ASP.NET Using VB.NET

Robert J. Oberg
Arun Ganesh
Srini Manickam

Student Guide
Revision 1.0

Object Innovations Course 426
ASP.NET Using VB.NET
Rev. 1.0

Student Guide

Information in this document is subject to change without notice. Companies, names and data used in examples herein are fictitious unless otherwise noted. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Object Innovations.

Product and company names mentioned herein are the trademarks or registered trademarks of their respective owners.

Copyright ©2002 Object Innovations, Inc. All rights reserved.

Object Innovations, Inc.
4515 Emory Lane
Charlotte, NC 28211
704-362-5413
www.ObjectInnovations.com

Printed in the United States of America.
# Table of Contents (Overview)

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction to ASP.NET</td>
</tr>
<tr>
<td>2</td>
<td>Web Forms Architecture</td>
</tr>
<tr>
<td>3</td>
<td>ASP.NET and HTTP</td>
</tr>
<tr>
<td>4</td>
<td>Web Applications Using Visual Studio</td>
</tr>
<tr>
<td>5</td>
<td>State Management and Web Applications</td>
</tr>
<tr>
<td>6</td>
<td>Server Controls and Data Binding</td>
</tr>
<tr>
<td>7</td>
<td>ASP.NET Configuration and Security</td>
</tr>
<tr>
<td>8</td>
<td>Caching in ASP.NET</td>
</tr>
<tr>
<td>9</td>
<td>Introduction to Web Services</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Appendix A</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Learning Resources</td>
</tr>
</tbody>
</table>
# Table of Contents (Detailed)

## Chapter 1  Introduction to ASP.NET
- What Is ASP.NET? 3
- Web Application Fundamentals 4
- Setting up the Web Examples 5
- Creating a Virtual Directory 9
- Home Page for ASP.NET Examples 11
- An Echo Program 12
- ASP.NET Features 16
- Compiled Code 18
- Server Controls 19
- Browser Independence 20
- Separation of Code and Content 21
- State Management 22
- Lab 1 23
- Summary 24

## Chapter 2  Web Forms Architecture
- Web Forms Architecture 29
- Code-Behind Version of Echo Example 30
- HelloCodebehind.aspx 31
- HelloCodebehind.aspx.vb 32
- Page Class 33
- Inheriting from Page Class 34
- Web Forms Page Life Cycle 36
- View State 39
- Web Forms Event Model 40
- Page Processing 42
- Page Events 43
- Page Properties 44
- Sample Program 45
- Page Directive 49
- Tracing 51
- Lab 2 53
- Summary 54

## Chapter 3  ASP.NET and HTTP
- Classical Web Programming 59
- Active Server Pages Object Model 60
- Request and Response Objects 61
- Request and Response Example 62
- HTML Code 63
<table>
<thead>
<tr>
<th>Chapter 5</th>
<th>State Management and Web Applications</th>
<th>117</th>
</tr>
</thead>
<tbody>
<tr>
<td>Session and Application State</td>
<td>Example Program</td>
<td>120</td>
</tr>
<tr>
<td>Session Object</td>
<td>Global.asax.vb</td>
<td>121</td>
</tr>
<tr>
<td>Global.asax.vb</td>
<td></td>
<td>122</td>
</tr>
<tr>
<td></td>
<td></td>
<td>123</td>
</tr>
</tbody>
</table>
Chapter 8  Caching In ASP.NET ................................................................. 221

Introduction ............................................................................................ 223
What is Caching? ...................................................................................... 224
Need for Caching (Why Cache?) ............................................................. 225
Data to be Cached – Time Frame ............................................................ 226
ASP vs. ASP.NET Response Model ......................................................... 227
Caching in ASP.NET ................................................................................ 228
Three Types of Caching in ASP.NET ...................................................... 229
Output Caching ....................................................................................... 230
@ OutputCache Directive ......................................................................... 231
Simple Output Caching Example ............................................................ 232
@ OutputCache – Attributes in Detail ...................................................... 235
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual Studio Web Service Demo</td>
<td>290</td>
</tr>
<tr>
<td>Hello World Web Service</td>
<td>291</td>
</tr>
<tr>
<td>Test Client</td>
<td>292</td>
</tr>
<tr>
<td>Providing a Namespace</td>
<td>293</td>
</tr>
<tr>
<td>Client Program</td>
<td>294</td>
</tr>
<tr>
<td>Add Web Reference Dialog</td>
<td>295</td>
</tr>
<tr>
<td>Hello Service Operations</td>
<td>296</td>
</tr>
<tr>
<td>Web References Subdirectory</td>
<td>297</td>
</tr>
<tr>
<td>HelloClient Code</td>
<td>298</td>
</tr>
<tr>
<td>Lab 9A</td>
<td>299</td>
</tr>
<tr>
<td>Lab 9B</td>
<td>300</td>
</tr>
<tr>
<td>Summary</td>
<td>301</td>
</tr>
<tr>
<td>Appendix A   Learning Resources</td>
<td>309</td>
</tr>
</tbody>
</table>
Chapter 4

Web Applications Using Visual Studio
Web Applications Using Visual Studio

Objectives

*After completing this unit you will be able to:*

- Create Web applications using Visual Studio and ASP.NET.
- Use the Forms Designer to visually lay out Web forms.
- Deploy Web applications.
- Configure Web projects as Web applications under IIS.
- Use the `Global.asax.vb` file to work with application state in Web applications.
- Use data binding in your Web applications.
Using Visual Studio

• We have examined the fundamentals of ASP.NET and have created some simple Web pages.

• To carry the story further it will be very helpful to start using Visual Studio .NET.
  – Everything we do could also be accomplished using only the .NET Framework SDK, but our work will be much easier using the facilities of Visual Studio.
  – A special kind of project, an “ASP.NET Web Application,” creates the boilerplate code.
  – The Forms Designer makes it very easy to create Web forms by dragging controls from a palette.
  – We can add event handlers for controls in a manner very similar to the way event handlers are added in Windows Forms.
  – In fact, the whole Web application development process takes on many of the rapid application development (RAD) characteristics typical of Visual Basic.

• In this chapter we will introduce the Web application development features of Visual Studio with a simple demo.

• In a lab you will create the first step of our Acme Travel Web site.
Configuring Web Server Connection

- Before getting started you may wish to check, and possibly change, your Visual Studio Web Server Connection setting.

- The two options are File share and FrontPage.
  - If you are doing all your development on a local computer, you might find File share to be faster and more convenient.

- To access this setting, select the Visual Studio menu Tools | Options. . .
  - Choose Web Settings underneath Projects.

- You can then set the Preferred Access Method by using a radio button, as illustrated in the figure.
Demonstration: Creating a Web App

1. Verify that `c:\OIC\AspVb` is a virtual directory with alias `AspVb` by opening the URL `http://localhost/AspVb` and checking that you find the home page of example programs.


   ![New Project Dialog]

   You will now see the design mode for the Web form `WebForm1.aspx` with a message indicating that the page you are working on is in grid layout mode. (The alternative is flow layout mode, and you can specify which by using the `pageLayout` property of the form.)
The Toolbox

3. Bring up the Toolbox from the View menu (if not already showing).

4. Drag a Label control, a TextBox, a Button, and another Label onto the form, as shown.

5. Change the Text property of the labels to “Your name” and blank. Change the Text property of the Button to “Echo”. Move and resize the controls as needed so that the form looks good.

6. Change the (ID) of the TextBox to `txtName`, of the Button to `cmdButton`, and of the second label to `lblName`.

7. Double-click the Echo Button. This will add an event handler, and will bring up the code-behind file for the form.
Creating a Web App (Cont’d)

8. Add code to the event handler to create a greeting message from the name passed in the TextBox and store this greeting in the second Label.

Private Sub cmdEcho_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdEcho.Click
    lblName.Text = "Hello, " & txtName.Text
End Sub
9. Build and run the application. Internet Explorer will be brought up to display your new Web page. The behavior should be the same as the hand-crafted Hello.aspx page we illustrated in Chapter 1.
Deploying a Web Application

- Developing a Web application using Visual Studio is quite straightforward.
  - You can do all your work within Visual Studio, including testing your application.
  - When you start a Web application within Visual Studio, Internet Explorer will be brought up automatically.
  - And it is easy to debug, as you will see in the lab.

- Deploying a Web application created using Visual Studio is also easy, but you need to be aware of a few things.

  1. The Project | Copy Project... menu can be used to deploy a Web project from Visual Studio.

  2. Visual Studio precompiles Web pages, storing the executable in the bin folder.

  3. The Src attribute in the Page directive is not used.

  4. Instead, the Inherits attribute is used to specify the Page class.

  5. The directory containing the Web pages must be marked as a Web application. This marking is performed automatically by Visual Studio when you deploy the application.

  6. If you copy the files to another directory, possibly on another system, you must perform the marking as an application yourself, which you can do using Internet Services Manager.
Using Project | Copy Project...

• To illustrate using Visual Studio to deploy a Web project, let’s deploy the EchoVb application we have created.

• We will deploy it to a new directory EchoVb2 in the Deploy subdirectory of c:\OIC\AspVb.

1. Using Windows Explorer, create a new directory EchoVb2 underneath Deploy.

2. Bring up the Copy Project dialog from the menu Project | Copy Project…

3. Enter the following information (see the screen capture on the next page).

   − http://localhost/AspVb/Deploy/EchoVb2/ for Destination project folder

   − File share for Web access method

   − C:\OIC\AspVb\Deploy\EchoVb2 for Path

   − “Only files needed to run this application” for Copy
4. You can test the deployment by using Internet Explorer. Enter the following URL:

http://localhost/AspVb/Deploy/EchoVb2/Webform1.aspx
Precompiled Web Page

- Examining the files in the folder Deploy\EchoVb2, you will see no code-behind file WebForm1.aspx.cs.
  - Instead, in the bin folder you will see the DLL EchoVb.dll.

- Examining the file WebForm1.aspx, we see there is no Src attribute.
  - Instead, the Inherits attribute specifies the Page class WebForm1, which is implemented in the assembly EchoVb.dll.

```xml
<%@ Page Language="vb" AutoEventWireup="false"
Codebehind="WebForm1.aspx.vb"
Inherits="EchoVb.WebForm1" %>
```
Configuring a Virtual Directory as an Application

- The identical files you copied to `Deploy\EchoVb2` are also provided in the directory `EchoRun`.
  - Try the URL
    
    http://localhost/AspVb/Chap04/EchoRun/WebForm1.aspx

    in Internet Explorer. You will obtain a configuration error, as illustrated in the screen capture on the next page.
Configuration Error

Server Error in '/AspVb' Application.

Configuration Error

Description: An error occurred during the processing of a configuration file required to service this request. Please review the specific error details below and modify your configuration file appropriately.

Parser Error Message: It is an error to use a section registered as allowDefinition='MachineToApplication' beyond application level. This error can be caused by a virtual directory not being configured as an application in IIS.

Source Error:

Line 23: "Forms", "Passport" and "None"
Line 24: -->
Line 25: <authentication mode="Windows" />
Line 26:
Line 27:

Source File: C:\OLC\AspVb\Chap04\EchoRun\web.config   Line: 25

- The key sentence in the error message is: “This error can be caused by a virtual directory not being configured as an application in IIS.”
Configuring as an Application

- To fix, use Internet Services Manager.
  - You can find Internet Services Manager under Administrative Tools in the Control Panel.

1. Find the folder **EchoRun** under **Chap04** in the virtual directory **AspVb**.

2. Right-click and choose properties. Click Create. See the figure.

3. You will then see “EchoRun” suggested as the application name. Accept all the suggested settings and click OK.

4. Now try again in Internet Explorer. You should be successful in bringing up the application.
Copying the Source Code

- A Web application project is not quite as simple to move around as other Visual Studio projects.

- To gain some insight into working with Web application projects, let’s copy the source code of our *EchoVb* application “by hand.”
  
  You could of course also use Project | Copy Project.

1. Close Visual Studio, and copy the whole directory *EchoVb* from *Demos* to the *Deploy* folder.

2. Examine the files in the *EchoVb* folder. Notice that there are files with extension *\.vbproj* and *\.vbproj.webinfo*, but there is no solution file with extension *\.sln*. Where do you suppose this file is?


4. Back in *Deploy\EchoVb*, double-click on *EchoVb.vbproj*. You get an error message.

![Microsoft Development Environment](image)
Copying the Source Code

5. To fix this error, edit the file `EchoVb.vbproj.webinfo` so that the URL points to the right file with respect to the virtual directory.

```xml
<VisualStudioUNCWeb>
    <Web URLPath = "http://localhost/AspVb/Deploy/EchoVb/EchoVb.vbproj" />
</VisualStudioUNCWeb>
```

6. Now double-click on `EchoVb.vbproj` again. This time the project should open successfully.

7. Build the solution. You will be prompted to save the file `EchoVb.sln` (which this time will be saved in the same folder as the other files of the project).

8. Try to run the project. You will get an error message.

![Error Message]

```
Running the project requires setting an initial Web page. To set this page, right-click the desired page in the Solution Explorer and select "Set As Start Page".
```

9. Follow the directions in the message box and make `WebForm1.aspx` into the start page.

10. Build and run again. This time you will a configuration error like the one shown on page 97. Fix it by using IIS to configure the folder as an application, as described on page 97. Now the application should run as expected.

- This same application with all source files is also available in the folder `Chap04\EchoVb`. 
ASP.NET Applications

• An ASP.NET application consists of all the Web pages and code files that can be invoked from a virtual directory and its subdirectories on a Web server.

• Besides .aspx files and code-behind files such as those we have already examined, an application can also have a global.asax file and a configuration file config.web.
  – In this chapter we examine the features of ASP.NET applications.
  – We will look at configuration files in Chapter 7.
Sessions

• To appreciate the Web application support provided by ASP.NET, we need to understand the concept of a Web session.

• HTTP is a stateless protocol. This means that there is no direct way for a Web browser to know whether a sequence of requests is from the same client or from different clients.
  
  − A Web server such as IIS can provide a mechanism to classify requests coming from a single client into a logical session.
  
  − ASP.NET makes it very easy to work with sessions, as we will see in more detail in the next chapter.
Global.asax

- An ASP.NET application can optionally contain a file Global.asax, which contains code for responding to application-level events raised by ASP.NET.

- This file resides in the root directory of the application.
  - Visual Studio will automatically create a Global.asax file for you when you create an ASP.NET Web Application project.
  - If you do not have a Global.asax file in your application, ASP.NET will assume you have not defined any handlers for application-level events.

- Global.asax is compiled into a dynamically generated .NET Framework class derived from HttpApplication.
Web Application Life Cycle

• The typical life cycle of a Web application would consist of these events:

  − **Application_Start** is raised only once during an application’s lifetime, on the first instance of **HttpApplication**. An application starts the first time it is run by IIS for the first user. In your event handler you can initialize a state that is shared by the entire application.

  − **Session_Start** is raised at the start of each session. Here you can initialize session variables.

  − **Application_BeginRequest** is raised at the start of an individual request. Normally you can do your request processing in the Page class.

  − **Application_EndRequest** is raised at the end of a request.

  − **Session_End** is raised at the end of each session. Normally you do not need to do cleanup of data initialized in **Session_Start**, because garbage collection will take care of normal cleanup for you. However, if you have opened an expensive resource, such as a database connection, you may wish to call the **Dispose** method here.

• **Application_End** is raised at the very end of an application’s lifetime, when the last instance of **HttpApplication** is torn down.
Application State Example

- To illustrate how to maintain state information across page requests, consider the ListDemo example.
  - You can add names, which will be remembered and shown in a listbox.
Global.asax.vb

• We store the names in the ArrayList \textit{list}, which is a shared data member of the \textit{Global} class.

• The object \textit{list} is instantiated in the \textit{Application\_Start} method.

' Global.asax.vb

Imports System.Web
Imports System.Web.SessionState

Public Class Global
    Inherits System.Web.HttpApplication

    Public Shared list As ArrayList

    #Region " Component Designer Generated Code "
    ...
    Sub Application\_Start(ByVal sender As Object, ByVal e As EventArgs)
        ' Fires when the application is started
        list = New ArrayList()
    End Sub
    
    '...

    #End Region
Data Binding

- Next we will populate a listbox with the string data stored in list.

- We make use of the data binding capability of the ListBox control.

- You might think data binding is only used with a database.
  - However, in .NET data binding is much more general, and can be applied to other data sources besides databases.

- The .NET Framework provides a number of data binding options.

- A very simple option is binding to an ArrayList.
  - This option works perfectly in our example, because we are storing the strings in the ArrayList list.
Data Binding Code Example

- Our code is in the handler of the Add button.

Private Sub cmdAdd_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles cmdAdd.Click
    Global.list.Add(txtName.Text)
    lstNames.DataSource = Global.list
    DataBind()
End Sub

- The DataSource property of ListBox is used to get or set the data source that populates the items of the listbox.

- The call to DataBind() binds all the server controls on the form to their data source, which results in the controls being populated with data from the data source.
  - The DataBind method can also be invoked on the server controls individually.
  - DataBind is a method of the Control class, and is inherited by the Page class and by specific server control classes.
Lab 4

Hotel Information Web Page

In this lab you will create a simple Web page that displays information about hotels. Dropdown listboxes are provided to show cities and hotels. Selecting a city from the first dropdown will cause the hotels in that city to be shown in the second dropdown. We obtain the hotel information from the Hotel.dll component, and we use data binding to populate the listboxes. As a source for the Hotel.dll component used in the lab, we provide a Windows Forms application, AcmeGui. The Hotel.dll component you need is in the folder AcmeGui.

Detailed instructions are contained in the Lab 4 write-up at the end of the chapter.

Suggested time: 60 minutes
Summary

- Visual Studio makes it easy to develop ASP.NET Web applications.

- With the Forms Designer you can visually lay out Web forms, and with a click you can add event handlers.

- You can deploy Web applications using Visual Studio, or you can just copy files and edit the .vbproj.webinfo file.

- You must configure Web projects as Web applications under IIS in order for them to run properly.

- You can use the Global.asax.vb file to work with application state in Web applications.

- Data binding makes it easy to map list controls to data sources of various sorts.
Lab 4

Hotel Information Web Page

Introduction

In this lab you will create a simple Web page that displays information about hotels. Dropdown listboxes are provided to show cities and hotels. Selecting a city from the first dropdown will cause the hotels in that city to be shown in the second dropdown. We obtain the hotel information from the `Hotel.dll` component, and we use data binding to populate the listboxes. As a source for the `Hotel.dll` component used in the lab, we provide a Windows Forms application, `AcmeGui`. The `Hotel.dll` component you need is in the folder `AcmeGui`.

Suggested Time: 60 minutes

Root Directory: OIC\AspVb

Directories: Labs\Lab4 (do your work here)
              Chap04\AcmeGui (source for Hotel.dll)
              Chap04\AcmeWeb\Step0 (answer)

Creating an ASP.NET Web Application

1. Use the Windows File Manager to create a folder `AcmeWeb` in Labs\Lab4.

2. In Visual Studio select the menu File | New | Project.....

3. In the New Project dialog box choose “Visual Basic Projects” as the Project Type and “ASP.NET Web Application” as the Template.

4. Enter http://localhost/AspVb/Labs/Lab4/AcmeWeb as the Location of your project.

5. Click OK. The project files will then be created in \OIC\AspVb\Labs\Lab4\AcmeWeb. The Visual Studio solution AcmeWeb.sln will then be created under MyDocuments\Visual Studio Projects\AcmeWeb.

Using the Form Designer

1. Bring up the Toolbox from the View menu, if not already showing. Make sure the Web Forms tab is selected.

2. Drag two Label controls and two DropDownList controls onto the form.

3. Change the Text property of the Labels to “City” and “Hotel”. Resize the DropDownList controls to look as shown in the screen capture on the next page.
4. Change the (ID) of the DropDownList controls to `listCities` and `listHotels`.

**Initializing the HotelBroker**

1. Copy `Hotel.dll` from `Chap04\AcmeGui` to `Labs\Lab4\AcmeWeb\bin`.

2. In your `AcmeWeb`, project add a reference to `Hotel.dll`.

3. As shown in the following code fragment, in `Global.asax.vb`, add the following line near the top of the file. (Use the View Code button to show the code.)

   ```vbnet
   Imports OI.NetVb.Acme
   ```


5. Add code to `Application_Start` to instantiate `HotelBroker`.

   ```vbnet
   Imports System.Web
   Imports System.Web.SessionState
   Imports OI.NetVb.Acme
   
   Public Class Global
       Inherits System.Web.HttpApplication

       #Region " Component Designer Generated Code "

       
       Public Shared broker As HotelBroker

       Sub Application_Start(ByVal sender As Object, ByVal e As EventArgs)
           ' Fires when the application is started
   ```
broker = New HotelBroker()
End Sub
...


Imports OI.NetVb.Acme

Public Class WebForm1
    Inherits System.Web.UI.Page
    ...
    Private Shared hotelBroker As HotelBroker
    ...

Data Binding

Next we will populate the first DropDownList with the city data, which can be obtained by the GetCities method of HotelBroker. We make use of the data binding capability of the DropDownList control. You might think data binding is only used with a database. However, in .NET data binding is much more general, and can be applied to other data sources besides databases. Binding a control to a database is very useful for two-tier, client/server applications. However, we are implementing a three-tier application, in which the presentation logic, whether implemented using Windows Forms or Web Forms, talks to a business logic component and not directly to the database. So we will bind the control to an ArrayList.

The .NET Framework provides a number of data binding options, which can facilitate binding to data obtained through a middle-tier component. A very simple option is binding to an ArrayList. This option works perfectly in our example, because we need to populate the DropDownList of cities with strings, and the GetCities method returns an array list of strings.

The bottom line is that all we need to do to populate the listCities DropDownList is to add the following code to the Page_Load method of the WebForm1 class.

Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    'Put user code to initialize the page here
    broker = Global.broker
    Dim cities As ArrayList = broker.GetCities()
    listCities.DataSource = cities
    DataBind()
End Sub

The call to DataBind() binds all the server controls on the form to their data source, which results in the controls being populated with data from the data source. The DataBind method can also be invoked on the server controls individually. DataBind is a
method of the **Control** class, and is inherited by the **Page** class and by specific server control classes.

You can now build and run the project. Running a Web application under Visual Studio will bring up Internet Explorer to access the application over HTTP. When you drop down the list of cities, you will indeed see the cities returned by the HotelBroker component.

---

**Initializing the Hotels**

We can populate the second DropDownList with hotel data using a similar procedure. It is a little bit more involved, because `GetHotels` returns an array list of `HotelListItem` structures rather than strings. We want to populate the `listHotels` DropDownList with the names of the hotels. The helper method `BindHotels` loops through the array list of hotels and creates an array list of hotel names, which is bound to `listHotels`. Here is the complete code, which adds the logic for initializing the hotels for the first city (which has index 0).

```vbnet
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
    'Put user code to initialize the page here
    broker = Global.broker
    Dim cities As ArrayList = broker.GetCities()
    listCities.DataSource = cities
    Dim hotels As ArrayList = _
        broker.GetHotels(CStr(cities(0)))
    BindHotels(hotels)
```

---

**The HotelBroker Interface**

The HotelBroker interface defines a method to get a city index and a method to get the list of hotels for the city.

```vbnet
Public Interface IHotelBroker
        ' Get a city index
    Function GetCityIndex() As Integer
    ' Get the list of hotels for a city
    Function GetHotels(cityIndex As Integer) As ArrayList
End Interface
```
Private Sub BindHotels(ByVal hotels As ArrayList)
    Dim hotelNames As ArrayList = _
        New ArrayList(hotels.Count)
    Dim hotel As HotelListItem
    For Each hotel In hotels
        hotelNames.Add(hotel.HotelName.Trim())
    Next
    listHotels.DataSource = hotelNames
End Sub

Selecting a City

Finally, we implement the feature that selecting a city causes the hotels for the selected city to be displayed. We can add an event handler for selecting a city by double-clicking on the listCities DropDownList control. The is a shortcut for adding a handler for the primary event for the control. Another method for adding an event handler for this control is to select listCities from the first dropdown in the WebForm1.aspx.vb code window. You can then choose an event from the second dropdown.

Private Sub listCities_SelectedIndexChanged( ByVal sender As System.Object, ByVal e As System.EventArgs) Handles listCities.SelectedIndexChanged
    Dim city As String = listCities.SelectedItem.Text
    Dim hotels As ArrayList = broker.GetHotels(city)
    BindHotels(hotels)
    DataBind()
End Sub

The second method allows you to add a handler for any event of the control. Here is the code for the SelectedIndexChanged event.

Private Sub listCities_SelectedIndexChanged( _
    ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles listCities.SelectedIndexChanged
    Dim city As String = listCities.SelectedItem.Text
    Dim hotels As ArrayList = broker.GetHotels(city)
    BindHotels(hotels)
    DataBind()
End Sub
Build and run the project. Unfortunately, the event does not seem to be recognized by the server. What do you suppose the problem is?

Debugging

One advantage of using Visual Studio for developing your ASP.NET applications is the ease of debugging. You can set breakpoints, single-step, examine the values of variables, and so forth, in your code-behind files just as you would with any other Visual Studio program. All you have to do is build your project in Debug mode (the default) and start the program from within Visual Studio using Debug | Start (or F5 at the keyboard or the toolbar button ).

As an example, set a breakpoint on the first line of the SelectedIndexChanged event handler for listCities. The breakpoint is not hit!

AutoPostBack

For an event to be recognized by the server, you must have a postback to the server. Such a postback happens automatically for a button click, but not for other events. Once this problem is recognized, the remedy is simple. In the Properties window for the cities DropDownList control, change the AutoPostBack property to true. The screen capture illustrates setting the AutoPostBack property.

Set the AutoPostBack property to True, and build and run. Now the breakpoint should be hit.

But no matter what city you select, the hotels dropdown listbox seems to be stuck at the hotels for the first city. What do you suppose is wrong?
IsPostBack Property

The problem is that we are *always* invoking the code in **Page_Init** to populate the hotels dropdown with hotels from cities(0). What we want is only to perform this initialization once, and not on the post backs. The fix is test the **IsPostBack** property.

```vbnet
Private Sub Page_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
  'Put user code to initialize the page here
  If Not IsPostBack Then
    broker = Global.broker
    Dim cities As ArrayList = broker.GetCities()
    listCities.DataSource = cities
    Dim hotels As ArrayList = _
      broker.GetHotels(CStr(cities(0)))
    BindHotels(hotels)
    DataBind()
  End If
End Sub
```