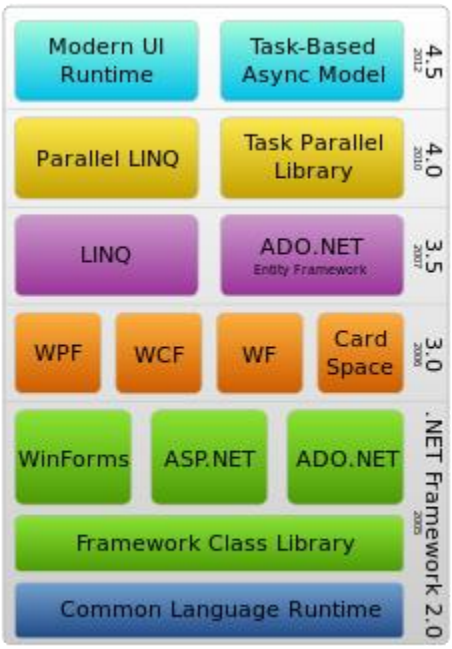
.NET stands for Network Enable Technology.

Introduction

.NET Framework (pronounced dot net) is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large class library known as Framework Class Library (FCL) and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for .NET Framework execute in a software environment (as contrasted to hardware environment), known as Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. FCL and CLR together constitute .NET Framework.

History:

1960's	Structured programming language C, PASCAL, BASIC
1970'S	DBMS packages- Dbase, FoxPro, Symphony, paradox etc.
1980's	GUI Windows –Visual Basic, Visual C++, Visual FoxPro etc.
1990's	Java- Sun Micro Systems



The .NET Framework Stack

Overview of .NET Framework release history

Generation	Version number	CLR version	Release date	Development tool	Distributed with		Supersedes
					Windows	Windows Server	
1.0	1.0.3705.0	1.0	2002-02-13	Visual Studio .NET	N/A	N/A	N/A
1.1	1.1.4322.573	1.1	2003-04-24	Visual Studio .NET 2003	N/A	2003	1.0
2.0	2.0.50727.42	2.0	2005-11-07	Visual Studio 2005	N/A	2003, 2008 SP2, 2008 R2 SP1	N/A
3.0	3.0.4506.30	2.0	2006-11-06	Microsoft Blend	Vista	2008 SP2, 2008 R2 SP1	2.0
3.5	3.5.21022.8	2.0	2007-11-19	Visual Studio 2008	7, 8, 8.1	2008 R2 SP1	2.0, 3.0
4.0	4.0.30319.1	4	2010-04-12	Visual Studio 2010	N/A	N/A	N/A
4.5	4.5.50709.17929	4	2012-08-15	Visual Studio 2012	8	2012	4.0
4.5.1	4.5.50938.18408	4	2013-10-17	Visual Studio 2013	8.1	2012 R2	4.0, 4.5
4.5.2	4.5.51641	4	2014-05-05	Visual Studio 2013	N/A	N/A	4.0, 4.5, 4.5.1

Principal Design Features

Interoperability

Because computer systems commonly require interaction between newer and older applications, .NET Framework provides means to access functionality implemented in newer and older programs that execute outside .NET environment. Access to COM components is provided in **System.Runtime.InteropServices** **and** **System.EnterpriseServices** namespaces of the framework; access to other functionality is achieved using the P/Invoke (**Platform Invocation Services**, commonly referred to as **P/Invoke**, is a feature of Common Language Infrastructure implementations, like Microsoft's Common Language Runtime, that enables managed code to call native code.) feature.

Common Language Runtime engine

Common Language Runtime (CLR) serves as the execution engine of .NET Framework. All .NET programs execute under the supervision of CLR, guaranteeing certain properties and behaviors in the areas of memory management, security, and exception handling.

Framework Class Library

Framework Class Library (FCL) is a library of functionality available to all languages using .NET Framework. FCL provides classes that encapsulate a number of common functions, including file reading and writing, graphic rendering, database interaction, XML document manipulation, and so on. It consists of classes, interfaces of reusable types that integrates CLR.

Simplified deployment

.NET Framework includes design features and tools which help manage the installation of computer software to ensure that it does not interfere with previously installed software, and that it conforms to security requirements.

Security

The design addresses some of the vulnerabilities, such as buffer overflows, which have been exploited by malicious software. Additionally, .NET provides a common security model for all applications.

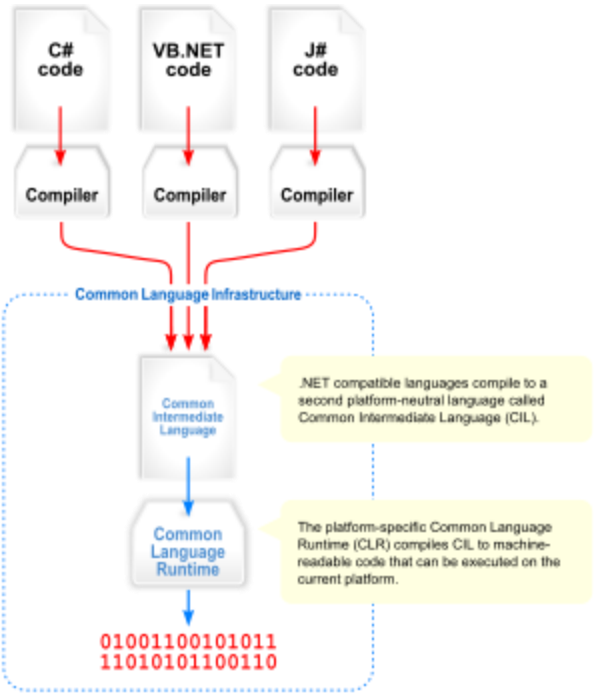
Portability

While Microsoft has never implemented the full framework on any system except Microsoft Windows, it has engineered the framework to be platform-agnostic, and cross-platform implementations are available for other operating systems (see Silverlight and Alternative implementations). Microsoft submitted the specifications for CLI (which includes the core class libraries, CTS, and the Common Intermediate Language C#, and C++/CLI to both ECMA and ISO, making them available as official standards. This makes it possible for third parties to create compatible implementations of the framework and its languages on other platforms.

Architecture

Common Language Infrastructure (CLI)

The purpose of CLI is to provide a language-neutral platform for application development and execution, including functions for exception handling, garbage collection, security, and interoperability. By implementing the core aspects of .NET Framework within the scope of CLI, this functionality will not be tied to a single language but will be available across the many languages supported by the framework. Microsoft's implementation of CLI is CLR.



Common Intermediate Language (CIL) code is housed in CLI assemblies. As mandated by the specification, assemblies are stored in Portable Executable (PE) format, common on Windows platform for all DLL and EXE files. The assembly consists of one or more files, one of which must contain the manifest, which has the metadata for the assembly. The complete name of an assembly (not to be confused with the filename on disk) contains its simple text name, version number, culture, and public keytoken. Assemblies are considered equivalent if they share the same complete name, excluding the revision of the version number. A private key can also be used by the creator of the assembly for strong naming. The public key token identifies which private key an assembly is signed with. Only the creator of the keypair (typically .NET developer signing the assembly) can sign assemblies that have the same strong name as a previous version assembly, since the creator is in possession of the private key. Strong naming is required to add assemblies to Global Assembly Cache.

Security

NET has its own security mechanism with two general features: Code Access Security (CAS), and validation and verification. CAS is based on evidence that is associated with a specific assembly. Typically the evidence is the source of the assembly (whether it is installed on the local machine or has been downloaded from the intranet or Internet). CAS uses evidence to determine the permissions granted to the code. Other code can demand that calling code is granted a specified permission. The demand causes CLR to perform a call stack walk: every assembly of each method in the call stack is checked for the required permission; if any assembly is not granted the permission a security exception is thrown.

Class library

NET Framework includes a set of standard class libraries. The class library is organized in a hierarchy of namespaces. Most of the built-in APIs are part of either `System.*` or `Microsoft.*` namespaces. These class libraries implement a large number of common functions, such as file reading and writing, graphic rendering, database interaction, and XML document manipulation, among others. .NET class libraries are available to all CLI compliant languages. .NET Framework class library is divided into two parts: FCL and Base Class Library (BCL).

FCL includes a small subset of the entire class library and is the core set of classes that serve as the basic API of CLR. Classes in `mscorlib.dll` and some classes in `System.dll` and `System.core.dll` are part of FCL. FCL classes are available in .NET Framework as well as its alternative implementations including .NET Compact Framework, Microsoft Silverlight and Mono.

BCL is a superset of FCL and refers to the entire class library that ship with .NET Framework. It includes an expanded set of libraries, including Windows Forms, ADO.NET, ASP.NET, Language Integrated Query, Windows Presentation Foundation, and Windows Communication Foundation among others. BCL is much larger in scope than standard libraries for languages like C++, and comparable in scope to standard libraries of Java.

Namespaces in the FCL
`System`
`System.Diagnostics`
`System.Globalization`
`System.Resources`
`System.Text`
`System.Runtime.Serialization`
`System.Data`

Memory management

CLR frees the developer from the burden of managing memory (allocating and freeing up when done); it handles memory management itself by detecting when memory can be safely freed. Instantiations of .NET types (objects) are allocated from the managed heap; a pool of memory managed by CLR. As long as there exists a reference to an object, which might be either a direct reference to an object or via a graph of objects, the object is considered to be in use. When there is no reference to an object, and it cannot be reached or used, it becomes garbage, eligible for collection. .NET Framework includes a garbage collector which runs periodically, on a separate thread from the application's thread, that enumerates all the unusable objects and reclaims the memory allocated to them.

.NET Garbage Collector (GC) is a non-deterministic, compacting, and mark-and-sweep garbage collector. GC runs only when a certain amount of memory has been used or there is enough pressure for memory on the system. Since it is not guaranteed when the conditions to reclaim memory are reached, GC runs are non-deterministic. Each .NET application has a set of roots, which are pointers to objects on the managed heap (*managed objects*). These include references to static objects and objects defined as local variables or method parameters currently in scope, as well as objects referred to by CPU registers. When GC runs, it pauses the application, and for each object referred to in the root, it recursively enumerates all the objects reachable from the root objects and marks them as reachable. It uses CLI metadata and reflection to discover the objects encapsulated by an object, and then recursively walk them. It then enumerates all the objects on the heap (which were initially allocated contiguously) using reflection. All objects not marked as reachable are garbage. This is the *mark* phase. Since the memory held by garbage is not of any consequence, it is considered free space. However, this leaves chunks of free space between objects which were initially contiguous. The objects are then *compacted* together to make used memory contiguous again. Any reference to an object invalidated by moving the object is updated by GC to reflect the new location. The application is resumed after the garbage collection is over.

GC used by .NET Framework is also generational. Objects are assigned a generation; newly created objects belong to Generation 0. The objects that survive a garbage collection are tagged as Generation 1, and the Generation 1 objects that survive another collection are Generation 2 objects. .NET Framework uses up to Generation 2 objects. Higher generation objects are garbage collected less frequently than lower generation objects. This helps increase the efficiency of garbage collection, as older objects tend to have a longer lifetime than newer objects. Thus, by eliminating older (and thus more likely to survive a collection) objects from the scope of a collection run, fewer objects need to be checked and compacted.

Top ten reasons to move to .NET

- Standard Integration: XML, SOAP (Simple Object Access Protocol) and more.
- Case of development.
- Web service Support.
- Standard toolset for any .NET languages.
- Support for mobile devices.
- Managed code.
- Platform independent.
- No learn for learning resources.
- Modernized languages.
- Standard base types across languages.

Following list will give you an idea about various types of application that we can develop on .NET.

1. ASP.NET Web applications: These include dynamic and data driven browser based applications.
2. Windows Form based applications: These refer to traditional rich client applications.
3. Console applications: These refer to traditional DOS kind of applications like batch scripts.
4. Component Libraries: This refers to components that typically encapsulate some business logic.
5. Windows Custom Controls: As with traditional ActiveX controls, you can develop your own windows controls.
6. Web Custom Controls: The concept of custom controls can be extended to web applications allowing code reuse and modularization.
7. Web services: They are "web callable" functionality available via industry standards like HTTP, XML and SOAP.

Features of .NET

- **Rich Functionality out of the box**

NET framework provides a rich set of functionality out of the box. It contains hundreds of classes that provide variety of functionality ready to use in your applications. This means that as a developer you need not go into low level details of many operations such as file IO, network communication and so on.

- **Easy development of web applications**

ASP.NET is a technology available on .NET platform for developing dynamic and data driven web applications. ASP.NET provides an event driven programming model with complex user interface. ASP.NET server controls provide advanced user interface elements (like calendar and grids) that save lot of coding from programmer's side.

- **OOPs Support**

The advantages of Object Oriented programming are well known. .NET provides a fully object oriented environment. The philosophy of .NET is – "Object is mother of all."

- **Multi-Language Support**

Generally enterprises have varying skill sets. For example, a company might have people with skills in Visual Basic, C++, and Java etc. It is an experience that whenever a new language or environment is invented existing skills are outdated. This naturally increases cost of training and learning curve. .NET provides something attractive in this area. It supports multiple languages. This means that if you have skills in C++, you need not throw them but just mould them to suit .NET environment. Currently four languages are available right out of the box namely – Visual Basic.NET, C# (pronounced as C-sharp), Jscript.NET and Managed C++ (a dialect of Visual C++). There are many vendors that are working on developing language compilers for other languages (20+ language compilers are already available). The beauty of multi-language support lies in the fact that even though the syntax of each language is different, the basic capabilities of each language remain at par with one another.

- **Multi-Device Support**

Modern life style is increasingly embracing mobile and wireless devices such as PDAs, mobiles and handheld PCs. . .NET provides promising platform for programming such devices.

- **Automatic memory management**

While developing applications developers had to develop an eye on system resources like memory. Memory leaks were major reason in failure of applications. .NET takes this worry away from developer by handling memory on its own. The garbage collector takes care of freeing unused objects at appropriate intervals.

- **Security**

Windows platform was always criticized for poor security mechanisms. Microsoft has taken great efforts to make .NET platform safe and secure for enterprise applications. Features such as type safety, code access security and role based authentication make overall application more robust and secure.

Installing the .NET Framework SDK

Understanding the .NET Platform and its layers

The Common Language Runtime (CLR)

It is considered as the heart of the .NET framework. .NET applications are compiled to a common language known as Microsoft Intermediate Language or "IL". The CLR, then, handles the compiling the IL to machine language, at which point the program is executed.

The CLR environment is also referred to as a managed environment, in which common services, such as garbage collection and security, are automatically provided.

The .NET Class Framework

The next layer up in the framework is called the .NET Class Framework also referred as .NET base class library. The .NET Class Framework consists of several thousand type definitions, where each type exposes some functionality. All in all, the CLR and the .NET Class Framework allow developers to build the following kinds of applications:

- Web Services. Components that can be accessed over the Internet very easily.
- Web Forms. HTML based applications (Web Sites).
- Windows Forms. Rich Windows GUI applications. Windows form applications can take advantage of controls, mouse and keyboard events and can talk directly to the underlying OS.
- Windows Console Applications. Compilers, utilities and tools are typically implemented as console applications.
- Windows Services. It is possible to build service applications controllable via the Windows Service Control Manager (SCM) using the .NET Framework.
- Component Library. .NET Framework allows you to build stand-alone components (types) that may be easily incorporated into any of the above mentioned application types.

ADO.NET: Data and XML

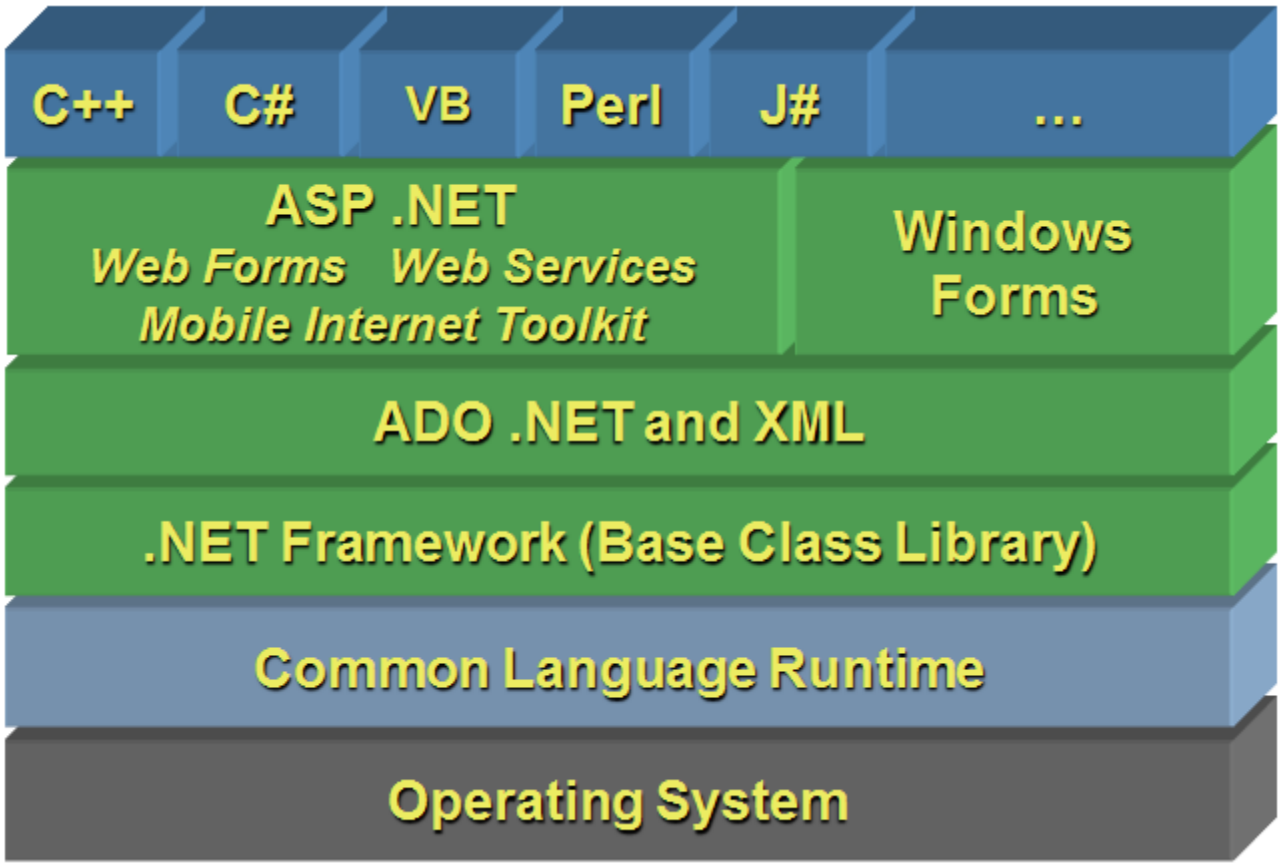
ADO.NET is the next generation of Microsoft ActiveX Data Object (ADO) technology. ADO.NET is heavily dependent on XML for representation of data. It also provides an improved support for the disconnected programming model.

ADO.NET's DataSet object is the core component of the disconnected architecture of ADO.NET. The DataSet can also be populated with data from an XML source, whether it is a file or an XML stream.

User Interface

The next layer consists of the user and programming interface that allows .NET to interact with the outside world. The following are the types of interaction interfaces that are supported by the .NET framework:

- Web Forms
- Windows Forms
- Web Services



Languages

The CLR allows objects created in one language be treated as equal citizens by code written in a completely different language. To make this possible, Microsoft has defined a Common Language Specification (CLS) that details for compiler vendors the minimum set of features that their compilers must support if they are to target the runtime.

The .NET Framework provides a run-time environment called the Common Language Runtime, which manages the execution of code and provides services that make the development process easier. Compilers and tools expose the runtime's functionality and enable you to write code that benefits from this managed execution environment. Code developed with a language compiler that targets the runtime is called **managed code.**

What is meant by Platform Dependent?

Every operating system contains a specific set of instruction set. This instruction set format will be varied from operating system to operating system. Thus an application developed in one operating system is not executed in other operating system. Therefore, these types of applications are called platform dependent.

What do you mean by managed code?

- The code that gets executed with the help of Operating System is called “Unmanaged Code”.
- The code that gets executed with the help of CLR is called “Managed Code”.
- The code that targets CLR while execution is called “Managed Code”.
- Managed code is faster in execution.

The .NET Class Framework

The next layer up in the framework is called the .NET Class Framework also referred as .NET base class library. The .NET Class Framework consists of several thousand type definitions, where each type exposes some functionality. All in all, the CLR and the .NET Class Framework allow developers to build the following kinds of applications:

- Web Services. Components that can be accessed over the Internet very easily.

- Web Forms. HTML based applications (Web Sites).
- Windows Forms. Rich Windows GUI applications. Windows form applications can take advantage of controls, mouse and keyboard events and can talk directly to the underlying OS.
- Windows Console Applications. Compilers, utilities and tools are typically implemented as console applications.
- Windows Services. It is possible to build service applications controllable via the Windows Service Control Manager (SCM) using the .NET Framework.
- Component Library. .NET Framework allows you to build stand-alone components (types) that may be easily incorporated into any of the above mentioned application types.
- As per .NET, a DLL contains platform independent code in the form of Byte code.
- DLL is also called as Managed Code.
- DLL is also called as Assembly.

Hardware Requirements:
RAM: 128MB RAM or Higher
Hard disk: 80GB with 300 MB free space at the root directory.
Monitory, Keyboard, Mouse.
Software Requirements:
OS: Windows XP/2000/NT/2003/Vista/Windows7/Windows8/Windows 8.1
Environment/ suite: Microsoft Visual Studio 2010
Webserver: IIS (Internet Information Services)
Database: SQL Server/ Any other database.

FCL-Framework Class Library

JAVA	**DOTNET**
Packages	Namespace
Classes	Classes
Methods	Methods

"A collection of Namespaces is called an Assembly". Assembly available under C:\Windows\assembly
"A Collection of Assemblies is called "Framework Class Library" (FCL).
DLL is also called "Microsoft Intermediate Language" (MSIL).

.NET is partially platform independent.

Every operating system contains a specific set of instruction set. This instruction set format will be varied from operating system to operating system. Thus an application developed in one operating system is not executed in other operating system. Therefore, these types of applications are called platform dependent.

Java	.class	JVM JVM JVM	Windows OS Linux OS Solaris OS

.NET	DLL/EXE	CLR Mono CLR Compact CLR No CLR	Windows OS Linux OS LWP Devices Win 95/98 DOS

Common language runtime is the runtime environment for C#&VB, which is similar to JRE in java.

DLL file contains reusable code.

Project1 Project2	Logic(DLL)

In .NET the compiled code converted into managed code which is similarly called Byte code in java

DOT NET	JAVA
CLR	JRE
MANAGED CODE	BYTE CODE

What do you mean by managed code?

- The code that gets executed with the help of Operating System is called “Unmanaged Code”.
- The code that gets executed with the help of CLR is called “Managed Code”.
- The code that targets CLR while execution is called “Managed Code”.
- Managed code is faster in execution.

What is a DLL as per .NET?

- As per .NET, a DLL contains platform independent code in the form of Byte code.
- DLL is also called as Managed Code.
- DLL is also called as Assembly.

Hardware Requirements:

RAM: 128MB RAM or Higher
Hard disk: 80GB with 300 MB free space at the root directory.
Monitory, Keyboard, Mouse.

Software Requirements:

OS: Windows XP/2000/NT/2003/Vista/Windows7/Windows8/Windows 8.1

Environment/ suite: Microsoft Visual Studio 2010
Webserver: IIS (Internet Information Services)
Database: SQL Server/ Any other database.

FCL-Framework Class Library

JAVA	DOTNET
Packages	Namespace
Classes	Classes
Methods	Methods

“A collection of Namespaces is called an Assembly”. Assembly available under

C:\Windows\assembly

“A Collection of Assemblies is called “Framework Class Library” (FCL).

DLL is also called “Microsoft Intermediate Language” (MSIL).