Exercise: ASP.NET Fundamentals (C#)

Objectives

After completing this lab, you will be able to:

- Create a Web site with Visual Studio
- Create and link multiple pages in a site
- Use ASP.NET server controls to interact with clients

Prerequisites

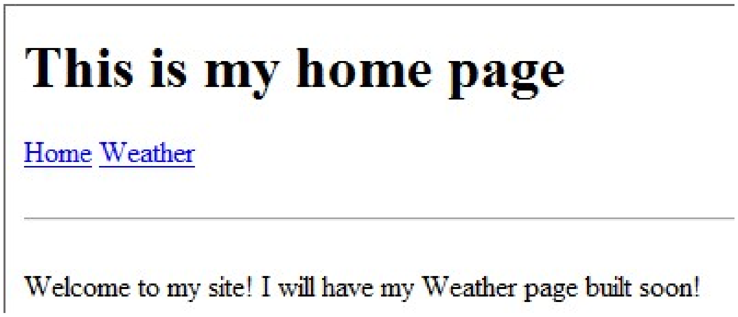
Before working on this lab, you must have:

- Basic understanding of HTML
- Familiarity with Web interfaces

Exercise 1
Creating a new Web site

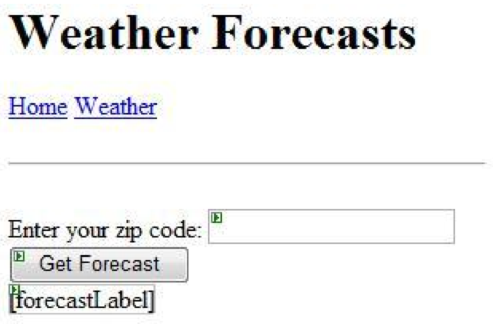
In this exercise, you will create a new Web site with Visual Studio. You will then add a second page (after the single default page that is added for you) for users to look up weather conditions (to be built in Exercise 2). You will also set up hyperlinks on each page for navigation, and lay out some text to distinguish one page from another.

Tasks	Detailed steps
1. Create a new Web Application with Visual Web Studio.	**a.** Launch Microsoft Visual Studio. **b.** Select **File | New ASP.NET Web Application...** . **c.** **Call the app MyWeatherSite**.
2. Add content to your home page – some title text and a pair of hyperlinks for navigation.	**a.** Create a new page, **Default.aspx**, . This is the home page for your site. **b.** Switch to **Design** mode by selecting the **Design** tab at the bottom of the window. **c.** Type in some text, like **This is my home page** (feel free to be more creative than that!) **d.** Select the text, and using the **Formatting** toolbar, select a **Block Format** of **H1**. **e.** Hit **Enter** after your title text, and type in **Home Weather** for navigation links. **f.** Select the word **Home** and **Convert to Hyperlink** Use the **Browse** button in the **Hyperlink** dialog to choose your **Default.aspx** page.
3. Add a new page, Weather.aspx, to your site and link to it from your home page.	**a.** In the **Solution Explorer** window, right-click on the **Web site** and select **Add New Item…** . **b.** Select **Web Form**, and name the file **Weather.aspx** **c.** Switch your new page into Design view, and type some text at the top of the page to indicate that users will be able to look up weather forecasts, something like **Weather Forecasts**. **g.** Select the text, and using the **Formatting** toolbar, select a **Block Format** of **H1**, as you did before on the home page. **h.** Open your **Default.aspx** page again, and turn the word **Weather** into a hyperlink pointing to your new page. Select the word **Weather** and select the **Convert to Hyperlink** button from the **Formatting** menu. **i.** Use the **Browse** button in the **Hyperlink** dialog to choose your **Weather.aspx** page.

	j. Finally, open the **HTML** tab in the toolbox and drag a **Horizontal Rule** under the links you just enabled (just to provide some separation). Feel free to type text under the rule to welcome users to the site. Your completed page should look something like:  This is my home page Home Weather Welcome to my site! I will have my Weather page built soon!
4. Copy the navigation links from **Default.aspx** to your **Weather.aspx** page.	**a.** With your **Default.aspx** page open in design mode, select the two links and the horizontal rule with your mouse and copy them (**Ctrl-C** or **Edit | Copy**). **b.** Open your **Weather.aspx** page in design mode, hit **Enter** after the title text, and paste the contents of the clipboard at that location (**Ctrl-V** or **Edit | Paste**). It should now have the same links and horizontal rule that your **Default.aspx** page does.
5. Try running your new site!	**c.** To test your page in the browser, select Debug/Start without debugging (or Ctrl-F5) to launch the page in the browser **d.** Verify that you can use the links on both of your pages to navigate between them.

Exercise 2
Creating a dynamic page with ASP.NET server controls

In this exercise, you will build your Weather.aspx page to interactively collect a Zipcode from the user and display the current (fabricated) weather forecast using ASP.NET server controls.

Tasks	Detailed steps
1. Use the designer to create a form for the user to input his/her zip code to retrieve the weather forecast.	**a.** Open your Weather.aspx page in Design view. **b.** Use the "Standard" controls in the Toolbox to lay out the controls shown in the image below.  Weather Forecasts Home Weather Enter your zip code: Get Forecast [forecastLabel] **c.** Begin by clicking under the Horizontal Rule you added earlier. **d.** Type in the text **Enter your zip code:** . **e.** From the **Toolbox** on the left, drag a **TextBox** control from the Standard tab adjacent to your text. Set the **ID** attribute of the **TextBox** to **zipcodeTextBox** in the **Properties** window. **f.** Click after the **TextBox** on the designer and hit **Shift-Enter** to insert a line break. **g.** Drag a **Button** from the **Toolbox** on the left under the text you added. Set the **ID** attribute of the **Button** to **getForecastButton** and set its **Text** attribute to **Get Forecast** in the **Properties** window. **h.** Click after the **Button** on the designer and hit **Shift-Enter** again to insert another line break. **i.** Drag a **Label** from the **Toolbox** on the left under the **Button** you just added. Set its **ID** attribute to **forecastLabel** and its **Text** attribute to nothing (delete the **Label1** text that is there) in the **Properties** window.
2. Add a handler for the click event of the Get Forecast button.	**a.** Double click on the **Button** in design mode. This will add a new method to your associated code behind file called **getforecastButton_Click** which is declaratively wired up to the **Click** event of the **Button**.
3. Implement the Get Forecast button handler to generate a random forecast, and set the result as the text in the	**a.** Our goal here is to generate a forecast with a random number (isn't that how weather forecasters do it anyway? ☺). There is a class available called Random which has a method called Next we can call to retrieve a random number from 0 to 1-n where n is the value we pass into the function. To keep it simple, we will just use 3 different forecasts, so we

Label.

will pass in the value 3, as shown here:

```
Random r = new Random();
 int val = r.Next(0,3);
```

b. Next we will declare a string to store the forecast in, and initialize it to an empty string:

```
string forecast = "";
```

c. Now we will write a cascading if/else block to assign a forecast string based on the random value we generated (feel free to change the forecasts to your liking, or increase the random number and number of forecasts):

```
if (val == 0)
    forecast = "warm and sunny.";
else if (val == 1)
    forecast = "cold and windy.";
else if (val == 2)
    forecast = "rain!";
```

d. Finally, we will dynamically set the Text value of the Label we added earlier to the new forecast. To make it feel customized to the user, let’s format it to say “The forecast at 00000 is…” extracting the zipcode from the Text property of the zipcodeTextBox we added.

```
forecastLabel.Text = "The weather forcast " +
            zipcodeTextBox.Text + " is " + forecast;
```

e. When you are done, the entire function should look like:

```
protected void getForecastButton_Click(
                    object sender, EventArgs e)
{
  Random r = new Random();
  int val = r.Next(0,3);
  string forecast = "";

  if (val == 0)
    forecast = "warm and sunny.";
  else if (val == 1)
    forecast = "cold and windy.";
  else if (val == 2)
    forecast = "rain!";

  forecastLabel.Text = "The weather forcast " +
                  zipcodeTextBox.Text + " is " + forecast;
}
```

4. Run and test!

a. Try running your new page with Debug/Start without debugging (Ctrl-F5) and verify that you can type in a zipcode and retrieve a forecast. How accurate is the forecast? ☺