Обработка на заявки с помощта на PHP – лабораторно упражнение

Какво включва това упражнение?

- Архитектура клиент-сървър: основни понятия
- Използване на HTML форма за изпращане на данни към сървъра. Методите Post и Get – особености.
- Механизъм на получаване на данните от HTML форми и обработката им с помощта на PHP.
- Много примери.

1. Архитектура клиент-сървър

В лекция 3, подчертахме, че PHP е скриптов език, като PHP кода се изпълнява на страната на сървъра. Когато става въпрос за сървър, не може да не стане въпрос и за клиент, защото двете понятия са неразривно свързани. Обединява ги компютърната архитектура клиент-сървър.

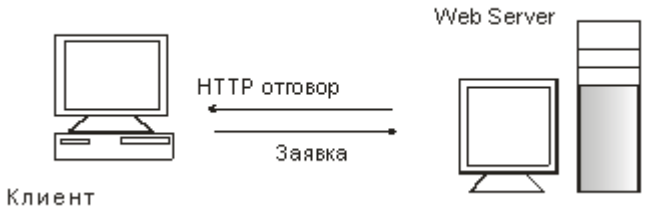
Архитектура клиент-сървър: Същността на тази компютърна архитектура е в разделянето на функциите на системата между двете подсистеми:

- Клиент, който изпраща запитване (заявка) за изпълнение на някакво действие към сървър

и

- Сървър, който изпълнява тази заявка.

Взаимодействието между клиента и сървъра става посредством множество специални протоколи за комуникация, на различни нива, като за нас интерес представлява само протокола на потребителско ниво - протокол HTTP (Hyper Text Transfer Protokol). Неговата асиметричност обуславя изказаните по-горе точно определените роли на клиента и на сървъра в архитектурата клиент-сървър.



Какво ще разбираме под «сървър» в архитектурата клиент-сървър?

Под сървър, в архитектурата клиент-сървър ние ще разбираме web-сървър (сървър WWW или http-сървър). На практика web-сървърът е програма, чиято основна роля е да получава и изпълнява заявки от множество клиенти. За целта, web-сървърът непрекъснато проверява («слуша») определен порт (80-ти по подразбиране) за наличие на заявка. В случай че такава се появи, web-сървърът веднага я изпълнява или я добавя в опашката на чакащите за изпълнение задачи (в случай че е зает да изпълнява друга задача).

Известни web-сървъри: сървър Apache на група Apache, Internet Information Server (IIS) на компания Microsoft, SunOne на фирма Sun Microsystems, WebLogic на фирма BEA Systems, IAS (Inprise Application Server) на фирма Borland, WebSphere на фирма IBM, OAS (Oracle Application Server).

Какво ние ще разбираме под «клиент» в архитектурата клиент-сървър?

От своя страна http-клиентите са Интернет браузъри (например, Internet Explorer, Opera или Mozilla), тоест приложения, чрез които потребителите осъществяват достъп до ресурсите, които се менажират от web-сървъра (http-сървъра). Ролята на http клиента (например Internet Explorer или друг браузър) е да формира и изпрати клиентската заявка (в соответствие с протокола HTTP) до web-сървъра, и впоследствие да получи, анализира и представи по подходящ начин на потребителя отговора, върнат от сървъра.

В протокола HTTP не се указва кой може да инициира или затваря връзката между клиента и сървъра.

Но на практика, връзката се инициира като правило от клиента, който заявява вид обслужване, тоест прави HTTP заявка, получава съответните резултати (HTTP отговор) и съобщава за завършване на работата.
И пак, на практика, след изпращане на данните, сървърът е този, който инициира затваряне на връзката.
От услугите на един сървър често се ползват няколко клиента одновременно. Ето защо всеки сървър е длъжен да има достатъчно голяма производителност и да обезпечава безопасност на данните.

Методи, предоставени от протокол HTTP за предаване на заявки към сървъра и формат на тези заявки
За реализиране на описания принцип заявка/отговор, протоколът HTTP предоставя набор от методи (9 на брой, между които GET и POST), които указват целите на клиентската заявки към сървъра. Ако в адресната лента на браузъра потребителят въведе:

http://www.uni-varna.bg/students/index.php,
на практика той (http-клиентът) инициира взаимодействието със сървъра, като му изпраща запитване (заявка), в следния **пълен формат**:

HTTP метод: GET, POST и др.
URL адрес: www.uni-varna.bg/students/index.php
HTTP версия: например HTTP 1.1
Съобщение, състоящо се от:
- хедъри (заглавия) - съдържат информация за типа на предаваните данни, информация за клиента, изпращащ запитването (User-Agent: тип на браузъра, версия на ОС и др.);
- тяло (не задължителен елемент, например GET – няма тяло, но POST - има. Съдържа информация, която се предава от клиента към сървъра.

2. Използване на HTML форма за изпращане на данни към сървъра. Методите Post и Get – особености.

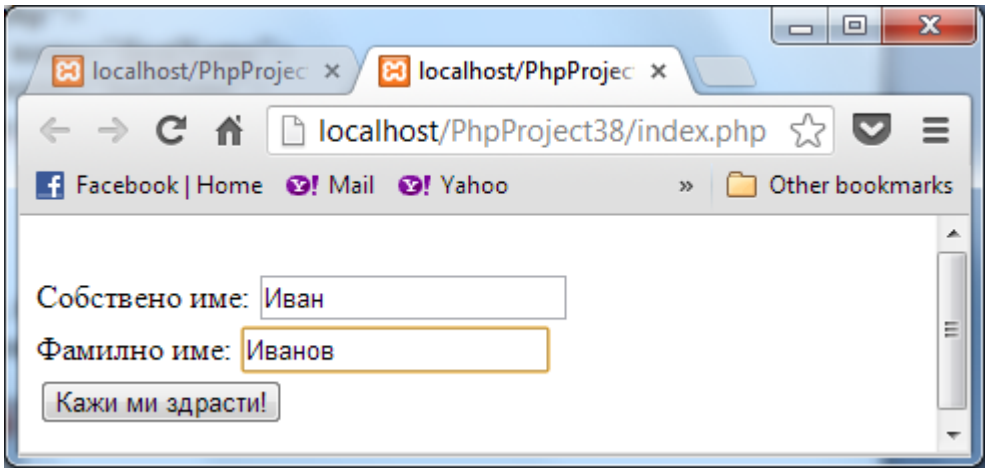
Задача 1: Нека да създадем html формуляр за въвеждане на име и фамилия, например Иван Иванов.
Нека при натискане на бутона с надпис «Кажи ми здрасти» се извежда отговор Здравей, Иван Иванов.
Независимо, че са HTML формулярите са вече разглеждани, ще припомним, че те се указват с таг <form> за начало и таг </form> за край, а между тези два тага се разполагат всички input компоненти на формуляра, указани най-общо по следния начин:
<input type="тип" name="име" />.
- На мястото на "тип" стои конкретния тип елемент (text, например), а "име" е уникално име на елемента (name, family, mail), което позволява неговото управление при обработката на формуляра.
- С помощта на атрибутите action и тага method, на формуляра се задават име на програмата, която ще обработва данните от формата и метода за достъп (Get, Post), например **<form action="example2.php" method=POST>.**
- Изпращането на данните от формата става след натискане на бутон - input елемент от тип submit, а отказът става чрез натискане на бутон reset, които се описват по следния начин:

<input type=submit value="Submit">
<input type=reset value="Reset">

Примерна реализация на задача 1 с метод Get:

Index.php
<html>
<head>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">

```
</head>
<body>
<form method="get" action="example2.php">
<br>Собствено име: <input type="text" name="firstName">
<br>Фамилно име: <input type="text" name="lastName">
<br><input type="submit" value="Кажи ми здрасти!">
</form>
</body>
</html>
```

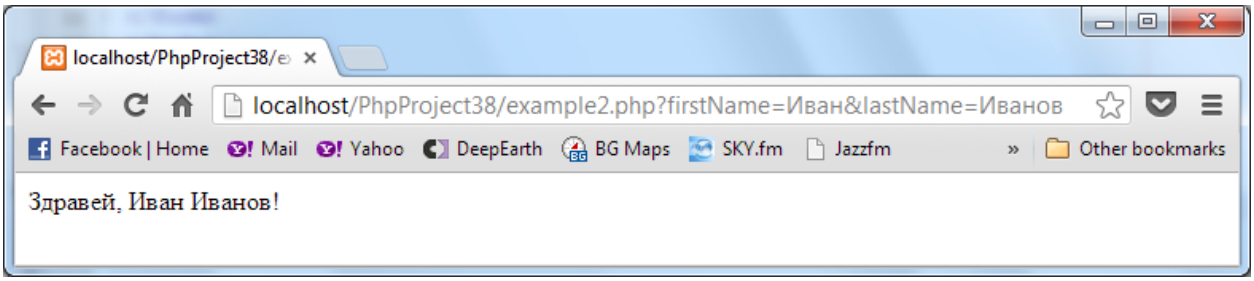


Example2.php

```
<?php
print '<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">';
print "Здравей, " . $_GET['firstName'] . ' ' . $_GET['lastName'].'!';
?>
```

Забележка: Първият оператор print в example2.php съобщава на браузъра, че следващото го съдържание е на кирилица.

Така, ако сме въвели за име Иван, а за фамилия: Иванов, то резултатът е:



Коментар:
При натискане на бутона „Кажи ми здрасти” в HTML формуляра, браузърът проверява по кой метод ще се изпращат данните до сървъра. Ако данните ще се изпращат по метода Get, браузърът автоматично формира низа от параметри firstName=Иван&lastName=Иванов, добавя го към адреса на програмата,която ще обработва заявката (в случая, http://localhost/PhpProject38/example2.php), след знак ? и изпраща get заявката към web сървъра:
http://localhost/PhpProject38/example2.php?firstName=Иван&lastName=Иванов

Как PHP обработва заявките на клиента изпратени по метод get (чрез html форма)?
В example2.php имаме:
print "Здравей, " . $_GET['firstName'] . ' ' . $_GET['lastName'].'!';

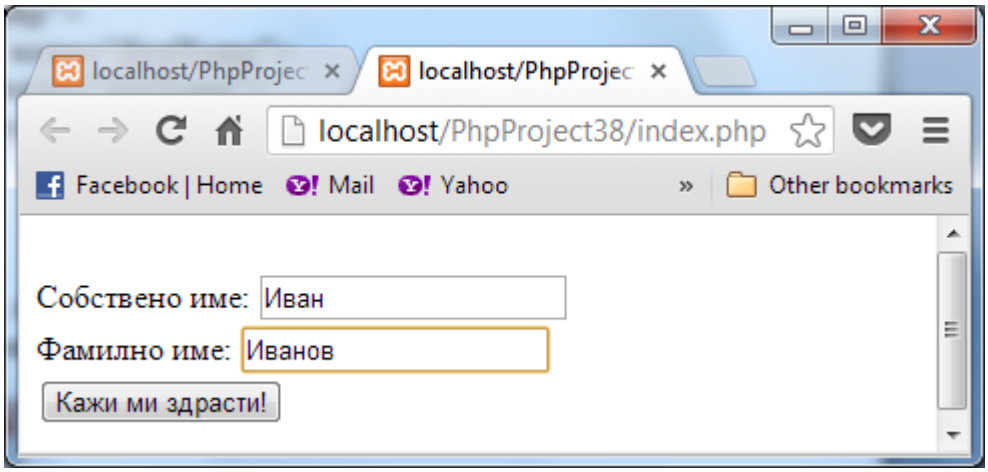
Вътре в PHP-скрипта, обработващ заявката на клиента, данните изпратени по метод get от html формуляр се обработват със суперглобалния масив $_GET (или суперглобалния масив $_REQUEST). Или, когато във формуляра имаме двойка елементи firstName и lastName, то в обработващия скрипт example2.php използваме масива $_GET по следния начин:
$_GET['firstName'] и $_GET['lastName']
Аналогично:
$_REQUEST['firstName'] и $_REQUEST['lastName']

Нека да модифицираме примера, заменяйки GET с POST:

Примерна реализация на задача 1 с метод POST:

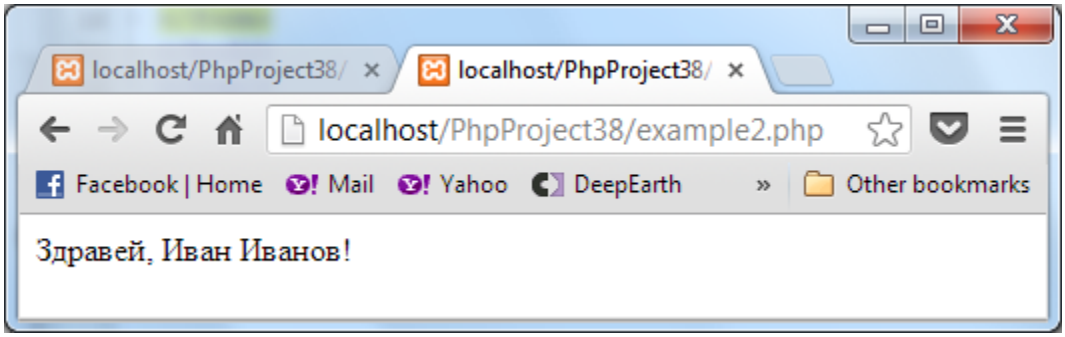
Index.php

```
<html>
<head>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
</head>
<body>
<form method="post" action="example2.php">
<br>Собствено име: <input type="text" name="firstName">
<br>Фамилно име: <input type="text" name="lastName">
<br><input type="submit" value="Кажи ми здрасти!">
</form>
</body>
</html>
```



Example2.php

```
<?php
print '<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">';
print "Здравей, " . $_POST['firstName'] . ' ' . $_POST['lastName'].'!';
?>
```



Вижда се, че: Ако заменим в index.php и файла example2.php GET с POST, то след натискане на бутона „Кажи ми здрасти” в index.php, в лентата на браузъра ще видим само:
http://localhost/PhpProject38/example2.php
Тоест, въведените данни не се визуализират в адресната лента.

Как PHP обработва заявките на клиента изпратени по метод post (чрез html форма)?
В example2.php имаме:
print "Здравей, " . $_POST['firstName'] . ' ' . $_POST['lastName'].'!';
Вътре в PHP-скрипта, обработващ заявката на клиента, данните изпратени по метод post от html формуляр се обработват със суперглобалния масив $_POST (или суперглобалния масив $_REQUEST). Или, когато във формуляра имаме двойка елементи firstName и lastName, то в обработващия скрипт example2.php използваме масива $_POST по следния начин:
$_POST['firstName'] и $_POST['lastName']
Аналогично:
$_REQUEST['firstName'] и $_REQUEST['lastName']

Example2.php (с $_REQUEST)
```
<?php
print '<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">';
print "Здравей, " . $_REQUEST['firstName'] . ' ' . $_REQUEST['lastName'].'!';
?>
```

Изводи:

- Съществен недостатък при предаване с GET е, че всеки може да види (и да преправи) стойностите на параметрите в URL полето на браузера, преди да ги изпрати на сървъра. Ето защо **не се препоръчва използване на този метод за предаване на информация, която трябва да е конфеденциална** (например парола) или такава, от която зависи безопасността на работа на програмата или сървъра.
- С Get заявки клиентът (браузърът) **може да изпрати ограничено (зависи от самия браузър, например за Internet Explorer – до 2083 байта) количество данни**, които се прикрепят към адреса на ресурса (вижда се от примера), тоест не се препоръчва използване на този метод за предаване на по-голяма по-обем информация.
- Метод **POST** е препоръчван за предаване на сървъра на **отговорна|или голяма** по обем информация (пароли, пощенски съобщения, данни за добавяне към БД и др.), тъй като данните от формата се предават на сървъра чрез тялото на съобщението (ползвателя не вижда предаваните към сървъра данни в URL полето на браузъра си).
- Ако данните се предават от HTML форма чрез метод get, то в обработващия скрипт се използват суперглобални масиви $_GET или $_REQUEST за достъп до данните.
- Ако данните се предават от HTML форма чрез метод post, то в обработващия скрипт се използват суперглобални масиви $_POST или $_REQUEST за достъп до данните.

Задача 2: Да създадем html формуляр за въвеждане на дължина и ширина на правоъгълник. Нека при натискане на бутона с надпис «GO» се извежда площта на правоъгълника.

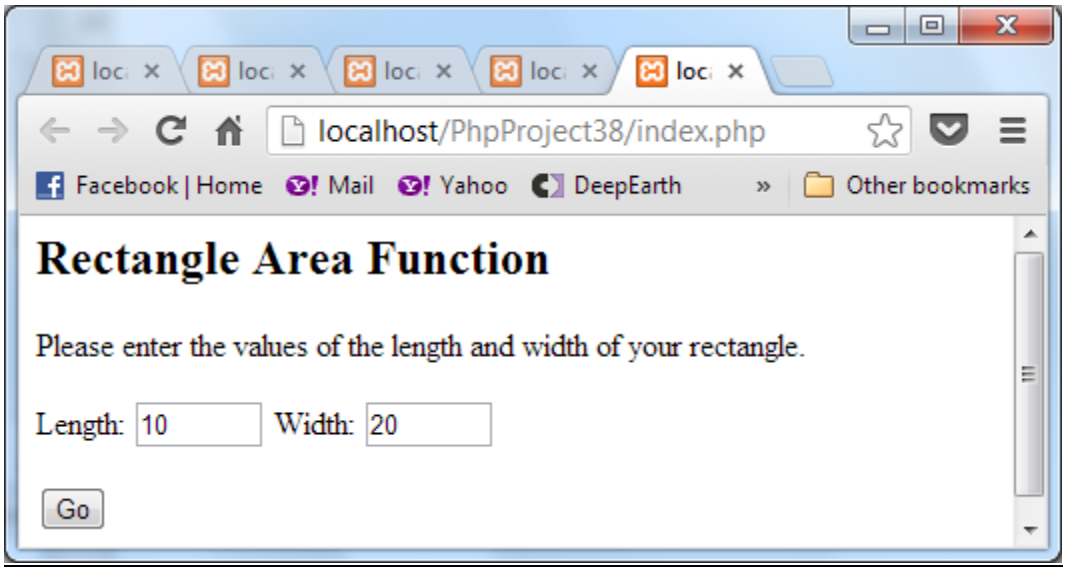
Примерна реализация:

Index.php
```
<html>
<head>
<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">
</head>
<body>
<h2>Rectangle Area Function</h2>
```

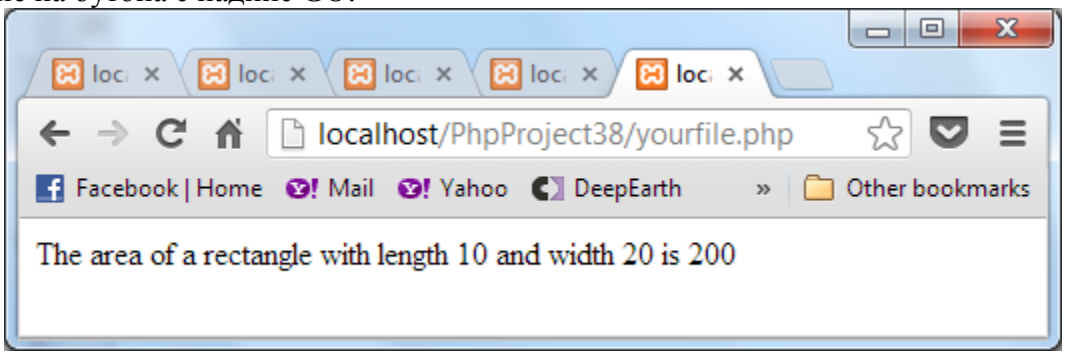
```
<form action="yourfile.php" method="post">
<p>Please enter the values of the length and width of your rectangle.</p>
<p>Length: <input type="text" name="length" size="5" />
Width: <input type="text" name="width" size="5" /></p>
<input type="submit" name="submit" value="Go" />
</form>
</body>
</html>
```

yourfile.php

```
<?php
if(isset ($_POST['submit'])){
//Retrieve user values.
$l = $_POST['length'];
$w = $_POST['width'];
echo "The area of a rectangle with length $l and width $w is " . $l*$w;
}
?>
```



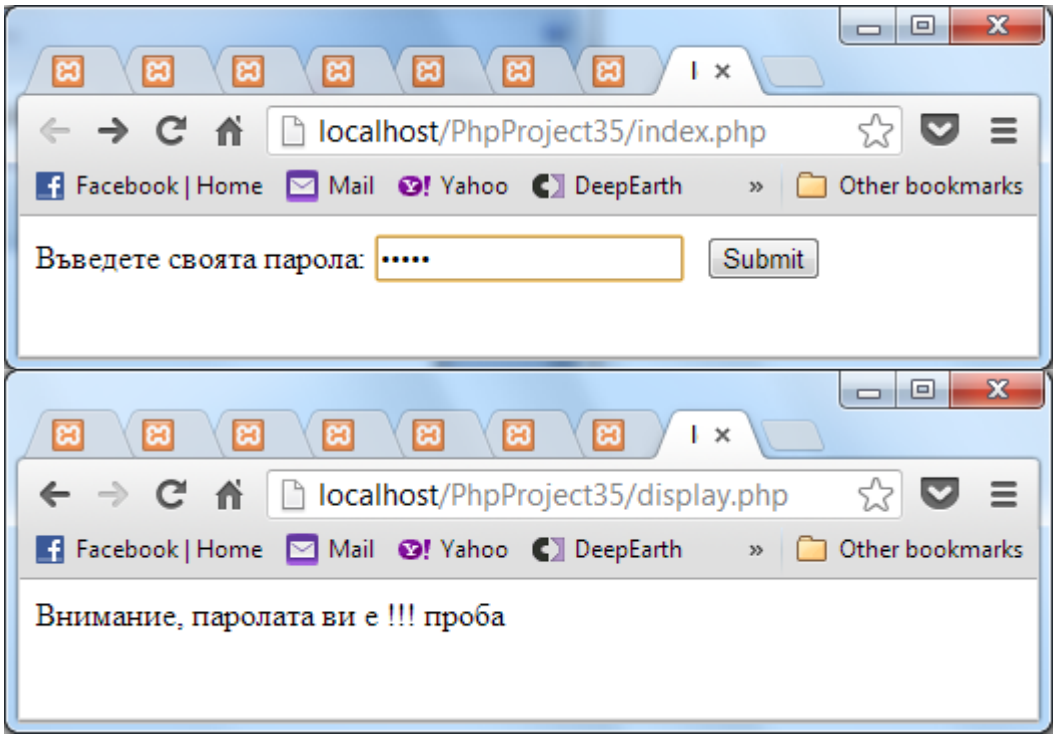
След натискане на бутона с надпис GO:



Задача 3: Самостоятелно: преработете задача 2, така че да пресмятате лице и обем на паралелепипед. Тествайте с методи get и post.

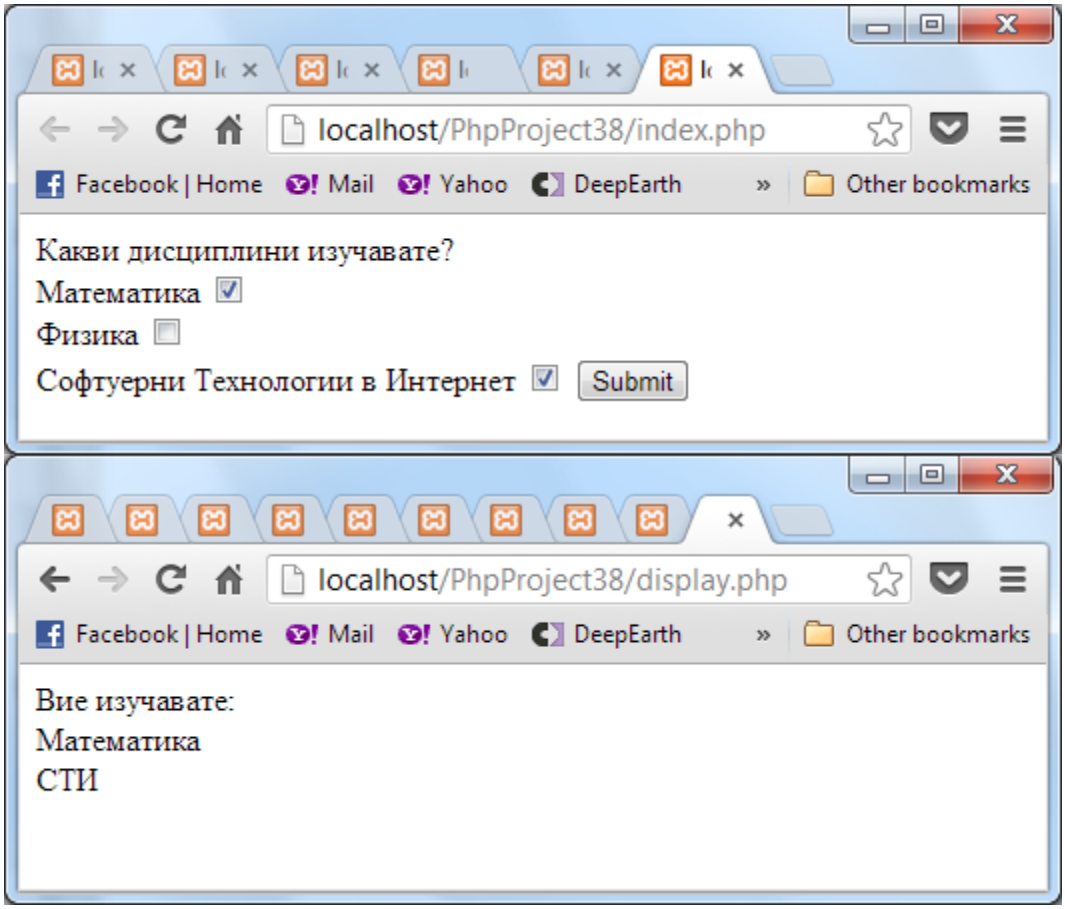
Задача 4: Самостоятелно: създайте форма за въвеждане на парола. Използвайте подходящия метод за изпращане на данните към сървъра.

Примерен интерфейс:



Забележка: Типът на input елемент в този случай е "password".

Задача 4: Създайте форма за множествен избор, с елементи от тип checkbox. Тествайте с методи get и post.
Примерен интерфейс:



Реализация:

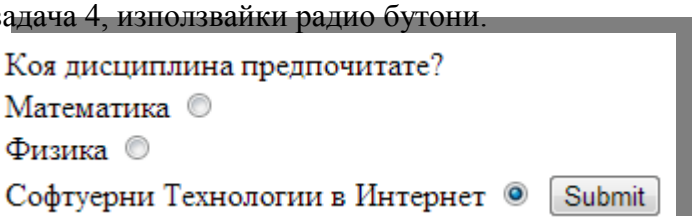
index.php

```
<!DOCTYPE html>
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title></title>
   </head>
   <body>
<form method="post" action="display.php">
Какви дисциплини изучавате?
<br />
Математика <input type="checkbox" name="Choice[]" value="Математика" checked="checked" />
<br />
Физика  <input type="checkbox" name="Choice[]" value="Физика" />
<br />
Софтуерни Технологии в Интернет <input type="checkbox" name="Choice[]" value="СТИ" />
<input type="submit" name="submit" value=Submit>
</form>
</body>
</html>
```

display.php

```
<?php
print '<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=utf-8">';
echo "Вие изучавате:";
foreach ($_POST['Choice'] as $names)
{ echo "<br>$names";}
?>
```

Задача 5: Модифицирайте задача 4, използвайки радио бутони.



След натискане на бутона: