Introduction to the .NET Platform

.NET (pronounced as dot net) is a platform introduced by Microsoft to develop, deploy, host, maintain and execute various applications. The Microsoft .NET Platform provides all the tools, libraries and technologies that you need to build various applications. .NET benefits end user by providing applications of higher capability, quality and security. .NET platform provides the ability to quickly build, deploy, manage, and use connected, security-enhanced solutions with web services. *.NET Platform provides several core technologies as shown below:*

- The .NET Framework
- The .NET Services
- The .NET Enterprise Servers
- Visual Studio .NET

The .NET Framework

The .NET Framework is based on a new common language runtime. The common language runtime provides a common set of services for projects built in Microsoft Visual Studio .NET IDE, regardless of the language. These services provide a key building blocks for applications of any type, across all application tiers. The .NET Framework is explained in great detail later in this module.

.NET My Services

.NET my services is a set of user-centric XML Web services. With .NET My service, users receive relevant information as they need it, delivered to the devices they are using , and based on preferences they have established. Using .NET My services, applications can communicate directly by using SOAP and XML from any platform that supports SOAP.

Core .NET My Services include:

- .NET Passport authentication.
- The ability to send alerts and manage preferences for receiving alerts.
- The storage of personal information (including contacts, e-mails, calendar, profiles, lists, electronic wallet, and physical location).
- The ability to maintain document stores, save application settings, record favorite web sites, and note devices owned.

The .NET Enterprise Servers

The .NET Enterprise Servers provide scalability, reliability, management, integration within and across organization, and many other features. Some of the server names are given below:

A. Microsoft SQL Server
B. Microsoft SharePoint Server
C. Microsoft Content Management Server

There are many more that you can learn in advance learnings.

Visual Studio .NET

Visual Studio.NET provides a development environments for building applications on the .NET Framework. It provides important enabling technologies to simplify the creation, deployment, and ongoing evolution of secure, scalable, highly available Web Applications and XML Web services. You will learn more about it in coming module.

Overview of the .NET Framework:

.NET Framework is nothing but a set of components, configuration file, documentation and other similar items which are bundled together. The .NET Framework provides the necessary compile-time and run-time foundation to build and run .NET-based applications. The .NET Framework is one the most popular and widely used integrated software development environments today.

Earlier software developer faced a lot of difficulties while integrating code that was written in different programming languages. This was because each language required a different execution environment to execute the code written in that language. For E.g. the code written by using Visual Basic 6.0 required a different execution environment than that required by code written by using Visual C++.

Later Microsoft introduced .NET Framework, which provided programmers a single platform to develop console, windows, and web applications in various programming languages, such as Visual Basic and Visual C#. The .NET Framework helped to reduce the complexities involved in developing large and reliable .NET applications.

Microsoft's released versions of .NET Framework:

- .NET Framework 1.0: In 2002.
- .NET Framework 1.1: In 2003.
- .NET Framework 2.0: In 2005.
- .NET Framework 3.0: In 2006.
- .NET Framework 3.5: In 2007.
- .NET Framework 4.0: In 2010.
- .NET Framework 4.5: In 2012.

By using .NET Framework latest version 4.5, you can build application for windows, windows phone, windows store, windows server and windows azure.

Further .NET Framework is designed to fulfill the following objectives:

- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that promotes safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the .NET Framework can integrate with any other code.

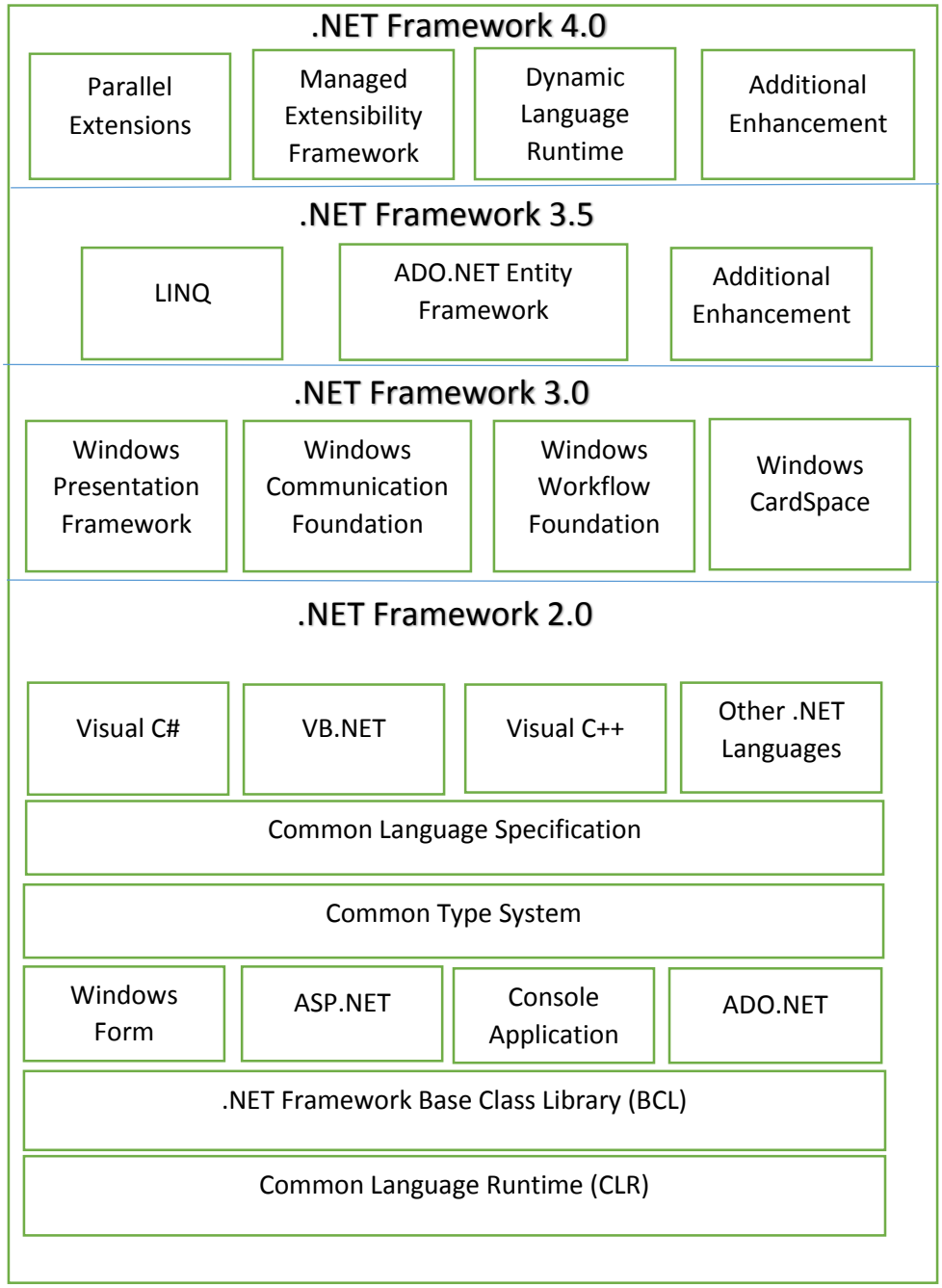
Why .NET is called .NET?

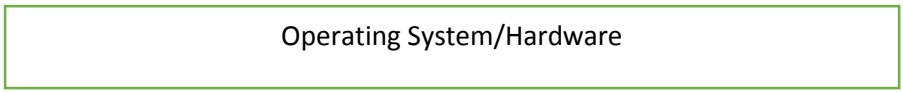
Microsoft started development on the .NET in the late of 1990s, originally under the name of "NGWS (Next Generation Windows Services)" . So from beginning Microsoft wanted to develop such platform which will provide a good connectivity for information, people, systems and devices through it and that could be possible only by the medium of internet. .NET is a business strategy from Microsoft that provide a collection of programming support for web services which is the ability to use web rather than your own computer for various services.

The reason of putting Dot (.) before NET is to give message that all the classes available under namespaces and classes members are accessed through dot. The OOPS concept of C# is incomplete without dot. So dot has very important role in .NET.

Architecture of .NET Framework 4.5

The .NET Framework consists primarily of a gigantic library of code. .NET Architecture consists of various components and each provide specific functionality. .NET Framework 2.0, 3.0, 3.5 and 4.0 (along with their service packs) form the foundation of .NET Framework 4.5 .In other words, the architecture of .NET Framework 4.5 includes the components of .NET Framework 2.0, 3.0, 3.5 and 4.0 along with some additional enhancements.





Explaining the Components of .NET Framework 4.5

Some of the basic components of .NET Framework 4.5 are as follows:

- CLR
- CTS and CLS
- Assemblies and metadata
- .NET Framework Class Library(FCL)
- BCL
- Windows Forms
- ADO.NET
- ASP.NET and ASP.NET AJAX
- WPF
- WCF
- WF
- WCS
- LINQ

Common Language Runtime (CLR)

"The CLR is the execution environment for code written for the .NET Framework"

The CLR manages the execution of .NET Code, including following services:

1. Memory allocation and garbage collection (which helps avoid memory leaks).
2. Security (including applying differing trust levels to code from different sources).
3. Thread Management.
4. Enforcing type-safety, and many other tasks.
5. Language Interoperability.

In simple words you can say that CLR manages the execution of .NET applications. The code that runs under CLR or target CLR is known as **managed code**, while **unmanaged code** is the code that is compiled into native or machine code and is directly executed by an operating system.

Benefits of Managed Code:

1. Cross-language integration.
2. Enhanced security.
3. Type safety.
4. Memory Management.
5. Secure and stable applications.

The CLR works with every language available for the .NET Framework, so there is no need to have a separate runtime for each language.

Interoperability between various .NET language, such as visual C#, visual basic, or Visual C++ is facilitated by CLR, as it provides a common environment for the execution of code written in any of these languages. All .NET applications, when run are compiled into an intermediate code called the Microsoft Intermediate Language (MSIL) or IL code, by the language compiler. The MSIL code is then used by the Just-In-Time (JIT) compiler to compile the code into the native machine code, which is the final executable code.

Understanding CLR and Execution of .NET Application in Depth

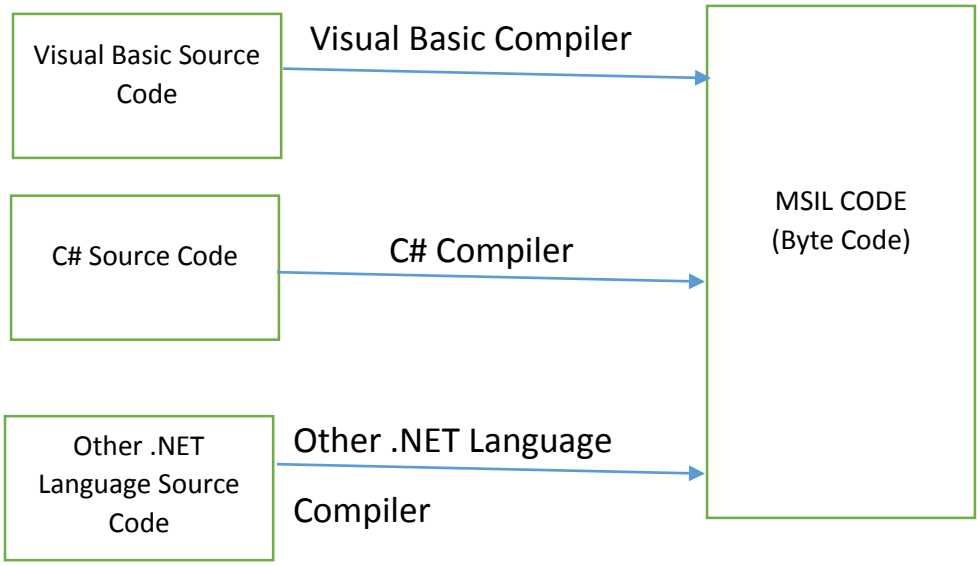
Step 1: Compiling to the Microsoft Intermediate Language (MSIL)

The compiler for a .NET language takes a source code file and produces an output file called an assembly. An assembly is either an executable or a DLL. The process is illustrated in below figure.

The code in an assembly is not native machine code, but an intermediate language called Common Intermediate Language (CIL).

An assembly, among other things, contains the following items:

A. The program's CIL
B. Metadata about the types used in the program.
C. Metadata about references to other assemblies.



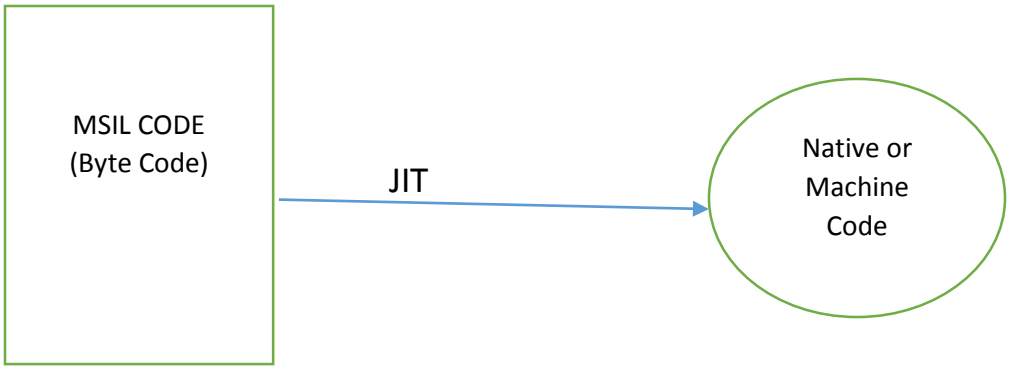
The acronym for the intermediate language has changed over time, and different references use different terms. Two other terms for the CIL that you might encounter are IL (Intermediate Language) and MSIL (Microsoft Intermediate Language), which was used during initial development and early documentation.

Step 2: Compiling to Native Code and Execution

The program's CIL is not compiled to native machine code until it is called to run. At run time, the CLR performs the following steps:

A. It checks the assembly's security characteristics.
B. It allocates space in memory.
C. It sends the assembly's executable code to the Just-In-Time (JIT) compiler portions of it to native code.

The executable code in the assembly is compiled by the JIT compiler as it is needed. It is then cached in case it is needed for execution again later in the program. Using this process means that code that isn't called isn't compiled to native code, and code that is called is only compiled once.



Common Type System (CTS) and Common Language Specification (CLS)

An important component of the .NET Framework that provides support for cross-language integration among .NET-enabled programming languages or Language Interoperability is CTS. CTS defined a common set of types that can be used in applications written in different .NET languages and the operations that can be performed on these types. Two CTS-compliant languages do not require type conversions, when calling a code written in one language from a code written in another language. For all the languages supported by the .NET Framework, CTS provides a base set of data types; however, each language uses aliases for the base data types provided by CTS. For instance, the keyword integer in visual basic and int in C# refer to 32-bit signed integer. CTS therefore uses the data type int32 to represent a 32-bit integer value.

CLS is a subset of CTS and defined a set of basic languages features and programming constructs that all .NET applications have in common and the rules which they must confirm. This enables interoperability between two .NET-compliant languages. The rules defined in CLS serve as guidelines for third-party compilers, designers, and library builders. CLS being a subset of CTS, the languages supported by CLS can use each other's class libraries similar to their own. Application programming interface (APIs), which are designed by following the rules defined in CLS, can be used by all .NET-compliant languages.

Assemblies and Metadata

An assembly is the basic building block of any .NET application. It is a single, logical deployment unit. Self-describing in nature, assemblies contain all the information that is needed by CLR to execute an application that targets CLR. Every .NET application is compiled into an assembly, which forms a unit of deployment, versioning, code reuse, and security of the application. This implies that everything you do in .NET application occurs in the scope of an assembly.

An assembly is created in the form of a Portable Executable (PE) file. This PE file can either be a dynamic link library (.dll file) or an executable file (.exe file) that contains the MSIL code of the compiled application.

An assembly is a collection of types and resources of types and resources that include assembly metadata, and type metadata besides the MSIL. Assembly metadata is the metadata about the assembly itself and its components, while type metadata is the metadata about the types that are required for the assembly, such as classes, interfaces, enumerations, and structures.

Types of Assemblies:

There are basically two type of assemblies which can be created in .NET:

A. Private Assembly: It is used by a single .NET application and stored in the application's directory.
B. Shared Assembly: It is referenced by more than one application. Shared assembly must have a unique name (called strong name) and are stored in Global Assembly Cache (GAC).

Metadata describes a program that is in the form of binary information stored in an assembly. When code is compiled in a PE file, the metadata is inserted into one part of the file, while the code is converted into MSIL and inserted into another part of the file. Metadata describes every type and member of program. At runtime, CLR loads the metadata into memory and finds information about the code, such as the classes and members specified in a program.

Metadata contains information about the code inside a program in a language-neutral manner. It includes the following information:

A. Assembly information such as the identity of the assembly which can be a name, version, culture, or public key. The types of assemblies, other referenced assemblies and security permissions.
B. Information about types, such as name, visibility, base class, interfaces used, and members (methods, fields, properties, events, and nested types).
C. Attribute information which modifies the types and members of a class.

The information contained in the metadata of a .NET application is used by CLR to create objects, access data and call member functions used in the assembly at run time.

.NET Framework Class Library (FCL)

.NET Framework consist of classes, interfaces, and value types that help in speeding up the application development process and provide access to system functionality. To facilitate interoperability between languages, .NET Framework should be CLS-compliant so that the code developed in programming language can be used from another programming language whose compiler conforms to the CLS.

The .NET applications, components, and controls are built on the foundation of .NET Framework types. These types performs the following functions:

- Represent base data types and exceptions.
- Encapsulate data structures.
- Perform input/output operations.
- Access information about loaded types.
- Call .NET Framework security checks.
- Facilitate data access, rich client-side graphical user interface (GUI), and server-controlled client-side GUI.

The FCL is a huge library of reusable types meant to be used by managed codes. It is an object-oriented library that is used in component-based applications. The FCL is made up of a hierarchy of namespaces that exposes classes, structures, interfaces, enumerations, and delegates. The namespaces are logically defined by their functionality. For example, The System.Data namespace contains the functionalities available for accessing databases. This namespace is further broken down into the System.Data.SqlClient and System.Data.OleDb namespaces. The System.Data.SqlClient namespace exposes the functionality specific to SQL Server, Whereas, the Syste.Data.OleDb namespace exposes a specific functionality for accessing Object Linking and Embedding Database (OLEDB) data sources. Specific assemblies within the FCL do not define the bounds of a namespace, instead they are focused on functionality and logical grouping. There are more than 20,000 classes in FCL, all logically grouped in a hierarchical manner. In other words, you can write applications that make use of objects in the FCL to read files, display windows, and perform various tasks. However, to exploit the true potential of classes, you can extend the FCL according to the requirements of your application and then write your application code.

The following points need to be remembered while using the FCL:

- The classes, interfaces, structures, and enumerated values are collectively referred as types.
- The different types in the framework are arranged in a hierarchy of namespaces. This solves the problem faced in name collisions. Namespaces are arranged in a hierarchy that has words separated by Dots (.) , therefor we can include namespace, such as System.IO.Stream and System.Windows.Forms.Form, for a code that uses a Stream class or a Form class respectively.
- The System namespace is a parent of all the types available in the .NET FCL. It contains base data type's classes, such as Object, Int, Array, and Short.
- All the types must have a base class. The exception to this rule is the System.Object type, which acts as the base type for all the types in .NET Framework. This implies that if you create a class that does not explicitly declare a base class, then the compiler implicitly declares its base class to the Object class.

Sometimes, FCL is referred to as BCL, which is incorrect. Actually, FCL is a superset of BCL.

Base Class Library (BCL)

The BCL is the library that contains some common set of functionalities, such as reading from and writing in files, designing graphics, and interacting with database, for all the .NET Framework languages. Microsoft has upgraded the BCL with every release of .NET Framework. Some improvements and new functionalities are added in .NET Framework 4.5 that will be discussed in brief in later modules.

Windows Forms

Windows Forms are Windows-based GUI applications that users can install and run on their computers. The basic component of a Windows Forms applications is a form, which is a visual rectangular area on which either information is displayed to a user or some input from the user is accepted. A variety of controls can be placed inside this form to enhance the UI of the applications. The .NET Framework facilitates the development of Windows Forms applications by using a .NET-compliant language. A windows Forms application is an event-driven application, which means that some action is performed whenever an event, such as mouse click or pressing of a key in the keyboard takes place.

ADO.NET (ActiveX Database Objects)

ADO.NET is a Microsoft technology that provides a consistent way to access data sources and databases of all types. It provides access to various data sources, such as Microsoft SQL server, and data sources exposed through Object Linking and Embedding Data Base (OLE DB) and Extensible Markup Language (XML). ADO.NET can be used to connect to data source to retrieve, manipulate, and update the data. The most important feature of ADO.NET is disconnected data architecture. In this architecture, .NET application are connected to the databases only till data is being retrieved or modified. The data access operation in ADO.NET is made possible with the help of two components, namely datasets and the data provider. Datasets, which refer to a cached set of database records, follow disconnected data architecture to access or modify data. Therefore, applications do not require connecting to the database to process each record. In addition, all database operations are performed on the dataset instead of the database. The data provider, on the other hand, can be thought of as a bridge that helps to connect an application to a database. Using a data provider, an application can execute commands on a database and retrieve results from the database. More information regarding ADO.NET is covered in the coming modules.

ASP.NET and ASP.NET AJAX

ASP.NET is a part of the Microsoft .NET Framework, which with its powerful and rich set of features enables .NET programmers to develop enterprise-level server-based dynamic and interactive Web applications. With the ASP.NET Web development model, you can add HyperText Markup Language (HTML), Web Server, and custom as well as user controls to a Web Form. In ASP.NET Web applications, a Web Form is the basic UI entity. You can also specify the behavior of a Web server and the custom or user controls and add functionalities for these controls in the code of a Web application.

An ASP.NET application can be written by using any of the .NET language, such as Visual Basic, Visual C# or Visual C++.

AJAX is acronym for Asynchronous JavaScript and XML and was earlier known as Atlas. It is an extension to ASP.NET, which means that the AJAX functionality can be added to ASP.NET applications. AJAX facilitates the transfer of information between a Web server and a client in such a way that AJAX –based applications do not require a Web page to be reloaded again while certain modifications are being made to the page. Using AJAX, .NET developers can send only the modified

portions of web page through asynchronous calls to the web server. This decreases network traffic and the processing time of a web page on the web server. Both Google Search Engine and Gmail are an example of a popular AJAX-enabled Web applications.

Windows Presentation Foundation (WPF)

WPF is a component of .NET Framework 3.0 and issued to create applications with visually appealing UIs. Formerly known as Avalon, WPF combines two-dimensional (2D) and three-dimensional (3D) graphics, Microsoft Word documents, Portable Document Format (PDF) files, and multimedia into a single framework. It also offers a consistent programming model to develop WPF applications, in which the UI and business logic can be clearly separated. In addition, WPF offers Extensible Application Markup Language (XAML), which is a markup language created by Microsoft. With XAML, both .NET application developers and designers can easily and efficiently work and collaborate on the development process of WPF applications. You can also use WPF to create a wide range of standalone and web applications.

Windows Communication Foundation (WCF)

One of the main technologies introduced with .NET Framework 3.0 was WCF, which provides a unified programming model to build service-oriented applications. For example, you can create a service-oriented application in which a service developed in .NET is used by another application that is built on a different platform or programming language. WCF combines and extends the capabilities of distributed systems, .NET remoting and web services. All these features help service-oriented applications to communicate with one another, thereby simplifying the development process.

Windows Workflow Foundation (WF)

WF is a Microsoft technology introduced in .NET Framework 3.0, which helps define, execute and manage workflows. A workflow generally refers to a sequence of tasks that are performed to complete a given procedure or produce some result. WF applications consist of a series of shapes and icons that represent some actions. Before the advent of WF, application developers used to write both the business logic and its implementation in .NET language, such as C#, Visual Basic. However with WF the business logic code can be defined in a workflow application while the actual implementation code is written in a .NET language.

Therefore, WF provides a declarative programming model to build workflow applications that only has the business logic code.

Windows CardSpace (WCS)

WCS is a client software introduced by Microsoft that enables users to use their digital identities over the Internet in a familiar, simple and secure way. WCS can be thought of as an online virtual information card or an ID card for the Internet which can be used to validate a user's identity. Using WCS, .NET programmers can build websites and software that are prone to identity related attacks, such as phishing. WCS takes care of the problems of traditional online security mechanisms by reducing the dependence on user names and passwords. WCS uses a separate desktop and cryptographically strong authentication to ensure secure online transactions.

Language-Integrated Query (LINQ)

LINQ is a .NET Framework component that adds native data querying capabilities to .NET languages. It has syntax similar to Structured Query Language (SQL), which means that you can use LINQ to write statements in any .NET language, such as Visual C# and Visual Basic. LINQ offers a consistent model to work with data across various data sources and formats, such as SQL Server databases, datasets objects in a collection and XML documents. The operations performed by a LINQ query typically consist of three main actions:

A. Obtaining a data source.
B. Creating the query.
C. Executing the query.

Describing the benefits of .NET Framework 4.5

The .NET Framework is an essential Windows component that supports the development and deployment of console, windows, web-based applications and services. It is designed to provide a code execution or runtime environment with minimal versioning conflicts and application deployment issues.

The .NET Framework offers a lot of benefits to store developers such as the following:

- **Comprehensive and consistent programming model:** Provides a consistent object-oriented programming model across various languages. You can use this model to create programs to perform different tasks, such as connecting to databases and retrieving data from them as well as reading from files and writing to them.
- **Cross-platform support:** Enables interoperability among those Windows operating systems that support Common Language Runtime (CLR). In other words, any Windows operating system that supports CLR can execute a .NET application.
- **Cross-language interoperability:** Facilitates the reuse of code as the .NET Framework enables code written in one programming language, such as C# to interact with code written in a different programming language such as Visual Basic. This helps to improve the efficiency of the overall development process.
- **Automatic resource management:** Manages and frees resources, such as files, memory and database connections when you no longer need them.
- **Ease of application deployment and maintenance:** Enables .NET applications to be easily deployed. For applications that are not based on the .NET Framework, you simply need to copy the application along with the components it requires, in the directory of the target computer. With the .NET Framework it is possible to quickly install and deploy applications such that the installation of new components does not affect the already existing applications.
- **Easy for developers to use:** In the .NET Framework, code is organized into hierarchical namespaces and classes. The .NET Framework provides a common type system, referred to as the unified type system, which is used by any .NET-compatible language. In the unified type system, all languages elements are objects. There are no variant types, there is only one string type, and all string data is Unicode.

Note: Console Applications are command-line based applications that accept input and display the output at the command-line. Such type of applications have a very simple user interface (UI) and often require little or no user interaction. Applications that have a Graphical User Interface (GUI) and can run locally on user's computers are known as Windows applications. These applications can be created by using any of the .NET programming language, such as C# and Visual Basic, while taking advantage of the .NET Framework debugging tools. Applications that are meant to be hosted through a Web

Languages used in .NET Framework 4.5: One of the imperative features of the .NET is the facility to program in multiple languages, which allows programmers to use their favorite languages. Any language that conforms to the common language specification (CLS) can run on the common language runtime. In the .NeT Framework, Microsoft provides Visual Basic, Visual C#, Visual C++, Visual J#, Microsoft Jscript and many more up to 59 languages. Languages supported by .Net languages are having the fellow features:

- **Standardization by ECMA.**
- **Components Based.**
- **Object Oriented.**
- **Robust and Maintainable Code.**

Key features of visual studio IDE

Visual Studio is an IDE from Microsoft that contains a complete set of development tools for building .NET applications. Visual Studio offers a rich set of tools and features that not only enable the application developers to write and modify their programs, but also helps detect and correct errors. The Visual Studio IDE includes a Code Editor, which is a windows where you can write the source code for an application that you are developing. An integrated debugger is also available in visual studio, which is a tool that helps to monitor the behavior of an application while it is executing. If there is any error in the application you can use the integrated debugger to determine the exact location of an error. `The integrated debugger provides support for IntelliSense, which is a useful feature of Visual Studio and helps to increase the productivity of a .NET application developer. The IntelliSense feature helps in the development of applications as it automatically generates code in the code editor. In addition visual studio provides templates for creating applications such as Class Library, Windows Control Library and Web Control Library. Templates to create Console, Windows, Web applications and Web services are also available in Visual Studio 2012.

Visual Studio 2012 targets .Net Framework 4.5. It helps to minimize the development time of .NET applications by providing different tools to integrate designers into the overall development process.

Visual Studio 2012 also uses a multi-paradigm programming language called Visual F#. This language combines the features of functional programming languages such as C, dynamic programming languages such as Ruby and Python, and object-oriented programming languages such as Java and C#.

Note: A functional programming language is one that focuses more on what a program should do rather than specifying how it should do. A dynamic programming language, also referred to as a weakly typed programming language, enables you to declare variables without defining their data types. The resolution of the actual data type of any variable declared in a dynamic language is deferred in run time. Any programming language that is centered on the concept of objects and follow the principles of abstraction, encapsulation, polymorphism and inheritance is known as an object-oriented programming language.

Before you start working with Visual Studio 2012, you need to know something about it's main features and enhancements. The following is a list of the main features and enhancements introduced in Visual Studio 2012.

- Project Compatibility.
- Configuration changes in ASP.NET 4.5 Website templates.
- Native support in Internet Information Services (IIS) 7 for ASP.NET routing.
- HyperText Markup Language (HTML) editor.
- JavaScript Editor.
- Cascading Style Sheet (CSS) editor.
- The Page inspector tool.
- Publishing.
- IIS Express
- Support for developing Windows Store App.
- Enhanced Cloud capability.
- New tabs in start page.
- Application Lifecycle Management (ALM) and Team Foundation Server (TFS).
- Managing the application lifecycle.
- Modeling applications.
- Developing application and better communication in a team.
- Automating and debugging builds.
- Microsoft Test Manager.

You will learn and implement its features in the coming modules.