

ASP.NET - EVENT HANDLING

An event is an action or occurrence such as a mouse click, a key press, mouse movements, or any system-generated notification. A process communicates through events. For example, interrupts are system-generated events. When events occur, the application should be able to respond to it and manage it.

Events in ASP.NET raised at the client machine, and handled at the server machine. For example, a user clicks a button displayed in the browser. A Click event is raised. The browser handles this client-side event by posting it to the server.

The server has a subroutine describing what to do when the event is raised; it is called the event-handler. Therefore, when the event message is transmitted to the server, it checks whether the Click event has an associated event handler. If it has, the event handler is executed.

Event Arguments

ASP.NET event handlers generally take two parameters and return void. The first parameter represents the object raising the event and the second parameter is event argument.

The general syntax of an event is:

```
private void EventName (object sender, EventArgs e);
```

Application and Session Events

The most important application events are:

- **Application_Start** - It is raised when the application/website is started.
- **Application_End** - It is raised when the application/website is stopped.

Similarly, the most used Session events are:

- **Session_Start** - It is raised when a user first requests a page from the application.
- **Session_End** - It is raised when the session ends.

Page and Control Events

Common page and control events are:

- **DataBinding** - It is raised when a control binds to a data source.
- **Disposed** - It is raised when the page or the control is released.
- **Error** - It is a page event, occurs when an unhandled exception is thrown.
- **Init** - It is raised when the page or the control is initialized.
- **Load** - It is raised when the page or a control is loaded.
- **PreRender** - It is raised when the page or the control is to be rendered.
- **Unload** - It is raised when the page or control is unloaded from memory.

Event Handling Using Controls

All ASP.NET controls are implemented as classes, and they have events which are fired when a user performs a certain action on them. For example, when a user clicks a button the 'Click' event is generated. For handling events, there are in-built attributes and event handlers. Event handler is coded to respond to an event, and take appropriate action on it.

By default, Visual Studio creates an event handler by including a Handles clause on the Sub procedure. This clause names the control and event that the procedure handles.

The ASP tag for a button control:

```
<asp:Button ID="btnCancel" runat="server" Text="Cancel" />
```

The event handler for the Click event:

```
Protected Sub btnCancel_Click(ByVal sender As Object, ByVal e As System.EventArgs)  
    Handles btnCancel.Click  
End Sub
```

An event can also be coded without Handles clause. Then, the handler must be named according to the appropriate event attribute of the control.

The ASP tag for a button control:

```
<asp:Button ID="btnCancel" runat="server" Text="Cancel" OnClick="btnCancel_Click" />
```

The event handler for the Click event:

```
Protected Sub btnCancel_Click(ByVal sender As Object, ByVal e As System.EventArgs)  
End Sub
```

The common control events are:

Event	Attribute	Controls
Click	OnClick	Button, image button, link button, image map
Command	OnCommand	Button, image button, link button
TextChanged	OnTextChanged	Text box
SelectedIndexChanged	OnSelectedIndexChanged	Drop-down list, list box, radio button list, check box list.
CheckedChanged	OnCheckedChanged	Check box, radio button

Some events cause the form to be posted back to the server immediately, these are called the postback events. For example, the click event such as, Button.Click.

Some events are not posted back to the server immediately, these are called non-postback events.

For example, the change events or selection events such as TextBox.TextChanged or CheckBox.CheckedChanged. The nonpostback events could be made to post back immediately by setting their AutoPostBack property to true.

Default Events

The default event for the Page object is Load event. Similarly, every control has a default event. For example, default event for the button control is the Click event.

The default event handler could be created in Visual Studio, just by double clicking the control in design view. The following table shows some of the default events for common controls:

Control	Default Event
AdRotator	AdCreated
BulletedList	Click
Button	Click
Calendar	SelectionChanged
CheckBox	CheckedChanged
CheckBoxList	SelectedIndexChanged
DataGrid	SelectedIndexChanged
DataList	SelectedIndexChanged
DropDownList	SelectedIndexChanged
HyperLink	Click
ImageButton	Click
ImageMap	Click
LinkButton	Click
ListBox	SelectedIndexChanged
Menu	MenuItemClick
RadioButton	CheckedChanged
RadioButtonList	SelectedIndexChanged

Example

This example includes a simple page with a label control and a button control on it. As the page events such as Page_Load, Page_Init, Page_PreRender etc. take place, it sends a message, which is displayed by the label control. When the button is clicked, the Button_Click event is raised and that also sends a message to be displayed on the label.

Create a new website and drag a label control and a button control on it from the control tool box. Using the properties window, set the IDs of the controls as .lblmessage. and .btnclick. respectively. Set the Text property of the Button control as 'Click'.

The markup file .aspx:

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="Default.aspx.cs"
    Inherits="eventdemo._Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >

    <head runat="server">
        <title>Untitled Page</title>
    </head>

    <body>
        <form >
            <div>
                <asp:Label ID="lblmessage" runat="server" >

                </asp:Label>
            </div>
        </form >
    </body>
</html>
```

```

        <br />
        <br />
        <br />
        <asp:Button ID="btnclick" runat="server" Text="Click"
onclick="btnclick_Click" />
    </div>
</form>
</body>

</html>

```

Double click on the design view to move to the code behind file. The Page_Load event is automatically created without any code in it. Write down the following self-explanatory code lines:

```

using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;

using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;

using System.Xml.Linq;

namespace eventdemo {

    public partial class _Default : System.Web.UI.Page {

        protected void Page_Load(object sender, EventArgs e) {
            lblmessage.Text += "Page load event handled. <br />";

            if (Page.IsPostBack) {
                lblmessage.Text += "Page post back event handled.<br/>";
            }
        }

        protected void Page_Init(object sender, EventArgs e) {
            lblmessage.Text += "Page initialization event handled.<br/>";
        }

        protected void Page_PreRender(object sender, EventArgs e) {
            lblmessage.Text += "Page prerender event handled. <br/>";
        }

        protected void btnclick_Click(object sender, EventArgs e) {
            lblmessage.Text += "Button click event handled. <br/>";
        }
    }
}

```

Execute the page. The label shows page load, page initialization and, the page pre-render events. Click the button to see effect:



Page post back event handled.
Button click event handled.
Page prerender event handled.

Click

Loading [Mathjax]/jax/output/HTML-CSS/jax.js