Chapter 10
Data Controls
Overview

- This chapter continues the broader topic of working with data in ASP.NET by focusing on how to work with ASP.NET’s powerful multivalue data controls.

- It covers:
  - Repeater Control
  - DataList Control
  - FormView Control
  - DetailsView Control
  - GridView Control
Data Controls

- ASP.NET has five controls dedicated to the display of multiple values from a data source.
- These controls have several similarities:
  - They all use template tags to separate the control data from its presentation.
  - That is, the control does not provide a user interface; rather, the user interface is customized by the developer via nested template elements.
  - They also all support data binding.
    - In fact, these controls can only display content via data binding.
  - These data controls vary in their flexibility, features, and speed.
DataList Control

- Displays rows of data as items in a list.
- The presentation of the data can be customized via templates. Inline editing and deleting of data is also supported by this control.
Repeater Control

- Provides a list of data.
- It is unique in that it provides complete flexibility in regard to the HTML presentation of the data.
- However, because there is no default look, the developer must completely describe via templates how the data is to be rendered.
DetailsView Control

- Displays a single record of data at a time. It provides buttons for navigating from record to record.
- As well, it supports the updating, insertion, or deletion of records.
- It can be used in conjunction with the GridView control to provide a master-detail view of your data.
FormView Control

- Like the DetailsView control, the FormView control displays a single record of data at a time and supports the editing of the record.

- Unlike the DetailsView control, the FormView control requires the use of templates to define the rendering of each item, so the developer can completely customize the appearance of the record.
GridView Control

- Displays data in a tabular (row-column) grid.
  - It provides the most features of the data controls.
- It not only supports the editing and deleting of data, it also supports the sorting and paging of data.
- Although it also uses templates to customize the appearance of the data, customizing the presentation requires more work than the DataList.
  - There is a default look to the GridView, so little work is required to displaying an entire table of data.
Understanding Templates

- The content and layout of list items in these controls are defined using templates.
- A template defines:
  - the HTML elements,
  - the ASP.NET controls,
  - the layout of these controls,
  - the data to be displayed via data-binding expressions.
- Style formatting of a template, such as fonts and colors, is set via the unique style element for that template.
The following table lists the available general templates:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemTemplate</td>
<td>Specifies the content and layout of items within a list.</td>
</tr>
<tr>
<td>AlternatingItemTemplate</td>
<td>Specifies the content and layout of alternating items.</td>
</tr>
<tr>
<td>SeparatorTemplate</td>
<td>Specifies what should appear between items (and alternating items).</td>
</tr>
<tr>
<td>SelectedItemTemplate</td>
<td>Specifies the content and layout of selected items.</td>
</tr>
<tr>
<td>EditItemTemplate</td>
<td>Specifies the content and layout of edited items.</td>
</tr>
<tr>
<td>HeaderTemplate</td>
<td>Specifies the content and layout of the header of the list.</td>
</tr>
<tr>
<td>FooterTemplate</td>
<td>Specifies the content and layout of the footer of the list.</td>
</tr>
</tbody>
</table>
Data-Binding Expressions

- Within the various item templates (e.g., ItemTemplate, EditItemTemplate), you need to specify which data source fields you want to display.

- ASP.NET provides a special declarative syntax for specifying data elements, generally referred to as **data-binding expressions**.

- The syntax for a data-binding expression is:

```csharp
<%# expression %>
```

- E.g.

```csharp
<%# Eval("LastName") %>
<%# Eval("Price", "{0:c}" ) %>
```

**composite formatting string**
DataList Control

- The developer specifies which particular data items in the data source will be displayed in a repeating list.
- This is done by using data-binding expressions within the ItemTemplate element.

```xml
<asp:SqlDataSource ID="dsPubs" runat="server"
    ProviderName="System.Data.SqlClient"
    ConnectionString="<%$ ConnectionStrings:BookCatalog %>"
    SelectCommand="Select PublisherId, PublisherName From Publishers" />

<asp:DataList ID="dlstPub" runat="server" DataSourceID="dsPubs">
    <ItemTemplate>
        <%-# Eval("PublisherName") %>
    </ItemTemplate>
</asp:DataList>
```
<asp:DataList ID="dlstPub" runat="server" DataSourceID="dsPubs"
  CellPadding="5" RepeatLayout="Table">

  <HeaderTemplate>
    <h2>Publishers</h2>
  </HeaderTemplate>

  <ItemTemplate>
    <%# Eval("PublisherName") %>
  </ItemTemplate>

  <ItemStyle BackColor="AntiqueWhite" ForeColor="Brown"
    Font-Size="Medium" Width="150" Height="25" />

  <AlternatingItemStyle BackColor="silver" />
  <SeparatorStyle BackColor="crimson" Height="5" />
  <SeparatorTemplate></SeparatorTemplate>

</asp:DataList>
Design-Time Support

15 Data Controls

COPYRIGHT © 2007 RANDY CONNOLLY

CORE INTERNET APPLICATION DEVELOPMENT WITH ASP.NET 2.0
Prentice Hall, 2007
www.randyleonell.com/core
Repeater Control

- Displays items in a repeating list.
- Unlike the DataList control, the Repeater does not have an inherent look; for example, it does not by default create a vertical list of data or a HTML `<table>` element.
- Instead, you must provide the layout for the control via templates.
  - The Repeater only creates the HTML that you specify within the ItemTemplate element.
- Although this usually means more work (by the developer) is required to display a simple list of data, it may be preferred by developers who prefer the complete control that the Repeater provides over the generated HTML.
<asp:Repeater ID="rptArtists" runat="server" 
    DataSourceID="dsBooks">
    <HeaderTemplate>
        <h2 class="listHeading">Recent Books</h2>
    </HeaderTemplate>
    <ItemTemplate>
        <dt><%# Eval("Title") %></dt>
        <dd class="bookImage">
            <asp:Image ID="imgBook" runat="server"
                ImageUrl='<%# String.Format("images/{0}.gif",Eval("ISBN")) %>'
                AlternateText='<%# Eval("Title") %>' />
        </dd>
        <dd class="publisher">
            <%# Eval("PublisherName") %>
            (<%# Eval("YearPublished") %>)
        </dd>
        <dd>
            <%# Eval("BriefDescription") %>
        </dd>
    </ItemTemplate>
    <FooterTemplate>
</FooterTemplate>
</asp:Repeater>
FormView Control

- Displays a single record of data.
- Provides a built-in mechanism for navigating from record to record;
- Supports the updating, insertion, and deletion of a record.
<asp:FormView ID="frmvBooks" runat="server" DataSourceID="dsBooks" CssClass="entireForm">

<HeaderTemplate>
<h2>Book Details</h2>
</HeaderTemplate>

<HeaderStyle CssClass="singleBookTitle" />

<ItemTemplate>
<div class="singleBook">
ISBN: <br />
<asp:Label ID="labIsbn" runat="server"
    Text='<%# Eval("ISBN") %>' CssClass="dataValues" />
<br />
Title: <br />
<asp:Label ID="labTitle" runat="server"
    Text='<%# Eval("Title") %>' CssClass="dataValues" />
<br />
Publisher:<br />
<asp:Label ID="labPublisher" runat="server"
    Text='<%# Eval("PublisherName") %>'
    CssClass="dataValues" />
<br />
Year:<br />
<asp:Label ID="labDesc" runat="server"
    Text='<%# Eval("YearPublished") %>'
    CssClass="dataValues" />
<br />
Description:<br />
<asp:Label ID="lab1" runat="server"
    Text='<%# Eval("BriefDescription") %>'
    CssClass="dataValues" />
<br />
</div>
</ItemTemplate>
</asp:FormView>
Although the FormView control displays a single record of data, it can accept a data source containing multiple records. You can allow the user to move from record to record by setting the control’s AllowPaging property to true.
The FormView control can also be used to provide the user with the ability to edit the data in a single record.
DetailsView Control

- Displays a single record of data at a time, provides a way to navigate from record to record, and supports the updating, insertion, or deletion of a record.

- The DetailsView control differs from the FormView control in that it does not require the use of templates for defining the look of the control.
  - it renders each field in the data source as a row in an HTML `<table>` element.
<asp:DetailsView ID="detvBooks" runat="server" DataSourceID="dsBooks" />

Test DetailsView

This example illustrates how to use a DetailsView control.

ISBN 0201033012
Title Design Patterns: Elements of Reusable Object-Oriented Software
PublisherName Addison-Wesley
PublisherId 1
YearPublished 1994

Capturing a wealth of experience about the design of object-oriented software, four top-notch designers present a catalog of simple and succinct solutions to commonly occurring design problems. Previously undocumented, these 23 patterns allow designers to create more flexible, elegant, and ultimately reusable designs without having to rediscover the design solutions themselves.
DetailsView Control

```xml
<asp:DetailsView ID="detvBooks" runat="server"
    DataSourceID="dsBooks" GridLines="None"
    AllowPaging="true" CellPadding="4" ForeColor="#333333">
    <RowStyle VerticalAlign="Top" BackColor="#EFF3FB"/>
    <PagerStyle CssClass="detailsPagerStyle"/>
    <FieldHeaderStyle BackColor="#DEE8F5" Font-Bold="True"/>
    <HeaderStyle CssClass="detailsHeaderStyle"/>
    <AlternatingRowStyle BackColor="White"/>
</asp:DetailsView>
```
GridView Control

- Provides a way to display data in a tabular (row-column) grid.
- You can use different field types to customize the visual display of the data within the grid.
- Also like the DetailsView control, the GridView supports the paging of data as well as the automatic modification and saving of data when bound to a data source control.
- The GridView also allows the user to sort or select data rows.
<asp:GridView ID="grdSample" runat="server" DataSourceID="dsBooks"/>
<asp:GridView ID="grdSample" runat="server"
    DataSourceID="dsBooks"
    AutoGenerateColumns="False" BackColor="White"
    BorderColor="#DEDFDE" BorderStyle="None" BorderWidth="1px"
    CellPadding="4" GridLines="Vertical" ForeColor="Black">

    <Columns>
        <asp:ImageField HeaderText="" DataImageUrlField="Isbn"
            DataImageUrlFormatString="images/{0}.gif"
            DataAlternateTextField="Title"
            DataAlternateTextFormatString="Book cover for {0}"
            ReadOnly="True" />
        <asp:BoundField DataField="ISBN" HeaderText="Isbn"
            ReadOnly="True"/>
        <asp:BoundField DataField="Title" HeaderText="Title"/>
        <asp:BoundField DataField="PublisherName"
            HeaderText="Publisher"/>
        <asp:BoundField DataField="YearPublished"
            HeaderText="Year"/>
    </Columns>

    <RowStyle CssClass="gridRow" />
    <HeaderStyle CssClass="gridHeader" />
    <AlternatingRowStyle CssClass="gridAlt" />
</asp:GridView>
GridView Control

CommandField

ButtonField
Pagination

- The GridView control supports pagination.
  - That is, rather than display all the records in the data source, pagination lets the control break its listing up into multiple, smaller pages displaying a subset of the complete data.
  - The user can then move to a specific page of data via pagination navigation buttons.

- Pagination can be enabled for any GridView, FormView, or DetailsView control by enabling its AllowPaging property.
  - The number of records to appear in each page can be set via the PageSize property.
Pagination

- How does it work?
  - The answer depends on what type of data source control is being used.
    - If a SqlDataSource control is being used, pagination effectively operates at the control level.
    - If an ObjectDataSource control is being used, pagination must be implemented by the underlying object.
In a GridView with pagination using a SqlDataSource control,

- all the records that match the SELECT statement are read and stored into an internal DataSet every time the page is requested or posted back, regardless of whether pagination is enabled or not enabled.

- If the SELECT statement returns 100000 records, but the GridView only displays 10 records per page, this is clearly an inefficient approach for a frequently requested page.

- In such a case, it would be better to refactor so that only the required data for the page is being requested by the database.

- This can be done at the DBMS level or at an ObjectDataSource level.
GridView Sorting

- The GridView control allows the user to change the visual sort order of the data displayed within the control.
- When sorting is enabled for the GridView (by setting the AllowSorting property of the control to true), the user can change the sort order of the data by clicking the hyperlinked column header for that field.
- Furthermore, the user can toggle between an ascending or descending sort order.
<asp:GridView ID="grdSample" runat="server" 
    DataSourceID="dsBooks" 
    AutoGenerateColumns="False" CellPadding="4" 
    GridLines="Vertical" CssClass="bookList" 
    AllowSorting="true" >

    <Columns>
        <asp:BoundField DataField="ISBN" HeaderText="Isbn" 
            SortExpression="Isbn" />
        <asp:BoundField DataField="Title" HeaderText="Title" 
            SortExpression="Title" />
        <asp:BoundField DataField="PublisherName" 
            HeaderText="Publisher" 
            SortExpression="PublisherName" />
        <asp:BoundField DataField="YearPublished" 
            HeaderText="Year" 
            SortExpression="YearPublished" />
    </Columns>

    ... 
</asp:GridView>
The GridView control supports the in-place editing of data within a row by the user.

In fact, you can add editing to a GridView control in almost exactly the same manner as with the DetailsView control.
Nested GridView Controls

- You can nest GridView controls in order to display data that has a one-to-many relationship.