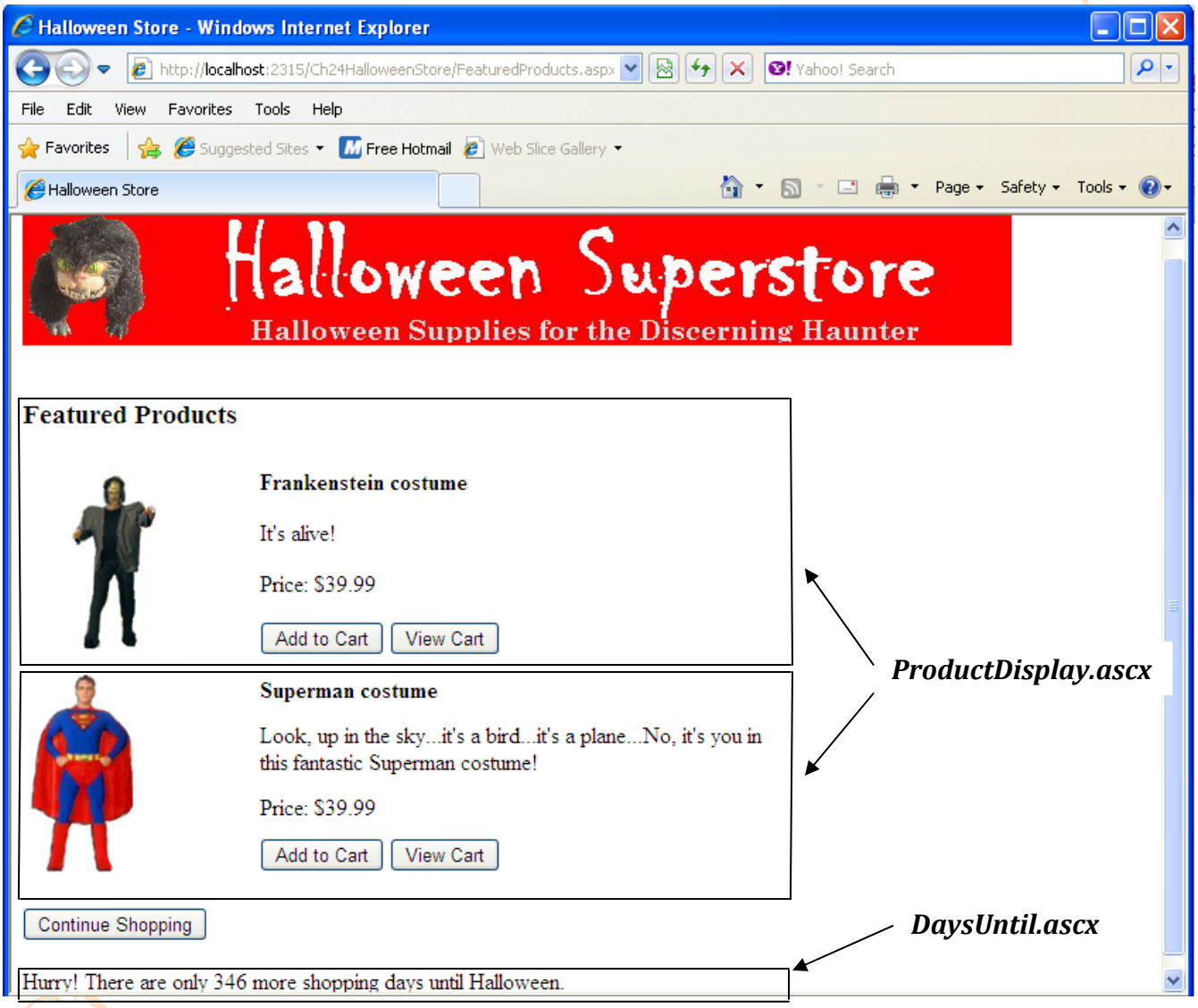
# Lecture 9

## User controls development

A ***user control*** is a special type of ASP.NET page that can be added to other pages. In figure below, you can see a web page that contains three user controls.




This page uses three user controls.

When you decide what to include in each user control, keep in mind that the major benefit of user controls is that they are reusable. That means that you want to design a user control so it can be used in two or more places within a web site.

If you need to, you can change all the pages that contain a user control by changing the user control. In contrast, if you don't use a user control, you have to change each page separately.

**The main concepts:**

- A *user control* is a special type of ASP.NET web page that has been converted to a control. A user control is stored in a separate file so that it can be added to any form in a project.
- User controls are typically used to develop page elements that can be used in multiple places within a web site.
- Like an ASP.NET web page, a user control can handle *Page* events such as the *Load* event. You code the event handlers for those events in the code-behind file for the control.

**A procedure for creating and using user controls**

1. Add a new user control to the project
2. Design the user control using the User Control Designer
3. Add the code to implement the control using the Code Editor
4. Add the user control to any form in the project
5. If necessary, set any properties of the user control using the *Properties* window
6. If necessary, add code to the form that contains the user control using the Code Editor.

After you create a user control, you can add it to any form in the project.

## Creating of the user controls

To add a user control to a project you need:

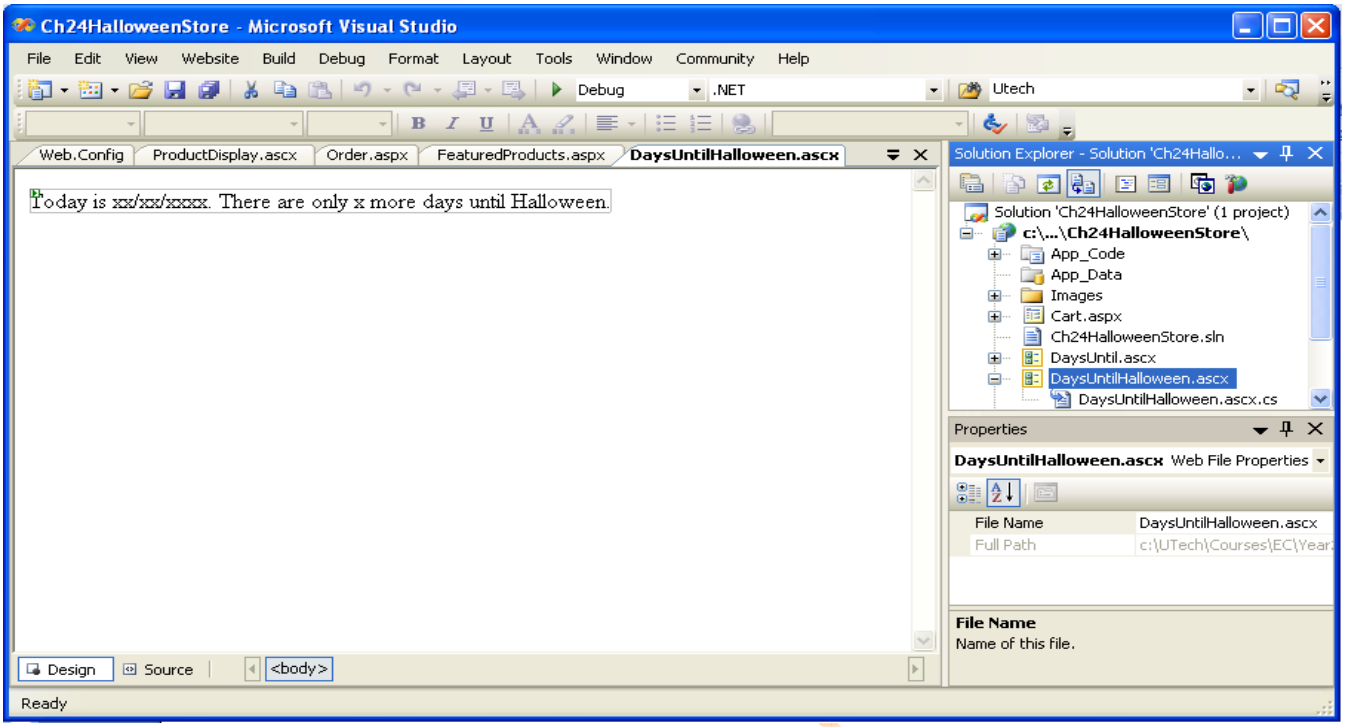
1. Use the *Website* → *Add New Item* command to display the *Add New Item* dialog box. Then, use the *Web User Control* template to create a user control with the name you specify.
2. When you add a user control, it will appear in the Solution Explorer with an extension of *ascx*.
3. You can use the *User Control Designer* to add ASP.NET controls and HTML elements to the control. A user control should not include a *Body* or a *Form* element.
4. The *ascx* code for a user control begins with a *Control Directive* that's generated by Visual Studio when you create the user control. This directive includes *Language*, *AutoEventWireup*, *CodeFile*, and *Inherits* attributes just as the *Page* directive of a page does.

**The *ascx* code for the *DaysUntilHalloween* control**

```
<%@ Control Language="C#" AutoEventWireup="true"
   CodeFile="DaysUntilHalloween.ascx.cs" Inherits="DaysUntilHalloween" %>

<asp:Label ID="lblDaysUntilHalloween" runat="server">
   Today is xx/xx/xxxx. There are only x more days until Halloween.
</asp:Label>
```

This user control shown below:




## Writing a programming code for a user control

Before begin coding user controls, you need to be familiar with the sequence of events that are raised for a page with user control:

1. The *Load* event for the page
2. The *Load* event for each user control
3. Control events caused by user interaction with the page
4. The *PreRender* event for the page
5. The *PreRender* event for each user control

To start ASP.NET raises the *Load* event of the page. Then, it raises the *Load* event for each user control that the page contains. By default, the *Load* for the user controls are raised in the order in which the controls are added to the form. To avoid problems, however, you should code your user controls so they don't depend on other user controls.

After the *Load* events are raised and processed, ASP.NET raises the control events that result from the user interacting with the page. If the user selects a check box, for example, the *CheckChanged* event of that control is raised. Finally, the *PreRender* event of the page is raised, followed by the *PreRender* event of each user control.

### The C# code for the *DaysUntilHalloween* user control

```
public partial class DaysUntilHalloween : System.Web.UI.UserControl
{
  protected void Page_Load(object sender, EventArgs e)
  {
    if (!IsPostBack)
    {
```

```
            lblDaysUntilHalloween.Text = "Today is " + DateTime.Today.ToShortDateString();
            lblDaysUntilHalloween.Text += ". There are only ";
            lblDaysUntilHalloween.Text += this.GetDaysUntilHalloween();
            lblDaysUntilHalloween.Text += " more days until Halloween.";
        }
    }

    private int GetDaysUntilHalloween()
    {
        DateTime halloween = new DateTime(DateTime.Today.Year, 10, 31);

        if (DateTime.Today > halloween)
            halloween = halloween.AddYears(1);

        TimeSpan timeUntil = halloween - DateTime.Today;

        return timeUntil.Days;
    }
}
```

The *UserControl* class provides many of the same properties as the *Page* class. For example, you can use the *Request* and *Response* properties to work with the *HttpRequest* and *HttpResponse* objects, and you can use the *Session* property to work with the *HttpSessionState* object. You can also use the *IsPostBack* property to determine whether the page that contains the user control is being displayed for the first time or if it is being posted back to the server.

In addition to the properties, methods, and events a user control inherits from the *UserControl* class, you can add your own properties, methods, and events to a user control. Then, you can access those properties, methods, and events from the page that contains the control.

**Description:**

- You can use the Code Editor to add code to the code-behind file that's generated for a user control. The code-behind file should only contain code that's directly related to the user control. It shouldn't contain code that depends on code in other controls.
- You can define properties and methods within a user control and then access them from the page that contains the control. You can also define and raise events within a user control and then respond to them from the page that contains the control.
- The class for a user control inherits the *System.Web.UI.UserControl* class. This class includes many of the same properties as a page, including *IsPostBack, Request, Response, Server*, and *Session*. This class also contains a *Page* property that refers to the *Page* object that contains the control.

## How to expose properties, methods, and events

The C# code for the *DaysUntil* control, presented below, exposes four properties. The first two properties let the developer change the text that's displayed before and after the number of days until a specified date. And the second two properties let the developer specify the date by setting the month and the day. As a result, you could display a message that indicates the number of days until Christmas or any other day of the year. This makes the control more flexible and increase the chance that it will be reused.

```
using System.ComponentModel;

public partial class DaysUntil : System.Web.UI.UserControl
{
   private string textBefore = "There are only";
   private string textAfter = "days until Halloween.";
   private int month = 10;
   private int day = 31;

   [Category("Appearance")]
   public string TextBefore
   {
      get
      {
         return textBefore;
      }
      set
      {
         textBefore = value;
      }
   }

   [Category("Appearance")]
   public string TextAfter
   {
      get
      {
         return textAfter;
      }
      set
      {
         textAfter = value;
      }
   }

   [Category("Appearance")]
   public int Month
```

```
{
    get
    {
        return month;
    }
    set
    {
        month = value;
    }
}

[Category("Appearance")]
public int Day
{
    get
    {
        return day;
    }
    set
    {
        day = value;
    }
}
```

The values of four properties are stored in the private variables that are declared at the beginning of the class. These variables are assigned default values that are used if the developer doesn't set the corresponding properties.

After the variables declarations is the code that declares the four public properties. Each property begins with a designer attribute that's code in square brackets. This attribute determines how the property will be displayed in the Web Forms Designer. In that example, all four properties include a *Category* attribute that specifies the category in the *Properties* window where the property will appear. Here, all four properties will be displayed in the *Appearance* category.

To use designer attributes like this, a user control must include the *System.ComponentModel* namespace as shown here. Then, the control can use any of the attributes defined by this namespace.

Below you can see the event handler for the Load event of the control.

```
protected void Page_Load(object sender, EventArgs e)
{
    lblDaysUntil.Text = textBefore + " ";
    lblDaysUntil.Text += this.DaysUntilDate();
    lblDaysUntil.Text += " " + textAfter;
}

private int DaysUntilDate()
{
    DateTime targetDate = new DateTime(DateTime.Today.Year, Month, Day);
```

```
        if (DateTime.Today > targetDate)
        {
            targetDate = targetDate.AddYears(1);
        }

        TimeSpan timeUntil = targetDate - DateTime.Today;

        return timeUntil.Days;
    }
}
```

To start, the value of the *TextBefore* property is assigned to the label, followed by a space. Then, the value returned by the *DaysUntilDate* method is appended to the label. This method uses the *Month* and *Day* properties to get the number of days until the specified date. Finally, a space and the value stored in the *TextAfter* property are appended to the label.

Although the *DaysUntil* control only exposes properties, it could also expose methods or events. To expose a method or event, you just code it with the public keyword like any other method or event. You can also expose a field by declaring it with the public keyword. However, you're more likely to use public properties since public fields aren't displayed in the *Properties* window of the Web Forms Designer.

## Converting a Page to a User Control

Sometimes the easiest way to develop a user control is to put it in a web page first, test it on its own, and then translate the page to a user control. Even if you don't follow this approach, you might still end up with a portion of a user interface that you want to extract from a page and reuse in multiple places.

Overall, this process is straightforward cut-and-past operation. However, you need to watch for a few points:

- Remove all *<html>, <body>,* and *<form>* tags. These tags appear once in a page, so they can't be added to user controls (which might appear multiple times in a single page).
- If there is a *Page* directive, change it to a *Control* directive and remove the attributes that the control directive does not support, such as *AspCompat, Buffer, ClientTarget, CodePage, Culture, EnableSessionState, EnableViewStateMac, ErrorPage, LCID, ResponseEncoding, Trace, TraceMode,* and *Transaction*.
- If you aren't using the code-behind model, make sure you still include a class name in the *Control* directive by supplying the *ClassName* attribute. This way, the web page that consumes the control can be strongly typed, which allows it to access properties and methods you've added to the control.
- Change the file extension from *.aspx* to *.ascx*.

## How to use user controls

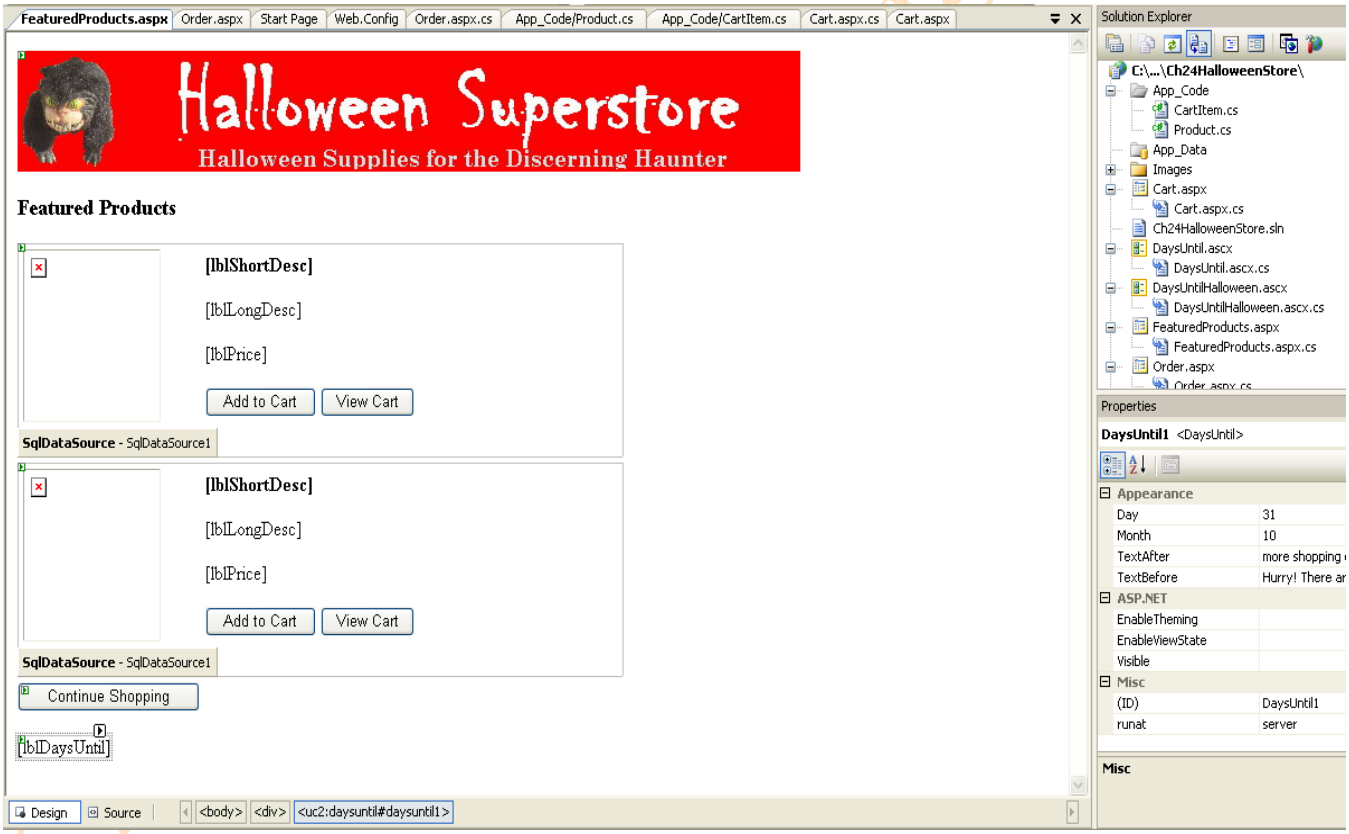
After you design and code a user control, you can add it to any form in the web site that needs it. To add a user control to a web form you simply drag the control from the Solution Explorer to the location

where you want it displayed on the form. Then, ASP.NET generates a tag that registers the control so it can be used on the form. In addition, it generates a tag that displays the control.

To set a public property of a user control, select the control in the Web Forms Designer and use the *Properties* window to set the property. Any custom properties you have defined for the control appear in this window, along with some of the properties provided by the *UserControl* class.

The form shown below is the *FeaturedProducts* form. It consists of the three user controls. The *DaysUntil* control at the bottom of the form is selected, and you can see the custom properties that are exposed by this control in the *Properties* window. Notice that all four of these properties – *Day, Month, TextAfter,* and *TextBefore* – are listed in the *Appearance* category of this window. That's because *"Appearance"* was specified for the *Category* attribute of each property.

In addition to the custom properties for a ser control, you can also set some of the properties that are defined by the *UserControl* class that all user controls inherit. In this figure, for example, you can see the *EnableTheming* property. In addition to this property, you can set the *EnableViewState, Visible, ID,* and *Runat* properties.




The *aspx* code below is an example of the code, generated by the Web Forms Designer when you add user control to a web form. As you can see, it starts with two *Register* directives that register the two user controls. Each *Register* directive includes three attributes: *Src, TagName, and TagPrefix*. The *Src* attribute specifies the name of the *ascx* file that defines the user control. The *TagName* attribute associates a name with the control. And the *TagPrefix* attribute associates a prefix with the control.

**The *aspx* code for the Feature Products form that uses user controls**

```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="FeaturedProducts.aspx.cs"
Inherits="FeaturedProducts" %>
<%@ Register Src="ProductDisplay.ascx" TagName="ProductDisplay" TagPrefix="uc1" %>
<%@ Register Src="DaysUntil.ascx" TagName="DaysUntil" TagPrefix="uc2" %>
<!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 id="Head1" runat="server">
   <title>Halloween Store</title>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:Image ID="Image1" runat="server"
         ImageUrl="~/Images/banner.jpg" /><br /><br />
      <h3>Featured Products</h3>
      <uc1:ProductDisplay id="ProductDisplay1" runat="server"
         ProductID="frankc01">
      </uc1:ProductDisplay>
      <uc1:ProductDisplay id="ProductDisplay2" runat="server"
         ProductID="super01">
      </uc1:ProductDisplay>
      <br />
      <asp:Button ID="btnContinue" runat="server"
         PostBackUrl="~/Order.aspx" Text="Continue Shopping" /><br />
      <br />
      <uc2:DaysUntil ID="DaysUntil1" runat="server"
         Day="31" Month="10"
         TextBefore="Hurry! There are only"
         TextAfter="more shopping days until Halloween." />
   </div>
   </form>
</body>

</html>
```

**Code that sets a property of the control from the page that contains it**

```
   protected void Page_Load(object sender, EventArgs e)
   {
      if (!IsPostBack)
         ddlProducts.DataBind();

      ProductDisplay1.ProductID = ddlProducts.SelectedValue;
   }
```

## Writing a code that works with properties, methods, and events

Fragment of code below sows how to wrte  code that works with the properties of a user control. The code example shows how you can use the *Load* event handler of the page to set a property of a user control. Specifically, it shows how you can set the *ProductID* property of the *ProductDisplay* control named *ProductDisplay1* to the product *ID* that's selected in the drop-down list named *ddlProducts*.

```
protected void Page_Load(object sender, EventArgs e)
{
   if (!IsPostBack)
      ddlProducts.DataBind();

   ProductDisplay1.ProductID = ddlProducts.SelectedValue;
}
```

Although this fragment of code only shows an example for working with properties, the same principles apply to working with methods or events. For example, if the class for a user control defines a public method, you can call it from the *Load* event handler of the page. In addition, if the class for a user control defines and raises a public event, the page that contains the control contain an event handler to handle that event.