Chapter -1 Introduction to .NET Framework and ASP.NET

1.1 Microsoft .NET Framework Overview

.NET is a collection of tools, technologies, and languages that all work together in a Framework to provide the solutions that are needed to easily build and deploy truly robust enterprise applications.

The .NET Framework supports multiple programming languages in a manner that allows language interoperability, whereby each language can utilize code written in other languages.

These .NET applications are able to easily communicate with one another and provide information and application logic, regardless of platforms and languages.

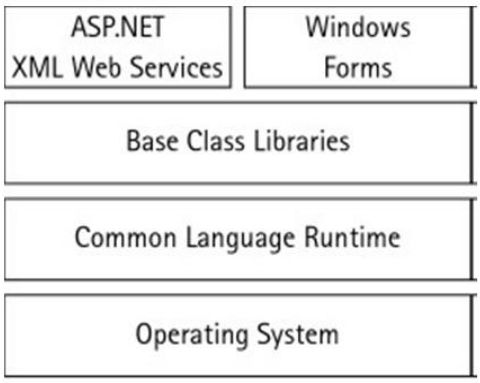


Figure: Structure of .NET Framework

The first layer of .NET Framework is operating System. OS which supports .NET Frameworks are Windows XP, Windows 2000 and later version. Second layer of .NET Framework is CLR (Common Language Runtime).The CLR is the engine that manages the execution of the code.

The next layer up is the .NET Framework base classes. This layer contains classes, value types, and interfaces that you will use often in your development process.
The third layer of Framework is ASP.NET and windows forms. Using ASP.NET, it's now possible to build web applications that are even more functional than Win32 applications of the past. The second part of the top layer of the .NET Framework is the Windows forms section. This is where you can build the traditional executable applications that you built with Visual Basic 6.0 in the past. There are some new features here as well, such as a new drawing class and the capability to program these applications in any of the available .NET languages.

.NET Framework consist of

- CLR(Common Language Runtime)
- Class Libraries
- Support for multiple programming language

1.1.1 .NET ARCHITECTURE

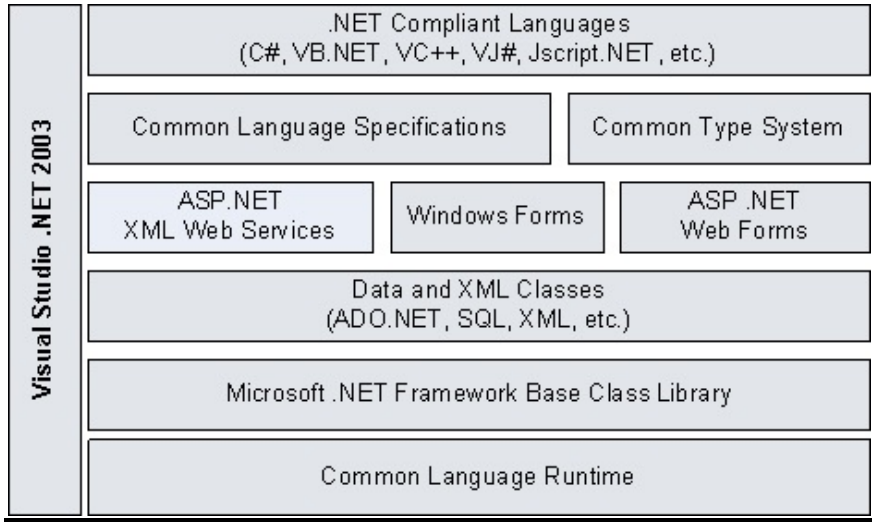


Figure: .NET Architecture

The .NET Framework is factored into several components. The most important component of the framework is the CLR as the heart and soul of the .NET architecture. Every application written using the Framework depends on the CLR. Among other things, the CLR provides a common set of data types, acting as a foundation for C#, VB, and all other languages that target the .NET Framework. Because this foundation is the same no matter which language they choose, developers see a more consistent environment.

The CLS is a statement of rules that allow each language to interoperate. For example, the CLS guarantees that Visual Basic's idea of an integer is the same as C#. Because the two languages agree on the format of the data, they can transparently share the data. The CLS defines not only type information, but also method invocation, error handling, and so forth.

The middle layer called Application Class Libraries and Services. This layer represents the rich set of libraries and APIs that have been created to support virtually all aspects of programming. Graphical user interface APIs or Windows Forms, database APIs through ADO.NET, XML processing, regular expression handling, and so forth are all part of this layer.

Visual Studio .NET is an important part of the .NET Framework because it provides a means for the programmer to access the framework at any level. A programmer can use Visual Studio .NET to write code in many supported managed languages, or he can bypass the CLR altogether and write unmanaged code with Visual Studio .NET.

1.1.2 .NET Framework Components

Following are major components of .net framework

1.1.2.1Common Language Runtime (CLR)

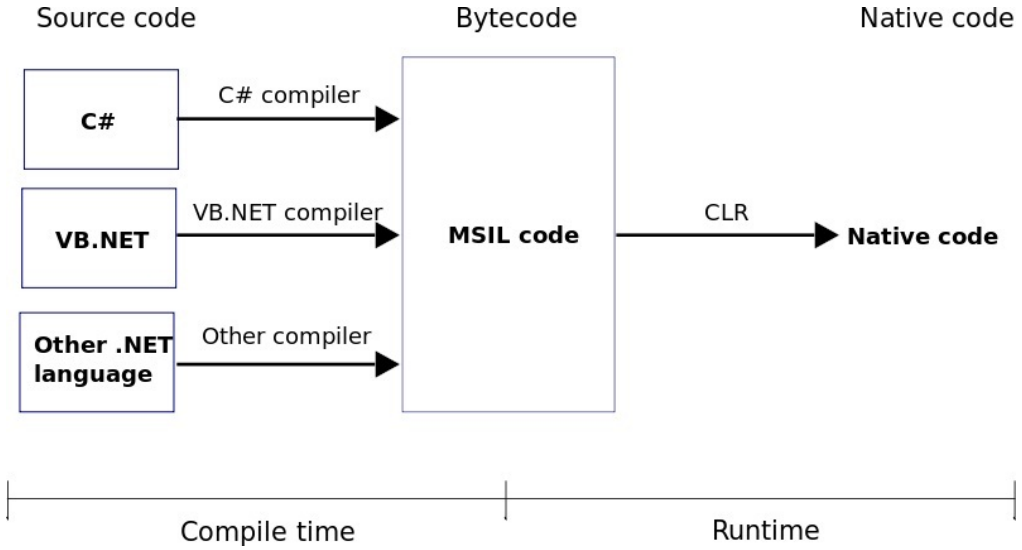


Figure : Common Language Runtime

Common Language Runtime (CLR) is a managed execution environment that is part of Microsoft's .NET framework. CLR manages the execution of programs written in different supported languages. The Common Language Runtime (CLR) manages the execution of programs written in different supported languages. CLR transforms source code into a byte code known as Common Intermediate Language (CIL). At run time, CLR handles the execution of the CIL (Common Intermediate Language) previously known as Microsoft Intermediate Language (MSIL) code.

Developers using the CLR write code in a language such as C# or VB.NET. At compile time, a .NET compiler converts such code into CIL code. At runtime, the CLR's just-in-time compiler converts the CIL code into native to the operating system. Alternatively, the CIL code can be compiled to native code in a separate step prior to runtime by using the Native Image Generator (NGEN). This speeds up all later runs of the software as the CIL-to-native compilation is to no longer necessary.

1.1.2.2 Common Language Infrastructure (CLI)

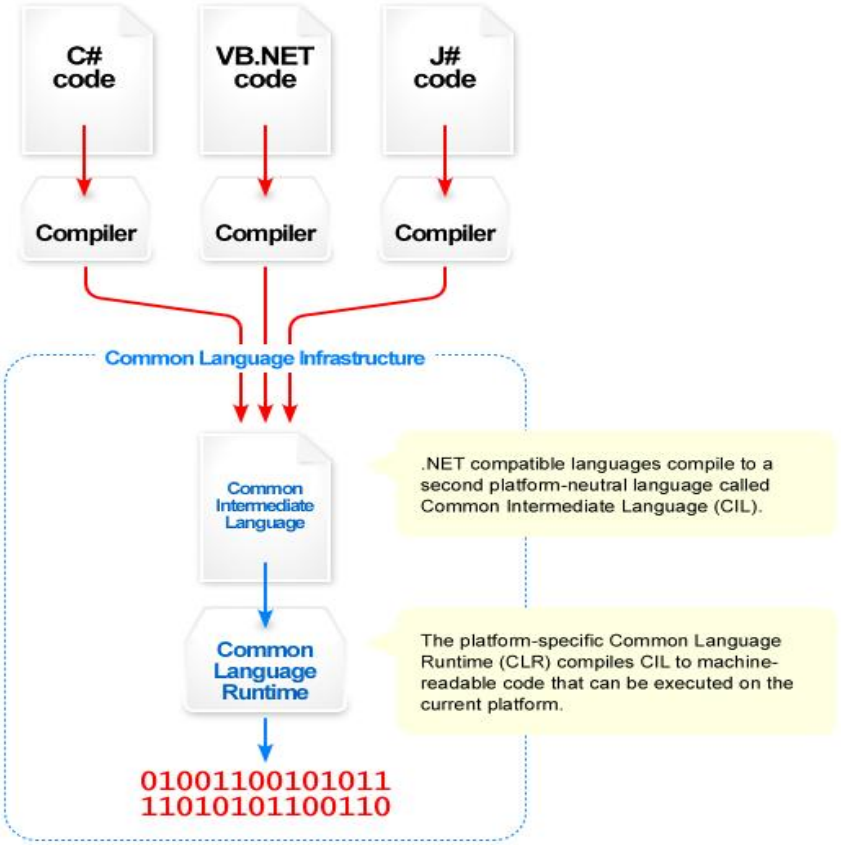


Figure : Common Language Infrastructure (CLI)

The Common Language Infrastructure (CLI) is an open specification developed by Microsoft that describes the executable code and runtime environment that form the core of the Microsoft.NET Framework. The specification defines an environment that allows multiple high-level languages to be used on different computer platforms without being rewritten for specific architectures.

The CLI specification is divided into four aspects:

1. The Common Type System (CTS)- A set of types and operations that are shared by many programming languages.

2. Metadata- Information about program structure is language-agnostic, so that it can be referenced between languages and tools, making it easy to work with code written in a language you are not using.

3. Common Language Specification (CLS)- A set of base rules which any language targeting the CLI should conform to in order to interoperate with other CLS-compliant languages.

4. Virtual Execution System (VES) - The VES loads and executes CLI-Compatible programs, using the metadata to combine separately generated pieces of code at runtime. All compatible languages compile to Common Intermediate Language (CIL), which is an intermediate language that is abstracted from the platform hardware. When the code is executed, the platform-specific VES will compile the CIL to the machine language according to the specific hardware.

1.1.2.3 Common Type System (CTS)

The Common Type System (CTS) is the core part if the CLR that takes care of cross language integration. The CTS offers a wide range of types and operations that are found in many programming languages.

As .NET Framework is language independent and support over 20 different programming languages, many programmers will write data types in their own programming languages.

For example, an integer variable in C# is written as int, whereas in Visual Basic it is written as integer. Therefore in .NET Framework you have single class called System.Int32 to interpret these variables. Similarly, for the Array List data type .NET Framework has a common type called System.Collections.Arraylist. In .NET Framework, System.Object is common base type from where all the other types are derived.

This system is called Common Type System. The types in .NET Framework are the base on which .NET applications, components, and controls are built, Common Type System in .NET Framework defines how data types are going to be declared and managed in runtime.

The Common Type System supports two general categories of types:

- **Value types** include simple types (char, int, float, etc), enum types and struct types.
- **Reference types** include class types, interface types, delegate types and array types.

1. Value types:

Value types directly contain their data and instances of values types are either allocated on the stack or allocated inline in a structure. Value types can be built-in (implemented by the runtime), user-defined, or enumerations. For a list of built-in value types, see the .NET Framework Class Library.

2. Reference types:

Reference types store a reference to the value's memory address, and are allocated on the heap. Reference types can be self-describing types, pointer types, or interface types. The type of a reference type can be determined from values of self-describing types. Self-describing types are further split into arrays and class types. The class types are user-defined classes, boxed value types and delegates.

1.1.2.4 Common Language Specification (CLS)

The Common Language Specification defines a set of rules compilers must follow for types (both value and reference) so that objects of types written in different languages can interact.

Note: System. Object is CLS compliant.

Rules for CLS

The most important rules, and which apply to public and protected members are

- All types appearing in a method prototype must be CLS-compliant.
- Array elements must have a CLS-compliant element type. Arrays must also be 0- indexed.
- A CLS compliant class must inherit from a CLs-compliant class.
- Although method names in CLS-compliant classes are not case-sensitive, no two method names can be different only in case of the letters in their name.
- Enumerations must be of type Int16, Int32 or Int64. Enumerations of other types are not compliant.
- Also there are several naming guidelines, but they are not mandatory.

1.1.2.5 Garbage Collection

- Garbage collection is the process of detecting when objects are no longer in use and automatically destroying those objects, thus freeing memory.
- Garbage collection (GC) is a form of **automatic memory management.** After certain duration of time garbage collector, or just collector, attempts to reclaim garbage, or memory occupied by objects that are no longer in use by the program
- Garbage collection is also portrayed as the opposite of **manual memory management**, which requires the programmer to specify which objects to deallocate and return to the memory system.

- Garbage collection is not a new concept. It has been used in other languages for quite some time. In fact, Java has a garbage collection system in place. Other languages, such as C++, do not have garbage collection. C++ developers themselves are required to take care of destruction of objects and the freeing of memory. This results in a number of problems, such as memory leaks. If the developer forgets to free objects from the application, memory allocation of the application grows, sometimes substantially. Also, freeing objects too early cause's application bugs to crop up these kinds of errors are, in most cases, quite difficult to track down.
- In .NET, this new garbage collector works that you as a developer are no longer required to monitor your code for unneeded objects and destroy them. The garbage collector will take care of all this for you.

.1.1.2.6 ASSEMBLY

- An assembly is a collection of types and resources that forms a logical unit of functionality. Assemblies can be thought of as the building blocks of your applications.
- Without an associated assembly, code will not be able to compile from IL. When you are using the JIT compiler to compile your code from managed code to machine code, the JIT compiler will look for the IL code that is stored in portable executable (PE) file along with the associated assembly manifest.
- Every time you build a Web From or Windows Form application in .NET, you are actually building an assembly. Every one of these applications will contain at least one assembly.
- When an application is started in .NET, the application will look for an assembly in the installation folder. Assemblies that are stored in a local installation folder are referred to as private assemblies. If the application cannot find the assembly within the installation folder, the application will run to the Global Assembly Cache (GAV)/Public assembly for the assembly.

- The GAC is place where you can store assemblies that you want to share across applications. You can find the assemblies that are stored in the GAC in the WINNT\ASSEMBLY folder in your local disk drive.

Structure of an Assembly

- Assemblies contain code that is executed by the Common Language Runtime. The great thing about assemblies is that they are self-describing. All the details about the assembly are stored within the assembly itself.
- In the Windows DNA world (previous versions), COM stored all its self- describing data within the server's registry, and so installing (as well as uninstalling) COM components was a tedious task. .NET assembly stores this information within itself

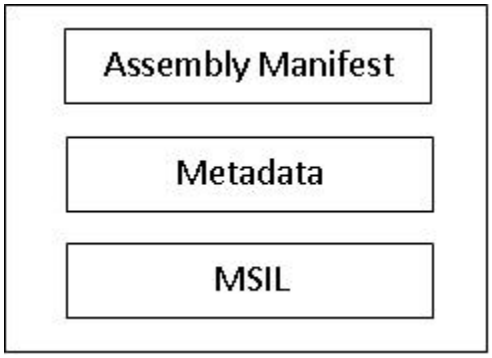


Figure: The structure of an assembly

- The assembly manifest is where the details of the assembly are stored. The Assembly is stored within the DLL or EXE itself.
- The assembly manifest also stores the version number of the assembly to ensure that the application always uses the correct version. When you are going to have multiple versions of an assembly on the machine, it is important to label them carefully so that the CLR knows which one to use. Version numbers in assemblies are constructed in the following manner:

Syntax
<major version>.<minor version>.<build number>.<revision>

The assembly manifest contains these items:

- The assembly name and version
- The culture or language the assembly supports (not required in all assemblies)
- The public key for any strong name assigned to the assembly (not required in all Assemblies)
- A list of files in the assembly with hash information
- Information on exported types
- Information on referenced assemblies

- The metadata contains information on the types that are exposed by the assembly such as security permission information, class and interface information and other assembly information.
- MSIL stands for Microsoft Intermediate Language. We can call it as Intermediate Language (IL) or Common Intermediate Language (CIL). During the compile time, the compiler convert the source code into Microsoft Intermediate Language (MSIL) .Microsoft Intermediate Language (MSIL) is a CPU-independent set of instructions that can be efficiently converted to the native code. During the runtime the Common (CLR)'s Just In Time (JIT) compiler converts the Microsoft Intermediate Language (MSIL) code into native code to the Operating System.

Types of assembly:

- Private assembly
- Public assembly

1.1.2.7 Namespaces

The .NET Framework is made up to hundreds of classes. Many of the applications that you build in .NET are going to take advantage of these classes in one way or another. Because the number of classes is so large and you will need to get at them in a logical fashion, the .NET Framework organizes these classes into a class structure called a namespace. There are a number of namespaces, and they are organized in an understandable and straightforward way.

You can import a namespace into your application in the following manner:

VB

```
Imports System.Data
```

If you are going to import more than one namespace into your application, do as shown in the following code:

VB

```
Imports System.Data
Imports System.Data.Oledb
```

By importing a namespace into your application, you no longer need to fully qualify the class. For example, if you do not import the namespace into your application, you must write out the full class name.

Example

```
Myconnection=”Initial Catalog=Northwind;Data Source=localhost;
Integrated Security=SSPI;”
Dim conn As New System.Data.SqlConnection (myconnection)
```

If you do import the System.Data.SQLClient namespace into your application or to the page where you need it, you can refer to the class quite simply when building your .NET applications in the .NET Framework, a number of namespaces are already automatically imported into your application for use throughout. An ASP.NET Web application automatically imports the following namespaces:

Namespaces
System
System.Data
System.Drawing
System.Web

1.1.2.8 THE BASE CLASS LIBRARIES

The base class libraries (BCL) are a set of classes, value types and interfaces that give you access to specific developer utilities and various system functions. Table gives a brief description of some of the classes available with the .NET Framework.

Namespaces Definitions

Namespace	Description
System	Provides base data types and almost 100 Classes that deal with situations like exception handling, mathematical functions and garbage collection.
System.CodeDom	Provides the classes needed to produce source
System. Collections	Provides access to collection classes such as lists, queues, bit arrays, hash tables and dictionaries.
System.ComponentModel	Provides classes that are used to implement runtime and design time and implement runtime and design-time behavior.
System. Data	Provides classes that allow data access and manipulation to SQL Server and OleDb data
System.IO	Provides classes that allow access to file and stream control and manipulation.

SERVICES

The CLR allows programmers to ignore many details of the specific CPU that will execute the program. It also provides other important services, including the following:

- Memory management
- Thread management
- Exception handling
- Garbage collection
- Security

CLR is like a JVM of java.

When we compile our .NET program it converts it in IL or MSIL (Microsoft

Intermediate Language). Then CLR convert it in Native code is called JIT (Just-In-Time Compilation) CLR directly communicate with OS (Operating System)

1.2 Basics of ASP.NET

1.2.1 Features of ASP.NET

1. Interoperability

Computer systems commonly require interaction between new and older applications, the .NET Framework provides means to access functionality that is implemented in programs that execute outside the .NET environment.

2. In-built memory management

Visual Studio .NET provides a fully object oriented environment. Visual Studio .NET supports handling of memory on its own. The garbage collector takes responsibility for free up unused objects for regular intervals.

3. Multi Language and Multi Device Support

Visual Studio .NET supports multi languages. The beauty of multi-language support lies in the fact that even though the syntax of each language is different, the basic environment of developing software is same. Visual Studio .NET supports Multi Device. We can create Mobile or PDA etc. device supported software.

4. Common Language Runtime

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

5. Base Class Library

The Base Class Library (BCL), part of the Framework class library (FCL), is a library of functionality available to all languages using the .NET Framework. The BCL provides classes which encapsulate a number ofcommon functions, including file reading and writing, graphic rendering, database interaction, XML document manipulation and so on.

6. Simplified deployment

The .NET Framework includes design features and tools that 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.

7. Security

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

8. Language Independence

One of the principal design features of Microsoft .NET Framework is that it is language independent. It consists of Common Type System, also Known as CTS. It is a kind of standard defining all the data types and programming constructs supported by the CLR and the Conforming to the Common Language Infrastructure (CLI) standards. This feature makes the .NET Framework as language Independent Framework.

9. Portability

This principal design feature of Microsoft .NET Framework permits it to be platform agnostic, and therefore it is compatible with different platforms. This feature of Microsoft .NET Framework also makes a way for third parties to develop compatible implementations of this Framework and its languages on platforms other than Microsoft in which they might or might not be interacting with each other.

1.2.2 Difference between ASP and ASP.NET

Classic ASP

- ASP has limited oops support and no built in support for xml.
- Limited development and debugging tool is available. Very difficult to debug code.
- In ASP only two languages are available for scripting, like VB script and Jscript/Javascript.
- ASP is not well structured. ASP has mixed html and server side scripting.
- Error handling system is very poor in ASP.
- In ASP you must place all directives on the first line of a page within the same delimiting block. For example:

Example
<%LANGUAGE="VBSCRIPT" CODEPAGE="932"%>

- ASP is Interpreted language based on scripting language like JavaScript and VB script.
- In classic ASP if you want to update any code then need to often stop and restart the server.
- ASP is not having inbuilt facility for validation of controls.
- Client and server side validations both were headache for developers.

ASP.NET

- ASP .NET is fully Object Oriented Programming.
- ASP.NET has full xml support for easy data exchange.
- Different types of tools and compilers available. Mostly as a development framework visual studio users more.
- Very easy to debug code.
- ASP.NET we can use C# or VB.NET as server side coding.
- In ASP.NET business logic and codes and Design phrase all are separate. So that it's easy for user to make changes in code.
- Error handling is very good.
- In ASP.NET, you are now required to place the Language directive with a

Page directive
<%@Page Language="VB" CodePage="932"%> <%@QutputCache Duration="60" VaryByParam="none" %>

- State management support.

1.2.3 Web Application & Web Pages

Web Application

A web application or web app is any application software that runs in a web browser and is created in a browser-supported programming (such as the

combination of JavaScript, HTML and CSS) and relies on a common web browser to render the application.

Web Page

A web page (or webpage) is a web document that is suitable for the World Wide Web and the web browser. A web browser displays a web page on a monitor or mobile device. The web page is what displays, but the term also refers to a computer file, usually written in HTML or comparable markup language, whose main distinction is to provide hypertext that will navigate to other web pages via links. Web browsers coordinate web resources centered on the written web page, such as style sheets, scripts and images, to present the web page.

1.2.4 Client/ Server Architecture

- Web applications (ASP.Net) use client/server architecture.
- The Web application resides on a server and responds to requests from multiple clients over the Internet, as shown in this figure.

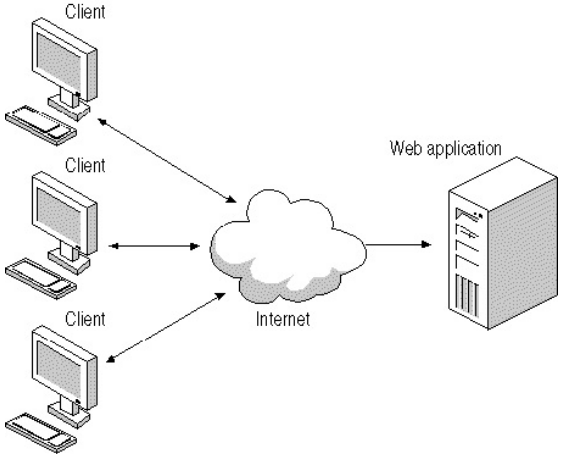


Figure : Client Server Architecture

- On the client side, the Web application is hosted by a browser.
- The application's user interface takes the form of Hypertext Markup Language (HTML) pages that are interpreted and displayed by the client's browser.
- On the server side, the Web application runs under Microsoft Internet

Information Services (IIS).

- IIS manages the application, passes requests from clients to the application, and returns the application's responses to the client.
- These requests and responses are passed across the Internet using Hypertext Transport Protocol (HTTP).
- The figure below shows how the client and server interact over the Internet.

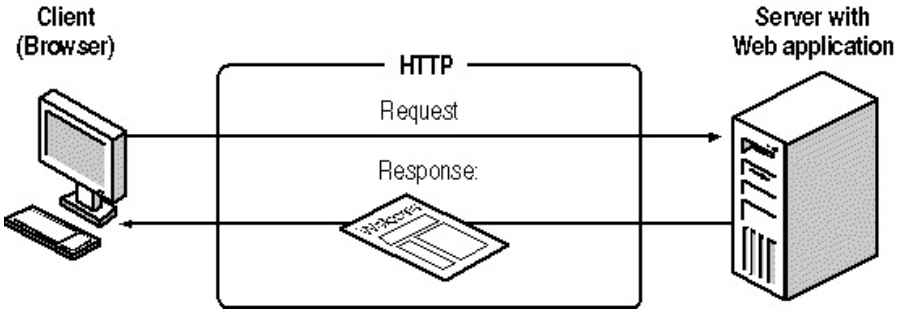


Figure: Request Response Object

- The Web application composes responses to requests from resources found on the server.
- These resources include the executable code running on the server (what we think of as the "application" in "traditional" programming), Web forms, HTML pages, image files, and other media that make up the content of the application.
- Web applications are much like traditional Web sites, except that the content presented to the user is actually composed dynamically by executable, rather than being served from a static page stored on the server.
- The figure below shows how a Web application composes the HTML to be returned to a user.

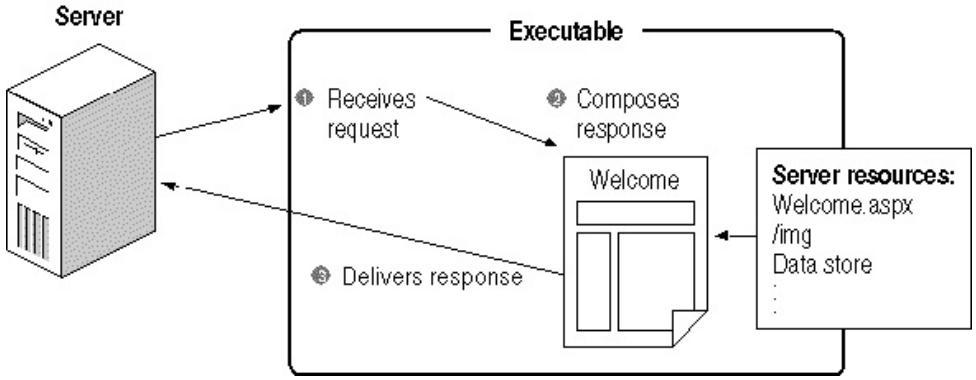


Figure: Web form Processing

1.2.4 Parts of Websites

1. HTML
2. XHTML
3. CSS
4. Client Side coding and Server Side coding

1. HTML

- HTML stands for Hypertext Markup Language
- It is not a Programming Language, it is a markup language used to create web pages.
- HTML is written in the form of HTML elements consisting of markup tags enclosed in angle brackets (like <html>).
- A web browser can read HTML files and compose them into visible or audible web pages
- With the help of HTML we can develop static pages.

2. XHTML

- XHTML stands for Extensible Hypertext Markup Language
- XHTML is almost identical to HTML 4.01
- XHTML is a stricter and cleaner version of HTML
- XHTML is HTML defined as an XML application
- XHTML is supported by all major browsers

3. CSS

- Cascading Style Sheets (CSS) is a style sheet language used for describing the look and formatting of a document written in a markup language.
- It's a separation of document content from document presentation
- Instead of assigning attributes to each element on your page individually, you can create a general rule that that applies attributes whenever a web browser encounters an instance of an element or an element that is assigned to a certain style sheet.

4. Client Side Coding and Server Side Coding

- Client-side scripting generally refers to the class of computer programs on the web that are executed client-side, by the user's web browser
- Client-side scripts are often embedded within an HTML document (hence known as an "embedded script"), but they may also be contained in a separate file, which is referenced by the document (or documents) that use it (hence known as an "external script"). Upon request, the necessary files are sent to the user's computer by the web server (or servers) on which they reside. The user's web browser executes the script, and then displays the document, including any visible output from the script. Client-side scripts may also contain instructions for the browser to follow in response to certain user actions, (e.g., clicking a button). Often, these instructions can be followed without further communication with the server.
- Server-side scripts, written in languages such as Perl, PHP, and server-side VBScript, are executed by the web server when the user requests a document. They produce output in a format understandable by web browsers (usually HTML), which is then sent to the user's computer. The user cannot see the script's source code (unless the author publishes the code separately), and may not even be aware that a script was executed. Documents produced by server-side scripts may, in turn, contain client-side scripts.
- Client-side scripts have greater access to the information and functions available on the user's browser, whereas server-side scripts have greater access to the information and functions available on the server.

1.3 Creating First Web Application in ASP.NET

1.3.1 Introduction to Visual Studio 2008

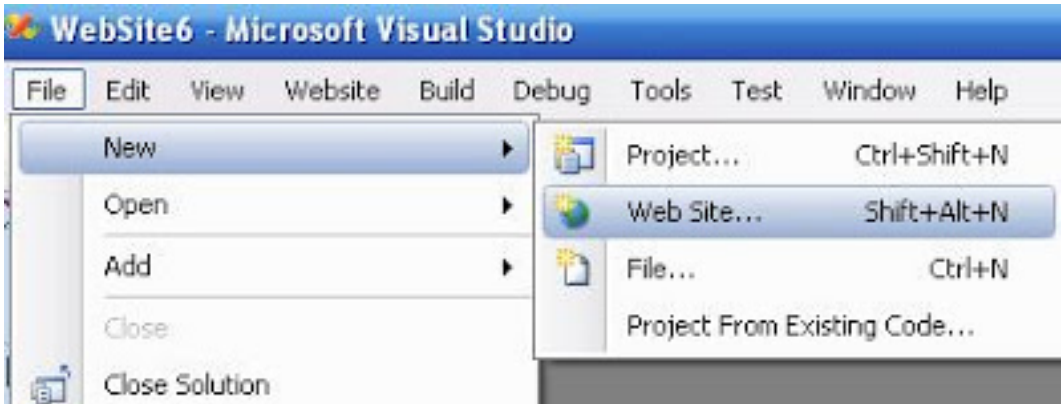
Microsoft Visual Studio is an integrated development environment (IDE) from Microsoft. It is used to develop computer programs for Microsoft Windows super family of operating systems. We can develop Web Applications as well as Windows Applications.

1.3.2 Creating a New Web Project

Step 1.Open your Microsoft Visual Studio 2008.

Click on Start menu->All Programs->Microsoft Visual Studio 2008

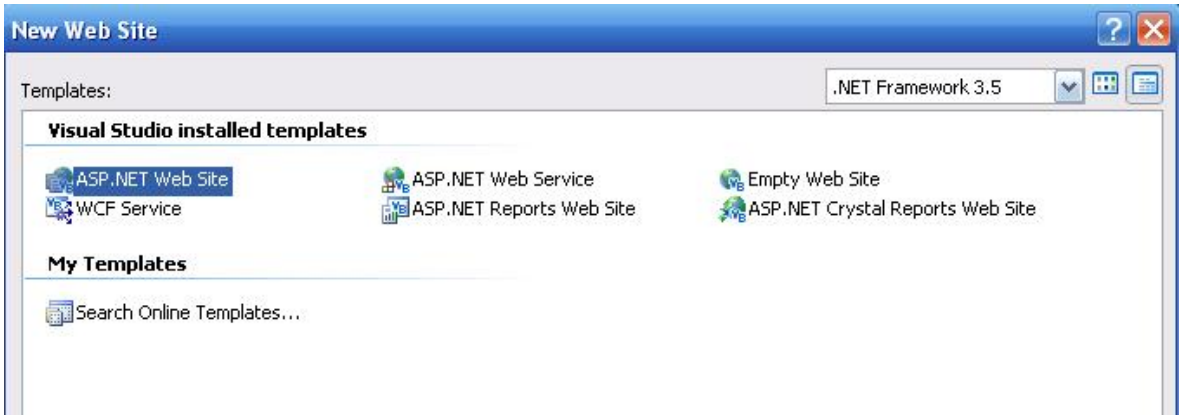
Step 2. In the File Menu, Click on New and then Website



Creating a new website

Step 3.

(a) Select Asp.Net Website from the window



(b) Location: Select File System for storing the File on the hard drive and set location where you want to save your website project like C:,D:,E: etc and name your website project.

(c) Language: Select Visual Basic language.

(d) Click on ok button to proceed.

Step 4.

Now we will see few files on the left side in Solution Explorer panel. There are website related files like **web.config, default.aspx, default.aspx.vb and App_data.**

Figure : Solution Explorer

As you see in image and this representation of files shows that your project has been created. Now we will check or run our project but before run, we have add some message or code in code behind file (default.aspx.vb).

Step 5. Double click (or simply press F7 function key to open code behind file) on default.aspx.vb file in solution explorer panel then file will be open.

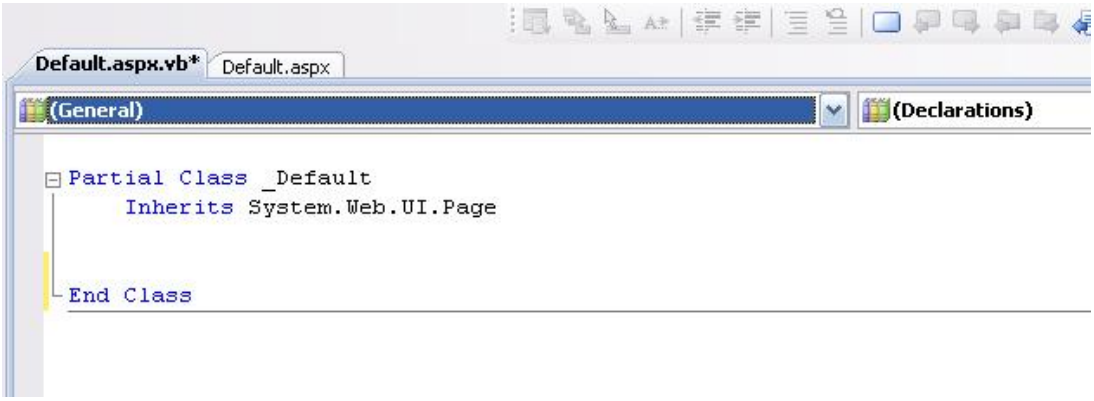
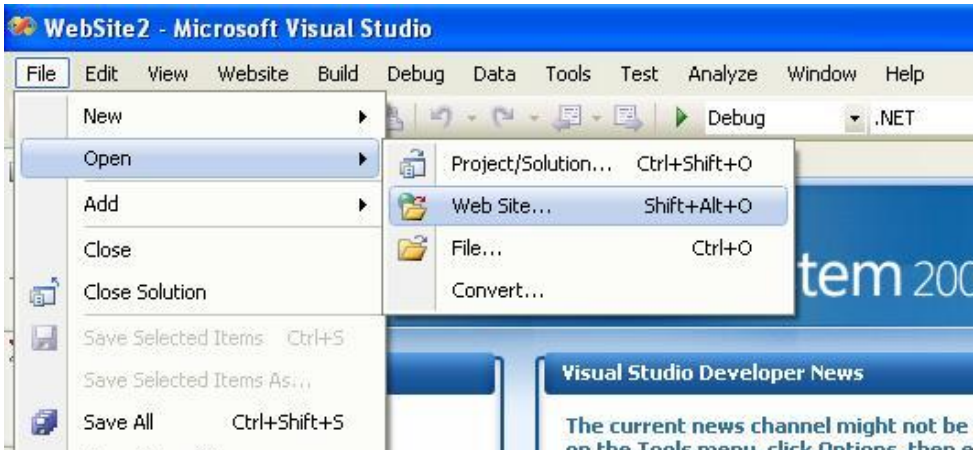


Figure : Code Window

1.3.3. Opening an Existing Web Project

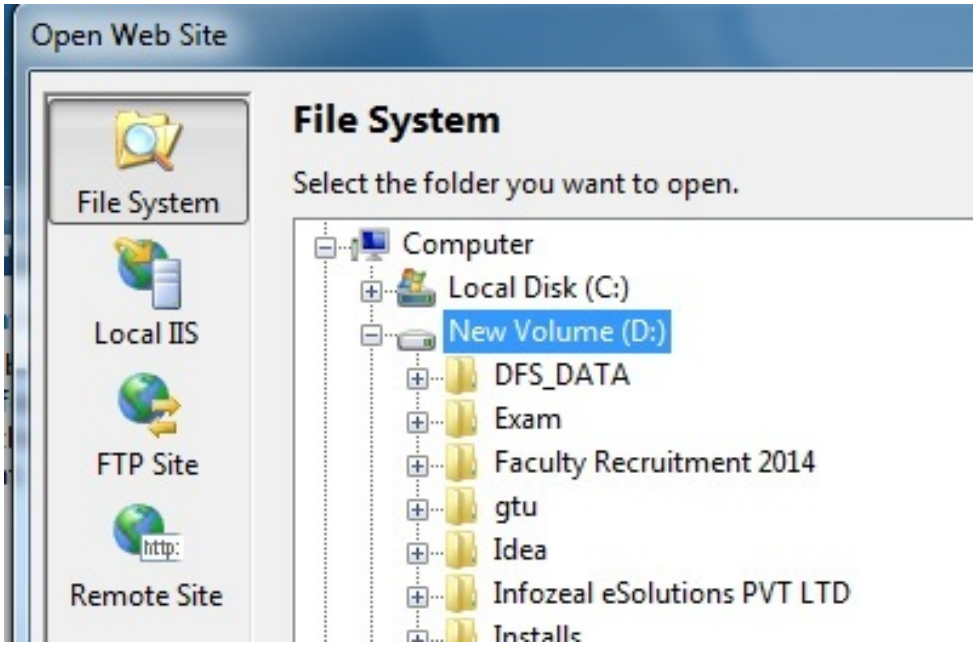
Step 1 Open the Visual Studio 2008 solution.

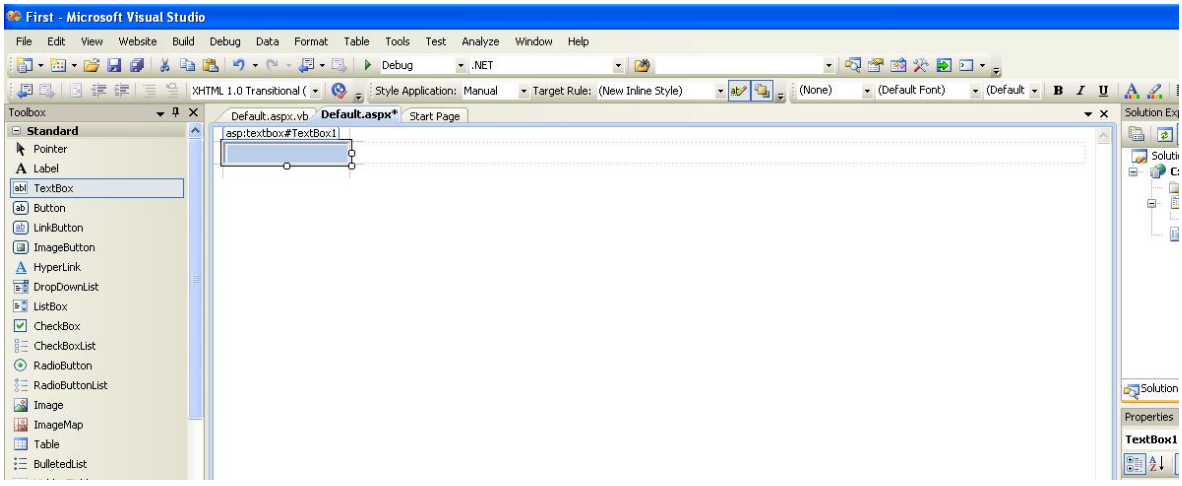
In the File menu, click on Open Menu, and then click Web Site.



Opening an Existing Website

Step2. Select the project folder that you want to open, and then click **Open**.

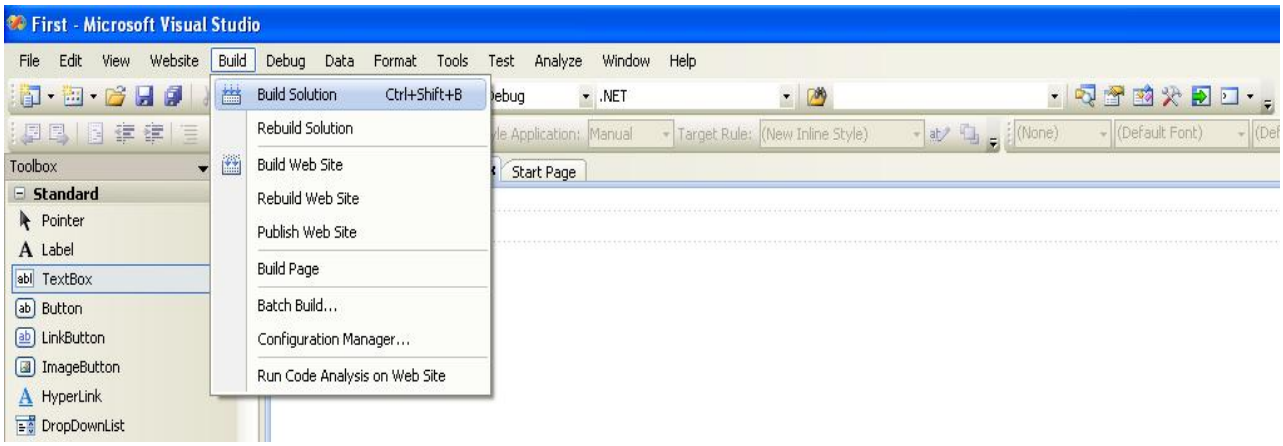




Existing Website

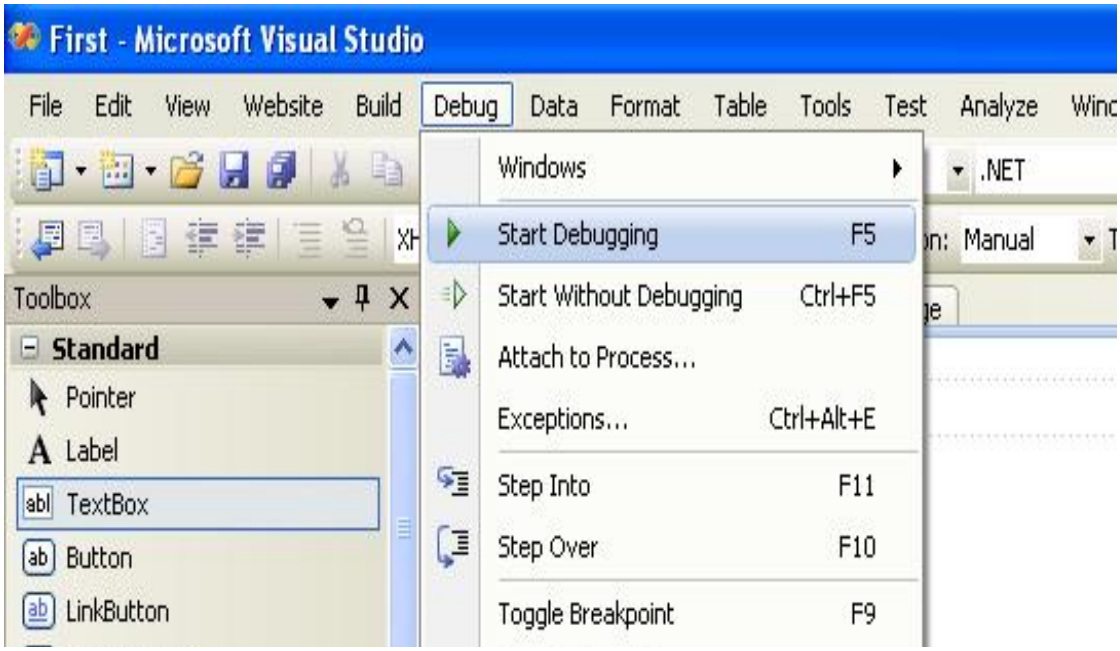
1.3.4. Building Website

Step 1. In the Build menu, click on Build Solution or Press Ctrl+Shift+B Shortcut Key.



Step 2. To debug the project

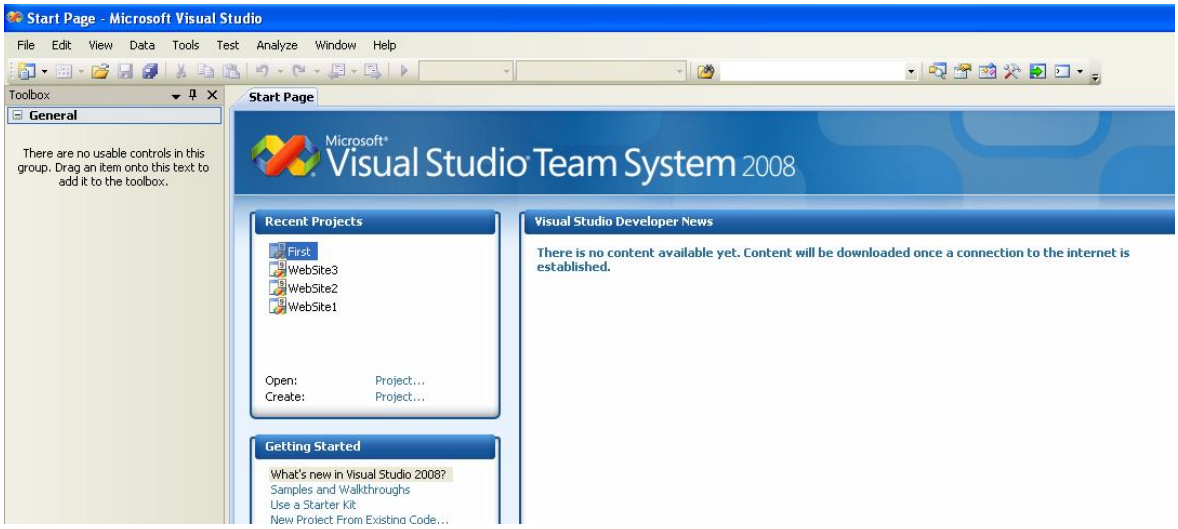
In the Debug menu, click on Start Debugging or Press F5 Key as shown below.



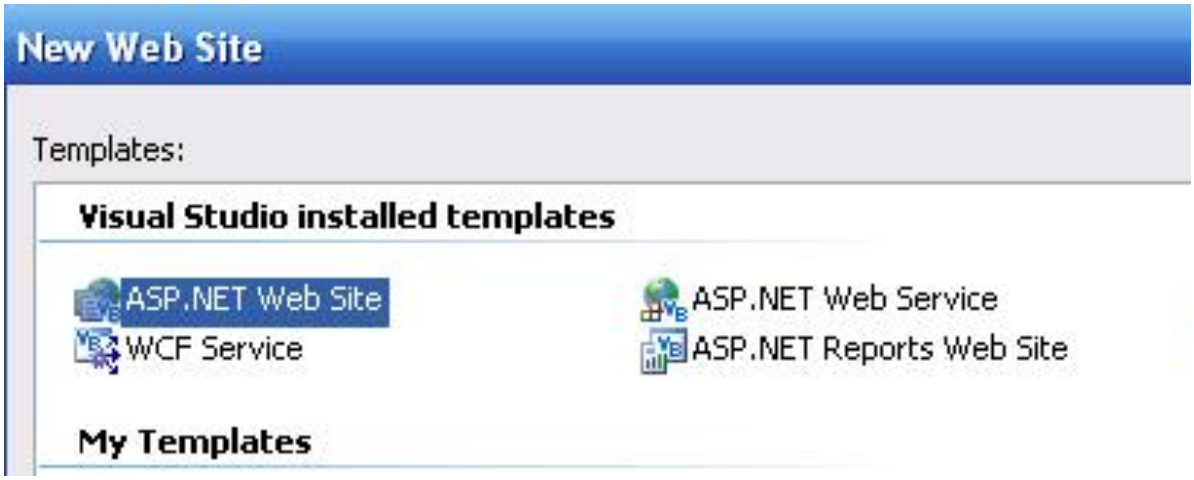
Debugging

1.3.5. Set up of work environment, start page, the menu system, toolbar, the new project dialog box, graphical designer, code designer.

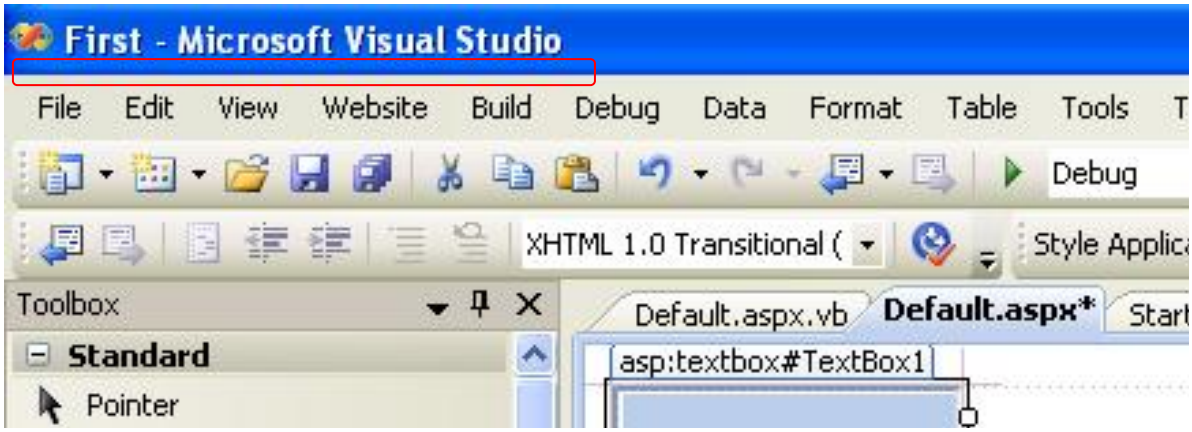
- **Start Page**



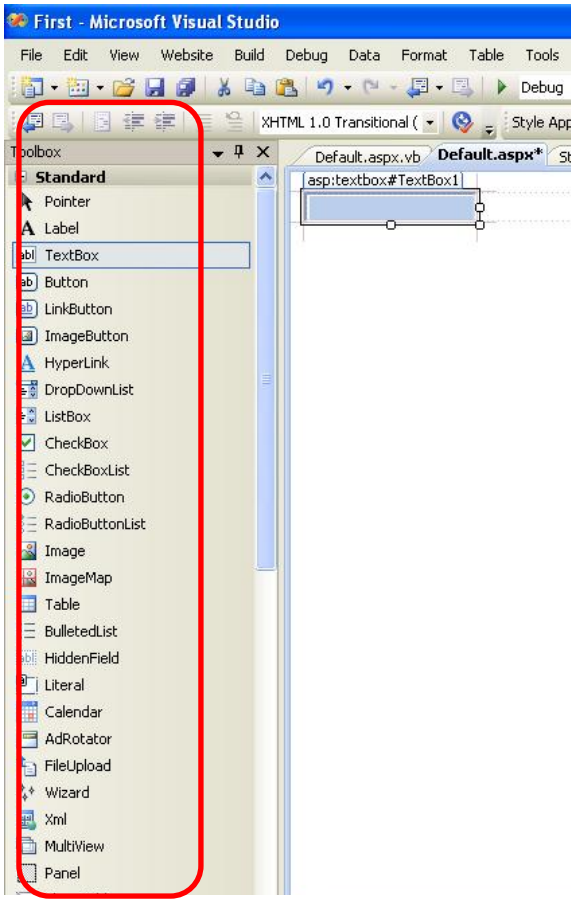
- **New Project Dialog box**



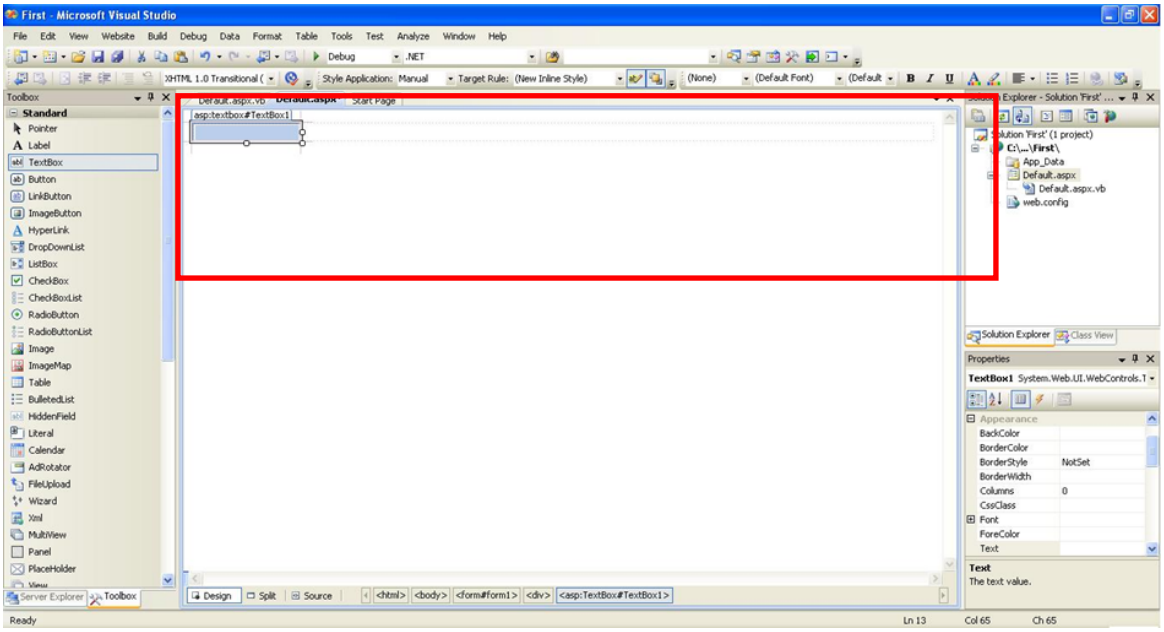
- **Menu bar**


First - Microsoft Visual Studio
File Edit View Website Build Debug Data Format Table Tools
Debug
XHTML 1.0 Transitional (
Toolbox
Standard
Pointer
Default.aspx.vb
Default.aspx*
asp:textbox#TextBox1

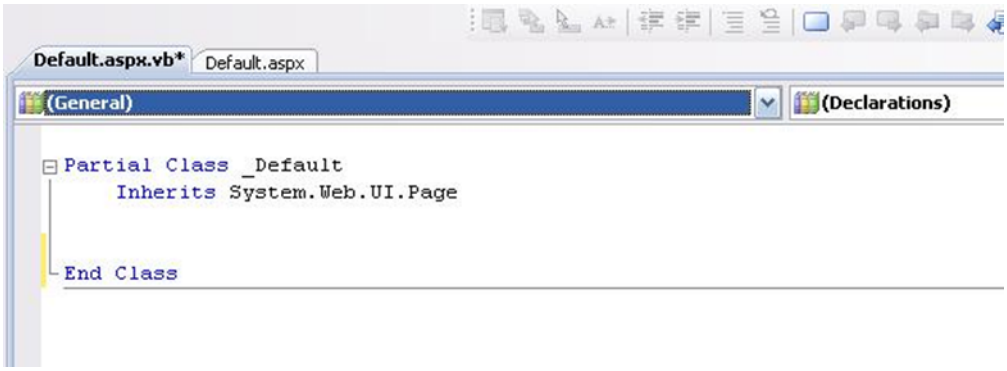
- **Toolbar**


First - Microsoft Visual Studio
File Edit View Website Build Debug Data Format Table Tools
Debug
XHTML 1.0 Transitional (
Style App
Toolbox
Standard
Pointer
Label
TextBox
Button
LinkButton
ImageButton
HyperLink
DropDownList
ListBox
CheckBox
CheckBoxList
RadioButton
RadioButtonList
Image
ImageMap
Table
BulletedList
HiddenField
Literal
Calendar
AdRotator
FileUpload
Wizard
Xml
MultiView
Panel
Default.aspx.vb
Default.aspx*
asp:textbox#TextBox1

➢ Graphical Designer Window



➢ Code Designer Window



Exercise

1. What is ASP? Explain in brief.
2. Explain client-server architecture with diagram.
3. Explain the disadvantages of ASP.
4. What is ASP.NET? List & explain the features & advantages of .net.
5. Explain .net framework architecture with diagram.
6. Write a brief note on the following : 1) CLR 2) CLS 3) CTS
7. Write a short note on assemblies.
8. With the help of proper diagram, explain the work mechanism of ASP.NET.

********************************END OF CHAPTER ********************************