What is .NET ?

Framework basics

Document Prepared by :
Sehul Soni
Call : +91-9925025622
Email : sehulsoni@gmail.com
Chat : sehulsoni@yahoo.com
Talk : sehulsoni1011 (Skype)

What is .NET ?

Introduction

In this article, I will give an understanding of what is .NET and why it came into existence. We will also see some of the core building blocks of .NET and how it is layered.

Why .NET

The world of computing till date has been confused. We have had various languages struggling to interoperate with each other, developers undergoing huge learning curves to shift from one language to another or from one application type to another, non-standard ways of modeling applications and designing solutions and huge syntactic differences between languages. The list goes on....

Past years have seen some comfort in the form of enterprise "glue" applications and standards like COM, which put-forth a binary standard of interoperability between application components. But in reality, this was not always true (VB COM found it very difficult to take on VC++ COM). Also, as applications increased in their reach, it was found that rather than re-inventing the wheel for a solution, it was better to take the "service" of another applications specialized for a piece of work.

Thus from a paradigm where applications replicated code to provide common services, we have moved to a paradigm where applications are built as "collaborative units" of components working together. This simple shift has led to the collapse of the current set of architectures and demanded a new programming model:

- A model where applications can be built as reusable components and are sharable over the internet.
- A model that encourages applications to be shared as a "service" (read web services).
- A model that enables true "interoperability" wherein the language used is only a matter of choice, thus enabling organizations to take advantage of existing skill sets.

Enter .NET. The .NET Framework is a new computing platform developed by Microsoft that simplifies application development in the highly distributed environment of the internet. .NET is much more than just a platform for developing for the internet, but it is intended for this purpose predominantly, because here, others methods have failed in the past.

Overview of .NET

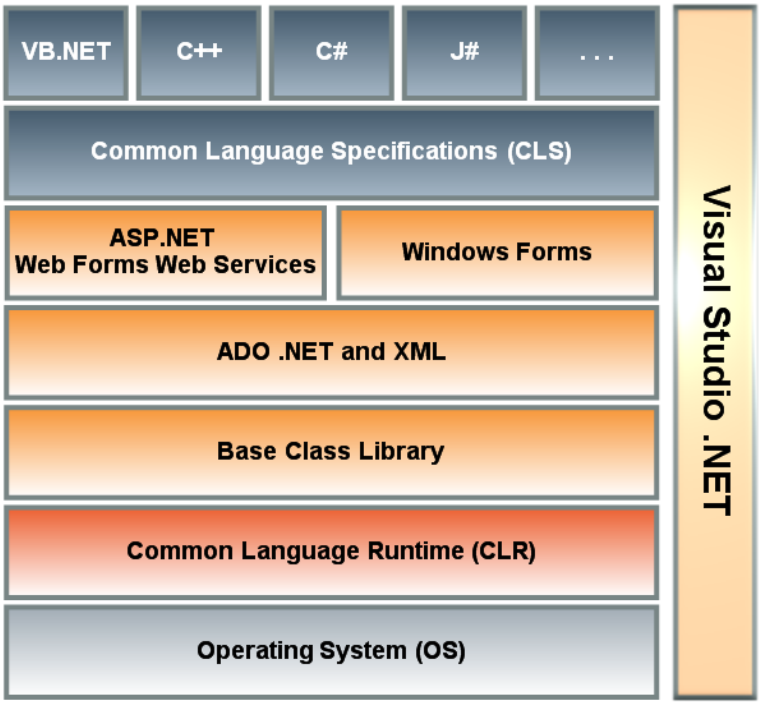
The .NET Framework has been developed to cater to the following objectives and requirements:

- To provide a consistent object-oriented environment to develop applications.
- To provide a code execution environment that simplifies deployment and versioning.

- To provide a code execution environment that guarantees the safety of the code that is executing. This includes both code developed internally by an organization or for code developed by 3rd party vendors.
- To provide a code execution environment that eliminates the issues faced by scripted environments with respect to performance.
- To provide a common programming model where the choice of a programming language becomes a matter of choice.

The .NET Framework is made up of two major components: the common language runtime (CLR) and the framework class library (FCL). The CLR is the foundation of the .NET Framework and provides various services that applications can use. The CLR also forms the environment that other applications run on. The FCL is a collection of over 7000+ types that cater to all the services, and data structures that applications will ever need.

The following diagram shows the .NET Framework, its hierarchy and the associated toolset. The diagram is so famous that you can spend some time memorizing its layout!!



At the base of the diagram, you see the operating system which can be (theoretically) any platform. The Common Language Runtime (CLR) is the substrate that abstracts the underlying operating system from your code. The minute it does this, it means that your code has to run using the services provided by the CLR and we get a new name called managed code. The CLR provides its services to applications by providing a standard set of library classes that abstract all the tasks that you will ever need. These classes are called as the Base Class Libraries. On

top of this, other development platforms and applications are built (like ASP.NET, ADO.NET and so on). Language compilers that need to generate code for the CLR must adhere to a common set of specifications as laid down by the Common Language Specification (CLS). Above this, you have all the popular .NET languages.

Common Language Runtime

The CLR is the platform on which applications are hosted and executed. The CLR also provides a set of services that applications can use to access various resources (like arrays, collections, operating system folders etc). Since this runtime "manages" the execution of your code, code that works on the CLR is called as managed code. Any other code, you guessed it, is called unmanaged code.

When compilers release code to run on the CLR, they do not release machine language code. Rather, an intermediate language code is used called Microsoft Intermediate Language (MSIL). MSIL is like an object-oriented version of assembly language and is platform independent. It has a rich set of instructions that enable efficient representation of the code. When a code starts to execute, a process knowing as Just in Time Compilation (JIT) converts the MSIL code into the native processor instructions of the platform, which is then executed. This is shown in the following diagrams:

Diagram 1:

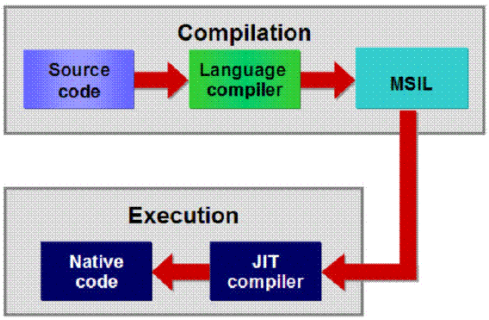
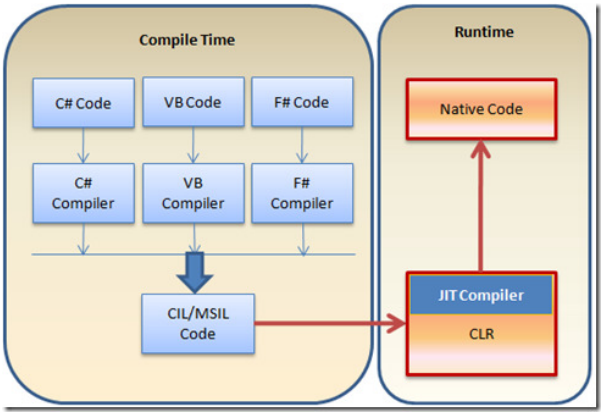


Diagram 2:



Note that this conversion happens only once. Subsequent calls to the code will execute the native version only. Once the application dies down and is started again, this process is repeated.

The following are some of the benefits of the CLR:

- Performance improvements.
- The ability to easily use components developed in other languages.
- Extensible types provided by a class library.
- New language features such as inheritance, interfaces, and overloading for object-oriented programming; support for explicit free threading that allows creation of multithreaded, scalable applications; support for structured exception handling and custom attributes.

Common Language Specification

Language interoperability is the ability of code to interact with code that is written using a different programming language. Language interoperability can help maximize code reuse and, therefore, improve the efficiency of the development process. Because developers use a wide variety of tools and technologies, each of which might support different features and types, it has historically been difficult to ensure language interoperability. However, language compilers and tools that target the common language runtime benefit from the runtime's built-in support for language interoperability. To ensure that you can develop managed code that can be fully used by developers using any programming language, a set of language features and rules for using them called the Common Language Specification (CLS) has been defined. Components that follow these rules and expose only CLS features are considered CLS-compliant.

Common Type System

The common type system defines how types are declared, used, and managed in the runtime, and is also an important part of the runtime's support for cross-language integration. The common type system performs the following functions:

- Establishes a framework that enables cross-language integration, type safety, and high performance code execution.
- Provides an object-oriented model that supports the complete implementation of many programming languages.
- Defines rules that languages must follow, which helps ensure that objects written in different languages can interact with each other. For example, if you have created a class in VB.NET, you can inherit from it in a C# program.

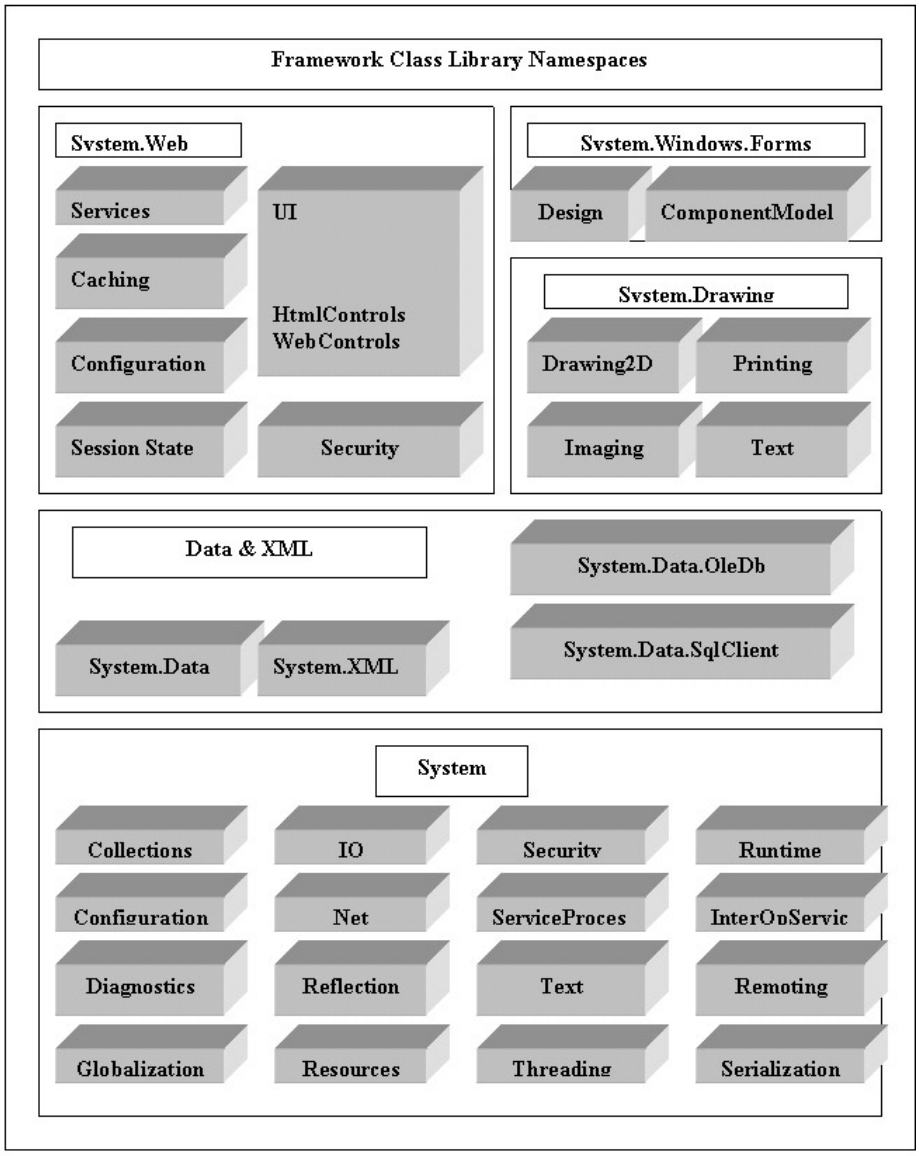
Framework Class Library

Windows programmers coding in C, tend to rely on the Windows API and functions in third-party DLLs to get their job done. C++ programmers often use class libraries of their own creation or standard class libraries such as MFC. Visual Basic programmers use the Visual Basic API, which is an abstraction of the underlying operating system API.

In the .NET Framework, all these anachronistic APIs are done away with. Rather a new set of functions branded as the framework class library are introduced which contain more than 7000 types.

To make learning and using the FCL more manageable, Microsoft has divided the FCL into hierarchical namespaces. The FCL has about 100 namespaces in all. Each namespace holds classes and other types that share a common purpose. For example, much of the window manager portion of the Windows API is encapsulated in the System.Windows.Forms namespace. In this namespace classes that represent windows, dialog boxes, menus, and other elements commonly used in GUI applications are present. A separate namespace called System.Collections holds classes representing hash tables, resizable arrays, and other data containers. Yet another namespace, System.IO, contains classes for doing file I/O.

The following diagram shows the FCL classes and their associated namespaces.



** Article is taken from www.codeproject.com and images are taken from google.