PRACTICAL: 1

AIM: Getting acquainted with Visual Studio environment. (Create new web project, open existing web project, building website, and study of toolbars, menu etc.)

This walkthrough provides you with an introduction to Microsoft Visual Web Developer. It guides you through creating a simple page by using Visual Web Developer, illustrating the basic techniques of creating a new page, adding controls, and writing code.

Tasks illustrated in this walkthrough include:

* Creating a file system Web site.
* Familiarizing yourself with Visual Web Developer.
* Creating a single-file ASP.NET page in Visual Web Developer.
* Adding controls.
* Adding event handlers.
* Running pages with the Web Servers in Visual Web Developer.
* The .NET Framework

## Creating a Web Site and Page

In this part of the walkthrough, you will create a Web site and add a new page to it. You will also add HTML text and run the page in your Web browser.

For this walkthrough, you will create a file system Web site that does not require that you work with Microsoft Internet Information Services (IIS). Instead, you will create and run your page in the local file system.

A file system Web site is one that stores pages and other files in a folder that you choose somewhere on your local computer. Other Web site options include a local IIS Web site, which stores your files in a subfolder of the local IIS root (typically, \Inetpub\Wwwroot\). An FTP site stores files on a remote server that you gain access to across the Internet by using File Transfer Protocol (FTP). A remote site stores files on a remote server that you can access across a local network. For more information, see Walkthrough: Editing Web Sites with FTP in Visual Web Developer. Also, Web site files can be stored in a source control system such as Visual SourceSafe. For more information, see Introducing Source Control.

**To create a file system Web site**

1. Open Visual Web Developer.
2. On the File menu, click New Web Site.

The New Web Site dialog box appears, as shown in the following screen shot.

### New Web Site dialog box

The programming language you choose will be the default for your Web site. However, you can use multiple languages in the same Web application by creating pages and components in different programming languages.

3.

Under Visual Studio installed templates, click ASP.NET Web Site.

When you create a Web site, you specify a template. Each template creates a Web

application that contains different files and folders. In this walkthrough, you are creating

a Web site based on the ASP.NET Web Site template, which creates some folders and a

few default files.

4.

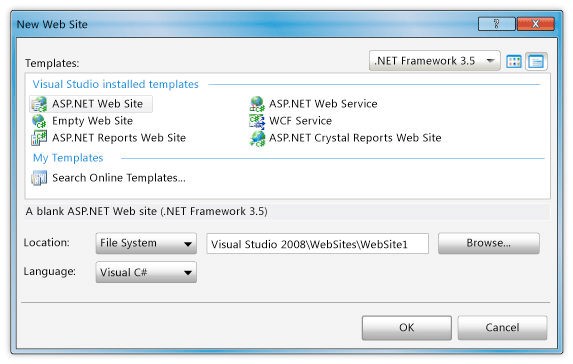
In the Location box, select the File System box, and then enter the name of the folder

where you want to keep the pages of your Web site.

For example, type the folder name C:\BasicWebSite.

5.

In the Language list, select Visual Basic or Visual C#.



6. Click OK.

Visual Web Developer creates the folder and a new page named Default.aspx. When a new page is created, by default Visual Web Developer displays the page in Source view, where you can see the page's HTML elements. The following screen shot shows the Source view of a default Web page.

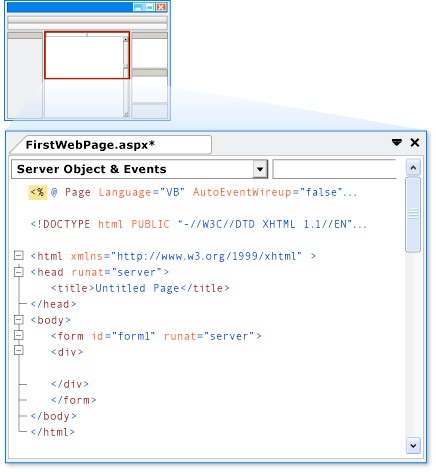
### Source view of default page

**A Tour of Visual Web Developer**

Before you proceed with working on the page, it is useful to familiarize yourself with the Visual

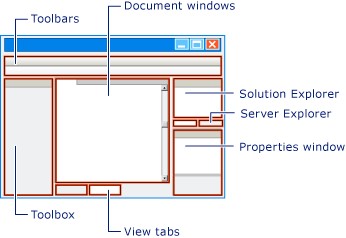
Web Developer development environment. The following illustration shows you the windows

and tools that are available in Visual Web Developer.



This diagram shows default windows and window locations. The View menu allows you to display additional windows, and to rearrange and resize windows to suit your preferences. If changes have already been made to the window arrangement, what you see will not match the diagram.

### Diagram of Visual Web Developer environment



**To familiarize yourself with the Web designer in Visual Web Developer**

• Examine the preceding illustration and match the text to the following list, which describes the most commonly used windows and tools. (Not all windows and tools that you see are listed here, only those marked in the preceding illustration.) o Toolbars. Provide commands for formatting text, finding text, and so on. Some toolbars are available only when you are working in Design view.

* Solution Explorer. Displays the files and folders in your Web site.
* Document window. Displays the documents you are working on in tabbed windows. You can switch between documents by clicking tabs. o Properties window. Allows you to change settings for the page, HTML elements, controls, and other objects. o View tabs. Present you with different views of the same document. Design view is a near-WYSIWYG editing surface. Source view is the HTML editor for the page. Split view displays both the Design view and the Source view for the document. You will work with the Design and Source views later in this walkthrough. If you prefer to open Web pages in Design view, on the Tools menu, click Options, select the HTML Designer node, and change the Start Pages In option. o Toolbox. Provides controls and HTML elements that you can drag onto your page. Toolbox elements are grouped by common function. o Server Explorer/Database Explorer. Displays database connections. If Server Explorer is not visible in Visual Web Developer, on the View menu, click Server Explorer or Database Explorer.

**Creating a New Web Forms Page**

When you create a new Web site, Visual Web Developer adds an ASP.NET page (Web Forms page) named Default.aspx. You can use the Default.aspx page as the home page for your Web site. However, for this walkthrough, you will create and work with a new page.

**To add a page to the Web site**

1. Close the Default.aspx page. To do this, right-click the tab containing the file name and select Close.
2. In Solution Explorer, right-click the Web site (for example, C:\BasicWebSite), and then click Add New Item.
3. Under Visual Studio installed templates, click Web Form. The following screen shot

shows the Add New Item dialog box.

Add New Item dialog box

4.

In the Name box, type FirstWebPage.

5.

In the Language list, choose the programming language you prefer to use (Visual Basic,

C#, or J#). When you created the Web site, you specified a default language. However,

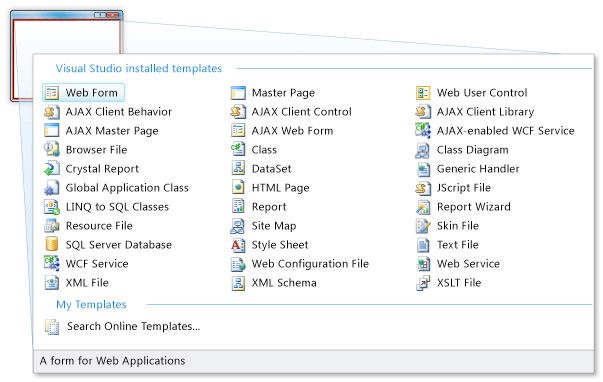
each time you create a new page or component for your Web site, you can change the

language from the default. You can use different programming languages in the same

Web site.

6.

Clear the Place code in separate file check box.



In this walkthrough, you are creating a single-file page with the code and HTML in the same page. The code for ASP.NET pages can be located either in the page or in a separate class file. To learn more about keeping the code in a separate file, see Walkthrough: Creating a Basic Web Page with Code Separation in Visual Web Developer. 7. Click Add.

Visual Web Developer creates the new page and opens it.

**Adding HTML to the Page**

In this part of the walkthrough, you will add some static text to the page.

**To add text to the page**

1.

At the bottom of the document window, click the Design tab to switch to Design view.

Design view displays the page that you are working on in a WYSIWYG-like way. At this

point, you do not have any text or controls on the page, so the page is blank except for a

dashed line that outlines a rectangle. This rectangle represents a div element on the page.

2.

Click inside the rectangle that is outlined by a dashed line.

3.

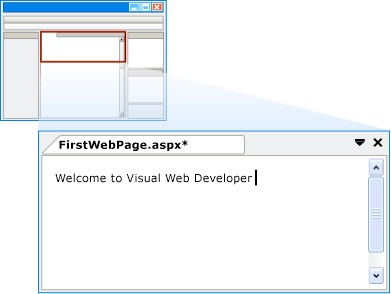
Type Welcome to Visual Web Developer and press Enter twice.

The following screen shot shows the text you typed in Design view.

Welcome text as seen in Design view

4.

Switch to Source view.



You can see the HTML that you created by typing in Design view, as shown in the following screen shot.

### Welcome text as seen in Source view

**Running the Page**

Before you proceed with adding controls to the page, you can try running it. To run a page, you

need a Web server. In a production Web site, you use IIS as your Web server. However, to test a

page, you can use the ASP.NET Development Server, which runs locally and does not require

IIS. For file system Web sites, the default Web server in Visual Web Developer is the ASP.NET

Development Server.

**To run the page**



1. In Solution Explorer, right-click FirstPage.aspx and select Set as Start Page.
2. Press CTRL+F5 to run the page.

Visual Web Developer starts the ASP.NET Development Server. An icon appears on the toolbar to indicate that the Visual Web Developer Web server is running, as shown in the following screen shot.

### Visual Web Developer Web server icon

The page is displayed in the browser. Although the page you created has an extension of .aspx, it currently runs like any HTML page.

To display a page in the browser you can also right-click the page in Solution Explorer and select View in Browser.

If the browser displays a 502 error or an error indicating that the page cannot be displayed, you might need to configure your browser to bypass proxy servers for local requests. For details, see How to: Bypass a Proxy Server for Local Web Requests.

3. Close the browser.

## Adding and Programming Controls

In this part of the walkthrough, you will add a Button, a TextBox, and a Label control to the page and write code to handle the Click event for the Button control.

You will now add server controls to the page. Server controls, which include buttons, labels, text boxes, and other familiar controls, provide typical form-processing capabilities for your ASP.NET Web pages. However, you can program the controls with code that runs on the server, not the client.

**To add controls to the page**

1. Click the Design tab to switch to Design view.
2. Put the insertion point at the end of the Welcome to Visual Web Developer text and press ENTER five or more times to make some room in the div element box.
3. In the Toolbox, expand the Standard group.
4. Drag a TextBox control onto the page and position it in the middle of the div element box that has Welcome to Visual Web Developer in the first line.
5. Drag a Button control onto the page and position it to the right of the TextBox control.
6. Drag a Label control onto the page and position it on a separate line below the Button control.
7. Put the insertion point above the TextBox control, and then type Enter your name:.

This static HTML text is the caption for the TextBox control. You can mix static HTML and server controls on the same page. The following screen shot shows how the three controls appear in Design view.

### Controls in Design View

**Setting Control Properties**

Visual Web Developer offers you various ways to set the properties of controls on the page. In

this part of the walkthrough, you will set properties in both Design view and Source view.

**To set control properties**

1.

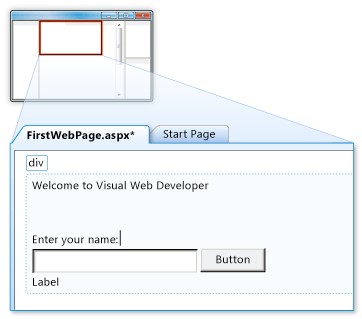
Select the Button control, and then in the Properties window, set Text to Display Name,

as shown in the following screen shot.

Changed Button control text

2.

Switch to Source view.



Source view displays the HTML for the page, including the elements that Visual Web Developer has created for the server controls. Controls are declared using HTML-like syntax, except that the tags use the prefix asp: and include the attribute runat="server".

Control properties are declared as attributes. For example, when you set the Text property for the Button control, in step 1, you were actually setting the Text attribute in the control's markup.

Note that all the controls are inside a <form> element, which also has the attribute runat="server". The runat="server" attribute and the asp: prefix for control tags mark the controls so that they are processed by ASP.NET on the server when the page runs. Code outside of <form runat="server"> and <script runat="server"> elements is sent whose opening tag contains the runat="server" attribute.

unchanged to the browser, which is why the ASP.NET code must be inside an element

3.

Put the insertion point after asp:Label in the <asp:Label> tag, and then press

SPACEBAR.

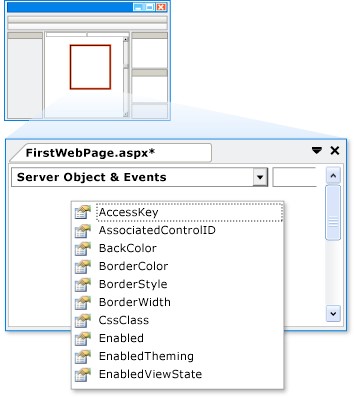
A drop-down list appears that displays the list of properties you can set for a Label

control. This feature, referred to as IntelliSense, helps you in Source view with the syntax

of server controls, HTML elements, and other items on the page. The following screen

shot shows the IntelliSense drop-down list for the Label control.

IntelliSense for Label control



1. Select ForeColor and then type an equal sign and a quotation mark (="). IntelliSense displays a list of colors.

**Note:**

You can display an IntelliSense drop-down list at any time by pressing CTRL+J.

1. Select a color for the Label control's text. Make sure you select a color that is dark enough to read against a white background.

The ForeColor attribute is completed with the color that you have selected, including the closing quotation mark.

**Programming the Button Control**

For this walkthrough, you will write code that reads the name that the user enters into the text box and then displays the name in the Label control.

**To add a default button event handler**

1. Switch to Design view.
2. Double-click the Button control.

Visual Web Developer switches to Source view and creates a skeleton event handler for the Button control's default event, the Click event.

**Note:**

Double-clicking a control in Design view is just one of several ways you can create event handlers.

1. Inside the handler, type the following:

Label1.

When you type the period after Label, Visual Web Developer displays a list of available members for the Label control, as shown in the following screen shot.

### Available Label control members

4.

Finish the Click event handler for the button so that it reads as shown in the following

code example.

VB

Protected

Sub

Button1\_Click(

ByVal

sender

As

Object

,

ByVal

e

As

System.EventArgs)

Label1.Text = Textbox1.Text &

", welcome to Visual Web Developer!"

End

Sub

C#

protected

void

Button1\_Click(

object

sender, System.EventArgs e)

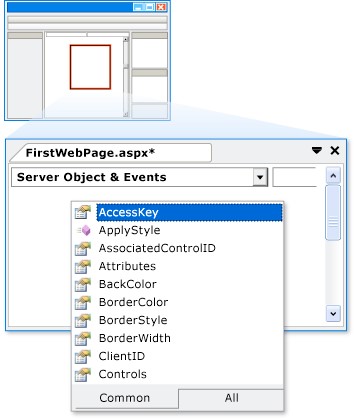
{

Label1.Text = TextBox1.Text +

", welcome to Visual Web Developer!"

;

}



5. Scroll down to the <asp:Button> element. Note that the <asp:Button> element now has the attribute OnClick="Button1\_Click". This attribute binds the button's Click event to the handler method you coded in step 4.

Event handler methods can have any name; the name you see is the default name created by Visual Web Developer. The important point is that the name used for the OnClick attribute must match the name of a method in the page.

If you are using Visual Basic with code separation, Visual Web Developer does not add an explicit OnClick attribute. Instead, the event is bound to the handler method using a Handles keyword on the handler declaration itself.

**Running the Page**

You can now test the server controls on the page.

**To run the page**

1. Press CTRL+F5 to run the page in the browser.

The page again runs using the ASP.NET Development Server.

1. Enter a name into the text box and click the button.

The name you entered is displayed in the Label control. Note that when you click the button, the page is posted to the Web server. ASP.NET then recreates the page, runs your code (in this case, the Button control's Click event handler runs), and then sends the new page to the browser. If you watch the status bar in the browser, you can see that the page is making a round trip to the Web server each time you click the button.

1. In the browser, view the source of the page you are running.

In the page source code, you see only ordinary HTML; you do not see the <asp:> elements that you were working with in Source view. When the page runs, ASP.NET processes the server controls and renders HTML elements to the page that perform the functions that represent the control. For example, the <asp:Button> control is rendered as the HTML element <input type="submit">.

1. Close the browser.

## Working with Additional Controls

In this part of the walkthrough, you will work with the Calendar control, which displays dates a month at a time. The Calendar control is a more complex control than the button, text box, and label you have been working with and illustrates some further capabilities of server controls.

In this section, you will add a Calendar control to the page and format it.

**To add a Calendar control**

1. In Visual Web Developer, switch to Design view.
2. From the Standard section of the Toolbox, drag a Calendar control onto the page and position it below the div element that contains the other controls:

The calendar's smart tag panel is displayed. The panel displays commands that make it

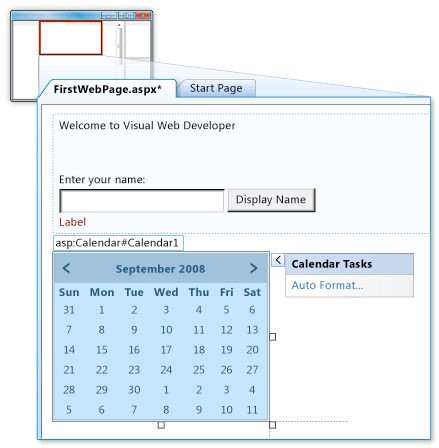
easy for you to perform the most common tasks for the selected control. The following

screen shot shows the Calendar control as rendered in Design view.

Calendar control in Design view

3.

In the smart tag panel, choose Auto Format.



The Auto Format dialog box is displayed, which allows you to select a formatting scheme for the calendar. The following screen shot shows the Auto Format dialog box for the Calendar control.

### Auto Format dialog box for the Calendar control

4.

From the Select a scheme list, select Simple and then click OK.

5.

Switch to Source view.

You can see the <asp:Calendar> element. This element is much longer than the elements

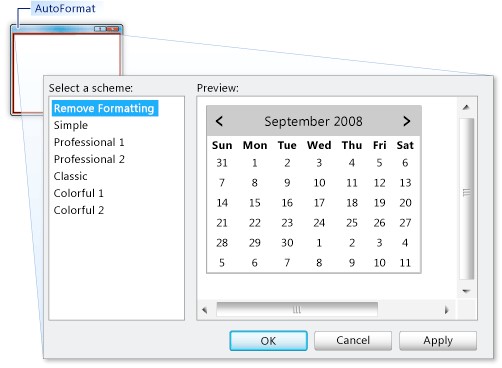
for the simple controls you created earlier. It also includes subelements, such as

<

WeekEndDayStyle>, which represent various formatting settings. The following screen

shot shows the Calendar control in Source view.

Calendar control in Source view



**Programming the Calendar Control**

In this section, you will program the Calendar control to display the currently selected date.

**To program the Calendar control**

1.

In Design view, double-click the Calendar control.

A new event handler is created in Source view.

2.

Finish the SelectionChanged event handler with the following highlighted code.

VB

Protected

Sub

Calendar1\_SelectionChanged(

ByVal

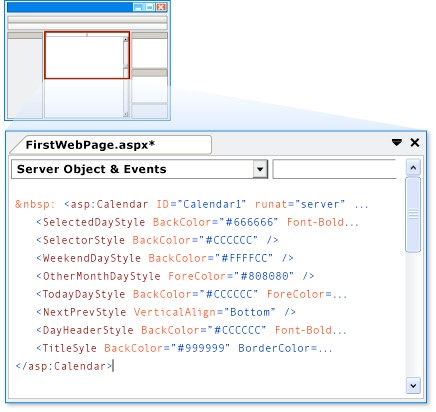
sender

As

Object

,

ByVal



e As System.EventArgs)

Label1.Text = Calendar1.SelectedDate.ToLongDateString()

End Sub

### C#

protected void Calendar1\_SelectionChanged(object sender,

System.EventArgs e)

{

Label1.Text = Calendar1.SelectedDate.ToLongDateString(); }

**Running the Page**

You can now test the calendar.

**To run the page**

1.

Press CTRL+F5 to run the page in the browser.

2.

Click a date in the calendar.

The date you clicked is displayed in the Label control.

3.

In the browser, view the source code for the page.

Note that the Calendar control has been rendered to the page as a table, with each day as

a <td> element containing an <a> element.

4.

Close the browser.

|  |  |  |  |
| --- | --- | --- | --- |
| Start Date | End/Sign Date | Faculty Sign | Remarks/Grade |
|  |  |  |  |