

ASP.NET

Komal

*Dronacharya College of Engineering
Haryana*

ASP.net?

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 is an [open-source^{\[2\]} server-side Web application framework](#) designed for [Web development](#) to produce [dynamic Web pages](#). It was developed by [Microsoft](#) to allow [programmers](#) to build dynamic [web sites](#), [web applications](#) and [web services](#).

It was first released in January 2002 with version 1.0 of the [.NET Framework](#), and is the successor to Microsoft's [Active Server Pages](#) (ASP) technology. 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 .Net Framework family also includes two versions for mobile or embedded device use. A reduced version of the framework, the .NET Compact Framework, is available on Windows CE platforms, including Windows Mobile devices such as smartphones. Additionally, the .NET Micro Framework is targeted at severely resource-constrained devices.

Characteristics of Asp.Net Pages

- Directives
- User controls
- Custom Controls
- Rendering Techniques
- State management
- Application
- Session state
- Performance

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.

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.

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.
- **Init Complete** - InitComplete event allows tracking of view state. All the controls turn on view-state tracking.
- **Load View State** - LoadViewState event allows loading view state information into the controls.
- **Load Post Data** - During this phase, the contents of all the input fields are defined with the <form> tag are processed.
- **Pre Load** - 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.
- **Load Complete** - 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.

- **PreRender Complete** - As the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.
- **Save State Complete** - 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.

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. VB2008 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.

REFERENCE

- <https://en.wikipedia.org/wiki/ASP.NET>
- <http://blog.spec-india.com/characteristics-of-asp-net-pages>
- <https://kirandn.wordpress.com/about/characteristics-of-asp-net/>
- <http://www.microsoft.com/technet/prodtechnol/WindowsServer2003/Library/IIS/ad2a22b9-135c-432a-bc9f-c67f074242b7.mspx?mfr=true>
- http://www.tutorialspoint.com/asp.net/asp.net_environment_setup.htm