The Calendar Control

This control creates a functionally rich and good-looking calendar box that shows one month at a time. The user can move from month to month, select a date, and even select a range of days (if multiple selection is allowed). The Calendar control has many properties that, taken together, allow you to
change almost every part of this control. For example, you can fine-tune the foreground and background colors, the font, the title, the format of the date, the currently selected date, and so on. The Calendar also provides events that enable you to react when the user changes the current month (VisibleMonthChanged), when the user selects a date (SelectionChanged), and when the Calendar is about to render a day (DayRender).
The following Calendar tag sets a few basic properties:

```
<asp:Calendar runat="server" ID="Calendar1" ForeColor="red"
BackColor="lightyellow" />
```

The most important Calendar event is SelectionChanged, which fires every time a user clicks a date.
Here's a basic event handler that responds to the SelectionChanged event and displays the selected date:

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
    lblDates.Text = "You selected: " +
    Calendar1.SelectedDate.ToLongDateString();
    }
```

Note Every user interaction with the calendar triggers a postback. This allows you to react to the selection event immediately, and it allows the Calendar to rerender its interface, thereby showing a new month or newly selected dates. The Calendar does not use the AutoPostBack property.
You can also allow users to select entire weeks or months as well as single dates, or you can 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. The type of selection is set through the Calendar.SelectionMode property. You may also need to set the Calendar.FirstDayOfWeek property to configure how a week is selected. (For example, if you set FirstDayOfWeek to the

enumerated value Monday, weeks will be selected from Monday to Sunday.)

When you allow multiple date selection (by setting Calendar.SelectionMode to something other than Day), you need to examine the SelectedDates property instead of the SelectedDate property. SelectedDates provides a collection of all the selected dates, which you can examine, as shown here:

```
protected void Calendar1_SelectionChanged(object sender, EventArgs e)
    {
        lblDates.Text = "You selected these dates:<br />";
        foreach (DateTime dt in Calendar1.SelectedDates)
    {
        lblDates.Text += dt.ToLongDateString() + "<br />";
    }
    }
```

The Calendar control exposes many more formatting-related properties, many of which map to the underlying HTML table representation (such as CellSpacing, CellPadding, Caption, and CaptionAlign). Additionally, you can individually tweak portions of the controls through grouped formatting settings called styles (which expose color, font, and alignment options). Example properties include DayHeaderStyle, DayStyle, NextPrevStyle, OtherMonthDayStyle, SelectedDayStyle, TitleStyle, TodayDayStyle, and WeekendDayStyle. You can change the subproperties for all of these styles using the Properties window.

Finally, by handling the DayRender event, you can completely change the appearance of the cell
being rendered. 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 control is really a sophisticated HTML table.) For example, you could highlight an important date or even add extra controls or HTML content in the cell. Here's an example that changes the background and foreground colors of the weekend days and also makes them nonclickable so that the user can't choose those days:

```
protected void Calendar1_DayRender(object sender, DayRenderEventArgs
e)
    {
```

```
        if (e.Day.IsWeekend)
    {
        e.Cell.BackColor = System.Drawing.Color.Green;
        e.Cell.ForeColor = System.Drawing.Color.Yellow;
        e.Day.IsSelectable = false;
    }
    }
```

Figure 4-16 shows the result.

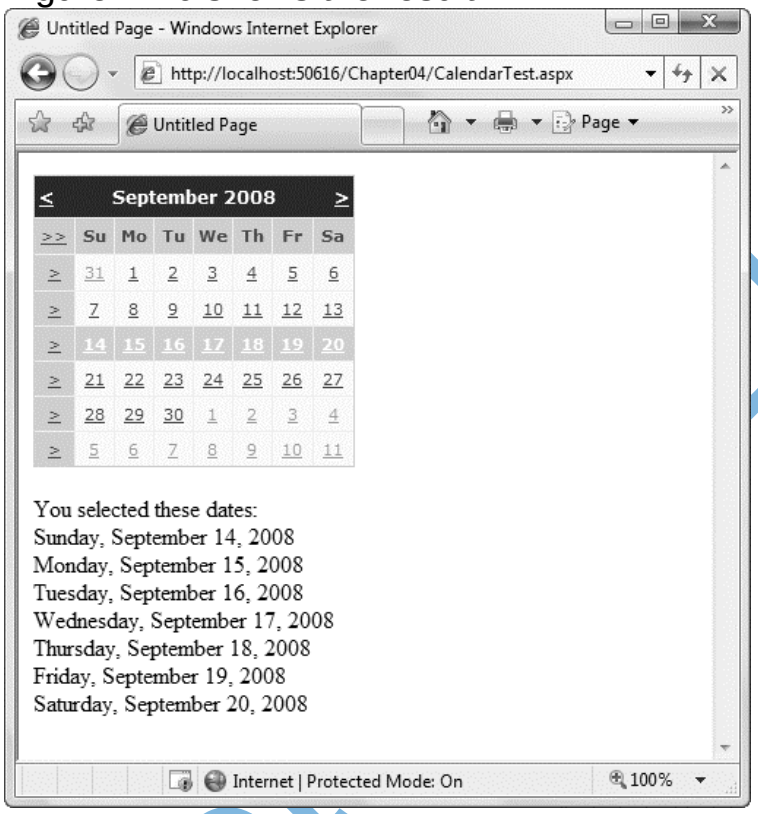

Figure 4-16. The Calendar control

Tip If you're using a design tool such as Visual Studio, you can even set an entire related color scheme using the built-in designer. Simply select the Auto Format link in the smart tag. You'll be presented with a list of predefined formats that set various style properties.

Summary

In this chapter you learned the basics of the core server controls included with ASP.NET, such as HTML server controls, web controls, list controls, validation controls, and rich controls. You also learned how to use ASP.NET controls from your web-page code, access their properties, and

handle their server-side events. Finally, you learned how to validate potentially problematic user input with the validation controls. In the next chapter, you'll learn how pages come together to form web applications.