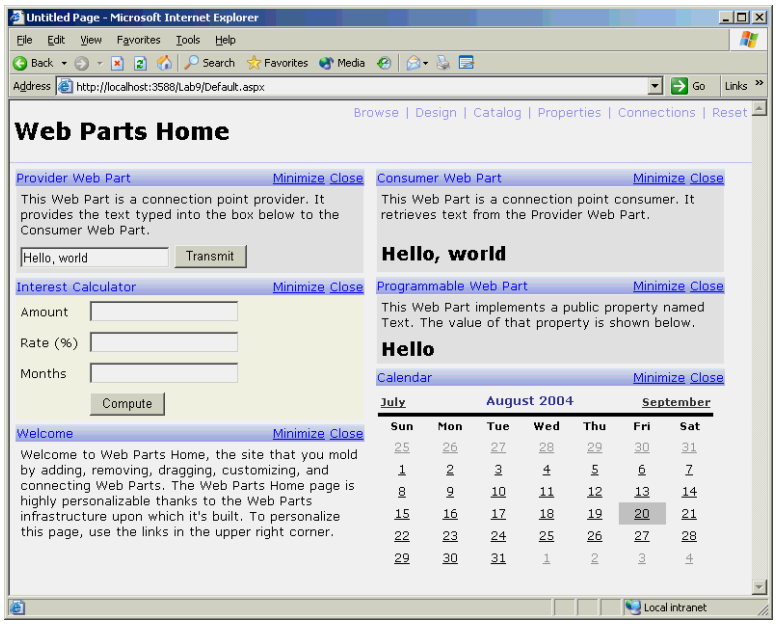
Lab 9: *Creating Personalizable applications using Web Parts*

Estimated time to complete this lab: 45 minutes

Web Parts is a framework for building highly customizable portal-style pages. You compose Web Parts pages from “Web Parts,” which can be Web controls, user controls, or custom controls. End users can customize Web Parts pages by changing the page layout, adding and removing Web Parts, editing Web Parts properties, establishing connections between Web Parts, and more. Changes made to a Web Parts page are persisted by the Web Parts framework. Web Parts pages exhibit a degree of sophistication seldom seen in Web applications today, and they do it without requiring you to write reams of code. In fact, as with many of the other new features in ASP.NET 2.0, the vast majority of what you can accomplish with the Web Parts can be accomplished declaratively.

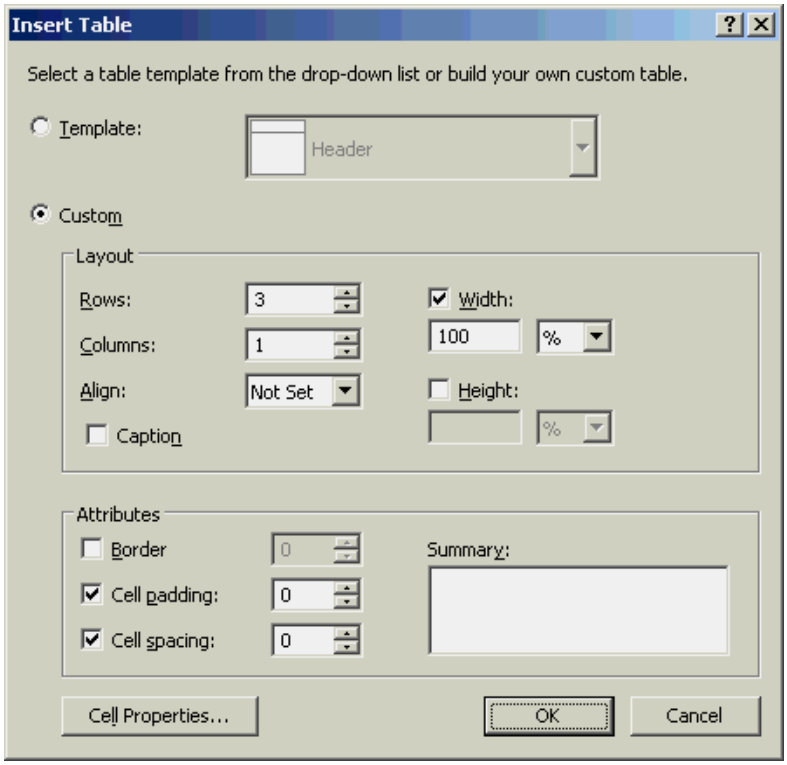
In this lab, you’ll build a Web Parts page and use it to familiarize yourself with the Web Parts infrastructure built into ASP.NET. Here’s a preview of what you’ll be building:



Exercise 1
Enable Failure Audit events

In this exercise, you'll create a new Web site and turn it into a Web Parts site. Then you'll add Web Parts and Web Part zones and stylize the zones to customize the look of the Web Parts.

Tasks	Detailed Steps
1. Create a Web Site	**a.** Start Microsoft Visual Studio. **b.** Select "New Web Site" from Visual Studio's File menu. **c.** In the New Web Site dialog, choose "Visual C# or Visual Basic" ype and "ASP.NET Web Site" as the template type. Type "C:\HOL\Web\Starter\<VB or CS>\lsb9 into the Location box and click OK to create the Web site.

2. Lay out the page	**a.** Open Default.aspx in the designer if it isn't open already and switch to Source view.
	b. Add the following statements to the <head> element:
	`<style>`
	`<!--`
	`  td { font-family: verdana; font-size: 10pt; }`
	`-->`
	`</style>`
	c. Modify the <body> element so that it reads as follows
	`<body topmargin="0" leftmargin="0" bottommargin="0" rightmargin="0">`
	d. Switch to Design view and add a WebPart Manager control to the page by dragging it from the Toolbox and dropping it onto the design surface.
	e. Use the "Layout->Insert Table" command to insert a 3-row, 1-column table beneath the WebPartManager. Set the table's width to 100% and cell padding and cell spacing to 0, as shown below
	 

f. Set the middle table row's height to 1 pixel and background color ("BgColor") to #c0c0ff.

g. Position the cursor in the top table row and use the "Layout->Insert Table" command to insert a 1-row, 2-column table into the row. Set the table's width to 100%, height to 64 pixels, cell padding to 4, and cell spacing to 0.

h. Switch to Source view and examine the HTML for the table that was just added. If the table's <td> tags contain attributes setting the cells' widths, delete those attributes.

i. Go back to Design view and place the cursor in the left-hand cell of the new table. Set the cell's Align property to "left" and its VAlign property to "middle."

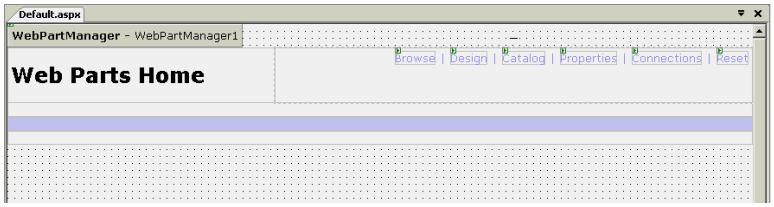
j. Place the cursor in the right-hand cell of the new table. Set the cell's Align property to "right" and its VAlign property to "top."

k. Place the cursor back in the left-hand table cell. Use the Formatting toolbar to set the text to bold 18-point Verdana. Then type "Web Parts Home."

l. Switch to Source view and insert the following statements between the <td> and </td> tags representing the right-hand table cell.

```
<span style="color: #9090ff">
   <asp:LinkButton ID="LinkButton1" ForeColor="#9090ff" Text="Browse"
    style="Text-Decoration: none" Runat="server" />
   |
   <asp:LinkButton ID="LinkButton2" ForeColor="#9090ff" Text="Design"
    style="Text-Decoration: none" Runat="server" />
   |
   <asp:LinkButton ID="LinkButton3" ForeColor="#9090ff" Text="Catalog"
    style="Text-Decoration: none" Runat="server" />
   |
   <asp:LinkButton ID="LinkButton4" ForeColor="#9090ff" Text="Properties"
    style="Text-Decoration: none" Runat="server" />
   |
   <asp:LinkButton ID="LinkButton5" ForeColor="#9090ff" Text="Connections"
    style="Text-Decoration: none" Runat="server" />
   |
   <asp:LinkButton ID="LinkButton6" ForeColor="#9090ff" Text="Reset"
    style="Text-Decoration: none" Runat="server" />
</span>
```

m. Switch back to Design view. Default.aspx should now look like this in the designer:

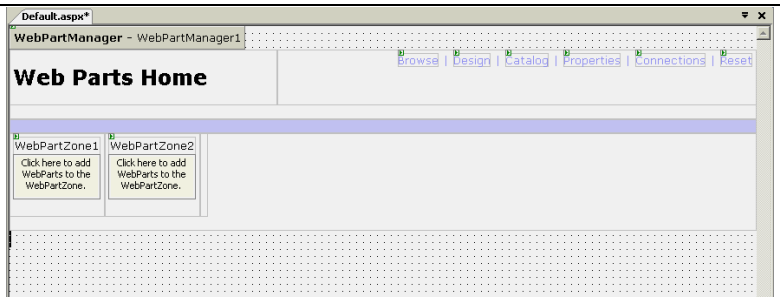


n. Position the cursor in the bottom row of the 3-row table that you added in Step 5. Then use the "Layout->Insert Table" command to insert a 1-row, 3-column table. Set the table's width to 3 pixels, cell padding to 4, and cell spacing to 0.

o. For each of the three cells in the table you just added, set the width to 1 pixel, the Align property to "left," and the VAlign property to "top."

p. Add a WebPartZone control to the page by dragging it from the Toolbox and dropping it into the leftmost cell of the table you just added.

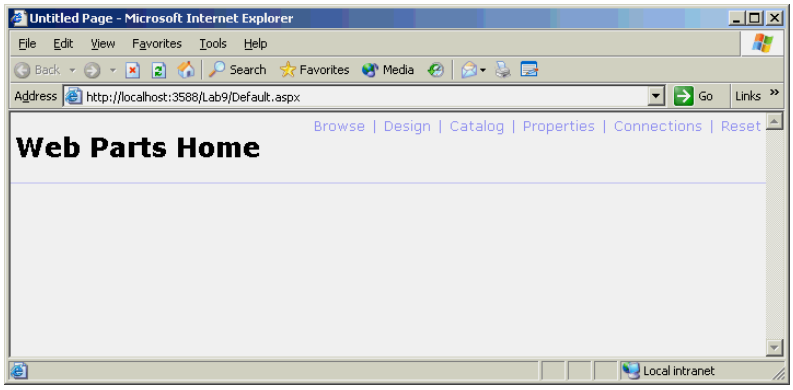
q. Add another WebPartZone control to the page by dragging it from the Toolbox and dropping it into the middle cell of the table you just added. Default.aspx should now look like this in the designer:



r. Select the first WebPartZone that you added (WebPartZone1). Then go to the Properties window and set the control's HeaderText property to "Zone 1."

s. Select the second WebPartZone that you added (WebPartZone2). Then go to the Properties window and set the control's HeaderText property to "Zone 2."

t. Press Ctrl+F5 to launch Default.aspx in your browser. Verify that the resulting page resembles the one shown below.



u. Close your browser and return to Visual Studio.

3. Add a "Welcome" Web Part

a. Right-click C:\..\Lab9 in the Solution Explorer window and select "Add New Item" from the context menu.

b. Select "Web User Control" as the template type and type "Welcome.ascx" into the Name box. Make sure C# is selected in the Language drop-down and check the "Place code in separate file" box. Then click OK to add a user control to the project.

c. Add the following HTML to Welcome.ascx underneath the @ Control directive:

```
<table width="360px" cellpadding="4" cellspacing="0">
  <tr>
    <td align="left" valign="top">
      Welcome to Web Parts Home, the site that you mold by adding,
      removing, dragging, customizing, and connecting Web Parts. The Web
      Parts Home page is highly personalizable thanks to the Web Parts
      infrastructure upon which it's built. To personalize this page, use
      the links in the upper right corner.
    </td>
  </tr>
</table>
```

d. Save your changes to Welcome.ascx and close it.

e. Return to Default.aspx in the designer and switch to Design view if you're not already in Design view. Then click inside WebPartZone1 where it says "Click here to add WebParts to the WebPartZone."

f. Drag Welcome.ascx from the Solution Explorer window and drop it into WebPartZone1.

g. Switch to Source view and manually add a Title="Welcome" attribute to the tag that

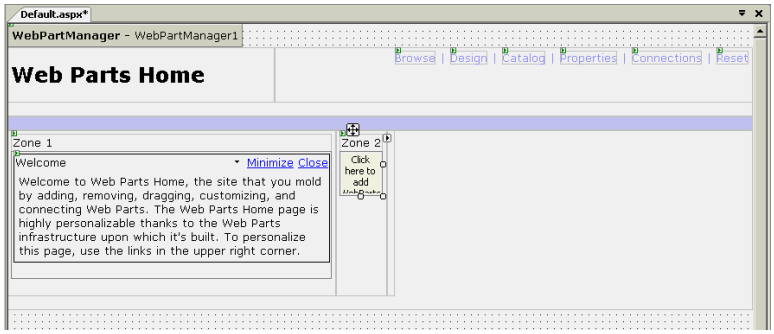
declares an instance of Welcome.ascx in the <ZoneTemplate>. Here's what the modified tag looks like:

```
<uc1:Welcome Title="Welcome" Runat="server" ID="Welcome1" />
```

NOTE:

Visual Studio will show a squiggly underline under the Title attribute as if the attribute is invalid. Visual Studio knows that the user control doesn't have a property named Title. What Visual Studio doesn't know is that at run-time, the user control will be wrapped in a GenericWebPart that DOES have a Title property. Therefore, you can ignore the squiggles.

h. Go back to Design view. Default.aspx should now look like this in the designer:



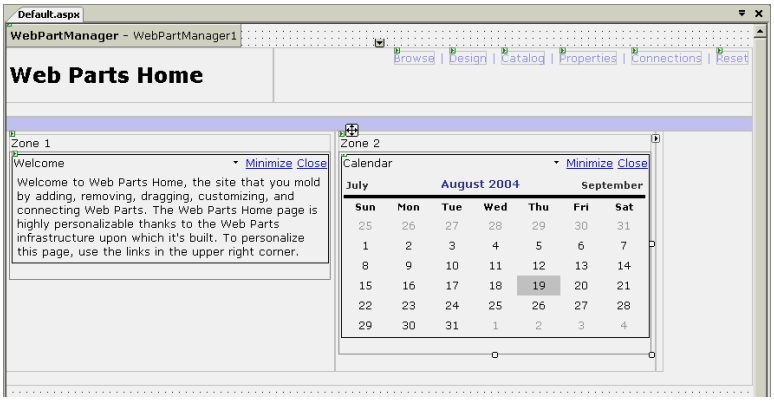
4. Add a Calendar Web Part

a. Any Web control can serve as a Web Part if it's added to a Web Part zone. To demonstrate, click "Click here to add WebParts to the WebPartZone" in Zone 2 and drop a Calendar control into it.

b. Use the "Auto Format" command to apply the "Professional 1" format to the Calendar.

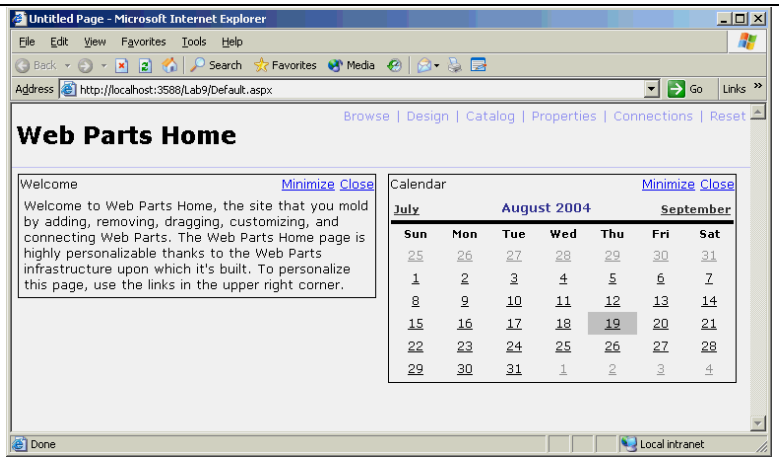
c. Switch to Source view and manually add a Title="Calendar" attribute to the <asp:Calendar> tag in the Web Part zone's <ZoneTemplate>.

d. Go back to Design view. Default.aspx should now look like this in the designer:



5. Stylize the Web Parts

a. Press Ctrl+F5 to launch Default.aspx in your browser. Here's what you should see:



b. Close your browser and return to Visual Studio.

c. The two Web Parts that you added to the page are visible, but at present they look very plain. To fix that, begin by switching to Source view and adding the following statements to the <style> element you added to Default.aspx in Task 2:

```
.TitleBar
{
   background-image: url("images/titlebargradient.jpg");
   background-repeat: repeat-x;
   background-position: 0 0;
   color: blue
}
```

d. Right-click C:\..\Lab9 in the Solution Explorer window and use the “New Folder” command to create a folder named Images.

e. Right-click the Images folder in Solution Explorer and use the “Add Existing Item” command to add TitleBarGradient.jpg to the Images folder. You’ll find TitleBarGradient.jpg in the C:\HOL\Web\LabFiles\Images directory.

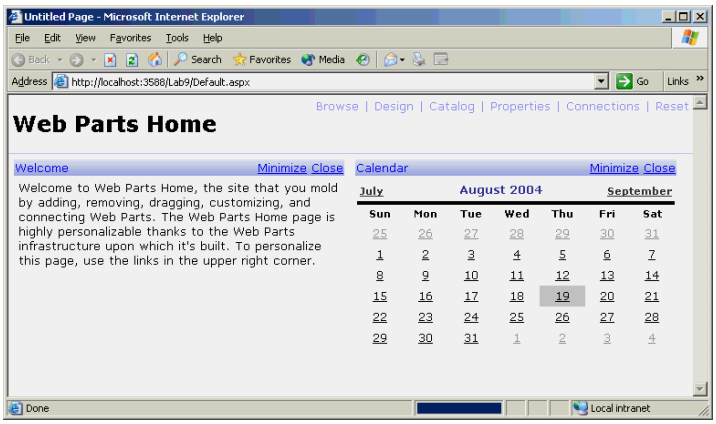
f. Switch back to Design view and select WebPartZone1. Then go to the Properties window and set the zone’s PartTitleStyle-CssClass property to “TitleBar” (referring to the CSS class you implemented in step 3).

g. In the Properties window, set WebPartZone1’s PartChromeType property to “TitleOnly.”

h. Select WebPartZone2 and set its PartTitleStyle-CssClass property to “TitleBar”.

i. In the Properties window, set WebPartZone2’s PartChromeType property to “TitleOnly.”

j. Press Ctrl+F5 to launch Default.aspx in your browser. Here’s what you should see:



k. Close your browser and return to Visual Studio.

Exercise 2
Enable interactive layout editing

.In this exercise, you'll add code to switch the page to Design display mode, thereby enabling the page layout to be modified interactively.

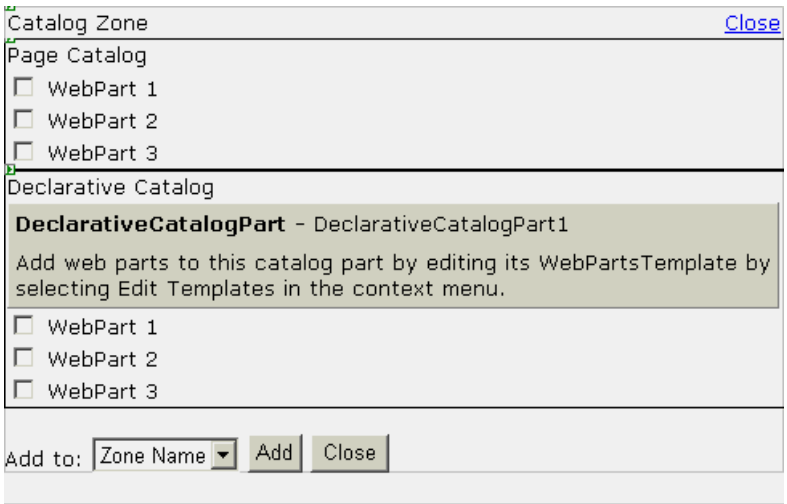
Tasks	Detailed Steps

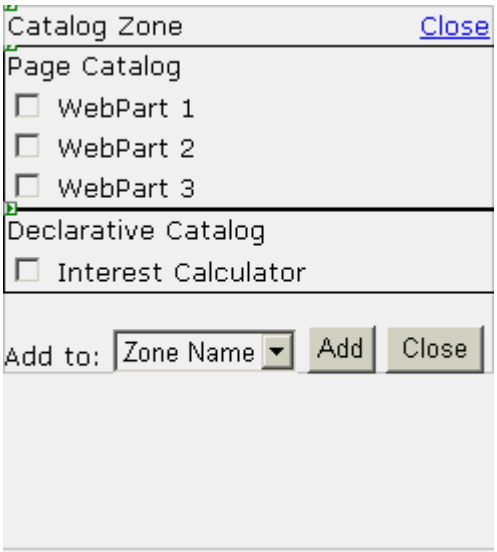
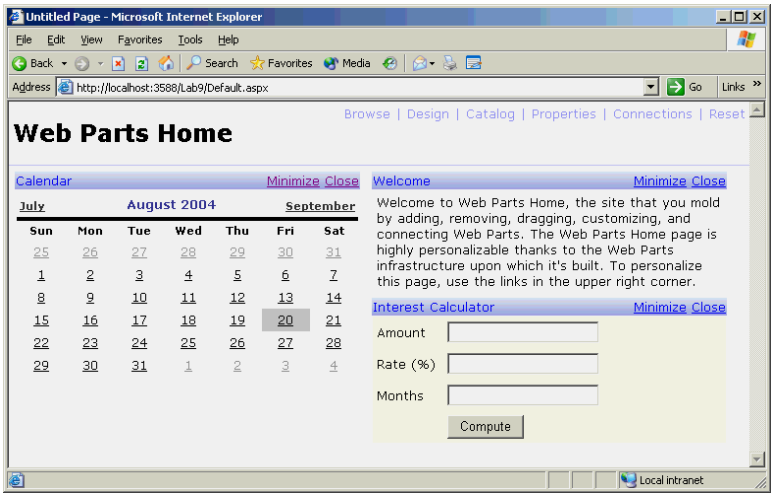
1. Add code to change display modes	**a.** Open Default.aspx in the designer and switch to Design view.
	b. Double-click the "Browse" LinkButton in the upper-right corner to add a handler for Click events. Add the following statement to the body of the handler:
	C#
	`WebPartManager1.DisplayMode = WebPartManager.BrowseDisplayMode;`
	VB
	`WebPartManager1.DisplayMode = WebPartManager.BrowseDisplayMode`
	c. Go back to Default.aspx in the designer and double-click the "Design" LinkButton. Add the following statement to the handler created by Visual Studio
	C#
	`WebPartManager1.DisplayMode = WebPartManager.DesignDisplayMode;`
	VB
	`WebPartManager1.DisplayMode = WebPartManager.DesignDisplayMode`
	d. Save your changes to Default.aspx.<cs or vb> and close it.

2. Edit the page layout	**a.** Press Ctrl+F5 to launch Default.aspx in your browser. **b.** Click the "Design" link in the page's upper-right corner to switch to Design display mode. **c.** Drag the Welcome Web Part from Zone 1 to Zone 2. **d.** Drag the Calendar Web Part from Zone 2 to Zone 1. **e.** Click the "Browse" link in the page's upper-right corner to switch back to Browse display mode. Congratulations! You just experienced drag-and-drop layout editing, Web Parts-style. **f.** Close your browser and return to Visual Studio

Exercise 3
Add a Catalog zone

In this exercise, you'll add a CatalogZone to Default.aspx, populate it with CatalogParts, and populate a DeclarativeCatalogPart with a prebuilt Web Part component. The CatalogZone will enable users to add new Web Parts to the page as well as restore Web Parts that are closed.

Tasks	Detailed Steps
1. Add another Web Part to the project	**a.** Right-click C:\..\Lab9 in the Solution Explorer window and use the "Add Existing Item" command to add Interest.ascx to the project. Interest.ascx is a prebuilt user control that you'll find in the C:\HOL\Web\LabFiles\Components folder. **b.** Confirm that Interest.ascx appears in Solution Explorer, and that there's a plus sign next to it. Click the plus sign and verify that Interest.ascx.cs was imported, also. It is OK to mix the languages in your application, so VB programmers will be importing this same file.
2. Add a CatalogZone control	**a.** Open Default.aspx in the designer and switch to Design view if you're not already in Design view. **b.** Drag a CatalogZone control from the Toolbox and drop it into the table cell to the right of the cell that contains WebPartZone2. **c.** Check the "View in catalog mode" box in the "Common CatalogZone Tasks" menu to show a WYSIWYG view of the CatalogZone. **d.** Click "Click here to add CatalogParts to the CatalogZone." **e.** Grab a PageCatalogPart control from the Toolbox and drop it into the CatalogZone. **f.** Grab a DeclarativeCatalogPart control from the Toolbox and drop it into the CatalogZone underneath the PageCatalogPart. The CatalogZone should now look like this in the designer: 
3. Add a Web Part to the DeclarativeCatalogPart	**a.** Click the small arrow next to the DeclarativeCatalogPart to display the "Common DeclarativeCatalogPart Tasks" menu and click "Edit Templates." **b.** Click "To edit this template, click here and type text or drag a control from the Toolbox." **c.** Drag Interest.ascx from the Solution Exporer window and drop it into the DeclarativeCatalogPart control.

	d. Display the “Common DeclarativeCatalogPart Tasks” menu and click “End Template Editing.” **e.** Switch to Source view and find the <uc2:Interest> tag in the DeclarativeCatalogPart’s <WebPartsTemplate>. Manually add a Title=“Interest Calculator” attribute to the tag. **f.** Go back to Design view. The CatalogZone control should now look like this in the designer: 
4. Add code to change display modes	**a.** Return to Default.aspx in the designer and double-click the “Catalog” LinkButton. Add the following statement to the handler created by Visual Studio: `WebPartManager1.DisplayMode = WebPartManager.CatalogDisplayMode;` **b.** Save your changes to Default.aspx.cs and close it.
5. Test the Catalog Zone	**a.** Press Ctrl+F5 to launch Default.aspx in your browser. **b.** Click the “Catalog” link in the page’s upper-right corner. **c.** When the catalog appears, click “Declarative Catalog (1).” Then check the “Interest Calculator” box and click Add. An Interest Calculator Web Part should appear in Zone 1. **d.** Drag the Interest Calculator Web Part from Zone 1 to Zone 2, positioning it below the Welcome Web Part. **e.** Click “Browse” in the page’s upper-right corner to return to Browse display mode. The page should look like this in the browser: 

	f. Use the Interest Calculator Web Part to calcuate the monthly payment that you'd incur if you borrowed $100,000 for 240 months at 10% interest. **g.** Click the Close button in the Welcome Web Part. The Web Part should disappear from the page. **h.** Click the "Catalog" link in the page's upper-right corner. **i.** When the catalog appears, click "Page Catalog (1)." Then check the "Welcome" box, select "Zone 2" in the drop-down list, and click Add. The Welcome Web Part should reappear in Zone 2. **j.** Click "Browse" in the page's upper-right corner to return to Browse display mode. **k.** Close your browser and return to Visual Studio.

Exercise 4
Add an Editor zone

.In this exercise, you'll create a Web Part that displays a text string whose value comes from a public property. Then you'll add an EditorZone control and EditorParts enabling this and other Web Part properties to be edited by users.

Tasks	Detailed Steps

1. Create a programmable Web Part	**a.** Right-click C:\..\Lab9 in the Solution Explorer window and use the "Add New Item" command to add a user control named ProgrammableWebPart.ascx to the project.
	b. Add the following HTML to ProgrammableWebPart.ascx:
	`<table width="360px" cellpadding="4" cellspacing="0" bgcolor="#ececec">`
	`  <tr>`
	`  <td align="left" valign="top">`
	`  This Web Part implements a public property named Text.`
	`  The value of that property is shown below.`
	`  </td>`
	`  </tr>`
	`  <tr>`
	`  <td>`
	`  <span style="font-size: 14pt; font-weight: bold">`
	`  <asp:Label ID="Label1" Runat="server" />`
	`  </span>`
	`  </td>`
	`  </tr>`
	`</table>`
	c. Add the following code to the ProgrammableWebPart class in ProgrammableWebPart.ascx.<cs or vb>:
	C#
	`string _text = "Hello";`
	
	`public string Text`
	`{`
	`  get { return _text; }`
	`  set`
	`  {`
	`  Label1.Text = value;`
	`  _text = value;`
	`  }`
	`}`
	
	`protected void Page_Load (Object sender, EventArgs e)`
	`{`
	`  Label1.Text = _text;`
	`}`
	VB
	`  Dim _text As String = "Hello"`
	`  Public Property Text() As String`
	`  Get`
	`  Return _text`

```
        End Get
        Set(ByVal value As String)
            Label1.Text = value
            _text = value
        End Set
    End Property

Protected Sub Page_Load(ByVal sender As Object, ByVal e As
System.EventArgs) Handles Me.Load
        Label1.Text = _text
    End Sub
```

d. Open Default.aspx in Design view. Display the DeclarativeCatalogPart's "Common DeclarativeCatalogPart Tasks" menu and click "Edit Templates."

e. Drag ProgrammableWebPart.ascx from the Solution Explorer window and drop it into the DeclarativeCatalogPart.

f. Display the DeclarativeCatalogPart's "Common DeclarativeCatalogPart Tasks" menu again and click "End Template Editing."

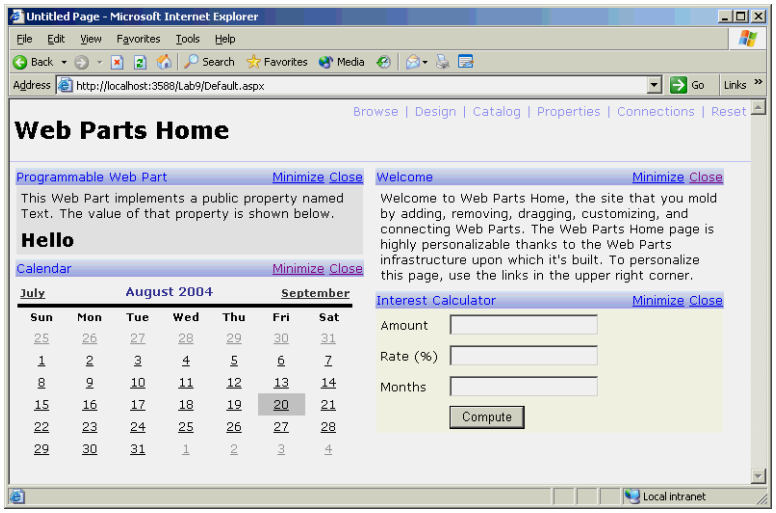
g. Switch to Source view and find the <uc3:ProgrammableWebPart> tag added to the DeclarativeCatalogPart. Manually add a Title="Programmable Web Part" attribute to the tag.

h. Press Ctrl+F5 to launch Default.aspx in your browser.

i. Click "Catalog" to display the Catalog zone.

j. Click "Declarative Catalog (2)" in the Catalog zone. Check the "Programmable Web Part" box and then click Add to add a Programmable Web Part to the page.

k. Click "Browse" to return to Browse display mode. The page should now look like this:



l. Close your browser and return to Visual Studio.

2. Add an EditorZone control

a. Open Default.aspx in the designer and switch to Design view.

b. Drag an EditorZone control from the Toolbox and drop it into the table cell containing the CatalogZone.

c. Display the "Common EditorZone Tasks" menu and check the "View in Edit Mode" box.

d. Click "Click here to add EditorParts to the EditorZone."

e. Drag an AppearanceEditorPart control from the Toolbox and drop it

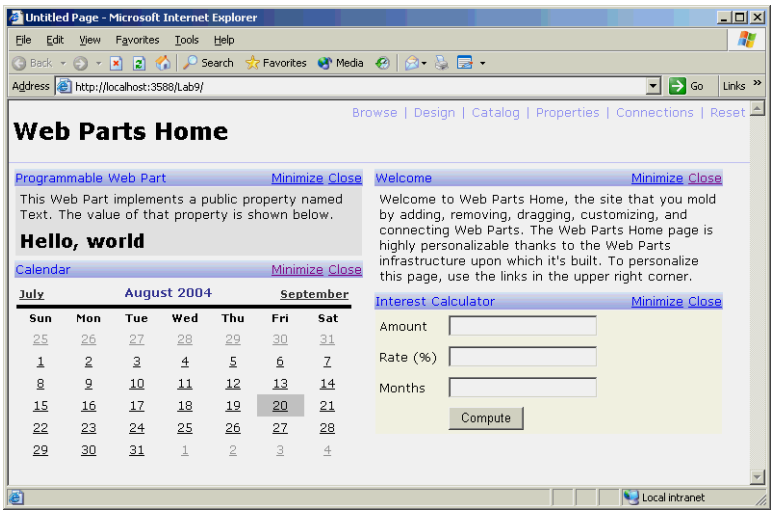
	into the EditorZone.
	f. Drag a BehaviorEditorPart control from the Toolbox and drop it into the EditorZone.
	g. Drag a LayoutEditorPart control from the Toolbox and drop it into the EditorZone.
	h. Drag a PropertyGridEditorPart control from the Toolbox and drop it into the EditorZone.
	i. Switch to Source View and verify that the EditorParts you just added were added to the EditorZone's <ZoneTemplate>. If they weren't, use cut-and-paste to move them to the <ZoneTemplate>.
	j. Double-click the "Properties" LinkButton in the upper-right corner of the page to add a handler for Click events. Add the following statement to the body of the handler:
	C#
	`WebPartManager1.DisplayMode = WebPartManager.EditDisplayMode;`
	VB
	`WebPartManager1.DisplayMode = _ WebPartManager.EditDisplayMode`
	k. Press Ctrl+F5 to launch Default.aspx in your browser.
	l. Click the "Properties" link in the upper-right corner of the page to display the EditorZone and EditorParts.
	m. Click the downward-pointing arrow just to the left of the word "Minimize" in the Programmable Web Part's title bar. Select "Edit" from the menu that appears.
	n. Inspect the CatalogZone that appears on the right side of the page. Does it include a PropertyGridEditorPart? Why not?
	o. Click "Browse" to return to Browse display mode.
	p. Close your browser and return to Visual Studio.

3. Expose the Text property in the PropertyGridEditorPart	**a.** Open ProgrammableWebPart.ascx.cs in the program editor.
	b. Modify the Text property so that it can be edited in the PropertyGridEditorPart. Also modify it to make the value assigned to it persistent. Do this by adding the following attributes above the property declaration
	C#
	`[WebBrowsable]`
	`[Personalizable]`
	VB
	`<WebBrowsable> _`
	`<Personalizable> _`
	c. Press Ctrl+F5 to launch Default.aspx in your browser.
	d. Click the "Properties" link in the upper-right corner of the page to display the EditorZone and EditorParts.
	e. Click the downward-pointing arrow just to the left of the word

"Minimize" in the Programmable Web Part's title bar. Select "Edit" from the menu.

f. Use the PropertyGridEditorPart to change Programmable Web Part's Text property from "Hello" to "Hello, world." Click OK to apply the change and close the EditorZone.

g. Verify that "Hello, world" now appears in the Programmable Web Part, as shown below.



h. Close your browser and return to Visual Studio.

i. Press Ctrl+F5 to launch Default.aspx again.

j. Verify that the property value "stuck"—that is, that Programmable Web Part still shows "Hello, world." Then close your browser and return to Visual Studio.

Exercise 5
Add a Connections zone

In this exercise, you'll add two Web Parts to the page: one that's a connection point provider, and another that's a connection point consumer. Then you'll add a ConnectionsZone control to the page and dynamically connect the two Web Parts so that text typed into one appears in the other.

Tasks	Detailed Steps

k. .Add a ConnectionsZone control	**a.** Open Default.aspx in the designer and switch to Design view.
	b. Drag a ConnectionsZone from the Toolbox and drop it into the table cell containing the CatalogZone and EditorZone.
	c. Double-click the "Connections" LinkButton in the upper-right corner of the page to add a handler for Click events. Add the following statement to the body of the handler:
	C#
	`WebPartManager1.DisplayMode = WebPartManager.ConnectDisplayMode;`
	VB
	`WebPartManager1.DisplayMode = _ WebPartManager.ConnectDisplayMode`
	d. Press Ctrl+F5 to launch Default.aspx in your browser.
	e. Click "Connections" in the upper-right corner of the page. Nothing happens. Why?
	f. Close your browser and return to Visual Studio

g. Define a connections interface
.
	a. Right-click C:\..\Lab9 in the Solution Explorer window and use the "New Folder" command to add a Code folder to the Web site.
	b. Right-click the Code folder in Solution Explorer and use the "Add New Item" command to create a source code file named ITextTransfer.<cs or vb>. Select "Class" as the template type and your selected language.
	c. Change the class definition in ITextTransfer.<cs or vb> into an interface definition as shown here:
	C#
	`public interface ITextTransfer { string GetText (); }`
	VB
	`Public Interface ITextTransfer Function GetText() As String End Interface`
	d. Save your changes and close ITextTransfer.<cs or vb>.

e. Add a provider Web Part	**a.** Right-click C:\..\Lab9 in the Solution Explorer window and use the "Add New Item" command to add a user control named ProviderWebPart.ascx to the project. **b.** Add the following HTML to ProviderWebPart.ascx:

```
<table width="360px" cellpadding="4" cellspacing="0"
bgcolor="#ececec">
  <tr>
    <td align="left" valign="top">
      This Web Part is a connection point provider.
      It provides the text typed into the box below to the
      Consumer Web Part.
    </td>
  </tr>
  <tr>
    <td>
      <asp:TextBox ID="TextBox1" MaxLength="16" Runat="server"
/>
      <asp:Button Text="Transmit" Runat="server" />
    </td>
  </tr>
</table>
```

c. Add an Implements ItextTransfer directive to the class, in C# you do this by adding a “, ItextTransfer at the end of the class statement, in VB you use the Implements keyword after the class statement. Your class statement should look as follows:

C#

```
public partial class ProviderWebPart : System.Web.UI.UserControl ,
ITextTransfer
```

VB

```
Partial Class ProviderWebPart
   Inherits System.Web.UI.UserControl
   Implements ITextTransfer
```

d. Add the following code to the ProviderWebPart_ascx class in ProviderWebPart.ascx.<cs or vb>

C#

```
[ConnectionProvider ("Text", "TextProvider")]
public ITextTransfer GetTextTransferInterface ()
{
   return (ITextTRansfer) this;
}

public string GetText ()
{
   return TextBox1.Text;
}
```

VB

```
<ConnectionProvider("Text", "TextProvider")> _
Public Function GetTextTransferInterface() As ITextTransfer
    Return CType(Me, ITextTransfer)
End Function
Public Function GetText() As String Implements ITextTransfer.GetText
    Return TextBox1.Text
End Function
```

e. Open Default.aspx in Design view. Display the DeclarativeCatalogPart’s “Common DeclarativeCatalogPart Tasks” menu and click “Edit Templates.”

f. Drag ProviderWebPart.ascx from the Solution Explorer window and drop it into the DeclarativeCatalogPart.

g. Display the DeclarativeCatalogPart’s “Common

	DeclarativeCatalogPart Tasks" menu again and click "End Template Editing." **h.** Switch to Source view and find the <uc4:ProviderWebPart> tag added to the DeclarativeCatalogPart. Manually add a Title="Provider Web Part" attribute to the tag.

i. Add a consumer Web Part	**a.** Right-click C:\..\Lab9 in the Solution Explorer window and use the "Add New Item" command to add a user control named ConsumerWebPart.ascx to the project.
	b. Add the following HTML to ConsumerWebPart.ascx:
	`<table width="360px" height="88pt" cellpadding="4" cellspacing="0" bgcolor="#ececec">`
	`<tr>`
	`<td align="left" valign="top">`
	`This Web Part is a connection point consumer.`
	`It retrieves text from the Provider Web Part.`
	`</td>`
	`</tr>`
	`<tr>`
	`<td>`
	`<asp:Label ID="Label1" Font-Size="14pt" Font-Bold="true" Runat="server" />`
	`</td>`
	`</tr>`
	`</table>`
	c. Add the following code to the ConsumerWebPart_ascx class in ConsumerWebPart.ascx.cs:
	C#
	`[ConnectionConsumer ("Text", "TextConsumer")]`
	`public void GetTextTransferInterface (ITextTransfer provider)`
	`{`
	`Label1.Text = provider.GetText ();`
	`}`
	VB
	`<ConnectionConsumer("Text", "TextConsumer")> _`
	`Public Sub _`
	`GetTextTransferInterface(ByVal provider As ITextTransfer)`
	`Label1.Text = provider.GetText()`
	`End Sub`
	d. Open Default.aspx in Design view. Display the DeclarativeCatalogPart's "Common DeclarativeCatalogPart Tasks" menu and click "Edit Templates."
	e. Drag ConsumerWebPart.ascx from the Solution Explorer window and drop it into the DeclarativeCatalogPart.
	f. Display the DeclarativeCatalogPart's "Common DeclarativeCatalogPart Tasks" menu again and click "End Template Editing."
	g. Switch to Source view and find the <uc5: ConsumerWebPart> tag added to the DeclarativeCatalogPart. Manually add a Title="Consumer Web Part" attribute to the tag.

h. Connect the Web Parts	**a.** Press Ctrl+F5 to launch Default.aspx in your browser. **b.** Click the "Catalog" link in the upper-right corner of the page.

c. Use the Catalog zone's DeclarativeCatalogPart to add a Provider Web Part to Zone 1.

d. Use the Catalog zone's DeclarativeCatalogPart to add a Consumer Web Part to Zone 2.

e. Click the "Connections" link in the upper-right corner of the page.

f. Click the downward-pointing arrow in the Consumer Web Part's title bar and select "Connect" from the menu.

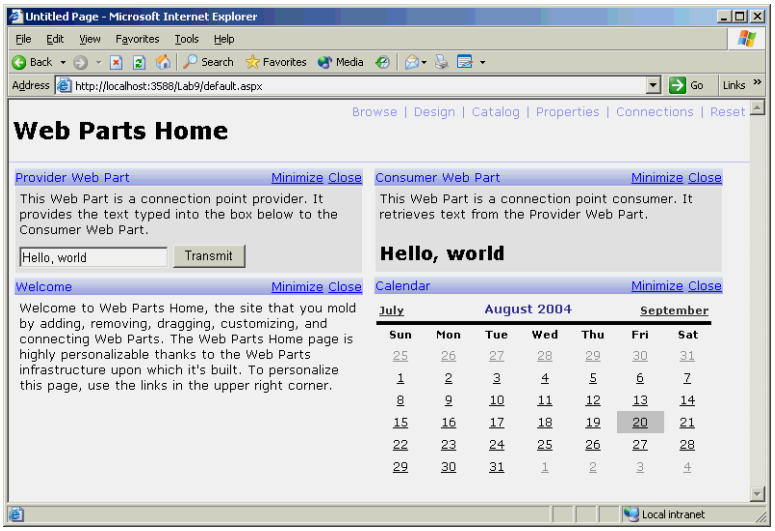
g. In the ConnectionsZone control that appears on the right, click "Create a connection to a provider."

h. Pull down the drop-down list labeled "From" and select "Provider Web Part."

i. Click the Connect button.

j. Click "Browse" in the upper-right corner of the page to return to Browse display mode.

k. Type "Hello, world" into the text box in the Provider Web Part and click the Transmit button. Verify that "Hello, world" appears in the Consumer Web Part, as shown below.



l. Close your browser and return to Visual Studio.

Exercise 6
Add a reset feature

In this exercise, you'll apply a finishing touch to the site by allowing users to reset—that is, undo changes to—Default.aspx. All it takes is a call to the PersonalizationAdministration.ResetUserState method, which deletes the personalization data generated when you add Web Parts to the page, create connections between Web Parts, and so on.

Tasks	Detailed Steps

1. Write a handler for the Reset LinkButton	**a.** Open Default.aspx in the designer and switch to Design view.
	b. Default.aspx already contains a LinkButton labeled "Reset." Double-click that LinkButton to create a handler for its Click events.
	c. Add the following statements to the handler generated by Visual Studio:
	C#
	`PersonalizationAdministration.ResetUserState ("~/Default.aspx");`
	`Response.Redirect (Request.FilePath);`
	VB
	`PersonalizationAdministration.ResetUserState ("~/Default.aspx")`
	`Response.Redirect (Request.FilePath)`
	d. Save your changes and close Default.aspx

2. Test the Reset LinkButton	**a.** Press Ctrl+F5 to launch Default.aspx in your browser. **b.** Click "Reset" in the page's upper-right corner. **c.** Verify that the page returns to its default state with the Welcome Web Part on the left, the Calendar Web Part on the right, and no other Web Parts visible on the page.

Summary

Here's a recap of what you learned in this lab:

- How to create a Web Parts page by adding a WebPartManager
- How to use conventional Web controls (such as Calendars) as Web Parts
- How to use user controls as Web Parts
- How to use DesignDisplayMode to enable interactive layout editing
- How to use CatalogZones and CatalogParts to allow Web Parts to be added to a page
- How to use EditorZones and EditorParts to allow Web Parts to be customized
- How to expose custom Web Part properties in PropertyGridEditorParts
- How to create connectable Web Parts and connect them with ConnectionsZones
- How to delete per-user personalization data

Take a moment to review the site's source code. As you do, here are some questions to ponder:

- *Where did ASP.NET store the data that persists the page layout?*

- *If you wanted to make the text typed into the Provider Web Part persistent, how would you go about it?*

- *If you wanted to display a message box requesting confirmation before calling PersonalizationAdministration.ResetUserState when the Reset button is clicked, how would you do it?*