TOPIC 2 → Introduction to ASP.NET

LEARNING OUTCOME

By the end of this topic, students will be able to:

- Describe ASP.NET
- Create ASP.NET Web Forms Model
- Integrate ASP.NET Component Model
- Identify Components of .Net Framework
- Explain the Environment Setup and Installation Steps
- Describe the Visual Studio IDE
- Work with Views and Windows
- Add Folders and Files to Website Project
- Build and Run an ASP.NET Project
- Describe ASP.NET Life Cycle

Introduction

ASP.NET is a web application framework developed and marketed by Microsoft to allow programmers to build dynamic web sites. It allows you to use a full featured programming language such as C# or VB.NET to build web applications easily.

ASP.NET is a web development platform, which provides a programming model, a comprehensive software infrastructure and various services required to build up robust web applications for PC, as well as mobile devices.

ASP.NET works on top of the HTTP protocol, and uses the HTTP commands and policies to set a browser-to-server bilateral communication and cooperation.

ASP.NET is a part of Microsoft .Net platform. ASP.NET applications are compiled codes, written using the extensible and reusable components or objects present in .Net framework. These codes can use the entire hierarchy of classes in .Net framework.

The ASP.NET application codes can be written in any of the following languages:

- C#
- Visual Basic.Net
- Jscript
- J#

ASP.NET is used to produce interactive, data-driven web applications over the internet. It consists of a large number of controls such as text boxes, buttons, and labels for assembling, configuring, and manipulating code to create HTML pages.

ASP.NET Web Forms Model
ASP.NET web forms extend the event-driven model of interaction to the web applications. The browser submits a web form to the web server and the server returns a full markup page or HTML page in response.

All client side user activities are forwarded to the server for stateful processing. The server processes the output of the client actions and triggers the reactions.
Now, HTTP is a stateless protocol. ASP.NET framework helps in storing the information regarding the state of the application, which consists of:

- Page state
- Session state

The page state is the state of the client, i.e., the content of various input fields in the web form. The session state is the collective information obtained from various pages the user visited and worked with, i.e., the overall session state. To clear the concept, let us take an example of a shopping cart.

User adds items to a shopping cart. Items are selected from a page, say the items page, and the total collected items and price are shown on a different page, say the cart page. Only HTTP cannot keep track of all the information coming from various pages. ASP.NET session state and server side infrastructure keeps track of the information collected globally over a session.

The ASP.NET runtime carries the page state to and from the server across page requests while generating ASP.NET runtime codes, and incorporates the state of the server side components in hidden fields.
This way, the server becomes aware of the overall application state and operates in a two-tiered connected way.

The ASP.NET Component Model
The ASP.NET component model provides various building blocks of ASP.NET pages. Basically it is an object model, which describes:

- Server side counterparts of almost all HTML elements or tags, such as <form> and <input>.
- Server controls, which help in developing complex user-interface. For example, the Calendar control or the Gridview control.

ASP.NET is a technology, which works on the .Net framework that contains all web-related functionalities. The .Net framework is made of an object-oriented hierarchy. An ASP.NET web application is made of pages. When a user requests an ASP.NET page, the IIS delegates the processing of the page to the ASP.NET runtime system.

The ASP.NET runtime transforms the .aspx page into an instance of a class, which inherits from the base class page of the .Net framework. Therefore, each ASP.NET page is an object and all its components i.e., the server-side controls are also objects.

Components of .Net Framework 3.5
Before going to the next session on Visual Studio.Net, let us go through at the various components of the .Net framework 3.5. The following table describes the components of the .Net framework 3.5 and the job they perform:

Components	Description
Common Language Runtime or CLR	It performs memory management, exception handling, debugging, security checking, thread execution, code execution, code safety, verification, and compilation. The code that is directly managed by the CLR is called the managed code. When the managed code is compiled, the compiler converts the source code into a CPU

	independent intermediate language (IL) code. A Just In Time (JIT) compiler compiles the IL code into native code, which is CPU specific.
.Net Framework Class Library	It contains a huge library of reusable classes, interfaces, structures, and enumerated values, which are collectively called types.
Common Language Specification	It contains the specifications for the .Net supported languages and implementation of language integration.
Common Type System	It provides guidelines for declaring, using, and managing types at runtime, and cross-language communication.
Metadata and Assemblies	Metadata is the binary information describing the program, which is either stored in a portable executable file (PE) or in the memory. Assembly is a logical unit consisting of the assembly manifest, type metadata, IL code, and a set of resources like image files.
Windows Forms	Windows Forms contain the graphical representation of any window displayed in the application.
ASP.NET and ASP.NET AJAX	ASP.NET is the web development model and AJAX is an extension of ASP.NET for developing and implementing AJAX functionality. ASP.NET AJAX contains the components that allow the developer to update data on a website without a complete reload of the page.
ADO.NET	It is the technology used for working with data and databases. It provides access to data sources like SQL server, OLE DB, XML etc. The ADO.NET allows connection to data sources for retrieving, manipulating, and updating data.
Windows Workflow Foundation (WF)	It helps in building workflow-based applications in Windows. It contains activities, workflow runtime, workflow designer, and a rules engine.
Windows Presentation Foundation	It provides a separation between the user interface and the business logic. It helps in developing visually stunning interfaces using documents, media, two and three dimensional graphics, animations, and more.
Windows Communication Foundation (WCF)	It is the technology used for building and executing connected systems.
Windows CardSpace	It provides safety for accessing resources and sharing personal information on the internet.
LINQ	It imparts data querying capabilities to .Net languages using a syntax which is similar to the tradition query language SQL.

Environment Setup

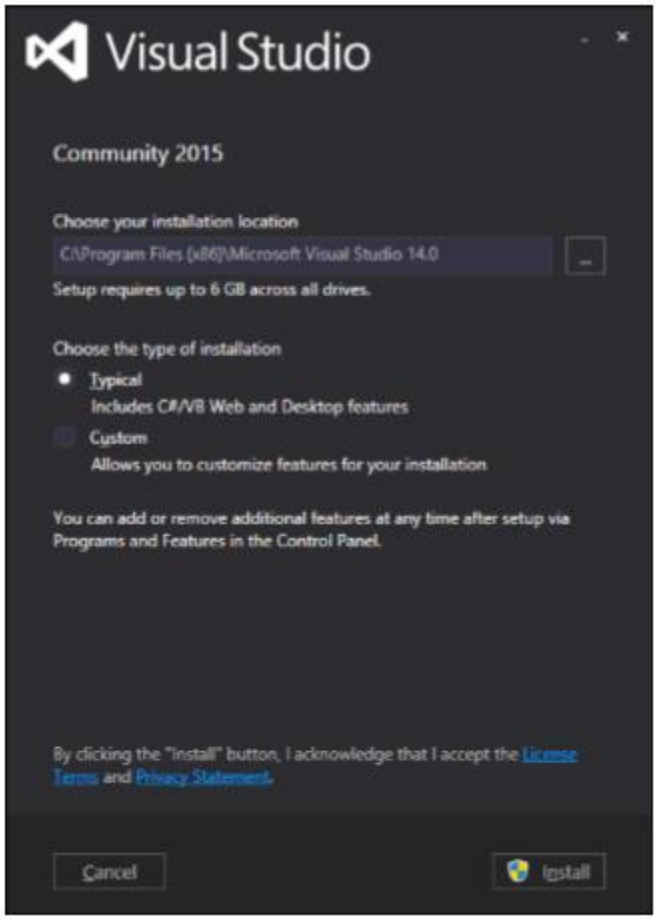
ASP.NET provides an abstraction layer on top of HTTP on which the web applications are built. It provides high-level entities such as classes and components within an object-oriented paradigm. The key development tool for building ASP.NET applications and front ends is Visual Studio. In this class, we work with Visual Studio 2015.

Visual Studio is an integrated development environment for writing, compiling, and debugging the code. It provides a complete set of development tools for building ASP.NET web applications, web services, desktop applications, and mobile applications.

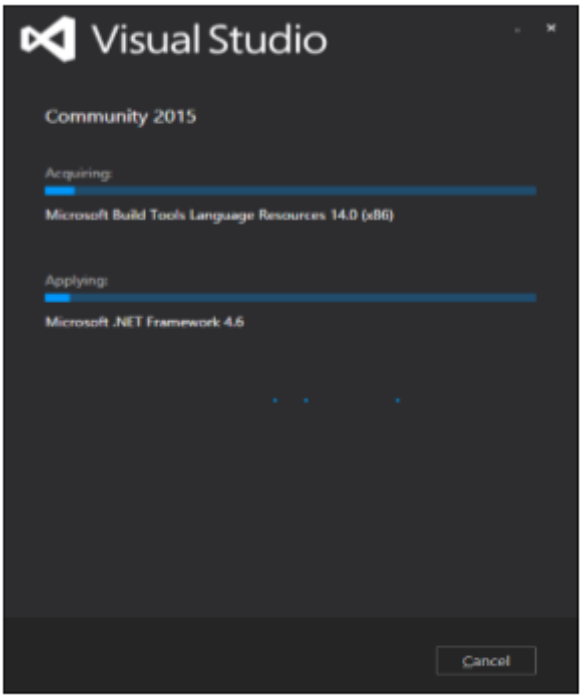
Installation

Microsoft provides a free version of visual studio which also contains SQL Server and it can be downloaded from www.visualstudio.com

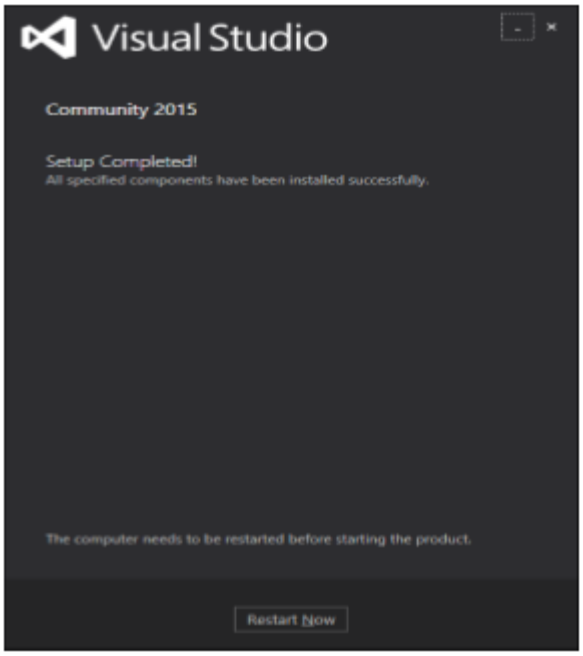
Step 1 − Once downloading is complete, run the installer. The following dialog will be displayed.



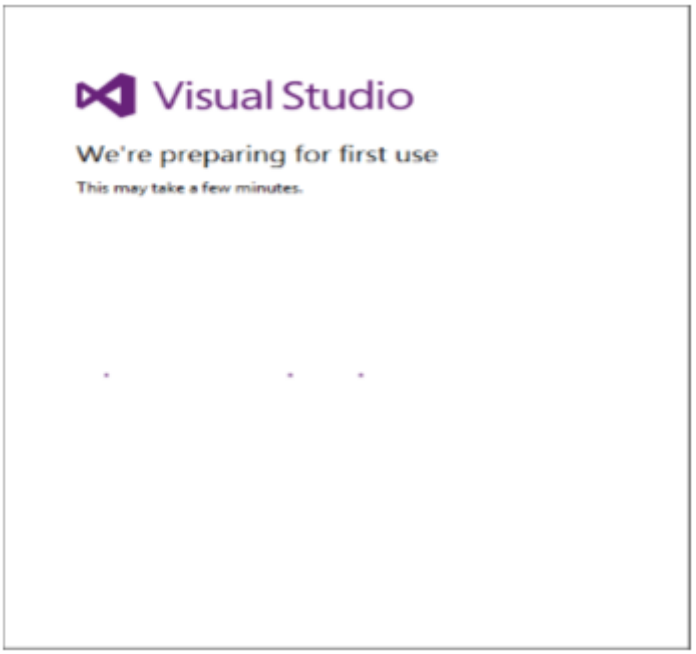
Step 2 − Click on the Install button and it will start the installation process.



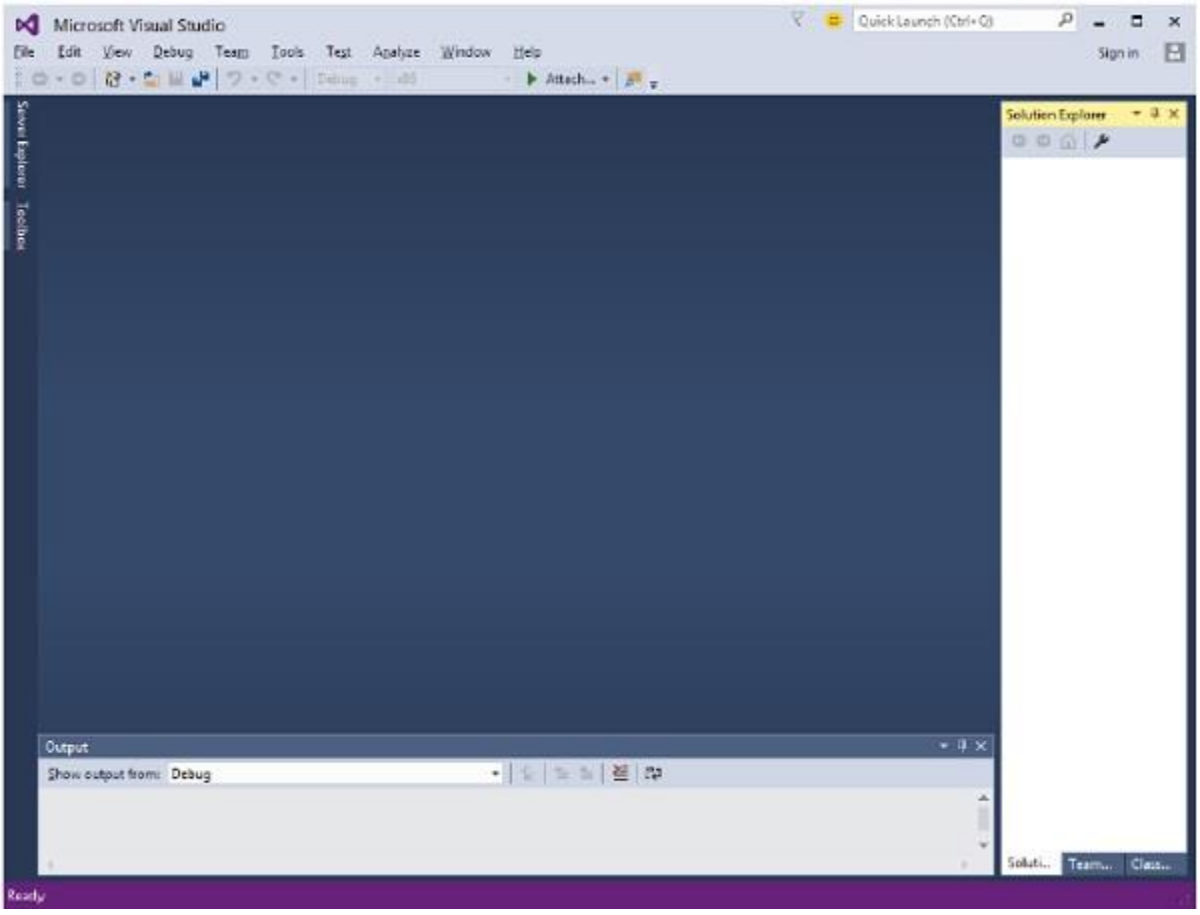
Step 3 − Once the installation process is completed successfully, you will see the following dialog. Close this dialog and restart your computer if required.



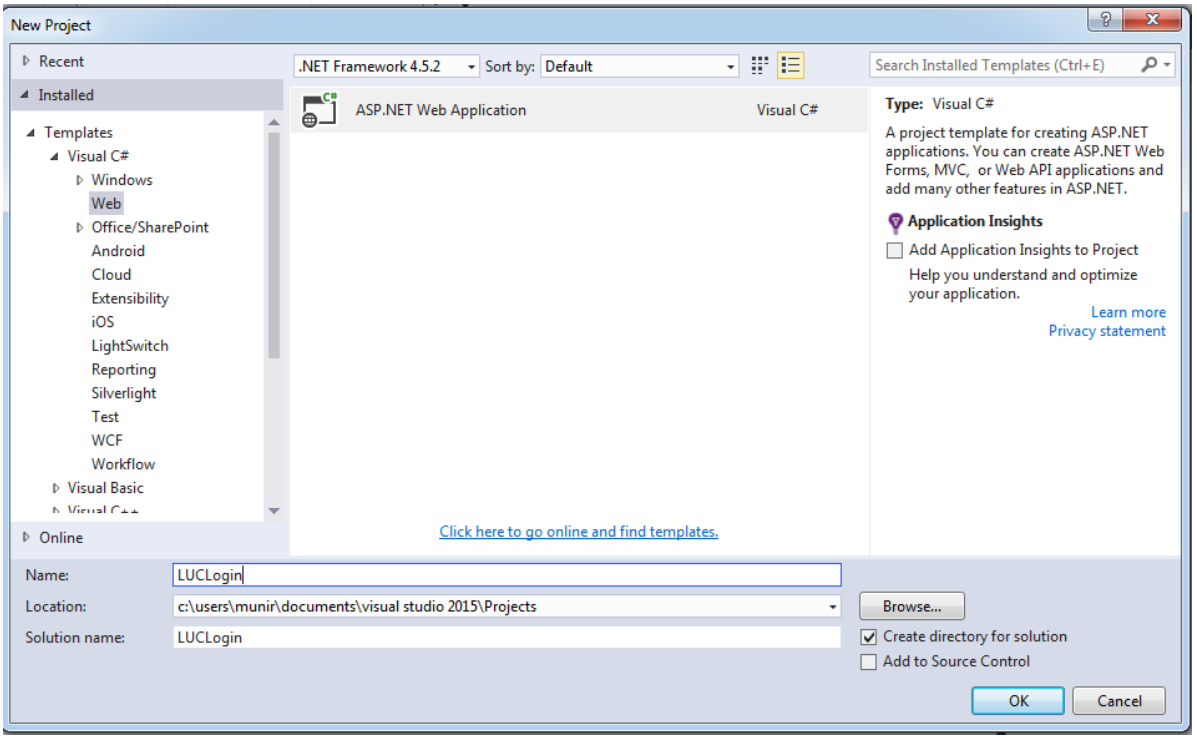
Step 4 − Open Visual Studio from start Menu which will open the following dialog. It will be a while for the first time for preparation.



Step 5 − Once all is done you will see the main window of Visual studio.

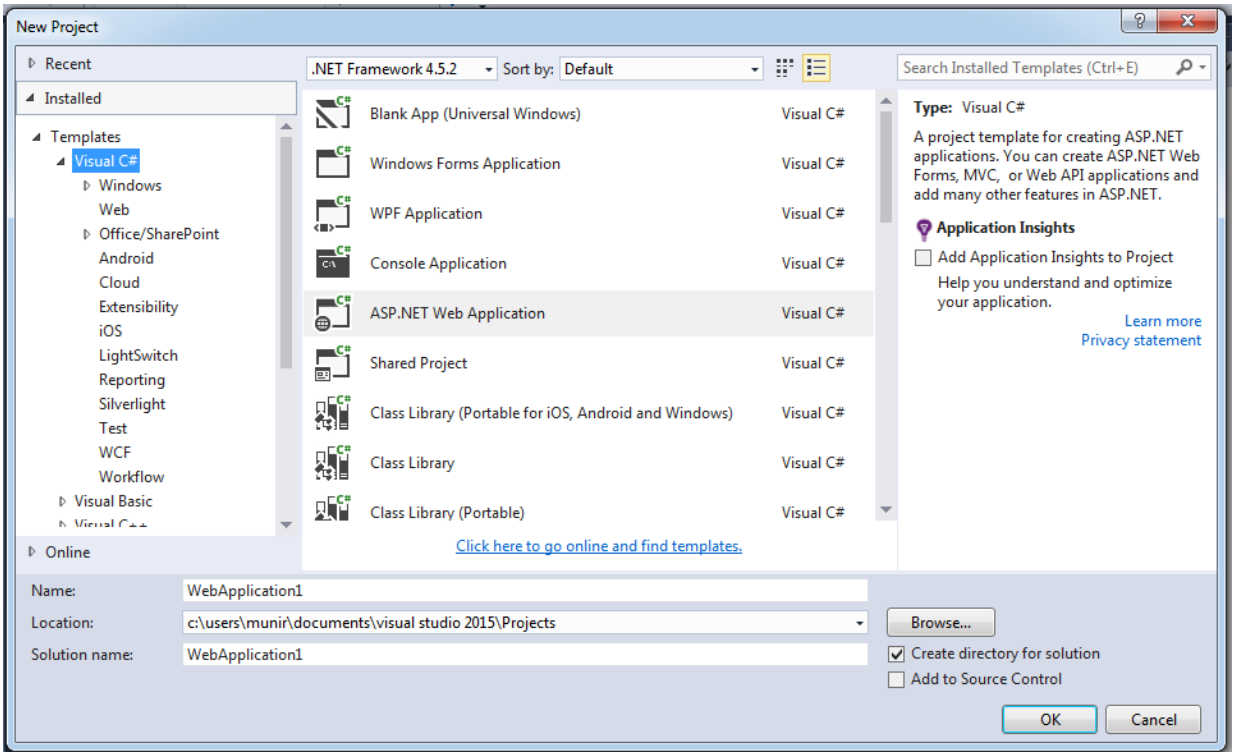


To start a new project, Click on `File → New → Project`



The Visual Studio IDE

The new project window allows choosing an application template from the available templates.



When you start a new web site, ASP.NET provides the starting folders and files for the site, including two files for the first web form of the site.

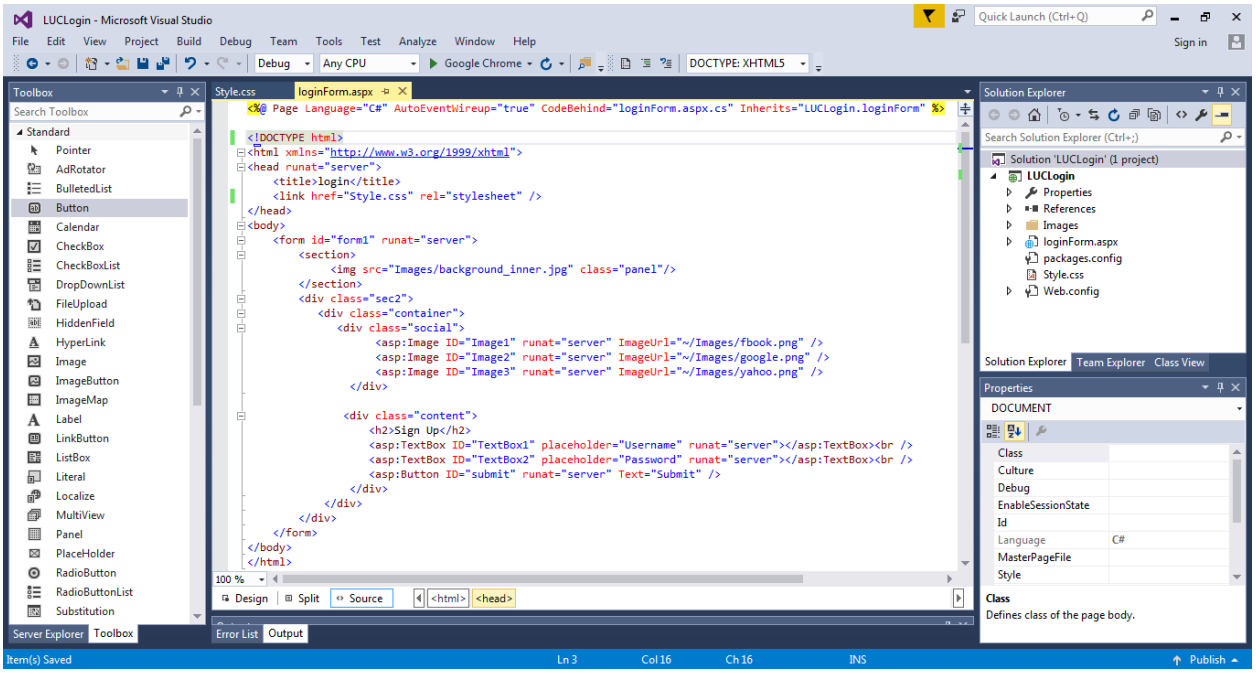
The file named Default.aspx contains the HTML and asp code that defines the form, and the file named Default.aspx.cs (for C# coding) or the file named Default.aspx.vb (for VB coding) contains the code in the language you have chosen and this code is responsible for the actions performed on a form.

The primary window in the Visual Studio IDE is the Web Forms Designer window. Other supporting windows are the Toolbox, the Solution Explorer, and the Properties window. You use the designer to design a web form, to add code to the control on the form so that the form works according to your need, you use the code editor.

Working with Views and Windows

You can work with windows in the following ways:

- To change the Web Forms Designer from one view to another, click on the Design or source button.
- To close a window, click on the close button on the upper right corner and to redisplay, select it from the View menu.
- To hide a window, click on its Auto Hide button. The window then changes into a tab. To display again, click the Auto Hide button again.
- To change the size of a window, just drag it.



Adding Folders and Files to your Website

When a new web form is created, Visual Studio automatically generates the starting HTML for the form and displays it in Source view of the web forms designer. The Solution Explorer is used to add any other files, folders or any existing item on the web site.

- To add a standard folder, right-click on the project or folder under which you are going to add the folder in the Solution Explorer and choose New Folder.
- To add an ASP.NET folder, right-click on the project in the Solution Explorer and select the folder from the list.
- To add an existing item to the site, right-click on the project or folder under which you are going to add the item in the Solution Explorer and select from the dialog box.

Projects and Solutions

A typical ASP.NET application consists of many items: the web content files (.aspx), source files (.cs files), assemblies (.dll and .exe files), data source files (.mdb files), references, icons, user controls and miscellaneous other files and folders. All these files that make up the website are contained in a Solution.

When a new website is created. Visual studio automatically creates the solution and displays it in the solution explorer.

Solutions may contain one or more projects. A project contains content files, source files, and other files like data sources and image files. Generally, the contents of a project are compiled into an assembly as an executable file (.exe) or a dynamic link library (.dll) file.
Typically a project contains the following content files:

- Page file (.aspx)
- User control (.ascx)
- Web service (.asmx)
- Master page (.master)
- Site map (.sitemap)
- Website configuration file (.config)

Building and Running a Project

You can execute an application by:

- Selecting Start

- Selecting Start Without Debugging from the Debug menu,
- pressing F5
- Ctrl-F5

The program is built meaning, the .exe or the .dll files are generated by selecting a command from the Build menu.

ASP.NET Life Cycle
ASP.NET life cycle specifies, how:

- ASP.NET processes pages to produce dynamic output
- The application and its pages are instantiated and processed
- ASP.NET compiles the pages dynamically

The ASP.NET life cycle could be divided into two groups:

- Application Life Cycle
- Page Life Cycle

ASP.NET Application Life Cycle
The application life cycle has the following stages:

- User makes a request for accessing application resource, a page. Browser sends this request to the web server.
- A unified pipeline receives the first request and the following events take place:
    - An object of the class ApplicationManager is created.
    - An object of the class HostingEnvironment is created to provide information regarding the resources.
    - Top level items in the application are compiled.
- Response objects are created. The application objects such as HttpContext, HttpRequest and HttpResponse are created and initialized.
- An instance of the HttpApplication object is created and assigned to the request.
- The request is processed by the HttpApplication class. Different events are raised by this class for processing the request.

ASP.NET Page Life Cycle
When a page is requested, it is loaded into the server memory, processed, and sent to the browser. Then it is unloaded from the memory. At each of these steps, methods and events are available, which could be overridden according to the need of the application. In other words, you can write your own code to override the default code.

The Page class creates a hierarchical tree of all the controls on the page. All the components on the page, except the directives, are part of this control tree. You can see the control tree by adding trace= "true" to the page directive. We will cover page directives and tracing under 'directives' and 'event handling'.
The page life cycle phases are:

- Initialization
- Instantiation of the controls on the page
- Restoration and maintenance of the state
- Execution of the event handler codes
- Page rendering

Understanding the page cycle helps in writing codes for making some specific thing happen at any stage of the page life cycle. It also helps in writing custom controls and initializing them at right time, populate their properties with view-state data and run control behavior code.
Following are the different stages of an ASP.NET page:

- Page request - When ASP.NET gets a page request, it decides whether to parse and compile the page, or there would be a cached version of the page; accordingly the response is sent.
- Starting of page life cycle - At this stage, the Request and Response objects are set. If the request is an old request or post back, the IsPostBack property of the page is set to true. The UICulture property of the page is also set.
- Page initialization - At this stage, the controls on the page are assigned unique ID by setting the UniqueID property and the themes are applied. For a new request, postback data is loaded and the control properties are restored to the view-state values.
- Page load - At this stage, control properties are set using the view state and control state values.
- Validation - Validate method of the validation control is called and on its successful execution, the IsValid property of the page is set to true.
- Postback event handling - If the request is a postback (old request), the related event handler is invoked.
- Page rendering - At this stage, view state for the page and all controls are saved. The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Response property of page.
- Unload - The rendered page is sent to the client and page properties, such as Response and Request, are unloaded and all cleanup done.

ASP.NET Page Life Cycle Events

At each stage of the page life cycle, the page raises some events, which could be coded. An event handler is basically a function or subroutine, bound to the event, using declarative attributes such as Onclick or handle.

Following are the page life cycle events:

- PreInit - PreInit is the first event in page life cycle. It checks the IsPostBack property and determines whether the page is a postback. It sets the themes and master pages, creates dynamic controls, and gets and sets profile property values. This event can be handled by overloading the OnPreInit method or creating a Page_PreInit handler.
- Init - Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page_Init handler.
- InitComplete - InitComplete event allows tracking of view state. All the controls turn on view-state tracking.
- LoadViewState - LoadViewState event allows loading view state information into the controls.
- LoadPostData - During this phase, the contents of all the input fields are defined with the <form> tag are processed.
- PreLoad - PreLoad occurs before the post back data is loaded in the controls. This event can be handled by overloading the OnPreLoad method or creating a Page_PreLoad handler.
- Load - The Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page_Load handler.
- LoadComplete - The loading process is completed, control event handlers are run, and page validation takes place. This event can be handled by overloading the OnLoadComplete method or creating a Page_LoadComplete handler
- PreRender - The PreRender event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.
- PreRenderComplete - As the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.

- SaveStateComplete - State of control on the page is saved. Personalization, control state and view state information is saved. The HTML markup is generated. This stage can be handled by overriding the Render method or creating a Page_Render handler.
- UnLoad - The UnLoad phase is the last phase of the page life cycle. It raises the UnLoad event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as database connections, are freed. This event can be handled by modifying the OnUnLoad method or creating a Page_UnLoad handler.

ACTIVITY

1. What is page state and session state?
2. Describe ASP.NET Component Model
3. Highlight 5 Components of .Net Framework
4. Describe the procedure to add Folders and Files to a Web project
5. Discuss ASP.NET Application Life Cycle
6. List the phases of ASP.NET page life cycle
7. Highlight the stages of an ASP.NET page life cycle
8. Highlight the events of an ASP.NET page life cycle

KEYWORDS

- ASP.NET
- Web Form
- Page
- Session
- .NET Framework

SUMMARY

This chapter presents the following topics:

- Introduction to ASP.NET
- ASP.NET Web Forms Model
- The ASP.NET Component Model
- Components of .Net Framework 3.5
- Environment Setup and Installation
- The Visual Studio IDE
- Working with Views and Windows
- Adding Folders and Files to your Website
- Projects and Solutions
- Building and Running a Project
- ASP.NET Life Cycle