**ASP.NET Page Life Cycle**

[**http://www.codeproject.com/Articles/52857/ASP-NET-Page-Life-Cycle**](http://www.codeproject.com/Articles/52857/ASP-NET-Page-Life-Cycle)

By [Indrajeet Sutar](http://www.codeproject.com/script/Membership/View.aspx?mid=6201101), 17 Jan 2010

Top of Form

* [Download source - 3.9 KB](http://www.codeproject.com/KB/aspnet/Page_Life_Cycle_Explain/PageLifeCycle.zip)

**Introduction**

This is my first article on ASP.NET and I will try to explain the basics of ASP.NET page life cycle and an easy understanding of its usage.

It is very important to understand ASP.NET page life cycle for many reasons, mainly for understanding where we should place particular methods and when we should set page related properties. Additionally if we are developing custom server controls, then clear understanding of page life cycle is very useful to correct initialization of control, setting properties with view-state data, and control’s code (Control events are subject to ASP.NET page only).

Page life cycle depends on whether it is requested for the first time or it is after postback (page request of itself) and finalize to the web server. When a web page is requested to the web server, it goes through a series of sequence of steps/events (like initialization, instantiating controls, restoring and maintaining state, running event handler code, and rendering) before it is returned back to the web browser.

When we use and manipulate page life cycle events correctly, then it becomes a handy and powerful tool for developing web applications.

**Background**

**IIS**: It is the default web server with Microsoft .NET. Internet Information Server helps to deploy web sites/web application. IIS web server receives requests for a web resource (file), it checks the extension of the file (e.g. .aspx, .ascx, .ashx and .asmx) and then it determines which ISAPI extension should handle this request and passes this request to proper ISAPI extension.

**ASPNET\_ISAPI.DLL**: IIS loads this DLL and sends the page request to this DLL. This DLL loads the HTTPRuntime for further processing.

**ASPNET\_WP.EXE**: It contains an Application Pool. Each Application Pool can contain any number of Applications. Application Pool is also called as AppDomain. When a web page is requested, IIS looks for the application pool under which the current application is running and forwards the request to the respective worker process.

**Explanation**

General Page Life Cycle Stages each and every time the browser sends the request, page instance is created. HTTP runtime invokes ProcessRequest and starts page execution.

**Table Showing Stage and Corresponding Events**

| Stage | Events/Method |
| --- | --- |
| Initialization of the page | Page\_Init |
| Loading of the View State | LoadViewState |
| Processing of the Postback data | LoadPostData |
| Loading of Page | Page\_Load |
| Notification of PostBack | RaisePostDataChangedEvent |
| Handling of PostBack Event | RaisePostBackEvent |
| Pre Rendering of Page | Page\_PreRender |
| Saving of view state | SaveViewState |
| Rendering of Page | Page\_Render |
| Unloading of the Page | Page\_UnLoad |

**Setting Up Of the Page**

ASP.NET determines that the page request by a user requires to be parsing and compiling or to render cached copy of the page to be sent. Thus it comes very much before the beginning of the page life cycle. After this, it is also checked that request is a normal request, postback, cross-page postback or callback. The page constructor creates a tree of controls as soon as the HTTP runtime instantiates the page class to perform the request.

**Events**

**PreInit**

This event is the beginning of the page life cycle. All page controls are initialized and the properties are set according to the aspx source code.

* Possible to change or set Master page, themes
* Creates or re-creates dynamic controls
* Reads or sets Profile property values

**Init**

First, the Init event for the Page object occurs, then Init event occurs for each control on the Page. Viewstate information is not available at this stage.

* Controls have been initialized
* Theme skins applied if any
* Initialize control properties

**InitComplete**

This event is used for processing tasks that require all initialization to be complete.

**PreLoad**

This event is used before performing any processing that should occur before Load. Use this event if you need to perform processing on your page or control before the Load event. Before the Page instance raises this event, it loads view state for itself and all controls, and then processes any postback data included with the Request instance.

**Load**

Set properties in controls and establish database connections.

**Control Events**

These are control specific events such as – Button Click, DropDownIndexChanged, etc.

**Load Complete**

This event is used for performing those tasks which require load has been completed.

**PreRender**

In this event, Page ensures that all child controls are created. Page calls EnsureChildControls for all controls, including itself. Every control whose datasource/databind property is set calls for its databind method.

**SaveStateComplete**

This event occurs after viewstate is encoded and saved for the page and for all controls.

**Render**

Every ASP.NET control has render method and the page instance calls this method to output the control’s markup, after this event any changes to the page or controls are ignored.

**Unload**

Unload event is used to do cleanup tasks like closing the database connections, closing of the open files, completing logging or other requested tasks. Unload events occurs for each control on page control tree first and after that page.

Bottom of Form