Unit 10

11.0 Rich Controls (Text Book Chapter 10)

11.0 Rich Controls

Rich controls are web controls that model complex user interface elements. Although no strict definition exists for what is and what isn't a rich control, the term commonly describes a web control that has an object model that's distinctly separate from the HTML it generates. A typical rich control can be programmed as a single object (and added to a web page with a single control tag) but renders itself using a complex sequence of HTML elements. Rich controls can also react to user actions (like a mouse click on a specific region of the control) and raise more meaningful events that your code can respond to on the web server. In other words, rich controls give you a way to create advanced user interfaces in your web pages without writing lines of convoluted HTML.

In this chapter, you'll take a look at several web controls that have no direct equivalent in the world of ordinary HTML. You'll start with the Calendar, which provides slick date-selection functionality. Next, you'll consider the **AdRotator**, which gives you an easy way to insert a randomly selected image into a web page. Finally, you'll learn how to Create sophisticated pages with multiple views using two advanced container controls: the **MultiView** and the Wizard. These controls allow you to pack a miniature application into a single page. Using them, you can handle a multistep task without redirecting the user from one page to another.

NOTE:
ASP.NET includes numerous rich controls that are discussed elsewhere as needed in these class notes, including rich data controls, security controls, and controls tailored for web portals. In this chapter, you'll focus on a few useful web controls that don't fit neatly into any of these categories. All of these controls appear in the Standard tab of the Visual Studio Toolbox.

11.0 Rich Controls

11.1 Calendar Control

11.1.1 Default Calendar

The Calendar control presents a miniature calendar that you can place in any web page. Like most rich controls, the Calendar can be programmed as a single object (and defined in a single simple tag), but it renders itself with dozens of lines of HTML output.

```
<asp:Calendar id="MyCalendar" runat="server" />
```

The Calendar control presents a single-month view, as shown in Figure 12 The user can navigate from month to month using the navigational arrows, at which point the page is posted back and ASP.NET automatically provides a new page with the correct month values. You don't need to write any additional event-handling code to manage this process. When the user clicks a date, the date becomes highlighted in a gray box (by default). You can retrieve the selected day in your code as a **DateTime** object from the **Calendar.SelectedDate** property. This basic set of features may provide everything you need in your application. Alternatively, you can configure different selection modes to allow users to select entire weeks or months or to render the control as a static calendar that doesn't allow selection. The only fact you must remember is that if you allow month selection, the user can also select a single week or a day. Similarly, if you allow week selection, the user can also select a single day.

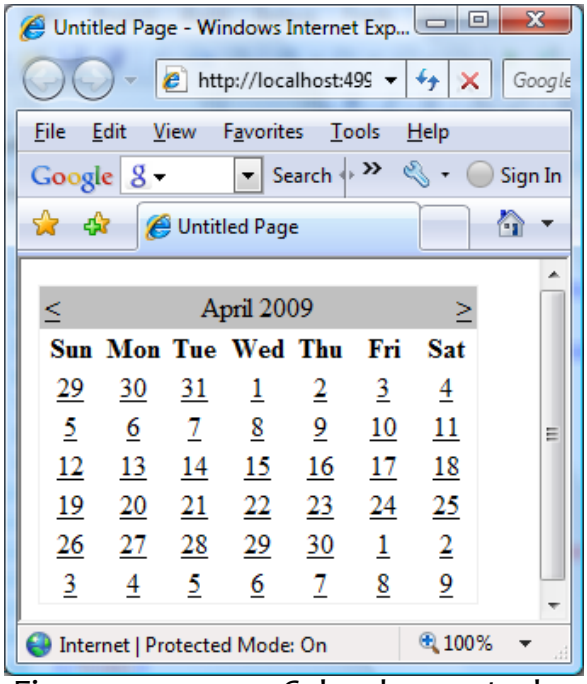

Figure 11.1-1– asp:Calendar control

You set the type of selection through the **Calendar.SelectionMode** property.

You may also need to set the **Calendar.FirstDayOfWeek** property to configure how a week is selected. (For example, set **FirstDayOfWeek** to the enumerated value Sunday, and weeks will be selected from Sunday to Saturday.)

➢11.0 Rich Controls

When you allow multiple date selection, you need to examine the **SelectedDates** property, which provides a collection of all the selected dates. You can loop through this collection using the `foreach` syntax. The following code demonstrates this technique:

You see a slight modification to the asp:Calendar where the SelectedMode property is now set to "DayWeekMonth". So you can select a day or a week or a whole month at a time.

```
<asp:Calendar id="MyCalendar" SelectionMode="DayWeekMonth" runat="server" />
<asp:Label id="lblDates"  runat="server" />
```

Example 11.1-1a Calendar Control with D/M/W Selectable

```
lblDates.Text = "You selected these dates:<br />";

foreach (DateTime dt in MyCalendar.SelectedDates)
{
   lblDates.Text += dt.ToLongDateString() + "<br />";
}
```

Example 11.1-1b Snippet of code behind

Here are some screen shots

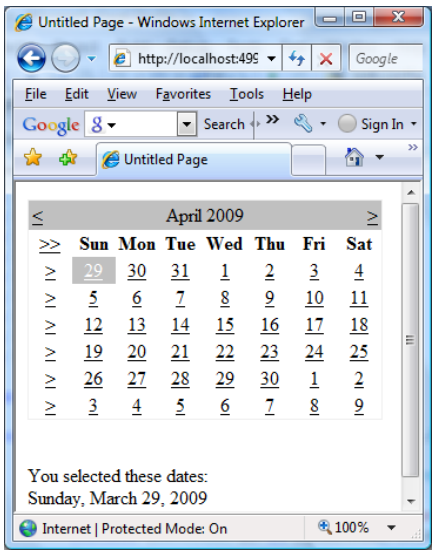

Figure 11.1-2a

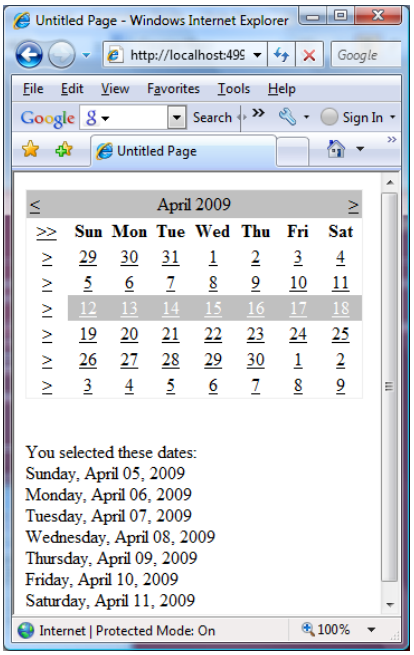

Figure 11.1-2b

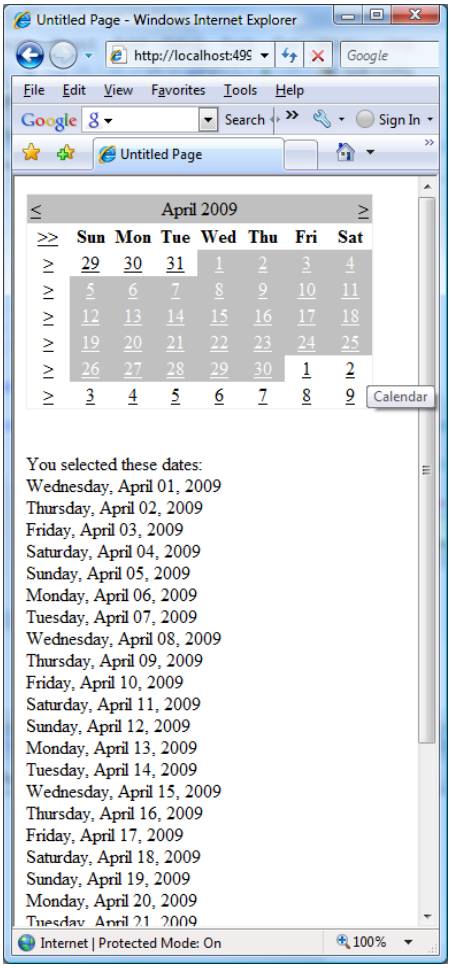

Figure 11.1-2c

11.1.2 Other Common Calendar Properties

The Calendar control provides a whole host of formatting-related properties. You can set various parts of the calendar, like the header, selector, and various day types, by using one of the style properties (for example, **WeekendDayStyle**).

Each of these style properties references a full-featured **TableItemStyle** object that provides properties for coloring, border style, font, and alignment. Taken together, they allow you to modify almost any part of the calendar's appearance. Table in figure 11.1-3 lists the style properties that the Calendar control provides.

Table Properties for Calendar Styles

Member	Description
DayHeaderStyle	The style for the section of the Calendar that displays the days of the week (as column headers).
DayStyle	The default style for the dates in the current month.
NextPrevStyle	The style for the navigation controls in the title section that move from month to month.
OtherMonthDayStyle	The style for the dates that aren't in the currently displayed month. These dates are used to "fill in" the calendar grid. For example, the first few cells in the topmost row may display the last few days from the previous month.
SelectedDayStyle	The style for the selected dates on the calendar.
SelectorStyle	The style for the week and month date-selection controls.
TitleStyle	The style for the title section.
TodayDayStyle	The style for the date designated as today (represented by the TodaysDate property of the Calendar control).
WeekendDayStyle	The style for dates that fall on the weekend.

Figure 11.1-3 – Common style properties in asp:Calendar

You can adjust each style using the Properties window. For a quick shortcut, you can set an entire related color scheme using the Calendar's Auto Format feature. To do so, start by selecting the Calendar on the design surface of a web form. Then, click the arrow icon that appears next to its top-right corner to show the Calendar's smart tag, and click the Auto Format link. You'll be presented with a list of predefined formats that set the style properties, as shown in Figure 11.1-4.

Using BackColor, ForeColor and TodayDayStyle properties:
Here is an example where some of the properties said above are customized in a asp:Calendar control.

```
<asp:Calendar id="MyCalendar" SelectionMode="DayWeekMonth" runat="server" DayHeaderStyle-
    BackColor="#FF3300" TodayDayStyle-ForeColor="#FF9933" WeekendDayStyle-
    ForeColor="#0033CC" /> <br /> <br />
```

Example 11.1-2 Using Common Style properties on a asp:Calendar

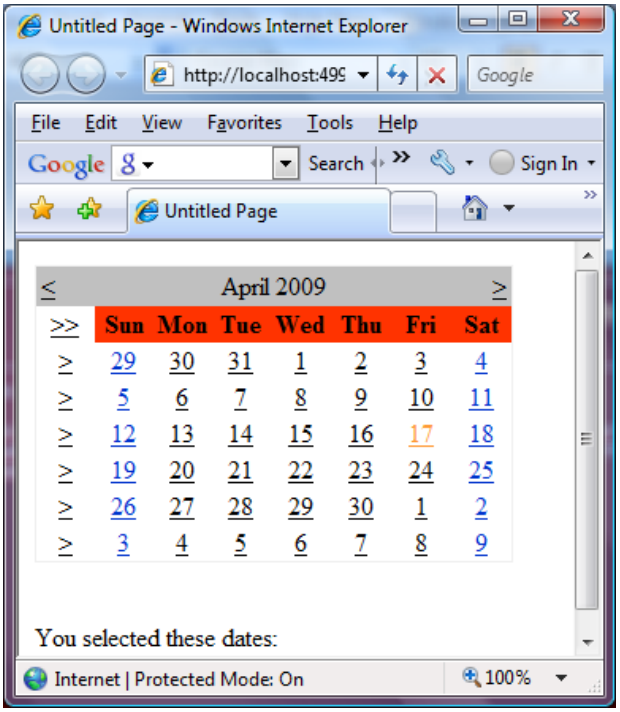

Figure 11.1-4Using Common Style properties on a asp:Calendar

You can also use additional properties to hide some elements or conFigure the text they display. For example, properties that start with "Show" (such as ShowDayHeader, ShowTitle, and ShowGridLines) can be used to hide or show a specific visual element. Properties that end in "Text" (such as PrevMonthText, NextMonthText, and SelectWeekText) allow you to set the text that's shown in part of the calendar.

Restricting Access to Days:

In most situations where you need to use a calendar for selection, you don't want to allow the user to select any date in the calendar. For example, the user might be booking an appointment or choosing a delivery date-two services that are generally provided only on set days. The Calendar control makes it surprisingly easy to implement this logic. In fact, if you've worked with the date and time controls on the Windows platform, you'll quickly recognize that the ASP.NET versions are far superior.

The basic approach to restricting dates is to write an event handler for the **Calendar.DayRender** event. This event occurs when the Calendar control is about to create a month to display to the user. This event gives you the chance to examine the date that is being added to the current month (through the **e.Day** property) and decide whether it should be selectable or restricted.

The following code makes it impossible to select any weekend days or days in years greater than 2013:

```
protected void MyCalendar_DayRender(Object source, DayRenderEventArgs e)
{
   // Restrict dates after the year 2010 and those on the weekend.
   if (e.Day.IsWeekend || e.Day.Date.Year > 2010)
   {
      e.Day.IsSelectable = false;
   }
}
```

Example 11.1-3 Restricting access to days

The **e.Day** object is an instance of the **CalendarDay class**. CalendarDay class provides various properties. Table in figure 11.1-5 describes some of the most useful.

Table CalendarDay Properties	
Property	**Description**
Date	The DateTime object that represents this date.
IsWeekend	True if this date falls on a Saturday or Sunday.
IsToday	True if this value matches the Calendar.TodaysDate property, which is set to the current day by default.
IsOtherMonth	True if this date doesn't belong to the current month but is displayed to fill in the first or last row. For example, this might be the last day of the previous month or the next day of the following month.
IsSelectable	Allows you to conFigure whether the user can select this day.

Figure 11.1-5 CalenderDay class properties

Day**Render Event:**

The **DayRender** event is extremely powerful. Besides allowing you to tailor what dates are selectable, it also allows you to configure the cell where the date is located through the **e.Cell property**. (The calendar is displayed using an HTML table.) For example, you could highlight an important date or even add information. Here's an example that highlights a single day—the fifth of May—by adding a new Label control in the table cell for that day:

```
protected void MyCalendar_DayRender(Object source, DayRenderEventArgs e)
{
   // Check for May 5 in any year, and format it.
   if (e.Day.Date.Day == 5 && e.Day.Date.Month == 5)
   {
      e.Cell.BackColor = System.Drawing.Color.Yellow;

   // Add some static text to the cell.
      Label lbl = new Label();
      lbl.Text = "<br />My Birthday!";
      e.Cell.Controls.Add(lbl);
   }
}
```

Example 11.1-4 Using CalendarDay properties

➢11.0 Rich Controls

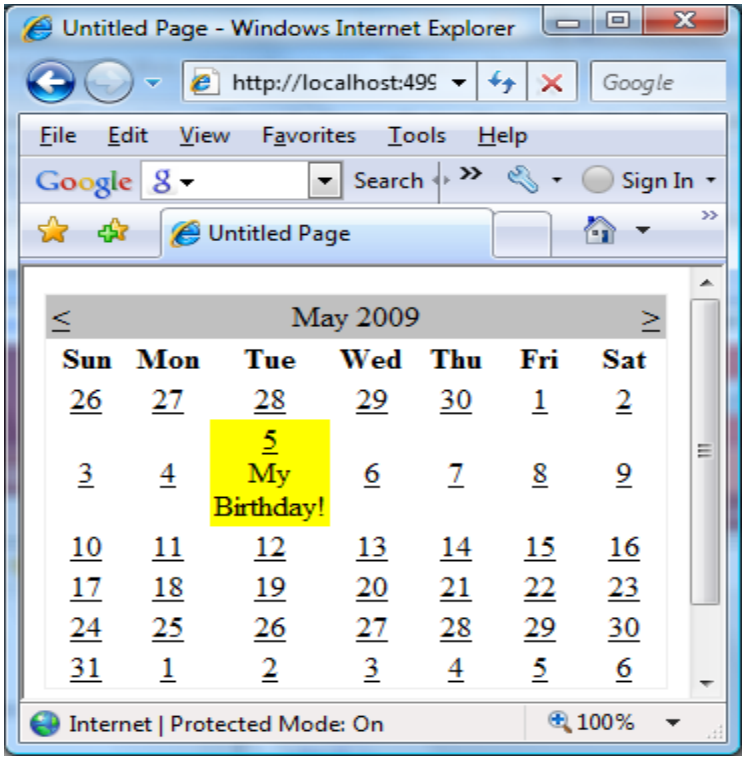

Figure 11.1-6 DayRender

The Calendar control provides two other useful events: SelectionChanged and VisibleMonthChanged. These occur immediately after the user selects a new day or browses to a new month (using the next month and previous month links). You can react to these events and update other portions of the web page to correspond to the current calendar month. For example, you could design a page that lets you schedule a meeting in two steps. First, you choose the appropriate day. Then, you choose one of the available times on that day.

The following code demonstrates an example how SelectionChanged event is trapped and customize the calendar.

➢11.0 Rich Controls

```
<form id="form1" runat="server">
 <div>
  <asp:Calendar ID="MyCalendar" runat="server"
      SelectionMode="DayWeekMonth"
     ShowGridLines="True"
      OnSelectionChanged=
                "MyCalendar_SelectionChanged">

       <SelectedDayStyle BackColor="Yellow"
                  ForeColor="Red">
       </SelectedDayStyle>

     </asp:Calendar>

  <asp:Label id="lblDates"  runat="server" />
 </div>
</form>
```

Example 11.1-5a Handling SectionChanged event - Presentation

```
protected void MyCalendar_SelectionChanged
                    (Object source, EventArgs e)
{
 //Clear the current text.
  lblDates.Text = "";

 //Iterate through the SelectedDates collection
 //and display the dates selected in the Calendar
  //control.
 foreach(DateTime day in
     MyCalendar.SelectedDates)
 {
  lblDates.Text +=
    day.Date.ToShortDateString() + "<br />";
 }
}
```

Example 11.1-5b Handling SectionChanged event code behind

Output of example 11.1-5a and 11.1-5b is show in Figure 11.1-7.

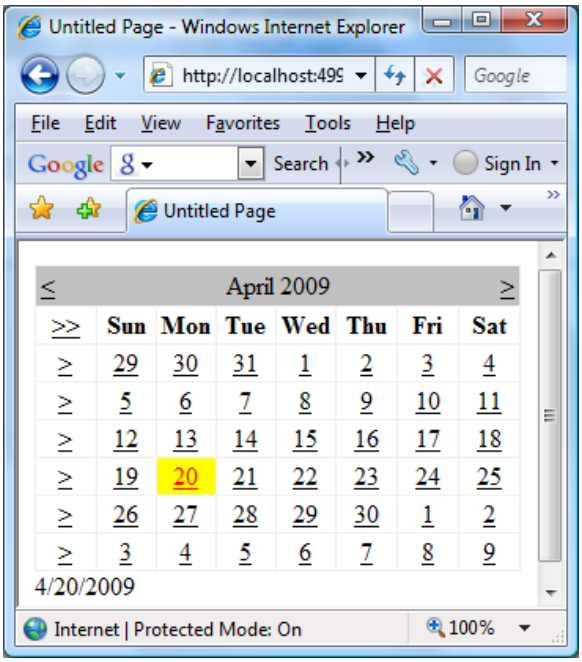

Figure 11.1-7

See other Calendar Class Members

11.2 AdRotator Control

The basic purpose of the AdRotator is to provide a graphic on a page that is chosen randomly from a group of possible images. In other words, every time the page is requested, an image is selected at random and displayed, which is the "rotation" indicated by the name AdRotator. One use of the AdRotator is to show banner-style advertisements on a page, but you can use it any time you want to vary an image randomly.

Using ASP.NET, it wouldn't be too difficult to implement an AdRotator type of design on your own. You could react to the Page.Load event, generate a random number, and then use that number to choose from a list of predetermined image files. You could even store the list in the web.config file so that it can be easily modified separately as part of the application's configuration. Of course, if you wanted to enable several pages with a random image, you would either have to repeat the code or create your own custom control. The AdRotator provides these features for free. The Advertisement File

The AdRotator stores its list of image files in an XML file. This file uses the format shown here:

11.2.1 The Advertisement File

The AdRotator stores its list of image files in an XML file. This file uses the format shown here:

```
<Advertisements>
<Ad>
<ImageUrl>prosetech.jpg</ImageUrl>
<NavigateUrl>http://www.prosetech.com</NavigateUrl>
<AlternateText>ProseTech Site</AlternateText>
<Impressions>1</Impressions>
<Keyword>Computer</Keyword>
</Ad>
</Advertisements>
```

➢11.0 Rich Controls

This example shows a single possible advertisement, which the **AdRotator** control picks at random from the list of advertisements. To add more advertisements, you would create multiple <Ad> elements and place them all inside the root <Advertisements> element:

```
<Advertisements>
<Ad>
<!-- First ad here. -->
</Ad>
<Ad>
<!-- Second ad here. -->
</Ad>
</Advertisements>
```

Each <Ad> element has a number of other important properties that configure the link, the image and the frequency, as described in the table in figure 11.2-1

Element	Description
ImageUrl	The image that will be displayed. This can be a relative link (a file in the current directory) or a fully qualified Internet URL.
NavigateUrl	The link that will be followed if the user clicks the banner. This can be a relative or fully qualified URL.
AlternateText	The text that will be displayed instead of the picture if it cannot be displayed. This text will also be used as a tooltip in some newer browsers.
Impressions	A number that sets how often an advertisement will appear. This number isrelative to the numbers specified for other ads. For example, a banner with the value 10 will be shown twice as often (on average) as the banner with the value 5.
Keyword	A keyword that identifies a group of advertisements. You can use this for filtering.For example, you could create ten advertisements and give half of choose to filter the possible advertisements to include only one of these groups.choose to filter the possible advertisements to include only one of these groups.them the keyword Retail and the other half the keyword Computer. The web page can then choose to filter the possible advertisements to include only one of these groups.

Figure 11.2-1 - AdRotator Properties

11.2-2 The AdRotator Class

The actual AdRotator class provides a limited set of properties. You specify both the appropriate advertisement file in the AdvertisementFile property and the type of window that the link should follow (the Target window). The target can name a specific frame, or it can use one of the values defined in table in figure 11.2-2.

Target	Description
_blank	The link opens a new unframed window.
_parent	The link opens in the parent of the current frame.
_self	The link opens in the current frame.
_top	The link opens in the topmost frame of the current window (so the link appears in the full window).

Figure 11.2-1 - AdRotator Target Window Types

Optionally, you can set the KeywordFilter property so that the banner will be chosen from a specific keyword group. This is a fully configured

AdRotator tag:

```
<asp:AdRotator id="Ads" runat="server"
    AdvertisementFile="MainAds.xml"
Target="_blank" KeywordFilter="Computer" />
```

some related content or a link, as shown in Figure 11.2-3

Additionally, you can react to the AdRotator.AdCreated event. This occurs when the page is being created and an image is randomly chosen from the advertisements file. This event provides you with information about the image that you can use to customize the rest of your page. For example, you might display some related content or a link, as shown in Figure 11.2-3

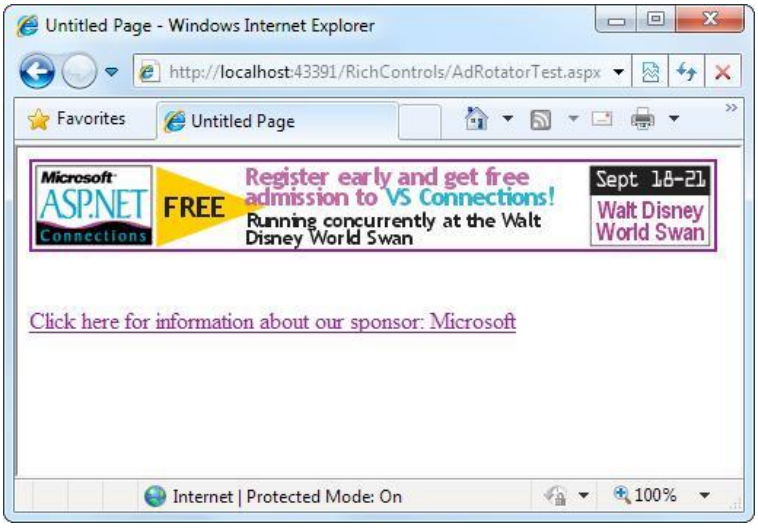

Figure 11.2-3 - AdRotator An AdRotator with synchronized content

The event handling code for this example simply configures a HyperLink control named lnkBanner based on the randomly selected advertisement:

```
protected void Ads_AdCreated(Object sender, AdCreatedEventArgs e)
{
//Synchronize the Hyperlink control.
  lnkBanner.NavigateUrl = e.NavigateUrl;
//Syncrhonize the text of the link.
  lnkBanner.Text = "Click here for information about our sponsor: ";
  lnkBanner.Text += e.AlternateText;
}
```

As you can see, rich controls such as the Calendar and AdRotator don't just add a sophisticated HTML output; they also include an event framework that allows you to take charge of the control's behavior and integrate it into your application.

11.3 Multiview Control

The MultiView control represents a control that acts as a container for groups of View controls. It allows you to define a group of View controls, where each View control contains child controls, for example, in an online survey application.

Your application can then render a specific View control to the client based on criteria such as user identity, user preferences, or information passed in a query string parameter. The following figure illustrates a MultiView control hosting 3 View controls.

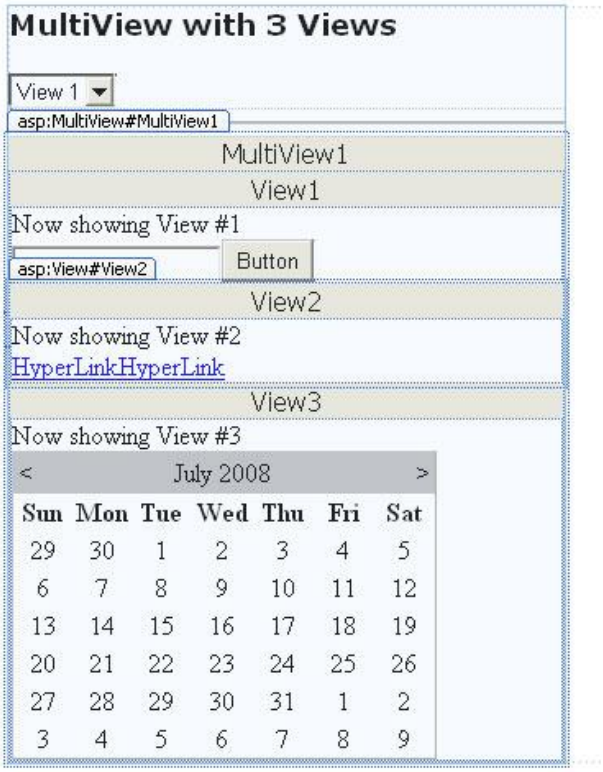

Figure 11.3-1 - 3 View MultiView control

The source code for the above example is shown in next page.

11.0 Rich Controls

```
<%@ Page Language="C#" AutoEventWireup="true"
 CodeFile="how-to-use-MultiView-c.aspx.cs" Inherits="how_to_use_MultiView_c" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
                      "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
<title>How to Use MultiView</title>
</head>
<body>
   <form id="form1" runat="server">
   <div style="width: 339px">
     <h3>
          <font face="Verdana">MultiView with 3 Views</font>
        </h3>
        <asp:DropDownList ID="DropDownList1“ OnSelectedIndexChanged="DropDownList1_SelectedIndexChanged"
          runat="server" AutoPostBack="True">
          <asp:ListItem Value="0">View 1</asp:ListItem>
          <asp:ListItem Value="1">View 2</asp:ListItem>
          <asp:ListItem Value="2">View 3</asp:ListItem>
        </asp:DropDownList><br />
        <hr />
        <asp:MultiView ID="MultiView1" runat="server" ActiveViewIndex="0">
          <asp:View ID="View1" runat="server">
            Now showing View #1<br />
            <asp:TextBox ID="TextBox1" runat="server"></asp:TextBox><strong> </strong>
            <asp:Button ID="Button1" runat="server" Text="Button" /></asp:View>
          <asp:View ID="View2" runat="server">
            Now showing View #2<br />
            <asp:HyperLink ForeColor="#FF9933" ID="HyperLink1" runat="server"
              NavigateUrl="http://www.asp.net">HyperLink</asp:HyperLink>
           <asp:HyperLink ForeColor="#FF9933" ID="HyperLink2" runat="server“
                    NavigateUrl="http://www.asp.net">HyperLink</asp:HyperLink></asp:View>
          <asp:View ID="View3" runat="server">
            Now showing View #3<br />
            <asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
          </asp:View>
        </asp:MultiView>
   </div>
   </form>
</body>
</html>
```

Fortunately, the MultiView includes some built-in smarts that can save you a lot of trouble. Here's how it works: the MultiView recognizes button controls with specific command names. (Technically, a button control is any control that implements the IButtonControl interface, including the Button, ImageButton, and LinkButton.) If you add a button control to the view that uses one of these recognized command names, the button gets some automatic functionality. Using this technique, you can create navigation buttons without writing any code.

Table in figure 11.3-2 lists all the recognized command names. Each command name also has a corresponding static field in the MultiView class, so you can easily get the right command name if you choose to set it programmatically.

Table Recognized Command Names for the MultiView

Command Name	MultiView Field	Description
PrevView	PreviousViewCommandName	Moves to the previous view.
NextView	NextViewCommandName	Moves to the next view.
SwitchViewByID	SwitchViewByIDCommandName	Moves to the view with a specific ID (string name). The ID is taken from the CommandArgument property of the button control.
SwitchViewByIndex	SwitchViewByIndexCommandName	Moves to the view with a specific numeric index. The index is taken from the CommandArgument property of the button control.

Figure 11.3-2 - MultiView Command Names

11.4 GridView Control

A recurring task in software development is to display tabular data. ASP.NET provides a number of tools for showing tabular data in a grid, including the GridView control. With the GridView control, you can display, edit, and delete data from many different kinds of data sources, including databases, XML files, and business objects that expose data.

You can use the GridView control to do the following:

- Automatically bind to and display data from a data source control.
- Select, sort, page through, edit, and delete data from a data source control.

Additionally, you can customize the appearance and behavior of the GridView control by doing the following:

- Specifying custom columns and styles.
- Utilizing templates to create custom user interface (UI) elements.
- Adding your own code to the functionality of the GridView control by handling events.

11.4-2 Data Binding with the GridView Control

The GridView control provides you with two options for binding to data:

Data binding using the DataSourceID property, which allows you to bind the ridView control to a data source control. This is the recommended approach because it allows the GridView control to take advantage of the capabilities of the data source control and provide built-in functionality for sorting, paging, and updating.

Data binding using the DataSource property, which allows you to bind to various objects, including ADO.NET datasets and data readers. This approach requires you to write code for any additional functionality such as sorting, paging, and updating.

11.0 Rich Controls

When you bind to a data source using the DataSourceID property, the GridView control supports two-way data binding. In addition to the control displaying returned data, you can enable the control to automatically support update and delete operations on the bound data. The following example demonstrates how a data source could be bounded to a GridvView.

```
<%@ Page Language="C#" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<script runat="server">
</script>
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
 <title>GridView example: how to use GridView in asp.net</title>
</head>
<body>
  <form id="form1" runat="server">
  <div>
    <asp:GridView ID="GridView1" runat="server" AllowPaging="True"
      AllowSorting="True" AutoGenerateColumns="False" DataKeyNames="CustomerID"
      DataSourceID="SqlDataSource1">
      <Columns>
        <asp:BoundField DataField="CustomerID" HeaderText="CustomerID" ReadOnly="True"
          SortExpression="CustomerID" />
        <asp:BoundField DataField="CompanyName" HeaderText="CompanyName"
          SortExpression="CompanyName" />
        <asp:BoundField DataField="ContactName" HeaderText="ContactName"
          SortExpression="ContactName" />
        <asp:BoundField DataField="ContactTitle" HeaderText="ContactTitle"
          SortExpression="ContactTitle" />
        <asp:BoundField DataField="Address" HeaderText="Address"
          SortExpression="Address" />
        <asp:BoundField DataField="City" HeaderText="City" SortExpression="City" />
        <asp:BoundField DataField="Region" HeaderText="Region"
          SortExpression="Region" />
        <asp:BoundField DataField="PostalCode" HeaderText="PostalCode"
          SortExpression="PostalCode" />
        <asp:BoundField DataField="Country" HeaderText="Country"
          SortExpression="Country" />
        <asp:BoundField DataField="Phone" HeaderText="Phone" SortExpression="Phone" />
        <asp:BoundField DataField="Fax" HeaderText="Fax" SortExpression="Fax" />
      </Columns>
    </asp:GridView>
    <asp:SqlDataSource ID="SqlDataSource1" runat="server"
      ConnectionString="<%$ ConnectionStrings:AppConnectionString1 %>"
      SelectCommand="SELECT * FROM [Customers]"></asp:SqlDataSource>
  </div>
  </form>
</body>
</html>
```

Example 11.4-1 Binding a Data Source to a GridView Control

➢11.0 Rich Controls

CustomerID	CompanyName	ContactName	ContactTitle	Address	City	Region	PostalCode	Country	Phone	Fax
TORTU	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	Region	05023	Mexico	(5) 555-3932	
TRADH	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	Region	05023	Mexico	(5) 555-3932	
TRAIH	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	Region	05023	Mexico	(5) 555-3932	
VAFFE	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	Region	05023	Mexico	(5) 555-3932	
VICTE	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	Region	05023	Mexico	(5) 555-3932	
VINET	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	Region	05023	Mexico	(5) 555-3932	
WANDK	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	Region	05023	Mexico	(5) 555-3932	
WARTH	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	Region	05023	Mexico	(5) 555-3932	
WELLI	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	Region	05023	Mexico	(5) 555-3932	
WHITC	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	México D.F.	Region	05023	Mexico	(5) 555-3932	
1 2 3 4 5 6 7 8 9 10										

Figure 11.4-1 - GridView output from a Database Table

We will discuss more about data binding with GridView in a later chapter.