C# Corner

ASP.NET TAGS IN WEB FORMS

THE COMPLETE GUIDE

Sangeetha S

Sasikala S

ASP.NET Tags in Web Forms:

A Complete Guide

Sangeetha S and Sasikala S

Author – Sangeetha S and Sasikala S
Publisher – C# Corner
Editorial Team – Deepak Tewatia, Baibhav Kumar
Publishing Team – Praveen Kumar
Promotional & Media – Rohit Tomar

Author Bio

Sangeetha S has 6+ years' Experience in the software profession and a strong passion for application development, sharing knowledge, and writing articles, blogs, and books with knowledge of software development technologies such as PHP, Java, front-end HTML, CSS, jQuery, and JavaScript.
Also experienced in .NET MVC, .NET Core, ASP.NET Web Forms, C#, and VB. Net. Interest in Android app development and Creating Web application development. Skilled with working experience in a software development solution provider and creating sustainable applications with customer stratification.
Furthermore, Oracle certified in Java, C# Corner MVP, Coursera certifications in Android development and Django Web framework Certifications, 3+ Python Certifications, and two AI (Artificial Intelligence) Certifications in developing technologies.

Co-Author Bio

Sasikala has been working as a Software Developer for the past 6 years in Dot Net languages like C#, VB, Python, and Java. She is passionate about software development and sound, and she has primarily worked in ASP.NET WEB and desktop applications. She is also interested in analyzing and providing solutions for problem statements in a coding environment.
She is involved in sharing knowledge through reading and writing articles, Blogs, and Following Forums like C# Corner. She has recently completed her Coursera Certifications in 4 Python and AI Courses, and she was certified as a Java 8 Oracle developer in 2018.

Table of Contents:

1

Introduction to ASP.NET Web Forms & Tags

Overview

In this chapter, we explore ASP.NET Web Forms, covering its foundational concepts and the use of ASP tags in building web applications. Prepare yourself for a practical journey that will enable you to leverage the .NET Framework's tools and libraries to create web apps with rich user interfaces and advanced features.

ASP Tags in Web Forms

ASP.NET Web Forms are used to create a Dynamic Web page, which can manage data with interactive web designs. Usually, web applications are designed using HTML and CSS, along with some libraries based on their usage. In the case of Web Forms, it's different. Instead of using html tags here, we use ASP tags.

What are ASP Tags?

In ASP.NET Web Forms, ASPX tags are server-side control elements, and these tags are processed on the server and rendered as HTML on the client side. Also, these Tags are written in the .aspx markup pages. Here, we will discuss the most commonly used ASPX tags and tasp: Labele.

ASPX Tags have some attributes which will be their common to all Tags and common usages.

- **ID:** Sets the ID to an element that will be unique. Help in referencing the element in server-side code and client-side scripts.
- **runat:** Indicates that the element is a server-side control. Ensure the element is processed on the server.
- **Text:** Sets the text displayed on the element. Commonly used with controls like buttons and labels.
- **CssClass:** CSS classes apply for styling the elements. Helps in applying consistent styles across elements.
- **Enabled:** Determines whether a control is interactive.it has values true (default) or false. When set to false, the control is displayed but is not interactive (e.g., a button cannot be clicked, a textbox cannot be edited).
- **Visible:** Determines whether a control is rendered on the page. it has values true (default) or false. When set to false, the control is not rendered in the HTML output, and it takes up no space on the page.

ASP: Button

When the button is clicked, trigger signal events on the server.

```
<asp:Button
    ID="btnid1"
    Text="Button1"
    OnClick="Button1_Click"
    CssClass="btn btn-danger btn-sm"
    Enabled="true"
    Visible="true"
    CausesValidation="false"
    CommandName="MyCommand"
    CommandArgument="MyArgument"
    PostBackUrl="~/AnotherPage.aspx"
    ToolTip="Click this button"
    AccessKey="B"
    TabIndex="1"
    runat="server"
/>
```

Attributes:

- **OnClick:** The event handler that is triggered when the button is clicked.
- **CausesValidation:** Determines whether clicking the button triggers validation.
- **CommandName:** Used to specify a command name that can be used in event handling.
- **CommandArgument:** Used to specify an argument that can be passed to the event handler.
- **PostBackUrl:** Specifies the URL to which the form is posted when the button is clicked.
- **ToolTip:** The text displayed when the mouse pointer hovers over the button.
- **AccessKey:** Specifies a shortcut key to navigate to the button quickly.
- **TabIndex:** Specifies the tab order of the button within the form.

ASP:Textbox :

This Control allows users to provide input text.

```
<asp:TextBox
    ID="Textbox1"
    CssClass="form-control"
    TextMode="SingleLine"
    MaxLength="20"
    ReadOnly="false"
    ToolTip="Enter your text here"
    AutoPostBack="true"
    Columns="30"
    Placeholder="Enter text"
    TabIndex="1"
    ValidationGroup="Group1"
    runat="server"
/>
```

Attributes:

- **TextMode:** Specifies the type of text box (e.g., SingleLine, MultiLine, Password).
- **MaxLength:** Specifies the maximum number of characters allowed in the textbox.
- **ReadOnly:** Specifies whether the textbox is read-only.
- **ToolTip:** The text displayed when the mouse pointer hovers over the textbox.
- **AutoPostBack:** Specifies whether the textbox should post back to the server when its content changes.
- **Columns:** Specifies the width of the textbox in characters.
- **Rows:** Specifies the height of the textbox in lines (used with TextMode="MultiLine").
- **Placeholder:** Specifies a short hint that describes the expected value of the input field.
- **TabIndex:** Specifies the tab order of the textbox within the form.
- **ValidationGroup:** Specifies the group of controls for which validation is performed when the textbox causes a postback.

ASP:Label

This Displays static text or dynamic data on web page.

Attributes: ID, Text, CssClass, Visible

```
<asp:Label
    runat="server"
    AssociatedControlID="Textbox1">
    Label
</asp:Label>

<asp:TextBox
    ID="Textbox1"
    runat="server">
</asp:TextBox>
```

Here AssociatedControlID is used to link the label to another control on the page. By clicking the label will move the focus to the associated control, making it easier for users to interact with forms.

ASP:CheckBox

Attributes: ID, Text, Checked, CssClass, Enabled, Visible

- **Usage:** Represents a checkbox that can be checked or unchecked.

```
<asp:CheckBox
    ID="CheckBox1"
    runat="server"
    Text="Accept Terms and Conditions"
/>
```

ASP:RadioButton

This represents a radio button that can be selected.

Attributes: ID, Text, GroupName, Checked, CssClass, Enabled, Visible

```
<asp:RadioButton
    ID="RadioButton1"
    runat="server"
    GroupName="Options"
    Text="Option 1"
/>

<asp:RadioButton
    ID="RadioButton2"
    runat="server"
    GroupName="Options"
    Text="Option 2"
/>
```

By using GroupName attributes each radio button can be grouped together.

ASP:DropDownList

This control is typically used to provide a dropdown list of items for users to select from. It can be populated with data from a database, a collection, or static items defined in the markup.

```
<asp:DropDownList
    ID="DropDownList1"
    DataSource="<%# YourDataSource %>"
    DataTextField="Name"
    DataValueField="ID"
    SelectedValue="1"
    CssClass="form-control"
    Enabled="true"
    Visible="true"
    AppendDataBoundItems="true"
    AutoPostBack="true"
    ToolTip="Select an item"
    TabIndex="1"
    ValidationGroup="Group1"
    runat="server">

    <asp:ListItem Text="Select an option" Value="" />

</asp:DropDownList>
```

Attributes:

- **DataSource:** The data source from which the dropdown list is populated.
- **DataTextField:** The field from the data source to display as the text of the list items.
- **DataValueField:** The field from the data source to use as the value of the list items.
- **SelectedValue:** The value of the currently selected item.
- **AutoPostBack:** Specifies whether the dropdown list should post back to the server when its selected value changes.
- **ToolTip:** The text displayed when the mouse pointer hovers over the dropdown list.
- **TabIndex:** Specifies the tab order of the dropdown list within the form.
- **ValidationGroup:** Specifies the group of controls for which validation is performed when the dropdown list causes a postback.
- **Items:** Collection of list items that can be added statically.

ASP:LinkButton

The LinkButton control creates a hyperlink-style button that can post back to the server.

Attributes:

- **Text:** The text displayed on the button.
- **CommandName:** The name of the command to be executed.
- **CommandArgument:** An argument passed to the command.
- **OnClick:** Event handler for the click event.
- **PostBackUrl:** URL to navigate to after the button is clicked.
- **CssClass:** CSS class for styling.
- **Enabled:** Indicates whether the button is enabled.

- **ToolTip:** Text displayed when the mouse hovers over the button.
- **Visible:** Indicates whether the button is visible.

```
<asp:LinkButton
    ID="LinkButton1"
    runat="server"
    Text="Click Me"
    OnClick="LinkButton1_Click"
    CssClass="link-button"
    Enabled="true"
    ToolTip="Click to perform action">
</asp:LinkButton>
```

Example:

This code sample uses all the tags explained above to create a form where users can enter their details. Additionally, the LinkButton tag uses the PostBackUrl attribute, which allows navigation to the next page upon clicking the button. Similarly, the asp:Button tag performs the same action using the OnClick attribute.

ASPTags.aspx

```
<%@ Page
    Language="vb"
    AutoEventWireup="false"
    CodeBehind="ASPTags.aspx.vb"
    Inherits="ASPWEBFORMS.ASPTags"
%>
```

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body style="background-color: lightgray; display: flex; align-items:
center; justify-content: center;">
    <form id="form1" runat="server">

        <asp:LinkButton ID="lnlbtn1" Text=">>> Home" runat="server"
PostBackUrl="~/Default.aspx"></asp:LinkButton>

        <h2>
            <span style="color: #ff00dc;">Please Fill the
Details</span>
        </h2>

        <div class="body-content">

            <div class="row">
                <asp:Label runat="server"
AssociatedControlID="Textbox1">Name</asp:Label>
```

```
                <asp:TextBox ID="Textbox1"
runat="server"></asp:TextBox>
            </div>

            <div class="row">
                <asp:Label runat="server">Gender</asp:Label>
                <asp:RadioButton ID="RadioButton1" runat="server"
GroupName="Options" Text="Male" />
                <asp:RadioButton ID="RadioButton2" runat="server"
GroupName="Options" Text="Female" />
            </div>
            <br />

            <div class="row">
                <asp:Label runat="server"
AssociatedControlID="CheckBox1"></asp:Label>
                <asp:CheckBox ID="CheckBox1" runat="server" Text="I AM
not a Robot" />
            </div>

            <div class="row">
                <asp:Label runat="server" Text="Are you Interested in
Coding....?" AssociatedControlID="DropDownList1"></asp:Label>
                <br />
                <asp:DropDownList ID="DropDownList1" CssClass="form-
control" runat="server">
                    <asp:ListItem Text="Select an option" Value="" />
                    <asp:ListItem Text="Yes" Value="Yes" />
                    <asp:ListItem Text="No" Value="No" />
                </asp:DropDownList>
            </div>
            <br />

            <asp:Button ID="btnid1" Text="Submit"
                        OnClick="Button1_Click"
                        CssClass="btn btn-danger btn-sm"
                        Enabled="true"
                        Visible="true"
                        CausesValidation="false"
                        CommandName="MyCommand"
                        CommandArgument="MyArgument"
                        PostBackUrl="~/AspTagsPage2.aspx"
                        ToolTip="Click this button"
                        AccessKey="B"
                        TabIndex="1"
                        runat="server" />
        </div>
```

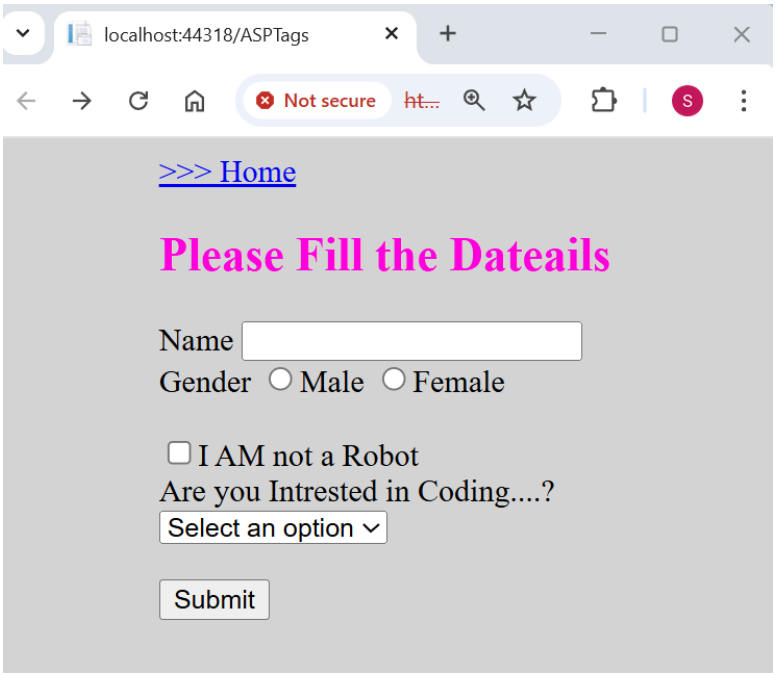
```
    </form>
</body>
</html>
```

ASPTags.aspx.vs

The server-side code for the Page_Load and Button_Click event functions is shown in the ASPTags.aspx.vs. This code primarily loads the ASP tags in a format that is viewable by the user.

```
Public Class ASPTags
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        ' Your Page_Load code here
    End Sub

    Protected Sub Button1_Click(ByVal sender As Object, ByVal e As
EventArgs)
        ' Your Button1_Click code here
    End Sub

End Class
```

Output



2

Data Populating ASP Tags

Overview

In this chapter, we discuss how to populate data in tabular, list, or Excel-like formats to enhance readability for users. ASP.NET Web Forms offers a variety of controls, such as ListView, DataList, and GridView, that make it easy to present structured data in a clean and organized manner.

ASP:ListView

The control is used to display data in a highly customizable format. It allows for detailed control over the layout and appearance of the data. This is ideal for scenarios where need to display data with a custom layout, such as a grid, list, or any other format. suitable for complex layouts.

```
<asp:ListView
    ID="ListView1"
    runat="server"
    DataSource="<%# YourDataSource %>">

    <LayoutTemplate>
        <div class="listview-layout">
            <asp:PlaceHolder
                runat="server"
                ID="itemPlaceholder">
            </asp:PlaceHolder>8
        </div>
    </LayoutTemplate>

    <ItemTemplate>
        <div class="listview-item">
            <%# Eval("Name") %>
        </div>
    </ItemTemplate>

    <EmptyDataTemplate>
        <div>No data available.</div>
    </EmptyDataTemplate>

</asp:ListView>
```

Attributes:

- **ID:** Unique identifier for the control.
- **DataSource:** The data source from which the ListView is populated.
- **ItemTemplate:** Defines the template for each item in the ListView.
- **LayoutTemplate:** Defines the overall layout of the ListView.
- **EmptyDataTemplate:** Defines the template to display when there is no data.

Sample Code

```
<%@ Page Language="vb" AutoEventWireup="false"
CodeBehind="AspTagsPage2.aspx.vb" Inherits="ASPWEBFORMS.AspTagsPage2"
%>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>AspTagsPage2</title>
    <style type="text/css">
        .listview-table {
```

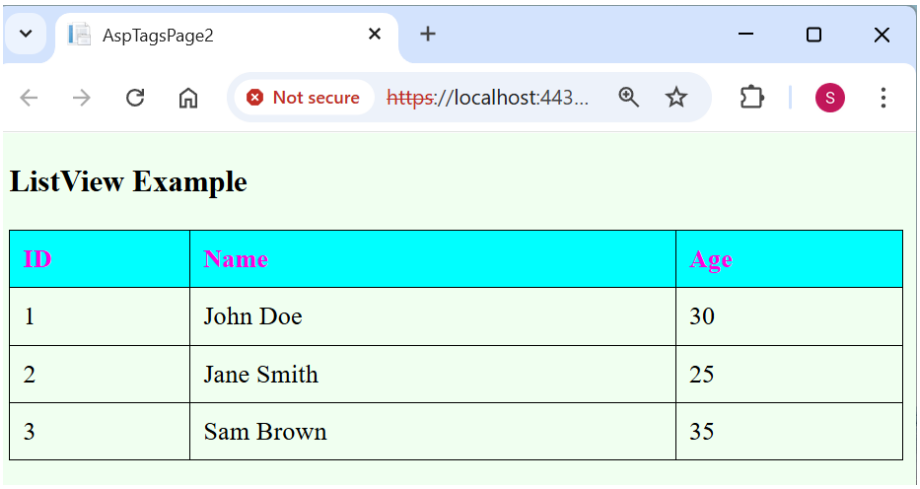
```
            width: 100%;
            border-collapse: collapse;
        }

        .listview-table th, .listview-table td {
            border: 1px solid #000;
            padding: 8px;
            text-align: left;
        }

        .listview-table th {
            background-color: #0ff;
            color: #ff00dc;
        }
    </style>
</head>
<body style="background-color:honeydew;">
    <form id="form1" runat="server">
        <h3>ListView Example</h3>

        <asp:ListView ID="ListView1" runat="server">
            <LayoutTemplate>
                <table class="listview-table">
                    <tr>
                        <th>ID</th>
                        <th>Name</th>
                        <th>Age</th>
                    </tr>
                    <asp:PlaceHolder ID="itemPlaceholder"
runat="server"></asp:PlaceHolder>
                </table>
            </LayoutTemplate>

            <ItemTemplate>
                <tr>
                    <td><%# Eval("ID") %></td>
                    <td><%# Eval("Name") %></td>
                    <td><%# Eval("Age") %></td>
                </tr>
            </ItemTemplate>
        </asp:ListView>
    </form>
</body>
</html>
```

Server-Side Code

```
Imports System
Imports System.Web.UI
```

```
Partial Public Class AspTagsPage2
    Inherits Page

    Protected Sub Page_Load(sender As Object, e As EventArgs) Handles
Me.Load
        BindDataToListView()
    End Sub

    Private Sub BindDataToListView()
        Dim dt As New DataTable()
        dt.Columns.Add("ID")
        dt.Columns.Add("Name")
        dt.Columns.Add("Age")

        ' Add rows to the DataTable
        dt.Rows.Add("1", "John Doe", "30")
        dt.Rows.Add("2", "Jane Smith", "25")
        dt.Rows.Add("3", "Sam Brown", "35")

        ' Bind the DataTable to the ListView
        ListView1.DataSource = dt
        ListView1.DataBind()
    End Sub

End Class
```

Output:



ASP:DataList

The <asp:DataList> control is used to display data in a repeated list format. It is simpler than the ListView and is suitable for scenarios where you need to display data in a tabular or list format.

```
<asp:DataList
    ID="DataList1"
    runat="server"
    DataSource="<%# YourDataSource %>"
```

```
    RepeatDirection="Vertical"
    RepeatColumns="1">

    <ItemTemplate>
        <div class="datalist-item">
            <%# Eval("Name") %>
        </div>
    </ItemTemplate>

</asp:DataList>
```

Attributes:

- **ID:** Unique identifier for the control.
- **DataSource:** The data source from which the DataList is populated.
- **ItemTemplate:** Defines the template for each item in the DataList.
- **RepeatDirection:** Specifies the direction in which items are repeated (Horizontal or Vertical).
- **RepeatColumns:** Specifies the number of columns to display when items are repeated horizontally.

Sample Code:

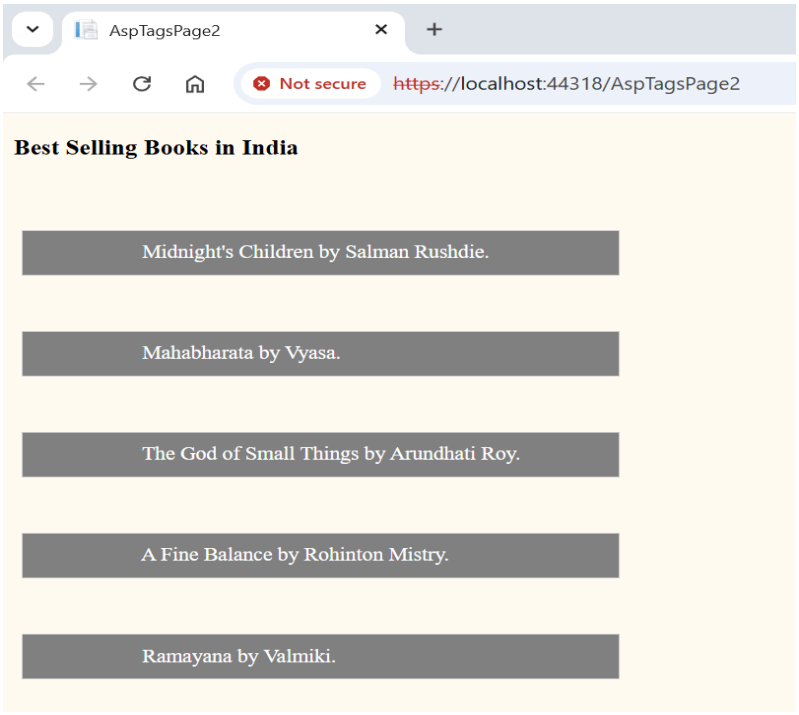
```
<%@ Page Language="vb" AutoEventWireup="false"
CodeBehind="AspTagsPage2.aspx.vb" Inherits="ASPWEBFORMS.AspTagsPage2"
%>
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>AspTagsPage2</title>
    <style type="text/css">
        .datalist-item {
            border: 1px solid #ccc;
            padding: 10px;
            margin: 5px;
            background-color: #808080;
            color: white;
        }
    </style>
</head>
<body style="background-color:floralwhite;">
    <form id="form1" runat="server">
        <h3>Best Selling Books in India</h3>

        <asp:DataList
            ID="DataList1"
            runat="server"
            RepeatDirection="Vertical"
            RepeatColumns="1">

            <ItemTemplate>
```

```
                <div class="datalist-item">
                    <%# Eval("Name") %>
                </div>
            </ItemTemplate>

        </asp:DataList>

    </form>
</body>
</html>
```

Server-Side Code

```
Imports System
Imports System.Web.UI

Partial Public Class AspTagsPage2
    Inherits Page

    Protected Sub Page_Load(sender As Object, e As EventArgs) Handles
Me.Load
        If Not IsPostBack Then
            Dim data As New List(Of Item) From {
                New Item With {.Name = "Midnight's Children by Salman
Rushdie."},
                New Item With {.Name = "Mahabharata by Vyasa."},
                New Item With {.Name = "The God of Small Things by
Arundhati Roy."},
                New Item With {.Name = "A Fine Balance by Rohinton
Mistry."},
                New Item With {.Name = "Ramayana by Valmiki."}
            }

            DataList1.DataSource = data
            DataList1.DataBind()
        End If
    End Sub

    Public Class Item
        Public Property Name As String
    End Class
End Class
```

Output:



ASP:GridView

GridView is a powerful and flexible control used to display, sort, and manage tabular data in a web application. It allows to bind data from various data sources, such as databases, and provides features like paging, sorting, and editing.

```
<asp:GridView
    ID="GridView1"
    runat="server"
    AutoGenerateColumns="True">
</asp:GridView>
```

ASP:BoundField

The <asp:BoundField> is used to display the value of a data field in a data-bound control like GridView or DetailsView. It automatically binds to a specific field in the data source and displays its value.

Attributes:

- **DataField:** Specifies the name of the data field to bind to.
- **HeaderText:** Sets the text for the header of the column.
- **SortExpression:** Enables sorting based on the field.
- **DataFormatString:** Formats the data displayed in the field.
- **HtmlEncode:** Indicates whether to HTML encode the field's value.

```
<asp:GridView
    ID="GridView1"
    runat="server"
    AutoGenerateColumns="False">
```

```
    <Columns>
        <asp:BoundField DataField="Name" HeaderText="Name" />
        <asp:BoundField DataField="Age" HeaderText="Age"
DataFormatString="{0:N0}" />
    </Columns>

</asp:GridView>
```

Sample Code:

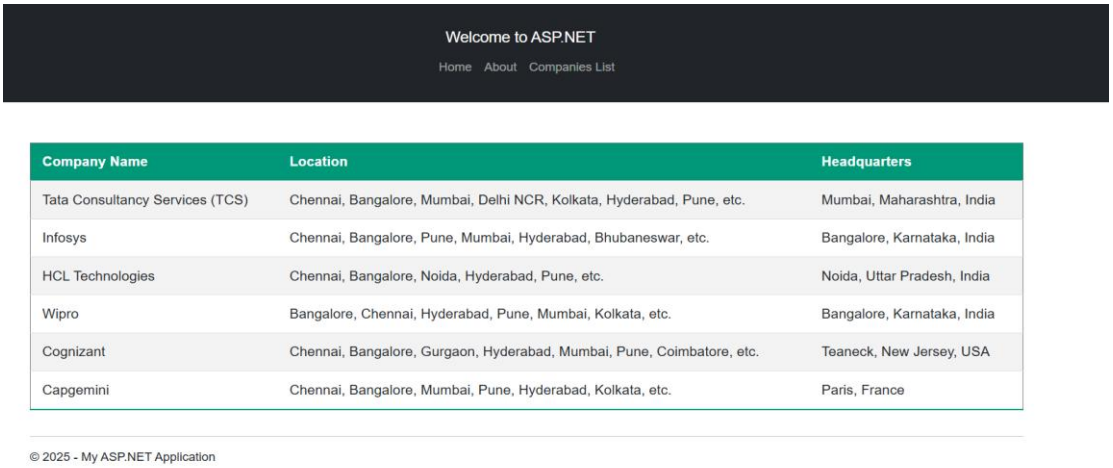
```
<%@ Page Title="Contact" Language="VB" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Contact.aspx.vb"
Inherits="WebApplication1.Contact" %>

<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent"
runat="server">

    <style>
        .gridview {
            width: 100%;
            border-collapse: collapse;
            margin: 20px 0;
            font-size: 18px;
            text-align: left;
        }

        .gridview th, .gridview td {
            padding: 12px 15px;
        }

        .gridview thead tr {
            background-color: #009879;
            color: #ffffff;
            text-align: left;
            font-weight: bold;
        }

        .gridview tbody tr {
            border-bottom: 1px solid #dddddd;
        }

        .gridview tbody tr:nth-of-type(even) {
            background-color: #f3f3f3;
        }

        .gridview tbody tr:last-of-type {
            border-bottom: 2px solid #009879;
        }
```

```
        .gridview tbody tr:hover {
            background-color: #f1f1f1;
        }

        .gridview-column {
            padding: 10px;
        }

        .gridview-header {
            background-color: #009879;
            color: white;
            padding: 10px;
        }
    </style>

    <asp:GridView
        ID="GridView1"
        runat="server"
        AutoGenerateColumns="False"
        CssClass="gridview">

        <Columns>
            <asp:BoundField
                DataField="Company Name"
                HeaderText="Company Name"
                ItemStyle-CssClass="gridview-column"
                HeaderStyle-CssClass="gridview-header" />

            <asp:BoundField
                DataField="Location"
                HeaderText="Location"
                ItemStyle-CssClass="gridview-column"
                HeaderStyle-CssClass="gridview-header" />

            <asp:BoundField
                DataField="Headquarters"
                HeaderText="Headquarters"
                ItemStyle-CssClass="gridview-column"
                HeaderStyle-CssClass="gridview-header" />
        </Columns>

    </asp:GridView>

</asp:Content>
```

Server-Side Code

```
Public Class ASPTags

    Public Class Contact
        Inherits Page

        Protected Sub Page_Load(ByVal sender As Object, ByVal e As
EventArgs) Handles Me.Load
            BindGridView()
        End Sub

        Private Sub BindGridView()
            ' Create a DataTable and add columns
            Dim dt As New DataTable()
            dt.Columns.Add("Company Name")
            dt.Columns.Add("Location")
            dt.Columns.Add("Headquarters")

            ' Add rows to the DataTable
            dt.Rows.Add("Tata Consultancy Services (TCS)", "Chennai,
Bangalore, Mumbai, Delhi NCR, Kolkata, Hyderabad, Pune, etc.", "Mumbai,
Maharashtra, India")
            dt.Rows.Add("Infosys", "Chennai, Bangalore, Pune, Mumbai,
Hyderabad, Bhubaneswar, etc.", "Bangalore, Karnataka, India")
            dt.Rows.Add("HCL Technologies", "Chennai, Bangalore, Noida,
Hyderabad, Pune, etc.", "Noida, Uttar Pradesh, India")
            dt.Rows.Add("Wipro", "Bangalore, Chennai, Hyderabad, Pune,
Mumbai, Kolkata, etc.", "Bangalore, Karnataka, India")
            dt.Rows.Add("Cognizant", "Chennai, Bangalore, Gurgaon,
Hyderabad, Mumbai, Pune, Coimbatore, etc.", "Teaneck, New Jersey, USA")
            dt.Rows.Add("Capgemini", "Chennai, Bangalore, Mumbai, Pune,
Hyderabad, Kolkata, etc.", "Paris, France")

            ' Bind the DataTable to the GridView
            GridView1.DataSource = dt
            GridView1.DataBind()
        End Sub

    End Class

End Class
```

Output:



ASP:TemplateField

The <asp:TemplateField> provides more flexibility by allowing you to define custom templates for displaying and editing data. You can use various controls within the templates to create complex layouts.

Attributes:

- **ItemTemplate:** Defines the template for displaying data items.
- **EditItemTemplate:** Defines the template for editing data items.
- **HeaderTemplate:** Defines the template for the header.
- **FooterTemplate:** Defines the template for the footer.

```
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False">
    <Columns>
        <asp:TemplateField HeaderText="Name">
            <ItemTemplate>
                <asp:Label ID="Label1" runat="server" Text='<%#
Eval("Name") %>' />
            </ItemTemplate>
            <EditItemTemplate>
                <asp:TextBox ID="TextBox1" runat="server" Text='<%#
Bind("Name") %>' />
            </EditItemTemplate>
        </asp:TemplateField>
        <asp:TemplateField HeaderText="Age">
            <ItemTemplate>
                <asp:Label ID="Label2" runat="server" Text='<%#
Eval("Age") %>' />
            </ItemTemplate>
            <EditItemTemplate>
                <asp:TextBox ID="TextBox2" runat="server" Text='<%#
Bind("Age") %>' />
            </EditItemTemplate>
        </asp:TemplateField>
    </Columns>
</asp:GridView>
```

Sample Code:

```
<%@ Page Title="Contact" Language="VB" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Contact.aspx.vb"
Inherits="WebApplication1.Contact" %>
<asp:Content ID="BodyContent" ContentPlaceHolderID="MainContent"
runat="server">
    <style>
        .gridview {
            width: 100%;
            border-collapse: collapse;
            margin: 20px 0;
            font-size: 18px;
            text-align: left;
        }
        .gridview th, .gridview td {
            padding: 12px 15px;
        }
        .gridview thead tr {
            background-color: #009879;
            color: #ffffff;
            text-align: left;
            font-weight: bold;
        }
        .gridview tbody tr {
            border-bottom: 1px solid #dddddd;
        }
        .gridview tbody tr:nth-of-type(even) {
            background-color: #f3f3f3;
        }
        .gridview tbody tr:last-of-type {
            border-bottom: 2px solid #009879;
        }
        .gridview tbody tr:hover {
            background-color: #f1f1f1;
        }
        .gridview-column {
            padding: 10px;
        }
        .gridview-header {
            background-color: #009879;
            color: white;
            padding: 10px;
        }
    </style>

    <div class="divcontent">
        <asp:GridView ID="GridView2" runat="server" CssClass="gridview"
AutoGenerateColumns="False"
```

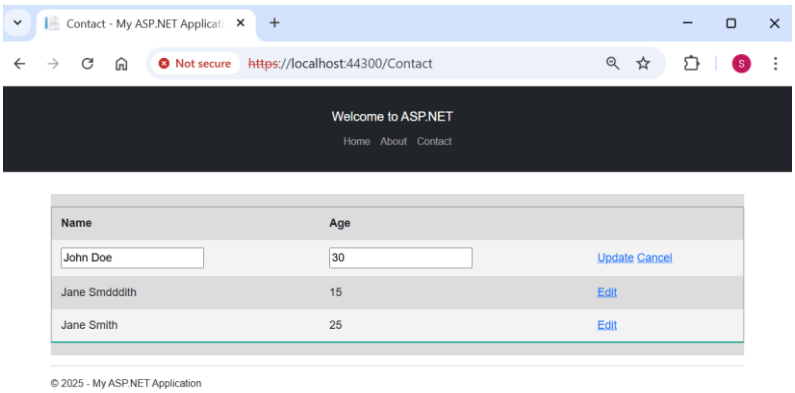
```
            OnRowEditing="GridView1_RowEditing"
            OnRowUpdating="GridView1_RowUpdating"
            OnRowCancelingEdit="GridView1_RowCancelingEdit">
            <Columns>
                <asp:TemplateField HeaderText="Name">
                    <ItemTemplate>
                        <asp:Label ID="Label1" runat="server" Text='<%#
Eval("Name") %>' />
                    </ItemTemplate>
                    <EditItemTemplate>
                        <asp:TextBox ID="TextBox1" runat="server"
Text='<%# Bind("Name") %>' />
                    </EditItemTemplate>
                </asp:TemplateField>
                <asp:TemplateField HeaderText="Age">
                    <ItemTemplate>
                        <asp:Label ID="Label2" runat="server" Text='<%#
Eval("Age") %>' />
                    </ItemTemplate>
                    <EditItemTemplate>
                        <asp:TextBox ID="TextBox2" runat="server"
Text='<%# Bind("Age") %>' />
                    </EditItemTemplate>
                </asp:TemplateField>
                <asp:CommandField ShowEditButton="True" />
            </Columns>
        </asp:GridView>
    </div>
</asp:Content>
```

Server-Side Code

```
Public Class Contact
    Inherits Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
EventArgs) Handles Me.Load
        If Not IsPostBack Then
            BindGridView1()
        End If
    End Sub

    Private Sub BindGridView1()
        Dim data As New List(Of Person) From {
            New Person With {.Name = "John Doe", .Age = 30},
            New Person With {.Name = "Jane Smdddith", .Age = 15},
            New Person With {.Name = "Jane Smith", .Age = 25}
        }
        GridView2.DataSource = data
```

```
        GridView2.DataBind()
    End Sub

    Protected Sub GridView1_RowEditing(ByVal sender As Object, ByVal e
As GridViewEditEventArgs)
        GridView2.EditIndex = e.NewEditIndex
        BindGridView1()
    End Sub

    Protected Sub GridView1_RowUpdating(ByVal sender As Object, ByVal e
As GridViewUpdateEventArgs)
        Dim row As GridViewRow = GridView2.Rows(e.RowIndex)
        Dim name As String = CType(row.FindControl("TextBox1"),
TextBox).Text
        Dim age As Integer =
Integer.Parse(CType(row.FindControl("TextBox2"), TextBox).Text)

        Dim data As New List(Of Person) From {
            New Person With {.Name = name, .Age = age}
        }
        GridView2.EditIndex = -1
        GridView2.DataSource = data
        GridView2.DataBind()
    End Sub

    Protected Sub GridView1_RowCancelingEdit(ByVal sender As Object,
ByVal e As GridViewCancelEditEventArgs)
        GridView2.EditIndex = -1
        BindGridView1()
    End Sub

    Public Class Person
        Public Property Name As String
        Public Property Age As Integer
    End Class
End Class
```

Output



The <asp:TemplateField> is particularly useful when you need to customize the display or editing experience beyond what <asp:BoundField> can offer.

ASP:Repeater

This control is used to display a repeated list of items that bind data from database or collections. This is useful for creating dynamic lists, tables. Repeater can be nested within other controls or repeaters. It supports various templates (ItemTemplate, HeaderTemplate, FooterTemplate, etc.), which can be used to define the structure and style of the repeated items.

Attributes:

- **ID:** Unique identifier for the control.
- **DataSource:** Source of data for the repeater.
- **ItemTemplate:** Template for displaying each item in the list.
- **HeaderTemplate:** Template for the header section.
- **FooterTemplate:** Template for the footer section.

```
<asp:Repeater ID="Repeater1" runat="server">
    <HeaderTemplate>
        <h2>Product List</h2>
    </HeaderTemplate>

    <ItemTemplate>
        <div class="product-item">
            <%# Eval("ProductName") %> - <%# Eval("Price", "{0:C}") %>
        </div>
    </ItemTemplate>

    <FooterTemplate>
        <p>Total Products: <%# Container.DataItemCount %></p>
    </FooterTemplate>
</asp:Repeater>
```

Sample Code:

```
Protected Sub Page_Load(sender As Object, e As EventArgs) Handles
Me.Load
    If Not IsPostBack Then
        Dim products As New List(Of Product) From {
            New Product With {.ProductName = "Product 1", .Price =
19.99D, .AvailableinNumbers = 10},
            New Product With {.ProductName = "Product 2", .Price =
29.99D, .AvailableinNumbers = 5},
            New Product With {.ProductName = "Product 3", .Price =
9.99D, .AvailableinNumbers = 20}
        }

        Repeater1.DataSource = products
        Repeater1.DataBind()
    End If
End Sub
```

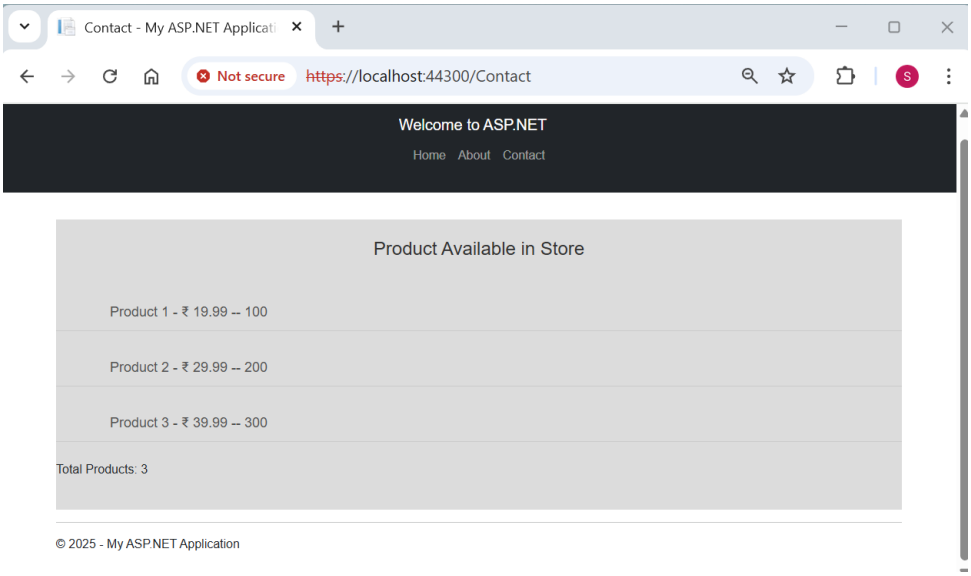
```
Protected Sub Repeater1_ItemDataBound(sender As Object, e As
RepeaterItemEventArgs) Handles Repeater1.ItemDataBound
    If e.Item.ItemType = ListItemType.Footer Then
        Dim lblTotal As Label =
CType(e.Item.FindControl("lblTotalProducts"), Label)
        If lblTotal IsNot Nothing Then
            lblTotal.Text = Repeater1.Items.Count.ToString()
        End If
    End If
End Sub

Public Class Product
    Public Property ProductName As String
    Public Property Price As Decimal
    Public Property AvailableinNumbers As Integer
End Class
```

Server-Side Code

```
Public Class Contact
    Inherits Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
EventArgs) Handles Me.Load
        If Not IsPostBack Then
            BindRepeater()
        End If
    End Sub

    Private Sub BindRepeater()
        Dim dt As New DataTable()
        dt.Columns.Add("ProductName", GetType(String))
        dt.Columns.Add("Price", GetType(Decimal))
        dt.Columns.Add("AvailableinNumbers", GetType(Integer))
        dt.Rows.Add("Product 1", 19.99D, 100)
        dt.Rows.Add("Product 2", 29.99D, 200)
        dt.Rows.Add("Product 3", 39.99D, 300)

        Repeater1.DataSource = dt
        Repeater1.DataBind()
    End Sub

    Protected Sub Repeater1_ItemDataBound(sender As Object, e As
RepeaterItemEventArgs) Handles Repeater1.ItemDataBound
        If e.Item.ItemType = ListItemType.Footer Then
            Dim lblTotalProducts As Label =
CType(e.Item.FindControl("lblTotalProducts"), Label)
            If lblTotalProducts IsNot Nothing Then
```

```
                lblTotalProducts.Text =
Repeater1.Items.Count.ToString()
            End If
        End If
    End Sub

End Class
```

Output



3

Some of Other ASP Tags

Overview

In this chapter, we explore control, action, and event-handling tags in ASP.NET Web Forms, including features like regular expressions and file upload events. These event handler tags play a crucial role in creating responsive and interactive client-side components within web applications.

ASP:CaptchaControl

The CaptchaControl is used to add CAPTCHA functionality to your web forms to prevent automated submissions.

Attributes:

- **CaptchaWidth:** Width of the CAPTCHA image.
- **CaptchaHeight:** Height of the CAPTCHA image.
- **CaptchaLength:** Number of characters in the CAPTCHA.
- **CaptchaFont:** Font used for the CAPTCHA text.
- **CaptchaBackgroundColor:** Background color of the CAPTCHA image.
- **CaptchaTextColor:** Color of the CAPTCHA text.
- **CaptchaNoise:** Level of noise in the CAPTCHA image.
- **CaptchaLineNoise:** Level of line noise in the CAPTCHA image.

```
<asp:CaptchaControl ID="CaptchaControl1"
    runat="server"
    CaptchaWidth="200"
    CaptchaHeight="50"
    CaptchaLength="6"
    CaptchaFont="Arial"
    CaptchaBackgroundColor="White"
    CaptchaTextColor="Black"
    CaptchaNoise="Medium"
    CaptchaLineNoise="Low">
</asp:CaptchaControl>
```

ASP:RegularExpressionValidator

This control allows to validate the input of another control such as textbox, dropdown based on the id given in ControlToValidate.

Attributes:

- **ControlToValidate:** The ID of the control to validate.
- **ValidationExpression:** The regular expression to use for validation.
- **ErrorMessage:** The error message to display if validation fails.
- **Display:** How the error message is displayed (None, Static, Dynamic).
- **CssClass:** CSS class for styling the error message.
- **Enabled:** Indicates whether the validator is enabled.
- **InitialValue:** The initial value of the control to validate.
- **SetFocusOnError:** Indicates whether to set focus on the control when validation fails.

```
<asp:RegularExpressionValidator
    ID="RegexValidator1"
    runat="server"
    ControlToValidate="TextBox1"
    ValidationExpression="^\d{4}$"
    ErrorMessage="Please enter a 4-digit number."
    Display="Dynamic"
    CssClass="error-message"
    Enabled="true"
```

```
    SetFocusOnError="true">
</asp:RegularExpressionValidator>
```

ASP:CalendarExtender

CalendarExtender control is part of the AJAX Control Toolkit and extends a Textbox control to display a calendar for date selection.

Attributes:

- **TargetControlID:** The ID of the TextBox control to extend.
- **Format:** The date format to use (e.g., MM/dd/yyyy).
- **PopupButtonID:** The ID of the button that triggers the calendar popup.
- **CssClass:** CSS class for styling the calendar.
- **Enabled:** Indicates whether the extender is enabled.
- **StartDate:** The earliest date selectable in the calendar.
- **EndDate:** The latest date is selectable in the calendar.

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>

<ajaxToolkit:CalendarExtender
    ID="CalendarExtender1"
    runat="server"
    TargetControlID="TextBox1"
    Format="MM/dd/yyyy"
    PopupButtonID="PopupButton1"
    CssClass="calendar-extender"
    Enabled="true"
    StartDate="01/01/2020"
    EndDate="12/31/2025">
</ajaxToolkit:CalendarExtender>
```

ASP:FileUpload:

This is used to upload files to the server, which can upload one to many files.

Attributes:

- **FileName:** The name of the uploaded file.
- **HasFile:** Indicates whether a file has been uploaded.
- **PostedFile:** Provides access to the uploaded file.
- **ToolTip:** Text displayed when the mouse hovers over the control.
- **AllowMultiple:** Allows multiple file upload.

```
<asp:FileUpload
    ID="FileUpload1"
    runat="server"
    CssClass="file-upload"
    Enabled="true"
    ToolTip="Upload your file here"
    Visible="true"
    AllowMultiple="true">
</asp:FileUpload>
```

ASP:Calendar

The asp:Calendar control is used to display a calendar on a webpage, allowing users to select dates. It provides various attributes to customize its appearance and behaviours.

Attributes:

- **ID:** Unique identifier for the control.
- **SelectedDate:** The date that is currently selected.
- **VisibleDate:** The month and year that the calendar displays.
- **DayStyle, WeekendDayStyle, TodayDayStyle:** CSS styles for different types of days.
- **OnSelectionChanged:** Event handler for when the selected date changes.

```
<asp:Calendar
    ID="Calendar1"
    runat="server"
    SelectedDate="2025-03-27"
    VisibleDate="2025-03"
    DayStyle-CssClass="day-class"
    WeekendDayStyle-CssClass="weekend-class"
    TodayDayStyle-CssClass="today-class"
    OnSelectionChanged="Calendar1_SelectionChanged" />
```

ASP:HyperLink

This is used to create hyperlinks that navigate to other pages or resources. When clicking on hyper link it directly hits the URL to web page.

Attributes:

- **ID:** Unique identifier for the control.
- **NavigateUrl:** URL to which the hyperlink points.
- **Text:** Text displayed for the hyperlink.
- **CssClass:** CSS class for styling the hyperlink.
- **Visible:** Boolean value indicating whether the control is visible.

```
<asp:HyperLink
    ID="HyperLink1"
    NavigateUrl="https://www.google.com"
    Text="Visit Example"
    CssClass="link-class"
    Visible="true"
    runat="server" />
```

This creates a hyperlink that navigates to "https://www.example.com" with the text "Visit Example".

ASP:Image

asp:Image control is useful when some picture, or images to be shown on a webpage.it can be logo of company, some symbols, etc.

Attributes:

- **ID:** Unique identifier for the control.

- **ImageUrl:** URL of the image to be displayed.
- **AlternateText:** Text displayed if the image cannot be loaded.
- **CssClass:** CSS class for styling the image.
- **Visible:** Boolean value indicating whether the control is visible.

```
<asp:Image
    ID="Image1"
    ImageUrl="~/images/logo.png"
    AlternateText="Company Logo"
    CssClass="image-class"
    Visible="true"
    runat="server" />
```

This displays an image located at "~/images/logo.png" with the alternate text "Company Logo".

In below code snippet some ASP tags like RegularExpressionValidator, FileUpload, Image tags are explained.Here RegularExpressionValidator Asp tag can be used with other elements like, Textbox, dropdown which will validate the user input is correct or not and show the indicator that if it's not valid input. Here with email id Textbox is added with RegularExpressionValidator by using ControlToValidate attribute.

Sample Code:

```
<%@ Page Title="Home Page" Language="VB" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Default.aspx.vb"
Inherits="WebApplication1._Default" %>

<asp:Content ID="Content1" ContentPlaceHolderID="MainContent"
runat="server">
    <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css" />

    <main>
        <form id="form1" runat="server">
            <!-- Email Input with Validation -->
            <asp:TextBox ID="txtEmail" runat="server"></asp:TextBox>
            <asp:RegularExpressionValidator
                ID="revEmail"
                runat="server"
                ControlToValidate="txtEmail"
                ErrorMessage="Please enter a valid email address."
                ValidationExpression="^\w+@[a-zA-Z_]+?\.[a-zA-Z]{2,3}$"
                ForeColor="Red">
            </asp:RegularExpressionValidator>
            <br /><br />

            <!-- File Upload Control -->
            <asp:FileUpload
                ID="FileUpload1"
                runat="server"
```

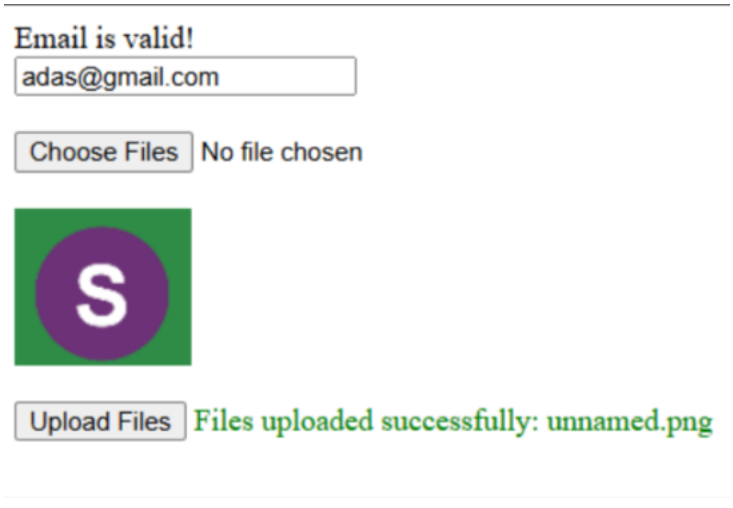
```
                CssClass="file-upload"
                Enabled="true"
                ToolTip="Upload your file here"
                Visible="true"
                AllowMultiple="true" />
            <br /><br />

            <!-- Company Logo -->
            <asp:Image
                ID="Image1"
                runat="server"
                ImageUrl="~/images/im.png"
                AlternateText="Company Logo"
                CssClass="image-class"
                Visible="true" />
            <br /><br />

            <!-- Upload Button and Message Label -->
            <asp:Button
                ID="btnUpload"
                runat="server"
                Text="Upload Files"
                OnClick="btnUpload_Click" />
            <asp:Label
                ID="lblMessage"
                runat="server"
                ForeColor="Green">
            </asp:Label>
        </form>
    </main>
</asp:Content>
```

Server-Side Code

```
Imports System.IO

Public Class _Default
    Inherits Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
EventArgs) Handles Me.Load
        ' You can add any page load logic here if needed
    End Sub

    Protected Sub btnUpload_Click(sender As Object, e As EventArgs)
Handles btnUpload.Click
        If FileUpload1.HasFiles Then
            Dim uploadPath As String = Server.MapPath("~/Uploads/")
            Dim uploadedFiles As New List(Of String)
```

```
            For Each file As HttpPostedFile In FileUpload1.PostedFiles
                Dim fileName As String =
Path.GetFileName(file.FileName)
                Dim fullPath As String = Path.Combine(uploadPath,
fileName)
                file.SaveAs(fullPath)
                uploadedFiles.Add(fileName)
            Next

            lblMessage.Text = "Files uploaded successfully: " &
String.Join(", ", uploadedFiles)
            lblMessage.ForeColor = System.Drawing.Color.Green
        Else
            lblMessage.Text = "Please select at least one file to
upload."
            lblMessage.ForeColor = System.Drawing.Color.Red
        End If

        If Page.IsValid Then
            ' Proceed with processing after validation
            Response.Write("Email is valid!")
        End If
    End Sub

End Class
```

Output:



4

Main Content Populating ASP Tags

Overview

In this chapter, we explore special ASP.NET Web Forms tags such as Content and UpdatePanel, which are used to integrate server-side and client-side code within HTML markup. These tags help in building a more interactive and dynamic user interface, enhancing the overall user experience in web applications.

ASP:Panel

This is a kind of container which can have multiple other controls like textbox, button, checkbox. Will be useful to design a form.

Attributes:

- **ID:** Unique identifier for the control.
- **CssClass:** CSS class for styling the panel.
- **Visible:** Boolean value indicating whether the control is visible.
- **Usage:** This control acts as a container for other controls.

```
<asp:Panel ID="Panel1" CssClass="panel-class" Visible="true"
runat="server">
    <asp:Label ID="Label1" Text="Hello, World!" runat="server" />
    <asp:Button ID="Button1" Text="Click Me" runat="server" />
</asp:Panel>
```

ASP:Content:

This control defines content that will be placed in a ContentPlaceHolder on a master page. This allows us to create a consistent layout across multiple pages.

ASP:UpdatePanel:

This control allows us to perform partial page updates; it means only a portion of the page is refreshed instead of the whole page. This can improve the user experience by making the page more responsive.

ASP:UpdateProgress:

This control provides feedback to the user during asynchronous postbacks, such as showing a loading indicator while an UpdatePanel is being updated.

ASP:ModalPopupExtender:

This control is part of the AJAX Control Toolkit and is used to display a modal dialog box. It can be used to create pop-up dialogs that require user interaction before they can return to the main page. Here's asp:Content, asp:UpdatePanel, asp:UpdateProgress, and asp:ModalPopupExtender all together in an ASP.NET Web Forms application.

Example:

Master Page (Site.Master)

```
<%@ Master Language="VB" AutoEventWireup="true"
CodeBehind="Site.master.vb" Inherits="ASPWEBFORMS.SiteMaster" %>
<!DOCTYPE html>
<html>
<head runat="server">
    <title></title>
    <asp:ContentPlaceHolder ID="head"
runat="server"></asp:ContentPlaceHolder>
</head>
```

```
<body style="background-color: lightblue; align-items: center; justify-
content: center;">
    <form id="form1" runat="server">
        <asp:ScriptManager ID="ScriptManager1"
runat="server"></asp:ScriptManager>
        <asp:ContentPlaceHolder ID="MainContent"
runat="server"></asp:ContentPlaceHolder>
    </form>
</body>
</html>
```

Default Page (Default.aspx)

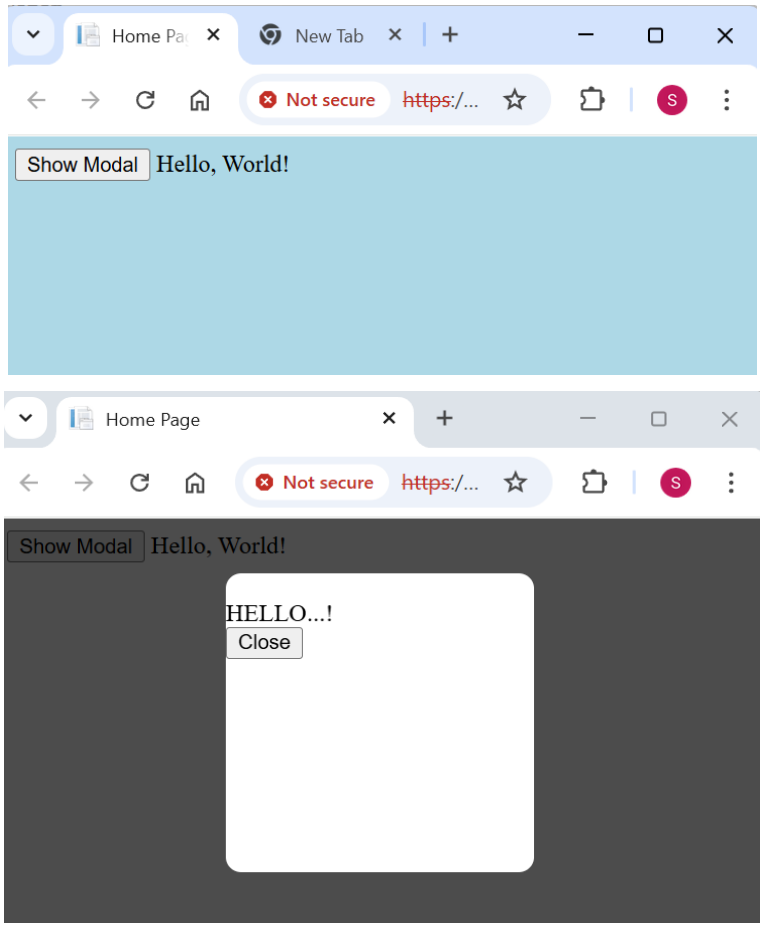
```
<%@ Page Title="Home Page" Language="VB" MasterPageFile="~/Site.Master"
AutoEventWireup="true" CodeBehind="Default.aspx.vb"
Inherits="ASPWEBFORMS._Default" %>
<%@ Register Assembly="AjaxControlToolkit"
Namespace="AjaxControlToolkit" TagPrefix="asp" %>

<asp:Content ID="Content1" ContentPlaceHolderID="head" runat="server">
    <style>
        .modalBackground {
            background-color: #000;
            filter: alpha(opacity=70);
            opacity: 0.7;
        }

        .modalPopup {
            border: 5px medium black;
            border-radius: 10px;
            background-color: #FFFFFF;
            width: 200px;
            max-height: 200px;
            height: 100%;
            overflow: auto;
        }

        .divWaiting {
            position: absolute;
            background-color: #FAFAFA;
            z-index: 2147483647 !important;
            opacity: 0.8;
            overflow: hidden;
            text-align: center;
            top: 0;
            left: 0;
            height: 150%;
            width: 100%;
            padding-top: 20%;
```

```
        }
    </style>
</asp:Content>

<asp:Content ID="Content2" ContentPlaceHolderID="MainContent"
runat="server">
    <asp:UpdatePanel ID="UpdatePanel1" runat="server">
        <ContentTemplate>

            <asp:Button ID="btnShowModal" runat="server" Text="Show
Modal" OnClick="btnShowModal_Click" />

            <asp:Label ID="lblMessage" runat="server" Text="Hello,
World!" />

            <asp:UpdateProgress ID="UpdateProgress1" runat="server"
AssociatedUpdatePanelID="UpdatePanel1">
                <ProgressTemplate>
                    <div class="divWaiting">
                        <img src="loading.gif" alt="Loading..." />
                    </div>
                </ProgressTemplate>
            </asp:UpdateProgress>

            <asp:ModalPopupExtender
                ID="ModalPopupExtender1"
                runat="server"
                TargetControlID="btnShowModal"
                PopupControlID="pnlModal"
                BackgroundCssClass="modalBackground" />

            <asp:Panel ID="pnlModal" runat="server"
CssClass="modalPopup" style="display: none;">
                <br />
                <asp:Label ID="lab1" runat="server"
Text="HELLO...!"></asp:Label>
                <br />
                <asp:Button ID="btnClose" runat="server" Text="Close"
OnClick="btnClose_Click" />
            </asp:Panel>

        </ContentTemplate>
    </asp:UpdatePanel>
</asp:Content>
```

Code-Behind (Default.aspx.vb)

```
Public Class _Default
    Inherits Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
EventArgs) Handles Me.Load
        ' Optional: Add any logic needed on page load
    End Sub

    Protected Sub btnShowModal_Click(ByVal sender As Object, ByVal e As
EventArgs)
        ModalPopupExtender1.Show()
    End Sub

    Protected Sub btnClose_Click(ByVal sender As Object, ByVal e As
EventArgs)
        ModalPopupExtender1.Hide()
    End Sub

End Class
```

Output



Explanation

- **Master Page (Site.Master):** Defines the layout with ContentPlaceHolder controls for the head and main content.
- **Content Page (Default.aspx):** Uses asp:Content to fill the placeholders in the master page.
- **asp:UpdatePanel:** contains a button and a label to demonstrate partial page updates.
- **asp:UpdateProgress:** Shows a loading indicator during asynchronous postbacks.
- asp:Panel acts as the modal dialog, which is shown/hidden using asp:ModalPopupExtender.

Conclusion

With this, we conclude our discussion on ASP tags. I hope we have covered most of the essential ASPX tags. By utilizing these tags, we can create dynamic and robust web applications.

This content will be useful for .net web app developer and asp .net learner. Also, it’s efficient reference for basic understanding and helpful for fast development.

C# Corner

OUR MISSION

Free Education is Our Basic Need! Our mission is to empower millions of developers worldwide by providing the latest unbiased news, advice, and tools for learning, sharing, and career growth. We're passionate about nurturing the next young generation and help them not only to become great programmers, but also exceptional human beings.

ABOUT US

CSharp Inc, headquartered in Philadelphia, PA, is an online global community of software developers. C# Corner served 29.4 million visitors in year 2022. We publish the latest news and articles on cutting-edge software development topics. Developers share their knowledge and connect via content, forums, and chapters. Thousands of members benefit from our monthly events, webinars, and conferences. All conferences are managed under Global Tech Conferences, a CSharp Inc sister company. We also provide tools for career growth such as career advice, resume writing, training, certifications, books and white-papers, and videos. We also connect developers with their potential employers via our Job board. Visit C# Corner

MORE BOOKS