ASP.NET (4.6)

ASP.NET is a unified Web development model that includes the services necessary for you to build enterprise-class Web applications with a minimum of coding. ASP.NET is part of the .NET Framework, and when coding ASP.NET applications you have access to classes in the .NET Framework

ASP.NET offers three frameworks for creating web applications: ASP.NET Web Forms, ASP.NET MVC, and ASP.NET Web Pages.

ASP.NET Web forms (aspx)- RAD and WYSIWYG

ASP.NET Web Pages – cshtml and vbhtml

Whenever an ASP.Net application starts the execution begins froms Global.asax files and events in it like Application\_start,Application\_End,Session\_start,Session\_End,Application\_Error.

When a request comes to IIS then it will first enter HTTP.Sys part of IIS Kernel where it will find routing for the related Application Pool which serves this request.Then,this request is send to the User Mode of IIS where WAS services pass it to the related Application Pool. In an application pool, a worker process is executing which learns the extension of the request and initiate HttpRequest Processing Pipeline.

HttpRequest Pipeline consist of series of Http Modules and HttpHandler. Whenever some preprocessing is required on the request it is done by using Http Modules and following list of events are fired.

* **Begin Request** - When the new request gets created, the Begin Request will get fired and it will be the first event which will be always raised during the request processing.
* **Authenticate Request** - This event confirms that the authentication configuration has authenticated the request and subscription to this event ensures that before processing the attached handler and module, it is authenticated.
* **Authorize Request** - This event confirms that the request has been authorized by ASP.NET. To implement the custom authorization, you can use this event.
* **Resolve Request Cache** - Once the authorization event gets completed, this event calls the caching module to serve the request from the cache, bypassing execution of the event handler.
* **Map Request Handler** - This event is used by ASP.NET framework to check the extension of the file and accordingly checks the handler for the request.
* **Acquire Request State** - When this event is raised, ASP.NET acquires the state information (Session State) that is associated with the request. In this case the request must has a valid Session ID.
* **Execute Request Handler** - When the handler generates the output, this event gets raised. This is the only request which is not exposed by the HttpApplication class.
* **Release Request State** - This event gets raised after ASP.NET finishes executing all request handlers. It also signals ASP.NET State modules to save the current request state.
* **Update Request Cache** - Once ASP.NET finishes the execution of event handlers, this event gets fired to let the caching modules store responses that will be reused to serve identical requests from the cache.
* **Log Request** - This request gets fired before ASP.NET performs any logging for the current request. This event can also be used to perform the custom logging.

After this httpHandler fires and initates the page life cycle.

ASP.net page life Cycle

* Preinit-

Check the [IsPostBack](https://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) property to determine whether this is the first time the page is being processed. The [IsCallback](https://msdn.microsoft.com/en-us/library/system.web.ui.page.iscallback.aspx) and [IsCrossPagePostBack](https://msdn.microsoft.com/en-us/library/system.web.ui.page.iscrosspagepostback.aspx) properties have also been set at this time.Create or re-create dynamic controls.Set a master page dynamically.Set the [Theme](https://msdn.microsoft.com/en-us/library/system.web.ui.page.theme.aspx) property dynamically.Read or set profile property values.

* Init

Raised after all controls have been initialized and any skin settings have been applied. Use this event to read or initialize control properties.

* InitComplete

Use this event to make changes to view state that you want to make sure are persisted after the next postback.

* PreLoad

Raised after the page loads view state for itself and all controls

* Load

The [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object calls the [OnLoad](https://msdn.microsoft.com/en-us/library/system.web.ui.control.onload.aspx) method on the [Page](https://msdn.microsoft.com/en-us/library/system.web.ui.page.aspx) object, and then recursively does the same for each child control until the page and all controls are loaded. The [Load](https://msdn.microsoft.com/en-us/library/system.web.ui.control.load.aspx) event of individual controls occurs after the [Load](https://msdn.microsoft.com/en-us/library/system.web.ui.control.load.aspx) event of the page.Use the [OnLoad](https://msdn.microsoft.com/en-us/library/system.web.ui.control.onload.aspx) event method to set properties in controls and to establish database connections.

* Control events – Textbox,button events
* [LoadComplete](https://msdn.microsoft.com/en-us/library/system.web.ui.page.loadcomplete.aspx) - Use this event for tasks that require that all other controls on the page be loaded.
* [PreRender](https://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx) –
* [PreRender](https://msdn.microsoft.com/en-us/library/system.web.ui.control.prerender.aspx)Complete - Raised after each data bound control whose [DataSourceID](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.databoundcontrol.datasourceid.aspx) property is set calls its [DataBind](https://msdn.microsoft.com/en-us/library/system.web.ui.control.databind.aspx) method
* SaveStateComplete - Raised after view state and control state have been saved for the page and for all controls.
* Render - All ASP.NET Web server controls have a [Render](https://msdn.microsoft.com/en-us/library/system.web.ui.control.render.aspx) method that writes out the control's markup to send to the browser.
* Unload - Raised for each control and then for the page.

After Postback

* **Page\_LoadState Event** - During the round trips of client-to-server and server-to-client, ASP.NET maintains the state of the controls and the page, in the form of ViewState. ViewState contains the control ID and the value of the control, as a key-value pair. This ViewState is loaded during the post back of the request. **After InitComplete**
* **Page\_ProcessPostData Event** - This event processes the controls state by updating the control values with the data which is post back by the request.
* **Server Control Events** - Here all the events of the controls will be processed like Button click event, Textbox TextChanged event etc. **After Page\_Load**

**HttpHandler and Modules**

ASP.NET uses Http Handler and Http Modules mechanisms to process incoming ASP.NET requests, generate a response, and return that response to the client.

ASP.NET passes each incoming request through a layer of preprocessing HttpModules in the pipeline. ASP.NET allows multiple modules to exist in the pipeline for each request. After the incoming request has passed through each module, it is passed to the HttpHandler which serves the request. Notice that although a single request may pass through many different modules,it can be processed by one handler only. The handler is generally responsible for creating a response to the incoming HTTP request. After the handler has completed execution and generated a response, the response is passed back through a series of post-processing modules, before it is returned to the client.

**HTTPMODULES**

HttpModules are simple classes that can plug themselves into the request-processing pipeline. They do this by hooking into a handful of events thrown by the application as it processes the HTTP request. To create an HttpModule, you simply create a class that derives from the System.Web.IHttpModule interface.

Whenever a custom Http Module is Loaded it create instance of HttpApplication using which we can get current Request,Response and Session. Various HttpRequest event are fired too like – BeginRequest,AuthenticateRequest,AuthorizeRequest etc

This interface requires you to implement two methods: Init and Dispose

<system.webServer>

<modules>

<add name="AppendMessage" type="AppendMessage, App\_Code" />

</modules>

</system.webServer>

Uses:

*Security*: For authenticating a request before the request is handled.

*Statistics and Logging*: Since modules are called for every request they can be used for gathering statistics and for logging information.

*Custom header*:  Since response can be modified, one can add custom header information to the response. Like File and Image compression mechanism

HttpHandlers(\*.ashx)

Handlers are the last stop for incoming HTTP requests and are ultimately the point in the requestprocessing

pipeline that is responsible for serving up the requested content, be it an ASPX page, .ashx ,HTML,

plaintext, or an image.

It is inherited from IHttpHandler and has ProcessRequest Method and IsReusable property

It can be used to check server health to ie (Ping IIS)

**State Management Techniques in ASP.net**

Client Side:

* View State - View state is used automatically by the ASP.NET page framework to persist information that must be preserved between postbacks. View state provides state information for a specific ASP.NET page.

Set value in view state

  //Value of Textbox1 and TectBox2 is assigin on the ViewState

    ViewState["name"] = TextBox1.Text;

    ViewState["password"] = TextBox2.Text;

Get value from viewstate

 if (ViewState["name"] != null)

    {

        TextBox1.Text = ViewState["name"].ToString();

    }

    if (ViewState["password"] != null)

    {

         TextBox2.Text = ViewState["password"].ToString();

    }

To disable view state <%Page Language="C#" EnableViewState="false";%>

To secure view state

<%Page Language="C#" EnableViewState="true ViewStateEncryptionMode="Always" %>

ViewStateEncryptionMode :Always ,Auto,Never

<%Page Language="C#" EnableViewState="true"  EnableViewStateMac="true";%>

* Control State
* Cookies – size 4096 bytes

Set cookies

HttpCookie aCookie = new HttpCookie("lastVisit");

aCookie.Value = DateTime.Now.ToString();

aCookie.Expires = DateTime.Now.AddDays(1);

Response.Cookies.Add(aCookie);

Read Cookies

if(Request.Cookies["userName"] != null)

{

HttpCookie aCookie = Request.Cookies["userName"];

Label1.Text = Server.HtmlEncode(aCookie.Value);

}

* Query String
* Hidden Fields

Server side

Application State

Session State

Profile Properties