Д. Н. Колисниченко

# PHP 5

## Самоучитель

Санкт-Петербург
2007

http://all-ebooks.com

Колисниченко Д. Н.

**САМОУЧИТЕЛЬ PHP 5** — СПб.: Наука и Техника, 2007. — 640 с., ил.

Серия «Самоучитель»

---

Данная книга является превосходным учебным пособием, позволяющим быстро освоить и эффективно использовать PHP. Здесь вы найдете все, что для этого необходимо: от описания настройки PHP, основ CGI и базовых конструкций языка до работы с сессиями, каналами, сокетами и другими специальными технологиями PHP. Особое место занимает рассмотрение взаимодействия PHP с базами данных.

Книга максимально ориентирована на достижение реальных практических результатов. В деталях описано, как создать на PHP свою гостевую книгу, чат, форум, почтовую рассылку на сайте, новостную ленту и даже интернет-магазин. Приведены готовые скрипты. Отдельно рассмотрена методика создания системы автоматического управления содержимым сайта. Кроме создания своих скриптов, приведено описание уже готовых профессиональных продуктов (бесплатных и открытых): форума phpBB и системы PHP-Nuke. Эти продукты применяются на огромном количестве сайтов и позволяют «дважды не изобретать велосипед», а использовать уже готовые решения.

Книга отличается простотой и ясностью изложения материала. Написана опытным и известным автором. Лучший выбор для начинающих изучать PHP.

Контактные телефоны издательства:
(812) 567 70 25, (812) 567 70 26, (044) 516 38 66

Официальный сайт: www.nit.com.ru

© Наука и техника (оригинал-макет), 2007
© Колисниченко Д. Н.

**http://all-ebooks.com**

# СОДЕРЖАНИЕ


**ГЛАВА 1. ОСНОВНЫЕ ТЕХНОЛОГИИ ИНТЕРНЕТА ..... 19**

1.1. ЧТО ТАКОЕ СЕТЬ? ..... 20
1.2. ЧТО ТАКОЕ ПРОТОКОЛ? ПОПУЛЯРНЫЕ ПРОТОКОЛЫ ИНТЕРНЕТА ..... 26
1.3. СЕТЕВОЕ ВЗАИМОДЕЙСТВИЕ ..... 28
1.4. АДРЕСАЦИЯ В СЕТИ ИНТЕРНЕТ ..... 37
1.4.1. Вид IP-адреса и способы его назначения ..... 37
1.4.2. Классы IP-сетей ..... 39
1.4.3. URL-адреса и доменные имена ..... 40
1.5. СЛУЖБА ДОМЕННЫХ ИМЕН DNS ..... 42
1.6. WORLD WIDE WEB И HTML-ДОКУМЕНТЫ ..... 44
1.7. ПОРТЫ И ДЕМОНЫ ..... 45
1.8. СТРУКТУРА ПАКЕТОВ IP И TCP ..... 47
1.9. ГЛОССАРИЙ ..... 48

**ГЛАВА 2. ИНТЕРФЕЙС CGI ..... 50**

2.1. ПРИНЦИПЫ РАБОТЫ CGI ..... 51
2.1.1. Общие положения ..... 51
2.1.2. Взаимодействие сервера с браузером ..... 52
2.1.3. Роль интерфейса CGI ..... 52
2.1.4. Немного подробнее о CGI ..... 53
2.1.5. Способы создания CGI-сценариев ..... 54
2.2. МЕТОДЫ ПЕРЕДАЧИ ДАННЫХ GET И POST ..... 55
2.2.1. Передача данных серверу и CGI-сценариям ..... 55
2.2.2. Метод GET ..... 57
*Формирование ответа сервером* ..... 58
2.2.3. Метод POST ..... 60
2.3. ПЕРЕМЕННЫЕ ОКРУЖЕНИЯ CGI ..... 60
2.4. ЗАГОЛОВКИ HTTP ..... 62
2.4.1. Описание основных заголовков HTTP ..... 62
2.4.2. Правила формирования имен переменных окружения ..... 64
2.5. КОДЫ ОТВЕТОВ HTTP ..... 64


http://all-ebooks.com

2.6. ПОДРОБНЕЕ ОБ URL ..... 67
2.7. ТИПЫ MIME ..... 68
2.8. PHP И CGI ..... 68

**ГЛАВА 3.** УСТАНОВКА И НАСТРОЙКА НЕОБХОДИМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ..... 70

3.1. ВЫБОР НЕОБХОДИМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ ..... 71
3.2. APACHE + PHP + MYSQL В WINDOWS ..... 74
3.3. APACHE + PHP + MYSQL В LINUX ..... 82
3.4. ВЫБОР РЕДАКТОРА PHP-СЦЕНАРИЕВ ..... 92

**ГЛАВА 4.** ВЕБ-СЕРВЕР APACHE ..... 94

4.1. КАКИЕ ФУНКЦИИ ВЫПОЛНЯЕТ APACHE ..... 95
4.2. КОНФИГУРАЦИОННЫЕ ФАЙЛЫ APACHE ..... 96
4.3. КОНФИГУРАЦИОННЫЙ ФАЙЛ HTTPD.CONF И ЕГО ДИРЕКТИВЫ ..... 97
4.3.1. Общие директивы ..... 98
4.3.2. Директивы протоколирования ..... 100
4.3.3. Директивы управления производительностью ..... 100
4.3.4. Директивы обеспечения постоянного соединения с клиентом .. 101
4.3.5. Директивы создания виртуальных узлов ..... 102
4.3.6. Директивы настройки отображения каталогов ..... 102
4.3.7. Директивы обработки MIME-типов ..... 103
4.3.8. Директивы для работы с многоязычными документами ..... 104
4.3.9. Директивы перенаправления ..... 105
4.3.10. Директивы обработки ошибок ..... 106
4.4. ДИРЕКТИВЫ УПРАВЛЕНИЯ ДОСТУПОМ К ОТДЕЛЬНЫМ КАТАЛОГАМ ..... 106
4.4.1. Блок директив Directory ..... 107
4.4.2. Блок директив Limit ..... 108
4.4.3. Блок директив Location ..... 110
4.5. КАТАЛОГИ ПОЛЬЗОВАТЕЛЕЙ ..... 110
4.6. ВИРТУАЛЬНЫЕ ВЕБ-СЕРВЕРЫ ..... 111
4.6.1. Виртуальные серверы с идентификацией по имени ..... 112
4.6.2. Виртуальные серверы с идентификацией по IP-адресу ..... 113
4.7. КОНФИГУРИРОВАНИЕ APACHE С ПОМОЩЬЮ SYSTEM-CONFIG-HTTPD ..... 115



http://all-ebooks.com

ЧАСТЬ II. ОСНОВНЫЕ ПОНЯТИЯ, МЕХАНИЗМЫ И ФУНКЦИИ PHP... 116

ГЛАВА 5. ОСНОВЫ ЯЗЫКА PHP ... 117

5.1. СТРУКТУРА ПРОГРАММ В PHP. ПЕРВАЯ ПРОГРАММА ... 118

5.1.1. Стандартный вид PHP-сценария ... 118

Общие понятия и описание PHP ... 118

Пишем первую программу (скрипт) на PHP ... 121

5.1.2. PHP-сценарии в теле HTML-документов ... 125

5.1.3. Комментарии в PHP-сценариях ... 126

5.2. ПЕРЕМЕННЫЕ И ТИПЫ ДАННЫХ ... 126

Типы данных ... 127

Функции определения и задания типа переменных ... 128

Логические переменные и их особенности в PHP ... 129

5.3. КОНСТАНТЫ ... 130

5.4. ОПЕРАЦИИ С ПЕРМЕННЫМИ ... 132

5.4.1. Присвоение значений. Оператор присваивания ... 132

5.4.2. Проверка существования переменной ... 133

5.4.3. Удаление переменных ... 135

5.5. ВЫРАЖЕНИЯ И ОПЕРАЦИИ ... 135

5.5.1. Понятие «выражение» в PHP ... 135

5.5.2. Операции в PHP ... 136

Арифметические операции ... 136

Битовые операции ... 137

Логические выражения ... 137

Приоритет операций ... 139

5.5.3. Строки и строковые выражения ... 139

Строки в апострофах и строки в кавычках ... 139

Строковые операции ... 140

Многострочные строковые переменные ... 140

5.5.4. Оператор эквивалентности (===) ... 142

5.6. ССЫЛКИ ... 143

5.6.1. Жесткие ссылки ... 143

5.6.2. Символические ссылки ... 144

**5.7. ОСНОВНЫЕ КОНСТРУКЦИИ ЯЗЫКА ..... 144**

5.7.1. Условный оператор ..... 144

5.7.2. Циклы ..... 146

Цикл с предусловием while ..... 146

Цикл с постусловием do-while ..... 147

Цикл со счетчиком for ..... 147

Альтернативный синтаксис ..... 148

Вложенные циклы ..... 149

Операторы break и continue ..... 150

Цикл перебора массивов foreach ..... 151

5.7.3. Оператор switch-case (конструкция выбора) ..... 151

**5.8. СОБИРАЕМ ОДИН СКРИПТ ИЗ НЕСКОЛЬКИХ ..... 154**

5.8.1. Инструкция require ..... 154

5.8.2. Инструкция include ..... 154

5.8.3. Особенности использования include и require ..... 155

5.8.4. Инструкции однократного включения include_once и require_once ..... 156

**5.9. ОБРАБОТКА ОШИБОК ..... 156**

**ГЛАВА 6. ПЕРЕДАЧА ПАРАМЕТРОВ СЦЕНАРИЯМ ..... 158**

**6.1. ФОРМЫ В HTML-ДОКУМЕНТАХ ..... 159**

6.1.1. Общие положения ..... 159

6.1.2. Элементы форм ..... 160

6.1.3. Тег FORM – контейнер форм ..... 162

6.1.4. Тег INPUT и способы его использования ..... 162

Однострочные поля ввода ..... 162

Поле ввода пароля ..... 163

Скрытое текстовое поле ..... 163

Независимые переключатели (флажки) ..... 164

Зависимые переключатели (радиокнопки) ..... 164

Кнопка отправки формы ..... 166

Кнопка сброса ..... 166

Кнопка отправки с индивидуальным рисунком ..... 166

6.1.5. Ввод многострочного текста. Тег TEXTAREA ..... 167

6.1.6. Списки выбора. Тег SELECT ..... 168



http://all-ebooks.com

Списки с единственным выбором ..... 168
Списки множественного выбора ..... 170
6.1.7. Загрузка файлов ..... 170

**6.2. ПЕРЕДАЧА ПАРАМЕТРОВ С ПОМОЩЬЮ ФОРМЫ ..... 171**
**6.3. МЕХАНИЗМ COOKIES ..... 176**
6.3.1. Принцип работы механизма Cookies ..... 176
6.3.2. Программирование Cookies ..... 180
Установка Cookies ..... 180
Чтение значений Cookies ..... 181

**6.4. ДИРЕКТИВА REGISTER_GLOBALS ..... 182**
6.4.1. Почему опасно использовать register_globals? ..... 182
6.4.2. Массив $GLOBALS ..... 185

**ГЛАВА 7. МАССИВЫ И СПИСКИ ..... 187**

**7.1. ПОНЯТИЯ МАССИВА И СПИСКА ..... 188**
**7.2. АССОЦИАТИВНЫЕ МАССИВЫ ..... 190**
**7.3. ИНСТРУКЦИИ LIST() ..... 191**
**7.4. ИНСТРУКЦИЯ ARRAY() И СОЗДАНИЕ МНОГОМЕРНЫХ МАССИВОВ ..... 191**
**7.5. ОПЕРАЦИИ НАД МАССИВАМИ ..... 193**
7.5.1. Удаление массивов ..... 193
7.5.2. Слияние массивов ..... 193
Стандартное слияние ..... 193
Слияние с помощью функции array_merge() ..... 194
7.5.3. Перебор массива ..... 195
Косвенный перебор ..... 195
Прямой перебор ..... 196
Недостатки косвенного перебора ..... 196

**7.6. СОРТИРОВКА МАССИВОВ ..... 197**
7.6.1. Сортировка массива по значению. Функции asort() и arsort() ..... 197
7.6.2. Сортировка массивов по ключам. Функции ksort() и krsort() ..... 197
7.6.3. Сортировка списка ..... 198
7.6.4. Переворачивание массива. Функция array_reverse() ..... 198
7.6.5. Перемешивание списков. Функция shuffle() ..... 199

http://all-ebooks.com

7.6.6. Пользовательская сортировка массивов. Функции uksort(), uasort() и usort() ..... 199

**7.7. ПОЛУЧЕНИЕ ЧАСТИ МАССИВА ..... 201**

**7.8. ДОБАВЛЕНИЕ И УДАЛЕНИЕ ЭЛЕМЕНТОВ МАССИВА ..... 202**

7.8.1. Добавление в конец массива ..... 202

7.8.2. Удаление последнего элемента списка ..... 202

7.8.3. Добавление в начало списка. Удаление первого элемента списка ..... 203

**7.9. ПЕРЕМЕННЫЕ И МАССИВЫ. ВЗАИМНЫЕ ПЕРЕХОДЫ ..... 204**

7.9.1. Упаковка переменных в массив. Функция compact() ..... 204

7.9.2. Извлечение переменных из массива. Функция extract() ..... 204

**7.10. ПАРА ПОЛЕЗНЫХ ФУНКЦИЙ ДЛЯ РАБОТЫ С МАССИВАМИ ..... 205**

Разбор URL ..... 205

Создание диапазона чисел ..... 206

**7.11. НЕБОЛЬШОЙ ПРАКТИЧЕСКИЙ ПРИМЕР ..... 206**

**ГЛАВА 8. ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ В PHP ..... 208**

**8.1. ЧТО ТАКОЕ ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ ..... 209**

**8.2. СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ФУНКЦИЙ ..... 212**

8.2.1. Объявление функций ..... 212

8.2.2. Первая функция ..... 213

**8.3. ПЕРЕДАЧА ФУНКЦИИ ПАРАМЕТРОВ ..... 214**

**8.4. ПАРАМЕТРЫ ПО УМОЛЧАНИЮ ..... 216**

**8.5. ПЕРЕМЕННОЕ ЧИСЛО ПАРАМЕТРОВ В ФУНКЦИИ ..... 218**

**8.6. ЛОКАЛЬНЫЕ И ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ ..... 219**

**8.7. СТАТИЧЕСКИЕ ПЕРЕМЕННЫЕ ..... 221**

**8.8. РЕКУРСИЯ ..... 222**

**8.9. ПОЛЕЗНЫЕ ПРИМЕРЫ ..... 222**

8.9.1. Инициализация переменных ..... 223

8.9.2. Параметры-массивы ..... 224

8.9.3. Возврат функцией ссылки ..... 225

**8.10. УСЛОВНО ОПРЕДЕЛЯЕМЫЕ ФУНКЦИИ ..... 226**



http://all-ebooks.com

**ГЛАВА 9. ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ ..... 227**

**9.1. ОПЕРАЦИИ НАД СТРОКАМИ ..... 228**

9.1.1. Строки в PHP ..... 228

9.1.2. Конкатенация строк ..... 229

9.1.3. Сравнение строк ..... 229

**9.2. ОСНОВНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ ..... 231**

**9.3. ЗАМЕНА В СТРОКЕ ..... 232**

**9.4. ФУНКЦИИ ДЛЯ РАБОТЫ С ОТДЕЛЬНЫМИ СИМВОЛАМИ ..... 233**

**9.5. ИЗМЕНЕНИЕ РЕГИСТРА СИМВОЛОВ. УСТАНОВКА И НАСТРОЙКА ЛОКАЛИ ..... 234**

**9.6. ПРЕОБРАЗОВАНИЕ СТРОК ..... 235**

9.6.1. Функции удаления пробелов ..... 235

9.6.2. URL-кодирование и декодирование ..... 236

9.6.3. Экранирование ..... 237

9.6.4. Замена специальных символов на их HTML-эквиваленты ..... 238

**9.7. ПРЕОБРАЗОВАНИЕ КОДИРОВОК ..... 238**

**9.8. СПЕЦИАЛЬНЫЕ ФУНКЦИИ ..... 239**

9.8.1. Хэш-функции ..... 239

9.8.2. Функция serialize() ..... 241

9.8.3. Функция explode(): разбиение на подстроки ..... 242

**ГЛАВА 10. ПОЛЕЗНЫЕ СТАНДАРТНЫЕ ФУНКЦИИ ..... 246**

**10.1. РАБОТА С ДАТОЙ И ВРЕМЕНЕМ ..... 247**

**10.2. ГЕНЕРАТОР СЛУЧАЙНЫХ ЧИСЕЛ. ВЫВОД БАННЕРОВ СЛУЧАЙНЫМ ОБРАЗОМ ..... 248**

**10.3. МАТЕМАТИЧЕСКИЕ ФУНКЦИИ ..... 250**

**10.4. ПРЕОБРАЗОВАНИЯ СИСТЕМ СЧИСЛЕНИЯ ..... 251**

**10.5. ЗАПУСК ВНЕШНИХ ПРОГРАММ ..... 252**

**10.6. НЕКОТОРЫЕ ИНТЕРЕСНЫЕ ФУНКЦИИ PHP ..... 257**

10.6.1. Отображение исходного кода. Функция show_source() ..... 257

10.6.2. Функция sleep(): поспим немного ..... 258

http://all-ebooks.com

**ГЛАВА 11. РАБОТА С ФАЙЛАМИ И КАТАЛОГАМИ ..... 259**

**11.1. ОБЩИЕ ЗАМЕЧАНИЯ ..... 260**

**11.2. ОСНОВНЫЕ ОПЕРАЦИИ С ФАЙЛАМИ ..... 262**

11.2.1. Открытие файла ..... 262

11.2.2. Конструкция or die() ..... 263

11.2.3. Временные файлы ..... 264

11.2.4. Чтение и запись в бинарном режиме ..... 265

11.2.5. Построчное чтение и запись ..... 270

11.2.6. Закрытие файлов ..... 271

*11.2.7. Функция file(). Чтение файла в список* ..... 271

11.2.8. Чтение файла в строку ..... 271

11.2.9. Чтение CSV-файлов ..... 273

**11.3. ФУНКЦИИ ДЛЯ РАБОТЫ С ФАЙЛАМИ ..... 274**

11.3.1. Функции манипулирования файлами ..... 275

11.3.2. Функции для работы с именами файлов ..... 276

11.3.3. Функции определения типа и параметров файла ..... 277

11.3.4. Специальные функции ..... 279

11.3.5. «Разборки» с INI-файлами. Функция parse_ini_file() ..... 280

**11.4. ПРАВА ДОСТУПА ..... 282**

11.4.1. Права доступа в ОС Linux ..... 282

11.4.2. Права доступа в Windows ..... 284

11.4.3. Функции определения прав доступа к файлу ..... 285

**11.5. БЛОКИРОВАНИЕ ДОСТУПА ..... 285**

11.5.1. Общие положения ..... 285

11.5.2. Виды блокировки ..... 286

11.5.3. Реализация блокировки в PHP ..... 287

11.5.4. Запись данных в начало файла ..... 289

**11.6. ФУНКЦИИ ДЛЯ РАБОТЫ С КАТАЛОГАМИ ..... 291**

11.6.1. Манипуляции с каталогами ..... 291

11.6.2. Открытие, чтение и закрытие каталогов ..... 291

**11.7. КАНАЛЫ ..... 293**



http://all-ebooks.com

11.7.1. Общие положения ... 293
11.7.2. Управление каналами на языке C ... 293
11.7.3. Управление каналами в PHP ... 296

**ЧАСТЬ III. РАСШИРЕННЫЕ ВОЗМОЖНОСТИ PHP ... 298**

**ГЛАВА 12. PHP И БАЗЫ ДАННЫХ ... 299**

**12.1. ОБЩИЕ ПОЛОЖЕНИЯ ... 300**

**12.2. ОСНОВЫ SQL ... 301**

12.2.1. Базовая терминология ... 302
12.2.2. Поля и их типы ... 303
12.2.3. Создание таблиц ... 305
12.2.4. Добавление данных в таблицу ... 306
12.2.5. Обновление записей ... 307
12.2.6. Удаление записей ... 308
12.2.7. Выбор записей ... 308
12.2.8. Дополнительные функции MIN, MAX, AVG, SUM ... 310
12.2.9. Группировка и сортировка записей ... 310
12.2.10. Ключи ... 311
12.2.11. Удаление полей и таблиц ... 314
12.2.12. Отключение от базы данных ... 315

**12.3. СЕРВЕР БАЗ ДАННЫХ MYSQL ... 315**

12.3.1. Клиентская и серверные части MySQL ... 315
12.3.2. Некоторые полезные приемы при работе с MySQL ... 317
Резервное копирование базы данных ... 317

**12.4. ФУНКЦИИ PHP ДЛЯ РАБОТЫ С MYSQL ... 317**

12.4.1. Соединение PHP-сценариев с таблицами MySQL ... 317
12.4.2. Выбор базы данных ... 319
12.4.3. Обработка ошибок ... 319
12.4.4. Обработка результата SQL-запроса ... 319
12.4.5. Получение информации о результате SQL-запроса ... 322

**12.5. СЕРВЕР БАЗ ДАННЫХ SQLITE ... 325**

http://all-ebooks.com

12.5.1. Преимущества и недостатки SQLite ..... 325
12.5.2.Установка SQLite ..... 328

**12.6. ФУНКЦИИ И КОНСТАНТЫ PHP ДЛЯ РАБОТЫ С SQLITE ..... 329**

12.6.1. Список функций SQLite ..... 329
12.6.2. Открытие и закрытие базы данных ..... 332
12.6.3. Выполнение запросов ..... 332
12.6.4. Функции для обработки результата запроса ..... 333
12.6.5. Информационные функции ..... 335

**12.7. ПОСТРАНИЧНЫЙ ВЫВОД ИЗ БАЗЫ ДАННЫХ ..... 335**

**ГЛАВА 13. ГРАФИЧЕСКИЕ ФУНКЦИИ PHP ..... 340**

**13.1. ОСНОВНЫЕ ПОНЯТИЯ ..... 341**

13.1.1. Библиотека GD ..... 341
13.1.2. Порядок создания графического оформления сайтов в PHP ... 342

**13.2. ВЫВОД ГОТОВОГО ИЗОБРАЖЕНИЯ ..... 342**

13.2.1. Графические форматы ..... 344
13.2.2. Определение размера картинки ..... 345

**13.3. ГРАФИЧЕСКИЕ ПРИМИТИВЫ ..... 345**

13.3.1. Создание (определение) цвета ..... 345
Задание нового цвета в палитре рисунка ..... 345
Подбор ближайшего цвета из палитры ..... 346
Эффект прозрачности ..... 347
13.3.2. Рисование графических примитивов ..... 347
Рисование точки ..... 347
Рисование линий ..... 348
Рисование прямоугольников ..... 348
Рисование многоугольников ..... 348
Рисование эллипсов и дуг ..... 349
Закрашивание области ..... 349

**13.4. ДОБАВЛЕНИЕ НАДПИСЕЙ К ИЗОБРАЖЕНИЮ ..... 351**

**13.5. РАБОТА С TTF-ШРИФТАМИ ..... 352**

**13.6. КОПИРОВАНИЕ ФРАГМЕНТОВ ИЗОБРАЖЕНИЯ ..... 355**

**13.7. СЦЕНАРИЙ ДИНАМИЧЕСКОГО ИЗМЕНЕНИЯ РАЗМЕРОВ ИЗОБРАЖЕНИЙ ..... 357**



http://all-ebooks.com

**ГЛАВА 14.** СЕТЕВЫЕ ФУНКЦИИ PHP ..... 361

14.1. СОКЕТЫ ..... 362

14.1.1. Понятие сокета. Функции для работы с сокетами ..... 362

Открытие и использование сокетов ..... 362

Блокирующий/неблокирующий режимы ..... 364

14.1.2. Отправка электронных писем с помощью сокетов ..... 364

14.2. РАБОТА С DNS ..... 368

14.2.1. DNS-функции ..... 368

14.2.2. Корректное разрешение IP-адреса в доменное имя ..... 369

14.3. ФУНКЦИИ ДЛЯ РАБОТЫ С FTP ..... 369

**ГЛАВА 15.** ЯЗЫК РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ REGEX ..... 373

15.1. ЧТО МЫ ИЩЕМ? МЫ И САМИ НЕ ЗНАЕМ ..... 374

15.2. РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ ..... 375

15.3. ЯЗЫК REGEX ..... 375

15.3.1. Литералы ..... 376

15.3.2. Управляющие конструкции ..... 378

15.3.3. Псевдосимволы ..... 379

15.3.4. Альтернативный оператор ..... 380

15.3.5. Скобки ..... 380

15.4. ФУНКЦИИ PHP ДЛЯ РАБОТЫ С РЕГУЛЯРНЫМИ ВЫРАЖЕНИЯМИ ..... 381

15.5. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ ..... 383

15.5.1. Автоматическое добавление ссылок ..... 383

15.5.2. Преобразование адресов электронной почты ..... 384

15.5.3. Преобразование имен файлов ..... 384

**ЧАСТЬ IV. СТРОИМ СВОЙ САЙТ** ..... 386

**ГЛАВА 16.** ДОМЕН + ХОСТИНГ. КУРС МОЛОДОГО САЙТОСТРОИТЕЛЯ ..... 387

16.1. БЕСПЛАТНЫЙ ХОСТИНГ ..... 388

16.2. КОММЕРЧЕСКИЙ ХОСТИНГ ..... 389

http://all-ebooks.com

16.2.1. Выбор доменного имени ... 389
16.2.2. Кому принадлежит доменное имя? ... 391
16.2.3. Наличие скрытого редиректа ... 393
16.2.4. Ежемесячная плата за хостинг ... 394
16.2.5. Средство управления сайтом ... 396
16.2.6. Скрытые (косвенные) платежи ... 396

**ГЛАВА 17. ПАНЕЛЬ УПРАВЛЕНИЯ ХОСТИНГОМ ... 398**

**17.1. ПАНЕЛЬ УПРАВЛЕНИЯ DIRECTADMIN ... 399**

17.1.1. Управление учетной записью ... 400
17.1.2. Управление доменами и поддоменами ... 403
17.1.3. Менеджер файлов ... 405
Удаление файлов и копирование файлов через буфер обмена ... 406
Создание файлов и каталогов. Загрузка новых файлов ... 407
Архивирование файлов ... 408
Копирование/Переименование/Редактирование файла ... 408
17.1.4. Управление почтой ... 408
POP3-аккаунты ... 409
Сборщик всей почты ... 410
Пересылка почты ... 410
Фильтры ... 410
17.1.5. Дополнительные функции ... 411

**17.2. ПАНЕЛЬ УПРАВЛЕНИЯ CPANEL ... 414**

17.2.1. Управление почтовыми аккаунтами ... 415
Добавить/удалить email аккаунты ... 416
Фильтры ... 417
Псевдонимы/Переадресаторы ... 418
Автоответчики ... 419
17.2.2. Инструменты управления сайтом ... 419
17.2.3. Хотлинк-защита ... 421
17.2.4. Менеджер ФТП ... 421
17.2.5. Поддомены ... 423
17.2.6. Базы данных MySQL ... 424
17.2.7. Файловый менеджер ... 426



http://all-ebooks.com

17.2.8. Резервное копирование ..... 427

**ГЛАВА 18. PHP-NUKE — СИСТЕМА УПРАВЛЕНИЯ СОДЕРЖИМЫМ САЙТА (КОНТЕНТОМ) ..... 429**

**18.1. ВОЗМОЖНОСТИ И ОСОБЕННОСТИ PHP-NUKE ..... 430**

**18.2. УСТАНОВКА PHP-NUKE ..... 432**

**18.3. ПЕРВЫЙ ЗАПУСК ..... 435**

**18.4. ИЗМЕНЕНИЕ ПАРАМЕТРОВ САЙТА ..... 436**

**18.5. СОЗДАНИЕ НЕБОЛЬШОГО САЙТА ..... 441**

18.5.1. Работа с блоками ..... 441

18.5.2. Создание опросов (голосований) ..... 447

18.5.3. Добавление новостей ..... 448

**18.6. ЧТО ЕЩЕ МОЖЕТ PHP-NUKE ..... 450**

**18.7. ДРУГИЕ CMS ..... 451**

**ГЛАВА 19. СОЗДАЕМ НА САЙТЕ ПОЧТОВУЮ РАССЫЛКУ ..... 458**

**19.1. ИСПОЛЬЗОВАНИЕ ФУНКЦИИ MAIL() ДЛЯ ОТПРАВКИ СООБЩЕНИЙ ..... 459**

**19.2. ЕСЛИ СООБЩЕНИЯ НЕ ОТПРАВЛЯЮТСЯ ..... 461**

**19.3. ОТПРАВЛЕНИЕ ПИСЕМ В ФОРМАТЕ HTML И ВЛОЖЕНИЙ ..... 461**

19.3.1. Общие положения ..... 461

19.3.2. Использование класса HtmlMimeMail ..... 462

**19.4. ПРИЕМ ПИСЕМ ПО ПРОТОКОЛУ POP3 ..... 466**

19.4.1. Общие сведения о механизме приема писем и протоколе POP3 ..... 466

19.4.2. Использование класса POP3 ..... 467

**ГЛАВА 20. ФОРУМ PHPBB ..... 473**

**20.1. УСТАНОВКА ФОРУМА PHPBB ..... 474**

**20.2. АДМИНИСТРИРОВАНИЕ ФОРУМА. ОСНОВНЫЕ ПАРАМЕТРЫ ФОРУМА ..... 477**

**20.3. УПРАВЛЕНИЕ ФОРУМАМИ ..... 483**

**20.4. УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ ..... 486**

**20.5. МАССОВАЯ РАССЫЛКА ..... 489**

http://all-ebooks.com

20.6. СТИЛИ ..... 489

20.7. МОДЫ ..... 490

ГЛАВА 21. ДЕЛАЕМ ЧАТ ДЛЯ СВОЕГО САЙТА ..... 491

21.1. ПРИНЦИП СОЗДАНИЯ ЧАТА ..... 492

21.2. ПРОСТОЙ ЧАТ – SIMPLECHAT ..... 493

21.2.1. Установка, настройка и работа с чатом ..... 494

21.2.2. Проблемы при установке ..... 497

Некорректная работа ..... 497

Некорректное отображение даты и времени ..... 497

Я не могу очистить чат! ..... 498

21.2.3. Полезные советы ..... 498

21.3. ПРОФЕССИОНАЛЬНЫЙ ЧАТ CHAT.PHP.SPB.RU ..... 499

21.3.1. Установка и настройка чата под Windows ..... 501

21.3.2. Установка и настройка чата под Linux ..... 502

ГЛАВА 22. ГОСТЕВАЯ КНИГА ..... 505

22.1. СОЗДАЕМ САМИ ГОСТЕВУЮ КНИГУ ..... 506

22.2. ИСПОЛЬЗУЕМ ГОТОВЫЙ СКРИПТ ГОСТЕВОЙ КНИГИ ..... 511

ГЛАВА 23. ЛЕНТА НОВОСТЕЙ ..... 514

23.1. ПРИНЦИП ОРГАНИЗАЦИИ НОВОСТНОЙ ЛЕНТЫ ..... 515

23.2. FUSION NEWS MANAGEMENT – ПРОФЕССИОНАЛЬНОЕ УПРАВЛЕНИЕ НОВОСТЯМИ ..... 516

23.2.1. Устанавливаем скрипт ..... 516

23.2.2. Работа со скриптом ..... 521

ГЛАВА 24. ГОЛОСОВАНИЕ НА САЙТЕ ..... 524

24.1. ПИШЕМ ОПРОС САМИ ..... 525

24.2. ПРОФЕССИОНАЛЬНЫЙ СЦЕНАРИЙ ADVANCED POLL ..... 528

24.2.1. Общее описание и установка Advanced Poll ..... 528

24.2.2. Настройка Advanced Poll. Центр управления ..... 529



http://all-ebooks.com

24.2.3. Создание и редактирование опроса ..... 531
24.2.4. Вывод опроса на веб-страницу ..... 532
24.2.5. Вывод нескольких опросов на одной странице ..... 534

**ГЛАВА 25. СЧЕТЧИК И АНАЛИЗ СЕРВЕРНОЙ СТАТИСТИКИ ..... 536**

**25.1. СЧЕТЧИК И ПРОТОКОЛИРОВАНИЕ ПОСЕЩЕНИЙ НА САЙТЕ ..... 537**

25.1.1. Простой счетчик ..... 537
25.1.2. Графический счетчик ..... 539
25.1.3. Протоколирование посещений сайта ..... 541
25.1.4. Разработка системы статистики ..... 544

**25.2. АНАЛИЗ СЕРВЕРНОЙ СТАТИСТИКИ. ПРОГРАММА WEBALIZER ..... 548**

25.2.1. Методика сбора и анализа серверной статистики. Серверные журналы ..... 548
25.2.2. Установка и запуск Webalizer ..... 551
25.2.3. Конфигурационный файл Webalizer ..... 553

**ГЛАВА 26. СЕССИИ В PHP ..... 556**

**26.1. ОБЗОР МЕХАНИЗМА СЕССИЙ ..... 557**

26.1.1. Когда не пригодны Cookies ..... 557
26.1.2. Механизм сессий: принцип работы ..... 558

**26.2. ПРИМЕНЕНИЕ СЕССИЙ ..... 559**

26.2.1. Инициализация сессии и регистрация сессионных переменных ..... 559
26.2.2. Имя сессии ..... 561
26.2.3. Устаревший способ обойти использование Cookies в сессиях 561

**26.3. ОБРАБОТЧИКИ СЕССИЙ ..... 562**

**ГЛАВА 27. СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ ШАБЛОНОВ. МОДУЛИ ..... 566**

**27.1. РАЗДЕЛЕНИЕ ПРОЕКТА НА ДИЗАЙНЕРСКУЮ ЧАСТЬ И ПРОГРАММИРОВАНИЕ ..... 567**
**27.2. ИСПОЛЬЗОВАНИЕ МОДУЛЕЙ. ПОНЯТИЕ БИБЛИОТЕКАРЯ ..... 568**

27.2.1. Общие положения ..... 568
27.2.2. Создание библиотекаря ..... 568

http://all-ebooks.com

27.2.3. Подключение модулей с помощью библиотекаря ..... 571

27.3. ИСПОЛЬЗОВАНИЕ ШАБЛОНОВ ..... 572

ГЛАВА 28. ПИШЕМ СВОЙ ИНТЕРНЕТ-МАГАЗИН ..... 580

28.1. ПОСТАНОВКА ЗАДАЧИ ..... 581
28.2. СТРУКТУРА ИНТЕРНЕТ-МАГАЗИНА ..... 582
28.3. ПИШЕМ СЦЕНАРИЙ ..... 583
28.4. ПИШЕМ СЦЕНАРИЙ ИНТЕРНЕТ-МАГАЗИНА ..... 586

ГЛАВА 29. ДИСТАНЦИОННОЕ ОБУЧЕНИЕ – СИСТЕМА ПРОВЕРКИ ЗНАНИЙ ..... 597

29.1. ОБЩИЕ ПОЛОЖЕНИЯ ..... 598
29.2. ПОСТАНОВКА ЗАДАЧИ ..... 599
29.3. НЕОБХОДИМЫЕ ТАБЛИЦЫ ..... 600
29.4. ПИШЕМ СЦЕНАРИЙ ..... 602

ГЛАВА 30. ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ И PHP ..... 613

30.1. ОСНОВНЫЕ ПРИНЦИПЫ ООП ..... 614
30.2. СОЗДАНИЕ КЛАССА В PHP ..... 616
30.3. НАСЛЕДОВАНИЕ КЛАССОВ ..... 618

30.3.1. Основы наследования ..... 618
30.3.2. Переопределение методов ..... 618
30.3.3. Ключевые слова final и abstract ..... 619
30.3.4. Интерфейсы ..... 622

30.4. КОНСТРУКТОР И ДЕСТРУКТОР ..... 623
30.5. ДОСТУП К ЧЛЕНАМ КЛАССА ..... 625
30.6. СТАТИЧЕСКИЕ ЧЛЕНЫ КЛАССА ..... 627
30.7. КОПИРОВАНИЕ И СРАВНЕНИЕ ОБЪЕКТОВ ..... 629



http://all-ebooks.com

# ГЛАВА 1.

# ОСНОВНЫЕ ТЕХНОЛОГИИ ИНТЕРНЕТА

http://all-ebooks.com

Прежде чем приступить к созданию веб-сайтов и веб-порталов, необходимо иметь хотя бы общее представление о том, как устроен и функционирует Интернет. Без этой информации вы не сможете изучить язык PHP, так как его технологии опираются на основные технологии Интернета и тесно взаимосвязаны с ними. Тем не менее базовый набор знаний об устройстве и принципах Интернета не так уж велик. Вполне возможно, что вы уже частично или полностью обладаете этой информацией, поэтому вы можете пропускать знакомые разделы в этой главе.

## 1.1. ЧТО ТАКОЕ СЕТЬ?

Итак, что мы будем считать сетью? Сеть – это совокупность компьютеров и других сетевых устройств, объединенных между собой средой передачи данных и коммуникационным оборудованием. Это общее определение сети. При этом в качестве среды передачи данных могут использоваться как провода и кабели различного типа (проводные технологии), так и радиоэфир (беспроводные технологии).

### 1.1.1. Территориальная классификация сетей

Существует много классификаций сетей. Одной из них является территориальная. Согласно этой классификации, сети бывают трех видов:

- Локальные (Local Area Network, LAN) – это относительно небольшие сети масштаба предприятия.


http://all-ebooks.com

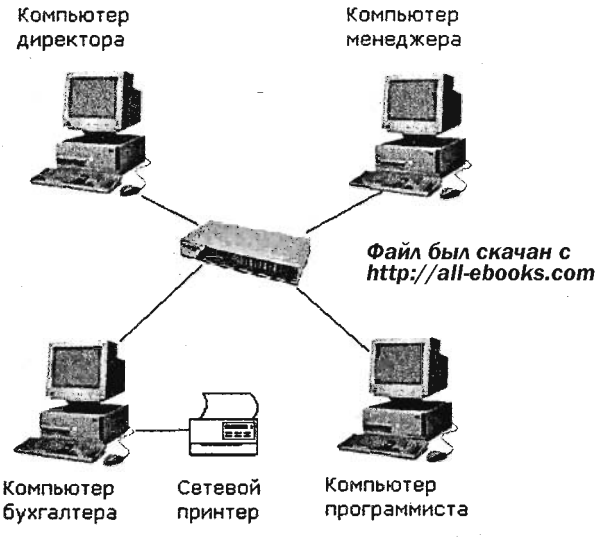



*Рис. 1.1. Локальная сеть*

- Муниципальные (Metropolitan Area Network, MAN) – сети масштаба города.
- Глобальные (Wide Area Network, WAN) – сети, охватывающие значительное географическое пространство – регион, страну, континент.

Самым тривиальным примером локальной сети являются два компьютера, соединенные с помощью витой пары (или другого кабеля). А самым удачным примером глобальной сети как раз и является Интернет.

### 1.1.2. Архитектура сети: одноранговая и клиент/сервер

Начнем с самого главного – архитектуры сети. Существуют две основные архитектуры сети: одноранговая (peer-to-peer) и клиент/сервер, причем вторая практически вытеснила первую.

В одноранговой сети все компьютеры равноправны – имеют один ранг. Поэтому любой компьютер может выступать как в роли сервера, предоставляющего свои ресурсы (файлы, принтеры) другому компьютеру, так и в роли клиента, использующего предоставленные ему ресурсы.

В отличие от одноранговой сети, в сети клиент/сервер существует один или несколько главных компьютеров – серверов. Все остальные компьютеры

http://all-ebooks.com

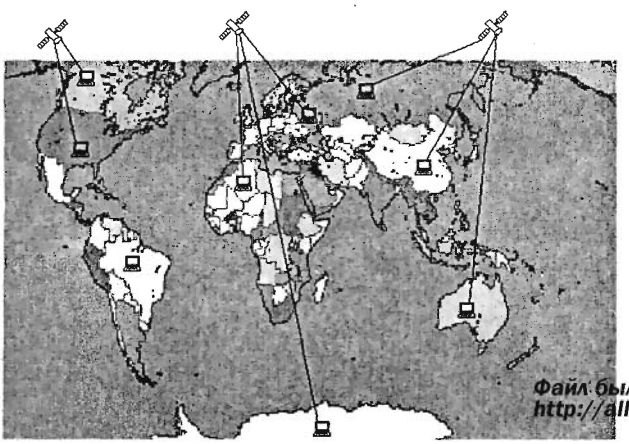

Файл был скачан с
http://all-ebooks.com

*Рис. 1.2. Глобальная сеть*

сети называются клиентами или рабочими станциями (workstations). Сервер – это специальный компьютер, который предоставляет определенные услуги другим компьютерам. Существуют различные виды серверов (в зависимости от предоставляемых ими услуг): серверы баз данных, файловые серверы, серверы печати, почтовые серверы, веб-серверы и т. д.

### Одноранговые сети (peer-to-peer)

Одноранговая архитектура встречается преимущественно в домашних сетях или небольших офисах. В самом простом случае для организации такой сети нужна всего лишь пара компьютеров, снабженных сетевыми платами, и коаксиальный кабель. Нужна еще пара Т-коннекторов (соединителей) и терминаторов (заглушек), но я обещал сильно не углубляться. Если вам нужно соединить только два компьютера, то для этого понадобится немного витой пары и не нужно даже никаких терминаторов и коннекторов!

Сначала сеть создается физически, то есть компьютеры соединяются друг с другом посредством коаксиального кабеля. После этого нужно настроить сеть программно. Для этого необходимо, чтобы на компьютерах были установлены сетевые операционные системы (Linux, FreeBSD, Windows 98, Windows NT/2000/XP/2003) или операционные системы с поддержкой сетевых функций (Windows 95, Windows for Workgroups). Программные настройки сети выполняются средствами выбранной операционной системы.

Компьютеры в одноранговой сети объединяются в рабочие группы. Каждая рабочая группа имеет свой идентификатор – имя рабочей группы. Если



http://all-ebooks.com

вы сейчас работаете в Windows XP, узнать имя рабочей группы вы можете, щелкнув правой кнопкой на пиктограмме Мой компьютер и выбрав команду Свойства. В появившемся окне вам нужно перейти на вкладку Имя компьютера (рис. 1.3).

Для примера допустим, что в вашей одноранговой сети три компьютера A, B, C. Первые два входят в рабочую группу WG1, а компьютер C – в рабочую группу WG2 (рис. 1.4).

Даже несмотря на то, что компьютеры входят в один сегмент сети (физически подключены к одному кабелю), компьютеры A и B не будут «видеть» компьютер C, а компьютер C не будет видеть компьютеров A и B. Если выполнить команду поиска компьютера в Windows XP (Пуск -> Найти -> Компьютеры), компьютер «увидит» компьютеры A и B, но будет сообщено, что они находятся в другой рабочей группе.

В одноранговой сети предусмотрено единственное ограничение доступа к общему ресурсу (например, принтеру): защита ресурса паролем, который пользователь должен ввести, чтобы получить доступ к этому ресурсу. Это называется управлением доступом на уровне ресурсов.

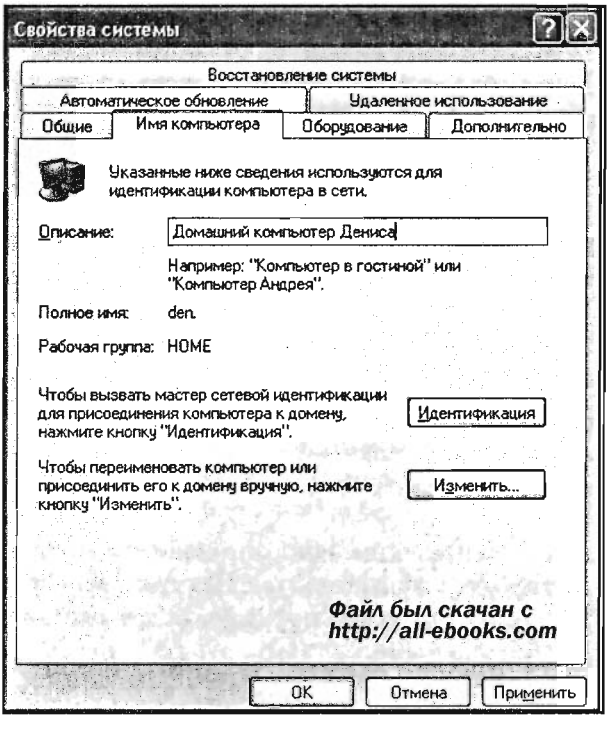



***Рис. 1.3. Идентификатор рабочей группы в ОС Windows XP***

http://all-ebooks.com

*Файл был скачан с*
*http://all-ebooks.com*

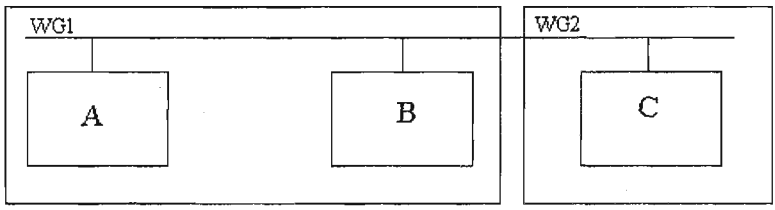



***Рис. 1.4. Схема одноранговой сети***

В сети клиент/сервер используется другой способ управления доступом – на уровне пользователей. В этом случае можно разрешить доступ к ресурсу только определенным пользователям.

Например, ваш компьютер А через сеть могут использовать два пользователя: Иванов и Петров. К этому компьютеру подключен принтер, доступный по сети. Но вы не хотите, чтобы кто угодно печатал на вашем принтере, и установили пароль для доступа к этому ресурсу. Если у вас одноранговая сеть, то любой, кто узнает этот пароль, сможет использовать ваш принтер. В случае с сетью клиент/сервер вы можете разрешить использовать ваш принтер только Иванову или только Петрову (можно и обоим).

### Сети клиент/сервер

Для получения доступа к ресурсу в сети клиент/сервер пользователь должен ввести свой уникальный идентификатор – регистрационное имя (логин) и пароль. Логины пользователей общедоступны и служат, например, для того, чтобы связаться с данным пользователем по электронной почте (естественно, кроме логина необходимо знать и имя сервера электронной почты, на котором зарегистрирован этот пользователь).

Использование логина и пароля для доступа к ресурсам называется идентификацией, а подтверждение (проверка подлинности) имени пользователя паролем – аутентификацией. Идентификация + аутентификация = авторизация. Иногда термин «аутентификация» означает проверку подлинности в широком смысле слова.

После рассмотрения архитектуры одноранговой сети можно прийти к выводу, что единственное преимущество этой архитектуры – это ее простота и дешевизна. Сети клиент/сервер обеспечивают более высокий уровень производительности и безопасности.

В сети клиент/сервер существует один или несколько главных компьютеров – серверов. Все остальные компьютеры сети называются клиентами или рабочими станциями.



http://all-ebooks.com

Как я уже писал выше, сервер – это специальный компьютер, который предоставляет определенные услуги другим компьютерам. Для экономии средств, как правило, один сервер сочетает в себе функции нескольких серверов, например, почтовик может быть также и веб-сервером. Услуги, которые может предоставлять сервер, ограничиваются только его аппаратными возможностями: чем мощнее компьютер, тем больше услуг и с большим качеством он может предоставлять. Поэтому на роль сервера обычно выбирается самый мощный из имеющихся компьютеров.

Сеть клиент/сервер более дорога в построении и обслуживании, чем простая одноранговая сеть. Так почему же Интернет построен с использованием именно этой технологии? Неужели никто не задумывался над экономией средств?

Оказывается, всемирная сеть просто не может быть одноранговой. Почему? Можно назвать множество причин, но мы ограничимся только двумя.

Первая причина связана с адресацией компьютеров в сети: чтобы отправить сообщение пользователю за определенным компьютером, вы должны однозначно идентифицировать этот компьютер. Человеку проще запоминать символьные имена, например `comp_den` или `comp_markus`, а компьютеру удобнее оперировать с числами – IP-адресами. Следовательно, нужна программа, которая автоматически преобразовывала бы символьные имена в соответствующие IP-адреса.

В одноранговой сети, где все узлы (компьютеры) по определению равноправны, сведения о соответствии имён адресам хранятся на каждом узле, в файлах HOSTS. Эта технология разрешения имён унаследована от предшественницы Интернета – сети ARPAnet. Администратор должен поддерживать содержимое всех этих файлов в актуальном состоянии, отражающем все изменения конфигурации сети (например, добавление новых узлов). А теперь представьте себе, что эта сеть объединяет сотни тысяч компьютеров (на самом деле в Интернете их гораздо больше)! Представляете, как неэффективно будет работать описанная схема? Поэтому было предпринято решение создать службу доменных имен – DNS (Domain Name System), которая бы и занималась преобразованием символьных имен в IP-адрес и наоборот.

Служба DNS работает так: к DNS-серверу (выделенному для функций DNS компьютеру) обращается компьютер с просьбой преобразовать символьное имя в IP-адрес. Если DNS-сервер может разрешить имя компьютера, он возвращает его IP-адрес. Все это делается незаметно для пользователя – ему нужно только указать символьное имя искомого узла, а все остальное его не касается. Администратору тоже удобно: ему нужно всего один раз

http://all-ebooks.com

отредактировать базу данных сервера DNS. Как видите, служба DNS подразумевает использование технологии клиент/сервер.

Вторая причина связана с авторизацией. Представьте, что в сети зарегистрированы сотни тысяч пользователей. Один из них входит в сеть и пытается обратиться к компьютеру `comp_den`. Очевидно, что в одноранговой сети список всех пользователей, которым разрешён доступ к `comp_den`, вместе с их паролями должен храниться на самом `comp_den`. То есть на каждом из компьютеров, ресурсы которых могут понадобиться пользователю, должны быть продублированы регистрационные данные этого пользователя. Если пользователь сменит пароль, его придётся менять на всех узлах сети.

Намного проще выделить для авторизации один или несколько серверов. В этом случае учётные данные пользователей хранятся в одном месте, что выгодно как с точки зрения экономии трафика, так и из соображений безопасности: сервер легче защитить от взлома, чем все компьютеры сети.

## 1.2. ЧТО ТАКОЕ ПРОТОКОЛ? ПОПУЛЯРНЫЕ ПРОТОКОЛЫ ИНТЕРНЕТА

Теперь пора уже перейти к протоколам, в частности, к протоколу TCP/IP, который лежит в основе сети Интернет. Протокол – это совокупность правил, определяющая взаимодействие абонентов вычислительной системы (в нашем случае – сети) и описывающая способ выполнения определенного класса функций. Говоря простым языком, протокол – это набор правил, по которым компьютеры взаимодействуют между собой.

Необходимость протоколов обусловлена тем, что в сети могут взаимодействовать компьютеры с самым разным программным обеспечением (операционными системами) и самым разным аппаратным устройством. Чтобы все подключенные к сети компьютеры могли понимать друг друга, необходимы общие наборы правил. Такими наборами правил и являются протоколы. Для разного рода взаимодействий используются разные правила, а значит и разные протоколы. Давайте рассмотрим основные протоколы, используемые в Интернете.

**Протокол TCP/IP**. Основополагающим протоколом сети Интернет служит протокол TCP/IP (Transmission Control Protocol/Internet Protocol - Протокол Управления Передачей/Интернет-Протокол). Если быть точными, TCP/IP – это не один протокол, а два:

- Транспортный протокол TCP обеспечивает гарантированную доставку данных по сети.



http://all-ebooks.com

- Адресный протокол IP отвечает за адресацию всей сети. Иными словами, благодаря использованию протокола IP каждый узел сети имеет свой индивидуальный адрес (IP-адрес). По этим адресам и осуществляется передача данных. Широко используемые в Интернете символьные адреса (`www.rambler.ru`, `www.dkws.org.ua`, `www.microsoft.com` и т. п.) являются лишь словесными обозначениями IP-адресов. Сделано это для удобства, поскольку человеку намного удобнее запомнить символьный адрес, чем числовой IP-адрес. Когда вы вводите символьный адрес в адресную строку браузера, он автоматически преобразуется в IP-адрес. Обмен данными в Интернете осуществляется только по IP-адресам.

За сопоставление символьных адресов и числовых IP-адресов отвечает специальная служба – DNS. Ошибка в этом процессе может привести к тому, что вы будете обращаться к одному серверу, а попадать на другой или вообще никуда не попадать, а наблюдать у себя на экране сообщение об ошибке. Подробнее на рассмотрении протокола TCP/IP и службы DNS мы остановимся чуть позже, когда будем изучать передачу данных в сети Интернет. Сейчас же перечислим ещё несколько немаловажных протоколов.

**Протокол ICMP** (Internet Control Message Protocol) – протокол межсетевых управляющих сообщений. С помощью этого протокола компьютеры и устройства сети обмениваются друг с другом управляющей информацией. К примеру, этот протокол используется для передачи сообщений об ошибках, проверки доступности узла и т. д.

**Протокол FTP** (File Transfer Protocol) – протокол передачи файлов. Служит для обмена файлами между компьютерами. Например, вам нужно передать файл на сервер или, наоборот, скачать файл с сервера. Для этого вам нужно подключиться к файловому серверу (он же FTP-сервер) и выполнить необходимую вам операцию скачивания или закачки. Подключение осуществляется с помощью FTP-клиента. Простейший FTP-клиент входит в состав практически любой операционной системы. Кстати, просматривать FTP-серверы могут и обычные браузеры.

**Протокол HTTP** (Hyper Text Transfer Protocol) – протокол обмена гипертекстовой информацией, то есть документами HTML. Вы, наверное, слышали, что HTML является базовым языком для создания веб-страниц. Так вот, протокол HTTP предназначен для передачи веб-страниц по сети. Таким образом, протокол HTTP используется веб-серверами, а браузеры – программы, служащие для просмотра веб-страниц, – являются HTTP-клиентами.

**Протоколы POP и SMTP**. Протокол POP (Post Office Protocol) – протокол почтового отделения. Этот протокол используется для получения электрон-

http://all-ebooks.com

ной почты с почтовых серверов. А для передачи электронной почты служит протокол SMTP (Simple Mail Transfer Protocol).

**Протокол IMAP.** Для чтения почты может использоваться еще один протокол – IMAP. Его отличие от протокола POP состоит в том, что пользователь читает сообщения электронной почты, не загружая их на свой компьютер. Все сообщения хранятся на сервере. При удалении сообщения оно удаляется с сервера.

**Протокол SLIP** (Serial Line Internet Protocol) – протокол подключения к сети Интернет по последовательной линии. Используется для установления связи с удаленными узлами через низкоскоростные последовательные интерфейсы. В настоящее время вытеснен протоколом PPP и практически не используется.

**Протокол PPP** (Point-to-Point Protocol) – обеспечивает управление конфигурацией, обнаружение ошибок и повышенную безопасность при передаче данных на более высоком уровне, чем протокол SLIP. Поэтому при настройке сервера рекомендуется использовать именно этот протокол. Протокол PPP рассмотрен в RFC 1547 и RFC 1661.

**Протокол RIP** (Routing Information Protocol) – используется для маршрутизации пакетов в компьютерных сетях. Для маршрутизации также используется протокол OSPF (Open Shortest Path First), который является более эффективным, чем RIP.

## 1.3. СЕТЕВОЕ ВЗАИМОДЕЙСТВИЕ

### 1.3.1. В КАКОМ ВИДЕ ИНФОРМАЦИЯ ПЕРЕДАЕТСЯ ПО СЕТИ

Любая сеть связи работает с использованием определенного метода коммутации абонентов. Наиболее распространены три метода коммутации:

- коммутация каналов;
- коммутация пакетов;
- коммутация сообщений.

Для использования метода коммутации каналов нужен физический канал для прямой передачи данных между узлами. Физический канал состоит из последовательно соединенных канальных участков. Участки объединяются между собой коммутаторами, которые устанавливаются между конечными узлами сети.



http://all-ebooks.com

Коммутация каналов удобна для телефонных сетей. Для передачи компьютерных данных сети с коммутацией каналов совершенно не годятся из-за слишком большого коэффициента пульсации трафика.

Что это такое? Представьте, что вы заходите на почтовый сервер и запрашиваете список сообщений. В ответ на ваш запрос сервер порождает сравнительно небольшой трафик, содержащий только заголовки полученных писем: отправитель, дата, тема, размер сообщения. Потом вы загружаете на свой компьютер одно из полученных сообщений – письмо от вашего друга с десятком качественных фотографий общим размером в 10 Мб. Получение этого сообщения порождает интенсивный обмен данными. Просмотрев фотографии, вы, в свою очередь, отправляете ему свои фото, что также создает большую нагрузку на канал связи. Если сеть, в которой вы работаете, использует метод коммутации каналов, то канал между вами и сервером большую часть времени будет простаивать, занимая и без того ограниченные ресурсы коммутатора.

Поэтому для передачи компьютерного трафика был разработан другой метод – коммутация пакетов. При коммутации пакетов данные разбиваются на части ограниченного размера – пакеты. Пакеты передаются как независимые блоки, а на компьютере-получателе снова собираются воедино. Пакетная передача позволяет сбалансировать нагрузку на канал связи и обеспечить наиболее эффективное его использование.

Пакет состоит из заголовков и поля данных. Заголовок содержит служебную информацию – адрес отправителя, адрес получателя, порт назначения и другую информацию. Поле данных содержит передаваемые вами данные. Ясное дело, что не все сразу, а только их часть. Любая сеть накладывает свои ограничения на длину пакетов, например, для сети Ethernet минимальный размер пакета равен 72 байтам, а максимальный – 1526 байтам. Минимальный размер обусловлен длиной необходимой служебной информации: если её не указать, то мы не сможем отправить даже пустой пакет. Мы еще вернемся к устройству пакетов и рассмотрим заголовок пакета подробнее.

Мы забыли о сетях с коммутацией сообщений. В чистом виде они сегодня не существуют, поскольку в свое время они послужили прототипом сетей с коммутацией пакетов.

### 1.3.2. Семиуровневая модель взаимодействия открытых систем OSI

В начале 80-х годов международной организацией по стандартизации (ISO -- International Organization for Standardization) была разработана модель взаимодействия открытых систем (OSI -- Open System Interconnection). В

другой литературе вы можете встретить и другие названия этой модели: сокращенное – модель OSI или более полное – семиуровневая модель взаимодействия открытых систем OSI.

Средства взаимодействия (рис. 1.5) в модели OSI делятся на сеть уровней, каждый из которых призван решать свой круг задач:

1. Физический.
2. Канальный.
3. Сетевой.
4. Транспортный.
5. Сеансовый.
6. Представительный.
7. Прикладной.

Файл был скачан с
http://all-ebooks.com

Благодаря разделению на уровни задача сетевого взаимодействия разбивается на несколько подзадач. Это позволяет при разработке новых способов

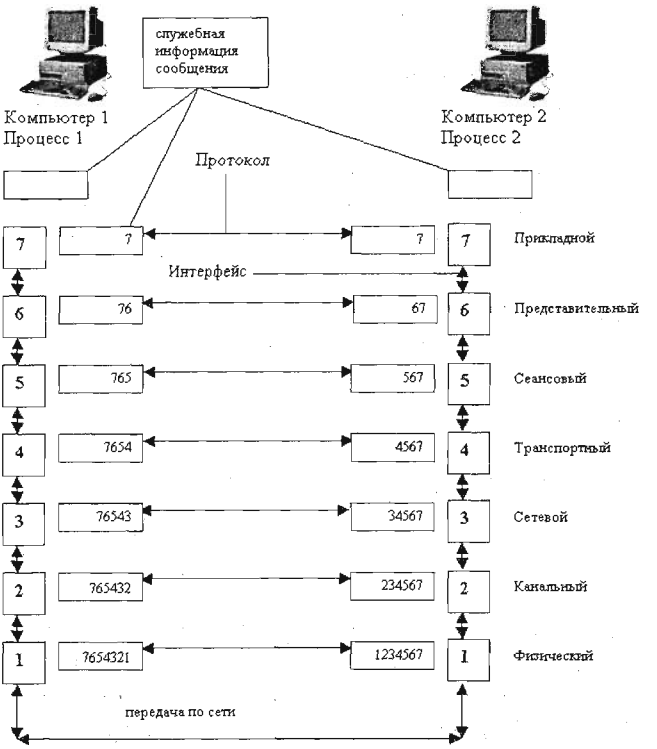



*Рис. 1.5. Модель OSI*



http://all-ebooks.com

и инструментов сетевого взаимодействия не создавать их с нуля, а использовать уже готовые решения, заменив только некоторые части.

Непосредственно друг с другом компьютеры (открытые системы) взаимодействуют только на физическом уровне. Все остальные уровни напрямую взаимодействуют только с выше- и нижележащими уровнями: пользуются услугами нижележащего и предоставляют услуги вышележащему. Друг с другом такие уровни контактируют косвенным образом, через посредство нижележащих уровней.

> **Примечание.**
>
> В некоторых случаях сетевого взаимодействия физический уровень как таковой отсутствует, при этом его функции выполняет самый нижележащий уровень.

Из рис. 1.5 видно, что по мере прохождения сообщения через уровни модели OSI к пересылаемым данным добавляется служебная информация, свидетельствующая о прохождении данных через определенный уровень.

Рассмотрим взаимодействие двух компьютеров более подробно на примере файловой службы. Допустим, вам (компьютер 1) нужно записать какую-нибудь информацию в файл на удаленном компьютере 2.

Взаимодействие между компьютерами вы обычно осуществляете с помощью каких-либо приложений, обладающим специальным набором функций. Эти приложения работают на самом высоком уровне модели взаимодействия – прикладном. Поэтому, когда вы укажете, что хотите записать определенные данные в файл, будет сформировано соответствующее сообщение. В поле данных этого сообщения и будет записана передаваемая в файл информация.

Сформированное сообщение с прикладного уровня будет передано на представительский уровень. На этом уровне в заголовок добавляются указания для представительского уровня компьютера-адресата. Потом сообщение передается сеансовому уровню, который добавляет свою информацию, и т. д. Как изменяется заголовок, видно из рис. 1.5. Процесс вложения одного протокола в другой называется инкапсуляцией. В процессе прохождения исходного блока данных (сообщения) по уровням оно разбивается на более мелкие фрагменты для пересылки их по сети.

Когда сообщение поступает на компьютер-адресат, оно принимается физическим уровнем и передается вверх с уровня на уровень. Каждый уровень анализирует содержимое заголовка своего уровня, выполняет содержащиеся в нем указания, затем удаляет относящуюся к себе информацию из заголовка и передает сообщение далее вышележащему уровню. Этот процесс называется декапсуляцией.

Далее приведено описание уровней взаимодействия.

## 1.3.3. Уровни физического взаимодействия

### Физический уровень (Physical Layer)

Физический уровень передает биты по физическим каналам связи, например коаксиальному кабелю или витой паре. Другими словами, именно этот уровень непосредственно производит передачу данных.

На этом уровне определяются характеристики электрических сигналов, которые передают дискретную информацию, например: тип кодирования, скорость передачи сигналов. К этому уровню также относятся характеристики физических сред передачи данных: полоса пропускания, волновое сопротивление, помехозащищенность.

Функции физического уровня реализуются сетевым адаптером или последовательным портом. Примером протокола физического уровня может послужить спецификация 100Base-TX (технология Fast Ethernet).

### Канальный уровень (Data Link Layer)

Канальный уровень отвечает за передачу данных между узлами в рамках одной локальной сети. При этом под узлом понимается любое устройство, подключенное к сети.

Этот уровень выполняет адресацию по физическим адресам (MAC-адресам), «вшитым» в сетевые адаптеры предприятием-изготовителем. Каждый сетевой адаптер имеет свой уникальный MAC-адрес, то есть вы не найдете двух сетевых плат с одним и тем же MAC-адресом.

Канальный уровень переводит поступившую с верхнего уровня информацию в биты, которые потом будут переданы физическим уровнем по сети. Он разбивает пересылаемую информацию на фрагменты данных – кадры (frames).

На этом уровне открытые системы обмениваются именно кадрами. Процесс пересылки выглядит примерно так: канальный уровень отправляет кадр физическому уровню, который передаёт кадр в сеть. Каждый узел сети получает этот кадр и проверяет, соответствует ли адрес пункта назначения адресу этого узла. Если адреса совпадают, канальный уровень принимает кадр и передает наверх вышележащим уровням. Если же адреса не совпада-



http://all-ebooks.com

ют, то он просто игнорирует кадр. Таким образом, сеть на канальном уровне является широковещательной.

В используемых в локальных сетях протоколах канального уровня заложена определенная топология. Топологией называется способ организации физических связей и способы их адресации. Канальный уровень обеспечивает доставку данных между узлами в сети с определенной топологией, то есть для которой он разработан. К основным топологиям (рис. 1.6) относятся:

1. Общая шина.
2. Кольцо.
3. Звезда.

Протоколы канального уровня используются компьютерами, мостами, маршрутизаторами. Глобальные сети (в том числе и Интернет) редко обладают регулярной топологией, поэтому канальный уровень обеспечивает связь только между компьютерами, соединенными индивидуальной линией связи.

Для доставки данных через всю глобальную сеть используются средства сетевого уровня (протоколы «точка-точка»). Примерами протоколов «точка-точка» могут послужить PPP, LAP-B.

### Сетевой уровень (Network Layer)

Данный уровень служит для образования единой транспортной системы, которая объединяет несколько сетей. Другими словами сетевой уровень обеспечивает межсетевое взаимодействие.

Протоколы канального уровня передают кадры между узлами только в рамках сети с соответствующей топологией. Нельзя передать кадр канального уровня узлу, который находится в другой сети. Данное ограничение не

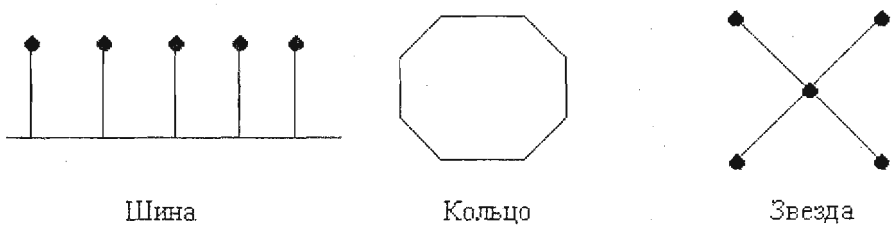



***Рис. 1.6. Основные топологии локальных компьютерных сетей***

позволяет строить сети с развитой структурой или сети с избыточностью связей, а именно такой сетью является Интернет.

Построить большую локальную сеть на канальном уровне невозможно также из-за физических ограничений. В частности, производительность сети из 1024 узлов в одном сегменте (что разрешается спецификацией 10Base-T ) вас не обрадует: как уже сказано, на канальном уровне сеть является широковещательной, то есть каждый кадр доставляется каждому компьютеру в сегменте, а это создаёт огромную нагрузку на сеть и сказывается на быстроте сетевого взаимодействия. Все это приводит к необходимости другого решения для больших сетей. Именно такое решение и призван реализовать сетевой уровень.

На сетевом уровне термин сеть следует понимать как совокупность компьютеров, соединенных в соответствии с одной из основных топологий и использующих для передачи данных один из протоколов канального уровня.

Сети соединяются специальными устройствами – маршрутизаторами. Маршрутизатор собирает информацию о топологии межсетевых соединений и на основании этой информации пересылает пакеты сетевого уровня в сеть назначения.

Чтобы передать сообщение от компьютера-отправителя компьютеру-адресату, который находится в другой сети, нужно совершить некоторое количество транзитных передач между сетями. Иногда их еще называют хопами (от англ. hop – прыжок). При этом каждый раз выбирается подходящий маршрут.

Сообщения на сетевом уровне называются пакетами. На сетевом уровне работают несколько видов протоколов. Прежде всего это сетевые протоколы, которые обеспечивают передвижение пакетов по сети, в том числе в другую сеть. Поэтому довольно часто к сетевому уровню относят протоколы маршрутизации (routing protocols) -- RIP и OSPF.

Еще одним видом протоколов, работающих на сетевом уровне, являются протоколы разрешения адреса – Address Resolution Protocol (ARP). Хотя эти протоколы иногда относят и к канальному уровню.

Классические примеры протоколов сетевого уровня: IP (стек TCP/IP), IPX (стек Novell).

### Транспортный уровень (Transport Layer)

На пути от отправителя к получателю пакеты могут быть искажены или утеряны. Некоторые приложения самостоятельно выполняют обработку



http://all-ebooks.com

ошибок при передаче данных, но большинство все же предпочитают иметь дело с надежным соединением, которое как раз и призван обеспечить транспортный уровень. Этот уровень обеспечивает требуемую приложению или верхнему уровню (сеансовому или прикладному) надежность доставки пакетов. На транспортном уровне определены пять классов сервиса:

1. Срочность.
2. Восстановление прерванной связи.
3. Наличие средств мультиплексирования нескольких соединений.
4. Обнаружение ошибок.
5. Исправление ошибок.

Обычно уровни модели OSI, начиная с транспортного уровня и выше, реализуются на программном уровне соответствующими компонентами операционных систем.
Примеры протоколов транспортного уровня: TCP и UDP (стек TCP/IP), SPX (стек Novell).

### Сеансовый уровень (Session Layer)

Сеансовый уровень устанавливает и разрывает соединения между компьютерами, управляет диалогом между ними, а также предоставляет средства синхронизации. Средства синхронизации позволяют вставлять определенную контрольную информацию в длинные передачи (точки), чтобы в случае обрыва связи можно было вернуться назад (к последней точке) и продолжить передачу с места обрыва.

Сеанс – это логическое соединение между компьютерами. Каждый сеанс имеет три фазы:

1. Установление соединения. Здесь узлы «договариваются» между собой о протоколах и параметрах связи.
2. Передача информации.
3. Разрыв связи.

Не нужно путать сеанс сетевого уровня с сеансом связи. Пользователь может установить соединение с Интернетом, но не устанавливать ни с кем логического соединения, то есть не принимать и не передавать данные.

### Представительный уровень (Presentation Layer)

Представительный уровень изменяет форму передаваемой информации, но не изменяет ее содержания. Например, средствами этого уровня может быть

выполнено преобразование информации из одной кодировки в другую. Также на этом уровне выполняется шифрование и дешифрование данных.

Пример протокола представительного уровня: SSL (Secure Socket Layer), обеспечивающий секретность обмена данными.

#### Прикладной уровень (Application Layer)

Прикладной уровень представляет собой набор разнообразных протоколов, с помощью которых пользователи сети получают доступ к совместно используемым ресурсам. Единица данных называется сообщением.

Примеры протоколов: HTTP, FTP, TFTP, SMTP, POP, SMB, NFS.

### 1.3.4. Интернет и модель OSI

При взаимодействии открытой системы и Интернета модель OSI упрощается, так как некоторые протоколы Интернета включают в себя функции нескольких уровней. Если к сети Интернет подключается один пользователь, а не вся сеть, то автоматически исчезают канальный и физический уровни, потому что нет сетевых адаптеров, а значит, нет и физических адресов. В данном случае конечным протоколом будет протокол типа «точка-точка», например PPP. В этот протокол будут вложены все остальные.

**Таблица 1.1**. *Соответствие уровней стека TCP/IP модели OSI*

| Уровень модели OSI | Протокол | Уровень стека TCP/IP |
|---|---|---|
| 7,6 | WWW (HTTP), FTP, TFTP, SMTP, POP, telnet, WAIS, SNMP | 1 |
| 5,4 | TCP, UDP | 2 |
| 3 | IP, ICMP, RIP, OSPF, ARP | 3 |
| 2,1 | Ethernet, PPP, SLIP | 4 |

Стек TCP/IP разработан по заказу Министерства обороны США для сети ARPAnet, которая стала прототипом современного Интернета. Стек стал популярным благодаря развитию операционной системы Unix, которая его использовала.

На нижнем уровне стек поддерживает все популярные стандартные протоколы физического и канального уровней: Ethernet, Fast Ethernet, Token Ring, FDDI, SLIP, PPP, ISDN.

Основными протоколами стека TCP/IP, как нетрудно догадаться, являются протоколы TCP и IP. Протокол TCP гарантирует надежность доставки па-



http://all-ebooks.com

кета, а IP – его продвижение по сети. В модели OSI протокол IP относится к сетевому уровню, а TCP – к транспортному.

За долгие годы развития стека TCP/IP для него было разработано много протоколов прикладного уровня: HTTP (Протокол Передачи Гипертекстовой Информации), FTP (Протокол Передачи файлов), TFTP (Тривиальный FTP), SMTP (Простой Протокол Передачи Почты), telnet (протокол эмуляции терминала) и др. Все эти протоколы относятся к представительскому и прикладному уровням модели OSI.

## 1.4. АДРЕСАЦИЯ В СЕТИ ИНТЕРНЕТ

### 1.4.1. Вид IP-адреса и способы его назначения

Любому компьютеру в IP-сети (TCP/IP-сети) назначен уникальный адрес, который называется IP-адресом. IP-адрес – это 32-разрядное число, которое принято записывать в десятичном формате в виде четырех чисел, разделенных точками, например:

111.111.213.232

127.0.0.1

192.168.9.2

При условии, что ваша сеть подключена к Интернету, протокол TCP/IP обеспечивает работу вашей сетевой программы с любым компьютером в мире, как будто тот находится в локальной сети. Уникальность IP-адреса достигается достаточно просто – IP-адреса назначаются централизованно Сетевым Информационным Центром (NIC, Network Information Center).

Для понимания остальной информации нужно отметить, что существуют локальные (LAN, Local Area Networks) и региональные (Wide Area Networks) сети. Сеть Интернет сначала была региональной (ARPAnet), а потом стала глобальной, объединив все региональные сети мира. Если ваша локальная (или даже региональная) сеть не соединена с Интернетом, то внутри сети вы можете использовать любые IP-адреса без согласования с NIC. Обычно в локальных сетях используются особые IP-адреса, о которых мы поговорим немного позже.

Любую сеть, независимо от типа (LAN или WAN), можно разделить на подсети. Причины разбиения сети на подсети кроются в ранних версиях протокола IP. Тогда существовало несколько сетей класса A, содержащих не-

http://all-ebooks.com

сколько миллионов узлов (о классах читайте далее). Помимо всего прочего, в таких сетях очень велика вероятность коллизий, то есть одновременного доступа двух или более узлов к среде передачи данных. Управлять такой сетью крайне неудобно, да и сеть будет перегружена собственным трафиком. Чем меньше сеть, тем эффективнее можно ею управлять.

К другим причинам разделения относят возможность использовать в разных подсетях различные технологии (Ethernet, Token Ring, FDDI, ATM). Вы не можете смешивать эти технологии в одной подсети, однако их можно объединить в одной сети, разбитой на подсети.

Разделение на подсети может быть также произведено из соображений безопасности.

Как я уже писал, каждый компьютер в сети имеет свой уникальный адрес. Но оказывается, что и сеть (подсеть) также имеет свой уникальный адрес. Под сетью можно понимать «пачку» IP-адресов, идущих подряд, то есть 192.168.1.0 – 192.168.1.255. Самый младший и самый старший адреса резервируются. Младший (192.168.1.0) является адресом сети, а старший является широковещательным (broadcast) адресом сети. Адрес сети может потребоваться, когда нужно указать всю сеть (подсеть), например, при задании маршрутизации для этой сети.

Представьте, что у вас есть две отдельные сети и вам нужно объединить их в одну. Тогда эта одна «большая» сеть станет называться сетью, а две «маленькие» – подсетями. Устройство, которое будет обеспечивать связь этих сетей (маршрутизацию), называется, как уже было отмечено выше, маршрутизатором. Маршрутизатор может быть как аппаратным (отдельное устройство), так и программным. В роли программного маршрутизатора может выступать любой компьютер с двумя (или более) сетевыми интерфейсами, например двумя сетевыми платами. В качестве операционной системы может быть установлена любая сетевая операционная система, поддерживающая перенаправление пакетов IPv4-Forwarding. Такой операционной системой может быть Linux, FreeBSD, любая UNIX-система, Windows NT/2000.

Для каждой подсети определена ее маска. Фактически маска – это размер сети, то есть число адресов в сети. Маску принято записывать в десятичнопобайтном виде:

255.255.255.0 – маска на 256 адресов (0...255)

255.255.255.192 – маска на 64 адреса (192...255)

255.255.0.0 – маска на 65536 адресов (256x256)



http://all-ebooks.com

### 1.4.2. Классы IP-сетей

IP-сети делятся на пять классов: A, B, C, D и E, причём два последних класса зарезервированы для специальных целей.
Сети класса A – это огромные сети. Маска сети класса A: 255.0.0.0. В каждой сети такого класса может находиться 16777216 адресов. Адреса таких сетей лежат в промежутке 1.0.0.0 – 126.0.0.0, а адреса узлов (компьютеров) имеют вид 125.*.*.*

Сети класса B – это средние сети. Маска такой сети — 255.255.0.0. Эта сеть содержит 65536 адресов. Диапазон адресов таких сетей 128.0.0.0 – 191.255.0.0. Адреса узлов имеют вид 136.12.*.*

Сеть класса C — маленькие сети. Содержат 256 адресов (на самом деле всего 254 узла, так как номера 0 и 255 зарезервированы). Маска сети класса C – 255.255.255.0. Интервал адресов: 192.0.1.0 – 223.255.255.0. Адреса узлов имеют вид: 195.136.12.*
Класс сети определить очень легко. Для этого нужно перевести десятичное представление адреса сети в двоичное. Например, адрес сети 128.11.1.0 в двоичном представлении будет выглядеть так:

```
10000000 00001011 00000001 00000000
```

А 192.168.1.0 — так:

```
11000000 10101000 00000001 00000000
```

Если адрес начинается с последовательности битов 10, то данная сеть относится к классу B, а если с последовательности 110, то к классу C.
Если адрес начинается с последовательности 1110, то сеть является сетью класса D, а сам адрес является особым – групповым (multicast). Если в пакете указан адрес сети класса D, то этот пакет должны получить все узлы, которым присвоен данный адрес.
Адреса класса E зарезервированы для будущего применения. В табл. 1.2 приведены сравнительные характеристики сетей классов A, B, C, D и E.

**Таблица 1.2.** *Характеристики сетей различных классов*

| Класс | Первые биты | Диапазон адресов | Количество узлов |
|---|---|---|---|
| A | 0 | 1.0.0.0 — 126.0.0.0 | 16777216 ($2^{24}$) |
| B | 10 | 128.0.0.0 — 191.255.0.0 | 65536 ($2^{16}$) |
| C | 110 | 192.0.1.0 — 223.255.255.0 | 256 ($2^{8}$) |
| D | 1110 | 224.0.0.0 — 239.255.255.255 | Multicast |
| E | 11110 | 240.0.0.0 — 247.255.255.255 | Зарезервирован |

http://all-ebooks.com

Теперь самое время немного сказать о специальных адресах, о которых я упомянул немного выше. Если весь IP-адрес состоит из нулей (0.0.0.0), то он обозначает адрес того узла, который сгенерировал этот пакет.

Адрес 255.255.255.255 – это широковещательный адрес. Пакет с таким адресом будет рассылаться всем узлам, которые находятся в той же сети, что и источник пакета. Это явление называется ограниченным широковещанием. Существует также другая рассылка, которая называется широковещательным сообщением. В этом случае вместо номера узла стоят все единицы в двоичном представлении (255). Например, 192.168.2.255. Это означает, что данный пакет будет рассылаться всем узлам сети 192.168.2.0.

Особое значение имеет IP-адрес 127.0.0.1 – это адрес локального интерфейса. Он используется для тестирования сетевых программ и взаимодействия сетевых процессов. При попытке отправить пакет по этому адресу данные не передаются по сети, а возвращаются протоколам верхних уровней, как только что принятые. При этом образуется как бы петля, поэтому адрес 127.0.0.1 называется также loopback (англ. «обратная петля»). В IP-сети запрещается использовать IP-адреса, которые начинаются со 127. Любой адрес подсети 127.0.0.0 относится к локальному компьютеру, например: 127.0.0.1, 127.0.0.5, 127.77.0.6.

#### IP-АДРЕСА ДЛЯ ВНУТРЕННЕГО ИСПОЛЬЗОВАНИЯ

Существует также специальные адреса, которые зарезервированы для несвязанных локальных сетей, то есть сетей, которые используют протокол IP, но не подключены к Интернету. Вот эти адреса:

1. 10.0.0.0 (сеть класса A, маска сети 255.0.0.0).
2. 172.16.0.0 -- 172.31.0.0 (16 сетей класса B, маска каждой сети 255.255.0.0).
3. 192.168.0.0 -- 192.168.255.0 (256 сетей класса C, маска каждой сети 255.255.255.0).

В этой книге я старался использовать именно такие адреса, чтобы не вызвать пересечение с реальными IP-адресами.

### 1.4.3. URL-АДРЕСА И ДОМЕННЫЕ ИМЕНА

URL (Universal Resource Locator) - это универсальный идентификатор ресурса. Чтобы просмотреть какую-нибудь веб-страницу (точнее, запросить


http://all-ebooks.com

любой документ через Интернет), вы должны знать ее имя и адрес, а также протокол, с помощью которого происходит передача веб-страницы. Все это вместе (протокол, адрес и имя) называются URL. Для веб-страниц обычно указывает протокол http.

Примеры URL:

```
http://www.dkws.org.ua/
ftp://server.ru/apache-php-mysql.zip
```

Рассмотрим следующий URL:

```
http://www.dkws.org.ua/text/main.html
```

Из чего состоит имя сервера `www.dkws.org.ua`? Для ответа на этот вопрос вспомним, для чего предназначена служба DNS. Правильно, для преобразования символьного имени компьютера в его IP-адрес. Так как компьютеров в Интернете очень много, то база данных серверов DNS должна быть иерархической. Предположим, что компьютеров у нас мало. Тогда базу данных можно было бы сделать «плоской», например такой:

```
Имя IP-адрес

host1  111.111.111.111
host2  111.111.111.112
...
hostN  111.111.NNN.NNN
```

Представляете, какого размера был бы файл, содержащий такую таблицу? А сколько бы длился поиск нужного имени? Поэтому для удобства (и, конечно же, для оптимизации поиска) все компьютеры в Интернете разбиты на группы – зоны или домены. Обычно на поддержку одного домена выделяют два компьютера - первичный и вторичный серверы DNS. Если домены маленькие, можно настроить один сервер так, чтобы он поддерживал несколько доменов. Так обычно и делается в относительно небольших корпоративных сетях.

Вернемся к имени сервера `www.dkws.org.ua`. Здесь `ua` – это домен, принадлежащий Украине. Домен может содержать поддомены: `org` – это поддомен домена `ua`. В свою очередь, `dkws` – это поддомен домена `org.ua`. А что такое `www`? А это и есть имя компьютера в домене `dkws.org.ua`. Возможно, в домене не будет настоящего имени `www` (ну никто из пользователей не захотел так назвать свой компьютер). Тогда администратор домена создаст псевдоним `www` для какого-нибудь компьютера, например для компьютера server1, на котором запущен веб-сервер. В следующем пункте мы подробно рассмотрим работу службы DNS.

http://all-ebooks.com

## 1.5. СЛУЖБА ДОМЕННЫХ ИМЕН DNS

Для преобразования IP-адреса в символьное имя и обратно используется служба доменных имен – DNS (Domain Name System). Обычно на любом сервере устанавливается своя служба DNS, даже если этот сервер не поддерживает домена. В отличие от одноранговой сети, в IP-сети компьютеры объединяются в домены, а не в рабочие группы. На самом деле понятие домен гораздо шире, чем рабочая группа, но пока остановимся на таком определении.

В предыдущем пункте мы упомянули термин «зона». Чем отличается домен от зоны? С доменом все ясно – это группа компьютеров с одинаковой правой частью доменного имени. Если у нас есть домен `company.com`, то его составляют все компьютеры, у которых доменное имя заканчивается строкой `company.com`. Однако, домен `company.com` большой. Кроме того, он содержит поддомены `bill.company.com`, `sec.company.com`, `sales.company.com`, `xyz.company.com` и т. п. Мы можем управлять всем доменом `company.com` на одном сервере имен или можем создать независимые серверы имен для некоторых поддоменов.

Пусть первый (главный) сервер имен обслуживает домен `company.com` и поддомены `sec.company.com`, `sales.company.com` и `dev.company.com`, а второй – домены `mail.company.com` и `ftp.company.com`. Тогда домены `company.com`, `sec.company.com`, `sales.company.com` и `dev.company.com` образуют одну зону, а `mail.company.com` и `ftp.company.com` – другую. Таким образом, зона – это часть домена, которая управляется отдельным сервером имен. Зона, содержащая домены низшего уровня, называется «подчиненной зоной» ("subordinate zone").

Предположим, адрес веб-сервера вашего подразделения выглядит так: `http://www.department.firma.isp.ru`. Рассмотрим, что происходит, когда пользователь вводит в окне браузера этот адрес. Сначала отправляется запрос на разрешение (преобразование) имени в IP-адрес серверу DNS, который принадлежит провайдеру пользователя. Если такое имя есть в кэше DNS-сервера провайдера (для определенности назовем его user-dns), он возвращает IP-адрес, и браузер устанавливает соединение с этим компьютером. Если же такого адреса в кэше сервера DNS не оказалось, DNS-сервер провайдера обращается к серверу, обслуживающему домен наивысшего уровня, то есть к корню дерева (рис. 1.8). Тот обращается к домену `ru` (а этот сервер пусть называется `ru-dns`).

Сервер `ru-dns` в свою очередь обращается к серверу, обслуживающему домен `isp` (это ваш провайдер). Сервер `isp` обращается к серверу,


http://all-ebooks.com

отвечающему за домен firma, а он уже к серверу, отвечающему за домен department, который и возвращает IP-адрес компьютера www.department.firma.isp.ru. Таким образом, получается своеобразная цепочка. Ясно, что если эта цепочка оборвется на каком-нибудь звене, то пользователю, точнее, серверу DNS user-dns, будет сообщено о невозможности разрешения имени компьютера в IP-адрес.

Вся структура службы DNS является иерархической. Существуют домены первого, второго, третьего и так далее уровней. В рассмотренном примере доменом первого уровня является ru, второго – isp, третьего – firma, а department -- четвертого уровня (рис. 1.8).

Корневой домен управляется центром InterNIC. Домены верхнего (первого) уровня (называемые также TLD-доменами, то есть Top Level Domains) назначаются для каждой страны (табл. 1.3).

**Таблица 1.3.** *Обозначения стран по стандарту ISO 3166*

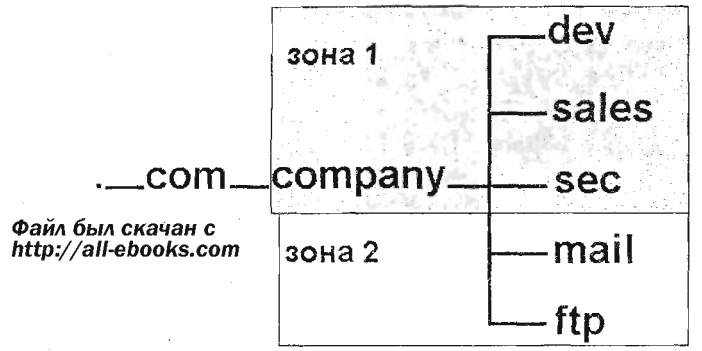
| Домен | Страна | Домен | Страна |
|---|---|---|---|
| Ru | Россия | ua | Украина |
| By | Белоруссия | lt | Литва |
| Lv | Латвия | ee | Эстония |
| Md | Молдова | kz | Казахстан |
| Tr | Турция | ro | Румыния |
| Iq | Ирак | ir | Иран |
| Il | Израиль | tm | Туркменистан |
| Pl | Польша | it | Италия |
| Es | Испания | gb | Великобритания |
| Fr | Франция | de | Германия |
| Id | Индонезия | vn | Вьетнам |
| Gr | Греция | va | Ватикан |
| At | Австрия | co | Колумбия |
| Hu | Венгрия | mx | Мексика |

Для США и Канады единый домен отсутствует, но иногда используется обозначение us. Обозначения стран соответствуют международному стандарту ISO 3166. Данные сведения могут быть получены по адресу ftp:/ftp.ripe.net/iso3166-countrycodes.

Для различных типов организаций могут использоваться такие обозначения:

1. com -- коммерческие организации (например, yahoo.com).
2. edu -- образовательные учреждения (например, mit.edu).
3. gov -- правительственные организации (например, nasa.gov).

http://all-ebooks.com




Файл был скачан с
http://all-ebooks.com

*Рис. 1.7. Домен company.com и две зоны*

4. `org` -- некоммерческие организации (например, `linux.org`).

5. `net` -- обычно провайдеры (например, `ukr.net`).

По данным ISC (Internet Software Consortium) по состоянию на июль 2005 года зарегистрировано более 353 миллионов узлов сети Интернет (рис. 1.9). Для сравнения - в январе 2003 года был «всего лишь» 171 миллион узлов, то есть за два года количество узлов увеличилось в более чем 2 раза. Данные сведения публикуются с разрешения ISC.

## 1.6. WORLD WIDE WEB И HTML-ДОКУМЕНТЫ

World Wide Web переводится с английского как «Всемирная паутина». Сейчас попробуем разобраться, почему именно так, а не иначе. WWW-серверы (их часто называют просто веб-серверами) предназначены для передачи гипертекстовой информации. Для отображения этой информации предназначены специальные программы – браузеры. Вы запускаете браузер, вводите имя веб-сервера и созерцаете веб-страницу.

Чем гипертекстовая информация отличается от обычной текстовой? Основное отличие заключается в том, что гипертекстовый документ может содержать ссылку на другой такой же документ (или любой другой файл). Причем документ, на который ссылаются, может находиться на совершенно другом сервере. В свою очередь, этот документ может содержать ссылки на другие документы. Если представить все это в глобальном плане, то получается паутина ссылок, охватывающая весь мир.

Язык HTML (Hyper Text Markup Language) используется для написания HTML-документов (гипертекстовых документов). Кроме ссылок на другие



http://all-ebooks.com

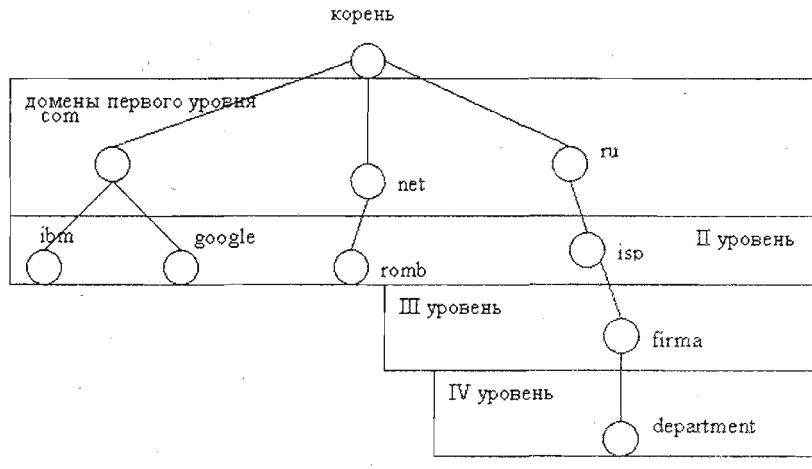



***Рис. 1.8. Иерархическая структура системы доменных имен***

Язык HTML (Hyper Text Markup Language) используется для написания HTML-документов (гипертекстовых документов). Кроме ссылок на другие документы, HTML-документ может содержать форматирование информации для лучшего ее восприятия. Служебная информация заключается в специальные теги. Вот пример небольшого HTML-документа:

**Листинг 1.1. Пример HTML-документа**

```
<html>
<body>
<h1>Привет, мир!</h1>
<a href="doc1.html">Это ссылка на документ doc1.html,
находящийся на этом сервере</a>
<br><a href="http://server.ru/doc2.html">А это ссылка на
документ doc2.html, находящийся на сервере server.ru</a>
</body>
</html>
```

Данный HTML-документ использует теги `html`, `body`, `h1`, `br` и `a`. Подробно на протоколе HTTP (Hyper Text Transfer Protocol) и HTML-документах останавливаться сейчас не будем – в книге мы еще не раз будем говорить о них.

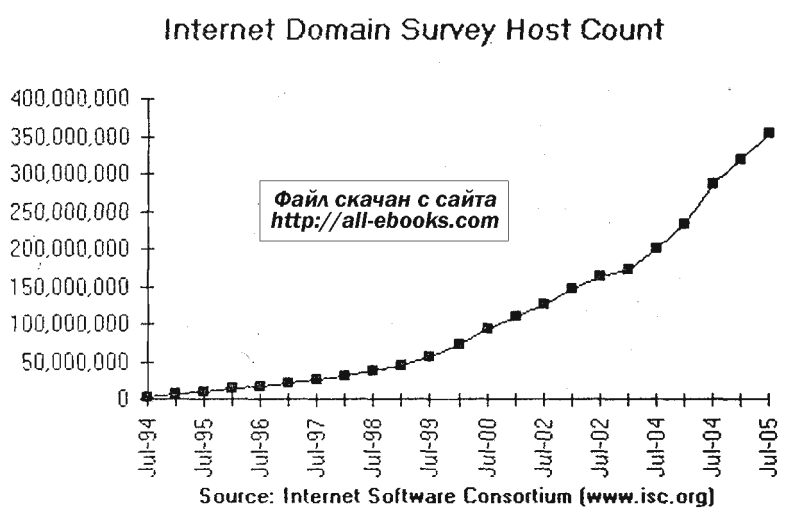
http://all-ebooks.com

*правителя и получателя все понятно, осталось сказать, что же такое порт.* Дело в том, что сразу несколько приложений на одном компьютере могут осуществлять обмен данными через сеть. При этом, если в качестве адресата указывать только IP-адрес получателя, то приложения выполняемые на нем, не смогут разобраться кому из них предназначены присланные данные.

Можно, конечно, каждому приложению назначить свой IP-адрес, однако такой подход приведет к перерасходу адресного пространства.

Для решения этой проблемы служит механизм портов. Каждый порт имеет свой номер. Каждой сетевой программе, которая работает по протоколу TCP/IP, сопоставлен свой номер порта. Например, 80 – это порт WWW-сервера (обычно в качестве него работает программа Apache), а 53 – это порт системы доменных имен.

Термин демон означает программу, которая выполняется в фоновом режиме и дополняет операционную систему каким-нибудь сервисом. Как правило, пользователь не замечает работы демона. Предположим, что наш компьютер получил пакет, в заголовке которого сказано, что данный пакет предназначен для программы с номером 21 (это номер порта демона FTP). Суперсервер (это тоже демон), запущенный на нашем компьютере, ищет программу с номером 21 и запускает ее, передав ей пакет. Если программа




Файл скачан с сайта
http://all-ebooks.com

***Рис. 1.9. Динамика роста узлов в сети Интернет***



http://all-ebooks.com

компьютер получил пакет, в заголовке которого сказано, что данный пакет предназначен для программы с номером 21 (это номер порта демона FTP). Суперсервер (это тоже демон), запущенный на нашем компьютере, ищет программу с номером 21 и запускает ее, передав ей пакет. Если программа с номером 21 не установлена на нашем компьютере, суперсервер сообщает удаленному компьютеру, что «заданный порт недоступен».

Программа с номером 21 может работать и в автономном режиме (standalone), как демон. То есть она сама будет контролировать входящие пакеты, обрабатывая только пакеты, в заголовке которых указан порт с номером 21.

## 1.8. СТРУКТУРА ПАКЕТОВ IP И TCP

Когда мы рассматривали метод коммутации пакетов, мы узнали, что пакет состоит из заголовка и поля данных, содержащего передаваемые данные. Тогда мы были не готовы к подробному рассмотрению структуры заголовка пакеты, но сейчас можем сделать это.

Протокол IP не ориентирован на соединение, поэтому не обеспечивает надежности доставки данных. Поля, описание которых приведено в табл. 1.4, представляют собой IP-заголовок и добавляются к пакету при его получении с транспортного уровня.

Протокол TCP, в отличие от протокола IP, ориентирован на установление соединения и обеспечивает надежную доставку данных. Структура TCP-пакета описана в таблице 1.5.

**Таблица 1.4.** *Структура заголовка IP-пакета*

| Поле | Описание |
|---|---|
| Source IP-address (IP-адрес отправителя) | Отправитель пакета |
| Destination IP-address (IP-адрес получателя) | Получатель пакета |
| Protocol (Протокол) | TCP или UDP |
| Checksum (Контрольная сумма) | Значение для проверки целостности пакета |
| TTL (Time to Live, время жизни пакета) | Определяет, сколько секунд дейтаграмма может находиться в сети. Предотвращает бесконечное блуждание пакетов в сети. Значение TTL автоматически уменьшается на одну или более секунд при проходе через каждый маршрутизатор сети |
| Version | Версия протокола IP — 4 или 6 |

http://all-ebooks.com

Протокол TCP, в отличие от протокола IP, ориентирован на установление соединения и обеспечивает надежную доставку данных. Структура TCP-пакета описана в таблице 1.5.

**Таблица 1.5.** *Структура TCP-пакета*

| Поле | Описание |
|---|---|
| Source port (Порт отправителя) | Порт TCP узла-отправителя |
| Destination Port (Порт получателя) | Порт TCP узла-получателя |
| Sequence Number (Порядковый номер) | Номер последовательности пакетов |
| Acknowledgement Number (Номер подтверждения) | Порядковый номер байта, который локальный узел рассчитывает получить следующим |
| Data Length (Длина данных) | Длина TCP-пакета |
| Reserved (Зарезервировано) | Зарезервировано для будущего использования |
| Flags (Флаги) | Описание содержимого сегмента |
| Window (Окно) | Показывает доступное место в окне протокола TCP |
| Checksum (Контрольная сумма) | Значение для проверки целостности пакета |
| Urgent Pointer (Указатель срочности) | При отправке срочных данных (поле Flags) в этом поле задается граница области срочных данных |

## 1.9. ГЛОССАРИЙ

В этом небольшом пункте приведены описания основных терминов, которые использовались в первой главе, а также некоторых новых.

- **Аутентификация** – проверка подлинности.
- **Интерфейс** –средства и правила взаимодействия компонент системы между собой.
- **Маршрутизатор (router)** – устройство для пересылки пакетов. Маршрутизатор собирает информацию о топологии межсетевых соединений и на основании этой информации пересылает пакеты сетевого уровня в сеть назначения. Маршрутизатор может быть программным и аппаратным.
- **Маршрутизация (routing)** – процесс передачи пакетов данных между двумя подсетями.



http://all-ebooks.com

- **Порт** – физическое или логическое устройство, через которое осуществляется процесс приема-передачи данных.
- **Протокол** – совокупность правил, определяющая взаимодействие абонентов вычислительной системы и описывающая способ выполнения определенного класса функций.
- **Сервер** – специальный компьютер, который предоставляет определенные услуги другим компьютерам.
- **Сокет (socket)** – во многом аналогичен дескриптору файла (file handle). Сокет обеспечивает конечную точку соединения. Приложение, создавая сокет, указывает три параметра: IP-адрес узла, протокол (TCP или UDP) и порт, используемый приложением.
- **Узел (host)** – устройство, подключенное к сети.
- **Шлюз (Gateway)** – межсетевой преобразователь. Выполняет функции, аналогичные мосту, но используется для связи сетей разных типов, например LAN и WAN. Обычно для устройства, которое связывает две локальные сети, используется термин «маршрутизатор», а для устройства, которое соединяет вашу локальную сеть с Интернетом, – «шлюз», хотя оба эти устройства выполняют одну и ту же функцию – маршрутизацию пакетов

http://all-ebooks.com

# ГЛАВА 2.

## ИНТЕРФЕЙС CGI

http://all-ebooks.com

## 2.1. ПРИНЦИПЫ РАБОТЫ CGI

### 2.1.1. Общие положения

CGI (Common Gateway Interface – Общий Шлюзовой Интерфейс) – это набор правил и соглашений, которые должны соблюдаться различными веб-серверами при выполнении веб-приложений.

В этой главе мы рассмотрим традиционное CGI-программирование, почти не касаясь PHP. Это сделано преднамеренно: чтобы вы почувствовали разницу между написанием CGI-программы и PHP-программы. Вскоре поймете, почему язык PHP получил такое распространение в веб-программировании. Вы это узнаете не из моих слов, а исходя из собственной практики.

Предположим, что в поле «Адрес» браузера вы ввели следующий URL: `http://www.server.ru/path/picture.gif`, в результате чего браузер отобразил картинку. Вы уверены, что видите на экране именно содержимое файла `picture.gif` из каталога `path` на сервере `www.server.ru`?

На самом деле файл `picture.gif`, как и путь `path`, могут вообще не существовать. Дело в том, что администратор может настроить псевдонимы каталогов (alias) для любых объектов на сервере так, что действительное имя ресурса останется неизвестным пользователю. При этом производимые над ресурсом манипуляции не будут отличаться ничем, кроме использования псевдонима вместо реального имени.

Однако это еще не самое удивительное. Самое интересное может заключаться в том, что файл `picture.gif` может быть вовсе не картинкой, а

http://all-ebooks.com

программой, которая передаст браузеру картинку в формате GIF. Вот это уже настоящее проявление CGI. Если первого эффекта мы могли достичь средствами сервера, то во втором случае без CGI нам просто не обойтись.

Таким образом, CGI представляет собой мощный механизм формирования документов «на лету». Кроме того, данный механизм абсолютно незаметен для пользователя, которому все равно, каким образом в его браузере появилась картинка – браузер загрузил файл или файл передала ему программа.

### 2.1.2. Взаимодействие сервера с браузером

Что же происходит на самом деле, когда вы вводите какой-нибудь URL в строку адреса браузера? Браузер извлекает из URL протокол (например, `http`) и имя сервера, после чего обращается к указанному серверу согласно указанному протоколу (набору правил).

Компьютер – это «железо», которое ничего не может сделать без программного обеспечения. На компьютере, к которому обращается браузер, должна быть установлена программа – веб-сервер, которая в действительности будет обслуживать наши запросы. Должно быть понятным, что, когда мы говорим «сервер», должны иметь в виду *программу*, а *не компьютер*. Прошу на это обратить внимание.

Браузер определил, что имя сервера есть в базе данных DNS, узнал его IP-адрес и, используя его, подключился к серверу. Сервер ждет команды от нашего браузера. Браузер посылает запрос – сообщение вроде «Мне нужен документ `/path/doc1.html`». Сервер просматривает содержимое каталога `/path`, находит там файл `doc1.html` и возвращает его браузеру как обыкновенный текст. Кроме текста, сервер возвращает также тип документа. Браузер отображает документы согласно информации о типе, переданной сервером. Например, если сервер скажет, что файл `doc1.html` является текстом, то в окне браузера мы увидим код HTML. Обычно сервер так не делает, и мы видим обычную веб-страницу с картинками и форматированием.

Если запрашиваемый документ не найден, сервер может возвратить документ, содержащий сообщение об ошибке или просто код ошибки (например, 500 - внутренняя ошибка сервера). В последнем случае браузер, получив код ошибки, возвращает встроенное в него сообщение об этой ошибке.

### 2.1.3. Роль интерфейса CGI

Интерфейс CGI выполняет функции шлюза (переходника) между различными программами, установленными на веб-сервере, и браузером пользова-



http://all-ebooks.com

теля. Когда браузер запрашивает определенный ресурс, программа-сервер запускает программу, расположенную в каталоге `cgi-bin`, а та уже возвращает результат, который передает пользователю.

> **Примечание.**
>
> Необходимо отметить, что запросы браузера не всегда обрабатываются `cgi`-программами. Очень часто запрашиваемые ресурсы выдаются браузеру напрямую. То есть, если браузер запросил `http://www.server.ru/path/picture.gif`, то ему и выдается файл `picture.gif`.

Использование CGI-программ обычно требуется, когда необходимо в браузере отобразить веб-страницу, сформированную на основе некоторых данных или на основе каких-то действий пользователя на сайте. Например, вы ввели в форму какие-то данные, а на их основе формируется выдаваемая вам страница. Или еще пример: зачастую сайты создаются на нескольких языках. Так вот CGI-программа может по IP-адресу определить вашу страну и сразу отобразить сайт на родном вам языке.

### 2.1.4. Немного подробнее о CGI

Давайте разберемся, что именно происходит при запуске CGI-программы. CGI-программа, – это обычная программа, выводящая что-то. Механизм CGI перенаправляет вывод программы на веб-сервер, а тот, в свою очередь, в браузер пользователя. С точки зрения программы, нет ничего необычного: она «думает», что выводит информацию на экран (stdout), даже не подозревая, что ее вывод будет перенаправлен и его увидит весь Интернет.

Не всякая программа может служить CGI-программой, потому что CGI-программа должна предварять свой вывод определенными заголовками сервера, в частности, указывать тип выводимой информации. Например, если мы хотим отправить картинку в формате GIF, мы должны отправить заголовок `Content-type: image/gif`.

CGI-программа принимает параметры точно так же, как любая программа. Например, если нам нужно передать сценарию `cgi.exe` строку запроса `name=Denis&email=denis@locahost`, то нужно просто вызвать программу `cgi.exe` с параметром:

```
cgi.exe name=Denis&email=denis@locahost
```

Если программа предназначена для CGI, то нужно позаботиться о ее взаимодействии с сервером. Простейшее взаимодействие заключается в том, что программа извлекает информацию о сервере из переменных окружения.

Примерно так работает механизм CGI. Чуть позже мы рассмотрим, как превратить обыкновенную программу на языке C в CGI-программу.

### 2.1.5. Способы создания CGI-сценариев

CGI-программы очень часто называют CGI-сценариями или просто сценариями из-за того, что они написаны, как правило, на каком-нибудь интерпретирующем языке (Perl, Bash и др.). Однако никто вам не мешает написать обыкновенную CGI-программу, например, на языке C. В этом случае у вас будет исполняемый файл (exe-файл в Windows), который будет обрабатывать данные пользователя.

Попробуем создать небольшой CGI-сценарий (листинг 2.1). Напишем его на языке командного интерпретатора CMD.EXE (COMMAND.COM в Windows 9x). По сути, это будет обычный CMD-файл (или BAT - расширение можете выбрать на ваше усмотрение, результат будет одинаковым), выводящий нужную нам информацию:

**Листинг 2.1. Файл test-cgi.cmd**

```
@echo off
echo Content-type: text/html
echo.
echo.
echo Hello!
```

Файл `test-cgi.cmd` нужно поместить в каталог `cgi-bin` вашего веб-сервера (сервер может запускать только программы, находящиеся в этом каталоге).

Рассмотрим, что делает наш сценарий. Сначала он сообщает серверу формат передаваемых данных: `Content-type: text/html`. Затем мы выводим два символа перевода строки (`LF` или `\n`) – команда «echo.», сообщая серверу, что мы больше не будем выводить никаких заголовков и приступаем к выводу данных.

В Linux нет понятия «расширение файла». В Linux тип файла определяется не по его «расширению», а по содержимому. Вы можете написать аналогичный сценарий на языке Bash (Bourne Again Shell):

**Листинг 2.2. Файл test.gif**

```
#!/bin/bash
echo "Content-type: text/html\n\n"
echo "Hello!"
```



http://all-ebooks.com

Файл можно преспокойно сохранить под именем `test.gif` и сделать его исполняемым с помощью команды:

```
chmod  +x test.gif
```

С виду будто бы картинка, правда? Но на самом деле файл можно запустить и он выведет следующую информацию:

```
Content-type: text/html
Hello!
```

Если поместить этот файл в каталог `cgi-bin`, то в окне браузера вы увидите строку «Hello!», но никак не картинку в формате GIF. Точно таким же образом поступают программы, выводящие изображения: сначала они выводят формат (например, `image/gif`), а затем содержимое файла в формате GIF.

Благодаря тому, что в Linux отсутствует понятие «расширение файла», мы можем достичь еще большего уровня прозрачности. В Windows запускаемая программа должна иметь расширение `.com`, `.exe`, `.bat`. Понятное дело, что пользователь догадается, что он имеет дело с программой. В Linux запуск программы происходит совершенно незаметно для пользователя.

## 2.2. МЕТОДЫ ПЕРЕДАЧИ ДАННЫХ GET И POST

### 2.2.1. Передача данных серверу и CGI-сценариям

Программы предназначены для того, чтобы решать определенные задачи. Но кто же будет писать программу, которая решает конкретную задачу? Программы должны решать все задачи определенного класса. Например, следующая программа не является универсальной:

```
a = 2 + 2;
print a;
```

В данном исполнении программа каждый раз будет выводить одно и то же число. Сделаем программу более универсальной:

```
read a, b;
c = a + b;
print c;
```

Точно такая же ситуация сложилась вокруг CGI-сценариев. Для работы сценарию нужно передать определенные параметры, он их обработает и передаст ответ. Поскольку браузер не позволяет вводить параметры напрямую с клавиатуры, он передает их с помощью форм: вы вводите значения

http://all-ebooks.com

параметров в форму, затем браузер складывает все параметры в одну строку – строку запроса – и передает ее сценарию.

Порядок работы любой CGI-программы выглядит следующим образом:

- Получить данные, переданные пользователем.
- Обработать данные.
- Вывести результат.
- Завершить работу.

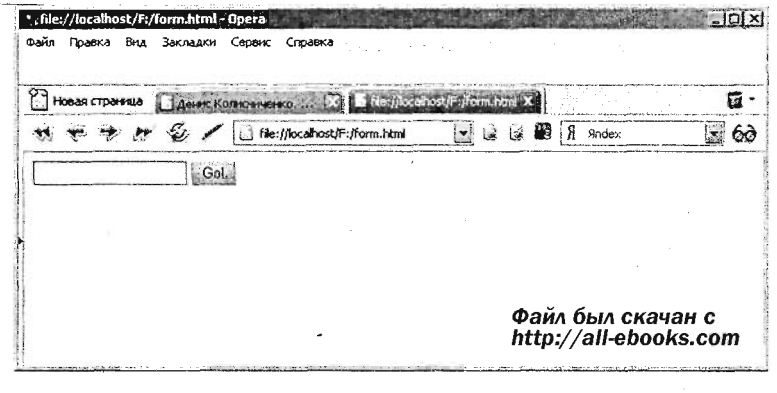
Как вы уже знаете, передача данных пользователя в HTML реализуется с помощью форм. Рассмотрим небольшую форму, передающую имя пользователя (листинг 2.3).

**Листинг 2.3. Пример формы**

```
<form action="/cgi/delete.cgi" method="get">
<input type="text" name="user_name">
<input type="submit" name="go" value="Go!">
</form>
```

Если данную форму сохранить как файл в формате HTML, то в окне браузера вы увидите поле для ввода имени и кнопку для передачи параметра на сервер (рис. 2.1.).

Теперь давайте разберемся с самой формой. Когда пользователь введет имя и нажмет кнопку «Go!», введенное имя будет передано сценарию `delete.cgi` в качестве параметра с именем `user_name`. Метод передачи данных задается параметром `method` тэга `<form>`. Сейчас мы используем




Файл был скачан с
http://all-ebooks.com

*Рис. 2.1. Форма в окне браузера*



http://all-ebooks.com

метод GET. Нажмите кнопку «Go!», и в адресной строке браузера вы увидите примерно следующее

```
http://localhost/cgi/delete.cgi?user_name=Denis
```

> **Примечание.**
>
> Если форма размещена не на сервере (то есть не в каталоге DocumentRoot сервера), вы ничего не увидите, а браузер сообщит вам, что нет файла `delete.cgi`. Желательно создать файл `delete.cgi` в каталоге CGI-сценариев.

Как видите, после имени узла и пути к CGI-программе ей передаются введенные пользователем данные в виде параметров. Параметр при пересылке присоединяется к URL-адресу сценария через знак вопроса (?). Если параметров несколько, они разделяются амперсандом (&).

```
http://localhost/cgi/delete.cgi?user_name=Denis&sex=Male
```

Метод передачи определяет, как именно данные будут переданы сценарию. Метод, при котором передаваемые параметры отображаются в строке браузера, называется методом GET. Если вы не видите передаваемых параметров, значит, используется метод POST.

После того, как параметры переданы сценарию, задача сценария состоит в том, чтобы выполнить разбор строки, то есть выделить пары `имя_параметра=значение`. Теперь рассмотрим оба эти метода подробнее.

## 2.2.2. Метод GET

Ранее было сказано, что взаимодействие сервера (серверной программы) и браузера осуществляется в виде запросов и ответов. Когда мы вводим в адресной строке браузера строку `string`, браузер последовательно выполняет следующие действия:

- Выделяет из нее имя сервера и порт (если он указан);
- Устанавливает соединение с сервером и посылает ему запрос:

```
GET string HTTP/1.0\n
....разная информация....
\n\n
```

В запросе `\n` означает перевод на новую строку, а `\n\n` - маркер окончания запроса. Если сервер не получит маркера окончания, то запрос он обрабатывать не будет.

http://all-ebooks.com

После строки, начинающейся со слова GET (так называемой GET-строки), в запросе могут содержаться другие строки. Эти строки называются заголовками, впрочем, как и сама GET-строка. Заголовков может быть довольно много. Их формирует браузер. Однако для запроса документа нужен только один GET. Чуть позже мы рассмотрим заголовки подробнее, а сейчас займемся разбором переданных сценарию параметров.

Если браузер вызывает сценарий, то, как правило, он передает ему какие-нибудь параметры:

```
GET сценарий?параметр=значение&параметр=значение HTTP/1.0\n
```

Как видите, после имени запрашиваемого документа следует знак вопроса, а за ним - строка параметров.

### Формирование ответа сервером

Для формирования ответа сервера будем использовать специальную *CGI-программу* (именно программу, поскольку мы ее откомпилируем, и она будет запускаться без помощи какой-либо посторонней программы), которую мы напишем на языке C. Эта программа будет только отображать присланные ей параметры в окне браузера, не разбирая их.

Скомпилируйте программу, код которой приведён в листинге 2.4, и поместите исполняемый файл в каталог CGI-сценариев. Обычно это каталог `/var/www/cgi-bin/` (я предполагаю, что вы работаете в Linux).

**Листинг 2.4. Простейшее CGI-приложение**

```
/* cgi.c - простейшее CGI-приложение */
#include <stdio.h>
#include <stdlib.h>

int main()
{
char *Remote_IP = getenv("REMOTE_ADDR");
char *Query_String = getenv("QUERY_STRING");

printf("Content-type: text/html\n\n");

printf("<html><body><h1>Привет!</h1><p>");
printf("Ваш IP: %s",Remote_IP);
printf("<p>Переданные вами параметры: %s",Query_String);
printf("</body></html>");

return 0;
}
```



http://all-ebooks.com

Давайте разберемся, что делает программа. Сначала она получает значение переменной окружения REMOTE_ADDR, а потом – значение переменной QUERY_STRING. Первая содержит значение IP-адреса удаленного пользователя, а вторая – строку переданных параметров. Для обращения к переменным окружения в языке C используется функция `getenv()`.

Затем программа выводит на стандартный вывод значения обеих переменных. Вот здесь и заключается огромная хитрость CGI. Если запустить программу просто в консоли, а не через браузер, то на экране мы увидим:

```
Content-type: text/html
<html><body><h1>Привет!</h1><p>Ваш IP: <p>Переданные вами параметры:
</body></html>
```

Значения переменных окружения REMOTE_ADDR и QUERY_STRING будут пусты, так как эту программу запустили мы сами, а не сервер. Сервер, запустив программу, читает все данные, которые программа выводит на стандартный вывод (stdout) и передает программе пользователя – браузеру.

Если запустить программу, используя браузер, в окне браузера мы увидим (при условии что мы передали такие параметры: `name=Denis&email=den@localhost`):

```
Привет!
Ваш IP: 127.0.0.1
Переданные вами параметры: name=Denis&email=den@localhost
```

Вам, как программисту, осталось сделать немного: разобрать строку запроса на пары «параметр=значение» и создать двумерный массив, сопоставляющий именам параметров их значения. Потом, когда вам понадобится значение параметра, вы переберёте все элементы массива в поисках нужного имени параметра, а по имени извлечёте его значение.

Вот и первое преимущество языка PHP: на C или на каком-либо языке командных сценариев вам пришлось бы писать разбор строки параметров самостоятельно, а из программы на PHP вы можете непосредственно обращаться к значению любого параметра, поставив перед именем параметра знак доллара ($):

```
echo "Ваше имя: $name";
echo "Email: $email";
```

PHP сам создает переменные, которые называются так же, как и переданные параметры. Это значит, что если сценарию передать параметр `email`, то в процессе работы сценария, сразу же после его запуска, появится пере-

http://all-ebooks.com

менная `$email`, с которой можно будет работать как с обыкновенной переменной.

Следует отметить, что такое поведение интерпретатора PHP зависит от директивы `register_globals` в файле конфигурации `php.ini`. Если она включена, то PHP будет работать так, как было описано выше. Об этой директиве мы еще поговорим, но не в этой главе.

### 2.2.3. Метод POST

Мы только что рассмотрели метод передачи GET и все, что с ним связано. Теперь приступим к следующему методу – методу POST. Как я уже говорил, если используется метод POST, переданные сценарию параметры не отображаются в окне браузера. Да и сама передача происходит иначе:

```
POST сценарий HTTP/1.0\n
Content-length: 5\n
\n
Hello
```

Как видите, сам блок пересылаемых данных находится уже после маркера окончания. Это приводит к тому, что сервер никак не интерпретирует полученные данные, а сразу передает их сценарию. В этом и состоит принципиальное отличие метода POST.

Сценарий должен самостоятельно прочитать определенное число байтов, указанное в заголовке `Content-length` (длина слова Hello составляет пять символов). Поэтому при использовании метода POST всегда необходимо указывать значение переменной `Content-length`.

Но вы можете не беспокоиться: всю грязную работу за вас выполнит PHP, и вы сможете обращаться к переданным параметрам так же, как и в прошлом случае: параметры, переданные методом POST, будут преобразованы в переменные, с которыми вы сможете работать в своем сценарии.

Метод POST обычно используется для передачи больших объемов данных (например, для загрузки файлов на сервер) или тогда, когда нужно скрыть значения передаваемых параметров от глаз пользователя. Дело в том, что в отличие от метода GET передаваемые параметры и их значения не отображаются в окне браузера.

## 2.3. Переменные окружения CGI

Мы уже знакомы с двумя переменными окружения – `REMOTE_ADDR` и `QUERY_STRING`. Переменные окружения в языке PHP можно использовать как самые обыкновенные переменные:



http://all-ebooks.com

```
echo "IP: $REMOTE_ADDR";
echo "Параметры: $QUERY_STRING";
```

В этом пункте мы поговорим об остальных переменных окружения, которые можно разбить на четыре большие группы:

- Формируемые сервером переменные.
- Специальные переменные сервера Apache.
- Переменные HTTP-полей запроса.
- Переменные SSL-соединения (защищенного соединения).

Первые три группы мы рассмотрим сейчас, а последнюю – в одной исследующих глав. Все эти переменные представлены в табл. 2.1 – 2.3 соответственно.

**Таблица 2.1.** *Формируемые сервером переменные*

| Переменная | Описание |
|---|---|
| AUTH_TYPE | Используемая схема аутентификации. Обычно 'BASIC' |
| CONTENT_LENGTH | Длина содержимого (например, при передаче POST-данных) |
| CONTENT_TYPE | MIME-тип содержимого, например text/html |
| GATEWAY_INTERFACE | Версия CGI, например, CGI/1.1 |
| PATH_INFO | HTTP-путь к сценарию |
| PATH_TRANSLATED | Полный путь к сценарию |
| REMOTE_ADDR | IP-адрес удаленного пользователя |
| REMOTE_HOST | Доменное имя удаленного компьютера. Доменное имя определятся веб-сервером с помощью службы DNS. Директива HostnameLookups сервера Apache разрешает (или запрещает) преобразование IP-адреса в доменное имя |
| REMOTE_PORT | Порт, закрепленный за браузером для получения ответа от сервера |
| REMOTE_USER | Имя пользователя, прошедшего аутентификацию |
| QUERY_STRING | Строка переданных серверу параметров |
| SERVER_ADDR | IP-адрес сервера |
| SERVER_NAME | Доменное имя сервера. Определяется директивой Server-Name файла конфигурации |
| SERVER_PORT | TCP-порт веб-сервера, обычно 80 |
| SERVER_PROTOCOL | Версия протокола HTTP. Например, HTTP/1.1 |
| SERVER_SOFTWARE | Программное обеспечение сервера |
| SCRIPT_NAME | HTTP-путь к сценарию |
| SCRIPT_FILENAME | Имя файла сценария в файловой системе сервера (физический путь). Например, /var/www/cgi-bin/script.cgi |

**Таблица 2.2.** *Специальные переменные сервера Apache*

| Переменная | Описание |
|---|---|
| DOCUMENT_ROOT | Физический путь к корневому WWW-каталогу сервера. Например, /var/www/html |
| SERVER_ADMIN | Адрес электронной почты администратора сервера |
| SERVER_SIGNATURE | Подпись сервера. Например, Apache/1.3.3 Server at www.firma.ru Port 80 |

**Таблица 2.3.** *Переменные HTTP-полей запроса*

| Переменная | Описание |
|---|---|
| HTTP_HOST | Имя виртуального хоста, которому адресован запрос |
| HTTP_USER_AGENT | Программное обеспечение удаленного пользователя. Обычно данная переменная содержит название и версию браузера |
| HTTP_ACCEPT | Список поддерживаемых клиентом типов содержимого. В последнее время вместо списка браузеры возвращают значение *.*, что означает все типы |
| HTTP_ACCEPT_LANGUAGE | Список поддерживаемых языков в порядке предпочтения, например ru,en. |
| HTTP_ACCEPT_ENCODING | Список поддерживаемых методов сжатия |
| HTTP_ACCEPT_CHARSET | Список поддерживаемых кодировок |
| HTTP_CONNECTION | Тип соединения. Возможны два варианты:<br>• Keep-alive, если после ответа на запрос не нужно разрывать соединения;<br>• Close, если нужно закрыть соединение сразу после ответа на запрос |
| HTTP_REFERER | Значение поля REFERER. В этом поле браузер передает URL ресурса, который ссылался на наш сервер. Например, если пользователь перешел на наш сайт со страницы http://dkws.org.ua/index.html, то значением поля REFERER будет http://dkws.org.ua/index.html. |
| HTTP_X_FORWARDED_FOR | Если пользователь работает через прокси-сервер, то в этом поле будет IP-адрес узла, обратившегося к прокси-серверу. Если данное поле уже содержит значение, то новое значение будет добавлено через запятую |

## 2.4. ЗАГОЛОВКИ HTTP

### 2.4.1. Описание основных заголовков HTTP

Заголовки HTTP используются для «общения» браузера и веб-сервера. Например, когда браузер запрашивает какой-нибудь документ, то он посылает



http://all-ebooks.com

заголовок GET, а когда сервер возвращает тип документа, то он делает это в заголовке Content-type.

В этой главе мы уже говорили о заголовках протокола HTTP, но не рассматривали их достаточно подробно. Основные HTTP-заголовки описаны в табл. 2.4.

**Таблица 2.4.** *Основные HTTP-заголовки*

| Заголовок | Описание |
|---|---|
| Accept | Предназначен для информирования сервера о типах данных, которые поддерживаются клиентом. Используется переменная окружения HTTP_ACCEPT. Пример:<br>Accept: text/html, text/plain, image/gif<br>В последнее время вместо списка типов указывается «*.*», что означает «все типы» |
| Content-type | Предназначен для идентификации типа передаваемых данных. Используется переменная окружения CONTENT_TYPE. Обычно для этого заголовка указывается значение application/x-www-form-urlencoded, указывающее формат, в котором все не алфавитно-цифровые данные кодируются специальным образом. О других MIME-типах вы узнаете чуть позже. Пример:<br>Content-type: text/plain |
| Content-length | содержит строку, в которой записана длина передаваемых данных в байтах при использовании метода передачи POST. За заголовком закреплена переменная окружения CONTENT_LENGTH |
| Cookie | В этом заголовке хранятся все Cookies. О Cookies более подробно мы поговорим позже, а пока лишь отметим, что данный заголовок использует переменную окружения HTTP_COOKIE. Для установки Cookies используется заголовок Set-Cookie |
| GET | На этом заголовке мы останавливались ранее. Формат заголовка следующий:<br>GET сценарий?параметры HTTP/1.0<br>Заголовок GET использует следующие переменные окружения:<br>• REQUEST_URI – запрашиваемый идентификатор ресурса (именно URI, а не URL);<br>• QUERY_STRING – передаваемые сценарию параметры;<br>• REQUEST_METHOD – метод передачи информации. В данном случае эта переменная будет содержать значение GET |

**Таблица 2.4.** *Основные HTTP-заголовки (продолжение)*

| | |
|---|---|
| Location | Получив заголовок Location вместе с указанным в нем URL, сервер обязан немедленно перейти по указанному URL, не дожидаясь, пока тело документа загрузится. Пример:<br>Location: http://www.dkws.org.ua |
| POST | Этот заголовок использует те же переменные окружения, что и заголовок GET (переменная REQUEST_METHOD содержит значение POST). Как уже отмечалось ранее, данные методом POST можно передавать в конце всех заголовков. Пример:<br>POST сценарий?параметры HTTP/1.0 |
| Pragma | Используется для различных целей, одна из которых – это запрет кэширования документа. Пример:<br>Pragma: no-cache |
| Server | Данный заголовок содержит название и версию программного обеспечения сервера. Например:<br>Server: Apache/1.3.23 (Unix) (Red-Hat/Linux) mod_ssl/2.8.7 OpenSSL/0.9.6b DAV/1.0.3 PHP/4.1.2 mod_perl/1.26 configured |
| Referer | С помощью этого заголовка мы можем узнать ссылающийся на нас сервер. Это помогает выявлять обратные ссылки, что особенно бывает полезно при анализе эффективности сетевой рекламы. Переменная окружения: HTTP_REFERER |
| User-Agent | Уточняет версию браузера, например User-Agent: Mozilla/5.0 (compatible; Konqueror/3.0.0-10; Linux). Переменная окружения: HTTP_USER_AGENT |

### 2.4.2. Правила формирования имен переменных окружения

Ознакомившись с названиями заголовков и соответствующих им переменных окружения, вы, наверное, заметили простое правило, согласно которому образуются имена переменных окружения:

- Все символы переводятся в верхний регистр;
- В начале имени добавляется HTTP_;
- Символы "-" заменяются знаком подчеркивания "_".

## 2.5. КОДЫ ОТВЕТОВ HTTP

Знание кодов возврата в большей степени полезно для администратора, но не повредит и рядовому веб-программисту, поскольку последний является администратором своего домашнего веб-сервера.



http://all-ebooks.com

Скорее всего, вы уже знакомы с некоторыми кодами возврата протокола HTTP. По крайней мере, код 404 - `Not Found` - знают все. Иногда с этой ошибкой сталкиваешься несколько раз в день.

На самом деле ошибка 404 является самой распространенной не только потому, что, например, при работе над сложным проектом веб-мастер забыл правильно изменить ссылки или нужный документ был попросту удален. В большинстве случаев ошибка 404 является той единственной, которая показывается пользователю. Например, код нормального завершения операции (то есть страница найдена) - 200, но вы ведь его не видите, когда получаете страницу. А иногда все ошибки из диапазона 400-415 вообще расцениваются как ошибка 404.

Кроме уже известных нам кодов 404 и 200 определено множество других кодов возврата. Всего определено пять категорий кодов, каждая из которых может содержать 100 кодов возврата.

Коды диапазона 100-199 называются *информационными*. С помощью этих кодов возвращается различная информация, относящаяся к запросу.

Диапазон 200-299 является *«диапазоном успеха»*. Коды, которые относятся к этому диапазону, сообщают об удачном завершении той или иной операции. К этому диапазону относятся также коды, сообщающие, что запрашиваемый файл пуст (no content) и что запрос был принят.

Коды 300-399 являются кодами *перенаправления*. Коды 400-499 являются *ошибками клиента*. Например, пользователь может запросить несуществующий документ. К этому же диапазону относятся ошибки «Forbidden» и «Unauthorized». И первая, и вторая сообщает, что у вас нет доступа к какому-нибудь ресурсу, но в первом случае доступ просто запрещен, а во втором вы не являетесь авторизированным клиентом.

Коды 500-599 являются *ошибками сервера*. Эти ошибки возникают при внутренней ошибке сервера или же при неправильном его конфигурировании, когда сервер просто не может правильно возвратить ответ на запрос. Источником ошибок этого типа бывают обычно ошибки сценариев, расположенных на вашем сервере.

Некоторые интерпретаторы сценариев (например, PHP) никогда не допускают такой ошибки. Если в вашем PHP-сценарии есть ошибка, интерпретатор PHP просто возвратит клиенту сообщение об этой ошибке. Со стороны сервера это будет выглядеть как нормальный ответ клиенту, поэтому ошибка 5xx при использовании PHP у вас практически никогда не возникнет. В отличие от PHP интерпретатор Perl генерирует ошибку 500 и записывает сообщение об ошибке в файл протокола сервера HTTP.


http://all-ebooks.com

Краткий (основной) список кодов возврата вы найдете в табл. 2.5.

**Таблица 2.5.** *Таблица кодов возврата HTTP*

| Код | Описание |
|---|---|
| 100 | Продолжить (Continue) |
| 101 | Переключение протоколов (switching protocols) |
| 200 | OK |
| 201 | Создан (created) |
| 202 | Принят |
| 203 | Неавторитетная информация (non-authoritative information) |
| 204 | Файл пуст (no content) |
| 205 | Сброс содержимого (reset content) |
| 206 | Частичное содержимое (partial content) |
| 300 | Множественный выбор (multiple choices) |
| 301 | Перемещен постоянно (moved permanently) |
| 302 | Перемещен временно (moved temporarily) |
| 303 | Смотри другой (see other) |
| 304 | Не измененный (not modified) |
| 305 | Использовать прокси-сервер (use proxy) |
| 400 | Неправильный запрос (bad request) |
| 401 | Неавторизирован (unauthorized) |
| 402 | Платный доступ (Payment Required) |
| 403 | Нет доступа (forbidden) |
| 404 | Не найден |
| 405 | Метод не разрешен (method not allowed) |
| 406 | Неприемлемый (not acceptable) |
| 407 | Требуется аутентификация прокси-сервера (proxy authentication required) |
| 408 | Превышение таймаута запроса (request time-out) |
| 400 | Конфликт |
| 411 | Требуется длина (при использовании метода POST) |
| 412 | Не выполнено предыдущее условие (precondition failed) |
| 413 | Объект запроса слишком велик (request entity too large) |
| 414 | Запрашиваемый URL слишком велик (request-URL too large) |
| 415 | Неподдерживаемый тип данных (unsupported media type) |
| 500 | Ошибка сервера |
| 501 | Не реализовано (not implemented) |
| 502 | Неправильный шлюз (Bad Gateway) |
| 503 | Нехватка ресурсов (out of resources) |
| 504 | Превышен таймаут шлюза |
| 505 | Неподдерживаемая версия HTTP (HTTP version not supported) |



http://all-ebooks.com

## 2.6. ПОДРОБНЕЕ ОБ URL

URL (Universal Resource Locator – Универсальный идентификатор ресурса) – нечто большее, нежели просто адрес. Рассмотрим формат URL:

```
протокол://имя_узла:порт/путь/к/документу/документ?параметры
```

### 2.6.1. Протокол

Протокол указывает браузеру, какой протокол нужно использовать. Как правило, кроме HTTP современные браузеры поддерживают как минимум два протокола: HTTPS (защищенный HTTP) и FTP. В этих случаях вместо предваряющего http:// в начале URL будут стоять https:// или ftp://.

Протокол – необязательная часть URL. Если он не указан, по умолчанию будет использоваться протокол HTTP.

### 2.6.2. Имя узла

После протокола указывается имя узла в формате `узел.домен`. Если домен не указан, то браузер будет считать, что узел находится в локальной сети.

Предположим, что вы ввели в адресную строку `www.firma.ru`. Так вот, `www` – это всего лишь каноническое имя сервера, которое присваивается ему сервером DNS. В большинстве случаев настоящее имя звучит по-другому (например, `denis.firma.ru`). Когда сервер DNS получает запрос о разрешении имени `www.firma.ru`, он сообщает клиенту IP-адрес компьютера `denis.firma.ru`.

Вместо имени узла можно указывать IP-адрес.

### 2.6.3. Порт

Обычно для сервиса HTTP используется порт 80. Но администратор может назначить этому сервису другой порт, например 81. В этом случае требуется указать порт явно.

### 2.6.4. Путь к документу и имя документа

Путь может быть как физическим, так и виртуальным (созданным с помощью псевдонимов).

Документ может быть сценарием. В этом случае строка передаваемых ему параметров отделяется от имени документа знаком "?".

http://all-ebooks.com

### 2.6.5. URL и URI

Какая разница между URL и URI (Universal Resource Identifier)? Очень часто эти два термина путают – и не мудрено: ведь в переводе на русский язык оба эти термина звучат одинаково. URL – это полный физический путь к документу (вместе с указанием протокола и узла), а URI – это часть URL, расположенная после имени узла.

```
http://www.firma.ru/news/index.html - это URL
/news/index.html - это URI
```

## 2.7. ТИПЫ MIME

Типы MIME используются для идентификации типа содержимого. Перечень всех типов MIME, поддерживаемых вашей системой, находится в файле `/etc/mime.types` (или, если вы работаете в Windows, `c:\Program Files\Apache Group\Apache\conf`). Тип MIME указывается в формате: тип/подтип. Вот основные типы MIME:

- application – приложение (программа или документ, связанный с программой). Например, `application/rtf`;
- audio – аудио файл (`audio/mpeg`);
- image – графический файл (`audio/png`);
- text – текст (`text/html` – в формате HTML, `text/plain` - простой).

MIME-тип пересылаемых по сети данных указывается в заголовке Content-type.

## 2.8. PHP И CGI

PHP является основным конкурентом CGI. Сейчас, хоть и написано много CGI-приложений, они постепенно вытесняются PHP. Почему? Да потому, что PHP предназначен именно для веб-программирования: в нём уже решены типичные задачи разбора строки параметров, извлечения заголовков HTTP и т. п., так что вам как программисту не придётся писать эти процедуры самому.

В PHP переданные сценарию параметры автоматически становятся переменными сценария (при условии, что директива `register_globals` включена), с которыми можно работать, как с обыкновенными переменны-



http://all-ebooks.com

ми. То же самое происходит с переменными окружения сервера. Вам не нужно изобретать функции для работы с различными базами данных – MySQL, PostgresSQL, Informix, Sybase, SQLite и др. Все это уже есть в PHP.

К тому же PHP является полноценным языком программирования, синтаксис которого очень напоминает C, что способствует быстрому освоению языка. Библиотека стандартных функций PHP очень богата, и все, что нужно программисту, всегда под рукой.

http://all-ebooks.com

# ГЛАВА 3.

# УСТАНОВКА И НАСТРОЙКА НЕОБХОДИМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

http://all-ebooks.com

## 3.1. ВЫБОР НЕОБХОДИМОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

### 3.1.1. Выбор операционной системы

Для начала необходимо определиться с операционной системой. По большому счету, все равно, какую операционную систему вы будете использовать. Ведь есть версии PHP и для Windows, и для Linux. Для абсолютного большинства проектов так оно и есть. Вы можете спокойно создавать PHP-сценарий на своем Windows-компьютере, а затем отправлять его на сервер, не беспокоясь по поводу того, под какой операционной системой работает сервер (Linux или Windows). Главное, чтобы на нем поддерживались PHP и MySQL.

Однако если вы всерьез решили заняться созданием профессиональных сайтов на основе PHP-сценариев, то вам не обойтись без учета того, под какой операционной системой работает сервер. При этом стоит иметь в виду, что PHP изначально создавался под Unix-подобные операционные системы (Linux, FreeBSD и т. п.) и некоторые детали и возможности до сих пор доступны только под Unix.

Таким образом, использование PHP под Windows не позволяет полностью реализовать как возможности операционной системы, так и возможности PHP. Прошу заметить, что здесь идет речь о профессиональных тонкостях, знание которых подразумевает серьезную подготовку и глубокое понимание функционального устройства ОС. Однако именно эти тонкости отличают профессиональный проект от непрофессионального.

http://all-ebooks.com

Под Windows существует аналог PHP, который называется ASP (Active Server Pages). Он полностью ориентирован на Windows и позволяет учитывать все особенности данной операционной системы. При этом в качестве HTTP-сервера вместо Apache используется IIS (Internet Information Server) – специальный продукт компании Microsoft. Хотя нужно отметить, что можно настроить связку IIS + PHP, но стоит ли это делать? Такую возможность мы рассматривать в книге не будем.

Итак, профессиональное использование PHP ориентировано на Linux (Unix). Поэтому желательно иметь соответствующий хостинг. Тем не менее в книге будут рассмотрены оба варианта установки, настройки и использования PHP: под Linux и под Windows.

Желательно, чтобы на сервере и на вашем компьютере (на котором вы будете создавать и тестировать свои PHP-сценарии) стояли одинаковые операционные системы. Иногда бывает так, что PHP-программа отлично работает на вашем домашнем Windows-компьютере, но отказывается работать или работает некорректно на Linux-сервере. Поскольку для работы PHP-проектов желательно использовать Linux (Unix)-хостинг, желательно, чтобы и на компьютере разработчика была установлена ОС Linux. Если вы рано или поздно решите установить у себя Linux, рекомендую вам свою книгу «Самоучитель Linux. Установка, настройка, использование», вышедшую в издательстве «Наука и Техника». В этой книге вы найдете всю необходимую информацию для работы с этой операционной системой.

**Примечание.**

В издательстве «Наука и Техника» вышли еще две мои книги, посвященные Linux: «Linux-сервер своими руками» и «Linux. Полное руководство». Но эти книги более сложны и адресованы в первую очередь тем, кто хочет создать и настроить свой сервер на основе Linux.

### 3.1.2. Выбор программы

Для изучения языка PHP вам понадобятся следующие программы:

- веб-сервер Apache. Это и есть та самая программа, которая превратит ваш домашний компьютер в настоящий веб-сервер.
- Собственно сам PHP4 или PHP5. Его нужно установить сразу же после Apache.
- Сервер баз данных MySQL. Практически ни один серьезный проект не обходится без использования баз данных, поэтому не стоит откладывать установку MySQL надолго.



http://all-ebooks.com

- Дополнительные модули для Apache (`mod_php`) и PHP (`php_mysql`). Первый из них «встраивает» PHP в сервер Apache, благодаря чему Apache обрабатывает PHP-сценарии «на лету». Второй модуль добавляет в PHP поддержку MySQL - без него вы не сможете работать с MySQL. Стоит отметить, что в Windows эти модули выполнены в виде динамических библиотек (DLL).

Сейчас нужно остановиться и все-таки сделать выбор операционной системы. Linux предпочтительнее, поскольку в большинстве случаев у хостинг-провайдера сервер будет работать под управлением Linux (Unix). С другой стороны, если вы совсем не знаете этой операционной системы, спешить с ее установкой не стоит. Осваивайте PHP под Windows, а когда Linux будет действительно вам необходим, вы всегда сможете его установить.

Если вы выбрали Linux, то могу вас обрадовать: все необходимое программное обеспечение (Apache, PHP, MySQL) уже есть в вашем дистрибутиве, если только вы не выбрали урезанную ознакомительную (например, LiveCD) его версию. Для профессионального освоения Linux и PHP вам нужна полноценная версия Linux, которую вы можете заказать в одном из многочисленных интернет-магазинов.

Вы выбрали Windows? Тогда разрешите сделать вам небольшой подарок. Ранее (в первых трех изданиях книги) я «заставлял» читателей скачивать с сайта www.apache.org версию Apache для Windows, с сайта www.php.net - Windows-версию PHP, а с сайта www.mysql.org - версию MySQL для Windows. После этого вам нужно было установить соответствующие программные продукты и связать их вместе. Но у некоторых пользователей по тем или иным причинам не получалось этого делать.

Отныне все будет проще. Вам нужно будет скачать с моего сайта архив, в котором будет уже настроенный веб-сервер Apache с поддержкой PHP. Все, что вам останется, это скачать и установить MySQL (он заметно большего размера, поэтому я не позволил себе выложить его на свой сайт). Поскольку PHP в архиве уже поддерживает MySQL, то для инициализации связки Apache+PHP+MySQL вам нужно сначала запустить Apache, а потом MySQL , и все.

Стоит отметить, что в связке Apache+PHP+MySQL сервер Apache наиболее сложен в конфигурировании. Он предоставляет столько возможностей, что некоторые пользователи теряются: что означает эта директива, а что – та? Вот поэтому серверу Apache в этом издании книги посвящена отдельная глава, читать которую я рекомендую только Linux-пользователям. Если вы пока не собираетесь профессионально заниматься PHP-разработкой, можете ее пропустить.

http://all-ebooks.com

Нужно также отметить, что на данный момент активно используются две версии Apache - 1.3.x и 2.x. Но для нас на данном этапе версия Apache не имеет значения, поскольку Apache нам необходим только для вывода результатов наших сценариев, поэтому его версия роли не играет - PHP будет одинаково работать как с версией 1.3.x, так и с 2.0.

В этой главе мы поговорим о минимально необходимой настройке нашего программного обеспечения. Начнем мы с Windows.

## 3.2. APACHE + PHP + MYSQL В WINDOWS

Как я уже говорил, Windows-пользователям я сделал небольшой подарок, а именно заархивировал полностью работающую версию Apache 1.3. Да, это не самая новая версия, но, как я уже отмечал, нам для работы ее вполне хватит. Итак, заходим на страничку `http://www.dkws.org.ua`. Потом идем в раздел «Книги». Найдите в списке вашу книгу - рядом с ней вы найдете ссылочку на дополнительный материал для книги. Перейдите на эту страницу - все, что вам останется, - это скачать один из архивов:

- **apache-php-4**: данный архив содержит сервер Apache 1.3 и PHP 4 с поддержкой MySQL.
- **apache-php-5** (*рекомендую*): этот архив содержит Apache 1.3 и PHP 5 с поддержкой:
    - MySQL: вы сможете подключаться к серверу баз данных MySQL;
    - сокетов: вы сможете работать с сокетами;
    - протокола IMAP: вы сможете использовать протоколы IMAP и POP;
    - графической библиотеки GD2: вы сможете создавать и выводить графические изображения с помощью PHP;
    - протокола SOAP: вы сможете строить распределенные веб-сервисы.

Первый архив - это минимально необходимый набор PHP-программиста. Он занимает 710 Кб. Второй архив «весит» 4 Мб и укомплектован заметно лучше. Если вам нужно все-все-все, то вы можете скачать файл `php-5.1.2-Win32.zip` с сайта `www.php.net` (занимает 9 Мб), но в этом случае вам нужно будет самому настраивать связку Apache + PHP.



http://all-ebooks.com

**Примечание**

Если вам нужна поддержка SQLite, то соответствующие динамические библиотеки вы также сможете скачать с моего сайта `http://www.dkws.org.ua`.

## 3.2.1. Установка Apache и PHP4

Итак, предположим, что вы скачали архив `apache-php-4.zip`. Распакуйте его в каталог `c:\`. Если вам нужно установить его в другой каталог, вам нужно будет редактировать файл конфигурации, но об этом мы поговорим позже.

Предположим, что вы распаковали его все-таки в каталог `c:\`. В результате у вас появится подкаталог `c:\web`. В нем будет два подкаталога: `apache` и `www`. Первый содержит сам Apache и его конфигурационные файлы, а второй будет корневым каталогом вашего сервера, куда вы будете помещать свои HTML- и PHP-файлы. Сервер уже настроен и, если вас не раздражает имя сервера, `denserver` (вы его все равно не увидите), вы можете сразу его запустить.

Для этого перейдите в каталог `c:\web\apache` и запустите программу `apache.exe`. Перед запуском убедитесь, что в вашем Windows-каталоге (обычно `C:\Windows`) нет файла `php.ini`, поскольку в нем могут быть настройки, конфликтующие с настройками уже сконфигурированного сервера. Если все нормально, вы должны будете увидеть сообщение:

```
Apache/1.3.12 (Win32) PHP/4.0.6 running...
```

Запускайте ваш браузер и подключайтесь к узлу `localhost` (`http://localhost`):

*Рис. 3.1. Apache запущен*

http://all-ebooks.com

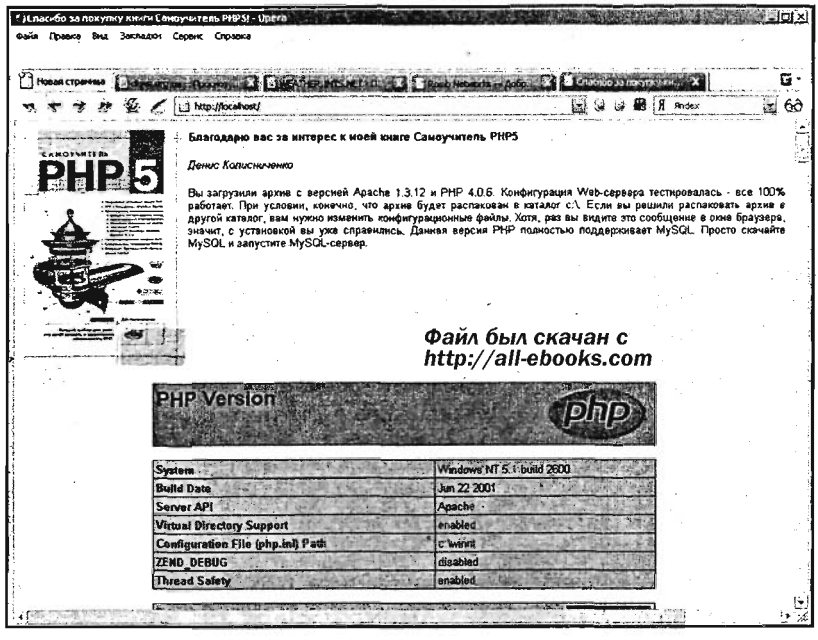



*Рис. 3.2. Apache 1.3.12 и PHP 4.0.6 - работают!*

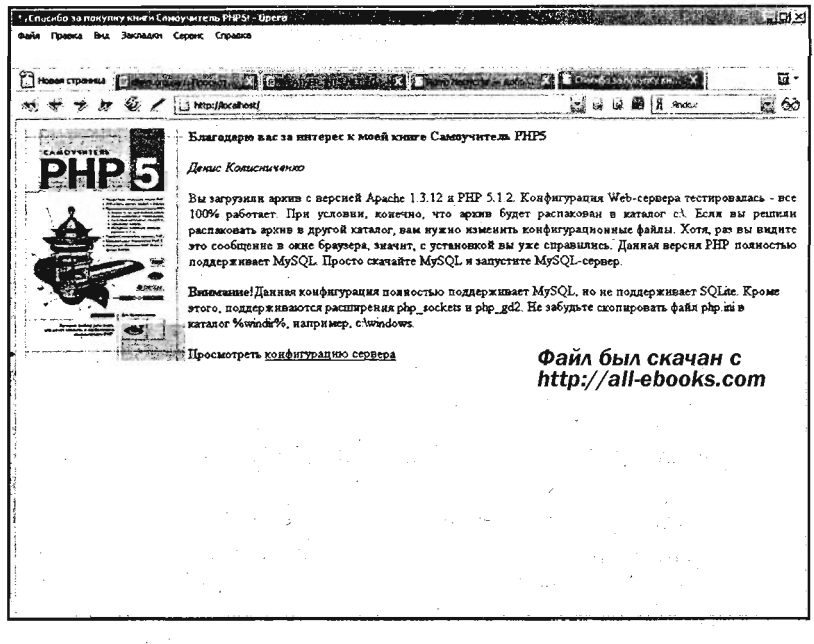



*Рис. 3.3. Apache 1.3.12 и PHP 5.1.2 - работают!*



http://all-ebooks.com

*Рис. 3.4. Apache (php5) запущен*

## 3.2.2. Установка Apache и PHP5

Как и в предыдущем случае, архив apache-php-5 нужно распаковать в каталог `c:\`. После этого у вас появится каталог `c:\web5`. Зайдите в него. В нем будет файл `php.ini`, который нужно скопировать в каталог `c:\windows`.

После этого запустите программу `c:\web5\apache\apache.exe`. Вы увидите сообщение:

```
Apache/1.3.12 (Win32) PHP/5.1.2 running...
```

Теперь проверим, как работает этот сервер. Запустите ваш браузер и подключитесь к узлу `http://localhost` (или `http://127.0.0.1` – напоминаю, что это IP-адрес локального интерфейса). Если вы увидите страничку, изображенную на рис. 3.3, значит, сервер работает правильно.

Как видите, мы только что с минимальными затратами установили веб-сервер Apache и PHP. В следующем пункте мы поговорим об установке Apache в каталог, отличный от `c:\`.

## 3.2.3. Установка Apache в другой каталог

Если вы установили Apache в другой каталог, вам нужно немного изменить файлы конфигурации Apache и PHP. Для Windows-пользователей есть еще один подарок. Я написал небольшой конфигуратор Apache, а вам остаётся только запустить его и выбрать каталог, в который вы распаковали архив. Все остальные параметры можно оставить по умолчанию. Скачать конфи-

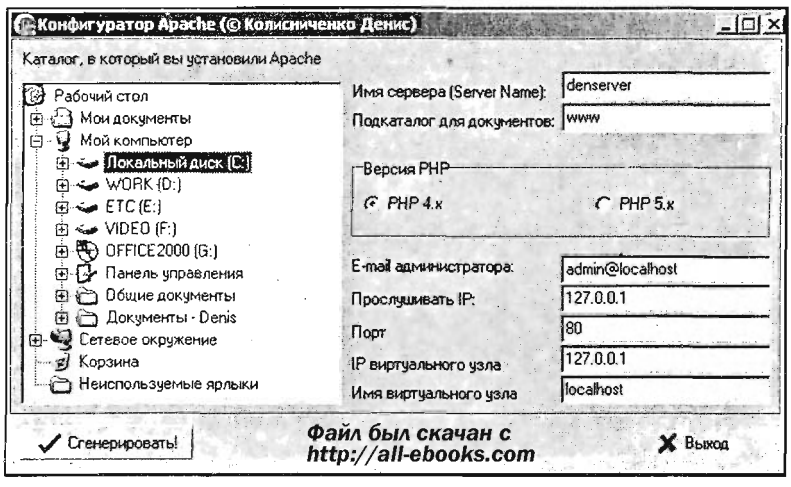



***Рис. 3.5. Конфигуратор Apache***

гуратор `httpdconf` можно с моего сайта `http://dkws.org.ua`. Скачайте архив с конфигуратором и распакуйте его в любой каталог. Запустите программу `httpdconf.exe` (см. рис. 3.5).

Выберите каталог, в который вы распаковали архив. Если вы распаковали его в корневой каталог диска `D:\`, выберите каталог `D:\web` (или `D:\web5`). Остальные параметры можно не изменять. Потом нажмите кнопку **Сгенерировать**. Программа создаст файл `httpd.conf` в корневом каталоге диска C:\. Этот файл нужно скопировать в каталог `<dir>\apache\conf` (`<dir>` - это каталог, в который вы установили Apache). Перед этим не забудьте на всякий случай сохранить оригинальный файл `httpd.conf`.

Если вы любите все держать под контролем, рассмотрим упрощенный листинг файла `httpd.conf`. Внимательно читайте комментарии, и вы все поймете:

**Листинг 3.1. Файл конфигурации httpd.conf Windows-версии Apache**

```
# корневой каталог сервера
ServerRoot "c:/web5/apache"

# имя сервера - данное имя пользователи должны будут вводить в
строку адреса
# браузера, однако, чтобы эта директива работала так, как нужно,
вам необходимо
# зарегистрировать данное имя в системе DNS. Поскольку у вас
домашний
```



http://all-ebooks.com

```
# сервер, регистрировать имя негде, поэтому данное значение роли
не играет
ServerName denserver

# тип сервера - автономный, в Windows - только такой
ServerType standalone

# Все ниже перечисленные параметры мы подробно рассмотрим
# в следующей главе, когда будем рассматривать сервер Apache
# Назначение этих директив одинаковое, как для Windows-, так и для
# Linux-версии
PidFile logs/httpd.pid
ScoreBoardFile logs/apache_status
Timeout 300
KeepAlive On
MaxKeepAliveRequests 100
KeepAliveTimeout 15
MaxRequestsPerChild 0
ThreadsPerChild 50

# Файлы-индексы: эти файлы будет искать веб-сервер,
# когда задана директория, но не указано имя документа
DirectoryIndex index.phtml index.php index.html

# Имя файла доступа
AccessFileName .htaccess

# До следующего комментария ничего не изменяйте
<Files .htaccess>
    Order allow,deny
    Deny from all
</Files>

<Files ~ "\\.inc$">
    Order allow,deny
    Deny from all
</Files>

UseCanonicalName Off
TypesConfig conf/mime.types
DefaultType text/plain
<IfModule mod_mime_magic.c>
    MIMEMagicFile conf/magic
</IfModule>
HostnameLookups On
ErrorLog logs/error.log
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-
Agent}i\"" combined
```

http://all-ebooks.com

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
LogFormat "%{Referer}i -> %U" referer
LogFormat "%{User-agent}i" agent
CustomLog logs/access.log common
ServerSignature On

# Псевдоним каталога с пиктограммами для сервера Apache
Alias /icons/ "c:/web5/Apache/icons/"
AddIconByEncoding (CMP,/icons/compressed.gif) x-compress x-gzip
AddIconByType (TXT,/icons/text.gif) text/*
AddIconByType (IMG,/icons/image2.gif) image/*
AddIconByType (SND,/icons/sound2.gif) audio/*
AddIconByType (VID,/icons/movie.gif) video/*
AddIcon /icons/binary.gif .bin .exe
...
DefaultIcon /icons/unknown.gif

HeaderName HEADER
IndexIgnore .??* *~ *# HEADER* README* RCS CVS *,v *,t
AddEncoding x-compress Z
AddEncoding x-gzip gz tgz
AddLanguage ru .ru
LanguagePriority en ru

AddType application/x-tar .tgz
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-
response-1.0
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0

# Директивы "внедрения" PHP5
LoadModule php5_module modules/php5apache.dll
AddModule mod_php5.c

AddType application/x-httpd-php phtml php php5
AddType application/x-httpd-php-source .phps

# выводим все ошибки
php_value error_reporting E_ALL
php_value magic_quotes_gpc 0

# включаем register_globals
php_value register_globals 1

# Прослушиваем IP 127.0.0.1, порт 80
Listen 127.0.0.1:80
# Имя виртуального узла - 127.0.0.1
```



http://all-ebooks.com

```
NameVirtualHost 127.0.0.1:*

# Описание виртуального узла
<VirtualHost 127.0.0.1:*>
    php_value magic_quotes_gpc 0
    php_value register_globals 1
    ServerAdmin dhsilabs@locahost

# Каталог с HTML-документами сервера. В этом каталоге вы будете
# создавать свои PHP-сценарии и прочие HTML-документы
    DocumentRoot c:/web5/www
    ServerName 127.0.0.1
    ErrorLog logs/127.0.0.1.err
    CustomLog logs/127.0.0.1.log common
</VirtualHost>
```

Скорее всего, в этом файле конфигурации вам не все понятно. Например, зачем нам виртуальный узел и вообще что это такое? Но не спешите: в следующей главе, посвященной настройке Apache, вы найдете ответы на все ваши вопросы.

Если вы выбрали архив с PHP5, то, кроме редактирования файла конфигурации Apache, вам еще нужно отредактировать файл `php.ini`, который должен находиться в каталоге Windows. Откройте этот файл и найдите в нем строку:

```
extension_dir = "c:/web5/php_ext/"
```

Директива `extension_dir` задает каталог, в котором PHP будет искать необходимые ему расширения. Если вы изменили местоположение каталога с Apache, укажите новое имя каталога, например:

```
extension_dir = "d:/wwww/php_ext/"
```

### 3.2.4. MySQL для Windows

Сервер MySQL для Windows можно загрузить с сайта `http://www.mysql.com`. Программа установки установит сервер в каталог `C:\Program Files\MySQL\`. В отличие от Linux-версии Windows-версия MySQL не требует никакой настройки, она готова к работе сразу же после установки. Конечно, желательно изменить пароль пользователя root. О том, как это сделать, вы узнаете в п. 3.3.

Для запуска MySQL запустите программу `c:\Program Files\MySQL\bin\mysqld.exe`. После этого можно запустить программу-клиент (`c:\Program Files\MySQL\bin\mysql.exe`) и попытаться подключиться к серверу.

## 3.3. APACHE + PHP + MYSQL В LINUX

### 3.3.1. Установка пакетов

Установка необходимого программного обеспечения в Linux немного сложнее, чем в Windows, несмотря на то, что все программы уже есть в составе вашего дистрибутива. Сложность заключается в том, что в разные дистрибутивы включены разные версии пакетов, а иногда различаются и имена пакетов. Мало того, в дистрибутивах разных линий отличаются и средства установки пакетов. В Windows же вне зависимости от версии установка программ осуществляется одинаково. Вот поэтому я и предупреждал вас, что нужно знать Linux, прежде чем начать программировать на PHP, иначе вы просто не сможете установить нужные пакеты.

Хотя я, скорее всего, драматизирую – ведь если вы справились с установкой Linux, следовательно, курс молодого бойца вы уже прошли. Настройку связки Apache + PHP + MySQL я буду приводить на примере моего домашнего дистрибутива – Linux Mandrake 10.1. Если у вас другой дистрибутив, например SuSE, то вы сами должны знать, что и к чему. Поймите меня правильно, если описывать установку нашей связки в каждом отдельно взятом дистрибутиве, то получится книжка объемом не намного меньше той, которую вы держите в руках.

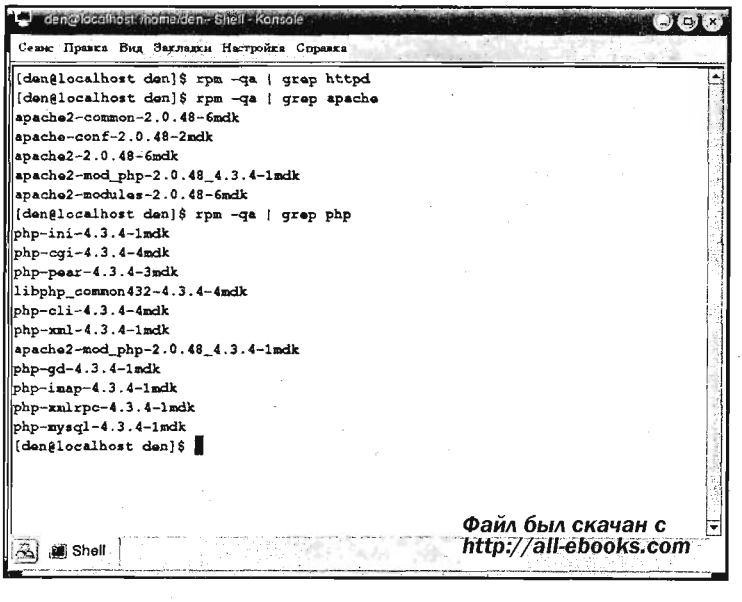



*Рис. 3.6. Apache2 и PHP установлены*



http://all-ebooks.com

Первым делом вам нужно установить Apache. В некоторых дистрибутивах пакет, содержащий этот веб-сервер, может называться `apache`, в других – `httpd`. В Linux Mandrake 10.1 пакет, содержащий веб-сервер, называется `apache`. Для установки программного обеспечения я буду использовать менеджер пакетов RpmDrake, а также утилиту `rpm`.

Перед установкой нужно убедиться, что Apache не установлен, поскольку велика вероятность того, что он уже будет установлен по умолчанию. Для того чтобы убедиться в этом, выполните в консоли (или в X-терминале) команду:

```
$ rpm -qa | grep apache
```

Если в ответ на эту команду вы увидите следующий список пакетов, значит, Apache у вас уже установлен.

```
apache2-common-2.0.48-6mdk
apache-conf-2.0.48-2mdk
apache2-2.0.48-6mdk
apache2-mod_php-2.0.48_4.3.4-1mdk
apache2-modules-2.0.48-6mdk
```

В данном случае у вас установлен Apache версии 2 (пакеты `apache2-common` и `apache2`), утилита конфигурации веб-сервера

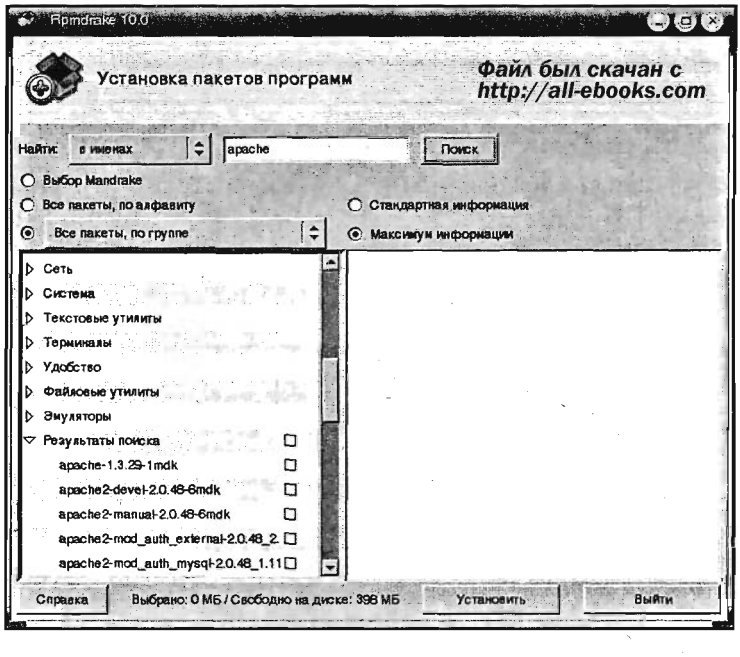



***Рис. 3.7. Менеджер пакетов RpmDrake***

(`apache-conf`), модуль поддержки PHP (`apache2-mod_php`), а также различные дополнительные модули (`apache2-modules`).

Если же список пуст, самое время запустить RpmDrake (рис. 3.7). Произведите поиск по слову apache.

Как вы видите, кроме Apache2, в состав дистрибутива входит и "классика" – Apache версии 1.3.29. Его устанавливать не нужно – зачем, если есть возможность сразу использовать новую версию. Также не нужно устанавливать все пакеты, относящиеся к Apache2. Вам вполне хватит приведенного выше списка пакетов.

Теперь перейдем к установке PHP. В моем случае все пакеты уже были установлены. Но у вас – случай другой, поэтому, чтобы не гадать, введите команду:

```
$ rpm -qa | grep php
```

Вы увидите следующий список пакетов (рис. 3.6):

```
php-ini-4.3.4-1mdk
php-cgi-4.3.4-4mdk
php-pear-4.3.4-3mdk
libphp_common432-4.3.4-4mdk
php-cli-4.3.4-4mdk
php-xml-4.3.4-1mdk
apache2-mod_php-2.0.48_4.3.4-1mdk
php-gd-4.3.4-1mdk
php-imap-4.3.4-1mdk
php-xmlrpc-4.3.4-1mdk
php-mysql-4.3.4-1mdk
```

Давайте разберемся, для чего используется каждый пакет. В этом вам поможет следующая таблица.

**Таблица 3.1.** *Назначение пакетов для PHP*

| Пакет | Описание |
|---|---|
| php-ini-4.3.4-1mdk | Файл конфигурации PHP |
| php-cgi-4.3.4-4mdk | CGI-версия PHP для связки с веб-серверами, отличными от Apache. Для связки с Apache используется пакет apache2-mod_php. В принципе можно связать Apache и с CGI-версией, однако стоит ли это делать, если уже есть специальный модуль для Apache? |
| php-pear-4.3.4-3mdk | PEAR (PHP Extension and Application Repository) – это хранилище расширений и приложений PHP. С помощью PEAR вы сможете найти нужные вам PHP-расширения. PEAR – это своеобразный аналог Perl'овского CPAN |



http://all-ebooks.com

**Таблица 3.1.** *Назначение пакетов для PHP(продолжение)*

| Пакет | Описание |
|---|---|
| libphp_common432-4.3.4-4mdk | Общие библиотеки |
| php-cli-4.3.4-4mdk | Command Line Interface – интерфейс командной строки, используется для запуска PHP из командной строки, а не через Apache |
| php-xml-4.3.4-1mdk | Данный пакет добавляет к PHP поддержку XML |
| apache2-mod_php-2.0.48_4.3.4-1mdk | Добавляет к Apache поддержку PHP |
| php-gd-4.3.4-1mdk | Графическая библиотека gd. Позволяет работать в PHP с графическими изображениями, например создавать графические баннеры |
| php-imap-4.3.4-1mdk | Поддержка протокола IMAP |
| php-xmlrpc-4.3.4-1mdk | Добавляет поддержку XMLRPC в PHP |
| php-mysql-4.3.4-1mdk | Поддержка MySQL |

Теперь, когда Apache и PHP установлены, самое время установить MySQL. Установка MySQL в Windows проблем не вызывает – вам нужно только запустить программу установки, а все необходимые действия произведет именно она. В Linux вам нужно немного потрудиться, но ничего, сейчас все установим и настроим.

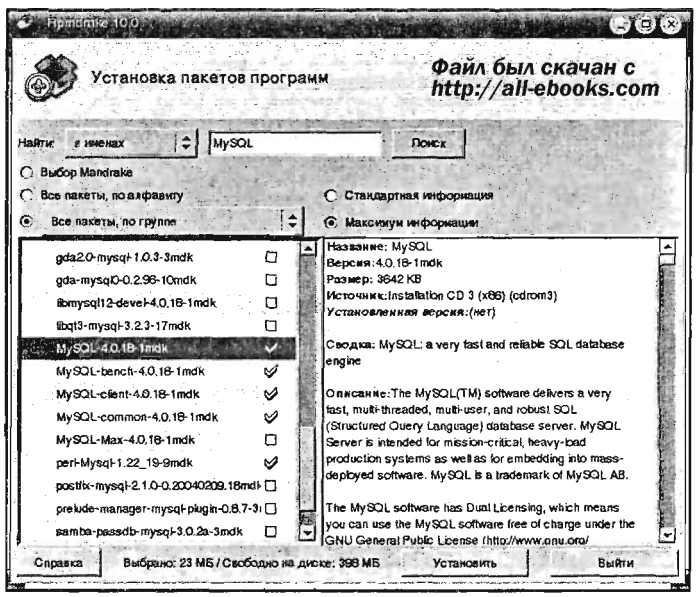



***Рис. 3.8. Установка MySQL***

Как обычно, давайте убедимся, что MySQL не установлен:

```
$ rpm -qa | grep MySQL
```

Скорее всего, в ответ на эту команду вы не получите никакого вывода, поскольку MySQL редко когда устанавливается по умолчанию. Запускаем менеджер пакетов RpmDrake и пытаемся найти MySQL.

На рис. 3.8 изображены все пакеты, которые вы должны установить. Нажимаем кнопочку **Установить** и временно превращаемся в ди-джея, сменяющего диски. Я насчитал три смены диска за все время установки.

## 3.3.2. Настройка MySQL

Все, установка MySQL завершена, но нам еще нужно выполнить настройку этого сервера. Прежде всего вводим команду `mysql_install_db`:

Команда `mysql_install_db` создает системную базу данных mysql. База данных mysql содержит следующие таблицы:

- db
- host
- user

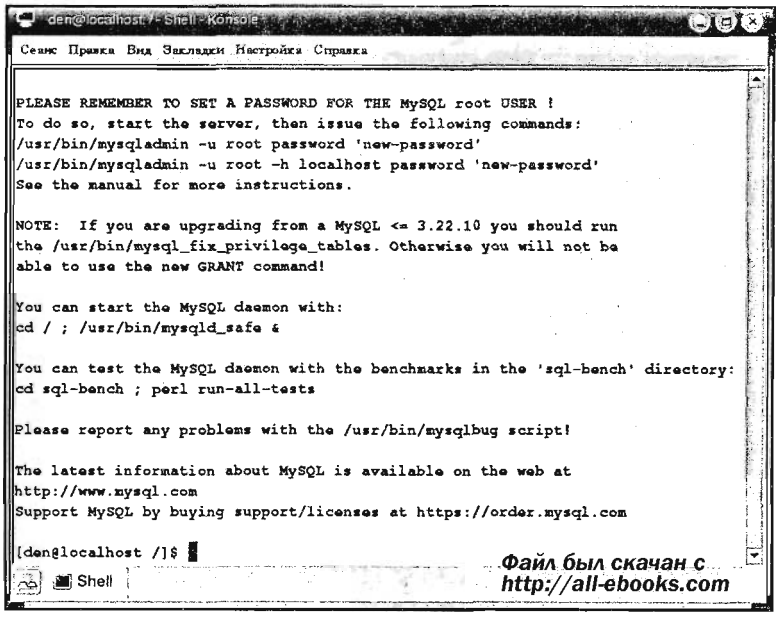



*Рис. 3.9. Создание основной базы данных MySQL*



http://all-ebooks.com

Таблица **user** содержит пароли всех пользователей, в том числе и пользователя root. По умолчанию пароль для пользователя root не устанавливается, то есть после установки MySQL у пользователя root нет пароля.

> **Примечание.**
>
> Системный пользователь root и администратор root сервера баз данных MySQL – это разные пользователи, не следует их путать.

Если вы не планируете с данного компьютера выходить ни в Интернет, ни в локальную сеть, можете не устанавливать пароль пользователя root. Однако если ваш компьютер подключен к Интернету или хотя бы к локальной сети, работа без пароля может привести к печальным результатам: любой желающий сможет подключиться к вашему серверу MySQL с максимальными привилегиями и натворить все, что ему угодно.

Для изменения пароля пользователя root (в частности, для его установки) сначала запустите сам MySQL (все дальнейшие действия нужно выполнять от имени root):

```
# service mysql start
```

После этого подключитесь к базе данных mysql как пользователь root:

```
# mysql -u root mysql
```

Как только запустится MySQL-клиент, введите следующий SQL-запрос:

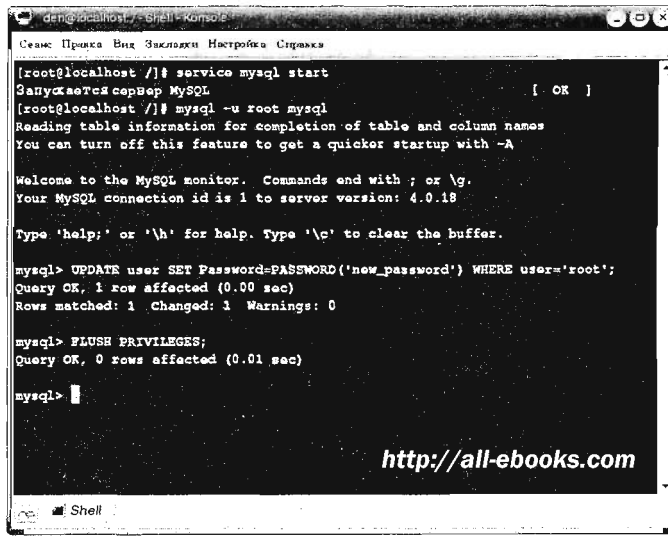



***Рис. 3.10. Изменение пароля***

http://all-ebooks.com

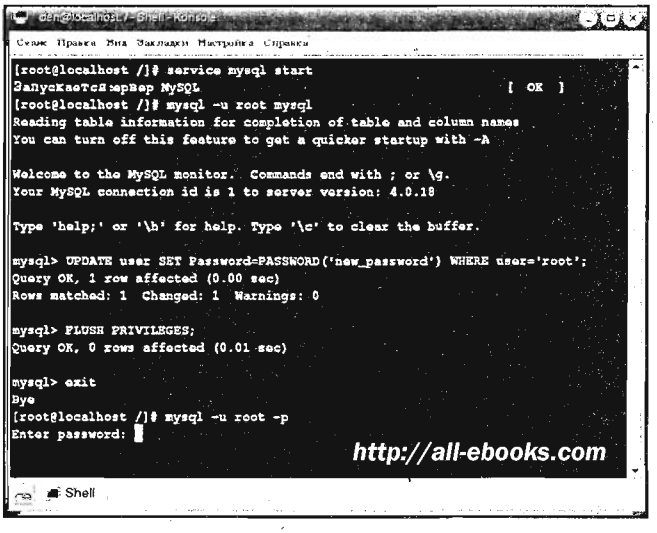



***Рис. 3.11. Вход на сервер с использованием пароля***

```
UPDATE user SET Password=PASSWORD('new_password') WHERE
user='root';
```

Ясно, что вместо пароля `'new_password'` нужно ввести что-нибудь поинтереснее. После этого введите запрос

```
FLUSH PRIVILEGES;
```

и выйдите из клиента (команда **exit**).

Чтобы убедиться, что пароль установлен, попробуем подключиться к серверу еще раз, с использованием пароля:

```
# mysql -u root -p
```

Введите установленный пароль – если сервер вас "пропустит", значит, все нормально.

### Пользователи и привилегии

В своей работе демон **mysqld** использует файл журнала `/var/log/mysql.log`. Именно в него заносятся все транзакции, а также все команды, которые ввел пользователь. После установки сервера нужно внести пользователей, которые имеют право работать с сервером баз данных. Введите следующий запрос:

```
GRANT ALL PRIVILEGES ON *.* TO admin@localhost IDENTIFIED BY
'password' WITH GRANT OPTION;
```



http://all-ebooks.com

Введенный вами запрос создаст пользователя admin, который будет иметь право выполнять любые операции со всеми базами данных. Этот пользователь будет иметь право подключаться к серверу с компьютера localhost, используя пароль `password`.

Маска *.* определяет, к каким базам данных и таблицам имеет право подключаться тот или иной пользователь. Первая звездочка определяет базу, а вторая – таблицу. Если вам нужно, чтобы пользователь admin имел право подключаться с любого хоста, используйте знак процента вместо имени хоста. В этом случае запрос будет выглядеть так:

```
GRANT ALL PRIVILEGES ON *.* TO admin@"%" IDENTIFIED BY ' password'
WITH GRANT OPTION;
```

Вместо всех полномочий вы можете определить, какие действия может выполнять с базой тот или иной пользователь. Если вы являетесь хостинг-провайдером и предоставляете доступ пользователю к его базе данных, то вы можете использовать следующий запрос:

```
GRANT CREATE,DROP,SELECT,INSERT,UPDATE,DELETE,INDEX ON user.* TO
user@% IDENTIFIED BY 'user_password';
```

Перед выполнением данного запроса необходимо создать базу данных **user**. Приведённый запрос позволяет пользователю user выполнять все операции с его базой данных.

Полный список полномочий представлен в табл. 3.2.

**Таблица 3.2.** *Полномочия пользователей сервера MySQL*

| Полномочия | Описание |
|---|---|
| SELECT, INSERT, UPDATE, DELETE | Одноименные операции с данными: пользователь имеет право просматривать, добавлять, модифицировать, удалять данные в таблицах базы данных |
| INDEX | Пользователь имеет право производить операции с индексами таблиц |
| REFERENCES | Пользователь имеет право работать со ссылками в базах данных и таблицах |
| CREATE, DROP | Создание и удаление таблиц и баз данных |
| GRANT, ALTER | Операции с полномочиями |
| RELOAD, SHUTDOWN, PROCESS | Пользователь имеет право перезагружать, останавливать сервер и просматривать все процессы (подключения) |

Если запрос GRANT у вас не работает, то вы можете внести пользователя непосредственно в таблицу user базы данных mysql. Структура таблицы user выглядит следующим образом:

http://all-ebooks.com

```
Host User Password Select_priv Insert_priv Update_priv Delete_priv
Create_priv Drop_priv Reload_priv Shutdown_priv Process_priv File_
priv
```

Поля **Host**, **User**, **Password** – это, соответственно, узел, с которого пользователю разрешается доступ, имя пользователя, пароль пользователя. Все остальные поля задают полномочия для пользователя. Если выполнение какой-нибудь операции разрешено пользователю, соответствующее поле должно быть равным «Y». В противном случае установите значение «N».

Например, нам нужно создать пользователя admin, который имеет все полномочия. Это можно сделать с помощью такого запроса SQL:

```
INSERT INTO user(Host,User,Password,Select_priv,Insert_priv,Update_
priv,Delete_priv,      Create_priv,Drop_priv,Reload_priv,Shutdown_
priv,Process_priv,File_priv) VALUES ('localhost','admin',password(
'4td56ls12'),'Y','Y','Y','Y','Y','Y','Y','Y','Y','Y','Y');
```

При вводе запроса обратите внимание на регистр названий полей. Сервер MySQL различает прописные и строчные буквы! С помощью вышеприведенного запроса был создан пользователь admin, который имеет право регистрироваться на сервере только из узла localhost. Если вам нужно разрешить регистрацию из любого узла сети, используйте знак процента, однако это не совсем корректно с точки зрения безопасности. Пользователь admin обладает всеми возможными привилегиями. Пароль пользователя -- `4td56ls12`.

Для создания обыкновенного пользователя используйте следующий запрос:

```
INSERT INTO user(Host,User,Password,Select_priv,Insert_priv,Update_
priv,Delete_priv) VALUES('%','user',password('123456'),'Y','Y','Y'
,'Y');
```

### 3.3.3. Запуск серверов

Для запуска веб-серверов Apache и MySQL введите команды:

```
service httpd start
service mysql start
```

Обратите внимание, сервис веб-сервера называется `httpd`, а не `apache`. Теперь самое время позаботиться о том, чтобы Apache и MySQL запускались автоматически при запуске системы. Для этого введите команду **drakxservices** и убедитесь, что оба сервиса запускаются при загрузке системы:



http://all-ebooks.com

*Рис. 3.12. Автоматический запуск Apache и MySQL*

## 3.3.4. Тестируем

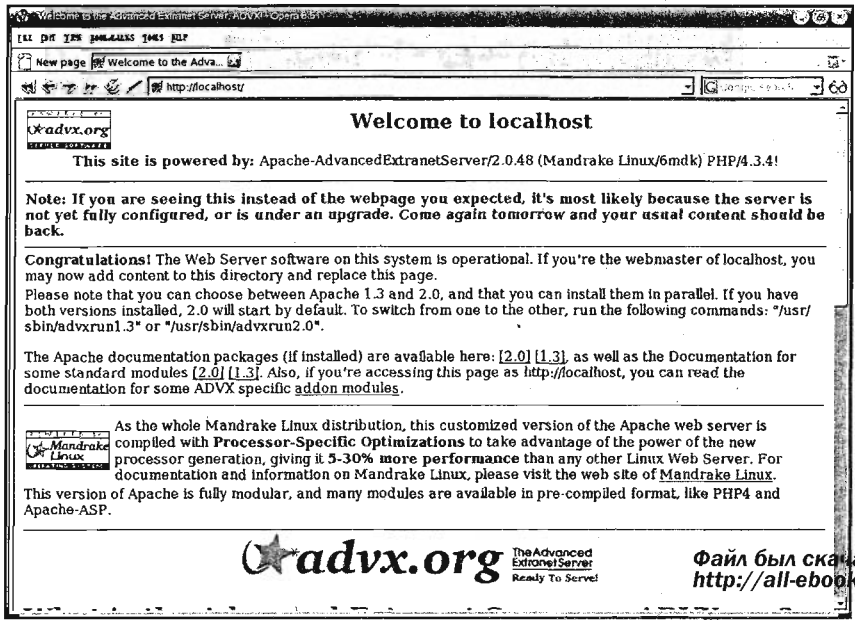
Убедитесь, что Apache и MySQL запущены, запустите любимый браузер и введите в поле адреса `http://localhost`. Вы должны увидеть стандартную страницу приветствия Apache 2.

Нужно отметить, что, начиная со второй версии, Apache уже называется немного иначе: Apache-AdvancedExtranetServer. Как видите, Apache повзрослел, и имя его стало длиннее, солиднее.

Но это еще не все. Создайте файл `test.php` следующего содержимого:

**Листинг 3.2. Сценарий для проверки конфигурации Apache+PHP+MySQL**

```
<?
// Используется имя пользователя root и пароль passwd
if(!mysql_connect("localhost","root","passwd"))
{
echo "Не могу соединиться с сервером\n";
echo mysql_error();
exit;
}
echo "Работает!";
?>
```




Файл был скачан с
http://all-ebooks.com

***Рис. 3.13. Apache работает!***

Сохраните этот файл в каталоге `/var/www/html`. После этого вернитесь в браузер и введите в строке адреса следующий URL:

```
http://localhost/test.php
```

Если после загрузки документа в окне браузера увидите слово «Работает!», значит, вы все сделали правильно.

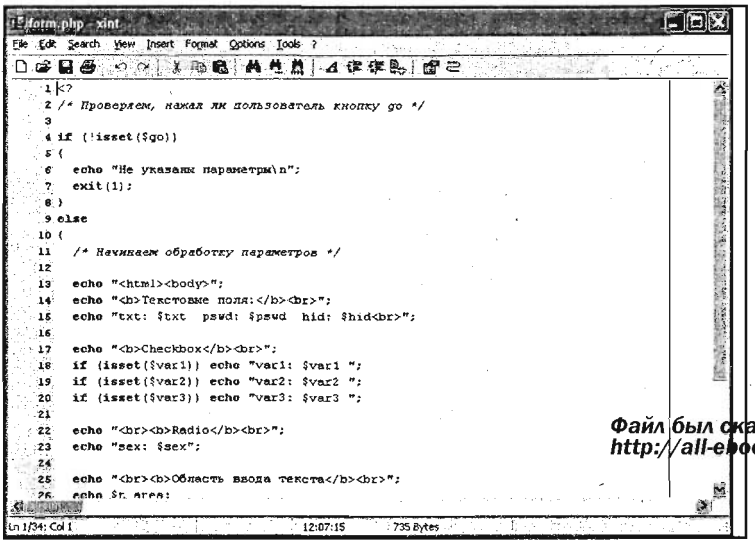
## 3.4. ВЫБОР РЕДАКТОРА PHP-СЦЕНАРИЕВ

Каким редактором вы предполагаете пользоваться для редактирования ваших PHP-сценариев? В Linux этот вопрос и не возникает – там есть из чего выбрать. Редакторов много, причем практически все они поддерживают подсветку синтаксиса PHP. На худой конец можно использовать подсветку синтаксиса для языка C.

В Windows можно использовать Блокнот, но он несколько неудобен. Во-первых, нет подсветки синтаксиса, во-вторых, номера строк не отображаются слева, как бы нам очень хотелось. Можно, конечно, для редактирования использовать редактор HomeSite или другой подобной системы, но это все равно, что из пушки по воробьям стрелять. Я рекомендую использовать



http://all-ebooks.com




Файл был скачан с
http://all-ebooks.com

***Рис. 3.14. Редактор xint***

компактный редактор xint. Этот редактор компактный, надежный, обладает подсветкой синтаксиса не только PHP, но и других языков (HTML, Perl, C, Java, Pascal и т. д.). Скачать редактор можно с моего сайта:

`http://www.dkws.org.ua/progs/xint.rar`

Или с домашней странички автора:

`http://www.xtort.net/index.php`

Для выбора подсветки синтаксиса выполните команду меню **Format, Highlighting, PHP**. В остальном xint - это очень простой и удобный редактор. Исследуйте его меню, чтобы узнать все его функции.

http://all-ebooks.com

# ГЛАВА 4.

## ВЕБ-СЕРВЕР APACHE

http://all-ebooks.com

## 4.1. КАКИЕ ФУНКЦИИ ВЫПОЛНЯЕТ APACHE

Эта глава посвящена популярному HTTP-серверу Apache. Этот сервер возник из веб-сервера NCSA, разработанного в Национальном центре разработок суперкомпьютеров Иллинойского университета. В 1994 году из проекта NCSA ушел главный разработчик, оставив многих последователей самостоятельно разбираться в своем сервере. Со временем начали появляться исправления и дополнения к серверу NCSA — заплатки (патчи).

А в апреле 1995 года вышла первая версия сервера Apache, основанного на версии 1.3 сервера NCSA, которая просто вобрала в себя все известные на тот момент исправления NCSA. Отсюда появилось и само название Apache — «А PatCHy». Позже Apache стал самостоятельной разработкой, и сейчас он поддерживается группой программистов-добровольцев Apache Group.

Сервер Apache разрабатывался для ОС Linux и UNIX, но со временем были выпущены его версии и для ОС Windows и OS/2.

Хочу также отметить, что, кроме Apache, для ОС Linux существуют и другие веб-серверы: Red Hat Secure Server, Apache-SSL, Netscape Enterprise Server и т. д.

Помните, еще в первой главе было сказано, что функции веб-сервера выполняет не компьютер, а программа, установленная на нем? Так вот, Apache и является той самой программой. Когда браузер пользователя подключается к веб-серверу и посылает заголовок GET (это просьба передать файл), его запрос обрабатывает именно Apache. Apache проверяет, существует ли указанный в заголовке GET файл, и если да, то отправляет его вместе с заголовками браузеру.

Apache стал фактическим стандартом веб-сервера в Интернете. Его основным конкурентом является IIS (Internet Information Server) от Microsoft, который работает под Windows. Apache, хотя и существуют его версии под Windows, преимущественно устанавливается на Unix-подобные операционные системы (Linux и FreeBSD). Нужно также заметить, что большинство хостеров используют именно Apache, а не IIS. IIS устанавливается в основном на корпоративных серверах, работающих под Windows. Сейчас мы не будем останавливаться на характеристиках и особенностях работы Apache и тем более не будем сравнивать его с IIS. Лучше перейдем сразу к настройке этого сервера (ведь установили-то мы его в прошлой главе, не так ли?).

Сервер Apache нам просто необходим: без него мы не сможем протестировать наши сценарии.

## 4.2. КОНФИГУРАЦИОННЫЕ ФАЙЛЫ APACHE

После установки Apache следует отредактировать следующие файлы:

- `/etc/httpd/conf/httpd.conf` — основной файл конфигурации. Для Apache 2.x. этот файл может также называться `httpd2.conf`;
- `/etc/logrotate.d/apache` или `/etc/logrotate.d/httpd` (в версии 2.0) — файл ротации журналов;
- `/etc/sysconfig/httpd` — системный файл конфигурации;
- `/etc/init.d/httpd` — файл инициализации Apache.

Для старых версий Apache (до версии 1.3) характерно наличие файлов `srm.conf` и `access.conf`. Обычно эти файлы находятся в каталоге `/etc/httpd/conf`. В файле `srm.conf` задаются параметры документов, которые размещены на сервере. Файл `access.conf` содержит параметры доступа к серверу. Начиная с версии 1.3, рекомендуется все директивы, которые раньше находились в файлах `srm.conf` и `access.conf`, помещать в файл `httpd.conf`.

Файл `httpd.conf` — это основной файл конфигурации сервера. В нем содержится техническое описание работы сервера. Начиная с версии 1.3, появилось несколько дополнительных конфигурационных файлов: `apache-mime.types`, `vhosts/vhosts.conf`, `vhosts/VirtualHomePages.conf`, `vhosts/DynamicVHosts.conf`. В файле `apachemime.types` содержатся типы MIME, поддерживаемые сервером Apache.



http://all-ebooks.com

Файлы `vhosts.conf`, `VirtualHomePages.conf`, `DynamicVHosts.conf` относятся к конфигурированию виртуальных веб-серверов, о которых речь пойдет немного позже.

> **Примечание.**
>
> MIME (Multipurpose Internet Mail Extensions) — многоцелевое расширение электронной почты в сети Интернет. Используется не только при работе с электронной почтой, но и служит для описания различных типов данных (текстовых, графических и т. п.). Описание типа MIME включает в себя наименование типа, подтипа и расширение (например, text/plain txt).

## 4.3. КОНФИГУРАЦИОННЫЙ ФАЙЛ HTTPD.CONF И ЕГО ДИРЕКТИВЫ

Директивой называется параметр сервера, влияющий на его поведение. Директивы бывают простыми и сложными, которые также называются блоками директив. Простые директивы изменяют только одно свойство сервера, например его имя, а блоки директив позволяют изменять сразу несколько свойств какого-нибудь объекта, например каталога.

Как уже отмечалось ранее, файл `httpd.conf` (`httpd2.conf`) содержит практически все директивы, необходимые для работы сервера. Директивы конфигурационного файла сервера Apache можно условно разделить на такие группы:

1. **Общие.** К общим директивам относятся глобальные директивы, влияющие на работу всего веб-сервера. Это директивы ServerName, ServerType, Port, User и Group, ServerAdmin, ServerRoot, PidFile, DocumentRoot, UserDir.
2. **Директивы протоколирования:** ErrorLog, TransferLog, HostnameLookups.
3. **Директивы ограничения доступа:** AllowOverride, Options, Limit.
4. **Директивы управления производительностью:** StartServers, MaxSpareServers, MinSpareServers, а также директива CacheNegotiatedDocs.
5. **Директивы обеспечения постоянного соединения с клиентом:** Timeout, KeepAlive, KeepAliveTimeout.
6. **Директивы настройки отображения каталога.** Оформить отображение каталогов можно с помощью директив настройки отображения каталогов: Directorylndex, FancyIndexing и AddIconByType.

7. **Директивы обработки ошибок.** Директивой обработки ошибок HTTP-сервера является директива ErrorDocument. С ее помощью можно установить реакцию на любую ошибку сервера, например на ошибку 404 (документ не найден).
8. **Директивы перенаправления:** Redirect, Alias и ScriptAlias.
9. **Директивы для работы с многоязычными документами:** AddLanguage и LanguagePriority.
10. **Директивы обработки MIME-типов.** Настроить свой сервер для обработки различных MIME-типов можно с помощью директив DefaultType, AddEncoding, AddType, AddHandler и Action.
11. **Директивы создания виртуальных узлов**: VirtualHost, Listen, BindAddress.

Все эти директивы редактировать вам вряд ли придется: нужно будет задать только значения директив ServerName и ServerAdmin. По умолчанию остальные директивы содержат вполне разумные значения. Далее приведено описание директив, используемых в файле `httpd.conf`. Директивы управления доступом к отдельным каталогам вынесены в отдельный раздел.

### 4.3.1. Общие директивы

Общие директивы изменяют глобальные параметры сервера: его имя, тип, порт, адрес администратора. Значения, указанные глобальными директивами, влияют на работу всего сервера.

**`ServerName`** – директива, которая определяет имя сервера Apache. Здесь должно быть указано официальное имя сервера в таком виде, в котором оно должно появляться в строке адреса браузера. Это имя должно быть зарегистрировано в сервере DNS вашей сети.

**`ServerType`** – директива, которая определяет тип сервера. По умолчанию используется значение `standalone`. Если вы хотите достичь максимальной производительности вашего веб-сервера, не изменяйте эту опцию.

**`ServerAdmin`** – директива, задающая электронный адрес веб-мастера вашего веб-сервера. Если возникнут какие-то проблемы, связанные с работой сервера, то по этому адресу будет отправлено соответствующее сообщение. Обычно используется значение `webmaster@Your_Host.com`. Пользователь webmaster, как правило, реально не существует в системе. Для сопоставления псевдонима webmaster реальному (регистрационному) имени пользователя служит файл псевдонимов электронной почты



http://all-ebooks.com

/etc/aliases, состоящий из строк вида «псевдоним : имя». После модификации этого файла нужно ввести от имени суперпользователя команду newaliases.

В Windows такого файла нет, а псевдонимы задаются с помощью программы настройки установленного у вас почтового сервера.

**Port** – директива, задающая номер порта, который будет использоваться для установки соединения. По умолчанию используется порт 80. Если вы хотите запустить сервер Apache с использованием этого или любого другого порта, номер которого меньше 1024, вы должны обладать правами суперпользователя. Если у вас нет таких прав, вы можете запустить сервер для работы с портом, номер которого превышает значение 1024. Обычно используется номер 8080 или 8000.

**User** и **Group** – определяют идентификаторы пользователя (User) и группы (Group), от имени которых будет работать сервер. Эти идентификаторы присваиваются серверу, если он запущен в автономном режиме. Можно использовать как имена пользователей, так и их числовые эквиваленты (UID). По умолчанию используется имя пользователя nobody. Из соображений безопасности не рекомендуется изменять это значение и присваивать имя реального пользователя. В этом случае веб-сервер получит доступ только к тем файлам, которые разрешены для чтения всем пользователям. Нужно заметить, что указанный пользователь и группа должны существовать в вашей системе. *Ни в коем случае не запускайте сервер от имени пользователя* root!

**ServerRoot** – в этой директиве указывается местонахождение файлов конфигурации сервера Apache. По умолчанию для этих целей используется каталог /etc/httpd.

**DocumentRoot** – директива, определяющая местонахождение корневого каталога документов вашего сервера.

**UserDir** – эта директива задает названия подкаталога в домашнем каталоге пользователя, *из которого берутся документы*. В этом случае вы активизируете возможность использования пользовательских каталогов. Если вы не хотите включать эту возможность, укажите UserDir DISABLED. Более подробно эта директива будет рассмотрена позже.

**PidFile** – с помощью этой директивы указывается имя файла, в котором будет регистрироваться исходный процесс сервера. Идентификатор процесса (PID), записанный в этом файле, можно использовать для останова или перезапуска сервера при написании собственных сценариев. Файл PidFile создается только в том случае, если сервер Apache запущен в автономном режиме.

### 4.3.2. Директивы протоколирования

Директивы протоколирования управляют процессом протоколирования работы сервера. С помощью этих директив вы можете определить, какие события подлежат протоколированию (записи в журналы), а какие нет.

**`HostnameLookups on | off`.** Сервер Apache ведет журнал доступа других компьютеров. Если вы включите данную опцию (on), то в журнал будет записано доменное имя компьютера-клиента. Если эта опция выключена (off), в журнал будет записан IP-адрес клиента. Включение данной опции замедляет работу сервера, так как требуется дополнительное время на ожидание ответа от сервера DNS.

**`ErrorLog`** и **`TransferLog`** – эти директивы определяют расположение журналов сервера Apache. Обычно для этих целей используется каталог `/etc/httpd/logs`, который является ссылкой на каталог `/var/log/httpd` или на любой другой. В журнале errorlog протоколируются диагностические сообщения, а также сообщения об ошибках, порождаемых CGI-сценариями. В журнале `transferlog` протоколируются запросы клиентов. Если включена директива `HostNameLookups`, то вместо IP-адресов клиентов будут регистрироваться имена компьютеров.

### 4.3.3. Директивы управления производительностью

Поэкспериментировав с этими директивами и выбрав оптимальные значения, можно добиться существенного повышения производительности вашего сервера.

Сервер Apache для каждого соединения запускает отдельную копию, которая будет обрабатывать запросы клиента. Управлять запущенными копиями позволяют директивы **StartServers**, **MinSpareServers**, **MaxSpareServers**.

**StartServers**, **MinSpareServers**, **MaxSpareServers**. Директива **StartServers** задает количество копий, которые будут созданы при запуске исходной копии сервера. Исходная копия сервера получает запросы и передает их свободным копиям. Это позволяет равномерно распределить нагрузку между отдельными процессами и повысить производительность сервера, однако на практике все не так хорошо, как хотелось бы. Существенного прироста производительности можно добиться только в случае большой загрузки сервера. По умолчанию запускается пять копий сервера.

Если число поступающих запросов превышает количество запущенных копий сервера, запускаются дополнительные процессы-серверы. Эти процессы не завершаются после обработки своего запроса, а продолжают находить-



http://all-ebooks.com

ся в памяти. Директива **MaxSpareServers** позволяет указать максимальное число таких процессов. Если это количество превышено, то лишние процессы завершаются. Если количество «серверов на подхвате» меньше, чем задано директивой **MinSpareServers**, запускаются дополнительные копии.

Для работы этих директив необходимо, чтобы сервер был запущен в автономном режиме.

**CacheNegotiatedDocs** – эта директива позволяет прокси-серверу, например SQUID, не кэшировать документы, которые не генерируются автоматически, то есть в процессе выполнения различных сценариев. Согласно протоколу `HTTP/1.0`, сервер Apache с каждым пакетом посылает заголовок `Pragma: no-cache` прокси-серверу, что позволяет отключить кэширование документов (в протоколе `HTTP/1.1` вместо Pragma используется Cache-Control). Если вы включите данную директиву, то разрешите прокси-серверу кэшировать документы. К сожалению, далеко не все прокси-серверы отключают кэширование после получения данного заголовка. При написании своего CGI-сценария вам, скорее всего, придется самому выводить заголовок Pragma (или Cache-Control) и мета-тэги, указывающие дату последнего обновления документа.

### 4.3.4. Директивы обеспечения постоянного соединения с клиентом

Данные директивы обеспечивают постоянное соединение с клиентом, а также управляют параметрами установленного соединения.

**Timeout** – задает промежуток времени в секундах, в течение которого сервер продолжает попытки возобновления приостановленной передачи данных. Значение директивы **Timeout** распространяется не только на передачу, но и на прием данных. Если вам нужно получать большие файлы, рекомендую увеличить данное значение.

**KeepAlive** – разрешает постоянные соединения, то есть такие соединения, в которых производится более одного запроса за один раз.

**KeepAliveTimeOut** – данная директива определяет таймаут для постоянного соединения.

**MaxClients.** Иногда поступающих запросов так много, что компьютеру не хватает ресурсов для загрузки новых копий сервера в память и их выполнения. Директива **MaxClients** (значение по умолчанию – 150) определяет

максимальное число копий сервера, которые могут выполняться одновременно.

**MaxRequestsPerChild**. После обработки определенного количества запросов, указанного в этой директиве, копия сервера завершается, а вместо нее запускается новая.

### 4.3.5. Директивы создания виртуальных узлов

Довольно часто пользователь не может позволить себе такую роскошь, как собственный физический веб-сервер (отдельный компьютер, постоянно подключенный к Интернету, на котором запущена программа веб-сервера Apache). Поэтому современные веб-серверы поддерживают виртуальные хосты. У вас будет собственное имя, например `www.firma.ru`, но в действительности ваши файлы не будут размещены на отдельном компьютере `www.firma.ru`, а будут находиться на сервере провайдера (или хостера) – `www.isp.ru`.

Благодаря технологии виртуальных узлов один сервер хостинг-провайдера обслуживает сотни или даже тысячи сайтов.

На своем домашнем веб-сервере вы также можете создать один или несколько виртуальных узлов, если, конечно, вам это нужно.

**Listen** – позволяет вам связывать Apache с определенным IP-адресом и (или) дополнительными портами.

**BindAddress** – данная директива используется для поддержки виртуальных хостов и применяется, чтобы сообщить серверу, какой IP-адрес следует прослушивать. Значением данной директивы может быть «*» (любой адрес), IP-адрес или полное имя домена.

Блок директив `VirtualHost` будет рассмотрен позже, при описании создания и использования виртуальных хостов.

### 4.3.6. Директивы настройки отображения каталогов

Данные директивы позволяют «украсить» ваш веб-сервер, например, сопоставить значок с типом файла или определить оформление каталога.

**Directorylndex** – позволяет задать имя документа, который будет возвращен по запросу, не содержащему никакого имени. С помощью данной директивы



http://all-ebooks.com

можно задать несколько имен файлов. Значением по умолчанию является `index.html index.php index.htm index.shtml index.cgi Default.htm default.htm index.php3`. Например, если вы введете в адресную строку браузера только `http://localhost`, то будет возвращен один из указанных в директиве `DirectoryIndex` документов.

**FancyIndexing.** При получении запроса, не содержащего имени документа, сервер передаст один из файлов, указанных в директиве **DirectoryIndex**. Если этих файлов не существует, клиенту будет возвращено оглавление каталога. При включении директивы **FancyIndexing** в оглавлении каталога будут использованы значки и описания файлов. Если директива **FancyIndexing** выключена, оглавление будет представлено в более простом виде.

**AddIconByType** – сопоставляет значок типу файла. Значок будет использоваться при выводе каталога, если включена директива **FancyIndexing.** Директива **AddIconByType** имеет следующий формат:

```
AddIconByType (TEXT, URL)  mime-type
```

Параметр TEXT определяет текстовое описание типа, которое увидят пользователи, использующие текстовый браузер, или пользователи, у которых отключено отображение рисунков. Параметр URL определяет адрес значка, а параметр **mime-type** – это тип файла, которому сопоставляется значок. Полный перечень MIME-типов приведен в файле apache-mime.types. В качестве имени файла можно задать не только MIME-тип, но и символы, которыми заканчивается имя файла (см. листинг 4.1), но для этого нужно использовать директиву **AddIcon** вместо **AddIconByType**.

**Листинг 4.1. Фрагмент файла httpd.conf**

```
AddIconByType (VID,/icons/movie.gif) video/*
AddIcon /icons/binary.gif .bin .exe
```

Первая директива в листинге сопоставляет типу **video** значок `/icons/movie.gif`. Вторая директива сопоставляет бинарным файлам `*.bin` и `*.exe` значок `/icons/binary.gif`. Значок по умолчанию задается директивой DefaultIcon.

### 4.3.7. Директивы обработки MIME-типов

Как вы помните, в Windows существует такое понятие, как расширение имени файла, указывающее тип данных, хранящихся в файле. По расширению можно связать с файлами определенного типа программу, в которой

они будут открываться по умолчанию. Например, когда вы щелкаете по файлу с расширением `.txt`, запускается Блокнот, а при щелчке на файле с расширением `.doc` будет запущен Word. Точно такой же механизм реализован в Apache. Вы можете сопоставить типу файла обработчик этого типа – программу, которая будет запущена для обработки файла. До того, как появился модуль `mod_php`, программа PHP объявлялась обработчиком файлов с расширением `.php`. Сервер запускал PHP, передавал ему файл, а потом возвращал пользователю результат.

**DefaultType.** Если запрашиваемый клиентом тип не соответствует ни одному из MIME-типов, используется MIME-тип, указанный в директиве DefaultType.

**AddEncoding.** Для сокращения времени передачи файла клиентам используется сжатие данных. Браузеры имеют встроенные программы для распаковки, запускаемые при получении архивов определенных MIME-типов. Именно эти MIME-типы и указываются в директиве AddEncoding.

**AddType** – с помощью этой директивы можно добавить новый MIME-тип, который не указан в файле `apache-mime.types`.

**AddHandler** и **Action**. Директива **AddHandler** позволяет сопоставить определенному MIME-типу какой-либо обработчик. А с помощью директивы Action можно определить какое-нибудь действие для обработчика. Например, вы можете запустить какую-нибудь программу для обработки файла данного типа. Использование этих директив, я думаю, лучше всего продемонстрировать на примере (см. листинг 4.2).

**Листинг 4.2. Применение директив AddHandler и Action**

```
AddHandler text/dhtml dhtml
Action text/dhtml /cgi-bin/dhtml-parse
```

### 4.3.8. Директивы для работы с многоязычными документами

Если на вашем сервере будут выложены документы на разных языках (например, вы делаете две версии сайта – русскую и английскую), целесообразно настроить директивы управления языком. Тогда пользователю не нужно будет щелкать мышкой на ссылке «In English» или «На Русском» – сервер, исходя из настроек браузера, сам определит, какой язык предпочитает пользователь, и отобразит нужную страницу:



http://all-ebooks.com

**AddLanguage**. В большинстве браузеров можно задать предпочитаемый язык. Благодаря этому вы можете предоставлять документы на разных языках. Директива **AddLanguage** сопоставляет расширение файла аббревиатуре языка. Для русского языка используется аббревиатура **ru**, для английского – **en**. В корневом каталоге вашего сервера могут находиться несколько индексных файлов на разных языках. Например, для русского языка нужно использовать имя файла `index.html.ru`, а для английского – `index.html.en`.

**LanguagePriority**. Если на вашем сервере размещены документы на разных языках, то с помощью директивы **LanguagePriority** можно указать приоритеты различных языков. Например, вы установили директиву **LanguagePriority** так:

```
LanguagePriority en ru
```

Клиент вводит в строке адреса своего браузера адрес `http://www.server.com/`. Если в свойствах браузера имеется возможность задать предпочитаемый язык, то возвращен будет файл на нужном языке, если такой существует. Если браузер клиента не поддерживает такую возможность, будет возвращен файл на языке, имеющем наиболее высокий приоритет.

В рассмотренном случае (см. пример) это английский язык. Если файла на нужном языке (например, немецком) не существует, то будет возвращен файл на английском языке. Для того чтобы сервер поддерживал нужный вам язык, предварительно установите правильное значение директивы **AddLanguage**.

### 4.3.9. Директивы перенаправления

Когда требуется перенаправлять пользователя на другой ресурс? Например, вы сменили хостера, и из-за этого изменилось имя вашего сайта. Обычно при таком «переезде» у вас есть определенное время, чтобы сообщить вашим пользователям об этом.

Проще всего установить на старом сервере перенаправление на новый: во-первых, пользователи узнают ваш новый адрес, во-вторых, им не придется вводить этот новый адрес вручную – сервер сделает все автоматически.

А может быть, вы изменили структуру своего сайта, перенеся файлы в другой каталог. Но пользователи не знают об этом и по-прежнему обращаются к старому каталогу. Чтобы не доставлять им неудобств, просто создайте ре-

http://all-ebooks.com

директ на новый каталог, и сервер автоматически перенаправит пользователей на него.

**Redirect**. Используйте директиву **Redirect**, чтобы перенаправить запрос в другой каталог или на другой сервер. Например, вы перенесли данные из каталога `/users/den` в каталог `/den`.

Если при этом старый URL-адрес был `http://www.host.com/users/den`, то новый станет `http://www.host.com/den/`. Используйте для этого следующую директиву:

```
Redirect /users/den /den
```

Можно также перенаправить запрос на другой сервер:

```
Redirect /users/den/ www.den.domain.com
```

При этом допускается использование как нового, так и старого URL-адреса.

**Alias** – с помощью директивы **Alias** можно предоставить доступ не только к файлам, находящимся в каталоге, указанном директивой **DocumentRoot**, и его подкаталогах, но и в других каталогах. По умолчанию определен только псевдоним для каталога `/icons`.

**ScriptAlias** – аналогична директиве **Alias**, но позволяет задать месторасположение каталога для CGI-сценариев.

### 4.3.10. Директивы обработки ошибок

Директива обработки ошибок всего одна, но она очень полезная. Например, произошла ошибка 404 (файл не найден). Вы можете сопоставить этой ошибке URL, на который будет перенаправлен браузер пользователя, если данная ошибка произошла. Обычно перенаправление устанавливают на документ, содержащий логотип сайта и сообщение об ошибке.

**ErrorDocument** – директива, сопоставляющая коды ошибок сервера URL-адресам на этом же сервере.

## 4.4. Директивы управления доступом к отдельным каталогам

Теперь рассмотрим директивы управления доступом к отдельным каталогам. Вы можете определить отдельные параметры для каждого каталога вашего сервера – оформление каталога, параметры доступа к этому каталогу.



http://all-ebooks.com

### 4.4.1. Блок директив Directory

Directory – директива, определяющая свойства каталога (см. листинг 4.3).

**Листинг 4.3. Директива Directory**

```
<Directory />
Options Indexes Includes FollowSymLinks
AllowOverride None
</Directory>
```

Свойства каталога можно указывать в директиве **Directory** или в файле `.htaccess`, который находится в том каталоге, для которого необходимо установить нужные параметры.

В блоке **Directory** могут находиться директивы управления доступом. К ним относятся директивы **AllowOverride**, **Options**, **Limit**. Рассмотрим по порядку все эти директивы. Директива **AllowOverride** может принимать значения, указанные в табл. 4.1.

**Таблица 4.1.** *Значения директивы AllowOverride*

| Значение | Описание |
|---|---|
| None | Сервер Apache будет игнорировать файлы .htaccess. Рекомендую установить данную опцию, так как это повысит производительность сервера |
| All | Пользователи имеют право переопределять в файлах .htaccess глобальные параметры доступа. Из соображений безопасности лучше не использовать этот режим |
| Options | Разрешает использовать директиву Options |
| Limit | Разрешает использовать директиву Limit |
| AuthConfig | Разрешает использование директив AuthName, AuthType, AuthUserFile и AuthGroupFile |
| FileInfo | Разрешает использовать в файлах .htaccess директивы AddType и AddEncoding |

С помощью директивы **Options** можно определить функции сервера, которые будут доступны для использования в определяемом каталоге. Данную директиву можно использовать как в файле `httpd.conf`, так и в файлах `.htaccess`. Допустимые опции для директивы **Options** представлены в табл. 4.2.

http://all-ebooks.com

**Таблица 4.2.** *Допустимые опции директивы Options*

| Опция | Описание |
|---|---|
| None | Не разрешается использование каких-либо функций |
| All | Разрешаются все функции |
| FollowSymLinks | Разрешается использовать символические ссылки. С точки зрения безопасности не рекомендуется использовать этот режим |
| SymLinksIfOwnerMatch | Разрешается использование символических ссылок, если ссылка указывает на объект, который принадлежит тому же пользователю, что и ссылка |
| ExecCGI | Разрешается выполнение CGI-сценариев |
| Indexes | Если эта опция выключена, сервер не будет передавать содержимое каталога при отсутствии файла index.html |
| Includes | Разрешено использование серверных включений. Рекомендую отключить эту опцию, поскольку это сильно нагружает сервер |
| IncludesNoExec | Разрешает использование серверных включений, но запрещает запуск из них внешних программ |

Использование директивы **Limit** имеет следующий вид:

```
Limit метод
```

Директива **Limit** ограничивает доступ к файлам в определенном каталоге. Параметр метод определяет метод передачи, например, GET или POST. Директиву **Limit** можно использовать внутри блоков **Directory**, **Location** или в файле `.htacccess`.

### 4.4.2. Блок директив Limit

В блоке **Limit** можно использовать такие директивы: **allow** (разрешить), **deny** (запретить), **order** (порядок), **require** (требуется). После директивы **allow** следует слово **from**, после которого можно указать IP-адрес, адрес сети, домен или просто имя компьютера. Слово **all** обозначает все компьютеры. Например, вам требуется запретить доступ всем компьютерам, кроме компьютеров, которые входят в домен ru (листинг 4.4).

**Листинг 4.4. Директивы allow, deny**

```
order deny, allow
deny from all
allow from ru
```



http://all-ebooks.com

Следующий пример показывает, как разрешить доступ компьютерам только из вашей сети (листинг 4.5). Допустим, что ваша сеть имеет адрес 192.168.1.0.

**Листинг 4.5. Разрешение доступа только из подсети 192.168.1.0**

```
order deny, allow
deny from all
allow from 192.168.1.
```

Директива order определяет порядок выполнения директив **allow** и **deny**. Кроме значений «allow,deny» и «deny,allow», директива **order** может содержать значение **mutual-failure**. В этом случае доступ будет отказан всем компьютерам, которые явно не указаны в списке **allow**.

Директиву require можно использовать для защиты каталога паролем. После названия директивы должен следовать список элементов: имена пользователей и групп, которые указаны в директивах **AuthUserFile** и **AuthGroupFile**. Можно использовать параметр **valid-user**, который требует, чтобы сервер предоставлял доступ любому пользователю, имя которого имеется в директиве **AuthUserFile**, если тот введет правильный пароль. Пример использования директив **Limit**, **require**, **AuthUserFile** приведен в листинге 4.6.

**Листинг 4.6. Защита документов паролем**

```
<Directory *>
AuthUserFile /var/secure/.htpasswd
AuthName Security
AuthType Basic
<Limit GET>
order deny,allow
deny form all
allow from mydomain.ru
require valid-user
</Limit>
</Directory>
```

В листинге 4.6 для аутентификации используется файл паролей `.htpasswd`, который можно создать с помощью программы **htpasswd**. Директива **Limit** разрешает доступ к любому каталогу сервера только пользователям домена `mydomain.ru`

Кроме параметра **valid-user**, допускается использование параметра users или groups. Данные параметры разрешают доступ только определенным

пользователям или группам пользователей. Пример использования параметра users приведен в листинге 4.7.

**Листинг 4.7. Применение параметра users**

```
<Directory /users>
AuthType Basic
AuthUserFile /var/users/.htpasswd
AuthName UsersDir
<Limit GET POST>
 require users denis lena max
</Limit>
</Directory>
```

### 4.4.3. Блок директив Location

С помощью блока директив **Location** можно задать определенный URL-адрес, который предназначен для обозначения каталогов, файлов или групп файлов. Обозначить группу файлов можно с помощью шаблонов, например, шаблон `*.html` определяет все файлы, имена которых заканчиваются на `.html`. В URL-адрес не включаются протокол и имя сервера. Пример описания блока Location представлен в листинге 4.8.

**Листинг 4.8. Блок Location**

```
<Location URL>
    директивы управления доступом
</Location>
```

## 4.5. КАТАЛОГИ ПОЛЬЗОВАТЕЛЕЙ

Директива **UserDir** включает поддержку пользовательских каталогов. Эта директива определяет общее название подкаталога в домашних каталогах всех пользователей. По умолчанию используется каталог `public_html`. Данная возможность очень удобна при использовании ее в большой корпорации, где каждый сотрудник имеет собственную страничку. Раньше эта возможность часто использовалась на серверах, предоставляющих бесплатный хостинг. Может быть, помните адреса вида `http://www.chat.ru/~mypage`?

Сейчас же все чаще используется технология виртуальных серверов, которую мы рассмотрим в следующем пункте, но знать, что такое каталоги пользователей и как с ними работать, тоже не помешает. Тем более что домашние



http://all-ebooks.com

каталоги настраиваются намного быстрее и проще, чем виртуальный сервер – нужно всего лишь определить директиву **UserDir** и указать местоположение домашних каталогов.

Чтобы получить доступ к файлам, расположенным в этих каталогах, нужно указать после имени сервера регистрационное имя пользователя через косую черту и тильду. Например, пусть имя сервера `www.server.com`, имя пользователя – `denis`. Тогда URL будет выглядеть так: `http://www.server.com/~denis/`. При этом сервер самостоятельно определит, где именно расположен домашний каталог пользователя. Если домашний каталог пользователя `/home/den`, то сервер передаст клиенту файл `/home/den/public_html/index.html`.

## 4.6. ВИРТУАЛЬНЫЕ ВЕБ-СЕРВЕРЫ

Концепция виртуальных хостов позволяет серверу Apache поддерживать несколько веб-узлов. Получается, что один веб-сервер заменяет несколько серверов, и вместо одного узла пользователи видят отдельные веб-узлы. Это очень удобно, если нужно организовать персональные веб-узлы пользователей или собственные веб-узлы подразделений компании, например `develop.mycompany.com`.

Сервер Apache можно настроить несколькими способами:

- запускать один сервер, который будет прослушивать все обращения к виртуальным серверам;
- запускать отдельный процесс для каждого виртуального сервера.

В первом случае один сервер будет одновременно обслуживать все виртуальные. Если вас интересует такой вариант, нужно настраивать виртуальные серверы с помощью директивы **VirtualHost**. Настройка отдельных процессов для каждого сервера осуществляется с помощью директив **Listen** и **BindAddress**.

В этом разделе я буду рассматривать именно первый случай. Внутри блока директивы **VirtualHost** можно использовать любые директивы, кроме **ServerType**, **BindAddress**, **Listen**, **NameVirtualHost**, **ServerRoot**, **TypesConfig**, **PidFile**, **MinRequestPerChild**, **MaxSpareServers**, **MinSpareServers**, так как некоторые из них относятся к основному HTTP-серверу (например, **ServerType**), а некоторые – ко второму варианту настройки виртуальных серверов и здесь неприемлемы. Обязательно должны присутствовать директивы **ServerName**, **DocumentRoot**, **ServerAdmin** и **ErrorLog**.

В зависимости от версии и от настроек Apache виртуальные узлы могут прописываться либо в файле `httpd.conf`, либо в файле `vhosts.conf`.

Виртуальные серверы можно идентифицировать по имени или по IP-адресу.

### 4.6.1. Виртуальные серверы с идентификацией по имени

Идентификация по имени имеет существенное преимущество перед идентификацией по IP-адресу: вы не ограничены количеством адресов, имеющимся в вашем распоряжении. Вы можете использовать любое количество виртуальных серверов, и при этом вам не потребуются дополнительные адреса. Такое возможно благодаря использованию протокола `HTTP/1.1`, поддерживаемого всеми современными браузерами.

Поддержка виртуальных хостов обеспечивается директивами **VirtualHost** и **NameVirtualHost**. Если ваша система имеет только один IP-адрес, его нужно указать в директиве **NameVirtualHost**. Все виртуальные серверы будут работать по этому IP-адресу, для чего его нужно указать в каждой директиве **VirtualHost**.

Каждый блок директив **VirtualHost** описывает параметры отдельного виртуального сервера (листинг 4.9). Серверы различаются своими доменными именами, которые обязательно нужно указывать как параметры директивы **ServerName**. Это необходимо для того, чтобы избежать поиска виртуального сервера службой DNS – вы же не хотите, чтобы при неудачном поиске виртуальный сервер был заблокирован?

Допустим, у вас имеется основной сервер `www.dhsilabs.com` и вы хотите создать еще один виртуальный: `lib.dhsilabs.com`. Пример приведен в листинге 4.9.

**Листинг 4.9. Два виртуальных сервера**

```
ServerName den.dhsilabs.com

<NameVirtualHost 192.168.1.1>

<VirtualHost 192.168.1.1>
    ServerName www.dhsilabs.com
    ServerAdmin webmaster@den.dhsilabs.com
    DocumentRoot /var/httpd/www/html
    ErrorLog /var/https/www/logs/error.log,
    TransferLog logs/access.log
</VirtualHost>
```



http://all-ebooks.com

```
<VirtualHost 192.168.1.1>
    ServerName lib.dhsilabs.com
    ServerAdmin webmaster@den.dhsilabs.com
    DocumentRoot /var/httpd/lib/html
    ErrorLog /var/https/lib/logs/error.log
    TransferLog logs/access.log
</VirtualHost>
```

Если ваша система имеет только один IP-адрес, доступ к основному серверу станет невозможным, то есть вы не сможете использовать его напрямую. Можно основной сервер заявить в качестве виртуального, что я и сделал в листинге 4.9: основной сервер www является виртуальным. При наличии двух IP-адресов можно один присвоить основному серверу, а другой виртуальному.

Сервер Apache позволяет использовать несколько доменных имен для доступа к одному серверу, например:

```
ServerAlias www.dhsilabs.com www2.dhsilabs.com
```

При этом запросы, посланные по IP-адресам, которые присвоены вашим виртуальным хостам, должны соответствовать одному из указанных доменных имен. Чтобы зафиксировать запросы, не соответствующие ни одному их этих имен, нужно с помощью опции **default:*** создать виртуальный хост, который будет обслуживать такие запросы:

```
<VitrualHost _default_:*>
```

Обратите внимание на то, что в рассмотренном примере адреса `www.dhsilabs.com` и `lib.dhsilabs.com` должны быть прописаны в DNS.

### 4.6.2. Виртуальные серверы с идентификацией по IP-адресу

В директиве **VirtualHost** в качестве адресов можно использовать доменные имена, но лучше указывать IP-адрес, причем действительный, а не виртуальный. В этом случае вы не будете зависеть от DNS при разрешении имени. Также потребуется один IP-адрес для вашего основного сервера. Если же распределить все адреса между виртуальными серверами, то нельзя будет получить доступ к основному серверу.

**Листинг 4.10. Идентификация по IP-адресу**

```
<VirtualHost 192.168.1.2>
    ServerName www.dhsilabs.com
    ServerAdmin webmaster@den.dhsilabs.com
```

```
    DocumentRoot /var/httpd/www/html
    ErrorLog /var/https/www/logs/error.log
</VirtualHost>

<VirtualHost lib.dhsilabs.com>
    ServerName lib.dhsilabs.com
    ServerAdmin webmaster@den.dhsilabs.com
    DocumentRoot /var/httpd/lib/html
    ErrorLog /var/https/lib/logs/error.log
</VirtualHost>
```

В приведенном примере (листинг 4.10) сконфигурированы два виртуальных сервера с идентификацией по IP-адресу. Один из них использует сам IP-адрес, а другой – доменное имя, соответствующее IP-адресу.

При конфигурировании виртуальных серверов можно использовать опцию **ExecCGI**, которая разрешает выполнение CGI-скриптов на виртуальном сервере. Ниже приведен пример для почтового веб-интерфейса (листинг 4.11)

**Листинг 4.11. Подключение почтового веб-интерфейса**

```
# Файл httpd.conf

<Directory /home/httpd/mail>
    order deny,allow
    deny from all
    allow from localhost
    allow from 192.168
    allow from 123.123.123.123
    Options ExecCGI
</Directory>


# Файл vhosts.conf

<VirtualHost 123.123.123.123>
    ServerAdmin webmaster@den.dhsilabs.com
    DocumentRoot /home/httpd/mail
    ServerPath /mail
    ServerName wwwmail.dhsilabs.com
    ErrorLog logs/error_log
    TransferLog logs/access_log
    ErrorDocument 403 http://www.dhsilabs.com/messages/error403.html
</VirtualHost>

# Error 403 - попытка доступа извне, то есть почтовый интерфейс
будет
# доступен только из локальной сети
```



http://all-ebooks.com

## 4.7. КОНФИГУРИРОВАНИЕ APACHE С ПОМОЩЬЮ SYSTEM-CONFIG-HTTPD

Графические конфигураторы существенно облегчают процесс настройки любого сервера, будь то Apache или ProFTPD. Но за все нужно платить. В данном случае за комфорт приходится расплачиваться гибкостью в настройке. С помощью конфигуратора вы никогда не настроите сервер так точно, как редактируя его конфигурационный файл вручную. Обычно конфигураторы выполняют базовую настройку, но не более.

В разных дистрибутивах есть разные конфигураторы для Apache. Эта утилита необязательна, поэтому в вашем дистрибутиве её может вообще не оказаться, и придется искать её в Интернете. Например, для Fedora Core разработан конфигуратор **system-httpd-conf**, но в моем дистрибутиве его не было, пришлось загрузить через `http://rpmfind.net`.

Конфигуратор **system-config-httpd** (рис. 4.1) обеспечивает только базовую настройку сервера Apache, но тем не менее он «умеет» настраивать виртуальные узлы. Так что хоть какая-то польза от него вам будет.

Приводить полный листинг конфигурационного файла Apache я не вижу смысла – он довольно большой и имеется у вас с точно такими же комментариями, как и у меня, разве что на английском. Надеюсь, что, прочитав эту главу, вы уже и без комментариев знаете, для чего служит та или иная директива.

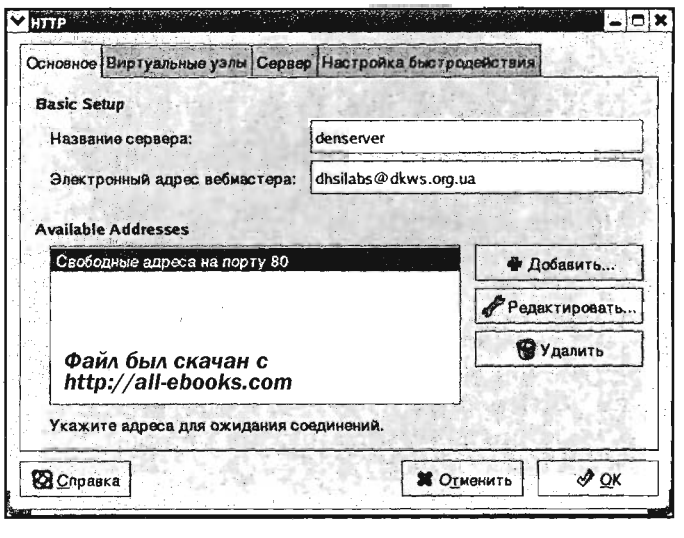



*Рис. 4.1. Конфигуратор system-config-httpd*

http://all-ebooks.com

# ЧАСТЬ II.

# ОСНОВНЫЕ ПОНЯТИЯ, МЕХАНИЗМЫ И ФУНКЦИИ PHP

http://all-ebooks.com

# ГЛАВА 5.

# ОСНОВЫ ЯЗЫКА PHP

http://all-ebooks.com

## 5.1. СТРУКТУРА ПРОГРАММ В PHP. ПЕРВАЯ ПРОГРАММА

### 5.1.1. Стандартный вид PHP-сценария

#### Общие понятия и описание PHP

Язык программирования PHP, специально предназначенный для работы в Интернете, сочетает достоинства языков C и Perl. Синтаксис языка очень сильно напоминает C — если вы программируете на C, то очень скоро это заметите.

В официальной документации язык PHP подается как server-side HTML-embedded scripting language, то есть как встраиваемый в HTML скриптовый язык с обработкой на сервере. Это позволяет сразу же отметить следующее:

- Обработка PHP-кода производится на стороне сервера еще до того, как Web-страница будет передана браузеру. Это в первую очередь отличает язык PHP от языка JavaScript.
- PHP-код может быть непосредственно встроен в HTML-код страницы. Это выгодно отличает PHP от языка Perl.



http://all-ebooks.com

Мы знаем, что языки программирования бывают двух видов: интерпретируемые и компилируемые. Чем же является PHP? Чтобы выяснить это, нам нужно разобраться в терминологии.

Программа, переводящая код, написанный на одном языке программирования, на другой, называется транслятором. Например, транслятором будет программа p2c, которая переводит Pascal-код в C-код.

Компилятор — это тоже транслятор. Он переводит код, написанный на языке высокого уровня, в машинный код. Язык высокого уровня понятен человеку и близок к человеческому (Pascal, C, PHP), а машинный код (0 и 1) понятен компьютеру — с двоичными данными компьютеру проще работать. В результате работы компилятора, как правило, создается двоичный исполняемый файл (в Windows/DOS — с расширением .exe или .com), который можно выполнять без помощи компилятора.

Интерпретатор — это совершенно другое средство программирования, не имеющее ничего общего с компилятором. Интерпретатор ничего не переводит, а только исполняет код. Он анализирует код программы и исполняет каждую ее строку. Например, предположим, что интерпретатор нашел строку «A = A + 7». Он анализирует и проверяет ее корректность, а затем ищет переменную A. Если переменная была объявлена до строки «A = A + 7», то интерпретатор увеличивает ее значение на 7 — он выполняет код. Была бы вместо строки «A = A + 7» строка «print "Hello"», наш интерпретатор вывел бы на экран строку «Hello». Выполнив код, интерпретатор не создает исполняемый файл, поэтому для следующего запуска вашего кода опять нужен интерпретатор.

По производительности интерпретируемые языки значительно уступают компилируемым, поскольку машинный код выполняется намного быстрее. Зато интерпретаторы позволяют полностью контролировать программу во время ее исполнения.

PHP не является ни компилятором, ни интерпретатором, а представляет собою нечто среднее. Рассмотрим, как PHP обрабатывает сценарии. На вход PHP подается сценарий. Он переводит (транслирует) его в специальный байт-код (внутреннее представление). Затем PHP выполняет байт-код (а не код самой программы!), не создавая при этом исполняемого файла. Байт-код значительно компактнее обыкновенного кода программы, поэтому он интерпретируется (выполняется) быстрее. Как видите, PHP больше является интерпретатором, нежели компилятором. Для чего нужна была такая двойная работа — сначала переводить во внутреннее представление, а потом — выполнять? Неужели не проще сразу выполнить каждую строку программы?

http://all-ebooks.com

Вдумайтесь в это выражение «каждую строку программы». Представим, что у нас есть цикл:

```
for  (I=0;I<10;I++)
{
оператор_1;
оператор_100;
}
```

Наш цикл будет «крутиться» не бесконечно, а всего лишь десять раз. За каждый из этих десяти проходов интерпретатор должен проанализировать и выполнить 100 строк кода. Итого ему нужно проанализировать и выполнить (я не зря дважды упоминаю эту фразу) 10x100 = 1000 строк кода. Если перевести один раз весь цикл в байт-код, то анализировать ему придется в 10 раз меньше! А это значит, что ваши сценарии будут выполняться в 10 раз быстрее.

Выходит, что PHP является интерпретирующим транслятором. Стоит отметить, что PHP версии 3 был «чистым» интерпретатором и только с появлением четвертой версии сценарии стали выполняться значительно быстрее. Пятая версия, как и четвертая, является интерпретирующим транслятором.

Однако, что бы ни делали разработчики PHP, откомпилированные двоичные исполняемые файлы работают значительно быстрее — в десятки, а может и в сотни раз. Но производительность PHP вполне достаточна для создания интернет-приложений. Сейчас объясню почему.

Пусть вышеупомянутый цикл, переписанный на языке С, выполняется за 0.1 секунды. На языке PHP он будет выполняться в 10 раз дольше — за 1 секунду. Однако передача пользователю данных— результатов работы цикла — будет длиться 3 секунды. Получаем, соответственно, в сумме 3.1 и 4 секунды. Согласитесь, разница между 3.1 и 4 секунды незначительна, как пользователь Интернета вы даже ее не почувствуете. Даже если у вас быстрый выделенный канал, разница все равно не будет заметна — может сказаться нагрузка на сервер или другие факторы, даже не зависящие от PHP.

Использование интерпретатора (а значит и PHP) имеет свои неоспоримые преимущества:

- Вам не нужно заботиться об освобождении выделенной памяти, не нужно закрывать файл по окончании работы с ним — за вас все сделает интерпретатор, поскольку ваша программа выполняется под его бдительным контролем.



http://all-ebooks.com

- Вам не нужно думать о типах переменных, вам также не нужно объявлять переменную до её первого использования.
- Отладка программы и обнаружение ошибок существенно упрощаются: интерпретатор непременно сообщит вам, если что-то пошло не так.
- Вы можете написать программу, которая будет писать другую программу (да, это так), а потом включит в себя код только что написанной программы и выполнит его. Об этой возможности мы поговорим отдельно, когда наступит время.

### Пишем первую программу (скрипт) на PHP

Согласно традиции, знакомство с любым языком программирования начинается с программы Hello, World! Эта программа ничего замечательного не делает, она только выводит на экран надпись «Hello World!». Мы сейчас немного отступим от традиции и напишем программу, которая будет здороваться не со всем миром (все равно этого никто не услышит), а только с нами. Текст программы (сценария, или скрипта) приведен в листинге 5.1.

**Листинг 5.1. Первая программа на PHP**

```
<?
echo "<html><body><h1>";
echo "Hello, $my_name";
echo "</h1></body></html>";
?>
```

Прежде чем запустить программу, ее нужно правильно «установить» на сервере. Для этого сохраните сценарий под именем `first.php` и скопируйте его в каталог `DocumentRoot` вашего сервера. Если вы работаете под Linux, таким каталогом будет `/var/www/html` (в старых версиях — `/home/httpd/html`) Если вы работаете под Windows и следовали моим рекомендациям в главе 3, то каталогом `DocumentRoot` будет `c:\web5\www`.

> **Примечание.**
>
> Для корректной работы этого сценария директива register_globals в вашем файле конфигурации php.ini должна быть включена. Если вы воспользовались интерпретатором PHP, загруженным с сайта dkws.org.ua, данная директива будет включена по умолчанию. Если же вы скачали PHP с www.php.net или другого сайта, проверьте, включена ли эта директива.

В отличие от CGI-программ, сценарии PHP являются для сервера обыкновенными документами и их не нужно помещать в каталог `cgi-bin`. Также не нужно делать их исполняемыми (в случае, если вы работаете под Linux).

Выполнив вышеуказанные рекомендации по установке, введите в адресную строку браузера:

```
http://localhost/first.php?my_name=Denis
```

В результате в окне браузера вы увидите Hello, Denis (рис. 5.1).

Если теперь страницу, отображенную в браузере, просмотреть в виде HTML-кода, то можно увидеть следующее:

```
<html><body><h1>Hello, Denis</h1></body></html>
```

Как видите, PHP-код выполнился, подставил нужное значение переменной `$my_name` и выдал готовую HTML-страницу.

Сейчас мы запускали PHP-сценарий через CGI-интерфейс, передав переменной `$my_name` значение Denis. Однако любую PHP-программу можно запустить и так:

```
php file.php
```

При этом CGI-переменные будут недоступны — ведь вы запустили программу из командной строки. Демонстрационные PHP-программы, не требующие CGI, в этой книге я буду запускать именно так.

**Примечание.**

Вводить текст программы лучше в текстовом редакторе, поддерживающем подсветку синтаксиса PHP. Если вы работаете под Linux, могу порекомендовать три редактора: K Write (команда kwrite), Kate (kate) и Quanta (quanta). Под Windows также не составляет труда найти такой редактор. Если вы не найдете редактора с подсветкой синтаксиса PHP, можно использовать редактор с подсветкой синтаксиса языка C.

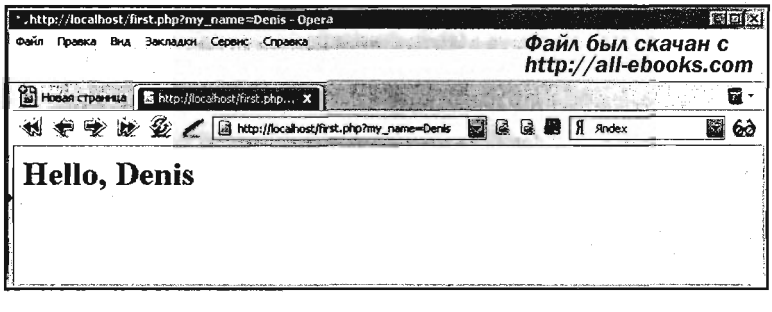



***Рис. 5.1. Первая PHP-программа***



http://all-ebooks.com

Теперь рассмотрим саму программу. Код PHP заключается в специальные теги <? и ?>. После тега начала сценария (<?) следует первый оператор — **echo**. Оператор **echo** осуществляет вывод информации. Его с полной уверенностью можно назвать самым главным оператором — ведь без него был бы невозможен вывод информации в браузер и пользователь не увидел бы результата работы сценария.

Выводимая оператором **echo** строка заключается в кавычки. Первый оператор выводит теги HTML, BODY и H1. В принципе, можно было бы и не выводить первые два тега, но нужно привыкнуть к дисциплине с самого начала.

Второй оператор echo «Hello, Smy_name»; выводит слово Hello и значение переменной `Smy_name`. Перед именами всех переменных ставится знак доллара. Это правило кажется необычным только поначалу, довольно скоро вы привыкнете к нему.

Присваивание переменной какого-либо значения осуществляется оператором присваивания, который обозначается значком «=» (равно). Примеры использования переменных:

```
$i=0;
$d=7;
$i=5 + $d;    // $i = 12
```

Еще один пример:

```
$word1 = "Hello";
$word2 = "world";

echo "$word1, $word2";
echo "$word1, word2";
```

Первый оператор выведет в браузер строку «Hello, world», а второй — «Hello, word2». В этом примере мы определили две переменные — `$word1` и `$word2`. Так как мы забыли дописать знак доллара во втором операторе, PHP воспринял «word2» как обыкновенную строку, а не как переменную.

Обратите внимание, что в нашем первом примере (`first.php`) мы не объявили переменную `$my_name`. Ее объявил автоматически сам PHP — ведь это параметр, переданный сценарию браузером. В этом заключается еще одно преимущество PHP перед другими языками программирования.

Вы когда-нибудь программировали на Pascal или C? Обращение к параметру, переданному программе, осуществляется по его номеру. Например, прочитать значение первого параметра можно, вызвав Pascal-функцию

`ParamStr(1)`. В C параметры хранятся в специальном массиве, обращение к которому, как обычно, выполняется по индексу элемента. Ощущаете разницу? То есть, если мы программируем на Pascal или на C, нам обязательно нужно знать номер переданного параметра.

Например, программе, копирующей файл, требуется передать два параметра: имя исходного файла и имя файла назначения. Если мы перепутаем порядок файлов, может произойти непоправимое — исходный файл будет перезаписан файлом назначения. Для программы на PHP порядок следования параметров не важен:

`src=file1&dest=file2` или `dest=file2&src=file1`

(так передаются браузером параметры нашему сценарию).

Очень удобно! И это не говоря уже о том, что PHP сам выполняет разбор переданной строки, объявляет и инициализирует переменные.

Третий оператор нашей программы в особых комментариях не нуждается, как и тег окончания PHP-программы (?>).

Модифицируем немного наш текст программы `first.php`, чтобы он выглядел так:

```
<?
echo   "<html><body><h1>";
echo   "Hello, $my_name";
echo    "<br> $var";
echo    "</h1></body></html>";
?>
```

Если мы не передадим программе параметр `$var`, то есть, как обычно, запустим программу только с параметром `$my_name`, то в окне браузера мы увидим сообщение о том, что переменная `$var` не определена. Для того чтобы избавиться от этой ошибки, непосредственно перед оператором echo "<br> $var"; добавьте строку:

```
$var=" ";
```

Переменная будет объявлена и инициализирована. В PHP, как и в C++, вы можете объявлять переменные в любом месте сценария, но до места первого использования переменной.

Любой другой интерпретатор, например Perl, в подобной ситуации сгенерировал бы сообщение 500: «Ошибка сценария». Интерпретатор PHP решает проблему просто и элегантно, избавляя тем самым программиста от просмотра многотомных журналов веб-сервера.



http://all-ebooks.com

### 5.1.2. PHP-сценарии в теле HTML-документов

Мы только что рассмотрели структуру PHP-программы, и можно было бы перейти к рассмотрению переменных. Однако PHP-сценарии можно записывать и по-другому.

Сценарий из листинга 5.1 можно оформить так:

```
<html><body><h1>
<? echo "Hello, $my_name"; ?>
</h1></body></html>
```

Удивлены? Обыкновенный HTML-код корректно обрабатывается интерпретатором, поэтому у вас не будет ошибок при выполнении этого сценария.

Давайте разберемся, как правильно включать HTML-код в состав вашего сценария. Если интерпретатор PHP встречает скобки <? ?>, то он считает, что они содержат PHP-код, и пытается выполнить его. Все, что находится за пределами скобок, считается простым текстом и передается браузеру безо всяких изменений. Вот ещё один пример:

```
<html><body>
<? if ($REMOTE_ADDR=="127.0.0.1") { ?>
<h1>Добро пожаловать домой!</h1>
<? } else { ?>
<h1>Привет, гость!</h1>
<? } ?>
</html></body>
```

Если IP-адрес удаленного компьютера равен 127.0.0.1, мы выводим HTML-код «<h1>Добро пожаловать домой!</h1>» без использования оператора echo. В противном случае мы выводим другое сообщение — «<h1>Привет, гость!</h1>». Таким образом, теги <hl> и </hl> воспринимаются и выводятся как обычный текст.

> **Примечание.**
>
> В приведенном примере был использован условный оператор if, который позволяет выполнять определенные условия в зависимости от истинности или ложности условия. После условия указывается действие, которое следует выполнять в случае его истинности, а после слова else — в случае ложности. Подробнее условный оператор будет рассмотрен далее.

При включении HTML-кода в ваши сценарии будьте внимательны, потому что легко запутаться. Вообще я рекомендую выводить HTML-код только с помощью echo, так будет проще.

### 5.1.3. Комментарии в PHP-сценариях

Немаловажным элементом любой программы являются комментарии. Настоятельно рекомендую не пренебрегать ими. Когда через месяц вам нужно будет вернуться к сценарию, который вы пишете сейчас, комментарии помогут вам значительно быстрее разобраться в вашей же программе.

Комментарии в PHP могут быть трех типов:

```
#   Это комментарий
//  Это тоже комментарий
/ *
A   это
длинный многострочный
комментарий
* /
```

Я предпочитаю использовать комментарии второго и третьего типа. Хотя какой из них использовать — дело вкуса.

## 5.2. ПЕРЕМЕННЫЕ И ТИПЫ ДАННЫХ

Практически во всех языках программирования есть такое понятие, как переменная. Не является исключением и язык PHP.

Переменная — это область оперативной памяти, доступ к которой осуществляется по имени. Все данные, с которыми работает программа, хранятся в виде переменных.

Как я уже сказал, в программе на языке PHP перед именем переменной обязательно должен стоять знак доллара. По этому символу интерпретатор выделяет переменные из текста программы.

В официальной документации PHP сказано, что имена переменных могут состоять не только из латинских букв, но из любых символов, код которых больше 127. Это означает, что имена переменных могут содержать русские буквы. Однако я настоятельно советую вам не использовать русскоязычные имена — ведь в разных кодировках буквы кириллицы имеют разные коды.

Определим правила задания переменных в PHP:

- Имя переменной должно начинаться со знака доллара $.
- Имя переменной не должно содержать никаких других символов, кроме символов латинского алфавита, цифр и знака подчеркивания.



http://all-ebooks.com

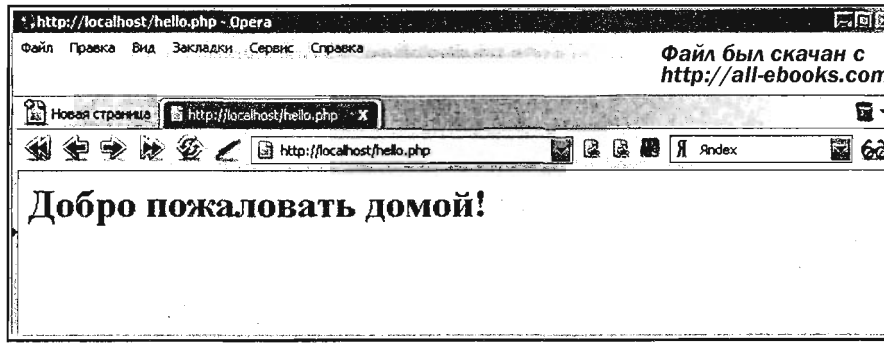



***Рис. 5.2. Добро пожаловать домой!***

Если вы будете использовать это правило, вы существенно облегчите себе жизнь.

- Имена переменных в PHP, как и в C, чувствительны к регистру символов, то есть `$a` и `$A` – это совершенно разные переменные.

- Объявлять переменную можно в любом месте программы, но до места первого ее использования.

При объявлении переменных можно не скупиться — объявляйте столько, сколько вам нужно. Привычные методы оптимизации, которые применялись 20 лет назад, во времена первых компьютеров (например, чрезмерная экономия памяти — чем меньше переменных использует программа, тем лучше), не сработают: ваша программа не будет работать быстрее, если в ущерб собственному комфорту вы будете использовать одну переменную вместо 5. К тому же можно наделать ошибок, которые очень трудно будет найти.

### Типы данных

Переменные в PHP могут содержать данные любого типа. Исключение составляют константы, которые могут содержать только число или строку.

Указателей, знакомых вам по языку C (то есть переменных, содержащих не значение, а адрес в памяти, по которому находится это значение), в языке PHP не существует. При присваивании переменная копируется «один в один», какую бы структуру она не имела.

Предположим, что у нас есть две переменные: массив и строка. Если первой переменной мы присвоим значение второй, то наш массив будет навсегда

потерян, а переменная, которая содержала массив, будет содержать строку — это и означает, что копируется не только значение переменной, как в других языках, но и структура.
Правда, начиная с 4-й версии, в PHP появились жесткие и мягкие ссылки, но о них мы поговорим позже.

Рассмотрим следующее объявление переменных:

```
$a = 10;
$A = "Hello";
```

Как вы видите, при объявлении переменных мы не указали их тип. Выбор типа осуществляется самим интерпретатором. Однако иногда PHP может ошибиться и сопоставить переменной неправильный тип. В таких случаях нужно задавать тип явно. О том, как это сделать, мы поговорим чуть позже, а пока рассмотрим основные типы переменных (табл. 5.1).

**Таблица 5.1.** *Типы переменных в PHP*

| Тип переменной | Описание |
|---|---|
| Integer | Целое число. Диапазон значений: от -2 147 483 648 до -2 147 483 647 (32 бита). Данный тип соответствует типу long int в языке C (или longint в Pascal) |
| double (или float) | Вещественное число очень большой точности |
| String | Строка. Это очень важный тип данных, поскольку в большинстве случаев сценарии занимаются обработкой строк. Поэтому строкам посвящена целая глава |
| Array | Массив. О массивах мы также поговорим в отдельной главе |
| Object | Язык PHP поддерживает принципы объектно-ориентированного программирования, хотя и самые простые. Программист может создавать свои объекты и использовать их в программе. Доступ к отдельным элементам объекта осуществляется с помощью оператора -> |
| Bool | Логический тип данных. Переменные этого типа могут принимать только одно из двух значений: true (истина) или false (ложь) |

### Функции определения и задания типа переменных

Язык PHP предоставляет много средств для определения типа переменных. Вы можете использовать целых семь функций для определения типа:

- `is_int($x)` или `is_integer($x)` — возвращает `true`, если переданная переменная — целое число.
- `is_double($x)` или `is_float($x)` — возвращает `true`, если переданная переменная — вещественное число.



http://all-ebooks.com

- `is_string($x)` — возвращает `true`, если переданная переменная — строка.
- `is_array($x)` — возвращает `true`, если переданная переменная — массив.
- `is_object($x)` — возвращает `true`, если переменная является объектом.
- `is_bool($x)` — возвращает `true`, если переменная объявлена, как логическая.
- `gettype($x)` — возвращает строку, описывающую тип переменной: `integer`, `double`, `string`, `object`, `array`, `bool` или `unknown type`, если невозможно определить тип. Тип переменной определить невозможно, когда он не встроен в PHP, а добавляется с помощью модулей, расширяющих возможности языка.

Если PHP неправильно определил тип переменной, можно указать его явно. Для этого используется функция `settype($x, $type)`, где `$type` — это одна из строк, возвращаемых функцией `gettype()`. Например:

```
settype($x,"double");
```

Функция возвращает значение `false`, если невозможно привести тип переменной `$x` к указанному типу `$type`.

### Логические переменные и их особенности в PHP

Наверное, вы заметили, что большинство функций для определения типа (все, кроме `gettype()`) возвращают логическое значение `true` или `false`. Сейчас самое время поговорить о логических переменных и связанных с ними «подводных камнях».

В PHP логически эквивалентны любое не равное нулю число, любая непустая строка, значение `true`: все они являются истиной. Значение `false`, пустая строка, нуль — это ложь.

Поэкспериментируем немного с логическими переменными. Мы будем использовать условный оператор `if`, о котором я еще не говорил, но, если вы знакомы с любым алгоритмическим языком (Си, Паскаль и т.п), без труда поймете смысл этого оператора. Тем более что текст примеров сопровождается подробными комментариями.

Итак, создайте сценарий `logical.php`:

```
<?
echo false;
echo true;
?>
```


http://all-ebooks.com

Первый оператор выводит пустую строку, а второй выведет «1».

В использовании логических переменных в PHP имеется еще одна особенность. Если в операторах сравнения (= =, !=, <,>) один тип является логическим, то и второй также воспринимается, как логический. Модифицируем наш сценарий так:

```
<?
$x=10;
if ($x == 1) echo "Переменная равна 1\n";
if ($x == true) echo "Переменная равна True\n";
?>
```

В первой строке переменной `$x` было присвоено значение 10. Затем `$x` сравнивается с 1 и если `$x` равно 1, то выводится строка «Переменная равна 1». Затем значение переменной `$x` сравнивается со значением `true`. И если `$x` равно `true`, то выводится строка «Переменная равна True». Поскольку ненулевые числа эквивалентны истине, должна быть выведена только вторая строка. Так оно и происходит.
Рассмотрим еще один пример:

```
<?
$x = 100;
$y = true;

echo "x= $x\n";
echo "y= $y\n";

if ($x==$y) echo "X=Y";
?>
```

Вот здесь уж совсем смешная ситуация: сначала программа сообщает, что X = 100, Y = 1, а затем, что X = Y. А происходит это потому, что переменная `$y` имеет логический тип, следовательно, оператор сравнения трактует переменную `$x` тоже как логическую. Поскольку `$x` не равна 0, `false` или пустой строке, считается, что она равна `true`. Вот и выходит, что `$x` = `$y`.

## 5.3. КОНСТАНТЫ

Константы содержат постоянные значения. В отличие от переменной, вы не можете изменить значение, присвоенное константе при ее объявлении. Константы удобно использовать для хранения значений, которые не должны изменяться во время работы программы (например, имя разработчика программы).

Для определения константы служит функция `define`. Эта функция имеет следующий формат:



http://all-ebooks.com

```
define ($name, $value, $case_sen)
```

где `$name` — имя константы, `$value` — значение константы, а `$case_sen` — необязательный параметр логического типа, указывающий, следует ли учитывать регистр символов (`true`) или нет (`false`).

Пример:

```
define("pi",3.14,true);
echo pi;
```

Обратите внимание, что после объявления константы ее можно использовать без предваряющего символа доллара.

При объявлении констант нужно помнить два важных момента:

- Имя константы должно быть заключено в кавычки, а не в апострофы.
- Нельзя объявлять две константы с одним и тем же именем.

Для проверки существования константы можно использовать функцию `defined()`. Эта функция возвращает `true`, если константа объявлена, и `false`, если не объявлена.

Пример:

```
define("pi",3.14,true);
if (defined("pi")==true) echo "Константа объявлена";
```

При программировании на PHP вы можете использовать следующие стандартные константы (см. табл. 5.2).

**Таблица 5.2.** *Стандартные константы*

| Имя | Значение |
|---|---|
| __FILE__ (два знака подчеркивания до FILE и два после) | Содержит имя программы, которая выполняется в данный момент |
| __LINE__ | Номер строки, которую сейчас обрабатывает интерпретатор |
| PHP_OS | Имя и версия операционной системы, под которой запущен PHP |
| PHP_VERSION | Версия PHP |
| TRUE, true, FALSE, false | Уже знакомые нам логические константы |

## 5.4. ОПЕРАЦИИ С ПЕРЕМЕННЫМИ

Мы уже рассмотрели две важнейшие операции с переменными — определение и установку типа переменной. Поэтому нам осталось рассмотреть всего три операции:

- Присвоение значения.
- Проверка существования переменной.
- Уничтожение переменной.

Вы можете спросить, а как же арифметические и другие операции? Все остальные операции специфичны для конкретного типа переменной. Например, операция умножения совершенно бессмысленна для двух строк или числа и строки.

Тем не менее PHP является очень гибким языком и выполняет преобразования числа в строку (и наоборот) незаметно для нас. Когда я говорил, что операция умножения бессмысленна для двух строк, я был не совсем прав. Выполните этот сценарий, и вы убедитесь в этом:

```
$x = "10";     // строка
$y = "2";      // и это строка
echo $x*$y;
```

Сценарий вместо сообщения об ошибке выведет на экран число 20. Как видите, если строка допускает преобразование типа, то PHP работает со строками, как с обыкновенными числами.

### 5.4.1. Присвоение значений. Оператор присваивания

Оператор присваивания позволяет присвоить переменной некоторое значение. Нет такого языка программирования, в котором не было бы оператора присваивания. Поэтому я вас сильно не удивлю, если скажу, что в PHP он тоже есть:

```
$x    =    4;
$y    =    $x;
```

Вот, что вам нужно знать об этом операторе и правилах присваивания:

1. Переменной можно присвоить: константу; значение, возвращаемое функцией; значение другой переменной; значение выражения (например, `$x + $y * 2 - 7`); ссылку на другую переменную.



http://all-ebooks.com

2. В PHP нет указателей, поэтому если в переменную `$x` вы поместите файл размером 500 Кб, а потом присвоите ее значение переменной `$y`, то будет создана точная копия переменной `$x`, которая тоже будет занимать 500 Кб. Итого у вас в памяти будет почти один мегабайт информации. Это нужно учитывать при создании так называемых временных переменных. Желательно после окончания работы с такой переменной освободить память, то есть уничтожить переменную.
3. Интерпретатор сам выполняет преобразование типов, но иногда привести тип переменной к другому просто невозможно. Например, как можно преобразовать строку «1212го3te» в число?
4. Как я уже говорил, при присваивании создается точная копия переменной, копируется также и тип переменной. Это означает, что если вы массиву присвоите число, весь массив будет потерян.

Работа оператора присваивания состоит из двух последовательных этапов:

- Вычисление выражения, стоящего в правой части.
- Присвоение переменной, стоящей в левой части, вычисленного значения.

Таким образом, из-за того, что процесс присваивания состоит из двух раздельных этапов, в левой и правой части оператора присваивания может находиться одна и та же переменная. Например: `$x = $x+4`.

Кроме того, так же как и в языке C, в PHP существует возможность использования операции присваивания в следующем виде: переменная операция=значение.

Например:

```
$x +=10; // эквивалентно $x = $x + 10;
$x *=10; // эквивалентно $x = $x * 10;
```

## 5.4.2. Проверка существования переменной

Проверка существования переменной — это очень удобная возможность языка программирования. Вы можете спросить: «А зачем она нужна? Что, я не помню, объявлял я переменную или нет?» Сейчас поясню. Помните, я говорил, что переменные может объявлять сам PHP — при разборе параметров. Так вот, благодаря возможности проверки существования переменной вы можете проверить, передан ли сценарию определенный параметр, а только потом начинать с ним работать. Рассмотрим два сценария.

http://all-ebooks.com

Сценарий 1:

```
<?
$x    =     $my_name;
?>
```

Сценарий 2:

```
<?
if (isset($my_name))
{
  $x = $my_name;
}
?>
```

В первом случае мы сразу начинаем работать с переменной. Если по каким-либо причинам интерпретатор не передал сценарию параметра m_name, мы получим сообщение о том, что переменная не определена. Во втором сценарии мы с помощью функции isset ($my_name) проверяем существование переменной $my_name, а только потом начинаем с ней работать. В результате ваш сценарий не прекратит свою работу из-за банальной ошибки пользователя, если тот забудет ввести свое имя.

Конечно, если проверять все переданные параметры, наша программа станет намного длиннее. Иногда на проверке можно сэкономить. Предположим, нам нужно написать на PHP анкету для ввода имени и адреса электронной почты. Алгоритм нашего сценария будет такой:

- Если пользователь не передал параметры, значит, мы выводим форму анкеты.

- Если параметры переданы (то есть если пользователь нажал кнопку Submit на форме), значит, можно начать обработку параметров.

Выходит, что нам достаточно проверить существование только одной переменной, устанавливаемой, когда пользователь нажимает кнопку:

```
<html><head><title>Анкета</title></head><body>
<?
if(!isset($go))
  {
    // переменная go (кнопка) не установлена, выводим форму:
    echo "<form action=anketa.php method=post>";
    /* Выводим поля анкеты, а затем кнопку go */
    echo "<input type=submit name=go value=Send></form>";
}
else
{
  // параметры введены пользователем, начинаем обработку
}
?>
```



http://all-ebooks.com

Проверив существование переменной `$go`, мы начинаем обработку параметров. В выводимой форме я использовал метод POST. Для PHP-сценариев нет совершенно никакой разницы, каким методом им будут переданы параметры, поэтому из соображений безопасности лучше использовать метод POST (никто из-за спины не увидит, что ввел пользователь), да и выглядит он более эстетично (ничего лишнего в строке адреса не отображается).

### 5.4.3. Удаление переменных

Чтобы не занимать память, можно удалить ненужные нам переменные. Это делается с помощью функции `unset()`. Эта функция удаляет указанную в скобках переменную из памяти, и программа продолжает выполняться, как будто бы эта переменная вообще не была инициализирована.

```
<?
$a    =    // читаем_большой_файл;
обрабатываем_файл;
unset($a);    // освобождаем память
?>
```

Эта функция особенно полезна и даже необходима при обработке больших объемов данных, чтобы они не оставались в памяти компьютера и не замедляли его работу.

## 5.5. ВЫРАЖЕНИЯ И ОПЕРАЦИИ

### 5.5.1. Понятие «выражение» в PHP

Выражения являются теми «кирпичиками», из которых состоят PHP-программы. Практически все, что вы пишете в программе, является выражением. Под выражением понимается любая конструкция, у которой есть значение. Самое простое выражение — это константа, стоящая в правой части оператора присваивания:

```
$x    =    100;
```

Разберемся, что есть что. `$x` — это переменная, = — это оператор присваивания, а 100 — это и есть выражение. Его значение — число 100.

Выражением может служить и переменная, если ей сопоставлено определенное значение. Например:

```
$x    =    5;
$y    =    $x;
```

http://all-ebooks.com

В первом операторе выражением является константа 5, во втором — переменная $x, так как ранее ей было присвоено значение 5. Но это еще не все, $y = $x тоже является выражением! Ведь не вызывает сомнения то, что его значение равно 5. Это, в свою очередь, позволяет использовать следующие операторы:

```
$y = $x = 5;
```

или

```
$y = ($x = 5);
```

Таким образом, оператор присваивания можно использовать в середине выражения:

```
$y = 1;
$x = 100*($y = 2) * $y;
```

Очевидно, что переменной $x будет присвоено значение 100*2*2 = 400.

## 5.5.2. Операции в PHP

### Арифметические операции

Как и в любом другом языке, мы можем использовать арифметические операции:

- X + Y — сложение;
- X — Y — вычитание;
- X * Y — умножение;
- X / Y — деление;
- X % Y — остаток от деления X на Y.

Также возможно использование скобок. Приоритет одних арифметических операций над другими и изменение приоритетов при использовании скобок в арифметических выражениях соответствуют обычным математическим правилам.

К арифметическим операциям можно также отнести операции инкремента и декремента:

- операция инкремента **$x++** увеличивает значение на 1;
- операция декремента **$x--** уменьшает значение на 1.



http://all-ebooks.com

### Битовые операции

Эти операции предназначены для установки или снятия групп битов целочисленной переменной. Ведь любое число — это просто последовательность битов. Целые числа в PHP — 32-разрядные, поэтому для представления одного числа используются 32 бита:

```
0000  0000  0000  0000  0000  0000  0000  0000 - это ноль

0000  0000  0000  0000  0000  0000  0000  0001 - это 1
0000  0000  0000  0000  0000  0000  0000  0010 - это 2
0000  0000  0000  0000  0000  0000  0000  0011 - это 3
0000  0000  0000  0000  0000  0000  0000  0100 - это 4
0000  0000  0000  0000  0000  0000  0000  0101 - это 5
...
0000  0000  0000  0000  0000  0000  0000  1111 - это 15
...
```

Битовые операции:

- `x & y` – будут установлены те биты, которые установлены в `x` и `y` одновременно

  x = 1111 (15) y = 1100 (12) Результат = 1100 (12)

- `x | b` – будут установлены только те биты, которые были установлены или в x, или в y:

  x = 0100 (4) y = 0101 (5) Результат = 0101 (5)

- `~x` – инвертирование битов:

  x = 1001 Результат = 0110

- `x << y` – поразрядный сдвиг битов x влево на y разрядов:

  x = 0010 y = 0001 (1 разряд) Результат = 0100

- `x >> y` - поразрядный сдвиг битов x вправо на y разрядов:

  x = 0010 y = 0001 (1 разряд) Результат = 0001

### Логические выражения

Логические выражения — это выражения, результатом которых может быть или истина, или ложь. Примеры логических выражений:

http://all-ebooks.com

```
$x = true;              // $x = истина
$x = $y < 0;            // $x = true, если $y < 0
$x = $y == 0;           // $x = true, если $y = 0
$x = $y>=0&&$y<=10      // $x = true, если $y < =0 и $y<=10,
                        // то есть $y в диапазоне от 0 до 10
```

В логических выражениях можно использовать операторы сравнения и логические операции.

Операторы сравнения:

**==** — равно;

**!=** — не равно;

**<** — меньше;

**>** — больше;

**>=** — больше или равно;

**<=** — меньше или равно.

В PHP нельзя сравнивать массивы и объекты, разрешается сравнивать только скалярные переменные.

Логические операции:

- **!x** – возвращает `true`, если `x` – ложь и наоборот (отрицание, NOT);
- **x && y** – возвращает `true`, если `x` и `y` – истинны (умножение, И, AND);
- **x || y** – возвращает `true`, если истинно хотя бы одно из значений или все они истинны (сложение, ИЛИ, OR).

Несмотря на то что логическим значениям эквивалентны числа, операции инкремента (++) и декремента (--) с логическими переменными не работают, то есть не являются логическими операциями. Чтобы убедиться в этом, запустим следующий сценарий:

```
<?
$x = false; // эквивалентно 0
echo $x;
$x++; // может быть, инкремент 0 даст 1?
echo $x;
?>
```

Этот сценарий выведет две пустые строки. Это и означает, что операция инкремента для логических значений не работает. То же самое можно сказать и об операции декремента.



http://all-ebooks.com

### Приоритет операций

Сколько будет 2+2*2? Правильно, 6, поскольку операция умножения выполняется перед операцией сложения. Это значит, что приоритет операции умножения выше.

Операции с более высоким уровнем приоритета выполняются в первую очередь. Если вы в чем-то сомневаетесь, используйте скобки. Приоритет арифметических и логических операций представлен в табл. 5.3.

**Таблица 5.3.** *Приоритет операций в PHP*

| Приоритет | Оператор | Порядок выполнения |
|---|---|---|
| 13 | (постфикс)++ (постфикс )— | слева направо |
| 12 | ++(префикс) --(префикс) | справа налево |
| 11 | * / % | слева направо |
| 10 | + - | |
| 9 | << >> | |
| 8 | < <= > >= | |
| 7 | == != | |
| 6 | & | |
| 5 | ^ | |
| 4 | \| | |
| 3 | && | |
| 2 | \|\| | |
| 1 | = += .= *= /= %= >>= <<== &= ^= \|= | справа налево |

Файл был скачан с http://all-ebooks.com

## 5.5.3. Строки и строковые выражения

### Строки в апострофах и строки в кавычках

Рассмотрим небольшой пример:

```
$s = "Hello";
echo "$s";
echo '$s';
```

Оба оператора echo выведут строки. Первый оператор echo выведет строку «Hello», а второй — «$s». Между строками в кавычках и в апострофах существует большая разница. Если строка заключена в апострофы, то все символы трактуются как есть. Исключения составляют последовательности \' и \\. Первая представляет собой обыкновенный апостроф, а вторая — обратную наклонную черту.

http://all-ebooks.com

Строки в кавычках позволяют, кроме всего прочего, вывести значение переменной, поэтому никогда не используйте строку в апострофах для этой цели. Предположим, что у нас есть переменная `$a = 10`. Проанализируем вывод двух операторов echo:

```
echo "$a"; // строка в кавычках
echo '$a'; // строка в апострофах
```

Первый оператор выведет 10 — значение переменной `$a`. Второй напечатает то, что заключено между апострофами — `$a`. Улавливаете разницу?

Кроме того, строки в кавычках могут содержать следующие специальные символы:

`\n` — символ новой строки;

`\r` — символы возврата каретки;

`\t` — символ табуляции;

`\ $` — знак доллара;

`\"` — кавычка;

`\\` — обратный слэш;

`\xNN` — символ с кодом NN (в шестнадцатеричной системе).

### Строковые операции

Строковых операций в PHP всего две:

`$s1.$s2` — конкатенация (слияние) двух строк;

`$s1[n]` — обращение к символу с номером n в строке.

Все остальные действия над строками выполняются стандартными функциями.

*Присвоить значение строке очень просто:*

```
$s   =    "hello";
$s   =    "Это   большая
многострочная  строковая константа";
```

### Многострочные строковые переменные

Начиная с четвертой версии PHP, появились так называемые Here-документы. Это еще один способ записи многострочных констант. Длина строки в PHP ограничена только объемом свободной оперативной памяти, поэтому в переменную можно записать даже небольшой файл. Записать большое текстовое сообщение можно так:



http://all-ebooks.com

```
$s =<<<LABEL
Текст
...
LABEL;
```

При использовании этого способа записи многострочных строковых переменных нужно помнить три правила:

- LABEL может быть любым идентификатором.
- После LABEL и до конца строки не должно быть непробельных символов.
- После завершающего LABEL должна стоять точка с запятой.

Как соединить значение строковой переменной со строковой константой? Попробуем вывести строку «Hello world» так:

```
<?
$a = "Hell";
echo "$ao world";
?>
```

Ничего не получится: интерпретатор сообщит, что переменная `$ao` не найдена. Чтобы выйти из этой ситуации, можно явно выполнить операцию слияния строк:

```
echo $a."o world";
```

А можно взять имя переменной в фигурные скобки:

```
echo "${a}o world"
```

или

```
echo "{$a}o world"
```

Рекомендуется использовать запись `${a}`, поскольку иногда фигурные скобки могут трактоваться как спецсимволы.

Строки в обратных апострофах (вызов внешней программы)

Вы хотите запустить внешнюю программу и присвоить переменной результат ее работы? Нет ничего проще! Для этой цели используйте строки в обратных апострофах (на клавиатуре под символом тильды):

```
$directory = `dir`;  // обратите внимание, именно обратные
                     // апострофы ``, а не прямые '' !
echo "<h1>Содержимое каталога:</h1><br><pre> $directory";
```

### 5.5.4. Оператор эквивалентности (===)

Иногда обычный оператор сравнения = = (два знака равенства) работает не совсем так, как нам хочется. Рассмотрим следующий пример:

```
<?
$x = 0;  // число, равно 0
$s = "";  // пустая строка

echo gettype($x)."\n";
echo gettype($s)."\n";

if ($x==$s) echo "x = s";
?>
```

Переменные `$x` и `$s` разного типа, в чём нас убеждает вывод результата вызова функции `gettype()`, но программа сообщает, что их значения равны! Все дело в том, что PHP трактует одну из переменных (обе они эквивалентны логическому false) как логическую, а если хотя бы одна переменная логическая, то вторая тоже считается логической. Поэтому мы получаем false = = false, следовательно, программа выведет строку «x=s».

Чтобы избежать подобной ситуации, нужно использовать оператор эквивалентности = = = (три знака равенства). Если модифицировать нашу программу так:

```
<?
$x = 0;  // число, равно 0
$s = "";  // пустая строка

echo gettype($x)."\n";
echo gettype($s)."\n";

if ($x===$s) echo "x = s";
?>
```

то сейчас строка «x=s» выведена не будет.

Проведем еще один эксперимент с использованием массивов. Понимаю, что о массивах в PHP вы еще ничего не знаете, но сейчас просто необходимо забежать немного вперед.

```
<?
$A = array('1'=>'0001'); // используются апострофы, а не кавычки
$B = array('2'=>'0010');

if ($A == $B) echo "A= =B";
if ($A === $B) echo "A= = =B";

?>
```



http://all-ebooks.com

Будет выведена первая строка, но не вторая. В версиях PHP 4.0.x и ниже массивы $A и $B преобразовывались в строки "array", которые и сравнивались. Вот из-за этого оператор == неправильно работал. Оператор === с самого начала лишен этого недостатка. Точно могу сказать, что в версии 4.1.2 оператор == работает корректно, не говоря уже о пятой версии PHP.

Для оператора === должен быть обратный оператор !===, но его нет. Если вы напишете:

```
if ($A !=== $B) echo "A<>B";
```

то получите сообщение об ошибке. Но можно обойти это неудобство так:

```
if (!($A === $B)) echo "A<>B";
```

## 5.6. Ссылки

Как уже отмечалось ранее, в PHP нет такого понятия, как указатель. Хорошо это или плохо, сказать сложно. Указатели являются одновременно достоинством и недостатком языка C: в них слишком легко запутаться, а ошибки, связанные с указателями, очень трудно найти. Лично мне больше нравится идеология указателей в языке Pascal: вы можете нормально работать как с указателями, так и без них.

Во всяком случае, самое большее, на что можно рассчитывать в PHP, это ссылочные переменные или просто ссылки. Ссылки бывают двух типов: символические и жесткие. Последний тип ссылок появился в четвертой версии интерпретатора.

### 5.6.1. Жесткие ссылки

Жесткая ссылка – это просто псевдоним другой переменной. Ссылки на ссылку на переменную (многоуровневые ссылки), как в языке Perl, не поддерживаются.

Для создания жесткой ссылки используется оператор &:

```
<?
$x = 77;
$link = &$x;

$link = 66;
echo $x;
?>
```

Нетрудно догадаться, что выведет программа: число 66. Для разрыва связи между переменной и ссылкой используется функция `UnSet`:

```
UnSet($link);
```

Саму переменную нельзя удалить до тех пор, пока на нее ссылается хотя бы одна ссылка.

### 5.6.2. Символические ссылки

Символическая ссылка – это обыкновенная переменная, содержащая имя другой переменной. Для доступа к значению ссылки используется двойной знак доллара:

```
<?
$x = 77;
$link = "x";          // символическая ссылка на переменную $x
echo $$link;          // выводит 77
echo $link;           // выводит x
$$link = 66;          // аналогично $x = 66
echo $x;              // выведет 66

?>
```

Я не рекомендую использовать символические ссылки, чтобы не запутывать и без того запутанную программу. Даже к жестким ссылкам стоит прибегать только в самых крайних случаях, когда без ссылки никак не обойтись.

## 5.7. Основные конструкции языка

Конструкции языка PHP очень похожи на конструкции языка C, поэтому, если вы знаете язык C, вы очень быстро освоите PHP. Нужно заметить, что синтаксис PHP очень лаконичен, а мы знаем, что краткость – сестра таланта. PHP еще раз позволяет нам убедиться в этом.

### 5.7.1. Условный оператор

Конструкция `if-else` (условный оператор) является, наверное, самой распространенной конструкцией в программах. Какая же серьезная программа не использует эту конструкцию?

Синтаксис конструкции `if-else` таков:



http://all-ebooks.com

```
if (логическое выражение)
 оператор_1;
else
 оператор_2;
```

Первый оператор будет выполнен, если логическое выражение истинно, второй — если логическое выражение ложно. Можно также использовать сокращенную форму конструкции, без второго оператора:

```
if (логическое выражение)
 оператор_1;
```

В этом случае мы можем обойтись без «else». Если вам нужно выполнить не один оператор, а целую группу, тогда эти операторы следует поместить в блок операторов { }:

```
if (логическое выражение)
 {
 оператор_1;
 ...
 оператор_i;
 }
else
 {
  оператор_i+1;
 ...
  оператор_n;
 }
```

В этой книге я еще ни разу не привлекал вашего внимания к разделителю операторов, символу точки с запятой ";". Операторы в блоке { } можно записать в одну строку:

```
if (логическое выражение)  {  оператор_1;  ...  оператор_i;  }
```

В этом случае после фигурной скобки можно ставить точку с запятой, а можно и не ставить. Но если вы думаете использовать полную конструкцию (то есть с "else"), тогда точка с запятой не ставится. Если же вместо блока операторов используется только один из операторов (см. общий синтаксис), тогда точка с запятой после него (перед else) обязательна:

```
if ($a>5) echo "a>5";
  else echo "a<=5";
```

Существует еще один вариант конструкции if-else. Вот его синтаксис:

```
if (логическое выражение 1)
 оператор_1;
elseif ( логическое выражение 2 )
 оператор_2;
else
 оператор_3;
```

По-моему, эта конструкция не очень удобна, поэтому я советую использовать ее классический вариант.

## 5.7.2. Циклы

На втором месте по частоте использования после условного оператора находятся циклы. Цикл позволяет повторить один или несколько операторов определенное, зависящее от условия цикла, количество раз. Повторяемые операторы называются телом цикла. Один проход цикла называется итерацией.

Как и C, PHP поддерживает три вида циклов:

- Цикл с предусловием (while).
- Цикл с постусловием (do-while).
- Цикл со счетчиком (for).

### Цикл с предусловием while

Принцип работы цикла с предусловием:

- Вычисляется значение логического выражения.
- Если значение истинно, выполняется тело цикла, в противном случае переходим на следующий за циклом оператор.

Синтаксис цикла с предусловием выглядит следующим образом:

```
while ( логическое выражение )
    инструкция;
```

В данном случае телом цикла является "инструкция". Обычно тело цикла состоит из большего числа операторов. Простейший пример цикла:

```
<?

$i=0;
while($i++ < 10) echo $i;

?>
```

Данная программа выведет строку 12345678910. Обратите внимание на последовательность выполнения операций при проверке условия (`$i++ < 10`). Сначала проверяется условие, а только потом увеличивается значение переменной. Если же мы поставим операцию инкремента перед



http://all-ebooks.com

переменной (++$i < 10), то сначала будет выполнено увеличение переменной, а только потом сравнение. В результате мы получили бы строку 123456789.

Этот же цикл можно было бы записать по-другому:

```
$i = 0;
while ($i<10)
 {
 $i ++;// увеличение счетчика
 echo $i;
 }
```

Если мы увеличим счетчик после выполнения оператора echo, мы получим строку 0123456789. Но в любом случае у нас будет 10 итераций.

### Цикл с постусловием do-while

Цикл с постусловием отличается от цикла с предусловием тем, что сначала выполняется тело цикла, а только потом уж проверяется условие. Таким образом, тело цикла хотя бы один раз, но будет обязательно выполнено.

Синтаксис цикла такой:

```
do
 {
 // тело цикла
 }
while (условие);
```

Попробуем с помощью цикла с постусловием получить строку 12345678910. Поскольку в этом цикле сначала выполняется тело, а потом проверяется условие, то нам нужно установить начальное значение счетчика в единицу:

```
<?
$i = 1;
do echo $i; while ($i++ <10);
?>
```

Напомню, что мы можем увеличивать значение счетчика в теле цикла (так даже понятнее).

### Цикл со счетчиком for

Цикл со счетчиком используется для выполнения тела цикла наперёд заданное количество раз. Если цикл **while** удобен для ожидания некоторого события (например, возврата вызываемой функцией значения true), то цикл **for** удобнее для таких задач, как заполнение массива.

Синтаксис цикла **for**:

```
for (команды_инициализации; условие; команды_после_итерации) { тело
цикла }
```

Оператор for начинает свою работу с выполнения команд инициализации. Эти команды выполняются только один раз. После этого проверяется условие: если оно истинно, выполняется тело цикла. После того как будет выполнен последний оператор тела, выполняются команды «после итерации».

Затем снова проверяется условие, в случае если оно истинно, выполняется тело цикла и поститерационные команды и т. д.

Выведем привычную для нас строку:

```
<?

for ($i=0; $i<10; $i++) echo $i;

?>
```

Данный сценарий выведет строку 0123456789. Если мы хотим получить строку 12345678910, то нужно обеспечить увеличение счетчика при проверке логического выражения:

```
<?

for ($i=0; $i++<10;) echo $i;

?>
```

Как видите, команды после итерации можно опустить.

Оба списка, и **команды_инициализации**, и **команды_после_итерации**, могут состоять из нескольких команд. В этом случае команды нужно разделять запятыми:

```
for ($i=0, $j=0; $i<10; $i++, $j++) echo $i;
```

Как вы уже успели заметить, цикл со счетчиком является вариацией цикла с предусловием, так как сначала в нем проверяется условие, а только потом выполняется тело.

#### Альтернативный синтаксис

Циклы **while** и **for** имеют альтернативный синтаксис. Для цикла **while** он таков:

```
while (логическое_выражение):
```



http://all-ebooks.com

```
    тело цикла
endwhile;
```

А для цикла **for** - такой:

```
for (команды_инициализации; условие; команды_после_итерации):
    тело цикла
endfor;
```

Обратите внимание на двоеточие — его наличие обязательно! Альтернативный синтаксис также предусмотрен для оператора **if-else**:

```
if (логическое_выражение):
    команды;
elseif (логическое_выражение2):
    команды;
else:
    команды;
endif;
```

Использовать данный синтаксис или нет, ваше личное дело. Я же предпочитаю использовать «классику» — обычный синтаксис.

Использовать альтернативный синтаксис имеет смысл, если вы предпочитаете включать HTML-код в PHP-сценарии, а не выводить его с помощью оператора echo.

В этом случае альтернативный синтаксис облегчает использование операторов **if-else**, **while** и **for**:

```
<?if  ($REMOTE_ADDR=="127.0.0.1") :?>
<h1>Добро пожаловать домой!</h1>
<?else:?>
<h1>Добро пожаловать на наш сайт!</h1>
<?endif;  ?>
```

#### Вложенные циклы

Каждый цикл может содержать в своем теле несколько циклов, называемых вложенными циклами. Вкладываться может неограниченное число циклов, причем все они могут быть разного типа — **while**, **do** или **for**.

Рассмотрим следующий код:

```
for ($I=0; $I<10; $I++) {
    echo "$I\n";
    for ($J=0; $J<5; $J++) echo "$J ";
    echo "\n";
}
```

Вложенный цикл выводит строку 0 1 2 3 4 десять раз, по числу итераций внешнего цикла for.

### Операторы break и continue

Управлять количеством итераций цикла позволяют операторы **break** и **continue**. Первый прерывает работу всего цикла, а второй – только текущей итерации. Рассмотрим следующие примеры:

```
<?
$i = 0;
while ($i++ < 10)
 {
 if ($i==3) break;
 echo "Итерация $i\n";
 }

?>
```

В этом примере после второй итерации работа цикла будет прервана и программа выведет строки:

```
Итерация 1
Итерация 2
```

Если же вместо оператора **break** использовать оператор **continue**, будет прервана работа только третьей итерации и мы получим вывод:

```
Итерация 1
Итерация 2
Итерация 4
Итерация 5
Итерация 6
Итерация 7
Итерация 8
Итерация 9
Итерация 10
```

Предположим, что у нас есть вложенные циклы, например:

```
for (...)
 {
    for (...)
    {
      for (...)
    }
 }
```



http://all-ebooks.com

Как прервать работу первого (внешнего) цикла, если оператор **break** находится в теле третьего? Номер прерываемого цикла можно передать функции **break** как параметр: break(1). Нумерация циклов начинается с единицы.

#### Цикл перебора массивов foreach

В четвертой версии PHP появился еще один вид цикла – **foreach**, предназначенный специально для перебора массивов.

Синтаксис данного цикла таков:

```
foreach (массив as $ключ=>$значение)
    операторы;
```

Операторы, содержащиеся в теле цикла, будут выполнены для каждого элемента массива. Переменная `$ключ` будет содержать имя ключа элемента, а переменная `$значение` — значение элемента.

Следующий пример отобразит значение массива `$GLOBALS`, содержащего глобальные переменные (будут выведены как имена переменных, так и их значения):

```
<?
foreach($GLOBALS as $key=>$value)
 echo "$key = $value\n";

?>
```

Вывод программы выглядит примерно как на рис. 5.3

> **Примечание.**
>
> Массив — это упорядоченный набор данных. Каждый элемент массива имеет индекс или ключ. Ключ служит для однозначной идентификации элемента массива. В одном массиве не может быть двух элементов с одинаковыми ключами.

В PHP есть один очень полезный стандартный массив `$GLOBALS`. Ключом данного массива является имя переменной, а значением элемента массива — значение переменной. Об этом важном массиве мы еще не раз поговорим в этой книге.

### 5.7.3. Оператор switch-case (конструкция выбора)

Нам осталось рассмотреть очень важную конструкцию — **switch-case**. Она предназначена для выбора действий в зависимости от значения указанного

выражения. Конструкция **switch-case** чем-то напоминает оператор **if-else**, который, по сути, является ее аналогом. Конструкцию выбора нужно использовать, если предполагаемых вариантов много (скажем, больше 5) и для каждого варианта нужно выполнить специфические действия. В этом случае использовать оператор выбора **if-else** просто неудобно.

Синтаксис **switch-case** таков:

```
switch (выражение) {
case значение1 :  команды_1; [break;]
...
case значениеN :  команды_N; [break;]
[default: команды_по_умолчанию; [break;]]
}
```

Альтернативный синтаксис конструкции выглядит так:

```
switch (выражение):
  case значение1 : команды_1; [break;]
   ...
  case значениеN : команды_N; [break;]
[default: команды_по_умолчанию; [break;]]
endswitch;
```

Работает эта конструкция следующим образом:

- Вычисляется значение выражения.
- Просматривается набор значений. Если вычисленное значение равно `значению_i`, то выполняется блок команд `команды_i`. После этого, если оператор break присутствует, выполнение оператора выбора завершается. Если оператора break нет, то выполняются блоки `команды_i+1, команды_i+2, .., команды_N`.
- Если ни одно значение из набора не совпало со значением выражения, тогда выполняется блок **default**, если он присутствует.

Небольшой пример:

```
<?
$age=21;
switch ($age)
{
case 20 : echo "Вам двадцать лет\n";
case 21 : echo "Вам двадцать один год\n";
case 22 : echo "Вам двадцать два года\n";
default: echo "Вам $age лет\n";
}
?>
```



http://all-ebooks.com

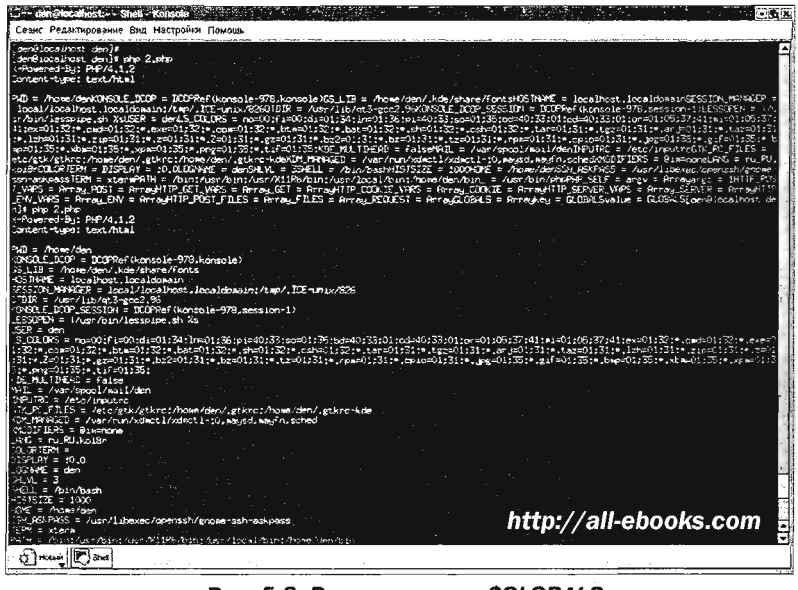



***Рис. 5.3. Вывод массива $GLOBALS***

Если переменная `$age` равна 21, то будут выведены строки:

```
Вам двадцать один год
Вам двадцать два года
Вам 21 лет
```

Исправить этот сценарий так, чтобы он выводил только одну нужную строку, можно добавлением операторов **break**:

```
<?
$age=21;
switch ($age)
{
case 20 : echo "Вам двадцать лет\n"; break;
case 21 : echo "Вам двадцать один год\n"; break;
case 22 : echo "Вам двадцать два года\n"; break;
default: echo "Вам $age лет\n"; break;
}
?>
```

Если вы знаете язык Pascal, то помните, что в нем есть конструкция case, позволяющая указывать диапазоны значений:

```
case (age)
10..20 : writeln('Вам от 10 до 20 лет');
...
end;
```

http://all-ebooks.com

В PHP такую возможность можно реализовать так (диапазон 20..22):

```
<?
$age=21;
switch ($age)
{
case 20 :
case 21 :
case 22 : echo "Вам от 20 до 22\n"; break;
}
?>
```

## 5.8. СОБИРАЕМ ОДИН СКРИПТ ИЗ НЕСКОЛЬКИХ

### 5.8.1. Инструкция require

Инструкция **require** позволяет включить код до выполнения нашего сценария. Общий синтаксис такой:

```
require "имя_файла";
```

Разберемся подробнее, что же делает require. Интерпретатор анализирует сценарий и, найдя **require**, просто заменяет ее содержимым указанного файла. Это может быть HTML-файл или сценарий на PHP. В последнем случае содержимое файла должно быть обрамлено тегами <? и ?>, как и для обычного сценария. Вставка содержимого файла происходит до выполнения сценария — прочитав следующий пункт, вы заметите разницу.

Очень удобно использовать эту инструкцию для включения HTML-заголовков. Например:

Файл header.html:

```
<html><head>
<title>My Company's Official Web Page</title>
</head>
<body>
```

Сценарий:

```
require "header.html"
/* тело документа */
```

### 5.8.2. Инструкция include

В отличие от **require** директива **include** позволяет включить код в сценарий во время выполнения сценария.



http://all-ebooks.com

Общий синтаксис такой:

```
include "имя_файла";
```

Возможно, вам нужно динамически менять заголовок в зависимости от ситуации:

```
for ($i=0; $i<4; $i++) {
include "header{$i}.html";
}
```

Создайте четыре файла: `header0.html`, `header1.html`, `header2.html` и `header3.html`. Пусть каждый из них содержит соответствующую имени файла цифру — от 0 до 3. При выполнении сценария вы увидите вывод 0123. Данный вывод означает, что содержимое файлов вставляется в сценарий во время его выполнения. Замените **include** на **require**, и вы почувствуете разницу.

Обратите внимание на фигурные скобки. При таком контексте использования **include** они обязательны. Почему? Интерпретатор просто заменяет **include** содержимым файла. Представьте, что наш файл содержит не один, а несколько операторов. Получится следующее:

```
for ($i=0; $i<4; $i++)
operator1;
operator2;
operator3;
```

Как вы видите, мы получаем совсем не то, на что рассчитывали. Мы хотели, чтобы все три оператора выполнились четыре раза, а на самом деле четыре раза выполняется только первый оператор, а остальные два будут выполнены после окончания работы цикла. Отсюда вывод: если вы забудете заключить **include** в фигурные скобки (в цикле), то ваша программа будет работать некорректно. Всегда заключайте инструкцию **include** в фигурные скобки!

### 5.8.3. Особенности использования include и require

Что же происходит на самом деле при использовании **include**? Мы знаем, что PHP преобразует наш сценарий в собственное внутреннее представление. Первую строку, вторую, третью... пока он не дойдет до инструкции **include**. Затем PHP прекращает транслировать наш сценарий и переключается на указанный в **include** файл. Вполне объяснимо: интерпретатор не знает, что будет в этом файле, поэтому не может отложить его трансляцию «до лучших времен».

Если у вас большая программа и она подключает дополнительные файлы с помощью **include**, вряд ли она будет работать быстро, поскольку из-за особенностей **include** быстродействие резко снижается.

С require таких проблем нет: файл включается до выполнения сценария, то есть на момент трансляции он уже включен в наш сценарий и PHP нет необходимости ждать, пока оттранслируется указанный файл.

Что же использовать? Если вам нужно подключить какой-то дополнительный модуль (например, класс **HtmlMimeMail**, который будет рассматриваться в этой книге), используйте **require**. Если вы точно знаете, что вам нужно включить какой-то файл не один раз, используйте **include**, однако старайтесь по возможности использовать **require** везде, где только можно.

### 5.8.4. Инструкции однократного включения include_once и require_once

Ваша программа разрослась до такой степени, что вы не можете вспомнить, какой файл вы включали, а какой — нет? Такого быть не может! Неужели не видно, что и где включается?

Такая ситуация встречается довольно часто при разработке действительно больших сценариев. Предположим, что у нас есть сценарий `main.php`, в который включены файлы `header.php`, `code.php`, `footer.php`. В файл `code.php` тоже включается несколько файлов. В одном из этих файлов находится инструкция:

```
require    "header.php";
```

Получается, что файл `header.php` включен дважды. Двойное включение может стать причиной серьезной ошибки, которую очень трудно найти.

Разработчики PHP предусмотрели выход из этой ситуации. Используя инструкции однократного включения `include_once` и `require_once`, вы можете быть уверены: ни один файл не будет включен дважды. Работают эти инструкции точно так же, как и **include** и **require**, с одним лишь отличием: перед включением файла интерпретатор проверяет, включался ли этот файл уже или нет. Если да, то файл не будет включен в состав сценария.

## 5.9. Обработка ошибок

Интерпретатор PHP позволяет программисту определить, какие сообщения об ошибках нужно выводить, а какие нет. Пока вы отлаживаете программу, я рекомендую выводить все сообщения об ошибках и все предупреждения, а потом, когда программа нормально работает, выводить только сообщения об ошибках.



http://all-ebooks.com

Параметр **error_reporting** в файле конфигурации PHP (`/etc/php.ini` или `c:\windows\php.ini`) позволяет определить степень обработки и протоколирования ошибок. Возможные значения:

- `E_ALL` – все ошибки и предупреждения;
- `E_ERROR` – фатальные ошибки во время выполнения программы (например, невозможно открыть файл);
- `E_WARNING` – предупреждения времени выполнения (не фатальные ошибки);
- `E_PARSE` – синтаксические ошибки;
- `E_NOTICE` – замечания времени выполнения (например, сообщение о том, что переменная не инициализирована).

Кроме вышеперечисленных, возможны и другие значения, но они не столь важны. Я рекомендую включить оповещения обо всех ошибках и предупреждениях:

```
error_reporting = E_ALL
```

Когда программа отлажена, можно отключить замечания времени выполнения. Для отключения какого-нибудь вида ошибок используется тильда:

```
error_reporting  =  E_ALL & ~E_NOTICE
```

Если вы не можете отредактировать файл `php.ini` (например, у вас нет на это прав), то можете из сценария вызвать функцию `error_reporting()`:

```
error_reporting(E_ALL);
```

Следует учитывать, что действие директивы `error_reporting` в файле конфигурации распространяется на все ваши сценарии, а вызов функции `error_reporting()` – только на текущий сценарий.

Существует еще один способ избежать вывода сообщения об ошибке – это использование оператора @. Например, если вы не хотите, чтобы пользователь видел сообщение об ошибке какой-нибудь стандартной функции, нужно указать оператор @ перед именем функции, например:

```
@mysql_query("create table tbl (no int, info text)");
```

# ГЛАВА 6.

# ПЕРЕДАЧА ПАРАМЕТРОВ СЦЕНАРИЯМ

http://all-ebooks.com

## 6.1. ФОРМЫ В HTML-ДОКУМЕНТАХ

### 6.1.1. Общие положения

В этом небольшом разделе мы рассмотрим создание HTML-форм. Возможно, вы уже изучали этот материал, когда осваивали HTML. В этом случае вы можете сразу перейти к п. 6.2, в котором мы непосредственно будем рассматривать передачу параметров с помощью форм. Однако при изучении языка HTML формы обычно изучаются поверхностно, так как без знания CGI они ни к чему. Поэтому я все-таки рекомендую восстановить в памяти механизм создания форм. Чтобы вам не обращаться к другим книгам, я специально выделил небольшой раздел, посвященный данному вопросу.

Итак, формы предназначены для отправки информации CGI-приложению, то есть нашему сценарию. Рассмотрим простую форму ввода имени:

**Листинг 6.1. Наша первая форма**

```
<form action="http://localhost/1.php" method="GET">
Имя: <input type = text name=user_name><br>
<input type=submit value="Передать">
</form>
```

Наша форма содержит всего два элемента: поле для ввода имени и кнопку с надписью «Передать». Что происходит, когда пользователь вводит имя и нажимает на кнопку «Передать»? Браузер передаст сценарию `1.php` пара-

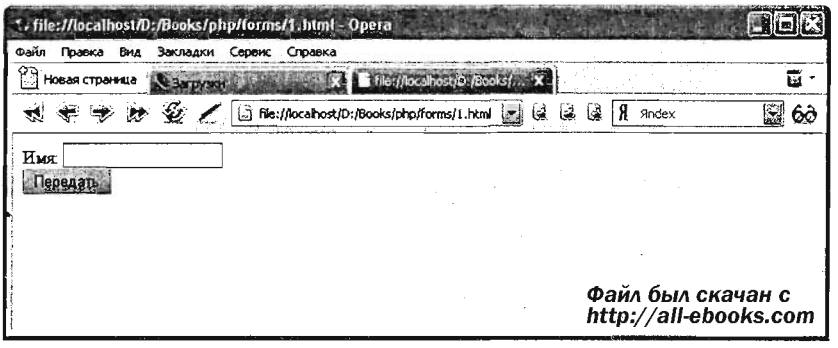



***Рис. 6.1. Форма передачи имени в окне браузера***

метр `user_name`. В качестве значения параметра будет введенное пользователем имя. Так как используется метод GET, то в строке адреса вы увидите:

```
http://localhost/1.php?user_name=Denis
```

## 6.1.2. Элементы форм

Форма в HTML-документе реализуется тегом-контейнером FORM, в котором задаются все управляющие элементы — поля ввода, кнопки и т. д. Если управляющие элементы указаны вне содержимого тега FORM, то они не создают форму, а используются для построения пользовательского интерфейса на веб-странице, то есть для привнесения в нее различных кнопок, флажков, полей ввода. Обработку таких элементов выполняют скрипты, включенные в HTML-документ.

Например, управляющий элемент (тег) TEXTAREA часто используется для создания окна с полосой прокрутки внутри документа для вывода большого текста, который играет второстепенную роль. Обычно так отображаются тексты лицензионных соглашений, тексты больших комментариев или правила пользования данным веб-ресурсом.

Имена элементам формы присваиваются через их атрибут NAME.

Каждый элемент формы имеет начальное и конечное значения строкового типа. Начальное значение присваивается по умолчанию, конечное вводится пользователем.

Начальные значения элементов не меняются, благодаря чему возможно «сбросить» форму, то есть снова присвоить всем её элементам первоначальные значения.



http://all-ebooks.com

В HTML 4.01 определены следующие типы управляющих элементов:

- **Кнопки** — задаются с помощью элементов BUTTON и INPUT. Различают:
  - **кнопки отправки** — при нажатии на них форма посылается серверу;
  - **кнопки сброса** — при их нажатии управляющим элементам присваиваются начальные значения;
  - **прочие кнопки** — кнопки, для которых не указано действие, выполняемое по умолчанию при их нажатии.

- **Зависимые переключатели** (переключатели с зависимой фиксацией) — задаются элементом INPUT и представляют собой переключатели «вкл\выкл». Если несколько зависимых переключателей имеют одинаковые имена, то они являются взаимоисключающими. Это значит, что если одна из них ставится в положение «вкл», то все остальные автоматически — в положение «выкл». Именно это и является преимуществом их использования.

- **Независимые переключатели** (переключатели с независимой фиксацией) — задаются элементом INPUT и представляют собой переключатели «вкл\выкл». В отличие от зависимых независимые переключатели могут принимать и изменять значения независимо от остальных переключателей, даже если последние имеют такое же имя.

- **Меню** — реализуется с помощью элементов SELECT, OPTGROUP и OPTION. Меню предоставляют пользователю список возможных вариантов выбора.

- **Ввод текста** — реализуется элементами INPUT, если вводится одна строка, и элементами TEXTAREA, если несколько строк. В обоих случаях введенный текст становится текущим значением управляющего элемента.

- **Выбор файлов** — позволяет вместе с формой отправлять выбранные файлы, реализуется HTML-элементом INPUT.

- **Скрытые управляющие элементы** — создаются управляющим элементом INPUT.

Как видите, очень много элементов задаются с помощью одного и того же тега INPUT. Этот тег действительно является универсальным.

### 6.1.3. Тег FORM – контейнер форм

Как уже говорилось, форма в HTML-документе реализуется тегом-контейнером FORM. Этот тег своими атрибутами указывает адрес сценария, которому будет послана форма, способ пересылки и характеристику данных, содержащихся в форме. Начальный и конечный теги FORM задают границы формы, поэтому их указывать обязательно. Пример приводить еще раз не буду, он был приведен в п. 6.1.1. Однако атрибуты тега FORM подробно рассмотреть все-таки следует:

- **action** – единственный обязательный атрибут. В качестве его значения указывается URL запрашиваемой CGI-программы, которая будет обрабатывать данные, содержащиеся в форме. Допустимо использовать запись MAILTO:URL, благодаря которой форма будет послана по электронной почте. Если атрибут ACTION все-таки не указан, то содержимое формы будет отправлено на URL, с которого загружалась данная веб-страница.
- **method** – определяет метод HTTP, используемый для пересылки данных формы от браузера к серверу. Атрибут METHOD может принимать два значения: GET или POST.
- **enctype** – необязательный атрибут. Указывает тип содержимого формы, используемый для определения формата кодирования при ее пересылке. В HTML определены два возможных значения для атрибута ENCTYPE:
    - APPLICATION/X-WWW-FORM-URLENCODED (используется по умолчанию).
    - MULTIPART/FORM-DATA.

### 6.1.4. Тег INPUT и способы его использования

Элемент INPUT является наиболее употребительным тегом, так как с его помощью реализуются основные функции формы. Он позволяет создавать внутри формы поля ввода строки текста, имени файла, пароля и др.

Обратите внимание на особенность INPUT – у него нет конечного тега. Атрибуты и особенности использования INPUT зависят от способа его использования. Эти способы мы сейчас и рассмотрим.

#### Однострочные поля ввода

Наиболее часто используются поля ввода – ведь даже кнопка является полем ввода. Мы уже знакомы с полем ввода текстовой информации, поэтому



http://all-ebooks.com

обзор полей ввода начнем именно с текстового поля. Формат тега INPUT для создания поля ввода текстовой строки:

```
<input type=text
    name=имя_параметра
    [value=значение]
    [size=размер]
    [maxlen=длина]
>
```

Этот тег создает поле ввода с максимально допустимой длиной текста **maxlen** и размером в **size** знакомест. Если задан атрибут **value**, то в поле будет изначально отображаться указанная строка. В квадратных скобках помечены необязательные атрибуты.

### Поле ввода пароля

Возможно, вы создаете какой-нибудь серьезный проект, требующий авторизации, то есть проверки имени пользователя и пароля. Конечно, имя пользователя можно ввести с помощью обыкновенного текстового поля.

А вот как быть с паролем? Ведь он не должен отображаться на экране? В этом случае вам поможет поле ввода пароля:

```
<input type=password
    name=имя_параметра
    [value=значение]
    [size=размер]
    [maxlen=длина]
>
```

Принцип работы данного тега точно такой же, как и текстового. Разница заключается в том, что вводимая информация в поле не отображается, а заменяется «звездочками».

Следует иметь в виду, что не рекомендуется устанавливать значение **value** (значение по умолчанию) из соображений безопасности. Да, в окне браузера данное значение не отображается, но стоит просмотреть исходный HTML-код, и пароль будет виден невооруженным глазом.

### Скрытое текстовое поле

Для передачи служебной информации (о которой пользователь даже не должен подозревать) используются скрытые поля. С помощью таких полей, например, могут передаваться параметры настройки.

```
<input type=hidden
    name=имя_параметра
    value=значение
>
```

Такие поля передаются серверу, но на web-странице не отображаются.

### Независимые переключатели (флажки)

Очень часто пользователю, заполняющему форму у себя в браузере, необходимо дать возможность указать свои настройки, ответив на некоторые вопросы «да» или «нет». Для этого форма содержит ряд значений, около каждого из которых помещается небольшое квадратное поле. Пользователь может поставить в этом поле галочку (значение будет выбрано) или снять её (значение не выбрано).

Реализовать это можно опять же с помощью тега INPUT. Для этого только необходимо в качестве значения атрибута **type** указать **checkbox**:

```
<input type=checkbox
  name=имя
  value=значение
  [checked]
>
```

Если флажок был включен на момент нажатия кнопки отправки данных, то сценарию будет передан параметр **имя=значение**. Если же флажок выключен, то сценарию не будет передано ничего, как если бы этого флажка вообще не было на форме.

Переключатель может быть по умолчанию либо включен, либо выключен. Чтобы переключатель был по умолчанию включен, необходимо для него указать атрибут **checked**.

Переключатель **checkbox** называется независимым, так как его состояние не зависит от состояния других переключателей **checkbox**. Таким образом, в одной форме может быть одновременно выбрано несколько переключателей (рис. 6.2).

### Зависимые переключатели (радиокнопки)

Зависимые переключатели тоже реализуются с помощью тега INPUT, но значением атрибута **type** является **radio**. Зависимость их состоит в том, что на форме может быть только один включенный переключатель типа **radio**.



http://all-ebooks.com

Точнее, если в форме присутствует несколько одноименных зависимых переключателей, то включен может быть только один из них.

При выборе одного переключателя все одноименные зависимые переключатели автоматически выключаются. В качестве имени переключателей воспринимается значение атрибута **name**. Рассмотрим такой пример, приведенный в листинге 6.1.

**Листинг 6.2. Форма с зависимыми переключателями**

```
<form action="http://localhost/sex.php" method="post">
<input type=radio
    name=sex
    value=male
    checked
> Мужчина
<input type=radio
    name=sex
    value=female
> Женщина
<input type=submit
    name=go
    value=Передать
>
</form>
```

В окне браузера наша форма будет выглядеть так, как показано на рис. 6.3.

Первый переключатель (со значением «male») включен по умолчанию (мы установили для него атрибут **checked**). Как только пользователь нажмет

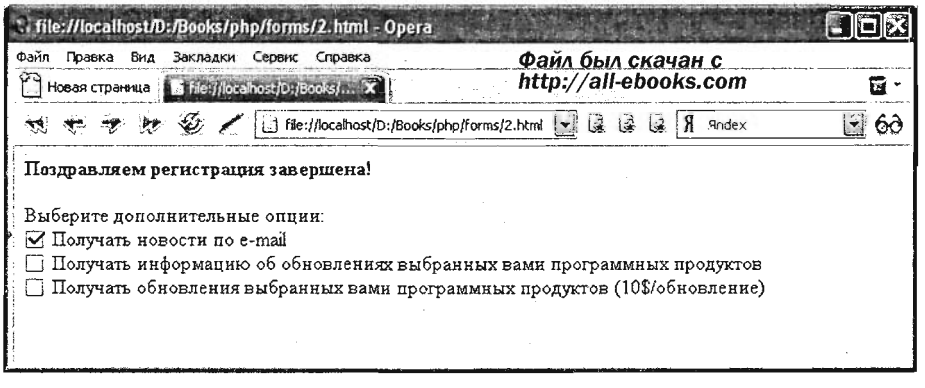



*Рис. 6.2. Использование независимых переключателей checkbox*

кнопку Передать, сценарию `sex.php` будет передан параметр sex (атрибут name обоих переключателей) со значением «male». Если же пользователь выберет другой вариант («female»), на форме будет отмечена другая радиокнопка и сценарию будет передано соответствующее значение параметра **sex**.

### Кнопка отправки формы

Последними элементами управления типа INPUT являются кнопки. Кнопка отправки служит для отправки сценарию введенных в форму значений. Синтаксис тега INPUT при этом таков:

```
<input type=submit
    [name=имя]
    [value=надпись_кнопки]
>
```

Необязательный атрибут **value** задает надпись на кнопке отправки. Необязательный атрибут name определяет имя кнопки. Если он не указан, то скрипту будут переданы введенные в форму значения, и всё. Если же атрибут name для кнопки будет указан, то дополнительно к данным формы будет отправлена пара **имя=значение** от самой кнопки. Я советую обязательно указывать этот атрибут — он нам очень пригодится при программировании на PHP.

Посмотреть, как выглядит кнопка отправки, можно на рис. 6.3.

### Кнопка сброса

Кроме кнопки «Submit», есть еще кнопка «Reset», которая сбрасывает форму — устанавливает для всех элементов формы значения по умолчанию. Желательно, чтобы на вашей форме была такая кнопка, особенно если это большая форма. Наличие данной кнопки облегчает очистку формы, если были введены неправильные параметры.

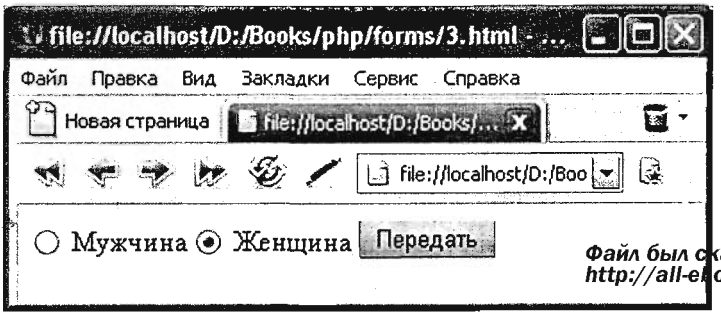
```
<input type=reset
    [value=надпись_кнопки]
>
```

### Кнопка отправки с индивидуальным рисунком

Вместо кнопки «Submit» можно использовать рисунок. При щелчке по этому рисунку произойдет то же, что и при нажатии на кнопку «Submit».



http://all-ebooks.com




Файл был скачан с
http://all-ebooks.com

*Рис. 6.3. Зависимые переключатели radio*

Однако, кроме этого, сценарию будут переданы координаты места произведения щелчка на рисунке. Координаты будут переданы в формате: `имя.x=коор_X`, `имя.y=коор_Y`.

```
<input type=image
    name=имя
    src=рисунок
>
```

## 6.1.5. Ввод многострочного текста. Тег TEXTAREA

Мы переходим к рассмотрению многострочного поля ввода. В HTML эта возможность реализуется с помощью тега TEXTAREA. Поле, создаваемое этим тегом, позволяет вводить и отправлять не одну строку, а сразу несколько. Формат тега TEXTAREA таков:

```
<textarea
    name=имя
    [cols=ширина в символах]
    [rows=высота в символах]
    wrap=тип_переноса
> Текст по умолчанию </textarea>
```

Несколько замечаний относительно использования атрибутов: необязательные параметры `cols` и `rows` указывать всё же желательно. Первый из них задает количество символов в строке, а второй — количество строк в области.

Атрибут **wrap** определяет, как будет выглядеть текст в поле ввода:

- **Virtual** — справа от текстового поля выводится полоска прокрутки. Вводимый пользователем текст выглядит разбитым на строки, а символ новой строки вставляется при нажатии клавиши Enter.

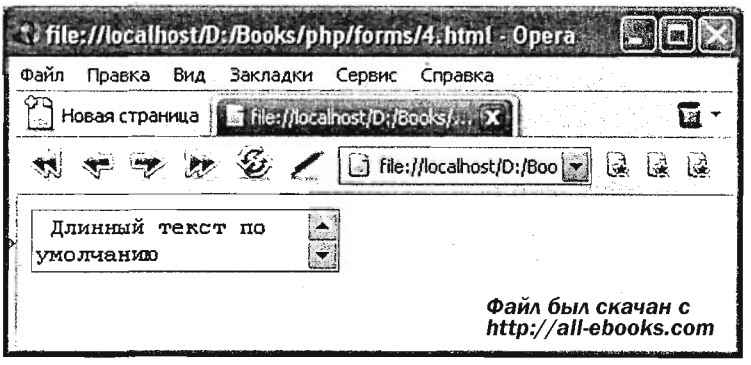



*Рис. 6.4. Поле TEXTAREA*

- **Physical** — этот тип зависит от браузера и в разных браузерах может вести себя по-разному.
- **None** — текст выглядит в точности так, как пользователь его вводит. Если текст не умещается в одну строку, появляется горизонтальная полоска прокрутки.

Стоит заметить, что наиболее удобным является тип **Virtual**. Внешний вид поля можно увидеть на рис. 6.4.

## 6.1.6. СПИСКИ ВЫБОРА. ТЕГ SELECT

### Списки с единственным выбором

Довольно часто нам нужно представить какие-нибудь данные в виде списка и предусмотреть возможность выбора в этом списке. В HTML это реализуется в виде списков выбора, создавать которые можно с помощью тега SELECT. Список выбора позволяет выбрать один вариант из множества. Можно было бы использовать зависимые переключатели **radio**, но это не очень рационально. Например, если нам нужно, чтобы пользователь выбрал месяц, то один список будет занимать намного меньше места в окне браузера, чем целых двенадцать радиокнопок. Вот пример списка выбора, реализованного с помощью тега SELECT:

```
<select name=month size=1>
    <option value=01>Январь</option>
    <option value=02>Февраль</option>
    <option value=03>Март</option>
    <option value=04>Апрель</option>
    <option value=05 selected>Май</option>
```



http://all-ebooks.com

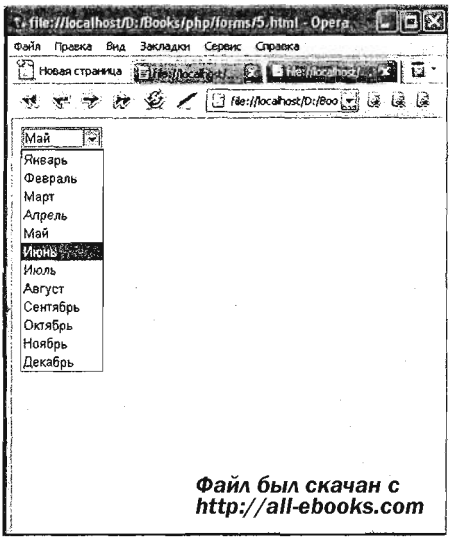



***Рис. 6.5. Тег SELECT. Список с единственным выбором***

```
    <option value=06>Июнь</option>
    <option value=07>Июль</option>
    <option value=08>Август</option>
    <option value=09>Сентябрь</option>
    <option value=10>Октябрь</option>
    <option value=11>Ноябрь</option>
    <option value=12>Декабрь</option>
</select>
```

Как видно, варианты выбора задаются с помощью тегов **option**. При отправке формы сценарию будет отослано значение **value** выбранного элемента списка.

Атрибут name определяет имя параметра, который будет передан сценарию.

Если значение атрибута **size** равно 1, то список будет выпадающим. Если вы укажете значение **size=n** (n больше 1), то список будет занимать на форме n строк и будет снабжен полоской прокрутки.

Значение, выбранное в списке по умолчанию, можно указать с помощью атрибута **selected** для соответствующего тега **option**. В приведенном примере месяц по умолчанию — Май (рис. 6.5).

Атрибут **value** необязателен. Если его не указать, то будет передана строка, заключенная в тег option.

http://all-ebooks.com

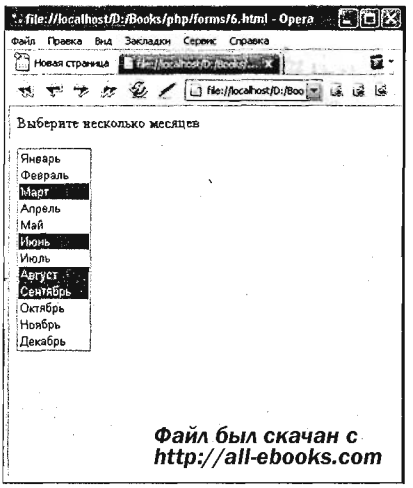



*Рис. 6.6. Тег SELECT. Список с множественным выбором*

### Списки множественного выбора

С помощью тега SELECT можно создавать и списки множественного выбора. В таких списках можно выбрать не одно, а сразу несколько значений. Чтобы сделать список с множественным выбором, необходимо для тега SELECT указать атрибут **multiple**:

```
<select name=month size=12 multiple>
    <option value=01>Январь</option>
    <option value=02>Февраль</option>
    ...
```

Если пользователь выберет несколько вариантов, то сценарию будут переданы параметры в форме:

```
имя=значение&имя=значение...&имя=значение
```

О том, как обработать такие параметры с помощью PHP, вы узнаете далее, в п. 6.2. Пример использования списка множественного выбора показан на рис. 6.6.

## 6.1.7. Загрузка файлов

Тег INPUT позволяет реализовывать еще одну возможность, а именно: он позволяет создавать поле выбора файла для отправки. При этом формат тега таков:

```
<input type=file name=имя [value=имя_файла]
```



http://all-ebooks.com

Более подробно о загрузке файлов мы поговорим в соответствующей главе.

## 6.2. ПЕРЕДАЧА ПАРАМЕТРОВ С ПОМОЩЬЮ ФОРМЫ

В этом пункте мы поговорим о передаче параметров с помощью форм данных. Создадим HTML-документ с формой, использующей все типы полей, описанные в п.6.1. А затем напишем сценарий, который будет обрабатывать переданные параметры. Код HTML-документа приведен в листинге 6.3.

**Листинг 6.3. Все типы полей в одном HTML-документе**

```
<html>

<head>
<title>Форма данных</title>
</head>

<body>

<form action="http://www.dkws.org.ua/form.php" method="POST">
Текстовое поле txt <input type="text" name="txt"><br>
    Поле ввода пароля pswd <input type="password" name="pswd"><br>
    Скрытое поле hid <input type="hidden" name="hid"
                    value="Видишь значение? И я не вижу, а оно
есть!"><br>
<hr>
    <input type="checkbox" name="var1" value="1" checked>Вариант 1
(по умолчанию)
    <input type="checkbox" name="var2" value="2">Вариант 2
    <input type="checkbox" name="var3" value="3">Вариант 3
<hr>
    <input type="radio" name="sex" value="male" checked>Мужчина
    <input type="radio" name="sex" value="female">Женщина
<hr>
    <textarea name="t_area" cols="20" rows="5" wrap="virtual">
    Текст по умолчанию</textarea>

<hr>

<p>Список без множественного выбора:
<select name=month size=1>
    <option value=01>Январь</option>
    <option value=02>Февраль</option>
    <option value=03>Март</option>
```

http://all-ebooks.com

```
    <option value=04>Апрель</option>
    <option value=05 selected>Май</option>
    <option value=06>Июнь</option>
    <option value=07>Июль</option>
    <option value=08>Август</option>
    <option value=09>Сентябрь</option>
    <option value=10>Октябрь</option>
    <option value=11>Ноябрь</option>
    <option value=12>Декабрь</option>
</select>
<br>Список с множественным выбором:
<br><select name="months[]" size=12 multiple>
    <option value=01>Январь</option>
    <option value=02>Февраль</option>
    <option value=03>Март</option>
    <option value=04>Апрель</option>
    <option value=05 selected>Май</option>
    <option value=06>Июнь</option>
    <option value=07>Июль</option>
    <option value=08>Август</option>
    <option value=09>Сентябрь</option>
    <option value=10>Октябрь</option>
    <option value=11>Ноябрь</option>
    <option value=12>Декабрь</option>
</select>
<hr>
    <input type="submit" name="go" value="Отправить">
    <input type="reset" value="Сброс">
</form>

</body>
</html>
```

В окне браузера наша форма будет выглядеть так, как на рис. 6.7.

Когда пользователь нажмет кнопку go, браузер передаст сценарию следующие параметры:

1) `txt` — значение текстового поля;
2) `pswd` — значения поля ввода пароля;
3) `hid` — значение скрытого поля;
4) параметры `var1`, `var2` и `var3` будут переданы только, если соответствующие им переключатели активны (может быть передано одно из значений, любые два значения, все три значения или же ни одно из значений);
5) `sex` — значение группы зависимых переключателей (будет передано одно из значений — male или female);



http://all-ebooks.com

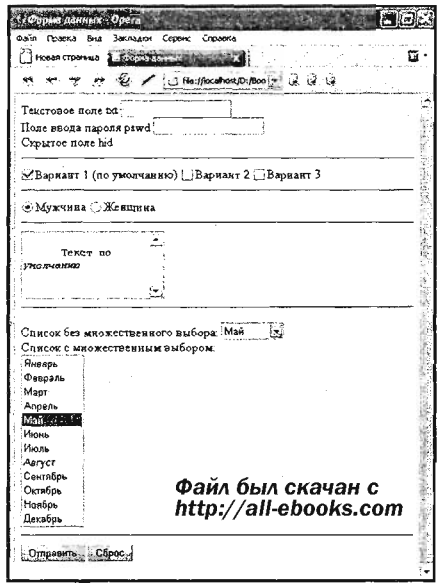



***Рис. 6.7. Форма в окне браузера***

6) `t_area` — содержимое текстовой области;

7) `month` — значение обыкновенного списка;

8) `months` — значение множественного списка.

Теперь нам нужно разобраться, как обработать все эти параметры. Параметры `txt`, `pswd`, `hid` и `t_area` обрабатываются легко. Интерпретатор создает для каждого параметра переменную с соответствующим именем. Например, для отображения значения параметра `txt` достаточно написать:

```
echo    "$txt ";
```

С параметрами `var*` дело обстоит несколько сложнее. Если переключатель неактивен, то параметры вообще не будут переданы на сервер, как будто бы их не было на форме. Следовательно, не будет создана глобальная переменная; при попытке обращения к такой переменной мы получим сообщение о том, что переменная не существует. Поэтому просто написать `echo $var1;` мы не можем: нужно предварительно проверить, существует ли переменная:

```
if (isset($var1)) echo "var1: $var1 ";
if (isset($var2)) echo "var2: $var2 ";
```

Только после проверки существования можно начинать работу с переменной.

http://all-ebooks.com

Параметр sex, то есть наш зависимый переключатель, обрабатывается легко: браузер передает только одно из значений. Другими словами, значение параметра sex будет равно либо «male», либо «female». Поэтому можно просто вывести значение этого параметра:

```
echo $sex;
```

Точно так же обрабатывается параметр $month: это обыкновенный список без возможности множественного выбора, поэтому, по сути, его можно представить, как набор из 12 зависимых переключателей.

Сложнее всего обработать список с множественным выбором. Ведь в этом случае параметры передаются так:

```
name=value1&name=value2&name=value3 ...
```

При стандартном подходе переменная $name будет содержать только последнее значение (valueN). Однако разработчики PHP позаботились об этом: множественный список можно представить в виде массива, а обработать его элементы можно с помощью цикла foreach. Нам даже не нужно знать количество переданных элементов списка. Предварительно лишь нужно сообщить PHP, что мы будем передавать массив:

```
<select name="months[]" size=12 multiple>
```

Квадратные скобки — это признак массива. Обрабатывается массив так:

```
foreach($months as $key=>$value) echo "<br>$key  = $value";
```

Окончательный текст сценария form.php представлен в листинге 6.4.

**Листинг 6.4. Сценарий form.php**

```
<?
/* Проверяем, нажал ли пользователь кнопку go */

if (!isset($go))
{
  echo "Не указаны параметры\n";
  exit(1);
}
else
{
  /* Начинаем обработку параметров */

  echo "<html><body>";
  echo "<b>Текстовые поля:</b><br>";
```



http://all-ebooks.com

```
  echo "txt: $txt  pswd: $pswd  hid: $hid<br>";

  echo "<b>Checkbox</b><br>";
  if (isset($var1)) echo "var1: $var1 ";
  if (isset($var2)) echo "var2: $var2 ";
  if (isset($var3)) echo "var3: $var3 ";

  echo "<br><b>Radio</b><br>";
  echo "sex: $sex";

  echo "<br><b>Область ввода текста</b><br>";
  echo $t_area;

  echo "<br><b>Список month:</b> $month";
  echo '"<br><b>Список months[]:</b><br>";

  foreach ($months as $key=>$value)
    echo "<br> $key = $value";
}
?>
```

Предположим, что в текстовое поле `txt` мы ввели имя 777, в поле пароля — 555555, выбрали только варианты 1 и 2, установили параметр `sex` в значение male, заполнили текстовую область, в первом списке выбрали Май, а во втором — Июнь и Сентябрь. В результате мы получим такой вывод сценария:

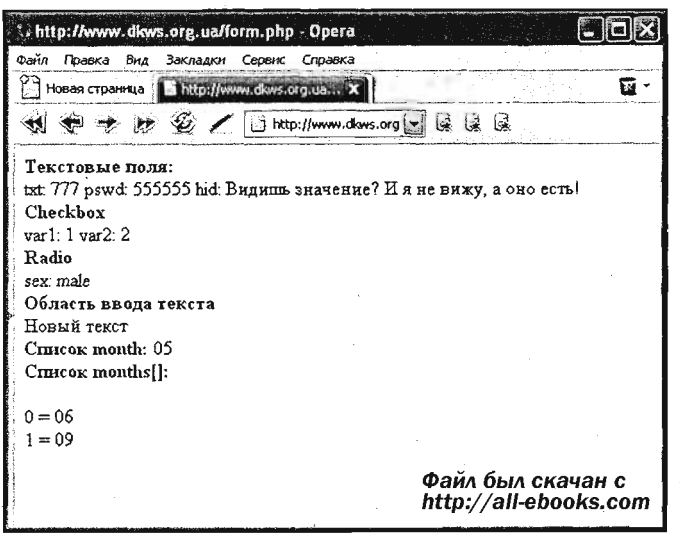



*Рис. 6.8. Результаты работы сценария form.php*

http://all-ebooks.com

Вывод сценария в особых комментариях не нуждается, кроме результатов обработки списков. Как вы видите, передается не строка («Май»), а заданное значение value (05). Если не указывать значение value, то значением параметра станет строка, заключенная в тег option. Также обратите внимание на трансляцию множественного списка: выводится номер элемента (0, 1, 2) и значения (01, 02, 03) элементов массива.

## 6.3. МЕХАНИЗМ COOKIES

### 6.3.1. Принцип работы механизма Cookies

Мне не хочется забивать вам голову техническими подробностями, поэтому я кратко скажу, что такое Cookies и... нет, не с чем их едят, а как с ними нужно работать.

Представим, что перед нами поставлена такая задача: нужно написать индивидуальный счетчик посещений для каждого клиента сайта нашей компании. То есть мы должны показывать пользователю не общее количество посещений сайта, а сколько раз обращался к сайту именно он.

Можно решить эту задачу двумя способами. Первый способ заключается в том, чтобы вести учет по IP-адресу. Для этого нужна база данных, состоящая всего из одной таблицы, структура которой примерно такова:

```
IP-адрес          Посещения
111.111.111.111   23
111.123.215.211   100
```

Когда пользователь заходит на наш сайт, нам нужно определить его IP, найти в базе данных информацию о его посещениях, увеличить счетчик на единицу и вывести его в браузер. Определять IP-адрес клиента мы уже сумеем (переменная `$REMOTE_ADDR`), поэтому нам несложно написать такую программу. Однако при использовании данного способа рано или поздно «всплывут» две проблемы:

- Для каждого IP-адреса нужно будет вести учет в одной таблице, которая, скорее всего, будет большой, а из этого следует, что мы нерационально используем процессорное время и дисковое пространство.



http://all-ebooks.com

- У большинства домашних пользователей IP-адреса динамические. То есть сегодня у него адрес 111.111.111.111, завтра — 111.111.111.118. И попробуй потом определить, кто есть кто. Можно ввести регистрацию с использованием имени пользователя и пароля, но это еще больше увеличит нашу базу данных и создаст неудобства для клиентов.

Ясно, что описанный способ для нашей задачи непригоден. Остается использовать второй способ, который намного легче в реализации. Мы устанавливаем Cookie — переменную, которая будет храниться на диске у пользователя. Эта переменная и будет хранить информацию о посещениях. Она будет считываться сценарием при обращении пользователя к серверу. Выгода очевидна. Во-первых, нам не нужно хранить ненужную информацию о IP-адресах, во-вторых, нам не страшны динамические IP-адреса, поскольку каждый клиент сам хранит данные о своих посещениях.

Теперь понятно, для чего используются Cookies — для хранения небольшого объема данных на стороне клиента. Такими данными могут быть его логин (но ни в коем случае не пароль!), настройки сайта (цветовая гамма, язык) и т. п.

Почему небезопасно хранить пароль в Cookies? Получив от сервера переменную, браузер сохраняет ее на диске в открытом виде, то есть не кодируя ее. Любой, кому не лень, может просмотреть значения этой пере-

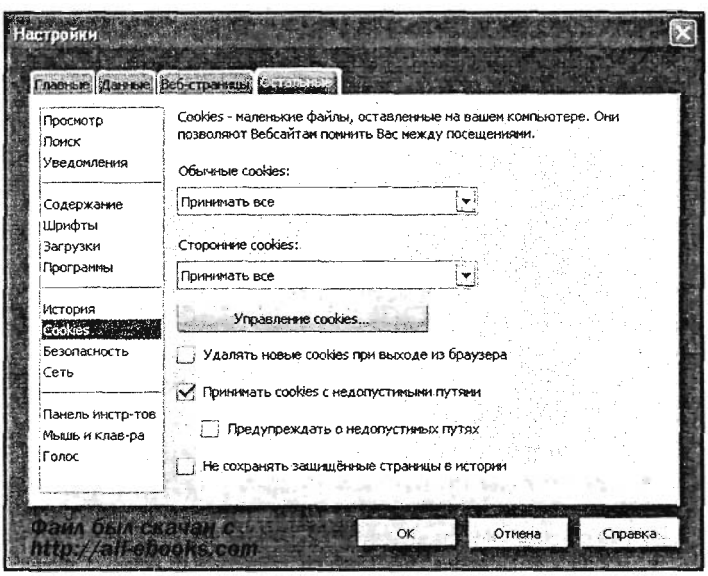



***Рис. 6.9. Управление параметрами Cookies***

http://all-ebooks.com

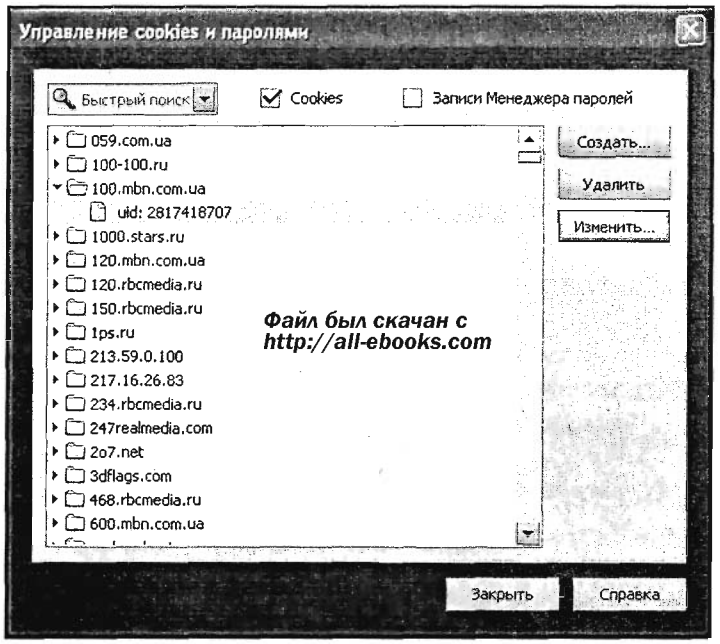



***Рис. 6.10. Установленные Cookies (браузер Opera)***

менной. Кроме того, в некоторых браузерах Cookies вообще сохраняются в обыкновенном текстовом файле, а некоторые даже позволяют просмотреть их. Например, в браузере Opera для просмотра Cookies выполните команду меню Сервис→Настройки→Cookies (рис. 6.9).

В этом разделе вы можете указать, хотите ли вы принимать Cookies или нет. Нажав кнопку Управление cookies, вы можете управлять уже принятыми Cookies (рис. 6.10)

Нажав кнопку Изменить, вы сможете просмотреть установленные Cookies, а также изменить их значения.

Перед тем как начать работу с Cookies, позволю себе пару замечаний относительно выбора браузера. Для отладки ваших сценариев, если они используют Cookies, я рекомендую использовать браузеры Opera или Mozilla, поскольку они позволяют эффективно управлять Cookies. Если эти браузеры вас почему-то не устраивают, используйте Konqueror, который входит в состав любого дистрибутива Linux.

Internet Explorer – не самый удачный выбор браузера для отладки сценариев, использующих Cookies, потому что управлять Cookies он не умеет.



http://all-ebooks.com

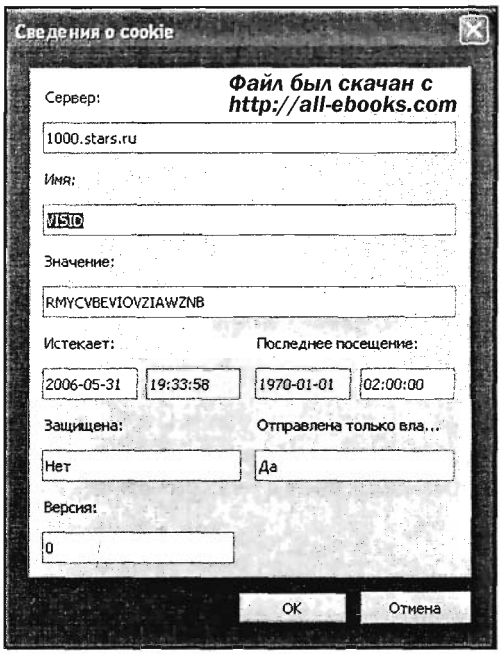



*Рис. 6.11. Редактирование Cookies*

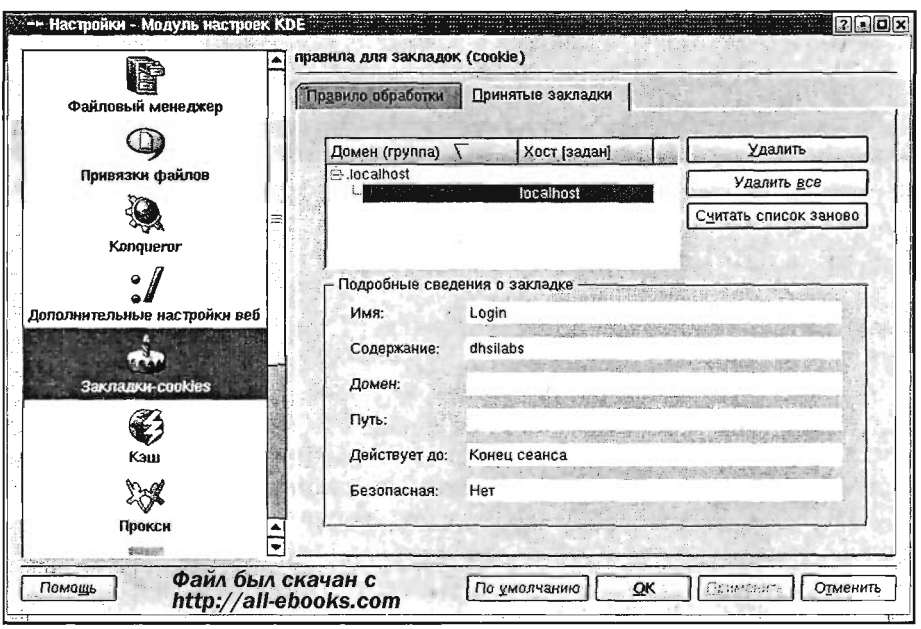



*Рис. 6.12. Окно настройки браузера Konqueror*

http://all-ebooks.com

## 6.3.2. Программирование Cookies

### Установка Cookies

Вот теперь можно приступить к программированию Cookies. Для установки Cookies используется функция `SetCookie()`. В общем случае для этой функции можно указать целых шесть параметров, но обязательным из них является только один — name:

- `name` — задает имя (строка), закрепленное за Cookie.
- `value` — определяет значение переменной (строка).
- `expire` — время жизни переменной (целое число). Если этот параметр не указан, Cookie будет «жить» до конца сессии, то есть до закрытия браузера. Если время указано, то, когда оно истечет, Cookie самоуничтожится.
- `path` — путь к Cookie (строка).
- `domain` — домен (строка). В качестве значения устанавливается имя хоста, установившего Cookie.
- `secure` — передача через HTTPS-соединение.

Обычно используются только три первых параметра. В листинге 6.5 приведены несколько примеров установки Cookies.

**Листинг 6.5. Примеры установки Cookies**

```
<?

// до конца сессии
SetCookie("Test","value");

// время жизни - один час после установки (1 час = 3600 секунд)
SetCookie("My_cookie","One hour",time()+3600);

?>
```

При использовании Cookies нужно иметь в виду, что Cookies должны быть установлены до первого вывода сценария (будь то оператор echo или вывод какой-либо функции). Поэтому желательно устанавливать Cookies в самом начале сценария. Дело в том, что Cookies устанавливаются с помощью определенного заголовка сервера, а если вы выводите что-нибудь, то это означает, что начинается тело документа. В результате Cookies не будут установлены. Вот что вы получите, если установка Cookies невозможна:



http://all-ebooks.com

```
Warning: Cannot add header information - headers already
sent by (output started at /var/www/html/cookie.php:2) in
/var/www/ html/cookie.php on line 4
```

Что делать, если вы получаете это сообщение, а никаких операторов echo до вызова `SetCookie()` в вашем сценарии нет? Убедитесь в том, что маркер начала программы «<?» стоит в самом начале файла. Любые символы, предшествующие «<?», PHP воспринимает как фрагмент HTML-документа. Вывод тела документа прекращает формирование заголовка, а Cookies устанавливаются именно при формировании заголовка документа.

### Чтение значений Cookies

Получить доступ к Cookies достаточно просто. Все они хранятся в переменной окружения `HTTP_COOKIE`. Если мы установили несколько Cookies, то в переменной `HTTP_COOKIE` они будут перечислены через точку с запятой. Например:

```
mycookie1=value1;mycookie2=value2.
```

Для чтения значения Cookies можно обращаться к ним как к обыкновенным переменным:

```
echo $Test;
```

Только не забудьте проверить существование переменной. В заключение приведу пример «индивидуального счетчика» посещений. Код сценария приведен в листинге 6.6.

**Листинг 6.6. Индивидуальный счетчик, реализованный с помощью Cookies**

```
<?

// Если переменная не существует, объявляем ее
if (!isset($Ind_Counter)) $Ind_Counter = 0;

// Увеличиваем счетчик
$Ind_Counter++;

// Устанавливаем Cookie, которая будет жить до переустановки
пользователем
// Windows (или пока он вручную не удалит эту Cookie), то есть очень
долго.
// Если вам интересно, то браузер удалит ее 18 июля 2029 года
//   в 8 часов 49 минут по московскому времени

SetCookie("Ind_Counter", 0x6FFFFFFF);

echo "Вы посетили этот сайт $Ind_Counter раз";
?>
```

http://all-ebooks.com

## 6.4. ДИРЕКТИВА REGISTER_GLOBALS

### 6.4.1. Почему опасно использовать register_globals?

Директива `register_globals` файла конфигурации `php.ini` используется для регистрации параметров, переданных программе, как обычных локальных переменных. Например, при передаче сценарию параметра `param` (`script.php?param=value`) вы его сразу можете использовать в вашей программе как обычную переменную:

```
echo $param;
```

Если же директива выключена (по умолчанию), то доступ к параметрам, переданным сценарию, осуществляется через массив `GLOBALS`:

```
echo $GLOBALS['param'];
```

Но такой способ несколько неудобен, поэтому очень часто директиву `register_globals` приходится включать самостоятельно. Возникает вопрос: почему директива `register_globals`, существенно облегчающая жизнь программистам, по умолчанию выключена? Оказывается, из соображений безопасности: ее использование опасно прежде всего для хостинга самого пользователя, а также для сервера хостинг-провайдера.

Лучше всего пояснить это на примере. Предположим, вашему проекту нужна система аутентификации, проверяющая имя пользователя и пароль. Если имя пользователя и пароль правильны, система предоставляет доступ, например, к панели управления сайтом. Рассмотрим небольшой пример кода сценария `login.php` (пусть он будет называться именно так):

```
...
if (user_correct($user, $passwd) == "1") $auth = $true;

if ($auth==true) {
 echo "предоставляем доступ";
}
else die("Access denied\n");

...
```

Из этого фрагмента видно, что сценарию с помощью формы ввода имени пользователя и пароля передаются два параметра: `user` и `passwd`, содержащие соответственно имя пользователя и пароль. Затем эти два параметра должны быть переданы функции `user_correct()`, проверяющей правильность имени и пароля. Если все в норме, функция возвращает 1, в противном случае 0.



http://all-ebooks.com

Оператор if сравнивает возвращенное функцией значение с 1: если оба значения равны, то переменная `$auth` устанавливается в true, в противном случае ничего не делается. То есть, если `$auth = true`, то пользователь успешно прошел аутентификацию.

Второй оператор if сравнивает значение переменной `$auth` со значением true. Если они не равны, то выводится всем известное сообщение «Access denied" (В доступе отказано). Если же `$auth = true`, то выводится строка «предоставляем доступ».

Теперь проведем небольшой эксперимент – побудем в роли злоумышленника. Для большей определенности приведу полный исходный код экспериментального сценария.

```
<?
function user_correct($user, $passwd) { return 0; }

if (user_correct($user, $passwd) == "1") $auth = $true;

if ($auth==true) {
 echo "предоставляем доступ";
}
else die("Access denied\n");

?>
```

Обратите внимание: функция `user_correct` всегда возвращает значение 0. То есть по идее мы никогда не должны увидеть строку «предоставляем доступ». Попробуйте передать любое имя пользователя и пароль сценарию:

```
http://localhost/login.php?user=den&passwd=1
```

Какое бы имя пользователя и пароль вы ни указали, вы увидите строку «Access denied». А теперь попробуйте ввести следующий URL:

```
http://localhost/login.php?user=den&passwd=1&auth=true
```

Что мы получаем в ответ? Правильно, «предоставляем доступ» (рис. 6.13).

Почему это произошло? Да потому, что PHP регистрирует все переменные массива GLOBALS (переменные HTTP-сервера и CGI-переменные, то есть параметры сценария). После этого уже невозможно понять, была ли эта переменная объявлена программистом или это CGI-параметры, переданные пользователем. Из-за этого злоумышленник может переопределять значения переменных сценария. Если же директива `register_globals` выключена и доступ к параметрам, переданным сценарию, осуществляется через массив GLOBALS, такого никогда не произойдет, поскольку идет четкое разделение между CGI-переменными и переменными программы.

http://all-ebooks.com

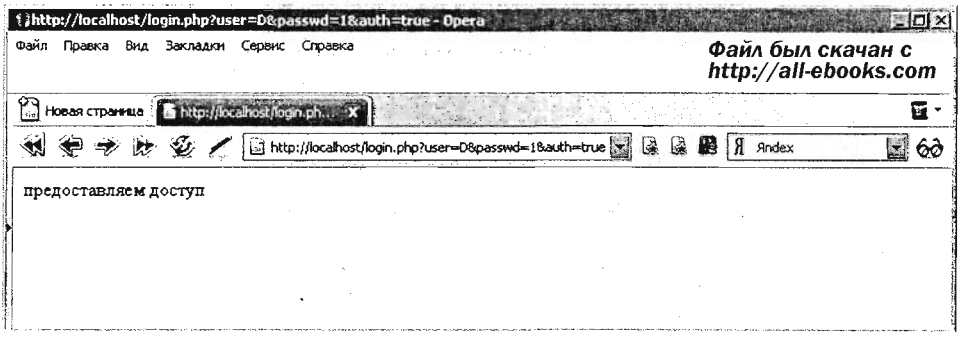



***Рис. 6.13. Сценарий взломан***

Обратиться к параметрам `user` и `passwd` через массив `$GLOBALS` можно так:

```
$GLOBALS['user']
$GLOBALS['passwd']
```

Даже если злоумышленник укажет параметр `auth`, ничего страшного не случится: сценарий же обращается к переменной `$auth`, а не к элементу массива `$GLOBALS['auth']`. Но это только в случае, если `register_globals` отключена.

Спрашивается, а откуда злоумышленник будет знать имя моей переменной? Не секрет, что большинство серьезных PHP-проектов не пишутся с нуля, а используются уже написанные кем-то сценарии. Вы же не пишете сами форум? Вы скачали форум с какого-то сайта. Злоумышленник тоже может скачать этот же форум и узнать имена всех переменных, которые его интересуют.

Конечно, в приведенном сценарии программист допустил грубейшую ошибку – он не определил значение переменной `$auth` в случае, когда функция `user_correct()` возвращает 0. Если бы его сценарий выглядел так:

```
<?
function user_correct($user, $passwd) { return 0; }

if (user_correct($user, $passwd) == "1") $auth = $true;
else $auth=false;

if ($auth==true) {
 echo "предоставляем доступ";
}
else die("Access denied\n");
?>
```



http://all-ebooks.com

то наша попытка взлома бы не удалась. Но хостинг-провайдеры не могут быть уверенными в уровне квалификации каждого программиста, которому предоставляется хостинг, поэтому директиву `register_globals` часто отключают.

## 6.4.2. Массив $GLOBALS

Кроме переменных окружения, массив `$GLOBALS` содержит значения всех переменных вашей программы. Например, пусть в нашей программе объявлена переменная

```
$Test = "String";
```

Тогда получить доступ к значению этой переменной можно через массив `$GLOBALS`:

```
echo "$GLOBALS[Test]";
```

В следующей таблице описаны наиболее полезные переменные массива `$GLOBALS`

**Таблица 6.1.** *Переменные окружения*

| Переменная | Значение |
|---|---|
| PWD | Текущий каталог |
| HOSTNAME | Имя узла |
| QTDIR | Каталог, в который установлена библиотека Qt (в Linux) |
| USER | Имя пользователя |
| MAIL | Почтовый ящик пользователя |
| HOME | Домашний каталог пользователя |
| SHELL | Командная оболочка |
| LANG | Язык системы (для каждого пользователя может быть разным) |
| PATH | Путь для поиска программ |
| HTTP_POST_VARS | Массив, содержащий параметры, переданные сценарию методом POST |
| HTTP_GET_VARS | Массив, содержащий GET-параметры |
| HTTP_ENV_VARS | Массив, содержащий переменные окружения |
| HTTP_COOKIE_VARS | Информация о Cookies |
| GLOBALS | А это сам массив GLOBALS. Так как массив GLOBALS – это глобальная переменная типа array, он также присутствует в этом списке |

Вот что вам нужно знать об этом массиве:

- Данный массив является глобальным, то есть он доступен как из основной программы, так и из любой функции.
- Добавление элемента в массив равноценно добавлению глобальной переменной.

Над массивом `$GLOBALS` можно выполнять все операции, которые можно выполнять над массивами, кроме трех:

- Нельзя присвоить массив какой-то переменной с помощью оператора = .
- Нельзя передать массив функции (да это и незачем, ведь он и так доступен).
- Нельзя удалить массив с помощью функции `UnSet()` – я пробовал, не вышло.



http://all-ebooks.com

# ГЛАВА 7.

## МАССИВЫ И СПИСКИ

http://all-ebooks.com

## 7.1. ПОНЯТИЯ МАССИВА И СПИСКА

Массив — это упорядоченный набор данных. Каждый элемент массива имеет индекс или ключ. Индекс (ключ) служит для однозначной идентификации элемента внутри массива. В одном массиве не может быть двух элементов с одинаковыми индексами.

Работать без массива не очень удобно, думаю, вы и сами это понимаете. Существует ряд задач, где без использования массива просто не обойтись. Примером может послужить простой список друзей. Не будете же вы создавать 20 переменных, содержащих имена ваших друзей? Гораздо рациональнее создать один массив, содержащий 20 элементов.

Если вы программировали на C или Pascal, то знаете, что длина массива ограничена. При этом, если вы наперед не знаете, сколько элементов будет содержать ваш массив, значит, вам нужно использовать динамические структуры (список, стек, очередь).

В языке PHP все гораздо проще: длину массива не нужно указывать при объявлении массива, она будет автоматически увеличена при добавлении нового элемента в массив. Все это становится возможным только благодаря тому, что PHP — интерпретатор, а не компилятор.

Очень важной особенностью PHP является то, что он, в отличие от других языков, позволяет создавать массивы любой сложности непосредственно в теле программы (сценария). Это бывает очень полезным, когда требуется создать массив, на основе некоторых, заранее неизвестных параметров.



http://all-ebooks.com

Рассмотрим простейший способ инициализации массива:

```
$People[0]="Коля";
$People[1]="Витя";
$People[2]="Дима";
$People[3]="Марк";
```

Интерпретатор PHP узнает о том, что создается массив, по квадратным скобкам. В квадратных скобках указывается индекс (ключ) отдельного элемента массива. В приведенном примере мы объявили массив `$People`, состоящий из четырех элементов. Однако в дальнейшем нам никто не помешает инициализировать еще один (или сколько нужно) новый элемент массива. В PHP размер массива не задается явно, как в других языках программирования.

Обратиться к элементу массива можно по имени массива и индексу элемента внутри массива:

```
echo    $People[3];
```

Зная количество элементов массива, мы можем вывести все элементы массива с помощью цикла:

```
for($i=0; $i<количество; $i++) echo "$People[$i] <br>";
```

Если мы не знаем точное количество элементов массива, можно использовать функцию `count()`. Эта стандартная функция возвращает размер массива, то есть количество элементов в нем. Соответственно, с использованием функции `count()` вышеприведенный цикл выглядит следующим образом:

```
for($i=0; $i<count($People); $i++) echo "$People[$i] <br>";
```

Массивы, индексами которых являются числа, начинающиеся с нуля без пропусков, будем называть списками. С технической точки зрения разницы между массивами и списками нет.

Не знаю, как вам, но мне удобнее создавать списки без указания индекса — пусть это сделает за меня PHP. Ведь я могу ненароком указать уже существующий индекс. Кроме того, такое автоматическое создание массивов удобно при записи текстового файла в массив для последующей обработки.

Вот пример автоматического создания массива:

```
$People[]="Коля";
$People[]="Витя";
$People[]="Дима";
```

При этом PHP начнет нумерацию с наименьшего незанятого значения индекса. В частности, если переменная `$People` только сейчас инициализируется, то нумерация будет начата с нуля. В дальнейшем с каждым новым элементом массива индекс будет увеличиваться на единицу.

Как видите, все просто и удобно — не нужно указывать индексы. Однако это как раз тот случай, когда мы не знаем точного числа элементов в массиве и должны использовать функцию `count()`.

Рассмотрим небольшой пример: чтение текстового файла в массив. Забегая вперед, скажу, что функция `fopen()` используется для открытия файла, функция `feof()` истинна, если достигнут конец файла, а функция `fgets()` читает строку из файла.

При работе с файлами мы не знаем, сколько строк будет содержать конкретный файл. Если бы мы программировали на C или Pascal, нам нужно было бы создать динамический список. У начинающих программистов, не знакомых с принципами работы с динамической памятью, это могло бы вызвать затруднения. В PHP все просто: вы можете добавить сколько угодно элементов в массив, не указывая их номера. Новые элементы массива будут добавлены в конец массива.

```
// Открываем файл
$f = fopen("file.txt","rt") or die("Ошибка");
// Читаем построчно
while(!feof($f))
{
// Заносим новый элемент (строку, прочитанную из файла)
// в конец массива
$File[]= fgets($f,50);
     }
// Закрываем файл
 fclose ($f);
```

## 7.2. АССОЦИАТИВНЫЕ МАССИВЫ

Мы договорились, что массивы, индексами которых являются числа, начинающиеся с нуля, будем называть списками. Однако в PHP индексом массива может быть и строка. Причем на эту строку не накладывается никаких ограничений: она может содержать пробелы, специальные символы, также нет ограничений на длину строки.
Массивы, индексами которых являются строки, называются ассоциативными. Индексы ассоциативных массивов называются ключами. Пример ассоциативного массива:



http://all-ebooks.com

```
$People["Иванов"] = "Иван Иванович";
$People["Сидоров"] = "Николай Петрович";
$People["Петров"] = "Петр Александрович";
```

Доступ к элементам ассоциативных массивов осуществляется так же, как и к элементам обыкновенных массивов, и называется доступом по ключу:

```
echo $People["Иванов"];
```

## 7.3. ИНСТРУКЦИИ LIST()

Предположим, что у нас есть массив, состоящий из трех элементов:

```
$People[0] = "Иван";
$People[1] = "Николай";
$People[2] = "Петр";
```

Теперь допустим, что в какой-то момент нам необходимо присвоить значения трех элементов массива трем переменным: `$Ivanov`, `$Sidorov`, `$Petrov` соответственно. Это можно сделать так:

```
$Ivanov = $People[0];
$Sidorov = $People[1];
$Petrov = $People[2];
```

Согласитесь, что если массив большой, то такой подход не очень удобен. Существует более рациональное решение — использовать инструкцию `list()`, которая специально для этого предназначена:

```
list($Ivanov,$Sidorov,$Petrov) = $People;
```

Если же нам нужны имена только Сидорова и Петрова, первый параметр можно пропустить:

```
list(,$Sidorov,$Petrov) = $People;
```

## 7.4. ИНСТРУКЦИЯ ARRAY() И СОЗДАНИЕ МНОГОМЕРНЫХ МАССИВОВ

Инструкция `array()` служит специально для создания массивов. Она позволяет создавать пустые массивы, чего никак нельзя сделать рассмотренными ранее механизмами. Методы использования инструкции `array()` рассмотрим на следующих примерах:

http://all-ebooks.com

```
$A = array(); // создает пустой массив
  $A  =  array("Иванов"=>"Иван","Петров"=>"Петр");  //  создает
//ассоциативный
// массив с двумя элементами

$A = array("Иванов","Петров");  // создает список с двумя
  // элементами. Индексы нумеруются с нуля.
```

Инструкцию `array()` удобно использовать для создания многомерных массивов. Предположим, что мы хотим создать массив, элементами которого будут ассоциативные массивы с ключами `name`, `sex` и `email`.

Создать такой массив можно с помощью следующих операторов:

```
$A["Ivanov"] =
array("name"=>"Иванов И.",  "sex"=>"M", "email" => "ivanov@mail.ru");
$A["Petrov"]  =
array("name"=>"Петров И.", "sex"=>"M", "email" => "petrov@mail.ru");
$A["Sidorova"]    =
array("name"=>"Сидорова Н", "sex"=>"F", "email" => "sidorova@mail.ru");
```

То же самое можно записать в следующем виде:

```
$A = array (
   "Ivanov" =
    array("name"=>"Иванов И.", "sex"=>"M", "email" =>  "ivanov@mail.ru"),
   "Petrov" =
   array("name"=>"Петров И.", "sex"=>"M", "email" => "petrov@mail.ru"),
   "Sidorova" =
  array("name"=>"Сидорова Н", "sex"=>"F", "email" => "sidorova@mail,ru")
);
```

Добраться до элементов массива можно так:

```
echo $A["Ivanov"]["name"];    // выведет Иванов И.
echo $A["Ivanov"]["email"];  // выведет ivanov@ivanov.org
```

Многомерные массивы в PHP больше положи на записи (record) в языке Pascal или структуры в языке C, чем на многомерные массивы в обычном понимании.



http://all-ebooks.com

## 7.5. ОПЕРАЦИИ НАД МАССИВАМИ

Мы уже познакомились с основной операцией над массивами — доступ по ключу, поэтому рассматривать ее сейчас я не вижу смысла. Сейчас мы рассмотрим три операции с массивами:

- Удаление.
- Слияние.
- Перебор.

### 7.5.1. Удаление массивов

Для удаления массива, как и любой другой переменной, используется уже знакомая нам функция `UnSet()`:

```
UnSet($People);
```

Обычно удаление может потребоваться для очень большого массива, чтобы освободить используемую им память.

### 7.5.2. Слияние массивов

#### Стандартное слияние

Слияние (конкатенация) — это операция создания массива, состоящего из элементов нескольких других массивов. Слияние массивов — это очень опасная операция, поэтому старайтесь избегать ее. Результат слияния подчиняется своей логике, забыв о которой можно потерять данные. Слияние реализуется при помощи оператора +.

Итак, пусть у нас есть два массива:

```
  $A = array("1"=>"первый","2"=>"второй");
$B = array("3"=>"третий","4"=>"четвертый");
```

Теперь давайте сольем эти два массива в один массив `$C`:

```
$C = $A + $B;
```

Сразу нужно сказать, что оператор «+» для массивов не коммутативен, а это значит, что `$A + $B` не равно `$B + $A`. В результате слияния `$A + $B` мы получим:

```
"1"=>"первый","2"=>"второй","3"=>"третий","4"=>"четвертый"
  // именно в таком порядке
```

А в результате `$B + $A` получим такой массив:

```
"3"=>"третий","4"=>"четвертый","1"=>"первый","2"=>"второй"
```

Особенно внимательным нужно быть при слиянии списков. Предположим, у нас опять есть два массива:

```
$A = array(10,11,12);
$B = array(15,17,18);
```

В результате слияния `$C = $A + $B` мы получим следующий массив:

```
10,11,12
```

Неожиданный результат, не правда ли? Скорее всего, вы ожидали получить массив `array` (10, 11, 12, 15, 17, 18). Однако в PHP на этот счет свои правила. При слиянии массивов, в которых содержатся элементы с одинаковыми индексами (ключами), в результирующем массиве останется элемент первого массива, причем на том же самом месте. А в приведенном примере элементы массива `$A` имели индексы 1, 2, 3 и элементы массива `$B` имели индексы 1, 2, 3. А поскольку массив `$A` при сложении стоит первым, то его элементы и являются результирующими. Можете убедиться в этом еще раз следующим образом:

```
$A = array(10,11,12);
$B = array(15,17,18);
$A = $A + $B;              // или $A+= $B;
```

В этом примере элементы массива `$A` не изменятся. Изменить элементы массива можно только при помощи прямого перебора (см. следующий раздел).

### Слияние с помощью функции array_merge()

Функция `array_merge()` предназначена для слияния двух массивов. Эта функция устраняет все недостатки оператора +. Она соединяет массивы, переданные ей в качестве аргументов, в один большой массив. Если в соединяемых массивах встречаются одинаковые ключи (ключ=>значение), то в результирующий массив помещается пара ключ=>значение из массива, который стоит правее в списке аргументов функции `array_merge()`

Однако это правило не затрагивает списки, то есть массивы с числовыми индексами. В этом случае элементы с повторяющимся числовым индексом помещаются в конец результирующего массива в любом случае.



http://all-ebooks.com

Проиллюстрируем сказанное примером:

```
$A = array(10,11,12);
$B = array(15,17,18);
$C = $A + $B;                    // $C = array (10,11,12);
$C = array_merge($A,$B);    // $C = array(10,11,12,15,17,18);
```

Таким образом, функцию `array_merge()` очень удобно использовать для *слияния списков*.

### 7.5.3. Перебор массива

Для перебора массивов в PHP можно использовать два метода:

- косвенный перебор;
- прямой перебор.

#### Косвенный перебор

Принцип косвенного перебора заключается в том, что сначала вычисляется очередной индекс (ключ), а затем по этому индексу косвенно находится значение элемента массива. Например:

```
$People[0]="Коля";
$People[1]="Витя";
$People[2]="Дима";
for($i=0; $i<count($People); $i++) echo "$People[$i] <br>";
```

Это мы с вами перебрали список, индексами которого являются целые числа. Для этого мы использовали цикл `for`, в котором перебирали индексы, последовательно увеличивая их на 1, и по индексам получали элементы массива.

Что делать, если нам нужно перебрать не список, а ассоциативный массив? Ведь в ассоциативном массиве индексом (ключом) является не число, а строка:

```
$People["Иванов"]  = "Иван";
$People["Сидоров"] = "Николай";
$People["Петров"]  = "Петр";
```

Выполнить перебор такого массива поможет следующий цикл:

```
for(Reset($People); ($k = key($People)); Next($People))
    echo "$k = $People[$k] <br>";
```

Здесь функция `Reset()` устанавливает текущий элемент массива `$People[]` на первую позицию, как бы «перематывает» массив на начало. Функция `key()` возвращает ключ текущего элемента. Если в массиве нет больше элементов, то эта функция возвратит 0, и цикл прекратит свою работу. Функция `Next()` перемещает текущий элемент на одну позицию вперед, то есть переходит к следующему элементу массива.

В теле цикла можно также использовать функцию `current()`, которая возвращает значение текущего элемента. С использованием этой функции вышеприведенный цикл можно переписать так:

```
for(Reset($People); ($k = key($People)); Next($People))
  echo "$k =".current($People)."<br>";
```

До сих пор мы перебирали ассоциативный массив с начала в конец. В PHP имеется возможность перебора такого массива и с конца. Для этого вам нужно лишь использовать функции `End()` и `Prev()` вместо функций `Reset()` и `Next()`. Первая из них устанавливает текущий элемент в конец массива, а вторая перемещает текущий элемент на одну позицию назад, то есть переходит к предыдущему элементу:

```
for(End($People); ($k = key($People)); Prev($People))
     echo "$k = {$People[$k]} <br>";
```

### Прямой перебор

Принцип прямого перебора заключается в том, чтобы при каждой итерации цикла одновременно получать и ключ, и значение текущего элемента массива. Для прямого перебора в PHP, начиная с версии 4, используется цикл `foreach`, который мы уже неоднократно использовали в этой книге:

```
foreach ($People as $key => $value) echo "$key = $value";
```

### Недостатки косвенного перебора

Косвенный перебор очень прост, а код, написанный с использованием косвенного перебора, легко читается. Даже если вы вообще не знаете PHP, вы понимаете, что здесь к чему. Однако косвенный перебор имеет существенные недостатки. Один из недостатков должен сразу броситься вам глаза: если в массиве встретится элемент с индексом 0, то после того, как он будет передан в цикл функцией `key()`, цикл прекратит свою работу.

И еще один недостаток: вы не можете использовать косвенный перебор во вложенных циклах, так как функция `Next()` второго цикла изменит положение текущего элемента в первом цикле.

Прямой перебор лишен этих недостатков.



http://all-ebooks.com

## 7.6. СОРТИРОВКА МАССИВОВ

Для сортировки массивов в PHP предназначено много различных функций, но мы сейчас рассмотрим только основные из них:

- `asort()` и `arsort()` — сортировка по значениям;
- `ksort()` — сортировка по ключам;
- `sort()` — сортировка списка;
- `array_reverce()` — «переворачивание» массива;
- `shuffle()` — перемешивание списка.

### 7.6.1. Сортировка массива по значению. Функции asort() и arsort()

Функция `asort()` выполняет сортировку массива по значениям элементов, располагая элементы массива таким образом, чтобы их значения шли в алфавитном порядке (если это строки) или в возрастающем порядке (если это числа). Функция `arsort()` выполняет сортировку по значениям в обратном порядке.

Предположим, что у нас есть такой массив:

```
$FIO = array("Sidorov"=>"Nikolay","Ivanov"=>"Alexandr","Petrov"=>
"Petr");
```

После выполнения функции `asort($FIO)` наш массив будет преобразован следующим образом:

```
array("Ivanov"=>"Alexandr","Sidorov"=>"Nikolay","Petrov"=>"Petr");
```

### 7.6.2. Сортировка массивов по ключам. Функции ksort() и krsort()

Для сортировки по ключам используется функция `ksort()`. При этом сортировка осуществляется по таким же принципам, что и функцией `asort()`, только не по значению, а по ключам. Если передать ей наш массив `$FIO`, то мы получим такой результат:

```
array("Ivanov"=>"Alexandr","Petrov"=>"Petr","Sidorov"=>"Nikolay");
```

Сравните полученный результат с результатом функции `asort()`. Для сортировки ключей в обратном порядке служит функция `krsort()`.

http://all-ebooks.com

### 7.6.3. Сортировка списка

Все четыре рассмотренные функции не разрывают пары **ключ=>значение**. Это очень важно. Например, у нас есть небольшой список:

```
$People = array ("Ivanov"=>"555-6677",
                 "Sidorov"=>"345-4567","Petrov"=>"777-5653");
```

После сортировки массива по ключам наш список будет выглядеть так:

```
$People = array("Ivanov"=>"555-6677",
                "Petrov"=>"777-5653","Sidorov"=>"345-4567");
```

Обратите внимание: пары **ключ=>значение** не разорваны, то есть у Петрова по-прежнему номер телефона 777-5653, а у Сидорова — 345-4567. Если бы пары были разорваны, номер телефона у Петрова был 345-4567, а у Сидорова — 777-5653, то есть наоборот.

Если вам нужно отсортировать список, то удобнее всего использовать функцию `sort()`. Данная функция разрывает пары **ключ=>значение**, поэтому ее нужно использовать только для списков.

Пусть у нас есть список:

```
$List = array("Name","Address","Email");
```

Если вывести список в виде ключ = >значение, то мы получим следующий результат:

```
0 => Name
1 => Address
2 => Email
```

После сортировки `sort($List)` у нас получится следующий список:

```
0 => Address
1 => Email
2 => Name
```

Изначальная связь между парами **ключ=>значение** была разорвана. Возникнет как раз та ситуация, которой мы хотели избежать в примере с номерами телефонов.

### 7.6.4. Переворачивание массива. Функция array_reverse()

Для переворачивания массива используется функция `array_reverse()`. Эта функция возвращает массив, элементы которого следуют в обратном



http://all-ebooks.com

порядке относительно исходного массива. Например, чтобы отсортировать массив в обратном порядке, можно использовать функцию `arsort()`, а можно упорядочить массив в прямом порядке функцией `asort()` и перевернуть его функцией `array_reverse()`.

**Примечание.**

Если вам нужно отсортировать массив в обратном алфавитном порядке, я рекомендую использовать функцию arsort() — она работает быстрее, чем последовательность функций asort() и array_reverse().

### 7.6.5. Перемешивание списков. Функция shuffle()

Функция `shuffle()`, так сказать, «перемешивает» список, то есть значения результирующего списка будут размещены в случайном порядке. Данная функция полностью изменяет массив, поэтому ее можно использовать только для списков.

Перед вызовом этой функции нужно инициализировать генератор случайных чисел, иначе при каждом новом вызове функции `shuffle()` результат перемешивания будет один и тот же (при перемешивании одного и того же массива). Для инициализации генератора случайных чисел используется функция `mt_srand()`:

```
mt_srand(time()* 100000);
shuffle($Arr);
```

**Примечание.**

Для генерирования случайных чисел рекомендуется использовать функции mt_rand() и mt_srand() вместо стандартных rand() и srand(), поскольку первые обеспечивают лучшее качество генерируемых случайных чисел.

### 7.6.6. Пользовательская сортировка массивов. Функции uksort(), uasort() и usort()

Иногда нужно отсортировать массив не в алфавитном порядке, а по более сложному критерию. Для этого используются функции `uksort()`, `uasort()` и `usort()` — пользовательские функции сортировки массивов. Каждой функции нужно передать два параметра:

- *массив, который нужно отсортировать;*
- функцию сравнения, которая определит, какой из элементов массива меньше, а какой больше согласно нашему критерию.

http://all-ebooks.com

Функции сравнения нужно передать два параметра — два элемента массива (`$el_1` и `$el_2`). В свою очередь функция должна возвратить одно из трех значений:

```
-1, если $el_1<$el_2;
0, если $el_1=$el_2;
1, если $el_1<$el_2.
```

Предположим, что у нас есть список файлов и каталогов `$FilesNDirs`. Нам нужно вывести сначала каталоги, а потом файлы. Сравнение будет производиться функцией `cmp()`, которую мы сами и напишем (листинг 7.1). Подробнее о создании и использовании пользовательских функций вы можете прочитать в гл. 8.

**Листинг 7.1. Функция сравнения cmp()**

```
function cmp($el_1,$el_2)
{
// Сначала каталог, а потом - файл
if (is_dir($el_1) && !is_dir($el_2 )) return -1;
if (!is_dir($el_1) && is_dir($el_2 )) return 1;

// Сравниваем по алфавиту
if($el_1<$el_2) return -1;
  elseif($el_1>$el_2) return 1;
    else return 0;
}
```

Чтобы отсортировать список `$FilesNDirs`, осталось вызвать функцию `uasort()`, передав ей нашу функцию сравнения:

```
uasort($FilesNDirs,"cmp");
```

> **Примечание.**
>
> В данном случае мы используем uasort() потому, что у нас простой список, а не ассоциативный массив. В последнем случае нужно было использовать функцию uksort().

Функция `uasort()` аналогична функции `uksort()`, только она сортирует не ключи, как `uksort`, а значения. Обе функции — `uksort()` и `uasort()` — сохраняют пары ключ=>значение.

Функция `usort()` похожа на функцию `sort()`, но критерий сортировки задается пользовательской функцией, которая передается в качестве второго параметра. Еще одна важная особенность этой функции заключается в том, что она не сохраняет пару ключ=>значение, поэтому пригодна лишь для сортировки списков, но ее ни в коем случае не нужно использовать для сортировки ассоциативных массивов.



http://all-ebooks.com

## 7.7. ПОЛУЧЕНИЕ ЧАСТИ МАССИВА

Для получения части массива используется функция `array_slice()`. Ей нужно передать три параметра:

- имя исходного массива, из которого будут выбираться элементы;
- смещение от начала массива, то есть сколько элементов от начала массива надо пропустить перед началом выборки;
- длина выборки, то есть количество элементов, которое нужно выбрать из исходного массива (необязательный параметр).

Эта функция возвращает часть указанного массива, начиная с указанного смещения от начала массива. Выборка элементов осуществляется подряд. Если длина выборки не указана, то выборка элементов осуществляется до конца исходного массива.

Вот небольшие примеры:

```
$A = array(xox, ror, tot, pop, mom); // задаем исходный массив
$B = array_slice($A,2);           // получаем массив $B,
    // в котором содержатся
    // все элементы массива $A,
    // начиная с третьей позиции
    // (т. к. два надо пропустить). Таким образом,
    // массив $B  будет содержать
    // элементы  tot, pop, mom.
$B = array_slice($A,0,2);         // Получаем массив $B, в котором
содержатся два
    // элемента  массива $A,
    // начиная с первого
    // (т. к. надо пропустить 0 элементов).
    // Таким образом, массив $B будет
    // содержать элементы  xox, ror.
```

Параметры смещение и длина могут быть отрицательными. В этом случае отсчет производится от конца массива:

```
$B = array_slice($A,2,-1);     // получаем массив $B,
// состоящий из элементов tot, pop.
```

Функция `array_splice()` похожа на функцию `array_slice()`. Ей нужно передать еще один дополнительный параметр — массив `$replace`. Функция `array_splice()` удаляет из массива выборку указанной длины начиная с указанного смещения и заменяет её элементами массива `$replace`, если он передан в качестве параметра. Вместо массива `$replace` можно передать одну строковую константу. Пример:

```
$A = array("Audi","Mercedes","Toyota","Honda","Renault");
array_splice($A,1, count($A),  "BMW");
// $A = array("Audi","BMW");
```

## 7.8. ДОБАВЛЕНИЕ И УДАЛЕНИЕ ЭЛЕМЕНТОВ МАССИВА

### 7.8.1. Добавление в конец массива

Ранее мы с вами освоили простейший способ добавления новых элементов в массив, осуществляемый с помощью оператора []. Однако в последних версиях PHP (начиная с 4) предусмотрена функция `array_push()`, которая добавляет элементы в конец массива так же, как и оператор []. Например, следующие два оператора аналогичны:

```
$A[] = 5;
array_push($A,5);
```

У вас может возникнуть естественный вопрос: зачем нужны два одинаковых способа делать одно и то же? Ответ заключается в удобстве. Если необходимо добавить один элемент в массив (как в приведенном выше примере), то удобнее использовать оператор []. Использование же функции `array_push()` оправдывает себя тогда, когда нам нужно добавить в конец массива несколько новых элементов. Например:

```
$A[] = 5;
$A[] = 7;
$A[] = 9;
$A[] = 10;
$A[] = 15;
```

Вместо этого можно записать:

```
array_push ($A, 5,7,9,10,15);
```

**Примечание.**

Функция array_push(), помимо всего прочего, возвращает новый размер массива. То есть если, помимо того, чтобы добавить новые элементы в массив, вам нужно узнать и его новую длину, то можно записать $D = array_push($A, 5, 7, 9, 10, 15). Таким образом, новая длина массива будет присвоена переменной $D.

### 7.8.2. Удаление последнего элемента списка

Функция `array_pop()` противоположна функции `array_push()`. Она «выталкивает» вершину списка (последний элемент списка). Другими словами, функция `array_pop()` возвращает значение последнего элемента массива и удаляет его из списка.



http://all-ebooks.com

Функции array_pop() нужно передать всего один параметр — имя массива (списка):

```
$A = array (xox, ror, tot, pop, mom);
$T=array_pop($A);          // переменной $T присвоили
// значение mom, а массив $A
// лишился этого последнего
// элемента и теперь состоит из
// элементов xox, ror, tot, pop.
array_pop($A);       // массив $A еще раз лишился
// последнего элемента и теперь
// состоит из элементов xox, ror, tot.
```

**Примечание.**

С помощью функций array_push() и array_pop() мы можем строить структуры LIFO (Last In First Out — *«последний вошел — первый вышел»*), *напоминающие стек*. Структуры типа стек широко применяются в программировании. Классическое применение стека — обратная польская запись, которая используется для разбора арифметических выражений и вычисления их значения.

### 7.8.3. Добавление в начало списка. Удаление первого элемента списка

Функции array_unshift() и array_shift() похожи на функции array_push() и array_pop(),только работают не с концом списка, а с его началом. Функция array_unshift() добавляет элементы в начало списка, а функция array_shift() возвращает значение первого элемента списка и удаляет его из списка (удаляет «голову» списка):

```
$A = array(3,4,5);
array_unshift($A,1,2);    // получаем $A = array(1,2,3,4,5)
array_shift($A)              // получаем $A = array(2,3,4,5);
```

**Примечание**

Функции array_unshift() и array_shift() полезно использовать, если вы хотите построить очередь типа FIFO (First In First Out — «первый вошел — первый вышел»). Очередь необходима для последовательной обработки элементов массива. Стек и очередь в языках C и Pascal являются основными динамическими структурами. Поскольку длина массивов в PHP ограничена свободными системными ресурсами, потребность в стеке и очереди исчезает. Функции для организации стека и очереди были написаны исключительно для удобства программиста — вдруг вы, программируя на C, привыкли работать именно со стеком или с очередью?

Функция array_unique() предназначена для извлечения уникальных значений массива вместе с ключами этих значений. Вот пример применения этой функции:

```
$Arr = array("a"=>"AAA","b"="BBB","c","d","e"="BBB", "d");
$Unique_Arr = array_unique($Arr);
// $Unique_Arr = array("a"=>"AAA","b"="BBB","c","d");
```

## 7.9. ПЕРЕМЕННЫЕ И МАССИВЫ. ВЗАИМНЫЕ ПЕРЕХОДЫ

### 7.9.1. Упаковка переменных в массив. Функция compact()

Функция `compact()` «упаковывает» переменные и их значения в массив. В результате получается ассоциативный массив, ключами которого являются имена переменных, а значениями — значения переменных.

Рассмотрим сказанное на примере. Предположим, что у нас есть две переменные:

```
$Name = "Вася";
$Email = "vasya@vvv.org";
```

Функция `compact("Name","Email")` упакует переменные в один массив:

```
$A = compact("Name","Email");
//получаем $A =array("Name"=>"Вася","Email" = "vasya@vvv.org");
```

Параметры функции `compact()` описаны как `mixed`, потому что параметры могут быть не только строками, но и списками строк. Если так оно и есть, функция перебирает все элементы этого списка и упаковывает переменные, имена которых она встретила. Вот небольшой пример:

```
$a = "AAA";
$b = "BBB";
$c = "CCC" ;
$Arr = array("b", "c" ) ;
$Massiv = compact("a",$Arr);
// $Massiv = array("a"=>"AAA","b" = "BBB","c" = "CCC");
```

### 7.9.2. Извлечение переменных из массива. Функция extract()

Существует функция, противоположная функции `compact()` — это функция `extract()`. Она создает переменные с именами, соответствующими именам ключей, и присваивает им значения элементов массива.

Распакуем наш «упакованный массив»:

```
extract($A);
```



http://all-ebooks.com

В процессе распаковки может оказаться, что уже существует переменная с именем, совпадающим с именем ключа в массиве $A. Для решения этой проблемы у функции extract() предусмотрен второй параметр, который позволяет определить поведение функции extract() в случае совпадения имен. Значение EXTROVERWRITE означает, что существующая переменная с именем ключа будет перезаписана, а EXTRSKIP — пропущена:

```
$Name = "Вася";
$Email = "vasya@vvv.org";
$A = compact("Name","Email");
$Name = "Ivan";

extract ($A,EXTR_OVERWRITE);
```

После выполнения этого фрагмента кода переменная **$Name** получит значение «Вася», так как функция extract() её перезапишет. Значение EXTROVERWRITE используется по умолчанию и может не указываться.

В PHP предусмотрено еще одно значение второго параметра функции extract() — это значение EXTRPREFIXALL. В этом случае при распаковке массива все имена создаваемых переменных будут предварены специальным префиксом, который нужно будет указать в качестве третьего параметра функции extract().

Например, если мы в рассмотренном выше примере распаковку произведем следующим образом:

```
extract($A,EXTR_PREFIX_ALL, "D_");
```

то на выходе будут созданы переменные $D_Name и $D_Email, то есть при создании переменных к имени ключа присоединится указанный префикс D.

## 7.10. ПАРА ПОЛЕЗНЫХ ФУНКЦИЙ ДЛЯ РАБОТЫ С МАССИВАМИ

### Разбор URL

Пусть у вас есть URL (например, http://www.dkws.org.ua/index.php), из которого требуется извлечь доменное имя. Поскольку PHP разрабатывался как инструмент программирования интернет-приложений, разумно будет предположить, что для этой типичной задачи в нём имеется стандартная функция. Она действительно есть: функция parse_url() специально предназначена для разбора URL. Она возвращает ассоциативный массив, содержащий различные фрагменты URL, например:

```
$fr_url = parse_url("http://www.dkws.org.ua/index.
```

```
php?page=articles");
echo $fr_url[host];         // выводим доменное имя
echo $fr_url[port];         // выводим порт (если указан)
echo $fr_url[path];         // выводим путь
echo $fr_url[user];         //  выводим  имя  пользователя  (если
указано)
echo $fr_url[password];     // выводим пароль (если указан)
echo $fr_url[query];        // выводим строку запроса (если указана)
```

Если вас интересует этот вопрос подробнее, посетите следующую страничку:

```
http://www.dkws.org.ua/phpbb2/viewtopic.php?t=91
```

#### Создание диапазона чисел

Иногда нужно создать диапазон чисел, например, для нумерации записей в выводимой таблице. Создать диапазон чисел можно очень просто:

```
$List = array();
for($i=10; $i<=100; $i++) $List[]=$i;
```

Но можно и еще проще: вызов функции `range(10, 100)` вернет список, заполненный целыми числами от 10 до 100 включительно.

## 7.11. НЕБОЛЬШОЙ ПРАКТИЧЕСКИЙ ПРИМЕР

Попробуем написать сценарий, выводящий содержимое каталога. Причем сценарий должен выводить содержимое каталога в алфавитном порядке и каталоги должны предшествовать файлам (листинг 7.2). Мы будем использовать ранее написанную и приведенную в листинге 7.1 функцию `cmp()`, которая сравнивает файлы и каталоги.

**Листинг 7.2. Сценарий files.php**

```
<?// Получаем содержимое текущего каталога.  Для нашего сценария
// текущим  будет  каталог  /var/www/html // $DocumentRoot)
exec("ls -1",$Files);
function cmp($el_1,$el_2)
{  if (is_dir($el_1) && !is_dir($el_2)) return -1;
   if (!is_dir($el_1) && is_dir($el_2)) return 1;
  if($el_1<$el_2) return -1;
   elseif($el_1>$el_2) return 1;   else return 0;}
uasort($Files,"cmp");
echo "<html><title>Сценарий files.php</title><body>";
```



http://all-ebooks.com

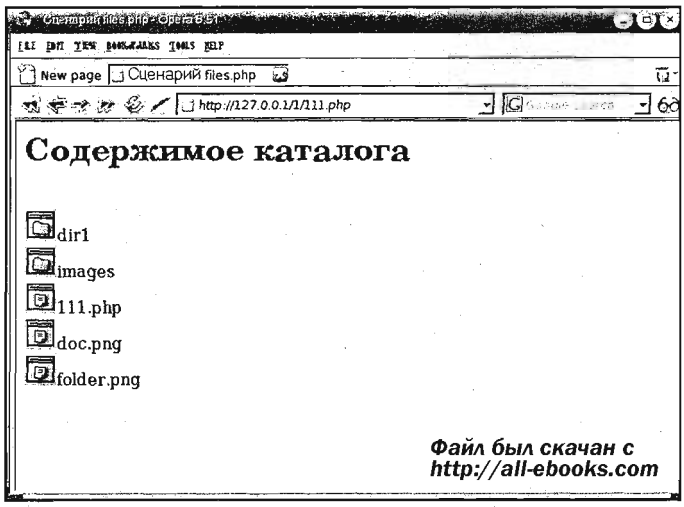



***Рис. 7.1. Результат работы сценария files.php***

```
echo "<h1>Содержимое каталога</h1><p>";
foreach($Files as $f)
  if (is_dir ($f))  echo "<br><img src=folder.png>$f";
else echo "<br><img src=doc.png>$f";
echo "</body></html>";
?>
```

Разберемся, как работает сценарий. Сначала мы формируем список файлов и каталогов. Можно было бы воспользоваться стандартными функциями PHP, но намного проще (в Linux) вызвать команды **ls** с параметром «-1» (вывод в одну колонку). Функция `exec()` вызывает команду **ls** и записывает ее вывод в список `$Files`.

Потом мы вызываем функцию `uasort()`, которая сортирует наш список в нужном нам порядке — сначала она сортирует каталоги в алфавитном порядке, а затем файлы. В результате мы получаем список, в котором сначала идут каталоги, отсортированные по алфавиту, а затем файлы (тоже отсортированные по алфавиту).

В цикле `foreach` мы анализируем элемент нашего списка: если он является каталогом, мы выводим рядом с его именем картинку папки, а если файлом — картинку документа (рис. 7.1).

# ГЛАВА 8.

# ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ В PHP

http://all-ebooks.com

## 8.1. ЧТО ТАКОЕ ПОЛЬЗОВАТЕЛЬСКИЕ ФУНКЦИИ

В любом языке программирования существуют подпрограммы. В языке C они называются функциями, в ассемблере — подпрограммами и вызываются инструкцией CALL, а в языке Pascal существует даже два вида подпрограмм: процедуры и функции.

Подпрограмма — это специальным образом оформленный фрагмент программы, к которому можно обратиться из любого места внутри программы. Подпрограммы существенно упрощают жизнь программистам: они улучшают читабельность исходного кода, сокращают исходный код (не нужно писать один и тот же фрагмент несколько раз). В языке PHP подпрограммы называются функциями.

До сих пор я упоминал только стандартные функции, предусмотренные самим языком PHP. Однако зачастую стандартных функций не хватает и возникает необходимость создания своей собственной функции, выполняющей нужные именно нам действия. Такие создаваемые самостоятельно функции и называются пользовательскими. Тому, как их создавать и использовать, посвящена эта глава.

Итак, в языке PHP подпрограммы называются функциями. Вот особенности использования функций в PHP:

- Вы можете использовать параметры, установленные по умолчанию. Это, в частности, позволяет вызывать одну и ту же функцию с переменным числом параметров.

http://all-ebooks.com

- Функция может возвращать значение любого типа.
- Область видимости переменных внутри функции, как и в других языках программирования, древовидная.
- Как в языке C, вы можете вернуться из функции с помощью инструкции return.
- Можно изменять значения переменных, переданных в качестве аргумента (как параметр-переменная в Паскале).

Язык PHP предоставляет программисту огромную свободу при создании собственных функций. При программировании на других языках мне очень не хватало параметров по умолчанию, которые позволяют написать функцию с различным числом параметров. В языке C++ появился механизм перегрузки функций, но параметры по умолчанию — это более гибкое средство. Ведь в C++ для каждой перегруженной функции нам нужно заново писать ее тело. А в языке PHP этого делать не нужно. В этой главе мы обязательно рассмотрим параметры по умолчанию.

Однако у PHP-функций, кроме преимуществ, есть и недостатки, причем существенные. Первый недостаток связан с объявлением локальной функции. В PHP вы не можете объявить локальную функцию, как это можно делать в других языках. То есть вы не можете создать функцию внутри другой функции таким образом, чтобы первая (вложенная) функция была видна только во второй функции. В PHP вложенная функция будет видна всей программе, то есть не будет локальной. Рассмотрим следующий пример:

**Листинг 8.1. Вложенность функций**

```
<?
function first()  {
 echo "First";
   function Second()
   {
     echo "Second";
   }
}
First();
second();

?>
```

Сценарий напечатает «FirstSecond» (рис.8.1). Это означает, что якобы локальная функция доступна во всей программе.



http://all-ebooks.com

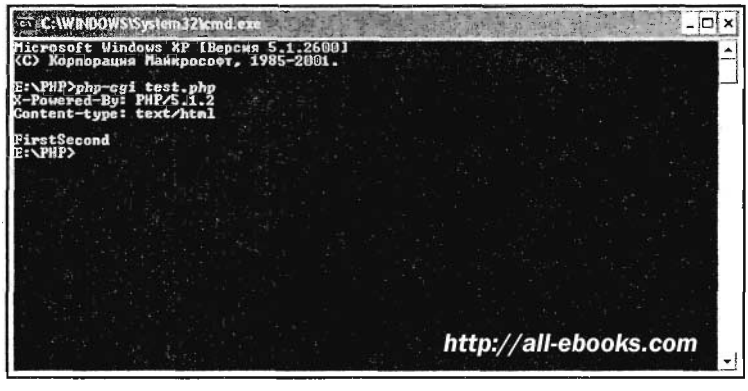



***Рис. 8.1. Вывод сценария 8.1.***

**Примечание.**

Обратите внимание на имена функций при объявлении (first и Second) и при вызове (First и second). Это не опечатка: PHP нечувствителен к регистру символов в именах функций. Вы можете вызвать функцию first(), FIRsT() или как-то иначе.

Второй недостаток связан с областью видимости функций. Но сначала немного скажем о глобальных и локальных переменных. Глобальные переменные — это переменные, которые видны и доступны всей программе со всеми входящими в нее подпрограммами (функциями). Локальные переменные, определенные внутри подпрограмм, видны и доступны только внутри той подпрограммы (функции), в которой они определены.

Так вот, в PHP все объявленные и используемые в функции переменные по умолчанию локальны для функции. Это значит, что в отличие от других языков программирования из тела функции невозможно изменить значение глобальной переменной. Если вы в функции будете использовать переменную с именем, идентичным имени глобальной переменной, то это никакого отношения к глобальной переменной иметь не будет. Просто в функции будет создана локальная переменная с таким же именем, невидимая за пределами этой функции.

Следующий пример подтверждает данный факт:

**Листинг 8.2. Локальные и глобальные переменные** *http://all-ebooks.com*

```
<?
$i = 10;

function change()  {
$i = 5;
```

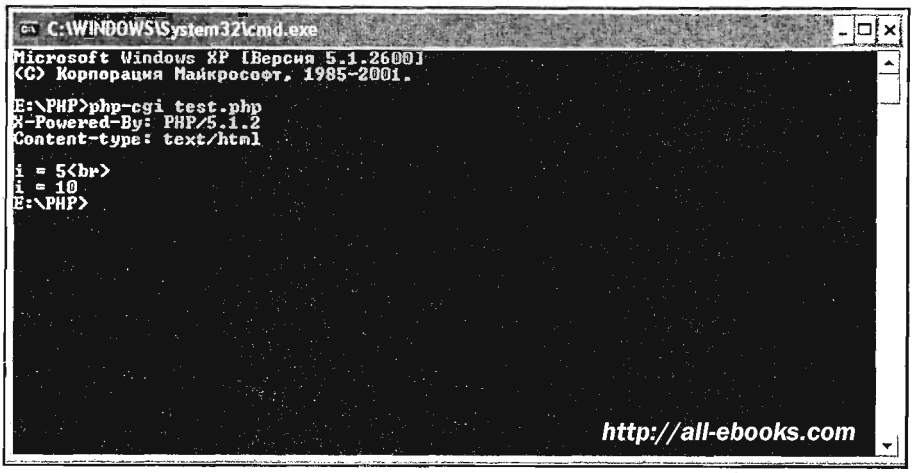



*Рис. 8.2. Вывод сценария 8.2.*

```
echo "i = $i<br>\n";
}
change();
echo "i = $i";
?>
```

Сценарий сначала напечатает число 5, а потом число 10 (рис. 8.2).

Тем не менее в PHP существует возможность избавиться от этого недостатка. Для этого предусмотрено использование специальной инструкции `global`, которая позволяет функции работать с глобальными переменными. Подробнее об этом мы будем говорить в п. 8.6.

## 8.2. СОЗДАНИЕ ПОЛЬЗОВАТЕЛЬСКИХ ФУНКЦИЙ

### 8.2.1. Объявление функций

Функция может быть объявлена в любой части программы (в пределах операторов <? и ?>), но до места первого ее использования. Другими словами, не нужно никакого предварительного объявления, как в других языка программирования, в частности в C. Дойдя до определения функции, транслятор проверит корректность определения и оттранслирует определение функции во внутреннее представление, но транслировать код он не станет. Данный подход является самым правильным — зачем транслировать код, который, возможно, никогда не будет использоваться?

Для объявления функций используется такой синтаксис:



http://all-ebooks.com

```
function Имя (аргумент1[=значение1],...,аргументl[=значениеl])
{
тело функции
}
```

Объявление функции начинается со служебного слова **function**, за которым следует имя функции, после имени — список аргументов в скобках. Тело функции заключается в фигурные скобки и может содержать произвольное число операторов.

В PHP функции могут обладать необязательными параметрами или параметрами по умолчанию — о них мы поговорим чуть позже.

Требования к именам функций:

- Имя функции может содержать русские буквы, но давать функциям русские имена по понятным причинам не рекомендуется.
- Имя функции не должно содержать пробелов и пробельных символов.
- Имя должно быть уникальным с точностью до регистра, то есть функции `Fl()` и `fl()` — это одна и та же функция.
- Функциям можно давать такие же имена, как и переменным, только без знака доллара в начале имени.

Так как тип возвращаемого значения может быть любым, его (тип) не указывают вовсе. Для передачи результата работы функции в основную программу используется инструкция return. Если функция ничего не возвращает, инструкцию return не указывают. Инструкция return может возвращать все что угодно, в том числе и массивы.

### 8.2.2. Первая функция

Мы уже знаем достаточно о функциях, чтобы написать собственную функцию. Например, следующая функция (листинг 8.3) возвращает массив из 10 случайных чисел:

**Листинг 8.3. Функция, возвращающая массив из 10 случайных чисел**

```
<?
function GetRand()    {

$LocalArr = array();
mt_srand(time());

for ($i=0; $i<10;$i++)
```

```
$LocalArr [] = mt_rand(0,100);

return $LocalArr;
}

$A = GetRand();
foreach($A as $K=>$v) echo "$v ";

?>
```

В этой функции сначала мы создаем пустой локальный массив $LocalArr, затем настраиваем генератор случайных чисел на новую последовательность вызовом mt_srand(). Чтобы при каждом новом запуске программы получать новую строку, мы инициализировали генератор текущим значением системного времени. Затем мы заполняем массив случайными числами и передаем его в основную программу, где он присваивается массиву $A. Цикл foreach выводит значения массива.

## 8.3. ПЕРЕДАЧА ФУНКЦИИ ПАРАМЕТРОВ

В большинстве языков программирования поддерживаются два способа передачи функции аргументов: по значению и по ссылке. Не является исключением и PHP.

Параметр, переданный по значению, доступен внутри функции только для чтения: функция может его использовать, но не может изменить. В качестве такого параметра необязательно указывать переменную: можно передать само значение. Отсюда название — параметр-значение.

Следующий фрагмент кода выведет строку «Параметр = 3» дважды:

```
function func($a)
{
echo "Параметр = $a\n" ;
}

$b = 3;
func(3);
func($b);
```

Параметром, переданным по ссылке, функция может распоряжаться как обычной переменной, отсюда название: параметр-переменная. Очевидно, что в качестве такого параметра нельзя передавать значение, нужно обязательно передать саму переменную.



http://all-ebooks.com

*Рис. 8.3. Десять случайных чисел*

Для указания того, что данный параметр передается по ссылке, нужно при объявлении функции предварить имя переменной символом амперсанда (листинг 8.4).

**Листинг 8.4. Передача переменной по ссылке**

```
<?
$First = $Second = 5;

echo "<PRE>";

function f1($f, &$s)
{
echo "Изменяем параметры\n";

$f = 7; $s = 10;

echo "First = $f, Second = $s\n";
}

echo "Значение переменных до вызова функции\n";

echo "First = $First Second = $Second\n";

f1($First, $Second);

echo "First = $First Second = $Second\n";

?>
```

Сначала переменные `$First` и `$Second` равны 5. Затем мы в теле функции изменяем оба параметра — обыкновенный и переданный по ссылке. Как показывает рис. 8.4, внутри функции изменились значения обоих параметров, а на вызывающей программе отразилось только изменение значения переменной `$Second`, переданной по ссылке.

## 8.4. ПАРАМЕТРЫ ПО УМОЛЧАНИЮ

Потребность в разработке функции с переменным количеством параметров обычно возникает по одной из двух причин:

- функция должна возвращать значения разных типов в зависимости от набора параметров;
- функция должна описывать максимально общий случай и для этого распоряжаться большим количеством параметров, но вызывают её чаще всего для частных случаев, когда значения большинства параметров заранее известны.

Введение параметров по умолчанию решает эту проблему. Если при вызове функции не указать параметр, которому приписано значение по умолчанию, то интерпретатор передаст функции это значение.

Параметры по умолчанию объявляются так:

```
имя_параметра=значение
```

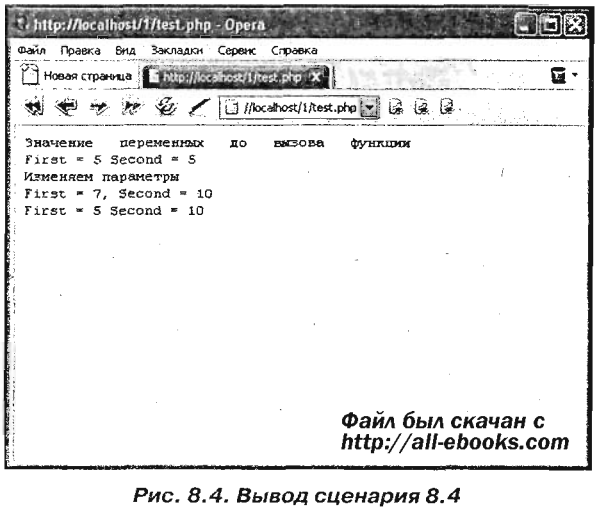



*Рис. 8.4. Вывод сценария 8.4*



http://all-ebooks.com

Например,

```
function my_func ($x = 10, $y = 5) { … }
```

Если функция имеет не только параметры по умолчанию, но и обычные параметры, то при объявлении функции параметры по умолчанию должны следовать после обычных. Пример:

```
function GetMax($Arr, $i = 0)  // правильное объявление
function GetMax($i = 0, $Arr ) // ошибка
```

Следующий пример (листинг 8.5) демонстрирует гибкость языка PHP. Предположим, что нам нужно написать функцию `Multi`, которая использует глобальный массив `Array`. Когда функции не переданы параметры, она возвращает максимальное значение элементов массива. Когда ей передано число, она пытается найти в массиве элемент с указанным значением и вернуть его индекс (если элемент не найден, вместо индекса будет возвращено значение -1).

**Листинг 8.5. Применение параметров по умолчанию**

```
<?

$Array = array();

function Multi($a="-999999") {

// глобальный массив Array() будет доступен внутри функции Multi
global $Array;

$i = 0;

if ($a=="-999999")
{
// функция была вызвана без параметров, т. е. получила значение по
умолчанию;
$max = $Array[0];

for($i=0; $i<10; $i++)
  if ($max < $Array[$i]) $max = $Array[$i];
return $max;
}
else
{
// поиск элемента со значением $a
for($i=0;$i<10;$i++)
  if ($Array[$i] == $a) return $i;
return -1;
}
}
```

http://all-ebooks.com

```
// основная программа
for($i=0;$i<10;$i++) $Array[]=$i+7;
echo Multi().”\n”;
echo Multi(10).”\n”;
?>
```

Если взяться реализовать то же самое на языке C++, то код увеличится в полтора раза, а тело функции придется переписывать два раза.

## 8.5. ПЕРЕМЕННОЕ ЧИСЛО ПАРАМЕТРОВ В ФУНКЦИИ

Параметры по умолчанию — это, конечно, очень полезные функции. Но иногда мы точно не знаем, сколько параметров будет передано нашей функции. Специально для такого случая разработчики PHP предусмотрели возможность использования переменного числа параметров. Напишем функцию `print_str()`, которая будет выводить переданные ей строки в виде таблицы HTML (листинг 8.6).

**Листинг 8.6. Функция print_str**

```
<?
function print_str()
{
echo “<table border=1 width=100%>”;
echo “<tr><td bgcolor=blue><font color=white>Номер</font></td>”;
echo “<td bgcolor=blue><font color=white>Параметр</font></td>”;
for  ($i = 0; $i<func_num_args();  $i++)
{
echo “<tr><td>$i</td><td>”.func_get_arg($i).”</td></tr>”;
}
echo “</table>”;
}
print_str(“Первый”,”Второй”,”Третий”,”Четвертый”);
?>
```

Основными в этой программе являются функции `func_num_args()` и `func_get_arg()`. Первая возвращает количество переданных функции аргументов, а вторая — аргумент с указанным номером.

Кроме данных двух функций, можно использовать еще одну — `func_get_args()`, которая возвращает список всех аргументов. Использовать ее еще удобнее, чем первые две:

```
foreach(func_get_args() as $a=>$v)
{
echo “<tr><td>$a</td><td>$v</td></tr>”;
}
```



http://all-ebooks.com

И еще один важный момент. Обратите внимание на объявление функции `print_str()`: в скобках не должно быть указано ни одного параметра.

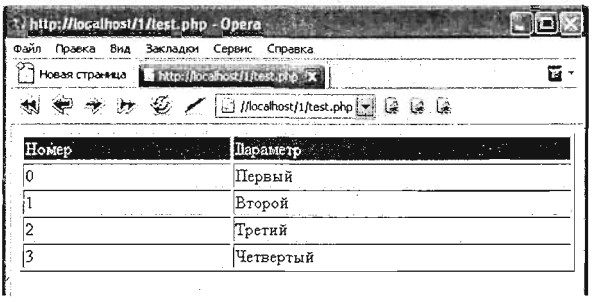



***Рис. 8.5. Созданная таблица параметров***

## 8.6. ЛОКАЛЬНЫЕ И ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ

Мы неоднократно сталкивались в этой книге с локальными и глобальными переменными. Локальные переменные доступны только в функции, а глобальные доступны как в основной программе, так и во всех вызываемых ею функциях. В листинге 8.7 приведен небольшой пример.

**Листинг 8.7. Использование глобальных и локальных переменных**

***Файл был скачан с http://all-ebooks.com***

```
<?
$i = 10;
$g = 7;

function  f1()
{
$x = 5;
$i = 0;
global $g;

echo "g = $g\n";
echo "i = $i\n";
}

f1();

// Переменная $x является локальной и недоступна в основной
// программе,
// поэтому мы получим сообщение о том, что данная переменная
// не определена (рис. 8.6).

echo "$x\n";
?>
```

http://all-ebooks.com

Переменные `$i` и `$g` — глобальные, они доступны во всей программе. Для передачи глобальных переменных в функцию используется инструкция **global**:

```
global список переменных
```

Если мы не выполним эту инструкцию, наша функция не получит доступ к переменной `$g`. Обратите внимание на две переменные `$i`: одна из них глобальная, а другая — локальная (в теле функции).

В функции будет использоваться локальная переменная, однако если мы объявим переменную глобальной:

```
global $g, $i;
```

то будет использоваться значение глобальной переменной. Попробуйте изменить функцию `f1()` так, как показано в листинге 8.8:

**Листинг 8.8. Некорректное использование глобальных и локальных переменных**

```
<?
$i = 10;
$g = 7;
function f1()
{

$i = 5;
echo $i;          // выведет 5
global $i;
echo $i;          // выведет 10

}
f1();

?>
```

Если функция изменит значение глобальной переменной, то вызывающая программа получит новое значение этой переменной:

```
<?
$g = 5;

function f1() { global $g; $g = 10; };

f1();

echo $g;     // выведет 10
?>
```



http://all-ebooks.com

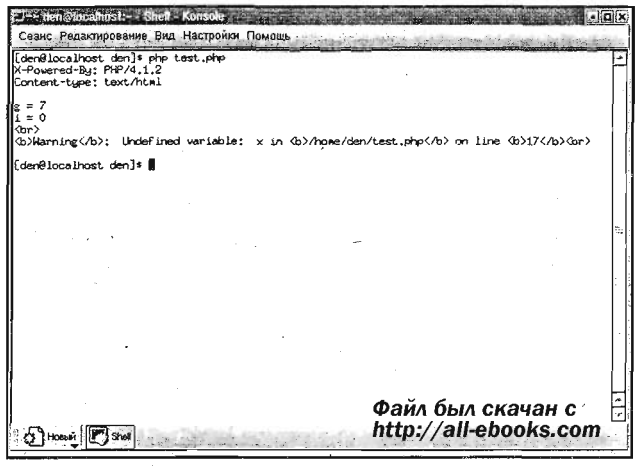



***Рис. 8.6. Локальные и глобальные переменные***

Мы уже говорили о массиве $GLOBALS, поэтому я не стану повторяться, а лишь скажу, что локальные переменные функций не являются элементами данного массива.

## 8.7. СТАТИЧЕСКИЕ ПЕРЕМЕННЫЕ

Кроме локальных и глобальных переменных, в PHP существует еще один тип переменных — статические. Если в теле функции объявлена статическая переменная, то компилятор не будет ее удалять после завершения работы функции. Следующий пример демонстрирует работу со статической переменной:

```
<?

function f1() { static $x=0; $x++; echo "$x "; }

for ($i = 0; $i++<5; ) f1();

?>
```

Следующий сценарий выведет строку 1 2 3 4 5. Если удалить инструкцию static, будет выведена строка 1 1 1 1 1, потому что переменная $x будет удаляться при завершении работы функции и обнуляться при каждом ее вызове.

Статические переменные не являются элементами $GLOBALS.

## 8.8. РЕКУРСИЯ

Рекурсией называется вызов функции самой себя, а такой вызов соответственно называется рекурсивным. Рекурсия бывает прямая и непрямая. Рассмотрим пример прямой рекурсии (листинг 8.9).

**Листинг 8.9. Прямая рекурсия**

```
<?
function fact($x)
{
if  ($x==0) return 1;
else return $x*fact($x-1);
}
echo fact(7);
?>
```

Как вы уже успели догадаться, функция fact() вычисляет факториал x. Функция fact() в своем теле вызывает саму себя — это и есть прямая рекурсия.

Непрямая рекурсия возникает, когда первая функция вызывает вторую, а вторая — первую.

Механизм рекурсии одновременно очень полезен и очень опасен. При создании рекурсивных функций нужно быть особенно внимательным, чтобы избежать зацикливания. Следующая функция (листинг 8.10) вызовет зацикливание.

**Листинг 8.10. Некорректное использование рекурсии**

```
function fact($x)
{

if ($x==0) return 1;
else return $x*fact($x);

}
```

## 8.9. ПОЛЕЗНЫЕ ПРИМЕРЫ

В этом пункте мы рассмотрим два небольших, но полезных примера — функцию init(), выполняющую инициализацию переменных, и функцию, демонстрирующую работу с массивами.



http://all-ebooks.com

### 8.9.1. Инициализация переменных

Начнем с более простой функции init(). Довольно часто при написании крупных сценариев возникает потребность в инициализации переменных, то есть присваивании переменным начальных значений. Например, нам нужно после каждой итерации цикла обнулить, скажем, пять-шесть переменных. Неудобно каждый раз писать

```
$a=0; $b=0; $c=0; $d=0; $e=0;
```

Гораздо удобнее вызвать функцию init(0,"a","b","c","d","e").

Работать она будет так: мы передаем ей несколько параметров, первый из которых — это значение, а все остальные — имена переменных, которым нужно присвоить это значение. Текст программы приведен в листинге 8.11.

**Листинг 8.11. Функция инициализации переменных**

```
<?
function init()
{
if (func_num_args() == 0) return 0;
else
{
  // извлекаем из списка параметров значение,
  // которое нужно присвоить переданным переменным
  $value = func_get_arg(0);

  for($i=1; $i<func_num_args(); $i++)
  {
    // извлекаем имя переменной, которую нужно инициализировать
    $var_to_set = func_get_arg($i);

    foreach($GLOBALS as $var=>$val)
      if ($var == $var_to_set) $GLOBALS[$var]=$value;
  }
}
}

$i = 7; $j = 6;

init(1,"i","j");

echo "$i $j\n";

?>
```

Теперь небольшой комментарий к функции init(). Если ей не переданы параметры, функция завершает свою работу и возвращает 0. В противном случае она получает первый параметр — $value. Значением этого параметра она будет инициализировать все переменные. Затем она просматривает все параметры, начиная с номера 1 (под номером 0 в списке параметров идёт значение). Если имя параметра совпадает с одним из ключей массива $GLOBALS, она присваивает значение $value элементу массива $GLOBALS с ключом $var. Напомню, что в массиве $GLOBALS имена переменных представлены в виде:

```
var = val
где var — это имя переменной, а val — ее значение.
```

После выполнения функции init(1,"i","j") переменным $i и $j присвоено значение 1.

## 8.9.2. Параметры-массивы

В этом небольшом пункте мы рассмотрим очень полезный пример, а именно передачу массива в качестве аргумента функции. Для этого мы напишем свою функцию array_param(). Наша функция будет работать не с обыкновенной скалярной переменной, а с целым массивом! Получив массив, она вычисляет номера минимального и максимального элементов, а затем рассчитывает среднее арифметическое элементов массива. В итоге функция возвращает список, состоящий из трех элементов:

- номера максимального элемента;
- номера минимального элемента;
- среднего арифметического.

Данная функция демонстрирует передачу массива в качестве параметра и возврат массива, полученного в результате работы функции, в вызывающую программу (листинг 8.12).

**Листинг 8.12. Передача и возврат массива**

```
<?
function array_param($Arr)
{

$min = $max = $Arr[0];    // сначала Min = Max = первый элемент
$mn_ind = $mx_ind =0;     // Это индексы минимального и макс.
```



http://all-ebooks.com

```
                                // элементов
$avg =0;                        // Среднее арифметическое

foreach($Arr as $k=>$v)
{

if ($max < $v) { $max=$v; $mx_ind=$k; }; // Вычисление $max
if ($min > $v) { $min=$v; $mn_ind=$k; }; // Вычисление min
$avg = $avg + $v;

}

$avg   =   $avg / count ($Arr);

$Res[]=$mx_ind;
$Res[]=$mn_ind;
$Res[]=$avg;

return   $Res;                          //Передача результата
}
for($i=1; $i<11; $i++) $Arr[]=$i;    //Заполнение массива Arr[]

foreach(array_param($Arr) as $v) echo "$v ";

?>
```

Функция array_param() принимает список, обрабатывает его и возвращает также список. Обратите внимание на обработку списка в основной программе. Мы могли бы присвоить возвращаемое функцией значение другому массиву, а потому работать с этим массивом:

```
$Res = array_param($Arr);

foreach($Res as $v) echo "$v ";    //Выводим только значения, но не
индексы
```

### 8.9.3. Возврат функцией ссылки

В этой главе мы уже рассмотрели механизм передачи параметра по ссылке, позволяющий функции изменять переданные ей параметры-переменные. Однако это еще не все. В PHP функции могут возвращать ссылки. Рассмотрим следующий пример (листинг 8.13).

**Листинг 8.13. Возврат ссылок функциями**

```
<?
// присваиваем переменной значение 100
$a=100;
```

```
// функция, возвращающая ссылку
function &SS()
{
// переменная $a - глобальная
global    $a;
// возвращаем ссылку на переменную $a;
retúrn    $a;
}

// присваиваем ссылку переменной $b
$b=&SS();
$b=0;
echo $a;    // выводит 0
?>
```

Программа выведет 0, а это означает, что `$a` и `$b` — синонимы, то есть, если вы измените значение одной переменной, автоматически будет изменено значение второй. Разорвать связь между объектом и ссылкой может функция `UnSet()`.

## 8.10. УСЛОВНО ОПРЕДЕЛЯЕМЫЕ ФУНКЦИИ

Уже было сказано, что в PHP функции могут возвращать значения разных типов в зависимости от переданных аргументов. PHP предоставляет разработчику еще большую свободу: в зависимости от внешних факторов (например, от типа операционной системы или ее версии) одна и та же функция может выполнять различные действия. Рассмотрим следующий пример:

```
<?
if ($OS=="Linux")
{
function GetOS()  { return "Linux"; }
}
elseif { function GetOS()  { return "Windows"; }}

GetOS() ;
?>
```

Конечно, оператор сравнения можно было бы использовать и внутри функции, но данный пример слишком тривиален, чтобы ощутить все удобство условно определяемых функций. Представьте, что у нас не простенькая функция, возвращающая тип операционной системы, а сложная функция, производящая аутентификацию пользователя в зависимости от многих факторов, в том числе и от типа операционной системы. Вот тогда условно определяемые функции нам очень пригодятся.



http://all-ebooks.com

# ГЛАВА 9.

# ФУНКЦИИ ДЛЯ РАБОТЫ СО СТРОКАМИ

http://all-ebooks.com

## 9.1. ОПЕРАЦИИ НАД СТРОКАМИ

### 9.1.1. Строки в PHP

Строка — это самый универсальный объект в мире PHP. Очень часто строки содержат числа (как целые, так и вещественные). Причем PHP позволяет работать с такими строками как с обыкновенными числами. Мы можем «упаковать» в строку любой объект, например массив, и работать с ним как со строкой. Как это сделать и для чего это нужно, вы узнаете в самом последнем пункте этой главы.

Чем же хороши сами строки в PHP? В отличие от C преимущества PHP-строк заключаются в следующем:

- Строка может содержать любые символы. Напомню, что C-строка не может содержать символы с кодом 0, поскольку первый такой символ трактуется как конец строки.
- Строка может содержать произвольное количество символов. Длина строки ограничивается только объемом оперативной памяти, что позволяет читать в строку целые файлы и работать с ними как с обыкновенными строками, например сравнивать.

Базовыми операциями над строками являются:

- Конкатенация строк (присоединение одной строки к другой).
- Сравнение строк.



http://all-ebooks.com

### 9.1.2. Конкатенация строк

В других языках программирования, например в языке Pascal, для конкатенации (присоединения) строк используется символ «+». Однако в PHP использование оператора «+» не рекомендуется: если строки содержат числа, то вместо объединения строк будет выполнена операция сложения двух чисел. Рассмотрим следующий сценарий (листинг 9.1).

**Листинг 9.1. Слияние строк оператором «+» и что из этого получается**

```
<?
$a = "20";
$b = "10";
$c = $a + $b;    // число + число

echo "$c\n";     // выведет 30

$d = "Hello";
$c = $a + $d;    // число + строка

echo "$c\n";     // выведет 20 ($a)
?>
```

Как видите, для конкатенации строк оператор «+» не подходит. Вместо него следует использовать оператор «.» (точка) — см. листинг 9.2.

**Листинг 9.2. Корректная конкатенация строк**

```
<?
$a = "20";
$b = "10";
$c = $a.$b;     // число + число
echo "$c\n";    // выведет 2010

$d = "Hello";
$c = $a.$d;     // число + строка
echo "$c\n";    // выведет 20Hello
?>
```

### 9.1.3. Сравнение строк

Подобная ситуация с преобразованием типов происходит и при сравнении строк. Операторы сравнения = = и != требуют преобразования типов. Поэтому они не должны использоваться для сравнения строк (листинг 9.3).

http://all-ebooks.com

*Рис. 9.1. Результат корректной конкатенации строк*

**Листинг 9.3. Сравнение строк**

```
<?
$a = 0;
$b = 1;
if ($a == "") echo "a - пустая строка";
if ($b == "") echo "b - пустая строка";
?>
```

Этот сценарий сообщит нам, что `$a` — пустая строка. Причина такого странного вывода в том, что пустая строка "" в ходе сравнения преобразуется в число 0. В PHP операнды сравниваются как строки только в том случае, если оба они являются строками. В противном случае они сравниваются как числа. При этом любая строка, которую интерпретатору не удастся перевести в число (в том числе и пустая строка), будет восприниматься как 0.

Можно придумывать некоторые обходные пути (заключать числа в кавычки, чтобы они воспринимались как строки, и т. п.), но при этом очень высока вероятность возникновения ошибок.

Лучше всего при сравнении использовать оператор эквивалентности = = = (тройной знак равенства). Этот оператор всегда позволяет производить корректно сравнение, так как он сравнивает величины и по значению, и по типу.

**Примечание.**

Для сравнения строк можно также использовать функцию strval(), однако она обладает определенными ограничениями и особенностями, что делает ее использование нежелательным. Поэтому эта функция больше не используется в PHP, начиная с версии 4.



http://all-ebooks.com

## 9.2. ОСНОВНЫЕ ФУНКЦИИ РАБОТЫ СО СТРОКАМИ

Функция `strlen()` возвращает длину строки, то есть количество символов, которое содержит строка:

```
$a = "Hello";
echo strlen($a);     // выведет 5
```

Функция `strpos()` ищет подстроку в строке. Функции нужно передать три параметра:

- строку, в которой мы будем искать подстроку;
- искомую подстроку;
- позицию (номер символа), с которой начинается поиск. Этот параметр необязателен и по умолчанию равен 0 (начало строки).

Если подстрока не найдена, функция возвращает false, в противном случае — номер позиции, с которой начинается вхождение подстроки в строку:

```
echo strpos("Hello","el");  // выведет 1
```

При работе с данной функцией начинающие программисты, в том числе и я когда-то, очень часто допускают одну и ту же ошибку. Вот она. Предположим, что нам нужно найти подстроку «ab» в строке «abba», и мы пишем следующее:

```
If (strpos("abba","ab") != false) echo "Строка ab найдена в abba";
```

Мы ожидали, что этот сценарий выведет «Строка ab найдена в abba», но он ничего не выводит! Дело в том, что подстрока «ab» входит в строку «abba», начиная с нулевой позиции, а 0 и false при использовании оператора сравнения != являются синонимами. Опытные программисты пишут так:

```
if (strpos("abba","ab") !== false) echo "Строка ab найдена в abba";
```

или так:

```
if (strpos("abba","ab") === false) echo "Строки ab нет в abba";
```

Функция `substr(string $str, int $from, int $length)` возвращает заданный участок строки, то есть подстроку. Возвращается подстрока строки `$str` длины `$length`, начиная с позиции `$from`. Если параметр `$from` будет отрицательным, то отсчет подстроки будет производиться с конца строки, а не с начала.

Параметр `$length` необязателен. Если его не указать, функция вернет подстроку от заданной позиции до конца строки.

http://all-ebooks.com

```
$str = "Book";
echo substr($str,1,1);     // выведет o
echo substr($str,-1,1);    // выведет k
```

Функция `strcmp($str1, $str2)` сравнивает две строки и возвращает следующие значения:

- 0, если строки равны;
- 1, если строка `$str1` лексикографически больше `$str2`;
- -1, если строка `$str1` «меньше» `$str2`.

Функция `strcasecmp(string $str1, string $str2)` сравнивает две строки без учета регистра символов, то есть строки «Book», «BOOK» и «book» для этой функции равны.

Функция `str_repeat(string $s, string $n)` повторяет указанную строку `$s` указанное количество (`$n`) раз:

```
echo str_repeat("String", 2);          // выведет "StringString"
```

## 9.3. ЗАМЕНА В СТРОКЕ

Функция `str_replace($str1, $str2, $source)` заменяет в строке `$source` все вхождения подстроки `$str1` на подстроку `$str2` и возвращает результат в виде новой строки. Исходная строка `$source` в процессе работы функции не изменяется.

Пусть нам нужно заменить символы новой строки `\n` на тег `<br>`. Это можно сделать так:

```
$str = str_replace("\n","<br>",$str);
```

или так:

```
$str = nl2br($str);
```

Функция `nl2br()` работает быстрее, чем `str_replace()`, однако она не учитывает символов перевода каретки, которые часто встречаются в текстовых файлах Windows.

Функция `WordWrap()` оказывается очень полезной, если нам нужно отформатировать текст письма перед его оправкой. Эта функция разбивает текст на строки, завершая каждую символом разрыва (по умолчанию `\n`). В каждой строке должно быть не больше 75 символов (вы можете указать другое количество). Пример использования функции `WordWrap()` приведен в листинге 9.4.



http://all-ebooks.com

**Листинг 9.4. Использование функции WordWrap()**

```
<?
$s = "Очень длинный блок текста, который нужно разбить на части
перед  отправкой адресату";
echo $s."\n\n";

// разбиваем по 15 символов

$s = WordWrap ($s, 15, "\n");

echo $s;
?>
```

**Примечание.**

Функция WordWrap() не разбивает блок на отдельные строки в смысле PHP, а лишь вставляет символ перевода строки через заданное число символов. Это влияет только на отображение исходной строки.

Функция `strip_tags()`. Очень полезной является функция `strip_tags()`, удаляющая из строки все HTML-теги:

```
$s = strip_tags($s);
```

Вы можете указать, какие теги не стоит удалять. Если вы не хотите удалять теги ссылок и тег заголовка <head>, тогда укажите их при вызове функции:

```
$s = strip_tags ($s, "<a><head>");
```

## 9.4. ФУНКЦИИ ДЛЯ РАБОТЫ С ОТДЕЛЬНЫМИ СИМВОЛАМИ

Как и в других языках программирования, в PHP мы можем работать с отдельно взятыми символами строки. Обратиться к любому символу строки можно по его индексу:

```
$str = "Book";
echo $str[0]; // выведет B
```

Если вы программировали на языке Pascal, вам должны быть знакомы функции `chr()` и `ord()`. В PHP эти функции работают почти так же, как и в Паскале. Функция `chr(int $code)` возвращает строку, состоящую из символа с кодом `$code`, а функция `ord($char)` — код символа `$char`:

```
echo chr(75);   // выведет K
echo ord('A');  // выведет 65 - код заглавной латинской буквы A
```

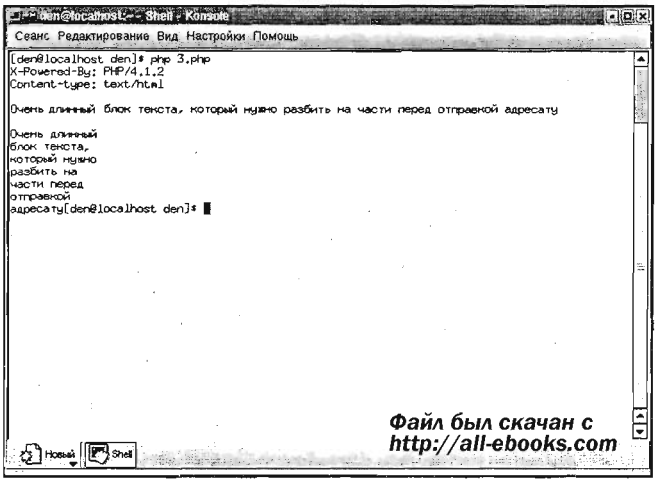



***Рис. 9.2. Результат работы WordWrap()***

Довольно интересной является функция `strrpos(string $str, char $chr)`. Хотя она и похожа названием на `strpos()`, но она выполняет совершенно иные действия. Она ищет последнее вхождение символа `$chr` в строку `$str`. Если строка `$chr` состоит из более одного символа, то будет использован первый из них.

Функция `strtr(string $s, string $from, string $to)` заменяет в строке `$s` все символы, которые встречаются в `$from`, на символы, которые расположенные в строке `$to` в тех же позициях, что и в `$from`. Чтобы было понятнее, давайте рассмотрим небольшой пример:

```
echo strtr("Book","o","a");
```

В строке «Book» все символы «o» будут заменены символами, стоящими в той же позиции в строке «a». В результате будет выведена строка «Baak».

## 9.5. ИЗМЕНЕНИЕ РЕГИСТРА СИМВОЛОВ. УСТАНОВКА И НАСТРОЙКА ЛОКАЛИ

Для перевода символов строки в нижний регистр используется функция `strtolower()`, а в верхний — `strtoupper()`:

```
$str = strtolower($str);
$str = strtoupper($str);
```

При изменении регистра русских букв могут возникнуть проблемы, избежать которых поможет правильная установка локали. Локалью будем на-



http://all-ebooks.com

зывать совокупность локальных настроек системы, таких как формат даты и времени, язык, кодировка.

Настройки локали очень зависят от операционной системы, поэтому если у вас дома все работает нормально, то это совсем не означает, что так и будет на машине вашего хостинг-провайдера.

Для установки локали используется функция `SetLocale()`, которой следует передать два параметра — категорию устанавливаемых параметров и локаль. Категория может быть следующей:

- LC_TYPE — используется для установки локали перевода символов нижний/верхний регистры (как раз то, что нам нужно).
- LC_NUMERIC — используется для установки локали для функции форматирования дробных чисел.
- LC_TIME — помогает настроить форматированный вывод даты и времени.
- LC_ALL — устанавливает все вышеперечисленные режимы.

Мы не будем рассматривать все категории, а остановимся на категории LC_TYPE, которая сейчас нужна нам. Любая локаль, установленная в системе, имеет свое уникальное имя, по которому можно к ней обратиться. Категория LC_TYPE устанавливает локаль для преобразования символов. Имя активизированной локали для категории LC_TYPE содержится в элементе с ключом LANG массива GLOBALS (это переменная окружения LANG). В моем случае — это ru_RU.koi8. Для корректного преобразования символов на моей машине нужно выполнить функцию:

```
SetLocale(LC_TYPE, "ru_RU.koi8");
```

Если вы работаете под Linux, в большинстве случаев вас устроит данный вызов (если, конечно, вы используете кодировку koi8-r). В большинстве случаев (в современных дистрибутивах) локаль уже настроена до вас, поэтому перед ее изменением с помощью `SetLocale()` следует сначала убедиться, что параметры настроены неправильно. На моей машине с самого начала все было нормально, поэтому я не изменял локаль.

## 9.6. ПРЕОБРАЗОВАНИЕ СТРОК

### 9.6.1. Функции удаления пробелов

В общем случае абсолютно невозможно предугадать, что может ввести в поле ввода пользователь. Иногда даже сам пользователь не может объяснить, почему он ввел это. Однако сценарий, если он создан добротно, должен уметь разобраться в предоставленных ему данных. Или по крайней

мере попытаться это сделать. Специально для этого в PHP предусмотрен целый ряд функций.

После приема параметров, прежде чем приступить к их обработке, нужно привести значения параметров к «товарному виду». Прежде всего нужно удалить лишние пробелы. Функция `trim()` удаляет пробельные символы в начале и в конце строки, например:

```
$s = "   string \n  ";
$s = trim($s);     // $s = "string"
```

Напомню, что пробельными символами считаются символы " ",\n,\r,\t.

Кроме функции `trim()`, можно использовать функции `ltrim()` и `chop()`. Первая удаляет пробельные символы только в начале строки, а вторая — только в ее конце.

### 9.6.2. URL-кодирование и декодирование

Очень полезными являются функции Url-кодирования и декодирования данных. Кодирование необходимо, если вам нужно передать через Интернет данные, содержащие специальные символы или символы национальных алфавитов (например, русского или украинского). Если вы собираетесь передавать параметр сценария, содержащий такие символы, то его необходимо подвергнуть Url-кодированию, чтобы избежать проблем, возникающих из-за возможного отсутствия поддержки 8-битных кодировок промежуточными серверами.

Для кодирования данных служит функция `UrlEncode()`, а для декодирования – функция `UrlDecode()`. Пример их совместной работы приведен в листинге 9.5.

**Листинг 9.5. Демонстрация совместной работы функций UrlEncode() и UrlDecode**

```
<?
$Param = "Русский   текст";
$Param = UrlEncode($Param);
echo "$Param\n";
$Param = UrlDecode($Param);
echo "$Param\n";
?>
```

Строка «Русский текст» (рис. 9.3) была преобразована в загадочную последовательность символов, причем пробел был заменен символом «+». В таком виде строка готова к передаче по Всемирной паутине. Функция



http://all-ebooks.com

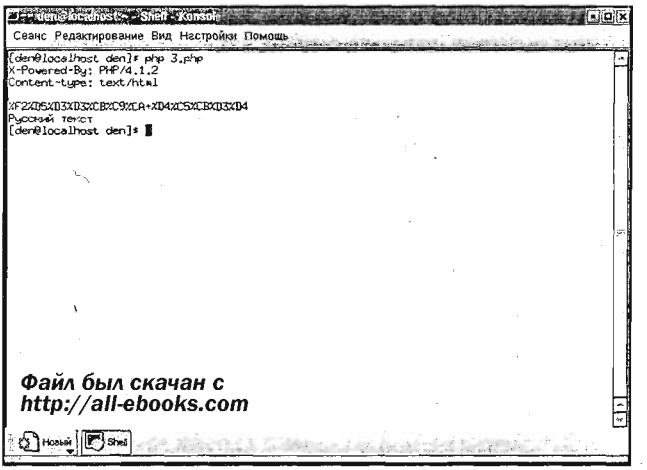



*Рис. 9.3. Функции UrlEncode() и UrlDecode()*

`UrlDecode()` безболезненно преобразовала эту последовательность обратно в исходную строку. Напомню, что кодирования требуют только параметры сценария, если они содержат специальные символы или символы национальных алфавитов.

Если вы хотите передать целый файл, его также желательно закодировать с помощью функции `UrlEncode()`, а потом раскодировать с помощью `UrlDecode()`, чтобы не вышло недоразумений.

А вот пример автоматического формирования ссылки `<a href>`:

```
$Param = "Русский текст";
echo "<a href=http://localhost/
script.php?parametrs=".UrlEncode($Param).">Click here</a>";
```

Кроме функций `UrlEncode()` и `UrlDecode()`, вы можете использовать функции `RawUrlEncode()` и `RawUrlDecode()`. Эти функции работают аналогично предыдущим, но не воспринимают символ «+» как пробел.

## 9.6.3. Экранирование

Вы знаете, что для вывода символа кавычек или обратной косой черты нужно экранировать их обратной косой чертой (обратным слэшем):

```
$S = "\\ \"'";
echo $S;
```

Рано или поздно вам будет нужно добавить перед указанными символами (или же, наоборот, удалить) лишние слэши. Для добавления слэшей используется функция `AddSlashes(string $str)`, а для удаления —

`StripSlashes(string $str)`. Обе функции возвращают модифицированную строку и не изменяют исходную.

### 9.6.4. Замена специальных символов на их HTML-эквиваленты

Функция `HtmlSpecialChars(string $str)` заменяет специальные символы (кавычки, «больше», «меньше» и другие) на их HTML-эквиваленты. Делается это для того, чтобы они на Web-странице выглядели самими собой.

Например, при использовании функции `HtmlSpecialChars()` знак «больше» в коде будет заменен строкой `&gt`, а кавычки — `&quot`. Пример использования `HtmlSpecialChars()` приведен в листинге 9.6.

**Листинг 9.6. Пример использования HtmlSpecialChars()**

```
<?
$S = "& < > \"" ;
$S =  HtmlSpecialChars($S);
echo $S;                    // выведет &amp; &lt; &gt; &quot;
?>
```

## 9.7. ПРЕОБРАЗОВАНИЕ КОДИРОВОК

Начиная с восьмидесятых годов «добрые» американцы решили, что у нас не хватит ума, чтобы написать русификатор знакового генератора компьютера. И они начали придумывать различные кодировки для русского языка. Одна только Microsoft изобрела две кодировки: CP866 (она же альтернативная) и Windows-1251 (она же ANSI). Компания Apple тоже решила не отставать и разработала кодировку x-mac-Cyrillic, которая используется в ОС MacOS. Помню, мне пришло письмо в этой кодировке, так я долго не мог его прочитать, пока не вспомнил о существовании MacOS.

Вдобавок ко всему ISO выпустила свою кодировку для русского языка — ISO8859-5. К сожалению, американцы забыли, что у нас есть своя родная кодировка — KOI8-R. Используя эту кодировку в DOS, Билл Гейтс в свое время действительно облегчил жизнь русскопишущему народу.

В результате на сегодня ситуация выглядит так: мой Linux, соседский Windows и еще чей-то DOS поддерживают русский язык... ровно до тех пор,



http://all-ebooks.com

пока нам не понадобится обменяться файлами. К счастью, острота проблемы частично снята тем, что почти все почтовые клиенты поддерживают все вышеперечисленные кодировки, а также существует масса программ, выполняющих преобразование кодировок.

В PHP есть одна очень полезная функция, преобразующая русский текст из одной кодировки в другую — `convert_cyr_string()`, которой нужно передать три параметра:

- исходную строку;
- исходную кодировку;
- требуемую кодировку.

Последние два параметра могут принимать следующие значения:

- k — KOI8-R;
- w — windows-1251;
- d или a — x-cp866;
- m — x-mac-cyrillic;
- i — iso8859-5.

Пример использования функции `convert_cyr_string()` приведен в листинге 9.7.

**Листинг 9.7. Пример использования функции convert_cyr_string()**

```
<?

// Преобразование слова "Привет" из кодировки koi8-r в windows-1251
echo convert_cyr_string("Привет", "k" , "w");        // выведет
"оПХБЕР"

?>
```

## 9.8. СПЕЦИАЛЬНЫЕ ФУНКЦИИ

### 9.8.1. Хэш-функции

В этом пункте мы рассмотрим две функции — `md5()` и `crc32()`.

Функция `md5()` кодирует строку, используя алгоритм MD5 (Message Digest Algorithm). Если быть предельно точным, то функция `md5()` возвращает хэш-код переданной строки.

Чем же хорош алгоритм MD5? Вероятность того, что две разные строки будут иметь одинаковый хэш-код, близка к нулю, а это позволяет эффективно использовать данную функцию для шифрования паролей. А также хорош тем, что для MD5 не существует алгоритма дешифровки, то есть зашифрованное с его помощью сообщение невозможно расшифровать. Для проверки подлинности пароля нужно зашифровать пароль заново и сравнить зашифрованные строки: если они равны, значит, пароль правильный (см. листинг 9.8).

**Листинг 9.8. Использование хэширования**

```
<?
 $a = "my_password";
 $b = "my_password";
 $c = "password";

if  (md5($a)===md5($b)) echo "1: пароль правильный";
if  (md5($a)===md5($c)) echo "2: пароль правильный";
?>
```

Сценарий выведет только первое сообщение, поскольку строки (значит, и их хэш-коды) равны. Вообще-то использовать эту функцию нужно немного не так:

- Пароль не должен храниться в сценарии, тем более в открытом виде. Пароль нужно хранить в зашифрованном виде в базе данных или в каком-нибудь файле на диске.
- Введенный пользователем пароль должен зашифровываться и сравниваться уже с зашифрованной строкой, которая прочитана из файла или БД.

Кроме функции `md5()`, вы можете использовать функцию `crypt()`, которая реализует алгоритм DES. Этот алгоритм не так эффективен, как MD5, поэтому его не рекомендуется использовать в серьезном проекте.

Функция `crc32()` используется для вычисления 32-битной контрольной суммы строки. Эта функция может вам пригодиться для регистрации какого-нибудь программного продукта через Internet, хотя я порекомендовал бы написать собственную функцию для этой цели, потому что `crc32()` — стандартная функция, и ее может использовать кто угодно.

```
<?
$SN = "012103343";
echo crc32($SN);
?>
```



http://all-ebooks.com

### 9.8.2. Функция serialize()

Допустим, что мы разрабатываем систему контроля знаний (попросту говоря, тест). Нам нужно предусмотреть продолжение теста с того места, где он был прерван.

Мы уже знакомы с технологией Cookies, поэтому знаем, как сохранить текущий номер вопроса, начиная с которого тест будет продолжен. А как запомнить номера вопросов (массив), которые были сгенерированы для студента? Сохранить массив в Cookies мы не можем, зато можно сохранить строку.

Функция `serialize()` позволяет преобразовать любой объект в строку, ею мы и воспользуемся для сохранения нашего массива в Cookies. Обратите внимание: строка, полученная в результате работы `serializeQ`, имеет специальный формат. Чтобы восстановить из этой строки объект обратно, используйте функцию `unserialize()`.

Вернемся к нашему тесту:

- Мы генерируем список номеров вопросов, которые сериализуем и помещаем в Cookies.
- В программе работаем с обыкновенным массивом, а в случае разрыва соединения (при повторном запуске) получаем массив из Cookies.

Как это реализуется программно, показано в листинге 9.9.

**Листинг 9.9. Демонстрация использования функции Serialize()**

```
<?

if (!isset($Questions))
{

// Cookies не установлены - первый запуск
for($i=0;$i<10;$i++) $Arr[]=$i;

// Преобразуем массив в строку
$Serialized_arr = serialize($Arr);

// Устанавливаем Cookies
setcookie("Questions",$Serialized_arr,time()+3600);

unset($Arr); // массив $Arr больше не существует
}
else
{
```

```
// Cookies установлены - "разворачиваем" массив

$Arr = unserialize($Questions);

foreach($Arr as $v) echo "$v ";

}
?>
```

В окне браузера при втором запуске данный сценарий выведет массив Arr:
0 1 2 3 4 5 6 7 8 9

### 9.8.3. Функция explode(): разбиение на подстроки

Функция `explode()` очень часто используется в реальных проектах, и мы в этом сейчас убедимся. Если вкратце, то эта функция предназначена для разделения строки на подстроки. Синтаксис ее такой:

```
array explode (string $separator, string $str [, int $limit])
```

В результате `explode()` возвращает массив строк. Каждая строка массива – это подстрока строки `str`. Строки формируются путем разбиения строки `str` по границам, указанным строкой `$separator`.

Например, у нас есть строка следующего формата:

```
user:password:home_dir:shell
```

Мы вызываем `explode()` и передаем ей эту строку и строку-разделитель, то есть двоеточие «:». В результате получаем массив строк. Элементами этого массива будут строки «user», «password», «home», «shell».

Параметр `$limit` используется довольно редко, но все же вам не помешает знать, для чего он используется. Если `$limit` задан, то возвращаемый массив будет содержать не более `$limit` элементов, причём неразобранный остаток строки `$str` помещается в последний элемент:

```
$A = explode(":","user:password:home_dir:shell", 3);
```

Массив `$A` будет содержать элементы: «user», «password» и «home_dir:shell». Последний элемент – это остаток исходной строки.

Я говорил о реальных проектах. Сейчас мы рассмотрим один из них. Предположим, что вы стали партнером Линукс-Центра или другого интернет-магазина. Это не реклама Линукс-цеитра, а просто пример из реальной жизни.

Получить категории товаров можно с помощью ссылки:



http://all-ebooks.com

```
http://www.linuxcenter.ru/trans/export_types.phtml?sid=stjcvd25
```

Но если ввести этот URL в адресную строку браузера, то получим мы вот что (рис. 9.4).

Согласитесь, если вывести такой блок текста на основную страницу, то много денег за участие в партнерской программе мы не заработаем. Поэтому нужно привести все это к «товарному» виду.

Сначала нужно разобраться с форматом файла. Формат файла описан на сайте самого интернет-магазина, но, даже глядя на вывод того сценария, все можно и так понять:

```
номер_категории_1\tссылка_на_категорию_товаров_1\tназвание_
категории_товаров_1\n
...
номер_категории_m\tссылка_на_категорию_товаров_m\tназвание_
категории_товаров_m\n
```

Каждую строку нам нужно превратить в ссылку:

```
<a href=ссылка_на_категорию_категорию>название_категории_товаров</a>
```

Массив строк мы получим с помощью функции `file()`, которую подробнее рассмотрим в главе 11, а пока сосредоточимся на функции `explode()`. Эта функция разделит каждую строку вывода на три подстроки: номер, ссылку

Файл был скачан с http://all-ebooks.com

***Рис. 9.4. Категории интернет-магазина***

и название. Номер мы выводить на свою страницу не будем, а вот ссылка и категория – это то, с помощью чего можно заработать немного денег.

В качестве разделителя будет выступать символ табуляции `\t`.

Пример кода приведен в листинге 9.10.

***Файл был скачан с http://all-ebooks.com***

**Листинг 9.10. Вывод категорий интернет-магазина**

```
$t = file('http://www.linuxcenter.ru/trans/export_types.phtml?sid=stjcvd25');
foreach($t as $s)
{

// преобразуем полученный массив в обычные переменные
// $num - номер категории
// $link - ссылка
// $name - название категории
list($num, $link, $name) = explode("\t",$s);
```

***Рис. 9.5. Вывод категорий товаров на сайте http://dkws.org.ua***



http://all-ebooks.com

```
echo "<br><a href=$link>$name</a>";
}
```

На рис. 9.5 вы можете увидеть, что из этого вышло.

# ГЛАВА 10.

# ПОЛЕЗНЫЕ СТАНДАРТНЫЕ ФУНКЦИИ

http://all-ebooks.com

## 10.1. РАБОТА С ДАТОЙ И ВРЕМЕНЕМ

Практически ни один серьезный проект не обходится без обработки дат и времен, поэтому не уделить внимание функции `date()` просто невозможно.

Функция `date(string $format [, int $timestamp])` принимает два параметра:

- формат даты/времени (строка);
- сама дата (или время). Это целое число, представляющее собой количество секунд, прошедших с начала эры Unix – с полуночи (по Гринвичу) 1 января 1970 года.

Функция форматирует указанную временную метку (либо текущую дату, если параметр `$timestamp` не задан) согласно указанной строке формата и возвращает полученную строку. Строка формата может содержать обычный текст, а также модификаторы (см. табл. 10.1), которые при выводе заменяются соответствующими значениями.

**Таблица 10.1**. *Модификаторы формата функции date()*

| Модификатор | Описание |
|---|---|
| U | Количество секунд, прошедших с начала эры Unix |
| Y | Год (4 цифры) |
| y | Год (2 цифры) |
| z | Номер дня от начала года (от 0 до 365) |
| F | Полное английское название месяца («December») |

**Таблица 10.1.** *Модификаторы формата функции date() (продолжение)*

| | |
|---|---|
| m | Номер месяца (2 цифры: от «01» до «12») |
| n | Номер месяца без предваряющего нуля: от «1» до «12» |
| M | Трехбуквенная английская аббревиатура месяца («Dec») |
| d | Номер дня в месяце (2 цифры: от «01» до «31») |
| j | Номер дня в месяце без предваряющего нуля: от «1» до «31» |
| l | Английское название дня недели («Monday») |
| w | Номер дня недели («0» - воскресенье, «1» - понедельник и т. д.) |
| D | Трехбуквенная английская аббревиатура дня недели («Mon») |
| A | До или после полудня: «AM» или «PM» |
| a | До или после полудня: «am» или «pm» |
| H | Часы (24-часовой формат) от «00» до «23» |
| h | Часы (12-часовой формат) от «01» до «12» |
| i | Минуты, от «00» до «59» |
| s | Секунды, от «00» до «59» |

Примеры использования функции `date()`:

```
echo date("Сегодня d.m.Y");
echo date('Дата создания файла d.m.Y',filectime ('test.php'));
```

При обработке введенных параметров очень полезной окажется функция `checkdate()`, которая проверяет корректность введенной даты. Функции нужно передать три параметра — номер месяца, число (день) и год (именно в такой последовательности). Дата считается корректной, если:

- год находится в диапазоне от 1900 до 32767;
- номер месяца находится в диапазоне от 1 до 12;
- число находится в диапазоне от 1 до количества дней в указанном месяце (функция учитывает, високосный год или нет).

В PHP есть еще несколько функций для работы с датой и временем, но пока рассмотренных функций вполне достаточно.

## 10.2. ГЕНЕРАТОР СЛУЧАЙНЫХ ЧИСЕЛ. ВЫВОД БАННЕРОВ СЛУЧАЙНЫМ ОБРАЗОМ

Чаще всего генерация случайных чисел нужна для отображения баннеров или криптографии. Для генерации случайных чисел служат функции `mt_rand()` и `mt_srand()`.



http://all-ebooks.com

Функция `mt_rand(int $from, int $to=RAND_MAX)` возвращает случайное число в диапазоне от `$from` до `$to`. Если второй параметр не задан, используется значение по умолчанию — константа RANDMAX. Значение данной константы можно узнать с помощью функции `mt_getrandmax()`.

В качестве примера сгенерируем 10 случайных чисел от 0 до 100:

```
for ($i=0; $i<10; $i++) echo mt_rand(0, 100)." ";
```

Сценарий выведет 10 случайных чисел. Запустите сценарий еще раз. И еще раз. Как видите, каждый раз сценарий выводит одну и ту же последовательность. Для настройки генератора случайных чисел на новую последовательность используется функция `mt_srand()`. С ней мы уже сталкивались, когда изучали перемешивание массивов в п. 7.6.5.

Функции `mt_srand()` нужно передать единственный параметр, на основании которого она настроит генератор на новую последовательность. В качестве параметра можно передать значение функции `time()`, которое возвращает текущее время в виде количества секунд, прошедших с начала эры Unix:

```
mt_srand(time());
```

В начале этого пункта я неосторожно обмолвился о баннерах. В одной из своих книг я описал, как с помощью SQUID (прокси-сервер) «вырезать» все баннеры. В этой книге, чтобы восстановить справедливость, я напишу сценарий, который выбирает случайный баннер и вставляет его в Web-страницу.

Предположим, что у нас есть файл `banner.txt`, в котором содержатся имена картинок-баннеров, которые нужно вставить в страницу. Подробнее

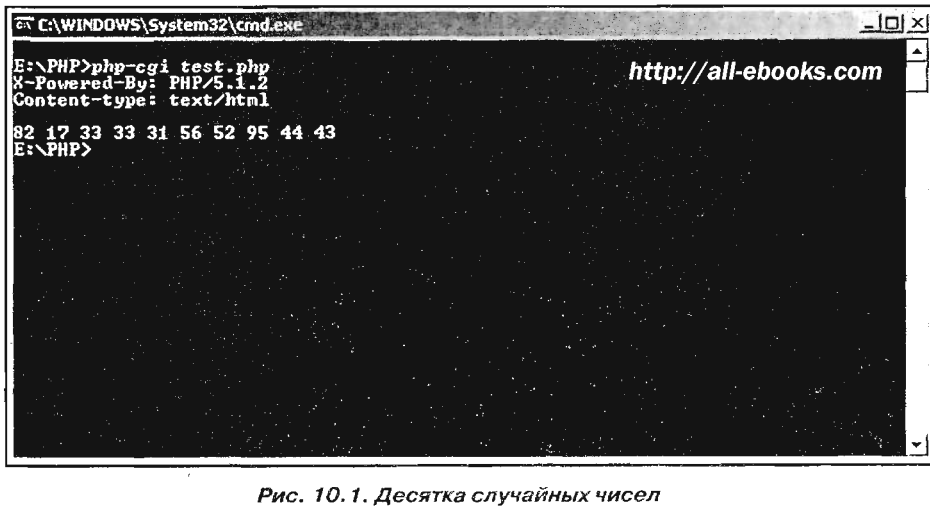



*Рис. 10.1. Десятка случайных чисел*

http://all-ebooks.com

о файлах мы поговорим в следующей главе, а сейчас не будем останавливаться на функциях для работы с файлами.

```
$Banners=file("banner.txt");
$k = mt_rand(0,99);
mt_srand(time()+(double)microtime()*$k);
$i = mt_rand(0, (count($Banners)-1));

echo "<IMG SRC=$Banners[$i]>";
```

Сначала мы получаем список строк, содержащихся в файле `banner.txt`. Затем настраиваем генератор на новую последовательность, получаем случайный индекс `$i` списка `$Banners` и выводим тег IMG. В качестве имени файла картинки (SRC) будет элемент списка `$Banners`, выбранный случайным образом.

## 10.3. МАТЕМАТИЧЕСКИЕ ФУНКЦИИ

Помните, мы писали функции для поиска минимального (или максимального) элемента массива? Зря мы это делали — в PHP есть стандартные функции: `min()` и `max()`.

Функция `max()` возвращает максимальный из переданных ей n аргументов. Например,

```
echo max(7 ,10 ,3 ,1, $a, $b)
```

выведет 10, если значения `$a` и `$b` меньше или равны 10.
Если функции передан один параметр, то он должен быть массивом, в котором будет произведен поиск максимального элемента.
Функция `min()` работает аналогично функции `max()`, но ищет минимальный аргумент (или элемент массива).

Математические функции PHP в особых примерах и объяснениях не нуждаются, поэтому удобнее всего представить их в виде таблицы (см. табл. 10.2).
Надеюсь, что эти функции пригодятся вам при программировании сценариев.

**Таблица 10.2.** *Математические функции PHP*

| Функция | Назначение |
|---|---|
| acos(float $x) | Вычисляет арккосинус аргумента $x |
| asin(float $x) | Вычисляет арксинус аргумента $x |



http://all-ebooks.com

**Таблица 10.2.** *Математические функции PHP (продолжение)*

| | |
|---|---|
| atan(float $x) | Вычисляет арктангенс аргумента $x |
| atan2(float $x, float $y) | Вычисляет арктангенс числа $x/$y и возвращает результат в радианах |
| sin(float $x) | Вычисляет синус аргумента $x. Аргумент задается в радианах |
| cos(float $x) | Вычисляет косинус аргумента $x. Аргумент задается в радианах |
| tan(float $x) | Вычисляет тангенс аргумента $x. Аргумент задается в радианах |
| pi() | Возвращает число Пи. Вместо функции pi() удобнее использовать константу M_PI |
| sqrt(float $x) | Возвращает квадратный корень $x. Напомню, что $x должен быть положительным |
| log(float $x) | Возвращает натуральный логарифм $x |
| exp(float $x) | Возвращает экспоненту аргумента (2,71828182845 в степени $x) |
| pow(float $x, $a) | Возвращает $x в степени $a |

Результат, возвращаемый всеми перечисленными в табл. 10.2 функциями, имеет тип `float`. Только число pi возвращается с типом `double`.

## 10.4. ПРЕОБРАЗОВАНИЯ СИСТЕМ СЧИСЛЕНИЯ

Эти функции не очень нужны при программировании сценариев, но, возможно, они вам пригодятся при работе на низком уровне с бинарными файлами.

Самая универсальная функция для преобразования кодировок — это функция `base_convert()`. Ей нужно передать три параметра:

- заданное в виде строки число, которое нужно преобразовать;
- основание системы счисления, в которой число представлено в данный момент;
- основание системы счисления, в которую нужно преобразовать число.

Последние два параметра могут принимать значения от 2 до 36 включительно. Первый параметр может содержать как цифры, так и латинские буквы. При этом цифры обозначают сами себя, буква «a» обозначает 10, буква «b» обозначает 11, и так далее до буквы «z».

Пример:

```
echo base_convert("8",10,2);      // выведет 1000
echo base_convert("F",16,10);     // выведет 15
echo base_convert("F",16,2);      // выведет 1111
```

Следующие две функции используются для преобразования числа из двоичной системы в десятичную и наоборот:

- `bindec()` — переводит двоичное число в десятичное;
- `decbin()` — переводит десятичное число в двоичное.

Пример:

```
echo bindec("1111");        // выведет 15
echo decbin("15");          // выведет 1111
```

Существует также функции для преобразования чисел из шестнадцатеричной и восьмеричной систем в десятичную и обратно:

- `hexdec()` — преобразует число из шестнадцатеричной системы в десятичную;
- `dechex()` — преобразует число из десятичной системы в шестнадцатеричную;
- `octdec()` — преобразует число из восьмеричной системы в десятичную;
- `decoct()` — преобразует число из десятичной системы в восьмеричную.

## 10.5. ЗАПУСК ВНЕШНИХ ПРОГРАММ

Большинство задач не требуют вызова внешних программ, поскольку без этого можно обойтись, используя стандартные функции PHP. Однако иногда удобнее вызвать внешнюю программу, чем использовать функции PHP. Самый простой пример — вывод оглавления каталога. Можно использовать функции PHP чтения каталога и его текущего элемента, которые будут рассматриваться в следующей главе, а можно написать всего лишь два оператора:

```
$dir = `ls`;                    // вывод оглавления текущего каталога
echo "<pre> $dir </pre>";
```

Для вызова внешней программы (в данном случае это Linux-команда `ls`, которая выводит содержимое каталога) заключите команду вызова в обратные апострофы.

> **Примечание.**
>
> Присваивание $dir =`ls` не сработает, если включен так называемый безопасный режим. Для его отключения установите параметр Safe_Mode=Off в файле /etc/php.ini.



http://all-ebooks.com

Команда `ls` слишком тривиальна. Рассмотрим примеры более полезных команд (в ОС Unix):

- `ps -ax` — выводит список всех запущенных процессов;
- `ps -ax | grep httpd` — информация о процессе httpd (сервер Apache);
- `ps -u den` — выводит список всех процессов, владельцем которых является пользователь den;
- `free` — информация о загрузке памяти;
- `df` — информация о загрузке дисковой памяти.

Все эти команды будут работать в ОС Linux (или другой Unix-подобной ОС).

Теперь рассмотрим стандартные функции PHP по запуску внешних программ. Если вы хотите, чтобы вывод программы был сразу перенаправлен в браузер, воспользуйтесь функцией `system()`:

```
system(string $command [, int $return_var]);
```

В качестве первого параметра этой функции передается имя запускаемой программы (команды). Второй параметр необязателен и обычно не задается. Однако если он задан (это обязательно должна быть переменная, а не константа), то в него помещается код возврата вызванной программы.

Если вы не хотите, чтобы весь вывод программы перенаправлялся в браузер, можно использовать функцию `exec(string $command [, list $array] [, int $return_var])`. Ей нужно передать три параметра:

- имя внешней программы (команды) — `$command`;
- массив — `$array`;
- переменную, в которую будет записан код возврата внешней программы — `$return_var`.

Последние два параметра необязательны, но очень желательны. Если вы не укажете массив, то функция передаст в основную программу только последнюю строку, выведенную программой. Если массив задан, то в него будет помещен вывод программы — по одной строке в каждый элемент массива. Если не будет указан третий параметр, то не узнаете, как завершилась программа — успешно или с ошибкой.

Стоит отметить, что функция `exec()`, как и `system()`, дожидается завершения программы и только потом передает управление основной программе. Это значит, что использование данных функций для больших программ недопустимо, иначе пользователь подумает, что сценарий завис.

http://all-ebooks.com

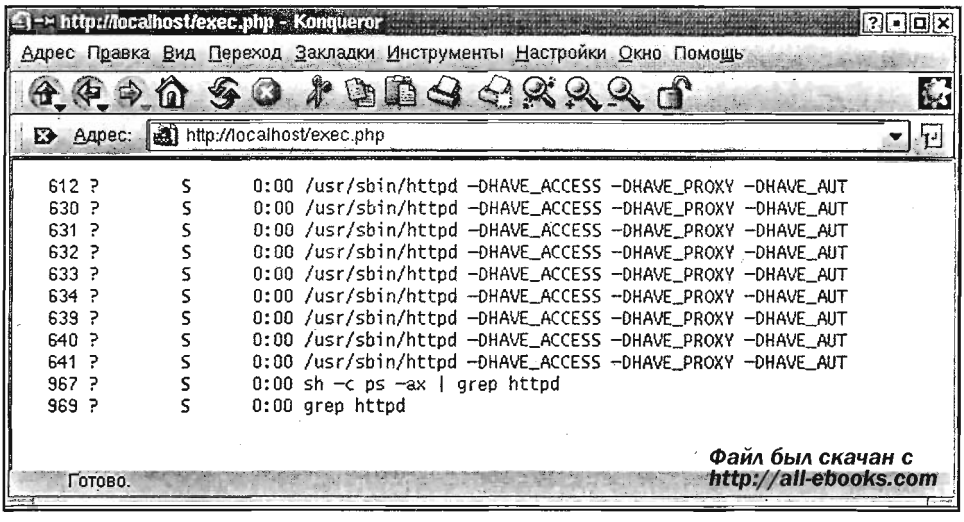



***Рис. 10.2. Вывод команды ps -ax | grep httpd***

Существует очень полезная функция `PassThru(string $command [, int $return_var])`, которая запускает команду и сразу же перенаправляет весь ее вывод «один в один» в браузер. Эта функция могла бы понадобиться для вывода в браузер картинок, если бы в PHP не было встроенных функций для этой цели. Тем не менее, если ваш PHP почему-то не поддерживает работу с графикой, вы можете воспользоваться этой функцией для вывода рисунков:

```
Header("Content-type: image/png");
PassThru ("cat picture.png");
```

Использовать вызов внешних программ нужно очень осторожно. По возможности старайтесь все же использовать стандартные функции PHP. Также старайтесь избегать вызова команд, введенных пользователем. Это значит, что вот так делать категорически не рекомендуется:

```
system($from_user)
```

(здесь переменная `$from_user` была получена из формы или Cookies). Ведь пользователь может ввести все, что угодно! Например, команда `cat /etc/passwd` выведет файл `passwd`, в котором содержатся данные об учетных записях пользователей системы (рис. 10.4). Конечно, при правильно настроенных правах доступа пользователь не получит доступа к файлу `/etc/shadow`, в котором хранятся пароли, но сообщать ему имена всех зарегистрированных в вашей системе пользователей тоже небезопасно.



http://all-ebooks.com

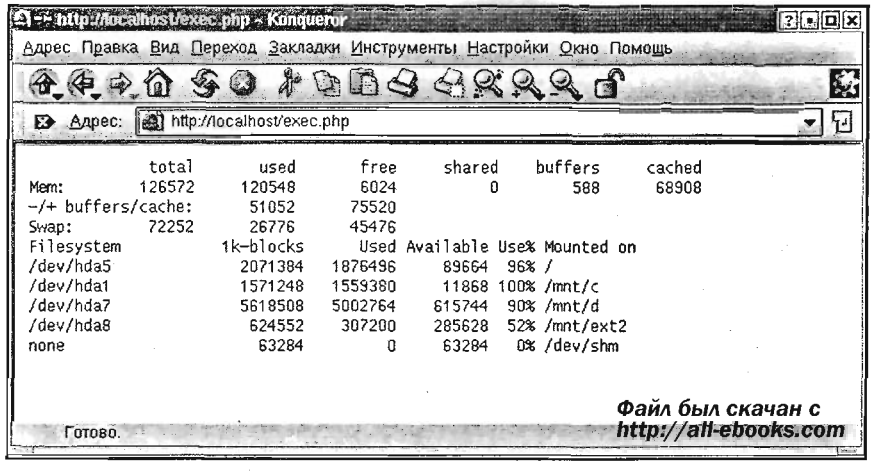



***Рис. 10.3. Вывод команд free; df***

Если нужно все-таки обеспечить выполнение команд пользователя, например, для смены каталога внутри домашнего каталога пользователя, используйте функцию `escapeshellcmd()`, которая экранирует все специальные символы слэшами. Например, вы хотите обеспечить смену каталога пользователя:

```
system("cd $dir");
```

Переменная `$dir` получена от пользователя. Если пользователь введет в качестве имени каталога строку «/; rm -R *», то он сможет удалить все файлы вашего сервера, если, конечно, сервер неправильно настроен.

На рис. 10.5 изображен результат работы функции `escapeshellcmd()` — первая строка до вызова функции, а вторая после. Специальные символы были предварены защитными слэшами — даже если пользователь введет такую разрушающую команду, у него ничего не выйдет.

**Примечание.**

Наверное, вас заинтересовал вполне уместный вопрос: а как пользователь может изменить каталог (то есть выполнить команду cd /home/usr/work), если все слэши блокируются? А для этого используется функция chdir(), которую мы рассмотрим позже.

Некоторые администраторы запускают сервер Apache от имени пользователя root. Это неправильно. В этом случае уж точно любой желающий сможет «убить» ваш сервер. Надо запускать сервер от имени пользователя apache или бесправного пользователя nobody. В этом случае пользователь не мо-

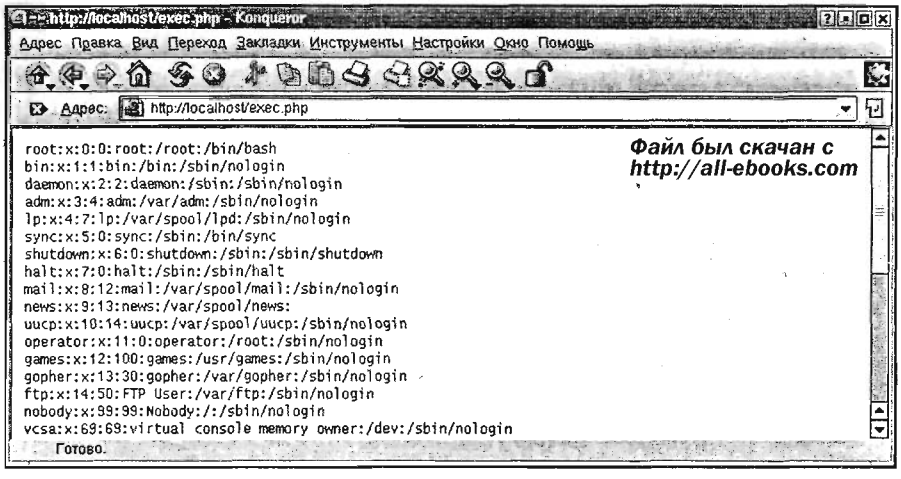



*Рис. 10.4. Вывод /etc/passwd*

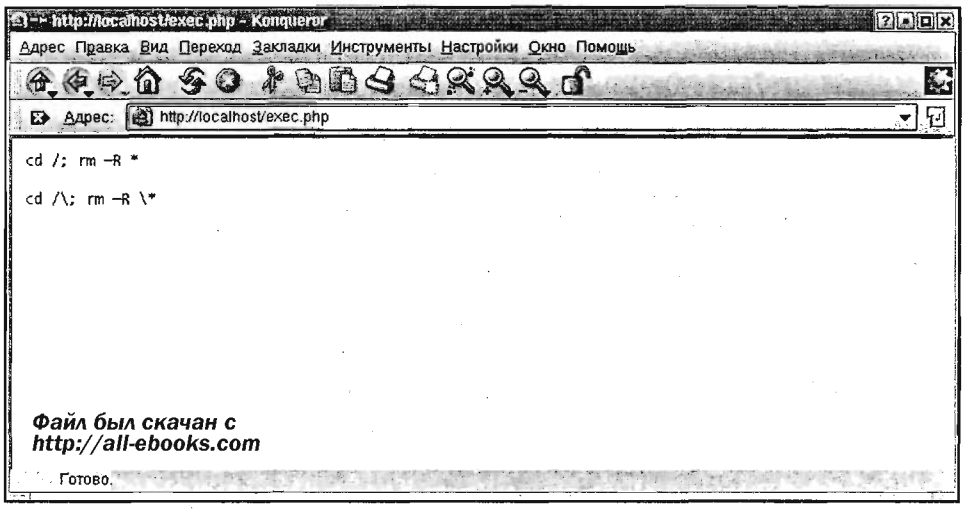



*Рис. 10.5. Результат работы escapeshellcmd()*

жет выполнить никаких разрушающих действий. Если администратором являетесь не вы, и проверить, от чьего имени запускается Apache, вы тоже не можете, обязательно используйте функцию `escapeshellcmd()` — это прежде всего в ваших интересах.Ведь восстановить систему — это дело двух-трех часов, а восстановить данные пользователей (ваши в том числе) порой вообще невозможно.



http://all-ebooks.com

## 10.6. НЕКОТОРЫЕ ИНТЕРЕСНЫЕ ФУНКЦИИ PHP

### 10.6.1. Отображение исходного кода. Функция show_source()

Предположим, что вы создали какой-то информационный ресурс, посвященный программированию на PHP. На вашем ресурсе, ясное дело, есть статьи, содержащие исходный код PHP. Да вот проблемка. Если просто сделать ссылку на php-файл, то он будет интерпретирован, то есть посетитель сайта увидит результат выполнения сценария, а не его исходный код. Неужели придется копировать исходный код в файл с расширением HTML, а потом его выводить? Но тогда возникает проблема с трансляцией специальных символов – тех же угловых скобок. К тому же хотелось бы оформить исходный код грамотно, с подсветкой синтаксиса.

Оказывается, разработчики PHP о вас уже позаботились. Они создали специально для вас функцию `show_source()`, выводящую исходный код сценария, указанного в качестве параметра. Причем функция не просто выводит исходный код, а делает это с подсветкой синтаксиса:

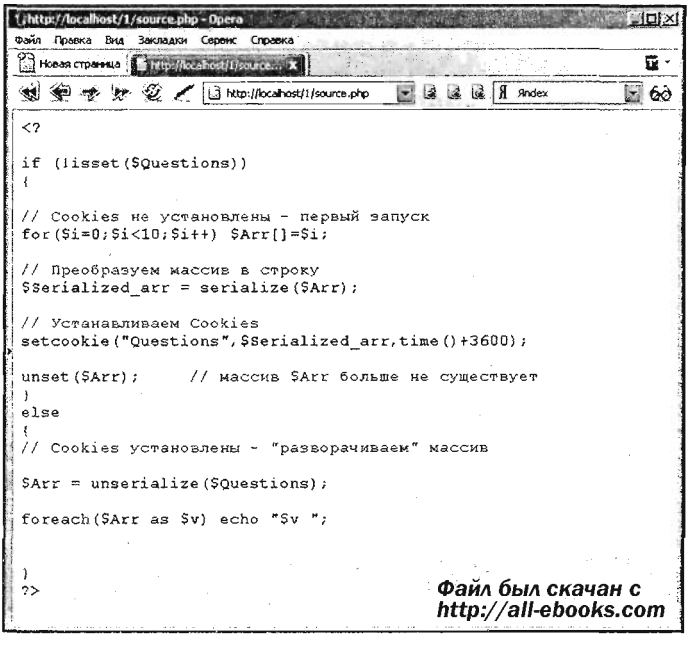



*Рис. 10.6. Вывод исходного текста сценария с подсветкой синтаксиса*

```
<?

// выводим исходный код сценария test.php
show_source("test.php");

?>
```

Результат работы функции представлен на рис. 10.6. Удобно? Я тоже так думаю.

### 10.6.2. Функция sleep(): поспим немного

Эта функция позволяет приостановить выполнение сценария на указанное количество секунд. Нужное количество секунд передается в качестве параметра функции `sleep()`:

```
sleep(2);     // "поспать" 2 секунды
```

При использовании этой функции помните о максимальном времени выполнения сценария: как бы не перестараться, иначе ваш сценарий «заснет», но уже больше не «проснется».



http://all-ebooks.com

# ГЛАВА 11.

# РАБОТА С ФАЙЛАМИ И КАТАЛОГАМИ

http://all-ebooks.com

## 11.1. ОБЩИЕ ЗАМЕЧАНИЯ

Прежде чем приступить к работе с файлами, нужно рассмотреть особенности работы с файлами при программировании на PHP и вообще под Unix. Даже если вы сейчас программируете под Windows, вам просто необходимо знать эти тонкости, поскольку у хостинг-провайдера обычно установлена Unix-подобная операционная система и ваша программа, нормально работающая под Windows, может не работать в Unix.

В отличие от Windows операционная система Linux чувствительна к регистру имени файла. Это означает, что файлы `A.txt` и `a.txt` — совершенно разные файлы. В операционной системе Linux полный путь к файлу записывается с помощью прямых слэшей (/), а не обратных (\), как в Windows. Но об этом вы можете не беспокоиться: для интерпретатора нет разницы, какие слэши вы используете — лишь бы только файл существовал и путь к нему был указан правильно.

Интерпретатор PHP позволяет работать не только с локальными файлами, но и с сетевыми соединениями. Например, вы можете открыть файл `http://www.dkws.org.ua/index.php` и работать с ним как с файлом, то есть прочитать, модифицировать и вывести в окно браузера. Точно так же вы можете работать и по протоколу FTP: просто укажите префикс протокола `ftp://`. Если у вас есть права на запись удаленных файлов, вы можете даже модифицировать эти файлы.

Самое главное отличие Windows-файлов от Unix-файлов проявляется при работе с файлом в текстовом режиме, когда данные читаются и записываются построчно. В Windows конец строки обозначается двумя символами:



http://all-ebooks.com

\r (возврат каретки) и \n (перевод строки). В Unix для этой цели служит единственный символ \n. Вот как один и тот же файл будет выглядеть в разных операционных системах:

| Linux | Windows | MacOS |
|---|---|---|
| Line1\n | Line1\r\n | Line1\n\r |
| Line2\n | Line2\r\n | Line2\n\r |
| Line3 | Line3 | Line3 |

Интерпретатор PHP работает в так называемом режиме трансляции символов новой строки. Если вы работаете в Linux и открываете файл в формате Windows, последовательность символов \r\n будет преобразована в один символ \n. Наоборот, если вы работаете в Windows и открываете Unix-файл, символ \n будет предварен символом \r. То же самое происходит при записи файлов. Режим *трансляции символов* новой строки еще называют *текстовым режимом* работы. Если отключить этот режим, интерпретатор будет работать в *бинарном режиме*.

В бинарном режиме никакой трансляции символов не производится, поэтому вы можете свободно работать с нетекстовыми файлами, например картинками, архивами и другими двоичными файлами.

Для чего я все это говорил — ведь для программиста преобразование символов происходит незаметно? Дело в том, что при чтении символы \r удаляются, а при записи на диск добавляются, следовательно, изменяется размер файла. Соответственно, изменяется и значение, возвращаемое функцией `filesize()`, которая выдает размер файла. Это значение будет не соответствовать размеру реально прочитанных данных. Если вы используете текстовый режим, то этот момент обязательно нужно учитывать.

В бинарном режиме вы можете делать с файлом все, что захочется. Например, можно полностью прочитать файл в строку, при этом данные никак не повредятся. Представьте такую ситуацию: вы открываете картинку в формате GIF в текстовом режиме. Это хоть и маловероятная процедура, но нам она позволит продемонстрировать, как подчас бывает необходим бинарный режим. Итак, вы открыли GIF-файл в текстовом режиме. Потом вы записываете содержимое файла в одну большую строку. Так как открытие файла произошло в текстовом режиме, из нашей строки «выпали» некоторые символы \r. Во-первых, длина нашей строки будет меньше (на количество символов \r) размера файла. А во-вторых, если мы потом попытаемся вывести эту строку (не забывайте — это же GIF-файл, но уже без символов \r) в браузер, то изображение будет искажено: ведь мы «вырезали» некоторые фрагменты изображения. Скорее всего, пользователь вообще ничего не уви-

http://all-ebooks.com

дит в окне браузера, потому что последний не сможет распознать формат GIF.

## 11.2. ОСНОВНЫЕ ОПЕРАЦИИ С ФАЙЛАМИ

Как и в любом языке программирования, работа с файлом осуществляется в три этапа:

- **Открытие файла.** Функция открытия файла возвращает целое число — идентификатор (дескриптор) файла. Дальнейшее обращение к файлу осуществляется через дескриптор, а не по имени.
- **Обработка файла** (чтение или запись).
- **Закрытие файла**.

### 11.2.1. Открытие файла

Функция `fopen($filename, $mode)` открывает файл (или URL) и возвращает его идентификатор. Эта функция принимает два параметра — имя файла (`$filename`) и режим (`$mode`), в котором необходимо открыть файл. Третий, необязательный, аргумент типа `bool` указывает, что делать, если `$filename` представляет собой относительное имя файла, а не полный путь: значение `true` требует использовать для поиска файла пути, указанные в `include_path`. Значение по умолчанию равно `false`.

В качестве имени файла можно указывать как абсолютный, так и относительный пути. При указании относительного пути используют каталоги «..» (две точки ) — родительский каталог и «.» (точка) — текущий каталог:

- `./array.txt` — файл находится в текущем каталоге;
- `../array.txt` — файл в родительском каталоге;
- `../command/array.txt` — файл в подкаталоге `command` родительского каталога.

Режимы открытия файла перечислены в табл. 11.1.

**Таблица 11.1.** *Режимы открытия файла*

| Режим | Назначение |
|---|---|
| r | Режим «только чтение». В этом режиме вы можете только читать файл, но не изменять его. После открытия указатель файла устанавливается на его начало (первый байт). Если указанного файла не существует, функция возвращает false |



http://all-ebooks.com

**Таблица 11.1.** *Режимы открытия файла (продолжение)*

| | |
|---|---|
| r+ | Файл открывается в режиме «чтение/запись», то есть вы можете не только читать, но и записывать данные в файл. После открытия файла указатель устанавливается на его начало. Как и в режиме «r», если файл не существует, функция возвращает false. ВАЖНО: если в момент записи указатель файла находится где-то в середине файла, то записываемые данные будут записаны поверх уже имеющихся, а не «вставлены» между ними |
| w | Будет создан новый пустой файл. Если на диске уже есть файл с таким именем, он будет перезаписан |
| w+ | Напоминает режим «r+», но со следующими отличиями:<br>— если файл не существует, он будет создан;<br>— если файл существует, он будет перезаписан.<br>Вы можете как читать, так и записывать данные в файл |
| a | Режим дополнения (append). Файл открывается в режиме записи, а указатель файла сдвигается в конец файла. Данные будут добавлены в конец файла. Если файл не существует, функция возвращает false |
| a+ | Самый универсальный режим. Файл открывается в режиме чтения и записи, при этом его содержимое не удаляется, а указатель файла перемещается на конец файла. Если файл не существует, он будет создан |
| b, t | После любого из вышеперечисленных режимов может следовать один из модификаторов «b» (открыть в бинарном режиме) или «t» (открыть в текстовом режиме). Если модификатор не указан, по умолчанию файл открывается в бинарном режиме |

## 11.2.2. Конструкция or die()

Функция `fopen()` возвращает `false`, если ей не удалось открыть файл: например, указанный файл не существует. Было бы удобно одним вызовом анализировать возвращаемое значение и, если оно равно `false`, завершать работу сценария с выводом подходящего сообщения. Существует конструкция, позволяющая это сделать:

```
<вызов_функции> or die("сообщение об ошибке");
```

Если вызов функции неуспешен (функция возвратила `false`), будет выведено сообщение об ошибке и работа сценария будет завершена.

Преимущество конструкции `or die()` заключается в том, что она, во-первых, позволяет писать более компактный код, а во-вторых, позволяет использовать себя вместе с присваиванием, так как приоритет оператора `or` ниже, чем оператора =.

Вот примеры открытия файлов:

http://all-ebooks.com

```
// Открываем файл /home/den/counter.txt в бинарном режиме
// "только чтение"
$f = fopen("/home/den/counter.txt","r") or die("Не могу открыть
файл\n");

// Открываем файл http://dkws.org.ua/index.php в текстовом
// режиме "только чтение"
$f = fopen("http://dkws.org.ua/index.php","rt") or
     die("Ошибка при открытии файла\n");

// Открываем FTP-соединение в бинарном режиме для записи
// (с указанием  логина и пароля)
$f = fopen("ftp://user:123456@ftp.narod.ru","r") or
     die("Ошибка при открытии файла\n");

// Открываем журнал для добавления сообщения в конец файла
// (текстовый режим)
$f = fopen("/home/den/counter.log","a+t") or
     die("Ошибка при открытии   файла\n");
```

**Примечание.**

Я рекомендую всегда использовать прямые слэши для указания полного имени файла даже в том случае, когда вы работаете в Windows. В PHP символ «\» требует экранирования, поэтому использовать его просто-напросто неудобно:

```
$f = fopen("c:\\home\\counter.txt","r");
```

Оператор `or` нельзя заменить логическим оператором ||, поскольку последний имеет более высокий приоритет. Рассмотрим два внешне похожих вызова:

```
$f = fopen("/home/den/counter.txt","r") or die("Ошибка!\n");
$f = fopen("/home/den/counter.txt","r" ) || die("Ошибка!\n");
```

В первом случае при отсутствии открываемого файла сценарий нормально завершит свою работу с выводом сообщения «Ошибка!». Во втором случае в этой же ситуации мы увидим ругательство интерпретатора:

```
<br>
<b>Warning</b>: fopen("/home/den/counter.txt", "r") - No such file
or directory in <b>/home/den/gl12,php</b> on line <b>2</b><br>
```

### 11.2.3. Временные файлы

При написании сложных проектов нам могут потребоваться временные файлы, то есть такие файлы, которые нужно удалить после завершения работы программы. Обычно в таких файлах сохраняются промежуточные результаты работы программы.



http://all-ebooks.com

Кроме того, временные файлы могут использоваться в качестве буфера при обработке больших файлов. Например, файл занимает 10 Мб, а свободной виртуальной памяти осталось только 8 Мб: в этом случае мы можем читать и обрабатывать файл по частям, сохраняя результаты обработки части файла во временный файл.

Для создания временного файла в PHP используется функция `tmpfile()`. Она вызывается без параметров, то есть от нас не требуется указывать имени создаваемого файла. Это имя выбирается случайным образом. Функция возвращает идентификатор созданного файла, после чего с временным файлом можно работать как с обыкновенным файлом, открытым в режиме чтения и записи. По окончании работы программы все временные файлы будут удалены. Временные файлы также удаляются из памяти при их закрытии.

Временные файлы сохраняются во временном каталоге. Обычно в Linux это каталог `/tmp`, а в Windows — `%WINDIR%\Temp`.

### 11.2.4. Чтение и запись в бинарном режиме

Мы уже умеем открывать файлы. Теперь поговорим об их обработке, то есть чтении и записи. На рис. 11.1 и 11.2 показано, как происходит чтение и запись бинарных файлов.

Для каждого открытого файла система хранит величину, которая называется указателем текущей позиции файла. При открытии файла этот указатель устанавливается на начало файла. С каждым вызовом функции чтения

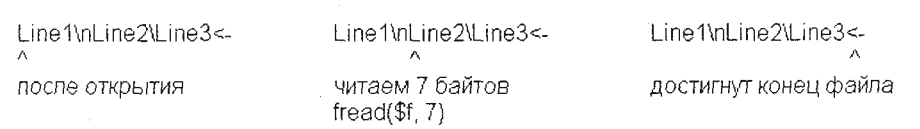



***Рис. 11.1. Чтение бинарного файла***

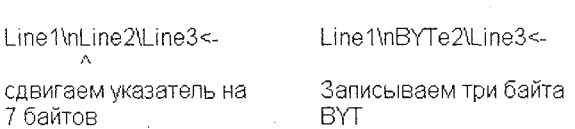



***Рис. 11.2. Запись бинарного файла***

файла указатель текущей позиции сдвигается на прочитанное количество байтов (рис. 11.1, напоминаю, что `\n` — это один символ).

Запись данных в файл начинается с позиции, на которую указывает указатель (рис. 11.2). Чтение двоичного (бинарного) файла осуществляется функцией `fread()`, принимающей два параметра: идентификатор файла и количество байтов, которое нужно прочитать. Функция возвращает строку, в которой будут прочитанные символы. Если в качестве параметра функции будет указано большее число символов, чем есть в файле на самом деле, то ошибки не будет — просто будет выведено все, что есть.

Пример чтения двоичного файла приведен в листинге 11.1.

**Листинг 11.1. Чтение бинарного файла**

```
<?
$f = fopen("/home/den/lines.txt", "r") or die("Ошибка!\n");
$s = fread($f, 4);
echo $s;
?>
```

Мы открыли файл в бинарном режиме «только чтение». Затем мы прочитали 4 байта и записали их в строку `$s`, которую вывели на экран. На рис. 11.3 мы видим содержимое файла `lines.txt`, а также вывод сценария — строку «Line».

Для записи символов в файл служит функция `fwrite()`, принимающая два параметра: идентификатор файла и строку, которую требуется записать в файл.

В приведенном выше примере после чтения четырех байтов указатель остановился на пятом байте: следующая операция чтения или записи начнется с пятой позиции. Это означает, что, если нам нужно записать что-нибудь после символов «Line», мы можем сразу после операции чтения записать нужные нам данные. Но мы ведь открыли файл в режиме чтения, поэтому сейчас запись невозможна. Исправим это упущение. Напишем сценарий, который будет работать так:

- открывать файл в режиме записи;
- «перематывать» файл, то есть смещать указатель текущей позиции к пятому байту;
- записывать данные в файл;
- сохранять изменения.

Текст сценария приведен в листинге 11.2.



http://all-ebooks.com

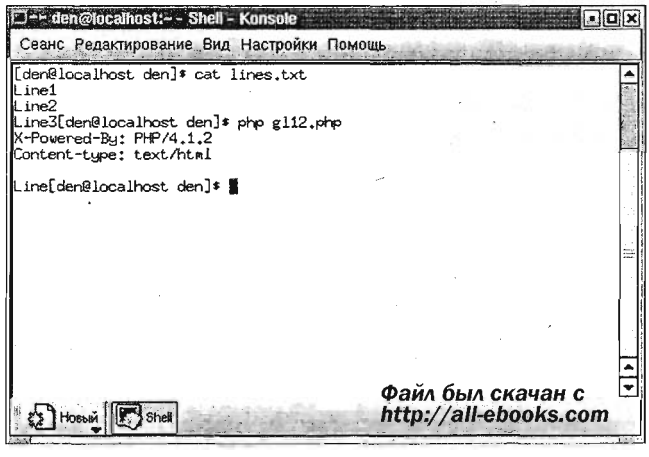



*Рис. 11.3. Вывод сценария 11.1*

**Листинг 11.2. Запись в бинарный файл**

```
<?
$f = fopen("/home/den/lines.txt", "r+") or die("Ошибка!\n");
fseek($f, 4, SEEK_SET);
fwrite($f, "BYTE");
fflush($f);
?>
```

Функция `fseek()` смещает указатель текущей позиции на заданное количество байтов от заданного места (в нашем случае на 4 байта от начала файла). Как видно из сценария 11.2, функция `fseek()` принимает три параметра: идентификатор файла, смещение (позицию, на которую необходимо перейти) и указание, откуда нужно отсчитывать смещение. В качестве указания используются следующие константы:

- SEEK_SET (или 0) — с начала файла;
- SEEK_CUR (или 1) — с текущей позиции;
- SEEK_END (или 2) — с конца файла.

При успешном завершении функция возвращает 0, а в случае ошибки -1.

Сразу после `fseek()` мы вызвали функцию `fwrite()`, которая записывает строку BYTE поверх (режим «r+») имевшихся в файле данных. Обратите внимание, что мы перезаписали четыре символа: `1`, `\n`, `Li` . А это означает, что строк в файле будет уже не три, а две (рис. 11.4), поскольку мы перезаписали символ новой строки.

Функция `fflush()` записывает на диск текущее состояние открытого файла, идентификатор которого принимает в качестве параметра. Сохранение

текущего состояния происходит и при закрытии файла, но функцию закрытия мы еще не рассматривали.

Рекомендуется вызывать `fflush()` в ходе обработки файла несколько раз, чтобы не потерять данные по причине, например, сбоя питания. Перед закрытием файла также желательно вызвать функцию `fflush()`, чтобы принудительно сбросить содержимое буферов ввода/вывода на диск. Функцию `fflush()` можно даже использовать вместо закрытия файла: интерпретатор PHP сам закроет файл при завершении работы сценария.

Если вы не хотите по десять раз вызывать функцию `fflush()`, вы можете установить размер буфера для файла таким, чтобы в случае отказа системы потеря данных была минимальной. Для установки размера буфера служит вызов `set_file_buffer(int $f, int $buffersize)`.

Можно вообще отключить буферизацию для определенного файла. При отключенной буферизации каждое изменение файла сразу же записывается на диск. Разумеется, безопасность данных достигается за счет снижения производительности сценария.

Следующий вызов отключает буферизацию для файла `$f`:

```
set_file_buffer($f, 0);
```

Теперь попробуем записать что-нибудь в конец файла. Стандартный способ сделать это заключается в открытии файла в режиме «a+». Но мы сейчас изучаем функцию `fseek()`, поэтому откроем файл в режиме «r+» и перемотаем его к концу вручную (листинг 11.3).

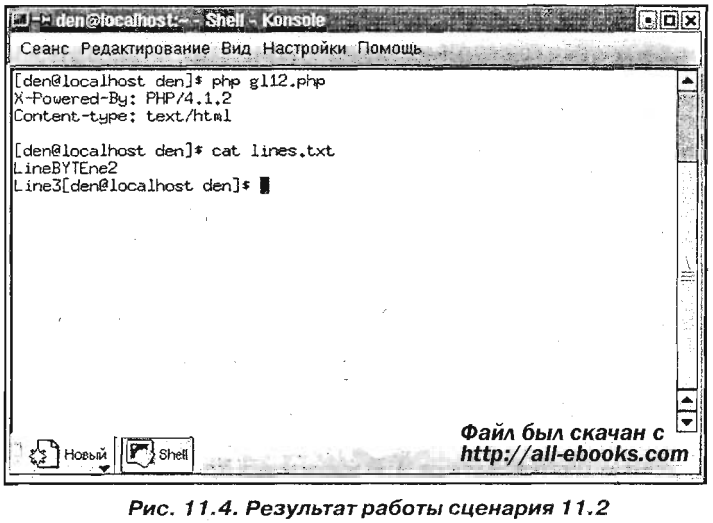



*Рис. 11.4. Результат работы сценария 11.2*



http://all-ebooks.com

**Листинг 11.3. Запись в конец файла**

```
<?
$f = fopen("/home/den/lines.txt", "r+") or die("Ошибка!\n");

fseek($f, 0, SEEK_END); // перемещаем указатель к смещению 0 от
конца файла
fwrite($f,"\nLine4");
fflush($f);
?>
```

Как видно на рис.11.5, мы добавили в файл еще одну строку (`\n` — символ новой строки). Размер файла увеличился на 6 байтов.

Теперь рассмотрим очень важную функцию — `feof()`, которая возвращает `true`, если достигнут конец файла. Применение этой функции продемонстрировано в листинге 11.4.

**Листинг 11.4. Обнаружение конца файла**

```
<?

$f = fopen("/home/den/lines.txt", "r+") or die("Ошибка!\n");

while (!feof($f))
{
  $s = fread($f,4);
  echo $s;
}

?>
```

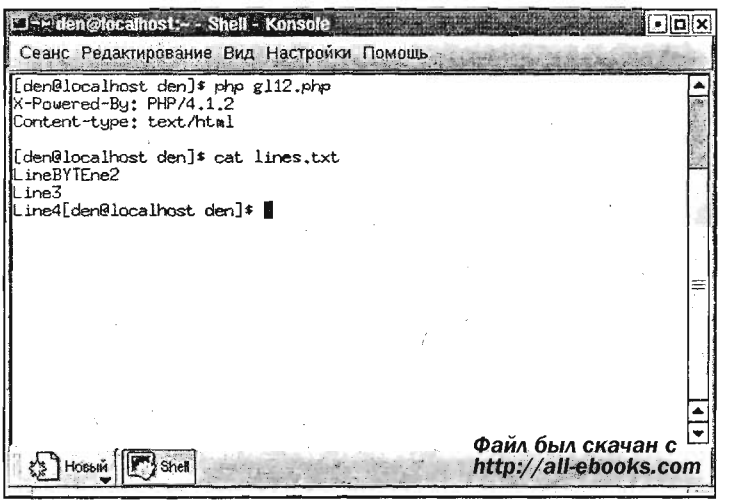



*Рис. 11.5. Результат работы сценария 11.3*

Этот сценарий читает данные из файла блоками по 4 байта и прекращает работу по достижении конца файла. Можно, конечно, читать данные посимвольно, но это неэффективно: гораздо эффективнее прочитать блок в строку, а потом работать посимвольно со строкой.

Если функции `fread()` не удастся прочесть заданное количество байтов, то она возвратит то, что удалось ей прочитать. Поэтому не нужно беспокоиться о том, что конец файла будет достигнут раньше времени.

Функцию `feof()` можно использовать при обработке как двоичных, так и текстовых файлов.

Прочесть весь файл в одну строку можно с помощью такого оператора:

```
$s = fread($f, filesize($f));
```

Однако для этого существуют более эффективные функции, например `file()`, которую мы рассмотрим позже.

Еще одна функция, полезная для работы с файлами в бинарном режиме — это `ftell()`, которая возвращает текущее положение указателя файла. Передать этой функции нужно единственный параметр — идентификатор файла.

### 11.2.5. Построчное чтение и запись

Текстовые файлы удобно читать и записывать по одной строке за раз. Для построчного чтения в PHP предусмотрена функция `fgets(int $f, int $length)`, которая читает из файла `$f` строку длиной `$length` символов. Символ новой строки также включается в результат. Если строка занимает больше чем `$length-1` символов, то возвращаются только первые `$length-1` символов строки. Если строка занимает менее `$length-1` символов, то возвращаются все символы до конца строки (т. е. до символа `\n`).

Обычно функция используется в контексте, показанном в листинге 11.5.

**Листинг 11.5. Построчное чтение файла**

```
<?
$f = fopen("/home/den/lines.txt", "rt") or die("Ошибка!\n");

while (!feof($f))
{
  $s = fgets($f,255);
  echo $s;
}

?>
```



http://all-ebooks.com

Для записи в файл одной строки в текстовом режиме служит функция `fputs(int $f, string $s)`, которая записывает строку `$s` в текстовый файл с идентификатором `$f`.

### 11.2.6. Закрытие файлов

Последняя операция над файлом — это его закрытие. Закрыть файл `$f` поможет функция `fclose($f)`. Строго говоря, закрывать файл вручную не требуется, поскольку по завершении работы сценария интерпретатор PHP сам закроет все открытые сценарием файлы. Однако закрывать файлы все же рекомендуется. Во-первых, это экономит память. Во-вторых, позволяет избежать потери записанных в файл данных в результате сбоя. В-третьих, закрытия файла требует культура программирования.

### 11.2.7. Функция file(). Чтение файла в список

Очень полезна при обработке текстовых файлов функция `file (string $filename)`, читающая файл целиком в список, каждый элемент которого в результате ее работы будет содержать одну строку, прочитанную из файла. Пусть у нас есть файл `lines.txt`:

```
Line1\n
Line2\n
Line3
```

После вызова `$List = file("lines.txt")` массив `$List` будет заполнен тремя элементами «Line1\n», «Line2\n» , «Line3». Нужно заметить, что функция не вырезает символы новой строки и не выполняет их трансляцию. Это означает, что если наш файл будет записан в Windows-формате:

```
Line1\r\n
Line2\r\n
Line3
```

то элементы массива `$List` будут содержать символы `\r\n`.

### 11.2.8. Чтение файла в строку

Используя функции `file()` и `join()`, можно прочитать весь файл в одну строку. Как это сделать, видно из сценария 11.6.

**Листинг 11.6. Чтение файла в одну строку**

```
<?
$s = join('',file("lines.txt"));
echo $s;
?>
```

С функцией `file()` мы уже знакомы. Остается выяснить, что собой представляет функция `join()`. Функция `join (string $separator, array $list)` возвращает строку, содержащую все элементы массива `$list`, последовательно расположенные и разделенные символами `$separator`.

Предположим, что нам нужно прочитать HTML-файл и удалить из него все теги. Нет ничего проще (листинг 11.7)!

**Листинг 11.7. Вывод HTML-файла без тегов**

```
<?
$Index = join('', file("http://dkws.org.ua/text/about.html"));
$Index = strip_tags($Index);
echo $Index;
?>
```

Сначала мы получаем файл с помощью функции `file()`. Обратите внимание: мы можем получить файл по протоколу HTTP и даже FTP. Функция `file()` возвращает список строк файла, а функция `join()` объединяет все элементы этого списка в строку, разделяя их пустой строкой. Результат работы функции `join()` мы записываем в строку `$Index`, которую выводим оператором `echo` после удаления HTML-тегов функцией `strip_tags()`.

Можно было бы обойтись без функции `join()`,обрабатывая элементы массива поочередно, но предложенный вариант работает намного быстрее. Построчная обработка может понадобиться, если размер выводимого файла слишком велик: она предотвратит задержку вывода в браузер, чтобы пользователь не подумал, что сценарий «завис». Пример построчной обработки файла приведен в листинге 11.8.

**Листинг 11.8. Построчный вывод HTML-файла без тегов**

```
<?
$Index = file("http://dkws.org.ua/text/about.html");

foreach($Index as $v)
{
  echo strip_tags($v);    // удаляем теги
  flush();               // выводим данные в браузер
}
?>
```

Функция `file_get_contents(string $filename)` работает еще проще, чем пара функций `file()` и `join()`: она сразу читает все содержимое файла в одну большую строку. Использовать ее очень просто:



http://all-ebooks.com

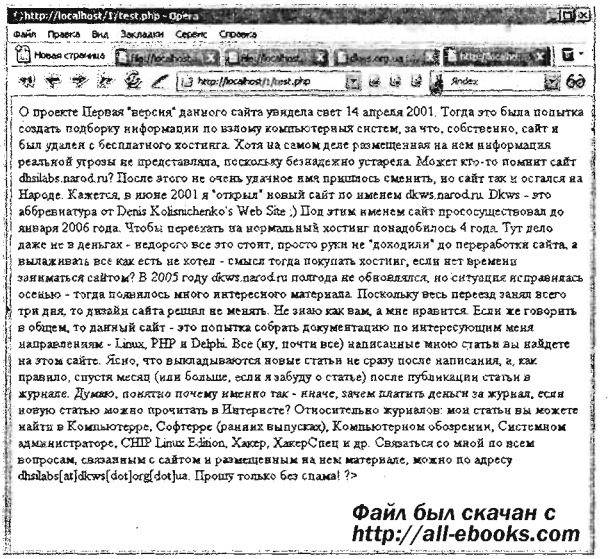



*Рис. 11.6. Сценарий 11.7 выведет только текст без HTML-тегов*

```
$s = file_get_contents("banner.txt"); // читаем banner.txt в строку $s
echo $s;                              // выводим строку $s
```

Эту функцию можно использовать не только для текстовых, но и для двоичных файлов.

## 11.2.9. Чтение CSV-файлов

Некоторые программы электронных таблиц, в том числе всем известная MS Excel, позволяют сохранять таблицы в формате CSV. Файл в формате CSV – это обычный текст, содержащий значения ячеек таблицы, разделенные точкой с запятой (рис. 11.7).

Прочитать CSV-файл позволяет функция `fgetcsv (int $file, int $length, char $separator = ';')`.

Функция читает из файла с идентификатором `$file` одну строку длины не более чем `$length` символов, разбивает ее по символу `$separator` и возвращает список, заполненный «полями» этой строки. Если достигнут конец файла, функция вернет `false`.

Символ-разделитель по умолчанию равен «,». Если в качестве параметра `$separator` передана строка из нескольких символов, то разделителем считается первый из них.

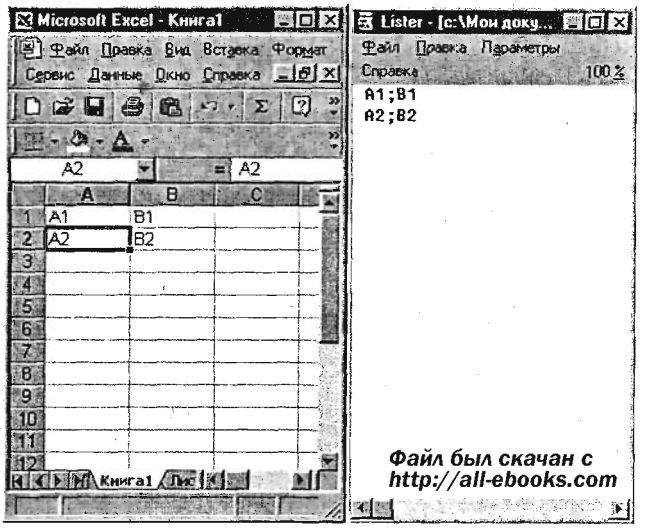



***Рис. 11.7. Файл в формате CSV***

В случае файла, изображенного на рис. 11.7, функция возвратит список из двух элементов «A1» и «B1».

**Листинг 11.9. Вывод CSV-файла**

```
<?
$f = fopen("Книга1.csv", "r") or die("Невозможно открыть файл");

echo "<PRE>";

for($i = 0; $array=fgetcsv($f,1024); $i++)
{

   $j = count($array);
   if ($j==1 && array [0]==="") continue;
   for ($k=0; $k<$j; $k++) echo "$array[$k] ";

   echo "<BR>";
}
fclose($f);
?>
```

## 11.3. ФУНКЦИИ ДЛЯ РАБОТЫ С ФАЙЛАМИ

PHP предоставляет большой выбор самых разных функций для работы с файлами. Все функции можно разделить на такие группы:



http://all-ebooks.com

- Функции манипулирования файлами.
- Функции для работы с именами файлов.
- Функции определения типа и параметров файла.
- Функции определения прав доступа к файлу.
- Специальные функции.

### 11.3.1. Функции манипулирования файлами

К этой группе относятся функции копирования, переименования, удаления файлов, а также некоторые другие функции.

Для копирования файлов служит функция `copy(string $source, string $dest)`. Она копирует файл с именем `$source` в файл с именем `$dest`. Если файл `$dest` существует, он будет перезаписан, поэтому вы должны проверить существование файла с помощью функции `file_exists()`, которая будет рассмотрена ниже.

Еще один момент, о котором вы должны позаботиться прежде, чем начать копирование. Вы должны иметь право на запись в каталог, в котором должен находиться файл `$dest`, и в сам файл, если он до вызова функции существовал, чтобы перезаписать его.

Функция `copy()` возвращает `true`, если файл удачно скопирован, и `false` в противном случае.

Переименовать файл можно с помощью функции `rename(string $oldname, string $newname)`. Если переименовать файл нельзя, функция возвратит `false`. Это может произойти в следующих случаях:

- файл `$oldname` не существует;
- у вас нет прав доступа к файлу `$oldname`;
- файл с именем `$newname` уже существует: обратите внимание, что функция `rename()` в отличие от функции `copy()` не перезаписывает файл;
- имя `$newname` находится на другой файловой системе, то есть на другом разделе диска или сетевой файловой системе;
- другие причины.

Удалить файл можно функцией `unlink(string $name)`. Функция возвращает `false`, если файл не удалось удалить.

**Примечание.**

У вас может возникнуть вопрос, почему функция удаления файла называется unlink(), а не как-нибудь вроде remove или delete. Дело в том, что файловая система Unix-подобных операционных систем позволяет создавать жесткие ссылки на файлы (hard links). Жесткую ссылку можно рассматривать просто как альтернативное имя файла: один и тот же файл (в терминах файловой системы – информационный узел, inode) может быть доступен по нескольким разным именам. Все жесткие ссылки равноправны, и к физическому удалению файла приводит только удаление последней жесткой ссылки на него. В Unix принято удалять файлы командой unlink, и PHP унаследовал эту традицию.

### 11.3.2. Функции для работы с именами файлов

Довольно часто приходится иметь дело с именами файлов. То требуется присоединить имя файла к имени каталога, то выделить имя из полного пути к файлу, то вычислить абсолютный путь по относительному. Для решения этих задач в PHP предусмотрена группа функций. Рассмотрим самые полезные из них.

- `file_exists(string $name)` проверяет существование файла;
- `basename(string $path)` выделяет имя файла из полного пути;
- `dirname(string $path)` возвращает имя каталога из полного пути;
- `realpath(string $path)` преобразует относительный путь к файлу в абсолютный;
- `tempname(string $dir, string $prefix)` генерирует имя для временного файла.

Функция `file_exists()` возвращает `true`, если файл с указанным именем существует, и `false` в противном случае:

```
<?
if (!file_exists("/home/den/lines.txt"))
{
  echo "Файл не существует";
}
else echo "Файл существует";
?>
```

Функция `basename()` возвращает основное имя файла, то есть имя файла без пути к нему:

```
echo basename("/home/den/lines.txt");  // выведет lines.txt
echo basename("/.");                   //  выведет .
```

Обратите внимание, функция `basename()` не проверяет существование файла, а просто возвращает часть строки, стоящую после последнего слэша.



http://all-ebooks.com

Эта функция правильно обрабатывает как прямые слэши, так и обратные (что важно для Windows).

Функция `dirname()` возвращает, наоборот, имя каталога:

```
echo dirname("/home/den/lines.txt"); // выведет /home/den
echo dirname("../lines.txt"); // выведет .. (родительский каталог)
echo dirname("lines.txt");    // выведет . (текущий каталог)
echo dirname ("/swap_file");  // выведет «/» в Linux или «\» в Windows
```

Функция `realpath()` преобразует относительный путь к файлу в абсолютный (полный):

```
echo realpath("./lines.txt");     // выведет /home/den/lines.txt
```

Функция `tempname(string $dir, string $prefix)` генерирует имя файла в каталоге `$dir` с префиксом `$prefix`, причем так, чтобы полученное имя было уникальным в пределах каталога. В качестве аргумента `$dir` обычно указывается временный каталог, а в качестве префикса имя нашего сценария, создающего временный файл, но это только рекомендация: вы можете указать все что угодно.

```
$fname = tempnam("/var/www/html/tmp","dev");
```

### 11.3.3. Функции определения типа и параметров файла

Для определения типа файла служит функция `filetype(string $filename)`, возвращающая тип файла в виде строки (табл. 11.2).

Подробную информацию о файле (массив из 13 элементов) возвращает функция `stat(string $filename)` (табл. 11.3).

**Таблица 11.2.** *Типы файлов*

| Возвращаемое значение | Тип файла |
|---|---|
| file | Обычный файл. Может быть как текстовым, так и двоичным |
| dir | Каталог |
| link | Символическая ссылка |
| fifo | FIFO-канал |
| block | Блочно-ориентированное устройство (например, жесткий диск /dev/hddN) |

**Таблица 11.2.** *Типы файлов(продолжение)*

| | |
|---|---|
| char | Символьно-ориентированное устройство (например, терминал /dev/ttyN) |
| unknown | Тип определить не удалось |

**Таблица 11.3.** *Параметры файла, возвращаемые функцией stat()*

| Ключ массива | Значение |
|---|---|
| 0 | Устройство |
| 1 | Номер inode |
| 2 | Атрибуты файла |
| 3 | Число жестких ссылок на файл |
| 4 | Числовой идентификатор (UID) владельца файла |
| 5 | Числовой идентификатор группы (GID) владельца файла |
| 6 | Тип устройства |
| 7 | Размер файла в байтах |
| 8 | Время последнего доступа (в секундах с 1 января 1970 года) |
| 9 | Время последнего изменения файла |
| 10 | Время последнего изменения атрибутов файла |
| 11 | Размер блока |
| 12 | Число блоков, которые занимает файл |

Сценарий, представленный в листинге 11.10, собирает детальную информацию о файле `/home/den/lines.txt`:

**Листинг 11.10. Детальная информация о файле**

```
<?
$f_info  = stat("/home/den/lines.txt");

echo "Файл:  /home/den/lines.txt\n";
echo "Устройство: \t\t$f_info[0]\n";
echo „Номер inode: \t\t$f_info[1]\n";
echo „Атрибуты \t\t$f_info[2]\n";
echo "Число жестких ссылок: \t$f_info[3]\n";
echo "UID $f_info[4] GID $f_info[5]\n";
echo "Тип устройства:  \t$f_info[6]\n";
echo "Размер:  \t\t$f_info[7]\n";
echo "Время последнего доступа: ".date("d M Y H:i:s",$f_info[8])."\n";
echo "Время последнего изменения файла: ".date("d M Y
        H:i:s",$f_info[9])."\n";

echo "Время последнего изменения атрибутов файла: ".date("d M Y
```



http://all-ebooks.com

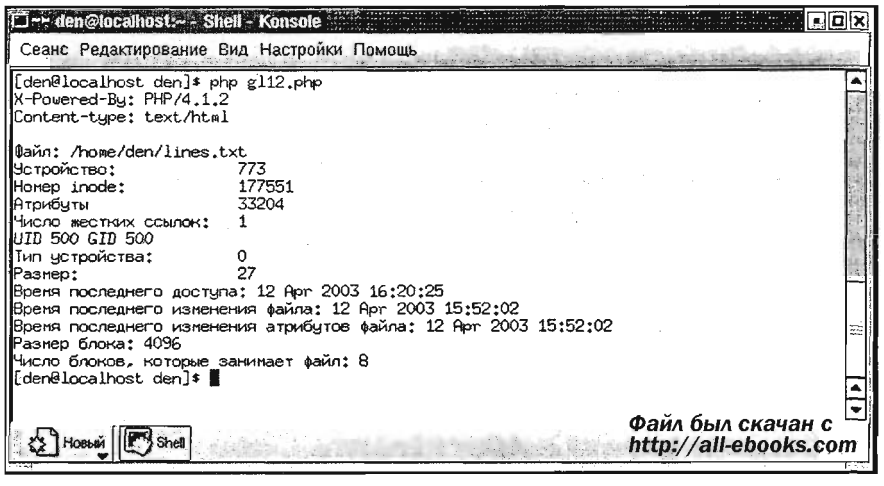



*Рис. 11.8. Информация о файле*

```
H:i:s",$f_info[10]).”\n”;
echo “Размер блока: $f_info[11]\n”;
echo “Число блоков, которые занимает файл: $f_info[12]\n” ;

?>
```

Если вы работаете под Windows 9x, многие элементы массива будут содержать пустую строку.

Если вы вызвали функцию `stat()` для символической ссылки, то будет возвращена информация о файле, на который ссылается ссылка. Чтобы получить информацию о самой ссылке, используйте функцию `lstat()`. Она возвращает массив такого же формата, как функция `stat()`.

### 11.3.4. Специальные функции

Разработчики PHP предусмотрели несколько полезных и удобных функций для работы со временем создания и модификации файла. Конечно, можно использовать функцию `stat()`, но если нам нужно узнать только время создания файла, то получать и обрабатывать массив не совсем рационально.

Функция `fileatime(string $filename)` возвращает время последнего доступа («a» в имени функции взято из слова «access») к файлу. Если файл не существует, она возвращает `false`.

Функция `filectime(string $filename)` возвращает время создания («c» — «create») файла. Функция `filemtime(string $filename)` возвращает время последнего изменения («m» — «modify») файла.

Пример использования функций:

```
$A =  fileatime("/home/den/lines.txt");
$C =  filectime("/home/den/lines.txt");
$M =  filemtime("/home/den/lines.txt");
echo   "Время последнего доступа: ",date("d M Y H:i:s",A) ."\n";
echo   "Время создания файла: ",date("d M Y H:i:s",C)."\n";
echo  "Время последнего изменения файла: ",date("d M Y H:i:s",M)."\
n";
```

Функция `touch(string $filename, int $time)` позволяет установить для файла `$filename` время последней модификации `$time`. Если второй параметр не указан, то временем модификации станет текущее время. Если же указанный файл не существует, то он будет создан.

Пример:

```
$M = filemtime("/home/den/lines.txt");
// устанавливаем время модификации файла some_file.txt таким
// же, как у файла lines.txt
touch("/home/den/some_file.txt",$M);
// устанавливаем время модификации на текущее
touch("/home/den/file.txt");
// Файл no_such_file не существует, поэтому функция возвратит
// false и создаст новый пустой файл с именем no_such_file
touch("/home/den/no_such_file");
```

Функция `disk_free_space()` возвращает объем свободной дисковой памяти в байтах на указанной файловой системе (для Linux) или разделе диска (для Windows):

```
$df = disk_free_space("/"); // сколько свободно на корневой файловой
системе (Linux)
$df = disk_free_space("c:"); // сколько свободно на диске C:
(Windows)
```

Функция `disk_total_space()` возвращает общий объем указанной файловой системы или раздела диска:

```
$dt = disk_total_space("/");   // общий размер корневой файловой
системы (Linux)
$dt = disk_total_space("c:"); // общий размер диска C: (Windows)
```

### 11.3.5. «Разборки» с INI-файлами. Функция parse_ini_file()

Предположим, что ваш проект «разросся» до такого уровня, что ему просто не обойтись без собственного файла конфигурации. Одним из самых удобных форматов для файла конфигурации остается формат INI-файлов:



http://all-ebooks.com

```
[Секция_1]
Параметр1 = Значение1
Параметр2 = Значение2
...
[Секция_2]
Параметр1 = Значение1
...
```

Предположим, что у нас есть следующий файл конфигурации `project.ini`:

```
[General]
Title = "Sample Title"
Header = "/usr/share/project/header.html"
Footer = "/usr/share/project/footer.html"

[DB]
User = db_user
Password = 123456
```

Для чтения файлов в формате INI в PHP предусмотрена специальная функция `parse_ini_file(string $filename, bool $use_sections=false)`.

Второй параметр задает режим работы функции: если он равен `false`, то функция будет работать без разбора секций, то есть секции будут игнорироваться; значение `true` требует от функции учитывать секции. Режим разбора секций нужен, когда разные секции могут содержать параметры с одинаковыми именами. По умолчанию секции игнорируются.

Для нашего файла в режиме без учета секций функция возвратит один большой ассоциативный массив:

```
Array (
[Title] => "Sample Title"
[Header] => "/usr/share/project/header.html"
[Footer] => "/usr/share/project/footer.html"
[User] => db_user
[Password] => 123456
)
```

В режиме учета секций функция вернет массив ассоциативных массивов. Каждый элемент этого массива – это ассоциативный массив, соответствующий определенной секции:

```
Array (
  Array (
[Title] => "Sample Title"
[Header] => "/usr/share/project/header.html"
[Footer] => "/usr/share/project/footer.html"
   )
```

http://all-ebooks.com

```
  Array (
[User] => db_user
[Password] => 123456
 )
)
```

Вывести значения конфигурационных параметров поможет следующий фрагмент кода:

```
<?
$ini = parse_ini_file("project.ini", false);

foreach ($ini as $key => $value) echo "$key = $value\n";

?>
```

## 11.4. ПРАВА ДОСТУПА

И в ОС Linux(Unix), и в ОС Windows у каждого файла и каталога есть так называемые права доступа. Права доступа определяют, кто имеет доступ к объекту (этому файлу или каталогу) и какие операции над объектом он может выполнять. Основных операций три: чтение, запись и исполнение.

### 11.4.1. Права доступа в ОС Linux

В ОС Linux право на чтение означает для файла разрешение просматривать и печатать этот файл, а для каталога — разрешение просматривать список содержащихся в нем файлов. Право на запись для файла означает возможность его редактирования, а для каталога — возможность создания и удаления в нем файлов.

Если для файла установлено право на исполнение, то его можно запускать как программу. Эта возможность используется при написании сценариев командных интерпретаторов. Право исполнения для каталога разрешает доступ к этому каталогу (переход в него по команде `cd` и просмотр оглавления), но не разрешает исполнения расположенных в нем файлов.

При установке прав доступа принимается во внимание категория пользователей. В общем случае существует три категории пользователей: владелец, группа и прочие.

- *Владелец* — пользователь, создавший файл. Само собой разумеется, для того, чтобы создать файл, вы должны иметь право записи в каталог, в котором вы создаете файл. При создании файла обычно ус-



http://all-ebooks.com

танавливаются права на чтение и запись для владельца и только на чтение для всех остальных пользователей.

- Пользователи объединяются в *группы*, например, для работы над одним проектом. Владелец может разрешить или запретить доступ к файлам для членов группы.
- *Прочие* — это все остальные пользователи.

Для каждой категории имеется свой набор прав доступа, просмотреть который вы можете с помощью команды `ls -l`:

```
> ls -l file.txt
-rw-r----- 1 den group 300 Feb 10 12:00 file.txt
```

Строка «-rw-r-----» - характеризует набор прав доступа к файлу `file.txt`, владельцем которого является пользователь den, входящий в группу `group`.

Первый символ — это тип файла: «-» означает файл, а «d» — каталог.

Следующие три символа «rw-» задают права доступа для владельца файла. Символ «r» — это право на чтение, «w» — на запись, а «x» — на выполнение. Права задаются именно в таком порядке: чтение, запись, выполнение. Если право на какой-нибудь вид доступа отсутствует, то ставится «-». В данном случае отсутствует право на выполнение.

Второй трехсимвольный набор задает права доступа для группы, а третий — для прочих пользователей. В нашем примере (r--) члены группы имеют право только на чтение, а другие пользователи вообще не имеют никакого доступа к файлу (---).

Для изменения прав доступа в Linux используется команда `chmod`. При этом права доступа можно задавать двумя способами: символьным и абсолютным. В этой книге будет рассмотрен абсолютный, поскольку он нам еще пригодится.

> **Примечание.**
>
> О символьном способе вы можете прочитать в моей книге «Linux-сервер своими руками», которая вышла в издательстве «Наука и техника».

Как мне кажется, абсолютный способ несколько удобнее, чем символьный, поскольку не нужно помнить символику прав доступа. Этот метод еще называют методом двоичных масок.

Для изменения прав доступа абсолютным методом используется та же команда `chmod`:

```
chmod число <файл|каталог>
```

Число называется маской прав доступа и представляет собой трехразрядное число в восьмеричной системе счисления. Разряды нумеруются слева направо, то есть в числе 764 цифра 7 – это первый разряд, 6 – второй и 4 – третий. Первый разряд задает права доступа для владельца файла, второй — для группы, третий — для остальных пользователей.

Одному разряду восьмеричной системы соответствует три разряда в двоичной. Двоичные разряды также нумеруются слева направо. Первый двоичный разряд задает право на чтение, второй — на запись, третий — на исполнение: двоичная цифра 1 говорит о наличии этого права, 0 — об отсутствии.

Таким образом, маска 764 = 111 110 100 означает: право на чтение, запись и исполнение для владельца файла; на чтение и запись для пользователей из группы владельца; только на чтение для всех остальных.

Чаще всего используется маска 644, разрешающая чтение и модификацию файла для владельца и только чтение для других пользователей.

Иногда набор прав доступа задается четырехзначным числом. В этом случае старшая восьмеричная цифра содержит флаги SETUID (4), SETGID (2), sticky-бит (1).

Пусть мы создали каталог командой `mkdir`. Какие права доступа он получит по умолчанию? Это будет маска, задающая полные права для всех (777) минус маска режима создания файлов, обычно равная 022. Разность равна 755, или 111 101 101, то есть полные права для владельца и права на чтение и просмотр оглавления для всех остальных.

Текущее значение маски режима создания файлов позволяет узнать команда `umask`. Эта же команда, введенная от имени суперпользователя, позволяет установить новое значение маски.

Обратите внимание на то, что для каталога обязательно должно быть установлено право исполнения, иначе к каталогу нельзя будет получить доступ.

### 11.4.2. Права доступа в Windows

В устаревших версиях Windows (95/98/ME) понятия прав доступа не существует, то есть любой пользователь имеет право читать, изменять и даже удалять любой файл. Ваш сценарий может сделать все, даже записать в ваш `autoexec.bat` команду

```
echo y | format c:
```



http://all-ebooks.com

Вот почему в Интернете нет ни одного сервера на базе Windows 98.

В операционных системах Windows NT/2000/XP существуют права доступа: вы можете разрешить читать/записывать и т. д. свои файлы конкретному пользователю или группе пользователей, но только в том случае, если ваш файл находится на файловой системе NTFS. Если же ваши диски отформатированы под FAT32, повторяется ситуация с Windows 9x.

### 11.4.3. Функции определения прав доступа к файлу

Познакомившись с понятием прав доступа, вернемся теперь к PHP. Рассмотрим функции, которые предоставляет PHP для проверки прав доступа.

- Функция `is_readable(string $filename)` возвращает `true`, если вызвавший ее сценарий может прочитать файл, то есть открыть его в режиме «r».
- Функция `is_writeable(string $filename)` возвращает `true`, если сценарий может записать данные в файл.
- Функция `is_executable(string $filename)` возвращает `true`, если файл исполняемый (является программой).

Эти функции рекомендуется использовать перед открытием важных файлов для обеспечения корректности доступа к ним.

## 11.5. БЛОКИРОВАНИЕ ДОСТУПА

### 11.5.1. Общие положения

Как UNIX, так и Windows NT/2000/XP – это многозадачные операционные системы, поэтому в них предусмотрена совместная (одновременная) работа нескольких программ с одним файлом.

Представьте себе, что независимые процессы А и Б пытаются открыть один и тот же файл на запись. Если разрешить запись обоим, то вместе они могут натворить с файлом такого, что вы потом никогда не поймете, что и откуда взялось. Поэтому и в UNIX (Linux), и в Windows есть механизмы блокирования доступа к файлу.

http://all-ebooks.com

Пусть процесс А открыл файл первым. Сразу же после открытия он блокирует файл, и процесс Б не сможет открыть этот файл, пока процесс А не снимет блокировку. Обратите внимание, *снимет блокировку,* а *не завершит работу.* Поэтому, когда вы установите блокировку, постарайтесь как можно быстрее ее снять — сразу же после окончания работы с файлом.

Предположим, что вы хотите обработать массив данных, которые хранятся в файле. После того как вы прочитаете массив из файла, вы должны освободить файл, чтобы он не был блокирован на время обработки вашей программой. Перед записью в файл опять заблокируйте файл, чтобы никакой другой процесс не повредил информацию. Конечно, в этом случае не исключена ситуация, когда другой процесс заблокирует файл и уже вам придется ждать его освобождения.

### 11.5.2. Виды блокировки

Существует два вида блокировки:

- *исключительная* — файл может использовать только процесс, открывший его (обычно применяется для записи данных в файл);
- *разделяемая* — процесс, открывший файл, может «поделиться» им с другими процессами, запущенными на этой же машине.

Предположим, что у нас есть какой-нибудь файл и два процесса, которым нужно записать информацию в этот файл. Процесс А первым открыл файл для записи и заблокировал его, используя *исключительную блокировку.* Тогда процесс Б будет ждать, пока процесс А не разблокирует файл или не закроет его. Только тогда он сможет получить доступ к файлу для записи. Исключительную блокировку нужно использовать тогда и только тогда, когда вам нужно что-нибудь записать в файл.

Если вам нужно только прочитать данные из файла, целесообразнее использовать разделяемую блокировку. Представьте, что мы пишем счетчик посещений. Допустим, что есть несколько процессов, которым нужно прочитать информацию. Если мы используем исключительную блокировку для чтения, эти процессы будут ждать, пока мы не разблокируем файл. Но ведь чтение — это не критическая операция, поэтому с точки зрения повышения производительности нужно предоставить другим процессам возможность прочитать файл.

Мы рассмотрели два случая совместного использования файла: когда несколько процессов одновременно пытаются записать что-нибудь в файл и когда несколько процессов одновременно пытаются прочитать файл. А что



http://all-ebooks.com

делать, если один процесс хочет записать в файл, а несколько — прочитать файл? В этом случае процессы, открывшие файл для чтения, будут обязаны подождать, пока первый процесс завершит запись и разблокирует файл. И наоборот, если в данное время какой-то процесс читает файл, процесс, которому нужно писать в файл, должен подождать, пока первый процесс не разблокирует файл.

Немного подытожим: если вам нужно писать в файл, обязательно используйте исключительную блокировку, а если читать — только разделяемую.

### 11.5.3. Реализация блокировки в PHP

Для блокирования файла в PHP служит функция `flock(int $f, int $operation [, $block=true])`. Ее первый параметр – это дескриптор открытого файла, второй ($operation) — режим блокировки. Режим блокировки может принимать следующие значения:

- LOCK_SH: разделяемая блокировка (разделяемый доступ для чтения). Другие процессы также могут открыть этот файл для чтения и читать из него данные. Однако если какому-либо из них понадобится записать что-либо в файл, то ему придется подождать своей очереди.
- LOCK_EX: исключительная блокировка (монопольный доступ для записи). При этом все остальные сценарии, кроме текущего, не могут ничего делать с файлом.
- LOCK_UN: снятие блокировки.
- LOCK_NB: если файл заблокирован другим процессом, то у вашего сценария есть две возможности: либо ждать, пока другой процесс «отдаст» файл, либо получить сообщение об ошибке («невозможно блокировать»), продолжить работу и повторить попытку как-нибудь в другой раз. Константа LOCK_NB, прибавленная к одной из вышеперечисленных, запрещает ожидание.

Значение `true` необязательного третьего аргумента требует приостановить выполнение сценария до тех пор, пока блокировка не станет возможна.

В качестве примера рассмотрим дописывание данных в конец файла. До сих пор мы считали, что в файл пишет исключительно наш сценарий, и обходились без блокировки, но теперь знаем, что это небезопасно. Правильный порядок записи приведен в листинге 11.11.

**Листинг 11.11. Запись в конец файла с блокировкой**

```
<?
$f = fopen($fname, "a+") or die("Ошибка!\n");

flock($f, LOCK_EX);      //   Ждем, пока не получим разрешение на
//     исключительную блокировку, и устанавливаем ее.
//     Функция flock() не вернет управление нашему
//     сценарию, пока блокировка не будет выполнена
//
fwrite ($f, $str."\n");
flock($f, LOCK_UN);      //     снимаем блокировку
fclose($f);
?>
```

Работа этого сценария безопасна: если несколько экземпляров этого сценария (или других PHP-сценариев, использующих механизм блокировки) захотят записать что-либо в файл, то они это сделают по очереди.

Для полноты картины приведу пример сценария, читающего из файла (листинг 11.12).

**Листинг 11.12. Чтение файла с блокировкой**

```
<?
$f = fopen($fname, "r") or die("Ошибка!\n");

// Ждем, пока не получим разрешение на разделяемую блокировку
flock($f, LOCK_SH);

// чтение
$s = fgets($f,255);

flock($f, LOCK_UN); // снимаем блокировку

// продолжаем работу
fclose($f);
?>
```

Зачастую бывает полезно не дожидаться, пока функция `flock()` разрешит доступ к файлу и позволит сценарию выполняться дальше. Можно ведь это время ожидания потратить с пользой и выполнить какие-либо другие действия. Это как раз и позволяет сделать константа LOCK_NB, прибавленная к основному режиму блокировки (листинг 11.13).

**Листинг 11.13. Пример блокировки без ожидания**

```
<?
$f = fopen($fname, "r+") or die("Ошибка!\n");
```



http://all-ebooks.com

```
$i = 0;

while( ! flock($f, LOCK_EX+LOCK_NB))
{
  $i++;
  echo "Попытка номер $i";
  sleep(1) ;   // засыпаем на 1 секунду
}

// Мы получили доступ, можно работать с файлом

fclose($f) ; // Закрываем файл - он автоматически разблокируется
?>
```

При работе с блокировками нужно учитывать, что они не передаются процессам-потомкам: если сценарий заблокировал файл и вызвал какой-то процесс, то этот процесс не получит доступа к файлу до тех пор, пока родительский процесс не разблокирует его.

Также необходимо помнить, что применение функции `flock()` возможно только к открытому файлу. На закрытый файл она никакого действия не возымеет.

При записи в файл с использованием блокировки лучше использовать режимы «r+» и «a+», но не «w».

### 11.5.4. Запись данных в начало файла

Необходимость записи данных в начало файла можно проиллюстрировать примером гостевой книги. Очевидно, что пользователь, заходящий на ваш сайт, хочет видеть самые свежие сообщения сверху. Можно было бы читать файл сообщений с конца, но мне кажется, что гораздо удобнее записывать новые сообщения в начало.

В PHP нет стандартных функций, позволяющих записать информацию в начало файла. Если поставить указатель файла на самое начало (при условии, что вы открыли файл в бинарном режиме) и что-то записать в файл, то новая информация перезапишет уже существующую. Однако следующий пример (листинг 11.14) продемонстрирует вам, как можно организовать запись в начало файла.

**Листинг 11.14. Запись в начало файла**

```
<?
$file_gb="gb.txt";          // файл гостевой книги
$file_tmp="gb_tmp.txt";     // временный файл
```

```
$str= "новая строка текста";

// проверяем, не было ли сбоя во время предыдущего запуска
сценария
if(file_exists($file_tmp)) die("Временный файл существует!");

// копируем содержимое файла в tmp
if(copy($file_gb, $file_tmp))
{
  // если удачно скопировался, можно перезаписывать основной файл
  if($w=fopen($file_gb,"w"))
  {
    flock($w, LOCK_EX);       // блокируем файл

    fwrite($w,$str."\n");  // записываем первой новую строку

    if(!$r=fopen($file_tmp,"r")) die("невозможно открыть файл");

    flock($r,1);

    while($str=fgets($r,10240)) // читаем построчно временный файл
    {
      fputs($w,$str);    // пишем построчно
    }
    flock($r, LOCK_UN);
    fclose($r);
    flock($w, LOCK_UN);
    fclose($w);
    unlink($file_tmp);     // удаляем временный файл
  }
}
?>
```

Сначала мы копируем файл, в начало которого собираемся записать данные, во временный файл. Потом перезаписываем исходный файл, записав в него новые данные. После этого мы построчно читаем временный файл и записываем данные из него в исходный файл. Конечно, можно поступить и по-другому: прочитать весь временный файл в одну строку, например, с помощью функции `file_get_contents()`, а потом сразу записать его в исходный файл. Но это я решил оставить вам в качестве домашнего задания.

В заключение этого пункта хочу напомнить, что в качестве веб-сервера у вас должен быть установлен только Apache. Дело в том, что блокировка файлов функцией `flock()` реализована на уровне процессов, и при использовании многопоточного сервера (например, IIS под Windows NT/2000) она не поможет. Блокировка на уровне процесса не защитит файл от PHP-сценария, запущенного в параллельном потоке одного и того же серверного процесса.



http://all-ebooks.com

## 11.6. ФУНКЦИИ ДЛЯ РАБОТЫ С КАТАЛОГАМИ

### 11.6.1. Манипуляции с каталогами

Следующие функции работают с каталогами в целом:

- `mkdir(string $dirname, int $perms)`: создает каталог с именем `$dirname` и правами доступа `$perms`. Права доступа указываются в абсолютном (числовом) виде.
- `rmdir(string $dir_name)`: удаляет *пустой* каталог.
- `getcwd()`: возвращает текущий каталог.
- `chdir(string $dirname)`: изменяет текущий каталог.

Например, следующий вызов создаст в текущем каталоге подкаталог `mydir` с правами доступа 0755:

```
mkdir("mydir",0755);
```

В связи с функцией `mkdir()` я хочу отметить два момента. Во-первых, вы должны иметь право на запись в тот каталог, в котором создаете подкаталог (в приведенном примере это текущий каталог). Во-вторых, число `$perms` должно быть восьмеричным. Для указания интерпретатору PHP, что числовую константу нужно воспринимать как восьмеричную, ее запись должна начинаться с цифры нуль.

Чаще всего создаваемому каталогу присваивают права 0770, что означает доступ на запись, чтение и исполнение для владельца файла и его группы.

Функция `rmdir()` удаляет только пустой каталог. Если каталог содержит файлы, он не будет удален, и функция вернет `false`.

Функция `getcwd()` возвращает строку – имя текущего каталога.

Последняя функция, `chdir()`, изменяет текущий каталог на указанный, например `chdir("/var/www/html/docs")`.

### 11.6.2. Открытие, чтение и закрытие каталогов

Каталог — это обыкновенный файл, содержащий информацию о находящихся «в нем» файлах. Последовательность работы с каталогом такая же, как и с файлом: открыть, прочитать (писать в каталог мы не можем) и закрыть. Для этого в PHP есть три функции:

- `opendir(string $dirname)`: открывает каталог, после чего мы можем читать его оглавление. Функция возвращает дескриптор каталога `$handle`, который нужно указывать в качестве аргумента при вызове других функций для работы с каталогом.

- `readdir(int $handle)`: считывает очередное имя файла или подкаталога, которые находятся в каталоге, заданном дескриптором `$handle`.
- `closedir(int $ handle)`: закрывает каталог.

Пример совместного использования всех трех функций приведен в листинге 11.15.

**Листинг 11.15. Открытие, чтение и закрытие каталогов**

```
<?
$dir = opendir("/var/www/html");

chdir("/var/www/html");
echo "<html><head><title>Печать Каталога</title></head><body>";
echo "<h1>Оглавление каталога</h1><p>" ;
echo "<table width = 100%>";

while ($d=readdir($dir))
{
  echo "<tr><td>";

  if (is_dir ($d)) echo "$d</td><td>Каталог</td>" ;
  if (is_file($d)) echo "$d</td><td>".filesize($d)."</td>";

  echo "<tr>";
}

echo "</table></body></html>" ;

closedir($dir);
?>
```

Сценарий 11.15 сначала открывает каталог, потом переходит в него, сделав его текущим. Далее в цикле `while` мы последовательно читаем оглавление каталога и анализируем каждую запись `$d`. Если `$d` является каталогом (функция `is_dir()` возвращает `true`), мы выводим в соответствующую ячейку таблицы слово «Каталог», а если `$d` — обычный файл (функция `is_file()`), то выводим его размер, полученный вызовом `filesize()`. Прочитав все записи каталога, мы его закрываем.

**Примечание.**

Цикл while с условием ($d=readdir($dir)) не рассчитан на случай, когда в каталоге окажется файл с именем «0»: помните, как PHP интерпретирует строки в логических выражениях? Переменная $d получит значение false, и цикл завершит работу. Поэтому надежнее будет записать условие цикла следующим образом: (false !== ($d = readdir($dir)))



http://all-ebooks.com

Каталог, как и файл, обладает указателем текущей позиции. Чтобы по десять раз не закрывать/открывать каталог, для сброса указателя можно использовать функцию `rewinddir()`, которая «перематывает» каталог на начало:

```
$dir = opendir("dir");
. . .
rewinddir($dir);
. . .
closedir($dir);
```

## 11.7. КАНАЛЫ

### 11.7.1. Общие положения

Язык PHP предусматривает функции для работы с каналами, но я, честно говоря, не вижу задач, в которых каналы понадобились бы веб-программисту. Областью их применения традиционно считается системное программирование. Поэтому я сначала введу основные понятия о каналах применительно к языку C и только потом расскажу, как каналы реализованы на PHP.

Канал — это связь между родительским и дочерним процессом. Каналы позволяют процессам обмениваться данными, минуя файлы, через оперативную память.

Конечно, можно из одной программы писать данные в файл, а второй программой их читать (не забывая о блокировке совместно используемого файла). Но такой подход нерационален в первую очередь из-за его низкого быстродействия, ведь скорость работы с файлами намного ниже скорости работы с оперативной памятью. Ясное дело, что для сохранения информации вам все равно придется работать с файлами, но для передачи «оперативной» информации другому процессу гораздо удобнее использовать каналы.

### 11.7.2. Управление каналами на языке C

Перенаправление ввода/вывода можно осуществить и программным путем, то есть без вмешательства пользователя. Наша программа сама перенаправит свой ввод/вывод в нужный файл или какой-нибудь программе.

Вызов `system()` порождает дочерний процесс, позволяя ему читать данные со стандартного ввода (`stdin`) и писать на стандартный вывод

http://all-ebooks.com

(stdout). Иногда нам нужно передать данные дочернему процессу или, наоборот, получить информацию от порожденного процесса. Другими словами, мы хотим, чтобы дочерний процесс получал данные не со стандартного ввода, а от родительского процесса или/и выводил информацию не на стандартный вывод, а передавал ее процессу-предку. Ввод/вывод между процессами осуществляется с помощью системного вызова popen():

```
FILE * popen(const char * команда, const char * режим_доступа);
```

Первый параметр — это имя исполняемого файла, который будет запущен как дочерний процесс. Второй параметр определяет режим доступа. Значение «r» указывает, что родительский процесс будет читать вывод дочернего. Значение «w», наоборот, указывает, что вывод родительского процесса поступит на стандартный ввод дочернего, то есть дочерний процесс будет читать данные, выводимые родительским. Режима двустороннего обмена данными не существует.

Вызов popen() возвращает указатель FILE или пустой указатель NULL в случае ошибки. Так же как и при работе с обыкновенными файлами, вы должны закрыть канал вызовом pclose() после завершения операции ввода/вывода. Во время работы с потоком рекомендую использовать вызов fflush() — как и в PHP, чтобы предотвратить задержки из-за буферизации.

Теперь несколько простых примеров. Предположим, что нам нужно вывести на стандартный вывод имена всех текстовых файлов, содержащихся в текущем каталоге. Это можно очень просто сделать с помощью вызова system():

```
system("ls *.txt");
```

Это уж совсем тривиальная задача — мы просто выводим данные, но никак не обрабатываем их. Как получить все имена текстовых файлов и обработать их в программе?

```
// запускаем команду ls в дочернем процессе, открываем к нему канал
на чтение
FILE *fp = popen("ls *.txt", "r");

// в цикле читаем имена всех текстовых файлов
while ((fname = fgets(buf,1024,fp);) != EOF)
{
// обрабатываем полученную строку fname
}

// закрываем канал
pclose(fp);
```



http://all-ebooks.com

Вот теперь мы готовы к тому, чтобы рассмотреть более серьезный пример. В этом примере мы будем передавать данные дочернему процессу. Задача такова: у нас есть две программы. Первая программа (листинг 11.16) передает строки данных второй, вторая (листинг 11.17) обрабатывает их и выводит результат на свой стандартный вывод.

**Листинг 11.16. Программа-родитель**

```
#include   <stdio.h>
#include   <sys/wait.h>
#include   <unistd.h>
int   main()
{
char buff[1024]={0};
// поток дочернего процесса (child process)
FILE * cp;
int status;

// Открываем канал для записи. Дочерний процесс - /usr/bin/child
cp = popen("/usr/bin/child", "w");

if (!cp)
{
  printf("Не могу открыть канал.\n");
  exit(1)
}

printf("Введите строку для передачи дочернему процессу " );

// читаем ввод пользователя
fgets(buff, sizeof(buff), stdin);

// передаем данные дочернему процессу
fprintf(cp, "%s\n", buff);
// "выталкиваем" содержимое буфера в канал
fflush(cp);

// закрываем канал и проверяем состояние вызова pclose()
status=pclose(cp);

if (!WIFEXITED(status))
   printf("ошибка при закрытии канала\n");

printf("Завершение работы родительского процесса\n");
return 0 ;
}
```

http://all-ebooks.com

Теперь несколько комментариев. Вызов `popen()` запускает дочерний процесс и устанавливает канал связи к нему. Если этого сделать почему-либо не удалось (например, отсутствует исполняемый файл `/usr/bin/child`), вызов `popen()` вернет NULL.

Вызов `fprintf()` выводит строку, принятую со стандартного ввода, в установленный канал точно так же, как если бы указатель `FILE * cp` указывал на обычный открытый файл. И точно так же во избежание задержек, связанных с буферизацией вывода, мы «выталкиваем» данные из буфера с помощью вызова `fflush()`. По окончании работы с каналом мы его закрываем и вызовом `pclose()`. При невозможности корректного закрытия канала мы выводим соответствующее предупреждение.

Код дочерней программы приведен в листинге 11.17.

**Листинг 11.17. Дочерняя программа**

```
#include   <stdio.h>
int   main()
{
    char buff[1024]={0} ;
    fgets(buff, sizeof (buff), stdin);
    printf("Прочитал со стандартного ввода: %s\n",buff);
    printf("Завершение работы дочернего процесса\n");
    return 0 ;
}
```

Если вы скомпилируете эту программу и поместите исполняемый файл в каталог `/usr/bin` под именем `child`, то, запустив родительскую программу, увидите примерно следующее:

```
Введите строку для передачи дочернему процессу hello
Прочитал со стандартного ввода: hello
Завершение работы дочернего процесса
Завершение работы родительского процесса
```

### 11.7.3. Управление каналами в PHP

В PHP также имеется функция `popen(string $command, string $mode)`, принимающая два параметра того же назначения, что ее аналог в языке C: имя запускаемой программы и режим работы канала — чтение «r» или запись «w».

Режим *«чтение»* нужно выбирать, если вы хотите *получить* информацию от дочернего процесса, а *«запись»* - если вы хотите *передать* данные порождаемому процессу. Как и в C, двусторонний обмен данными невозможен.

Пример использования функции `popen()` приведен в листинге 11.18.



http://all-ebooks.com

**Листинг 11.18. Управление каналами в PHP. Функция popen()**

```
<?
// Открываем   канал
$p = popen("/bin/ls","r");
$Files   =   array();

// Читаем вывод программы
for (;!feof($p);) $Files[] = gets($p,255);

// Закрываем   канал
pclose ($p);
?>
```

Особенностью вызова `popen()` является то, что дочерний процесс будет выполняться параллельно нашему процессу и интерпретатор не будет ждать, пока завершится дочерний процесс, а сразу перейдет на следующую строчку нашего сценария.

Теперь подведем итоги того, что сказано о каналах.

- Канал — это связь между родительским и дочерним процессом.
- Каналы можно использовать только в Unix-подобных операционных системах. (например, в Linux).
- В отличие от вызова `system()` программа, запущенная с помощью `popen()`, выполняется параллельно запустившей ее программе, то есть родительский процесс не приостанавливается, ожидая завершения дочернего.
- Вывод запущенной с помощью `popen()` программы не дублируется на стандартный вывод `(stdout)`.
- После открытия канала с ним можно работать как с обычным открытым файлом (не забывая о том, в каком режиме он открыт: на чтение или на запись).
- Канал нельзя открыть в режиме чтения и записи одновременно.

# ЧАСТЬ III.

# РАСШИРЕННЫЕ ВОЗМОЖНОСТИ PHP

http://all-ebooks.com

# ГЛАВА 12.

## PHP И БАЗЫ ДАННЫХ

http://all-ebooks.com

## 12.1. ОБЩИЕ ПОЛОЖЕНИЯ

Реляционная база данных – а именно и только такие мы будем рассматривать в этой книге – представляет собой совокупность связанных двумерных таблиц, хранящих данные. Базовым элементом таблицы является *поле*, обладающее двумя атрибутами: типом данных и хранящимся в нем значением – элементом данных. Набор полей (одного или разных типов) называется *записью*. *Таблица* – это совокупность записей одной структуры.

Программное обеспечение, которое управляет базой данных (создает новые базы данных, таблицы, а также позволяет использовать уже созданные объекты), называется системой управления базой данных (СУБД).

В PHP есть функции для работы с множеством СУБД (MS SQL Server, Oracle, Sybase и др.), но нам в большинстве случаев хватит возможностей баз данных MySQL или SQLite. MySQL – это своеобразный стандарт сетевых баз данных, пригодный для реализации самых разнообразных веб-проектов. Так уж исторически сложилось, что эти веб-проекты написаны преимущественно на языке PHP. Поэтому от связки PHP + MySQL далеко не уйдешь.

SQLite – сравнительно молодой продукт, поддержка которого появилась только в пятой версии PHP. Тесты показывают, что SQLite работает несколько быстрее, чем MySQL, но за все в этом мире нужно платить. Ценой быстродействия стали «урезанные» возможности SQLite. Например, в отличие от MySQL, SQLite не является сетевым: если база данных управляется SQLite, то она должна находиться на том же сервере, что и PHP-сценарий, тогда как база MySQL может быть расположена на другом сервере. Для большинства веб-проектов это не слишком серьезное ограничение, потому



http://all-ebooks.com

что и их база данных, и PHP-сценарий расположены на сервере хостинг-провайдера.

**Примечание.**

MySQL в узких кругах Web-программистов настолько популярен, что для него придумали даже собственное название – «мускул». Поэтому, если вы когда-нибудь услышите или увидите выражение «движок на мускуле», это означает, что сайт написан с использованием базы данных MySQL.

Стоит сразу же оговориться, что эта глава не претендует на полноту информации: СУБД MySQL и языку SQL посвящены целые тома, объемы которых превышают эту книгу. Однако для написания несложных сценариев работы с базой данных приведенного материала будет достаточно.

Как работает сервер SQL? Клиент посылает запрос, в котором он указывает, какие данные он хочет получить от сервера или какую операцию над данными он хочет выполнить; сервер посылает клиенту ответ, в котором указывается, выполнил ли сервер запрос клиента и, если сервер выполнил запрос, собственно результат запроса.

Для описания запросов клиента был разработан целый язык — SQL (Structured Query Language, структурированный язык запросов). С помощью запросов SQL вы можете:

- создавать базы данных и таблицы;
- добавлять информацию в таблицы;
- удалять информацию;
- модифицировать информацию;
- извлекать нужные вам данные.

Вот теперь можно перейти к следующему пункту, в котором рассматривается язык запросов SQL.

## 12.2. ОСНОВЫ SQL

Давайте договоримся, что будем писать запросы согласно стандарту SQL, то есть для улучшения восприятия будем разбивать их на части. Программа `mysql` допускает ввод запроса во всю строчку. Например, запрос, записанный в соответствии со стандартом SQL:

```
SELECT *
FROM S
WHERE Q > 10;
```

в программе mysql можно записать так (в конце каждого запроса должна стоять точка с запятой):

```
SELECT * FROM S WHERE Q > 10;
```

### 12.2.1. Базовая терминология

Если вы раньше работали с какой-нибудь СУБД, то смело можете пропустить этот пункт — ничего нового для себя вы не прочитаете. Но если вы сталкиваетесь с СУБД впервые, без основных терминов вам не обойтись:

- Поле (field) — это базовый элемент данных в БД. Поле имеет имя и тип. Например, поле Tovar_No, обозначающее количество товара, скорее всего, будет целым. Подробнее о типах полей мы поговорим чуть позже.
- Запись (record) — это набор полей, содержащих связанную информацию. Например, запись с полями C_No, C_Name и C_Address содержит информацию о клиенте — его номер, имя и адрес.
- Таблица (table) — это набор записей одинаковой структуры. Если у нас есть запись, состоящая из полей C_No, C_Name, C_Address, то все записи в таблице Clients будут иметь такую структуру.
- База данных (database) — это совокупность связанных таблиц. Например, в одной таблице может храниться информация о клиенте, а в другой — информация о заказе, который сделал клиент, и т. д.
- Индекс (index) — используется для быстрого поиска нужной записи в базе данных. Обычно поиск производится по значению одного поля или по значению нескольких полей.
- Первичный индекс (index) — управляет порядком отображения записей в таблице. Поле первичного индекса должно быть уникальным. Это значит, что первичный индекс возможно построить только по такому полю, для которого нет двух записей с одинаковым значением этого поля. В нашей таблице Clients, очевидно, первичный индекс должен строиться по полю C_No — по уникальному номеру клиента.
- Вторичный индекс (secondary index) — в отличие от первичного индекса может строиться по нескольким полям и необязательно должен быть уникальным. Вторичные индексы используются для связывания таблиц (об этом мы поговорим позже).
- Запрос (query) — оператор, выбирающий записи и поля из одной или нескольких таблиц. При этом выбираемые записи и поля должны соответствовать условию, заданному оператором.

Индексы также называются *ключами*. Соответственно первичный индекс называется первичным ключом, а вторичный — внешним ключом.



http://all-ebooks.com

## 12.2.2. Поля и их типы

С точки зрения MySQL (и некоторых других СУБД) база данных физически представляет собой каталог, содержащий бинарные файлы определенного формата — таблицы. Таблицы состоят из записей, а записи, в свою очередь, состоят из полей. Поле имеет два атрибута — имя и тип. Существуют следующие типы полей:

- целые;
- вещественные;
- строковые;
- бинарные;
- дата и время;
- перечисления и множества.

Названия, диапазоны и описания типов данных приведены в табл. 12.2 – 12.5.

**Таблица 12.1** *Целочисленные типы данных*

| Тип | Диапазон |
|---|---|
| TINYINT | от -128 до +127 |
| SMALLINT | от -32768 до +32767 |
| MEDIUMINT | от -8 388 608 до +8 3888 607 |
| INT | от -2 147 483 648 до 2 147 483 647 |
| BIGINT | от -9 223 372 036 854 775 808 до +9 223 372 036 854 775 807 |

Вещественные типы записываются в виде:

```
ТИП  (ДЛИНА, ЗНАКИ) [UNSIGNED]
```

ДЛИНА — это количество знакомест, в которых будет размещено все число при его передаче, а ЗНАКИ — это количество знаков после десятичной точки, которые будут учитываться. Если указан модификатор UNSIGNED, знак числа учитываться не будет.

**Таблица 12.2.** *Вещественные числа*

| Тип | Описание |
|---|---|
| FLOAT | Одинарная точность |
| DOUBLE | Двойная точность |
| REAL | То же, что DOUBLE |
| DECIMAL | Дробное число, хранится в виде строки |
| NUMERIC | То же, что DECIMAL |

Любая строка — это массив символов. При поиске с помощью оператора SELECT не учитывается регистр символов: строки «HELLO» и «HELLo» считаются одинаковыми.

Можно настроить базу данных на автоматическое перекодирование символов — в этом случае в базе данных строки будут храниться в одной кодировке, а выводиться — в другой.

В большинстве случаев вас устроит тип VARCHAR или просто CHAR, позволяющий хранить строки, содержащие до 255 символов. В скобках после типа указывается длина строки:

```
VARCHAR(20);
CHAR(20);
```

Если же 255 символов для вашей задачи не хватает, можно использовать другие строковые типы, например TEXT.

**Таблица 12.3.** *Строки*

| Тип | Описание |
|---|---|
| TINYTEXT | Максимальная длина строки — 255 символов |
| TEXT | Максимальная длина строки — 65535 символов (64 Кб) |
| MEDIUMTEXT | Максимальная длина строки — 16 777 215 символов |
| LONGTEXT | Максимальная длина строки — 4 294 967 295 символов |

Бинарные данные также можно использовать для хранения текста, но при поиске будет учитываться регистр символов. К тому же любой текстовый тип можно превратить в бинарный, указав модификатор BINARY:

```
VARCHAR(20)      BINARY
```

Если для базы данных установлена перекодировка символов, то бинарные данные в отличие от строковых «на лету» перекодироваться не будут.

**Таблица 12.4.** *Бинарные типы данных*

| Тип | Описание |
|---|---|
| TINYBLOB | Максимум — 255 символов |
| BLOB | Максимум — 65535 символов |
| MEDIUMBLOB | Максимум — 16 777 215 символов |
| LONGBLOB | Максимум — 4 294 967 295 символов |

**Таблица 12.5.** *Дата и время*

| Тип | Описание |
|---|---|
| DATE | Дата в формате ГГГГ-ММ-ДД |
| TIME | Время в формате ЧЧ:ММ:СС |



http://all-ebooks.com

**Таблица 12.5.** *Дата и время(продолжение)*

| | |
|---|---|
| TIMESTAMP | Дата и время в формате timestamp (секунд с начала эры Unix), однако при отображении оно выводится в виде ГГГГММДДЧЧММСС |
| DATETIME | Дата и время в формате ГГГГ-ММ-ДД ЧЧ:ММ:СС |

Другие типы данных MySQL рассматривать мы не будем, поскольку их использование в PHP нецелесообразно.

### 12.2.3. Создание таблиц

Создать таблицу позволяет оператор CREATE. Вот его синтаксис:

```
CREATE TABLE Имя_таблицы
(
Имя_поля1       Тип        Модификатор,
. . .
Имя_поляN       Тип        Модификатор,
[Первичный    ключ]
[Внешний    ключ]
)
```

Подробно о ключах мы поговорим позже, а пока будем использовать таблицы *без ключей*.

В качестве модификаторов можно использовать следующие значения:

- NOT NULL: поле не может содержать неопределенного значения (NULL), то есть поле должно быть явно инициализировано;
- PRIMARY KEY: поле будет первичным ключом — идентификатором записи, по которому можно однозначно идентифицировать запись;
- AUTO_INCREMENT: при вставке новой записи значение этого поля будет автоматически увеличено на единицу, поэтому в таблице никогда не будет двух записей с одинаковым значением этого поля;
- DEFAULT: задает значение, которое будет использовано по умолчанию, если при вставке записи поле не будет инициализировано явно. Значение по умолчанию задается так:

```
Имя_поля      Тип      DEFALUT      Значение
```

Например:

```
NO            INT      DEFAULT      0,
NAME          INT      DEFAULT      'Иванов'
```

Теперь создадим три таблицы — «Товар», «Клиенты» и «Заказы».

```
CREATE   TABLE CLIENTS
(
C_NO     int         NOT NULL,
FIO      char(40)    NOT NULL,
ADDRESS  char(30)    NOT NULL,
CITY     char(15)    NOT NULL,
PHONE    char(11)    NOT NULL
);
```

Таблица CLIENTS содержит поля C_NO (номер клиента), FIO (Фамилия, Имя, Отчество), ADDRESS (Адрес), CITY (Город) и PHONE (Телефон). Все эти поля не могут содержать пустого значения (NOT NULL).

```
CREATE TABLE TOVAR
(
T_NO       int                 NOT   NULL,
DESC       char(40)            NOT   NULL,
PRICE      double(9,2)         NOT   NULL,
QTY        double(9,2)         NOT   NULL
);
```

Эта таблица будет содержать данные о товарах. Поле DESC содержит название (описание) товара. PRICE – цена, QTY — количество товара на складе. Тип `double (9,2)` означает, что под значение числа отводится 9 знаков, из них два — под дробную часть.

```
CREATE            TABLE            ORDERS
(
O_NO         int             NOT   NULL,
DATE         date            NOT   NULL,
C_NO         int             NOT   NULL,
T_NO         int             NOT   NULL,
QUANTITY     double(9,2)     NOT   NULL,
AMOUNT       double(9,2)     NOT   NULL
);
```

Таблица ORDERS содержит сведения о заказах: номер заказа (O_NO), дату заказа (DATE), номер клиента (C_NO), номер товара (T_NO), количество (QUANTITY) и сумму всего заказа AMOUNT (то есть AMOUNT = QUANTITY * TOVAR.PRICE).

### 12.2.4. Добавление данных в таблицу

Для добавления записей служит оператор INSERT:

```
INSERT INTO Имя_таблицы [(Список полей)]
VALUES  (Список констант);
```



http://all-ebooks.com

Выполнение оператора INSERT создает в таблице новую запись, поля которой заполняются значениями, перечисленными в списке VALUES.

Добавим данные в только что созданные таблицы.

```
INSERT INTO CLIENTS
VALUES (1,'Иванов И.П.', 'Ленина 6', 'Кировоград','80522111111');
```

Порядок перечисления значений в списке VALUES должен совпадать с порядком, в котором соответствующие поля были перечислены при создании таблицы. Если вы хотите изменить порядок значений в операторе INSERT, то вы должны явно перечислить заполняемые поля:

```
INSERT INTO CLIENTS  (FIO,ADDRESS,C_NO,PHONE,CITY)
VALUES ('Иванов И.П.', 'Ленина 6',1,'80522111111','Кировоград');
```

Необязательно перечислять все поля записи. В общем случае вы можете добавить запись, заполнив только некоторые из ее полей, остальные при этом получат значение NULL (не определено):

```
INSERT INTO CLIENTS (C_NO, FIO)
VALUES  (2, 'Петров');
```

Однако в нашем случае сервер не выполнит такой запрос, поскольку при создании таблицы мы указали, что пустые значения полей недопустимы. Запись для клиента «Петров» придется добавлять полностью:

```
INSERT INTO CLIENTS
VALUES (2,'Петров','Пушкина 9','Кировоград','-');
```

Добавим данные в таблицу TOVAR:

```
INSERT INTO TOVAR
VALUES  (1,'Монитор LG',550.74);
```

Обратите внимание, что мы пока еще не указали первичные ключи таблицы, поэтому нам никто не мешает добавить в таблицу одинаковые записи. Добавить дату в поле DATE можно с помощью функции TO_DATE:

```
INSERT INTO ORDERS
VALUES (1,TO_DATE('01/07/06,'DD/MM/YY'),1,1,1,550.74);
```

Этот запрос означает внесение в таблицу заказа от первого июля 2006 года Ивановым И.П. (C_NO=1) одного (QUANTITY=1) монитора LG (T_NO=1).

### 12.2.5. Обновление записей

Синтаксис оператора UPDATE, который используется для обновления записей, выглядит так:

```
UPDATE Имя_таблицы
SET Поле1 = Значение1,  . . . , ПолеN = ЗначениеN
[WHERE Условие];
```

Если не задано условие WHERE, будет модифицирована вся таблица, а это может повлечь за собой непредсказуемые последствия, поскольку для всех записей будут установлены одинаковые значения полей. Поэтому всегда указывайте условие WHERE.

Предположим, что нам нужно обновить запись, если, например, клиент Иванов переехал в другой город и нам нужно отметить это событие в нашей базе данных. Это делается так:

```
UPDATE CLIENTS
SET CITY = 'Киев'
WHERE C_NO = 1;
```

Данный запрос нужно понимать так: найти запись, значение поля C_NO которой равно 1 (это код клиента Иванова), и сменить значение поля CITY на «Киев».

### 12.2.6. Удаление записей

Теперь удалим всех клиентов, номера которых превышают 10:

```
DELETE FROM CLIENTS
WHERE C_NO > 10;
```

С помощью команды DELETE можно удалить все записи таблицы, указав условие, которое подойдет для всех записей, например:

```
DELETE FROM CLIENTS;
```

Если вторая часть оператора DELETE-WHERE не указана, значит, действие оператора распространяется на все записи сразу.

### 12.2.7. Выбор записей

Добавление, изменение и удаление записей — это, безусловно, очень важные команды, но чаще всего вы будете использовать оператор SELECT, который выбирает данные из таблицы. Синтаксис этого оператора, наверное, наиболее сложен:

```
SELECT [DISTINCT|ALL] {*| [поле1 AS псевдоним]  [,..., полеN AS
псевдоним]}
FROM Имя_таблицы1 [,..., Имя_таблицыN]
[WHERE условие]
[GROUP BY список полей]  [HAVING условие]
[ORDER BY список полей]
```

Сейчас мы не будем рассматривать оператор SELECT полностью. Гораздо нагляднее будет продемонстрировать его использование на частных примерах.



http://all-ebooks.com

Например, чтобы получить все записи из таблицы CLIENTS, введите:

```
SELECT * FROM CLIENTS;
```

В результате вы получите такой ответ от сервера:

```
C_NO    FIO          ADDRESS          CITY            PHONE
1    Иванов И.П.    Ленина 6         Киев          80522111111
1    Иванов И.П.    Ленина 6         Киев          80522111111
2    Петров .       Пушкина 9        Кировоград        -
```

Обратите внимание на первые две записи — они одинаковые. Добавление одинаковых записей стало возможно благодаря тому, что мы не задали первичный ключ таблицы CLIENTS. Если вы хотите исключить одинаковые записи из ответа сервера (но не из таблицы!), введите запрос:

```
SELECT DISTINCT *
FROM CLIENTS;
```

Предположим, что вы хотите вывести только фамилию и номер телефона клиента, тогда введите такой запрос:

```
SELECT DISTINCT FIO, PHONE
FROM CLIENTS;
```

Теперь займемся усложнением наших запросов. Выведем все товары, цена которых превышает 500 рублей.

```
SELECT *
FROM TOVAR
WHERE PRICE > 500;
```

Вы можете использовать другие знаки отношений: <,>,=,<>,<=,>=.

> Если ваша компания обслуживает несколько однофамильцев, и вы хотите вывести информацию обо всех Ивановых, используйте шаблон LIKE:
>
> ```
> SELECT *
> FROM CLIENTS
> WHERE FIO LIKE '%Иванов%';
> ```

Запрос читается так: вывести всю информацию о клиентах, фамилия которых похожа на 'Иванов'. Если вы хотите выбрать данные из разных таблиц, перед именем поля нужно указывать имя таблицы. Следующий запрос выведет имена всех клиентов, которые хотя бы раз покупали у нас товар:

```
SELECT DISTINCT CLIENTS.FIO
FROM CLIENTS, ORDERS
WHERE CLIENTS.C_NO = ODREDS.C_NO;
```

Оператор SELECT позволяет использовать вложенные запросы. Однако MySQL их не поддерживает. Я это говорю для того, чтобы вы не пытались их использовать, тратя время впустую.

### 12.2.8. Дополнительные функции MIN, MAX, AVG, SUM

При работе с оператором SELECT вам доступно несколько полезных функций, вычисляющих количество элементов (COUNT), сумму элементов (SUM), максимальное и минимальное значения (MAX и MIN), а также среднее значение (AVG).

Следующие операторы выведут соответственно количество записей в таблице CLIENTS, самый дорогой товар на складе и сумму всех заказов.

```
SELECT COUNT(*)
FROM CLIENTS;

SELECT MAX(PRICE)
FROM TOVAR;

SELECT SUM(AMOUNT)
FROM ORDERS;
```

### 12.2.9. Группировка и сортировка записей

Оператор SELECT позволяет группировать возвращаемые значения. Например, клиент Иванов (C_NO=1) несколько раз заказывал у нас какой-то товар. Значит, его номер встречается в таблице ORDERS несколько раз. Другой клиент также мог сделать несколько заказов. Мы можем сгруппировать все записи по полю C_NO (номер клиента), а затем вывести сумму заказа каждого клиента.

```
SELECT CLIENTS.FIO, SUM(ORDERS.AMOUNT) AS TOTALSUM
FROM CLIENTS, ORDERS
WHERE CLIENTS.C_NO = ORDERS.C_NO
GROUP BY ORDERS.C_NO;
```

Группировку выполняет оператор GROUP BY, который является частью оператора SELECT. Оператор GROUP BY можно ограничить с помощью HAVING. Этот оператор отбирает строки, возвращаемые GROUP BY. HAVING можно считать аналогом WHERE, но только для GROUP BY:

```
HAVING   <условие>
```

Например, нас интересуют только клиенты, которые заказали товаров на общую сумму, превышающую 1000:



http://all-ebooks.com

```
SELECT CLIENTS.FIO, SUM(ORDERS.AMOUNT) AS TOTALSUM
FROM CLIENTS, ORDERS
WHERE CLIENTS.C_NO = ORDERS.C_NO
GROUP BY ORDERS.C_NO
HAVING TOTALSUM > 1000;
```

В этом запросе мы использовали псевдоним столбца TOTALSUM. В некоторых серверах SQL для определения псевдонима не нужно писать служебное слово AS, а некоторые требуют применения знака равенства:

```
SUM(ORDERS,AMOUNT)  TOTALSUM
```

*или*

```
TOTALSUM = SUM(ORDERS.AMOUNT).
```

Пока мы не установили первичный ключ, сортировка нашей таблицы не выполняется. Данные будут отображены в порядке занесения их в таблицу. Для сортировки по полю C_NO результата вывода таблицы CLIENTS служит следующий оператор (сама таблица при этом не сортируется):

```
SELECT *
FROM CLIENTS
ORDER BY C_NO;
```

## 12.2.10. Ключи

Предположим, что кто-то добавил в таблицу CLIENTS запись:

```
1    Сидоров    Егорова 11    Кировоград    80522345111
```

Получилось, что один и тот же номер (1) сопоставлен разным клиентам: Иванову и Сидорову. Кто из них будет оплачивать заказ, если в таблице заказов указан покупатель номер 1?

Для избежания подобной путаницы и нужны первичные ключи. Первичный ключ – это поле, значение которого однозначно идентифицирует запись. Понятно, что им может быть только такое поле, что в таблице нет двух записей с одинаковым значением этого поля. Первичный ключ можно указать при создании таблицы:

```
CREATE   TABLE CLIENTS
(
C_NO     int NOT NULL,
FIO      char(40) NOT NULL,
ADDRESS  char(30) NOT NULL,
CITY     char(15) NOT NULL,
PHONE    char(11) NOT NULL,
PRIMARY  KEY (C_NO)
);
```

Можно также добавить первичный ключ в уже готовую таблицу:

```
ALTER TABLE CLIENTS
ADD PRIMARY KEY (C_NO);
```

После этого запроса поле C_NO может содержать только уникальные значения. В качестве первичного ключа нельзя использовать поле, допускающее значение NULL.

Таблица ORDERS содержит сведения о заказах. По полю C_NO этой таблицы идентифицируется заказчик. Предположим, что в таблицу ORDERS кто-то ввел значение, которого нет в таблице CLIENTS: как понять, чей это заказ? Точно так же поле T_NO таблицы ORDERS обязано ссылаться на коды товаров, действительно присутствующие в таблице TOVAR.

Для организации такой связи между таблицами и введено понятие внешнего ключа. Внешний ключ – это ссылка поля одной таблицы на поле того же типа другой таблицы, являющееся ее первичным ключом, в результате чего таблицы оказываются связаны по этому полю. Наличие внешнего ключа гарантирует, что вы не сможете ввести в ссылающуюся таблицу значение, которого еще нет в ссылаемой. В этом состоит так называемая декларативная целостность базы данных.

Внешний ключ может состоять из нескольких полей. Тогда он должен ссылаться на столько же полей другой таблицы, причем поля должны быть того же типа.

Объявить внешний ключ можно как при создании таблицы, так и впоследствии, тем же запросом ALTER, что и при добавлении первичного ключа:

```
ALTER TABLE ORDERS
ADD FOREIGN KEY(C_NO) REFERENCES CLIENTS;
ALTER TABLE ORDERS
ADD FOREIGN KEY(T_NO) REFERENCES TOVAR;
```

Запрос ALTER позволяет не только добавлять ключи: он предназначен для реорганизации таблицы в целом. Вы хотите добавить еще одно поле? Или установить список допустимых значений для каждого из полей? Все это можно сделать с помощью запроса ALTER. Например, чтобы добавить в таблицу CLIENTS новое поле ZIP, введите

```
ALTER TABLE CLIENTS
ADD ZIP char(6) NULL;
```

Если в таблице уже есть хотя бы одна запись, то вы не можете добавить поле с модификатором NOT NULL.

Допустим, что наша компания работает с клиентами только из Киева и Кировограда. Тогда полезно будет ограничить допустимые значения поля CITY таблицы CLIENTS только этими двумя городами:



http://all-ebooks.com

```
ALTER TABLE CLIENTS
ADD      CONSTRAINT      INVALID_STATE      CHECK      (CITY      IN
('Кировоград','Киев'));
```

Планировать, какие таблицы будут связаны друг с другом и по каким полям, обычно можно (и нужно!) уже на этапе проектирования базы данных. Поэтому я покажу, как завести внешний ключ непосредственно при создании таблицы. Запрос CREATE в этом случае будет выглядеть следующим образом:

```
CREATE TABLE NEW_TABLE
(
/* Описания полей таблицы */
FOREIGN KEY KEY_NAME (LIST)
REFERENCES ANOTHER_TABLE  [(LIST2)]
[ON DELETE OPTION]
[ON UPDATE OPTION]
);
```

Здесь:

- KEY_NAME: имя ключа. Указывать его в принципе не обязательно, но безымянный ключ потом невозможно будет удалить. Так что для порядка имя лучше задать.
- LIST: список полей, входящих во внешний ключ, перечисленных через запятую.
- ANOTHER_TABLE: таблица, на которую ссылается внешний ключ.
- LIST2: список полей ссылаемой таблицы. Соответствие между полями внешнего ключа и полями из списка LIST2 устанавливается в порядке перечисления, и соответствующие друг другу поля должны иметь одинаковый тип. Если к тому же имена полей таблицы NEW_TABLE и соответствующих им полей таблицы ANOTHER_TABLE одинаковы, то список LIST2 можно вообще не указывать.

Необязательные параметры ON DELETE OPTION и ON UPDATE OPTION определяют, что СУБД должна сделать с записями таблицы NEW_TABLE, ссылающимися на удаленные (DELETE) или обновленные (UPDATE) поля таблицы ANOTHER_TABLE. Например, что делать с заказами клиента номер 1 в таблице ORDERS, если клиента с таким номером больше нет в таблице CLIENTS?

http://all-ebooks.com

Возможны следующие действия:

- CASCADE: удаление (обновление) записей, содержащих в поле ключа удаленное (обновленное) значение. В нашем случае удаление клиента с номером C_NO = 1 приведет к удалению из таблицы заказов всех записей о заказах, сделанных этим клиентом.
- NOACTION: запрет на удаление записи из ссылаемой таблицы, если на эту запись ссылается хотя бы одна запись в ссылающейся таблице. То есть вы не сможете удалить запись из таблицы CLIENTS до тех пор, пока в таблице ORDERS остаются записи о заказах, сделанных этим клиентом.
- SETNULL: значения всех полей в соответствующих записях ссылающейся таблицы будут заменены на NULL, если для каждого из этих полей допускается пустое значение.
- SET_DEFAULT: значения всех полей в соответствующих записях ссылающейся таблицы будут заменены на значение по умолчанию. Например, если вы укажете SET_DEFAULT 1, то при удалении клиента с любым номером его заказы будут приписываться клиенту с номером 1.

Какое действие выбрать? Несомненно, действие NOACTION является самым безопасным, а CASCADE — самым удобным для оператора.

Действие SETNULL я вообще не рекомендую использовать. Во-первых, оно небезопасно, а во-вторых, неудобно: из таблицы клиентов (основная таблица) запись удалится, а потом вам вручную придется удалять записи в таблице заказов (ссылающаяся таблица).

Действие SET_DEFAULT может быть полезно, если запись со значением поля ключа по умолчанию никогда не будет удалена из основной таблицы и, разумеется, если связывание с ней «бесхозных» записей ссылающейся таблицы имеет смысл в той предметной области, которую описывает конкретная база данных.

### 12.2.11. Удаление полей и таблиц

Как удалить поле? Стандартом SQL не предусмотрено удаление из таблицы столбцов, но MySQL всё-таки позволяет это сделать. Удалить поле ZIP, добавленное в предыдущем пункте, поможет тот же самый запрос ALTER:

```
ALTER TABLE CLIENTS
DROP ZIP;
```

Удалить таблицу еще проще:

```
DROP    ORDERS;
```



http://all-ebooks.com

### 12.2.12. Отключение от базы данных

Закончив работу с базой данных, управляемой MySQL, вы можете отключиться от нее по запросу DISCONNECT и подключиться к другой базе, введя запрос CONNECT.

Для некоторых серверов SQL запрос DISCONNECT не работает, а вместо CONNECT нужно использовать оператор USE.

При программировании на PHP запрос DISCONNECT вам не понадобится, поскольку для отключения служит функция `mysql_close()` (или `SQLite_close()`, если вы работаете с СУБД SQLite).

## 12.3. Сервер баз данных MySQL

Сервер баз данных MySQL надежен, устойчив, легок в использовании, является идеальным решением для малых и средних сетевых приложений баз данных. Вот такая краткая характеристика MySQL.

Поскольку сервер MySQL – сетевой, то у него должны быть клиентская и серверная части. Клиентская часть подключается к серверной, принимает запрос от пользователя, передает его на сервер и отображает результат – ответ от сервера.

Серверная часть получает запрос от клиента, обрабатывает его и отправляет результат обработки.

### 12.3.1. Клиентская и серверные части MySQL

Начнем с клиентской части СУБД MySQL. База данных в MySQL физически представляет собой каталог (обычно подкаталог каталога `/var/lib/mysql` в Linux или `c:\mysql\data` в Windows). Таблица – это набор файлов с расширениями frm, MYD, MYI.

Логически таблица является совокупностью записей, а запись – совокупностью полей разного типа (тип полей может повторяться). Имя базы данных уникально в пределах системы, таблицы — в пределах базы данных, а поля — в пределах таблицы.

Структуру базы данных можно просмотреть с помощью утилиты `mysqlshow`:

```
mysql show -p mysql
```

http://all-ebooks.com

Вы увидите список таблиц, которые находятся в базе данных mysql:

```
Database: mysql
+---------+
| Tables  |
+---------+
| db      |
|host     |
|user     |
+---------+
```

Утилиту `mysqlshow` можно запускать с дополнительными ключами, перечисленными в табл. 12.6.

**Таблица 12.6.** *Ключи утилиты mysqlshow*

| Ключ | Назначение |
|---|---|
| --host=hostname | Задает имя узла, на котором расположена база данных |
| --port=port_number | Определяет номер порта для сервера MySQL |
| --socket=socket | Указывает сокет |
| --user=username | Подключение к базе данных от имени указанного пользователя |
| -p | Требует ввода пароля |

Для самих же операций с данными служит программа `mysql`. Она и является клиентом сервера. Программа `mysql` принимает те же ключи, что и `mysqlshow`, а среди прочих ее ключей я хочу обратить ваше внимание только на один: «-s». Он подавляет большинство ненужных сообщений, выводимых клиентом. Я рекомендую вам использовать его всегда, особенно на медленных линиях связи.

Создать базу данных можно с помощью программы `mysqladmin`. Естественно, что использовать эту программу может только администратор, например:

```
mysqladmin -u admin -p create my_db
```

Естественно, пользователь admin, кроме того, что должен существовать, должен обладать соответствующими правами.

Каждый запрос MySQL должен заканчиваться точкой с запятой. Если вы введете

```
SELECT * FROM test
```

клиент `mysql` будет ждать ввода точки с запятой:

```
->
```



http://all-ebooks.com

### 12.3.2. Некоторые полезные приемы при работе с MySQL

Выбор кодировки
Предположим, вы сделали сайт объявлений (например, гостевую книгу) на базе MySQL. Ваш сайт написан в кодировке cp1251, и вы хотите, чтобы сообщения в базе хранились в той же кодировке. Но у вашего посетителя кодировка может быть любой!

Выполните следующие действия:

- Создайте базу данных средствами phpMyAdmin (эта программа будет рассмотрена в гл. 17).
- В phpMyAdmin введите запрос:

```
ALTER DATABASE имя_базы DEFAULT CHARACTER SET имя_кодировки
```

- Для выбора нужной кодировки сразу после подключения к MySQL добавьте в PHP-сценарий оператор:

```
mysql_query('SET NAMES имя_кодировки;');
```

В качестве имени кодировки можно использовать cp1251 или koi8_ru (не koi8-r!).

#### Резервное копирование базы данных

Резервное копирование базы данных в общем случае может быть выполнено с помощью программы mysqldump. Если база хранится на сервере вашего хостинг-провайдера, то вы не получите доступа к созданной таким образом резервной копии, поэтому для копирования вам придется использовать средства phpMyAdmin.

## 12.4. Функции PHP для работы с MySQL

### 12.4.1. Соединение PHP-сценариев с таблицами MySQL

Основной функцией является mysql_connect(), которая подключает наш сценарий к серверу баз данных и выполняет авторизацию пользователя.

```
mysql_connect([string $hostname][, string $user] [, string $password])
```

Все параметры функции являются необязательными, поскольку значения по умолчанию можно прописать в конфигурационном файле `php.ini`. Если вы хотите указать другие имя MySQL-узла, имя пользователя и пароль, вы всегда можете это сделать. Параметр `$hostname` может быть указан в виде «узел:порт».

Функция возвращает целый идентификатор соединения, служащий для всех дальнейших обращений к этому серверу БД. Следующий вызов функции `mysql_connect()` с теми же параметрами не откроет нового соединения, а вернет идентификатор существующего.

Для закрытия соединения предназначена функция `mysql_close([int $connection_id])`. Если у вас одновременно открыто несколько соединений, при вызове `mysql_close()` нужно указать идентификатор соединения, которое вы хотите закрыть. Соединение необязательно закрывать вручную: оно будет закрыто автоматически при завершении работы сценария. Помните только, что это плохой стиль программирования.

Если вы планируете использовать только одно соединение с базой данных за все время работы сценария, его идентификатор можно не сохранять и не указывать при вызове остальных функций.

Функция `mysql_connect()` устанавливает обыкновенное соединение с базой данных. Однако PHP позволяет создавать так называемые *постоянные соединения* — для этого служит функция `mysql_pconnect()`. Параметры у этой функции такие же, как и у `mysql_connect()`.

В чем разница между обыкновенным и постоянным соединением? Постоянное соединение не закрывается после завершения работы сценария, даже если сценарий вызвал функцию `mysql_close()`. Соединение «привязывается» к PID процесса-потомка Apache, от имени которого оно работает, и закрывается лишь тогда, когда удаляется процесс-владелец (например, перезагрузка или завершение работы сервера Apache).

PHP работает с постоянными соединениями примерно так: при вызове функции `mysql_pconnect()` PHP проверяет, было ли раньше установлено соединение. Если да, то возвращается его идентификатор, а если нет, открывается новое соединение и также возвращается идентификатор.

Недостатков у постоянных соединений практически нет. Постоянные соединения позволяют значительно снизить нагрузку на сервер, а также повысить скорость работы сценариев, использующих базу данных.

При работе с постоянными соединениями нужно следить, чтобы максимальное число клиентов Apache не превышало максимального числа кли-



http://all-ebooks.com

ентов MySQL, то есть значение параметра `MaxClients` в файле `httpd.conf` должно быть меньше или равно значению `max_user_connection` (параметр MySQL).

### 12.4.2. Выбор базы данных

Функция `mysql_select_db(string $db [, int $id])` выбирает базу данных, с которой будет работать сценарий. Если открыто только одно соединение, параметр `$id` можно не указывать.

Пример подключения к базе данных:

```
// Пытаемся установить соединение
if (!mysql_connect($SERVER,$USER,$PASSWD))
{
  echo "Не могу подключиться к серверу";
  exit;
}
// Выбираем базу данных
mysql_select_db($DB);
```

### 12.4.3. Обработка ошибок

Если произойдет ошибка, мы получим сообщение «Не могу подключиться к серверу», и все: сценарий завершит свою работу. Не очень-то удобно. Во время отладки сценария для уточнения подробностей ошибки можно воспользоваться двумя функциями:

- `mysql_errno(int $id);`
- `mysql_error(int $id);`

Первая функция возвращает номер ошибки, а вторая — сообщение, описывающее эту ошибку. Вместо того чтобы реагировать на все ошибки одним и тем же сообщением, лучше использовать такой вызов:

```
echo "ERROR ".mysql_errno().": ".mysql_error()."\n";
```

Теперь вам не нужно гадать, из-за чего произошла ошибка — вы сразу увидите сообщение об ошибке.

### 12.4.4. Обработка результата SQL-запроса

Все запросы к текущей базе данных отправляются функцией `mysql_query(string $query)`, принимающей один параметр: текст

запроса на языке SQL. Текст запроса может содержать пробельные символы и символы новой строки (\n).

Пример запроса:

```
$r = mysql_query("select * from hbd");
```

Функция mysql_query() возвращает идентификатор результата выполнения запроса, то есть набора данных, который сценарий должен обработать определенным образом. В нашем примере запрос вернет набор, состоящий из всех записей таблицы hbd. Вызов функции mysql_query() присвоит переменной $r идентификатор этого набора, по которому сценарий в дальнейшем будет обращаться к этому набору для его дальнейшей обработки.

Обработка полученных данных начинается с извлечения из набора отдельных записей, для чего в PHP предусмотрены следующие функции:

- mysql_result(): получить нужный элемент из набора записей;
- mysql_fetch_array(): занести запись в ассоциативный массив или числовой массив;
- mysql_fetch_row(): занести запись в массив;
- mysql_fetch_assoc(): занести запись в ассоциативный массив;
- mysql_fetch_object(): занести запись в объект.

Перед тем как начинать разбор полученного результата, обычно бывает необходимо определить количество содержащихся в нем записей и полей. Функция mysql_num_rows() позволяет узнать, сколько записей содержит наш результат:

```
$r = mysql_query("select * from hbd");
echo "В таблице hbd ".mysql_num_rows($r)." записей";
```

Запись состоит из полей. Узнать, сколько полей содержит каждая запись результата, позволяет функция mysql_num_fields():

```
$r = mysql_query("select * from hbd");
echo "В таблице hbd ".mysql_num_fields($r)." полей";
```

Мы уже можем получить количество полей таблицы и количество записей в выборке. Теперь можно извлечь значение конкретного поля конкретной записи, вызвав функцию mysql_result(int $result, int $row, mixed $field), где параметр $row указывает номер записи, а $field — имя или порядковый номер поля.

Предположим, что наш запрос вернул следующий набор данных:



http://all-ebooks.com

```
Email                    FirstName                    LastName
-------------------------------------------------------------
ivan@mail.ru             Ivan                         Ivanov
sid@mail.ru              Petr                         Sidorov
```

Следующий фрагмент кода позволяет вывести его в окно браузера:

```
$rows = mysql_num_rows($r);
$cols = mysql_num_fields($r);

echo "<PRE>";
for ($i=0; $i<$rows; $i++)
{
  for (j=0; $j<$cols; $j++)
      echo mysql_result($r, $i, $j)."\t";
      echo "\n";
}
echo "</PRE>";
```

Стоит отметить, что функция `mysql_result()` универсальна: она позволяет извлечь из результата запроса значение любого поля любой записи, то есть, зная количество записей и количество полей, вы можете «обойти» весь результат. Понятно, что этот способ разбора результата достаточно медленный. Поэтому для обработки больших наборов записей рекомендуется использовать функции, имя которых содержит «fetch»: `mysql_fetch_row()`, `mysql_fetch_array()` и т. п.

Функция `mysql_fetch_row(int $res)` возвращает строку, содержащую все поля текущей записи результата `$res`. Каждый следующий вызов функции перемещает указатель текущей записи на следующую позицию (как при работе с файлами) и возвращает следующую запись. Если записей больше нет, то возвращается значение `false`.

Пример использования:

```
$r=mysql_query("select * from hbd where month=\"$bd_m\" and day=\
"$bd_d\");


for ($i=0; $i<mysql_num_rows($r); $i++)
{
  $f=mysql_fetch_row($r);
  echo $f;
}
```

**Примечание.**

Обратите внимание на строку запроса: текстовые значения полей обязательно должны быть в кавычках — это требование MySQL.

Использовать функцию `mysql_fetch_row()` не всегда удобно, поскольку она возвращает все поля в од-

ной строке. Намного удобнее использовать функцию `mysql_fetch_array(int $res [, int $array_type=MYSQL_BOTH])`, возвращающую текущую запись в виде ассоциативного массива, ключами которого будут имена полей. Точнее будет сказать, что ассоциативный массив эта функция возвращает по умолчанию, а необязательный параметр `$array_type` позволяет указать более удобную для вас разновидность возвращаемого массива:

- MYSQL_ASSOC — ассоциативный массив;
- MYSQL_NUM — массив с числовыми индексами;
- MYSQL_BOTH — массив с двойными индексами, то есть такой, с которым можно работать и как с ассоциативным массивом, и как со списком.

В PHP есть одна функция, которая возвращает «чистый» (с одним индексом) ассоциативный массив: `mysql_fetch_assoc(int $res)`. Эта функция — просто синоним для `mysql_fetch_array($res, MYSQL_ASSOC)`.

Пример использования функции `mysql_fetch_array()`:

```
$r=mysql_query("select * from hbd where month=\"$bd_m\" and day=\
"$bd_d\");

for $i=0; $i<mysql_num_rows($r); $i++)
{
  $f=mysql_fetch_array($r);
  echo "$f[email] $f[name] $f[month] $f[day] $f[sex] <br>";
}
```

Как видите, использовать функцию `mysql_fetch_array()` намного удобнее, чем `mysql_fetch_row()`.

### 12.4.5. Получение информации о результате SQL-запроса

Мы рассмотрели практически все необходимые вам для работы функции. Однако PHP предоставляет еще несколько полезных функций, которые позволяют узнать информацию о результате.

Функция `mysql_field_name(int $result, int $offset)` возвращает имя поля, находящегося в результате `$result` по номеру `$offset`. Заметьте, что `$offset` – это номер столбца не в таблице, а именно в результате запроса. Нумерация полей начинается с 0.



http://all-ebooks.com

Функция `mysql field_type(int $result, int $offset)` возвращает тип поля с номером `$offset` в результате `$result`.

Функция `mysql_field_flags(int $result, int $offset)` возвращает строку, содержащую флаги (модификаторы) поля с номером `$offset`, разделенные пробелами. Все флаги, поддерживаемые MySQL, перечислены в табл. 12.7.

**Таблица 12.7.** *Флаги полей MySQL*

| Флаг | Назначение |
|---|---|
| not_null | Поле не может содержать неопределенного значения (NULL), то есть поле должно быть явно инициализировано |
| primary_key | Поле является первичным ключом |
| auto_increment | При вставке новой записи значение этого поля будет автоматически увеличено на единицу, поэтому в таблице никогда не будет двух записей с одинаковым значением этого поля |
| unique_key | Поле должно содержать уникальное значение |
| multiple_key | Индекс |
| blob | Поле может содержать бинарный блок данных |
| unsigned | Поле содержит беззнаковые числа |
| zerofill | Пробелы заменяются символами с кодом \0 |
| binary | Поле содержит двоичные данные |
| enum | Поле может содержать один элемент из нескольких возможных (элемент перечисления) |
| timestamp | В поле автоматически заносится текущая дата и время при его модификации |

Рассмотрим полноценный пример. Наш сценарий (листинг 12.1) получает содержимое одной таблицы БД и выводит его в окно браузера, оформив как таблицу HTML.

***Файл был скачан с http://all-ebooks.com***

**Листинг 12.1. Вывод содержимого таблицы БД**

```
<?
$SERVER = "localhost";
$USER = "user";
$PASSWD = "12345";
$DB = "my_db";

echo "<html><head></head><body>;
```

http://all-ebooks.com

```
//    Подключаемся
if(!mysql_connect($SERVER,$USER,$PASSWD))
{
echo "<h2>ERROR</h2>";
echo "</body></html>";
exit;
}

// Выбираем базу данных
mysql_select_db($DB);

// Выводим заголовок таблицы
echo "<table border=1 width=100% bgcolor=gold>";
echo "<tr><td>E-mail</td><td>Имя</td><td>Месяц</td>";
echo "<td>Число</td><td>Пол</td></tr>";

// Посылаем запрос
$r=mysql_query("select * from hbd");

// Выводим таблицу
for ($i = 0; $i<mysql_num_rows($r); $i++)
{
  echo "<tr>";

  $f=mysql_fetch_array($r);
  echo "<td>$f[email]</td><td>$f[name]</td><td>$f[month]</td>";
  echo "<td>$f[day]</td><td>$f[sex]</td>";

  echo "</tr>";
}
echo "</table></body></html>";
?>
```

Еще одна полезная функция, которая может пригодиться вам при написании ваших сценариев — это функция `mysql_list_tables(string $db [, int connect_id])`. Функция возвращает идентификатор результата (одна колонка), содержащего имена всех таблиц в указанной базе данных. Второй параметр, как обычно, это идентификатор соединения:

```
$r = mysql_list_tables("hbd");
while($row = mysql_fetch_row($r))
{
   echo $row."\n";
}
```



http://all-ebooks.com

## 12.5. СЕРВЕР БАЗ ДАННЫХ SQLITE

На сегодня самым популярным сервером баз данных, который чаще всего используется в связке с PHP, является MySQL. Его популярности способствовали три фактора:

- бесплатность;
- включение в состав большинства дистрибутивов Linux, благодаря чему он установлен на серверах большинства хостинг-провайдеров;
- встроенные в интерпретатор PHP функции для работы с MySQL.

Перед выпуском пятой версии PHP поддержка MySQL из-за лицензионных несогласий чуть было не была удалена из PHP. Вышла даже бета-версия PHP 5, которая не поддерживала MySQL. Вместо MySQL та версия поддерживала SQLite – вот тогда-то этот продукт и засветился в мире PHP-программирования. Если до этого о SQLite только начинали говорить, то после о нем уже заговорили на серьезном уровне.

Лицензионные разногласия были в итоге улажены, и множество сайтов, написанных поддержкой MySQL, не пришлось портировать на SQLite, потому что окончательная версия PHP5 стала поддерживать оба. В Интернете можно найти как коммерческий, так и бесплатный хостинг, на котором установлены оба сервера баз данных.

Пятая версия PHP для Windows, которую вы можете скачать с моего сайта `http://www.dkws.org.ua`, поддерживает и MySQL, и SQLite.

### 12.5.1. Преимущества и недостатки SQLite

SQLite, как и любой другой программный продукт, обладает своими преимуществами и недостатками. Основное преимущество и основной недостаток SQLite – это высокая производительность. С преимуществом-то все понятно, а вот почему производительность считается недостатком?

Дело в том, что производительность SQLite достигается не за счет уникального алгоритма, а из-за его ограниченных возможностей. В отличие от MySQL, SQLite является локальным сервером баз данных, то есть он «умеет» использовать базы данных, размещенные на локальном диске того компьютера, на котором выполняется ваш PHP-сценарий. Используя программу-клиент MySQL, мы можем подключаться к удаленным серверам MySQL, то есть не обязательно, чтобы наша база данных физически была

расположена на одном компьютере со сценарием. Может, для вас это и не критично, но я считаю, что SQLite уступает MySQL в универсальности.

Если быть предельно точным, вот основные свойства SQLite:

- поддерживает большинство спецификаций SQL92;
- база данных (включая таблицы и индексы) хранится в одном файле;
- максимальный размер файла базы данных – 2 терабайта;
- существуют версии для Linux и для Windows;
- файлы баз данных, созданные в Linux, могут использоваться в Windows и наоборот;
- работает в два раза быстрее, чем PostgreSQL и MySQL.

Чтобы не быть голословным, приведу результаты тестирования. Тестирование производилось на машине со следующей конфигурацией: AMD 1.6 ГГц Athlon, 1 Гб ОЗУ, жесткий диск EIDE. В качестве операционной системы использовалась Red Hat Linux 7.2, версии PostgreSQL и MySQL - 7.1.3 и 3.23.41 соответственно. Версия SQLite – 2.7.6.

Вот первый тест (вставка 1000 записей):

```
CREATE TABLE t1(a INTEGER, b INTEGER, c VARCHAR(100));
INSERT INTO t1 VALUES(1,13153,'thirteen thousand one hundred fifty
```

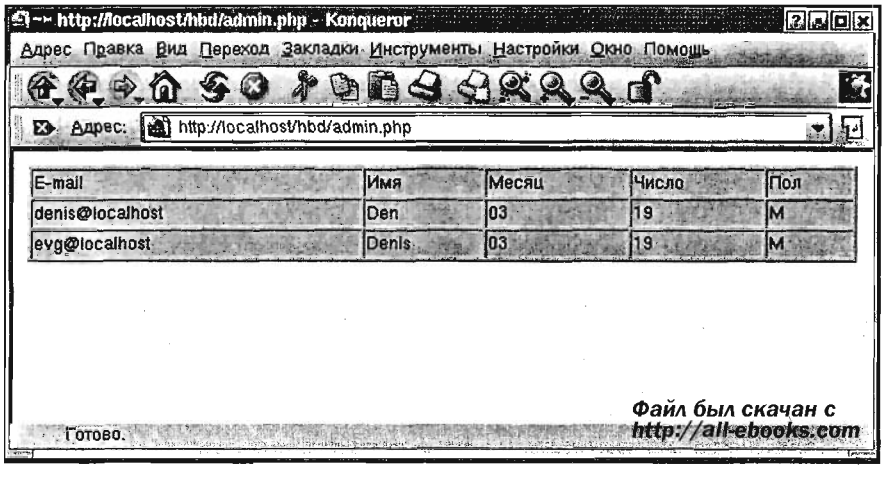



***Рис. 12.1. Вывод сценария, отображающего страницу***



http://all-ebooks.com

```
three');
INSERT INTO t1 VALUES(2,75560,'seventy five thousand five hundred
sixty');
... 995 строк пропущено
INSERT INTO t1 VALUES(998,66289,'sixty six thousand two hundred
eighty nine');
INSERT INTO t1 VALUES(999,24322,'twenty four thousand three hundred
twenty two');
INSERT INTO t1 VALUES(1000,94142,'ninety four thousand one hundred
forty two');
```

А вот результаты:

Второй тест – выборка без индексов:

```
BEGIN;
SELECT count(*), avg(b) FROM t2 WHERE b>=0 AND b
SELECT count(*), avg(b) FROM t2 WHERE b>=100 AND b
... 96 строк пропущено
SELECT count(*), avg(b) FROM t2 WHERE b>=9800 AND b
SELECT count(*), avg(b) FROM t2 WHERE b>=9900 AND b
COMMIT;
```

Третий тест – выборка со сравнением строки:

```
BEGIN;
SELECT count(*), avg(b) FROM t2 WHERE c LIKE '%one%';
SELECT count(*), avg(b) FROM t2 WHERE c LIKE '%two%';
... 96 строк пропущено
SELECT count(*), avg(b) FROM t2 WHERE c LIKE '%ninety nine%';
SELECT count(*), avg(b) FROM t2 WHERE c LIKE '%one hundred%';
COMMIT;
```

Как вы видите, при использовании простых запросов SQLite ненамного ушел от MySQL. В качестве сложного запроса будем использовать запрос, состоящий из 25000 операторов обновления (UPDATE):

```
BEGIN;
UPDATE t2 SET c='one hundred forty eight thousand three hundred
eighty two' WHERE a=1;
UPDATE t2 SET c='three hundred sixty six thousand five hundred two'
WHERE a=2;
... 24996 строк пропущено
UPDATE t2 SET c='three hundred eighty three thousand ninety nine'
WHERE a=24999;
UPDATE t2 SET c='two hundred fifty six thousand eight hundred thirty'
WHERE a=25000;
COMMIT;
```

Более подробно о тестировании вы можете прочитать на сайте `www.sqlite.org`.

### 12.5.2. Установка SQLite

Пятая версия PHP обладает встроенной поддержкой SQLite, поэтому вам не нужно производить никаких действий по подключению SQLite к PHP5: просто загрузите SQLite и работайте.

Скачать SQLite для Linux можно по адресам:

- `http://pecl.php.net/package/SQLite`;
- `http://sqlite.org`.

SQLite для Windows реализован в виде одной библиотеки `php_sqlite.dll`, скачать которую вы можете по адресу `http://snaps.php.net/win32/PECL_STABLE/php_sqlite.dll`.

Установка SQLite проблем не вызывает, поэтому подробно останавливаться на ней мы не будем. Если вы устанавливаете SQLite для Windows, то скачайте файл `php_sqlite.dll` и «пропишите» его в файле конфигурации `php.ini`:

```
extension=sqlite.dll
```

Сам файл нужно скопировать в каталог, заданный директивой `extension_dir` файла конфигурации `php.ini`.

Для установки SQLite для Linux скачайте файл `sqlite.tar.gz`. Затем введите следующую последовательность команд:

```
tar xzf sqlite.tar.gz  # Распаковываем архив в каталог "sqlite"
mkdir bld              # Создаем каталог для сборки SQLite
cd bld
../sqlite/configure    #  Запускаем конфигуратор
make                    #  Запускаем сборку SQLite
```

Сценарий конфигуратора `configure` для своей работы использует `autoconf 2.50` и `libtool`, поэтому убедитесь, что эти пакеты у вас установлены.

Конфигуратор проверит, установлены ли все необходимые для сборки и работы SQLite библиотеки и программы. Если чего-то не хватает, вы увидите соответствующее сообщение. Если же ваша система соответствует требованиям конфигуратора, вы можете запускать программу make.
После сборки SQLite запустите сценарий `install-sh` для установки SQLite, вспомогательных программ и сценариев, а также базы данных. Вспомогательными утилитами являются:



http://all-ebooks.com

- `showdb` – утилита для просмотра базы данных;
- `showjournal` – утилита для просмотра журнала базы данных;
- `diffdb` – утилита сравнения двух баз данных.

## 12.6. ФУНКЦИИ И КОНСТАНТЫ PHP ДЛЯ РАБОТЫ С SQLITE

Прежде чем перейти к рассмотрению функций для работы с SQLite, рассмотрим встроенные константы SQLite:

- SQLITE_ASSOC – результат будет возвращаться в виде ассоциативного массива;
- SQLITE_NUM – результат будет возвращен в виде списка;
- SQLITE_BOTH – двойное индексирование элементов массива: вы можете работать с массивом так, как если бы он был ассоциативным, и так, как если бы он был списком.

Эти константы действуют аналогично константам MYSQL_ASSOC, MYSQL_NUM и MYSQL_BOTH,.

### 12.6.1. Список функций SQLite

Теперь перейдем к рассмотрению списка функций (я их упорядочил в алфавитном порядке):

- `sqlite_array_query()` – выполняет SQL-запрос и возвращает результат – массив, тип которого задается одной из констант, перечисленных выше;
- `sqlite_busy_timeout()` – устанавливает тайм-аут;
- `sqlite_changes()` – возвращает число записей, которые были изменены в результате последнего SQL-запроса;
- `sqlite_close()` – закрывает базу данных;
- `sqlite_column()` – получает одно поле из записи, то есть возвращает один столбец из результата запроса;
- `sqlite_current()` – получает текущую запись из результата;

http://all-ebooks.com

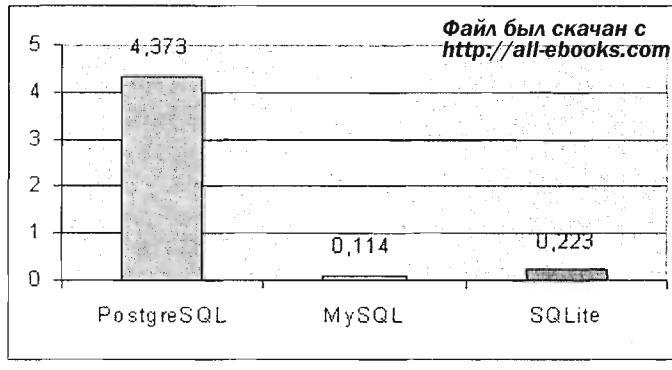



*Рис. 12.2. Результаты первого теста (в секундах)*

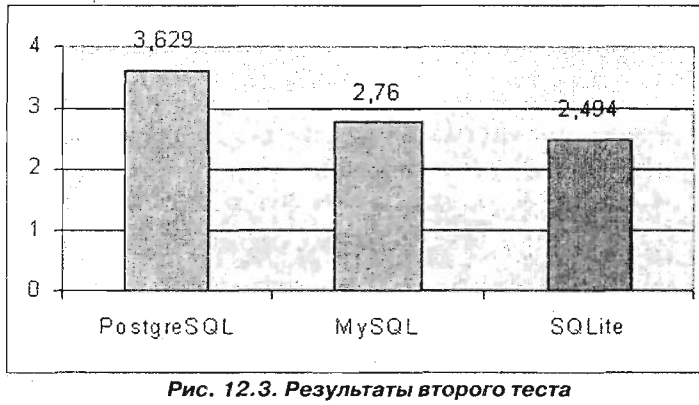



*Рис. 12.3. Результаты второго теста*

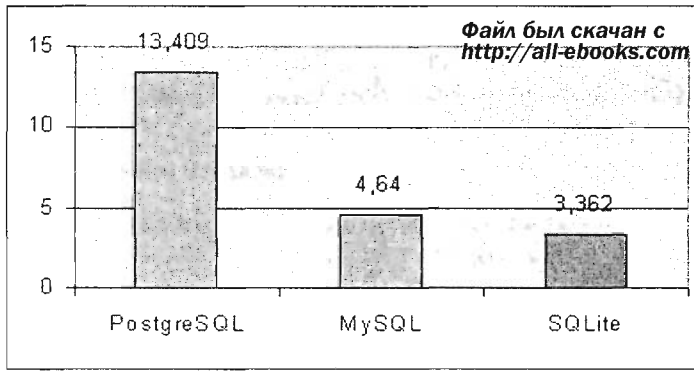



*Рис. 12.4. Результаты третьего теста*



http://all-ebooks.com

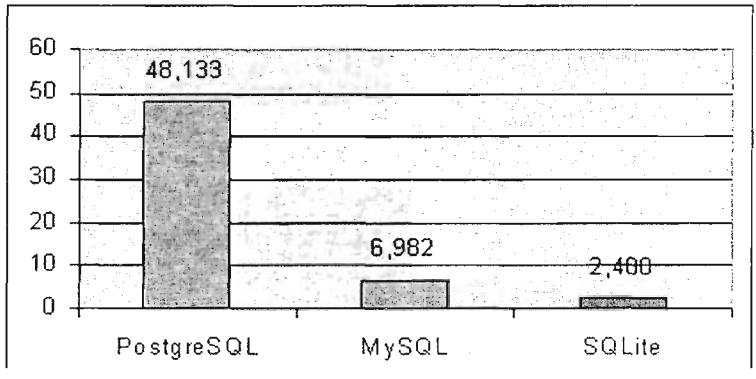



***Рис. 12.5. Результат четвертого теста***

- `sqlite_error_string()` — возвращает текстовое описание ошибки;
- `sqlite_fetch_array()` — получает следующую запись из результата в виде массива;
- `sqlite_fetch_single()` — возвращает значение первого поля из результата запроса в виде строки;
- `sqlite_fetch_string()` — псевдоним для `sqlite_fetch_single()`;
- `sqlite_field_name()` — возвращает имя поля;
- `sqlite_has_more()` — позволяет узнать, есть ли еще записи;
- `sqlite_last_error()` — возвращает код последней ошибки;
- `sqlite_last_insert_rowid()` — возвращает идентификатор последней добавленной записи;
- `sqlite_libversion()` — возвращает версию SQLite;
- `sqlite_next()` — переходит к следующей записи;
- `sqlite_num_fields()` — возвращает число полей (столбцов) в результате;
- `sqlite_num_rows()` — возвращает число записей в результате;
- `sqlite_open()` — открывает базу данных;
- `sqlite_popen()` — открывает постоянное «соединение» с базой данных. Если база данных не существует, она будет создана;

http://all-ebooks.com

- `sqlite_query()` — выполняет SQL-запрос и возвращает идентификатор результата;
- `sqlite_rewind()` — переходит к первой записи результата;
- `sqlite_seek()` — переходит к записи с указанным номером.

Я перечислил не все функции SQLite, однако перечисленных функций вам должно вполне хватить для работы с SQLite. Рассмотрим эти функции подробнее.

### 12.6.2. Открытие и закрытие базы данных

Для открытия базы данных служит функция

```
sqlite_open (string $filename [, int $mode [, string &error_message]])
```

Первый параметр — это имя файла таблицы, второй — режим доступа к таблице. По умолчанию используется режим 0666, позволяющий читать и записывать файл. Последний параметр, передаваемый по ссылке, — это строка, в которую будет записано сообщение об ошибке, если базу данных невозможно открыть. Функция возвращает дескриптор базы данных.

Функция `sqlite_popen()` с тем же набором параметров открывает постоянное соединение с базой данных. Постоянное соединение не разрывается при завершении работы сценария. При следующем запуске сценария функция `sqlite_popen()` ищет постоянное соединение; если оно существует, то она будет использовать его, а не открывать базу данных заново. При частых запусках сценария это позволяет повысить его производительность.

Функция `sqlite_close(resource $db)` принимает один параметр — *дескриптор открытой базы данных*.

### 12.6.3. Выполнение запросов

Функция `sqlite_query()` позволяет выполнить запрос к базе данных:

```
sqlite_query (resource $dbhandle, string $query)
sqlite_query (string $query, resource $dbhandle)
```

Обратите внимание, что функцию можно вызывать двояко: или сначала указывать дескриптор базы данных, а потом — SQL-запрос, или наоборот. Пример использования приведен в листинге 12.2.



http://all-ebooks.com

**Листинг 12.2. Открытие и закрытие базы данных**

```
<?
// открываем базу данных
if ($db = sqlite_open('mysqlitedb', 0666, $sqliteerror)) {
// создаем таблицу mytable
sqlite_query($db,'CREATE TABLE mytable (name varchar (20))');
// Добавляем в таблицу новые записи
sqlite_query($db,"INSERT INTO mytable VALUES ('Иванов И.И.')");
// Получаем результат
$result = sqlite_query($db,'select name from mytable');
// Выводим результат
var_dump(sqlite_fetch_array($result));
} else {
die ($sqliteerror);
}
?>
```

Если запрос невозможно выполнить, функция `sqlite_query()` возвращает `false`.

### 12.6.4. Функции для обработки результата запроса

Функция `sqlite_column (resource $result, mixed $index_or_name [, bool $decode_binary=true])` служит для извлечения из результата запроса одного столбца.

Первый аргумент — это дескриптор результата, второй — имя или номер поля, а третий указывает на необходимость декодирования данных, которые были закодированы функцией `sqlite_escape_string()`. Рекомендуется использовать значение по умолчанию всегда, кроме обращения к базам данных, созданным в других приложениях.

Функция `sqlite_fetch_array (resource $result [, int $type=SQLITE_BOTH [, bool $decode_binary=true]])` выбирает следующую запись из результата запроса, заданного аргументом `$result`, и возвращает ее в виде массива. Если результат не содержит больше данных, функция возвращает `false`.

Аргумент `$type` позволяет задать вид возвращаемого массива. Его значением может быть одна из констант SQLITE_ASSOC, SQLITE_NUM и (по умолчанию) SQLITE_BOTH.

Вызов функции `sqlite_array_query (resource $dbhandle, string $query [, int $type [, bool $decode_binary]])` позволяет сэкономить на вызове `sqlite_query()`: эта функция сразу и выполнит запрос, и извлечет из результата каждую запись в виде массива. Таким

образом, вместо сценария в листинге 12.3 можно написать единственный вызов:

```
$rows = sqlite_array_query ($db, 'SELECT * FROM mytable');
```

И он будет выполнен значительно быстрее!

**Листинг 12.3. Сценарий, эквивалентный sqlite_array_query()**

```
<?
$res = sqlite_query($db, 'SELECT * FROM mytable');
$rows = array();
while ($onerow = sqlite_fetch_array($res)) {
    $rows[] = $onerow;
}
?>
```

Функция `sqlite_fetch_single (resource $result [, bool $decode_binary]])`, как и `sqlite_fetch_array()`, выбирает из результата запроса следующую запись, но возвращает только первое ее поле в виде строки. Применение этой функции продемонстрировано в листинге 12.4.

**Листинг 12.4. Функция sqlite_fetch_single()**

```
<?
if ($db = sqlite_open('mysqlitedb', 0666, $sqliteerror))
{
  $query = 'SELECT * FROM mytable';
  $res = sqlite_query($db , $query);

  if (sqlite_num_rows($res) > 0) {
     echo sqlite_fetch_single($res);          // выведет: «Иванов
И.И.»
  }
  sqlite_close($db);
}
?>
```

Функция `sqlite_next(resource $result)` перемещает указатель текущей записи результата на следующую запись и возвращает `true`, если это удалось, и `false`, если достигнута последняя запись.

Функция `sqlite_rewind(resource $result)` перемещает указатель текущей записи на первую запись результата.

Функция `sqlite_seek(resource $result, int $rownum)` переходит к записи с указанным порядковым номером и возвращает `true`, если запись с этим номером найдена, и `false` в противном случае.



http://all-ebooks.com

### 12.6.5. Информационные функции

Функция `sqlite_changes(resource $db)` возвращает количество записей, которые были изменены последним SQL-запросом.

Функция `sqlite_field_name(resource $result, int $field_index)` возвращает имя поля по его порядковому номеру в результате запроса.

Функция `sqlite_has_more(resource $result)` возвращает `true`, если результат содержит еще записи, доступные для выборки, и `false` в противном случае.

Функция `sqlite_last_error(resource $db)` возвращает целочисленный код последней ошибки, а `sqlite_error_string(resource $db)` – строку ее описания.

Функции `sqlite_num_rows(resource $result)` и `sqlite_num fields(resource $result)` возвращают количество записей и полей в результате соответственно.

Функция `sqlite_busy_timeout(resource $db, int $milliseconds)` позволяет установить время, в течение которого SQLite ожидает готовности базы данных. По истечении этого времени, если файл базы данных заблокирован другим процессом, обратившаяся к нему функция SQLite завершится с ошибкой SQLITE_BUSY. Время задается в в миллисекундах и по умолчанию устанавливается при открытии базы данных равным 60 секундам.

## 12.7. Постраничный вывод из базы данных

Довольно часто нам нужно вывести сразу не все записи в базе данных, а организовать постраничный вывод. Во-первых, когда вы выводите, скажем, по 30 записей, а не сразу все 1000, то вывод осуществляется намного быстрее. Во-вторых, пользователю удобнее просматривать не все несколько сотен или даже тысяч записей, а только некоторые из них.

Сейчас мы рассмотрим код из реального проекта в Интернете, написанного мной. Поскольку проект коммерческий, то, чтобы не делать ему бесплатной рекламы в книге, его адрес в строке браузера я сотру. Но код будет полностью рабочим.

Суть проекта заключается в том, что имеется база данных с сотнями бизнес-предложений. Пользователь заходит на сайт, выбирает сферу бизнеса, например «Информационные технологии», и получает список предложений

других пользователей, касающихся бизнеса в сфере информационных технологий. Представьте себе лицо пользователя, когда он увидит сразу весь список. Конечно, можно воспользоваться поиском, но это уже потом, когда его браузер получит все несколько сотен предложений, каждое из которых – это не две строчки и номер телефона, а довольно подробное описание того или иного проекта. Если у пользователя к тому же медленное соединение (скажем, модем), то спасибо, я думаю, он вам не скажет.

Как же организовать постраничный вывод? Первое, что приходит на ум, – это получить все предложения из заданной пользователем области, а потом, в зависимости от переданных пользователем параметров, выводить ту или иную часть массива результата. Но в этом случае мы нерационально используем ресурсы нашего сервера баз данных, поскольку при любом обращении к сценарию будут получены все несколько сотен записей. Это первое. А второе – все это время пользователю нужно будет ждать, пока сценарий получит эти записи. Да, за трафик пользователь платить не будет (поскольку эти несколько сотен предложений не будут загружены на его компьютер), но подождать ему все-таки придется.

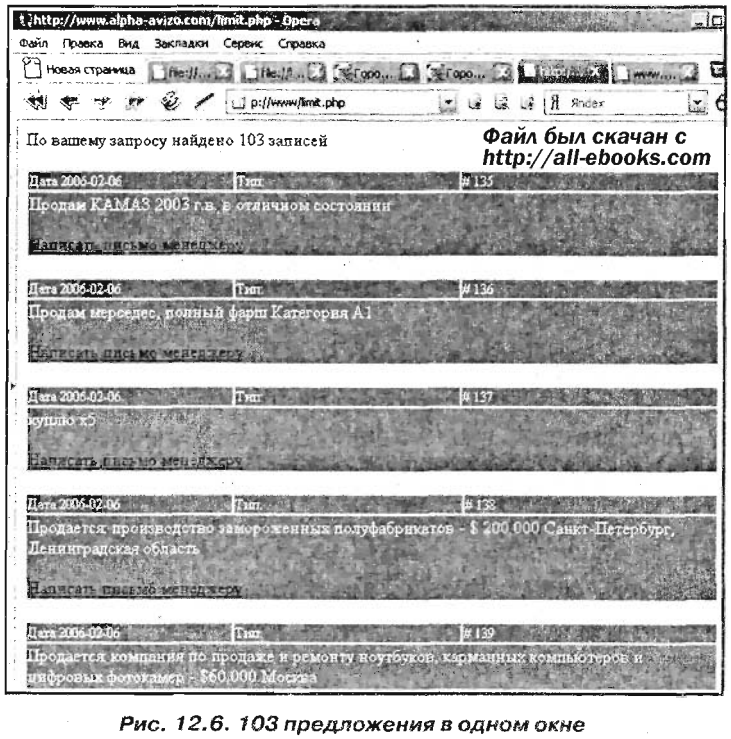



*Рис. 12.6. 103 предложения в одном окне*



http://all-ebooks.com

Попробуем сделать то же самое, но более рационально, то есть не перегружая сервер. У оператора SELECT есть инструкция LIMIT, позволяющая указывать ограничения на выбор записей из базы данных. Использовать ее нужно так:

```
SELECT … LIMIT <начальная_запись>, <смещение>
```

Смещение задает количество записей, которые нужно извлечь из таблицы, считая от начальной записи. Например:

```
SELECT * FROM prop LIMIT 0, 10  // вывести первые 10 записей
SELECT * FROM prop LIMIT 10, 10  // вывести следующие 10 записей...
```

Как несложно догадаться, в этом случае мы с помощью несложных SQL-запросов организовали постраничную выборку из таблицы prop по 10 записей. Рассчитывайте количество выбираемых записей исходя из того, сколько их может поместиться в окне браузера.

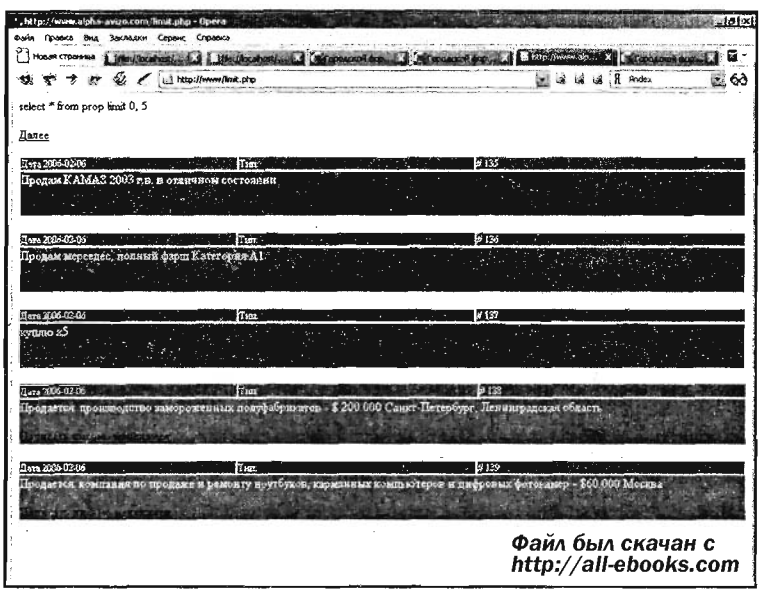
В листинге 12.5 приведен код сценария, выводящего по 5 записей на страницу.

**Листинг 12.5. Постраничный вывод из таблицы**

```
<?
mysql_connect("localhost","user","password");
mysql_select_db("bp");

mysql_query('SET NAMES cp1251;');  // выбираем кодировку

$N = 5;  // Показываем по 5 записей

// переменная $page задает номер страницы, которую мы должны
вывести
if (!isset($page)) $page=0;

// номер первой записи, выводимой на страницу $page
$records = $page * $N;

// SQL-запрос
$q = "select * from prop limit ".$records.", $N";

echo $q;  // отладочный вывод запроса
$r = mysql_query($q);  // выполняем запрос
$n = mysql_num_rows($r);    // получаем количество извлеченных
записей

$page++;
```

http://all-ebooks.com

```
// показываем ссылку на следующую страницу
echo "<p><a href=limit.php?page=$page>Далее</a>";

// выводим записи в соответствии с требованиями дизайна сайта
for ($i=0; $i<$N; $i++)
{
  $f = mysql_fetch_array($r);

  echo "<p><table width=100%>
    <tr><td bgcolor=5685a2 width=30%>
      <font name=tahoma size=2 color=white>Дата $f[dt]</font>";
  echo "<td bgcolor=5685a2>
   <font name=tahoma size=2 color=white>Тип:</font></td>";
  echo "<td bgcolor=5685a2>
   <font name=tahoma size=2 color=white># $f[no]</font></td>";
  echo "<tr><td colspan=3 bgcolor=797d8c>
    <font name=tahoma size=3 color=white>$f[txt]
     <p><a href=mailto:manager@localhost>Написать письмо менеджеру</a>
        </td></tr></table>";
}
?>
```




Файл был скачан с
http://all-ebooks.com

*Рис. 12.7. Выведено 5 записей*



http://all-ebooks.com

Как видите, в коде самого сценария нет ничего сложного. Возможно, вы скажете, что я забыл ссылку «Назад»? Так оно и есть. Но добавить ее несложно:

```
// в этом месте сценария мы уже увеличили $page++, поэтому номером
// предыдущей страницы будет $page-2
$pagePrev = $page-2;
if ($pagePrev <0) $pagePrev=0;
echo "<p><a href=limit.php?page=$pagePrev >Назад</a>";
```

# ГЛАВА 13.

# ГРАФИЧЕСКИЕ ФУНКЦИИ PHP

http://all-ebooks.com

## 13.1. ОСНОВНЫЕ ПОНЯТИЯ

### 13.1.1. Библиотека GD

При написании сценариев на PHP очень часто возникает необходимость создавать «на лету» различные рисунки: графики, счетчики или просто картинки, зависящие от каких-то данных.

Для того чтобы рядовой веб-программист почувствовал себя еще и художником, нужна библиотека GD. Если вы используете Linux, то, скорее всего, эта библиотека у вас уже установлена и PHP собран с ее поддержкой. Если же это не так, установите библиотеку GD и пересоберите PHP с поддержкой этой библиотеки. Как это сделать, описано в документации по PHP.

Если вы используете PHP для Windows, вам понадобится всего одна библиотека — `php_gd.dll`. Ее нужно скопировать в каталог, указанный в директиве `extension_dir` файла конфигурации `php.ini`. Затем добавьте в этот же файл следующую строку:

```
extension=php_gd.dll
```

Скорее всего, эта строка уже присутствует в разделе Dynamic Extensions, вам нужно лишь убрать комментарий.

**Примечание.**

Если вы скачали уже настроенную версию PHP5 с моего сайта http://www.dkws.org.ua, то вам ничего не нужно делать - все уже включено.

http://all-ebooks.com

### 13.1.2. Порядок создания графического оформления сайтов в PHP

Как и в любом другом языке программирования, который позволяет работать с графикой, совсем необязательно формировать картинку полностью. Возьмем простейший пример: вам нужно написать счетчик, содержащий логотип вашей компании. Не рисовать же замысловатую кривую, изображающую логотип, самостоятельно? Проще взять готовое изображение, поместить его слева, а справа вывести количество посещений сайта.

PHP позволяет это сделать: вы можете подготовить «прототип» будущей картинки в любимом графическом редакторе: The GIMP, Adobe Photoshop, да хоть в Paint. Потом ваш сценарий загрузит прототип как основу картинки, а дальше вы можете делать с ним все, что вам захочется: рисовать что-нибудь поверх прототипа, перемещать его фрагменты, добавлять надписи и т. д. В этой главе вы узнаете почти обо всех функциях графической библиотеки PHP.

## 13.2. ВЫВОД ГОТОВОГО ИЗОБРАЖЕНИЯ

После установки библиотеки проверим корректность ее работы. Создайте небольшой сценарий (листинг 13.1).

**Листинг 13.1. Сценарий test.php**

```
<?
$image = ImageCreateFromPng("image.png");
Header("Content-type: image/png");
ImagePng($image);
ImageDestroy($image);
?>
```

Использовать данный сценарий очень просто — в нужном месте вашего HTML-документа добавьте тег:

```
<img src=test.php>
```

Перед запуском сценария поместите в каталог, в котором расположен сценарий, файл `image.png`. Сценарий `test.php` не будет изменять наш рисунок `image.png`, он просто скопирует его в браузер. Вместо использования данного сценария мы могли бы просто использовать тег `<img src=image.png>`.

Сценарий демонстрирует несколько важных моментов при работе с рисунками: создание, вывод формата и сохранение рисунка.



http://all-ebooks.com

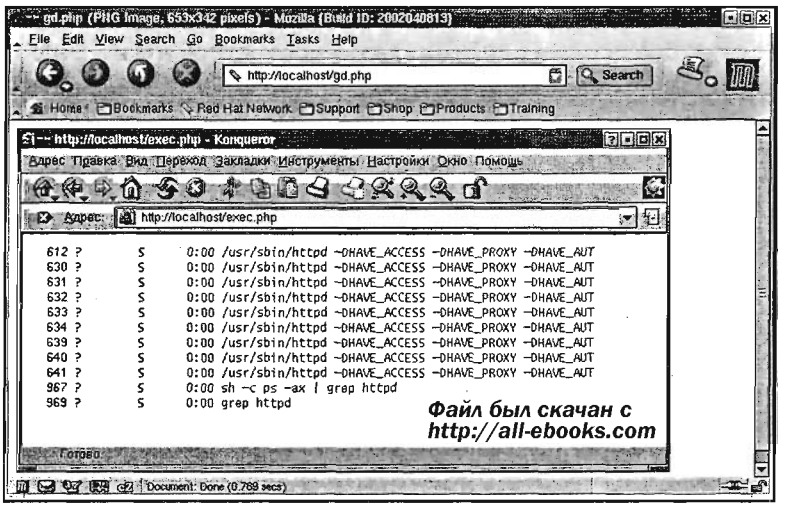



***Рис. 13.1. Вывод рисунка в браузер***

В первой строке мы загружаем в переменную `$image` содержимое файла `image.png`. Загруженное изображение распаковывается и в дальнейшем хранится в собственном растровом формате, подобном формату BMP, что обеспечивает высокую скорость выполнения операций над ним.

Неупакованный формат требует намного больше памяти, чем сжатый, поэтому перед пересылкой изображения по Интернету его требуется заново упаковать. Это мы и делаем в третьей строке, вызвав функцию `ImagePng()`, которая отправляет изображение в формате PNG на стандартный вывод, то есть в браузер. Предварительно (вторая строка) мы сообщили браузеру тип данных, которые собираемся выводить в его окно: рисунок формата PNG (image/png).

Наконец, функция `ImageDestroy()` освобождает память, выделенную для хранения рисунка.

В качестве прототипа вашего рисунка вы могли бы использовать файлы других графических форматов, например JPEG или GIF. Загрузить изображение в этих форматах позволяют функции `ImageCreateFromJpeg()` и `ImageCreateFromGif()` соответственно.

Итак, для вывода готовых изображений, хранимых в файлах графического формата, в PHP предусмотрены следующие функции:

- `ImageCreateFrom*()` — читает в переменную изображение из файла. В дальнейшем это изображение можно модифицировать средствами PHP независимо от формата исходного файла.

http://all-ebooks.com

- `ImagePng()`, `ImageGIF()` и `ImageJpeg()` — выводят переменную-изображение в окно браузера в соответствующем формате.
- `ImageDestroy()` — освобождает память. Если вы после вывода рисунка не планируете завершить работу сценария, обязательно освободите память, так как рисунок находится в памяти не в сжатом формате (PNG, JPEG, GIF), а в растровом, поэтому может занимать очень много памяти.

Если вы хотите нарисовать картинку с нуля, не используя заготовки, начните работу с вызова функции `ImageCreate(int $x, int $y)`. Она создаст пустой рисунок размером X × Y пикселей и вернет его дескриптор. Дальнейшая работа с рисунком производится через этот дескриптор.

### 13.2.1. Графические форматы

Следует отметить, что ваша библиотека может не поддерживать работу со всеми тремя форматами — PNG, JPEG и GIF. Поддержка формата PNG будет всегда, а вот поддержки формата GIF может не оказаться.

После того как рисунок уже сформирован, его нужно отправить на стандартный вывод — в браузер. Перед сохранением (выводом) нужно указать, и желательно правильно, формат рисунка. Это можно сделать с помощью функции `Header()`:

```
Header("Content-type: image/png"); // если рисунок в формате PNG
Header("Content-type: image/gif"); // если рисунок в формате GIF
Header("Content-type: image/jpeg");// если рисунок в формате JPEG
```

Почему я написал «желательно правильно»? Потому что некоторые браузеры сами определяют тип картинки. Например, вы отправляете заголовок, в котором сказано, что картинка в формате PNG, а сами выдаете браузеру картинку в формате GIF. Браузер по первым байтам самостоятельно определит тип рисунка и правильно отобразит его. Однако не нужно надеяться, что все браузеры такие умные: многие просто надеются на вашу честность и верят содержимому заголовка.

Напоминаю, что изображение, загруженное в память, хранится в растровом виде, а информация о его первоначальном формате теряется: вы можете вывести изображение в браузер в любом другом формате. Пример преобразования рисунка из формата PNG в формат JPEG приведен в листинге 13.2.



http://all-ebooks.com

**Листинг 13.2. Конвертация графических форматов**

```
<?
$image = ImageCreateFromPng("image.png");
Header("Content-type: image/jpeg");
ImageJpeg($image);
ImageDestroy($image);
?>
```

### 13.2.2. Определение размера картинки

После загрузки изображения полезно было бы знать его размеры (ширину и высоту), чтобы при выводе текста или графических примитивов не промахнуться, пытаясь вывести надпись длиной в 120 пикселей на рисунок шириной всего 100 пикселей.

При создании пустой картинки размер изображения нам уже известен, а вот при загрузке рисунка его точный размер нам неизвестен. Для определения горизонтального размера служит функция `ImageSX(resource $image)`, возвращающая ширину изображения `$image` в пикселях. Аналогично вертикальный размер рисунка вернет функция `ImageSY(resource $image)`.

## 13.3. Графические примитивы

Вам уже надоело рассматривать различные форматы и хочется что-нибудь нарисовать? Не спешите! Перед тем как нарисовать какой-нибудь примитив (линию, эллипс или дугу), нам нужно выбрать цвет и создать его. Создать цвет – значит сопоставить дескриптор тройке RGB-значений.

Тройка RGB (Red, Green, Blue) — это три числа в диапазоне от 0 до 255, которые определяют содержание красной, зеленой и синей составляющих в создаваемом цвете. Число 0 означает минимальную яркость, а 255 — максимальную. Например, тройка (0, 0, 0) задает черный цвет, а (255, 255, 255) — белый.

### 13.3.1. Создание (определение) цвета

#### Задание нового цвета в палитре рисунка

Средства библиотеки GD не позволяют задавать цвет привычным для веб-дизайнера способом, указав шестнадцатеричный код или стандартное

HTML-имя цвета («chocolate», «ivory», «peachpuff» и т. п.). Вам придётся создать новый цвет в палитре рисунка или выбрать уже существующий цвет из палитры. Работать с цветом можно только через его дескриптор, аналогично тому, как мы работаем с рисунком.

Палитра — это набор цветов, которые использованы в рисунке. Некоторые форматы, например GIF, поддерживают лишь небольшое количество цветов в рисунке (а именно 256).

Чем меньше цветов присутствует на рисунке, тем меньше рисунок занимает памяти. При небольшом размере палитры рисунок очень хорошо сжимается. Эффективность сжатия снижается при переходе через степень двойки: если ваш рисунок хорошо сжимался при 32 цветах, то добавление еще одного цвета заметно увеличит размер файла рисунка.

Создать новый цвет в палитре позволяет функция `ImageColorAllocate (resource $image, int $red, int $green, int $blue)`. Она добавляет в палитру новый цвет и возвращает его дескриптор. Таким образом, каждый вызов функции `ImageColorAllocate()` увеличивает число цветов в палитре на единицу.

Определять все нужные цвета я рекомендую в самом начале сценария. При этом желательно, чтобы идентификаторы цветов несли смысловую нагрузку, например `$white`, `$black`, а не `$c0`, `$c1`.

#### Подбор ближайшего цвета из палитры

Может получиться так, что с новым цветом палитра будет переполнена. Например, если вы работаете с изображением в формате GIF, то у него может быть всего 256 цветов. Что же произойдет, если достигнут предел — 256 цветов? Тогда функция `ImageColorAllocate()` попытается найти в палитре цвет, который, по ее мнению, ближе всех к запрашиваемому цвету.

Таким образом, функция `ImageColorAllocate()` действует примерно следующим образом: если есть свободные места в палитре, то создается новый цвет и добавляется в палитру, а если свободных мест нет, то производится поиск похожего цвета.

Однако в PHP предусмотрена специальная функция, которая изначально не задает новые цвета в палитре, а сразу ищет цвет, похожий на указанный. Это функция `ImageColorClosest(resource $image, int $red, int $green, int $blue)`, которая возвращает похожий цвет из уже имеющихся цветов в палитре. Эту функцию рекомендуется использовать, если вы не хотите, чтобы палитра рисунка разрасталась. При этом функцию `ImageColorAllocate()` следует использовать лишь в исключительных



http://all-ebooks.com

случаях, когда цвет нужно задать абсолютно точно либо в палитре нет оттенка, похожего на нужный.

Узнать текущее количество цветов в палитре позволяет функция `ImageColorsTotal(resource $image)`, возвращающая целое число.

#### Эффект прозрачности

Для некоторых частей рисунка бывает необходимо задать «прозрачный» цвет, то есть указать браузеру, что вместо прозрачного фрагмента следует отобразить фон документа HTML. Функция `ImageColorTransparent (resource $image, int $color)` указывает, что цвет, заданный дескриптором `$color`, в палитре изображения `$image` будет считаться прозрачным.

Пусть, например, наше изображение должно располагаться на красном фоне, и нам нужно сделать красные фрагменты рисунка прозрачными:

```
$red = ImageColorClosest($image,255,0,0);
// Мы используем функцию ImageColorClosest, потому
// что красный цвет уже есть в палитре
ImageColorTransparent($image,$red);
```

Помните, что не все форматы поддерживают прозрачность, например, JPEG прозрачность не поддерживает.

Вот теперь мы можем приступить к рисованию графических примитивов.

### 13.3.2. Рисование графических примитивов

#### Рисование точки

Простейшим примитивом является точка. Нарисовать ее можно с помощью функции `ImageSetPixel(resource $image, int $x, $y, int $color)`. Эта функция нарисует одну точку цвета `$color` с координатами (`$x, $y`). Координаты отсчитываются в пикселях от верхнего левого угла изображения `$image`.

Если вам нужно узнать цвет какой-нибудь точки изображения, используйте функцию `ImageColorAt(resource $image, int $x, int $y)`. Действие этой функции аналогично действию инструмента «Пипетка» во многих графических редакторах.

http://all-ebooks.com

### Рисование линий

Нарисовать линию можно функциями:

```
ImageLine (resource $image, int $x1, int $y1, int $x2, int $y2,
int $color)
```

и

```
ImageDashedLine (resource $image, int $x1, int $y1, int $x2,
int $y2, int $color).
```

Первая функция проводит между точками ($x1, $y1) и ($x2, $y2) цветом $color сплошную линию, а вторая — пунктирную.

### Рисование прямоугольников

Нарисовать цветной прямоугольник можно, вызвав функцию:

```
ImageFilledRectangle(resource $image, int $x1, int $y1, int $x2,
int $y2, int $color).
```

Эта функция нарисует в изображении $image закрашенный прямоугольник. Размеры прямоугольника задаются параметрами ($x1, $y1) и ($x2, $y2), указывающими координаты верхнего левого и нижнего правого углов соответственно. Цвет закраски задается дескриптором $color.

Для рисования незакрашенных прямоугольников в PHP служит функция

```
ImageRectangle(resource Simage, int $x1, int $y1, int $x2, int $y2,
int $color).
```

Она рисует прямоугольную границу цвета $color и толщины в 1 пиксель.

### Рисование многоугольников

С помощью функции

```
ImagePolygon(int $image, list $dots, int $num, int $color)
```

На изображении $image можно нарисовать многоугольник, заданный вершинами в количестве $num. Координаты вершин передаются в массиве $dots, заполняемом следующим образом:

```
$dots[0]=x0; $dots[1]=y0; $dots[2]=x1; $dots[3]=y1; и т. д.
```

Если вам нужно нарисовать закрашенный многоугольник, используйте функцию с теми же параметрами

```
ImageFilledPolygon (resource $image, list $dots, int $num, int
$color)
```



http://all-ebooks.com

### Рисование эллипсов и дуг

Для изображения дуги сектора используйте функцию

```
ImageArc(resource $image, int $x, int $y, int $width, int $height,
int $start_angle, int $end_angle, int $col)
```

Эта функция рисует дугу сектора эллипса от угла `$start_angle` до угла `$end_angle`. Углы отсчитываются в градусах от горизонтали против часовой стрелки. Сам эллипс должен вписаться в прямоугольник, заданный параметрами:

- `$x, $y` — координаты верхнего левого угла;
- `$width` — ширина;
- `$height` — высота.

Фигура не закрашивается, а цветом `$color` обводится ее контур. Понятно, что, чтобы нарисовать эллипс, а не дугу, следует в качестве значений `$start_angle` и `$end_angle` указать 0 и 360.

### Закрашивание области

Залить одним цветом произвольную область, начиная с точки (`$x, $y`), можно функцией

```
ImageFill(resource $image, int $x, int $y, int $color).
```

Эта функция закрасит все прилегающие пиксели, цвет которых совпадает с цветом начальной точки, цветом `$color`.

Если вам нужно закрасить, наоборот, все пиксели, цвет которых не совпадает с граничным, воспользуйтесь функцией

```
ImageFillToBorder(resource $image, int $x, $y, int $border, int
$color).
```

Она зальет цветом `$color` все точки, начиная с (`$x, $y`), пока не будет достигнута граница – цвет `$border`.

В качестве примера рассмотрим сценарий (листинг 13.3), который я использовал для создания рис. 13.2.

**Листинг 13.3. Рисование концентрических квадратов**

```
<?
// Создаем рисунок 200x200
$im = ImageCreate(200,200);
// первый цвет в палитре - это цвет фона
```

http://all-ebooks.com

```
// RGB (0,0,0) - это черный цвет
$black = imagecolorallocate($im,0,0,0);
// Создаем массив $c[] и заполняем его цветами
$c[] = imagecolorallocate($im,255,0,0);       // красный
$c[] = imagecolorallocate($im,0,255,0);       // зеленый
$c[] = imagecolorallocate($im,0,0,255);       // синий
$c[] = imagecolorallocate($im,255,0,255);     // ярко-фиолетовый
$c[]  =  imagecolorallocate($im,100,0,100);         //  фиолетовый
«вполнакала»
$c1 = 1; // индекс массива $c[]

// Рисуем концентрические квадраты
for ($i = 1; $i < 7; $i++)
{
$c1 = $i;
if ($c1>5) $c1 = 1; // если цвет > 5, то цвет = красный (1) ImageR
ectangle($im,10*$i,10*$i,200-10*$i,200-10*$i,$c[$c1-1]);
}

// Сообщаем браузеру формат картинки
Header("Content-type: image/png");
// Выводим изображение в формате PNG
ImagePng($im);
// Освобождаем память
ImageDestroy($im);

?>
```

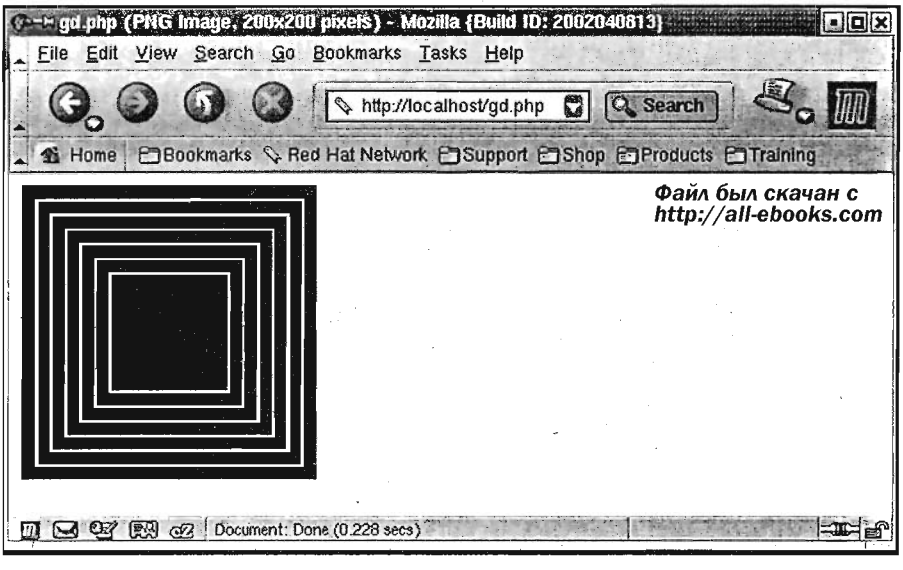



*Рис. 13.2. Концентрические квадраты*



http://all-ebooks.com

## 13.4. ДОБАВЛЕНИЕ НАДПИСЕЙ К ИЗОБРАЖЕНИЮ

Зачастую, например, при написании сценария-счетчика, возникает необходимость добавить к изображению строку текста. В PHP это можно сделать, используя функцию

```
ImageString(resource $image, int $font, int $x, int $y, string $s,
int $color).
```

Эта функция выведет строку `$s` шрифтом `$font` и цветом `$color` в горизонтальном направлении так, что левый верхний угол надписи попадет в точку (`$x`, `$y`). У стандартного встроенного шрифта можно менять только размер: от 1 (*наименьший*) до 5 (*наибольший*).

Вместо стандартного шрифта вы можете загрузить свой с помощью функции `ImageLoadFont(string $file)`, где `$file` — имя файла со шрифтом. При этом номер шрифта будет больше 5.

Файлы, в которых содержатся шрифты, — бинарные и имеют следующую структуру, представленную в табл. 13.1.

**Таблица 13.1.** *Формат файла шрифтов*

| Смещение от начала файла (в байтах) | Тип | Описание |
|---|---|---|
| 0...3 | long | Количество символов в шрифте |
| 4...7 | long | Индекс первого символа шрифта (обычно это пробел, 32) |
| 8...11 | long | Ширина каждого знака в пикселях – W |
| 12...15 | long | Высота каждого знака в пикселях – H |
| 16...EOF | array | Массив, содержащий информацию о каждом символе, по одному байту на пиксель. На один символ приходится WxH байтов |

Для вывода надписи (рис. 13.3) я использовал сценарий из листинга 13.4.

**Листинг 13.4. Вывод надписи поверх изображения**

```
<?
$im = ImageCreateFromPng("./screenshot.png");
$color = imagecolorallocate($im,255,0,0); // цвет шрифта - красный

// Наша строка - IP-адрес клиента + имя сервера
$str = "$REMOTE_ADDR $SERVER_NAME";

// Выводим строку $str в картинку $im, шрифтом 5, цветом $color,
// координаты начала строки (70,200)
```

http://all-ebooks.com

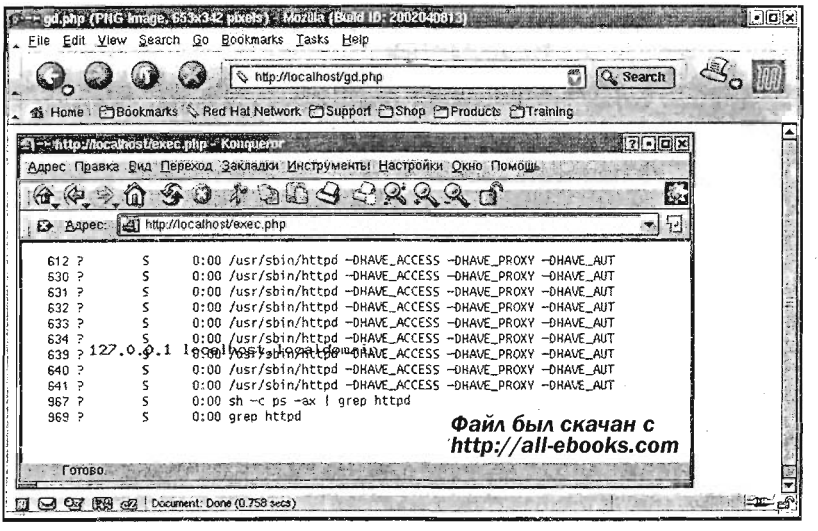



***Рис. 13.3. Вывод надписи***

```
imagestring($im,5,70,200,$str,$color);

Header("Content-type: image/png");
ImagePng($im);
ImageDestroy($im);
?>
```

Вывести строку в вертикальном направлении позволяет функция `ImageStringUp()`, принимающая тот же набор параметров, что и функция `ImageString()`.

## 13.5. РАБОТА С TTF-ШРИФТАМИ

Функции `ImageString()` и `ImageStringUp()` обладают существенным недостатком: они не позволяют вывести русский текст. Это объясняется тем, что данные функции используют фиксированные встроенные шрифты PHP. Можно, конечно, найти русский фиксированный шрифт, но проще использовать самый обыкновенный TTF (TrueType) шрифт. Во-первых, при этом у нас не будет проблем с русскими буквами, а во-вторых, TTF-шрифты очень распространены. Вы можете выбрать любой шрифт, который сделает вашу надпись привлекательнее. К тому же TTF-шрифты масштабируемы, то есть надпись можно вывести любым размером.

Для вывода строки с использованием TTF-шрифта служит функция:



http://all-ebooks.com

```
ImageTTFtext(resource $image, int $size, $int angle, int $x, $int
$y, int $color, string $font, string $message).
```

Аргументы этой функции таковы:

- `$image` — идентификатор картинки, в которую выводится текст;
- `$size` — размер шрифта, который будет использоваться при выводе строки;
- `$angle` — угол наклона выводимой строки;
- `$x, $y` — координаты верхнего левого угла выводимой строки;
- `$color` — идентификатор цвета;
- `$font` — полное имя файла шрифта;
- `$message` — выводимая строка.

Практически все параметры не нуждаются в объяснении, кроме параметров `$angle, $font` и `$message`. Угол `$angle` задается в градусах и отсчитывается от горизонтали против часовой стрелки. Имя файла шрифта должно быть полным, то есть нужно указывать полный путь к нему, например `/var/www/html/ttf/arial.ttf`. Если шрифт находится в том же каталоге, что и сценарий, обычно удобно использовать функцию `getcwd()`, возвращающую текущий каталог.

Сообщение `$message` должно быть в кодировке Windows 1251, потому что большинство TTF-шрифтов используют именно эту кодировку.

В листинге 13.5 приведен пример вывода строки на кириллице.

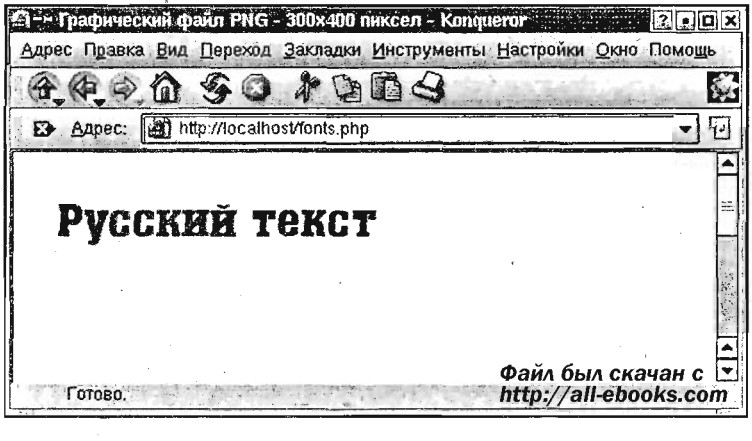



*Рис. 13.4. Вывод текста с использованием TTF-шрифта*

**Листинг 13.5. Вывод кириллицы**

```
<?
// Создаем картинку 300x400
$im = imagecreate(300,400);
// Определяем   белый   цвет
$white = imagecolorallocate($im, 255,255,255);
// Определяем   красный   цвет
$red = imagecolorallocate ($im, 255,0,0);

// Это наша строка
$str = "Русский   текст";
// Преобразовываем строку из кодировки KOI8-R в CP1251
// Вам не нужно этого делать, если вы работаете под Windows
$str = convert_cyr_string($str,"k","w");


// Выводим строку с использованием шрифта assuan.ttf,
// размер - 24 пункта,
// цвет - красный, координаты нижнего левого
// угла строки - (20,50)
imagettftext($im, 24,0,20,50,$red,getcwd()."/assuan.ttf",$str);

Header("Content-type: image/png");
// Выводим картинку в формате PNG
ImagePng($im);
?>
```

Помимо того, что функция `ImageTTFtext()` выводит текст, она в качестве своего значения возвращает массив-список, содержащий координаты прямоугольника, описанного вокруг строки (табл. 13.2).

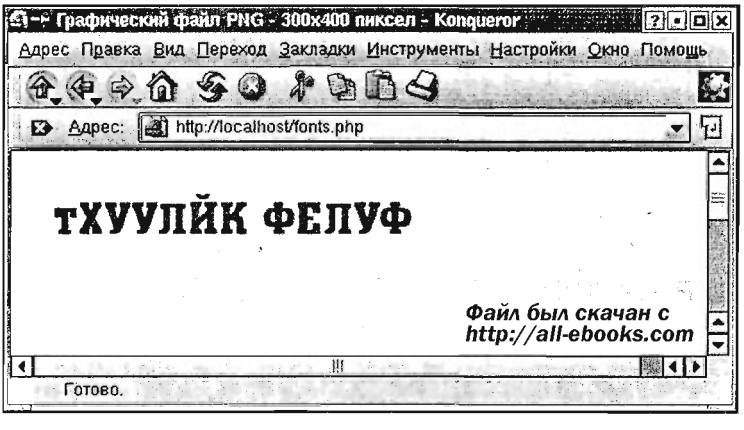



*Рис. 13.5. Вот что бывает, если программист не преобразовал кодировку*



http://all-ebooks.com

**Таблица 13.2.** *Результат функции imageTTFtext()*

| Индексы массива | Содержание |
|---|---|
| 0,1 | Координаты (x, y) левого верхнего угла |
| 2,3 | Координаты (x, y) правого верхнего угла |
| 4,5 | Координаты (x, y) правого нижнего угла |
| 6,7 | Координаты (x, y) левого нижнего угла |

Если вы часто работаете с TTF-шрифтами, вы должны оценить функцию `ImageTTFBox(int $size, int $angle, string $font, string $message)`, которая позволяет вычислить размер строки без вывода ее на изображение.

Функции нужно передать четыре параметра: размер шрифта, угол наклона строки, полное имя файла шрифта и саму строку.

Как и предыдущая функция, `ImageTTFBox()` возвращает массив-список, содержащий координаты прямоугольника, описанного вокруг строки. Внимание! Порядок вершин прямоугольника в нем другой (табл. 13.3).

Не понимаю, из каких соображений исходили разработчики PHP: если бы совпадали форматы списков, возвращаемых функциями `ImageTTFtext()` и `ImageTTFbox()`, работать с ними было бы гораздо проще.

```
$Rect = imageTTFbox(24,0,"/var/www/html/ttf/times.ttf",$str);
foreach ($Rect as $v) echo $v;
```

**Таблица 13.3.** *Результат функции imageTTFbox()*

| Индексы массива | Содержание |
|---|---|
| 0, 1 | Координаты (x, y) левого нижнего угла |
| 2,3 | Координаты (x, y) правого нижнего угла |
| 4, 5 | Координаты (x, y) правого верхнего угла |
| 6,7 | Координаты (x, y) левого верхнего угла |

## 13.6. КОПИРОВАНИЕ ФРАГМЕНТОВ ИЗОБРАЖЕНИЯ

Нам осталось рассмотреть очень мощную функцию, которая умеет копировать и масштабировать участки изображения как в пределах одного рисунка, так и из одного рисунка в другой.

```
ImageCopyResized(resource $dest, resource $src, int $destX, int
$destY, int $srcX, int $srcY, int $destW, int $destH, int $srcW,
int $srcH)
```

Выглядит просто ужасающе. Однако все намного проще, когда разберешься, что здесь к чему. Параметры `$dest` и `$src` задают рисунок-результат и рисунок-источник. Эти параметры могут совпадать.

Параметры `$srcX`, `$srcY`, `$srcW`, `$srcH` определяют область исходного изображения, над которой будет произведена операция: координаты ее левого верхнего угла, ширину и высоту.

Параметры `$destX`, `$destY`, `$destW`, `$destH` задают область рисунка-результата, в которую будет вписан прямоугольник, заданный предыдущими параметрами. Если ширина и высота прямоугольников не совпадают, то копируемая область будет сжата или растянута в соответствии с параметрами.

В PHP есть еще одна функция, позволяющая копировать части изображения. Это функция `ImageCopy()`:

```
imagecopy (resource $dest, resource $src, int $dest_x, int $dest_y,
int $src_x, int $src_y, int $src_w, int $src_h)
```

Функция `ImageCopy()` гораздо проще в использовании, чем предыдущая, но и менее универсальна. `ImageCopy()` копирует фрагмент изображения `$src` в изображение `$dest`. Копируемая область – это прямоугольник с левым верхним углом в точке (`$src_x`, `$src_y`), шириной `$src_w` и высотой `$src_h`. Скопированное изображение вставляется в изображение `$dest`, левым верхним углом в точку `$dest_x` и `$dest_y`.

Следует заметить, что в этой главе рассмотрены далеко не все графические функции, но и их вполне достаточно для типичных графических задач веб-программирования, сводящихся к построению графиков или выводу информации о посещениях узла. Для создания более сложной графики разработано множество специальных пакетов – GIMP, Photoshop и т. п.



http://all-ebooks.com

## 13.7. СЦЕНАРИЙ ДИНАМИЧЕСКОГО ИЗМЕНЕНИЯ РАЗМЕРОВ ИЗОБРАЖЕНИЙ

Предположим, что у нас есть довольно большая картинка. Вывод всей картинки не очень желателен, поскольку:

- Это не вписывается в дизайн сайта.
- Зачем пользователю загружать лишние объемы информации?

Гораздо эффективнее поступить так: сделать уменьшенную копию этого изображения, выложить ее на сервер вместе с оригинальным файлом и разместить на основной странице уменьшенное изображение. Если оно заинтересует пользователя, он щелкнет по нему и перейдет по ссылке к оригинальному изображению.

А теперь представьте, что на вашем сайте хранится галерея изображений, которую пополняют каждый день сотни пользователей. Неужели каждый файл придется обрабатывать вручную? Согласитесь, не очень хорошо.

Я предлагаю вам следующее решение: написать на PHP сценарий, который динамически в зависимости от переданных ему параметров будет выводить либо уменьшенное до определенного размера изображение, либо, наоборот, увеличенное. Если оригинальный рисунок требуется отображать на главной странице в уменьшенном виде, то поместите на страницу код сценария; если в увеличенном, то выложите на страницу сам файл рисунка, а ссылку с него поставьте на сценарий.

Сценарий `resize.php` (листинг 13.6) предназначен для вызова следующим образом:

```
http://server/resize.php?fn=<имя_файла_картинки.jpg>&type=<тип_
масштабирования>
```

Например, чтобы вывести на страницу картинку `bigimage.jpg` в уменьшенном виде, вызовите

```
http://server/resize.php?fn=bigimage.jpg&type=1
```

Наоборот, для увеличения картинки `smallimage.jpg` вставьте в тег `<img>` следующий вызов:

```
<img src=http://server/resize.php?fn=smallimage.jpg&type=2>
```

Теперь рассмотрим сам сценарий. Подробно описывать принцип его работы я не буду, поскольку соответствующие графические функции уже описаны

ранее. Все параметры и переменные прокомментированы, думаю, вы разберетесь (листинг 13.6):

**Листинг 13.6. Динамическое масштабирование изображений**

```
<?
// fn - имя файла
// type - способ масштабирования (1 уменьшить, 2 увеличить)
// q - качество сжатия JPEG
// src - исходное изображение
// dest - результирующее изображение
// w - ширина изображения
// ratio - коэффициент пропорциональности

if ($type == 1) $w = 120;    // уменьшить
if ($type == 2) $w = 218;  // увеличить

if (!isset($q)) $q = 90;  // качество jpeg по умолчанию

// создаём исходное изображение на основе
// исходного файла и получаем его размеры
$src = imagecreatefromjpeg($fn);
$w_src = imagesx($src);
$h_src = imagesy($src);

header("Content-type: image/jpeg");

// если размер исходного изображения
// отличается от требуемого размера
if ($w_src != $w)
{
       // вычисляем пропорции
       $ratio = $w_src/$w;
       $w_dest = round($w_src/$ratio);
       $h_dest = round($h_src/$ratio);

       // создаем результирующую картинку
       $dest = imagecreatetruecolor($w_dest,$h_dest);

       // строка, которая будет выведена поверх картинки
       $label = "www.dkws.org.ua";
       imagecopyresized($dest, $src, 0, 0, 0, 0, $w_dest, $h_dest,
$w_src, $h_src);

        // определяем координаты, по которым должен быть выведен
        //текст
        $size = 2; // размер шрифта выводимого текста
```



http://all-ebooks.com

```
        $x_text = $w_dest-imagefontwidth($size)*strlen($str)-3;
        $y_text = $h_dest-imagefontheight($size)-3;

        // каким цветом вывести текст, чтобы он не слился с фоном
        // картинки?
        // добавим элемент искусственного интеллекта:
        // пусть цвет определит сам сценарий.
        $white = imagecolorallocate($dest, 255, 255, 255);
        $black = imagecolorallocate($dest, 0, 0, 0);
        $gray = imagecolorallocate($dest, 127, 127, 127);

    if (imagecolorat($dest,$x_text,$y_text)>$gray) $color = $black;
    if (imagecolorat($dest,$x_text,$y_text)<$gray) $color = $white;

    // выводим текст
    imagestring($dest, $size, $x_text-1, $y_text-1, $label,$white-$color);
    imagestring($dest, $size, $x_text+1, $y_text+1, $label,$white-$color);
    imagestring($dest, $size, $x_text+1, $y_text-1, $label,$white-$color);
    imagestring($dest, $size, $x_text-1, $y_text+1, $label,$white-$color);
    imagestring($dest, $size, $x_text-1, $y_text,   $label,$white-$color);
    imagestring($dest, $size, $x_text+1, $y_text,   $label,$white-$color);
    imagestring($dest, $size, $x_text,   $y_text-1, $label,$white-$color);
    imagestring($dest, $size, $x_text,   $y_text+1, $label,$white-$color);

    imagestring($dest, $size, $x_text,   $y_text,   $label,$color);

        imagejpeg($dest,'',$q);      // выводим картинку в браузер
        imagedestroy($dest);         // очищаем память
        imagedestroy($src);
}
```

Если хотите увидеть сценарий в работе, зайдите на `www.nit.com.ru` – на этом сайте он используется для вывода обложек книг. Конечно, на универсальность данный сценарий не претендует, но вы всегда сможете изменить его, «заточив» под ваши нужды.

http://all-ebooks.com

# ГЛАВА 14.

# СЕТЕВЫЕ ФУНКЦИИ PHP

http://all-ebooks.com

## 14.1. СОКЕТЫ

### 14.1.1. Понятие сокета. Функции для работы с сокетами

#### Открытие и использование сокетов

Сокет во многом похож на дескриптор файла (то есть с сетевым соединением можно работать как с обычным файлом), но служит для несколько иных целей, а именно для передачи информации по сети.

Помните, функция `fopen()` позволяла открывать удаленные файлы, которые находились на веб-серверах? Но для многих задач этого явно недостаточно. Получить файл по протоколу HTTP мы можем, но как реализовать отправку электронных сообщений по протоколу SMTP или любую другую сетевую функцию на каком-либо другом, отличном от HTTP, протоколе? Конечно, в случае отправки письма можно вызвать программу sendmail (или mail) и передать ей письмо, чтобы она, в свою очередь, его отправила. Но это не очень удобно, потому что:

- Эта программа должна быть установлена на этом компьютере (соответственно компьютер должен работать под управлением Linux).
- Мы должны знать точный путь к программе.

А что делать, если программа не установлена или мы используем Windows? Как отправить сообщение? PHP позволяет использовать более гибкий инструмент — сокеты, позволяющие установить соединение по любому прото-



http://all-ebooks.com

колу. Это значит, что мы можем «общаться» с любой сетевой программой (нам только нужно знать протокол обмена) и реализовать любую сетевую задачу.

Пример использования SMTP я выбрал не случайно — в этой главе мы научимся отправлять письма с помощью сокетов. Потом, в следующей главе, мы будем использовать функцию `mail()` для этой цели— она проще, но иногда она работает некорректно. Поэтому все равно рекомендуется использовать сокеты.

Для открытия сокета служит функция

```
fsockopen (string $host [, int $port [, int &$errno [, string
&$errmsg [, float $timeout]]]] )
```

Первый параметр функции — это имя узла, к которому мы хотим подключиться, или его IP-адрес. Второй параметр — это порт службы. В нашем случае с отправкой письма по протоколу SMTP это будет порт 25. При возникновении ошибки в переменные `$errno` и `$errmsg`, если они указаны (что необязательно), будут записаны соответственно код ошибки и текст сообщения об ошибке.

После открытия сокета мы можем работать с ним как с обыкновенным файлом, то есть использовать функции `fwrite()`, `fread()`, `fgets()`, `fputs()` и другие. В отличие от канала сокет открывается одновременно на чтение и на запись.

Работу с протоколом SMTP через сокет мы рассмотрим в следующем пункте, а сейчас приведем пример, демонстрирующий работу через сокет с протоколом HTTP (листинг 14.1).

**Листинг 14.1. Работа через сокет с протоколом HTTP**

```
<?
// Открываем   сокет
$s = fsockopen("localhost",80);

// Отправляем команду GET - получить index.html (/)
fputs($s, "GET / HTTP/1.0\n\n");

// Читаем ответ
while (!feof($s)) echo fgets($s,1000);

// Закрываем сокет
fclose($s);
?>
```

http://all-ebooks.com

#### Блокирующий/неблокирующий режимы

Функция `socket_set_blocking(int $socket_descriptor, int $mode)` является псевдонимом для функции `stream_set_blocking (resource $stream, int $mode)`. Если параметр `$mode` принимает значение `false`, поток переводится в неблокирующий режим, если `true` — выполняется блокировка потока.

От режима блокирования зависит работа функций `fread()` и `fgets()`. В блокирующем режиме функция `fgets()` будет ждать до тех пор, пока в потоке не появятся данные, доступные для чтения, и она не сможет их прочитать. В неблокирующем режиме функция `fgets()` завершает работу с ошибкой, если данных прочитать не удалось.

> **Примечание.**
>
> Функция `socket_set_blocking()`, как и `stream_set_blocking()`, появилась в версии PHP 4.3.0. Если у вас более ранняя версия PHP, вы не сможете использовать режим блокировки потока.

### 14.1.2. Отправка электронных писем с помощью сокетов

Чтобы отправить письмо, мы должны знать, как сделать это вручную. Нет, не нажать кнопку «Новое сообщение», а затем кнопку «Отправить». Мы должны знать сам протокол SMTP, то есть понимать, что именно почтовая программа отправляет серверу и какие данные от него получает. Если вы не хотите вникать в некоторые детали SMTP, то пропустите этот раздел. Об отправке писем с помощью соответствующих функций PHP мы будем говорить в следующей главе.

Итак, если вы все-таки решились — запустите программу telnet:

```
telnet SMTP-узел  25
```

Если у вас Linux и программа sendmail (или другой агент MTA) установлена, можно использовать локальный хост — так будет даже лучше. Введите:

```
telnet localhost  25
```

Вы должны увидеть приветствие программы MTA (Mail Transfer Agent — Агент передачи почты).

```
220 localhost.localdomain ESMTP Sendmail 8.11.6/8.11.6; Wed, 05 Apr
2005 23:24:56 +0300
```

Программа поприветствовала нас и ждет нашей команды. Поздороваемся с программой:

```
HELO localhost
```



http://all-ebooks.com

Обратите внимание, что слово «привет» пишется HELO, а не HELLO. Вместо localhost нужно указать ваш узел, если у вас установлен агент MTA.

На наше приветствие программа ответит:

```
250 localhost.localdomain Hello localhost.localdomain [127.0.0.1],
pleased to meet you
```

Данное сообщение означает, что нам разрешена отправка почты.

> **Примечание.**
>
> Что делать, если вы работаете в Windows и у вас не установлен MTA? Тогда запустите программу telnet (она есть в любой операционной системе) и подключитесь к серверу провайдера: telnet mail.isp.ru 25. В команде **helo** нужно ввести имя своей машины, узнать которое поможет программа ipconfig.

После взаимного приветствия нужно представиться:

```
MAIL FROM: den@localhost
```

В команде MAIL FROM нужно указать свой адрес электронной почты. Ответ программы выглядит так:

```
250 2.1.0 den@localhost... Sender ok
```

Программа подтверждает, что имя отправителя из числа допустимых. Потом нужно указать адрес реципиента:

```
RCPT TO: lena@localhost
```

Ответ программы:

```
250  2.1.5 lena... Recipient ok
```

Вот теперь можно послать команду DATA, что означает, что мы будем передавать письмо.

```
DATA
```

Программа сообщит нам, что в качестве признака завершения ввода она ожидает строки, содержащей единственный символ точки:

```
354 Enter mail, end with "." on a line by itself
```

Введите сообщение из одной или нескольких строк, завершив его отдельно стоящей точкой, и увидите сообщение о том, что наше письмо поставлено в очередь на отправку:

```
250 2.0.0 m3GAOups1311 Message accepted for delivery
```

Сейчас можно заново повторить команды MAIL FROM, RCPT TO, DATA для создания нового сообщения, но мы этого делать не будем, поэтому введите команду QUIT.

http://all-ebooks.com

Приведенный пример демонстрирует обмен между почтовой программой и сервером по протоколу SMTP. Конечно, пример очень тривиален, потому что мы не использовали ни заголовка, ни аутентификации, но сейчас речь не о них. Главное, чтобы вы уловили идею работы с сокетом: нужно отправить команду, прочитать результат, отправить следующую команду, опять прочитать результат и т. д.

Сценарий, приведенный в листинге 14.2, автоматизирует проделанную выше работу.

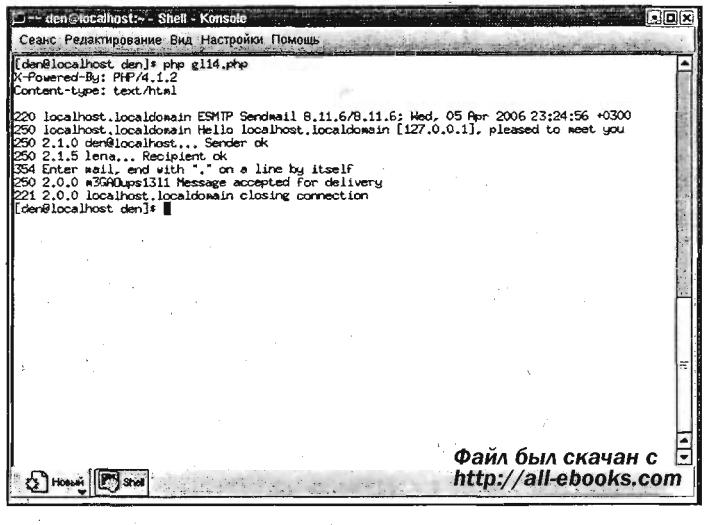
**Листинг 14.2. Работа через сокет с протоколом SMTP**

```
<?
$CRLF = "\r\n";
$r = 0;
$m = $line = " ";

//    Открываем   сокет

$s = fsockopen("localhost", "25", $r, $m);

//   Функция   чтения   ответа
function get_data()
{
  $return = " ";
  global $line, $CRLF, $s;
  while((strpos($return, $CRLF) === FALSE OR substr($line,3,1) !== ' '))

  {
    $line = fgets($s, 512);
    $return = $line;
    echo $line;
  }
}

// Функция отправки команды
function send_data($data)
{
  global $s, $CRLF;
  fwrite($s, $data.$CRLF, strlen($data)+2);
}
// Читаем приветствие
get_data();
// Здороваемся
send_data("HELO localhost");
// Читаем ответ
get_data();
```



http://all-ebooks.com

```
// Представляемся
send_data("MAIL FROM: den@localhost");
// Можно ли нам отправлять почту?
get_data();

// Указываем адрес реципиента
send_data("RCPT TO: lena");
// Правильный ли адрес?
get_data();

// Сейчас будем отправлять сообщение
send_data("DATA");
// Читаем ответ сервера
get_data();

// А это само сообщение, завершенное точкой на новой строке
send_data("Message",$CRLF.".".$CRLF);

// Поставлено ли сообщение в очередь?
get_data();

// Закрываем соединение
send_data("QUIT");
// Читаем ответ
get_data();

// закрываем сокет
fclose ($s);
?>
```




Файл был скачан с
http://all-ebooks.com

*Рис. 14.1. Отправка письма*

http://all-ebooks.com

Вывод программы вы можете увидеть на рис. 14.1.

Обратите внимание, что при работе с MTA символом конца строки считается не один символ \n, как в Unix, а два символа \r\n, как в Windows.

## 14.2. РАБОТА С DNS

### 14.2.1. DNS-функции

Для работы с системой доменных имен в PHP предназначены три функции:

- `gethostbyaddr(string $IP)` — возвращает имя узла по его IP-адресу, то есть выполняет преобразование IP-адреса в символьное имя узла.
- `gethostbyname(string $name)` — возвращает IP-адрес, соответствующий имени узла `$name`.
- `gethostbynamel(string $name)` — возвращает список (list) IP-адресов, которые ассоциируются с именем `$name`.

Зачем нужна третья из перечисленных функций? Одному доменному имени могут соответствовать несколько IP-адресов. Если нагрузка на сервер большая, то правильно настроенный сервер DNS будет каждый раз возвращать другой IP-адрес узла — IP-адрес зеркала. Выходит, что один клиент работает с сайтом `www.big_site.net`, подключившись по адресу 192.168.1.1, а второй работает с зеркалом этого сайта, подключившись по

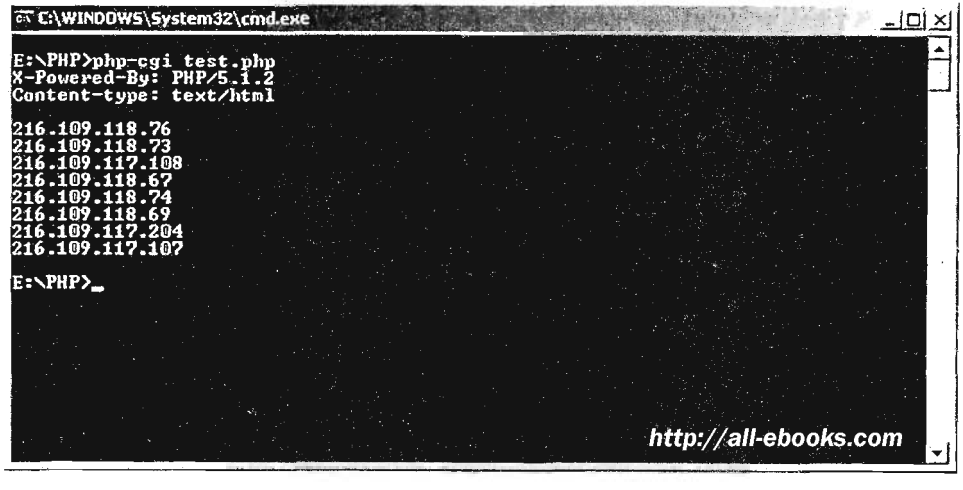



*Рис. 14.2. IP-адреса домена www.yahoo.com*



http://all-ebooks.com

адресу 192.168.1.2. Для клиентов это совершенно прозрачно, а нагрузка на серверы распределяется равномерно, вследствие чего повышается производительность работы сайта.

Функция `gethostbynamel()` возвращает все IP-адреса, связанные с данным доменным именем. Вот пример использования этой функции:

```
$ips = gethostbynamel("www.big_site.net");
foreach ($ips as $ip) echo "$ip <br>";
```

Применив этот фрагмент кода к имени `www.yahoo.com`, вы получите вывод, изображенный на рис. 14.2.

### 14.2.2. Корректное разрешение IP-адреса в доменное имя

Попробуем разрешить IP-адрес 111.111.111.333 (я специально использую несуществующий IP, чтобы не произошло недоразумений с владельцем этого IP) в доменное имя:

```
<?
echo gethostbyaddr("111.111.111.333");
?>
```

Функция `gethostbyaddr()` работает следующим образом: она обращается к узлу 111.111.111.333 и запрашивает у него доменное имя. Предположим, что злонамеренный владелец компьютера с адресом 111.111.111.333 настроил свой компьютер так, чтобы он сообщал, что имя его узла `www.yahoo.com`. Вот так и происходит подмена доменного имени.

Поэтому результат, полученный функцией `gethostbyaddr()`, рекомендуется перепроверять вызовом `gethostbyname()`:

```
$ip = "192.168.1.1";
$domain_name = gethostbyaddr($ip);

$ip2 = gethostbyname($domain_name);

// Если была ошибка, $ip2 = $domain_name
if ($ip2 === $domain_name) die("Неверное доменное имя");

if ($ip === $ip2) echo "$ip => $domain_name";
else echo "По адресу $ip находится хакер!!!";
```

## 14.3. ФУНКЦИИ ДЛЯ РАБОТЫ С FTP

FTP (File Transfer Protocol) был, есть и будет самым популярным протоколом обмена файлами. В этом пункте мы рассмотрим, как работать с FTP-

http://all-ebooks.com

сервером. Для работы с сервером мы будем использовать не сокеты, а стандартные функции PHP, что намного проще. Функции, предназначенные для работы с FTP-сервером, описаны в табл. 14.1.

**Таблица 14.1.** *Самые полезные FTP-функции*

| Функция | Назначение |
|---|---|
| bool ftp_cdup (resource ftp_stream) | Переходит в родительский каталог. Возвращает true, если переход удачен, false - в противном случае. Параметр ftp_stream - это идентификатор FTP-соединения |
| bool ftp_chdir (resource ftp_stream, string directory) | Переходит в каталог, указанный параметром directory |
| void ftp_close (resource ftp_stream) | Закрывает FTP-соединение ftp_stream |
| resource ftp_connect (string host [, int port [, int timeout]]) | Открывает FTP-соединение с узлом host, порт задается параметром port. Порт по умолчанию - 21. Третий параметр - это таймаут. По умолчанию таймаут равен 90 секундам |
| bool ftp_delete (resource ftp_stream, string path) | Удаляет файл на FTP-сервере, заданный параметром path |
| bool ftp_exec (resource ftp_stream, string command) | Позволяет выполнить команду на FTP-сервере, указанную параметром command<br>Возвращает true, если команду удалось выполнить |
| bool ftp_fget (resource ftp_stream, resource fp, string remote_file, int mode) | Загружает файл remote_file в файловый указатель fp. Параметр mode позволяет задать режим передачи файла:<br>• FTP_ASCII – текстовый;<br>• FTP_BINARY – двоичный (рекомендуется использовать даже для передачи локальных файлов)<br>В случае успешной загрузки возвращает true, иначе false |
| bool ftp_fput (resource ftp_stream, string remote_file, resource fp, int mode) | Загружает данные из файлового указателя fp в файл remote_file на удаленном сервере. Параметр mode – это режим загрузки (см. выше). Возвращает true, если загрузка завершена успешно, и false в противном случае |
| bool ftp_get (resource ftp_stream, string local_file, string remote_file, int mode) | Загружает файл remote_file с удаленного сервера сразу на локальный компьютер и сохраняет под именем local_file. Эта функция удобнее ftp_fget, которая работает через файловые указатели, поскольку сразу сама создает файл |



http://all-ebooks.com

**Таблица 14.1.** *Самые полезные FTP-функции (продолжение)*

| | |
|---|---|
| bool ftp_login (resource ftp_stream, string username, string password) | Регистрируется на FTP-сервере, то есть передает серверу имя пользователя (username) и пароль (password). Эту функцию нужно вызвать сразу после вызова ftp_connect(), то есть сразу после подключения к серверу |
| int ftp_mdtm (resource ftp_stream, string remote_file) | Возвращает время последнего изменения файла remote_file. Время возвращается в формате UNIX timestamp, то есть его можно передать функции date(). В случае ошибки функция возвращает -1. С каталогами эта функция не работает |
| string ftp_mkdir (resource ftp_stream, string directory) | Создает новый каталог directory. В случае успеха возвращает его имя (то, что вы указали в параметре directory), в случае неудачи false |
| array ftp_nlist (resource ftp_stream, string directory) | Возвращает массив (список) имен файлов в указанном каталоге (параметр directory) |
| bool ftp_pasv (resource ftp_stream, bool pasv) | Если параметр pasv = true, то происходит переключение в пассивный режим FTP |
| bool ftp_put (resource ftp_stream, string remote_file, string local_file, int mode) | Загружает локальный файл local_file на сервер под именем remote_file. Режим задается параметром mode |
| string ftp_pwd (resource ftp_stream) | Возвращает имя текущего каталога |
| bool ftp_rename (resource ftp_stream, string from, string to) | Переименовывает файл на сервере. Исходное имя задается параметром from, а конечное – to |
| bool ftp_rmdir (resource ftp_stream, string directory) | Удаляет указанный с помощью параметра directory каталог |
| int ftp_size (resource ftp_stream, string remote_file) | Возвращает размер заданного удаленного файла |

Продемонстрируем применение FTP-функций в обычном сеансе обмена файлами (листинг 14.3).

**Листинг 14.3. Загрузка файлов на FTP-сервер**

```
<?
// устанавливаем соединение с сервером ftp.firma.ru
$conn = ftp_connect("ftp.firma.ru");

// передаем имя пользователя и пароль
$result = ftp_login($conn, "pupkin", "123456rt");

// зашли на сервер или нет?
```

```
if ((!$conn) || (!$result)) {
        echo "Невозможно соединиться с FTP-сервером!";
        die;
    } else {
        echo "Подключение к серверу выполнено успешно!";
    }

// загружаем файл
$upload = ftp_put($conn, "report.doc", "/home/den/report.doc",
FTP_BINARY);

// загрузился ли файл?
if (!$upload) {
        echo "Произошла ошибка при загрузке файла";
    } else {
        echo "Файл успешно загружен на FTP-сервер";
    }

// закрываем FTP-соединение
ftp_close($conn);
?>
```

Не сильно увлекайтесь передачей больших файлов: помните об ограничении времени выполнения сценария. Может случиться такое, что сценарий просто не успеет передать файл, а его выполнение уже будет прекращено интерпретатором.



http://all-ebooks.com

# ГЛАВА 15.

# ЯЗЫК РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ REGEX

http://all-ebooks.com

## 15.1. ЧТО МЫ ИЩЕМ? МЫ И САМИ НЕ ЗНАЕМ...

Предположим, нам нужно написать программу, которая ищет какое-то слово в файле. Не правда ли, ничего сложного? Неважно, текстовый файл или двоичный, алгоритм поиска *известного* значения в основе своей прост: читаем файл блок за блоком (например, построчно) и проверяем, есть ли искомая подстрока в прочитанном блоке. Разумеется, нужно учесть возможность того, что искомая подстрока окажется в двух смежных блоках, но процесс поиска это не слишком усложнит.

Но существуют задачи, в которых образец для поиска известен не полностью. Например, требуется найти в строке «abs ecu brembo 777» все последовательности символов, окруженные пробелами. Можно, конечно, найти первый пробел, запоминать символы до второго пробела, вывести сохраненную последовательность символов и опять искать следующий пробел. Но такой алгоритм будет выглядеть, мягко говоря, неказистым.

Вот более практический пример: найти в файле все URL. Мы знаем, что URL имеют формат:

```
[http|ftp|gopher]://[узел].[домен][.домен][.домен] [.домен]
```

Имена всех доменов, кроме двух последних, могут представлять собой длинную последовательность символов, а два последних – от 2 до 4 символов. Попробуйте написать сценарий, выполняющий поиск ссылок, а потом посмотрите, каким он получился компактным, быстрым и эффективным. Шутка.



http://all-ebooks.com

## 15.2. РЕГУЛЯРНЫЕ ВЫРАЖЕНИЯ

Приведенные выше примеры объединяет то, что поиск строк, «похожих» на образец, легко выполнить вручную, но далеко не так легко автоматизировать. Конечно, при значительных объемах информации компьютер с самым неэффективным алгоритмом все равно обойдет человека – только потому, что компьютеру несвойственно уставать.

Как же заставить компьютер производить эффективный поиск информации с минимальными для нас затратами времени на написание программы? Выход есть – это механизм регулярных выражений.

Регулярные выражения – это инструкции, описывающие на специально разработанном языке закон «сходства» искомой строки с образцом. Наши два примера будут соответствовать следующим инструкциям:

- на образец похожа любая последовательность символов, с обеих сторон ограниченная пробелами;
- на образец похожа любая строка, содержащая подстроку «http», или «ftp», или «gopher», за которой следует подстрока «://», за которой следуют последовательности символов, разделенные точками, причем после последней последовательности нет точки, и все это завершено пробелом.

Все, что вам нужно сделать, – это вызвать определенную функцию, которой нужно передать одну из указанных инструкций, а все остальное эта функция сделает за вас.

Инструкции, пример которых приведен выше, называются *регулярными выражениями*. Проверка соответствия той или иной строки нашему выражению называется *сопоставлением* строки выражению, а результат поиска – *совпадением*.

В этой главе мы научимся переводить наши инструкции на язык регулярных выражений (RegEx). Поначалу переведенные инструкции будут выглядеть ужасающе, но я сделаю все, чтобы вы уловили суть и научились читать регулярные выражения без этой книги.

## 15.3. ЯЗЫК REGEX

Вы хорошо знакомы с возможностями текстового редактора MS Word? Тогда вы уже знаете и простейший из языков регулярных выражений. Именно на нем основан поиск с помощью подстановочных знаков: по шаблону «г?д» будут найдены слова «гад», «гид» и «год», по шаблону «г*д» – слова «год»

и «город», по шаблону «10{1,3}» – подстроки «10», «100» и «1000». Об остальных несложных правилах составления шаблонов из десятка подстановочных знаков вам всегда напомнит Справка MS Word.

Классический язык регулярных выражений RegEx, который мы будем рассматривать в этой книге, ненамного сложнее. Этот язык поддерживается практически всеми языками программирования, которые вообще поддерживают регулярные выражения. Кроме RegEx, существует еще одна очень распространенная разновидность языка регулярных выражений – Perl-Compatible Regular Expression (совместимые с Perl регулярные выражения). Думаю, не стоит особо говорить, что стандарт PCRE поддерживается языком Perl.

Что же касается PHP, то третья версия PHP поддерживает только RegEx, а, начиная с четвертой, RegEx и PCRE. PCRE обладает более широкими возможностями и более гибок, чем RegEx, но, как я уже говорил, RegEx – классика, и ее нужно знать, чтобы понять более сложные материи, такие как PCRE. О PCRE вы сможете прочитать в любой книге по языку Perl.

Каждое выражение языка RegEx состоит из одной или нескольких управляющих конструкций. Управляющие конструкции можно объединять в группы. Одна группа считается одной операцией.

Управляющая конструкция может состоять из:

- обычных символов, или *литералов*: каждый литерал представляет сам себя;
- управляющих конструкций: квантификаторов, альтернативного оператора, скобок и т. п.;
- псевдосимволов, представляющих невидимые символы, например конец строки.

### 15.3.1. Литералы

Начнем с самого простого – литералов. Как уже было сказано, каждый из таких символов представляет сам себя. Например, выражение 1234 совпадет только с буквальной последовательностью «1234».

К литералам относятся только алфавитно-цифровые символы, то есть A-Z, a-z, 0-9. Символы « . ? * + {} [] | $ ^ » считаются управляющими.

Если вы хотите найти подстроку, содержащую один из управляющих символов буквально (например, строку «a*b»), то искомый символ следует экранировать с помощью обратного слэша: a*b. Однако, поскольку мы



http://all-ebooks.com

программируем на PHP, символ «\» тоже нужно экранировать (п. 9.6.3). В итоге получаем:

- Искомая строка: a*b
- Регулярное выражение на языке RegEx: a*b
- Регулярное выражение на языке RegEx/PHP: a*b

Управляющие символы позволяют группировать символы для поиска не буквальных вхождений, а «похожих». Такой поиск – основное предназначение любого языка регулярных выражений, а группировка символов – его фундамент.

Помните, как задается маска имени файла? Символ «?» обозначает один любой символ в имени файла. В языке RegEx все так же, но вместо символа «?» используется символ точки. Например, регулярному выражению «1.3» соответствуют последовательности

```
103, 113, 123, 143, …, 193, 1a3,…,1z3,1A3,…,1Z3,
```

в которых между единицей и тройкой находится ровно один символ, который может быть любым. Строки «13», «1ab3» и т. п. выражению «1.3» соответствовать не будут.

Если мы ищем только те строки, в которых между единицей и тройкой находится любая цифра, то регулярное выражение пишется так:

```
1[01234567893]
```

Если на месте нашего «любого символа» должна быть буква, перечислите разрешенные буквы в квадратных скобках: например, 1[abx]3. Тогда выражению будут соответствовать строки: 1a3, 1b3, 1x3.

Стоп! А если нужно искать любую букву? Неужели нужно перечислять все 26 строчных букв и 26 заглавных букв? Конечно же нет. Если буквы/символы следуют в алфавитном порядке (или по возрастанию), можно указать знак диапазона «-»:

- a-z – все строчные буквы от a до z;
- A-Z – все заглавные буквы от A до Z;
- 0-9 – все цифры.

Вы можете указать подряд несколько диапазонов: так, выражению [a-nA-N] соответствует любая буква от A до N в любом регистре. Практический пример: для поиска одной шестнадцатеричной цифры используйте выражение [a-fA-F0-9].

http://all-ebooks.com

Любому алфавитно-цифровому символу соответствует группа [a-zA-Z0-9_].

В языке RegEx определены несколько макросов, позволяющих указывать алфавитно-цифровые символы нагляднее. Эти макросы перечислены в табл. 15.1.

**Таблица 15.1.** *Модификаторы RegEx*

| Модификатор | Что обозначает |
|---|---|
| [:alpha:] | Буква |
| [:digit:] | Цифра |
| [:alnum:] | Буква или цифра |
| [:space:] | Пробел или пробельный символ |
| [:blank:] | То же самое, что и space, но учитываются также символы с кодами 0 и 255 |
| [:cnrtl:] | Управляющий символ |
| [:graph:] | Символ псевдографики |
| [:lower:] | Строчная буква |
| [:upper:] | Заглавная буква |
| [:print:] | Печатаемый символ |
| [:punct:] | Знак пунктуации |
| [:xdigit:] | Шестнадцатеричная цифра: цифра или буква от A до F |

Квадратные скобки – это не знак группировки, а признак макроса, то есть их нужно дублировать: так, выражению ABC[0123456789] эквивалентна запись ABC[[:digit:]].

Если выражению должны соответствовать все символы, кроме указанных, то используйте управляющий символ «^». Его можно ставить перед одиночным символом, перечислением или диапазоном:

- [^=] – любой символ, кроме «=»;
- [^A-Z] – любой символ, кроме заглавных букв;
- [^ \t\r\n] – любой символ, кроме пробельных.

### 15.3.2. Управляющие конструкции

Как уже было отмечено, к данной группе относятся квантификаторы, альтернативный оператор, скобки и прочие конструкции, о которых сейчас будет идти речь.



http://all-ebooks.com

Квантификатор – это специальный символ, который используется для уточнений значения символов, идущих перед ним. В основном они служат для указания количества повторений определенных символов. Например, выражению «Z*_» соответствуют строки, содержащие сколько угодно (включая ноль) следующих подряд букв Z, после которых стоит знак подчеркивания. Существуют три квантификатора повторения:

- + – одно или более повторений;
- ? – ноль или одно повторение;
- * – ноль или более повторений.

Примеры:

[a-zA-Z]+-[a-zA-Z]+ : ищет слово, написанное через дефис.

\r?\n : ищет знак перевода каретки перед концом строки.

Квантификатор повторения будет стараться захватить как можно большее число символов. Он не предназначен для того, чтобы найти в строке, например «XXXXXXL» подстроку длиной в один или два символа «X». Для этого служат сложные квантификаторы, позволяющие задать точное число совпадений:

- A{n,k} – символ A может встречаться от n до k раз;
- A{n} — символ A должен встречаться ровно n раз;
- A{n, } — символ A может встречаться n или более раз.

Числа n и k должны принадлежать диапазону от 0 до 255.

### 15.3.3. Псевдосимволы

Эти символы обозначают какую-либо позицию в строке поиска. Например, символ ^ (крышка) означает начало строки, а символ $ – конец строки:

^123 — строки, которые начинаются с подстроки «123»;

123$ — строки, которые заканчиваются подстрокой «123»;

^123$ — этому выражению соответствует только строка «123».

Псевдосимвол [[:<:]] означает начало слова, а символ [[:>:]] – конец слова. Под словом подразумевается любая последовательность из букв и цифр: [[:alnum:]]+.

Внимание! Не путайте псевдосимвол ^ с управляющим символом ^, означающим «все, кроме» (п. 15.3.1). Крышка, стоящая в начале регулярного

выражения, соответствует началу строки, а крышка, стоящая после открывающей квадратной скобки, будет символом отрицания.

В языке RegEx существует только четыре псевдосимвола, зато в языке PCRE их столько, что только им можно было бы посвятить отдельную главу.

### 15.3.4. Альтернативный оператор

Мы рассматривали конструкцию [], которая позволяет задать группу альтернативных символов для определенного символа в строке. В RegEx есть альтернативный оператор (|), позволяющий задать альтернативные группы, например:

- A|b|c – эквивалентно выражению [Abc];
- 123|^1|9$ – этому выражению соответствуют строки, которые содержат подстроку «123», или начинаются с символа «1», или заканчиваются символом «9»;
- Terminator1|Terminator2|Terminator3 – содержат подстроки «Terminator1», «Terminator2» или «Terminator3».

### 15.3.5. Скобки

Запись «Terminator1|Terminator2|Terminator3» можно сократить. Делается это, как в школе – вынесением общей группы символов за скобки:

```
Terminator(1|2|3)
```

Это не единственное предназначение круглых скобок. Гораздо важнее другое их применение: группировка управляющих конструкций. Группировка позволяет не только найти совпадающую подстроку, но и одновременно разобрать ее на нумерованные фрагменты. Чтобы оценить пользу группировки, отвлечемся ненадолго от языка RegEx и вернемся к языку регулярных выражений MS Word.

Допустим, нам нужно найти в тексте даты в формате ДД-ММ-ГГГГ. Ничего сложного: вводим в поле поиска диалогового окна «Найти и заменить» шаблон

```
<[0-9]@-[0-9]@-[0-9]@>
```

Теперь представим, что мы должны изменить формат всех дат в документе на ГГГГ.ММ.ДД. Что вводить в поле замены? Здесь-то и нужно, чтобы поиск не только выделил найденную подстроку, но и присвоил значения трем переменным: первой – ДД, второй – ММ и третьей – ГГГГ. Тогда, введя в



http://all-ebooks.com

поле замены «\3.\2.\1», мы добьемся преобразования всех дат в нужный формат.

Взятие части шаблона в круглые скобки и указывает на то, что фрагмент найденной строки, совпавший с этой частью шаблона, требуется поместить в переменную. В результате успешного поиска создается столько же переменных, сколько в шаблоне пар скобок, а их именами служат порядковые номера пар, считая от 1.

Таким образом, на языке регулярных выражений MS Word замена формата даты выглядит так:

Найти: <([0-9]@)-([0-9]@)-([0-9]@)>

Заменить на: \3.\2.\1

Возвращаясь к языку RegEx, заметим, что регулярное выражение для поиска даты ДД-ММ-ГГГГ, разобранной на те же самые компоненты, на этом языке пишется так:

```
[[:<:]]([0-9]+)-([0-9]+)-([0-9]+)[[:>:]]
```

О том, куда будут помещены найденные компоненты даты, читайте в следующем пункте.

## 15.4. ФУНКЦИИ PHP ДЛЯ РАБОТЫ С РЕГУЛЯРНЫМИ ВЫРАЖЕНИЯМИ

При обработке регулярных выражений средствами PHP нужно помнить, что функция ожидает от вас регулярного выражения в виде строки. Не забывайте также экранировать все символы \ символом \.

Поиск по образцу `$expression` в строке `$str` выполняет функция

```
bool ereg(string $expression, string $str [, list &$matches])
```

Она сопоставляет регулярное выражение `$expression` строке `$str`. Если строка соответствует регулярному выражению, функция возвращает `true`, в противном случае `false`.

В список `$matches` – необязательный параметр, передаваемый по ссылке – функция помещает участки найденной строки, соответствующие фрагментам регулярного выражения `$expression`, взятым в круглые скобки. Сама найденная строка помещается в первый элемент списка, `$matches[0]`.

В отличие от языка регулярных выражений MS Word, вложенность скобок допускается.

http://all-ebooks.com

Следующий фрагмент кода демонстрирует разбор на компоненты даты в формате ДД-ММ-ГГГГ:

```
$s = " 31-12-2006 ";
ereg("^ *(([0-9]+)-([0-9]+)-([0-9]+)) *$", $s, $M);

// Выводим элементы совпадения:
echo "Совпадение целиком: $M[1] <br>";
echo "День: $M[2] <br>";
echo "Месяц: $M[3] <br>";
echo "Год: $M[4] <br>";
```

Почему найденная дата без окружающих пробелов попала в первый элемент массива M, а не только в нулевой? Все дело в четвертой, внешней, паре скобок: она дала первое найденное подвыражение. Следующие подвыражения, в порядке слева направо, попали в элементы массива M с индексами 2, 3 и 4.

Если вам нужно сразу же заменить найденную подстроку, используйте функцию `ereg_replace()`:

```
string ereg_replace(string $expression, string $replacement, string
$str)
```

Эта функция заменяет каждую подстроку строки `$str`, соответствующую регулярному выражению `$expression`, строкой `$replacement`.

В качестве примера рассмотрим вывод текста PHP-программы в окно браузера так, чтобы имена всех переменных были выделены жирным красным шрифтом. Очевидно, что для этого нужно обрамить имя каждой переменной тегами:

```
<b><font color=red>…</font></b>
```

Имя переменной в языке PHP начинается с символа $, за которым следует буква, за которой следует неопределенное (включая ноль) количество букв и цифр, после чего стоит пробел или знак препинания. Символ $ входит в число управляющих, поэтому для его буквального поиска его нужно экранировать. Таким образом, регулярным выражением для поиска переменной будет следующее:

```
\$[a-zA-Z_][[:alnum:]]*[[:>:]]
```

Найденную подстроку мы собираемся заменить ею же, но обрамленной тегами.

Для этого в строке `$replacement` нужно сослаться на найденный фрагмент `$expression`. Это делается точно так же, как в диалоге «Найти и заменить» MS Word: строки, совпадающие со взятыми в скобки подвыражениями, в результате успешного поиска помещаются в «переменные» с



http://all-ebooks.com

именами \n, где n – это порядковый номер пары скобок. В нашем случае аргумент `$replacement` выглядит так:

```
<b><font color=red>\1</font></b>
```

Учитывая, что и `$expression`, и `$replacement` – это строки PHP и экранирующий символ \ в них сам нуждается в экранировании, результирующий вызов будет таким:

```
$s = "<? $a=77; echo $a; ?>";  // фрагмент PHP-программы
echo ereg_replace("(\\$[a-zA-Z_][[:alnum:]]*[[:>:]])",
"<b><font color=red>\\1</font></b>", $s);
```

Следующая пара функций аналогична рассмотренным функциям `ereg()` и `ereg_replace()`, с той разницей, что они выполняют поиск и замену без учета регистра символов:

```
bool eregi(string $expression, string $str [, list &$matches])
string eregi_replace(string $expression, string $replacement,
string $str)
```

## 15.5. ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ РЕГУЛЯРНЫХ ВЫРАЖЕНИЙ

### 15.5.1. Автоматическое добавление ссылок

Начнем с моего любимого примера – с гипертекстовых ссылок. Почему именно с них? Да потому, что в свое время я довольно много времени потратил, чтобы составить нужное регулярное выражение!

Задача состоит в том, чтобы подготовить текстовый документ к публикации в Интернете, заменив все встречающиеся в нем URL гиперссылками:

```
<a href=$URL>$URL</a>
```

URL – это строка, состоящая из:

- имени протокола (http, https, ftp или gopher), за которым следует «://»;
- имени сервера (одна или несколько разделенных точками групп алфавитно-цифровых символов, кроме которых допускается символ «минус») и номера порта («:» и одна или несколько цифр, необязательно);
- необязательного пути к файлу;
- необязательной строки запроса (символ «?», за которым следуют одна или несколько пар «имя=значение», разделенных «&»).

Поскольку мы собираемся извлечь URL целиком, а не разбирать его на компоненты, регулярное выражение для него выглядит очень просто:

Протокол:

```
(https?|ftp|gopher)://
```

Имя узла:

```
[[:alnum:]-]+(\\.[[:alnum:]-]+)*
```

Все остальное:

```
(/[[:alnum:]+&.%]*(\\?[[:alnum:]?+&%]*)?)?
```

Получаем следующий фрагмент кода, решающий нашу задачу:

```
$h="[:alnum:]";

$text = eregi_replace(
    "((https?|ftp|gopher)://[$h-]+(\\.[$h-]+)*" .
      "(/[$h+&.%]*(\\?[$h?+&%]*)?)?)",
        '<a href="\\1">\\1</a>',$text);
```

## 15.5.2. Преобразование адресов электронной почты

Второй пример тоже связан с Интернетом – он занимается преобразованием адресов электронной почты. Задача та же: есть текст, в котором встречаются строки **пользователь@узел.домен**, которые нужно заменить ссылками вида:

```
<a href=mailto:адрес>адрес</a>
```

Решение задачи выглядит так:

```
$text = eregi_replace(
'([[:alnum:]-.]+@[[:alnum]-]+(\\.[[:alnum:]-]+)*(\\?([[:
alnum:]?+&%]*)?)?)',
'<a href=mailto:"\\1">\\1</a>',$text);
```

Мы составили регулярное выражение, выделяющее три части адреса: имя пользователя – алфавитно-цифровые символы до знака @; узел и домен; необязательные параметры сообщения (например, тема) – подвыражение, выделенное жирным шрифтом.

## 15.5.3. Преобразование имен файлов

Стандартной подзадачей при обработке файлов является сохранение исходного файла в виде архивной копии – файла с расширением .bak.



http://all-ebooks.com

Уточним задачу: если у имени файла расширение есть, то его нужно заменить на .bak; если расширения нет, то нужно добавить .bak. Следующий вызов решает эту задачу:

```
$fname=ereg_replace('([[:alnum:]])(\\.[[:alnum:].]*)?$','\\1.
bak',$fname);
```

Другая задача, связанная с именами файлов, заключается в том, чтобы извлечь из пути к файлу имя файла и имя родительского каталога. Решается она так:

```
$filename = ereg_replace(".*[\\/]","",$path);
$dir_name = ereg_replace(" [\\/]?[^\\/]*$","",$path);
```

# ЧАСТЬ IV.

# СТРОИМ СВОЙ САЙТ

http://all-ebooks.com

# ГЛАВА 16.

## ДОМЕН + ХОСТИНГ. КУРС МОЛОДОГО САЙТОСТРОИТЕЛЯ

http://all-ebooks.com

До сих пор все примеры вы тестировали на локальном веб-сервере, но рано или поздно вам захочется показать свой сайт всему миру. Поскольку организация полноценного веб-сервера с выделенным подключением к Интернету – дело довольно дорогое, свой сайт вы, скорее всего, будете размещать на чужом веб-сервере. Значит, настало время поговорить о выборе хостинга.

## 16.1. БЕСПЛАТНЫЙ ХОСТИНГ

Множество хостинг-провайдеров предоставляют так называемый бесплатный хостинг: доменное имя третьего уровня вроде ваш_сайт.провайдер.ru и определенное дисковое пространство.

Но, как все мы знаем, бесплатный сыр бывает только в мышеловке. Расплачиваться за дисковое пространство, предоставленное для вашего сайта, вы будете показами рекламных баннеров. Казалось бы, что тут неудобного? А вот что: некоторые компании не просто предлагают вам добавить их баннер в любое удобное вам место на вашей веб-страничке, а выводят его автоматически, из-за чего сильно страдает дизайн вашего сайта.

Кроме того, ресурсы бесплатного хостинга очень ограничены: не забывайте, что кроме дискового пространства и доменного имени вам нужно наличие хотя бы PHP и MySQL, которые вы найдете не на каждом бесплатном хостинге. Еще один полезный механизм, которого на бесплатном хостинге может не найтись, – это cron, служащий для периодического (по расписанию) запуска ваших заданий. Немаловажно и иметь учетную запись на FTP-сер-



http://all-ebooks.com

вере: гораздо удобнее обновлять сайт и вообще работать с файлами через FTP-клиент, а не средствами браузера.

Бесплатный хостинг можно использовать разве что для того, чтобы ознакомиться с услугами выбранного вами хостинг-провайдера: надежно ли работают серверы, удобна ли служба поддержки. Даже для тестирования ваших проектов я рекомендую вам купить коммерческий хостинг, тем более что цены на хостинг постоянно падают. Не верите? Я могу подсказать вам довольно неплохой коммерческий хостинг за 12,2$ в год. Рекламу хостинг-провайдера в книге делать не буду, но если вы напишете мне, я с удовольствием вам сообщу его адрес. Мой e-mail вы найдете на моем сайте `http://www.dkws.org.ua`. Там же вы сможете почитать отзывы пользователей о различных хостинг-провайдерах.

## 16.2. КОММЕРЧЕСКИЙ ХОСТИНГ

К выбору коммерческого хостинга нужно подходить очень осторожно, ведь вы рискуете своими собственными деньгами. За коммерческий хостинг вы заплатите вперед за полгода; если вам не повезет с провайдером, то потеряете деньги либо следующие полгода будете мучаться. Давайте поговорим о критериях, влияющих на выбор хостинга.

### 16.2.1. Выбор доменного имени

Все домены можно условно разделить на две группы:

- Международные доменные зоны (.COM, .NET, .ORG, .BIZ, .INFO, .WS и др.);
- Зоны государств (.RU, .UA и т. д.).

Конечно, это разделение довольно грубо, но для нас сейчас это неважно. Вам нужно знать, что, регистрируя домен в международной доменной зоне, вы подчеркиваете, что ваш сайт – международный, то есть будет интересен пользователям всех стран. Регистрация домена в зоне второй группы, наоборот, не означает, что он предназначен только для России или Украины: скорее она несет информацию о том, на каком языке представлены материалы на вашем сайте и каким законодательством они регулируются.

Если сайт нужен вам пока только в качестве экспериментальной площадки для тестирования ваших проектов, то имеет смысл выбрать самое дешевое имя третьего уровня вроде «ваш_сайт.kiev.ua». Неважно, что третий уро-

http://all-ebooks.com

вень выглядит несолидно: вы же не собираетесь показывать свой сайт никому, пока он не отлажен. Главное, чтобы на сервере хостинг-провайдера было установлено все необходимое программное обеспечение.

Сайт, предназначенный для того, чтобы служить лицом компании, целесообразно разместить в домене .COM, .RU или .UA. Однако тут есть свои нюансы. Владение сайтом в домене .COM безусловно престижно, как автомобиль Mercedes или ноутбук COMPAQ. Однако раскручивать такой сайт будет немного сложнее, потому что некоторые поисковые машины (например, Яндекс) не индексируют сайты домена .COM. Чтобы Яндекс проиндексировал ваш сайт, вам придется либо заплатить, либо убедить руководство Яндекса, что ваш сайт будет действительно полезен русскоязычным пользователям. Если у вас интересный информационный сайт, то можете пойти по второму пути. Если же сайт вам нужен исключительно для бизнеса, придется заплатить за престиж. Это и есть один из скрытых платежей, о которых мы еще поговорим.

Если сравнивать зоны .COM, .RU и .UA, то лучшим выбором будет зона .RU. Во-первых, ваш сайт будет нормально индексироваться Яндексом. Во-вторых, стоимость домена в зоне .RU довольно невелика, хотя на момент написания этих строк и выше, чем в домене .COM. Точных цен я называть не буду, потому что к моменту выхода книги из печати они могут измениться.

Домен .UA вообще невыгоден. Для его регистрации нужно свидетельство на знак для товаров и услуг. То есть для того, чтобы мне зарегистрировать домен, скажем, dkws.ua, я должен:

- зарегистрировать юридическое лицо (то есть открыть фирму);
- зарегистрировать торговую марку dkws на юридическое лицо (поверьте, это не дешевое удовольствие – цены начинаются от 200 долларов);
- заплатить 70$ в год за домен.

Как видите, обычному пользователю такой домен не нужен.

Советую присмотреться к доменам .COM.RU и .COM.UA: они недорогие и одновременно подчеркивают принадлежность сайта коммерческой структуре. В отличие от зоны .COM Яндекс их индексирует.

Если вам хочется чего-то эксклюзивного, то могу порекомендовать .SU. Если домен .COM я сравнивал с «Мерседесом», то .SU можно сравнить с «Бентли» или дорогущим «Ягуаром» ручной сборки. Да, за эксклюзивность нужно платить. Стоимость домена в этой зоне на данный момент превышала 100 долларов в год.



http://all-ebooks.com

Домен в зоне .ORG, .ORG.UA или .ORG.RU в самый раз подойдет для информационного проекта. Как правило, такие проекты бывают некоммерческими, поэтому размещение в этом домене сайта по продаже чего-либо – плохая идея. Для небольшого информационного проекта также подойдет домен .INFO, тем более что и стоимость регистрации в этом домене совсем невелика.

Домен .NET выбирать не стоит. Как правило, в этом домене регистрируются интернет-провайдеры и другие организации, чья деятельность непосредственно связана с сетью.

### 16.2.2. Кому принадлежит доменное имя?

Этот вопрос нужно уточнить *перед регистрацией* доменного имени. Если домен регистрируется на ваше имя, тогда все в порядке. В случае если вас не устроит хостинг, вы всегда сможете купить себе другой, не изменяя доменного имени.

Если же доменное имя зарегистрировано на хостинг-провайдера, то в случае отказа от хостинга вполне вероятно, что вам придется сменить доменное имя. Это крайне нежелательно, потому что раскрутку сайта после смены имени придется начинать с нуля. Если доменное имя регистрируется на хостинг-провайдера, то с таким провайдером я связываться не советую.

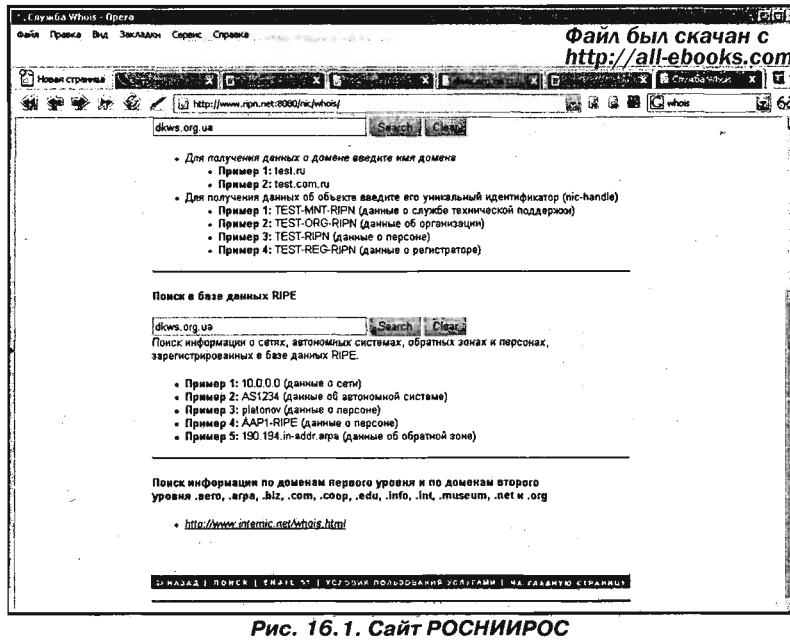



***Рис. 16.1. Сайт РОСНИИРОС***

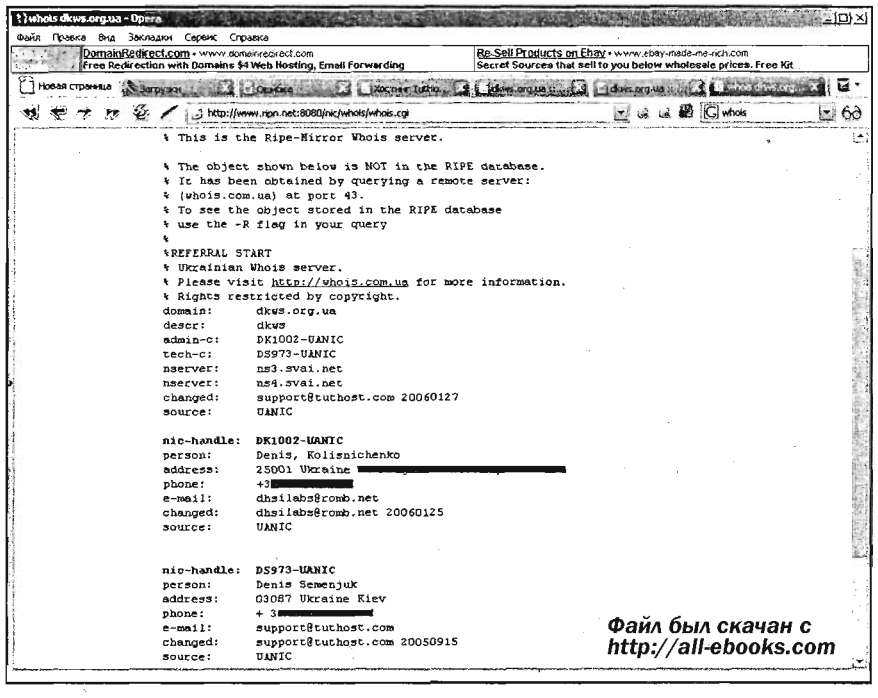



***Рис. 16.2. Информация о владельце доменного имени***

Чтобы немного сэкономить, вы можете заказать доменное имя у независимого регистратора, а хостинг – у не связанного с ним хостинг-провайдера. Этим вы убиваете двух зайцев:

- вы точно будете знать, что доменное имя регистрируется только на вас, потому что покупаете домен без хостинга;
- вы можете выиграть в цене, если выберете такого регистратора, у которого дешевле домен, и такого провайдера, у которого дешевле хостинг. Например, покупаете домен по новогодней акции за 8 долларов в год, а хостинг – по такой же акции у другого хостинг-провайдера, который не предоставляет скидки на домен, зато дает скидку на хостинг.

Если вы решили купить отдельно доменное имя, перед покупкой уточните, предоставляет ли регистратор доменного имени услугу скрытого редиректа. Если да, то покупайте, если нет – не нужно. Что такое скрытый редирект, я расскажу чуть позже.

Где можно зарегистрировать доменное имя? Зайдите в любой поисковик и введите запрос «регистрация доменного имени» или «качественный хос-



http://all-ebooks.com

тинг». Можно было бы привести несколько адресов, но не хочется делать рекламу тем или иным провайдерам.

Проверить, зарегистрирован ли домен, и если да, то на чье имя, можно на сайте Российского НИИ Развития Общественных Сетей:

```
http://www.ripn.net:8080/nic/whois
```

Введите доменное имя своего сайта (без http://) в поле для поиска по базе RIPE и нажмите кнопку Search (рис. 16.1).

После этого вы увидите детальную информацию о том, кто является владельцем вашего доменного имени. Как видно из рис. 16.2, домен `dkws.org.ua` принадлежит мне. Свой телефон и адрес я, по понятным причинам, стер.

### 16.2.3. Наличие скрытого редиректа

Я уже сказал, что доменное имя стоит покупать только у тех регистраторов, которые поддерживают скрытый редирект, а сейчас объясню, что это такое.

Допустим, вы зарегистрировали доменное имя, купили хостинг и разместили на нем свой сайт, но впоследствии решили перенести сайт на бесплатный хостинг. Может быть, у вас кончились деньги, а может быть, вас перестало устраивать качество услуг и вы собираетесь сменить хостинг-провайдера.

После регистрации на бесплатном хостинге ваш сайт получит невзрачный URL вроде «ваш_сайт2006.freehosting.com» вместо солидного «сайт.com», известного к тому же вашим пользователям. Вот для того, чтобы не сообщать им о смене адреса и не терять тех, до кого это сообщение не дойдет вовремя, и нужен редирект, то есть перенаправление запросов пользователя на ваш реальный сайт.

Как это происходит? Если вы просто установите в панели управления доменным именем WWW-редирект на свой новый URL, то все WWW-запросы к доменному имени «сайт.com» будут перенаправляться по адресу «ваш_сайт2006.freehosting.com». Этот процесс будет заметен для пользователя: введя в адресную строку браузера адрес `www.сайт.com`, после загрузки сайта он обнаружит, что в строке адреса указано уже адрес `ваш_сайт2006.freehosting.com`. Неопытного пользователя это может напугать, а опытного навести на нежелательные мысли о вашем финансовом положении.

Скрытый редирект – это перенаправление запросов незаметно для пользователя: загрузив ваш сайт с нового хостинга, пользователь будет по-прежнему видеть в адресной строке «www.сайт.com».

### 16.2.4. Ежемесячная плата за хостинг

Хостинг стоит вообще-то недорого, но загвоздка в том, что арендную плату хостеры берут не за месяц, а как минимум за три, а чаще всего за год. Если хостинг придется менять, вы потеряете деньги не в размере ежемесячной платы, а в значительно большем. Поэтому советую тщательно рассмотреть предлагаемые хостером тарифные планы и выбрать именно то, что подойдет вам.

Ежемесячная плата за хостинг зависит от следующих параметров:

- дискового пространства;
- количества поддоменов;
- количества почтовых аккаунтов;
- установленного программного обеспечения;
- наличия FTP-аккаунтов;
- количества баз данных.

Каждая дополнительная возможность, например FTP-доступ или дополнительный почтовый ящик, может несколько увеличить стоимость вашего хостинга. Это объясняется тем, что служба поддержки тратит дополнительное время на обслуживание вашего сайта. Подумайте, с каким сайтом будет возникать больше проблем (читайте - вопросов от пользователей): с сайтом на 10 Мб, 3 почтовыми аккаунтами, без FTP-доступа и только с PHP; или с сайтом на 100 Мб, неограниченными почтовыми аккаунтами, FTP-доступом и всевозможным программным обеспечением? Теперь давайте поговорим о каждом параметре отдельно.

`Дисковое пространство.` Тут все просто: чем больше места у вас в распоряжении, тем дороже будет хостинг. Можно найти хостинг объемом 10-15 мегабайт стоимостью 0,6$ в месяц. Здесь я могу вам посоветовать одно: не будьте жадными! Зачем вам хостинг на 100 Мб, если ваш сайт не дотягивает и до 10? Про запас? Вы должны понимать, что ваши странички физически записаны на одном большом массиве винчестеров, скажем, в несколько терабайтов. Если у вас сайт 30 Мб, а вам нужно сделать 50, поскольку вы планируете записать еще 10-15 Мб данных, вам нужно просто написать администратору и попросить его увеличить квоту.

`Количество поддоменов.` Количество поддоменов играет роль, когда вы хотите создать действительно большой сайт, например сайт какой-то корпорации. Тогда у каждого подразделения будет свой поддомен, например «sales.firma.com» – для отдела продаж, «dev.firma.com» – для отдела разработки, а «www.firma.com» – общий сайт фирмы.



http://all-ebooks.com

Опять-таки совет: покупайте ровно столько поддоменов, сколько вам нужно. Некоторые хостинг-провайдеры имеют тарифные планы с неограниченным числом поддоменов. Если такой план стоит недорого, покупайте. Если же дорого, подумайте, а нужен ли вам он?

`Почтовые аккаунты.` Хороший и недорогой тарифный план должен включать в себя не меньше 5-10 почтовых аккаунтов, а также неограниченное число почтовых редиректов, то есть записей в конфигурационных файлах сервера, обеспечивающих перенаправление почты. Если в вашем тарифном плане только 3 почтовых ящика, а все остальное – перенаправления, то нужно задуматься о смене хостинг-провайдера: многие хостеры предоставляют 10 и более почтовых ящиков при том же качестве обслуживания и даже за меньшую плату.

`FTP-аккаунты.` FTP-сервером будете пользоваться только вы для обновления сайта, поэтому FTP-аккаунта достаточно одного. Если в вашем тарифном плане нет FTP-аккуанта, то вам нужно либо выбрать более дорогой тарифный план, либо подыскать хостера с более низкими ценами, где вам по карману будет тариф с FTP-аккаунтом.

`Установленное программное обеспечение.` Нам как PHP-программистам необходимо, чтобы на сервере хостинг-провайдера было установлено следующее программное обеспечение:

- интерпретатор PHP версии 4 или 5;
- сервер баз данных MySQL или SQLLite (для PHP5);
- почтовая программа sendmail;
- сервер расписаний crond, позволяющий выполнять указанные вами программы в указанное вами время.

С первыми тремя программами ясно, а зачем crond? Сервер расписаний бывает очень полезен при разработке сложных проектов или для создания небольших «примочек» для вашего сайта. Предположим, что при регистрации на вашем сайте пользователь указывает свое имя, e-mail и дату рождения. Каждое утро crond будет запускать сценарий, поздравляющий пользователей, у которых на момент запуска сценария приходится день рождения. Вам не нужно каждый день заходить на сайт, чтобы разослать поздравительные сообщения вручную. Удобно? Конечно!

Если crond вам не предоставлен, то выкрутиться можно, вставив поздравляющий код в ваш основной сценарий `index.php`: тогда поздравления достанутся только тем пользователям, которые зашли на сайт в свой день рождения. Еще один недостаток этого способа состоит в том, что время загрузки вашего сайта увеличится. А если у вас `index.php` выполняет роль

шаблонизатора, то есть все страницы выводятся только через `index.php`, то ваш сайт будет откровенно «тормозить». Видите, скольких неудобств можно избежать, если позаботиться о наличии сервера расписаний?

Если кроме PHP, MySQL, sendmail и crond будет установлено дополнительное программное обеспечение, то это только плюс вашему хостеру. Обычно также устанавливается интерпретатор Perl, но он нам вообще не нужен, поскольку мы используем PHP. В качестве бонуса на вашем хостинге могут уже быть установлены гостевая книга, форум, чат, система голосований и многое другое. Если у вас такой хостинг, могу только поздравить вас – вам уже не нужно тратить время ни на установку форума, ни на создание гостевой.

Остался еще один момент: перед покупкой хостинга уточните, под управлением какой операционной системы работает сервер вашего хостинг-провайдера. Я бы не стал покупать хостинг, размещенный на сервере под управлением Windows. Думаю, не нужно объяснять почему.

### 16.2.5. Средство управления сайтом

Любое средство управления сайтом должно выполнять следующие функции (так сказать, минимальный джентльменский набор):

- управление поддоменами;
- управление WWW-редиректами;
- управление почтовыми аккаунтами и редиректами;
- управление FTP-аккаунтами;
- управление файлами, в том числе изменение прав доступа к файлам;
- управление заданиями crond;
- управление базами данных. Очень желательно, чтобы на хостинге было установлено средство управления базами данных phpMyAdmin.

Если ваш хостер не поддерживает хотя бы одну функцию, поищите другого, ведь перечисленные функции - это необходимый минимум.

### 16.2.6. Скрытые (косвенные) платежи

Вы счастливый обладатель самого дешевого хостинга? А вы не задумывались, почему он самый дешевый? Возможно, вы что-то не дочитали и в итоге не сэкономите, а даже переплатите?



http://all-ebooks.com

Наибольший удар по карману наносит оплата трафика. Допустим, за хостинг вы платите один доллар в месяц по тарифному плану, включающему неограниченный трафик внутри страны и всего 300 Мб международного трафика (все цифры, понятно, условные). Трафик считается в обоих направлениях, и ваши 300 Мб международного трафика закончатся очень быстро: если каждый пользователь с иностранным IP скачает полностью ваш 10-мегабайтный сайт, то выделенной вам квоты хватит всего на 30 пользователей.

Когда вы используете все 300 Мб, ваш сайт не закроют, а просто в конце месяца предложат оплатить счет, в котором трафик оплачивается помегабайтно. Теперь посчитайте сами: возможно, нужно было купить хостинг за 2 или 3 доллара, но с неограниченным трафиком?

У вас в голове не укладываются все возможные тарифы и условия различных хостинг-провайдеров? Я не буду рекомендовать какого-либо хостера – вывод должны сделать вы сами. Ознакомиться с отзывами о различных хостерах вы можете по адресу `http://www.hostobzor.ru`.

# ГЛАВА 17.

# ПАНЕЛЬ УПРАВЛЕНИЯ ХОСТИНГОМ

http://all-ebooks.com

Итак, вы – счастливый обладатель арендованного хостинга. Вы получили пароль для доступа к панели управления и к FTP-аккаунту. Что дальше? С FTP-аккаунтом все понятно: запустили FTP-клиент (в Windows рекомендую Total Commander), залили файлы на сервер, и все. Но одним управлением файлами при администрировании сайта вам не обойтись. А почтовые аккаунты? А поддомены? А редиректы? Все это и многое другое можно настроить только через панель управления. В этой главе мы рассмотрим следующие панели управления хостингом:

- **DirectAdmin** – довольно удачная (удобная и простая) панель управления;
- **Cpanel** – фактический стандарт: эта панель устанавливается на многих как зарубежных, так и отечественных хостингах.

Мы рассмотрим также phpMyAdmin: эта панель служит не для управления сайтом, а для управления базами данных. Она также является фактическим стандартом, и вы найдете ее практически на всех хостингах.

В предыдущих изданиях этой книги рассматривалась панель управления **Monster Controls**, но, как показала практика, на постсоветском пространстве эта панель оказалась не очень популярной. «Монстры» установлены преимущественно в США, а для отечественных сайтов хостинг в США очень нежелателен из-за медленного доступа к нему. Особенно это ощущается, если у вас или у ваших посетителей модем, а не выделенная линия. Поэтому в настоящем издании было решено заменить панель **MonsterControls** более перспективными и более популярными у нас панелями **DirectAdmin** и **Cpanel**.

Вообще различных панелей управления очень много. Например, довольно популярна панель **Fantastico**. Некоторые хостеры предлагают панели управления собственной разработки. Некоторые из «самодельных» панелей довольно удобны, а некоторые обладают весьма ограниченными функциями. Поскольку трудно угадать, какая именно панель управления будет у вас, мы рассмотрим две самые популярные.

## 17.1. ПАНЕЛЬ УПРАВЛЕНИЯ DIRECTADMIN

Как уже было ранее отмечено, панель **DirectAdmin** довольно удобна и проста в управлении. Именно поэтому она завоевала свое место под солнцем, отобрав часть этого места у **Cpanel**. На моем хостинге установлена именно эта панель управления.

Когда вы зарегистрируетесь в **DirectAdmin**, вы увидите панель управления, изображенную на рис. 17.1.

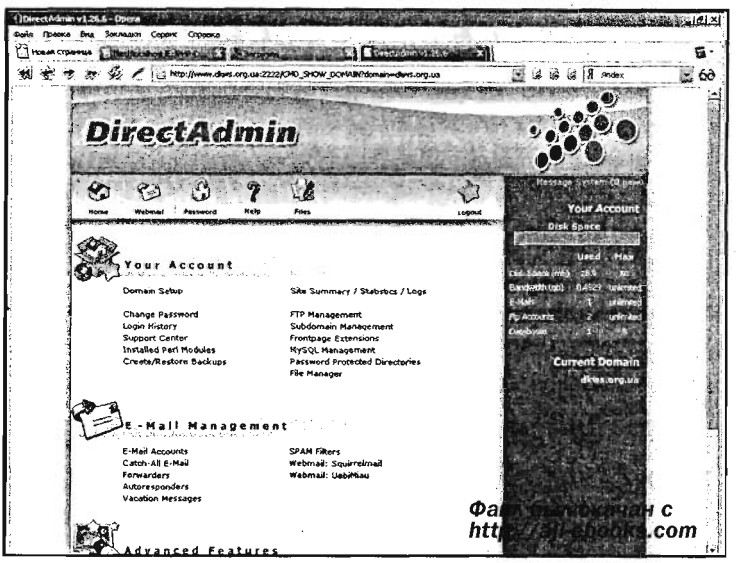



***Рис. 17.1. Панель управления DirectAdmin***

Первое, что бросается в глаза, — это интерфейс на английском языке. Как бы хорошо вы ни знали английский, а работать все равно удобнее в русскоязычном интерфейсе. Поэтому первым делом советую настроить панель управления на русский язык. Для этого:

1. Откройте секцию `Site summary / Statistics / Logs`.
2. Под статистикой будут отображены текущие установки сайта (рис.17.2). Найдите опцию `Language` и выберите из списка значение «ru».
3. Нажмите кнопку `Save`.

Прежде чем пойти дальше, обратите внимание на небольшую информационную панель в правом верхнем углу (рис.17.4). Она показывает, сколько ресурсов (мегабайтов дискового пространства, гигабайтов трафика, почтовых и FTP-аккаунтов, баз данных) предоставил вам хостинг-провайдер и сколько из них вы использовали.

### 17.1.1. Управление учетной записью

Для управления вашим сайтом служит раздел `Your Account`.
Начнем с самого необходимого – смены пароля доступа к панели управления. Сменить его очень просто: откройте ссылку `Изменить пароль`, введите старый пароль, новый пароль и подтверждение нового пароля.



http://all-ebooks.com

| Setting | Current Usage | Maximum Usage |
|---|---|---|
| Bandwidth (meg) | 504.6 | unlimited |
| Total Disk Usage (meg) Update | 28.9 | 50 |
| Email Disk Usage | 14.7 KB | |
| Database Disk Usage | 1.22 MB | |
| # of Domains | 1 | 5 |
| # of SubDomains | 1 | unlimited |
| POP Email Accounts | 1 | unlimited |
| Email Forwarders | 2 | unlimited |
| Mailing Lists | 0 | 0 |
| AutoResponders | 0 | unlimited |
| # of MySQL DBs | 1 | 5 |
| Domain Pointers | 0 | unlimited |
| Ftp Accounts | 2 | unlimited |
| Email | dhsilabs@mail.ru | Save |
| Name | dkwsorgu | Save |
| Language | ru | Save |

***Рис. 17.2. Выбор русского языка***

***Файл был скачан с http://all-ebooks.com***

***Рис. 17.3. DirectAdmin на русском языке***

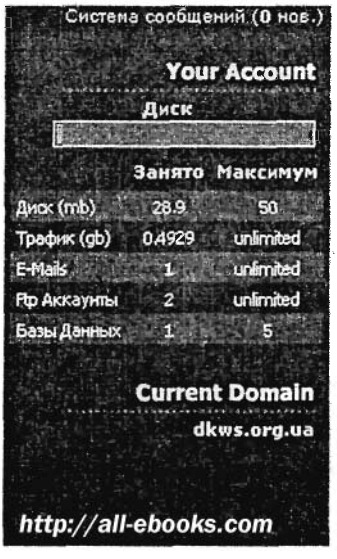



***Рис. 17.4. Сведения об имеющихся ресурсах***

Перечислим остальные модули раздела `Your Account`:

- `Управление доменами`: этот модуль требует особого внимания, поэтому мы рассмотрим его отдельно.
- `История входов`: этот модуль напомнит вам не только дату последнего посещения панели управления, но и IP-адрес, с которого вы заходили, и количество попыток администрирования сайта.
- `Служба поддержки`: этот модуль реализует «книгу жалоб и предложений», адресованную службе поддержки. Но на моем хостинге он не был активирован, поэтому воспользоваться службой поддержки так и не получилось.
- `Установленные Perl модули`: это для вас пока не актуально, вы ведь изучаете PHP.
- `Создать/Восстановить бекап`: этот модуль позволяет создать резервную копию всего сайта. Зайдите в этот модуль и выберите данные, которые вы хотите сохранить (рис. 17.5), после чего нажмите кнопку `Создать бекап`. После создания первой резервной копии чуть ниже появится список всех резервных копий и инструкции по восстановлению из резервной копии.
- `Статистика сайта / Общая статистика / Логи`: позволяет просмотреть статистику и журналы сайта, а также изменить некоторые настройки сайта, например язык, как было показано выше.



http://all-ebooks.com

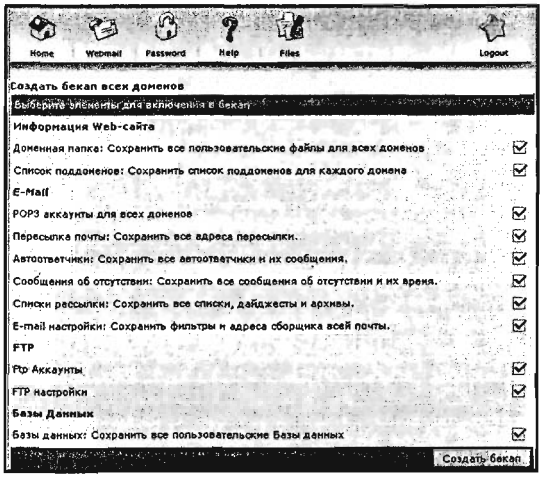



***Рис. 17.5. Создание резервной копии сайта***

- `Управление FTP`: позволяет создавать, удалять и изменять FTP-аккаунты (учетные записи для доступа к FTP-серверу).
- `Управление поддоменами`: мы рассмотрим этот модуль в следующем пункте, вместе с `Управлением доменами`.
- `Серверные расширения Frontpage`: этот модуль предназначен для управления расширениями Frontpage, установленными на сервере провайдера. Эти расширения обеспечивают поддержку компонентов, которые вы могли бы включить в свой сайт, если бы разрабатывали его в Frontpage. Мы не будем использовать этот модуль.
- `Управление MySQL`: позволяет создать новую базу данных, а также запустить средство управления базами данных phpMyAdmin.
- `Защита папок паролем`: с помощью этого модуля вы можете ограничить доступ к тому или иному каталогу сайта только определенным пользователям. Сделать это очень просто: входите в модуль, открываете `Выбрать папку для защиты` и устанавливаете пароль.
- `Менеджер файлов`: большой модуль, который мы также рассмотрим отдельно.

## 17.1.2. Управление доменами и поддоменами

В модуле `Управление доменами` вы можете:

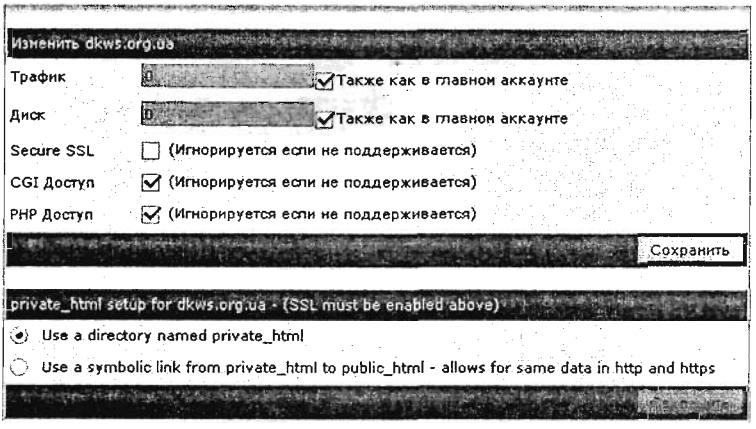



***Рис. 17.6. Изменение параметров домена***

- добавить другой домен (припарковать);
- изменить свое доменное имя;
- изменить параметры своего основного домена.

О первых двух действиях я рекомендую договариваться с вашим доменным регистратором. А вот третье мы сейчас рассмотрим. Выберите свой домен: появится панель управления вашим доменом (рис. 17.6).

Здесь вы можете:

- задать квоты (ограничения) на дисковое пространство и на трафик для вашего домена. Обычно это делать не нужно, поскольку квоты на дисковое пространство и трафик у вас ограничены вашим аккаунтом (учетной записью) и больше, чем вам выделено, вы все равно использовать не сможете;
- включить SSL, CGI, PHP;
- указать, в каком каталоге вы будете хранить документы, отдаваемые только по протоколу HTTPS. HTTPS – это расширение протокола HTTP, поддерживающее шифрование: данные, передаваемые по протоколу HTTP, упаковываются в протокол SSL. То есть, если вы не включите SSL, HTTPS будет для вас бесполезен.

Для изменения параметров поддоменов вам нужно вернуться в главную панель (кнопка `Home`) и выбрать модуль `Управление поддоменами`.



http://all-ebooks.com

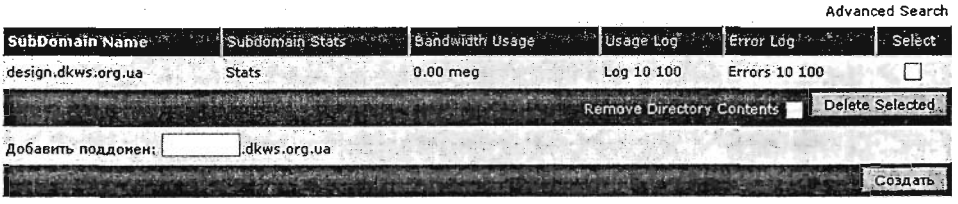



***Рис. 17.7. Панель управления поддоменами***

В этой панели (рис. 17.7) вы можете:

- просмотреть статистику для каждого поддомена (ссылка `Stats` в строке поддомена);
- просмотреть журнал использования домена (ссылка `Log`);
- просмотреть журнал ошибок (ссылка `Errors`);
- удалить поддомен;
- создать поддомен.

Начнем с создания поддомена. Например, вы хотите создать отдельный поддомен для отдела продаж и назвать его «sales». Введите это имя в поле внизу формы и нажмите кнопку `Создать`: у вас появится дополнительный поддомен «sales.ваш_домен», корневым каталогом которого станет созданный для вас каталог `public_html/sales`. В этот каталог вы будете загружать файлы для поддомена «sales» .

Чтобы удалить поддомен, установите флажок `Select` в строке этого поддомена и нажмите кнопку `Delete Selected`. Будьте внимательны: если вы выбрали несколько поддоменов, то удалены будут все выделенные поддомены. Чтобы одновременно удалить содержимое каталога удаляемого поддомена, установите флажок `Remove Directory Contents`.

### 17.1.3. Менеджер файлов

Файловый менеджер позволяет вам управлять файлами в каталоге, выделенном хостинг-провайдером для размещения вашего сайта. Содержимое этого каталога представлено на рис. 17.8. Запустить файловый менеджер вы можете из главной панели DirectAdmin (рис. 17.3), щелкнув по ссылке `Менеджер файлов`.

В каталоге `domains` хранятся все данные о вашем домене. Его подкаталогом является корневой каталог вашего основного домена

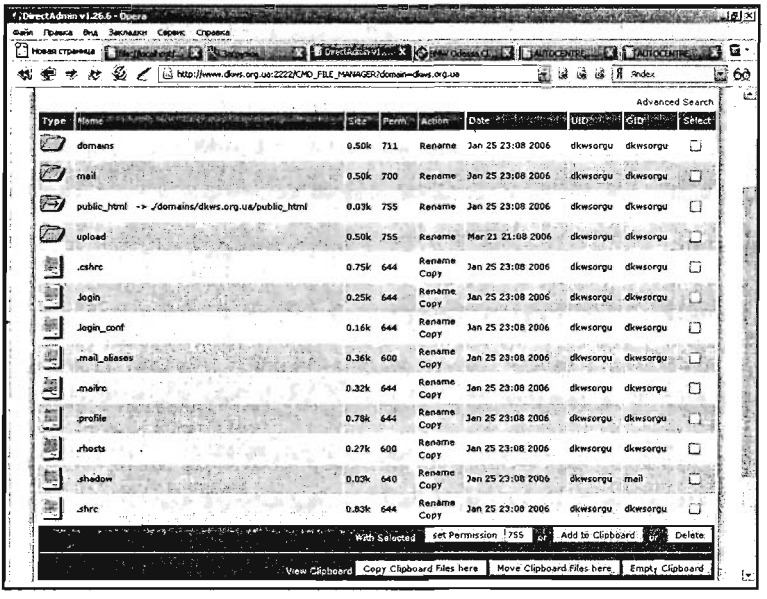



***Рис. 17.8. Менеджер файлов***

`/domains/<имя_вашего_домена>/public_html`, на который ведет ссылка `public_html`. В каталоге `mail` хранится почта домена (в специальных файлах). Каталога `upload` у вас не будет, поскольку я создал его для собственных целей (для закачки на хостинг файлов с последующей их загрузкой на другой компьютер).

Для управления вашими файлами вы можете смело перейти по ссылке `public_html`. Сразу должен отметить, что возможности менеджера файлов несколько ограничены, поэтому его можно использовать разве что для установки прав доступа к файлам – с этой задачей он справляется на отлично. Отметьте флажками те файлы, права доступа к которым нужно изменить, затем в нижней части панели введите нужное значение прав доступа и нажмите кнопку `Set Permissions`. А для остальных операций я рекомендую использовать какой-нибудь FTP-клиент (например, мой любимый Total Commander). Во всяком случае, мне удобнее работать с файлами именно так. Если же вы все-таки решились использовать именно Менеджер файлов панели управления, то давайте рассмотрим основные операции с файлами.

### Удаление файлов и копирование файлов через буфер обмена

Кнопка `Delete` служит для удаления выбранных файлов, а кнопка `Add to Clipboard` – для добавления файлов в буфер обмена. Перед тем, как вос-



http://all-ebooks.com

пользоваться одной из этих кнопок, файлы нужно выбрать, то есть отметить флажками в столбце Select.

Чтобы скопировать файлы в другой каталог, поместите их в буфер обмена (кнопка Add to Clipboard), перейдите в каталог назначения и нажмите кнопку Copy Clipboard Files here. Чтобы переместить файлы в другой каталог, нажмите вместо нее кнопку Move Clipboard Files here.

### Создание файлов и каталогов. Загрузка новых файлов

Для создания нового каталога введите его имя в поле Создать новую папку (рис.17.9) и нажмите кнопку Создать. Аналогично производится создание нового файла. Если рядом с кнопкой Создать (при создании файла) вы отметите флажок HTML-template, то будет создан не пустой файл, а заготовка веб-страницы, то есть текстовый файл с таким текстом:

```
<html>
<head>
     <title>sample1.html</title>
</head>
<body>

<center>
Welcome to sample1.html.
</center>

</body>
</html>
```

Файл был скачан с
http://all-ebooks.com

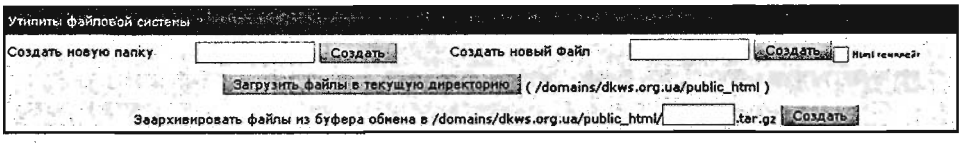



*Рис. 17.9. Утилиты файловой системы*

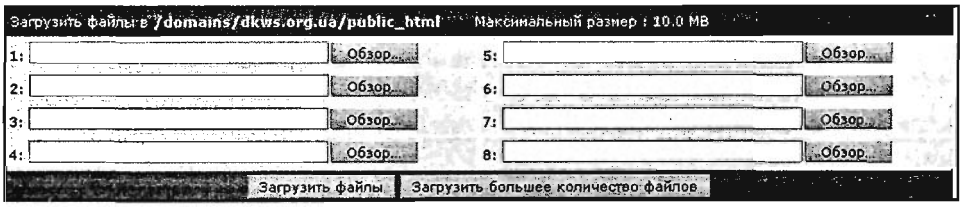



*Рис. 17.10. Форма загрузки файлов*

http://all-ebooks.com

Как видите, некоторые теги вам писать вручную уже не придется. Как говорится, мелочь, а приятно.

Для загрузки новых файлов на сайт (в текущий каталог) нажмите кнопку `Загрузить файлы в текущую директорию`. Появится форма для загрузки восьми файлов. Если вам этого покажется мало (вам нужно загрузить больше файлов), нажмите кнопку `Загрузить большее количество файлов`.

Выберите файлы, нажав кнопку `Обзор` напротив соответствующего поля ввода, и нажмите кнопку `Загрузить файлы`. Максимальный размер загрузки равен 10 Мб.

#### Архивирование файлов

Для архивирования файлов (формат архива tar.gz) сначала нужно добавить файлы в буфер обмена (см. выше), затем ввести имя архива (рис. 17.9) и нажать кнопку `Создать`.

#### Копирование/Переименование/Редактирование файла

В столбце `Action` (рис. 17.8) напротив каждого файла перечислены допустимые действия над этим файлом. Служебные файлы вы имеете право только переименовывать (`Rename`) и копировать (`Copy`), а для страниц вашего сайта разрешено также редактирование (`Edit`, рис. 17.11). Когда вы выбираете действие `Edit`, открывается окно редактора, в котором вы сможете изменить текстовый или HTML-файл. Чтобы просмотреть файл без редактирования, просто щелкните по имени файла.

### 17.1.4. Управление почтой

К управлению почтой относятся следующие модули:

- `POP3 аккаунты`: позволяет создать POP3-аккаунты, количество которых ограничено только вашим тарифным планом.
- `Сборщик всей почты`: позволяет собрать в одном месте всю нерассортированную почту, например письма, адресованные на ваш домен, но несуществующему адресату.

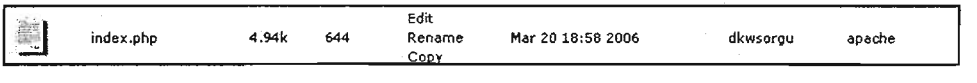



***Рис. 17.11. Действия над файлами***



http://all-ebooks.com

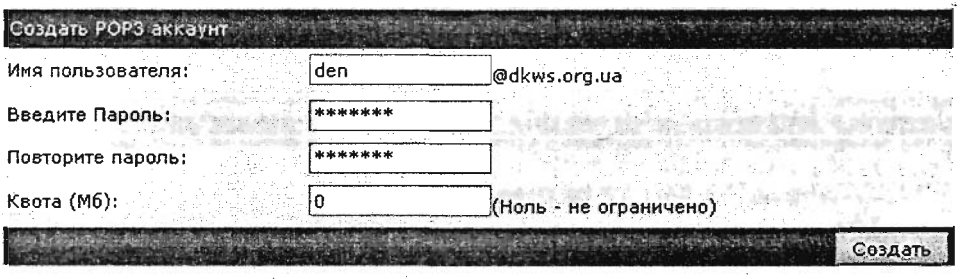



*Рис. 17.12. Создание нового POP3-аккаунта*

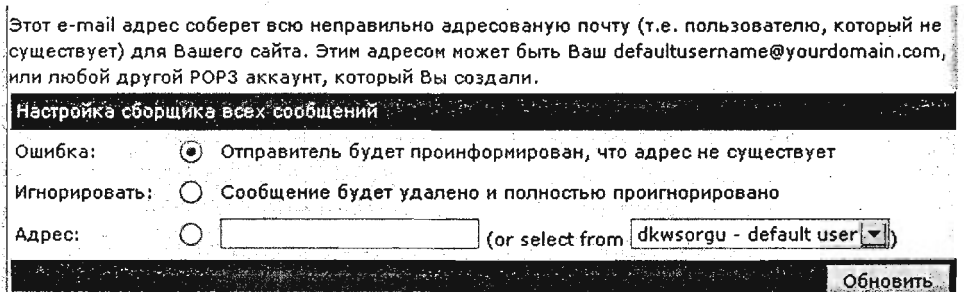



*Рис. 17.13. Сборщик почты*

- `Пересылка почты`: позволяет настроить списки пересылки почты.
- `Автоответчики`: с помощью этого модуля вы можете настроить свой автоответчик.
- `Сообщения об отсутствии`: здесь вы можете настроить сообщение, уведомляющее отправителя письма о том, что вы в отпуске.
- `Фильтры` - это модуль - аналог черного списка на Mail.ru. Вы можете добавить в него как отдельные адреса, так и целые домены.
- `Web-почта: Squirrelmail` и `Web-почта: UebiMiau` – это почтовые веб-интерфейсы. Мы не будем их рассматривать, потому что они не сложнее привычного Mail.Ru.

## POP3-аккаунты

Начнем с POP3-аккаунтов. Предположим, что у нас есть домен «dkws.org.ua». Если вы создадите почтовый аккаунт «den», то другие пользователи смогут отправить почту по адресу «den@dkws.org.ua». Попробуем создать такой аккаунт. Зайдите в модуль `Pop3 аккаунты` и перейдите по ссылке `Создать POP3 аккаунт`.

http://all-ebooks.com

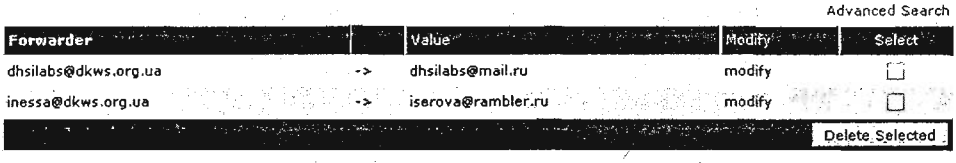



**Рис. 17.14. Список пересылки почты**

Вам нужно ввести имя аккаунта (латиницей), пароль, подтверждение пароля и квоту в мегабайтах. Если квота не задана (0), то размер почтового аккаунта будет ограничен только свободным местом на хостинге.

### Сборщик всей почты

Пусть некто отправил письмо по адресу «ivan@dkws.org.ua». Домен «dkws.org.ua» зарегистрирован, а вот почтовой учетной записи «ivan» на нем нет. Что же делать с таким письмом? Можно отправить его обратно с пометкой, что пользователь ivan не зарегистрирован на POP3-сервере. Можно полностью проигнорировать - ничего никому не отправлять. А можно отправить по заданному адресу, например на адрес администратора сайта. Любое из этих действий можно задать с помощью сборщика почты.

### Пересылка почты

Рассмотрим пример списка пересылки почты (рис. 17.14).

Мы регистрируем адрес «dhsilabs@dkws.org.ua», но все письма, которые приняты по этому адресу, будут отправлены на адрес `dhsilabs@mail.ru`. Это очень удобно. Например, у вас всего лишь 100 Мб дискового пространства, всего 5 из которых свободно. Ваш знакомый хочет передавать вам важную информацию общим объемом более 5 Мб. Вы можете установить переадресацию на какой-нибудь бесплатный почтовый сервер, на котором есть ваша учетная запись. И все будут довольны: получатель сможет отправить вам файлы без всякого ограничения (например, на `narod.ru` и на Рамблере выделяются почтовые ящики размером по 100 Мб), а вы сможете принять письма большого размера абсолютно бесплатно - вам не нужно будет платить за дополнительное место. При этом ваш адрес будет на `rambler.ru`, в собственном вашем домене.

### Фильтры

Давайте разберемся, как можно настроить фильтры. Если вам нужно заблокировать какой-то определенный адрес, тогда введите его в поле `Блокиро-`



http://all-ebooks.com

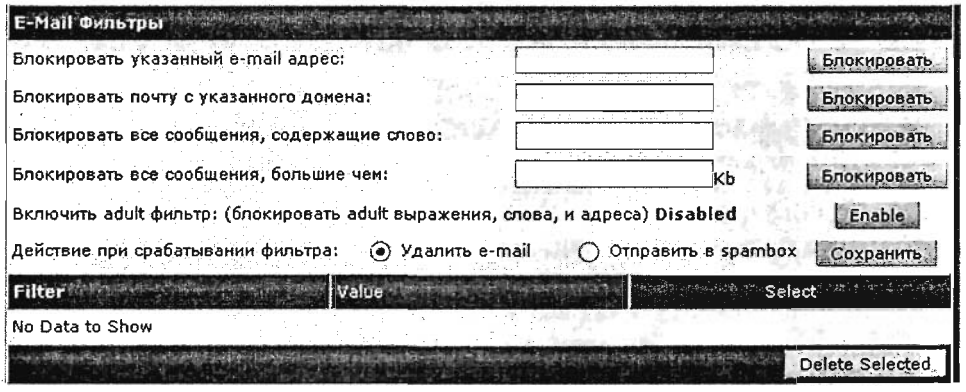



***Рис. 17.15. Окно настройки фильтров***

вать указанный e-mail адрес и нажмите кнопку Блокировать. Аналогично можно блокировать весь домен, но имейте в виду, что тогда вы не сможете получить почту ни от одного из пользователей этого домена. Также можно установить максимальный размер сообщения (в Кб).

Все сообщения, блокированные фильтром, могут быть либо удалены (если установлен режим Удалить e-mail), либо отправлены в ящик для спама (если установлен режим Отправить в spambox).

Также вы можете включить фильтр adult-содержимого (содержимого для взрослых), особенно это актуально, если вы посещаете «сайты для взрослых» и у вас хватило мужества оставить там свой e-mail.

### 17.1.5. Дополнительные функции

К дополнительным функциям относятся следующие модули:

- Системная информация: позволяет просмотреть информацию о сервере и об установленном на нем программном обеспечении.
- Cron задания: позволяет настроить выполнение задач по расписанию. Вам нужно, чтобы каждый день в определенное время запускался тот или иной сценарий? Тогда этот модуль для вас.
- Типы MIME: с его помощью можно отредактировать типы MIME.
- Обработчики Apache: позволяет установить обработчики Apache, используя графический интерфейс, а не конфигурационные файлы Web-сервера.

- `Собственные страницы ошибок`: вместо стандартной страницы с сообщением об ошибке вы можете показывать пользователям собственную, проявив все свои способности в области дизайна.
- `Интерфейс phpMyAdmin`: запускает утилиту управления базами данных phpMyAdmin.
- `Перенаправления сайта`: вы перенесли какую-то страницу в другой каталог, но пользователи все еще обращаются к ней по старому URL? Тогда вам нужно установить перенаправление запросов с исходной страницы на новый адрес.
- `Доменные указатели`: позволяет припарковать домен к вашему домену, то есть создать доменный указатель. Посоветуйтесь со службой поддержки относительно использования этого модуля.
- `SMTP Usage Stats`: позволяет просмотреть статистику использования SMTP.

В предыдущей главе я говорил, что наличие у хостинг-провайдера сервера расписаний crond будет весомым доводом в пользу выбора именно этого хостинга, и приводил в пример сценарий поздравления ваших пользвателей с днем рождения. Рассмотрим этот пример подробнее.

Пусть у вас есть сценарий `happy_birthday.php`, расположенный в корневом каталоге вашего сайта, то есть в каталоге `public_html`. Этот сце-

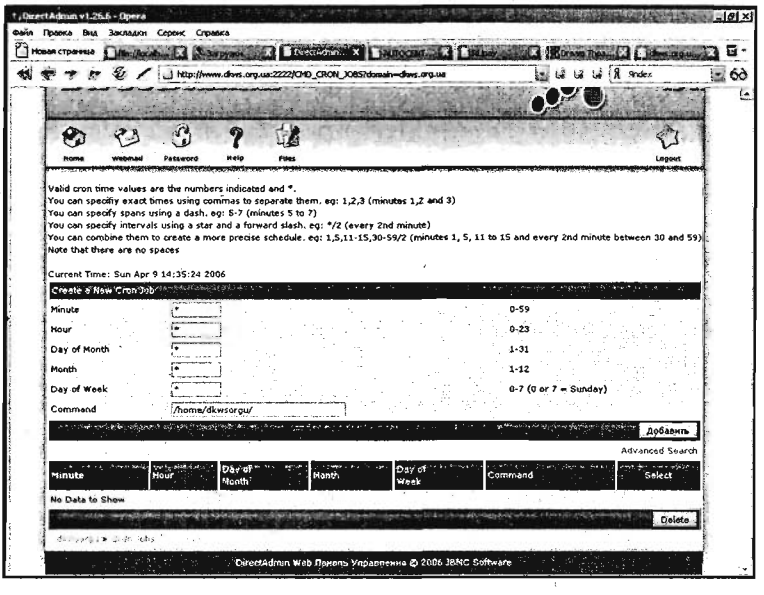



***Рис. 17.16. Создание нового задания Cron***



http://all-ebooks.com

нарий проверяет, у кого из зарегистрированных пользователей сегодня день рождения (дни рождения хранятся в базе данных) и отправляет этому пользователю поздравительную открытку. Вы хотите, чтобы этот сценарий запускался каждый день в 7 часов 15 минут.

Откройте модуль Cron задания (рис.17.16) и введите в поле Hour (часы) значение 7, а в поле Minute (минуты) значение 15. В полях Day of Month, Month и Day of Week оставьте значение по умолчанию «*», говорящее о том, что задание должно выполняться каждый день каждого месяца.

Осталось указать команду, которая будет выполняться в это время. Такой командой не может быть путь к PHP-сценарию, поскольку с точки зрения системы это не исполняемая программа, а обычный текстовый файл. Командный файл, запускающий наш PHP-сценарий, мы сейчас и напишем.

Для начала нужно знать, в каком каталоге установлен интерпретатор PHP: обычно это бывает каталог /usr/bin, но на всякий случай уточните его расположение у службы поддержки. После этого создайте в корневом каталоге вашей учетной записи файл с именем «hb» без расширения. Обратите внимание, что файл нужно создать в корневом каталоге учетной записи, а не сайта, то есть в том каталоге, который открывается первым, когда вы запускаете Менеджер файлов, а не в каталоге public_html. Для прав доступа к файлу hb установите значение 755, то есть объявите его исполняемым.

Содержанием командного сценария hb будут следующие строки:

```
#!/bin/bash
/usr/bin/php  /home/<ваш_логин>/public_html/happy_birthday.
php
```

> **Примечание.**
>
> Мы считаем, что у хостинг-провайдера установлена ОС Unix, и написали сценарий для bash – командной оболочки Unix. Первая строка указывает, что обработкой сценария должен заниматься интерпретатор bash, а вторая строка вызывает интерпретатор php и передает ему файл happy_birthday.php

Если файл happy_birthday.php находится в каталоге public_html не основного домена, тогда последнюю строку нужно изменить так:

```
/usr/bin/php  /home/<ваш_логин>/domains/<домен>/public_html/
happy_birthday.php
```

Вернитесь к заданиям Cron и введите в поле Command команду «/home/<ваш_логин>/hb». Первая часть команды (/home/<ваш_логин>/) уже будет указана, вам останется только добавить «hb».

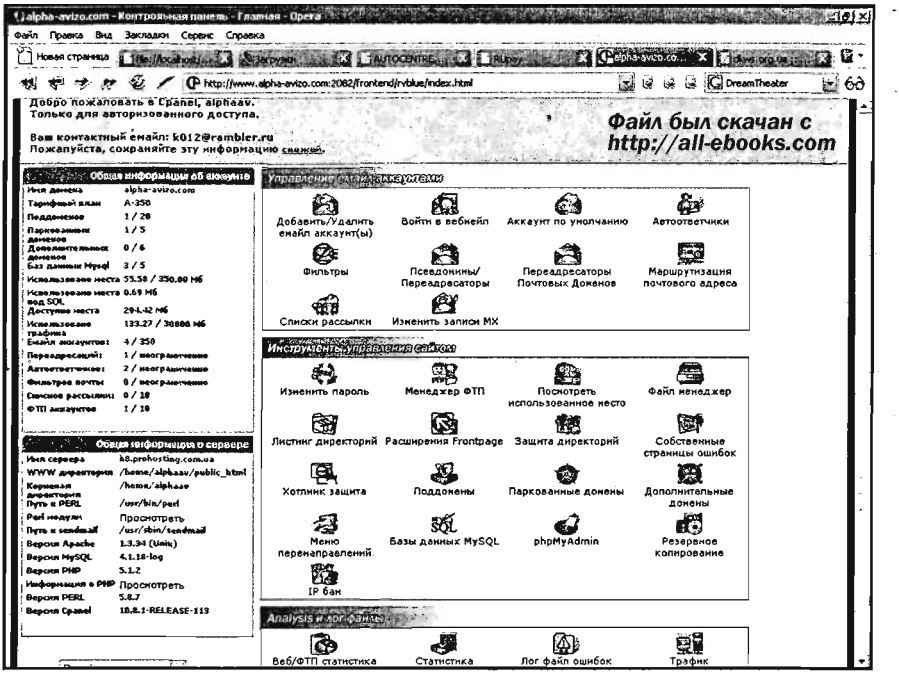



***Рис. 17.17. Панель управления Cpanel***

## 17.2. ПАНЕЛЬ УПРАВЛЕНИЯ CPANEL

Эта панель управления стала фактическим стандартом и встречается у хостинг-провайдеров чаще, чем **DirectAdmin**. Панель **Cpanel** выглядит так, как показано на рис. 17.17.

Если у вас панель **Cpanel** на английском языке, то переключить ее на русский очень просто. В самом низу окна находится модуль `Language`. Откройте его, выберите из списка русский язык (Russian), а потом нажмите кнопку `Change`. Можно поступить еще проще — под информационным блоком слева есть список выбора языка.

Все модули **Cpanel** мы рассматривать не будем. Причин тому две: либо модуль попросту не востребован (редко используется), либо прост до безобразия, так что любые комментарии просто бессмысленны. В частности, мы вообще не будем рассматривать раздел журналов и статистики: в нем вы разберетесь сами без особого труда.

Прежде чем перейти к знакомству с отдельными модулями **Cpanel**, обратите внимание на панель в левой части со сводной информацией о вашем хос-



http://all-ebooks.com

тинге. На ней вы можете узнать более подробно о вашем тарифном плане, а также о программном обеспечении, установленном на сервере.

### 17.2.1. Управление почтовыми аккаунтами

Для управления почтой Cpanel предоставляет следующие модули:

- `Добавить/удалить email аккаунты`: позволяет добавить, удалить, а также просмотреть настройки POP3-аккаунтов.
- `Фильтры`: служит для создания черного списка адресатов.
- `Списки рассылки`: очень полезный модуль, позволяющий создать собственную рассылку и управлять ею с панели управления.
- `Войти в вебмейл`: запускает почтовый веб-интерфейс.
- `Псевдонимы/Переадресаторы`: позволяет настроить почтовые псевдонимы и переадресацию.
- `Изменить записи MX`: позволяет изменить MX-записи домена. Перед использованием этого модуля проконсультируйтесь у службы поддержки.
- `Аккаунт по умолчанию`: позволяет указать e-mail адрес, на который будет направлена вся нерассортированная почта (например,

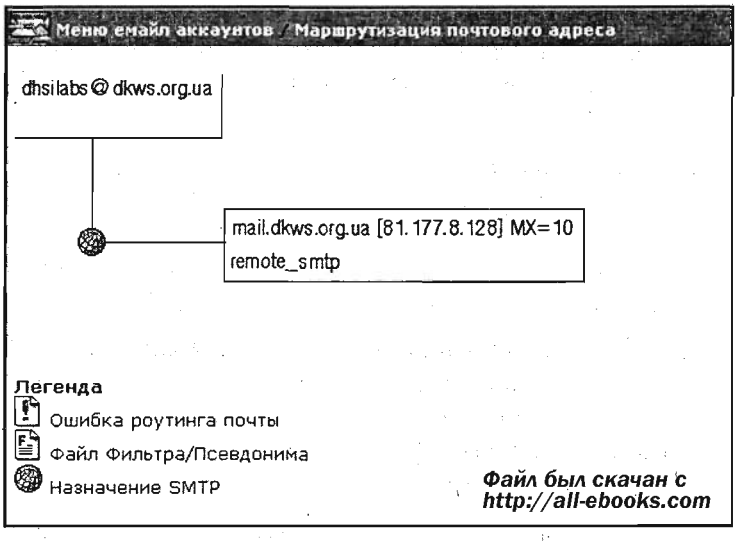



*Рис. 17.18. Маршрутизация почтового адреса*

сообщения, адресованные несуществующим пользователям вашего домена).

- `Переадресаторы почтовых доменов:` перед использованием этого модуля посоветуйтесь со службой поддержки, чтобы решить, нужно ли это вам.
- `Автоответчики:` название этого модуля говорит само за себя.
- `Маршрутизация почтового адреса:` позволяет узнать маршрут следования сообщений электронной почты до указанного вами адреса (рис. 17.18).

### Добавить/удалить email аккаунты

Этот модуль отображает все созданные POP3-аккаунты, а также занимаемое ими дисковое пространство и квоты на это пространство. Для просмотра почты того или иного аккаунта нажмите кнопку `Webmail` (рис. 17.19). Несмотря на то что вы администратор, у вас потребуют пароль для доступа к этому аккаунту.

Для удаления почтового аккаунта нажмите кнопку `Delete`. Кнопка `Quota` позволяет изменить квоту (ограничение на размер занимаемого дискового пространства) для почтового аккаунта. Кнопка `Password` позволяет изменить пароль для аккаунта, даже если вы не знаете старого пароля. Кнопка

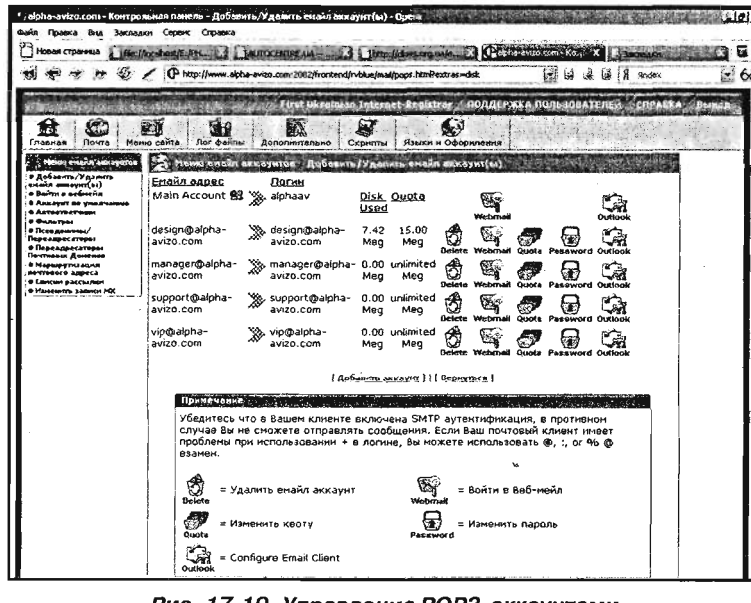



***Рис. 17.19. Управление POP3-аккаунтами***



http://all-ebooks.com

Outlook позволяет автоматически настроить почтовую программу Outlook для работы с тем или иным почтовым аккаунтом.

Как вы видите, менеджер POP3-аккаунтов в **Cpanel** более «продвинутый», чем в DirectAdmin. А что же с фильтрами?

### Фильтры

Организовать «черный список» вы можете с помощью модуля Фильтры. Этот модуль более гибкий, чем аналогичный в **DirectAdmin**, и поэтому сложнее в использовании. Откройте этот модуль и выберите Добавить фильтр. Перед вами появится форма подготовки фильтра черного списка (рис. 17.20).

Давайте разберемся, как создать фильтр.

- По какому полю фильтровать сообщения:
  - От: по отправителю;
  - Тема: по теме;
  - Кому: по получателю;
  - Тело: поиск будет произведен только по тексту сообщения;

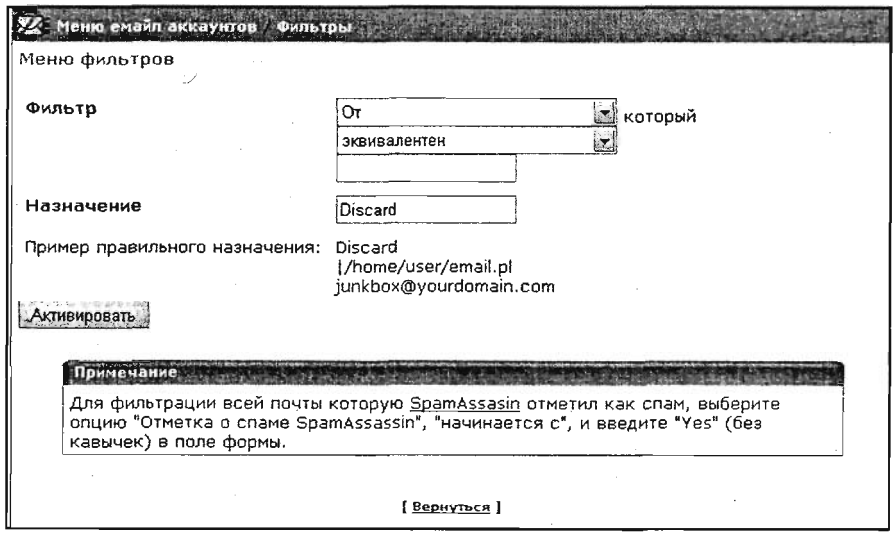



*Рис. 17.20. Создание почтового фильтра*

- Любой заголовок: поиск будет произведен по всем заголовкам сообщения.

- Условие поиска:

  - Эквивалентен: указанное вами значение должно точно совпадать со значением выбранного заголовка (От, Тема, Кому и т. д.);

  - Соответствует выражению regex: поддержка regex очень кстати, поскольку ничто не позволяет организовать поиск так гибко, как регулярные выражения. Надеюсь, вы разобрались с регулярными выражениями в главе 15.

  - Содержит: будьте осторожны с этим условием. Ведь если вы укажете подстроку «put», то под этот фильтр попадут также слова, содержащие эту подстроку, например «computer». Если вы хотите исключить именно слово « put », обязательно окружите его пробелами.

  - Начинается с: заголовок должен начинаться с указанного вами слова.

- Назначение: здесь вы можете указать, куда должно быть доставлено сообщение, удовлетворяющее заданному вами фильтру:

  - Discard: *сообщение будет удалено;*

  - |имя_файла - сообщение будет записано в указанный файл;

  - <e-mail> - сообщение будет отправлено по указанному e-mail адресу.

*Заполнив форму фильтра, нажмите кнопку* Активировать.

### Псевдонимы/Переадресаторы

Этот модуль позволяет настроить переадресацию почты. Допустим, у вас есть учетная запись на каком-то бесплатном почтовом сервере и вы хотите сэкономить на своем дисковом пространстве. Правильно, зачем платить за место, если это самое место можно получить бесплатно? Поэтому можно создать переадресатор. Для внешних пользователей переадресатор будет выглядеть так же, как и обычный почтовый адрес, то есть «имя@ваш_домен». Однако принятая почта не будет храниться на вашем сервере, а будет переадресована по указанному вами адресу.



http://all-ebooks.com

#### Автоответчики

Этот модуль позволяет создать автоответчики для любого POP3-аккаунта вашего домена. В отличие от **DirectAdmin**, **Cpanel** не позволяет создавать сообщения об отсутствии, что несколько неудобно, если отсутствуешь постоянно в один и тот же период времени (например, отпуск или плановые командировки). Но на это время можно отредактировать сообщения автоответчика.

### 17.2.2. Инструменты управления сайтом

В этой группе (рис. 17.17) находятся следующие модули:

- **Изменить пароль:** позволяет изменить пароль доступа к панели управления, зная старый пароль.
- **Листинг директорий:** позволяет включить/выключить листинг директорий для каждого каталога сайта. Например, пользователь вводит URL «http://domain/directory/». Если в каталоге `directory` нет файлов `index.*` (с расширениями `.htm`, `.php` и т. п.), то сайт может отобразить либо листинг каталога (то есть список файлов и подкаталогов), либо надпись «Access denied». Предпочтительно запретить вывод листинга директории.
- **Хотлинк защита:** об этом мы поговорим отдельно.
- **Меню перенаправлений:** позволяет установить HTTP-перенаправления для определенных страниц. Например, у вас была страница «http://ваш_домен/design/prices.html». Если вы перенесли ее в другой каталог или даже на другой сайт, то с помощью этого модуля вы сможете перенаправить HTTP-запросы к этой странице на новый адрес.
- **IP бан:** позволяет запретить доступ к вашему сайту из любого диапазона IP-адресов. Например, если кто-то пытается взломать ваш сайт, то вы можете закрыть доступ из всей его подсети.
- **Менеджер ФТП:** менеджер FTP в **Cpanel** намного удобнее, чем в DirectAdmin, поскольку позволяет не только управлять учетными записями FTP, но и прерывать любые FTP-соединения, а также завершать незавершенные FTP-соединения.
- **Расширения FrontPage:** аналогичный модуль **DirectAdmin** описан в п.17.1.2. Мы его использовать не будем.
- **Поддомены:** управление поддоменами мы рассмотрим отдельно.

- **База данных MySQL:** этот модуль мы тоже рассмотрим отдельно.
- **Просмотреть использованное место:** выводит, сколько дискового пространства занимает указанный каталог.
- **Защита директорий:** позволяет указать пароль для доступа к тому или иному каталогу сайта.
- **Паркованные домены:** домены, которые являются «синонимами» вашего основного домена. Все обращения к паркованным доменам будут переадресованы к вашему основному домену.
- **phpMyAdmin:** запускает интерфейс **phpMyAdmin**.
- **Файл менеджер:** менеджер файлов лучше рассмотреть отдельно, поскольку тут парой слов не обойдешься.
- ***Собственные страницы ошибок:*** *позволяет создать собственные* страницы с сообщениями об ошибках вместо стандартных.
- **Дополнительные домены:** позволяет определить дополнительные домены.
- **Резервное копирование:** создает резервную копию вашего сайта.

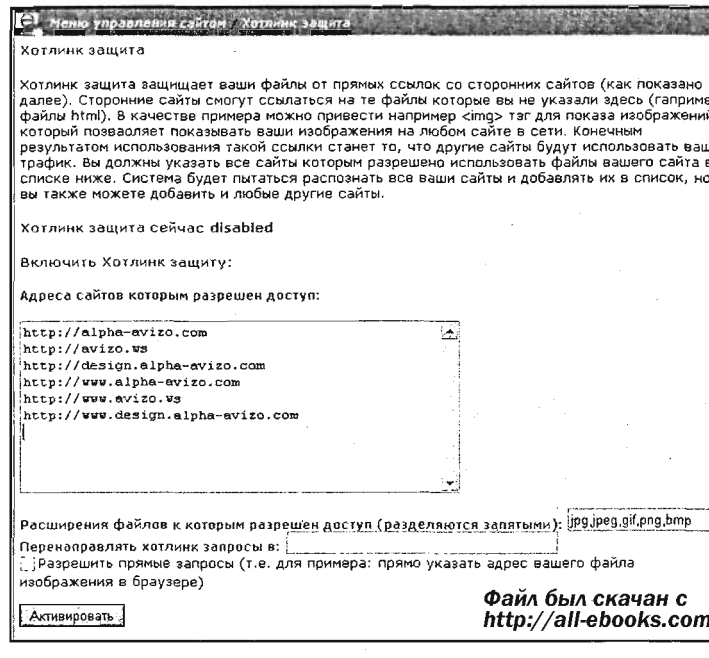



***Рис. 17.21. Настройка Hotlink-защиты***



http://all-ebooks.com

### 17.2.3. Хотлинк-защита

Hotlink-защита – это механизм, запрещающий чужим сайтам ссылаться на файлы, выложенные на вашем сайте. Зачем он нужен? Представьте, что вы выложили у себя какое-нибудь увесистое изображение, например карту страны размером 8 Мб. А некто Пупкин поставил у себя ссылку на это изображение. Откуда будут качать карту пользователи сайта Пупкина? От вас. И за исходящий трафик тоже будете платить вы. Мало того, что это невыгодно, так пользователи еще и будут принимать Пупкина за автора ваших материалов.

Если вы включите Hotlink-защиту, то сторонние сайты смогут ссылаться только на те файлы, расширения которых вы *не указали* в окне настроек (рис. 17.21). Доступ к файлам с этими расширениями вы можете предоставить другим *вашим же* сайтам, перечислив их адреса в списке разрешенных сайтов. После этого для включения Hotlink нажмите кнопку `Активировать`.

### 17.2.4. Менеджер ФТП

Как уже было отмечено, менеджер FTP состоит из двух частей (рис. 17.22): первая позволяет редактировать FTP-аккаунты, а вторая – управлять FTP-соединениями.

Рассмотрим первую часть менеджера. Для добавления FTP-аккаунта служит ссылка `Добавить ФТП аккаунт`. Вам нужно указать имя пользователя (логин), пароль, задать квоту (чтобы не ограничивать пользователя

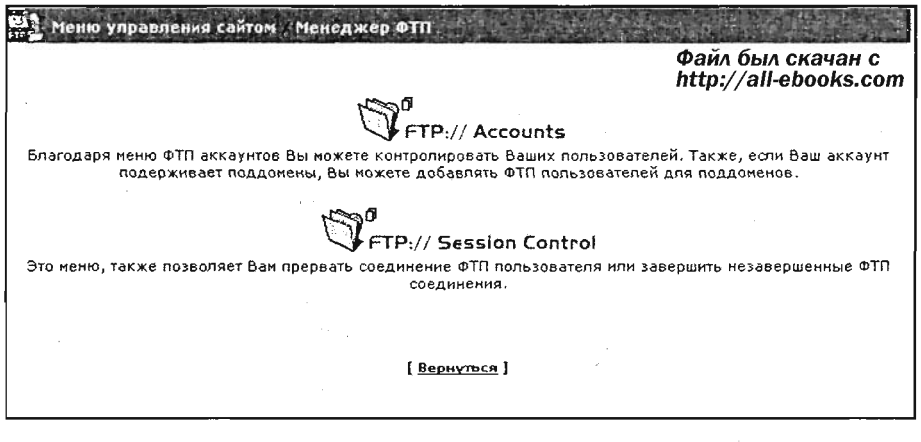



*Рис. 17.22. Менеджер FTP*

http://all-ebooks.com

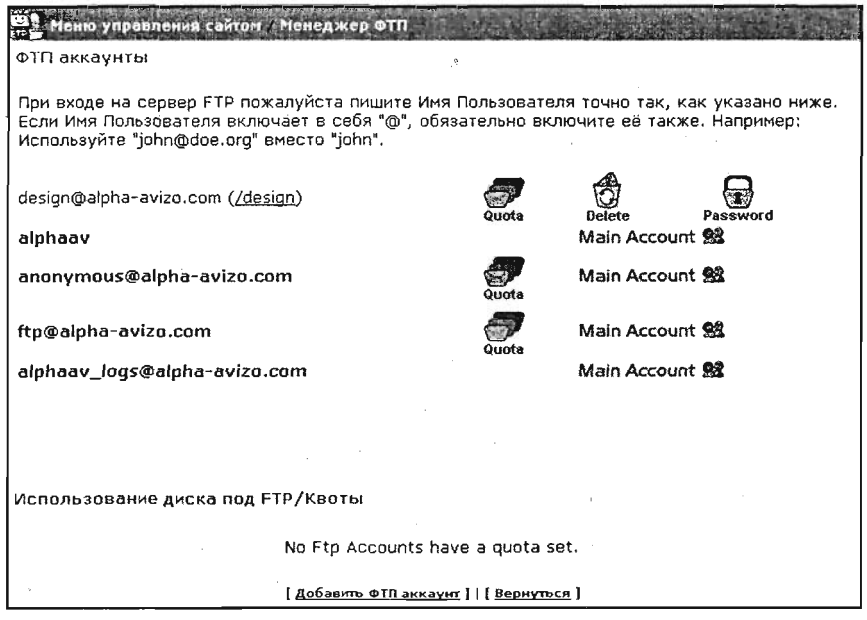



***Рис. 17.23. Управление FTP-аккаунтами***

квотой, укажите значение «unlimited»). Также вам нужно указать путь к каталогу пользователя относительно каталога «/home/<имя_пользователя_главной_учетной_записи>/public_html». Если вы укажете каталог `/`, то пользователь получит доступ к любому объекту в каталоге `public_html`, то есть он сможет удалять и модифицировать любые файлы и каталоги вашего сайта. Если же вы ограничите его собственным каталогом, например, `/user_files`, то пользователь сможет получить доступ только к подкаталогу `user_files` каталога `public_html`. Этот каталог станет для него «корневым», то есть пользователь не выйдет за пределы этого каталога, а любые операции над файлами он сможет выполнять только в каталоге `public_html/user_files` и его подкаталогах.

Когда вы создадите аккаунты, напротив каждого созданного вами (а не системного) будут указаны три пиктограммы: **Quota, Delete** и **Password**. Первая позволяет изменить квоту для FTP-аккаунта (то есть максимальный размер каталога `user_files` из предыдущего примера), **Delete** служит для удаления аккаунта, а **Password** – для изменения пароля.

Теперь перейдем ко второй части – к управлению FTP-соединениями. Когда вы войдете в этот модуль, вы увидите список текущих FTP-соединений (рис. 17.24), напротив каждого из них будет кнопка **Disconnect**, нажатие на которую позволит вам завершить это соединение.



http://all-ebooks.com

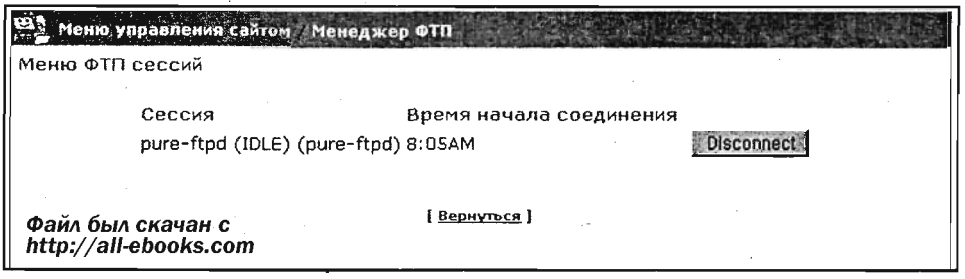



*Рис. 17.24. Завершение FTP-соединений*

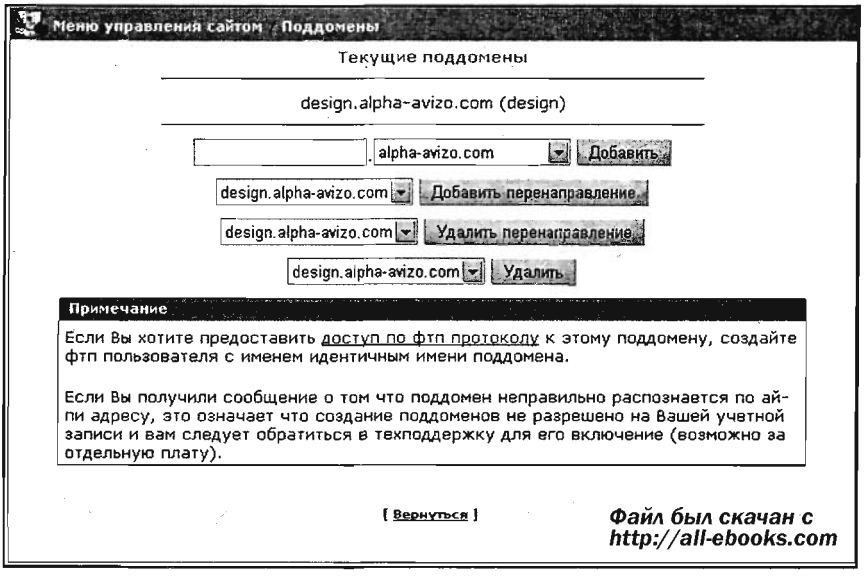



*Рис. 17.25. Поддомены*

## 17.2.5. Поддомены

Предположим, что у вашей фирмы (сайт которой вы разместили по адресу `firma.ru`) есть подразделение программистов, для которого вы хотите создать отдельный поддомен `dev.firma.ru`. Тогда вы заходите в модуль **Поддомены**, вводите имя поддомена «dev» и нажимаете кнопку **Добавить**.

Если вы хотите предоставить FTP-доступ к созданному поддомену, то создайте FTP-пользователя «dev», то есть имя FTP-пользователя должно совпадать с именем поддомена.

### 17.2.6. Базы данных MySQL

*Этот модуль позволяет быстро и удобно создать базу данных* MySQL без запуска интерфейса **phpMyAdmin**. Напротив каждой базы данных есть три кнопки (рис. 17.26):

- **Delete**: удалить базу данных;
- **Check**: проверить базу данных на целостность;
- **Repair:** восстановить базу данных в случае ошибки целостности.

Под каждой базой данных выписаны операторы подключения к этой базе на языках Perl и PHP.

В нижней части окна будет поле для ввода имени новой базы данных. Введите имя и нажмите кнопку **Создать базу данных MySQL**. Новая база данных получит имя, состоящее из введенной вами строки с префиксом – именем вашей учетной записи. Например, если вы вошли в Cpanel под именем «user», а *при создании базы данных ввели имя* «db1», *то вновь созданная* база данных будет называться «user_db1».

Как мы знаем, чтобы подключиться к базе данных, нужно указать имя сервера, имя MySQL-пользователя и его пароль. Список MySQL-пользователей отображается еще ниже (рис.17.27). Там же находится форма для создания нового пользователя.

Но одного указания пользователя и его пароля мало. Нужно, чтобы указанный пользователь имел доступ к выбранной с помощью функции `mysql_select_db()` таблицы. Установить привилегии пользователя можно в этом же окне. Для этого выберите пользователя и выберите базу дан-

Файл был скачан с
http://all-ebooks.com

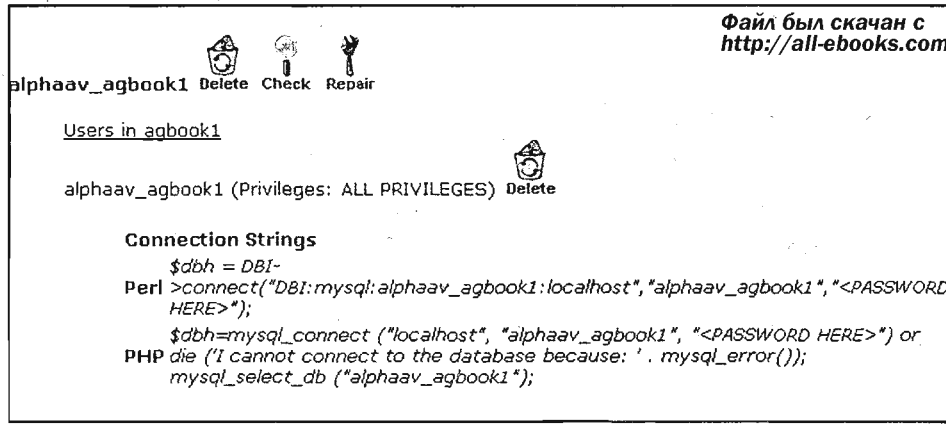



***Рис. 17.26. Список имеющихся баз данных***



http://all-ebooks.com

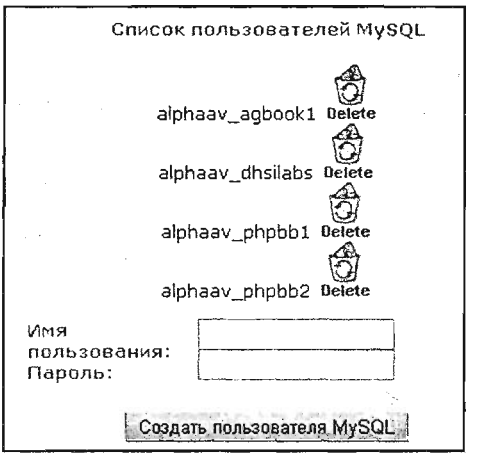



***Рис. 17.27. Список пользователей MySQL***

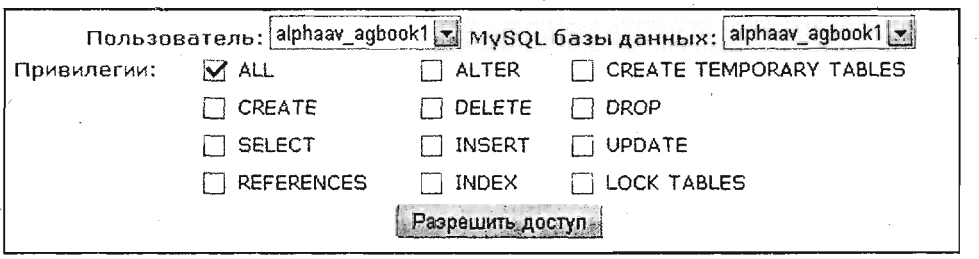



***Рис. 17.28. Привилегии пользователей***

ных MySQL, доступ к которой вы хотите предоставить этому пользователю. Доступ к базе устанавливается с помощью следующих привилегий:

- ALL – любые действия над базой данных;
- CREATE – право создавать таблицы в базе данных;
- SELECT – право использовать оператор SELECT, то есть производить выборку из таблиц базы данных;
- REFERENCES – право использования REFERENCES;
- ALTER – право изменять структуру таблиц;
- DELETE – право удалять записи из таблиц;
- INSERT – право добавлять записи в таблицы;
- INDEX – право индексирования таблиц;
- CREATE TEMPORARY TABLES – право создавать временные таблицы;

http://all-ebooks.com

- DROP – право удалять таблицы;
- UPDATE – право модифицировать записи в таблицах;
- LOCK TABLES – право «запирать» таблицы.

### 17.2.7. Файловый менеджер

На первый взгляд файловый менеджер Cpanel может показаться неудобным (рис.17.29), но к нему очень быстро привыкаешь.

Работать с ним очень просто:

- Для того чтобы перейти в нужный каталог, просто щелкните по ссылке с именем этого каталога.
- Для вызова меню операций с файлами, которое появится в правом верхнем углу окна, щелкните по имени файла. В меню входят следующие команды:
  - **Show file** – показать файл;
  - **Delete file** – удалить файл;

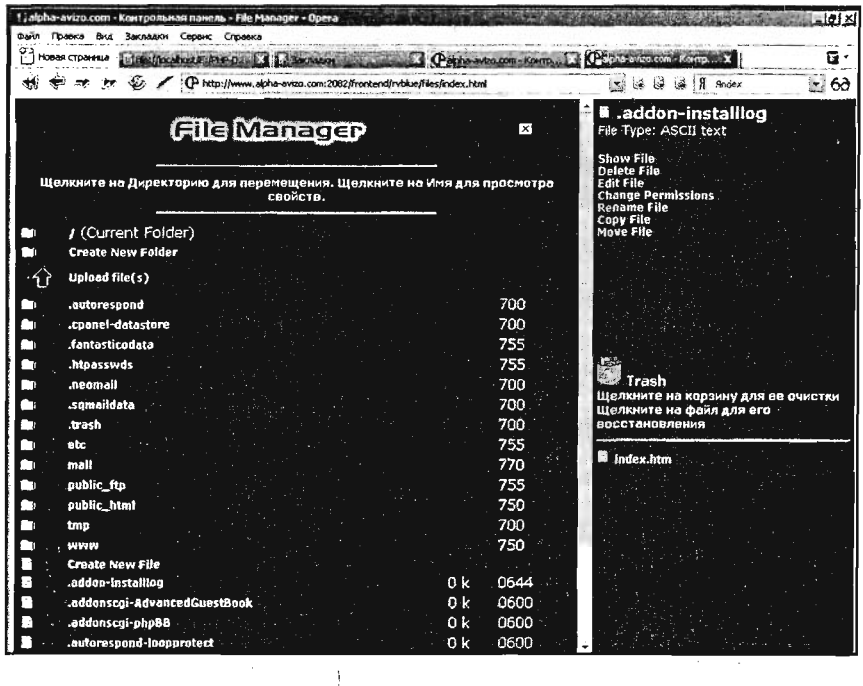



*Рис. 17.29. Файловый менеджер от Cpanel*



http://all-ebooks.com

*Рис. 17.30. Фрагмент окна файлового менеджера*

- **Edit file** – редактировать файл;
- **Change permissions** – изменить права доступа к файлу;
- **Rename file** – переименовать файл;
- **Copy file** – копировать файл;
- **Move file** – переместить файл.

- Для каталогов меню действий будет немного иным:
  - **Delete this folder and all files under it** – удалить каталог и все его содержимое;
  - **Rename this folder** – переименовать каталог;
  - **Change Permissions** – изменить права доступа к каталогу;
  - **Move this folder** – переместить этот каталог;
  - **Copy this folder** – копировать этот каталог.

Для создания нового каталога щелкните по ссылке **Create New Folder,** находящейся вверху списка каталогов. Аналогичная ссылка **Create New File** для создания нового файла находится выше списка файлов. Ссылка **Upload files,** расположенная под ссылкой **Create New Folder** (рис.17.30), позволяет загрузить новые файлы в текущий каталог.

### 17.2.8. Резервное копирование

Модуль резервного копирования с Cpanel тоже лучше, чем в DirectAdmin. Модуль резервного копирования от Cpanel позволяет:

- создать резервную копию всего сайта;
- создать резервную копию файлов, которые изменились за сегодня;
- скачать архив любой созданной вами базы данных;
- скачать архив псевдонимов/фильтров;
- восстановить сайт из резервной копии;

http://all-ebooks.com

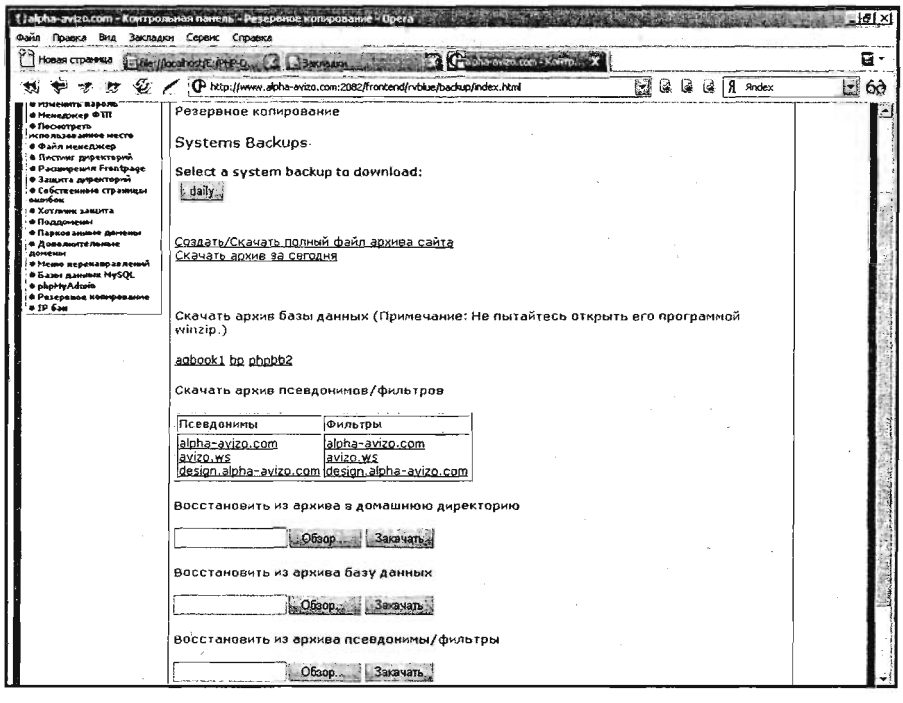



***Рис. 17.31. Модуль резервного копирования***

- восстановить базу данных по копии;
- восстановить псевдонимы и фильтры.

Рекомендую делать полную копию сайта один раз в неделю (при условии, что вы за эту неделю его изменяли), а ежедневное копирование -- в случае изменения файлов сайта.



http://all-ebooks.com

# ГЛАВА 18.

# PHP-NUKE — СИСТЕМА УПРАВЛЕНИЯ СОДЕРЖИМЫМ САЙТА (КОНТЕНТОМ)

http://all-ebooks.com

## 18.1. ВОЗМОЖНОСТИ И ОСОБЕННОСТИ PHP-NUKE

Создание сайта – дело весьма непростое. Смотрите сами: во-первых, нужно написать "движок" – сценарий, который будет выводить нужные страницы вашего сайта. Во-вторых, нужно позаботиться о дизайне. Если с PHP-программированием особых сложностей у вас не будет, то с дизайном настоящая проблема. Не у каждого разработчика получится придумать и реализовать качественный дизайн: для этого нужно не только быть настоящим художником, но и отлично владеть программой вроде Photoshop'а. А что делать, если над душой стоит начальник и говорит, что сайт нужен вчера?

К счастью, решение есть: это системы управления контентом сайта (CMS, Content Management System). Такая система сделает за вас почти все: предоставит «движок», нарисует дизайн, а вам останется только обеспечить контент: статьи, новости, форумы, голосования и прочее. Вместо разработки собственного дизайна вы сможете просто выбрать одну из имеющихся тем оформления или, если стандартные темы вас не устраивают, скачать из Интернета новую. Создать хороший сайт модульной структуры, позволяющей в дальнейшем расширять его возможности, на основе CMS можно всего за день.

В этой главе мы рассмотрим только одну из множества систем CMS: **PHP-Nuke**. Загрузить последнюю версию **PHP-Nuke** можно с официального сайта разработчиков `http://phpnuke.org`.

**PHP-Nuke** — это мощное средство управления контентом (содержимым) веб-сайта. Оно позволяет администратору:



http://all-ebooks.com

- создавать и размещать новости (статьи) и голосования;
- создавать резервную копию базы данных;
- управлять баннерами.
- управлять как контентом, так и внешним видом сайта, включая управление блоками (что это такое, рассмотрим чуть позже);
- создавать членские (закрытые) зоны;
- управлять учетными записями пользователей;
- создавать форумы. В качестве форума по умолчанию **PHP-Nuke** использует движок PHPBB, который мы рассмотрим отдельно в главе 20, поэтому сейчас не будем останавливаться на нем подробно;
- создавать рассылки (newsletters);
- рассылать пользователям сайта сообщения, в том числе по SMS.

Модульная конструкция **PHP-Nuke** позволяет подключать к сайту дополнительные модули. К числу стандартных модулей относятся:

- **Home** — обеспечивает переход на домашнюю страницу;
- **AvantGo** — просмотр статей;
- **Downloads** — обеспечивает загрузку на сайт полезных программ, скриптов, картинок или другой информации;
- **Feedback** — модуль обратной связи с администрацией сайта, позволяющий пользователю отправить сообщение администратору сайта, не зная его точного e-mail адреса;
- **PrivateMessages** — управляет личными сообщениями, то есть сообщениями, адресованными зарегистрированному пользователю сайта;
- **Recommend Us** — модуль рекомендации, позволяющий отправить ссылку на сайт вашему другу;
- **Search** — поиск статей по ключевому слову, теме, автору, по дате и другим критериям;
- **Statistics** — разнообразная статистика: например, какую операционную систему и какой браузер чаще всего используют пользователи сайта;
- **Stories Archive** — поиск статьи в архиве;

http://all-ebooks.com

- **Submit News** — модуль публикации новостей пользователями сайта с разрешения администратора;
- **Top 10** — лучшая десятка (выводятся списки самых читаемых и самых комментируемых статей);
- **Weblinks** — каталог ссылок;
- **Members list** — список пользователей;
- **WebMail** — веб-интерфейс управления почтой домена;
- **Your account** — управление вашей учетной записью.

Как вы видите, список функций и модулей впечатляет. Позже мы остановимся на некоторых модулях подробнее, а сейчас займемся установкой **PHP-Nuke**.

## 18.2. УСТАНОВКА PHP-NUKE

Как я уже сказал, скачать **PHP-Nuke** можно с сайта `http://phpnuke.org`. Он полностью бесплатен, единственное требование разработчика – не удалять баннеры-ссылки на сайт `http://phpnuke.org`.

Описание установки начнем, как и полагается, с системных требований. Для работы PHP-Nuke необходимы:

- правильно настроенный Linux;
- веб-сервер Apache;
- PHP версии не ниже 4.0;
- модуль `mod_php`, добавляющий поддержку PHP в Apache;
- сервер баз данных MySQL.

Все это доступно в любом Linux-дистрибутиве (ну почти в любом) либо на сервере хостинг-провайдера – при условии, что вы выбрали Unix-хостинг. А вот на домашнем компьютере под управлением Windows, к сожалению, установить PHP-Nuke не удастся. Поэтому, если вы хотите поэкспериментировать не в онлайне, а на локальном компьютере, установите на нем Linux.

Распакуйте содержимое архива `PHP-Nuke-X.X.tar.gz` (X.X — это версия **PHP-Nuke**, я использую версию 6.5) в корневой каталог вашего сервера, то есть в `$DocumentRoot`. Хотя сейчас я рекомендую вам распаковать его в каталог `$DocumentRoot/phpnuke` — чтобы ничего не нарушить, а потом вы всегда сможете перенести его в корневой каталог.



http://all-ebooks.com

**Примечание.**

Какую версию устанавливать? Все зависит от вашего хостинга. Если у вас самый дешевый тарифный план, то можно выбрать одну из ранних версий **PHP-Nuke**, например 6 (версии более "древние" выбирать не стоит). Эта версия занимает в архиве 3,7 Мб, а в распакованном виде 13,7 Мб. Последняя на момент написания этих строк версия (7.8) занимает 7,8 Мб в архиве (!), в развернутом виде она будет занимать чуть больше 28 Мб. Если вам выделено всего 30 Мб, то для работы вам уже ничего не останется, а ведь на сайт нужно загружать дополнительные материалы, картинки, файлы и т.д.

Существует два способа установки **PHP-Nuke**: «легкий» и «вам не повезло». Начнем с легкого способа. Просто распакуйте архив в корневой каталог вашего хостинга, затем откройте файл `html/config.php` и укажите в нем параметры доступа к серверу MySQL. Потом запустите браузер и введите в адресную строку:

```
http://<доменное_имя>/html/
```

Если вам повезет, **PHP-Nuke** сам настроит себя. Если же нет, тогда читайте дальше.

Создайте базу данных nuke. Это можно сделать с помощью какого-нибудь средства управления сервером MySQL или, если у вас есть доступ к консоли сервера (хотя бы по ssh), с помощью команды:

```
mysqladmin create nuke
```

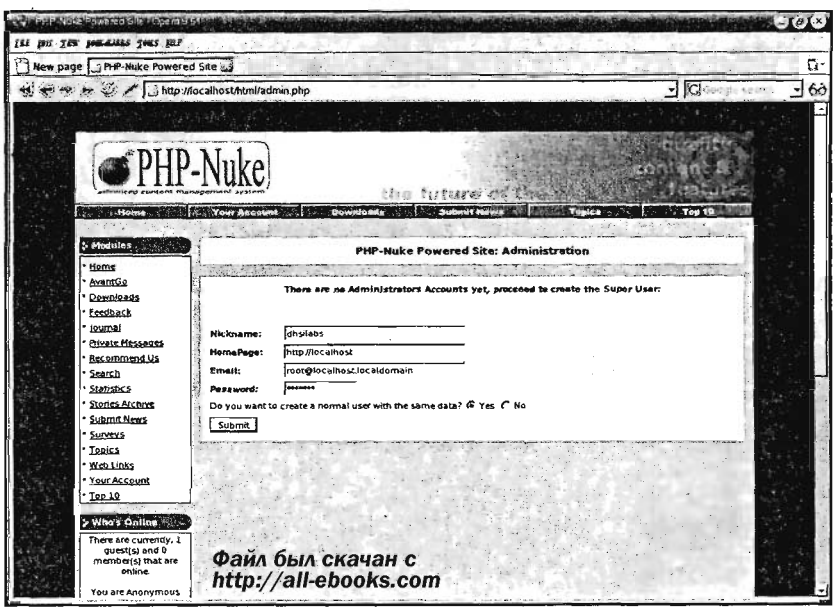



***Рис. 18.1. Создание учетной записи администратора***

Затем нужно создать таблицы. Запросы находятся в файле `sql/nuke.sql`

```
mysql nuke < nuke.sql
```

После этого измените права доступа к файлу `html/ultramode.txt`:

```
chmod 666 ultramode.txt
```

После этого запустите ваш браузер и введите следующий URL:

```
http://localhost/html/admin.php
```

> **Примечание.**
>
> Вместо localhost нужно ввести ваше доменное имя. Поскольку PHP-Nuke установлен на моем домашнем компьютере, то далее (и в иллюстрациях) будет использоваться имя компьютера localhost.

PHP-Nuke предложит вам создать учетную запись администратора (рис. 18.1). Внимание! Если вы увидите форму выбора языка интерфейса, ни в коем случае не изменяйте язык до создания первой учетной записи администратора!

Введите имя пользователя, URL (поскольку мы устанавливаем на домашнюю машину, то подойдет `http://localhost`), e-mail и пароль. Внизу спрашивается, нужно ли создать учетную запись обычного пользователя с таким же именем. Отметьте **Yes** – тогда вы сможете регистрироваться в системе не только как администратор, но и как обычный пользователь.

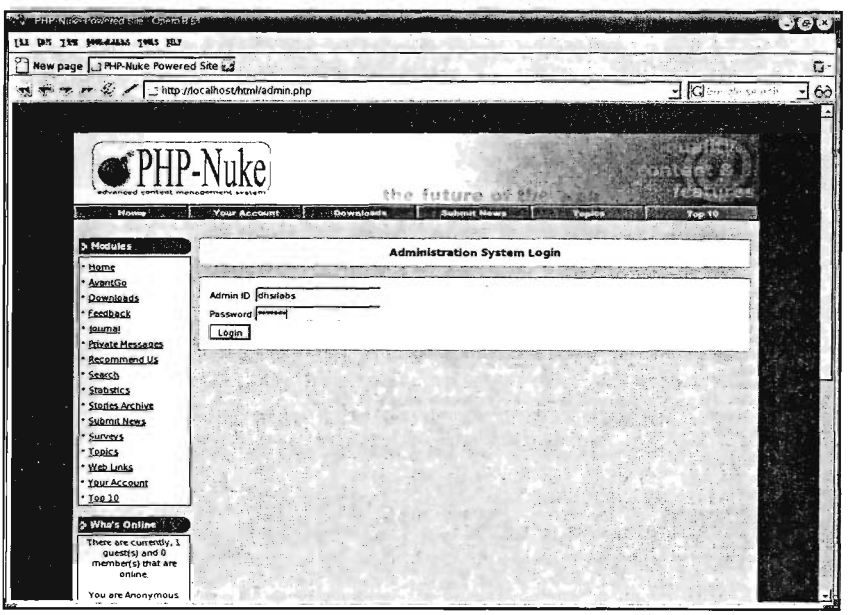



***Рис. 18.2. Регистрация в системе***



http://all-ebooks.com

Когда учетная запись администратора будет создана, **PHP-Nuke** предложит вам зарегистрироваться в системе как администратор (рис. 18.2).

Поздравляю вас с успешной установкой **PHP-Nuke**! В следующем пункте мы рассмотрим управление сайтом с помощью **PHP-Nuke**.

## 18.3. ПЕРВЫЙ ЗАПУСК

После регистрации вы увидите меню администрирования (рис. 18.3), которое позволяет вам:

- добавить новую статью (новость, блок);
- сделать резервную копию базы данных;
- добавить или удалить баннеры и блоки;
- редактировать контент;
- управлять загрузками (модуль **Downloads**);
- управлять учетными записями администраторов и рядовых пользователей;

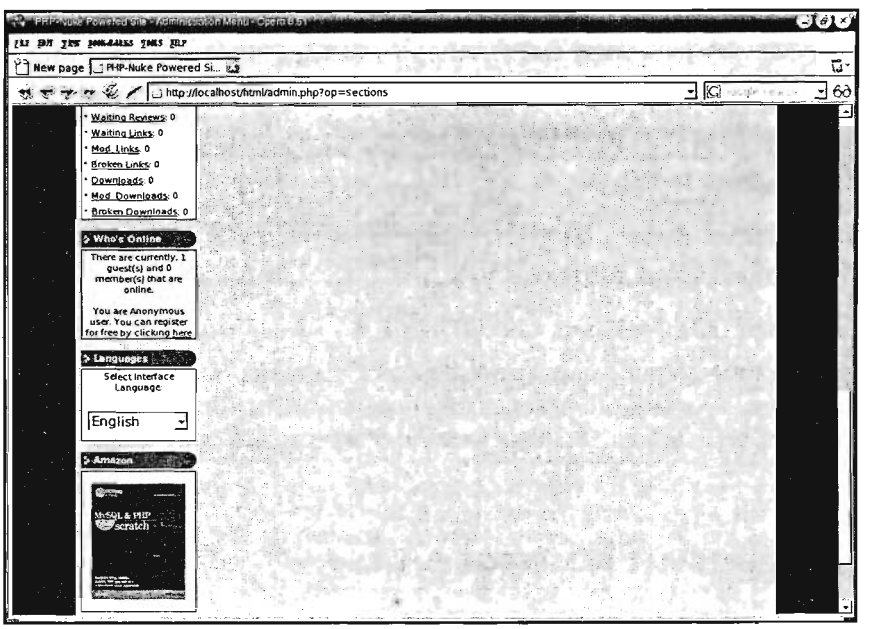



***Рис. 18.3. Меню администрирования прокручено до формы выбора языка***

http://all-ebooks.com

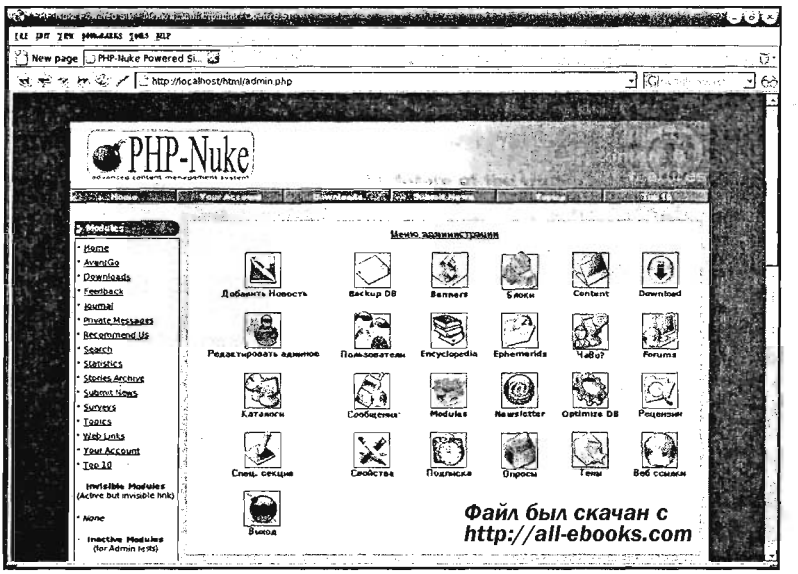



***Рис. 18.4. Панель управления сайтом на русском языке***

- администрировать форум PHPBB;
- управлять каталогами, модулями и сообщениями;
- создавать рассылки (newsletters), редактировать списки подписчиков;
- управлять опросами (голосованиями);
- создавать новые темы;
- включать/выключать, а также редактировать модули;
- управлять каталогом ссылок;
- изменять глобальные свойства сайта.

Первое, что хочется исправить, - это язык интерфейса. Что ни говори, а если предусмотрен русский, пользоваться англоязычным интерфейсом вообще не хочется. Слева на рис.18.3 вы видите форму для выбора языка.

## 18.4. ИЗМЕНЕНИЕ ПАРАМЕТРОВ САЙТА

Для настройки общих параметров сайта предназначен модуль **Свойства.** Нажав кнопку **Свойства** (рис.18.4), не забудьте пролистать окно вниз:



http://all-ebooks.com

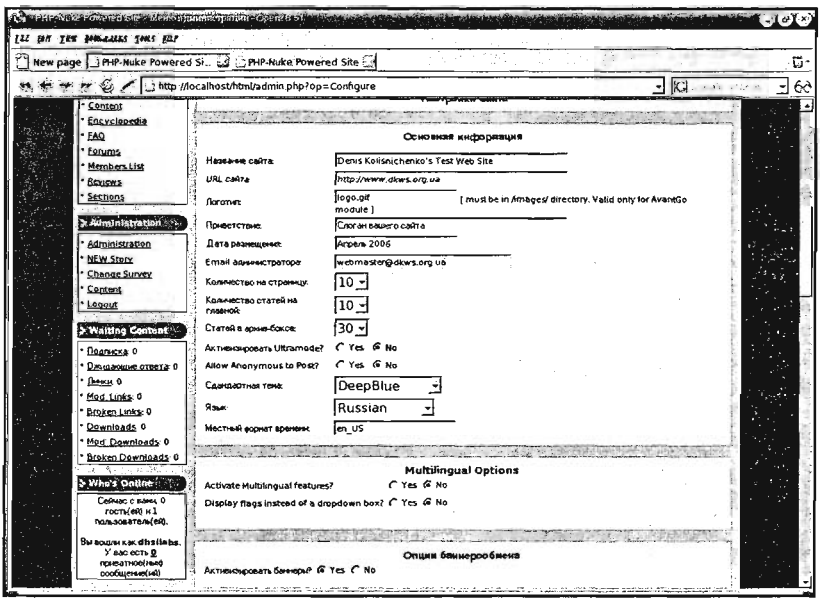



***Рис. 18.5. Свойства сайта***

**PHP-Nuke** отображает загруженный вами модуль под меню панели управления. Рекомендуемые настройки приведены в табл. 18.1.

**Таблица 18.1.** *Настройки сайта*

| Опция | Рекомендуемое значение |
|---|---|
| Название сайта | Казалось бы, все просто - что хочу, то и пишу, хоть "Персональный сайт Васи Пупкина". Но вспомните: "Как вы яхту назовете, так она и поплывет",- поэтому постарайтесь придумать для своего сайта оригинальное название |
| URL сайта | Укажите полный URL вашего сайта |
| Логотип | Тут нужно поработать дизайнеру, чтобы разработать специально для вас хороший логотип. После этого файл логотипа нужно поместить в каталог html/images. А пока пусть будет стандартный logo.html |
| Приветствие | Это слоган вашего сайта. Ну вроде "Крупнейший сайт знакомств" или "Хомосоветикус легких путей не ищет". В общем придумайте что-нибудь – тут можно писать все, что угодно, причем слоган не всегда должен соответствовать тематике сайта |
| Дата размещения | Когда вы разместили сайт? Вот сюда и запишем эту дату |

http://all-ebooks.com

**Таблица 18.1.** *Настройки сайта(продолжение)*

| Опция | Рекомендуемое значение |
|---|---|
| Email администратора | Это ваш e-mail. Укажите реальный адрес, поскольку в случае ошибки или другого сервисного сообщения PHP-Nuke будет уведомлять вас по этому адресу |
| Количество на страницу | Количество статей на странице. Это значение можно увеличить до 15, если статьи небольшие или, наоборот, уменьшить до 5, если статьи большие. Под статьей здесь подразумевается новостное сообщение, а не текст на 10-15 тысяч символов, как мы привыкли считать. Новостной пост, как правило, не должен превышать 3000 символов |
| Количество статей на главной | Желательно не загромождать главную страницу: она и так будет перегружена модулями PHP-Nuke, поэтому данное значение можно смело сократить до 5 |
| Статей в архив-боксе | Вот здесь предлагаемое по умолчанию значение 30 можно не изменять. Предполагается, что в день будет выкладываться один пост (одна новость), поэтому одна страница архива позволяет просмотреть новости сразу за весь месяц |
| Активизировать Ultramode? | Не задумываясь, оставьте все как есть – No |
| Allow Anonymous to Post? | Эта опция означает "Разрешить анонимным пользователям добавлять новости". Конечно же, No, если вы не хотите, чтобы ваш сайт заспамили |
| Стандартная тема | Установите что-то отличное от «DeepBlue», чтобы не сразу догадались, что ваш сайт создан на PHP-Nuke. Но не сейчас, а немного позже – когда вы уже разберетесь со всеми модулями. Если стандартных тем вам будет мало, на сайте http://phpnuke.org вы всегда сможете скачать дополнительные темы |
| Язык | Язык вашего сайта. Не забудьте выбрать русский! |
| Местный формат времени | Оставьте как есть, потом измените |
| Activate Multilingual features? | Активировать многоязычные функции? Нет, этого вам не нужно – зачем усложнять себе жизнь. Ваш сайт на русском, значит, на русском |
| Display flags instead of a drop-down box? | Эта опция нужна только тем, кто ответил Yes на предыдущий вопрос: она предлагает вместо выпадающего списка языков отображать флаги стран. Не стоит надеяться на то, что все ваши пользователи смогут опознать флаг своей страны: ответьте No |
| Активизировать баннеры? | Выключите эту опцию – рекламы хватит и без нее |



http://all-ebooks.com

**Таблица 18.1.** *Настройки сайта (продолжение)*

| Опция | Рекомендуемое значение |
|---|---|
| Сообщения внизу | Эта группа параметров позволяет установить оформление нижней части сайта. Пока ничего не изменяйте – вы ведь еще сайта не видели. А потом, когда разберетесь, где и что находится, тогда и поменяете эти надписи |
| Backend Конфигурация | Оставьте все как есть, ну, разве что можно изменить строку PHP-Nuke Powered Site |
| Сообщать о новых статьях на мыло? | Если включена эта опция, то в случае добавления пользователями новых статей вы будете получать об этом сообщение. Здесь уж решайте сами: при большой активности пользователей вы рискуете получать сотни сообщений в день |
| Email | Если вы ответили Yes на предыдущий вопрос, значит, сейчас вы должны указать адрес, на который будут отправляться уведомления о новых статьях |
| Заголовок | Тема сообщения о новом посте |
| Сообщение | Текст сообщения |
| Откуда письмо | От чьего имени отправлять уведомление |
| Тип модерации | PHP-Nuke предоставляет пользователям возможность комментировать посты. Разумеется, пользователи могут добавлять и некорректные комментарии, поэтому желательно, чтобы кто-то поддерживал порядок: одобрял публикацию приемлемых комментариев и удалял неприемлемые. Такая деятельность называется модерированием, и значение рассматриваемой опции указывает, кто будет модерировать комментарии:<br>• без модератора;<br>• пользователь: проверять комментарии будет пользователь, отправивший пост;<br>• администратор – любой из администраторов.<br>Наиболее оптимальный вариант – «пользователь»: пусть каждый сам следит за своим постом и несет ответственность за то, что в нем и под ним написано |
| Опции комментариев | Эта группа опций позволяет задать максимальный размер комментария (по умолчанию 4096 байтов) и имя, присваиваемое по умолчанию анонимным комментаторам. Если сайт на русском, то имя «Anonymous» желательно заменить на «Гость» |
| Разрешить графику в меню Администратора? | Тут уж сами решайте. С одной стороны, красиво, с другой – будет медленно грузиться, что критично, если у вас доступ по модему |

http://all-ebooks.com

**Таблица 18.1.** *Настройки сайта (продолжение)*

| Опция | Рекомендуемое значение |
|---|---|
| Комментировать Опросы? | Будут ли комментироваться опросы (голосования)? Вообще-то комментарии для голосования необязательны, поэтому можете выключить эту опцию |
| Activate Comments in Articles? | Активировать комментирование статей? Вот это нужная опция, включите ее – пусть каждый из пользователей выскажет все, что он думает по поводу того или иного поста |
| Минимальная длина пароля | Задает минимальную длину пароля пользователя |
| Activate Broadcast Messages | Разрешить широковещательные сообщения? Если такие сообщения разрешены, пользователь сможет отправить сообщение всем пользователям вашего сайта |
| Let users change News number in Home? | Разрешить ли пользователям изменять количество новостей? А почему бы и нет? Пусть себе изменяют. Ведь пользователь изменит только свои параметры – параметры остальных пользователей не будут затронуты |
| Censure Options | Опции цензуры. Лучше выключить, то есть установить режим «No filtering», поскольку эти опции «не дружат» с русским языком |
| Footer Message in all sent e-mails | Подпись в сообщениях, отправляемых от имени вашего сайта |
| Let your users send e-mail? | Разрешить ли пользователям отправлять сообщения электронной почты? Если у вас ограничен трафик, отключите эту функцию, а то они вам понаотправляют столько, что потом за хостинг не расплатитесь |
| Let users send attachments? | Если вы в предыдущем случае ответили Yes, то есть все-таки разрешили отправку сообщений, то хотя бы на этот вопрос ответьте No. Если вы ответите Yes, то этим вы разрешите пользователям к своим файлам прикреплять вложения, что существенно увеличит размер сообщения |
| Attachments temporal directory | Каталог для вложений. Должен быть доступен для записи |
| Let users view/read attachments? | Разрешить пользователям просматривать вложения, не загружая их на свой компьютер? Конечно, No. Представьте, что пользователи начнут обмениваться фотографиями, просматривая их средствами вашего сайта за счет вашего трафика |
| Received attachments temporal directory | Временный каталог для полученных вложений. Должен быть доступен для записи |
| Max. number of accounts | Максимальное количество учетных записей (-1 – без ограничений) |



http://all-ebooks.com

**Таблица 18.1.** *Настройки сайта (продолжение)*

| Опция | Рекомендуемое значение |
|---|---|
| Is this service based in a single account? | Этот сервис основан на одной учетной записи? |
| Default Single Account Name | Имя учетной записи (для POP/SMTP) по умолчанию |
| Default POP3 Mail Server | POP3-сервер по умолчанию |
| Filter header when forward | Перед перенаправлением письма отфильтровать заголовки. Полезная опция, иногда помогающая избавиться от спама |

## 18.5. СОЗДАНИЕ НЕБОЛЬШОГО САЙТА

Для создания сайта с помощью **PHP-Nuke** вам нужно выполнить следующие действия:

- установить параметры сайта;
- выбрать дизайн;
- выбрать блоки (модули) сайта и их расположение на странице.

### 18.5.1. Работа с блоками

Первое мы уже сделали, второе – всегда успеем, поскольку для этого вам достаточно при настройке сайта просто выбрать другую тему оформления. Самое главное – это настройка блоков сайта.

Зайдите в модуль **Блоки** панели администрирования **PHP-Nuke**.

Пока ничего не трогайте, а просто просмотрите ваш сайт, как он есть сейчас, чтобы понять, для чего служит тот или иной модуль. Для этого запустите браузер и введите URL

`http://localhost/html`. Вы увидите свой сайт таким, каким **PHP-Nuke** будет отображать его вашим будущим пользователям.

Не нравится? Конечно: во-первых, названия блоков на английском, а во-вторых, некоторые блоки вообще не нужны. Например, зачем вам блок с товарами от интернет-магазина Amazon, если вы не собираетесь получать с него деньги за показ его рекламы?

Теперь вернемся к окну, в котором у нас открыта панель администрирования сайта. Как управлять блоками? Для каждого блока выводится:

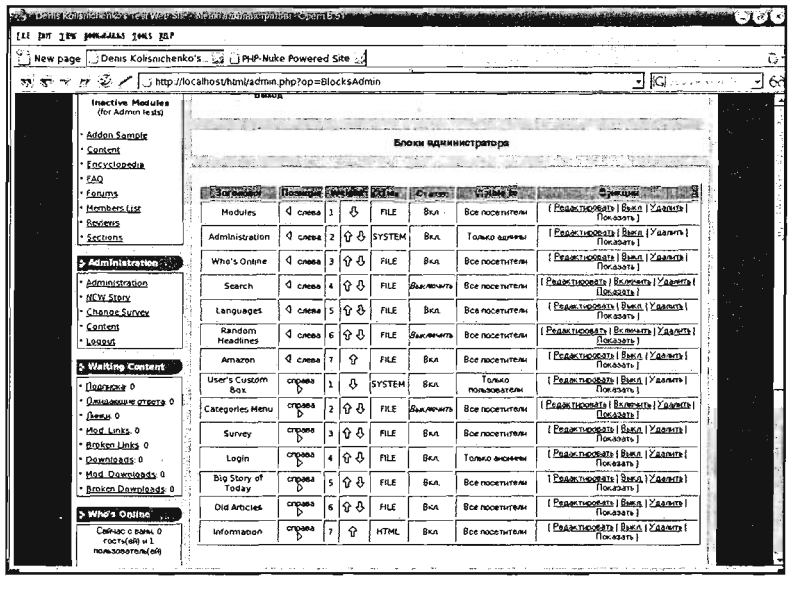



***Рис. 18.6. Список блоков сайта***

- его заголовок;
- позиция: в каком месте страницы должен быть помещен этот блок;
- ширина: сколько места занимает блок. Этот параметр блока изменению не подлежит, но вы должны учитывать его при выборе места для блока;
- стрелки вверх/вниз: позволяют изменить позицию блока относительно других блоков;
- тип: неизменяемый параметр, обычно указывающий, откуда берутся данные для заполнения блока информацией;
- статус: включен или выключен блок. Если блок выключен, он не будет отображаться на странице;
- visible to: кто будет видеть этот блок? Доступны следующие значения:
    - все посетители;
    - только пользователи;
    - только админы;
    - только анонимы;



http://all-ebooks.com

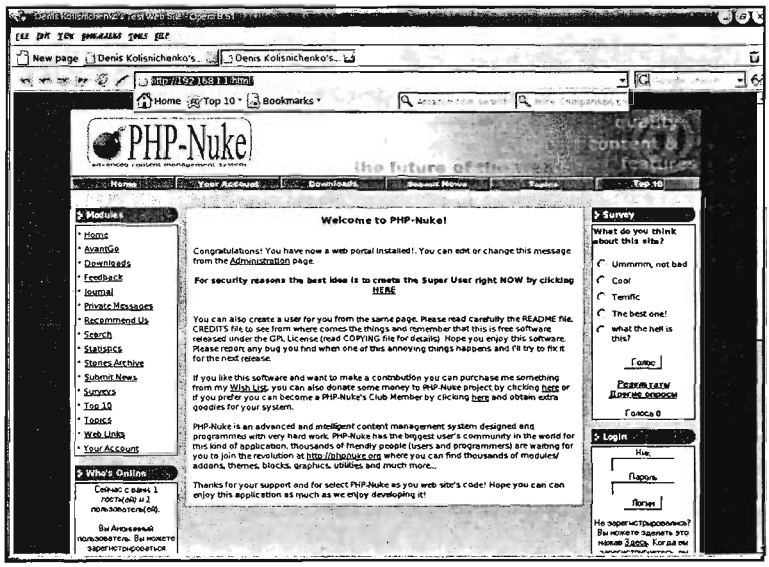



***Рис. 18.7. Так блоки образуют внешний вид сайта***

Возле каждого блока есть меню действий:

- **Редактировать** – используется для редактирования блока, например для изменения заголовка, позиции и других параметров;
- **Включить/выключить** – если блок выключен, он не будет отображаться на странице;
- **Удалить** – вообще удалить блок из списка блоков. Если он вам понадобится, то добавить его вы сможете с помощью формы ниже;
- **Показать** – показывает, как будет выглядеть блок.

На русскоязычных сайтах уже сложилась традиция именования блоков. Советую первым делом переименовать блоки согласно этой традиции (табл. 18.2), чтобы ваши пользователи не заблудились.

**Таблица 18.2.** *Рекомендуемые заголовки блоков*

| Блок | Русский заголовок | Комментарии |
|---|---|---|
| Modules | Меню | Можно назвать также «Модули» |
| Administration | Администрирование | Если вы – единственный администратор, то этот блок можно не переименовывать: все равно его никто больше не увидит |

**Таблица 18.2.** *Рекомендуемые заголовки блоков (продолжение)*

| Блок | Русский заголовок | Комментарии |
|---|---|---|
| Who's Online | На сайте | Этот модуль показывает зарегистрированных пользователей сайта, которые сейчас в онлайне |
| Search | Поиск | Блок поиска по сайту |
| Languages | Язык | Если вы не собираетесь разрешать пользователям менять язык интерфейса, выключите этот блок |
| Random Headlines | - | Выключите этот блок. Блок отображает случайные заголовки постов |
| Amazon | - | Выключите этот блок. Он отображает товары магазина Amazon |
| User's Custom Box | - | Это пользовательский блок: вы сами определяете, какие данные будете в нем отображать (точнее, каждый пользователь определяет это для себя). Пока просто выключите его |
| Categories Menu | - | Показывает меню категорий. Пока выключите его, чтобы не захламлять главную страницу |
| Survey | Опрос | Это блок голосования. О создании опросов мы поговорим позже |
| Login | Вход | Блок регистрации на сайте |
| Big Story of Today | Пост дня | Выводит самый популярный пост |
| Old Articles | Архив | Выводит заголовки старых статей |
| Information | Инфо | Информационный блок. Здесь вы можете добавить свой HTML-код, например кнопки каталогов, счетчики и т.п. |

Когда проделаете все это, сохраните изменения и снова просмотрите сайт:

Так лучше? Конечно, некоторые надписи на неродном языке все еще режут глаз. Поэтому давайте продолжим создание сайта дальше. Начнем с блока Modules – просто отключите его, а на его место поместите блок Вход.

Сначала выберите для блока **Вход** позицию "слева". В результате он окажется расположен под блоком **На сайте**. Чтобы передвинуть **Вход** выше блока **На сайте**, нажмите стрелку вверх напротив имени блока **Вход**.



http://all-ebooks.com

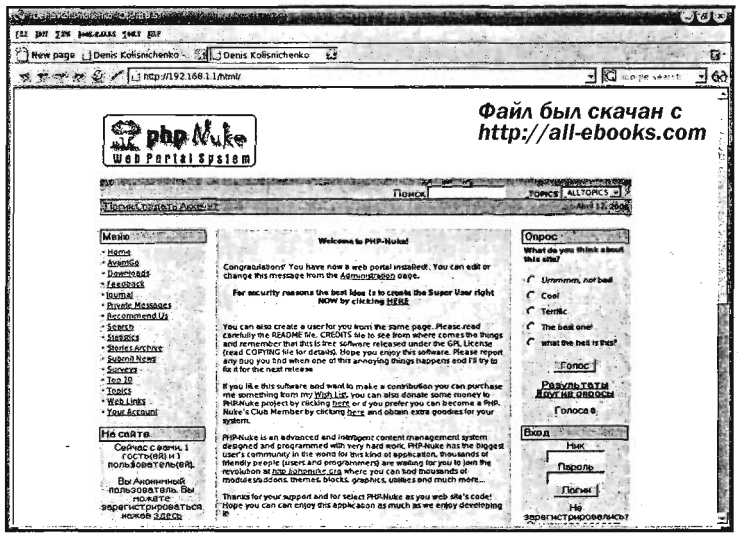



*Рис. 18.8. Модифицированный сайт (выбрана тема оформления «Karate»)*

Теперь, если посмотреть на сайт, вроде бы все нормально, осталось изменить текст сообщения «Welcome to PHP-Nuke» на главной странице (рис.18.8) и придумать свои опросы. Это не проблема. Войдите в модуль **Сообщения**.

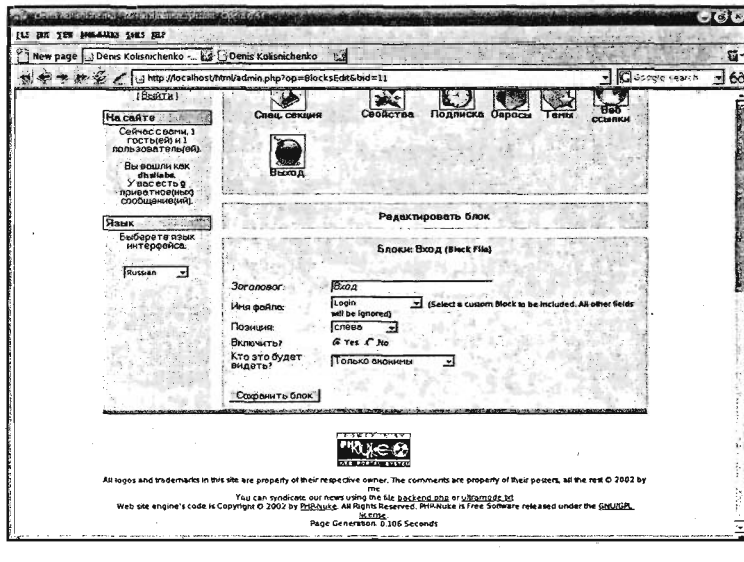



*Рис. 18.9. Редактирование блока*

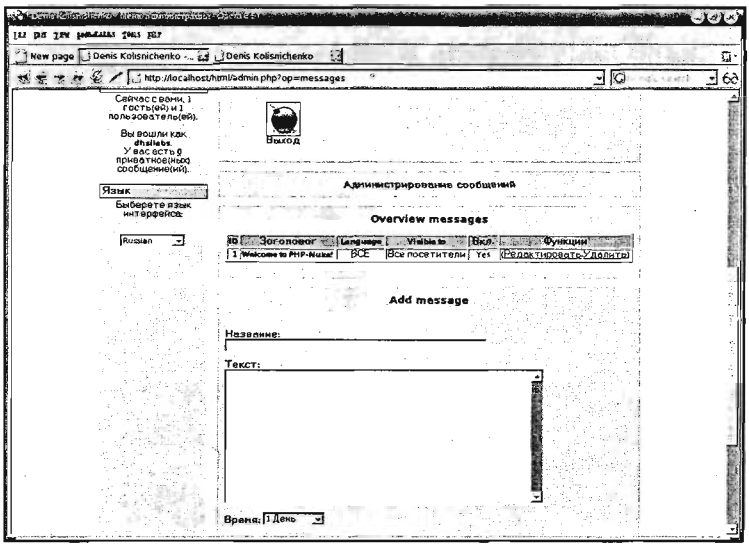



***Рис. 18.10. Редактирование сообщений***

Удалите сообщение «Welcome to PHP-Nuke» и введите новое сообщение в форму Add a message, расположенную под списком сообщений. При добавлении сообщения обратите внимание на следующие параметры:

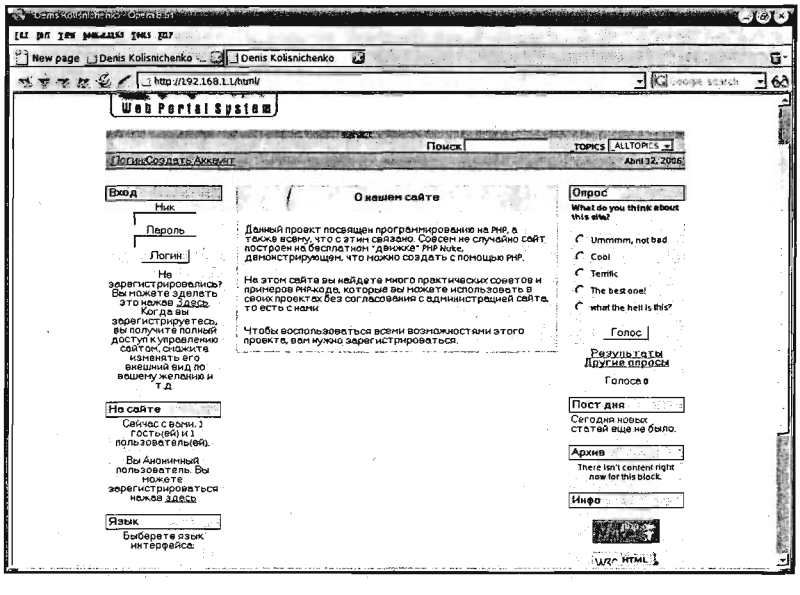



***Рис. 18.11. Еще чуть-чуть***



http://all-ebooks.com

- Время: сколько времени, считая с момента добавления, сообщение будет отображаться на сайте. Если выбрать значение «UNLIMITED», то сообщение будет отображаться до тех пор, пока вы не удалите его вручную.
- Active: если сообщение активно, значит, оно будет показано.
- Кто это будет видеть: определяет аудиторию, которой адресовано данное сообщение.

Как вы видите, **PHP-Nuke** – очень гибкий "движок", позволяющий определять разный контент (разные материалы) для разных целевых аудиторий.

Посмотрим, что вышло после добавления сообщения на русском языке (рис.18.11). Осталось всего ничего – добавить пару постов и опрос на русском языке.

## 18.5.2. Создание опросов (голосований)

Войдите в модуль **Опросы** и щелкните по ссылке **Удалить опрос**. Имеющийся опрос будет удален. Теперь снова войдите в модуль **Опросы** и заполните форму **Создать новый опрос** (рис.18.12). Как видите, все очень просто.

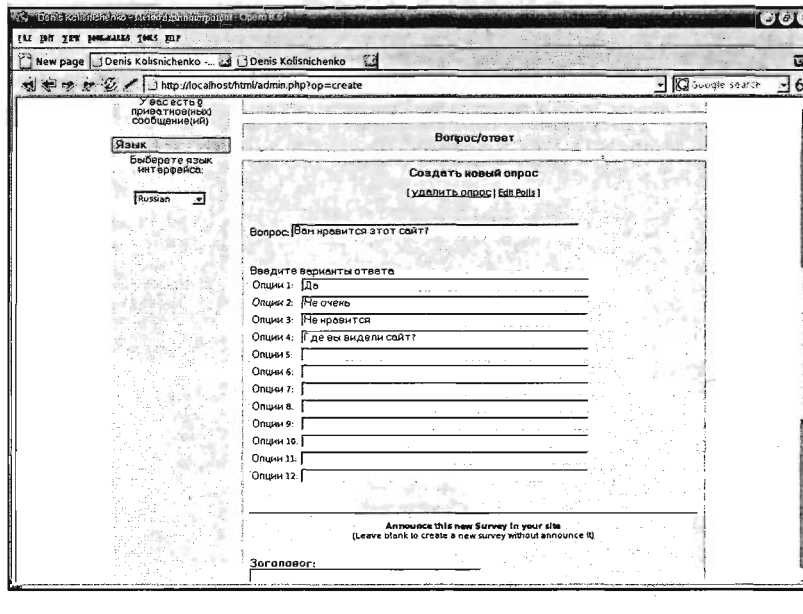



*Рис. 18.12. Создание нового опроса*

### 18.5.3. Добавление новостей

Зайдите в модуль **Добавить новость**. Введите заголовок новости, текст, расширенный текст новости и нажмите кнопку **Ok**. Обратите внимание на выпадающий список режимов возле кнопки **Ok**. Вы можете сразу поместить статью или выбрать режим предварительного просмотра. Рекомендую сначала просмотреть, как статья будет отображаться, а потом уже размещать ее: ведь если статья будет плохо выглядеть в окне браузера, вам придется удалить ее и начать все сначала.

Ниже формы ввода статьи находится форма, позволяющая добавить к вашей статье опрос. Видели на других сайтах голосования «Оцените эту статью» с набором оценок «2, 3, 4, 5»? **PHP-Nuke** позволяет организовать такое же для вашей статьи.

Теперь давайте просмотрим получившийся сайт (рис. 18.14). Можно было бы сказать, что все готово, если бы не это предупреждение:

```
Warning: setlocale(): Passing locale category name as string is
deprecated. Use the LC_* -constants instead. in /var/www/html/html/
mainfile.php on line 565
```

Если вы не хотите копать глубоко, а хотите просто от него избавиться, то откройте `mailfile.php` и перейдите к строке 565. Сейчас она выглядит так:

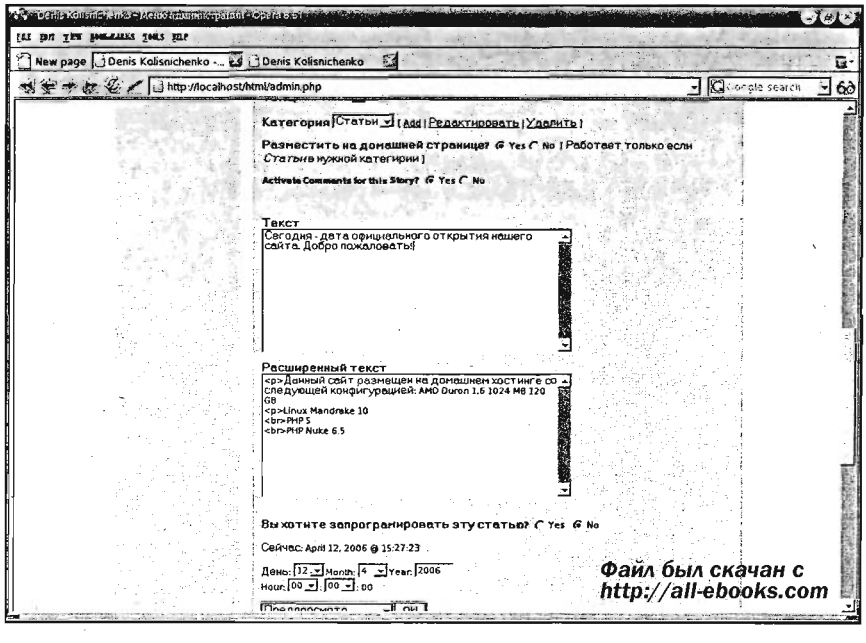



*Рис. 18.13. Добавление статьи*



http://all-ebooks.com

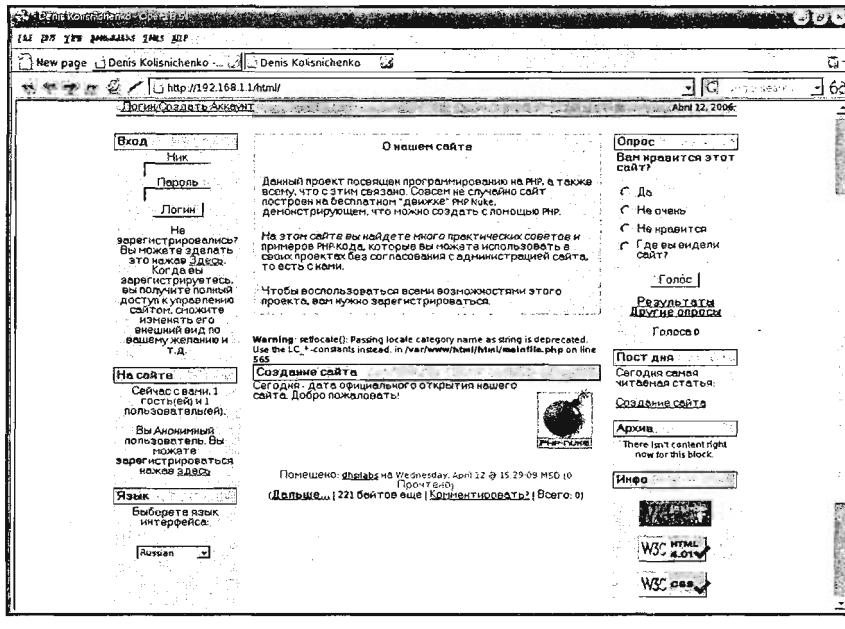



***Рис. 18.14. Сайт почти готов***

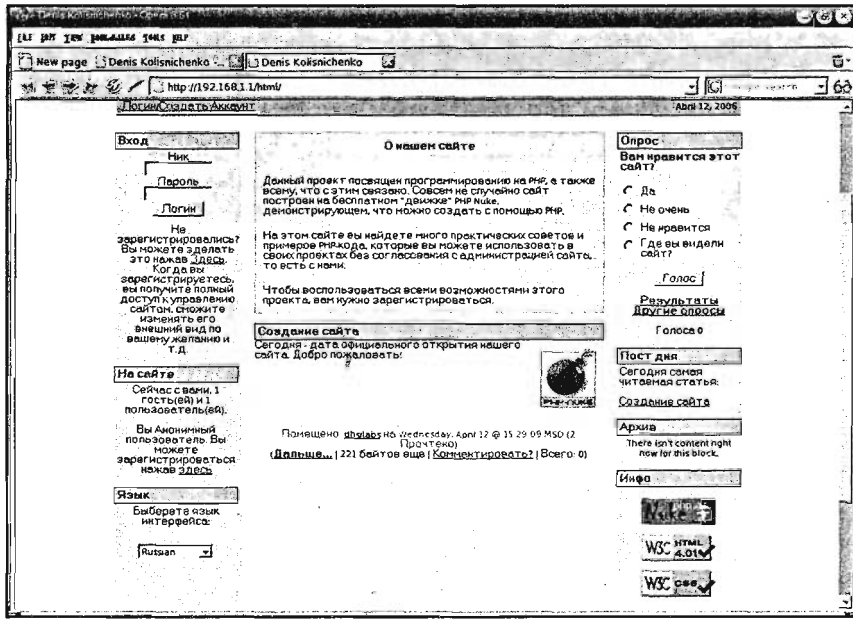



***Рис. 18.15. Сайт готов!***

```
setlocale ("LC_TIME", "$locale");
```

Перед вызовом функции добавьте @:

```
@setlocale ("LC_TIME", "$locale");
```

Вот теперь все нормально.

## 18.6. ЧТО ЕЩЕ МОЖЕТ PHP-NUKE

Мы рассмотрели далеко не все возможности **PHP-Nuke**. В табл. 18.3 кратко описаны не менее полезные модули, с которыми вы разберетесь сами. Надеюсь, что приведенных в ней сведений хватит для того, чтобы вы могли комфортно работать с **PHP-Nuke**.

**Таблица 18.3.** *Некоторые полезные модули PHP-Nuke*

| Модуль | Назначение |
|---|---|
| Backup DB | Создание резервной копии базы данных. Рекомендуется делать резервную копию раз в неделю (хотя бы не реже раза в месяц) |
| Banners | Управление баннерами |
| Content | Управление контентом. С помощью этого модуля вы можете изменять контент сайта, например добавить новую страницу |
| Download | Управляет категориями загрузок. Например, вы можете создать галерею фотографий или скриншотов, чтобы каждый пользователь мог закачать свою фотографию или понравившийся скриншот |
| Редактировать админов | Модуль управления учетными записями администраторов |
| Пользователи | Управление учетными записями пользователей |
| Forums | Управляет форумами. Форумы строятся на базе собственного форума pnPHPbb, который несколько отличается от популярного форума phpbb, но в тоже время очень похож на него. Форумы мы рассмотрим в главе 20 на примере phpbb |
| Modules | Управляет модулями сайта |
| Newsletter | Позволяет отправить письмо с новостями пользователям сайта: либо всем зарегистрированным пользователям, либо только тем, кто подписан на получение новостей. Отправлять всем пользователям рекомендуется только в том случае, если вы хотите сообщить что-то важное |
| Optimize DB | Модуль оптимизации базы данных |



http://all-ebooks.com

**Таблица 18.3.** *Некоторые полезные модули PHP-Nuke(продолжение)*

| Модуль | Назначение |
|---|---|
| Рецензии | Размещение рецензий |
| *Подписка* | *Управление подпиской* |
| Темы | Управление темами |
| *Веб ссылки* | *Позволяет создать собственный каталог ссылок* |

## 18.7. ДРУГИЕ CMS

Как я уже сказал, кроме PHP-Nuke, существует множество других систем управления контентом. В этом параграфе мы познакомимся с самыми популярными CMS.

**Mambo (http://mamboserver.ru)**

Очень солидная система, обладающая огромным числом модулей (на момент написания этих строк количество модулей перевалило за сотню). В Интернете можно найти огромное количество ресурсов по этой системе (темы, скины и т.д.), созданных пользователями-энтузиастами. Вот некоторые особенности системы: персонализация (каждый сам выбирает настройки сайта, как в PHP-Nuke), секции новостей (новости могут добавлять пользователи, а не только администратор), продукции, блочный дизайн, библиотека изображений.

Дизайнеры и разработчики могут использовать плагины для Dreamweaver и разнообразные WYSIWYG-редакторы.

Система очень хорошо документирована и проста в работе, поэтому с ней может разобраться даже новичок, не говоря уже о более опытном пользователе.

Оценка: 5

**Drupal (http://www.drupal.org)**

Хорошая система, обладающая полностью модульной структурой: в отличие от других систем, состоящих из ядра, к которому «прикручиваются» модули, здесь есть только модули.

Система занимает всего 1,7 Мб дискового пространства – не сравнить с PHP-Nuke или Mambo. Но это минимальный комплект. В Интернете есть

огромное количество различных модулей, есть отличная документация для разработчиков и пользователей. Все это делает систему весьма привлекательной.

Оценка: 4

**PhpBlaster (http://phpblaster.org)**

Что умеет PhpBlaster? Новости, опросы, архив файлов, каталог ссылок, форум minibb (урезанная версия phpbb), галереи, система личных сообщений – все это поддерживается. Система поддерживает несколько языков с возможностью динамического переключения языка интерфейса, как в PHP-Nuke. Темы оформления тоже можно менять «на лету».

Очень проста панель управления: обычная текстовая панель, от которой веет духом японского минимализма.

Места на диске занимает 2 Мб – согласитесь, это вам не 13,7 Мб, как в PHP-Nuke. В общем, можно попробовать, вдруг понравится.

Оценка: 4

**Envolution (http://www.envolution.com)**

Одна из самых удачных CMS. Популярность ее объясняется не наличием каких-то уникальных модулей, а грамотной реализацией модулей самых обычных. После установки система занимает 14 Мб на диске, а в базе данных MySQL будет 85 таблиц – вы поймете, почему так много, когда установите систему. Все тщательно продумано, "вылизано", очень удобно организована навигация. Русификация тоже выполнена грамотно: вам не попадется надпись "Зоголовог", как в PHP-Nuke. Envolution оставляет самое приятное впечатление.

Оценка: 5

**MDPro (http://www.maxdev.com)**

Проект довольно тяжеловат (занимает более 11 Мб), зато он обладает удачной панелью управления и очень хорошо русифицирован. В эти 11 Мб входит огромное количество модулей, из которых вам останется только выбрать нужные. Также имеется резервное копирование базы данных и ее оптимизация, что немаловажно. Если я не ставлю этому проекту пятерку, то только потому, что в нем нет ничего оригинального – вполне стандартная CMS.

Оценка: 4+

**Guppy (http://www.freeguppy.org)**



http://all-ebooks.com

Честно говоря, мне эта система не понравилась. Во-первых, она очень криво русифицирована. Хорошо Guppy "говорит" только на своем родном испанском. Во-вторых, во время загрузки отображается сообщение о том, что система загружается, а это мода прошлого века.

С другой стороны, эта система может заинтересовать пользователей, у которых нет возможности использовать MySQL (некоторые хостеры предоставляют доступ к MySQL только по дорогим тарифным планам). В целом же я бы посоветовал рациональнее использовать 1,9 Мб на диске, которые занимает Guppy: поставьте вместо нее тот же Drupal.

Оценка: 3

**vbPortal (http://phpportals.com)**

Портал, полностью интегрированный с vBulletin. Производит самое приятное впечатление, как и сам vBulletin. Позволяет очень легко, а главное, быстро, создать качественный портал.

Я нашел в сети очень интересный форум, посвященный vBulletin: `http://www.forumok.info/forumdisplay.php?f=65`. На этом форуме вы не только больше узнаете о vBulletin, но и скачаете всевозможные хаки и моды для этого движка.

Оценка: 5

**iPHPortal (http://phportal.informika.ru)**

Это CMS отечественной разработки. Ее можно назвать системой управления контентом, но точнее будет описать ее как среду разработки сайта. Программный комплекс iPHPortal обладает очень удобной системой построения сайта, огромным количеством модулей. Хорошо продумана система шаблонов: вы можете создать множество шаблонов для одной и той же страницы. Вы можете даже модифицировать структуру панели управления, изменив ее до неузнаваемости.

Система управления пользователями вообще заслуживает отдельного разговора. Она дает администратору возможность ограничить доступ других администраторов к разным разделам панели управления, сайта, позволяет организовать категории и группы доступа.

Ясно, что с русификацией у этой системы проблем не будет, да и с документацией тоже - все на русском.

Казалось бы, если все так хорошо, то почему многие пользователи до сих пор используют старый добрый PHP-Nuke, а не iPHPortal? К сожалению, в этой бочке меда есть пара ложек дегтя. Первая ложка – это довольно страш-

http://all-ebooks.com

ненький интерфейс, вторая – недружелюбие к новичкам: это все-таки скорее среда разработки, чем настоящая CMS.

Оценка: 4-

**dZeeemp (http://www.dzeeemp.org)**

Еще одна отечественная разработка. Недостаток ее в том, что у нее неблочная система управления сайтом, то есть вы не можете определить блоки и их расположение на странице, как в PHP-Nuke. Зато у этой системы есть гибкая система шаблонов, она поддерживает структуру с неограниченной вложенностью. Система управления пользователями тоже очень удобна.

Нужно отметить, что система относительно молодая, поэтому есть надежда, что она станет еще лучше. Хотя и в таком варианте она очень даже ничего. Естественно, интерфейс dZeeemp и вся документация на русском.

Оценка: 4

**BlindHomepage (http://www.bhp.x-sub.de)**

Это практически полный аналог PHP-Nuke, но у него есть одно отличие: он не использует MySQL, что позволяет применять его на "дешевых" хостингах. Эта система приятнее в использовании, чем Guppy.

Оценка: 4

**Sitellite (http://sitellite.org)**

Обладает огромным количеством функций, модулей, скинов, но за все это нужно платить: после установки система занимает более 50 Мб. Подойдет для создания корпоративных ресурсов внутреннего назначения. Почему именно корпоративных? Да потому что сайт на таком "тяжелом" движке будет хорошо работать в локальной сети, но не в Интернете, где все еще многие пользователи работают по модему.

Оценка: 5

**PHPC (http://www.phpc.ru)**

PHPC (сокращение от PHP Compile) представляет собой систему для разработки веб-сайтов. Как следует из названия, PHPC написана на PHP.

Эта система предлагает абсолютно новый, не похожий на другие системы, подход к созданию сайта. Она позволяет с нуля создать сайт с оригинальным дизайном, написанный на PHP. Звучит, конечно, не слишком правдоподобно, но вы сами сможете в этом убедиться, установив PHPC. Понятно, что для эффективной работы с этой системой желательно знать PHP и разбираться в MySQL.



http://all-ebooks.com

Вот некоторые особенности PHPC:

- разделение дизайна и PHP-кода;
- поддержка шаблонов;
- поддержка модулей расширения системы;
- поддержка стилей (позволяет быстро изменить дизайн сайта);
- поддержка нескольких языков;
- удобная панель администрирования и многое другое.

Оценка: 5

**Xoops2 (http://xoops2.ru)**

Эту систему можно использовать для создания самых разнообразных сайтов: она достаточно компактна, чтобы на ней "построить" домашнюю страничку, и в то же время достаточно масштабируема для большого ресурса. На ее основе можно организовать и каталог, и интернет-магазин.

Система уже не молодая, поэтому для нее написано очень много модулей, скинов, а также языковых пакетов. В системе есть встроенный поиск и поддержка RSS.

Оценка: 4+

**E-Xoops (http://e-xoops.ru)**

История этой системы такова. Была система Xoops v1. Потом она переросла в Xoops v2, а разработчики будущей E-Xoops взяли старую систему Xoops v1, немного модифицировали ее и назвали E-Xoops. По функциям система E-Xoops полностью схожа с Xoops v1. Предпочтительнее использовать Xoops2.

Оценка: 4-

**Phase CMS (http://phasecms.org/)**

Очень простенькая системка, подходящая для очень простеньких сайтов. Подойдет новичкам, ведь она не перегружена функциями.

Оценка: 4

**Subdreamer (http://www.subdreamer.com)**

Еще одна простенькая система, интерес к которой существенно снижает тот факт, что она коммерческая. Если вы готовы заплатить, то система позволит вам построить неплохой сайт. Она обладает понятной панелью управления

http://all-ebooks.com

и стандартным набором функций. Я поставил бы ей четверку с плюсом, но за платность сниму один балл.

Оценка: 3+

**iPROSITE (http://www.iprosite.com/)**

Как и в предыдущем случае, система коммерческая, обладающая стандартным набором функций: менеджер файлов, органайзер, поддержка модулей, WYSIWYG-редактирование и т.д. Поскольку система ничем особенным не отличается от остальных, но является коммерческой, то оценивается она ниже, чем бесплатная система с такими же функциями.

Оценка: 3

**Storyteller (http://www.esselbach.com)**

Вот этот коммерческий проект действительно достоин уважения. Storyteller рассчитан на большие проекты, а именно:

- поддерживает до 99 сайтов;
- обладает раздельным дизайном;
- позволяет экспортировать/импортировать любые данные в формате XML;
- обладает кэшем, что позволяет его использовать на сайтах с большой нагрузкой;
- может интегрироваться с огромным количеством форумов.

Ради справедливости нужно отметить, что есть и бесплатная версия, обладающая некоторыми ограничениями.

Оценка: 5

**TikiWiki (http://tikiwiki.ru)**

Данная система подойдет для создания веб-приложений, сайтов и порталов.

TikiWiki – это не просто CMS, а еще и среда для совместной разработки веб-приложений. Система обладает множеством встроенных параметров, которые могут подключаться/отключаться по мере необходимости. Среди особенностей TikiWiki можно отметить, что система ориентирована на расширяемость и интернациональность, о чем говорит качественно реализованная многоязыковая поддержка.

Можно сказать, что в TikiWiki есть все функции, которыми должны обладать системы управления контентом, а также свои собственные функции



http://all-ebooks.com

(например, функции, ориентированные на групповое создание сайтов). Масштабируемость системы позволяет использовать ее для создания как простых сайтов, так и больших порталов. Те, кому CMS как таковая не нужна, могут использовать TikiWiki как форум, чат или механизм проведения опросов.

Оценка: 5

**pMachine (http://www.pmachine.com)**

Эта бесплатная англоязычная CMS поставляется в двух версиях: Lite и Pro. Во второй добавлен форум, модуль рассылок, поддержка RSS и другое. В Интернете можно скачать модуль русификации.

Оценка: 3

# ГЛАВА 19.

# СОЗДАЕМ НА САЙТЕ ПОЧТОВУЮ РАССЫЛКУ

http://all-ebooks.com

## 19.1. ИСПОЛЬЗОВАНИЕ ФУНКЦИИ MAIL() ДЛЯ ОТПРАВКИ СООБЩЕНИЙ

В четырнадцатой главе мы рассмотрели, как отправлять электронные сообщения через сокеты. На примере мы убедились, что отправлять сообщения таким способом не очень удобно, потому что нужно знать особенности протокола SMTP.

В этой главе мы рассмотрим функцию `mail()`, позволяющую быстро отправить сообщение. Формат этой функции таков:

```
mail(string $email, string $subject, $string $message [,string $headers]);
```

Предположим, что нам нужно отправить письмо по адресу «ivan@ivanov.org», тема сообщения «Привет» и само сообщение — «Как дела?» Воспользуемся функцией `mail()`:

```
mail("ivan@ivanov.org", "Привет", "Как дела?");
```

Если ваша система правильно настроена, сообщение будет отправлено.

> **Примечание.**
>
> Для правильной настройки системы нужно отредактировать секцию `mail function` файла конфигурации `php.ini`. В нем нужно указать имя SMTP-сервера, если сервер работает под Windows, или путь к программе sendmail, если на сервере установлена ОС Linux.

В функции `mail()`, как вы уже поняли, первым аргументом задается адрес электронной почты получателя, а следующими двумя — само сообщение. Теперь разберемся с необязательным параметром `$headers`, позволяющим

задать различные служебные заголовки письма. Все заголовки знать вам не нужно, вот самые необходимые:

- **From** — содержит имя и адрес отправителя;
- **Reply-To** — задает адрес для ответа;
- **X-Mailer** — позволяет указать название почтовой программы;
- **Content-type** — определяет тип содержимого;
- **Content-Transfer-Encoding** — сколько битов кодируют один символ. Если вы планируете отправлять сообщения на русском языке, установите значение 8bit.

Вот пример отправки письма с указанием заголовков:

```
mail("ivan@ivanov.org","Привет","Как дела","From: sidorov@ukr.net\n",
    "Reply-to: root@firma.net\n",
    "Content-Type: text/plain; charset=\"koi8-r\"\n",
    "Content-Transfer-Encoding: 8bit");
```

Обратите внимание, что мы с помощью заголовков Content-type и Content-Transfer-Encoding установили кодировку KOI8-R, а также 8-битное кодирование символов при передаче сообщения.

Если не указать кодировку, то пользователь вместо русских букв увидит непонятные иероглифы, а если указать кодировку, но забыть указать параметр Content-Transfer-Encoding, то вместо русского текста он увидит так называемый transliterirovannyi tekst. Заголовок Content-type также указывает формат сообщения — text/plain. О форматах мы поговорим чуть позже.

Если вы хотите преобразовать заголовки сообщения и само сообщение из одной кодировки в другую, например из koi8-r в windows-1251, используйте функцию `convert_cyr_string()`, которая уже рассматривалась в этой книге (п.9.7).

Теперь займемся организацией рассылки. Пусть текст письма, которое нужно разослать, находится в файле `message.txt`, а список адресатов – в файле `users.txt`. Сценарий 19.1 автоматизирует рутинную работу.

**Листинг 19.1. Простой список рассылки**

```
<?
// Список-массив электронных адресов
$Users = file("users.txt");
// Сообщение
$Message = join('',file("message.txt"));
// Отправляем сообщение каждому пользователю
```



http://all-ebooks.com

```
foreach ($Users as $user) mail($user,"Test, please ignore",$Message);

?>
```

## 19.2. ЕСЛИ СООБЩЕНИЯ НЕ ОТПРАВЛЯЮТСЯ

Иногда функция `mail()` не может отправить сообщение. В большинстве случаев это происходит потому, что она не смогла правильно определить SMTP-сервер. Если у вас установлена и настроена программа sendmail или другой MTA (Mail Transfer Agent), у вас не будет проблем с отправлением писем. Если же нет, то вам придется использовать сокеты (глава 14) или класс `HtmlMimeMail`, который мы рассмотрим в следующем пункте.

## 19.3. ОТПРАВЛЕНИЕ ПИСЕМ В ФОРМАТЕ HTML И ВЛОЖЕНИЙ

### 19.3.1. Общие положения

MIME-формат `plain/text` идеально подходит для электронных сообщений, но иногда нам нужно послать сообщение с форматированием текста и/или картинкой. Поэтому нам просто не обойтись без формата `text/html`. Обычно в формате HTML отправляются рекламные сообщения.

Если вы хотите отправить простое HTML-сообщение без картинок, то можно воспользоваться и функцией `mail()`. Например:

```
$msg = "<html><body><h1>Привет!!!</h1></body></html>";

mail("ivan@ivanov.org","Привет",$msg,"Content-Type: text/html;
    charset=\"koi8-r\"\n");
```

Таким способом можно даже вставить в письмо картинку: просто поместите в тело HTML-документа тег `<img src=http://...>`. Сам файл рисунка передан не будет, но если почтовая программа пользователя поддерживает HTTP, то она загрузит и отобразит картинку сама.

Если мы хотим передать рисунок или другой нетекстовый файл прямо в письме, чтобы получателю не пришлось ждать, пока этот файл скачается, то HTML нам в этом не поможет. Нужно отправить сообщение в так называемом MIME-формате. Что представляет собой этот формат, говорить не стану — если вам интересно, создайте письмо с вложением в почтовой про-

грамме (например, в Outlook), сохраните его как файл, а потом просмотрите в текстовом редакторе.

### 19.3.2. Использование класса HtmlMimeMail

Класс – это понятие из объектно-ориентированного программирования, которым мы займемся только в 30 главе. А пока, если вы не знакомы с классами, можете рассматривать класс как тип переменной: некий «черный ящик», содержащий набор данных и методов (функций) для их обработки и предназначенный для решения определенной задачи. Программисту не обязательно знать алгоритм решения этой задачи, достаточно просто вызывать методы класса, а класс уже сам знает, что и как делать.

Класс `HtmlMimeMail`, разработанный Ричардом Гейесом (Richard Heyes), позволяет отправлять сообщение в MIME-формате. Скачать этот класс можно по адресу `www.phpguru.org/downloads/html.mime.mail/` или с моего сайта: `http://dkws.org.ua/mybooks/php/mime.zip`.

Распакуйте архив в каталог `/var/www/html/mime`. После распаковки класс пока еще не готов к работе. Чтобы каждый раз не указывать кодировку и параметры SMTP-сервера, пропишем их один раз. Откройте в текстовом редакторе файл `HtmlMimeMail.php` и найдите следующие строки:

```
$this->build_params['html_encoding'] = 'quoted-printable';
$this->build_params['text_encoding'] = '7bit' ;
$this->build_params['html_charset']  = 'ISO-8859-1';
$this->build_params['text_charset']  = 'ISO-8859-1';
$this->build_params['head_charset']  = 'ISO-8859-1';
```

Кодировка ISO-8859-1 не поддерживает кириллицу. Для отправки сообщений на русском языке нужно использовать кодировку KOI8-R или CP1251. Также для корректного отображения кириллицы нужно включить 8-битное кодирование символов. Разработчик класса `HtmlMimeMail` не предвидел, что класс будут использовать на территории СНГ, поэтому он указал кодировку ISO-8859-1 и 7-битное кодирование. Сами понимаете, нас эти параметры никак не устраивают. Поэтому без всякого зазрения совести заменяем их следующими:

```
$this->build_params['html_encoding'] = 'quoted-printable1;
$this->build_params['text_encoding'] = '8bit';
$this->build_params['html_charset']  = 'koi8-r';
$this->build_params['text_charset']  = 'koi8-r';
$this->build_params['head_charset']  = 'koi8-r';
```

Можно было бы и не изменять исходный текст класса, но не хочется устанавливать эти параметры в каждой программе, использующей класс. Исхо-



http://all-ebooks.com

дя из тех же самых соображений, укажем имя SMTP-сервера по умолчанию и порт:

```
$this->smtp_params['host'] = 'localhost';
$this->smtp_params['port'] = 25;
```

Если агент MTA не установлен на вашей машине, то имя «localhost» нужно заменить именем SMTP-сервера вашего провайдера.

Затем откройте файл `smtp.php` и найдите следующие строки:

```
$this->authenticated = FALSE;
$this->timeout = 5;
$this->status = SMTP_STATUS_NOT_CONNECTED;
$this->host = 'localhost';
$this->port = 25;
$this->helo = 'localhost';
$this->auth = FALSE;
$this->user = '';
$this->pass = '';
```

Если ваш SMTP-сервер требует аутентификации, то установите значение `TRUE` для параметра `authenticated` и укажите имя пользователя (`user`) и пароль (`pass`). Также нужно указать имя SMTP-сервера и его порт.

Если у вас медленный канал связи, будет полезным увеличение таймаута для функции `fsockopen()`: установите, например, значение 10 или даже 30 секунд.

Можно пробежаться по остальным файлам пакета и заменить значение параметра `charset` некоторых функций на «koi8-r».

Вот теперь ваш класс полностью готов к работе. Следующий пример (листинг 19.2) демонстрирует, как отправить письмо в формате HTML с картинками, прикрепив к нему файл zip-архива.

**Листинг 19.2. Пример использования класса HtmlMimeMail**

```
<?
// на время отладки включаем отображение всех ошибок
error_reporting(E_ALL);

// подключаем класс htmlMimeMail. Файл   htmlMimeMail.php
//  должен  находиться  в  одном  каталоге  с  нашим  сценарием
include('htmlMimeMail.php');

// создаем новый объект класса htmlMimeMail
$mail = new htmlMimeMail();
// загружаем картинку, которая станет фоновым изображением в HTML-документе
```

```
$background = $mail->getFile('background.gif');

// загружаем файл docs.zip, который будет прикреплен к письму
$attachment = $mail->getFile('docs.zip');

// загружаем письмо в текстовом формате - для тех почтовых
// программ,
// которые не поддерживают HTML
$text = $mail->getFile('message.txt');

// загружаем то же письмо в формате HTML
$html = $mail->getFile('message.html');

// передаем письмо объекту htmlMimeMail и сообщаем,
// что его формат - HTML
$mail->setHtml($html, $text);

// связываем картинку с HTML; MIME-тип: image/gif
$mail->addHtmlImage($background, 'background.gif', 'image/gif');

// прикрепляем файл. MIME-тип: application/zip
$mail->addAttachment($attachment, 'docs.zip', 'application/zip');

// адрес отправителя
$mail->setReturnPath('ivan@ivanov.com');

// А это - некоторые заголовки
$mail->setFrom('"Иван" <ivan@ivanov.com');
$mail->setSubject('Привет');
$mail->setHeader('X-Mailer', 'HTML Mime mail class (http://phpguru.
org)');

// Отправляем письмо по адресу: sidorov@ukr.net
$result = $mail->send(array('sidorov@ukr.net'), 'smtp');

 //Если есть ошибки, выводим их, а если нет - отображаем "Отправлено"
if (!$result)
{
  print_r($mail->errors);
} else {
  echo 'Отправлено';
}

?>
```

Я старался писать понятные комментарии, но, на всякий случай, прокомментирую еще раз. Чтобы отправить сообщение в формате MIME, нужно:



http://all-ebooks.com

1. Подключить класс и создать объект класса `HtmlMimeMail`.
2. В обыкновенные переменные прочитать все файлы (и вложения, и картинки, отображаемые в HTML-документе, и сам текст письма) с помощью метода `getFile()`.
3. Передать объекту `HtmlMimeMail` текст письма с помощью метода `setHtml()`. Вы можете прикрепить к письму несколько HTML-файлов, но только один будет основным, все остальные будут обыкновенными вложениями.
4. Связать картинки с HTML-файлом сообщения вызовом метода `addHtmlImage()`.
5. Добавить все вложения методом `addAttachment()`, указав при этом переменную вложения, название вложения (в качестве которого обычно указывают имя файла), а также его MIME-тип.
6. Задать обратный адрес и некоторые заголовки письма.
7. Отправить письмо методом `send()`.

Теперь рассмотрим еще один пример (листинг 19.3), демонстрирующий отправку обычного письма, состоящего из текстового (не HTML) сообщения и вложенного файла.

**Листинг 19.3. Отправка сообщения с вложением**

```
<?
error_reporting(E_ALL);
// подключаем   класс htmlMimeMail
include('htmlMimeMail.php');

// создаем новый объект класса htmlMimeMail
$mail = new htmlMimeMail();

// загружаем файл docs.zip, который будет вложен в письмо
$attachment = $mail->getFile('docs.zip');

// загружаем сообщение в текстовом формате
$text = $mail->getFile('msg.txt');

// передаем письмо объекту htmlMimeMail и сообщаем,
// что его формат - текстовый
$mail->setText($text);

// добавляем вложение
$mail->addAttachment($attachment, 'docs.zip', 'application/zip');
```

http://all-ebooks.com

```
// устанавливаем заголовки: имя отправителя
$mail->setFrom('Иван <ivan@localhost>');
// простая копия
$mail->setCc('Carbon Copy <cc@example.com»');
// "слепая" копия
$mail->setBcc('bec@example.com');

// отправляем сообщение
$result = $mail->send(array('"Сидоров" <sidorov@localhost>'));
echo $result ? 'Отправлено' : 'Ошибка';
?>
```

Как видите, ничего сложного нет, просто вместо метода `setHtml()` вызывается метод `SetText()`. Обратите также внимание на дополнительные заголовки: вызовом метода `setCc()` мы указываем адрес получателя копии письма (CC), а вызовом метода `SetBcc()` – адрес получателя скрытой копии (BCC).

Надеюсь, что приведенного материала вполне достаточно для работы с классом. Если вас интересует дополнительная информация — изучите исходные тексты класса: они достаточно информативны.

## 19.4. ПРИЕМ ПИСЕМ ПО ПРОТОКОЛУ POP3

### 19.4.1. Общие сведения о механизме приема писем и протоколе POP3

Мы рассмотрели класс `HtmlMimeMail`, позволяющий легко отправлять письма в формате MIME. Перед этим мы узнали, как можно отправить письмо: используя функцию `mail()` и напрямую — через сокеты. Теперь осталось разобраться, каким образом в PHP можно организовать получение писем.

Встроенной функции, насколько я знаю, нет, но зато можно написать сценарий, который через сокеты «свяжется» с сервером POP3 и получит сообщения. Конечно, для этого нужно знать протокол POP3, но его рассмотрение выходит за рамки этой книги. Поэтому мы пойдем по пути наименьшего сопротивления: будем использовать уже готовый класс POP3 (разработчик Manuel Lemos). С одной стороны, это правильно: зачем заново изобретать велосипед? А с другой — никто не мешает вам разобраться с протоколом POP3 и самому написать аналог класса POP3.

Минимальные знания о протоколе POP3 (Post Office Protocol ver. 3) нам все-таки необходимы. Программа-клиент, например The Bat!, Outlook, kmail



http://all-ebooks.com

или любая другая, поддерживающая POP3, подключается к почтовому серверу, который также должен поддерживать протокол POP3. После подключения программа-клиент отправляет серверу имя пользователя и его пароль. Если эти данные правильны, почтовый клиент получает список сообщений. Список возвращается в виде:

```
Номер сообщения / Размер сообщения
```

Каждое сообщение, кроме своего номера, имеет уникальный идентификатор, например 3ebcc2a500000001.

Почтовый клиент, получив список сообщений, может:

- Начать скачивание всех сообщений.
- Получить какое-нибудь определенное сообщение.
- Удалить с сервера все сообщения или только сообщения, удовлетворяющие определенному условию, например те, размер которых превышает 1 Мб. Клиент может также удалить с сервера скачанные сообщения либо оставить их там.

Действия почтового клиента зависят от того, как он запрограммирован изначально, а также от того, как он был впоследствии настроен пользователем.

## 19.4.2. Использование класса POP3

Вот теперь мы можем приступить к написанию собственного сценария, эмулирующего работу почтового клиента.

Класс POP3 можно скачать с моего сайта `http://dkws.org.ua/mybooks/php/`. Рассмотрим важнейшие методы класса (табл. 19.1).

**Таблица 19.1.** *Методы класса POP3*

| Метод | Назначение |
|---|---|
| Open() | Открывает новое соединение. До вызова этого метода нужно задать значение свойства hostname, т.е. указать имя сервера, к которому нужно подключиться |
| Login(string $user, string $pass, int $apop) | Регистрирует пользователя $user с паролем $pass. Этот метод нужно вызвать после установления соединения, то есть после успешного вызова Open(). Параметр $apop определяет, должна ли использоваться авторизация APOP (1) или нет (0) |

**Таблица 19.1.** *Методы класса POP3 (продолжение)*

| | |
|---|---|
| Statistics (&$messages,&$size) | Метод записывает в переменную $messages количество сообщений, а в переменную $size – общий объем всех сообщений |
| ListMessages (string $message, int $id) | Возвращает список сообщений. Нумерация сообщений начинается с единицы — это важно. Первый параметр задает количество возвращаемых сообщений. Если задана пустая строка, будут возвращены все сообщения. Если второй параметр ($id) = 1, то будет возвращен уникальный идентификатор каждого сообщения, иначе будет возвращен размер сообщения |
| Retrieve Message ($N,&$headers, &$body, int $lines) | Получает сообщение с номером $N. В переменные $headers и $body записываются заголовки и тело сообщения, соответственно. Последний параметр определяет, сколько строк сообщения будет возвращено. Если хотите получить все сообщение, укажите отрицательное значение |
| DeleteMessage(int $N) | Удаляет сообщение с номером $N |
| ResetDeletedMessages() | Снимает с сообщений отметки для удаления. Если вы отметите какие-нибудь сообщения методом DeleteMessage(), а потом вызовете ResetDeletedMessages(), то ни одно сообщение не будет удалено. Этот метод полезно использовать при отладке сценария, чтобы каждый раз не отправлять тестовые сообщения |

Теперь наших знаний вполне достаточно, чтобы написать сценарий получения сообщений (листинг 19.4).

**Листинг 19.4. Получение всех сообщений**

```
<?
// Подключаем класс pop3. Файл pop3.php должен находиться
// в одном каталоге с нашим сценарием
require('pop3.php');

// Задаем имя пользователя и пароль
$user="dhsilabs";
$password="1234567";

// Не использовать APOP
$apop=0;

// Создаем объект класса POP3
$pop3_connection=new pop3_class;
// Почтовый сервер - localhost. Если вы не настроили POP3 сервер
// на своей
```



http://all-ebooks.com

```
// машине, можете попробовать сервер провайдера или, например,
// pop.mail.ru
$pop3_connection->hostname="localhost";

// Открываем соединение
if(($error=$pop3_connection->Open())=="")
{

  echo "<PRE>Подключены к POP3-серверу $pop3_connection->hostname.
       </PRE>\n";

  // Регистрируемся - передаем имя и пароль
  if(($error=$pop3_connection->Login($user,$password,$apop))=="")
  {
    echo "<PRE>Пользователь &quot;$user&quot; зарегистрирован
         </PRE>\n";

    // Получаем статистику
   if(($error=$pop3_connection->Statistics(&$messages,&$size))=="")
    {
     echo "<PRE>В почтовом ящике $messages сообщений общим размером
$size байтов</PRE>\n";

     // Получаем список сообщений
     $result=$pop3_connection->ListMessages("",0);

     // Если предыдущий вызов возвратил переменную-массив,
     // значит,
     // в почтовом ящике есть сообщения
     if(GetType($result)=="array")
     {
       // Выводим номера сообщений и размер
       for(Reset($result),$message=0;
             $message<count($result); Next($result),$message++)
         echo "<PRE>Сообщение ", Key($result)," - ",
           $result[Key($result)]," байтов.</PRE>\n";

       // Получаем идентификаторы сообщений
       $result=$pop3_connection->ListMessages("",1);

       // Если предыдущий вызов возвратил переменную-массив,
       // значит, в почтовом ящике есть сообщения
       if(GetType($result)=="array")
       {
         // Выводим номера сообщений и размер
         for(Reset($result),$message=0;
              $message<count($result); Next($result),$message++)
           echo "<PRE>Сообщение ", Key($result),
           ", Unique ID - \"",$result[Key($result)],"\"</PRE>\n";
```



http://all-ebooks.com

```
          if($messages>0)
          {
          // Получаем каждое сообщение, нумерация с 1
          for($M=1; $M<=$messages; $M++)
          {
          // получаем сообщение с номером $M (первые 9 строк тела)
          if(($error=
        $pop3_connection->RetrieveMessage($M,&$headers,&$body,9))==="")
              {
               echo "<PRE>Сообщение $M :\n---Заголовки---</PRE>\n";
               for($line=0;$line<count($headers);$line++)
         echo "<PRE>",HtmlSpecialChars($headers[$line]),"</PRE>";
                echo "<PRE>---Конец заголовков---\n---Начало тела
сообщения---</PRE>\n";
               echo "<PRE>";
               for($line=0;$line<count($body);$line++)
                echo "$line: ".HtmlSpecialChars($body[$line])."\n";
                echo "<PRE>---Конец тела сообщения---</PRE>\n";

               // помечаем сообщение на удаление
               if(($error=$pop3_connection->DeleteMessage($M))==="")
               {
                echo "<PRE>Сообщение $M помечено на удаление.</PRE>\n";
                // сбрасываем список сообщений для удаления
        if(($error=$pop3_connection->ResetDeletedMessages())==="")
                    {
                    echo "<PRE>Пометка на удаление снята.</PRE>\n";
                    }
                } // if DeleteMessage
              } // if RetrieveMessage
            } // for messages
          } // if ($messages>0)

          if($error=="" && ($error=$pop3_connection->Close())=="")
          {
            echo "<PRE>Отключаемся от POP3-сервера
               &amp;quot;$pop3_connection->hostname&amp;quot;.</PRE>\n";
          }
        } // if(GetType($result)=="array")
        else
          $error=$result;
      } // if(GetType($result)=="array")
      else
       $error=$result;

    } // if Statistics
  } // if Login
}
```



http://all-ebooks.com

```
if($error!="")
  echo "<H2>Ошибка: ",HtmlSpecialChars($error),"</H2>";

?>
```

На рис. 19.1 и 19.2 вы видите вывод сценария 19.4.

Итак, в этой главе вы научились отправлять и принимать сообщения. Теперь ваших знаний вполне хватит, чтобы написать свой собственный веб-клиент для работы с электронной почтой.

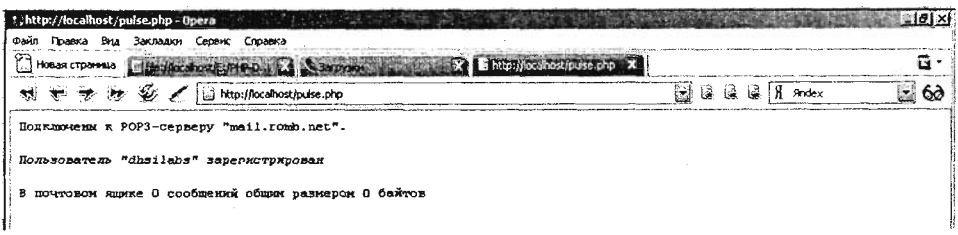



***Рис. 19.1. Нет сообщений***

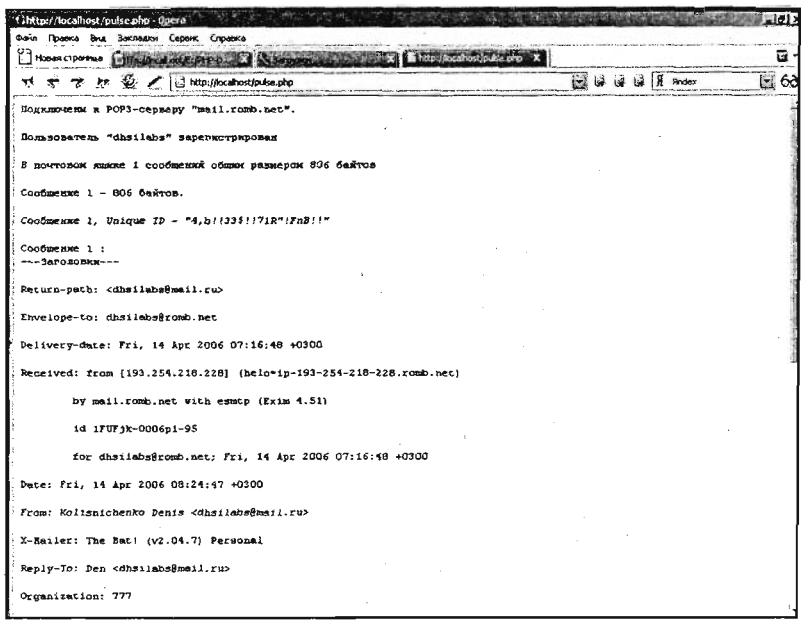



***Рис. 19.2. Есть 1 новое сообщение***

# ГЛАВА 20.

# ФОРУМ PHPBB

http://all-ebooks.com

Сейчас мы познакомимся с лучшим среди бесплатных форумов — форумом PHPBB. Действительно, поработав с этим форумом, чувствуется веяние профессионализма разработчиков. Несмотря на то, что форум бесплатный, работает он не хуже любого коммерческого, например, того же vbulletin (`www.vbulletin.com`).

Форум PHPBB обеспечивает:

- все функции форума;
- ведение списка пользователей;
- поддержку групп пользователей;
- создание профилей пользователя, то есть возможность определить индивидуальные настройки для каждого пользователя;
- рассылку новостей зарегистрированным пользователям форума;
- функции анти-флудинга;
- поиск;
- удобное администрирование форума.

Это только основные функции форума, со всеми остальными мы познакомимся в процессе работы с форумом.

## 20.1. УСТАНОВКА ФОРУМА PHPBB

Установка форума не вызывает проблем и не занимает много времени. Распакуйте архив `PHPbb-2.0.19.tar.gz` (версии более ранние, чем 2.0.19, устанавливать не стоит) в какой-нибудь ка-


http://all-ebooks.com

талог, а затем закачайте распакованные файлы на сервер, в каталог `$DocumentRoot/phpbb` (`/var/www/html/phpbb`).

Следующий шаг — измените права доступа для файла `config.php` так, чтобы разрешить модификацию этого файла всем:

```
chmod 666 config.php
```

После этого запустите браузер и введите в адресную строку URL сценария установки `install/install.php`:

```
http://localhost/phpbb/install/install.php
```

Сценарий установки попросит ввести вас следующую информацию:

- **Default board language** — язык интерфейса (у меня был доступен только английский).
- **Database type** — тип базы данных (MySQL 4.x).
- **Installation method** — метод установки: install (инсталляция).
- **Database Server Hostname** — имя сервера баз данных: обычно `localhost`.
- **Your database name** — имя базы данных (база данных уже должна существовать).
- **Database úser** — имя пользователя созданной базы данных MySQL.
- **Database password** — пароль пользователя MySQL.
- **Prefix for tables in database** — префикс для таблиц форума в базе данных. Префикс нужно указать, если вы используете одну общую базу данных, чтобы имена таблиц различных программных продуктов не конфликтовали друг с другом.
- **Admin email** — адрес электронной почты администратора.
- **Domain name** — доменное имя.
- **Server port** — номер порта Web-сервера (обычно 80).
- **Script path** — путь к форуму от корневого каталога Web-сервера (от каталога `$DocumentRoot`). У нас это `/phpbb/` .
- Имя пользователя администратора и его пароль.

На этом установку можно считать завершенной. Осталось только удалить каталоги `install` и `contrib` (из соображений безопасности), иначе форум работать не будет.

http://all-ebooks.com

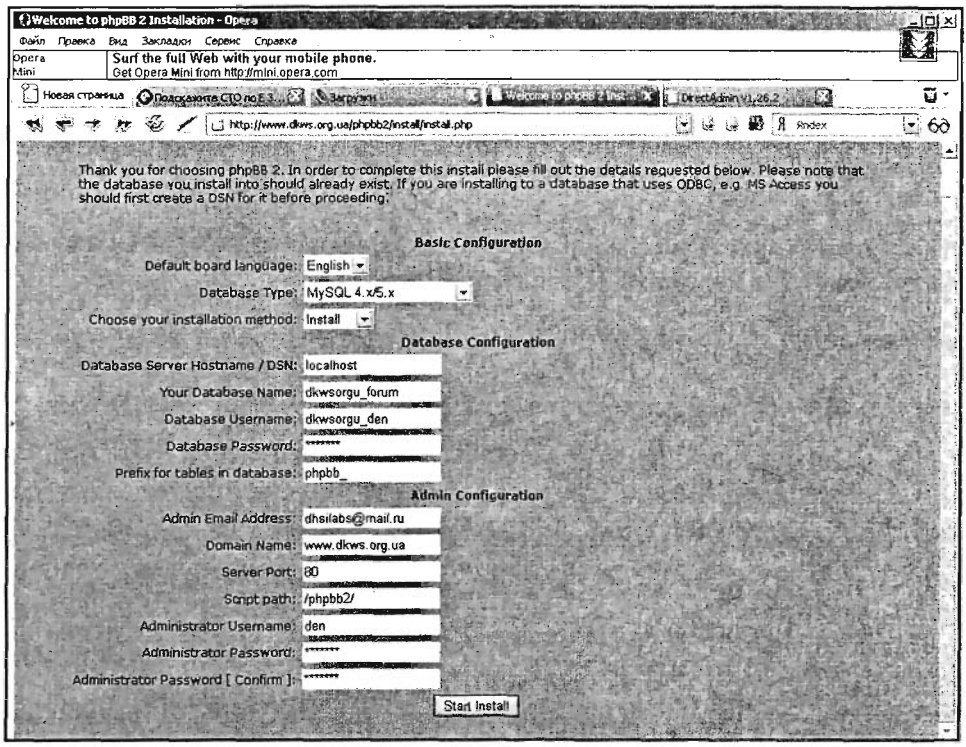



***Рис. 20.1. Установка форума***

После удаления этих каталогов измените права доступа к файлу `config.php` (теперь писать в этот файл имеете право только вы):

```
chmod 644 config.php
```

Теперь русифицируем форум — как бы хорошо мы ни знали английский, работать на родном языке гораздо приятнее. Зайдите на сайт `http://myphpbb.com.ru` и скачайте файл `lang_russian.zip` (вы найдете его в разделе «Русификации»). Я рекомендую скачать первый файл — это самый удачный перевод форума.

Загрузив архив, распакуйте его в каталог `$DocumentRoot/phpbb/language`. Все! Теперь вам в свойствах осталось только выбрать русский язык.

Работать с форумом просто — как обычно, нужно сперва зарегистрироваться (**Register**), затем войти (**Log In**), ну а дальше вы сами знаете, что делать.

При регистрации нового пользователя форум запрашивает следующую информацию (ее объем впечатляет, что говорит о «серьезности» форума):

- Имя пользователя, пароль и его E-mail.
- Номер ICQ, адрес AIM, Yahoo ID, MSN ID.
- Домашнюю страничку.
- Подпись пользователя (можно включить теги HTML, BB, а также смайлики).
- Отображать ли адрес электронной почты.
- Скрывать ли состояние online.
- Уведомлять ли при ответе на сообщение в форуме.
- Уведомлять ли о личном сообщении.
- Открывать ли новое окно при получении личного сообщения.
- Всегда ли добавлять подпись.
- Разрешать ли использование тегов HTML, BB и смайликов.
- Язык форума.
- Скин форума.
- Временную зону.
- Формат даты (синтаксис такой же, как и у функции `date()`).

## 20.2. АДМИНИСТРИРОВАНИЕ ФОРУМА. ОСНОВНЫЕ ПАРАМЕТРЫ ФОРУМА

Теперь поговорим об администрировании форума. Форум phpBB, подобно его коммерческим собратьям, обладает собственным центром управления. Чтобы вызвать центр управления, введите в адресную строку браузера адрес:

```
http://имя_узла/phpbb/admin/index.php
```

> **Примечание.**
>
> Чтобы панель управления форумом отображалась у вас на русском языке, выберите русский язык в собственном профиле. Также настоятельно рекомендую в настройках форума (см. ниже) установить русский как язык по умолчанию (раздел **General Admin/Configuration**, параметр **Default Language**), чтобы пользователям не пришлось, как вам, самостоятельно устанавливать русский язык.

Центр управления позволяет:

http://all-ebooks.com

- просматривать статистику форума;
- просматривать список подключенных к форуму пользователей;
- создавать новые форумы;
- создавать на форуме разделы для обсуждения отдельных тем;
- устанавливать уровень доступа к форуму;
- создавать резервную копию базы данных и восстанавливать базу данных из архива;
- изменять общие параметры форума;
- создавать стили форума.
- организовать массовую рассылку почты зарегистрированным пользователям;
- создавать новые группы;
- управлять учетными записями пользователей, назначать модераторов групп

и многое другое.

Первым делом зайдите в раздел **General Admin/Configuration** (Общие настройки/Конфигурация) и измените общие параметры форума. Назначение этих параметров приведено в табл. 20.1.

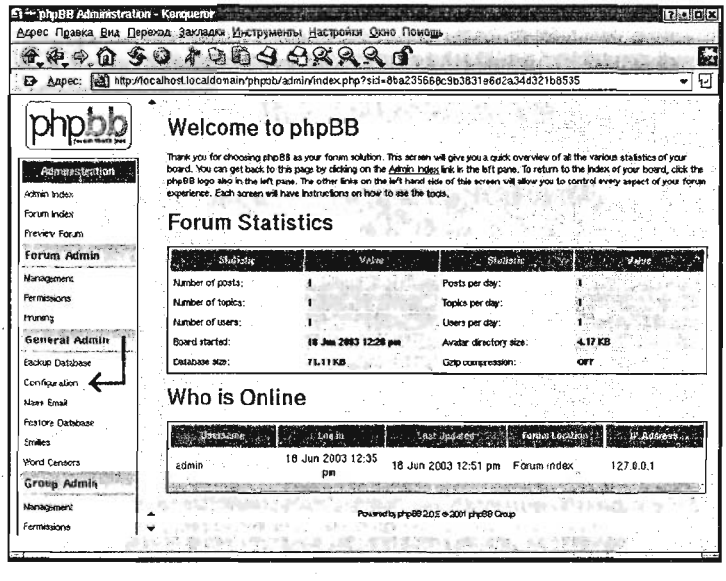



***Рис. 20.2. Панель администрирования форума***



http://all-ebooks.com

**Таблица 20.1.** *Общие параметры форума*

| Параметр | Назначение |
|---|---|
| **Общие настройки форумов** | |
| Имя сервера | Доменное имя вашего сервера, например www.dkws.org.ua |
| Порт сервера | Значение этого параметра, равное по умолчанию 80, нужно изменять только в том случае, когда вы точно знаете, что веб-сервер работает на другом порту |
| Путь к форумам | Путь к каталогу, в котором находится форум, относительно корневого каталога веб-сервера |
| Название сайта | Как вы яхту назовете, так она и поплывет! Это название будет написано крупным жирным шрифтом, поэтому его увидят все пользователи |
| Описание сайта | В описание можно вставлять HTML-код, например ссылку на главную страницу сайта или вообще меню сайта |
| Отключить форумы | Никогда не выбирайте «Да». Если вы уже установили это значение, не выходите из панели администрирования, иначе не сможете заново в нее зайти |
| Включить активизацию учётных записей | Включение этой опции приведет к тому, что создания учетной записи будет недостаточно для того, чтобы зарегистрироваться на форуме: новую учетную запись потребуется активировать. Активация может происходить в двух режимах:<br>• «Пользователь»: активация выполняется пользователем, то есть на указанный пользователем e-mail приходит письмо, содержащее код активации и просьбу активировать учетную запись;<br>• «Администратор»: активация выполняется администратором, то есть вами. Как только регистрируется новый пользователь, вам приходит письмо со ссылкой, перейдя по которой вы можете запретить регистрацию пользователя или разрешить ее.<br>В целях безопасности собственного форума обязательно включите эту опцию. Если у вас небольшой сайт, где мало пользователей, можете сами активировать их учетные записи (режим «Администратор»). Если же сайт большой, где каждый день могут регистрироваться десятки пользователей, лучше выбрать режим «Пользователь» |
| Включить код подтверждения | Эта опция служит для того, чтобы на вашем форуме не регистрировались роботы. Она требует, чтобы пользователь ввел число, изображенное на картинке в искаженном виде: считается, что распознать кривое изображение может человек, но не программа. Обязательно включите эту опцию, если не хотите, чтобы роботы завалили ваш форум спамом |

**Таблица 20.1.** *Общие параметры форума (продолжение)*

| | |
|---|---|
| Разрешить автоматический вход на форум | Автоматический режим более удобен для домашнего пользователя, который всегда работает за своим компьютером. В этом случае при следующем посещении форума пользователь будет зарегистрирован на форуме автоматически. Если же автоматический режим выключен, то при каждом посещении форума пользователю придется заново регистрироваться.<br>Разрешение автоматического режима опасно не для форума, а для ваших пользователей: если кто-то из них отметил режим «Запомнить меня», работая из интернет-кафе, то следующий посетитель кафе сможет писать сообщения от имени рассеянного пользователя. Обязательно предупреждайте пользователей, чтобы они покидали форум корректно, нажимая кнопку «Выход» |
| Автоматический вход на форум действителен | Срок в днях с момента последнего посещения, в течение которого пользователь может автоматически войти на форум. Рекомендуется значение 2. Значение 0 запретит автоматический вход |
| Рассылка e-mail сообщений через форумы | Включение этой опции даст пользователям право отправлять друг другу e-mail через ваш форум. Отключите ее, если хотите экономить трафик |
| Задержка «флудинга» | Позволяет указать минимальное время в секундах, которое должно пройти с момента отправки пользователем последнего сообщения |
| Тем на страницу | Максимальное количество тем форума, выводимое на одной странице |
| Сообщений на страницу | Максимальное количество сообщений на одной странице. Если вы рассчитываете на небольшие сообщения, можете установить, например, 20 или 30. Если же ваш форум предназначен для публикации объемистых статей, то установите 10 или даже 5 |
| Сообщений в «популярной» теме | Сколько сообщений должно быть в теме, чтобы она стала популярной |
| Стиль по умолчанию | Позволяет выбрать оформление (дизайн) вашего форума |
| Заменять стиль пользователя | Если пользователь выбрал другой стиль и эта опция включена, то стиль пользователя будет заменен стилем по умолчанию, то есть стиль по умолчанию будет подавлять стиль, установленный пользователем в собственном профайле |
| Язык по умолчанию | Ну, тут все, я думаю, понятно: русский |
| Формат даты | Синтаксис формата даты аналогичен функции date() в PHP |
| Часовой пояс | Поправка на местное время относительно гринвичского. Для справки: московское время – это +3 к Гринвичу (+4 – летнее московское), а киевское +2 (летнее +3) |



http://all-ebooks.com

**Таблица 20.1.** *Общие параметры форума (продолжение)*

| | |
|---|---|
| Включить сжатие GZip | Сжатие позволяет сэкономить дисковое пространство, но зато форум будет работать медленнее |
| Включить чистку форумов | Если эта опция включена, то старые сообщения будут удаляться из форума автоматически |
| **Настройки Куков** | |
| Все опции | Лучше оставить по умолчанию |
| **Личные сообщения** | |
| Личные сообщения | Включает/выключает возможность отправки личных сообщений. Сообщения отправляются локально, то есть копируются в соответствующую таблицу базы данных форума, а не пересылаются по электронной почте. Поэтому личные сообщения можно включать, не опасаясь, что они «скушают» драгоценный трафик |
| Макс. число сообщений в папке «папка» | Название опции говорит за себя |
| **Общие настройки форумов и пользователей** | |
| Макс. кол-во вариантов ответа в опросе | Значение по умолчанию равно 7. Это значение должно удовлетворить все потребности пользователей при создании опросов |
| Разрешить HTML | В целях безопасности HTML лучше не разрешать |
| Разрешённые теги HTML | Если вы разрешили HTML, то вам нужно указать, какие теги разрешены. Имена тегов указываются через запятую, например, b,i,u,pre |
| Разрешить BBCode | Теги, перечисленные выше, дают пользователю возможность изменить начертание шрифта. Той же цели можно добиться и запретив HTML: достаточно разрешить BBCode, позволяющий форматировать текст без помощи HTML |
| Разрешить смайлики | Разрешает/запрещает смайлики, любимые многими пользователями. На моем форуме смайлики нестандартные (рис. 20.3). Если вы хотите себе такие же, скачайте файл http://www.dkws.org.ua/files/smiles.rar (90K) и распакуйте его в каталог phpbb2/images/smiles |
| Разрешить подписи | Пользователям понравится возможность украшать свои сообщения художественной подписью, поэтому не отключайте эту опцию... |
| Макс. длина подписи | ...но ограничьте разрешенную длину подписи, например, 128 символами |
| Разрешить смену имени пользователя | Менять ник (логин) пользователю не нужно: если что, пусть создают новый, а этот удаляют |


http://all-ebooks.com

**Таблица 20.1.** *Общие параметры форума (продолжение)*

| **Настройки аватар** | |
|---|---|
| Разрешить аватар из галереи | Аватара – это небольшое изображение, которое ассоциируется с пользователем и отображается вместе с ником пользователя в каждом его сообщении. Форум phpBB и так очень "прожорлив" по отношению к трафику, поэтому аватары лучше отключить, ведь они создадут дополнительную нагрузку на сервер |

Все остальные параметры, не описанные в таблице, несущественны. Установив значения опций, нажмите кнопку Отправить, и все установленные параметры будут сохранены.

**Примечание.**

Выбор русского языка интерфейса приведет к изменению только текстовых надписей, а надписи кнопок и других графических элементов останутся на английском. Если вы хотите это исправить, скачайте новые кнопки для темы subSilver (основная тема форума) с `http://dkws.org.ua/files/subSilver_russian.zip` (18 K).

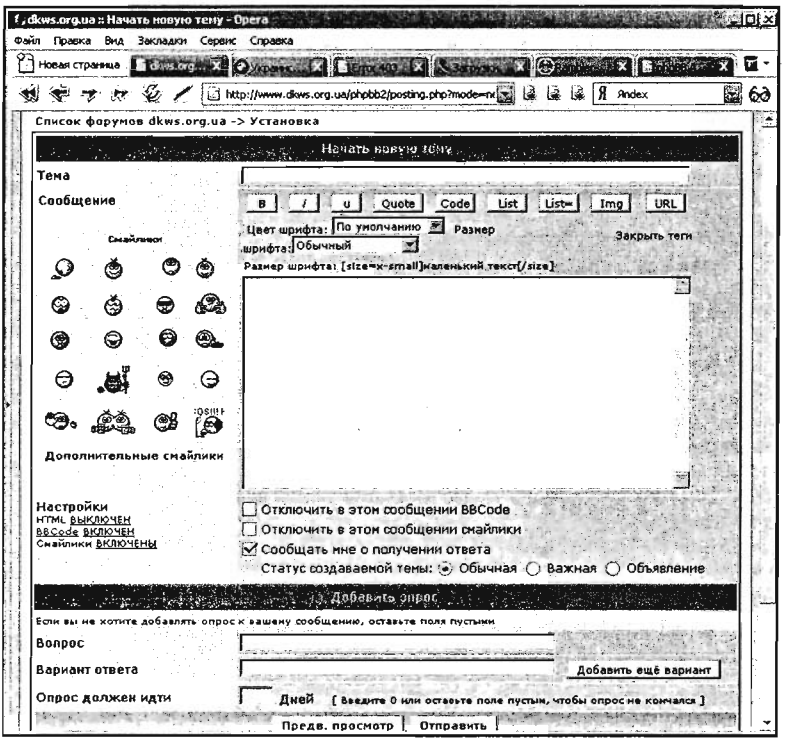



***Рис. 20.3. Нестандартные смайлы для вашего форума***



http://all-ebooks.com

## 20.3. УПРАВЛЕНИЕ ФОРУМАМИ

Для добавления, удаления или изменения какого-либо форума зайдите в раздел **Форумы/Управление** (рис. 20.4).

Для добавления нового форума введите его название в поле рядом с кнопкой **Создать новый форум** и нажмите эту кнопку. После этого вам нужно будет ввести описание форума, выбрать его категорию (она будет установлена по умолчанию), и, если нужно, установить опции автоматического удаления старых постов, что позволяет сэкономить дисковое пространство.

Категории объединяют форумы одного направления: например, вы можете создать категорию «Операционные системы», в которой будут форумы Linux, FreeBSD, MacOS, Windows.

Для создания новой категории предназначена кнопка **Создать новую категорию** в самом низу списка форумов. Нажимать ее нужно, ясно, после заполнения поля рядом с кнопкой.

Рядом с каждым форумом будут ссылки **Изменить, Удалить, Сдвинуть вверх/Сдвинуть вниз, Синхронизация**. С первыми двумя все ясно. Следу-

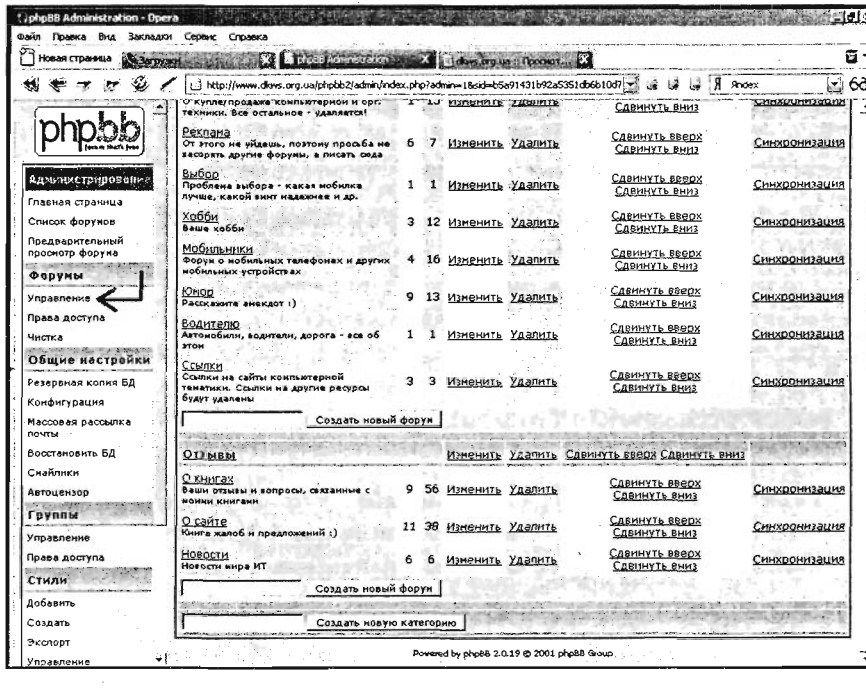



***Рис. 20.4. Управление форума***

http://all-ebooks.com

ющие две позволяют управлять отображением форума в категории. Осталось понять, что делает кнопка **Синхронизация**. Давайте разберемся.

Информация форума phpBB хранится в базе данных MySQL, что позволяет существенно ускорить работу форума. Но иногда информация в таблицах не успевает синхронизироваться из-за большой нагрузки на сервер (необязательно созданной вашим форумом: кроме вас, сервер MySQL обслуживает и других клиентов вашего провайдера). Если информация в таблицах не синхронизировалась, то форум начинает работать некорректно, например показывать ссылки на несуществующие темы или сообщения. Если ваш форум ведет себя так же, то вам нужно нажать ссылку **Синхронизация** напротив того форума, который ведет себя не так, как нужно. Эта операция совершенно безопасна для форума: если вы случайно нажали на ссылку **Синхронизация**, хуже форуму от этого не будет. В особо запущенных случаях операция синхронизации может занять некоторое время (минуту и больше), поэтому не спешите закрывать браузер.

В разделе **Форумы/Права доступа** у вас есть возможность установить права доступа к любому форуму. Выберите форум из списка и нажмите кнопку **Выбрать форум**. Для каждого форума можно выбрать один из следующих режимов (рис. 20.5):

- **Публичный**: все пользователи могут читать форум и оставлять сообщения. Зарегистрированные пользователи имеют право редактировать свои сообщения, принимать участие в опросах и создавать свои опросы. Модераторы и администраторы имеют право создавать специальные темы – «важные» и объявления.
- **Зарегистрированный**: незарегистрированные пользователи могут только читать сообщения, но не имеют права создавать новые темы или отвечать на сообщения других пользователей. У зарегистрированных пользователей этих ограничений нет. Кроме того, зарегистрированные пользователи имеют право создавать и участвовать в опросах, а также редактировать собственные сообщения. Модераторы имеют право, как и в предыдущем случае, создавать специальные темы.
- **Зарегистрированный [Спрятанный]**: этот форум будут «видеть» только зарегистрированные пользователи. Для зарегистрированных пользователей и модераторов данный режим полностью аналогичен предыдущему.
- **Приватный**: пользователи, которым специально не предоставили права посещения этого форума, могут только видеть, что он существует. Пользователи, которым такое право предоставлено, могут чи-



http://all-ebooks.com

тать, создавать новые темы, редактировать собственные сообщения, создавать и принимать участие в опросах. Модераторы имеют право создавать специальные темы.

- **Приватный [Спрятанный]**: аналогичен предыдущему, только пользователи, у которых нет прав на посещение форума, даже не увидят его.
- **Модераторы**: рядовые пользователи, включая зарегистрированных, могут только видеть, что этот форум существует. А вот посещать его – читать, создавать новые сообщения, новые темы, опросы, участвовать в опросах – имеют право только модераторы.
- **Модераторы [Спрятанный]**: аналогичен предыдущему, но рядовые пользователи даже не увидят, что форум существует.

В продвинутом режиме (**Advanced Mode**) вы можете определить, кто имеет право:

- просматривать сообщения (View);
- читать сообщения (Read);
- создавать новые сообщения (Post);

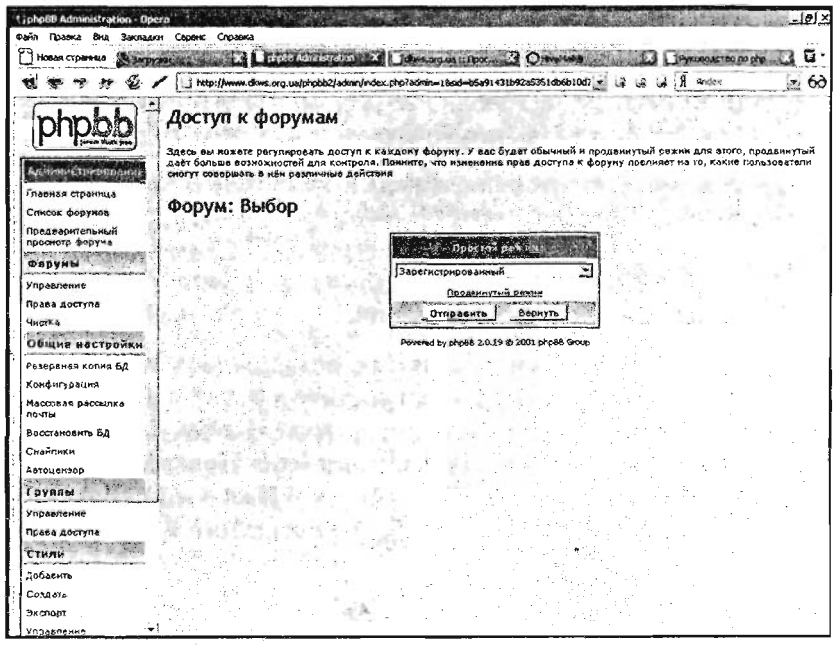



***Рис. 20.5. Назначение прав доступа для форума***

http://all-ebooks.com

- отвечать на сообщения (Reply);
- редактировать сообщения (Edit);
- удалять сообщения (Delete);
- голосовать (Vote).

Для каждого из этих действий вы можете указать одну из следующих категорий пользователей:

- все (All);
- только зарегистрированные (REG);
- модераторы (MOD);
- администратор (ADMIN).

Форумы могут занимать довольно много дискового пространства, поэтому периодически форум нужно «чистить» (скажем, раз в полгода). Конечно, период «генеральной уборки» сильно зависит от популярности форума. Именно для этого и предназначен раздел **Форумы/Чистка (Forums/Clean)**.

## 20.4. УПРАВЛЕНИЕ ПОЛЬЗОВАТЕЛЯМИ

В разделе **Пользователи** вы можете:

- редактировать черные списки;
- редактировать запрещенные имена;
- управлять учетными записями пользователей (редактировать, удалять их);
- назначать права доступа;
- присваивать пользователям звания.

Зачем нужны **Черные списки?** Допустим, пользователь вашего форума с ником «Vasya» утомил и вас, и других посетителей форума спамом или нецензурщиной. Задача разумного администратора – сделать так, чтобы Vasya на форуме больше не появился. Войдите в раздел **Пользователи/Черные списки** (рис. 20.6) и введите ник «Vasya» в поле **Имя** секции **Закрытие доступа отдельным пользователям**. Все, больше никакого Васи на форуме не будет.

Но все ли? Ведь нарушитель сможет тут же зарегистрироваться под новым именем и продолжить свое вредительство. Надежнее будет забанить его по IP-адресу.



http://all-ebooks.com

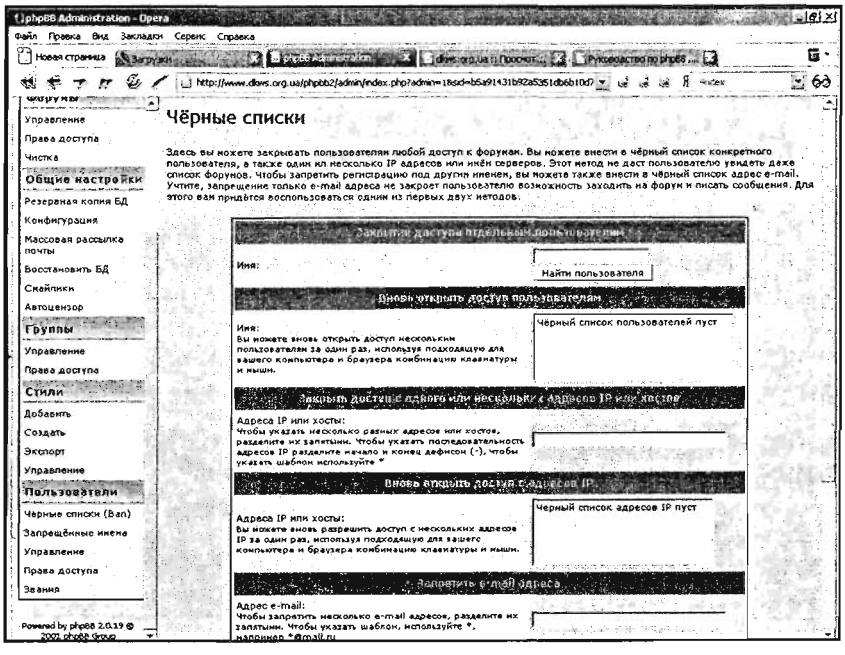



***Рис. 20.6. Управление черными списками***

Для этого вернитесь на форум (откройте в новом окне сам форум, а не администраторский модуль) и выведите список пользователей. Выберите из списка интересующего вас пользователя (рис. 20.7).

Нажмите ссылку **Найти все сообщения пользователя**. Вы увидите список всех сообщений, которые когда-либо отправлял данный пользователь. Перейдите к любой теме. Для определения IP-адреса пользователя нажмите кнопку **ip**. Вы увидите список всех IP-адресов, с каких когда-либо писал выбранный пользователь.

Если IP-адрес пользователя не менялся на протяжении нескольких дней, то он, скорее всего, пишет с постоянного IP (например, 192.168.77.10). Введите этот адрес в поле **Адреса IP или хосты** секции **Закрыть доступ...**, и больше ни один пользователь с этого адреса мешать вашему форуму не будет.

А что делать, если вы получили список, например, 192.168.77.55, 192.168.77.75, 192.168.77.101, говорящий о том, что нарушитель пишет с динамического адреса? Можно закрыть сразу всю сеть 192.168.77.*. Таким образом вы запретите доступ на форум не только нарушителю, но и всем пользователям из его сети, поэтому будьте осторожны: так можно и пол-России забанить.

http://all-ebooks.com

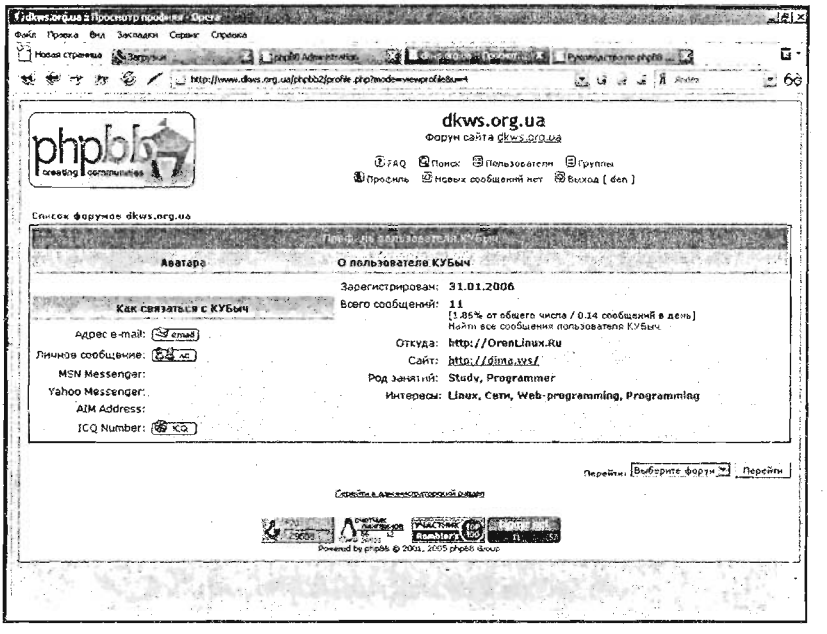



***Рис. 20.7. Информация о пользователе***

Раздел **Пользователи/Запрещенные имена** позволяет перечислить те ники, которые рядовым пользователям брать не полагается: «admin», «root», «moderator» и т.п. Конечно, сами по себе подобные имена никаких администраторских полномочий пользователю не дадут, но помогут ввести в заблуждение других посетителей форума. Согласитесь, что позволять рядовому пользователю подписываться «админом» непрофессионально со стороны настоящего админа.

Раздел **Пользователи/Управление** предназначен для редактирования/удаления учетной записи пользователя. Допустим, пользователь с ником «Pupkin» забыл свой пароль и просит вас выдать ему новый. Убедившись, что от имени «Pupkin» вам пишет именно тот, кто зарегистрировал этот ник (как вы это сделаете, меня не касается), войдите в раздел **Пользователи/Управление** и нажмите кнопку **Найти пользователя**.

В появившемся окошке введите «*» для вывода всех пользователей, а потом выберите нужного из списка. Если список слишком длинный, введите «P*», чтобы получить только тех пользователей, ники которых начинаются с буквы P.

Отметив нужный ник, нажмите кнопку **Выбрать пользователя**.



http://all-ebooks.com

Для изменения пароля введите новый пароль и его подтверждение. Также можете изменить другие данные пользователя в его анкете, если считаете это нужным.

Для удаления пользователя в самом низу страницы отметьте флажок **Щелкните здесь для удаления этого пользователя. Операцию нельзя будет отменить.**

Для внесения изменений нажмите кнопку **Отправить**.

В разделе **Пользователи/Права доступа** вы можете установить права доступа отдельного пользователя или присвоить ему статус модератора. Как и в предыдущем случае, сначала нужно выбрать пользователя из списка. После этого вы можете присвоить ему статус модератора любого форума.

Если вы не хотите присваивать пользователю статус модератора, а хотите просто изменить его права доступа, то щелкните по ссылке **Продвинутый режим** внизу странички. В этом режиме вы можете назначить права доступа пользователя к приватным форумам: разрешить или запретить чтение, создание тем и т.п.

## 20.5. МАССОВАЯ РАССЫЛКА

Возможно, вы хотите отправить всем зарегистрированным пользователям какое-то важное сообщение или просто сделать небольшую рассылку с сообщением о новых материалах на вашем сайте. Для этого служит модуль **Общие настройки/Массовая рассылка почты**.

## 20.6. СТИЛИ

Стили позволяют коренным образом изменить внешний вид вашего форума. Установить новый стиль очень просто: скачайте архив со стилем и распакуйте его в каталог `/phpbb2/templates`. При распаковке следите, чтобы ваш файловый менеджер не переводил символы в именах файлов в нижний регистр, иначе стили будут работать некорректно. Потом в настройках форума просто выберите нужный стиль.

Скачать стили для phpbb2 можно по адресу `http://www.phpbbguru.net/styles`. Также огромное количество стилей можно найти в Интернете.

Если вы не собираетесь радикально менять стиль, а просто хотите немного модифицировать стандартную схему SubSilver, то можете отредактировать следующие файлы:

- Логотип форума:
  `форум/templates/SubSilver/images/logo_phpBB.gif`
- Оформление верха и низа каждой страницы форума:
  `форум/templates/SubSilver/overall_header.tpl`
  `форум/templates/SubSilver/overall_footer.tpl`
- Фоны-заливки:
  `форум/templates/SubSilver/images/cellpic1.gif`
  `форум/templates/SubSilver/images/cellpic2.jpg`
  `форум/templates/SubSilver/images/cellpic3.gif`
  `форум/templates/SubSilver/images/cellpic.gif`

## 20.7. МОДЫ

Моды позволяют расширить возможности вашего форума phpbb, например, добавить новое поле в профиль пользователя, фотоальбом пользователя, функцию поздравления пользователя в день его рождения. С модами возможности форума действительно безграничны.

Моды выполнены в виде текстовых файлов. В этих файлах есть указания для администратора, то есть для вас, какие действия нужно выполнить, чтобы добавить ту или иную функцию. Обычно там находятся указания по работе с исходным текстом форума: имя файла, номер строки и новый фрагмент кода, который нужно вписать вместо старого. Нередко при модификации форума приходится модифицировать его таблицы. В этом случае нужные SQL-операторы указываются в самом начале мода.

Скачать моды можно с сайта `http://www.phpbbguru.net` или с моего сайта `http://dkws.org.ua` (в разделе PHP).



http://all-ebooks.com

# ГЛАВА 21.

# ДЕЛАЕМ ЧАТ ДЛЯ СВОЕГО САЙТА

http://all-ebooks.com

## 21.1. ПРИНЦИП СОЗДАНИЯ ЧАТА

Нужен ли вашему сайту чат? Если ваш сайт постоянно обновляется, значит, его должны посещать постоянные пользователи. Пусть их немного — пять, десять или пятьдесят человек – но тема вашего сайта объединяет эту группу людей. Так почему бы не предусмотреть возможность общения постоянных посетителей? Думаю, что я убедил вас: чат нужен. Сейчас мы рассмотрим идею создания простого чата — на одну комнату.

В любом чате должен быть модуль регистрации, который проверяет введенные посетителем имя и пароль, а также позволяет зарегистрировать нового посетителя. Если введенные имя и пароль правильны, то пользователь попадет в комнату общения. Ну, а дальше, думаю, все ясно: пользователь вводит сообщение, оно передается на сервер и записывается в файл, причем в начало файла (п. 11.5.4), чтобы пользователи видели новое сообщение сверху старых. Модуль, записывающий сообщение в файл, отображает новое содержимое этого файла – для экономии только первые N записей.

Формат файла может быть таким:

```
<login>:Сообщение
```

Писать чат вручную мы не будем. Почему? Да потому, что я не вижу смысла заново изобретать велосипед — уже написано очень много чатов, доступных с исходным кодом.

Если у вас есть время и желание написать самый лучший чат, то возьмите исходники профессионального чата, например чата CHAT.PHP.SPB.RU, посмотрите, как он написан, и сделайте свой.



http://all-ebooks.com

Здесь мы рассмотрим два чата: SimpleChat, простота которого соответствует его названию, и профессиональный чат CHAT.PHP.SPB.RU. Если ни один из них вам не понравится, посетите сайт `http://www.softlinks.ru`. Вы найдете много ссылок на различные чаты — от самых простых до самых сложных.

## 21.2. ПРОСТОЙ ЧАТ – SIMPLECHAT

Если вы заинтересованы в посещаемости вашего сайта, то, кроме интересной информации, привлекательного дизайна, желательно наличие еще двух атрибутов:

- Чата, в котором постоянные посетители смогут поделиться опытом, обсудить какие-нибудь связанные с тематикой сайта проблемы или просто пообщаться.
- Гостевой книги, где посетители могут оставлять мнения о вашем сайте — своего рода книга жалоб и предложений.

Перепробовав десяток различных чатов, я остановил свой выбор на сценарии SimpleChat! версии 1.3.

SimpleChat — это простой HTML-чат, написанный на PHP 4. Чат работает довольно быстро, а вывод сценария небольшой, поэтому большой нагрузки на трафик не будет. Желательно, чтобы ваш браузер поддерживал HTML 4 и JavaScript 1.2 — в этом случае чат будет выглядеть привлекательнее.

Основными преимуществами SimpleChat являются:

- Быстрая загрузка (не больше 9 Кб за каждое обновление).
- Простая установка.
- Не требует MySQL.
- Поддерживает форматирование текста — выделение жирным шрифтом, курсивом и цветом.
- 20 цветовых схем.
- Различные пиктограммы пользователей — пользователь сам выбирает свою пиктограмму.
- Поддержка смайликов и гиперссылок.
- Фильтр HTML-тегов.
- Автоматическое обновление через 5, 10, 20, 30 или 60 секунд.

- 150 строк истории (предыдущих сообщений).
- Список online-пользователей.
- Звуковое сопровождение — при получении нового сообщения вы услышите звуковой сигнал.
- Переведен на разные языки, в том числе и на русский.
- Оптимизирован под браузеры MSIE 4+, Netscape 4+, Opera 4+, Mozilla.
- Нормально работает как на Windows-, так и на Linux-хостинге.

Такое обилие функций для чата, который в архиве занимает 34 Кб — довольно неплохо. В то же время можно выделить сразу бросающиеся в глаза недостатки:

- Нет аутентификации пользователей.
- HTML-чат довольно легко взломать, то есть получить возможность писать сообщения от имени другого пользователя.
- Нет приватной комнаты.

Но, с другой стороны, эти недостатки не кажутся такими уж значительными, ведь чат — это не место для ведения серьезных переговоров. Для них рекомендуется использовать SSL-соединения.

Особенно мне понравилось, что этот чат работает сразу после установки — не нужно ничего изменять или переписывать заново, что часто бывает при использовании сценариев такого рода. Единственное, что мне пришлось изменить, - это настройки языка.

### 21.2.1. Установка, настройка и работа с чатом

Распакуйте архив `schat.zip` в каталог `/var/www/html/schat`. Если вы работаете под Linux, сразу измените права доступа к каталогу `data`:

```
chmod 777 /var/www/html/schat/data
```

Затем откройте файл `incl/hdr.inc` и замените последнюю строку

```
include "lang/lang-en.inc";
```

строкой

```
include "lang/lang-ru.inc ";
```

Сразу после подключения русского языка чат должен заработать. Чтобы попасть в него, введите в адресную строку браузера следующий URL:



http://all-ebooks.com

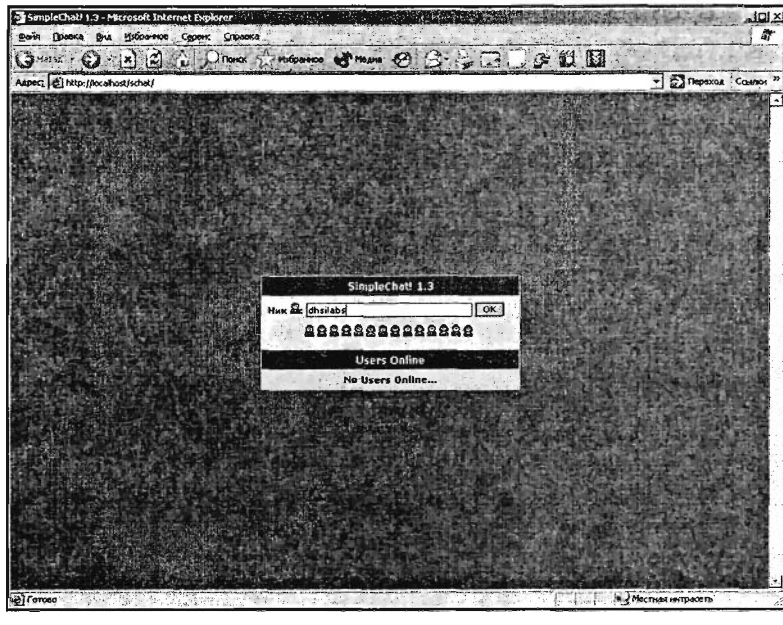



***Рис. 21.1. Регистрация в чате***

```
http://localhost/schat/index.php
```

Чтобы зарегистрироваться, введите свой ник и выберите пиктограмму (рис. 21.1).

SimpleChat отображает сообщения в формате «Дата/Время», «Пользователь», «Сообщения» (рис. 21.2). Справа вы можете видеть список online-пользователей — тех, кто зарегистрирован в чате в данный момент.

В нижней части окна находится панель управления чатом. Здесь вы можете:

- Отправить простое сообщение.
- Отправить форматированное сообщение (допускается использование цвета, изменение начертания, вставка смайликов и гиперссылок).
- Изменить цветовую гамму (в нижней части панели находятся 20 цветных прямоугольников — это кнопки выбора цветовой схемы).
- Включить/выключить звуковое сопровождение.
- Изменить режим работы чата: Чат/Архив/Выход/Очистить/Инфо.



http://all-ebooks.com

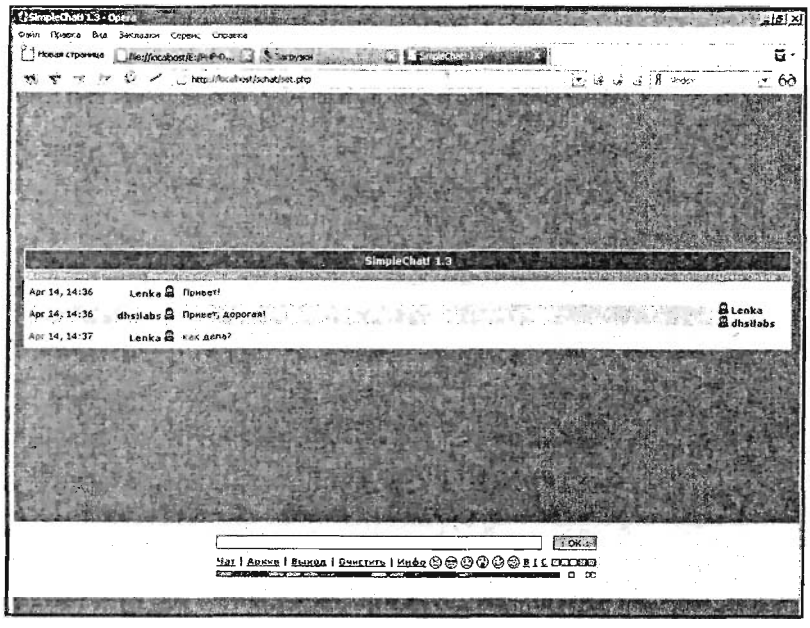



*Рис. 21.2. Чат-комната*

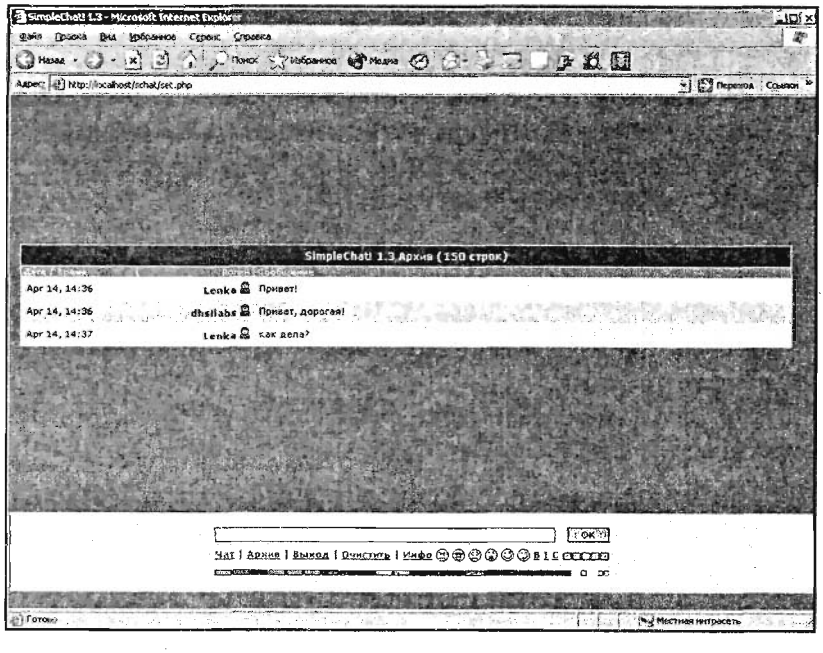



*Рис. 21.3. Архив сообщений*



http://all-ebooks.com

Чаще всего вы будете пользоваться режимами «Архив» и «Выход». С выходом-то все ясно. А вот «Архив» (рис. 21.3) придется немного модифицировать. По умолчанию в этом режиме отображаются 150 последних сообщений. Если вы хотите это число увеличить или уменьшить, понадобится отредактировать файл `index.php`, строка 45:

```
for($i=0;$i<150;$i++){if(isset($myar[$i])){$nr=$nr.$myar[$i]."\
n";}}wrl($nr);
```

Просто замените константу в выражении цикла нужным вам числом.

Вот, собственно, и все. Осталось разобрать типичные проблемы при установке, которые могут помешать чату работать корректно.

## 21.2.2. Проблемы при установке

### Некорректная работа

Если вы не можете:

- зарегистрироваться;
- отправить сообщение;
- изменить цветовую схему,

убедитесь, что в вашем файле `php.ini` включена директива `register_globals`:

```
register_globals=On
```

Если же изменить файл `php.ini` невозможно (в тех случаях, когда хостинг-провайдер не разрешает редактировать файлы конфигурации), то добавьте следующие строки в начало файла `incl/hdr.inc`:

```
ini_set('error_reporting',7);
import_request_variables("GPC");
```

### Некорректное отображение даты и времени

По умолчанию чат настроен на гринвичское время GMT (UTC). Для настройки чата на нужный вам часовой пояс замените 0 в первой строке файла `incl/hdr.inc` нужным вам значением (+/- число):

```
$datetime=gmdate('M j,  G:i',time()+0*3600);
$datetime=gmdate('M j,  G:i',time()+3*3600);
```

#### Я не могу очистить чат!

По умолчанию пароль для очищения чата — 12345. Для изменения пароля отредактируйте файл `clear.php` (строка 7):

```
if(isset($pass)&&$pass = ="12345") . . .
```

### 21.2.3. Полезные советы

Модульная структура чата открывает много интересных возможностей. Например, вы можете легко перевести чат на другой язык, вставить печать списка online-пользователей в любую страницу вашего сайта и многое другое.

Начнем с самого простого — перевода чата на другой язык. Если вы хотите перевести чат на какой-нибудь язык, скопируйте файл `lang/lang-ru.inc` в файл `lang/lang-XX.inc` и переведите сообщения на нужный вам язык, например украинский.

Вот фрагмент файла `lang-ru.inc`:

```
<?php
$crl[0]='10';
$crl[1]='<meta http-equiv="Content-Type" content="text/html;
charset=windows-1251">';
$crl[2] = 'SimpleChat! 1.3';
$crl[3] = 'Sound: On/Off';
$crl[4] = ' : OK : ';
$crl[5] = 'Отмена';
$crl[6] = 'Пароль';
$crl[7] = 'Поменять цвет...';
$crl[8] ='Злость';
$crl[9] = 'Здорово';
$crl[10] ='Огорчение';
$crl[11] = 'Кричать';
```

Тут же можно изменить кодировку сообщений.

Возможно, вы хотите просмотреть полный протокол чата без вызова самого чата. Откройте файл `data/log` — в нем вы найдете все, что вам нужно:

```
1053076834:|:May 16, 9:20:|:evg:|:w1:|: <b></b>
1053076782:|:May 16, 9:19:|:evg:|:w2:|:Как дела? <img src="pics/
m5.gif" width="15" height="15" hspace="2" alt="">
1053076582:|:May 16, 9:16:|:evg:|:w2:|:Привет!
1053076536:|:May 16, 9:15:|:denis:|:w1:|:Hello all
```

Вы можете модифицировать сценарий, добавив в него запись в протокол IP-адреса, если вам это нужно.



http://all-ebooks.com

Используя возможности чата, вы можете отображать список online-пользователей на любой странице вашего сайта. Для этого подключите файл `user.inc` к вашей странице:

```
<?php $pth='xxx'; include "$pth/incl/users.inc"; ?>
```

Здесь `'xxx'` — это относительный путь к чату.

После этого вызовите функцию `whos_online()`, передав ей строку-разделитель пользователей:

```
<?
whos_online('<br> ' ) ;
?>
```

Пример файла `index.php` с использованием функции `whos_online()` приведен в листинге 21.1.

**Листинг 21.1. Вывод списка пользователей**

```
<?php $pth='friends/simple-chat';include "$pth/incl/users.inc";
?>

<html><head>
<title>Основная страница</title>
</head><body>

<!-- вывод списка в одну строку -->
<div style="background-color:#cccccc">
<?php whos_online(' '); ?>
</div><br>

<!-- вывод списка в столбец -->
<div style="background-color:#aaaaaa">
<?php whos_online('<br>'); ?>
</div><br>

</body></html>
```

## 21.3. ПРОФЕССИОНАЛЬНЫЙ ЧАТ CHAT.PHP.SPB.RU

Чат CHAT.PHP.SPB.RU разработан Дмитрием Бородиным (`dima@php.spb.ru`) и доступен по адресу `http://php.spb.ru/chat`. Автор называет этот чат «правильным», и он совершенно прав. Достаточно беглого взгляда на особенности чата:

- Он не использует MySQL. В результате вы можете сэкономить на хостинге, не платя за поддержку MySQL. Кроме того, неиспользование MySQL упрощает установку чата.
- Чат не использует Java-апплетов и Flash-анимации, то есть может работать в любом браузере независимо от того, установлены ли у посетителя чата Java и Flash Player.
- Чат работает в одном окне, что помогает новичкам не запутаться в интерфейсе.
- Сообщения загружаются в чат плавно, без периодических обновлений окна.
- Сообщения, адресованные вам, подсвечиваются.
- Чат снабжен индикаторами связи компьютера пользователя с Интернетом и с чатом.
- Чат поддерживает постоянное соединение с сервером.
- Вместе с чатом предлагается множество дополнительных скинов, позволяющих пользователю легко настроить внешний вид чата.
- Минимальные требования к памяти: чату нужно всего 5 Мб для обслуживания 100 пользователей.
- Чат рассчитан на большую нагрузку — до 1000 пользователей, работающих одновременно.
- Защита от флуда встроена в механизм чата.
- По сети передаются только текстовые реплики пользователей, а не HTML-код, что снижает нагрузку на трафик.
- Новому пользователю при регистрации не удастся выбрать ник, похожий на уже зарегистрированный.
- Чат позволяет пользователю менять свои регистрационные данные, включая пароль, и удалять свой логин.
- Чат имеет возможность интеграции с базами пользователей от любого форума на MySQL — Phorum.org, phpBB, phpNuke и др.
- Настройка уровня безопасности, что актуально для интернет-кафе.
- Чат позволяет создавать чат-порталы, то есть объединять несколько чатов.

И это далеко не все возможности чата. На сайте `http://php.spb.ru/chat` вы найдете не только исходники чата, но и подробную документацию с описанием всех возможностей.



http://all-ebooks.com

### 21.3.1. Установка и настройка чата под Windows

Чат устанавливается полностью автоматически и очень быстро — за 10-15 секунд, в зависимости от производительности вашего компьютера. Скачайте файл `chat-php-spb-ru-04_60.exe`, запустите его. Это обыкновенный самораспаковывающийся RAR-архив. Нажмите кнопку «Извлечь» и подождите, пока чат будет распакован в каталог `c:\www` (каталог можно изменить).

После этого должны запуститься два черных окошка, которые не нужно закрывать. Через пару секунд запустится Internet Explorer и откроет страницу `http://127.0.0.1/`. Все! Чат готов к работе.

Если автоматическая установка у вас не прошла (как у меня), значит, сейчас самое время приступить к настройке чата вручную. Откройте файл

`c:\WWW\chat-php-spb-ru-04\www\__dima.php` и найдите в нем строку:

```
"$os=0";
```

Измените значение переменной `$os` на:

- 3, если у вас Windows 9x/ME;
- 2, если у вас Windows NT/2000/XP.

Потом найдите строки

"`$rand_password=...`" и "`$cookie_pass=...`" и впишите в них два случайных набора символов. Например, udfjHdtujn7dkjfd87.

После этого найдите переменную `$adminfirst` и присвойте ей значение — ник администратора чата:

```
$adminfirst = "admin";
```

Потом запустите файл `c:\www\chat-php-spb-ru-04\apache\apache.exe` — это обыкновенный сервер Apache для Windows. Если сервер не запустился, то вам придется его настроить (глава 4).

Если Apache запустился нормально, запустите файл `c:\www\chat-php-spb-ru-04\daemon\run_this.bat`. Откроется еще одно черное окошко, а затем — Internet Explorer. Если последний не запустился, запустите его вручную (http://127.0.0.1).

Теперь вы можете приступить к настройке чата «под себя». Если у вас возникнут какие-нибудь проблемы, обратитесь к разработчику чата.

### 21.3.2. Установка и настройка чата под Linux

Скачайте самораспаковывающийся архив `chat-php-spb-ru-04_60.exe`. Да, это тот же самый файл, который устанавливает чат под Windows, поэтому вам понадобится Windows для его распаковки. Распаковав файлы, перегрузитесь в Linux, примонтируйте свой Windows-раздел:

```
mount  /dev/hdaN  /mnt/diskc
```

и скопируйте файлы чата... куда? Сейчас покажу. Если вы устанавливаете чат на сервер хостинг-провайдера, то скопируйте всю файловую иерархию в подкаталог `chat` своего домашнего каталога.

На домашнем компьютере, где вы сами root, создайте нового пользователя wwwchat (группа — тоже wwwchat). Все дальнейшие операции будем выполнять от имени этого пользователя.

Чат можно устанавливать в любой каталог. Пусть для определенности это будет подкаталог `chat` вашего домашнего каталога `/home/wwwchat`. Тогда структура корневого каталога чата будет выглядеть так:

- /home/wwwchat/chat/www — PHP-скрипты
- /home/wwwchat/chat/www/src — секретный каталог PHP-скриптов
- /home/wwwchat/chat/www/skin — скины чата
- /home/wwwchat/chat/php_sess — файлы сессий PHP-скриптов
- /home/wwwchat/chat/daemon — чат-сервер

Структура каталога `daemon`:

- daemon/db2 — база данных юзеров: пароли не шифруются!
- daemon/lib — библиотеки для windows-версии чат-сервера
- daemon/bin — библиотеки для unix-версии чат-сервера
- daemon/logs — протоколы чат-сервера
- daemon/www-chat — временные файлы PHP-скриптов
- daemon/www-conf — конфигурация чата
- daemon/www-server — временные файлы чат-сервера
- daemon/www-sess — временные файлы пользователей

Далее из соображений безопасности нужно сделать так, чтобы писать в подкаталоги `/home/wwwchat` могли только чат-сервер, запущенный от имени



http://all-ebooks.com

wwwchat, и веб-сервер Apache. Для этого создайте новую группу (допустим, grpchat), внесите в нее пользователей wwwchat и apache, затем смените группу владельца `/home/wwwchat` на эту группу и установите для каталога права доступа «drwxr-x---»:

```
chown -R wwwchat:grpchat /home/wwwchat
chmod 750 /home/wwwchat
```

Для работы чата нужен еще один интерпретатор – TCL. Работа под TCL значительно снижает загрузку памяти и увеличивает производительность. Пакет TCL можно скачать отсюда:

```
http://telia.dl.sourceforge.net/sourceforge/tcl/tcl8.3.4.tar.gz
```

или отсюда:

```
http://php.spb.ru/files/tcl8.3.4.tar.gz
```

Распаковать TCL можно в любой каталог, но собирать и устанавливать его нужно в каталог `/home/wwwchat/chat/daemon`. Выполните следующие команды:

```
tar -xvzf tcl8.3.4.tar.gz
cd tcl8.3.4/unix
./configure --prefix=/home/полный/путь/к/daemon --disable-shared
make
make install
```

Вам понадобится еще одна библиотека TCL: `libtbcload1.3.so`. Скачать ее можно отсюда: `http://php.spb.ru/files/libtbcload1.3.so.zip`. Не распаковывая скачанный файл, просто уберите расширение .zip и скопируйте файл в `/home/wwwchat/chat/daemon`.

Осталось настроить в Apache виртуальный веб-узел (гл. 4.6) для чата. Отредактируйте файл `/home/wwwchat/chat/www/__dima.php`:

```
$wwwhost = «значение директивы ServerName конфигурационного файла
Apache»;
$os = 1 // Linux.
$rand_password = «набор символов, могущий служить паролем»;
$cookie_pass = «набор символов, могущий служить паролем»;
```

Отредактируйте конфигурационный файл чата `/home/wwwchat/www/daemon/dima_chat.conf`:

```
$wwwhost = "значение переменной $wwwhost в файле __dima.php";
#set library "tbcload13.dll";  // закомментируйте эту строку: она
для Windows
set library "./libtbcload1.3.so";  // раскомментируйте эту строку
```

Далее установите для всех подкаталогов (рекурсивно) daemon и php_sess права доступа 777:

```
chmod -R 777 /home/wwwchat/chat/daemon
chmod -R 777 /home/wwwchat/chat/php_sess
```

Сценарии, запускающие чат, находятся в каталоге daemon и называются run_this и run_background. Если этих файлов там нет, создайте их. Они должны содержать следующие команды (соответственно):

```
bin/tclsh8.3 dima_chat.conf
```

и

```
bin/tclsh8.3 dima_chat.conf >/dev/null 2>&1 &
```

Если у вас установлен интерпретатор TCL другой версии, или в другое место, или вы вообще не root и пользуетесь TCL из общего каталога (обычно /usr/lib/tcl*), то вместо bin/tclsh8.3 укажите правильный путь к интерпретатору – относительный либо абсолютный.

Запустите сценарий run_this. Если вы увидите следующий вывод, значит, чат настроен правильно:

```
Config loaded. Your TCL version: 8.3. Wait   please. . .

[11:25]------------------------------
[11:25]            CHAT.PHP.SPB.RU, ver 4.06, 06 September 2002
[11:25]------------------------------
[11:25]Chat software (c) Dmitry Borodin, Russia, St.-Petersburg
[11:25]------------------------------
[11:25]Before using, please read licence on http://php.spb.ru/chat!
[11:25]--------------------------------
[11:25]PID: 10495, TCL Version: 8.3.4, hostname: localhost
[11:25]Listing localhost:8888  ...
[11:25]Encoding iso8859-1 ...
[11:25]Found charset for IRC-frame: windows-1251 .
[11:25]Searching old session file...
[11:25]Opening main chat file: www-chat/chat.txt ...
[11:25]ok, file clean
[11:25]*** LOADED (0.35 sec)  ***
```

Если вместо этого вы увидели сообщение об ошибке, прочитайте внимательно файл unix.html, который вы найдете в архиве вместе с исходниками чата. В этом файле установка под Linux описана очень подробно.



http://all-ebooks.com

# ГЛАВА 22.

## ГОСТЕВАЯ КНИГА

http://all-ebooks.com

## 22.1. СОЗДАЕМ САМИ ГОСТЕВУЮ КНИГУ

Гостевая книга — это еще один полезный инструмент веб-мастера, позволяющий ему узнать мнение посетителей сайта о своей работе, то есть о собственно сайте. В этой главе мы рассмотрим, как написать гостевую книгу своими руками и как прикрутить к сайту готовый сценарий гостевой книги. Какой из предложенных вариантов использовать, решайте сами. Ведь гостевая книга — это не чат и не форум, на самостоятельную разработку которого уйдет много времени. Написать неплохую гостевую книгу можно за пару часов, и вполне вероятно, что написанный вами скрипт будет лучше готового.

Рассмотрим алгоритм нашей гостевой книги. Пользователь открывает страницу гостевой книги и видит все сообщения. На этой же странице находится форма для ввода нового сообщения. Посетитель вводит сообщение и нажимает кнопку «Submit». Мы записываем сообщение в начало файла, чтобы пользователи видели последние сообщения сверху.

В файл гостевой книги будем записывать следующие данные:

- дату и время отправки сообщения;
- e-mail пользователя;
- само сообщение.

Перед записью сообщения нужно удалить из него все HTML-теги. Это не только позволит сэкономить дисковое пространство, но и увеличит безопасность нашей гостевой книги. Форматом файла гостевой книги будет следующий:


http://all-ebooks.com

```
[дата e-mail] сообщение
```

Можно было бы сохранять также IP-адрес пользователя, но лучше этого не делать, потому что по традиции сообщения в гостевой книге анонимны. Если вам все-таки интересен IP-адрес, то вы можете найти его в протоколах веб-сервера (по дате и времени) или записывать его в файл, но не отображать на странице сообщений. Эта операция несколько усложнит наш сценарий, поэтому мы будем просто игнорировать IP-адреса.

В каком виде отображать адрес электронной почты? У вас может возникнуть соблазн выводить e-mail как ссылку, т.е. в виде тега <a href="mailto:user@server.domain">. Ни в коем случае не делайте так! Существуют программы, которые просматривают содержимое веб-страниц в поисках тегов <a>, содержащих электронные адреса, и вносят эти адреса в спамерские базы. Вы же не хотите, чтобы пользователям, оставившим сообщения в вашей гостевой книге, приходил спам?

Выводить сообщения будем с помощью функции `view()`. Ее исходный код представлен в листинге 22.1.

**Листинг 22.1. Функция view()**

```
function view()
{
  $Messages=file('./gbook/gb.txt');

  echo "<p><table width=100%>";

  $i = 0;
  foreach($Messages as $v)
  {
    $i++;
    if ($i % 2 == 0) echo "<tr><td>$v</td></tr>";
    else echo "<tr><td bgcolor = gray>$v</td></tr>";
    if ($i==$Max) break;
  }
}
```

Сначала мы читаем весь файл `gb.txt` в массив `$Messages`. Потом обрабатываем первые `$Max` элементов этого массива, выводя каждый элемент в соответствующую ячейку таблицы из одного столбца. Для большего удобства чтения четные строки таблицы отображаются на сером фоне.

Ограничивать количество выводимых сообщений нужно потому, что гостевая книга — это не форум: она предназначена не для бесед посетителей друг с другом, а для отзывов о вашем сайте. Вполне достаточно показывать только последние, скажем, 50 сообщений: во-первых, это экономит трафик,

http://all-ebooks.com

а во-вторых, пользователи не увидят старых отзывов о давно исправленных вами недостатках сайта.

Если вас такой подход не устраивает, перепишите функцию `view()`, организовав постраничный вывод. Сделать это можно следующим образом: во-первых, заменить цикл `foreach` циклом с параметром `for`, который лучше приспособлен к обработке массива-списка, а во-вторых, запоминать в Cookies номер последнего сообщения (`$last`), просмотренного пользователем. Вывод будет выглядеть следующим образом:

```
for($i=$last; $i<count($Messages); $i++) echo
  "<tr><td>$Messages[$i]</td></tr>";
```

Вот теперь можно приступить к написанию сценария `gbook.php`. Код сценария приведен в листинге 22.2. Этот же сценарий можно скачать с моего сайта `http://www.dkws.org.ua`.

**Листинг 22.2. Сценарий гостевой книги**

```
<?
echo "<html><head><title>Гостевая книга</title></head><body>";
echo "<h1>Гостевая книга/h1>";

$file_gb="./gbook/gb.txt";          // файл гостевой книги
$file_tmp="./gbook/gb_tmp.txt";     // временный файл
$Max=50;                            // к-во выводимых сообщений

// Функция для вывода файла gb.txt
function view()
{
  $Messages=file($file_gb);
  echo "<p><table width=100%>";
  $i=0;
  foreach($Messages as $v)
  {
    $i++;
    if ($i % 2 == 0) echo "<tr><td>$v</td></tr>" ;
    else echo "<tr><td bgcolor=gray>$v</td></tr>" ;
    // выводим не более $Max сообщений
    if ($i==$Max) break;
  }
}

if (!isset($Post))
{
  // Выводим форму для ввода нового сообщения
  echo "<table width=100% border=1>";
  echo "<tr><td><h2>Новое сообщение</h2><form method=post
```



http://all-ebooks.com

```
    action=$SCRIPT_NAME>";
  echo "Email: <input type=text name=email>" ;
  echo "Сообщение: <input type=text name=mes> <input type=submit
    name = Post></td></tr>";
  echo "</table></form>";

  // Выводим сообщения
  view();
}
else
{
  if(file_exists($file_tmp)) die("fatal error!");

  // Добавляем новое сообщение в начало файла
  if(copy($file_gb, $file_tmp))
  {
    if($w=fopen($file_gb,"w"))
    {
      // блокируем доступ к основному файлу
      flock($w,2);

      // $mes - это сообщение пользователя
      // перед внесением сообщения в файл удалим из него HTML-теги

      fwrite($w,"[".date("d.m.Y H:i:s").$email."]".
        strip_tags($mes)."\n");

      if(!$r = fopen($file_tmp,"r")) die("can't open file");
      flock($r,1);  // блокируем доступ к временному файлу

      while($mes=fgets($r,10240))
      {
        fputs($w,$mes);
      }

      // Снимаем блокировку и закрываем файлы
      flock($r,3);
      fclose($r);
      flock($w,3);
      fclose($w);

      // Удаляем временный файл
      unlink($file_tmp);
    }
  }
  echo "Сообщение успешно записано в файл";
  // выводим файл сообщений
  view();
}
?>
```

http://all-ebooks.com

**Примечание.**

Для корректной работы этого сценария под Windows нужно закомментировать все вызовы функций блокировки доступа к файлу, поскольку эти функции в Windows не поддерживаются.

Сохраните сценарий 22.2 под именем `gbook.php` в каталоге `$DocumentRoot` (по умолчанию этим каталогом является `/var/www/html` в Linux и `c:\web5\html` — в Windows). Затем создайте в каталоге `$DocumentRoot` подкаталог `gbook` и назначьте ему права доступа, позволяющие всем пользователям читать каталог и записывать в него файлы. Без права просматривать оглавление каталога можно обойтись: мы и так знаем, что там находится единственный файл `gb.txt`. Создайте этот файл (пустой) и задайте для него права 0666 (разрешить чтение, запись, но запретить выполнение):

```
cd /var/www/html
mkdir gbook
chmod 666 gbook
cd /var/www/html/gbook
touch gb.txt
chmod 666 gb.txt
```

**Примечание.**

Если вы программируете под Windows, никакие манипуляции по изменению прав доступа производить не нужно.

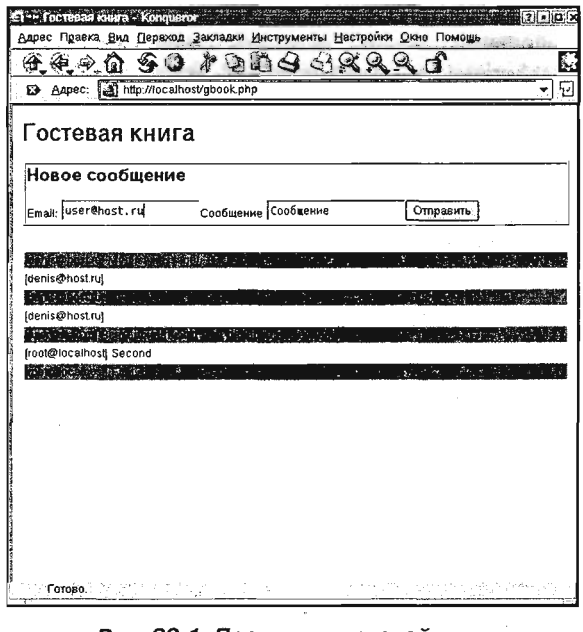



***Рис. 22.1. Просмотр гостевой книги***



http://all-ebooks.com

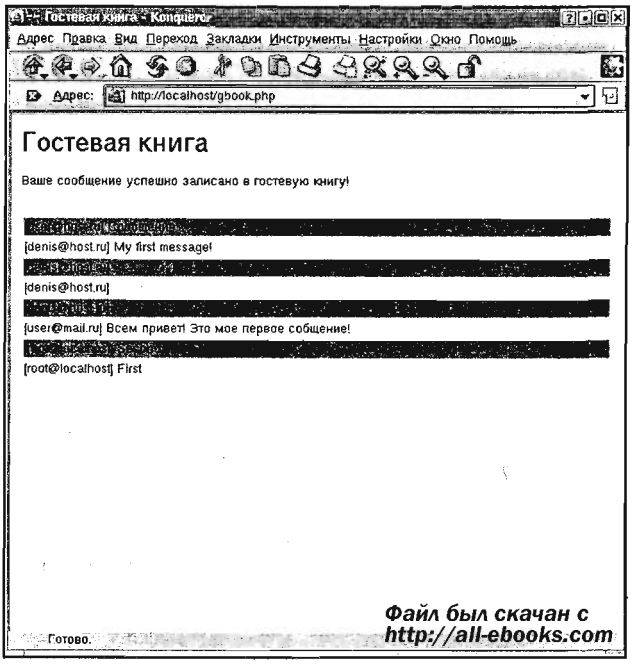



***Рис. 22.2. Сообщение добавлено***

Вот теперь можно запустить наш сценарий (см. рис. 22.1 и 22.2).

## 22.2. ИСПОЛЬЗУЕМ ГОТОВЫЙ СКРИПТ ГОСТЕВОЙ КНИГИ

В предыдущих изданиях этой книги я рассматривал сценарий гостевой книги GuestHere. Этот скрипт далек от совершенства, да еще и не русифицирован. В этой книге мы рассмотрим сценарий гостевой книги отечественного разработчика – PHP Guest Book 1.00. Вот особенности этой гостевой книги:

- хранение сообщений в базе данных MySQL;
- возможность администрирования гостевой книги;
- поддержка смайлов;
- русскоязычный интерфейс;
- анти-флуд (запрет пользователю оставлять больше одного сообщения за 10 минут).

http://all-ebooks.com

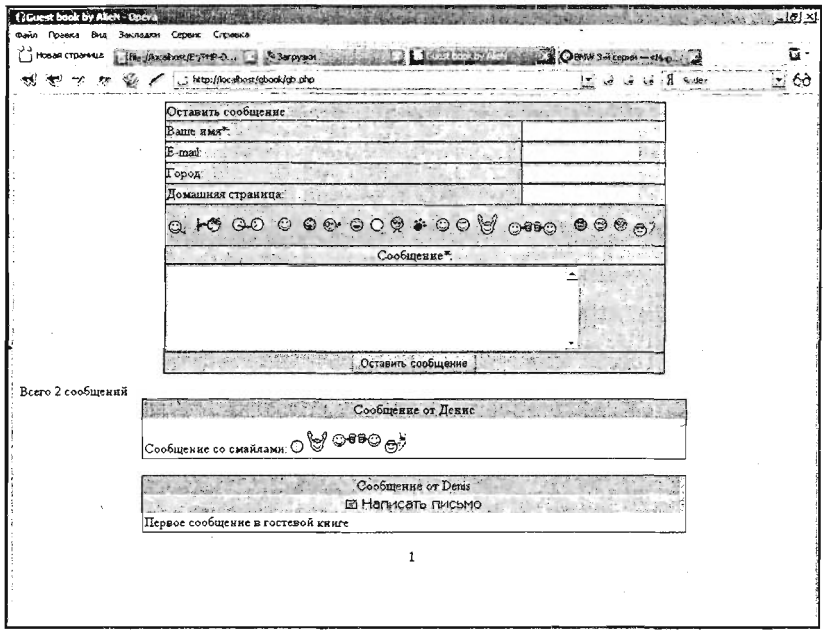



***Рис. 22.3. Гостевая книга в работе***

Этот сценарий можно скачать с моего сайта `http://dkws.org.ua` (раздел PHP) или найти на `http://softsearch.ru/programs`.

Итак, перейдем к установке гостевой книги. Убедитесь, что серверы Apache и MySQL запущены. Создайте подкаталог `gbook` в каталоге `$DocumentRoot` (`/var/www/html` в Linux или `c:\web5\www`, если вы скачали преднастроенный Apache с PHP5). Распакуйте архив с гостевой книгой в этот каталог.

В файле `dump.sql` находятся SQL-команды для создания таблицы `guestbook`, в которой будут храниться сообщения. Создавать отдельную базу данных ради одной таблицы не нужно – просто подключитесь к уже существующей базе данных и создайте в ней эту таблицу.

После создания таблицы нужно отредактировать файл `config.inc.php`. В этом файле нужно изменить параметры доступа к MySQL:

```
#### НАСТРОЙКИ АДМИНИСТРАТОРА
$admin["login"]="admin";  # ваш логин для администрирования гостевой
книги
$admin["password"]="123456";    # пароль администратора
$admin["email"]="ваш_email";
...
#### НАСТРОЙКИ БАЗЫ ДАННЫХ
```



http://all-ebooks.com

```
$mysqlhost="localhost";    # адрес узла, где установлена гостевая
книга
$mysqllogin="root";    # пользователь MySQL
$mysqlpassword="123456";    # пароль пользователя MySQL
$mysqldatabase="bp";     # имя базы данных, в которой находится
таблица guestbook
```

Чтобы войти в гостевую книгу, введите в адресную строку браузера

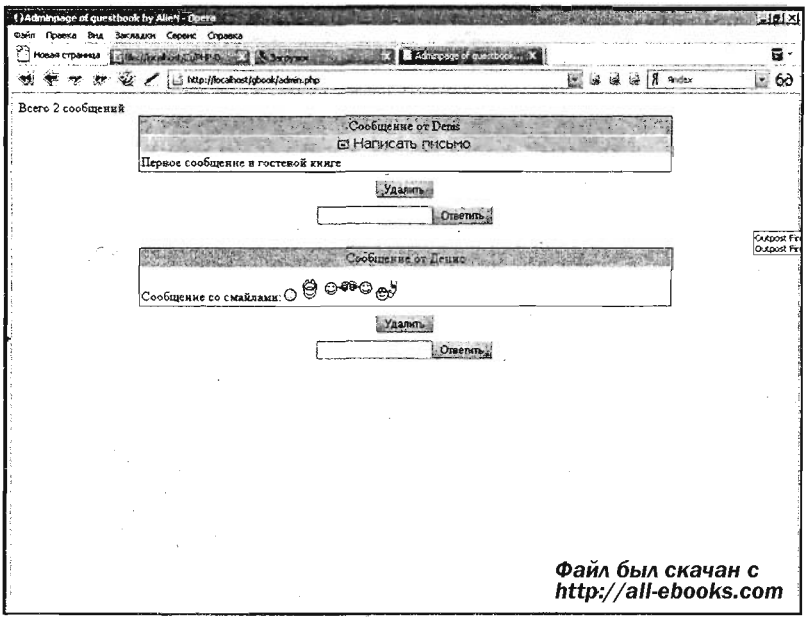
```
http://localhost/gbook/gb.php
```

Если вам не понравилось какое-то сообщение (например, оно содержит нецензурные выражения) или вы хотите ответить на него, войдите в админку гостевой книги:

```
http://localhost/gbook/admin.php
```

Здесь вы можете удалить любое сообщение или ответить на него (рис. 22.4).

Как видите, все очень просто: за пять минут мы получили довольно удобную и компактную гостевую книгу.




Файл был скачан с
http://all-ebooks.com

***Рис. 22.4. Администрирование гостевой книги***

# ГЛАВА 23.

# ЛЕНТА НОВОСТЕЙ

http://all-ebooks.com

## 23.1. ПРИНЦИП ОРГАНИЗАЦИИ НОВОСТНОЙ ЛЕНТЫ

Информационные сайты, порталы просто не могут обойтись без новостной ленты. Примеры новостных лент вы можете увидеть на многих информационных сайтах: `www.linux.ru`, `www.linuxrsp.ru`, `dkws.org.ua`. Обычно на главной странице отображается содержание новости в двух словах со ссылкой на страницу, где можно прочитать эту новость подробнее.

Новостная лента – это своеобразный симбиоз форума и системы голосования. Отличие от форума заключается в том, что новое сообщение (то есть новость) может добавить не кто угодно, а только администратор или лицо, отвечающее за наполнение новостной страницы. Рядовым пользователям, даже зарегистрированным, разрешается только оставлять свои комментарии к новостям. Строго говоря, любой форум можно превратить в новостную ленту, если разрешить создавать новые темы только модераторам.

Кроме функций форума, новостная лента выполняет функции системы голосования, так как обычно предоставляется возможность оценить материал.

Писать собственными руками новостную ленту мы не будем — зачем заново изобретать велосипед. Если же вас не устраивает готовая лента новостей, можно отредактировать «под себя» готовый продукт, а если и этот вариант вас не устраивает, тогда вам остается только одно — запастись терпением и писать систему самому.

http://all-ebooks.com

## 23.2. FUSION NEWS MANAGEMENT – ПРОФЕССИОНАЛЬНОЕ УПРАВЛЕНИЕ НОВОСТЯМИ

### 23.2.1. Устанавливаем скрипт

Если вы создаете информационный сайт, портал или просто ваш сайт часто обновляется и вы хотите сообщить пользователям, что же появилось нового, то вам не обойтись без ленты новостей. Как утверждает команда разработчиков FusionPHP, система Fusion News Management соответствует последним стандартам в области такого рода систем. Так почему же не использовать лучшее?

Система Fusion News Management предельно проста в эксплуатации, но, как оказалось, сложна в установке. Главным образом, сложность установки заключается в непродуманности сценария установки, предоставленного разработчиками системы. В принципе, ничего сложного в установке нет, но поработать руками (и головой!) все же придется. Систему можно взять на сайте `http://www.fusionphp.net` или на моем сайте: `http://www.dkws.org.ua/mybooks/php/news.zip`.

Итак, приступим к установке системы. Для начала распакуйте архив `news.zip` в каталог `/var/www/html/news`. В этом каталоге появятся два подкаталога — `images` и `upload`. Как сказано в руководстве по установке, нужно установить права 777 для каталога `upload` и запустить сценарий `upload/installer.php`. Скажу сразу, изменять права доступа пока не нужно — вы еще успеете это сделать, а сейчас запустите сценарий `installer.php`.

На первом этапе установки (рис. 23.1) сценарий предлагает нам изменить права доступа к подкаталогам каталога `upload` и файлам системы следующим образом:

- каталог `upload`: 777;
- подкаталоги каталога `upload`: 777;
- все файлы: 666;
- файлы `index.php`, `send.php`, `functions.php`, `comments.php`: 755.

**Примечание.**

Напомню, права доступа 777 для каталога разрешают чтение оглавления каталога и добавление файлов. Права доступа 666 для файла разрешают его чтение и запись, но запрещают выполнение. А права доступа 755 разрешают любые операции с файлом (в том числе и выполнение) владельцу файла, а всем остальным – чтение и запуск файла, но не его изменение. Честно говоря, сам не пойму, зачем нужно устанавливать право на выполнение для PHP-сценария, но без этого сценарий установки отказывается продолжить работу, поэтому нужно выполнить его требования.



http://all-ebooks.com

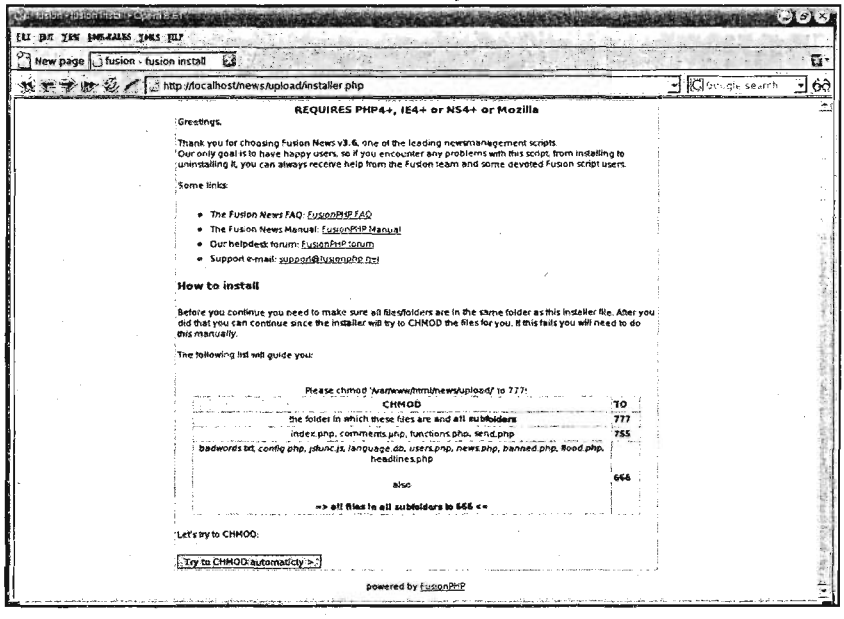



***Рис. 23.1. Первый шаг инсталлятора***

Попробуйте нажать на кнопку «Try to CHMOD automatically». Сценарий сам изменит права доступа согласно своим планам, но только в том случае, если ваша система настроена неправильно с точки зрения безопасности. Скорее всего, у вас ничего не выйдет, то есть вы не сможете установить систему автоматически, поскольку у вас нет полномочий для изменения прав доступа файлов, и вы увидите приблизительно то, что показано на рис. 23.2, — сообщение о необходимости изменения прав доступа.

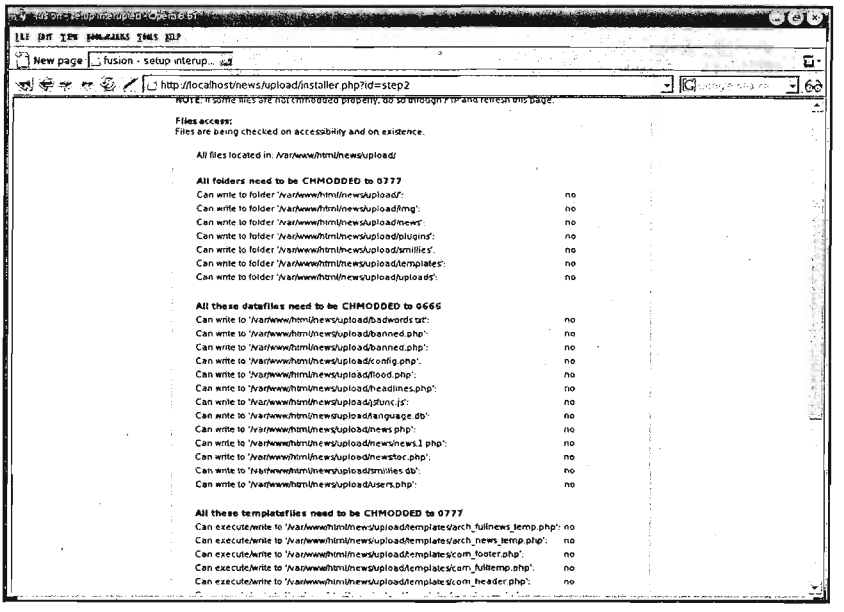
Будем изменять права доступа вручную, в соответствии с указанной выше последовательностью действий. Должен сказать, что указанная последовательность неправильна. Мы будем все делать в точности наоборот. Сначала мы изменим права доступа для всех файлов и каталогов каталога `upload`:

```
chmod -R 666 upload
```

Затем изменим права доступа (755) для файлов `index.php`, `functions.php`, `send.php` и `comments.php`. После этого установим права доступа 777 для всех подкаталогов каталога `upload` и для него самого. Правильная последовательность действий указана на рис. 23.3.

Теперь попробуем обновить страницу сценария `installer.php`. И что мы видим? Оказывается, нужно установить права доступа 755 не только для файлов `index.php`, `functions.php`, `send.php` и `comments.php`, но еще и для `archive.php`, `fullnews.php`, `plugins/check_chmod.php`, `search.php`, `upload.php`:

http://all-ebooks.com




*23.2. Неправильные права доступа - установка прервана*

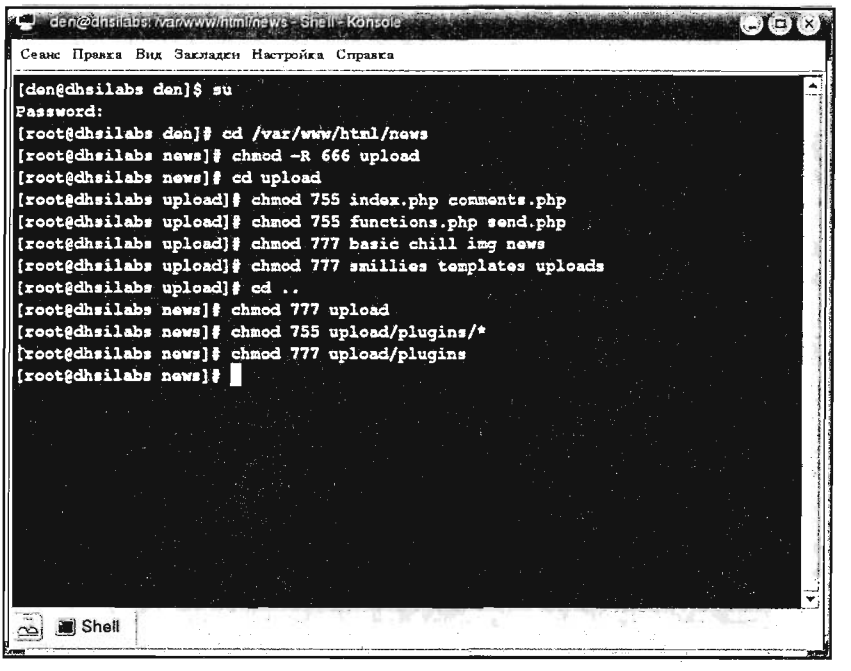



*Рис. 23.3. Изменение прав доступа*



http://all-ebooks.com

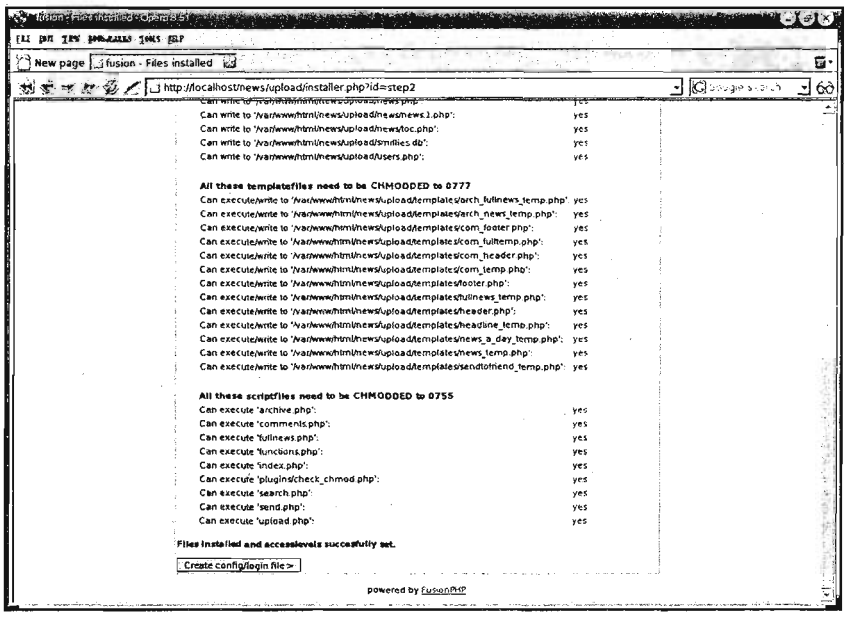



*Рис. 23.4. Права доступа установлены правильно*

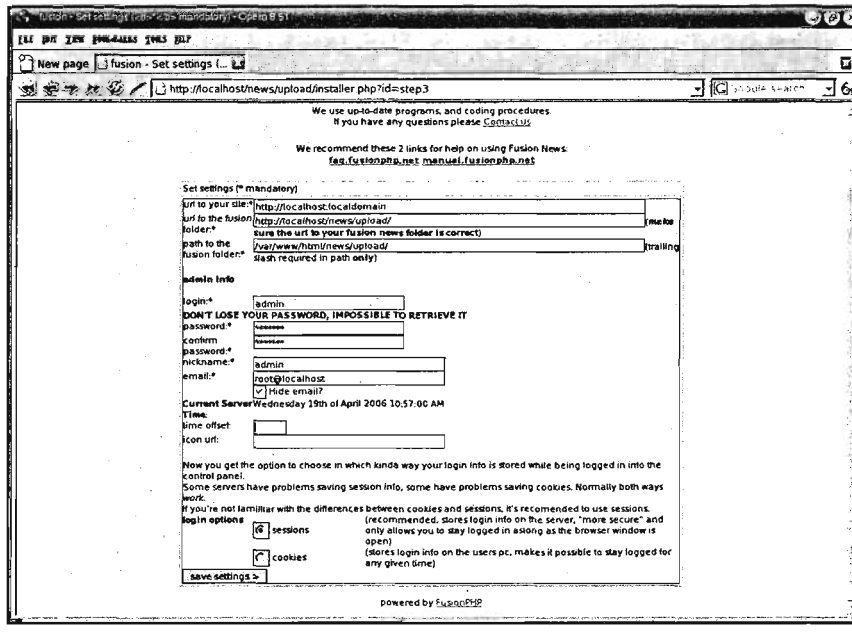



*Рис. 23.5. Ввод параметров системы*

http://all-ebooks.com

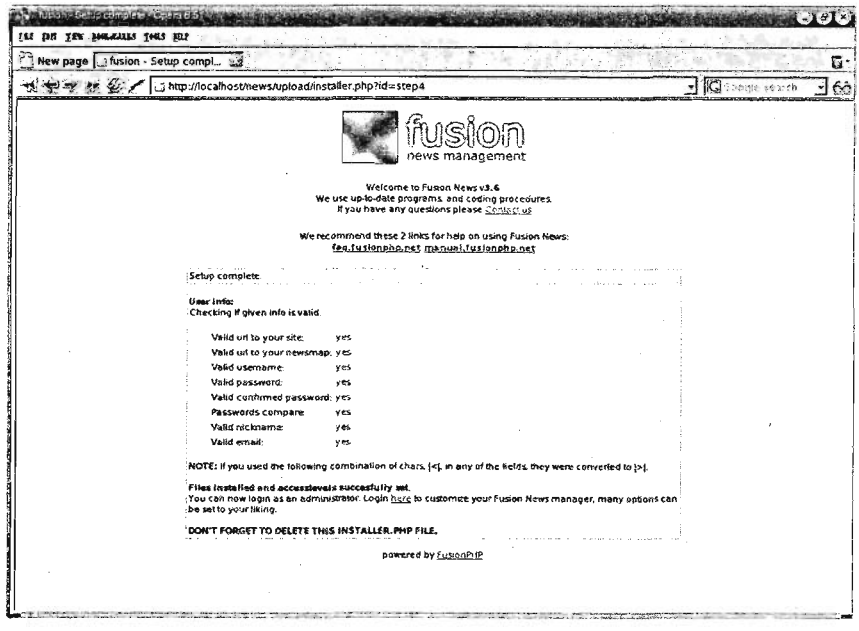



***Рис. 23.6. Установка завершена***

```
chmod 755 archive.php fullnews.php search.php upload.php
chmod 777 plugins
chmod 755 plugins/check_chmod.php
```

Вот теперь, после обновления содержимого окна браузера, сценарий предложит вам продолжить установку. Самое сложное уже сделано (рис. 23.4)!

Сейчас вам нужно ввести следующую информацию (рис. 23.5):

- URL вашего сайта, например `http://www.my_site.ru`.
- URL каталога Fusion News. В нашем случае это `http://www.my_site.ru/news/upload/`.
- Путь к каталогу Fusion News. У нас это `/var/www/html/news/upload/`.
- Логин администратора, например admin.
- Пароль и его подтверждение. Осторожно! Пароль восстановить нельзя.
- Ник администратора.
- E-mail администратора.



http://all-ebooks.com

Рекомендую использовать ник, отличный от логина администратора, а также установить режим «Hide email», чтобы никто не узнал вашего почтового адреса.

Последнее, что нужно сделать, - это выбрать режим работы системы — с использованием сессий или Cookies. Рекомендую использовать сессии, так как они более практичны. В этой книге мы еще поговорим о разнице между сессиями и Cookies.

Вот теперь установка завершена (рис. 23.6). Не забудьте удалить файл `installer.php`, чтобы злобные хакеры не взломали вашу систему путем изменения пароля администратора.

### 23.2.2. Работа со скриптом

Как я уже сказал, работать с системой чрезвычайно просто. В верхней части окна вы увидите восемь ссылок:

- main
- post news
- edit posts
- view news
- my webpage
- plugins
- help/update
- login

Для просмотра новостей нужно перейти по ссылке **view news**. Для добавления или редактирования новостей вам нужно сперва зарегистрироваться, а затем перейти по ссылке **post news** (добавить новость) или **edit posts** (редактировать новости).

Что вам еще нужно знать о системе? Во-первых, ее можно перевести на русский или любой другой язык. Для этого достаточно открыть файл `language.db` в любом текстовом редакторе и вручную перевести сообщения, выдаваемые системой, на требуемый язык. Во-вторых, все параметры, указанные вами при установке, находятся в файле `config.php`. У меня он выглядит так:

```
<?php
$fusion_id = "6LO9UY4844pN43fZb0vJ44fYF21WTcl9";
$cookies = " ";
$site = "http://localhost.localdomain";
$furl= "http://localhost/news/upload/";
$hurl = "http://localhost.localdomain";
$fpath = "/var/www/html/news/upload/";
$datefor = "m-d-Y H:i";
$numofposts = "10";
$numofh = "2";
$bb = "checked";
$ht = "";
$first = "no";
$post_per_day = "";
$wfpost = "checked";
$wfcom = "checked";

// Это - "скин". Параметры скина хранятся в одноименном
// каталоге (upload/chill)

$skin = "chill";
$cbwordwrap = "";
$wwwidth = "80";
$smilies = "checked";
$stfpop = "";
$comallowbr = "checked";
$stfwidth = "";
$stfheight = "";
$fslink = "full story";
$stflink = "send to friend";
$pclink = "comnents";
$fsnw = "";
$cbflood = "";
$floodtime = "30";
$comlength = "300";
$fullnewsw = "";
$fullnewsh = "";
$fullnewss = "";
$stfresize = "";
$stfscrolls = "";
$fullnewsz = "";
$htc = "";
$smilcom = "checked";
$bbc= "";
$compop= "";
$comscrolls= "";
$comresize= "";
$comheight= "";
$comwidth= "";
```

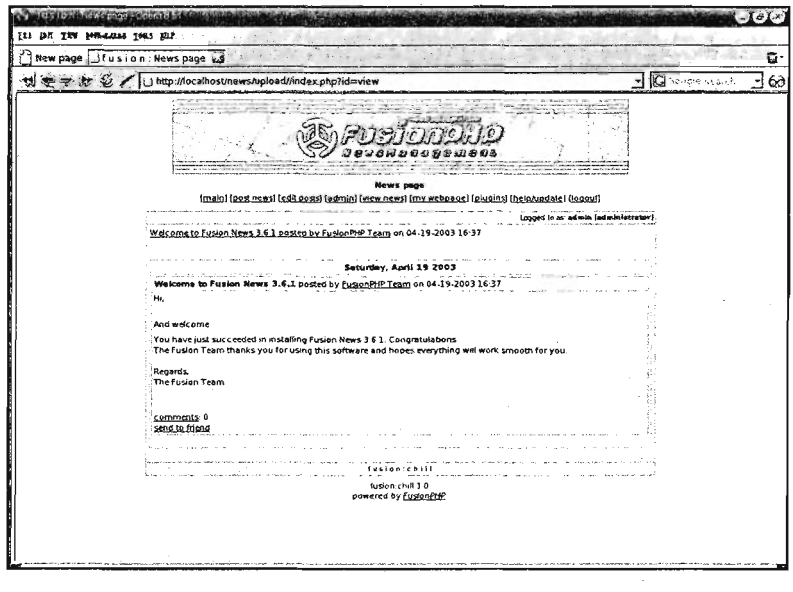


http://all-ebooks.com

```
$uploads_active= "checked";
$uploads_size= "1048576";
$uploads_ext= "gif&#124;jpg&#12 4;jpeg&#12 4;png"
$enable_rss = "checked";
$link_headline_fullstory = "";
$flip_news = "";
?>
```

Наверное, вам будет интересно, где же хранятся сами новости. Они находятся в каталоге `upload/news`, в файлах `toc.php` (оглавление, table of contents) и `news.x.php` (x – номер соответствующей новости). При желании можно редактировать эти файлы вручную — формат их предельно прост.

Пример использования новостной ленты Fusion News показан на рис. 23.7.




***23.7. Пример использования новостной ленты Fusion News***

# ГЛАВА 24.

# ГОЛОСОВАНИЕ НА САЙТЕ

http://all-ebooks.com

## 24.1. ПИШЕМ ОПРОС САМИ

Вы хотите знать, какой дизайн сайта больше нравится вашим пользователям — старый, новый или оба неудачны? А может быть, вам интересно знать, какую операционную систему предпочитают пользователи? Тогда вам не обойтись без голосования, или опроса пользователей сайта.

Сейчас мы напишем небольшой сценарий, позволяющий оценить дизайн нашего сайта. Вариантами ответов будут:

- Новый
- Старый
- Никакой
- Мне все равно

Наша система обработки опроса будет состоять из трех файлов:

- `vote.php` — обрабатывает ответ пользователя и записывает его в файл `vote.txt`;
- `vote.txt` — здесь будут храниться ответы пользователей;
- `vote.html` — форма, содержащая варианты ответов.

Начнем с самого простого — напишем форму с вариантами ответов (листинг 24.1).

http://all-ebooks.com

**Листинг 24.1. Файл vote.html**

```
<FORM METHOD="POST" action="vote.php">
<TABLE border="2" cellspacing="0" cellpadding="3"
bordercolor="#336699">
<TR><TD><TABLE BORDER=0>
<TR><TD bgcolor="#336699" font color=white>Какой дизайн вам больше
нравится?</font></TD></TR>
<TR><TD><INPUT TYPE=radio NAME=answer VALUE =1>Новый</TD></TR>
<TR><TD><INPUT TYPE=radio NAME=answer VALUE =2>Старый</TD></TR>
<TR><TD><INPUT TYPE=radio NAME=answer VALUE =3>Никакой</TD></TR>
<TR><TD><INPUT TYPE=radio NAME=answer VALUE =4>Мне все равно</TD>
</TR>
<TR><TD><INPUT TYPE=Submit NAME=vote VALUE="Голосовать!"></TD>
</TR>
<TR><TD><INPUT TYPE=Submit NAME= result VALUE="Результат"></TD>
</TR>
</TABLE></TD></TR></TABLE></FORM>
```

Далее создайте файл `vote.txt` следующего формата:

```
Результаты
0
0
0
0
```

Первая строка файла `vote.txt` – это просто комментарий, а следующие содержат количества голосов за варианты 1, 2, 3 и 4 (начальные значения – нули).

Вот теперь мы готовы написать сценарий, обрабатывающий ответы пользователей (листинг 24.2).

**Листинг 24.2. Сценарий vote.php**

```
<?
$file = "vote.txt"; // файл результатов
$data = file($file);
$i = 1;

// Количество строк в файле
$qty = count($data);
$n = 0;

// альтернативный синтаксис цикла (см. п. 5.7.2)
while ($i <= $qty):
```



http://all-ebooks.com

```
  $data[$i] = trim(str_replace("\n","", $data[$i]));
  $n = $n + $data[$i];
  $i++;
endwhile;

if ($answer != "") {
  echo "<br>Спасибо!";
  $data[$answer]++; $n++;

$res = "Результаты\n".$data[1]."\n".$data[2]."\n".$data[3]."\n".$data[4];
  $fp = @fopen($file,"w");

  if ($fp) { $counter=fputs($fp,$res); fclose($fp); }
  else { echo "Ошибка записи в файл"; }

} else { echo "<br>Результаты голосования"; }

echo "<br>Новый - <b>".$data[1]."</b>";
echo "<br>Старый - <b>".$data[2]."</b>";
echo "<br>Никакой - <b>".$data[3]."</b>";
echo "<br>Мне все равно - <b>".$data[4]."</b>";
echo "<br><br>Всего голосов: ".$n;

?>
```

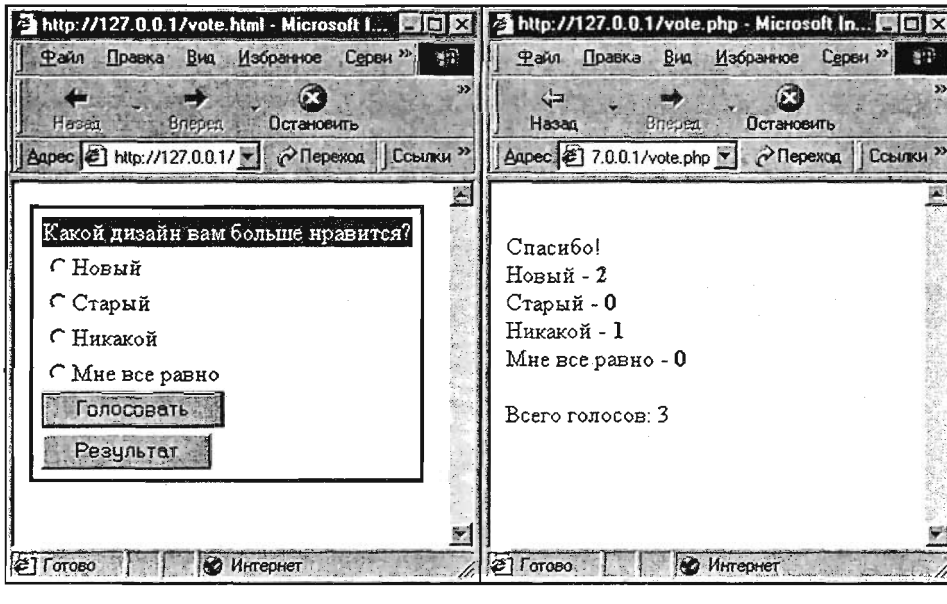



*Рис. 24.1. Сценарий голосования vote.php в работе*

В следующем пункте мы рассмотрим целую систему голосований, позволяющую обрабатывать не один опрос, а несколько.

## 24.2. ПРОФЕССИОНАЛЬНЫЙ СЦЕНАРИЙ ADVANCED POLL

### 24.2.1. Общее описание и установка Advanced Poll

Лучшим скриптом для голосования, на мой взгляд, является Advanced Poll. Этот сценарий позволяет создавать голосования и централизованно управлять ими. Поддерживается хранение результатов как в базе данных MySQL, так и в обычном текстовом файле на случай, если ваш хостинг-провайдер по тем или иным причинам не позволяет вам использовать MySQL.

Скрипт можно скачать с сайта `www.ishodniki.ru`. Распакуйте его в какой-нибудь локальный каталог. В архиве будет два каталога — `db` и `textfile`. Первый из них содержит версию скрипта с поддержкой MySQL, а второй – версию, использующую текстовый файл. В этой главе мы рассмотрим именно второй вариант, так как он немного проще в установке на сервер и не требует наличия сервера баз данных.

Скопируйте содержимое каталога `textfile` в каталог `$DocumentRoot/poll` (`/var/www/html/poll` под Linux, `c:\web5\www\poll` под Windows, если вы используете преднастроенную версию PHP5). В каталоге `$DocumentRoot/poll` окажутся следующие подкаталоги:

- `admin` — средства управления системой;
- `image` — картинки;
- `include` — файлы вложений;
- `lang` — настройки языка;
- `polldata` — данные опросов;
- `templates` — шаблоны;
- `misc` — в текстовой версии не используется.

После этого от вас потребуется выполнить пару несложных трюков, чтобы заставить систему работать на вашем сервере. Первое, что нужно сделать, — изменить права доступа к файлам и каталогам.

Установите права доступа для каталога `/var/www/html/poll/polldata` равными 777 (разрешены чтение, запись и просмотр каталога всем пользователям). Для всех файлов, находящихся в этом каталоге, установите права



http://all-ebooks.com

доступа 666 (разрешает чтение и запись, но запрещает выполнение). После этого установите права 777 для каталога `templates` и для всех его подкаталогов, а для файлов в `templates` и подкаталогов – 666.

Для каталога `include` установите права 755 (владельцу разрешено чтение, запись и просмотр каталога, всем остальным — только чтение и просмотр), а для файла `include/config.inc.php` — 666.

Теперь откройте файл `config.inc.php` и отредактируйте его так:

```
$pollvars['base_url'] = "/poll";
$pollvars['base_gif'] = "/poll/image";
```

Первая переменная указывает путь к каталогу с системой голосования относительно корневого каталога WWW-сервера, а вторая – путь к каталогу с картинками.

Вот и все! Теперь можно приступать к настройке голосования.

## 24.2.2. Настройка Advanced Poll. Центр управления

Чтобы попасть в центр управления голосованиями, введите в адресную строку браузера: `http://localhost/poll/admin/index.php`. Войдите в систему под именем «admin» с паролем «poll». Вы увидите окно, изображенное на рис. 24.2. Управление голосованиями производится с помощью инструментальных кнопок в верхней части окна:

- **Poll List** — список голосований;
- **Create a new poll** — создать новое голосование;
- **General Settings** — настройка;
- **Templates** — управление шаблонами;

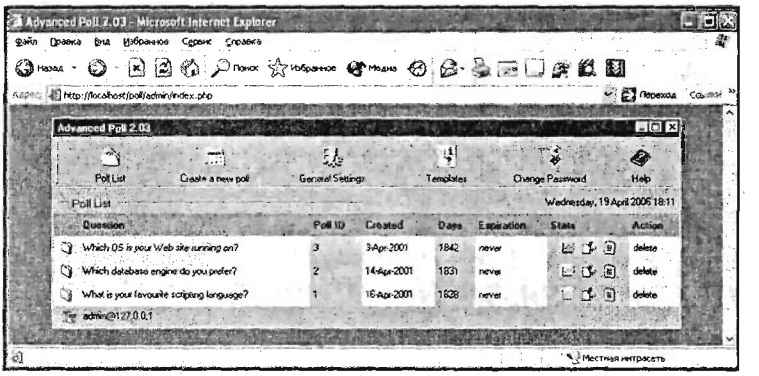



*Рис. 24.2. Центр управления голосованиями*

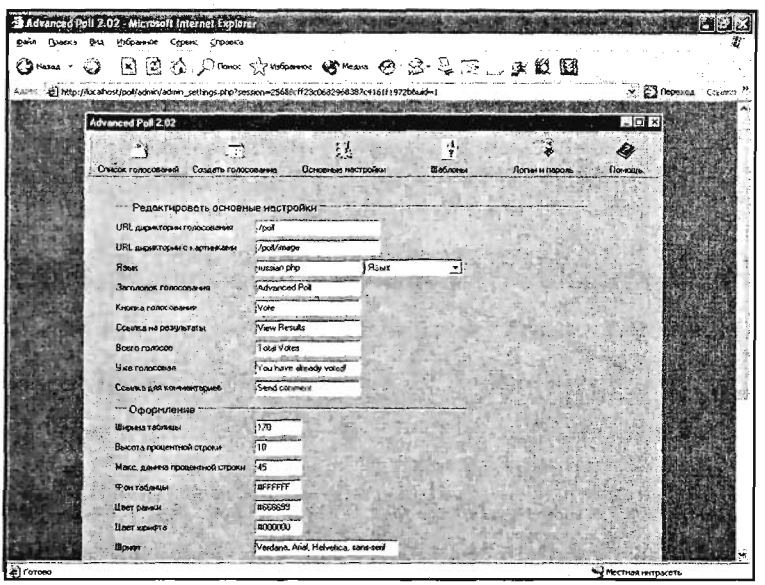



***Рис. 24.3. Раздел General Settings (Основные настройки)***

- **Change password** — изменение пароля администратора;
- **Help** — вызов справки.

Первым делом измените пароль администратора (по умолчанию «poll»). Далее войдите в раздел **General Settings** (рис. 24.3) и выберите русский язык. Потом отредактируйте конфигурационный файл `include/config.inc.php`, как показано в листинге 24.3.

**Листинг 24.3. Файл include/config.inc.php**

```
<?php
$pollvars['base_url'] = "/poll";
$pollvars['base_gif'] = "/poll/image";
$pollvars['lang'] = "russian.php";
$pollvars['title'] = "Advanced Poll";
$pollvars['vote_button'] = "Проголосовать";
$pollvars['result_text'] = "Результаты";
$pollvars['total_text'] = "Всего";
$pollvars['voted'] = "Вы уже голосовали!";
$pollvars['send_com'] = "Ваш комментарий";
...
?>
```



http://all-ebooks.com

### 24.2.3. Создание и редактирование опроса

Вот только теперь мы можем приступить к созданию опросов. Но перед этим рассмотрим список параметров для каждого опроса (раздел **Poll List**, рис. 24.2):

- **Вопрос (Question)** — вопрос для голосования (тема обсуждения);
- **ID голосования (Poll ID)** — идентификатор, который нужно указывать при обработке отдельного опроса;
- **Создано (Created)** — дата создания опроса;
- **Дней (Days)** — сколько дней прошло с момента создания;
- **Истечение (Expiration)** — дата окончания голосования;
- **Статистика (Stats)** — всевозможная статистика;
- **Действие (Action)** — обычно в этой колонке стоит действие удалить (delete), позволяющее удалить голосование

Вот теперь нажмите кнопку **Создать голосование (Create a new poll)**. На рис. 24.4 представлена форма, которую нужно заполнить. Все просто: введите вопрос и до 10 вариантов ответа на него. Можно включить протоколирование (**log**), установить начальное значение для каждого варианта ответа, а в нижней части окна установить общие параметры опроса:

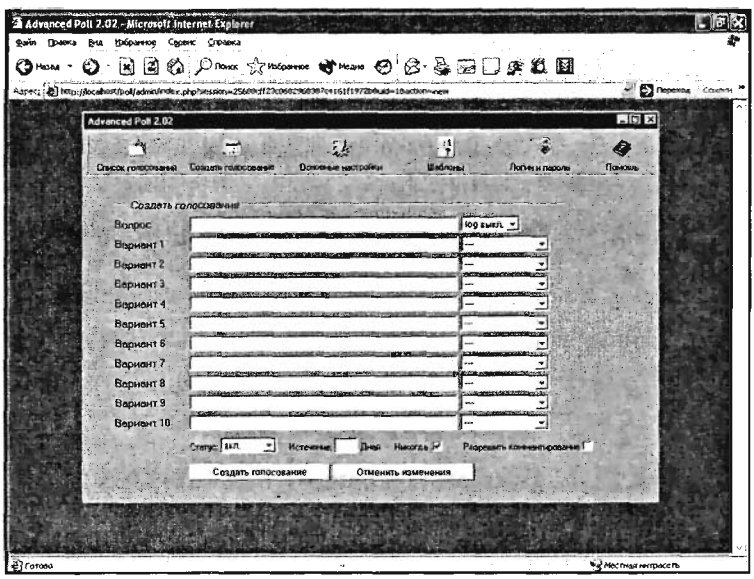



*Рис. 24.4. Создание опроса*

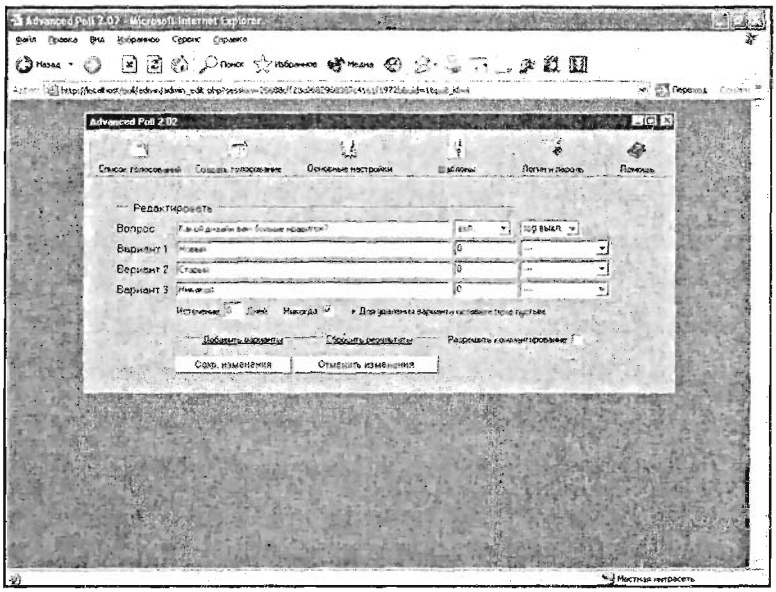



***Рис. 24.5. Редактирование опроса***

- активен ли он (**Статус enabled/disabled**), то есть отображается ли этот опрос в списке голосований;
- через сколько дней опрос перестанет быть активным (**Истечение, Expiration Days** или флажок **Никогда (Never)**);
- разрешается ли пользователям оставлять комментарии (**Allow comments**).

Заполнив форму, нажмите кнопку **Создать голосование** (**Create a new poll**), и новый опрос появится в списке голосований.

Для редактирования существующего опроса щелкните по нему в списке голосований. Появится форма, изображенная на рис. 24.5. Чтобы добавить дополнительные варианты ответа, щелкните по ссылке **Добавить варианты** (**Add options**). Чтобы удалить один из вариантов, просто очистите соответствующее ему поле ввода.

### 24.2.4. Вывод опроса на веб-страницу

Наше голосование создано. Что делать дальше? Как добавить форму голосования на наш сайт? Я советую вам подробно изучить сценарии `textfile/demo*.php`, в которых вы найдете нужные фрагменты кода.



http://all-ebooks.com

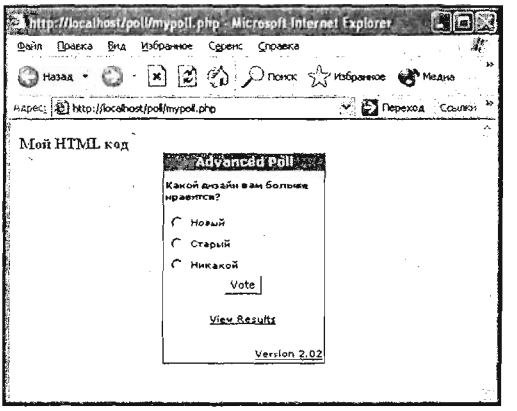



***Рис. 24.6. Результат работы сценария 24.4***

Сейчас мы напишем простенький сценарий (листинг 24.4), который будет отображать один из созданных вами опросов. Загляните в центр управления, чтобы узнать идентификатор этого опроса (**Poll ID**), и присвойте это значение переменной `$my_poll_id`.

**Листинг 24.4. Размещение опроса на веб-странице**

```
<?
/* Подключаем систему Poll ПЕРЕД вашим HTML-кодом */
include "./poll_cookie.php";
?>
<b>Мой HTML код</b>
<?

/* путь к каталогу, в котором установлена система */
$poll_path = "c:/web5/www/poll";
// идентификатор опроса
$my_poll_id=4;

// подключаем файл конфигурации и файл класса
require $poll_path."/include/config.inc.php";
require $poll_path."/include/class_poll.php";
$php_poll = new poll();

/* простое голосование */
echo "<center>";
$php_poll->set_template_set("simple");
echo $php_poll->poll_process($my_poll_id);
?>
```

Обратите внимание на строку

```
$php_poll->set_template_set("simple");
```

Перед тем, как отобразить опрос вызовом метода `poll_process()`, нужно указать, как оформить выводимый опрос. Это делается выбором шаблона: `"default"`, `"simple"`, `"popup"` и т.п. Доступные шаблоны находятся в каталоге `templates`, а их имена совпадают с именами подкаталогов этого каталога.

### 24.2.5. Вывод нескольких опросов на одной странице

Если вы хотите отобразить на одной странице несколько голосований — нет проблем: нужно просто несколько раз вызвать метод `poll_process()`:

```
$php_poll->set_template_set("simple");
echo $php_poll->poll_process(4);
echo $php_poll->poll_process(1);
echo $php_poll->poll_process(2);
```

Чтобы получить список доступных опросов, используйте класс `plist` вместо класса `poll`:

```
require $poll_path."/include/config.inc.php";
require $poll_path."/include/class_poll.php";
require $poll_path."/include/class_pollcomment.php";
require $poll_path."/include/class_plist.php";

$php_poll = new plist();

$php_poll->set_template("poll_list");
$php_poll->set_date_format("m/d/Y");
echo $php_poll->view_poll_list() ;
echo $php_poll->get_list_pages() ;
```

Сценарий 24.5 выводит на страницу три опроса в порядке слева направо, а под ними — список доступных голосований. Результат показан на рис. 24.7.

**Листинг 24.5. Вывод нескольких опросов**

```
<?
$poll_path = "c:/web5/www/poll";
require $poll_path."/include/config.inc.php";
require $poll_path."/include/class_poll.php";
require $poll_path."/include/class_pollcomment.php";
require $poll_path."/include/class_plist.php";
$php_poll = new plist();
/* первое */
echo "<center><TABLE><TR><TD valign=top>";
echo $php_poll->poll_process(4);
```



http://all-ebooks.com

```
echo "</TD><TD valign=top>";
/* второе */
$php_poll->set_max_bar_length(80);
echo $php_poll->poll_process(1);
echo "</TD><TD valign=top>";
/* третье */
$php_poll->set_template_set("popup");
if ($php_poll->is_valid_poll_id(2)) {
  echo $php_poll->display_poll(2);
}
echo "</TD></TR></TABLE>";

// шаблон оформления
$php_poll->set_template("poll_list");

// формат даты
$php_poll->set_date_format("m/d/Y");
echo $php_poll->view_poll_list();
echo $php_poll->get_list_pages();
?>
```

И последнее. Если вы хотите заменить стандартный заголовок «Advanced Poll» на что-нибудь русскоязычное, откройте файл `include/config.inc.php` и отредактируйте строку:

```
$pollvars['title1] = "Advanced Poll";
```

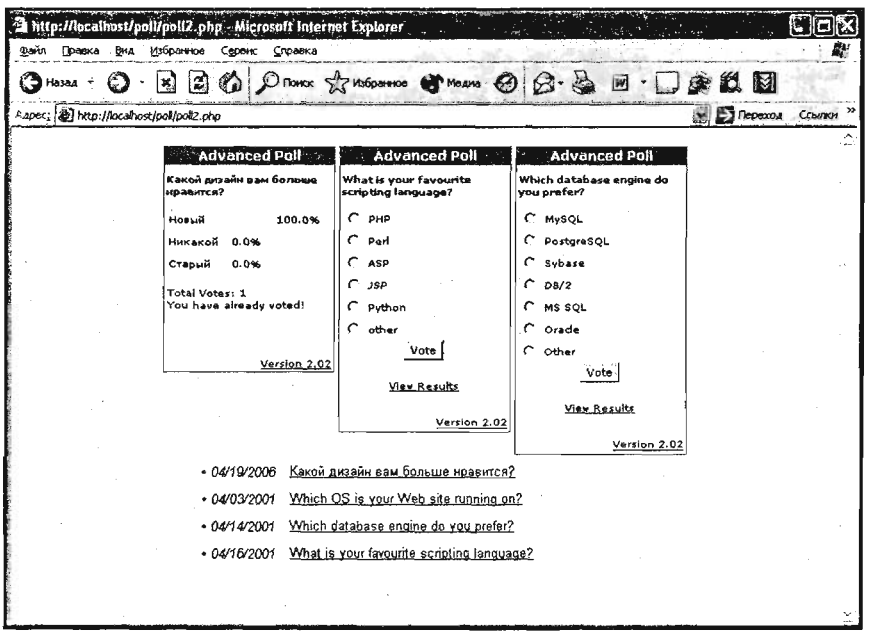



***Рис. 24.7. Вывод нескольких опросов***

# ГЛАВА 25.

# СЧЕТЧИК И АНАЛИЗ СЕРВЕРНОЙ СТАТИСТИКИ

http://all-ebooks.com

## 25.1. СЧЕТЧИК И ПРОТОКОЛИРОВАНИЕ ПОСЕЩЕНИЙ НА САЙТЕ

### 25.1.1. Простой счетчик

Нащей целью в этом параграфе будет разработка простого счетчика. Предлагаемое мною решение не претендует на универсальность, но гарантирует правильную работу счетчика, то есть то, что счетчик не будет реагировать на обновление страницы в браузере. Далее в этой главе мы рассмотрим более сложный счетчик, ведущий протокол посещений сайта.

Определим требования к нашему счетчику:

- Он должен подсчитывать только уникальных посетителей.
- Счетчик не должен реагировать на нажатие кнопки «Обновить» в браузере.
- Счетчик не должен вести учет наших собственных посещений (с локального хоста).

Считать уникальных посетителей можно по-разному. Узлы в Интернете идентифицируются по IP-адресам, поэтому первое, что приходит в голову, — записывать все уникальные IP-адреса в какой-нибудь файл (или базу данных). Но такой метод нам совершенно не подходит.

Во-первых, IP-адрес может быть динамическим: тот же самый пользователь может зайти на сайт во второй раз с другого IP-адреса. В этом случае не выполняется первое требование, предъявленное к нашему счетчику: уникальность посетителя.

http://all-ebooks.com

Во-вторых, поиск в файле – операция довольно медленная, не говоря уже о записи в файл (с обязательной блокировкой, поскольку нужно учитывать, что к нашему сайту может одновременно обращаться несколько пользователей).

В-третьих, рано или поздно размер нашего файла станет неприлично большим, а если хранить IP-адреса в базе данных, то наш счетчик уже нельзя будет назвать простым.

Значит, нам придется использовать Cookies. Алгоритм работы счетчика в этом случае будет таким: когда пользователь заходит на ваш сайт, сценарий проверяет, установлены ли у него определенные (наши) Cookies. Если да, то возвращается предыдущее значение счетчика, а если нет, то сценарий считает посетителя уникальным, увеличивает счетчик и «помечает» пользователя, устанавливая для него Cookie.

Наш сценарий (листинг 25.1) действительно прост: он содержит всего 17 строк кода.

**Листинг 25.1. Простой счетчик**

```
<?
$f = fopen("counter.dat", "a+") or die("Cannot open file");
flock($f,2);
$count=fread($f,100);
if(!isSet($Visited))
{
  $Visited =1;
  SetCookie("Visited",$Visited,time()+3600*24);
  // Не засчитываем посещения со своего узла
  $REMOTE_HOST=gethostbyaddr($REMOTE_ADDR);
  if(!($HTTP_HOST===$REMOTE_HOST))
    @$count=$count+1;
  ftruncate($f,0);
  fwrite($f,$count);
}
flock($f,3);
fclose($f);
echo $count;
?>
```

Нужно ли комментировать этот сценарий? Разве что вкратце. Значение счетчика мы храним в файле `counter.dat`, который открываем в режиме «a+», то есть для чтения и записи, причем указатель устанавливается на конец файла; в отличие от режима «a», если указанного файла не существовало, он будет создан.



http://all-ebooks.com

Затем мы блокируем файл. Последний параметр функции `flock()` определяет вид блокировки, в нашем случае это режим LOCKEX — исключительная блокировка: любому другому процессу, желающему открыть файл, придется ждать, пока мы завершим работу с файлом. Возможно, при большой нагрузке на сервер уместнее была бы разделяемая блокировка, но пока нам это не нужно.

Далее мы проверяем, существует ли Cookie: для этого достаточно проверить существование переменной `$Visited` с помощью функции `isSet()`. Если нет, мы создаем Cookie с именем `"Visited"` и временем существования одни сутки и записываем в него только что присвоенное значение переменной `$Visited`.

Имя узла, с которого пришел посетитель, мы проверяем вызовом функции `gethostbyaddr($REMOTE_ADDR)` и сравниваем с именем локального узла, хранящимся в переменной окружения `$HTTP_HOST`.

Наконец, мы записываем в файл новое значение счетчика `$count`. Разумеется, если значение не изменилось, то можно обойтись и без его записи: можете доработать это в качестве домашнего задания.

Чтобы отлаживать этот счетчик на своем локальном компьютере, закомментируйте проверку имени узла. При первом запуске сценария вы должны увидеть в окне браузера «1». При повторном посещении или нажатии на кнопку «Обновить» (в т.ч. и на «Shift» + «Обновить») значение счетчика не изменится.

Если ваш браузер – Opera, то вы можете просмотреть установленные Cookies, выбрав в меню **Сервис→Настройки→Cookies→Управление Cookies**. В Internet Explorer можно только удалить установленные Cookies, поэтому, чтобы значение счетчика увеличилось еще на единицу, выполните команду **Сервис→Свойства обозревателя→Общие→Удалить Cookie**. После этого счетчик засчитает вам еще одно посещение.

Наш счетчик страдает недостатком, общим для всех сценариев, основанных на Cookies: он бессилен против пользователей, заходящих на сайт с отключенными Cookies. Счетчик воспримет каждое такое посещение как уникальное.

### 25.1.2. Графический счетчик

Сейчас мы немного усложним наш счетчик. Дело в том, что текстовые счетчики в наше время не использует практически никто: обычно выводится какое-нибудь графическое изображение. Вот и заменим вывод результата при

http://all-ebooks.com

помощи вызова `echo $count` на вывод рисунка, содержащего результат (листинг 25.2).

**Листинг 25.2. Графический счетчик**

```
<?
$f = fopen("counter.dat","a+ ") or die("Cannot open file");
flock($f,2);
$count=fread($f,100);
if(!isSet($Visited))
{
  $Visited =1;
  SetCookie("Visited",$Visited,time()+3600*24);
  // Не засчитываем посещения со своего узла
  $REMOTE_HOST=gethostbyaddr($REMOTE_ADDR);
  if(!($HTTP_HOST===$REMOTE_HOST))
    @$count=$count+1;
    ftruncate($f,0);
    fwrite($f,$count);
}

flock($f,3);
fclose($f);

$img = ImageCreateFromPng("images/template.png");
$color = ImageColorAllocate($img, 0, 0, 255);
ImageString($img,3,5,9,"Visits : $count",$color);
Header("Content-type: image/png");
ImagePng($img);
ImageDestroy($img);
?>
```

Как видите, все очень просто: мы читаем фоновый рисунок счетчика из файла `images/template.png` (позаботьтесь заранее о его существовании). Далее мы указываем, что буквы будем выводить синим цветом, формируем и выводим строку, передаем браузеру тип изображения (PNG), выводим полученное изображение и освобождаем память, выделенную под рисунок.

Чтобы встроить этот счетчик в веб-страницу, укажите тег `<img src="counter.php">`.

Вы запустили этот сценарий и получили сообщение об ошибке: «Call to undefined function: imagecreatefrompng()...»? Значит, у вас не установлена библиотека GD, необходимая для работы с графикой. Если вы работаете под Windows, убедитесь, что у вас есть файл `php_gd.dll` и он прописан в `php.ini` (см. гл.13). В современных дистрибутивах Linux библиотека GD обычно присутствует, и PHP собран с ее поддержкой.



http://all-ebooks.com

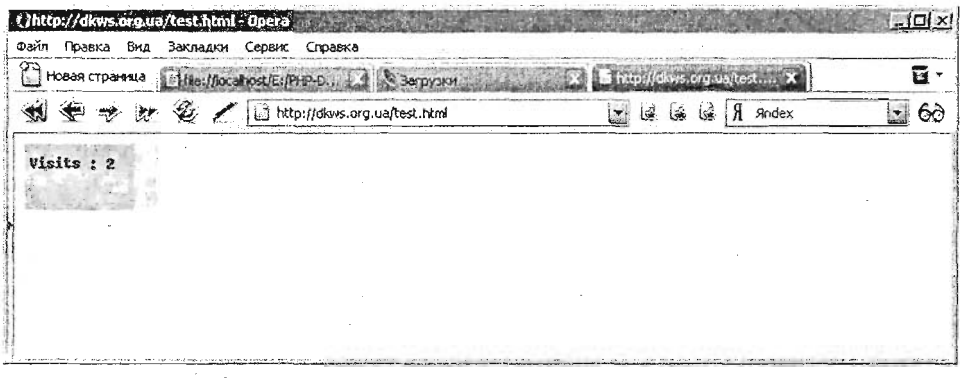



*Рис.25.1. Графический счетчик*

А вот у вашего хостинг-провайдера библиотеки GD может и не быть, так что, даже если сценарий работает на вашей локальной машине, это не гарантирует, что он заработает и на сайте.

Если вы настроили PHP на работу с графикой и все равно получаете сообщение об отсутствии функции `ImageCreateFromPng()`, попробуйте преобразовать файл рисунка в другой формат и применить функции `ImageCreateFromJpeg()` или `ImageCreateFromGif()`. Разумеется, Content-type в заголовке нужно тоже заменить на правильный тип, а для вывода изображения использовать функцию `ImageJpeg()` или `ImageGif()` соответственно.

### 25.1.3. Протоколирование посещений сайта

Допустим, мы хотим вести статистику посещений сайта: кто, когда, с каким браузером. Для этого посещения нужно протоколировать. Хранить полученные сведения мы будем в базе данных MySQL.

Есть два подхода к протоколированию: простой и правильный. Первый заключается в том, что мы сохраняем данные обо всех посетителях сайта подряд, не заботясь об их уникальности. Второй путь состоит в том, что мы вносим в базу только уникальных пользователей. Как вы догадались, мы пойдем по второму пути.

Вариантов решения задачи протоколирования множество. Вы можете использовать предложенный мною сценарий, а можете написать свой.

Так как наш счетчик уже считает уникальных посетителей, то можно было бы просто добавить один запрос SQL, который бы вносил в базу нужную

нам информацию. Но этот способ я считаю некорректным. Дело в том, что для ускорения поиска мы собираемся использовать таблицу с индексом. Если пользователь, вновь заходящий с того же самого IP-адреса, отключит и/или удалит Cookies, сценарий счетчика посчитает этого пользователя уникальным и попытается внести его в базу; но в таблице эта запись уже есть, и мы получим ошибку – выражаясь языком BDE (Borland Database Engine), "Key Violation", то есть нарушение индекса. И такую ошибку мы будем получать примерно в 30% случаев, потому что по статистике именно столько пользователей отключают Cookies. Значит, нам придется, не полагаясь на Cookies, проверять уникальность посетителя по базе.

Я предлагаю сохранять о посетителе следующие данные (поля таблицы):

- `IP  char (15)`
- `vdate  char (8)` // дата последнего посещения сайта (формат mm.dd.yy)
- `browser  char (127)` // название браузера посетителя

Поле IP (IP-адрес клиента), естественно, должно быть уникальным, то есть по нему мы будем строить индекс. 127 символов для имени браузера мы выделили с огромным запасом: вдруг Microsoft выпустит IE8, включив в его описание имена всех разработчиков?

Итак, приступим к реализации протоколирования. Нам понадобится всего одна таблица, поэтому новой базы данных создавать не надо: обойдемся существующей стандартной базой «test». Чтобы создать в ней новую таблицу, выполните команду

```
mysql test
```

(под Linux вам, скорее всего, нужно будет указать пароль: `mysql test -p` )

После этого введите запрос:.

```
create table stat(ip char(15) primary key, vdate char(8), browser
char(127));
```

В листинге 25.3 приведен фрагмент сценария, заносящий данные посетителя в нашу таблицу `stat`. Этот фрагмент нужно вставить в сценарий 25.1 после установки Cookies (строка 7).

**Листинг 25.3. Проверка уникальности посетителя**

```
// подключаемся к базе данных
mysql_connect("localhost");
mysql_select_db("test");
```



http://all-ebooks.com

```
// проверяем IP-адрес на уникальность
$Unique = 1;
$R = @mysql_query("SELECT * FROM stat") or die("Invalid SQL
query");

while($Rw=mysql_fetch_row($R))
{
  if($Rw[0]===$REMOTE_ADDR)
  {
    $Unique = 0;
    break(1);
  }
}
if($Unique)
{
  // если хост уникальный, заносим его в базу
  $vdate=date("d.m.y");
  @mysql_query("INSERT INTO stat
    VALUES('$REMOTE_ADDR','$vdate','$HTTP_USER_AGENT')");
}
else
{
  $vdate=date("d.m.y");
  mysql_query("UPDATE stat SET vdate='$vdate',
    browser='$HTTP_USER_AGENT' WHERE ip='$REMOTE_ADDR'");
}
```

Поскольку этот фрагмент кода предназначен для отладки на вашей домашней машине, мы подключаемся к базе данных (строка 2), используя имя пользователя по умолчанию. Если же сценарий должен работать на сервере вашего хостинг-провайдера, то придется подключаться с именем и паролем, выделенными вам провайдером (скорее всего, регистрационные данные для MySQL будут совпадать с вашим логином и паролем для входа в систему):

```
mysql_connect("localhost","user","password");
```

Разумеется, это должны быть имя и пароль, дающие право на создание таблицы.

Наш фрагмент кода начинает работу после того, как сценарий 25.1 по отсутствию Cookie предварительно определил посетителя как уникального. Мы читаем из таблицы все записи, ищем среди них IP-адрес посетителя `$REMOTE_ADDR` и, если он найден, сбрасываем флаг уникальности (переменная `$Unique`). Далее, если узел оказывается уникальным, мы заносим его в таблицу, а если нет – обновляем имеющуюся запись, сохраняя в ней дату последнего посещения и имя браузера пользователя.

И напоследок одно важное замечание: наверняка вам захочется вынести подключение к серверу MySQL в начало сценария. Этого не следует делать!

Эти строки могут породить какой-нибудь текст в случае ошибки подключения, и тогда мы не сможем установить Cookies. Информация о Cookies должна содержаться в заголовке, а если мы отправим какой-нибудь текст в браузер, то уже не сможем отправить продолжение заголовка, содержащего информацию о Cookies. В случае ошибки мы хотя бы успеем установить Cookies (если они включены).

### 25.1.4. Разработка системы статистики

Предположим, что вы создаете собственную систему статистики наподобие SpyLog. Другими словами, вы раздаете всем желающим счетчики, которые после установки на их сайтах будут считать посещения, а потом пользователи смогут просмотреть статистику.

Прежде всего нам нужно обеспечить идентификацию самих сайтов. Для этого будем использовать идентификационный номер сайта — id. Установить наш будущий счетчик на сайте клиента можно с помощью следующего тега:

```
<img src=counter.php?id=171>
```

В данном случае идентификатор сайта равен 171. Идентификатором по умолчанию будет номер 1 (наш сайт). Для подсчета посещений нашего сайта счетчик можно установить так:

```
<img src=counter.php>
```

А можно явно задать идентификатор, то есть id = 1.

Общее число уникальных посетителей для каждого сайта будет храниться в таблице ids базы данных counter. Хочу обратить ваше внимание на то, что в отличие от примера из предыдущего пункта, здесь мы будем использовать «свою собственную» базу данных. Структура таблицы ids выглядит так:

```
id int primary key, unq int
```

где id — идентификатор сайта, а unq — количество уникальных посещений.

Отдельная статистика для каждого сайта будет записываться в соответствующую таблицу `idN`, где N – это идентификатор сайта, например, `id171`. Структура таблиц `idN` будет такова:

```
ip char(15), vdate char(8), browser char (127)
```

Думаю, теперь все стало ясно. Нам осталось только реализовать все это программно. Полный текст счетчика с протоколированием приведен в листинге 25.4.



http://all-ebooks.com

**Листинг 25.4. Простая система статистики**

```
<?
// Идентификатор не указан - наш сайт
if (!isSet($id)) $id=1;

$table="id$id"; // Таблица для статистики

// подключаемся к базе данных
$SERVER = "localhost";
$USER = "user";
$PASSWD = "top_secret";
if(!mysql_connect($SERVER,$USER,$PASSWD))
{
  echo "<h1>Произошла ошибка при подключении к серверу MySQL</h1>
";
  echo "<p>Сервер $SERVER</p>";
  // Пароль никогда не выводим!!!!
  echo "<p>Имя пользователя $USER</p>";
  exit;
}

mysql_select_db("counter");

$Unique = 1;
$R=@mysql_query("SELECT * FROM $table") or die("Invalid SQL query 1");

while($Rw=mysql_fetch_row($R))
{
  if($Rw[0]===$REMOTE_ADDR)
  {
    $Unique = 0; break(1);
  }
}

// если хост уникальный...
if($Unique)
{
  $R=mysql_query("SELECT * FROM ids WHERE id=$id") or die("Invalid
query 2");
  $Rw=mysql_fetch_row($R);
  $counter=$Rw[1];
  $vdate=date("d.m.y");
  $counter=$counter+1;  // обновляем счетчик

  // обновляем информацию о посетителе
  @mysql_query("UPDATE ids SET unq=$counter WHERE id=$id");
  @mysql_query("INSERT INTO $table
    VALUES('$REMOTE_ADDR','$vdate','$HTTP_USER_AGENT')");
```

```
}
else
{
  $vdate=date("d.m.y");
  mysql_query("UPDATE $table SET vdate='$vdate',
    browser='$HTTP_USER_AGENT' WHERE ip='$REMOTE_ADDR'");
}

// выводим картинку
$img = ImageCreateFromPng("images/template.png");
$color = ImageColorAllocate($img, 0, 0, 255);
ImageString($img,3,5,9,"Visits : $count",$color);
Header("Content-type: image/png");
ImagePng($img);
ImageDestroy($img);
?>
```

Разумеется, для работы этого сценария необходимо, чтобы все таблицы `idN` уже существовали. Поэтому, прежде чем снабжать пользователя-владельца сайта кодом счетчика, этот сайт нужно зарегистрировать, то есть создать для него таблицу. Есть два метода решения этого вопроса: можно написать отдельный сценарий регистрации пользователей, а можно просто добавить в сценарий 25.4 запрос на создание таблицы после подключения к серверу. Второй способ не совсем корректен, потому что при уже имеющейся таблице такой запрос будет порождать ошибку 1050: таблица существует. Поэтому сейчас мы займемся написанием отдельного сценария регистрации (листинг 25.5).

Этот сценарий должен работать примерно так: находим номер N последнего зарегистрированного пользователя (он будет максимальным), потом присваиваем новому пользователю номер N+1 и создаем таблицу idN+1.

**Листинг 25.5. Регистрация новых пользователей**

```
<?
// подключаемся к базе данных
mysql_connect("localhost");
mysql_select_db("counter");
$R=mysql_query("SELECT * FROM ids") or die("Invalid SQL query 1");

$max=mysql_num_rows($result);
// новый номер
$N=$max+1;

// Таблица   для   статистики
$table="id$N";
```



http://all-ebooks.com

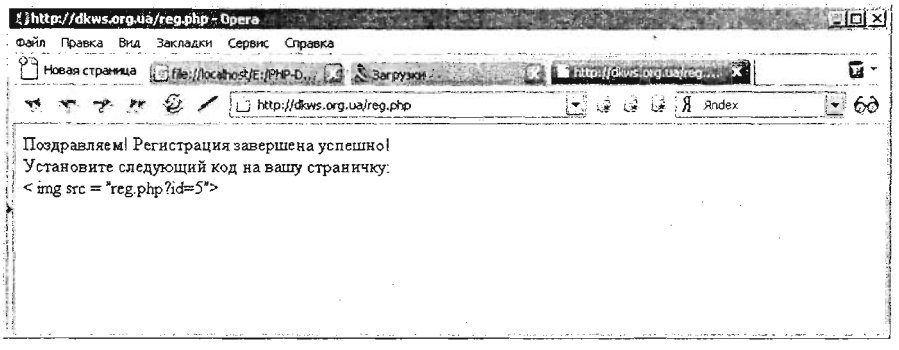



***Рис. 25.2. Регистрация нового пользователя***

```
mysql_query("insert into ids values ($N,0)");

// создаем таблицу
mysql_query("create table $table (ip char(15) primary key,
  vdate char(8), browser char (127))");

echo "Поздравляем! Регистрация завершена успешно!";
echo "<br>Установите следующий код на вашу страничку: ";
echo "<br>&lt img src = \"reg.php?id=$N\">";
?>
```

Вместо явного указания имени сценария (`reg.php`) вы можете использовать переменную окружения $SCRIPT_NAME. Как видите, сценарий для регистрации новых пользователей предельно прост. В этом сценарии я не рекомендую вам использовать символ @ перед названиями функций для работы с MySQL, чтобы в случае возникновения ошибки пользователь увидел соответствующее сообщение и понял причину отказа в регистрации.

Практически все готово для нашей системы статистики. Осталось написать небольшой сценарий, который бы показывал пользователю статистику обращений к его сайту. Этот сценарий приведен в листинге 25.6.

**Листинг 25.6. Сценарий просмотра статистики**

```
<?
mysql_connect("localhost");
mysql_select_db("counter");

if (!isSet($id)) $id=1;
$table="id$id";

$R=mysql_query("select unq from ids where id=$id");
$Rw=mysql_fetch_row($R);
```

```
$unq=$Rw[0];

$R=mysql_query("select * from $table");
echo "<html><body><h1>Здравствуйте, уважаемый пользователь!</h1>";
echo "Уникальных посетителей: $unq<br>";

// выводим таблицу со статистикой
echo "<table border>";
echo "<tr><td>Хост</td><td>Последнее посещение</td><td>Браузер
  </td></tr>";

while($Rw=mysql_fetch_row($R))
{
  echo "<tr>";
  for($i=0; $i<mysql_num_fields($R); $i++)
    echo "<td>$Rw[$i]</a></td>";
  echo "</tr>";
}
echo "</table>";
echo "</body>";
?>
```

Этот сценарий никак не аутентифицирует пользователя: при вызове сценария ему нужно передать всего один параметр — идентификатор сайта id, который любой желающий может увидеть на сайте зарегистрированного пользователя, просмотрев HTML-код страницы со счетчиком. Процедуру проверки правильности имени пользователя вы можете при необходимости добавить самостоятельно.

Сценарий можно еще модифицировать, добавив функцию разрешения IP-адреса в доменное имя, однако это замедлит работу сценария.

У нас получилась простенькая система статистики посещений, обрабатывающая только адрес узла, дату последнего посещения с этого узла и браузер пользователя. «Серьезные» системы статистики учитывают множество других данных. Например, страну (вы можете узнать, сколько пользователей зашло к вам из России, а сколько из Франции и США).

## 25.2. АНАЛИЗ СЕРВЕРНОЙ СТАТИСТИКИ. ПРОГРАММА WEBALIZER

### 25.2.1. Методика сбора и анализа серверной статистики. Серверные журналы

Владельцев сайта часто интересуют сведения о тех, кто обращался к их сайту: откуда приходил посетитель, что делал и т.п. Такого рода данные хра-



http://all-ebooks.com

нятся в журналах сервера, на котором расположен сайт. Вручную анализировать журналы не слишком удобно, поэтому лучше написать программу, автоматизирующую этот процесс. Но можно воспользоваться уже готовыми программами.

Одной из таких программ является программа Webalizer (`http://www.mrunix.net/webalizer`), разработанная Бредфортом Барретом (Bradford L. Barrett). Эта программа анализирует журналы сервера, собирает статистику и выдает результаты в формате HTML. Webalizer умеет также наглядно представлять статистические данные в виде гистограмм. Отчеты Webalizer ничем не уступают отчетам SpyLog и многим коммерческим счетчикам, а иногда даже превосходят их.

Программа Webalizer поддерживает следующие форматы серверных журналов: CLF (Common Log Format), форматы журналов FTP-серверов wuftpd и ProFTPD, а также формат журнала прокси-сервера Squid. Согласно формату CLF, файлы журналов должны иметь такую структуру:

```
IP ДФ Дата Действие КО ОПИ URL Браузер
```

где:

- **IP** — это IP-адрес удаленного узла;
- **ДФ** — дополнительные флаги;
- **Дата** — дата и время обращения к серверу;
- **Действие** — действие, произведенное сервером;
- **КО** — код ответа сервера, например, 200 — передача произведена успешно, 404 — ошибка: файл не найден;
- **ОПИ** — объем переданной информации;
- **URL** — URL, введенные в поле Location браузера.

Для иллюстрации этого формата приведу выдержку из файла `access_log`, в котором веб-сервер Apache учитывает данные о доступе пользователей:

```
192.168.1.1 - - [28/May/2006:15:27:32 +0300] "GET /mrtg/mrtg-r.gif
HTTP/1.1"
404 299 "http://www.server.ru/mrtg/unix-guide.html" "Mozilla/5.0
(X11;
U; Linux i686; en-US; rv:0.9.2.1) Gecho/20010901"
```

Эта запись означает, что 28 мая 2006 года в 15:27 некто с удаленного узла ввел в адресную строку браузера Mozilla URL «http://www.server.ru/mrtg/unix-guide.html» и пытался получить файл `/mrtg/mrtg-r.gif` методом GET по протоколу HTTP 1.1. Но у этого пользователя ничего не вышло, по-

тому что такой файл не найден на сервере, о чем свидетельствует код ответа 404. Размер страницы с сообщением об ошибке, переданной пользователю вместо запрошенного файла, составляет 299 байт.

Формат журналов серверов FTP отличается от формата CLF и имеет такую структуру:

```
Дата Время Год ВП Узел ОПИ Файл Режим_передачи СО ФП Пользователь
НП Служба
```

где:

- ВП — время, затраченное на передачу файла;
- СО — специальные операции;
- ФП — флаг пользователя;
- НП — направление передачи.

Вот выдержка из файла `Xferlog` – журнала сервера FTP:

```
Wed Jun 12 11:49:35 2002 1 localhost.localdomain 1490 /home/den/
vmware.html
a _ o r dhsilabs ftp 0 * c
Wed Jun 12 11:50:08 2002 1 localhost.localdomain 281 /home/den/
code.bin
a _ o r dhsilabs ftp 0 * c
Wed Jun 12 11:50:15 2002 1 localhost.localdomain 281 /home/den/
code.bin
a _ i r dhsilabs ftp 0 * c
Wed Jun 12 11:52:08 2002 1 localhost.localdomain 888 /home/den/
ftphosts.html
b _ i r dhsilabs ftp 0 * c
```

Из первой записи вы можете узнать, что пользователь dhsilabs зарегистрировался с удаленного узла «localhost.localdomain». Начало передачи файла `/home/den/vmware.html` произошло в среду, 12 июня 2002 года в 11:49. Общее время передачи — одна секунда, общий объем переданной информации – 1490 байт, файл передавался в режиме ASCII (a), не было произведено никаких специальных операций (_). Направление передачи (o) указывает на то, что пользователь скачивал файл с сервера. Пользователь dhsilabs является реальным (зарегистрированным) пользователем системы (r). Название службы, которая производила операцию — ftp.

Теперь рассмотрим четвертую запись. Тот же пользователь dhsilabs передал на сервер (направление i) файл `ftphosts.html` в двоичном (b) режиме.

Вторая и третья записи сообщают о загрузке с сервера и на сервер файла `code.bin` в текстовом (a) режиме.



http://all-ebooks.com

Тип пользователя обозначается символом возле имени пользователя. Символ «r» обозначает зарегистрированного в системе пользователя, у которого есть своя запись в файле `/etc/passwd`. Символ «g» означает гостевую регистрацию, а «a» — анонимную.

На основе такой информации Webalizer генерирует статистику. Информация, предоставляемая программой Webalizer, более информативна, чем информация обыкновенного счетчика. Webalizer может генерировать отчеты о посещении вашего веб-сервера за год, месяц, день и час. По умолчанию Webalizer помещает отчеты в каталог `<DocumentRoot>/usage`, но в настройках программы можно указать и другое место.

## 25.2.2. Установка и запуск Webalizer

Если вы работаете под Linux, то какая-то версия Webalizer уже входит в состав вашего дистрибутива, а последнюю версию программы можно загрузить с сайта `http://www.mrunix.net/webalizer`. Там же, в разделе Downloads, есть ссылка для скачивания Windows-версии программы.

Честно говоря, работа русской Windows-версии мне не очень понравилась. Дело в том, что весь вывод программы представлен в кодировке KOI8-R, а Windows использует CP1251. С текстовой информацией проблем нет — выбрал в браузере нужную кодировку, и все нормально, а вот с картинками действительно проблема: вместо русских букв непонятно что.

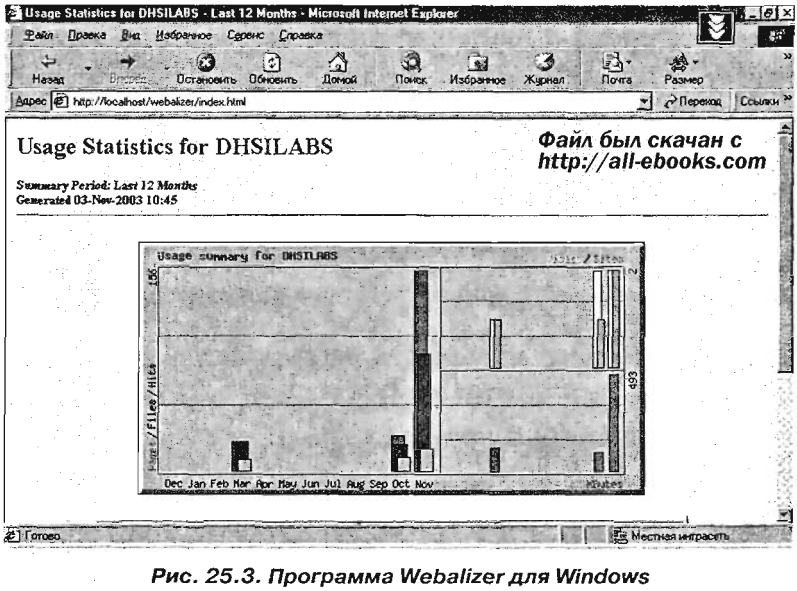



*Рис. 25.3. Программа Webalizer для Windows*

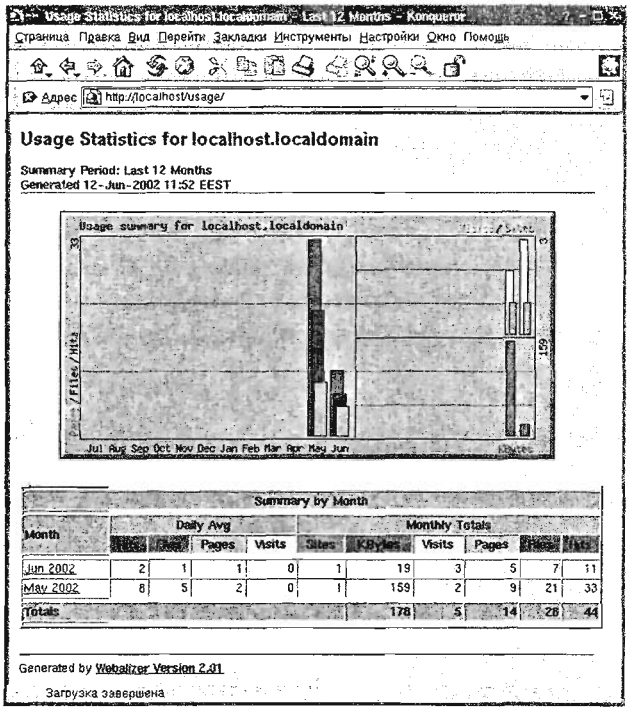



***Рис. 25.4. Программа Webalizer для Linux***

Далее мы будем работать в Linux, но в Windows все то же самое. Запустите программу webalizer:

```
webalizer
```

При запуске на экране вы ничего не увидите, но после завершения работы в каталоге `/var/www/html/usage` появятся файлы отчета. Затем убедитесь, что ваш веб-сервер запущен, и в окне браузера введите URL: `http://localhost/usage`.

В окне браузера вы увидите примерно то, что изображено на рис. 25.4.

Перейдя по ссылке на страницу, содержащую статистические данные за определенный месяц, вы можете просмотреть:

- Ежедневную статистику (Daily Statistics).
- Почасовую статистику (Hourly Statistics).
- Вводимые клиентами URL (URLS), определив таким образом самые посещаемые страницы вашего сайта.



http://all-ebooks.com

- Другую информацию (IP-адреса, страны клиентов, браузеры пользователей).

Весьма желательно добавить запуск программы сбора статистики в расписание демона crond для автоматического обновления статистики каждый час. Для этого в ваш файл `/etc/crontab` добавьте строку:

```
01 * * * * root /usr/bin/webalizer
```

Убедитесь, что демон crond запускается автоматически при старте системы.

Программу Webalizer можно запускать с ключами, перечисленными в табл. 25.1.

**Таблица 25.1.** *Аргументы программы Webalizer*

| Ключ | Назначение |
|---|---|
| -c файл | Указывает альтернативный файл конфигурации. По умолчанию используется файл /etc/webalizer.conf |
| -n имя | Указывает имя машины, которое будет отображено при выводе статистики |
| -o каталог | Указывает каталог, в который будут помещены файлы статистики |
| -t название | Указывает заголовок отчета |
| -F (clf \| ftp \| squid) | Указывает тип журнала: clf, ftp или squid |

### 25.2.3. Конфигурационный файл Webalizer

Если вам нужно считать посещения нескольких виртуальных Web-серверов, я рекомендую для каждого виртуального сервера создать свой файл конфигурации и запускать webalizer с единственным ключом -c, а все остальные настройки внести в конфигурационные файлы.

Рассмотрим директивы конфигурационного файла `/etc/webalizer.conf`.

- LogFile: определяет журнал сервера. Значение по умолчанию:

```
LogFile /var/log/httpd/access_log
```

- LogType: указывает тип журнала: web или ftp. Значение по умолчанию:

```
LogType web
```

- OutputDir: определяет каталог, куда Webalizer помещает отчеты. Значение по умолчанию:

http://all-ebooks.com

```
OutputDir /var/www/html/usage
```

- HistoryName: задает имя файла, в котором хранятся сведения о посещении веб-сервера на протяжении последнего года.

```
HistoryName webalizer.hist
```

Формат этого файла таков (месяц, год, статистические данные):

```
5  2002 33 21 1 159 28 31 9 2
6  2002 11 7 1 19 8 12 5 3
```

- Incremental: требует применять нарастающую обработку, то есть использовать несколько сравнительно небольших файлов журналов вместо одного огромного. Включать эту директиву полезно для больших сайтов, журналы которых ротируются чаще, чем раз в месяц (как принято на обычных веб-серверах). Для крупных сайтов целесообразно выполнять ротацию раз в неделю, сократив размер основного журнала. Текущие статистические данные при нарастающей обработке хранятся в файле, имя которого задает директива IncrementalName (по умолчанию это `webalizer.current`). По умолчанию нарастающая обработка отключена.
- ReportTitle: задает заголовок отчета. Значение по умолчанию:

```
ReportTitle Usage Statistics for
```

- HostName: имя веб-сервера. Обычно оно разрешается с помощью DNS, а эту директиву можно не определять.
- HTMLExtension: определяет расширение для файлов отчетов. Значение по умолчанию:

```
HTMLExtension html
```

- PageType: указывает расширения файлов, которые будут считаться Web-страницами. Значения по умолчанию:

```
PageType htm*
PageType cgi
```

- UseHTTPS: позволяет использовать безопасное соединение HTTPS для просмотра статистики. По умолчанию безопасное соединение не используется.
- Quiet: подавляет вывод на консоль информационных сообщений программы, что необходимо при запуске Webalizer из-под демона crond. Значение по умолчанию:

```
Quiet yes
```



http://all-ebooks.com

- ReallyQuiet: подавляет также вывод предупреждений и сообщений об ошибках. По умолчанию эта директива выключена:

```
ReallyQuiet no
```

Все остальные директивы не столь существенны и в основном относятся к внешнему виду создаваемых отчетов. С помощью этих директив вы можете настроить создаваемые директивы по своему вкусу. Я надеюсь, что этой информации вполне достаточно, чтобы настроить Webalizer и легко получать статистическую информацию о работе своего сайта.

http://all-ebooks.com

# ГЛАВА 26.

# СЕССИИ В PHP

http://all-ebooks.com

## 26.1. ОБЗОР МЕХАНИЗМА СЕССИЙ

### 26.1.1. Когда не пригодны Cookies

Одним из недостатков интерфейса CGI является то, что CGI-программа не может «общаться» с пользователем длительное время: она запускается, обрабатывает переданные ей параметры, выдает результат и заканчивает работу.

Представьте себе, что вы пишете анкету, в которой так много полей, что они не умещаются в одном окне браузера. Такие анкеты обычно приходится размещать на нескольких веб-страницах, отображаемых последовательно по мере заполнения анкеты, а отправлять введенные данные все сразу. Возникает вопрос: как в таком случае хранить данные, введенные пользователем на предыдущих страницах анкеты?

Можно, конечно, записывать их во временный файл. Но тогда нужно запретить заполнять анкету нескольким пользователям одновременно, иначе их данные перемешаются. Использовать базу данных? Но разворачивать целую базу для хранения сравнительно небольшого объема промежуточных данных – это стрельба из пушки по воробьям.

Более удачным решением выглядит хранение данных в Cookies. Но и оно не идеально: у пользователя, заполняющего анкету, Cookies могут быть отключены. Кроме того, не существует механизма согласования имен Cookies между разными сайтами. Допустим, мой сценарий записывает в Cookie с именем «DB» промежуточные результаты, а сценарий на чужом сайте хра-

http://all-ebooks.com

нит в «DB» имя базы данных. Тогда, если пользователь заглянет на один из этих сайтов в течение срока действия Cookie, установленной вторым, то какой-то из сценариев сработает некорректно.

Еще более удачное решение задачи идентификации пользователя – это механизм сессий, не требующий включения Cookies и не вызывающий коллизии имен.

### 26.1.2. Механизм сессий: принцип работы

Итак, сессии — это механизм, позволяющий однозначно идентифицировать пользователя, сохраняя для него на сервере все нужные переменные в течение всей сессии, то есть сеанса работы пользователя с сайтом.

Для идентификации пользователя служит идентификатор сессии — SID (Session IDentifer), передаваемый сервером браузеру пользователя и проверяемый при каждом обращении к серверу.

Идентификатор сессии SID — это обычная переменная, которая обычно называется $PHPSESSID. Мы уже знаем, что переменную можно передавать либо через Cookies, либо через POST/GET запрос (в качестве параметра). Язык PHP позволяет использовать оба этих метода для обмена SID между сервером и пользователем (браузером).

С использованием Cookies мы уже знакомы. Когда пользователь заходит на сайт, ему присваивается определенное значение SID и записывается в Cookies. На сервере заводится соответствующий файл, в котором будут храниться переменные текущего сеанса. В течение дальнейшей работы сервер будет брать значение SID из Cookies и по нему выбирать переменные из файла с переменными. Этот способ обмена SID удобен тем, что весь обмен происходит незаметно для пользователя, но не работает, если у пользователя отключены Cookies.

Гарантировать передачу SID может второй способ — через POST/GET запросы, то есть через адресную строку браузера. Суть метода заключается в следующем. Когда пользователь первый раз заходит на сайт, PHP-сценарий, перед тем как отправить страницу сайта пользователю, автоматически добавляет ко всем ссылкам значение SID. Например, если на странице стояла ссылка:

```
<a href=page2.php>Страница 2</a>
```

то после обработки эта ссылка будет выглядеть так:

```
<a href=page2.php?PHPSESSID="7401e923e7aca6d8c66bbf9db9cf6b08">Стр
аница 2</a>
```



http://all-ebooks.com

В результате пользователь, переходя по ссылкам, будет сам передавать серверу значение своего SID вместе с URL-адресом. Как видите, никаких Cookies не нужно, а единственное неудобство такого подхода — появление в адресной строке наборов непонятных символов, которые могут озадачить неопытного, но наблюдательного пользователя.

Кроме ссылок, автоматическому преобразованию подвергаются находящиеся на странице формы: к каждой форме добавляется скрытое поле

```
<input type="hidden" name="PHPSESSID"
    value="740Ie923e7aca6d8c66bbf9db9cf6b08">,
```

значение которого передается на сервер, когда пользователь отправляет заполненную форму.

За то, какой способ передачи идентификатора сессии SID использовать, отвечают две директивы файла `php.ini`:

- `session.use_cookies`: значение 1 означает передачу SID через Cookie, а значение 0 выключает этот способ передачи;
- `session.use_trans_sid`: значение 1 включает передачу SID через присоединение его к URL-адресам, а значение 0 отключает.

По умолчанию оба способа передачи SID включены. Когда пользователь в первый раз заходит на сайт, то ему и Cookies прописывается, и ссылки преобразуются. При следующем запросе сервер сначала пытается считать SID из Cookie. Если это удается, то он и при обработке последующих запросов будет ориентироваться на Cookies, а преобразование ссылок прекратит. Если же при втором запросе Cookie не читается, значит, у этого пользователя они отключены, и сервер прекратит запись Cookies, обменивая SID только через преобразованные ссылки.

## 26.2. ПРИМЕНЕНИЕ СЕССИЙ

### 26.2.1. Инициализация сессии и регистрация сессионных переменных

В сценарии 26.1 показан простейший пример использования сессий.

**Листинг 26.1. Простая сессия**

```
<?
session_start();
session_register("a");
```

```
$a=@$a+1;
echo "<html><body>Нажмите Reload, чтобы увеличить счетчик";
echo "<br>Счетчик:  $a";
echo "</body></html>";
?>
```

Функция `session_start()` инициализирует сессию, а функция `session_register("a")` регистрирует в этой сессии переменную `$a`. Имена переменных сессии указываются без знака доллара. Одну и ту же переменную можно регистрировать в сессии несколько раз: ее значение при повторной регистрации не обнуляется.

Запустите сценарий 26.1 и несколько раз нажмите кнопку браузера **Reload (Обновить)**. Вы увидите, как увеличивается значение счетчика. Счетчик будет сброшен, когда вы закроете окно браузера.

Способ обращения к сессионным переменным зависит от настроек в файле `php.ini`. Если директива `register_globals` включена, то обращение к сессионной переменной ничем не отличается от обращения к обычным переменным:

```
session_register("a");
$a = 5;
```

Если же она выключена (`register_globals=off`), то к сессионным переменным нужно обращаться через глобальный массив `$_SESSION` (в PHP версий до 4.1 вместо него использовался `$HTTP_SESSION_VARS`):

```
session_register("a");
$_SESSION["a"] = 5;
```

По умолчанию опция `register_globals` включена.

В том же файле `php.ini` вы можете указать каталог, в который будут записываться файлы с сессионными переменными. Для этого служит директива `session.save_path`:

```
session.save_path  =  /tmp
```

Путь к каталогу файлов сессии можно узнать и из сценария, вызвав функцию `session_save_path([string  $path])` без аргумента. Аргумент `$path` позволяет сменить этот каталог.

Следует отметить, что в общем случае сценарии, выполняющиеся одновременно, ничего не знают друг о друге. Это значит, что если вы запустите сценарий 26.1 и пару раз нажмете на кнопку **Обновить**, а потом, не закрывая окна, откроете другой экземпляр браузера и запустите в нем этот же сценарий, то в новом окне значение счетчика окажется сброшено. Если же вы откроете новое окно из уже работающего браузера (например, в Internet



http://all-ebooks.com

Explorer командой **Файл→Создать→Окно**), то в этом окне значение счетчика сохранится.

Проверить, зарегистрирована ли переменная в сессии, позволяет функция `session_is_registered(string $name)`. Отменить регистрацию сессионной переменной можно с помощью функции `session_unregister(string $name)`. Чтобы отменить регистрацию всех переменных сессии, вызовите функцию `session_unset()`.

### 26.2.2. Имя сессии

Если пользователь работает одновременно с несколькими сценариями, содержащими одноименные сессионные переменные, то их значения могут перепутаться. Во избежание этого сессии следует именовать. Имя сессии — это название группы сессий, к которой принадлежит данный SID.

Функцию `session_name(string $name)`, присваивающую сессии имя, нужно вызывать до инициализации сессии:

```
session_name("SessionOfIvan");
session_start();
session_register("a");
```

Если имя сессии не указано, PHP будет использовать имя сессии по умолчанию — PHPSESSID.

Функция `session_id([string $sid])`, вызванная без аргумента, возвращает значение идентификатора текущей сессии, а аргумент `$sid` позволяет установить новое значение текущего SID.

PHP позволяет инициализировать сессию автоматически, то есть без вызова функции `session_start()`, а сразу при запуске сценария. Это происходит, когда значение директивы `session.auto_start` в файле `php.ini` равно 1. Я не советую пользоваться этой возможностью, потому что создаваемая таким образом сессия будет безымянной, и вы не сможете задать ей имя даже вызовом функции `session_name()`.

### 26.2.3. Устаревший способ обойти использование Cookies в сессиях

Как я уже сказал, при отключенных пользователем Cookies PHP автоматически добавляет SID ко всем ссылкам на выдаваемой пользователю странице. Однако в старых версиях PHP этой возможности не существовало, и SID приходилось дописывать вручную. То же самое можно проделать и сейчас, и на всякий случай я покажу вам этот способ.

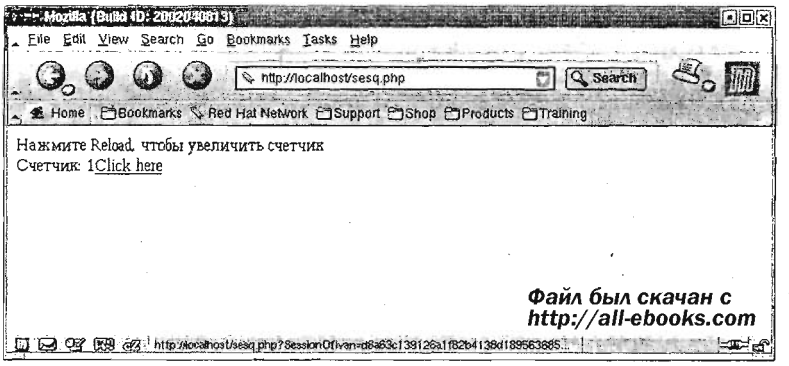



***Рис. 26.1. В строке состояния браузера видна преобразованная ссылка***

Сохраните сценарий 26.2 в файл `sesq.php` и перед его выполнением отключите Cookies в вашем браузере.

**Листинг 26.2. Преобразование ссылок вручную**

```
<?
session_name("SessionOfIvan");
session_start();
session_register("a");
$a=@$a+1;
echo "<html><body>Нажмите Reload, чтобы увеличить счетчик";
echo "<br>Счетчик: $a";
?>
<a href=sesq.php?<?=SID?>>Click here</a>"
```

Обратите внимание на последнюю ссылку: при отключенных Cookies PHP генерирует константу SID, считая, что данные об идентификаторе и имени сессии пришли из Cookies. Если Cookies включены, эта константа ничего не содержит. Для работы нашего сценария первый раз нужно вместо нажатия кнопки **Обновить** перейти по ссылке «Click here», а потом, как обычно, нажимать **Обновить**.

## 26.3. ОБРАБОТЧИКИ СЕССИЙ

Обработчики сессии – это PHP-функции, вызываемые, как только произошло какое-нибудь событие, связанное с сессией, например открытие или закрытие сессии. Для каждого события существует стандартный обработчик, выполняющий действия по умолчанию. Вы можете переопределить



http://all-ebooks.com

стандартные обработчики, если действия по умолчанию вас не устраивают. Для переопределения служит функция

```
session_set_save_handler(string $open, string $close, string $read,
  string $write, string $destroy, string $gc);
```

которую нужно вызывать до инициализации сессии (вызова `session_start()`), иначе она будет проигнорирована. В качестве аргументов ей нужно передать имена пользовательских функций, исполняющих роль обработчиков. Сигнатуры этих функций перечислены в табл. 26.1.

**Таблица 26.1.** *Стандартные обработчики сессий*

| Функция | Назначение |
|---|---|
| bool open (string $save_path, string $session_name) | Вызывается при инициализации сессии. Обычно этот обработчик открывает базу данных или файл для группы сессий $session_name |
| bool close() | Вызывается, когда нужно закрыть временное хранилище (вся информация уже записана) |
| string read (string $sid) | Вызывается для чтения сессионных переменных сессии $sid. Функция возвращает сериализованные данные сессии в формате: name1=value1; name2=value2;...;nameN=valueN; |
| bool write (string $sid, string $data) | Вызывается для записи переменных сессии $sid. Параметр $data задается в вышеописанном формате |
| bool destroy (string $sid) | Вызывается перед уничтожением сессии |
| gc (int $maxlifetime) | «Сборщик мусора» вызывается для окончательного удаления сессии после того, как пользователь покинет сайт. Аргумент $maxlifetime задает максимальное время (в секундах) существования сессии после последнего запроса пользователя |

Теперь рассмотрим пример переопределения обработчиков, который я позаимствовал из официальной документации по PHP (листинг 26.3).

**Листинг 26.3. Пример переопределения обработчиков handlers.php**

```
<?php
//  Обработчик открытия сессии
function open ($save_path, $session_name)
{
```

http://all-ebooks.com

```
  global $sess_save_path, $sess_session_name;

  $sess_save_path = $save_path;
  $sess_session_name = $session_name;
  return(true);
}

// Обработчик закрытия сессии
function close() {
  return(true);
}

// Чтение из сессии
function read ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "r"))
  {
    $sess_data = fread($fp, filesize($sess_file));
    return($sess_data);
  }
  else
  {
    return("");
  }
}

// Запись во временное хранилище
function write ($id, $sess_data)
{
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "w"))
  {
    return(fwrite($fp, $sess_data));
  } else {
    return(false);
  }
}
// уничтожение временного хранилища
function destroy ($id)
{
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  return(@unlink($sess_file));
}
```



http://all-ebooks.com

```
// здесь должен быть код сборщика мусора
function gc ($maxlifetime)
{
  return true;
}
session_set_save_handler   ("open",   "close",   "read",   "write",
"destroy", "gc");

session_start();

// продолжить нормальное использование сессий
?>
```

# ГЛАВА 27.

# СОЗДАНИЕ И ИСПОЛЬЗОВАНИЕ ШАБЛОНОВ. МОДУЛИ

http://all-ebooks.com

## 27.1. РАЗДЕЛЕНИЕ ПРОЕКТА НА ДИЗАЙНЕРСКУЮ ЧАСТЬ И ПРОГРАММИРОВАНИЕ

Для создания серьезного проекта необходимо, как минимум, два человека: один будет заниматься непосредственно программированием, а второй — разрабатывать дизайн будущего проекта. Такое разделение целесообразно практически всегда. Во-первых, универсальные специалисты встречаются довольно редко: кто-то лучше умеет программировать, а кто-то — рисовать. Лично я предпочитаю первое, поэтому в этой книге вы не найдете всевозможных тонкостей Web-дизайна.

Кроме того, любой проект необходимо документировать: хотя бы написать простенький Readme с инструкциями по установке проекта на сервер хостера. Для этого вам понадобится еще один человек — технический писатель, который доступным языком объяснит пользователю, как работать с вашим проектом.

Получается, что при разработке серьезного проекта происходит разделение на техническую и дизайнерскую части. К технической группе относятся программисты и технические писатели. Возможно, вам еще понадобится и системный администратор, который «разгрузит» вашу команду — вы будете заниматься программированием, не тратя время на администрирование вашей сети/сайта/домена.

К группе дизайнеров относится старший дизайнер и его помощники. Вообще, вы, как ведущий программист, будете контролировать работу всей группы: старший дизайнер будет отчитываться только вам, а работа его помощников и их количество вас вообще не должно интересовать. Вы можете даже отдать разработку дизайна на сторону – в какую-нибудь профессио-

http://all-ebooks.com

нальную студию веб-дизайна, чтобы сосредоточиться только на проектировании и воплощении своего сайта.

Все сказанное в значительной степени применимо и к случаю, когда проект создается и поддерживается одним человеком. В этом случае сам разработчик должен разделять два процесса: написание кода и создание дизайна. Важно, чтобы эти два процесса были как можно обособленны. Поясню на примере: можно сначала создать несколько сценариев — «кирпичики» сайта. А потом, разрабатывая дизайн сайта, уже эти кирпичики складывать и перекладывать, как вам будет угодно. При этом вам не придется при каждом изменении дизайна переписывать весь свой проект.

В этой главе мы уделим внимание технической части серьезного проекта, а именно созданию «библиотекаря», который поможет при разработке больших проектов.

## 27.2. ИСПОЛЬЗОВАНИЕ МОДУЛЕЙ. ПОНЯТИЕ БИБЛИОТЕКАРЯ

### 27.2.1. Общие положения

Практически во всех современных языках программирования высокого уровня предусмотрена возможность создания модулей. Основная программа подключает модуль, чтобы в процессе дальнейшей работы вызывать реализованные в этом модуле функции. Разработчик выигрывает вдвойне: во-первых, уменьшается размер кода основной программы и программа выглядит читабельнее, а во-вторых, вы можете написать несколько программ, используя один и тот же модуль.

Сейчас мы попробуем написать свой механизм загрузки модулей. Вы можете спросить: зачем писать свой механизм загрузки модулей, если есть инструкции `include` и `require` (`include_once` и `require_once`)? Дело в том, что хостеры часто настраивают систему так, чтобы инструкции `include` и `require` искали файлы только в определенных каталогах, доступа к которым у вас нет. С их стороны все правильно — они заботятся о безопасности, а вот нас это не очень устраивает. Иногда вообще невозможно подключить файл с помощью `include` и `require`. Придется искать решение своими силами.

### 27.2.2. Создание библиотекаря

Определим требования к нашему библиотекарю:

- Библиотекарь должен загружать модуль (библиотеку функций) только один раз.



http://all-ebooks.com

- Библиотекарь должен обеспечивать удобную настройку путей поиска каталогов, в которых хранятся модули.
- Загрузка модуля должна осуществляться с помощью какой-нибудь функции. Мы ее назовем `Load()`.
- Библиотекарь должен загружаться автоматически.

Исходный код библиотекаря приведен в листинге 27.1.

**Листинг 27.1. Подключаемый модуль librarian.php**

```
<?
if(!defined("LIBRARY_IS_LOADED"))
{
// Библиотека загружена
define("LIBRARY_IS_LOADED",1);

// Расширение библиотечных файлов по умолчанию
define("DefaultExt","lib");

// Пути поиска по умолчанию
$INC_DIRS = array(".","./lib","./include");

// Вспомогательные функции
//----------------------------------------
// Преобразует относительный путь в абсолютный
function absolute_path($path, $cur_path)
{
  // заменим обратные слэши прямыми
  $path = strtr(trim($path),"\\","/");
  // теперь разобьем путь по прямым слэшам
  $APaths = explode("/",$path);

  //  $cur_path - каталог, от которого отсчитывается
   //  относительный путь; если не указан, то используем текущий
каталог
  if ($cur_path==="") $Search=getcwd(); else $Search=$cur_path;

  foreach($APaths as $k=>$v)
  {
    if ($v!=".")
    {
      if(!$k && (strlen($v)>1&&$v[1]===":"||$v==="")) $Search=$v;
      elseif($v==="..")
      {
        if (strlen($Search)>1 && $Search[1]===":") continue;
        $p=dirname($Search);
```

```
                if($p==="/"||$p==="\\"||$p===".")  $Search="";  else
$Search=$p;
        }
        elseif($v!=="") $Search=$Search."/$v";
      }
    }
    return ($Search!==""?$Search:"/");
}

// Преобразует URL в абсолютный путь
function UrlToPath($path)
{
    $URL=dirname("$SCRIPT_NAME");
    $cURL=absolute_path(trim($path),$URL);
    return getenv("DOCUMENT_ROOT").$cURL;
}

// Преобразует все пути в массиве $INC_DIRS в абсолютные
function AbsoluteInclude()
{
    static $DIR="";
    global $INC_DIRS;

    if ($DIR!==$INC_DIRS)
    {
      for($i=0; $i<count($INC_DIRS); $i++)
      {
        $value=&$INC_DIRS[$i];
        if ($value[0]==="." && (strlen($value)==1 ||
          $value[1]==='\\'  ||  $value[1]==='/')) continue;
        $value=absolute_path($value);
      }
      $DIR=$INC_DIRS;
    }
}

//-----------------------------------------
// Главная функция загрузки модуля
function Load($library)
{
    global $INC_DIRS;
    static $DIR, $Last=0;

    AbsoluteInclude();
    $1=$Last;
    do
    {
      $dir=$INC_DIRS[$Last];
      if(@is_file($f="$dir/$library.".DefaultExt))
```



http://all-ebooks.com

```
    {
      $cwd=getcwd();
      chdir(dirname($f));
      foreach($GLOBALS as $k=>$v) global $$k;
      $res=include_once($f);
      AbsoluteInclude();
      chdir($cwd);
      return $res;
    }
    $Last=($Last+1)%count($INC_DIRS);
  } while ($Last!=$1);
  die("Library $library not found at ".join("; ",$INC_DIRS));
}

error_reporting(E_ALL);

// После закрывающего тега ?> не должно быть пробелов
// или иных символов!
} // if(!defined)
?>
```

Обратите внимание на следующие моменты. Массив `$INC_DIRS` содержит список каталогов, в которых будет производиться поиск файлов модулей. Сначала мы ищем файлы модулей в текущем каталоге, затем в каталоге `./lib`, а затем — в `./include`:

```
$INC_DIRS = array(".","./lib","./include");
```

По умолчанию файлы модулей должны иметь расширение `.lib`. Если вас это не устраивает, измените переменную `DefaultExt`:

```
define("DefaultExt","lib");
```

### 27.2.3. Подключение модулей с помощью библиотекаря

Возьмем заготовку модуля `unit.lib`, содержащего единственную функцию `myfunc()`:

```
<?
function myfunc()
{
  echo "Sample function from $SCRIPT_NAME library;
}
?>
```

Пусть файл `unit.lib` находится в одном их каталогов, перечисленных в коде `librarian.php` как пути поиска по умолчанию: текущем каталоге, `./lib` или `./include`. Тогда подключить этот модуль к основной программе мы можем так, как показано в сценарии 27.2.

**Листинг 27.2. Подключение модуля**

```
<?
// Сначала подключаем сам библиотекарь
include "librarian.php";

// Загружаем модуль unit.lib
Load("unit");
?>
```

Если же модуль `unit.lib` находится в другом каталоге – например, `./my_libs`, – то перед вызовом функции `Load()` нужно добавить этот каталог в пути поиска (листинг 27.3).

**Листинг 27.3. Подключение модуля из альтернативного каталога**

```
<?
// Подключаем сам библиотекарь
include "librarian.php";

// добавляем каталог к путям поиска
$INC_DIRS[]="./my_libs";

// Загружаем модуль
Load("unit");
?>
```

Осталось обеспечить автоматическую (а не посредством директивы `include`) загрузку самого библиотекаря. Если интерпретатор PHP установлен как модуль Apache – а у большинства хостинг-провайдеров это именно так – то задача решается просто. В этом случае директивы файла `php.ini` можно указывать в файле `.htaccess`. Создайте файл `.htaccess` в корневом каталоге своего виртуального веб-сервера и добавьте в него строку:

```
php_auto_prepend_file /home/ваше_имя/www/librarian.php
```

Можно обеспечить автозагрузку библиотекаря и по-другому, но тогда вам придется редактировать либо файл конфигурации Apache, либо файл конфигурации PHP, что вам не позволит сделать ни один хостинг-провайдер.

## 27.3. ИСПОЛЬЗОВАНИЕ ШАБЛОНОВ

Надеюсь, я убедил вас в том, что для разработки серьезного веб-проекта необходимо разделить команду на два лагеря: программистов и дизайнеров. Невозможно быть одновременно хорошим программистом и дизайнером:



http://all-ebooks.com

можно быть профессионалом только в одной области или любителем в обеих.

Задачу программиста мы уже облегчили, написав библиотечный модуль, позволяющий загружать отдельные модули проекта по мере необходимости. Сейчас мы пойдем дальше, упрощая задачу дизайнеру нашего проекта. Мы отделим код PHP-сценария от его HTML-шаблона, под которым понимается чистая веб-страница до внедрения в нее PHP-кода. Именно разделение PHP-кода и HTML-шаблона позволит спокойно заниматься созданием дизайна веб-страницы в любом HTML-редакторе, а PHP-скрипты подключать потом, к уже готовой странице.

До сих пор мы этим не занимались: во всех сценариях, приведенных в предыдущих главах, HTML-код выводился непосредственно из сценария оператором `echo`. Но попробуйте взглянуть на такой, например, фрагмент кода глазами дизайнера:

```
echo "<table width=100% border=2 cellspacing=0 cellpadding=3
  border color=$main_color><tr><td>";
echo "<form method=post action=$SCRIPT_NAME>";
echo  "<p  align=center>Заказчик  <input  type=text  name=client
size=30>";
echo "Адрес <input type=text name=addr size=30>";
echo "E-mail <input type=text name=email size=15>";
echo "</td></tr></table><p>";

echo "<table width=100% border=2 cellspacing=0 cellpadding=3
  bordercolor=$main_color><tr><td bgcolor=$main_color>
  <font color=white>Аппаратное обеспечение</td>";

echo "<td bgcolor=$main_color>
  <font color=white>Программное обеспечение</td></tr>" ;
echo "<tr><td valign=top>";

echo "<p><b>Процессор</b>";
echo "<br><input type=radio name=cpu value=C1 checked>Intel Celeron
1.7 Ghz";
echo "<br><input type=radio name=cpu value=C2>Intel Celeron 2 Ghz";
echo "<br><input type=radio name=cpu value=P4>Intel Pentium 4 2 Ghz";
echo "<br><input type=radio name=cpu value=P42>Intel Pentium 4 2.4 Ghz";

echo "<p><b>Оперативная память (DDR)</b>";
echo "<br><input type=radio name=ram value=256 checked>256 MB";
echo "<br><input type=radio name=ram value=512>512 MB";
echo "<br><input type=radio name=ram value=1024>1024 MB";

...
```

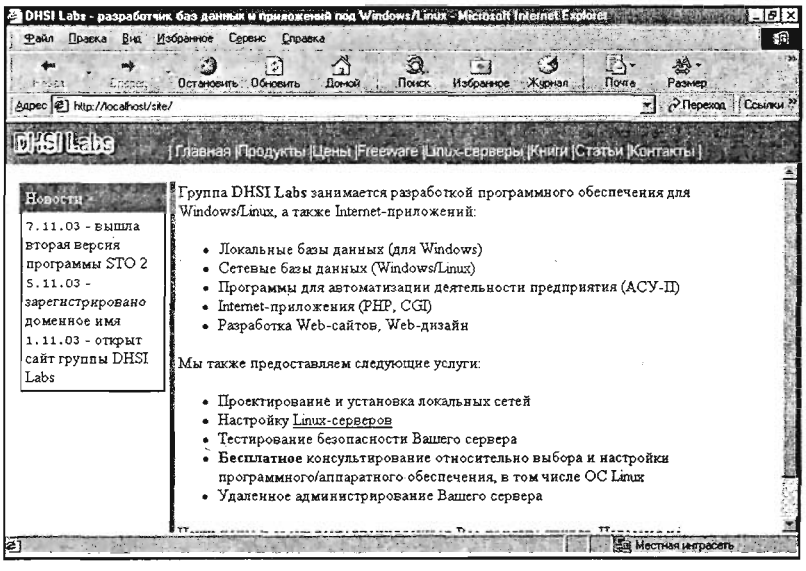



***Рис. 27.1. Лента новостей***

У дизайнера, не знакомого с PHP, глядя на все это, глаза просто разбегаются. Думаете, он решится на правку этого кода? Конечно, нет, потому что он боится что-то нарушить, и это вполне естественно. И что мы будем делать, когда придется переделывать дизайн проекта?

Включая HTML-код в код сценария, вы привязываете себя к этому сценарию: никто кроме вас не сможет его редактировать, а тем более переделывать. Возможно, пока вам это даже выгодно – почему бы не получить дополнительный доход за доработку сценария? Но на том уровне, когда вы будете начинать несколько новых крупных проектов в месяц, не считая поддержки старых, дополнительный доход превратится в дополнительную головную боль. А может быть, и дополнительные расходы: вам придется нанимать помощника для переработки ваших старых, написанных непрофессионально, сценариев.

Теперь самое время перейти к практике. Допустим, вы разрабатываете небольшую ленту новостей – примерно такую, как изображена на рис. 27.1.

Проект для отображения таблицы с новостями будет состоять из двух файлов:

- сценария `news.php`;
- файла с новостями — `add/news.html`.



http://all-ebooks.com

В файле news.html будут храниться новости в HTML-формате, но без тегов <html>,<head> и <body>, то есть собственно содержимое тега <body>. Примерное содержание этого файла:

```
<tt>7.11.03</tt> – вышла вторая версия программы STO 2
<tt>5.11.03</tt> – зарегистрировано доменное имя
<tt>1.11.03</tt> – открыт сайт группы DHSI Labs
```

Такая организация данных значительно упростит работу дизайнеру. Обратите внимание, что файл news.html находится в каталоге add. Я настоятельно рекомендую хранить PHP-скрипты и HTML-документы в отдельных каталогах, чтобы дизайнеру ни при каких обстоятельствах не требовалось заглядывать в каталог сценариев.

Сценарий, отображающий содержимое файла с новостями, приведен в листинге 27.4.

**Листинг 27.4. Сценарий news.php**

```
<?
$NewsFile = "add/news.html";
if (file_exists($NewsFile)) { $news = join('', file($NewsFile)); }
    else $news="No news today";

echo "<table width=100% border=2 cellspacing=0 cellpadding=3
            bordercolor=#336699><tr><td      bgcolor=#336699><font
color=white>Новости
</font></td></tr><tr><td>".$news."</td></tr></table>";
?>
```

Как вы видите, сценарий состоит всего из трех строк: в первой строке мы объявляем переменную $NewsFile, содержащую имя файла новостей, во второй — получаем содержимое этого файла, если он существует, а в третьей — оформляем таблицу: выводим рамку и в середине этой рамки помещаем содержимое файла новостей.

Но это еще не все. Оформление таблицы (синяя рамка) пока находится в сценарии и недоступно дизайнеру. Непорядок! Создадим два дополнительных шаблона: открывающий таблицу (заголовок) и закрывающий. Назовем их соответственно t_header.html и t_footer.html и поместим в каталог add, чтобы дизайнер мог с ними развлекаться.

Файл t_header.html:

```
<table    width=100%    border=2    cellspacing=0    cellpadding=3
bordercolor=#336699>
<tr>
<td bgcolor=#336699><font color=white>Новости</font></td>
```

```
</tr>
<tr><td>
```

Файл t_footer.html:

```
</td></tr>
</table>
```

Перепишем наш сценарий news.php так, чтобы он использовал оба новых шаблона. Окончательный код этого сценария приведен в листинге 27.5.

**Листинг 27.5. Окончательный вариант сценария news.php**

```
<?
$NewsFile = "add/news.html";
if (file_exists($NewsFile)) { $news = join('', file($NewsFile)); }
   else $news="No news today";

include "add/t_header.html";
echo $news;
include "add/t_footer.html";

?>
```

Наш сценарий стал проще даже для программиста: теперь ему не нужно знать, как дизайнер оформил выводимый текст. Большие куски HTML-кода усложняют восприятие исходного кода сценария; избавившись от них, мы убиваем сразу двух зайцев: помогаем и себе, и дизайнеру.

Осталось встроить сценарий вывода новостей в проект. Если файл, в который вы хотите внедрить сценарий news.php, сам является PHP-сценарием, тогда используйте директиву include или include_once:

```
include "news.php";
```

Можно также включить в сценарий модуль news.php с помощью библиотекаря:

```
Load("news");
```

Если же вы хотите внедрить сценарий news.php в HTML-документ, тогда вам нужно будет произвести следующие действия:

1. Вставить в нужное место в вашем HTML-файле строку <? include "news.php"; ?>. Например:

```
<table border=0 cols=3><tr>
<td width=20% valign=top>
<? include "news.php"; ?>
</td>
...
```



http://all-ebooks.com

2. Создать в каталоге, в котором находится HTML-документ, файл `.htaccess` и добавить в него строку, связывающую интерпретатор PHP с расширением .html:

```
AddHandler application/x-httpd-php.html
```

3. На всякий случай скопировать файл `.htaccess` в каталог, в котором находится сценарий `news.php`.

Давайте еще улучшим наш сценарий. Не всякую новость можно описать одной строкой: даже если вы умудритесь это сделать, ваш почтовый ящик будет забит сообщениями с просьбами рассказать о той или иной новости подробнее. Эту проблему можно решить так: при формировании файла новостей каждую новость записывать в виде:

```
<tt> дата </td> - <a href=DDMMYY.html target=_blank>Заголовок новости</a>
```

Теперь строка, которая раньше представляла собой всю новость, будет только ее заголовком – ссылкой, ведущей на подробное сообщение, открывающееся в новом окне браузера. Само сообщение хранится в файле `DDMMYY.html`, где `DDMMYY` – это дата новости.

Идем дальше. Хотелось бы, чтобы все страницы новостей были оформлены единообразно, то есть в файлах `DDMMYY.html` редактор новостей писал бы тот текст, который должен быть в теге `<body>`, а дизайн страниц задавался бы отдельно – файлами `news_header.html` и `news_footer.html`, автоматически добавляемыми к каждой новости.

Напишем сценарий `i.php`, выводящий подробное описание новости за дату `$d` (листинг 27.6).

**Листинг 27.6. Сценарий i.php**

```
<?
if (isset($d))
{
  $fname="./news/".$d.".html";
  if (file_exists($fname))
  {
    $text = join('', file($fname) );
  } else $text="Новость от $d не найдена";

  $text = join(file($fname),'');

  include "add/news_header.html";
  echo $text;
  include "add/news_footer.html";
}
?>
```

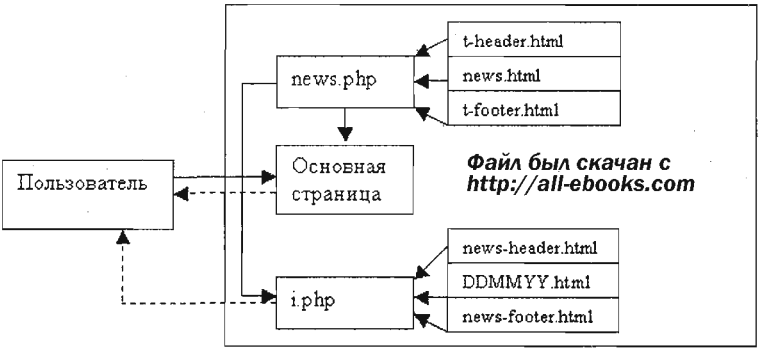



*Рис. 27.2. Схема передачи данных*

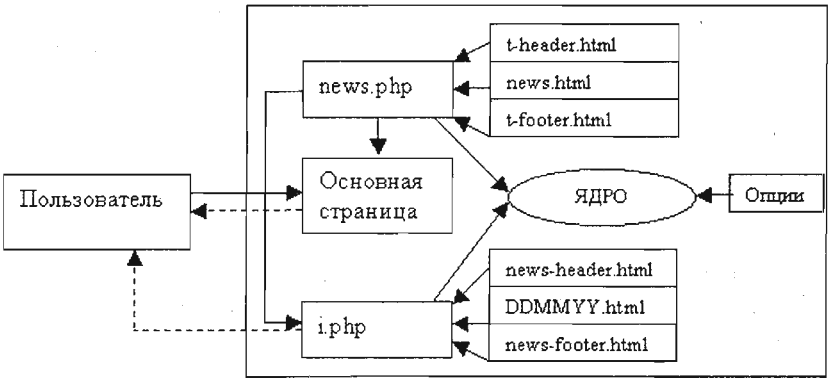



*Рис. 27.3. Схема передачи данных с использованием ядра*

Чтобы этот сценарий работал, ссылки в файле `add/news.html` должны выглядеть так:

```
<a href=i.php$d=DDMMYY.html target=_blank>
```

Итак, наша небольшая новостная лента состоит из таких файлов:

- `news.php` — сценарий, выводящий заголовки новостей.
- `i.php` — сценарий, выводящий подробное описание новости.
- `news.html` — «база данных» новостей.
- `t_header.html`, `t_footer.html` — шаблоны таблицы списка новостей.



http://all-ebooks.com

- `news_header.html`, `news_footer.html` — шаблоны оформления каждой новости.

На рис. 27.2 изображена схема нашей новостной ленты: направления передачи данных между пользователем, сценариями и шаблонами.

Вполне возможно, что вскоре наш проект разрастется до такой степени, что ему будут нужны вспомогательные функции, вызываемые как сценарием `news.php`, так и сценарием `i.php`. В этом случае имеет смысл вынести все общие функции в отдельный сценарий, который мы назовем ядром проекта — сценарий `kernel.php`. Ядро будет содержать общие функции и параметры проекта. Каждый сценарий проекта должен будет подключить этот файл либо с помощью инструкции `include` (`include_once`), либо загружать его средствами библиотекаря — не зря же мы его писали!

Если параметров проекта окажется слишком много, можно вынести их в отдельный файл `options.php`. С учетом файлов ядра и опций схема нашего проекта будет выглядеть так, как показано на рис. 27.3.

http://all-ebooks.com

# ГЛАВА 28.

# ПИШЕМ СВОЙ ИНТЕРНЕТ-МАГАЗИН

http://all-ebooks.com

## 28.1. ПОСТАНОВКА ЗАДАЧИ

Что такое интернет-магазин, наверняка знают все. Покупатель заходит на сайт, регистрируется, выбирает товар, затем принимает окончательное решение, хочет ли он купить выбранный товар или нет. Потом с его счета снимается определенная сумма, и заказанный товар присылается почтой на указанный покупателем адрес.

В сети существует множество виртуальных магазинов, в которых можно заказать практически все — от компакт-диска до автомобиля или золотых часов стоимостью 25000 американской валюты. Разве что продукты заказывать через Интернет не стоит: обычно доставка почтой занимает от нескольких дней до месяца, и вы рискуете умереть с голоду, пока дождетесь своего заказа. Проблема быстрой доставки успешно решена пока в очень немногих городах бывшего СССР.

Кроме скорости доставки, отечественные интернет-магазины отличаются от западных еще и системой оплаты. Если «там» можно заплатить кредитной карточкой, то у нас практически везде — только наложенным платежом: вы подтверждаете заказ по электронной почте, а оплачиваете доставленные товары на почте обычной. В результате вам приходится доплачивать до 25% стоимости заказа за услуги почты.

Почему бы не использовать кредитные карточки? Во-первых, карточки есть далеко не у всех, а те, у кого они есть, обычно стараются снимать с них деньги сразу после начисления. Во-вторых, большинство владельцев карточек опасаются доверять номер кредитки отечественным веб-сервисам. Во избежание, знаете ли. И в-третьих, для работы с кредитками нужно настро-

http://all-ebooks.com

ить SSL-соединение, написать сценарий, использующий SSL, а это уже тема для отдельной книги.

Итак, мы собираемся написать собственный виртуальный магазин. Для начала нужно определиться, что мы будем там продавать. Я предлагаю продавать компакт-диски и книги. Следовательно, у нас будет две категории товара — CD и Book.

Принцип работы нашего магазина будет следующим. Пользователь заходит на наш сайт и регистрируется, то есть вводит свою фамилию, почтовый адрес и e-mail. Не думаю, что один и тот же пользователь будет покупать что-либо более одного раза в месяц, учитывая скорость работы почты. Поэтому заводить специальную таблицу, в которой будет храниться информация о пользователе, я не вижу смысла. Даже если пользователь зайдет два или три раза с целью что-нибудь купить, ничего страшного не случится, если ему придется вводить заново свое имя и адрес. Зато нам не нужно будет хранить на диске многомегабайтную базу данных с данными о пользователях, которые, возможно, не сделали ни одной покупки.

Конечно, такая таблица пригодилась бы нам для статистики и для скидок самым активным покупателям, но при необходимости вы можете доделать все это сами.

Зарегистрировавшийся пользователь должен выбрать категорию товара — книги или компакты. После этого мы покажем ему список имеющихся товаров этой категории, он выберет нужные товары и укажет их количество. После этого, если он нажмет кнопку «Заказ», ему будет предоставлен счет: список заказанных товаров и общая сумма. Если он согласен оплатить эту сумму, то нажимает кнопку «Согласен», и мы высылаем на указанный e-mail письмо с просьбой подтвердить заказ. Если он ответит на письмо, то администрация магазина отправит ему наложенным платежом заказанные им товары.

## 28.2. СТРУКТУРА ИНТЕРНЕТ-МАГАЗИНА

В таблице товаров будут следующие поля:

- ID: идентификатор товара.
- CAT: категория. Поле содержит два символа: «BK» — книга, «CD» — компакт-диск.
- OPIS: описание товара (64 символа).
- PRICE: цена товара.



http://all-ebooks.com

Кроме этих характеристик товара, было бы полезно хранить год выпуска, дату поступления товара в магазин, оставшееся количество товара на складе, но я оставлю вам это для самостоятельной работы.

У нас получилась единственная таблица. Можно было бы поместить в базу данных `test`, но мы этого делать не будем, чтобы не мешать дальнейшему расширению функциональности магазина. Не мучаясь поисками оригинального названия, назовем наш магазин MyShop, а базу, соответственно, `myshop`.

Создать ее можно тремя способами:

- phpMyadmin — универсальное средство.
- Командой `mysqladmin create myshop` — если у вас есть доступ к консоли сервера и администратор MySQL разрешил вам создавать базы данных.
- Вызовом функции `mysql_createdb("myshop")` — аналогично, если вам даны полномочия на создание баз данных. Обычно для каждого пользователя хостинга провайдер заводит единственную базу данных, к которой обращаются все сценарии этого пользователя.

## 28.3. ПИШЕМ СЦЕНАРИЙ

Автоматизировать процесс создания и первичного заполнения базы данных нашего магазина поможет сценарий, приведенный в листинге 28.1. Он рассчитан на работу на вашем домашнем компьютере, где у вас есть любые полномочия. Для работы на сервере хостинг-провайдера этот сценарий придется модифицировать с учетом требований, которые хостер предъявляет к именам баз данных (обычно требуется, чтобы эти имена были в формате «имя_пользователя__имя_базы»). Если же хостер не предоставляет пользователям права создавать базы данных самостоятельно, то этот сценарий будет вообще бесполезен.

**Листинг 28.1. Файл setup.php**

```
<?
require "header.php";
require "options.php";

echo $head;
echo "<h1>Программа установки</h1>";
```

```
if(!mysql_connect($SERVER,$USER,$PASSWD))
{
  echo "<h1>Ошибка сервера MySQL. MySQL server error</h1>";
  echo "</body></html>";
  exit;
}

echo "<p>Пытаемся создать базу данных..."; echo $DB;

mysql_createdb($DB);

echo mysql_error(mysql_errno());

if (mysql_errno()==0)
{
  echo "<p>База данных создана успешно! Создаем таблицы...";

  mysql_select_db($DB);
  echo mysql_error();
  mysql_query('CREATE TABLE tovar (
    id int primary key,
    cat char(2) not null default \'CD\',
    opis char(64) not null default \'\',
    price numeric(9,2) not null default 0
  )');
  echo mysql_error();


  if (mysql_errno()===0)
  {
    echo "<p>Таблица создана. Добавляем данные";
    mysql_query('INSERT INTO tovar values(1,"BK","Колисниченко Д.Н.
      Linux-сервер своими руками, 2005",220)');
    mysql_query('INSERT INTO tovar values(2,"BK","Колисниченко Д.Н.
      Самоучитель Linux, 2005",175)');
    mysql_query('INSERT INTO tovar values(3,"CD","Linux Mandriva
      2007 6CD",235)');
       mysql_query('INSERT INTO tovar values(4,"CD","Linux FC7
5CD",200)');
    echo "<p>Инсталляция завершена!";
  }
  else { echo "<p>Ошибка при создании таблицы"; }
}
else
{
  echo "Ошибка при создании БД: ".mysql_error();
}
?>
```



http://all-ebooks.com

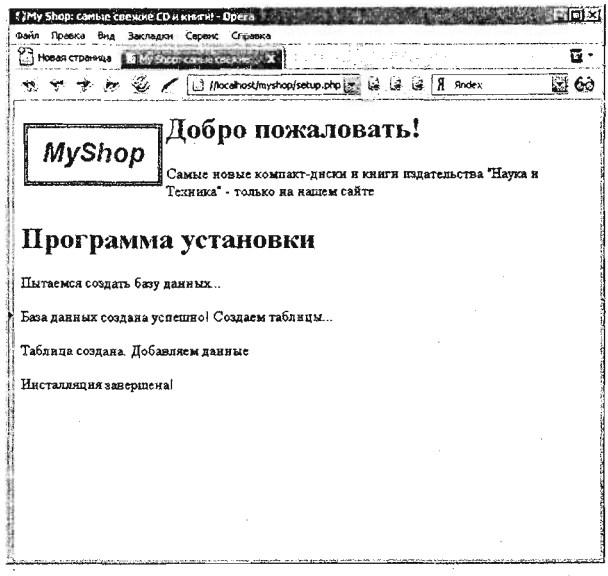



***Рис. 28.1. Установка интернет-магазина***

Обратите внимание, что параметры подключения — имя пользователя, пароль, а также имя самого сервера – мы берем из файла `options.php`, который является общим для всего проекта. Код этого сценария приведен в листинге 28.2.

**Листинг 28.2. Файл options.php**

```
<?
// Имя сервера; вместо него можно указать IP-адрес
$SERVER = "localhost";
// Имя пользователя БД
$USER = "root";
// Пароль
$PASSWD = "";
// Имя базы данных
$DB = "myshop";
// Адрес для подтверждения заказа
$REPLY = "buy@myshop.net";
?>
```

После успешного завершения работы программы-инсталлятора (рис. 28.1) не забудьте удалить файл `setup.php` — из соображений безопасности.

Я решил не разбивать проект на много файлов (сценариев), поэтому все действия магазина выполняет один-единственный файл — `index.php`.

Правда, я вынес заголовок в отдельный файл `header.php` (листинг 28.3), чтобы вы при необходимости могли изменить логотип, кодировку и другие параметры, не изменяя кода самого сценария.

**Листинг 28.3. Вспомогательный файл header.php**

```
<?
$head = '<html><head>
<meta http-equiv=Content-Type content = text/html; charset =
windows-1251>
<title>My Shop: самые свежие CD и книги!</title></head>
<body><table width=100% border=0 cols=2>
<tr>
<td><img src=myshop_logo.png></td>
<td><h1>Добро пожаловать!</h1><p>Самые новые компакт-диски
и книги издательства "Наука и Техника" - только на нашем сайте</
td></tr></table>';
?>
```

## 28.4. ПИШЕМ СЦЕНАРИЙ ИНТЕРНЕТ-МАГАЗИНА

Вот теперь перейдем к написанию сценария `index.php`. Для начала напомню схему работы магазина: пользователь заходит, регистрируется, выбирает категорию (книги или компакты), выбирает товар и нажимает кнопку «Заказать». После этого на указанный пользователем электронный адрес отправляется уведомление о сделанном заказе. Если он подтверждает заказ, то товар высылается ему наложенным платежом.

Регистрационные данные будут храниться в Cookies браузера до окончания сеанса, то есть закрытия окна браузера.

Сценарий использует вспомогательную функцию `ShowTovar()`, отображающую товары указанной категории. Функции нужно передать лишь один параметр — категорию. Код функции приведен в листинге 28.4.

**Листинг 28.4. Функция ShowTovar()**

```
/ Функция выводит товар указанной категории
// Категория по умолчанию - CD
function ShowTovar($ct="CD")
{
  // Глобальные переменные объявлены в файле options.php
  global $SERVER, $USER, $PASSWD, $DB;

  if (!mysql_connect($SERVER,$USER,$PASSWD))
```



http://all-ebooks.com

```
{
  echo $head;
  echo "<h1>Ошибка сервера MySQL. MySQL server error</h1>";
  echo "</body></html>";
  exit;
}
mysql_select_db($DB);

if($ct==="BK") echo "<h1>Книги</h1>";
else echo "<h1>Компакт-диски</h1>";

echo "<table WIDTH=100% border=2 cellspacing=0 cellpadding=3
  bordercolor=#336699>";

echo "<tr><td bgcolor=#336699><font color=white>Наименование</td>
  <td bgcolor=#336699><font color=white>Цена</td>
  <td bgcolor=#336699><font color=white>Заказ</td>
  </tr>";

$r=mysql_query("select * from tovar where cat=\"$ct\"");

echo "<form method=post action=index.php>";

echo mysql_error();

for ($i=0; $i<mysql_num_rows($r); $i++)
{
  echo "<tr>";
  $f=mysql_fetch_array($r);
  echo "<td>$f[opis]</td><td>$f[price]</td></tr>";
}

echo "</table>";
echo "<p><input type=submit name=buy value=Заказать>
  <input type=Reset value=Сброс></form>
<p>&copy Денис Колисниченко (dhsilabs at mail.ru), 2007</body></
html>";
}
```

Что ж, вывести список товаров действительно очень просто: достаточно подключиться к серверу, выбрать базу данных, выполнить запрос

```
select * from tovar where cat="категория"
```

и оформить результат в виде таблицы. А вот как организовать выбор товара?

Можно, например, поступить так: каждую строку в списке сделать ссылкой на сценарий, который будет заносить выбранный товар во временное хранилище на нашем диске или в Cookies браузера пользователя; чтобы вернуть-

ся к просмотру списка, пользователю придется нажимать кнопку «Назад». Согласитесь, этот способ не очень удобен.

Я предлагаю сделать следующее: возле каждого наименования товара вывести независимый переключатель. Пользователь отмечает выбранные товары и нажимает кнопку «Заказать». По нажатии этой кнопки значения флажков передаются нашему сценарию. Осталось понять, как именовать флажки, ведь конструкция `name = checkboxID` (где ID — идентификатор товара в таблице) выглядит весьма неуклюже.

Разработчики PHP об этом, разумеется, позаботились. Напоминаю (п. 6.2), что из формы можно передавать сценарию не только скалярные переменные, но и массивы. Мы будем передавать массив `Arr[]`, значениями элементов которого будут ID отмеченных товаров.

Модифицируем функцию `ShowTovar()` так, чтобы она отображала флажок около каждой товарной позиции:

```
for ($i=0; $i<mysql_num_rows($r); $i++)
{
  echo "<tr>";
  $f=mysql_fetch_array($r);
  echo "<td>$f[opis]</td><td>$f[price]</td>
  <td><input type=checkbox name=Arr[] value=".$f['id']."</tr>";
}
```

Продолжим создание нашего магазина. Для сохранения регистрационных данных пользователя в Cookies браузера будем использовать следующие переменные:

- `myshop_name` — имя пользователя;
- `myshop_familia` — фамилия;
- `myshop_otch` — отчество;
- `myshop_email` — электронный адрес;
- `myshop_country` — страна;
- `myshop_city` — город;
- `myshop_address` — адрес;
- `myshop_zip` — почтовый индекс.

Префикс «myshop_» необходим, чтобы наши имена переменных не пересекались с данными других сценариев, которые, возможно, используют эти же имена переменных.



http://all-ebooks.com

Первым делом нужно проверить, зарегистрирован ли пользователь. Для этого нужно проверить существование одной из двух переменных: `$register` (имя кнопки в форме регистрации) или `$myshop_email` (единственное поле, заполнение которого обязательно).

**Листинг 28.5. Регистрация пользователя**

```
// Проверяем, зарегистрирован ли пользователь
if (!isset($myshop_email))
{
  echo $head;
  echo "<p>Вы не зарегистрированы. Пожалуйста, оставьте
    информацию о себе. <PRE>";

  echo "<form action=index.php method=post>
   <table border=0 width=50%>
    <tr>
    <td>Ваше имя <td><input type=text name=myshop_name>
    <td>Фамилия <td><input type=text name=myshop_familia>
    <tr>
    <td>Отчество <td><input type=text name=myshop_otch>
    <td>E-mail <td><input type=text name=myshop_email>
    <tr>
    <td>Страна <td><input type=text name=myshop_country>
    <td>Город <td><input type=text name=myshop_city>
    <tr>
    <td>Адрес  <td><input type=text name=myshop_address>
    <td>Индекс <td><input type=text name=myshop_zip>
   </table>
  <p><input type=submit name=register value=OK>
  ";
}
else
{
 // E-mail указан, значит, пользователь заполнил форму.
 // Далее проверяем, на какие кнопки пользователь нажимал:
 // если на кнопку «Register», то записываем рег. данные в Cookies
  if (isset($register))
  {
    SetCookie("myshop_name",$myshop_name);
    SetCookie("myshop_familia",$myshop_family);
    SetCookie("myshop_otch",$myshop_otch);
    SetCookie("myshop_email",$myshop_email);
    SetCookie("myshop_country",$myshop_country);
    SetCookie("myshop_city",$myshop_city);
    SetCookie("myshop_address",$myshop_address);
    SetCookie("myshop_zip",$myshop_zip);
```

http://all-ebooks.com

```
  }
// Далее проверяем, выбрал ли пользователь категорию (если да, то
// выводим товары этой категории) и нажал ли кнопку «Заказ» (если да,
// то выводим счет и посылаем уведомление о заказе). См. листинг 28.6.
```

**Листинг 28.6. Сценарий index.php — полный исходный код**

```
<?
// Подключаем вспомогательные файлы
include "header.php";
include "options.php";

//==================================
// Функция выводит товар указанной категории, по умолчанию - CD
function ShowTovar($ct="CD")
{
  // Глобальные переменные объявлены в файле options.php
  global $SERVER, $USER, $PASSWD, $DB;

  if (!mysql_connect($SERVER,$USER,$PASSWD))
  {
    echo $head;
    echo "<h1>Ошибка сервера MySQL. MySQL server error</h1>";
    echo "</body></html>";
    exit;
  }
  mysql_select_db($DB);

  if($ct==="BK") echo "<h1>Книги</h1>";
  else echo "<h1>Компакт-диски</h1>";

  echo "<table WIDTH=100% border=2 cellspacing=0 cellpadding=3
    bordercolor=#336699>";

  echo "<tr><td bgcolor=#336699><font color=white>Наименование</td>
    <td bgcolor=#336699><font color=white>Цена</td>
    <td bgcolor=#336699><font color=white>Заказ</td>
    </tr>";

  $r=mysql_query("select * from tovar where cat=\"$ct\"");

  echo "<form method=post action=index.php>";
  echo mysql_error();
  for ($i=0; $i<mysql_num_rows($r); $i++)
  {
    echo "<tr>";
    $f=mysql_fetch_array($r);
    echo "<td>$f[opis]</td><td>$f[price]</td>
```



http://all-ebooks.com

```
    <td><input type=checkbox name=Arr[] value=".$f['id']."</tr>";
  }
  echo "</table>";
  echo "<p><input type=submit name=buy value=Заказать>
    <input type=Reset value=Сброс></form>
  <p>&copy Денис Колисниченко (dhsilabs at mail.ru), 2003</body></
html>";
}

//==================================
// Основная программа

// Проверяем, зарегистрирован ли пользователь
if (!isset($myshop_email))
{
  echo $head;
  echo "<p>Вы не зарегистрированы. Пожалуйста, оставьте
    информацию о себе. <PRE>";

  echo "<form action=index.php method=post>
   <table border=0 width=50%>
    <tr>
    <td>Ваше имя <td><input type=text name=myshop_name>
    <td>Фамилия <td><input type=text name=myshop_familia>
    <tr>
    <td>Отчество <td><input type=text name=myshop_otch>
    <td>E-mail <td><input type=text name=myshop_email>
    <tr>
    <td>Страна <td><input type=text name=myshop_country>
    <td>Город <td><input type=text name=myshop_city>
    <tr>
    <td>Адрес  <td><input type=text name=myshop_address>
    <td>Индекс <td><input type=text name=myshop_zip>
   </table>
  <p><input type=submit name=register value=OK>
  ";
}
else
{
  // E-mail указан, значит, пользователь заполнил форму.
  // Далее проверяем, на какие кнопки пользователь нажимал:
   // если на кнопку «Register», то записываем рег. данные в
Cookies
  if (isset($register))
  {
    SetCookie("myshop_name",$myshop_name);
    SetCookie("myshop_familia",$myshop_family);
    SetCookie("myshop_otch",$myshop_otch);
    SetCookie("myshop_email",$myshop_email);
```

http://all-ebooks.com

```
    SetCookie("myshop_country",$myshop_country);
    SetCookie("myshop_city",$myshop_city);
    SetCookie("myshop_address",$myshop_address);
    SetCookie("myshop_zip",$myshop_zip);
}

// Первый вывод нужно сделать только ПОСЛЕ установки Cookies,
// иначе они не
// будут установлены
echo $head;

// Пользователь выбрал товар и нажал кнопку «Заказ»?
if(isset($buy))
{
  // начинаем формировать тело письма с уведомлением о заказе
  $message = "Здравствуйте, $myshop_name!\n".date("d.m.Y") ." вы
     посетили наш Internet-магазин\n";

  $message = $message."Ваш заказ:\n";

  $q = "select * from tovar where";
  foreach($Arr as $k=>$v) $q=$q." (id=$v) or ";
    $q = substr($q,0,strlen($q)-4);

  if(!mysql_connect($SERVER,$USER,$PASSWD))
  {
    echo $head;
    echo "<h1>Ошибка сервера MySQL. MySQL server error</h1>";
    echo "</body></html>";
    exit;
  }

  mysql_select_db($DB);
  $r=mysql_query($q);

  // выводим заказ в окно браузера
  echo " <h1>Ваш заказ:\n</h1>";
  echo "<table WIDTH=100% border=2 cellspacing=0 cellpadding=3
    bordercolor=#336699>";

  echo "<tr><td bgcolor=#336699><font color=white>Наименование</td>
    <td bgcolor=#336699><font color=white>Цена</td></tr>";
  // добавляем заказанные позиции в окно браузера и в тело
  // письма,
  // одновременно вычисляя стоимость заказа
  $sum = 0;
  for ($i=0; $i<mysql_num_rows($r); $i++)
  {
```



http://all-ebooks.com

```
      echo "<tr>";
      $f=mysql_fetch_array($r);
      echo "<td>$f[opis]</td><td>$f[price]</td>" ;
      $message=$message."$f[opis] \t\t $f[price]\n" ;
      $sum+=$f['price'];
      echo "</tr>";
    }
    echo "</table>";
    echo "<p>Общая сумма заказа: $sum";
    // заканчиваем формирование письма и отсылаем его
    $message=$message."\nОбщая сумма: $sum\n
Для подтверждения заказа просто нажмите Reply\n\n
Спасибо за посещение нашего магазина!
----------------------------------------
Регистрационная карточка:\n
Фамилия $myshop_family\n
Имя $myshop_name\n
Отчество $myshop_otch\n
Email $myshop_email\n
Страна $myshop_country\n
Город $myshop_city\n
Адрес $myshop_address\n
Индекс $myshop_zip\n
";

  mail($myshop_email,"MySHOP: ваш заказ",$message,"From: $REPLY");
    echo "<p><b>Вам отправлено уведомление о заказе\n";

    exit;
  } // if(isset($buy))
  else
  {
    // Выводим категории или товар определенной категории
    if(!isset($myshop_cat))
    {
      echo "<h2>Вы хотите купить:</h2>
      <ul>
      <li><a href=index.php?myshop_cat=CD>Компакт-диск</li></a>
      <li><a href=index.php?myshop_cat=BK>Книгу</li></a>
      </ul>";
    }
    else
    {
      ShowTovar($myshop_cat);
    }
  }
}
?>
```

Вот теперь можно проверить работу сценария. Введите в адресную строку адрес сценария магазина `http://ваш_сайт/myshop/index.php`. Зарегистрируйтесь (рис. 28.2), выберите категорию товара (рис. 28.3), отметьте нужные товары (рис. 28.4) и нажмите кнопку «Заказать». Сценарий покажет вам ваш заказ (рис. 28.5), сообщит его стоимость и отправит уведомление о заказе. При получении уведомления вам достаточно нажать кнопку «Reply» для подтверждения заказа.

Готовую версию сценария можно взять на моем сайте `http://www.dkws.org.ua`. Чтобы установить его, скопируйте сценарий в каталог `$DocumentRoot/myshop`, отредактируйте файл `options.php` и запустите сценарий `setup.php`. Если ваш хостер не позволяет создавать базы данных, модифицируйте `setup.php` для работы с предоставленной вам базой или создайте таблицу вручную.

Вам для самостоятельной работы остается сценарий, позволяющий редактировать таблицу товаров.

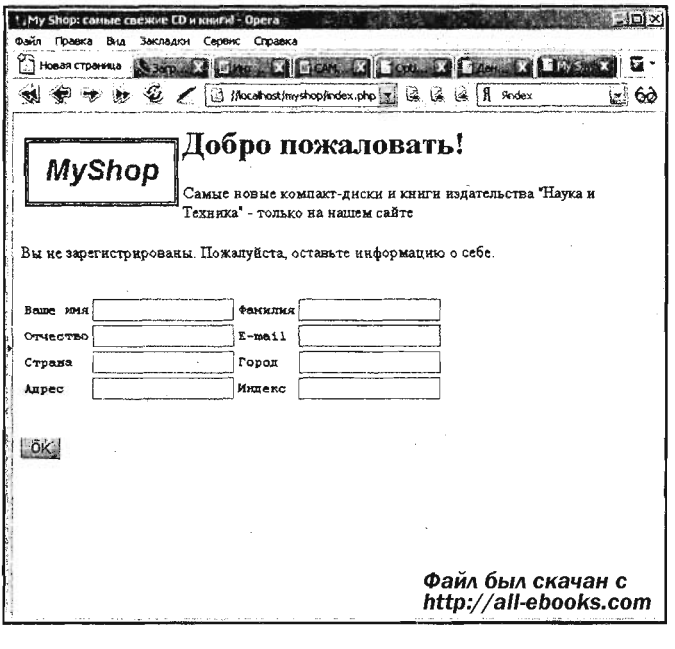



***Рис. 28.2. Регистрация в интернет-магазине***



http://all-ebooks.com

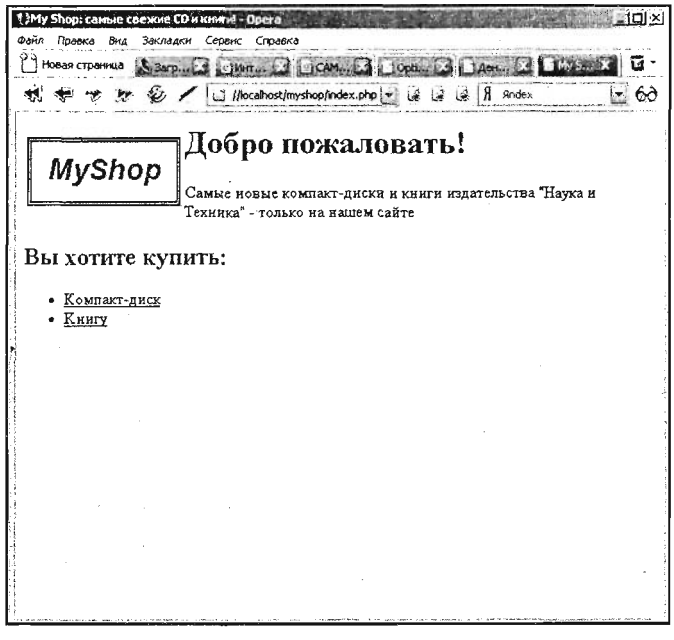



*Рис. 28.3. Выбор категории товара*

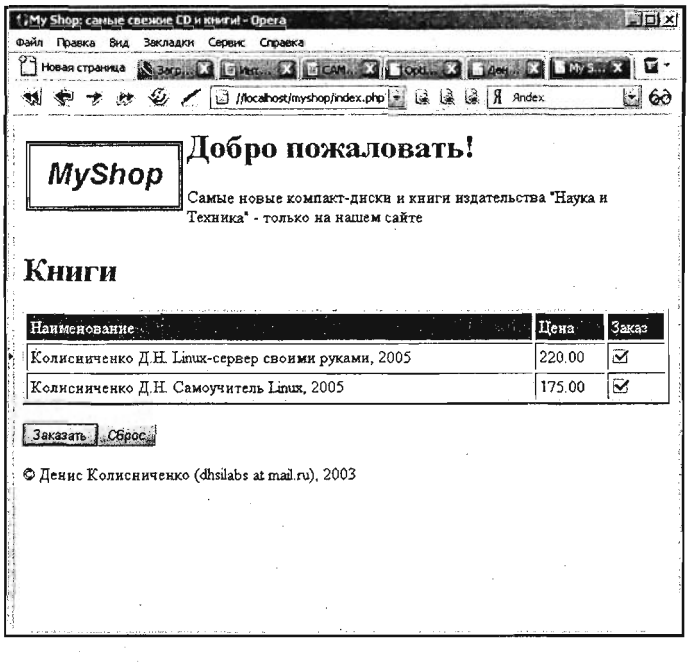
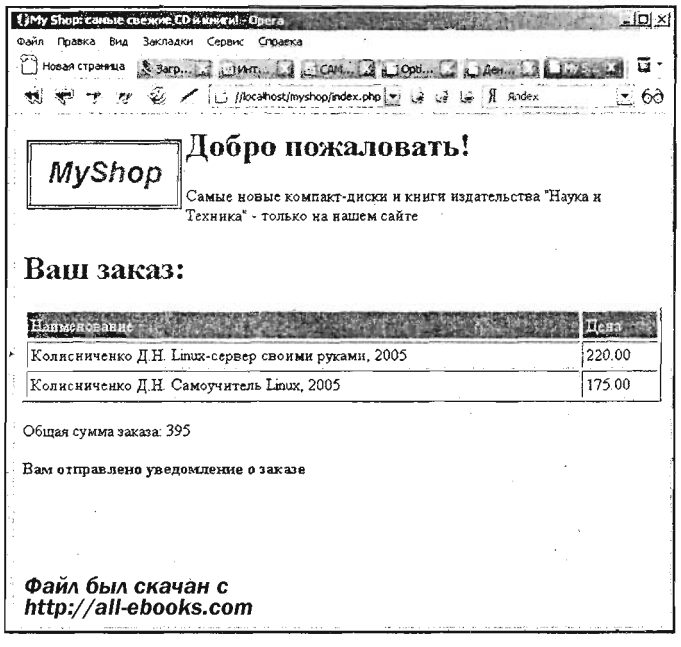



*Рис. 28.4. Выбор товара*

http://all-ebooks.com




*Файл был скачан с*
*http://all-ebooks.com*

***Рис. 28.5. Счет***



http://all-ebooks.com

# ГЛАВА 29.

# ДИСТАНЦИОННОЕ ОБУЧЕНИЕ – СИСТЕМА ПРОВЕРКИ ЗНАНИЙ

http://all-ebooks.com

## 29.1. ОБЩИЕ ПОЛОЖЕНИЯ

В прошлой главе мы рассмотрели интернет-магазин – веб-сервис, позволяющий совершать покупки, не выходя из дома. Но тратить деньги – не самое привлекательное из того, что можно делать, не вставая с дивана. Почему бы не научиться чему-нибудь полезному? В этой главе мы смоделируем систему дистанционного обучения – аналог интернет-магазина для тех, кто предпочитает запасаться не товарами, а знаниями.

Дистанционная форма обучения позволяет посещать лекции, не отрываясь от своего компьютера. Она вовсю набирает обороты и, скорее всего, получит официальное признание наряду со стационарной и заочной формами. Говорят, правда, что дистанционное обучение неэффективно, поскольку нет живого общения преподавателя и студентов. Лично я считаю, что живое общение нужно в школе, где учитель не только обучает, но и воспитывает учеников. В ВУЗы поступают люди, желающие чего-нибудь добиться, поэтому живое общение необязательно: воспитывать восемнадцатилетнего человека уже поздно. Вполне достаточно, если учитель подготовит студентам материалы для самостоятельного изучения, а потом проверит, как студенты эти материалы усвоили.

Для поклонников живого общения можно предложить системы обучения, использующие передачу видео и речи в реальном времени. При использовании таких систем студенты видят своего преподавателя, слышат его голос, могут задать вопросы, на которые преподаватель сразу же ответит. Вот только по диалапу, с трудом вытягивающему 33600 bps, подключаться к видеоуроку бесполезно: ничего не увидишь и не услышишь. А полумеры



http://all-ebooks.com

– общение в чате или просмотр скачанного видеоролика с лекцией – живого общения не заменят.

Самое трудное при удаленном обучении – это организовать проверку знаний. С одной стороны, компьютерный тест может оценить знания студента объективнее, чем живой эмоциональный экзаменатор. С другой стороны, где гарантия того, что при сдаче теста студент не откроет книгу и не перепишет правильные ответы? Поэтому обычные программы дистанционного обучения только передают студенту учебный материал и проверяют его получение (скачал ли студент лекцию №5 или даже не заходил на сайт?), но на экзамен студенту приходится являться лично. На экзамене, поскольку речь идет все-таки о программе, студент сдает тест автоматизированной системе, а роль преподавателя сводится к присмотру за шпаргалками.

В этой главе мы займемся именно экзаменующим модулем системы дистанционного обучения. Такие модули называются системами проверки знаний (СПЗ).

## 29.2. ПОСТАНОВКА ЗАДАЧИ

Мы собираемся написать СПЗ (назовем ее SmartTest), позволяющую публиковать тесты и отвечать на них удаленно. Это очень простая СПЗ. В ней предусмотрены следующие функции:

- **Восстановление после обрыва соединения** (считаем, что пользователь работает по модемному каналу, где обрывы случаются довольно часто; в локальной сети эта функция бесполезна). Если пользователь не закрыл окно браузера, то после восстановления соединения тест будет продолжен с места обрыва. Реализуется функция через механизм сессий, описанный в гл. 26.
- **Защита от несанкционированного доступа.** Пользователь, перед тем как пройти тест, должен быть зарегистрирован в таблице пользователей SmartTest. Добавить запись в эту таблицу может администратор СПЗ. Пароли передаются по сети в зашифрованном виде (используется алгоритм MD5) — не беспокойтесь, ваш пароль не увидит ни один сетевой сниффер.
- **Несколько вариантов правильных ответов.** Максимальное количество вариантов ответа — 6.
- **Для каждого теста ведется таблица ответов.** В нее заносится имя пользователя и его оценка. Пользователь может пройти тест только

http://all-ebooks.com

один раз. Если же ему нужно пересдать тест, тогда администратору нужно удалить его имя из таблицы ответов определенного теста.

- **Номера вопросов генерируются случайным образом.**

## 29.3. НЕОБХОДИМЫЕ ТАБЛИЦЫ

Подробно описывать работу сценария и комментировать сам сценарий, как в случае с интернет-магазином, я не стану. У вас уже достаточно квалификации, чтобы самостоятельно во всем разобраться. Подробно рассмотрим лишь структуру таблиц СПЗ SmartTest.

Сценарий `create_tables.php` создает таблицы:

- `ureg` — таблица регистрации пользователей;
- `reg` — таблица регистрации тестов;
- таблицы вопросов для зарегистрированных тестов;
- таблицы ответов.

Таблица `ureg` содержит логин, пароль и полное имя пользователя (дополнительные поля можно добавить по желанию). Эта таблица создается SQL-запросом:

```
CREATE TABLE ureg (
  login char(16) PRIMARY KEY,
  passwd char(128) NOT NULL,
  fullname char(32)
)
```

Чтобы опубликовать тест, его нужно зарегистрировать в системе, то есть внести в таблицу `reg`. Эта таблица создается следующим запросом:

```
CREATE TABLE reg (
  no int PRIMARY KEY,
  desc char(32),
  qmax int,
  tbl char(64),
  author char(32)
)
```

Поле NO — это порядковый номер теста, DESC — описание, QMAX — количество вопросов (необходимо при выборе случайного вопроса, чтобы выбранный номер не вышел за пределы допустимого). Поле AUTHOR содержит информацию об авторе теста.



http://all-ebooks.com

Вопросы теста хранятся в таблице, имя которой указано в поле TBL. Имя таблицы ответов формируется из имени таблицы вопросов дописыванием к нему суффикса «_a». Например, если вопросы теста находятся в таблице «linux», то ответы – в таблице «linux_a».

Таблица вопросов создается следующим запросом:

```
CREATE TABLE linux (
  no int PRIMARY KEY,
  quest TEXT,
  true char(6),
  dif char(1)
)
```

Поле NO — порядковый номер вопроса. Поле QUEST содержит текст вопроса вместе с вариантами ответов, например:

Какие фирмы участвовали в создании OS/2?

1. Microsoft
2. IBM
3. RedHat
4. Intel
5. Apple

Поле TRUE содержит номера правильных вариантов ответа. Если правильных вариантов несколько (в приведенном вопросе – 1 и 2), то они перечисляются без пробела: «12».

Поле DIF определяет сложность вопроса. Если его значение – «Y», то вопрос считается сложным и правильный ответ на него приносит студенту 3 балла. Простой вопрос приносит только 1 балл.

Таблица ответов еще более проста:

```
CREATE TABLE linux_a (
  login char (16) NOT NULL PRIMARY KEY,
  mark int,
  true int
)
```

Первое поле — это логин студента, второе — его оценка, третье — количество правильных ответов.

Сценарий создания таблиц приведен в листинге 29.1.

http://all-ebooks.com

**Листинг 29.1. Сценарий создания таблиц**

```
<?
mysql_connect("localhost");
mysql_select_db("smarttest");

// Создаем таблицу регистрации пользователей
mysql_query("CREATE TABLE ureg(login char(16)  PRIMARY  KEY,
  passwd char(128) NOT NULL, fullname char(32))");

// регистрируем пользователя
$pswd=md5("123456");
mysql_query("INSERT   INTO   ureg   values('user3','$pswd','Ivanov
Ivan')");

// Создаем таблицу регистрации тестов
mysql_query("CREATE TABLE reg (no int PRIMARY KEY, desc
  char(32),qmax int, tbl char(64), author char(32))");

// регистрируем тест на 100 вопросов
mysql_query("INSERT INTO reg VALUES('1','Linux Test', '100',
  'linux', 'Denis')");

// Создаем таблицы вопросов и ответов
mysql_query("CREATE TABLE linux(no int PRIMARY KEY, quest TEXT,
  true char(6), dif char (1))");
mysql_query("CREATE TABLE linux_a(login char(16) NOT NULL
  PRIMARY KEY, mark int, true int)");

// Вносим в таблицу 100 вопросов
for($i=0; $i<100; $i++)
{
  //
  $msg="Вопрос $i<br>Сколько вам лет?<br>$i?";
  echo $msg;
  if ($i%2==0) $ch='Y'; else $ch='N';
  mysql_query("INSERT INTO linux VALUES($i,'$msg','1','$ch')");
}
?>
```

## 29.4. ПИШЕМ СЦЕНАРИЙ

Я предлагаю следующую схему работы системы тестирования.

1. Пользователь регистрируется. Если логин или пароль неправильны, записываем в журнал попытку несанкционированного доступа и завершаем работу.



http://all-ebooks.com

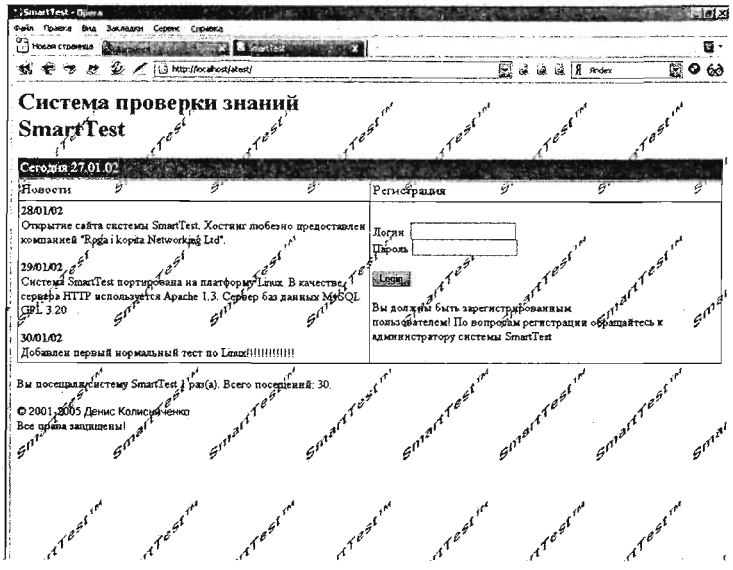



*Рис. 29.1. Регистрация в системе*

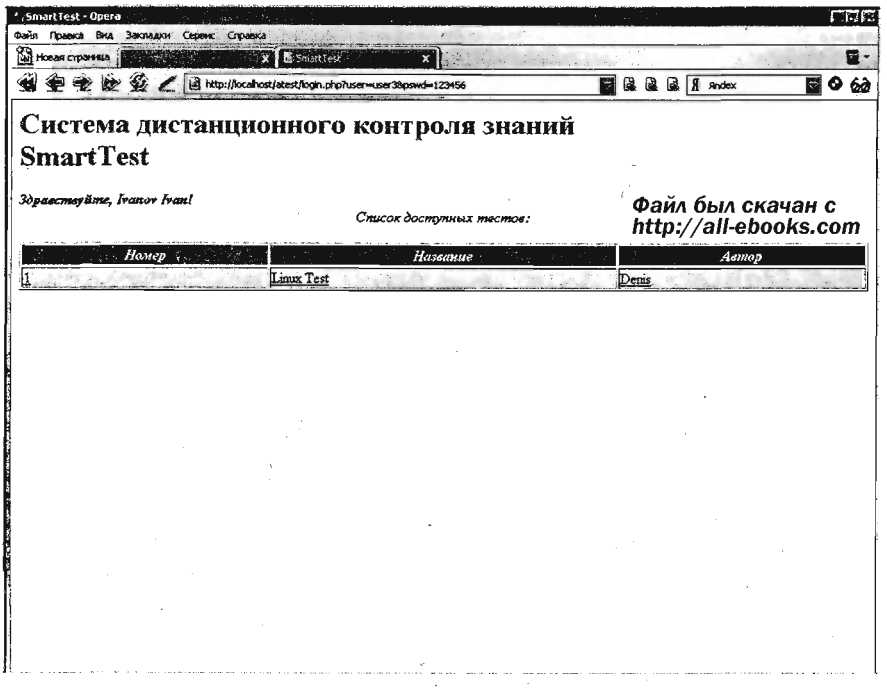



*Рис. 29.2. Список доступных тестов*

2. Выводим список тестов. Пользователь выбирает один из них. Если он уже проходил этот тест, то сообщаем ему об этом и просим выбрать другой тест; иначе переходим на шаг 3. Пересдать тест можно только с разрешения администратора, который удалит запись пользователя из таблицы результатов.
3. Генерируем 10 случайных номеров вопросов и записываем их в Cookies браузера.
4. Предъявляем по очереди эти 10 вопросов. Если пользователь ответил правильно, то анализируем сложность вопроса: за сложный вопрос добавляем 3 балла, за простой 1 балл.
5. Когда значение счетчика заданных вопросов достигнет 9 (предъявляемые вопросы нумеруются от 0), выводим результат и записываем его в таблицу ответов. Оценку вычисляем так: если пользователь набрал 90% максимально возможной (с учетом сложности) суммы или более, он получает 5, от 75% до 90% – 4, от 50% до 75% – 3, менее 50% — оценка 2.

Для лучшего понимания я разбил код сценария на две части: авторизацию пользователя (листинг 29.2) и собственно тестирование (листинг 29.3).

**Листинг 29.2. Сценарий авторизации пользователя**

```
<?
$LoginScript="$HTTP_HOST/smarttest/login.php";
$NewsFile="news.html";

// Устанавливаем cookie для индивидуального счетчика посещений
if(!isSet($STCount)) $STCount=0;
$STCount++;
SetCookie("STCount",$STCount,0x7FFFFFFF);

// Устанавливаем общий счетчик посещений
$f=fopen("counter.dat","a+");
flock($f,2);
$count=fread($f,100);
// Не засчитываем посещения со своего узла
$REMOTE_HOST=gethostbyaddr($REMOTE_ADDR);
if(!($HTTP_HOST===$REMOTE_HOST)) @$count=$count+ 1;
ftruncate($f,0);
fwrite($f,$count);
flock($f,3);
fclose ($f);

// Читаем файл с новостями
if (file_exists($NewsFile))
```



http://all-ebooks.com

```
{ $news = join('', file($NewsFile)); }
else $news="No news today";
// Формируем заголовок главной страницы СПЗ
echo "<html><head><meta http-equiv=Content-Type
  content=\"text/html; charset=windows-1251\">";
echo  "<title>SmartTest</title><link  rel=stylesheet  type=\"text/
css\"
  href=\"st.css\">";
echo "</head><body text=#000000 bgcolor=#FFFFFF
   link=#0000EE  vlink=#551A8B  link=#FF0000  background=smarttest.
jpg>";
// Формируем главную страницу СПЗ, содержащую также блок новостей
echo "<h1>Система проверки знаний<br>SmartTest</h1>";
echo "<table WIDTH=100% BGCOLOR=#336699><tr><td>";
echo "<font color=FFFFFF><font size=+1>".date("Сегодня d.m.Y").
  "</font></font></td></tr></table>";
echo "<table WIDTH=100% border=1 cellspacing=0
  cellpadding=3 bordercolor=#336699>";
echo "<tr><td width=50%><font color=#336699>
  <font size=+1>Новости</font></font></td>";
echo "<td width=50%><font color=#336699>
  <font size=+1>Регистрация</font></font></td></tr><tr><td>";
echo $news;
// Выводим регистрационную форму
echo "<td><form action=http://$LoginScript>";
echo "<br> Логин  <input type=text name=user>
  <br>Пароль <input type=password name=pswd>";
echo "<br><br><input type=submit value=Login><p>Вы должны
     быть  зарегистрированным  <br>пользователем!  По  вопросам
регистрации
   обращайтесь  к  <a  href=mailto:st-admin@host.com>администратору
системы
SmartTest</a>";
echo "</td></tr></table>";
echo "<p>Вы посещали систему SmartTest $STCount раз(а). Всего
  посещений: $count.";
echo "<p>&copy 2001-2003 <a href=http://dkws.narod.ru>Денис
  Колисниченко</a>";
</body></html>";
?>
```

**Листинг 29.3. Сценарий системы проверки знаний SmartTest**

```
<?
// Открываем сессию smarttest
session_name("smarttest");
session_start();
session_register("Login");
```

```
session_register("FullName");
session_register("UserName");

$Dir="/smarttest/logs";

// текст текущего вопроса. Переменная используется как глобальная
$text="";
//=================================================
// Функция выводит текст вопроса и флажки
// для выбора правильных вариантов ответа.
// Вопрос передается через глобальную переменную $text
function GetQuestText()
{
  global $Counter, $text;
  $tmp=$Counter+1;

  echo "<br><b>Вопрос $tmp</b><br>";
  echo "$text";
  echo "<br>";
  echo "<form action=$SCRIPT_NAME><br>";

  // Отображаем флажки с именами aN, где N от 1 до 6
  echo "<input type=checkbox name=a1 value=1>1
    <input type=checkbox name=a2 value=2>2 ";
  echo "<input type=checkbox name=a3 value=3>3
    <input type=checkbox name=a4 value=4>4 ";
  echo "<input type=checkbox name=a5 value=5>5
    <input type=checkbox name=a6 value=6>6 ";
  echo "<br><input type=submit value=\"Ответить!\">";
//echo "</form></body></html>";
}

//=================================================
// Функция подключения к серверу
function connect()
{
  mysql_connect("localhost", "root", "") or die(mysql_error());
  mysql_select_db("st");
}

//=================================================
// Функция выводит список тестов в виде таблицы
function GetTestList()
{
  global $SCRIPT_NAME;
  echo "<center><i>Список доступных тестов :</i></center><br>";
  echo "<table BORDER COLS=3 WIDTH=\"100%\" BGCOLOR=#CEFFCE>";
  echo "<tr ALIGN=CENTER BGCOLOR=#3333FF>";
  echo "<td><b><i><font color=#FFFFFF>Номер</font></i></b></td>"
```



http://all-ebooks.com

```
  echo "<td><b><i><font color=#FFFFFF>Название</font></i></b></td>";
  echo "<td><b><i><font color=#FFFFFF>Автор</font></i></b></td>"

  $res=mysql_query("SELECT no,desc,author FROM reg");
  while ($Row=mysql_fetch_row($res))
  {
    echo "<tr>";
    for($i=0; $i<mysql_num_fields($res); $i++)
    echo "<td><a href=$SCRIPT_NAME?tno=$Row[0]>$Row[$i]</a></td>";
    echo "</tr>";
  }
  echo "</table>";
}

//=================================================
// Функция проверяет введенные пользователем логин и пароль. Если
// они правильны, она присваивает глобальной переменной Login
// значение 1
function Register()
{
  global $user, $pswd, $Accept, $Login;
  global $UserName, $FullName;

  $Login=0;
  $res=mysql_query("SELECT * from ureg");

  while ($Row=mysql_fetch_row($res))
  {
    if($Row[0]==$user)
    {
      if($Row[1]===md5($pswd))
      { $FullName=$Row[2];
        $UserName=$user;
        $Login=1;
        break;
      }
    }
  }
}


//=================================================
// Основная программа
if ($Login==0)
{
  // Неправильные имя пользователя и/или пароль
  echo "<html><head>
  echo "<h1>Система дистанционного контроля знаний<br>SmartTest</h1>";
  echo "<center><img src=stop.jpg alt = STOP><h2>Access denied.</h2>";
```

```
  echo "<p>Проверьте имя пользователя и пароль!";
  echo "</body></html>";

  unset($Accept);
  session_unset();

  // запись попытки несанкционированного доступа в $Dir/access.log
  global $Dir;
  global $REMOTE_ADDR;

  @mkdir($Dir,755);
  $f=fopen("$Dir"."/access.log","a+") or Die("Cannot create
access.log");
  flock($f,2);
  $dt=date("d.m.y H.i.s");
  if(PHP_OS=="Linux") $NL="\n"; else $NL="\n\r";
   fputs($f,"$dt Access denied for user $user [$pswd]. IP:
$REMOTE_ADDR $NL");
  flock($f,3);
  fclose($f);
  exit;
}
else
{
  // Все Ок, выводим список тестов
  echo "<title>SmartTest</title></head><body bgcolor=white>";
   echo "<h1>Система дистанционного контроля знаний<br>SmartTest</
h1>";
  echo "<B><I>Здравствуйте, $Row[2]!</B></I><BR>";

  GetTestList();
  return;
}

connect();
if(!isSet($Login))
{
  Register();
}
else
{

  // Регистрируем в сессии служебные переменные
  session_register("TestNo");
  session_register("Arr");
  session_register("Counter");
  session_register("Table");
```



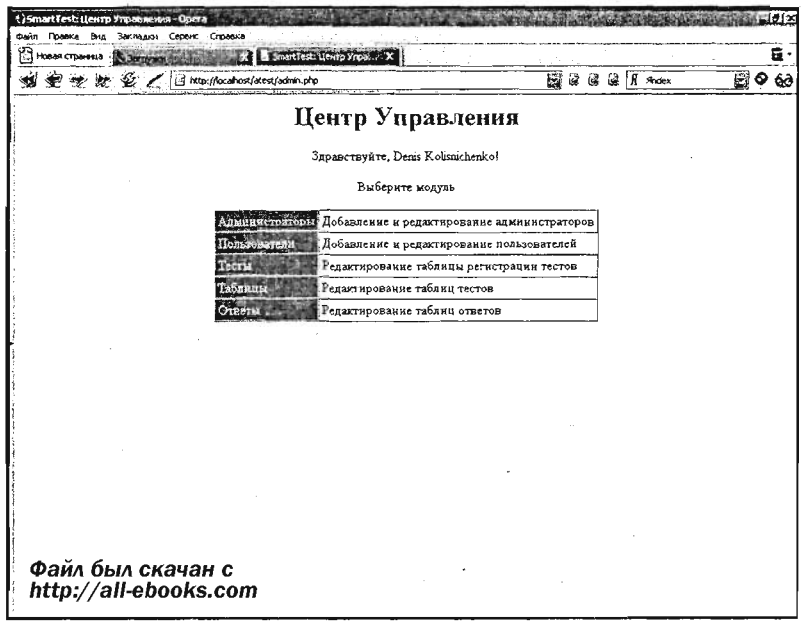
http://all-ebooks.com

```
session_register("True");
session_register("Max");
session_register("Prev");
session_register("Diff");
session_register("Total");
session_register("TT");

// При первом запуске устанавливаем номер теста и
// инициализируем базу
if(!isSet($TestNo))
{
  $Res=mysql_query("SELECT no,qmax,tbl FROM reg WHERE no=$tno");
  $Row=mysql_fetch_row($Res);
  $Max=$Row[1];
  $Table=$Row[2]; // имя таблицы вопросов
  $ATable=$Table."_a"; // имя таблицы ответов
  $Diff=$Row[3];
  $QDiff=0; $Total=0;
   $R=@mysql_query("SELECT * FROM $ATable") or die("Invalid SQL
query
      [$ATable]");

  while($Rw=mysql_fetch_row($R))
  {
    if($Rw[0]===$UserName)
    {
      echo "Вы уже проходили выбранный тест<br>";
      echo "Ваша оценка $Rw[1]. Вы правильно ответили на $Rw[2]
        вопросов(а)";
      GetTestList();
      exit;
    }
  }

  $TestNo=tno;
  $Counter=0;

  // получаем 10 случайных номеров; все номера должны быть
  // различны,
  // для проверки этого служит вспомогательная функция checkarr()
  function checkarr($Arr)
  {
    // возвращает $Res=1, если номера различны, и $Res=0, если
    // найдены совпадающие;
    $Res=1;
    for ($i=0; $i<10; $i++)
      for ($j=$i+1; $j<10; $j++)
        if($Arr[$i]===$Arr[$j])  { $Res = 0; return $Res; }
    return $Res;
```

```
    }

        // настраиваем генератор случайных чисел на новую
последовательность
    mt_srand(time()+(double)microtime()*1000000);
    // получаем массив номеров $Quest
    for ($i=0; $i<9;$i++) $Quest[]=mt_rand(0, $Max);
    while(checkarr($Quest)==0)
    {
      for ($i=0;$i<9;$i++) $Quest[$i]=mt_rand(0, $Max);
    }

    // записываем полученный массив в Cookie для возобновления
    // прерванного теста, предварительно преобразовав его в строку
    $Arr=serialize($Quest);

    // задаем первый вопрос
            $Res=mysql_query("SELECT * FROM $Table WHERE
no=$Quest[$Counter]");
    $Row=mysql_fetch_row($Res);
    $Prev=$Row[2];
    $text=$Row[1];
    $Diff=$Row[3];
    GetQuestText();
  } // if (!isSet($TestNo))
  else
  {
    // не первый запуск: получен ответ на предыдущий вопрос
    if(!isSet($a1)) $a1=""; if(!isSet($a2)) $a2="";
    if !isset($a3)) $a3=""; if(!isSet($a4)) $a4="";
    if(!isSet($a5)) $a5=""; if(!isSet($a6)) $a6="";
     // сравниваем ответы, преобразованные в строку, со значением
поля TRUE
    // (переменная $Prev)
    $Answer=$a1.$a2.$a3.$a4.$a5.$a6;
    if ($Answer===$Prev)
    {
      // количество правильных ответов
      $True=@$True+1;
      $Trusted='Y';
      // сумма с учетом сложности
      if ($Diff=='Y') $Total=$Total+3;
      else $Total=$Total+1;
    } else $Trusted='N';
    // максимальная сумма с учетом сложности
    if ($Diff=='Y') $TT=$TT+3;
    else $TT=$TT+1;

    if($Counter===9)
```



http://all-ebooks.com




Файл был скачан с
http://all-ebooks.com

***Рис. 29.3. Центр управления системой проверки знаний***

```
    {
      // 10 вопросов предъявлены, завершаем работу
      $Exp=($Total*100)/$TT;
      if ($Exp>=90) $Mark=5;
      else if($Exp>=75) $Mark=4;
      else if($Exp>=50) $Mark=3;
      else $Mark=2;

      mysql_query("INSERT INTO $ATable VALUES('$UserName','$Mark',
'$True')");
          $msg="<br>$FullName, вы ответили правильно на $True
вопросов(а)
        $msg=$msg+"<br>Вы набрали $Total баллов из возможных $TT
($Exp)%";
      $msg=$msg+"<br>Ваша оценка: $Mark";
      echo $msg;

      session_unset();
      exit;
    }

    // иначе задаем следующий вопрос
    $Quest=unserialize($Arr);
          $Res=mysql_query("SELECT * FROM $Table WHERE
```

http://all-ebooks.com

```
no=$Quest[$Counter]");
    $Row=mysql_fetch_row($Res);
    $Prev=$Row[2];
    $text=$Row[1];
    $Diff=$Row[3];
    $Counter=$Counter+1;
    GetQuestText();

  } //if(!isSet($TestNo))
} // if(!isSet($Login))
?>
```

Редактировать таблицы пользователей, тестов и вопросов можно с помощью средства phpMyAdmin, обычно предоставляемого хостинг-провайдерами. Но можно и написать собственный сценарий «панели администрирования» СПЗ. Для себя я его написал (рис. 29.3), а вам предоставляю это в качестве упражнения.



http://all-ebooks.com

# ГЛАВА 30.

# ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ И PHP

http://all-ebooks.com

## 30.1. ОСНОВНЫЕ ПРИНЦИПЫ ООП

Объектно-ориентированный подход (ООП) – это стратегия программирования, уделяющая основное внимание не последовательности операций, выполняемых программой, а структурам данных, описывающим объекты предметной области, которую программа должна обслуживать. Идея объектного подхода состоит в том, что программа представляет собой модель некоторого процесса, протекающего в реальном мире. Этот процесс может быть описан как взаимодействие совокупности объектов. У каждого объекта есть состояние, заданное набором параметров; объекту приписан также набор действий, которые тот может выполнять над другими объектами, и действий, которые можно выполнять над ним. Объекты взаимодействуют, посылая друг другу специальные сообщения. Получив сообщение, объект может изменить свое состояние или выполнить какие-либо действия.

**Основные понятия объектно-ориентированного программирования – это класс и объект класса. Классом называется модель некоторого множества однотипных предметов реального мира, а экземпляром класса, или объектом, – модель конкретного предмета из этого множества. Однотипными предметами считаются те, состояние которых описывается одним и тем же набором параметров, способные выполнять один и тот же набор операций. Описание класса состоит в перечислении этих параметров и операций. Объявленные в классе параметры называются членами класса, а функции, реализующие операции, – методами класса.**

С точки зрения языка программирования класс можно рассматривать как тип данных, а объект класса – как переменную этого типа. Таким образом,



http://all-ebooks.com

объектно-ориентированный подход предполагает, что разработчик программы определяет классы используемых в программе объектов, строит их описания и правила взаимодействия, затем создает экземпляры необходимых объектов и предоставляет им взаимодействовать по заданным правилам.

Объектный подход позволяет программисту мыслить категориями объектов предметной области, что уменьшает сложность и увеличивает читаемость программного кода. Программы, написанные с использованием объектного подхода, надежнее и проще в сопровождении, чем написанные с применением традиционного структурного подхода, так как их модульность позволяет вносить независимые изменения в разные части программы, сводя к минимуму риск ошибок программирования.

**Объектно-ориентированное программирование стоит на трех китах: инкапсуляции, наследовании и полиморфизме.**

**Инкапсуляция (encapsulation)** — это объединение мелких деталей в крупный объект, с которым программист работает как с единым целым. При этом подробности о том, как устроен и функционирует сам объект, остаются скрытыми от «внешнего мира», что не только приводит к значительному упрощению программы, но и защищает внутренние данные объекта от вмешательства извне.

В языках, поддерживающих объектный подход, инкапсуляцию обеспечивает возможность объявить члены и методы класса либо открытыми, либо закрытыми. Доступ к закрытым членам и методам класса возможен только изнутри объекта этого класса, а открытые члены и методы доступны из других частей программы.

Объектный синтаксис впервые появился в PHP версии 3, но до версии 5 реализацию инкапсуляции нельзя было считать полноценной: закрытые члены класса не поддерживались. Но в PHP 5, наконец, объектная модель усовершенствовалась настолько, что на PHP стало можно писать с применением тех же приемов, что и на C++.

**Наследование (inheritance)** – это механизм, позволяющий организовать иерархию классов. Говорят, что класс A (производный класс, или потомок) наследует от класса B (базового, или родительского класса), если объекты класса A обладают всеми свойствами и методами объектов класса B, а кроме них еще дополнительными свойствами, характерными только для объектов производного класса. Если рассматривать класс как «чертеж», по которому строятся объекты, то базовый класс – это самый общий план, на котором указаны только свойства, принадлежащие всем разновидностям объектов одной категории, а производные классы – подробные планы каждой отдельной разновидности.

http://all-ebooks.com

Иерархию классов следует проектировать так, чтобы она отражала классификацию объектов предметной области на роды, виды, подвиды и так далее. Количество уровней наследования в языке PHP не ограничено.

**Полиморфизм (polymorphism)** – это способность объектов, принадлежащих к различным классам, вести себя по-разному в ответ на вызов одного и того же метода, общего для всех классов. Принцип полиморфизма опирается на идею наследования: общий метод объявляется в базовом классе, а специфические для каждого класса действия программируются в результате переопределения этого метода в производных классах. Применение принципа полиморфизма позволяет разработать единый интерфейс для одинаковых манипуляций с объектами разной природы, что значительно облегчает написание сложных программ.

## 30.2. СОЗДАНИЕ КЛАССА В PHP

Итак, класс в языке PHP — это своеобразный тип переменной, а объект — это экземпляр класса, то есть переменная этого типа. До того, как использовать переменную-объект, класс необходимо объявить. Объявление класса в PHP начинается ключевым словом `class`:

```
class MyClass {

  // объявление членов класса - переменных любого типа, в том числе объектов
  public $SomeProperty
  // объявление методов класса
  function SomeMethod()
  {
    // тело функции
  }
}
```

В предыдущих версиях PHP члены класса требовалось объявлять с помощью ключевого слова `var`, но в PHP5 конструкция `var` устарела. Вместо нее появилась возможность управлять видимостью членов и методов класса извне, указывая для них так называемые модификаторы доступа: `public`, `private` и `protected`.

Модификатор `private` объявляет элемент класса (член или метод) закрытым. К закрытым элементам можно обращаться только изнутри текущего класса. Обратное действие имеет модификатор `public`, объявляющий элемент класса открытым. К открытым элементам можно обращаться отовсюду. Если не указывать никакого модификатора, то все будет работать так же, как в PHP4, то есть по умолчанию все члены класса открыты.



http://all-ebooks.com

Модификатор `protected` делает элемент класса защищенным. К защищенным членам и методам класса можно обращаться только из текущего класса и его производных классов.

Методы класса объявляются точно так же, как обычные функции. Метод может принимать параметры и возвращать значение.

Для создания объекта класса служит следующий синтаксис:

```
$obj = new MyClass;
```

Создав объект класса, мы получаем доступ к его членам и методам. Для этого служит оператор ->:

```
echo $obj->SomeProperty; // обратите внимание, что перед именем
                         // свойства
                         // не ставится знак $!
$obj->SomeMethod();
```

В отличие от обычной функции, метод класса может использовать не только переданные параметры и локальные переменные, но и члены класса. Для обращения к членам класса – переменным текущего объекта служит специальная переменная `$this`, указывающая на текущий объект класса.

Пример объявления и использования класса приведен в листинге 30.1.

**Листинг 30.1. Пример класса**

```
<?php
class HelloWorld
{
  // член класса
  public $message = 'Приветственное сообщение';
  // метод класса
  function SayHello()
  {
  // метод использует значение переменной $message текущего объекта
    echo $this->message;
  }
}
// создаем два экземпляра класса
$obj1 = new HelloWorld();
$obj2 = new HelloWorld();
// изменяем значение свойства $message для одного из экземпляров
$obj2->message = 'Другое сообщение';

$obj1->SayHello(); // выведет 'Приветственное сообщение'
$obj2->SayHello(); // выведет 'Другое сообщение'
?>
```

## 30.3. НАСЛЕДОВАНИЕ КЛАССОВ

### 30.3.1. Основы наследования

Наследование — это расширение возможностей уже существующего класса. Допустим, класс A обладает свойствами и методами, позволяющими выполнить почти все необходимые нам функции. Нужно создать новый класс B, умеющий все то, что умеет класс A, и еще немного. Нет необходимости в объявлении класса B перечислять свойства и методы класса A, достаточно объявить класс B потомком класса A. Для этого служит ключевое слово `extends`:

```
class BaseClass {
  function SayHello() { echo "Метод SayHello"; }
  function Report() { echo "Метод Report родительского класса"; }
}

class SubClass extends BaseClass {
  // добавляем свойство
  public $NewProperty;
  // добавляем метод
  function SubMethod() { echo "Метод SubMethod класса-потомка"; }
  // переопределяем метод (см. п.30.3.2)
  function Report() { echo "Метод Report класса-потомка"; }
}
```

Класс `BaseClass` называется базовым или родительским классом, класс `SubClass` — производным, или классом-потомком, или подклассом.

Теперь объекты класса `SubClass` могут вызывать как все методы класса `BaseClass`, так и свои собственные методы. Вы можете забыть о том, что класс `SubClass` унаследовал метод `SayHello()` от класса `BaseClass`: снаружи все выглядит так, будто класс `SubClass` реализует этот метод самостоятельно.

### 30.3.2. Переопределение методов

Вы убедились, что принцип наследования действительно позволяет расширить функциональность базового класса посредством добавления в производный класс новых свойств и методов. Более того, унаследованную функциональность можно изменить. Для этого нужно переопределить (override) метод родительского класса, то есть объявить в производном классе собственную версию этого метода с тем же именем и набором параметров. В вышеприведенном примере мы переопределили метод `Report()`:



http://all-ebooks.com

```
$obj1 = new BaseClass;
$obj2 = new SubClass;
$obj1->Report(); // выведет "Метод Report родительского класса"
$obj2->Report(); // выведет "Метод Report класса-потомка"
```

Потребность в переопределении метода возникает чаще всего тогда, когда метод базового класса реализует почти все, что требуется программисту, и производному классу остается нанести последние штрихи. В этом случае не обязательно копировать код переопределяемого метода в тело метода производного класса. Класс-потомок может просто вызвать метод класса-родителя. Для ссылки на метод родителя служит оператор области действия :: (двойное двоеточие):

```
class SubClass extends BaseClass {
  function Report()
  {
    parent::Report(); // вызов метода родительского класса
    // собственный код метода класса-потомка
    echo "Метод Report класса-потомка";
  }
}
```

### 30.3.3. Ключевые слова final и abstract

В PHP версии 5 появилась возможность управлять переопределением методов и наследованием классов с помощью ключевых слов `final` и `abstract`.

Метод, помеченный ключевым словом `final`, запрещено переопределять в наследующем классе. Если вы попытаетесь переопределить `final` метод, то выполнение сценария закончится фатальной ошибкой:

```
class BaseClass {
    final public function NoOverride() {
       echo "Метод BaseClass::NoOverride() переопределить нельзя";
    }
}

class SubClass extends BaseClass {
    public function NoOverride() {
        echo "Переопределенный метод SubClass::NoOverride()";
    }
}
// Ошибка: Cannot override final method BaseClass::NoOverride()
```

Ключевое слово `final` может находиться и перед объявлением класса. От класса, помеченного этим ключевым словом, вообще нельзя наследовать. Следующий пример вызовет ошибку:

```
final class FinalClass { }

class SubClass extends FinalClass { }
```

Ключевое слово `abstract` оказывает в точности обратное действие: метод, помеченный ключевым словом `abstract` (абстрактный метод), необходимо переопределить в наследующем классе. Абстрактный метод не имеет тела: по сути, такой метод – это просто уведомление о том, что в каком-то из потомков базового класса метод с данным именем и набором параметров будет реализован:

```
abstract class AbstractBaseClass {
     abstract function Report();
}
```

Класс, в котором объявлен хотя бы один абстрактный метод, сам является абстрактным, и его объявление должно начинаться с ключевого слова `abstract`. Очевидно, что экземпляр абстрактного класса создать невозможно. Абстрактные классы предназначены только для того, чтобы от них наследовать.

Для чего это нужно? Допустим, вы спроектировали два класса, описывающих объекты двух разных природ, и оказалось, что эти объекты позволяют производить над собой почти одинаковые манипуляции. Тогда, если вы объявите оба класса потомками одного общего предка, в котором объявите абстрактные методы, описывающие эти манипуляции, то вы получите стандартный интерфейс для обращения к объектам обоих классов. То есть в дальнейшем, вызывая метод, объявленный в классе-предке, вам не придется заботиться о том, к какому именно из классов-потомков принадлежит объект. Листинг 30.2 демонстрирует применение этого принципа.

**Листинг 30.2. Пример применения абстрактного класса**

```
<?php

abstract class BaseClass {

  abstract protected function ThisClassIs();

  public function Report() {
    echo "This class is: ".$this->ThisClassIs();
    }

}

class SubClass1 extends BaseClass {
    protected function ThisClassIs() {
```



http://all-ebooks.com

```
        return "SubClass1<br>";
    }
}

class SubClass2 extends BaseClass {

    protected function ThisClassIs() {
        return "SubClass2<br>";
    }
}

$obj1 = new SubClass1;
$obj2 = new SubClass2;
$obj = $obj1;
$obj->Report();
$obj = $obj2;
$obj->Report();
?>
```

Как видите, при вызове метода `Report()` для объекта `$obj` нам не нужно знать, к какому из классов-потомков `BaseClass` принадлежит этот объект: свой вариант метода `Report()` реализован в любом потомке `BaseClass`, поэтому мы можем вызывать эти варианты единообразно. Так в объектных языках реализуется принцип полиморфизма.

Что касается практической ценности примера 30.3, то мы решили с его помощью довольно важную задачу определения, к какому классу относится указанный объект. Но решили мы ее кривым и доморощенным способом: невозможно же в реальном сценарии прописывать в каждом классе метод, докладывающий имя этого класса. На самом деле для решения этой задачи в PHP предусмотрен оператор `instanceof`, возвращающий `true`, если указанный объект относится к указанному классу или одному из его потомков, и `false` в противном случае:

```
if ($obj1 instanceof BaseClass)
     echo '$obj1 is a BaseClass';
else
     echo '$obj1 is NOT a BaseClass'; // выведет: «$obj1 is a
                                      // BaseClass»
if ($obj1 instanceof SubClass1)
     echo '$obj1 is a SubClass1';
else
     echo '$obj1 is NOT a SubClass1'; // выведет: «$obj1 is a
                                      // SubClass1»
if ($obj1 instanceof SubClass2)
     echo '$obj1 is a SubClass2';
else
     echo '$obj1 is NOT a SubClass2'; // выведет: «$obj1 is NOT a
                                      // SubClass2»
```

http://all-ebooks.com

### 30.3.4. Интерфейсы

При построении модели предметной области часто обнаруживаются объекты, обладающие как бы «двумя природами». Классический пример – приложения, описывающие личный состав учебных заведений: есть класс «преподаватель», есть класс «студент» и есть сущность «аспиранты», проявляющая свойства как тех, так и других. От какого класса производить класс «аспирант»?

В языке C++ эту задачу решает идея множественного наследования: классу разрешается наследовать от нескольких базовых классов. В PHP версии 5 и тем более предыдущих версий множественное наследование не поддерживается, то есть класс может иметь только одного родителя. Однако механизм, позволяющий наделить объект двумя разными наборами свойств, в языке PHP предусмотрен. Это механизм интерфейсов.

Интерфейс можно рассматривать как абстрактный класс, не содержащий свойств и реализованных методов, а только константы и прототипы методов (то есть объявления без определения). Все методы интерфейса должны быть открытыми (`public`).

Для объявления интерфейса служит ключевое слово `interface`:

```
interface ISubst
{
   public function SetVariable($name, $value);
   public function SubstValues($str);
}
```

Класс не может наследовать интерфейс, но может реализовать его. Для этого в классе должны быть переопределены и реализованы все методы, которые объявлены в интерфейсе. Если хотя бы один из объявленных методов не реализован, то класс должен быть объявлен абстрактным.

Для заявления о том, что данный класс реализует данный интерфейс, служит ключевое слово `implements`:

```
class SubstClass implements ISubst
{
  // реализация методов, объявленных в ISubst;
  // собственные свойства и методы класса (необязательно)
}
```

Один класс может реализовать сразу несколько интерфейсов. В этом случае их имена нужно перечислить через запятую после ключевого слова `implements`.

Сценарий 30.3 демонстрирует реализацию интерфейса `ISubst` для замены имен переменных в строке значениями этих переменных.



http://all-ebooks.com

**Листинг 30.3. Пример применения интерфейса**

```
interface ISubst
{
   public function SetVariable($name, $value);
   public function SubstValues($str);
}

class SubstClass implements ISubst
{
  // список переменных, подлежащих подстановке значений
  private $vars = array();
  public function SetVariable($name, $value) {
    $this->vars[$name] = $value;
  }

  // функция заменяет в строке-аргументе каждое вхождение имени
  // переменной, взятого в фигурные скобки, на значение этой
  // переменной
  public function SubstValues($str) {
    foreach($this->vars as $name => $value) {
      $str = str_replace('{'.$name.'}', $value, $str);
    }
    return $str;
  }
}

$obj = new SubstClass;
$obj->SetVariable("v1", "foo");
$obj->SetVariable("v2", "bar");
$obj->SetVariable("v3", 42);
$str = 'Значения переменных:<li>v1={v1}<li>v2={v2}<li>v3={v3}';
$str = $obj->SubstValues($str);
echo $str;
?>
```

## 30.4. КОНСТРУКТОР И ДЕСТРУКТОР

Конструктор – это специальный метод класса, вызываемый автоматически каждый раз при создании экземпляра этого класса. В предыдущих версиях PHP имя этого метода должно было совпадать с именем класса, но в PHP 5 конструктор должен называться `__construct()` (два символа подчеркивания в начале слова).

Конструктор не обязан присутствовать в классе, но может оказаться полезным, если вы, например, хотите перед использованием объекта присвоить его свойствам начальные значения (иными словами, инициализировать объект).

Если конструктора в классе нет, то интерпретатор PHP пытается вызвать конструктор родительского класса и так далее, пока не будет достигнута вершина иерархии или найден какой-нибудь конструктор. PHP вызывает только один конструктор: если вам нужно, чтобы отработал конструктор не только производного класса, но и родительского, то вызов родительского конструктора вы должны прописать явно:

```
class BaseClass {
  function __construct() { echo "Конструктор класса BaseClass<br>";
}
}

class SubClass extends BaseClass {
  function __construct() {
    parent::__construct(); // вызов родительского конструктора
    echo "Конструктор класса SubClass<br>";
  }
}

$obj1 = new BaseClass();
$obj2 = new SubClass();
```

Если интерпретатор PHP, пройдя по иерархии классов, не обнаружит ни одного объявленного метода `__construct()`, то будет вызван конструктор предыдущей системы, то есть метод, имя которого совпадает с именем класса.

Деструктором называется специальный метод, вызываемый автоматически при уничтожении экземпляра класса. Назначение деструкторов – корректное освобождение ресурсов, занятых объектом: закрытие файлов, соединений, сокетов и т.п.

В PHP понятие деструктора появилось только в версии 5. Деструктор должен называться `__destruct()` (два символа подчеркивания в начале слова) и не принимать параметров.

**Листинг 30.4. Пример конструктора и деструктора**

```
<?php
class MyClass {
  public $name;
  function __construct()
  {
    echo "Конструктор <br>";
    $this->name = "MyClass";
  }
  function __destruct() {
```



http://all-ebooks.com

```
    echo "Уничтожается ".$this->name."<br>";
  }
}
$obj = new MyClass();
?>
```

В приведенном примере объект `$obj` будет уничтожен – а деструктор, соответственно, вызван – при завершении работы сценария. Для удаления объекта вручную служит та же самая функция `unset()`, что и для переменных необъектного типа.

Как и в случае с конструкторами, автоматически вызывается только один деструктор: первый, найденный в иерархии классов. Если вам нужно, чтобы деструктор родительского класса был выполнен тоже, напишите его вызов явно:

```
parent::__destruct();
```

Очевидно, что вызов родительского конструктора, если уж он нужен, должен быть первым вызовом в теле конструктора производного класса, а вызов родительского деструктора – последним в теле деструктора, иначе свойства, с которыми работает метод, могут оказаться еще не созданными или уже уничтоженными.

## 30.5. ДОСТУП К ЧЛЕНАМ КЛАССА

Сначала рассмотрим, как наследование классов влияет на доступ к членам класса, указанный с помощью модификаторов доступа (листинг 30.5).

**Листинг 30.5. Иллюстрация видимости членов класса**

```
<?php
class BaseClass
{
  public $public = 'Public';
  protected $protected = 'Protected';
  private $private = 'Private';

  function SayHello()
  {
    echo "BaseClass::SayHello ".$this->public;
    echo "BaseClass::SayHello ".$this->protected;
    echo "BaseClass::SayHello ".$this->private;
  }
}
```

```
class SubClass extends BaseClass
{
  // внутри производного класса доступны открытые и защищенные
  // члены базового класса. Их можно переопределять:
  protected $protected = 'SubProtected';

  function SayHello()
  {
    echo "SubClass::SayHello ".$this->public;
    echo "SubClass::SayHello ".$this->protected;
    echo "SubClass::SayHello ".$this->private;
  }
}

$obj1 = new BaseClass;
$obj1->SayHello (); // выведет все 3 переменных класса BaseClass
echo $obj1->public; // выведет 'Public'
//echo $obj1->protected; // раскомментируйте, и получите ошибку:
                         // защищенные
//echo $obj1->private; // и закрытые члены недоступны извне класса

$obj2 = new SubClass;
$obj2-> SayHello (); // выведет переменную $public класса BaseClass и
                     // переопределенную в SubClass переменную
$protected
echo $obj2->public; // выведет переменную $public класса BaseClass
echo $obj2->private; // не выведет ничего
//echo $obj2->protected; // раскомментируйте, и получите ошибку
?>
```

Приведенный пример иллюстрирует принцип инкапсуляции, или закрытия данных, впервые реализованный в PHP 5.

Еще одна полезная возможность, появившаяся в 5 версии, – это итератор свойств, позволяющий получить список всех открытых (public) свойств объекта: их имена и значения. Сценарий 30.6 демонстрирует применения итератора.

*Файл был скачан с http://all-ebooks.com*

**Листинг 30.6. Перебор списка свойств объекта**

```
<?php

class User {
  public $login = 'guest';
  public $realname = 'undefined';
  public $email = 'undefined';
```



http://all-ebooks.com

```
  protected $protected = 'protected';
  private   $password  = '12345';
}

$obj = new User;
$obj->realname = 'V.Pupkin';

foreach($obj as $key => $value) {
  echo "$key => $value<br>";
}
?>
```

Этот сценарий выведет:

```
login => guest
realname => V.Pupkin
email => undefined
```

Как видите, в итерацию включены только public свойства класса `User`.

## 30.6. СТАТИЧЕСКИЕ ЧЛЕНЫ КЛАССА

Статическими называются такие свойства и методы, которые принадлежат классу, а не его экземплярам (объектам). Доступ к статическим членам осуществляется через класс независимо от того, создан ли хоть один экземпляр этого класса:

```
MyClass::staticMethod();        // вызов статического метода
echo MyClass::$staticVariable; // обращение к статическому
                                // свойству
```

Обратите внимание, что при обращении к статическому свойству знак $ стоит перед именем переменной, а не именем объекта класса (что понятно, поскольку объекта класса `MyClass` не существует).

Для объявления свойства или метода статическим служит ключевое слово `static`, стоящее после модификатора доступа. При наличии ключевого слова `static` модификатор доступа можно не указывать: по умолчанию принимается значение `public`.

Поскольку статический метод вызывается в контексте класса, а не объекта, он не имеет доступа к переменным объекта. Статические методы могут оперировать только локальными переменными, принятыми параметрами и статическими свойствами. Для обращения к статическим переменным собс-

твенного класса нельзя использовать переменную $this, потому что она ссылается на текущий объект. Вместо нее служит ключевое слово self:

```
class MyClass {
  private static $staticVariable = "static";
  public static function staticMethod()
  {
    echo self::$staticVariable;
  }
}
```

Статическими могут быть объявлены не только переменные, но и константы класса. Как и к переменным, доступ к константам осуществляется через класс, а не через объект:

```
class MyClass {
  const SUCCESS = "Success";
  const FAILURE = "Failure";
}
echo MyClass::SUCCESS;
```

Сценарий 30.7 показывает, как можно применить статические члены класса для учета количества созданных экземпляров этого класса.

**Листинг 30.7. Счетчик экземпляров класса**

```
<?php

class MyCountableClass {
  private static $counter = 0;
  public static function getCount()
  {
    return self::$counter;
  }

  function __construct()
  {
    ++self::$counter;
  }
  function __destruct()
  {
    --self::$counter;
  }
}

$obj1 = new MyCountableClass;
$obj2 = new MyCountableClass;
//$obj = $obj1; // см. следующий параграф

echo "instances: ".MyCountableClass::getCount();
?>
```



http://all-ebooks.com

## 30.7. КОПИРОВАНИЕ И СРАВНЕНИЕ ОБЪЕКТОВ

### 30.7.1. Клонирование объектов

Запустите сценарий 30.7: как вы и ожидали, он выведет "instances: 2". Теперь раскомментируйте закомментированное присваивание

```
$obj = $obj1;
```

Если раньше вы работали с PHP 4, то, наверное, ожидали, что экземпляров класса окажется 3, однако сценарий по-прежнему выводит "instances: 2". Причиной этого является новая объектная модель PHP 5, об отличиях которой, влияющих на копирование объектов, мы поговорим в этом параграфе.

В отличие от PHP 4, ключевое слово new теперь возвращает дескриптор вновь созданного объекта, а не сам объект. В результате изменился смысл присваивания

```
$obj = $obj1;
```

В PHP 4 это присваивание скопирует в переменную $obj объект $obj1, и получатся два независимых объекта. В PHP 5 значением переменной $obj окажется дескриптор объекта $obj1, в результате чего обе переменные будут ссылаться на один и тот же объект.

Это же отличие проявляется при передаче объекта в качестве аргумента функции. В PHP 4 объект передавался по значению, то есть внутри функции существовала независимая копия переданного объекта, изменения которой никак не отражались в вызывающей программе. В PHP 5 же объекты передаются по ссылке, то есть функция имеет дело с тем же объектом, который известен в вызывающей программе.

Чтобы продемонстрировать это, измените пример 30.6 следующим образом. Добавьте в класс MyCountableClass свойство

```
  public $name = 'guest';
```

Определите функцию

```
function SetName($obj, $newname) {

  $obj->name = $newname;
     echo "inside SetName instances: ".MyCountableClass::
getCount()."<br>";
}
```

Вызовите эту функцию и выведите значение свойства $name, полученное объектом после ее завершения:

```
SetName($obj1, 'new name');
echo $obj1->name;
```

Вы убедитесь как в том, что вызов функции не меняет количества экземпляров класса, так и в том, что функция изменяет значения свойств переданного объекта.

Получается, что в PHP 5 для копирования объектов обычное присваивание непригодно и нужен какой-то другой механизм. Такой механизм предусмотрен. Он называется клонированием.

Для создания точной копии объекта служит ключевое слово `clone`:

```
$obj = clone $obj1;
```

В результате клонирования создается новый экземпляр класса, все переменные которого получают те же значения, что у исходного объекта на момент клонирования.

Возможности механизма клонирования шире, чем у обычного копирования: он предоставляет разработчику средство управлять копированием объектов. Это средство – метод `__clone()` (два символа подчеркивания в начале слова), который вы можете объявить в своем классе. Метод с именем `__clone()` автоматически вызывается в ходе клонирования объекта после того, как скопированы значения свойств исходного объекта. Если в классе клонируемого объекта нет метода `__clone()`, то копированием свойств процесс клонирования и ограничится.

Перепишем пример 30.7 так, чтобы класс `MyCountableClass` подсчитывал количество своих экземпляров с учетом клонирования (листинг 30.8).

**Листинг 30.8. Пример клонирования объектов**

```
<?php

class MyCountableClass {
  private static $counter = 0;
  public static function getCount()
  {
    return self::$counter;
  }

  function __construct()
  {
    ++self::$counter;
  }
  function __clone()
  {
    ++self::$counter;
  }
```



http://all-ebooks.com

```
  function __destruct()
  {
    --self::$counter;
  }
}

$obj1 = new MyCountableClass;
$obj2 = new MyCountableClass;

$obj3 = $obj1;
echo "После присваивания: ".MyCountableClass::getCount()."<br>";
$obj4 = clone $obj1;
echo "После клонирования: ".MyCountableClass::getCount()."<br>";
?>
```

**Группа подготовки издания:**

Зав. редакцией компьютерной литературы: *М.В. Финков*

Редактор: *О.И. Березкина*

Корректоры: *Е.Е. Кириллов, Н.Б. Сиразитдинова*

---

ООО «Наука и Техника»

Лицензия №000350 от 23 декабря 1999 года.

198097, г. Санкт-Петербург, ул. Маршала Говорова, д. 29.

Подписано в печать 31.05.2007. Формат 70x100 1/16.

Бумага газетная. Печать офсетная. Объем 40 п. л.

Тираж 5000. Заказ 1048

Отпечатано с готовых диапозитивов в ОАО «Техническая книга»
190005, Санкт-Петербург, Измайловский пр., 29.

**http://all-ebooks.com**