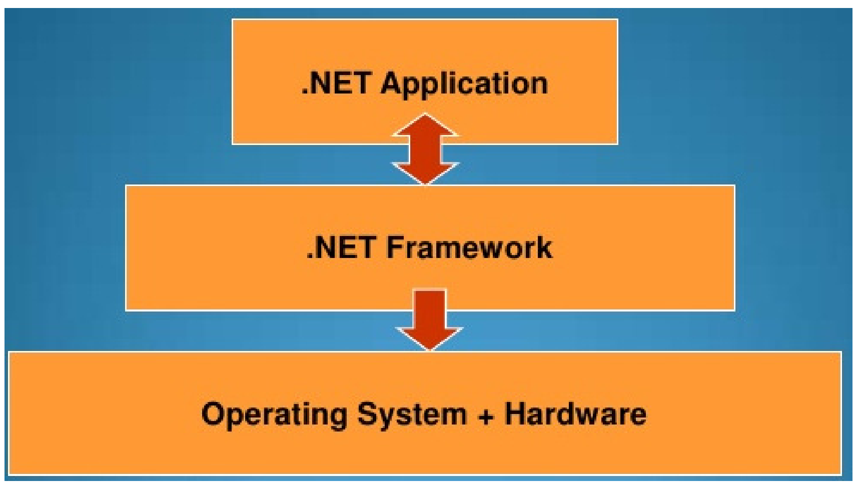
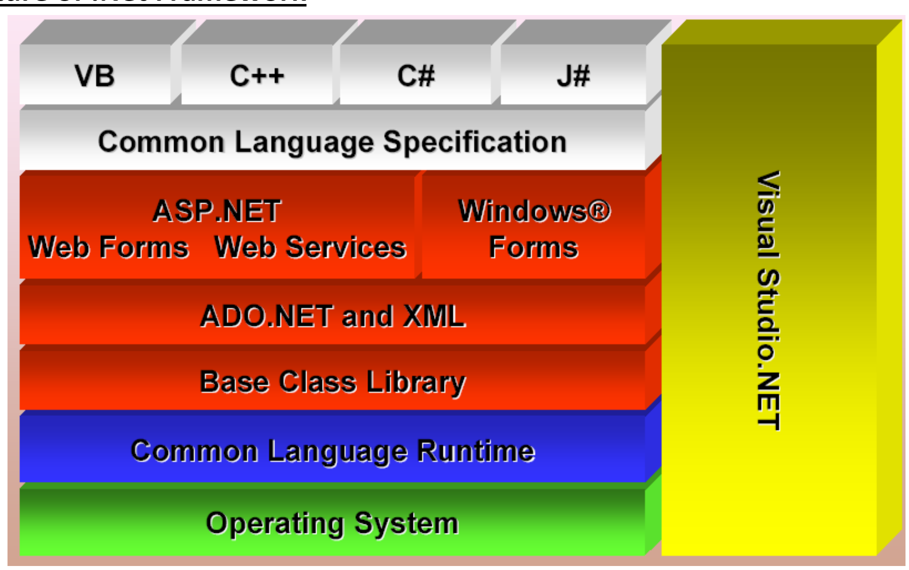
1.1 What is .Net Framework?

- Net Framework is a software development platform developed by Microsoft for developing **Windows based applications and web based application**. It consists of developer tools, programming languages, and libraries to build desktop and web applications.
- It is also used to build **websites, web services, and games**.

Narrow view of .NET Application



Architecture of .Net Framework

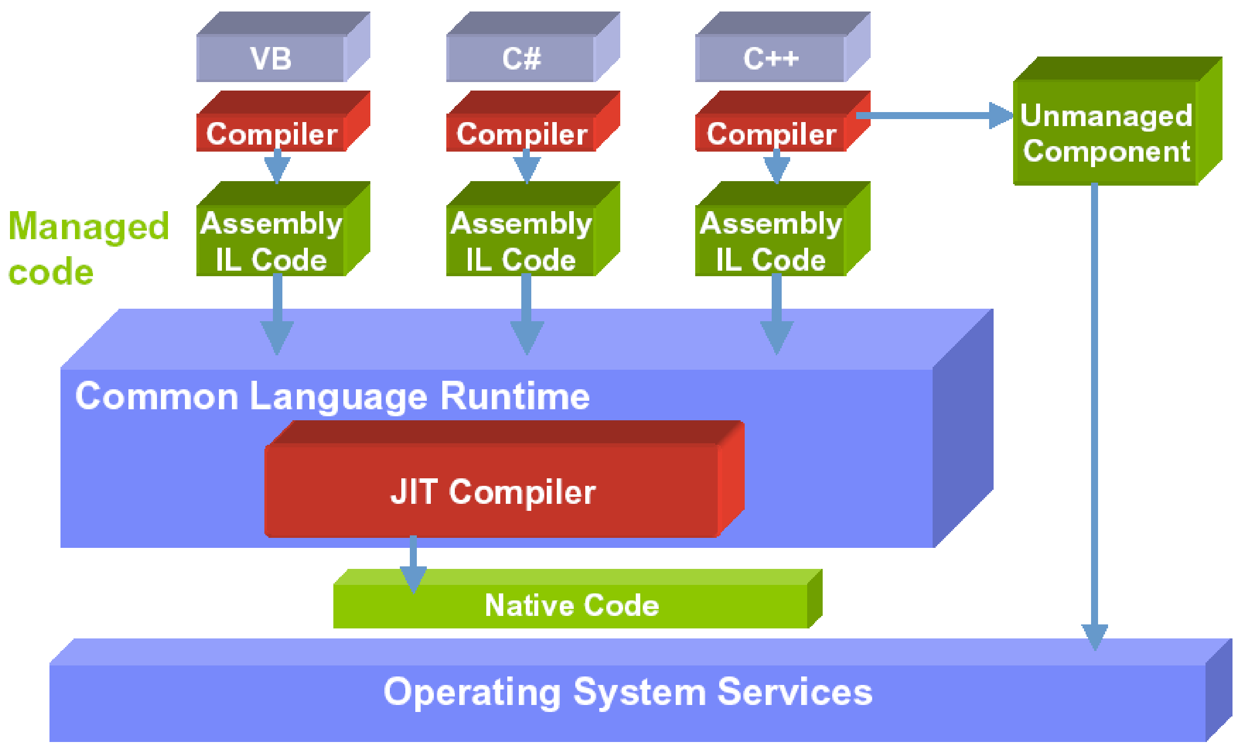


The .NET Framework consists of:

(1) Common Language Runtime (CLR)

- .NET Framework provides runtime environment called **Common Language Runtime (CLR).**
- It is the **heart** of .net framework. It is the engine that compiles and run the application. It uses MSIL code which is language independent for execution. The MSIL code is translated by JIT compiler.
- It provides an environment to run all the .NET Programs.
- The code which runs under the CLR is called as **Managed Code**.
- Programmers need not to worry on managing the memory if the programs are running under the CLR as it provides **memory management and thread management**.

Execution In CLR



Microsoft Intermediate Language

- MSIL stands for Microsoft Intermediate Language. We can call it as **Intermediate Language (IL)** or **Common Intermediate Language (CIL).**
- The .NET Compiler will create MSIL while you run your program and JIT (Just in Time compiler) will convert your Intermediate Language to machine code (Native Code).
- Microsoft Intermediate Language (MSIL) is a **CPU-independent** set of instructions that can be efficiently converted to the native code.
- Microsoft Intermediate Language is also called **Manage Code.**
- When a compiler produces Microsoft Intermediate Language (MSIL), it also produces **Metadata** (Data about Data).
- Metadata is nothing but a description of every namespace, class, method, Property etc.
- The Microsoft Intermediate Language (MSIL) and Metadata are contained in a **portable executable (PE) file.**
- Microsoft Intermediate Language (MSIL) includes instructions for loading, storing, initializing, and calling methods on objects.

Base Class Libraries (BCL)

- It is also known as **framework class library(FCL)**
- It is the object-oriented collection of reusable types. It is a Library of **prepackaged functionality and Applications**.
- It provides classes which encapsulate a number of common functions, including file reading & writing, Grahics rendering, database interaction and XML document manipulation.

(2) ADO.Net and XML

- It is also known as **data access layer**. With the help of this layer we can access relational databases. It work with XML and provides the disconnected Data Model.
- It is a part of BCL.it consists of two parts
    - Data Provider
    - Data Set

(3) Window Forms

- It is also known as Win Forms. It is used to create GUI for windows desktop application.
- it also provides integrated and unified way of developing GUI
- It has a rich variety of **windows controls** and user interface support like **Textbox, Button, Checkbox, Etc.**
- Using visual Studio.NET, we can simply design the GUI by dragging the controls on a form.

(4) Web Forms & web Services

- It provides a tool for web application. It is a part of ASP. Net
- It is the forms engine that provides **Browser –based user interface**.
- Web Forms are similar to Windows Forms in that they provide properties, methods, and events for the controls that are placed onto them.
- However, these UI elements render themselves in the appropriate markup language required by the request, e.g. HTML.
- If you use Microsoft Visual Studio® .NET, you will also get the familiar drag-and-drop interface used to create your UI for your Web application.

Web Services

- Web services are the applications that run on a web server and communicate with other application. It uses a series of **XML** based communicating protocols that respond to different requests.
- The protocols on which web services are built summarized below:
- UDDI (Universal Discovery and Description Integration)
- WSDL (Web services Description Language)
- SOAP (Simple Object Access Protocol)
- XML (Extensible Markup Language),HTTP(Hypertext Transfer Protocol),SMTP(Simple Mail Transfer Protocol)

(5) The Common Language Specification (CLS)

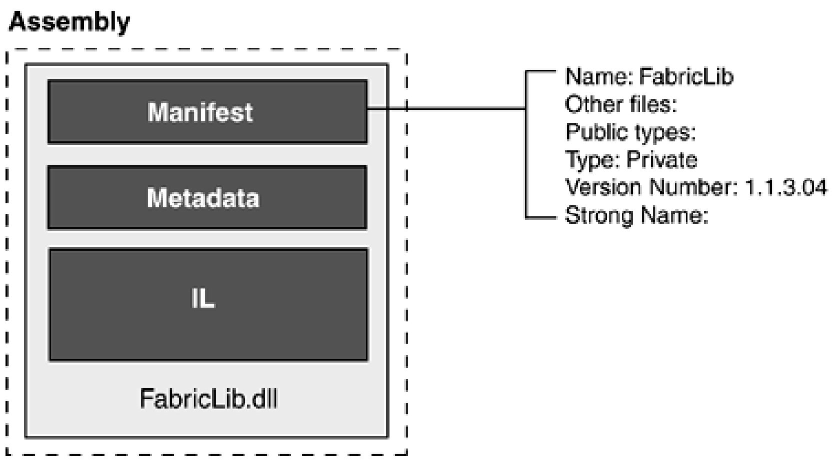
- It is a set of rules and constraints that all language must follow which want to be compatible with .NET framework.
- It is used to support the theme of .NET i.e. **unification** and **interoperability** (The ability of computer systems or software to exchange and make use of information). That means, if we want the code which we write in a language to be used by programs in other language(cross-language integration) then it should hold on to the CLS.
- Thus the CLS describes a set of features that are common different languages.

  CLS performs the following functions:

  - Establishes a framework that helps enable 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.

Some Concept which are important in .Net

Assembly All of the managed code that runs in .NET must be contained in an assembly. Logically, the assembly is referenced as one **EXE or DLL file**. Physically, it may consist of a collection of one or more files that contain code or resources such as images or XML data. An assembly is created when a .NET compatible compiler converts a file containing source code into a DLL or EXE file.



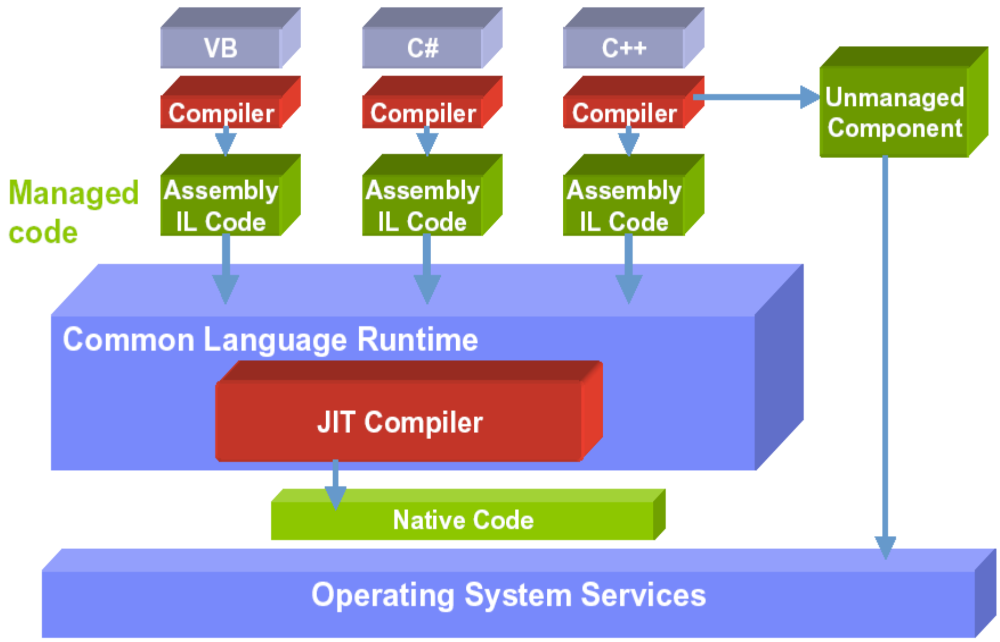
Manifest. Each assembly must have one file that contains a manifest. The manifest is a set of tables containing metadata that lists the names of all files in the assembly, references to external assemblies, and information such as name and version that identify the assembly.

Metadata When a compiler produces Microsoft Intermediate Language (MSIL), it also produces Metadata (**Data about Data**). Metadata is nothing but a description of every namespace, class, method, Property etc. **The Microsoft Intermediate Language (MSIL) and Metadata are contained in a portable executable (PE) file**. It allows loading and locating code, enforcing code security, generating native code, and providing reflection at runtime.

1.2 The Common Language Runtime

- .NET Framework provides runtime environment called Common Language Runtime (CLR).It is the **heart** of .net framework. It is the **engine** that compiles and run the application. It uses MSIL code which is language independent for execution. The MSIL code is translated by JIT compiler.
- It provides an environment to run all the .NET Programs. The code which runs under the CLR is called as **Managed Code**. Programmers need not to worry on managing the memory if the programs are running under the CLR as it provides memory management and thread management.

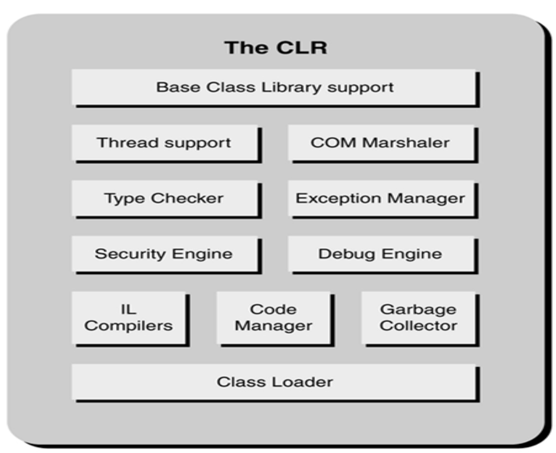
Execution in CLR



When the .NET program is compiled, the output of the compiler is not an executable file but a file that constraints a special type of code is called Microsoft intermediate language , which is a low level set of instructions understand by CLR.

- The MSIL defines a set of portable instructions that are CPU indendendent.
- It's the job of the CLR to translate this MSIL into native code when the program is executed, making the program to **run in any environment** for which the CLR is implemented. And that's how the .NET framework achieves **Portability (run in any environment)**. This MSIL is converts into Native code using JIT(Just In Time)compiler

Purpose of CLR



- **Thread Support:** Threads are managed under the Common Language Runtime. Threading means **parallel code execution**. Threads are basically light weight processes responsible for **multi-tasking** within a single application.
- **COM Marshaler**: It allows the communication between the application and COM objects.
- **Type Checker** Type checker will verify types used in the application with CTS or CLS standards supported by CLR, this provides type safety.
- **Exception Manager:** it handles all the runtime exceptions(Error) thrown by application
- **Security Engine**: It enforces security permissions at **code level security, folder level security, and machine level security** using Dot Net Framework setting and tools provided by Dot Net.
- **Debug Engine**: CLR allows us to perform debugging an application during runtime.
- **MSIL**: Microsoft Intermediate Language is considered to be the lowest form of human readable language. It is CPU independent and includes instructions of how to load, store, initialize objects. JIT converts this MSIL into native code which is independent on the CPU
- **Code Manager:** CLR manages code. When we compile a .NET application you don't generate code that can actually execute on your machine. You actually generate Microsoft Intermediate Language (MSIL or IL). All .NET code is IL code. IL code is also called Managed Code, because the .NET Common Language Runtime manages it.
- **Garbage Collector** Garbage Collector handles automatic memory management and it will release memory of unused objects in an application, this provides automatic memory management.
- **Class Loader:** as and when needed. It loads the class into the system memory.

1.3 The .Net Framework Class Library

- The Framework Class Library (FCL) is a collection of classes and other types (enumerations, structures, and interfaces) that are available to managed code written in any language that targets the CLR.
- The resources within the FCL are organized into **logical groupings** called **namespaces**. For example, types used for graphical operations are grouped into the **System.Drawing**.
- Types required for **file I/O** are members of the **System.IO namespace**. Namespaces represent a logical concept.
- The FCL comprises hundreds of assemblies (DLLs), and each assembly may contain multiple namespaces.

1.3.1 Namespace (IMP)

- Namespace is the Logical group of types (Class). We can say Namespace is a container (e.g. Class, Structures, Interfaces, and Enumerations etc).

Lists some of the most important namespaces in .NET.

Namespace	Use
System.Data System.Data.OracleClient System.Data.SqlClient System.Data.OleDb System.Data.Odbc	Classes used for database operations (**ADO.NET**). The client namespaces support Oracle and SQL Server, respectively; OledDb and Odbc define the data connection used.
System.IO	Provides **file and data stream I/O**. These classes provide a way to access the underlying file systems of the host operating system.
System.Windows.Forms	Classes used to build Windows desktop GUI applications. Controls including the ListBox, TextBox, DataGrid, and buttons are found here.
System.Xml	Types for processing XML.
System.Web	The Internet-related classes referred to as ASP.NET. They manage browser-server communication requirements, manipulate cookies, and contain the controls that adorn a Web page.
System.Web.Services	Web.Services includes those classes required for SOAP-based XML messaging.

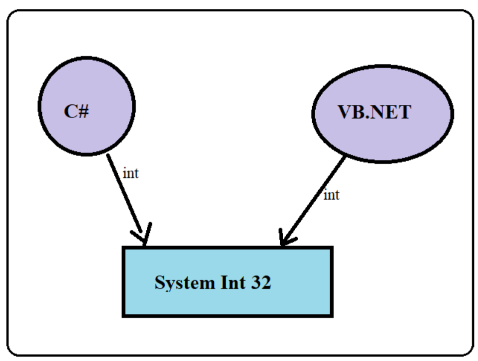
1.3.2 Common Type System (CTS)

It defines how types are declared, used, and managed in the **common language 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 helps enable 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.
- Provides a library that contains the primitive data types (such as Boolean, Byte, Char, Int32, and UInt64) used in application development**.**

For example, **C#** has an **int** data type and **VB.NET** has **Integer** data type. Hence a variable declared as an **int in C#** and **Integer in VB.NET**, finally after compilation, uses the same structure **Int32** from **CTS**.



1.3.2.1 Data Types

SrNo.	Data Types	Storage Size	Description
1	Short	2 bytes	-32,768 to 32,767
2	Integer	4 bytes	-2,147,483,648 to 2,147,483,647
3	Long	8 bytes	-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
4	Single	4 bytes	-3.402823E+38 to -1.401298E-45 for negative values; 1.401298E-45 to 3.402823E+38 for positive values
5	Double	8 bytes	-1.79769313486231E+308 to 4.94065645841247E-324 for negative values; 4.94065645841247E-324 to 1.79769313486231E+308 for positive values
6	Decimal	16 bytes	Signed integer that can have 28 digits on either side of decimal
7	Char	2 bytes	0 to 65535 (unsigned)
8	String	Variable length	0 to approximately 2 billion Unicode characters
9	Boolean	4 bytes	True or False
10	Byte	1 bytes	0 to 255 (unsigned)
11	Date	8 bytes	January 1, 0001 to December 31, 9999
12	Object	4 bytes	Any type can be stored in a variable of type Object