Asp.net Page Life Cycle

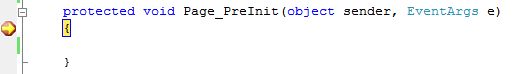
ASP.NET Page Life Cycle Events

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.

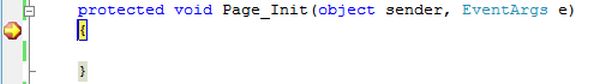
Following are the page life cycle events:

* **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.

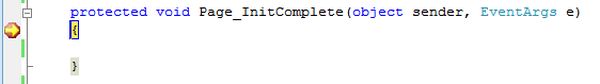
**PreInit**

1. Check the IsPostBack property to determine whether this is the first time the page is being processed.
2. Create or re-create dynamic controls.
3. Set a master page dynamically.
4. Set the Theme property dynamically.  
     
   

**Note:**If the request is a postback then the values of the controls have not yet been restored from the view state. If you set a control property at this stage, its value might be overwritten in the next event.  
 **Init**

1. This event fires after each control has been initialized.
2. Each control's UniqueID is set and any skin settings have been applied.
3. Use this event to read or initialize control properties.
4. The "Init" event is fired first for the bottom-most control in the hierarchy, and then fired up the hierarchy until it is fired for the page itself.  
     
   

**InitComplete**

1. Until now the viewstate values are not yet loaded, hence you can use this event to make changes to the view state that you want to ensure are persisted after the next postback.
2. Raised by the Page object.
3. Use this event for processing tasks that require all initialization to be complete.  
     
   

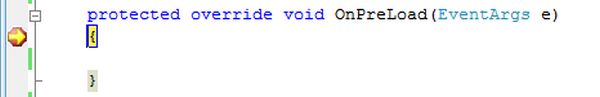
**OnPreLoad**

**(1)Loads ViewState** : ViewState data are loaded to controls

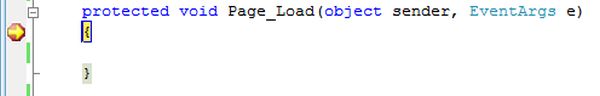
Note : The page viewstate is managed by ASP.NET and is used to persist information over a page roundtrip to the server. Viewstate information is saved as a string of name/value pairs and contains information such as control text or value. The viewstate is held in the value property of a hidden <input> control that is passed from page request to page request.

**(2)Loads Postback data** : postback data are now handed to the page controls

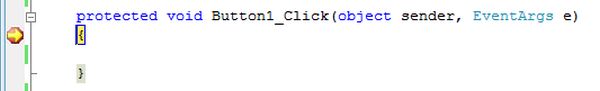
Note : During this phase of the page creation, form data that was posted to the server (termed postback data in ASP.NET) is processed against each control that requires it. Hence, the page fires the LoadPostData event and parses through the page to find each control and updates the control state with the correct postback data. ASP.NET updates the correct control by matching the control's unique ID with the name/value pair in the NameValueCollection. This is one reason that ASP.NET requires unique IDs for each control on any given page.

1. Raised after the page loads view state for itself and all controls, and after it processes postback data that is included with the Request instance.
2. 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.
3. Loads ViewState: ViewState data are loaded to controls.
4. Loads Postback data: Postback data are now handed to the page controls.  
     
   

**Load**

1. The Page object calls the OnLoad method on the Page object, and then recursively does the same for each child control until the page and all controls are loaded. The Load event of individual controls occurs after the Load event of the page.
2. This is the first place in the page lifecycle that all values are restored.
3. Most code checks the value of IsPostBack to avoid unnecessarily resetting state.
4. You may also call Validate and check the value of IsValid in this method.
5. You can also create dynamic controls in this method.
6. Use the OnLoad event method to set properties in controls and establish database connections.  
     
   

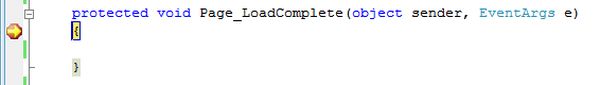
**Control PostBack Event(s)**

1. ASP.NET now calls any events on the page or its controls that caused the PostBack to occur.
2. Use these events to handle specific control events, such as a Button control's Click event or a TextBox control's TextChanged event.
3. In a postback request, if the page contains validator controls, check the IsValid property of the Page and of individual validation controls before performing any processing.
4. This is just an example of a control event. Here it is the button click event that caused the postback.  
     
   

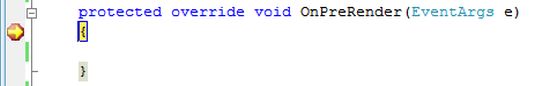
**LoadComplete**

1. Raised at the end of the event-handling stage.
2. Use this event for tasks that require that all other controls on the page be loaded.

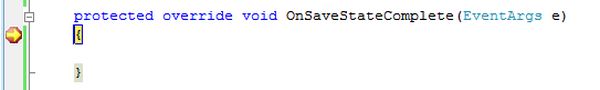
============= Or

1. It  raised at the end of the event handling stage.
2. It use this event for tasks that require that all other controls on the page be loaded.
3. All the event processing has been done in this event method.
4. 

**OnPreRender**

1. Raised after the Page object has created all controls that are required in order to render the page, including child controls of composite controls.
2. The Page object raises the PreRender event on the Page object, and then recursively does the same for each child control. The PreRender event of individual controls occurs after the PreRender event of the page.
3. The PreRender event of individual controls occurs after the PreRender event of the page.
4. PreRender event method is called when page has  created all the controls and displayed in browser.
5. Allows final changes to the page or its control.
6. This event takes place before saving ViewState, so any changes made here are saved.
7. For example: After this event, you cannot change any property of a button or change any viewstate value.
8. Each data bound control whose DataSourceID property is set calls its DataBind method.
9. Use the event to make final changes to the contents of the page or its controls.  
     
   

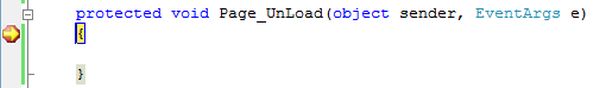
**OnSaveStateComplete**

1. Raised after view state and control state have been saved for the page and for all controls.
2. Before this event occurs, ViewState has been saved for the page and for all controls.
3. Any changes to the page or controls at this point will be ignored.
4. Use this event perform tasks that require the view state to be saved, but that do not make any changes to controls.  
     
   

**Render Method**

1. This is a method of the page object and its controls (and not an event).
2. The Render method generates the client-side HTML, Dynamic Hypertext Markup Language (DHTML), and script that are necessary to properly display a control at the browser.

**UnLoad**

1. This event is used for cleanup code.
2. At this point, all processing has occurred and it is safe to dispose of any remaining objects, including the Page object.
3. Cleanup can be performed on:  
   * Instances of classes, in other words objects
   * Closing opened files
   * Closing database connections.
4. This event occurs for each control and then for the page.
5. During the unload stage, the page and its controls have been rendered, so you cannot make further changes to the response stream.
6. If you attempt to call a method such as the Response.Write method then the page will throw an exception.  
     
   

**EXAMPLES**  
  
**Example 1: Control Values**  
  
In the following code, I have assigned the values to the label control on each event. When you run the code, you will see that in the "Page\_UnLoad", the values are not assigned to the label. WHY? Because, during the unload stage, the page and its controls have been rendered, so you cannot change the values.  
  
Please observe the code comments and output. It will help you to clearly understand the concepts.

public partial class PageLiftCycle : System.Web.UI.Page

{

    protected void Page\_PreInit(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        lblName.Text = lblName.Text + "<br/>" + "PreInit";

    }

    protected void Page\_Init(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        lblName.Text = lblName.Text + "<br/>" + "Init";

    }

    protected void Page\_InitComplete(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        lblName.Text = lblName.Text + "<br/>" + "InitComplete";

    }

    protected override void OnPreLoad(EventArgs e)

    {

        //Work and It will assign the values to label.

        //If the page is post back, then label contrl values will be loaded from view state.

        //E.g: If you string str = lblName.Text, then str will contain viewstate values.

        lblName.Text = lblName.Text + "<br/>" + "PreLoad";

    }

    protected void Page\_Load(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        lblName.Text = lblName.Text + "<br/>" + "Load";

    }

    protected void btnSubmit\_Click(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        lblName.Text = lblName.Text + "<br/>" + "btnSubmit\_Click";

    }

    protected void Page\_LoadComplete(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        lblName.Text = lblName.Text + "<br/>" + "LoadComplete";

    }

    protected override void OnPreRender(EventArgs e)

    {

        //Work and It will assign the values to label.

        lblName.Text = lblName.Text + "<br/>" + "PreRender";

    }

    protected override void OnSaveStateComplete(EventArgs e)

    {

        //Work and It will assign the values to label.

        //But "SaveStateComplete" values will not be available during post back. i.e. View state.

        lblName.Text = lblName.Text + "<br/>" + "SaveStateComplete";

    }

    protected void Page\_UnLoad(object sender, EventArgs e)

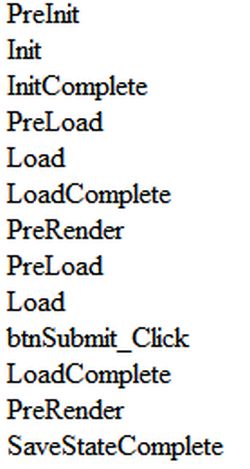
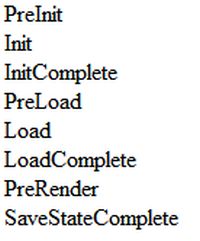
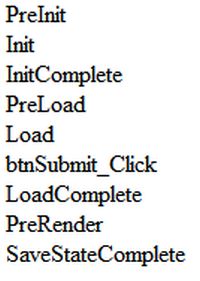
    {

        //Work and it will not effect label contrl, view stae and post back data.

        lblName.Text = lblName.Text + "<br/>" + "UnLoad";

    }

}

**Output**  
  
The first time the Page Load is output:  
  
  
  
When you click on the Submit Button output:  
  
  
  
During the first time of the page load with EnableViewState="false":  
  
  
When you click on the Submit Button output with EnableViewState="false":  
  
  
**Example 2: ViewState Values**  
  
Please observe the code comments and output. It will help you to clearly understand the concepts.

public partial class PageLiftCycle : System.Web.UI.Page

{

    protected void Page\_PreInit(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        //Note : If page is post back or first time call and you have not set any values to ViewState["value"], then

        //Convert.ToString(ViewState["value"]) is always empty.

        ViewState["value"] = Convert.ToString(ViewState["value"]) + "<br/>" + "PreInit";

        lblName.Text = Convert.ToString(ViewState["value"]);

    }

    protected void Page\_Init(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        //Note : If page is post back or first time call and you have not set any values to ViewState["value"] in privious events, then

        //Convert.ToString(ViewState["value"]) is always empty.

        ViewState["value"] = Convert.ToString(ViewState["value"]) + "<br/>" + "Init";

        lblName.Text = Convert.ToString(ViewState["value"]);

    }

    protected void Page\_InitComplete(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        //Note : If page is post back or first time call and you have not set any values to ViewState["value"] in privious events, then

        //Convert.ToString(ViewState["value"]) is always empty.

        ViewState["value"] = Convert.ToString(ViewState["value"]) + "<br/>" + "InitComplete";

        lblName.Text = Convert.ToString(ViewState["value"]);

    }

    protected override void OnPreLoad(EventArgs e)

    {

        //Work and It will assign the values to label.

        //Note : If page is post back and you have set or not set any values to ViewState["value"] in privious events, then

        //Convert.ToString(ViewState["value"]) will always have post back data.

        //E.g: If you string str = Convert.ToString(ViewState["value"]), then str will contain post back values.

        ViewState["value"] = Convert.ToString(ViewState["value"]) + "<br/>" + "PreLoad";

        lblName.Text = Convert.ToString(ViewState["value"]);

    }

    protected void Page\_Load(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        ViewState["value"] = Convert.ToString(ViewState["value"]) + "<br/>" + "Load";

        lblName.Text = Convert.ToString(ViewState["value"]);

    }

    protected void btnSubmit\_Click(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        ViewState["value"] = Convert.ToString(ViewState["value"]) + "<br/>" + "btnSubmit\_Click";

        lblName.Text = Convert.ToString(ViewState["value"]);

    }

    protected void Page\_LoadComplete(object sender, EventArgs e)

    {

        //Work and It will assign the values to label.

        ViewState["value"] = Convert.ToString(ViewState["value"]) + "<br/>" + "LoadComplete";

        lblName.Text = Convert.ToString(ViewState["value"]);

    }

    protected override void OnPreRender(EventArgs e)

    {

        //Work and It will assign the values to label.

        ViewState["value"] = Convert.ToString(ViewState["value"]) + "<br/>" + "PreRender";

        lblName.Text = Convert.ToString(ViewState["value"]);

    }

    protected override void OnSaveStateComplete(EventArgs e)

    {

        //Work and It will assign the values to label.

        //But "SaveStateComplete" values will not be available during post back. i.e. View state.

        ViewState["value"] = Convert.ToString(ViewState["value"]) + "<br/>" + "SaveStateComplete";

        lblName.Text = Convert.ToString(ViewState["value"]);

    }

    protected void Page\_UnLoad(object sender, EventArgs e)

    {

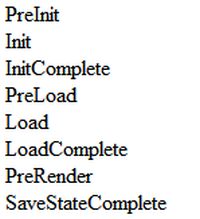
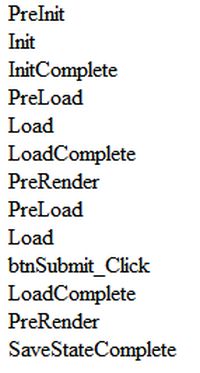
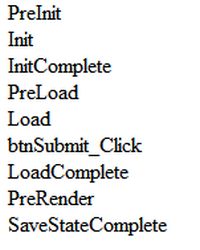
        //Work and it will not effect label contrl values, view state and post back data.

        ViewState["value"] = Convert.ToString(ViewState["value"]) + "<br/>" + "UnLoad";

        lblName.Text = Convert.ToString(ViewState["value"]);

    }

}

**Output**  
  
During the first the Time Page Load is output:  
  
  
  
When you click on the Submit Button output:  
  
  
  
During the first time the page loads with EnableViewState="false":  
  
  
  
When you click on the Submit Button the output with EnableViewState="false":  
  
  
  
**Example 3: ViewState Values**  
  
Please observe the code comments and output. It will help you to clearly understating the concepts.

public partial class PageLiftCycle : System.Web.UI.Page

{

    protected void Page\_PreInit(object sender, EventArgs e)

    {

        Response.Write("<br/>" + "PreInit");

    }

    protected void Page\_Init(object sender, EventArgs e)

    {

        Response.Write("<br/>" + "Init");

    }

    protected void Page\_InitComplete(object sender, EventArgs e)

    {

        Response.Write("<br/>" + "InitComplete");

    }

    protected override void OnPreLoad(EventArgs e)

    {

        Response.Write("<br/>" + "PreLoad");

    }

    protected void Page\_Load(object sender, EventArgs e)

    {

        Response.Write("<br/>" + "Load");

    }

    protected void Page\_LoadComplete(object sender, EventArgs e)

    {

        Response.Write("<br/>" + "LoadComplete");

    }

    protected override void OnPreRender(EventArgs e)

    {

        Response.Write("<br/>" + "PreRender");

    }

    protected override void OnSaveStateComplete(EventArgs e)

    {

        Response.Write("<br/>" + "SaveStateComplete");

    }

    protected void Page\_UnLoad(object sender, EventArgs e)

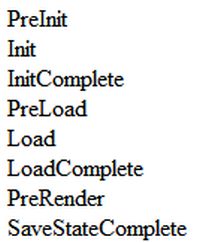
    {

        //Runtime Error : Response is not available in this context.

        //Response.Write("<br/>" + "UnLoad"); //Error

    }

}

**Output**  
  
  
  
Note: If you write Response.Write("<br/>" + "UnLoad"); in the Page\_UnLoad event, then it will genenrate the Runtime Error "Response is not available in this context".  
