SOS IN COMPUTER SCIENCE & APPLICATION
JIWAJI UNIVERSITY

Class : MBA (E-Commerce) II Semester

Subject : .Net Technologies

Paper Code: (204)

Topic: (i) ASP .NET Application and page life cycle.

(ii) ASP .NET Controls.

- 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

- 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:

- 1. User makes a request for accessing application resource, a page. Browser sends this request to the web server.
- 2. A unified pipeline receives the first request and the following events take place:
- i. An object of the class ApplicationManager is created.
- ii. An object of the class HostingEnvironment is created to provide information regarding the resources.
- iii. Top level items in the application are compiled.
- 3. Response objects are created. The application objects such as HttpContext, HttpRequest and HttpResponse are created and initialized.
- 4. An instance of the HttpApplication object is created and assigned to the request.
- 5. 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.

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

ASP.NET Page Life Cycle (cont..)

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.

ASP.NET Page Life Cycle (cont..)

- **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.

Web Form Controls–

ASP.NET – Server Controls

Server controls are tags that are understood by the server. There are three kinds of server controls:

- HTML Server Controls – Traditional HTML tags
- Web Server Controls – New ASP.NET tags
- Validation Server Controls – For input validation

ASP.NET – HTML Server Controls

HTML server controls are HTML tags understood by the server.

HTML elements in ASP.NET files are, by default, treated as text. To make these elements programmable, add a runat="server" attribute to the HTML element.

Web Form Controls (cont..)

This attribute indicates that the element should be treated as a server control. The id attribute is added to identify the server control. The id reference can be used to manipulate the server control at run time.

Note: All HTML server controls must be within a <form> tag with the runat="server" attribute. The runat="server" attribute indicates that the form should be processed on the server. It also indicates that the enclosed controls can be accessed by server scripts. In the following example we declare an HtmlAnchor server control in an .aspx file. Then we manipulate the HRef attribute of the HtmlAnchor control in an event handler (an event handler is a subroutine that executes code for a given event). The Page_Load event is one of many events that ASP.NET understands:

```
<script runat="server"> Sub Page_Load link1.HRef="http://www.abc.com" End Sub </script>
<html> <body> <form runat="server"> <a id="link1" runat="server">Visit again!</a>
</form> </body> </html>
```

Web Form Controls (cont..)

ASP.NET - Web Server Controls

Web server controls are special ASP.NET tags understood by the server. Like HTML server controls, Web server controls are also created on the server and they require a runat="server" attribute to work. However, Web server controls do not necessarily map to any existing HTML elements and they may represent more complex elements.

The syntax for creating a Web server control is:

```
<asp:control_name id="some_id" runat="server" />
```

In the following example we declare a Button server control in an .aspx file. Then we create an event handler for the Click event which changes the text on the button:

```
<script runat="server"> Sub submit(Source As Object, e As EventArgs) button1.Text="You clicked me!" End Sub </script> <html> <body> <form runat="server"> <asp:Button id="button1" Text="Click me!" runat="server" OnClick="submit"/> </form> </body> </html>
```

Web Form Controls (cont..)

ASP.NET - Validation Server Controls

Validation server controls are used to validate user-input. If the user-input does not pass validation, it will display an error message to the user. Each validation control performs a specific type of validation (like validating against a specific value or a range of values). By default, page validation is performed when a Button, ImageButton, or LinkButton control is clicked. You can prevent validation when a button control is clicked by setting the CausesValidation property to false. The syntax for creating a Validation server control is: <asp:control_name id="some_id" runat="server" />

In the following example we declare one TextBox control, one Button control, and one RangeValidator control in an .aspx file. If validation fails, the text "The value must be from 1 to 100!" will be displayed in the RangeValidator control:

Example

```
<html> <body> <form runat="server"> <p>Enter a number from 1 to 100:
<asp:TextBox id="tbox1" runat="server" /> <br /><br /> <asp:Button Text="Submit"
runat="server" /> </p> <p> <asp:RangeValidator ControlToValidate="tbox1"
MinimumValue="1" MaximumValue="100" Type="Integer" Text="The value must be
from 1 to 100!" runat="server" /> </p> </form> </body> </html>
```