# ASP.NET 4.x

View State

* <https://www.c-sharpcorner.com/UploadFile/225740/what-is-view-state-and-how-it-works-in-Asp-Net53/>

Session State

* Session state variables are available across all pages, but only for a given single session. Session variables are like single-user global data. Only the current session has access to its Session state.

Application State

* Application State variables are available across all pages and across all sessions. Application State variables are like multi-user global data. All sessions can read and write Application State variables.

**In an ASP.NET web application, Global.asax file conatins the application level events.**  
void Application\_Start(object sender, EventArgs e)  
{  
    // Code that runs on application startup  
}  
  
void Application\_End(object sender, EventArgs e)  
{  
    //  Code that runs on application shutdown  
}  
  
void Application\_Error(object sender, EventArgs e)  
{  
    // Code that runs when an unhandled error occurs  
}  
  
void Session\_Start(object sender, EventArgs e)  
{  
    // Code that runs when a new session is started  
}  
  
void Session\_End(object sender, EventArgs e)  
{  
    // Code that runs when a session ends.   
    // Note: The Session\_End event is raised only when the sessionstate mode  
    // is set to InProc in the Web.config file. If session mode is set to StateServer   
    // or SQLServer, the event is not raised.  
}

ViewState Vs Session State Vs Application State

|  |  |  |
| --- | --- | --- |
| ViewState | Session State | Application State |
| **1.** ViewState of a webform is available only with in that webform **2.** ViewState is stored on the page in a hidden field called \_ViewState. Because of this, the ViewState, will be lost, if you navigate awaya from the page, or if the broswer is closed.  **3.** ViewState is used by all asp.net controls to retain their state across postback | **1.** Session state variables are available across all pages, but only for a given single session. Session variables are like single-user global data. **2.** Session state variables are stored on the web server. **3.** SessionState variables are cleared, when the user session times out. The default is 20 minutes. This is configurable in web.config | **1.** Application State variables are available across all pages and across all sessions. Application State variables are like multi-user global data. **2.** Application State variables are stored on the web server. **3.** Application State variables are cleared, when the process hosting the application is restarted. |

Page Events

**The following are some of the commonly used events in the life cycle of an asp.net webform**. These events are shown in order of occurrence, except for, **Error event**, which occurs only if there is an unhandled exception.  
**PreInit** - As the name suggests, this event happens just before page initialization event starts.  IsPostBack, IsCallback and IsCrossPagePostBack properties are set at this stage. This event allows us to set the master page and theme of a web application dynamically. PreInit is extensively used when working with dynamic controls.

**Init** - Page Init, event occurs after the Init event, of all the individual controls on the webform. Use this event to read or initialize control properties. The server controls are loaded and initialized from the Web form’s view state.  
  
**InitComplete** - As the name says, this event gets raised immediately after page initialization.  
  
**PreLoad** - Happens just before the Page Load event.  
  
**Load** - Page Load event, occurs before the load event of all the individual controls on that webform.   
  
**Control Events** - After the Page load event, the control events like button's click, dropdownlist's selected index changed events are raised.  
  
**Load Complete** - This event is raised after the control events are handled.  
  
**PreRender** - This event is raised just before the rendering stage of the page.   
  
**PreRenderComplete** - Raised immediately after the PreRender event.

**Unload** - Raised for each control and then for the page. At this stage the page is, unloaded from memory.  
  
**Error** - This event occurs only if there is an unhandled exception.

Server Events

**Postback events** - These events submit the Web page, immediately to the server for processing. **Click event** of a button control is an example for **PostBack event**.  
  
**Cached events** - These events are saved in the page’s view state to be processed when a postback event occurs. **TextChanged** event of **TextBox** control, and **SelectedIndexChanged** event of a **DropDownList** control are examples of cached events. Cached events can be converted into postback events, by setting the **AutoPostBack** property of the control to true.  
  
**Validation events** - These events occur on the client, before the page is posted back to the server. All validation controls use these type of events.

Text Box Control

**Properties of a TextBox control**  
**1. TextMode Propertry** - SingleLine, MultiLine and Password.  
When you set the TextMode to MultiLine, use **Rows** property to control the number of lines to display for a MultiLine TextBox.  
  
**2. Text** - Use this property to set or get the Text from the TextBox.  
  
**3. MaxLength** - The maximum number of chatacters that a user can enter.  
  
**4. ReadOnly** - Set this property to true if you don't want the user to change the text in the TextBox.  
  
**5. ToolTip** - The tooltip is displayed when the mouse is over the control.  
  
**6. Columns** - Use this property to specify the width of the TextBox in characters  
  
**7. Height** - Set the height  
  
**8. Width** - Set the width  
  
**9. AutoPostBack** - By default, the TextChanged event of a TextBox control is cached in the viewstate, and is executed when the webform is submitted thru a postback by clicking the button control. If you want to change this behaviour, and post the webform immediately when the Text is changed, set AutoPostBack to true. Setting this property to true, will convert the cached event into a postback event.

**Events of TextBox:**  
**TextChanged** - This event is fired, when the text is changed.  
  
**Methods of a TextBox:**  
**Focus** - Set input focus onto the control.

Radio Button

**Important Properties of the Radio Button Control**  
**Checked** - This is a boolean property, that is used to check if the button is checked or not.  
  
**Text** - This is string property used to get or set the text associated with the radio button control  
  
**TextAlign** - right or left. On which side of the radio button the text should appear  
  
**AutoPostBack** - Set this property to true, if you want the webform to be posted immediately when the checked status of the radio button changes.  
  
**Group Name** - By default, the individual radio button selections, are not mutually exclusive. If you have a group of radio buttons, and if you want the selections among the group to be mutually exclusive, then use the same group name for all the radio button controls.  
  
**Events:**  
**CheckedChanged** - This event is fired when the checked status of the radio button control is changed.

Check Box

**Important Properties of the CheckBox Control**  
**Checked** - This is a boolean property, that is used to check if the check box is checked or not.  
**Text** - This is a string property used to get or set the text associated with the check box control  
**TextAlign** - right or left. On which side of the check box the text should appear  
**AutoPostBack** - Set this property to true, if you want the webform to be posted immediately when the checked status of the check box changes.  
  
**Methods:**  
**Focus()** - Just like TextBox, checkbox also supports, Focus() method. If you want to set the input focus, to a specific checkbox, Call this method for that check box control.  
  
**Events:**  
**CheckedChanged** - This event is fired when the checked status of the check button control is changed.

**HyperLink**

**Properties:**  
**Text**- The link text that will be shown to the user  
  
**Navigate URL** - The URL of the page to which the user will be sent  
  
**ImageURL** - The URL of the image, that will be displayed for the link. If you specify both the Text and ImageUrl, the image will be displayed instead of the text. If for some reason, the image is not unavailable, the text will be displayed.  
  
**Target** - If target is not specified, the web page to which the hyperlink is linked, will be displayed in the same window. If you set the Target to \_blank, the web page will be opened in a new window.  
  
**Methods:**  
**Focus**() - Call this method to Set the input focus when the page loads.  
  
**Events:**  
No HyperLink control specific events

Buttons

1. **Button** - The Button control is used to display a push button. Use the Text property to change the Text on the Button control.  
   **2. LinkButton** - LinkButton displays the button like a HyperLink. Use the Text property to change the LinkText.  
   **3. ImageButton** - ImageButton provides the flexibility of associating an Image with the button, using the ImageURL property.

### Command Event of an asp.net button control

When the Button is clicked, both the events are raised. **Click event happens before the Command event.**

**The click event handler and the command event handlers**, are attached to the respective **Click** and **Command** events in the HTML using **onclick**and **oncommand**attributes. The event handlers can also be attached programatically as shown below.  
protected void Page\_Load(object sender, EventArgs e)  
{  
    Button1.Click += new EventHandler(Button1\_Click);  
    Button1.Command += new CommandEventHandler(Button1\_Command);  
}

**If you have multiple button controls on a webform**, and if you want to programmatically determine which Button control is clicked, we can make use of **Command** event, along with **CommandName**and **CommandArgument**properties. Command event, makes it possible to have a single event handler method responding to the click event of multiple buttons. The command event, CommandName and CommandArgument properties are extremely useful when working with data-bound controls like Repeater, GridView, DataList. We will discuss about Repeater, GridView, and DataList in a later video session.

**Let's understand this with an example.** Consider the HTML below. Here we have 4 buttons. Notice that all the button controls have the same command event handler method - **oncommand="CommandButton\_Click"**. Also, notice the **CommandName**and **CommandArgument** properties. We will later use these properties, in the code behind to determine which button is clicked.

<asp:Button ID="PrintButton" runat="server" Text="Print" oncommand="CommandButton\_Click" CommandName="Print"/>  
  
<asp:Button ID="DeletButton" runat="server" Text="Delete" oncommand="CommandButton\_Click" CommandName="Delete"/>

<asp:Button ID="Top10Button" runat="server" Text="Show Top 10 Employees" oncommand="CommandButton\_Click"   
    CommandName="Show" CommandArgument="Top10"/>  
  
<asp:Button ID="Bottom10Button" runat="server" Text="Show Bottom 10 Employees" oncommand="CommandButton\_Click"   
    CommandName="Show" CommandArgument="Bottom10"/>  
          
<asp:Label ID="OutputLabel" runat="server"></asp:Label>

**protected void CommandButton\_Click(object sender, CommandEventArgs e)**  
**{**  
**switch (e.CommandName)**  
**{**  
**case "Print":**  
**OutputLabel.Text = "You clicked Print Button";**  
**break;**  
**case "Delete":**  
**OutputLabel.Text = "You clicked Delete Button";**  
**break;**  
**case "Show":**  
**if (e.CommandArgument.ToString() == "Top10")**  
**{**  
**OutputLabel.Text = "You clicked Show Top 10 Employees Button";**  
**}**  
**else**  
**{**  
**OutputLabel.Text = "You clicked Show Bottom 10 Employees Button";**  
**}**  
**break;**  
**default:**  
**OutputLabel.Text = "We don't know which button you clicked";**  
**break;**  
**}**  
**}**  
  
**Note:**All the 3 button controls - Button, LinkButton and ImageButton, expose Command event, the CommandName and CommandArgument properties.