ASP.NET

Appendix

Contents

- ASP.NET server controls
- Web Parts
- Data binding
- Web.config elements

ASP.NET Server Controls

ASP.NET Server Controls Advantages

- The ability to have the page automatically maintain the state of the control
- ASP.NET detects the level of the target browser, the appropriate HTML is generated for each browser
- The use of a compiled language instead of an interpreted script results in better performance
- The ability to bind to a data source
- Events can be raised by controls on the browser and easily handled by code on the server

Categories of ASP.NET Server Controls

- Basic controls
- Validation controls
- Data source controls
- Data view controls
- Login and security controls
- Master pages
- Rich controls

Label

- The **`Label`** server control is used to display text in the browser
- The text can be dynamically altered from server-side code
- The **`AccessKey`** attribute can be used to set a hot-key (an accelerator key) for an assigned server control
  - The user must use the Alt+<key> combination

```
<asp:Label ID="Label1" runat="server" AccessKey="N"
           AssociatedControlID="Textbox1">
    User<u>n</u>ame
</asp:Label>
<asp:TextBox ID="TextBox2" runat="server"></asp:TextBox>
```

Literal

- The **Literal** server control is similar to the **Label** control
  - The difference is that the **Label** control is rendered using the **** tag
- The **Mode** attribute allows to dictate how the text assigned to the control is interpreted by the ASP.NET engine

```
<asp:Literal ID="Literal1" Runat="server"
             Mode="Encode"
             Text="<b>Here is some text</b>">
</asp:Literal>
```

```
HTML source:
&lt;b&gt;Here is some text&lt;/b&gt;
```

```
Browser's output:
<b>Here is some text</b>
```

TextBox

- The **`TextBox`** control is mapped to one of three HTML elements:
  - A standard HTML text box

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

  - A text box for a password

```
<asp:TextBox ID="TextBox2" Runat="server" TextMode="Password" />
```

  - A multiline text box

```
<asp:TextBox ID="TextBox3" Runat="server" TextMode="MultiLine"
             Width="300px" Height="150px" />
```

- The **`Focus()`** method allows to dynamically place the end user's cursor in an appointed form element

TextBox cont.

- The **`OnTextChanged`** event is triggered when the user moves the cursor focus outside the text box

```
<asp:TextBox ID="TextBox1" Runat="server" AutoPostBack="True"
             OnTextChanged="TextBox1_TextChanged" />
```

```
protected void TextBox1_TextChanged(object sender, EventArgs e)
{
    Response.Write("OnTextChanged event triggered");
}
```

- The **`AutoCompleteType`** attribute allows to apply the auto-completion feature
  - A value of this attribute specifies the category of user data
  - Not all browser support this feature

Button

- Setting the **`CausesValidation`** property to **`false`** is a way to use a button that will not fire the validation process
- It is possible to have multiple buttons on a form working from a single event

```
<asp:Button ID="Button1" Runat="server" Text="Button 1"
    OnCommand="Button_Command" CommandName="DoSomething1" />
<asp:Button ID="Button2" Runat="server" Text="Button 2"
    OnCommand="Button_Command" CommandName="DoSomething2" />
```

```
protected void Button_Command(Object sender, CommandEventArgs e) {
    switch (e.CommandName) {
        case "DoSomething1":
            Response.Write("Button 1 was selected");
            break;
        case "DoSomething2":
            // ...
            break;
    }
}
```

Button cont.

- It is possible to specify both the client-side side and server-side event to fire by pushing one button
  - The postback to the server will occur after running client-side event

```
<script runat="server">
    protected void Button_Click(object sender, EventArgs e) {
        Response.Write("Postback!");
    }
</script>

<script language="javascript">
    function AlertHello() {
        alert('Hello ASP.NET');
    }
</script>

<asp:Button ID="Button" Runat="server" Text="Button"
    OnClientClick="AlertHello()" OnClick="Button_Click" />
```

LinkButton

- The **LinkButton** control is a button that looks like a typical hyperlink

```
<asp:LinkButton ID="LinkButton1"
                runat="server"
                onclick="LinkButton1_Click">
    The LinkButton control
</asp:LinkButton>
```

ImageButton

- The **ImageButton** control is a button rendered as an image

```
<asp:ImageButton ID="ImageButton1" runat="server"
    ImageUrl="sample.jpg"
    onclick="ImageButton1_Click" />
<asp:Literal ID="Literal4" runat="server"></asp:Literal>
```

- This control takes a different construction for the **OnClick** event
  - It allows to get coordinates of the clicked point

```
protected void ImageButton1_Click(
    object sender, ImageClickEventArgs e)
{
    Literal4.Text = string.Format("Click at ({0}, {1})", e.X, e.Y);
}
```

HyperLink

- The **`HyperLink`** server control allows to programmatically work with any hyperlinks on the Web page
- It is a great way to dynamically place hyperlinks on a Web page

```
<asp:HyperLink ID="HyperLink1" runat="server"
               NavigateUrl="http://www.google.com">
    HyperLink to google
</asp:HyperLink>
```

- It is possible to use an image as a link

```
<asp:HyperLink ID="HyperLink2" runat="server"
               NavigateUrl="http://www.google.com"
               ImageUrl="sample.gif" Target="_blank" />
```

DropDownList

- The select box generated by the **`DropDownList`** control displays a single item and allows the user to make a selection from a larger list of items
- When the **`AutoPostBack`** property is set to **`true`** and a selection is made, then the method provided through the **`OnSelectedIndexChanged`** event is fired

```
<asp:DropDownList ID="DropDownList1" runat="server">
    <asp:ListItem>First</asp:ListItem>
    <asp:ListItem>Second</asp:ListItem>
    <asp:ListItem>Third</asp:ListItem>
</asp:DropDownList>
```

ListBox

- The **SelectionMode** attribute allows users to make multiple selections

```
<asp:ListBox ID="ListBox1" runat="server" SelectionMode="Multiple">
    <asp:ListItem>First</asp:ListItem>
    <asp:ListItem>Second</asp:ListItem>
    <asp:ListItem>Third</asp:ListItem>
</asp:ListBox>
<asp:Button ID="Button4" runat="server" Text="Check selected items"
    onclick="Button4_Click" />
<asp:Literal ID="Literal5" runat="server" />
```

```
protected void Button4_Click(object sender, EventArgs e)
{
    StringBuilder sb = new StringBuilder("Selected: ");
    foreach (int i in ListBox1.GetSelectedIndices())
    {
        sb.AppendFormat("{0} ", ListBox1.Items[i].Text);
    }
    Literal5.Text = sb.ToString();
}
```

CheckBox

- The **`TextAlign`** property can be used to realign the text
- The **`AutoPostBack`** property is available

```
<asp:CheckBox ID="CheckBox1" runat="server" AutoPostBack="True"
    oncheckedchanged="CheckBox1_CheckedChanged"
    Text="This is a CheckBox"
    TextAlign="Left" />
```

```
protected void CheckBox1_CheckedChanged(object sender, EventArgs e)
{
    CheckBox1.Text += " - checked!";
}
```

CheckBoxList

- The **`CheckBoxList`** control allows to take multiple check boxes and create specific events for the entire group
- This control can be bound to a data source
- Formatting properties: **`RepeatDirection`**, **`RepeatColumns`**, **`BorderColor`**, **`BorderStyle`**, **`BorderWidth`**

```
<asp:CheckBoxList ID="CheckBoxList1" runat="server" />
```

```
string[] items = new string[]
    { "First", "Second", "Third", "Fourth", "Fifth" };
CheckBoxList1.DataSource = items;
CheckBoxList1.DataBind();
```

RadioButton

- The **`GroupName`** property enables the radio buttons on the Web page to work together

```
<table>
    <tr>
        <td>
            <asp:RadioButton ID="RadioButton1" runat="server"
                GroupName="Group1" Text="RadioButton no 1"/><br />
            <asp:RadioButton ID="RadioButton2" runat="server"
                GroupName="Group2" Text="RadioButton no 2"/>
        </td>
        <td>
            <asp:RadioButton ID="RadioButton3" runat="server"
                GroupName="Group1" Text="RadioButton no 3"/><br />
            <asp:RadioButton ID="RadioButton4" runat="server"
                GroupName="Group2" Text="RadioButton no 4"/>
        </td>
    </tr>
</table>
```

RadioButtonList

- Similar to the **CheckBoxList** control

```
<asp:RadioButtonList ID="RadioButtonList1" runat="server"
                     CellPadding="10" CellSpacing="10"
                     RepeatDirection="Horizontal" RepeatColumns="2">
    <asp:ListItem>First</asp:ListItem>
    <asp:ListItem>Second</asp:ListItem>
    <asp:ListItem>Third</asp:ListItem>
    <asp:ListItem>Fourth</asp:ListItem>
    <asp:ListItem>Fifth</asp:ListItem>
    <asp:ListItem>Sixth</asp:ListItem>
</asp:RadioButtonList>
```

BulletedList

- The **`BulletedList`** server control is rendered as a bulleted list of items
  - An ordered (using the HTML **`<ol>`** element) or unordered (using the HTML **`<ul>`** element) list can be produced
- Formatting options:
  - **`BulletStyle`** - **`Numbered, LowerAlpha, UpperAlpha, LowerRoman, UpperRoman, Disc, Circle, Square, NotSet, CustomImage`**
  - **`FirstBulletNumber`**
  - **`DisplayMode – Text, HyperLink, LinkButton`**

BulletedList cont.

```
<asp:BulletedList ID="BulletedList1" runat="server"
BulletStyle="Numbered"
    DisplayMode="LinkButton" onclick="BulletedList1_Click">
    <asp:ListItem>First</asp:ListItem>
    <asp:ListItem>Second</asp:ListItem>
    <asp:ListItem>Third</asp:ListItem>
    <asp:ListItem>Fourth</asp:ListItem>
    <asp:ListItem>Fifth</asp:ListItem>
    <asp:ListItem>Sixth</asp:ListItem>
</asp:BulletedList>
<asp:Literal ID="Literal7" runat="server"></asp:Literal>
```

```
protected void BulletedList1_Click(object sender,
    BulletedListEventArgs e)
{
    Literal7.Text = "Clicked: " + BulletedList1.Items[e.Index].Text;
}
```

Image

- The **`Image`** control is rendered as a static image, which does not support handling user clicks
- The **`DescriptionUrl`** property can be used to render the **`longdesc`** attribute of the **`<img>`** element (but this attribute is poorly supported by browsers)

```
<asp:Image ID="Image1" runat="server" ImageUrl="sample.jpg" />
```

ImageMap

- It enables to turn an image into a navigation menu
- Available types of hotspots:
  - **RectangleHotSpot, CircleHotSpot, PolygonHotSpot**

```
<asp:ImageMap ID="ImageMap1" runat="server" HotSpotMode="PostBack"
        ImageUrl="SampleMap.jpg" onclick="ImageMap1_Click">
    <asp:CircleHotSpot HotSpotMode="PostBack"
        PostBackValue="theCircle"
        Radius="50" X="100" Y="50" AlternateText="postback!" />
    <asp:RectangleHotSpot Bottom="70" HotSpotMode="Navigate"
        Left="200" NavigateUrl="http://www.google.com"
        Right="300" Top="20" AlternateText="google" />
</asp:ImageMap>
<asp:Literal ID="Literal8" runat="server"></asp:Literal>
```

```
protected void ImageMap1_Click(object sender, ImageMapEventArgs e) {
    if (e.PostBackValue == "theCircle") {
        Literal8.Text = "Circle clicked!";
    }
}
```

ImageMap cont.

- Properties of the **`ImageMap`** control:
  - **`AlternateText`**
  - **`GenerateEmptyAlternateText`**
  - **`HotSpotMode – Inactive, Navigate, NotSet, PostBack`**
  - **`HotSpots`** – a collection of **`HotSpot`** objects
- Properties of **`HotSpot`** objects:
  - **`AlternateText`**
  - **`HotSpotMode`**
  - **`NavigateUrl`**
  - **`PostBackValue`** – only relevant if the **`HotSpotMode`** is set to **`PostBack`**
  - **`Target`**

Table

```
<asp:Table ID="Table1" runat="server">
    <asp:TableRow runat="server">
        <asp:TableCell runat="server">First</asp:TableCell>
        <asp:TableCell runat="server">Second</asp:TableCell>
        <asp:TableCell runat="server">Third</asp:TableCell>
    </asp:TableRow>
    <asp:TableRow runat="server">
        <asp:TableCell runat="server">Fourth</asp:TableCell>
        <asp:TableCell runat="server">Fifth</asp:TableCell>
        <asp:TableCell runat="server">Sixth</asp:TableCell>
    </asp:TableRow>
</asp:Table>
```

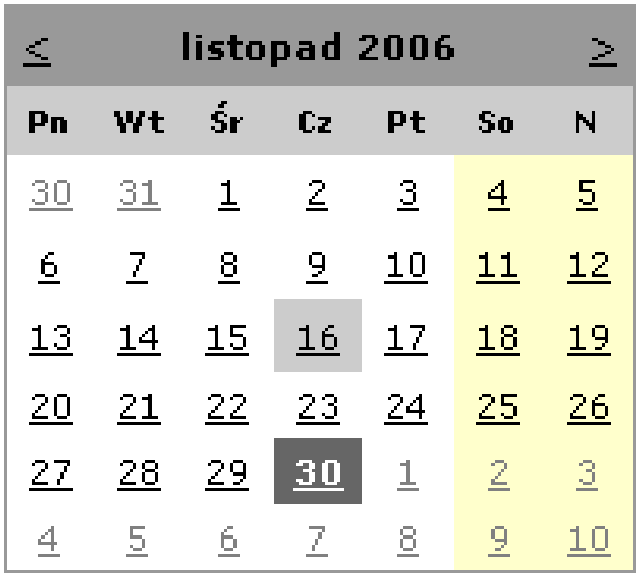
Table cont.

- Rows can be dynamically modified and added to the table

```
Table1.Rows[1].Cells.RemoveAt(2);
Table1.Rows[1].Cells[0].HorizontalAlign = HorizontalAlign.Center;
Table1.Rows[1].Cells[0].ColumnSpan = 2;

TableRow tr = new TableRow();
Table1.Rows.Add(tr);

TableCell tc = new TableCell();
tc.Text = "Seventh";
tr.Cells.Add(tc);

tc = new TableCell();
tc.Text = "Eight";
tr.Cells.Add(tc);

tc = new TableCell();
tc.Text = "Ninth";
tr.Cells.Add(tc);
```

Calendar

- Features of the **`Calendar`** control:
  - Displays a calendar showing a single month
  - Allows the user to select a day, week, or month
  - Allows the user to select a range of days
  - Allows the user to move to the next or previous month
  - Programmatically controls the display of specific days
  - It is customizable, with various properties and events



Calendar cont.

```
<asp:Calendar ID="Calendar1" runat="server" SelectedDate="2008-11-20"
      BackColor="White" BorderColor="#999999" CellPadding="4"
      DayNameFormat="Shortest" Font-Names="Verdana" Font-Size="8pt"
      ForeColor="Black" Height="180px" Width="200px"
      onselectionchanged="Calendar1_SelectionChanged">
    <SelectedDayStyle BackColor="#666666" Font-Bold="True" ForeColor="White" />
    <SelectorStyle BackColor="#CCCCCC" />
    <WeekendDayStyle BackColor="#FFFFCC" />
    <TodayDayStyle BackColor="#CCCCCC" ForeColor="Black" />
    <OtherMonthDayStyle ForeColor="#808080" />
    <NextPrevStyle VerticalAlign="Bottom" />
    <DayHeaderStyle BackColor="#CCCCCC" Font-Bold="True" Font-Size="7pt" />
    <TitleStyle BackColor="#999999" BorderColor="Black" Font-Bold="True" />
</asp:Calendar>
```

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
    Literal7.Text = "Selected day: " +
        Calendar1.SelectedDate.ToString();
}
```

AdRotator

- Displays a randomly selected advertisement banner on the Web page
- Advertisement information is stored in a separate XML file

```
<asp:AdRotator ID="AdRotator1" runat="server"
DataSourceID="XmlDataSource1" />

<asp:XmlDataSource ID="XmlDataSource1" runat="server"
    DataFile="~/App_Data/Ads.xml"></asp:XmlDataSource>
```

```
<?xml version="1.0" encoding="utf-8" ?>
<Advertisements>
    <Ad>
        <ImageUrl>~/Images/1st.jpg</ImageUrl>
        <NavigateUrl>http://www.1st.com</NavigateUrl>
        <AlternateText>1st</AlternateText>
        <Impressions>10</Impressions>
        <Keyword></Keyword>
    </Ad>
</Advertisements>
```

AdRotator cont.

- Useful properties:
  - **`AdvertisementFile`**
  - **`AlternateTextField`**, **`ImageUrlField`**, **`NavigateUrlFilter`** – names of elements from the advertisement file or data field in which the alternate text, image URL, or navigation URL respectively is stored
  - **`DataMember`**, **`DataSource`**, **`DatSourceID`**
  - **`KeywordFilter`**
  - **`Target`**
- The **`AdCreated`** event occurs once per round trip to the server after creation of the control, but before the page is rendered

FileUpload

```
<asp:FileUpload ID="FileUpload1" runat="server" /><br />
<asp:Button ID="Button1" runat="server" Text="Send the file"
    onclick="Button1_Click" /><br />
<asp:Literal ID="Literal8" runat="server"></asp:Literal>
```

```
protected void Button1_Click(object sender, EventArgs e) {
    if (FileUpload1.HasFile) {
        try {
            string destPath = Server.MapPath(
                "~/App_Data/" + FileUpload1.FileName);
            FileUpload1.SaveAs(destPath);
            Literal8.Text = string.Format(
                "File name: {0} <br>Size: {1} <br>Content type: {2}",
                FileUpload1.PostedFile.FileName,
                FileUpload1.PostedFile.ContentLength,
                FileUpload1.PostedFile.ContentType);
        }
        catch (Exception ex) {
            Literal8.Text = "ERROR: " + ex.Message.ToString();
        }
    }
    else {
        Literal8.Text = "No file to send.";
    }
}
```

FileUpload cont.

- This control puts an **<input type="file">** element on the Web page to enable the end user to upload files to the server
- The destination folder on the server has to be writable for the account used by ASP.NET
- The default size limitation for file upload is 4MB
  - It can be changed in the **Web.config** file
- The **FileUpload** control allows to place the content of the file into a **Stream** object

```
System.IO.Stream myStream = FileUpload1.FileContent;
```

Panel

- The **`Panel`** Web server control provides a container within the page for other controls
  - Controls put into the **`Panel`** control can be controlled as a unit
- Features of the **`Panel`** control:
  - Showing and hiding all contained controls
  - Changing attributes of all controls (e.g. the font)
  - Scrolling with scrollbars
  - Changing the horizontal alignment
  - Setting a background image

Panel cont.

- Useful properties:
  - **`BackImageUrl`** – if the image is smaller than the panel, it will be tiled
  - **`Direction`** – **`LeftToRight, RightToLeft, NotSet`**
  - **`GroupingText`** – causes the panel to render to the browser as the **`<fieldset>`** element rather than the **`<div>`** element
  - **`HorizontalAlign`** – **`Center, Justify, Left, Right, NotSet`**
  - **`ScrollBars`** – **`Auto, Both, Horizontal, Vertical, None`**
  - **`Wrap`** – if **`true`** (the default) the contents of the panel will wrap

MultiView and View

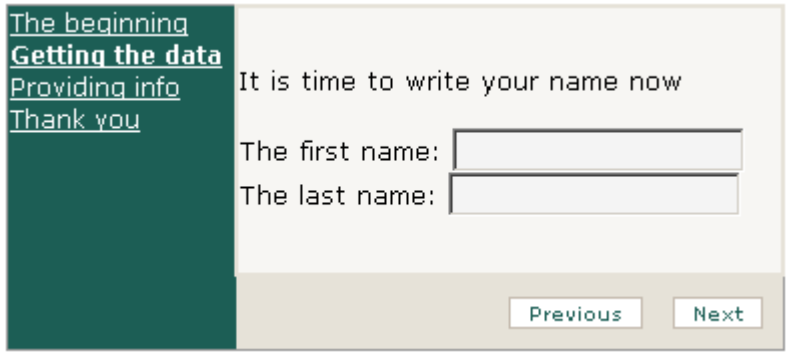
- The **`MultiView`** and **`View`** server controls work together to enable the capability to turn on/off sections of an ASP.NET page
  - It is similar to changing the visibility of **`Panel`** controls, but it is easier to manage

```
<asp:Button ID="Button3" runat="server" Text="Next view >>"
    onclick="Button3_Click" />
```

```
protected void Page_Load(object sender, EventArgs e) {
    if (!IsPostBack) {
        MultiView1.ActiveViewIndex = 0;
    }
}

protected void Button3_Click(object sender, EventArgs e) {
    if (MultiView1.ActiveViewIndex < MultiView1.Views.Count - 1) {
        MultiView1.ActiveViewIndex++;
    }
}
```

Wizard

- The **`Wizard`** control can be used when there is a need for step-by-step logic
- This control allows for a far greater degree of customization than does the **`MultiView`** control
- Available events:
  - **`ActiveStepChanged, CancelButtonClick, FinishButtonClick, NextButtonClick, PreviousButtonClick, SideBarButtonClick`**



PlaceHolder

- The `PlaceHolder` control is a placeholder to interject objects dynamically into the Web page
- This control does not produce any visible output and is only used as a container for other controls on the Web page
- The `Controls` collection can be used to add, insert, or remove a control

HiddenField

- The **`ValueChanged`** event is raised on postback when the **`Value`** property of the control is different from the previous posting

```
<asp:HiddenField ID="HiddenField1" runat="server" />
```

```
protected void Page_Load(object sender, EventArgs e)
{
    HiddenField1.Value = System.Guid.NewGuid().ToString();
}
```

```
<input type="hidden"
       name="HiddenField1"
       id="HiddenField1"
       value="a031e77c-379b-4b4a-887c-244ee69584d5" />
```

Xml

- The **`Xml`** server control provides a means of getting XML and transforming it using an XSL style sheet

```
<asp:Xml ID="Xml1" Runat="server"
         DocumentSource="~/MyXMLFile.xml"
         TransformSource="MyXSLFile.xslt">
</asp:Xml>
```

Validation Controls

- Validation controls provide an easy-to-use mechanism for all common types of standard validation plus ways to provide custom-written validation
- Validation controls allow to customize how error information is displayed to the user
- They can work with any controls put on the ASP.NET Web page, including both HTML and ASP.NET server controls

Client-Side and Server-Side Validation

- Client-side validation
  - Quick and responsive for the end user
  - If something is wrong with the form, using client-side validation ensures that the end user knows this as soon as possible
  - It can be easily hacked or disabled, because client-side validation code is written in JavaScript and is fully visible for the end user
- Server-side validation
  - Requires a postback to the server
  - Much slower, but definitely more secure
- The best approach is always to perform client-side validation first and then, after the form passes and is posted to the server, to perform the validation checks again using server-side validation

ASP.NET Validation Server Controls

- ASP.NET performs browser detection when generating the ASP.NET page
  - If the browser can support the JavaScript that ASP.NET can send its way, the validation occurs on the client-side
  - It can be turned off using the **`EnableClientScript`** property
- Even if the client-side validation is initiated on a page, ASP.NET still performs the server-side validation when it receives the submitted page
- Custom validation controls can be created
- Validation occurs in response to an event (in most cases, it is a button click event)
  - The **`CausesValidation`** property of the **`Button`**, **`LinkButton`**, and **`ImageButton`** server controls is set to **`true`** by default

Validation Controls

- **`RequiredFieldValidator`**
  - Ensures that the user does not skip a form entry field
- **`CompareValidator`**
  - Allows for comparisons between the user's input and another item using a comparison operator (equals, greater than, less than, etc.)
- **`RangeValidator`**
  - Checks the user's input based upon a range of numbers or characters
- **`RegularExpressionValidator`**
  - Checks that the user's entry matches a pattern defined by a regular expression (useful for e-mail addresses or phone numbers)
- **`CustomValidator`**
  - Checks the user's entry using custom-coded validation logic
- **`ValidationSummary`**
  - Displays all the error messages from the validation controls in one specific spot on the page

RequiredFieldValidator

- There must be a **`RequiredFieldValidator`** control for each form element on which a value-required rule should be enforced
- The text to show to the end user if the validation fails can be set in the **`ErrorMessage`** or **`Text`** property, or as a text between the opening and closing nodes
- When a control to validate has a value which must be changed by the end user, the **`InitialValue`** property should be used

RequiredFieldValidator cont.

```
This text box cannot be empty:
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
    runat="server" ControlToValidate="TextBox1"
    ErrorMessage="Empty!"
    ValidationGroup="ValGroup1"/>
```

```
Choose sChoose something different than the default item:
<asp:DropDownList ID="DropDownList1" runat="server">
    <asp:ListItem>-- choose something --</asp:ListItem>
    <asp:ListItem>First</asp:ListItem>
    <asp:ListItem>Second</asp:ListItem>
    <asp:ListItem>Third</asp:ListItem>
</asp:DropDownList>
<asp:RequiredFieldValidator ID="RequiredFieldValidator2"
    runat="server" ControlToValidate="DropDownList1"
    InitialValue="-- choose something --"
    ErrorMessage="Default value!" />
```

CompareValidator

- Validating against other controls

```
Password:
<asp:TextBox ID="TextBox4" runat="server" TextMode="Password" />
<br />
Retype password:
<asp:TextBox ID="TextBox5" runat="server" TextMode="Password" />
<br />
<asp:CompareValidator ID="CompareValidator1" runat="server"
    ControlToCompare="TextBox5" ControlToValidate="TextBox4"
    ErrorMessage="Passwords do not match!" />
```

CompareValidator cont.

- Validating against constants
  - Supported types: **`Currency`, `Date`, `Double`, `Integer`, `String`**
  - Supported operators: **`Equal`, `NotEqual`, `GreaterThan`, `GreaterThanEqual`, `LessThan`, `LessThanEqual`, `DataTypeCheck`**

```
Accept only numbers here:
<asp:TextBox ID="TextBox6" runat="server"></asp:TextBox>
<asp:CompareValidator ID="CompareValidator2" runat="server"
    ControlToValidate="TextBox6" ErrorMessage="Not a number!"
    Operator="DataTypeCheck" SetFocusOnError="True" Type="Double"
</asp:CompareValidator>
```

RangeValidator

- The **`RangeValidator`** control makes sure that the end user value or selection provided is between a specified range

```
Time:
<asp:TextBox ID="TextBox8" runat="server" Width="21px" />
:<asp:TextBox ID="TextBox9" runat="server" Width="23px" />
<asp:RangeValidator ID="RangeValidator1" runat="server"
    ControlToValidate="TextBox8"
    ErrorMessage="Wrong hour (0-23 accepted)!"
    MaximumValue="23" MinimumValue="0" Type="Integer">
</asp:RangeValidator>
<asp:RangeValidator ID="RangeValidator2" runat="server"
    ControlToValidate="TextBox9"
    ErrorMessage="Wrong minutes (0-59 accepted)!"
    MaximumValue="59" MinimumValue="0" Type="Integer">
</asp:RangeValidator>
```

RegularExpressionValidator

```
Email:
<asp:TextBox ID="TextBox10" runat="server"></asp:TextBox>
<asp:RegularExpressionValidator
    ID="RegularExpressionValidator1"
    runat="server"
    ErrorMessage="Incorrect email!"
    ControlToValidate="TextBox10"
    ValidationExpression=
                  "\w+([-+.']\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*" >
</asp:RegularExpressionValidator>
```

CustomValidator

- The client-side validation

```
<head runat="server">
    <title>CustomValidator</title>
    <script language="JavaScript">
        function CustomValidator1_ClientValidate(oSrc, args) {
            args.IsValid = (args.Value % 5 == 0);
        }
     </script>
</head>
<body>
<form id="form1" runat="server">
    Number: <asp:TextBox ID="TextBox11" Runat="server" /> 
    <asp:CustomValidator ID="CustomValidator1" runat="server"
        ControlToValidate="TextBox11" ErrorMessage="Wrong!"
        ClientValidationFunction="CustomValidator1_ClientValidate" />
    <asp:Button ID="Button1" OnClick="Button1_Click"
         Runat="server" Text="Button"></asp:Button>
</form>
</body>
```

CustomValidator cont.

- The server-side validation

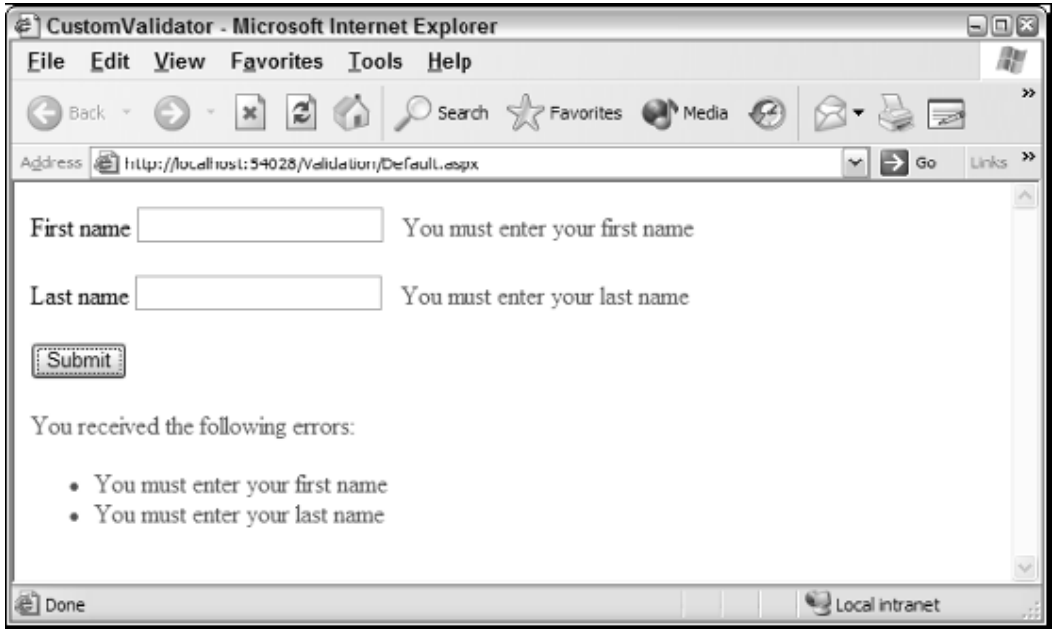
```
<asp:TextBox ID="TextBox11" runat="server"></asp:TextBox>
<asp:CustomValidator ID="CustomValidator1" runat="server"
    ControlToValidate="TextBox11" ErrorMessage="Wrong!"
    onservervalidate="CustomValidator1_ServerValidate">
</asp:CustomValidator>
```

```
protected void CustomValidator1_ServerValidate(object source,
        ServerValidateEventArgs args)
{
    int num;
    if (int.TryParse(args.Value, out num) && num % 5 == 0) {
        args.IsValid = true;
    } else {
        args.IsValid = false;
    }
}
```

ValidationSummary

- It is the reporting control used by the other validation controls on a page
  - It can be used to consolidate error reporting for all the validation errors that occur on a page

```
<asp:ValidationSummary ID="ValidationSummary1" runat="server"
    HeaderText="You received the following errors:" />
```



ValidationSummary cont.

- The **`ValidationSummary`** control shows values of the **`ErrorMessage`** properties of the validation controls
  - The `"*"` string is usually set as a value of the **`Text`** property of the validation controls
- The **`DisplayMode`** property can be used to change the display of the results to other types of format
  - **`BulletList`**, **`List`**, **`SingleParagraph`**
- The **`ShowMessageBox`** property set to **`true`** forces the browser to report errors in a message box

Using Images and Sounds

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
    ControlToValidate="TextBox11" Display="Dynamic"
    ErrorMessage="<img src='exclamation.png' />"
    onservervalidate="CustomValidator1_ServerValidate"
    ClientValidationFunction="CustomValidator1_ClientValidate"
    ValidationGroup="ValGroup4">
```

```
<asp:CustomValidator ID="CustomValidator1" runat="server"
    ControlToValidate="TextBox11" Display="Dynamic"
    ErrorMessage='<bgsource src="sound.wav"> '
    onservervalidate="CustomValidator1_ServerValidate"
    ClientValidationFunction="CustomValidator1_ClientValidate"
    ValidationGroup="ValGroup4">
```

Validation Groups

- The **`ValidationGroup`** property allows to separate the validation controls into groups
  - It allows to activate only the required validation controls when the end user clicks a button on the page

```
<asp:Button ID="Button1" Runat="server" Text="Login"
            ValidationGroup="Login" />
<br />
<asp:RequiredFieldValidator ID="RequiredFieldValidator1"
            Runat="server"
            ErrorMessage="* You must submit a username!"
            ControlToValidate="TextBox1"
            ValidationGroup="Login">
</asp:RequiredFieldValidator>
<asp:RequiredFieldValidator ID="RequiredFieldValidator2"
            Runat="server"
            ErrorMessage="* You must submit a password!"
            ControlToValidate="TextBox2"
            ValidationGroup="Login">
</asp:RequiredFieldValidator>
```

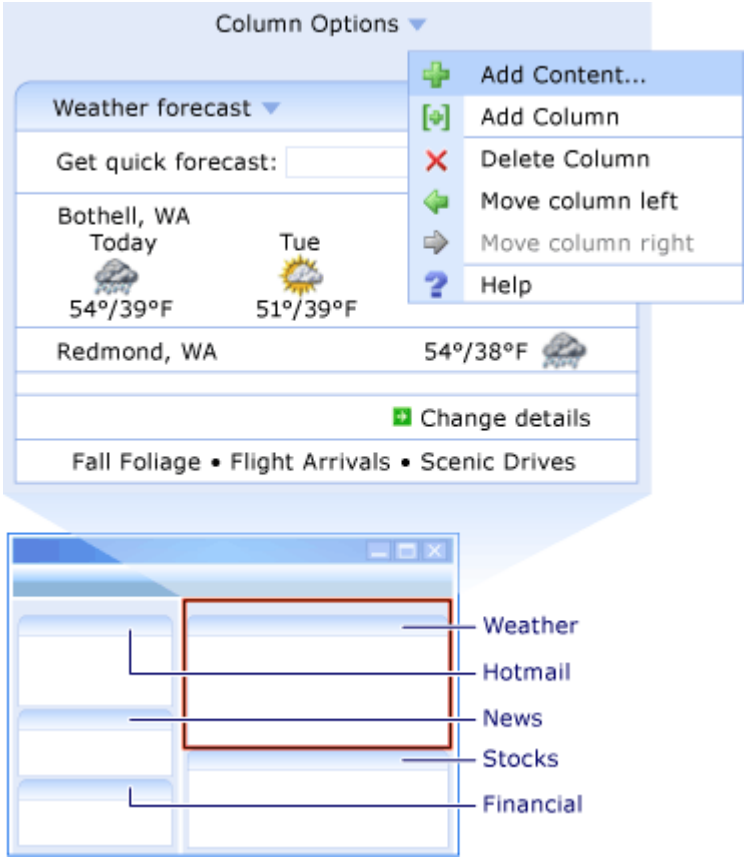
Web Parts

Web Parts

- ASP.NET Web Parts is an integrated set of controls for creating Web sites
- Web Parts enable end users to modify the content, appearance, and behaviour of Web pages directly from a browser
  - The modifications can be applied to all users on the site or to individual users
  - When a user modifies pages and controls, the settings can be saved to retain the user's personal preferences across future browser sessions (using the personalisation)

Possibilities for Users

- Web Parts allow end users to:
  - Personalise the page content
  - Personalise the page layout
  - Export and import controls
  - Create connections between controls
  - Manage and personalise site-level settings



Developer Scenarios

- The page development
  - Page developers can use visual design tools such as Visual Studio 2005 to create pages that use Web Parts
- The control development
  - Any existing ASP.NET control can be used as a Web Parts control, including standard Web server controls, custom server controls, and user controls
  - For maximum programmatic control, custom Web Parts controls derived from the **`WebPart`** class can be created
- The Web application development
  - A Web site that allows extensive user personalisation of the UI and content can be developed
    - A set of Web Parts controls
    - A consistent set of themes and skins
    - Catalogues of selectable Web Parts controls
    - Authentication services and role-based management

Web Parts Components

- The personalisation
- UI structural components
  - Rely on the personalisation
  - Provide the core structure and services needed by all Web Parts controls
  - One UI structural component required on every Web Parts page is the **`WebPartManager`** control
  - Zones act as layout managers on a Web Parts page
- Web Parts UI controls
  - Derived from the **`Part`** class
  - They comprise the primary UI on a Web Parts page

WebPartManager Control

- Required on every Web Parts page
- Invisible
- It has the critical task of coordinating all Web Parts controls on a page
  - It tracks all the individual Web Parts controls
  - It manages Web Parts zones, and which controls are in which zone
  - It tracks and controls the different display modes a page can be in (browse, connect, edit, or catalogue)
  - It controls whether personalisation changes apply to all users or to individual users
  - It initiates and tracks connections and communication between Web Parts controls

Zones

- Zones act as layout managers on a Web Parts page
- They contain and organize controls that derive from the **`Part`** class (part controls)
- They provide the ability to create a modular page layout in either horizontal or vertical orientation
- Zones also offer common and consistent UI elements (such as the header and footer style, title, border style, action buttons, and so on) for each control they contain
  - These common elements are known as the chrome of a control

Types of Zones

- **`CatalogZone`**
  - Contains the **`CatalogPart`** controls
  - Used to create a catalogue of Web Parts controls from which users can select controls to add to a page
- **`EditorZone`**
  - Contains the **`EditorPart`** controls
  - Used to enable users to edit and personalise Web Parts controls on a page
- **`WebPartZone`**
  - Contains and provides overall layout for the **`WebPart`** controls
- **`ConnectionsZone`**
  - Contains the **`WebPartConnection`** controls
  - Provides the UI that enables users to form connections between the **`WebPart`** and other server controls

Types of Web Parts UI Controls

- **`WebPart`**
  - Renders the primary UI
  - For maximum programmatic control, create custom Web Parts controls that derive from the base **`WebPart`** class
  - The **`GenericWebPart`** controls are created automatically by the **`WebPartManager`** control for server controls, user controls, or custom controls placed in a zone
- **`CatalogPart`**
  - Contains a list of available Web Parts controls that users can add to the page

Types of Web Parts UI Controls cont.

- **`WebPartConnection`**
  - Creates a connection between two Web Parts controls on a page
- **`EditorPart`**
  - Serves as the base class for the specialized editor controls
  - The **`AppearanceEditorPart`**, **`LayoutEditorPart`**, **`BehaviorEditorPart`**, and **`PropertyGridEditorPart`** controls allow users to personalise various aspects of the Web Parts UI controls on a page

Web Parts Page Display Modes

- A display mode is a special state that applies to an entire page
  - Certain UI elements are visible and enabled, while others are invisible and disabled
- A page can be in only one display mode at a time
- The **`WebPartManager`** control contains the implementation for the display modes that are available in the Web Parts control set
- Typically, you provide an UI that enables users to switch among display modes as needed
- The **`DisplayMode`** property can be used to change a page's display mode programmatically

Standard Display Modes

- **`BrowseDisplayMode`**
  - The normal mode in which end users view a page
- **`DesignDisplayMode`**
  - Enables users to drag Web Parts controls to change the layout of a page
- **`EditDisplayMode`**
  - Displays special editing UI elements and enables end users to edit the controls on a page
  - Allows dragging of controls
- **`CatalogDisplayMode`**
  - Displays special catalogue UI elements and enables end users to add and remove page controls
  - Allows dragging of controls
- **`ConnectDisplayMode`**
  - Displays special connections UI elements and enables end users to connect Web Parts controls

Creating Web Parts Solutions

1. Add the **`WebPartManager`** control to the page
2. Add the **`WebPartZone`** controls to the page
   - For example, one horizontal on the top (as a header) and three vertical (as columns)
3. Add controls to zones
   - Web server controls, user controls, custom controls, HTML server controls, raw text, HTML elements
4. Allow the end user to change the mode of the page
   - For example, the **`DropDownList`** control can be used (populated by iterating through the **`SupportedDisplayModes`** collection)

Creating Web Parts Solutions

5. Allow the end user to add Web Parts
   - Add the **`CatalogZone`** control
   - Add the **`PageCatalogPart`** control to the zone
     - It contains a title and check box list of items that can be selected by the user
6. In the **`Design`** mode, the end user can move Web Parts from one zone to another

Creating Web Parts Solutions

7. Allow the end user to use the **`Edit`** mode
   - Add the **`EditorZone`** control
   - Add one or many of the following controls:
     - **`AppearanceEditorPart`** – allows to change the Web Part's details, including the title, how the title appears, and other appearance-related items such as the item's height and width
     - **`BehaviorEditorPart`** – enables the end user to select whether the Web Part can be closed, minimized, or exported
     - **`LayoutEditorPart`** – allows to change the order in which Web Parts appear in a zone or move Web Parts from one zone to another
     - **`PropertyGridEditorPart`** – enables to modify properties that are defined in your own custom server controls

Custom Web Parts

- Developing a Web Part is quite similar to developing a custom control
  - It must be derived from **`System.Web.UI.WebControls.WebParts.WebPart`**
- The Web Part can be created by rendering HTML or composing from other controls
- The Web Part includes considerable functionality for integrating with the Web Part architecture

Data Binding

ASP.NET Data Access

- The `System.Data` (ADO.NET) and `System.Xml` namespaces can be used to write code to access data
- ASP.NET allows to perform data binding declaratively, no code is required for the most common data scenarios:
  - Selecting and displaying data
  - Sorting, paging, and caching data
  - Updating, inserting, and deleting data
  - Filtering data using run-time parameters
  - Creating master-detail scenarios using parameters
- Two types of server controls participate in the declarative data binding model:
  - Data source controls
  - Data-bound controls

Data Source Controls

- Data source controls are ASP.NET controls that manage the tasks of connecting to a data source, reading and writing data
  - They do not render any user interface
  - They act as an intermediary between a particular data store and other controls on the ASP.NET Web page
  - They enable rich capabilities for retrieving and modifying data, including querying, sorting, paging, filtering, updating, deleting, and inserting
- There are 7 data source controls:
  `ObjectDataSource`, `SqlDataSource`, `AccessDataSource`, `XmlDataSource`, `SiteMapDataSource`, `LinqDataSource`, `EntityDataSource` (3.5 SP1)

ObjectDataSource Control

- Allows to work with a business object or other class and create Web applications that rely on middle-tier objects to manage data
- Supports advanced sorting and paging scenarios unavailable with the other data source controls
- The control is designed to interact with an object that implements one or more methods to retrieve or modify data
  - The source object's data-retrieval methods must return a **`DataSet`**, **`DataTable`**, or **`DataView`** object, or an object that implements the **`IEnumerable`** interface
- Advanced paging scenarios can be also implemented if the source object accepts page size and record index information from the **`ObjectDataSource`** control

SqlDataSource Control

- It retrieves and modifies data using SQL commands
  - Allows to work with Microsoft SQL Server, OLE DB, ODBC, or Oracle databases
  - When used with SQL Server, supports advanced caching capabilities
- The **`SqlDataSource`** control can return results as the **`DataReader`** or **`DataSet`** object
  - It supports sorting, filtering, and caching when the results are returned as a **`DataSet`**

SqlDataSource Example

```
<asp:SqlDataSource
    id="SqlDataSource1"
    runat="server"
    DataSourceMode="DataReader"
    ConnectionString="<%$ ConnectionStrings:MyNorthwind%>"
    SelectCommand="SELECT LastName FROM Employees">
</asp:SqlDataSource>

<asp:ListBox
    id="ListBox1"
    runat="server"
    DataTextField="LastName"
    DataSourceID="SqlDataSource1">
</asp:ListBox>
```

```
<configuration>
    <!-- Other configuration settings -->
    <connectionStrings>
        <add name="MyNorthwind"
             providerName="System.Data.SqlClient"
             connectionString="server=.;database=NorthWind;
                               Integrated Security=SSPI" />
    </connectionStrings>
</configuration>
```

AccessDataSource Control

- The **`AccessDataSource`** control is a specialized version of the **`SqlDataSource`** control
  - It is designed to work specifically with Microsoft Access .mdb files
  - The **`AccessDataSource`** will not connect to an Access database that is password-protected; to retrieve data from a password-protected Access database, use the **`SqlDataSource`** control

```
<asp:AccessDataSource
    id="AccessDataSource1"
    DataFile="~/App_Data/Northwind.mdb"
    runat="server"
    SelectCommand="SELECT EmployeeID, LastName, FirstName
                   FROM Employees">
</asp:AccessDataSource>
```

LinqDataSource and EntityDataSource

- The **`LinqDataSource`** control enables to use LINQ in an ASP.NET Web page by setting properties in markup text
  - It uses LINQ to SQL to automatically generate the data commands
- The **`EntityDataSource`** control represents an Entity Data Model (EDM) to data-bound controls in an ASP.NET application
  - It was introduced with Entity Framework in .NET 3.5 SP1

XmlDataSource Control

- The `XmlDataSource` control can read either an XML file or string of XML
  - If the control is working with an XML file, it can write modified XML back to the source file
  - If a schema is available that describes the data, the `XmlDataSource` control can use the schema to expose data using typed members
- The XML data can be modified
  - An XSLT transformation allows to restructure the raw data
  - XPath expressions allow to filter the XML data to return only certain nodes in the XML tree

XmlDataSource Example

```
<asp:XmlDataSource id="PeopleDataSource" runat="server"
                   DataFile="~/App_Data/people.xml" />

<asp:TreeView id="PeopleTreeView" runat="server"
              DataSourceID="PeopleDataSource">
    <DataBindings>
        <asp:TreeNodeBinding DataMember="LastName"
                             TextField="#InnerText" />
        <asp:TreeNodeBinding DataMember="FirstName"
                             TextField="#InnerText" />
        <asp:TreeNodeBinding DataMember="Street"
                             TextField="#InnerText" />
        <asp:TreeNodeBinding DataMember="City"
                             TextField="#InnerText" />
    </DataBindings>
</asp:TreeView>
```

SiteMapDataSource Control

- It retrieves navigation data from a site-map provider, and then passes the data to controls that can display that data, such as the **`TreeView`** and **`Menu`** controls
  - This data includes information about the pages in your Web site, such as the URL, title, description, and location in the navigation hierarchy

Data Source Control Caching

- The caching is built into all the data source controls except the **`SiteMapDataSource`** control
  - These controls allow to create basic caching policies including a cache direction, expiration policies, and key dependencies

```
<asp:SqlDataSource ID="SqlDataSource1"
        Runat="server"
        SelectCommand="SELECT * FROM [Customers]"
ConnectionString="<%$ ConnectionStrings:AppConnectionString1 %>"
        DataSourceMode="DataSet"
        ConflictDetection="CompareAllValues"
        EnableCaching="True"
        CacheKeyDependency="SomeKey"
        CacheDuration="Infinite">
```

```
<connectionStrings>
    <add name="AppConnectionString1"
         connectionString="Server=localhost;
               User ID=sa;Password=password;Database=Northwind;
               Persist Security Info=True"
        providerName="System.Data.SqlClient" />
</connectionStrings>
```

Bound List Controls

- **`GridView`** – displays tabular data
- **`DetailsView`** – displays a single record from a data source (each data row represents a field in the record)
- **`FormView`** – works like the DetailsView, but it displays the data in custom templates
- **`Repeater`** – produces a list of individual items
- **`DataList`** – displays rows of database information in customizable format

Binding Data to Bound List Controls

- There are two ways to bind data to bound list controls:
  - Using the **`DataSourceID`** property – it allows to use a data source control
    - In this case, the **`GridView`** control provides built-in functionality for sorting, paging, and updating
    - When the **`DataSourceID`** property is used, the GridView control supports two-way data binding
  - Using the **`DataSource`** property – it allows to bind to various objects, including ADO.NET datasets and data readers
    - Any additional functionality such as sorting, paging, and updating must be coded manually

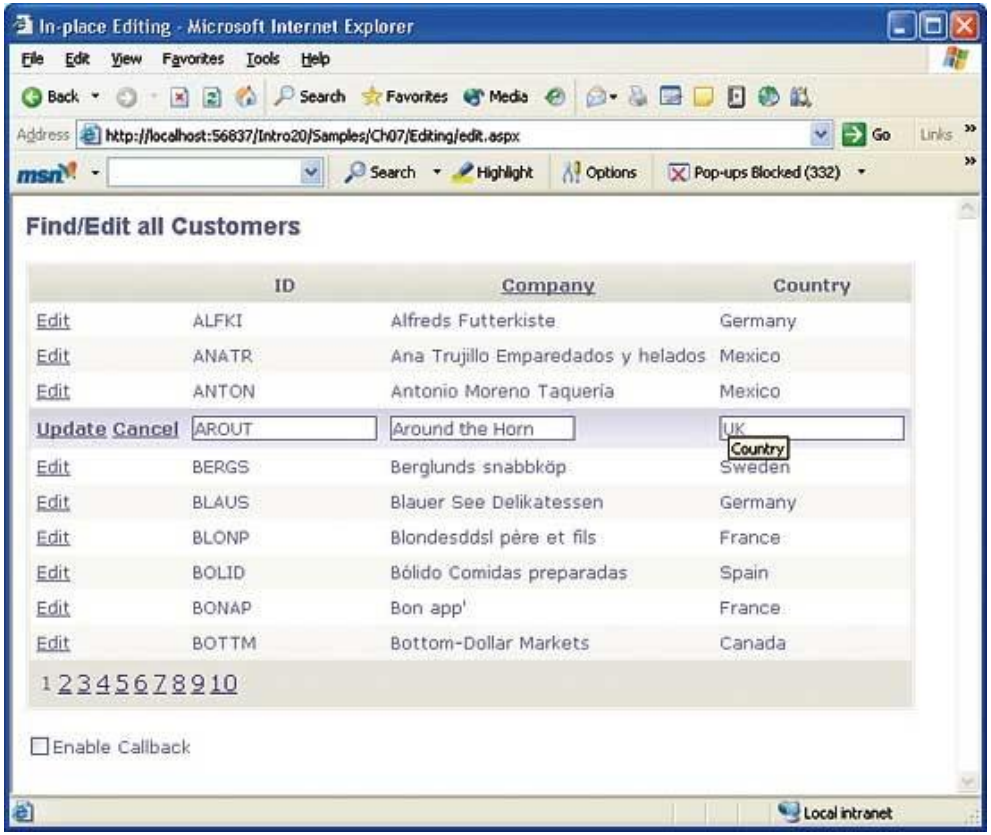
GridView Control

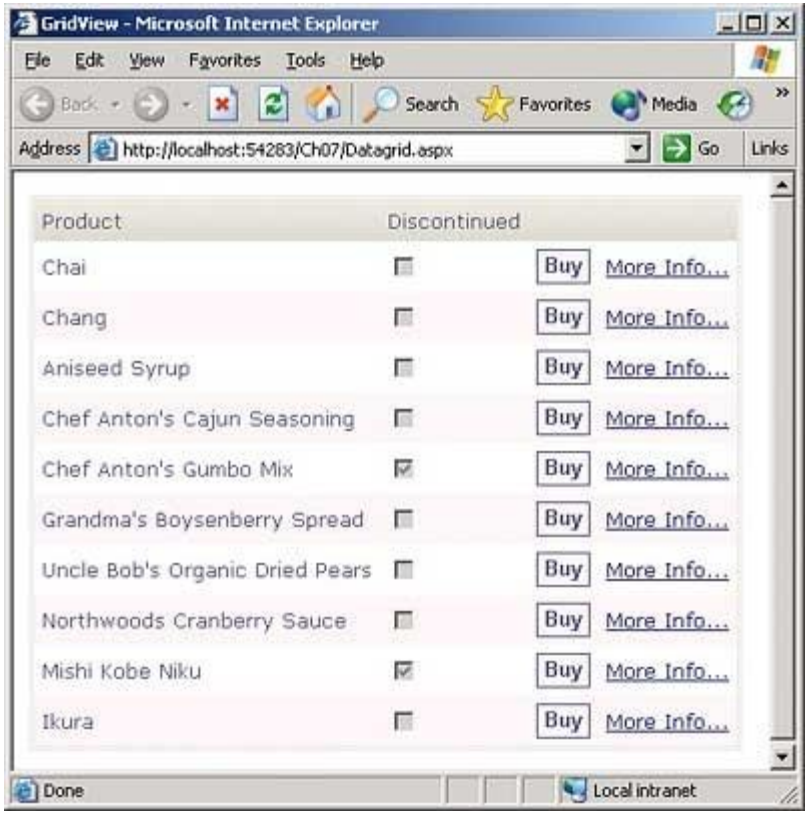
- Possibilities of custom formatting:
  - The layout, colour, font, and alignment of the **`GridView`** control's rows
  - The display of text and data contained in the rows
  - How to display data rows: as items, as alternating items, selected items, or edit-mode items
  - The format of columns (including possibility to choose a type of controls which should appear in the column)
- By default, the **`GridView`** control displays data in the read-only mode
  - The control also supports the edit mode
  - The control can automatically perform editing and deleting operations or the process of editing and deleting can be controlled programmatically

GridView Control cont.

- The **`GridView`** control supports sorting by a single column without requiring any programming
  - The sort functionality can be customized by using the **`Sorting`** event and providing the sort expression (using the **`SortExpression`** property)
- It provides a simple paging functionality
  - The paging functionality can be customized by using the **`PageTemplate`** property
- It provides events that occur both before and after a navigation or edit operation (e.g. **`RowDeleting`**, **`RowDeleted`**, **`PageIndexChanging`**, **`PageIndexChanged`**)

GridView Control cont.





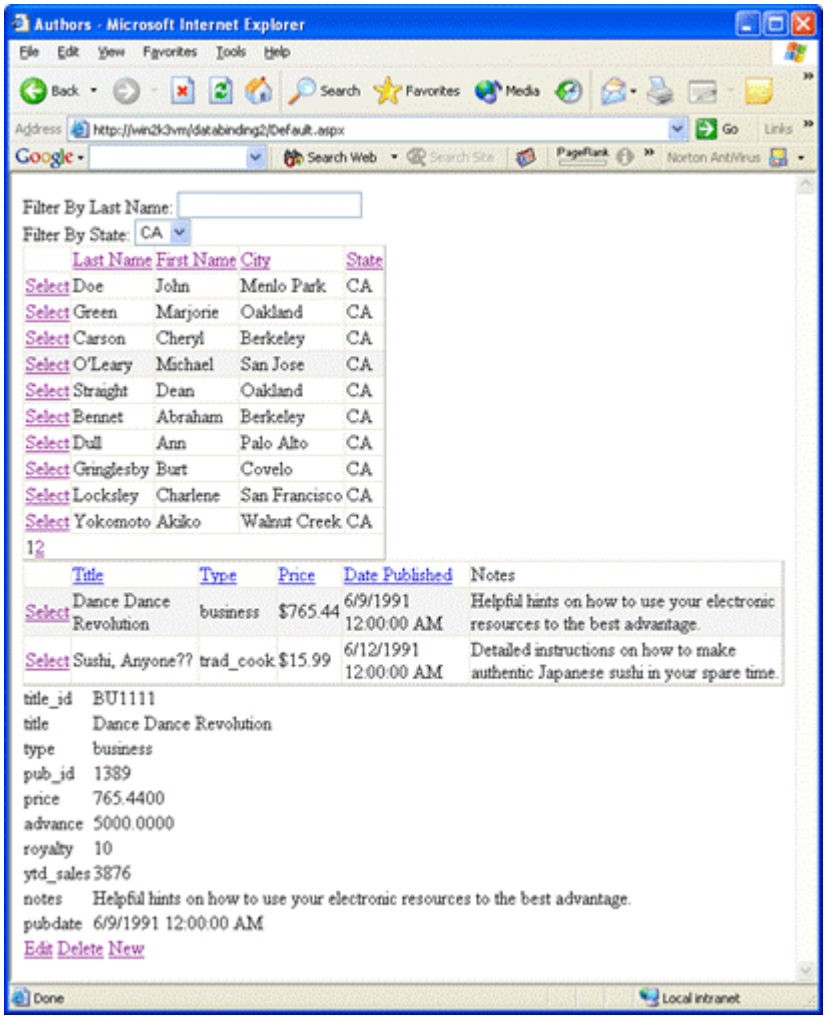
DetailsView Control

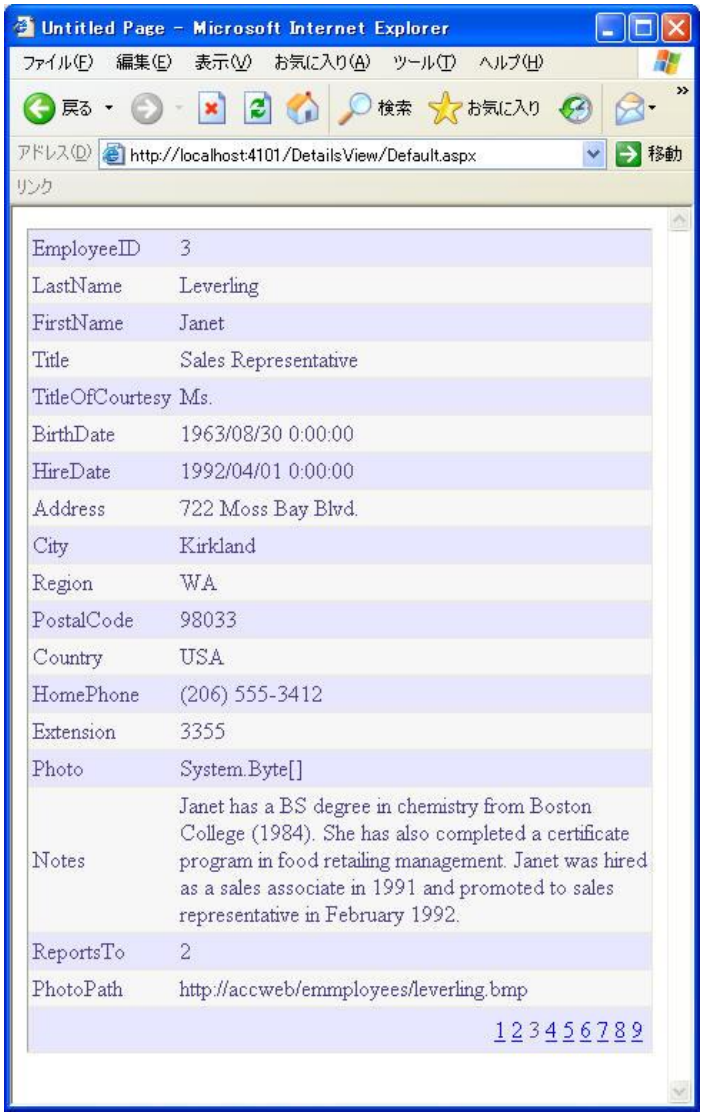
- It allows to display, edit, insert, or delete a single record at a time from its associated data source
  - By default it displays each field of a record on its own line
  - It displays only a single data record at a time, even if its data source exposes multiple records
- It does not support sorting
- The **`AutoGenerateEditButton`** and **`AutoGenerateInsertButton`** properties allow to support editing and inserting data
- The **`Fields`** collection properties determines internal controls used to render the data

DetailsView Control cont.

- The UI of the control can be customized by using style properties:
  - **`HeaderStyle, RowStyle, AlternatingRowStyle, CommandRowStyle, FooterStyle, PagerStyle, EmptyDataRowStyle`**
- Additional customization is available through templates:
  - **`EmptyDataTemplate, HeaderTemplate, FooterTemplate, PagerTemplate`**
- The event model of the **`DetailsView`** control is similar to that of the **`GridView`** control, but there is no support for selecting

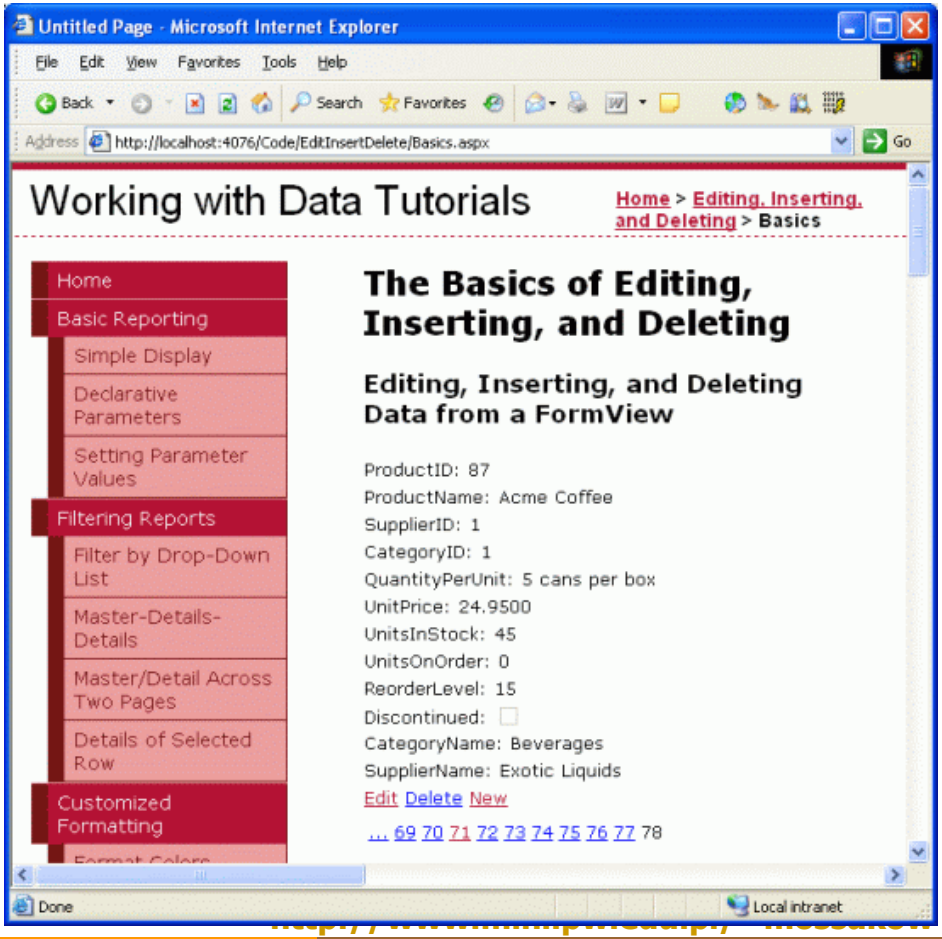
DetailsView Control cont.





FormView Control

- The **`FormView`** control is similar to the **`DetailsView`** control, it differs by the layout:
  - The **`DetailsView`** uses a tabular layout, while the **`FormView`** control allows to create a template containing controls to display individual fields from the record
  - The **`ItemTemplate`**, **`HeaderTemplate`**, **`FooterTemplate`**, and **`EmptyDataTemplate`** properties can be used

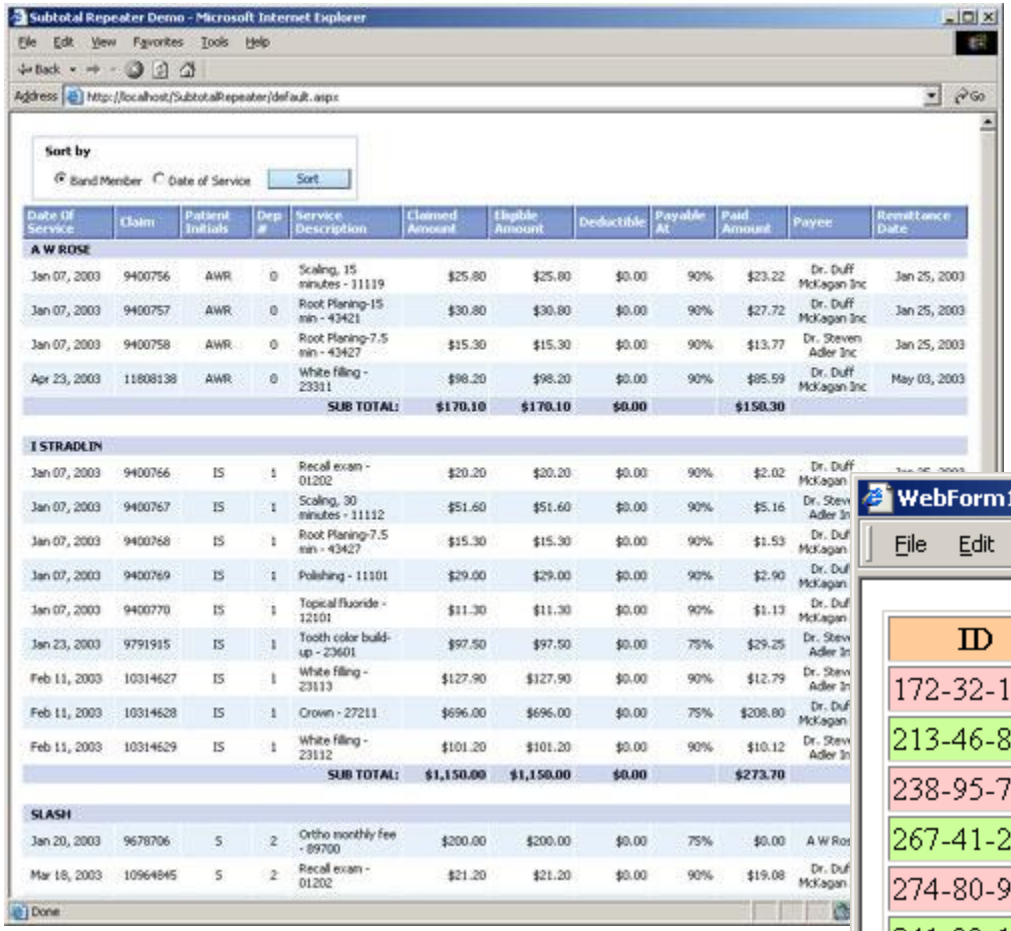


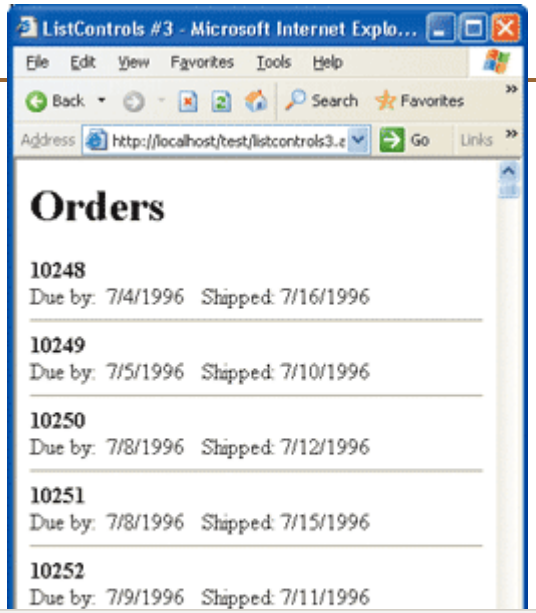
Repeater Control

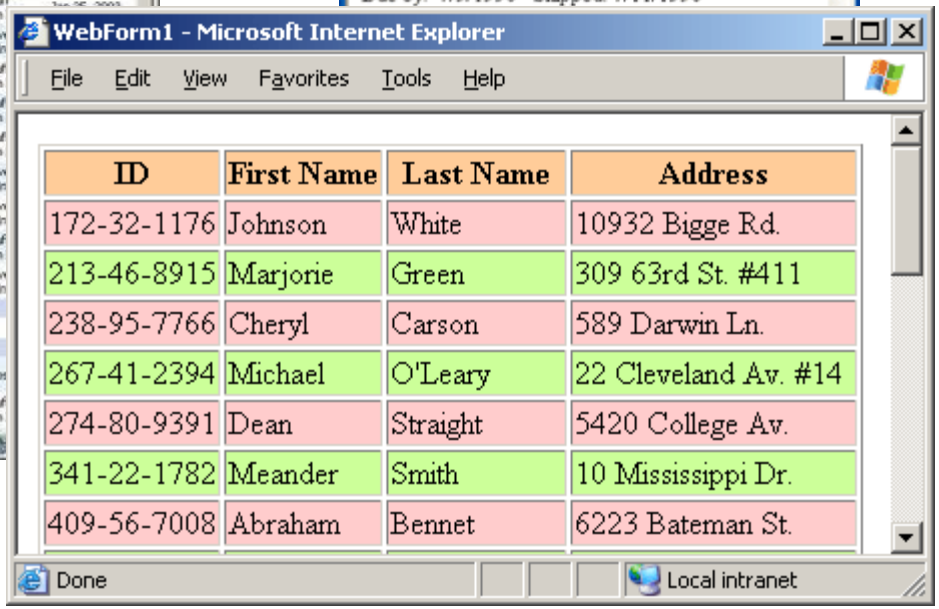
- The **`Repeater`** control is a container control that allows to create custom lists out of any data that is available to the page
- It does not have a built-in rendering of its own
  - The layout for the control must be provided by creating templates
  - **`SeparatorTemplate`**, **`AlternatingItemTemplate`**, **`HeaderTemplate`**, **`FooterTemplate`**, **`ItemTemplate`**
- When the page runs, the **`Repeater`** control loops through the records in the data source and renders an item for each record

```
<asp:Repeater ID="Repeater1" runat="server"
              DataSourceID="SqlDataSource1">
    <HeaderTemplate>
        <table><tr><th>Name</th><th>Description</th></tr>
    </HeaderTemplate>
    <ItemTemplate>
        <tr><td bgcolor="#CCFFCC">
        <asp:Label runat="server" ID="Label1"
                   Text='<%# Eval("CategoryName") %>' />
        </td><td bgcolor="#CCFFCC">
        <asp:Label runat="server" ID="Label2"
                   Text='<%# Eval("Description") %>' />
        </td></tr>
    </ItemTemplate>
    <AlternatingItemTemplate>
        <tr><td>
        <asp:Label runat="server" ID="Label3"
                   Text='<%# Eval("CategoryName") %>' />
        </td><td>
        <asp:Label runat="server" ID="Label4"
                   Text='<%# Eval("Description") %>' />
        </td></tr>
    </AlternatingItemTemplate>
    <FooterTemplate>
         </table>
    </FooterTemplate>
</asp:Repeater>
```

Repeater Control cont.



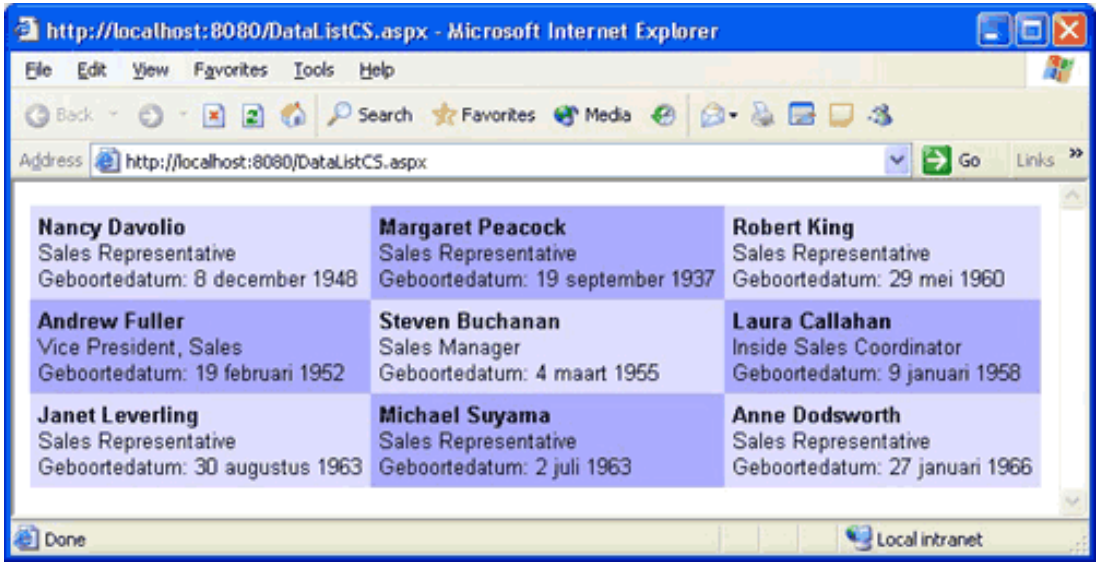




ID	First Name	Last Name	Address
172-32-1176	Johnson	White	10932 Bigge Rd.
213-46-8915	Marjorie	Green	309 63rd St. #411
238-95-7766	Cheryl	Carson	589 Darwin Ln.
267-41-2394	Michael	O'Leary	22 Cleveland Av. #14
274-80-9391	Dean	Straight	5420 College Av.
341-22-1782	Meander	Smith	10 Mississippi Dr.
409-56-7008	Abraham	Bennet	6223 Bateman St.

DataList Control

- The `DataList` control displays data in a format that can be defined using templates and styles
- It can display rows in different layouts, such as ordering them in columns or rows
- It uses an HTML table to lay out the rendering of items to which the template is applied, the following layout options are supported:
  flow layout,
  table layout,
  vertical and
  horizontal layout,
  number of columns



Responding to Button Events

- To respond to button events in the **`DataList`**, **`Repeater`**, and **`GridView`** controls:
  1. Include a **`Button`**, **`LinkButton`**, or **`ImageButton`** in a control template
  2. Set the button's **`CommandName`** property to a string that identifies its function, such as "sort" or "copy"
  3. Create a method for the **`ItemCommand`** event of the containing control.
     - In this method check the **`CommandName`** property and perform appropriate logic.

```
protected void DataList1_ItemCommand(object source,
                                     DataListCommandEventArgs e) {
    if (e.CommandName == "AddToCart") {
        // ...
    }
}
```

ListView Control

- It resembles the **`GridView`** control, except that it displays data by using user-defined templates instead of row fields
- Features:
  - Support for binding to data source controls
  - Customizable appearance through user-defined templates and styles
  - Built-in sorting capabilities
  - Built-in update and delete capabilities
  - Built-in insert capabilities
  - Support for paging capabilities by using a **`DataPager`** control
  - Built-in item selection capabilities
  - Programmatic access to the **`ListView`** object model to dynamically set properties, handle events, and so on

ListView Control cont.

First Previous Next Last

ID: **ALFKI**
Company name: **Alfreds Futterkiste--modified--**
Number of orders: **6**

ID: **ANATR**
Company name: **Ana Trujillo Emparedados y helados**
Number of orders: **4**

ID: **ANTON**
Company name: **Antonio Moreno Taquería**
Number of orders: **7**

ID: **AROUT**
Company name: **Around the Horn**
Number of orders: **13**

ID: **BERGS**
Company name: **Berglunds snabbköp**
Number of orders: **18**

ID: **BLAUS**
Company name: **Blauer See Delikatessen**
Number of orders: **7**

ID: BLAUS
Company name: Blauer See Delikatessen
Number of orders: 7

ID: BLONP
Company name: Blondesddsl pere et fils
Number of orders: 11

ID: BOLID
Company name: Bólido Comidas preparadas
Number of orders: 3

ID: BONAP
Company name: Bon app'
Number of orders: 17

ID: BOTTM
Company name: Bottom-Dollar Markets
Number of orders: 14

First Previous Next Last

Products


Chai2

Guaraná Fantástica


Sasquatch Ale

Steeleye Stout


Côte de Blaye


Chartreuse verte

1 2

DataPager Control

- The **`DataPager`** class is used to page data and to display navigation controls for data-bound controls that implement the **`IPageableItemContainer`** interface
  - e.g. the **`ListView`** control
- The **`DataPager`** control with the data-bound control by using the **`PagedControlID`** property
  - Alternatively, the **`DataPager`** control can be put inside the data-bound control hierarchy
    - e.g., in the **`ListView`** control, put this control inside the **`LayoutTemplate`** template.

Web.config Elements

<system.web> Elements

- **<anonymousIdentification>** - anonymous identification for application authorization
- **<authentication>** - authentication scheme that is used to identify users who view the application
  - **<forms>** - configuration for custom forms-based authentication
    - **<credentials>** - optional definitions of name and password credentials within the configuration file

```
<authentication mode="Windows">
   <credentials passwordFormat="SHA1">
      <clear />
      <user name="UserName"
            password="07B7F3EE06F278DB966BE960E7CBBD103DF30CA6"/>
   </credentials>
</authentication>
```

  - **<passport>** - Microsoft Passport authentication

<system.web> Elements cont.

- **<authorization>** - the authorization for a Web application
  - **<allow>, <deny>** - users, roles, verbs

```
<authorization>
    <allow roles="admins"/>
    <deny users="*"/>
</authorization>
```

- **<browserCaps>** - the settings and capabilities of supported browsers (deprecated in APS.NET 2.0)
- **<caching>** - the cache settings
  - **<cache>** - global application cache settings
  - **<outputCache>** - application-wide output-cache settings
  - **<outputCacheSettings>** - output-cache settings that can be applied to pages
  - **<sqlCacheDependencies>** - database caching

<system.web> Elements cont.

- **<clientTarget>** - adds aliases for specific user agents to an internal collection of user agent aliases
- **<compilation>** - all compilation settings that ASP.NET uses to compile applications
- **<customErrors>** - custom error messages
  - **<error>** - the custom error page for a given HTTP status code

```
<customErrors defaultRedirect="GenericError.htm"
              mode="RemoteOnly">
   <error statusCode="500" redirect="InternalError.htm"/>
</customErrors>
```

- **<deployment>** - whether the application is deployed in retail mode
- **<deviceFilters>** - specifies a device or a device class in the ASP.NET Mobile Capabilities system based on the user agent or browser

<system.web> Elements cont.

- **<globalization>** - the globalization settings

```
<globalization requestEncoding="iso-8859-1"
               responseEncoding="iso-8859-1"/>
```

- **<healthMonitoring>**
- **<hostingEnvironment>** - configuration settings that control the behaviour of the application hosting environment

```
<hostingEnvironment idleTimeout="Infinite"
                    shutdownTimeout="30"
                    shadowCopyBinAssemblies="true" />
```

- **<httpCookies>** - properties for cookies

```
<httpCookies httpOnlyCookies="true"
             requireSSL="false" />
```

<system.web> Elements cont.

- **<httpHandlers>** - settings for HTTP handlers

```
<httpHandlers>
    <add verb="*" path="*.New"
         type="MyHandler.New,MyHandler"/>
    <add verb="GET,HEAD" path="*.MyNewFileExtension"
         type="MyHandler.MNFEHandler,MyHandler.dll"/>
</httpHandlers>
```

- **<httpModules>** - settings for HTTP modules within the application

```
<httpModules>
    <add type="System.Web.Caching.OutputCacheModule"
         name="OutputCache"/>
    <add type="System.Web.SessionState.SessionStateModule"
         name="Session"/>
    <add type=Selector, selector.dll"
         name="Selector"/>
</httpModules>
```

<system.web> Elements cont.

- **<httpRuntime>** - run-time settings that determine how to process a request for the application
- **<identity>** - the identity of the Web application

```
<identity impersonate="true"
          userName="registry:HKLM\Software\AspNetProcess,Name"
          password="registry:HKLM\Software\AspNetProcess,Pwd"/>
```

- **<machineKey>** - keys for encryption and decryption of Forms authentication cookie data and view-state data
- **<membership>** - parameters for managing and authenticating user accounts by using the ASP.NET membership
- **<mobileControls>** - defines adapters sets that map ASP.NET mobile controls

<system.web> Elements cont.

- **<pages>** - globally defines page-specific configuration settings, such as ASP.NET directives for pages and controls that are within the scope of the configuration file

```
<pages buffer="true"
       enableSessionState="true"
       autoEventWireup="true"
       maintainScrollPositionOnPostBack="true"
       masterPageFile = "~/Masters/Page1.master"
    <controls>
        <add tagprefix="MyTags3"
             tagname="MyCtrl"
             source="MyControl.ascx"/>
    </controls>
</pages>
```

- **<processModel>** - configures the ASP.NET process model settings on IIS Web server (**Machine.config** only)

<system.web> Elements cont.

- **<profile>** - parameters for managing the user profile

```
<profile defaultProvider="SqlProvider">
    <providers>
        <clear />
        <add name="SqlProvider"
             type="System.Web.Profile.SqlProfileProvider"
             connectionStringName="SqlServices"
             applicationName="SampleApplication"
             description="Just a provider" />
    </providers>
    <properties>
        <add name="ZipCode" />
        <add name="CityAndState" />
    </properties>
</profile>
```

<system.web> Elements cont.

- <roleManager>

```
<configuration>
    <system.web>
        <roleManager defaultProvider="SqlProvider"
                     enabled="true"
                     cacheRolesInCookie="true"
                     cookieName=".ASPROLES"
                     cookieTimeout="30"
                     cookiePath="/"
                     cookieRequireSSL="false"
                     cookieSlidingExpiration="true"
                     cookieProtection="All" >
            <providers>
                <add name="SqlProvider"
                     type="System.Web.Security.SqlRoleProvider"
                     connectionStringName="SqlServices"
                     applicationName="SampleApplication" />
            </providers>
        </roleManager>
    </system.web>
</configuration>
```

<system.web> Elements cont.

- **<securityPolicy>** - a collection of mappings between security policy files and the trust level names for the security policy files
- **<sessionPageState>** - configures page view-state settings
- **<sessionState>**

```
<sessionState mode="SQLServer"
              cookieless="true"
              sqlConnectionString="Integrated Security=SSPI;
                                   data source=MySqlServer;"
              sqlCommandTimeout="10" />
```

- **<siteMap>** - configuration settings to support the navigation infrastructure

<system.web> Elements cont.

- **<trace>** - the ASP.NET code tracing service

```
<trace enabled="true"
       pageOutput="true"
       requestLimit="15"
       mostRecent="true" />
```

- **<trust>** - the level of code access security (CAS)
- **<urlMappings>** - defines a mapping that hides the real URL and maps it to a more user-friendly URL

```
<urlMappings enabled="true">
    <clear />
    <add url="~/Home.aspx"
         mappedUrl="~/Default.aspx?tab=home" />
    <remove url="~/Home2.aspx" />
</urlMappings>
```

<system.web> Elements cont.

- **<webControls>** - the shared location of the client script files
- **<webParts>** - a Web Parts provider and authorization
- **<webServices>** - controls the behaviour of ASP.NET Web services and their clients
- **<xhtmlConformance>** - configures the XHTML 1.0–conforming control rendering