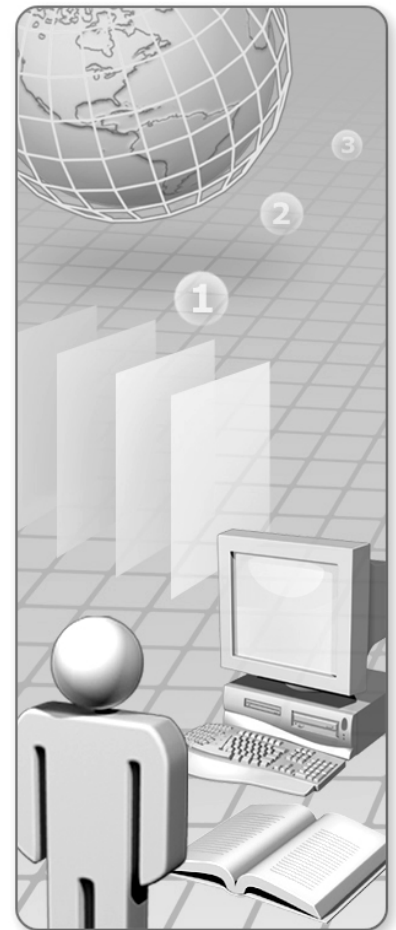


# Unit 6: Accessing and Displaying Data

---

## Contents

Overview	1
Database Connections and the Web.Config File	2
Relational Data and Data Source Controls	4
XML Data and Data Source Controls	6
Object Data and Data Source Controls	7
Lab Scenario	9
Lab Tasks and Objectives	10
Lab: Accessing and Displaying Data	12
Lab Discussion	35



# Overview



- Database Connections and the Web.Config File
- Relational Data and Data Source Controls
- XML Data and Data Source Controls
- Object Data and Data Source Controls
- Lab Scenario
- Lab Tasks and Objectives
- Lab: Accessing and Displaying Data

## Introduction

This unit describes how to add database connections to the Web.Config file and the benefits that this approach adds when building manageable Web applications. This unit then describes the new data controls for accessing data in a variety of formats. It includes details about using the **SqlDataSource** control, the **XmlDataSource** control, and the **ObjectDataSource** control. This unit also describes how user interface data controls are bound to the data source controls, and it includes a discussion about binding data-aware standard controls to data.

## Objectives

After completing this unit, you will be able to:

- Explain how to store and retrieve database connections by using the Web.Config file.
- Explain how to use data source controls to access relational data.
- Explain how to use data source controls to access XML data.
- Explain how to use data source controls to access object data.
- Create and retrieve database connections by using the Web.Config file.
- Access relational data by using the **SqlDataSource** control and data controls.
- Access XML data by using the **XmlDataSource** control and data controls.
- Access objects as data by using the **ObjectDataSource** control and data controls.

# Database Connections and the Web.Config File



- Database Connections and Connection Strings
- Using the Web.config File to Simplify Connection String Management
- Retrieving Connection Strings at Run Time

## Introduction

Web applications use database connections as the underlying mechanism for retrieving data from databases and for inserting, updating, and deleting data.

## Database Connections and Connection Strings

You define database connections in terms of a string that specifies the properties of the connection. These properties usually include the location of the database and security information for accessing the database. Different databases support different properties and therefore require different connection strings; the connection string format depends on the provider being used to open the connection. For example, the provider for Microsoft® SQL Server™ databases supports both Microsoft Windows® authenticated logins and SQL Server authenticated logins, and it requires the name of the server to be included in the connection string. Conversely, the provider for Microsoft Office Access does not support Windows authenticated logins, and it requires the file path to the Access database file rather than a server name.

## Using the Web.config File to Simplify Connection String Management

Although you can hard-code connection strings into your Web application pages, this approach can lead to a difficult-to-manage solution. It is not unusual for data sources to be moved, redefined, or upgraded after an application has been released. For example, you might initially deploy a small Web site with an Access database. As the usage and number of visitors to the site increases, you might decide to upgrade to SQL Server. If you have hard-coded all the connections to use the Access database, you will have to replace every occurrence with the new connection details for the SQL Server database. However, if you have used the Web.Config file to store connection strings, and your Web pages have retrieved the details from the Web.Config file at run time, your solution will be more manageable because you will need only to change the connection details in the Web.Config file.

Connection details are also typically stored in the Web.Config file to ease the deployment of the Web application from the development environment to the testing, staging, and production environments, each of which will typically have different database servers.

The following example shows part of a Web.Config file with a connection string:

```
<connectionStrings>
<add name="AdvWorks"
      connectionString="Server=MySQLSever;Database=AdventureWorks;
      Integrated Security=SSPI;Persist Security Info=True"
      providerName="System.Data.SqlClient"/>
</connectionStrings>
```

The **connectionStrings** element is part of the **configuration** element in the Web.Config file.

## Retrieving Connection Strings at Run Time

If you store a connection string in the **connectionStrings** section of the Web.Config file, you can retrieve the string at run time by using the **ConfigurationManager** class. This class provides a **ConnectionStrings** collection that enumerates the connection strings added to the **connectionStrings** section of the Web.Config file.

The following example shows how to retrieve connection strings at run time and how to use them to open a database connection:

### [Visual Basic]

```
Dim myDataString As String = _
    ConfigurationManager.ConnectionStrings("AdvWorks").ConnectionString
Dim sqlConn As System.Data.SqlClient.SqlConnection = _
    New System.Data.SqlClient.SqlConnection(myDataString)
sqlConn.Open()
```

### [Visual C#]

```
string myDataString =
    ConfigurationManager.ConnectionStrings["AdvWorks"].ConnectionString;
System.Data.SqlClient.SqlConnection sqlConn =
    new System.Data.SqlClient.SqlConnection(myDataString);
sqlConn.Open();
```

# Relational Data and Data Source Controls



- Accessing Relational Databases by Using Data Source Controls
- Displaying Relational Data on Web Pages
- Updating, Inserting, and Deleting Relational Data

## Introduction

Data-driven Web applications frequently retrieve data from databases and render it on Web pages. ASP.NET 2.0 provides controls for performing this type of operation. You can implement many data-driven scenarios using these controls without writing any data access code.

## Accessing Relational Databases by Using Data Source Controls

Microsoft ASP.NET provides data source controls for accessing the data stored in relational databases. For example, it provides the **SqlDataSource** control for accessing data stored in SQL Server databases, and it provides the **AccessDataSource** control for accessing data stored in Access databases. Regardless of the type of database used, the relational data source controls provide standard database functionality, such as retrieval of data, insertion of new data, updating of existing data, and data deletion.

## Displaying Relational Data on Web Pages

The basic process for displaying data from relational databases on Web pages is:

1. Add a data source control to the Web page, and then configure it to connect to the required database.
2. Specify the SELECT statement in the **SelectCommand** property of the data source control, to retrieve the data.
3. Bind data controls or data-aware controls to the data source control.

The data controls available to Web applications include the **GridView**, **DataList**, **DataRepeater**, and **FormView** controls. Data-aware controls are standard controls, such as a **DropDownList** control or a **ListBox** control, that you can bind to a data source.

## Updating, Inserting, and Deleting Relational Data

In addition to the **SelectCommand** property, data source controls provide the following properties:

- **UpdateCommand**
- **InsertCommand**
- **DeleteCommand**

These properties define the database commands to run to insert, update, or delete data from the database.

To enable data editing in a Web page, you can define templates for your data controls so that they include editable controls, such as **Textbox** and **DropDownList** controls, to allow data entry. You can define templates for scenarios such as adding new data and editing existing data. You can bind the controls in editable templates to fields in the data source.

In addition, you can include **Button** controls with specific **CommandName** arguments that invoke specific data operations. The **CommandName** property of a button defined in templates determines what action to take if a user clicks it. For example, the **CommandName** property **New** specifies that a **FormView** control should switch to the **InsertItemTemplate** from the **ItemTemplate**. Similarly, the **CommandName** property **Insert** specifies that data entered into the controls of an **InsertItemTemplate** should be saved to the database.

# XML Data and Data Source Controls



- Accessing XML Data by Using Data Source Controls
- Displaying XML Data on Web Pages

## Introduction

ASP.NET 2.0 provides controls for retrieving XML data from a data source and rendering it. You can use these controls without writing any data access code.

## Accessing XML Data by Using Data Source Controls

ASP.NET provides the **XmlDataSource** control for accessing XML data. You can configure the **XmlDataSource** control to retrieve data from XML files, Web services that return XML data, string variables that contain XML data, and in-memory **XmlDataDocument** objects.



**Note** XML data retrieved by the **XmlDataSource** control is read-only. Unlike the **SqlDataSource** control, the **XmlDataSource** control does not support commands or methods for updating the underlying data source. If you want to update XML data by using a data source control, you can use the **ObjectDataSource** control.

## Displaying XML Data on Web Pages

The basic process for displaying XML data on Web pages is:

1. Add an **XmlDataSource** control to the Web page, and configure it to connect to the required XML source.
2. Bind data-aware controls that are capable of displaying XML data to the **XmlDataSource** control.

Examples of controls that can display XML data include the **TreeView** control and the **Xml** control.

# Object Data and Data Source Controls



- Accessing Object Data by Using Data Source Controls
- Displaying Object Data on Web Pages
- Updating, Inserting, and Deleting Object Data

## Introduction

Web applications can also retrieve data from business objects and render it on Web pages. ASP.NET 2.0 provides controls for performing this type of operation, which you can use without writing any data access code.

## Accessing Object Data by Using Data Source Controls

ASP.NET provides the **ObjectDataSource** source control for accessing the data managed by business objects. The functionality of the business object defines the functionality of the **ObjectDataSource** control, such as whether data can be retrieved, inserted, updated, or deleted.

## Displaying Object Data on Web Pages

The process for displaying data from business objects is very similar to the process for relational data. The basic process is:

1. Add an **ObjectDataSource** control to the Web page, and then configure it to connect to the required business object.
2. Specify the name of the method used to retrieve data from the business object in the **SelectMethod** property.
3. Bind data controls or data-aware controls to the data source control.

Data controls include objects such as the **GridView**, **DataList**, **DataRepeater**, and **FormView** controls. Data-aware controls are standard controls that can be bound to a data source, such as a **DropDownList** control or a **ListBox** control.



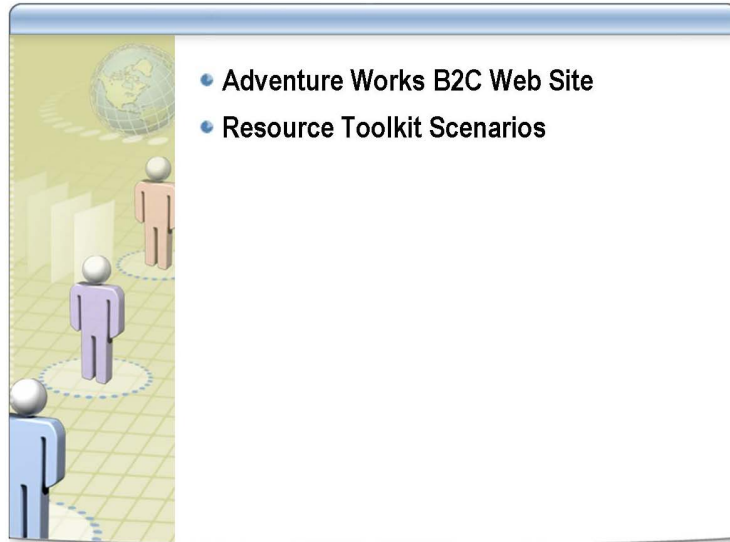
## Updating, Inserting, and Deleting Object Data

In addition to the **SelectMethod** property that defines the data to be retrieved from a business object, **ObjectDataSource** controls also support the following properties:

- **UpdateMethod**
- **InsertMethod**
- **DeleteMethod**

The process for using these properties is similar to that for using the commands of the relational data source controls described previously.

## Lab Scenario



### Adventure Works B2C Web Site

You are a developer in the Adventure Works organization, a fictitious bicycle manufacturer. You have been asked to assist in the development of the Business-to-Consumer (B2C) Web application and a related Business-to-Employee (B2E) extranet portal.

Decisions on the design of the application have already been made. You have been asked to carry out a number of specific tasks in order to implement various elements of this design. As part of the first phase of the B2C development, you have been asked to implement prototypes of pages that display and manipulate data from the AdventureWorks database. You have also been asked to implement prototypes of pages that display and manipulate data from custom object data sources and XML data from the TrailReport Web service.

### Resource Toolkit Scenarios

Before you start work on the lab, you should review the Scenario tab in the Resource Toolkit. The instructor will lead a group discussion of the scenarios for this lab.

## Lab Tasks and Objectives



Task	Objectives
Manage database connections by using the Web.Config file	<ul style="list-style-type: none"> <li>Add a database connection to Web.Config</li> <li>Programmatically retrieve and open the connection</li> </ul>
Display and manipulate relational data	<ul style="list-style-type: none"> <li>Add and configure data source controls for accessing relational data</li> <li>Add and configure data-bound controls for relational data</li> </ul>
Display and manipulate object data	<ul style="list-style-type: none"> <li>Add and configure data source controls for accessing object data</li> <li>Add and configure data-bound controls for object data</li> </ul>
Display and manipulate XML data	<ul style="list-style-type: none"> <li>Add and configure data source controls for accessing XML data</li> <li>Add and configure data-bound controls for XML data</li> </ul>

### Lab Tasks

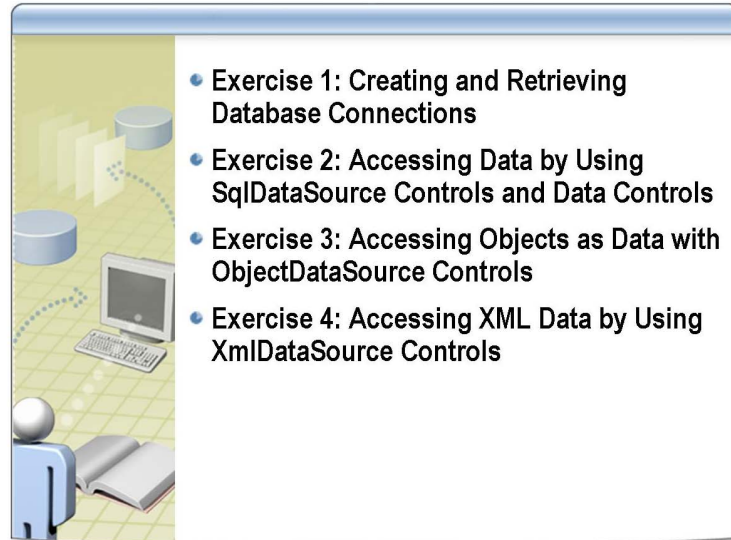
There are four tasks in this lab. Each one is designed to help you achieve one or more learning objectives. Resources are provided in the Resource Toolkit to help you complete the tasks. You should try to complete all of the tasks.

The tasks in this lab are:

- *Manage database connections by using the Web.Config file.* In this task, you will add a database connection string to the Web.Config file, and you will write code that opens a connection using the connection string at run time. The resource for this task is titled *How to: Create and Retrieve Database Connections*.
- *Display and manipulate relational data.* In this task, you will add and configure **SqlDataSource** controls for accessing the AdventureWorks database. These controls will use the connection string you defined in the previous task to access the database. You will also add and configure data-bound controls that manipulate and display the data retrieved by using the **SqlDataSource** controls. The resources for this task are titled:
  - *Products.aspx Page Design*
  - *How to: Access Data by Using SqlDataSource Controls*
  - *How to: Access Data by Using ASP.NET Data-Bound Controls*
  - *How to: Access and Display Data by Using the GridView Control*

- *Display and manipulate object data.* In this task, you will add and configure **ObjectDataSource** controls for accessing data returned by a custom business object. You will also add and configure data-bound controls that manipulate and display the data retrieved by using the **ObjectDataSource** controls. The resource for this task is titled *How to: Access Objects as Data by Using ObjectDataSource Controls*.
- *Display and manipulate XML data.* In this task, you will add and configure **XmlDataSource** controls for accessing XML data returned by the TrailReports Web service. You will also add and configure data-bound controls that manipulate and display the XML data retrieved by using the **XmlDataSource** controls. The resource for this task is titled *How to: Access XML Data by Using XmlDataSource Controls*.

## Lab: Accessing and Displaying Data



After completing this lab, you will be able to:

- Create and retrieve database connections from the Web.Config file.
- Access relational data by using the **SqlDataSource** control and data controls.
- Access XML data by using the **XmlDataSource** controls.
- Access objects as data by using the **ObjectDataSource** control.
- Review code that directly accesses Microsoft ADO.NET 2.0 objects.

Estimated time to complete this lab: **120 minutes**



---

**Important** You can choose to perform this workshop by using either Microsoft Visual Basic® or Microsoft Visual C#®. Code samples and lab solutions are provided in both languages.

---

### Lab Solutions

There are Visual Basic and Visual C# solution files associated with the labs in this workshop. You can find the lab solution files in the folder E:\Labfiles\Solution on the virtual machines.

### Lab Setup

For this lab, you will use the available Microsoft Virtual PC environment. Before you begin the lab, you must:

1. Start the **2543B-LON-DEV-01-06** virtual machine.
2. Log on to the **2543B-LON-DEV-01-06** virtual machine with the user name **Student** and the password **Pa\$\$w0rd**.



No additional setup is required for this lab.

## Exercise 1



### Creating and Retrieving Database Connections

In this exercise, you will add a connection string for the AdventureWorks database to the Web.Config file for the Web application. You will retrieve the connection string from the Web.Config file programmatically, and you will use it to open a connection to the database. You will retrieve the status of the connection to verify that the database is available to the Web application at run time.

#### Scenario

Tasks	Supporting information
<ol style="list-style-type: none"> <li>1. Add a connection string for the AdventureWorks database to the Web.Config file.</li> </ol>	<ol style="list-style-type: none"> <li>a. If you want to complete this lab by using Visual Basic, open the <b>VB.sln</b> file in the <b>E:\Labfiles\Starter\VB\</b> folder.</li> <li>b. If you want to complete this lab by using Visual C#, open the <b>CS.sln</b> file in the <b>E:\Labfiles\Starter\CS\</b> folder.</li> </ol> <p> <b>Note</b> The Microsoft Visual Studio® solution contains a Web application and a Web service.</p> <ol style="list-style-type: none"> <li>c. Open the Web.Config file for the Web application.</li> <li>d. Add a &lt;connectionStrings&gt; element <i>after</i> the <i>closing</i> tag of the <b>appSettings</b> element in the <b>configuration</b> section of the Web.Config file.</li> <li>e. Add an &lt;add&gt; element to the <b>connectionStrings</b> element, with the following attributes: <ul style="list-style-type: none"> <li>• name: <b>AdvWorks</b></li> <li>• connectionString: <b>Server=LON-DEV-01;</b> <ul style="list-style-type: none"> <li>↳ <b>Database=AdventureWorks;</b></li> <li>↳ <b>Integrated Security=SSPI;</b></li> <li>↳ <b>Persist Security Info=False</b></li> </ul> </li> <li>• providerName: <b>System.Data.SqlClient</b></li> </ul> </li> </ol> <p> See the resource in the Resource Toolkit, “How to: Create and Retrieve Database Connections.”</p>

*(continued)*

Tasks	Supporting information
<p>2. Programmatically retrieve the connection to the AdventureWorks database.</p>	<ul style="list-style-type: none"> <li>a. View the code-behind file for the <b>Diagnostics.aspx</b> page.</li> <li>b. Add a <b>try...catch</b> handler after the existing code in the <b>Page_Load</b> method.</li> <li>c. Add code to the <b>try</b> block that declares a string variable named <b>dbStatus</b>.</li> <li>d. Set the <b>dbStatus</b> variable to the <b>ConnectionString</b> property of the <b>AdvWorks</b> member of the <b>ConfigurationManager.ConnectionStrings</b> collection.</li> <li>e. After the code you have added, declare a <b>System.Data.SqlClient.SqlConnection</b> variable called <b>sqlConn</b>, and then set it to a new instance of the <b>System.Data.SqlClient.SqlConnection</b> class, passing in the <b>dbStatus</b> variable as a parameter to the constructor.</li> </ul> <p> See the resource in the Resource Toolkit, “How to: Create and Retrieve Database Connections.”</p>
<p>3. Programmatically open the connection and verify that it is open.</p>	<ul style="list-style-type: none"> <li>a. After the code you have just added, write a statement to invoke the <b>Open</b> method of the <b>sqlConn</b> object. If the database connection is opened successfully, set the <b>Text</b> property of the <b>lblDatabaseStatus</b> label object to the string literal <b>"Database is available"</b>. Otherwise, set the <b>Text</b> property of the <b>lblDatabaseStatus</b> label object to the string literal <b>"Database is currently unavailable"</b>.</li> <li>b. After opening and testing the status of the connection, add a statement that closes the database connection.</li> <li>c. In the <b>catch</b> block, set the <b>Text</b> property of the <b>lblDatabaseStatus</b> label object to the string literal <b>"Database is currently unavailable"</b>. This code informs the user that the database is unavailable if a run-time exception occurs when the user attempts to open the database connection.</li> </ul> <p> See the resource in the Resource Toolkit, “How to: Create and Retrieve Database Connections.”</p>
<p>4. Test the database connectivity.</p>	<ul style="list-style-type: none"> <li>a. Run the Web application. The <b>Default.aspx</b> page loads.</li> <li>b. Click the <b>Diagnostics</b> menu item.</li> <li>c. Verify that the text displayed in the <b>Database Diagnostics</b> section of the Web page is shown as <b>Database is available</b>.</li> <li>d. Close Microsoft Internet Explorer.</li> </ul>

## Exercise 2

### Accessing Data by Using **SqlDataSource** Controls and Data Controls

In this exercise, you will add an **SqlDataSource** control to the `Products.aspx` page. You will configure the control to connect to the AdventureWorks database by specifying that it uses the connection string that you added to the `Web.Config` file in Exercise 1. You will add an SQL `SELECT` query to the control to retrieve data from the AdventureWorks database. The data will be a list of product categories. You will display the product category data in a **DropDownList** control by binding the **DropDownList** control to the **SqlDataSource** control.

Next, you will add a data-bound **GridView** control to the `Products.aspx` page. You will bind the **GridView** control to a new **SqlDataSource** control. The `SELECT` query for this **SqlDataSource** control will return a list of subcategories and will accept a parameter supplied at run time. The parameter will be populated by referencing the selected item in the Categories **DropDownList** control. When the user changes the selected item in the Categories **DropDownList** control, the subcategories **GridView** control will update automatically.






You will add another data-bound **GridView** control to the `Products.aspx` page. You will bind this second **GridView** control to another **SqlDataSource** control. The `SELECT` query for this **SqlDataSource** control will return a list of products and will accept a parameter supplied at run time. The parameter will be populated at run time by referencing the selected item in the Subcategories **GridView**. When the user changes the selected item in the Subcategories **GridView**, the Products **GridView** will update automatically. You will also create a template that specifies what to display if no matching data is available in the database.

Next, you will add a **DetailsView** control to the page and bind it to another **SqlDataSource** control. These controls will display the details of the selected product by running a parameterized query, based on the value selected in the Products **GridView**. In addition to the standard data binding for fields and columns, you will add two templated fields to the **DetailsView** control. These templated fields will contain links that the user can click to see more details of a product, or to review comments made about a product.




Finally, you will add a **DataList** control and another new **SqlDataSource** control to the `ProductDetails.aspx` page. You will create a parameterized `SELECT` statement for the **SqlDataSource** control, where the parameter will be supplied at run time by the page's query string. You will also review the ADO.NET code required to retrieve images from the database.





## Scenario

Tasks	Supporting information
1. Add a <b>SqlDataSource</b> control to the Products page.	<p>a. Open the Products.aspx Web page in Design view.</p> <p>b. Add a <b>SqlDataSource</b> control to the page, next to the <b>Categories</b> label in the Products.aspx Web page.</p> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> <li>▪ “Products.aspx Page Design”</li> <li>▪ “How to: Access Data by Using SqlDataSource Controls”</li> </ul>
2. Set connection properties for the <b>SqlDataSource</b> control.	<p>a. Click <b>Configure Data Source</b> in the Smart Tag for the <b>SqlDataSource</b> control, and use the following information to configure the data source for the <b>SqlDataSource</b> control:</p> <ul style="list-style-type: none"> <li>• Data connection: <b>AdvWorks</b></li> </ul> <p> <b>Note</b> <i>AdvWorks</i> is the name used to identify the database connection you added to the Web.Config file in Exercise 1.</p> <ul style="list-style-type: none"> <li>• SQL statement used to select data:  SELECT Name, ProductCategoryID  FROM Production.ProductCategory  ORDER BY Production.ProductCategory.Name</li> </ul> <p> <b>Note</b> This SQL statement retrieves a list of product category names and IDs from the database, sorted alphabetically by product category name.</p> <p>b. Click <b>Test Query</b> and review the results. Four categories are displayed.</p> <p>c. Click <b>Finish</b>.</p> <p>d. Set the <b>ID</b> property of the <b>SqlDataSource</b> control to <b>dsCategory</b>.</p> <p> See the resource in the Resource Toolkit, “How to: Access Data by Using SqlDataSource Controls.”</p>
3. Bind a <b>DropDownList</b> control to the <b>SqlDataSource</b> control.	<p>a. Add a <b>DropDownList</b> control to the Web page, to the right of the <b>Categories</b> label.</p> <p>b. Set the following properties of the <b>DropDownList</b> control:</p> <ul style="list-style-type: none"> <li>• (ID): <b>ddCategory</b></li> <li>• AutoPostBack: <b>True</b></li> <li>• DataSourceID: <b>dsCategory</b></li> <li>• DataTextField: <b>Name</b></li> <li>• DataValueField: <b>ProductCategoryID</b></li> </ul> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> <li>▪ “Products.aspx Page Design”</li> <li>▪ “How to: Access Data by Using ASP.NET Data-Bound Controls”</li> </ul>




*(continued)*

Tasks	Supporting information
<p>4. Add and configure a Subcategories <b>GridView</b> control with an associated <b>SqlDataSource</b> control.</p>	<p>a. Add a <b>GridView</b> control to the first cell in the empty row underneath the <b>dsCategory</b> control.</p> <p> <b>Note</b> The cell is very small—ensure that you drop the <b>GridView</b> control in the leftmost cell of the empty row.</p> <p>b. Configure the <b>GridView</b> control to use a new data source called <b>dsSubcategory</b>, connecting using the <b>AdvWorks</b> connection string. Use the following SQL statement to retrieve the data:</p> <pre>SELECT ProductCategoryID, ProductSubcategoryID, Name FROM Production.ProductSubcategory WHERE (ProductCategoryID = @ProductCategoryID)</pre> <p> <b>Note</b> This SQL statement retrieves a list of product subcategory names from the database, along with Subcategory and Category IDs. The WHERE clause includes a parameter of <i>@ProductCategoryID</i>—this value will be supplied at run time by referencing the selected value in the <i>ddCategory</i> <b>DropDownList</b> control.</p> <p>c. Specify <b>ddCategory</b> as the source for the parameter and test the query by supplying an <b>Int32</b> value of <b>1</b>. Three subcategories are displayed.</p> <p>d. Set the following properties of the <b>GridView</b> control:</p> <ul style="list-style-type: none"> <li>• (ID): <b>gvwSubcategories</b></li> <li>• ForeColor: <b>#0e0e6c</b></li> <li>• GridLines: <b>None</b></li> </ul> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> <li>▪ “Products.aspx Page Design”</li> <li>▪ “How to: Access and Display Data by Using the GridView Control”</li> </ul>




*(continued)*

Tasks	Supporting information
<p>5. Define the columns for the <b>gvwSubcategories</b> control.</p>	<p>a. In the Smart Tag menu for the <b>gvwSubcategories</b> control, select the <b>Enable Selection</b> check box.</p> <p> <b>Note</b> Enabling selection specifies that the user can select a row in the <b>GridView</b> at run time. It adds a column to the <b>GridView</b> that the user can click to select a row.</p> <p>b. Edit the columns for the <b>gvwSubcategories</b> control. Remove the following items from the <b>Selected fields</b> list:</p> <ul style="list-style-type: none"> <li>• ProductCategoryID</li> <li>• ProductSubcategoryID</li> </ul> <p>c. Move the <b>Name</b> field above the <b>Select</b> field in the <b>Selected fields</b> list.</p> <p>d. Set the <b>HeaderText</b> property of the <b>Name</b> field to <b>Subcategories</b>.</p> <p>e. Set the following properties of the <b>Select</b> field:</p> <ul style="list-style-type: none"> <li>• HeaderText: &lt;Clear this property&gt;</li> <li>• SelectText: [&gt;]</li> <li>• ShowSelectButton: <b>True</b></li> </ul> <p> See the resource in the Resource Toolkit, “How to: Access and Display Data by Using the GridView Control.”</p>




*(continued)*

Tasks	Supporting information
<p>6. Add and configure a Products <b>GridView</b> control and associated <b>SqlDataSource</b> control.</p>	<p>a. Add another <b>GridView</b> control to the Web page and place it in the cell adjacent to the <b>gvwSubcategories GridView</b> control.</p> <p> <b>Note</b> This cell is also small—ensure that you drop the <b>GridView</b> control in the cell directly to the right of the <b>gvwSubcategories GridView</b> control.</p> <p>b. Configure the <b>GridView</b> control to use a new data source called <b>dsProduct</b>, connecting using the <b>AdvWorks</b> connection string. Use the following SQL statement to retrieve the data:</p> <pre>SELECT ProductID, ProductSubcategoryID, Name FROM Production.Product WHERE (ProductSubcategoryID=@ProductSubcategoryID)</pre> <p> <b>Note</b> This SQL statement retrieves a list of product names from the database, along with Product and Subcategory IDs. The WHERE clause includes a parameter of <i>@ProductSubcategoryID</i>—this value will be supplied at run time by referencing the selected value in the <b>gvwSubcategories GridView</b> control.</p> <p>c. Specify <b>gvwSubcategories</b> as the source for the parameter and test the query by supplying an <b>Int32</b> value of <b>2</b>. The details of several road bikes are displayed.</p> <p>d. Set the following properties of the <b>GridView</b> control:</p> <ul style="list-style-type: none"> <li>• (ID): <b>gvwProduct</b></li> <li>• ForeColor: <b>#0e0e6c</b></li> <li>• GridLines: <b>None</b></li> </ul> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> <li>▪ “Products.aspx Page Design”</li> <li>▪ “How to: Access and Display Data by Using the GridView Control”</li> </ul>



(continued)

Tasks	Supporting information
<p>7. Define columns and templates for the <b>gvwProduct GridView</b> control.</p>	<p>a. Using the Smart Tag menu for the <b>gvwProduct</b> control, enable selection and paging in this control.</p> <p> <b>Note</b> Enabling paging specifies that the GridView displays a limited number of items and allows the user to navigate to the next (or previous) page to view other items.</p> <p>b. Edit the columns for the <b>gvwProduct</b> control. Remove the following items from the <b>Selected Fields</b> list:</p> <ul style="list-style-type: none"> <li>• ProductID</li> <li>• ProductSubcategoryID</li> </ul> <p>c. Move the <b>Name</b> field above the <b>Select</b> field in the <b>Selected Fields</b> list.</p> <p>d. Set the following properties of the <b>Name</b> field:</p> <ul style="list-style-type: none"> <li>• ShowHeader: <b>False</b></li> <li>• ItemStyle Wrap: <b>False</b></li> </ul> <p> <b>Tip</b> Expand the <b>ItemStyle</b> property, and then scroll down until you can see the <b>Wrap</b> property.</p> <p>e. Set the following properties of the <b>Select</b> field:</p> <ul style="list-style-type: none"> <li>• HeaderText: &lt;Clear this property&gt;</li> <li>• SelectText: [ &gt; ]</li> <li>• ShowSelectButton: <b>True</b></li> </ul> <p>f. In the Smart Tag menu for the <b>gvwProduct</b> control, click <b>Edit Templates</b>.</p> <p>g. Ensure that <b>EmptyDataTemplate</b> is selected in the <b>Display</b> list.</p> <p>h. Type <b>Please select a subcategory</b> in the <b>EmptyDataTemplate</b> area of the <b>gvwProduct GridView</b> control.</p> <p>i. Click <b>End Template Editing</b> in the Smart Tag menu for the <b>gvwProduct</b> control.</p> <p> <b>Note</b> The <b>EmptyDataTemplate</b> area defines the content of the control when no data is returned from the underlying data source.</p> <p>j. Set the <b>PagerSettings</b> property for the <b>gvwProduct</b> control as follows:</p> <ul style="list-style-type: none"> <li>• Mode: <b>NextPrevious</b></li> <li>• NextPageText: [Next]</li> <li>• PreviousPageText: [Previous]</li> </ul> <p>k. Set the following property of the <b>SelectedRowStyle</b> property for the <b>gvwProduct</b> control:</p> <ul style="list-style-type: none"> <li>• BackColor: <b>#FFC0C0</b></li> </ul>

*(continued)*

Tasks	Supporting information
7. <i>(continued)</i>	<p>I. Switch to Source view and review the markup for the <b>GridView</b> controls and the <b>SqlDataSource</b> controls.</p> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> <li>▪ “Products.aspx Page Design”</li> <li>▪ “How to: Access and Display Data by Using the GridView Control”</li> </ul>
8. Add and configure a Product Details <b>DetailsView</b> control and associated <b>SqlDataSource</b> control.	<p>a. Add the markup from the following file to the cell that is adjacent to the <b>gvwProduct GridView</b> control.</p> <ul style="list-style-type: none"> <li>• E:\Labfiles\Starter\StarterText\Ex2-Step8-dvwProducts.txt</li> </ul> <p>b. Review the markup that you have just added. It defines a <b>DetailsView</b> control.</p> <p>c. Add the markup from the following file to the cell that contains the <b>dvwProduct</b> control you have just added</p> <ul style="list-style-type: none"> <li>• E:\Labfiles\Starter\StarterText\Ex2-Step8-dsProductDetails.txt</li> </ul> <p>d. Review the markup that you have just added. It defines a <b>SqlDataSource</b> control.</p> <p> <b>Note</b> The SQL statement used by the <b>SqlDataSource</b> control retrieves a list of product names, colors, and prices from the database, along with Product IDs. The WHERE clause includes a parameter of <i>@ProductID</i>—this value will be supplied at run time by referencing the selected value in the <b>gvwProduct GridView</b> control (as specified by the <code>&lt;SelectParameters&gt;</code> element).</p> <p>e. Review the <code>&lt;asp:TemplateField&gt;</code> elements in the markup for <b>dvwProduct</b> control. These items are used to build custom columns at run time. In this case, they build hyperlinks based on literal HTML markup, product IDs, and product names that are determined at run time.</p> <p> See the following resources in the Resource Toolkit:</p> <ul style="list-style-type: none"> <li>▪ “Products.aspx Page Design”</li> <li>▪ “How to: Access and Display Data by Using the GridView Control”</li> </ul>

(continued)

Tasks	Supporting information
<p>9. Add a <b>DataList</b> control to the ProductDetails.aspx page.</p>	<p>a. Open the code-behind file for the readImage.aspx page.</p> <p>b. Uncomment the code in the <b>Page_Load</b> method, and review it.</p> <p> <b>Note</b> This ADO.NET code programmatically accesses the data in the AdventureWorks database to retrieve product images as a stream of bytes. It then writes the stream of bytes that represent the images to the <b>Response</b> object. This response will be used by the ProductDetails.aspx page as the <b>src</b> attribute of &lt;img&gt; elements. The page can also be displayed in the browser directly and will return an image in response to the request.</p> <p>c. Open the ProductDetails.aspx page in Source view.</p> <p>d. If you are working with Visual Basic, add the markup from the following file to the empty space between the start and end tags of the content control.</p> <ul style="list-style-type: none"> <li>E:\Labfiles\Starter\StarterText\Ex2-Step9-VB_dlProductDetails.txt</li> </ul> <p>e. If you are working with Visual C#, add the markup from the following file to the empty space between the start and end tags of the content control.</p> <ul style="list-style-type: none"> <li>E:\Labfiles\Starter\StarterText\Ex2-Step9-CS_dlProductDetails.txt</li> </ul> <p>f. Review the markup that you have just added. It defines a <b>DataList</b> control with templated fields. The <b>DataList</b> control references a <b>SqlDataSource</b> control that you will add next.</p>
<p>10. Add an <b>SqlDataSourceControl</b> to the page.</p>	<p>a. After the closing tag of the <b>asp:DataList</b> element you added in step 9, add the markup from the following file:</p> <ul style="list-style-type: none"> <li>E:\Labfiles\Starter\StarterText\Ex2-Step10-dsProductDetails.txt</li> </ul> <p>b. Review the markup you have just added.</p> <p> <b>Note</b> The SQL statement used by the SqlDataSource control retrieves a list of product names, colors, and prices from the database, along with Product IDs. The WHERE clause includes a parameter of <i>@ProductID</i>—this value will be supplied at run time by referencing the QueryString of the page (as specified by the &lt;SelectParameters&gt; element).</p>

*(continued)*

Tasks	Supporting information
<p>11. Run and test the Web application.</p>	<ul style="list-style-type: none"> <li>a. Save all files and then run the Web application.</li> <li>b. Click the <b>Products</b> menu item. The <b>Categories</b> drop-down list is populated with four categories, and the <b>Subcategories</b> GridView displays subcategories from the Accessories category. The <b>Products</b> GridView displays the text <b>Please select a subcategory</b>. This is the text you specified for the <b>EmptyDataTemplate</b> of the <b>gvwProducts</b> control.</li> <li>c. Click <b>Bikes</b> in the <b>Categories</b> list. The <b>Subcategories</b> GridView control automatically updates to display the bike subcategories.</li> <li>d. Click [<b>&gt;</b>] for the <b>Mountain Bikes</b> subcategory. The <b>Products</b> GridView displays 10 mountain bike products at a time as a result of the <b>Paging</b> property that you set previously for the GridView. <ul style="list-style-type: none"> <li>• Click [<b>Next</b>] at the bottom of the <b>Products</b> GridView. The next 10 mountain bike products are displayed.</li> <li>• Click [<b>Previous</b>] at the bottom of the <b>Products</b> GridView. The first 10 mountain bike products are displayed again.</li> <li>• Click [<b>&gt;</b>] for the <b>Mountain-100 Silver, 44</b> product. The product details are displayed.</li> <li>• Click [<b>More Details</b>]. The ProductDetails.aspx page is displayed. The product image is retrieved from the database by the readImage.aspx page and returned as the source of the image control.</li> <li>• Click the product image. The readImage.aspx page opens in a new window and displays the large version of the product image.</li> <li>• Close both instances of Internet Explorer.</li> </ul> </li> </ul>



## Exercise 3


### Accessing Objects as Data with ObjectDataSource Controls

In this exercise, you will add a **FormView** control and associated **SqlDataSource** control to the ProductReview.aspx page. The **FormView** control will support two modes: viewing existing data from the database and adding new review data. You will define the templates for these two modes, and then you will test the entire database-driven features of the Web application.


### Scenario

Tasks	Supporting information
<ol style="list-style-type: none"> <li>1. Review the data-access code in the business object.</li> </ol>	<ol style="list-style-type: none"> <li>a. If you are working with Visual Basic, open the <b>Reviews.vb</b> file in the <b>App_Code</b> folder.</li> <li>b. If you are working with Visual C#, open the <b>Reviews.cs</b> file in the <b>App_Code</b> folder.</li> <li>c. Uncomment and review the code for the <b>GetReviews</b> method. <div data-bbox="641 856 701 919" data-label="Image"></div> <b>Note</b> This method retrieves data directly from the AdventureWorks database and returns it to the caller as an enumerable list. It will be used by the ProductReview.aspx page as the data source for an <b>ObjectDataSource</b> control. </li> <li>d. Uncomment and read through the code for the <b>AddReview</b> method.</li> <li>e. Save all files. <div data-bbox="641 1113 701 1176" data-label="Image"></div> <b>Note</b> This method inserts data into the AdventureWorks database. It will be used by the ProductReview.aspx page to handle inserts in response to the data-bound controls performing data additions. Also note that this code would require checks for malicious content (such as SQL injection attacks) in a real solution. These checks have been omitted for this lab to simplify the solution and to help you understand the process involved in adding data to a database. </li> </ol>

(continued)

Tasks	Supporting information
<p>2. Configure an <b>ObjectDataSource</b> control for business-data access.</p>	<ul style="list-style-type: none"> <li>a. Open the ProductReview.aspx page.</li> <li>b. Add an <b>ObjectDataSource</b> control to the <b>Content</b> control on the ProductReview.aspx page.</li> <li>c. Click <b>Configure Data Source</b> in the Smart Tag for the <b>ObjectDataSource</b> control. The <b>Configure Data Source</b> dialog box appears.</li> <li>d. Click <b>Reviews</b> in the <b>Choose your business object</b> list, and then click <b>Next</b>. <i>Reviews</i> is the name of the class in the code file for which you have just uncommented two methods.</li> <li>e. On the <b>SELECT</b> tab of the <b>Define Data Methods</b> page, select the <b>GetReviews</b> method in the <b>Choose a method</b> list.</li> <li>f. On the <b>INSERT</b> tab of the <b>Define Data Methods</b> page, select the <b>AddReview</b> method in the <b>Choose a method</b> list.</li> <li>g. Click <b>Next</b>.</li> <li>h. On the <b>Define Parameters</b> page, click <b>QueryString</b> in the <b>Parameter source</b> list.</li> <li>i. Type <b>ProductID</b> in the <b>QueryStringField</b> box, and then click <b>Finish</b>.</li> <li>j. Set the <b>ID</b> property of the <b>ObjectDataSource</b> control to <b>objReview</b>.</li> </ul> <p> See the resource in the Resource Toolkit, “How to: Access Objects as Data by Using ObjectDataSource Controls.”</p>
<p>3. Add and configure a <b>FormView</b> control.</p>	<ul style="list-style-type: none"> <li>a. Add a <b>FormView</b> control to the <b>Content</b> control on the ProductReview.aspx page.</li> <li>b. In the <b>Properties</b> window, in the <b>AllowPaging</b> list, click <b>True</b>.</li> <li>c. In the <b>ForeColor</b> box, type <b>#0E0E6C</b>.</li> <li>d. Set the data source of the <b>FormView</b> control to <b>objReview</b>.</li> </ul>



(continued)

Tasks	Supporting information						
4. Define the <b>EmptyDataTemplate</b> element for the <b>FormView</b> control.	<p>a. Switch to Source view.</p> <p>b. Add an <b>EmptyDataTemplate</b> element to the <b>FormView</b> control that contains the following items:</p> <table> <tr> <td>Literal text</td><td>No reviews exist for this product. Please add a review if you have used this product.</td></tr> <tr> <td colspan="2">A <b>&lt;br /&gt;</b> element.</td></tr> <tr> <td>The markup for an <b>asp:Button</b> control</td><td>           Add the following attributes to the <b>asp:Button</b> element:           <ul style="list-style-type: none"> <li>• ID: <b>btnAdd</b></li> <li>• Text: <b>Add review</b></li> <li>• CommandName: <b>New</b></li> <li>• runat: <b>server</b></li> </ul> </td></tr> </table>	Literal text	No reviews exist for this product. Please add a review if you have used this product.	A <b>&lt;br /&gt;</b> element.		The markup for an <b>asp:Button</b> control	Add the following attributes to the <b>asp:Button</b> element: <ul style="list-style-type: none"> <li>• ID: <b>btnAdd</b></li> <li>• Text: <b>Add review</b></li> <li>• CommandName: <b>New</b></li> <li>• runat: <b>server</b></li> </ul>
Literal text	No reviews exist for this product. Please add a review if you have used this product.						
A <b>&lt;br /&gt;</b> element.							
The markup for an <b>asp:Button</b> control	Add the following attributes to the <b>asp:Button</b> element: <ul style="list-style-type: none"> <li>• ID: <b>btnAdd</b></li> <li>• Text: <b>Add review</b></li> <li>• CommandName: <b>New</b></li> <li>• runat: <b>server</b></li> </ul>						
5. Define the <b>ItemTemplate</b> element for the <b>FormView</b> control.	<p> <b>Note</b> Use single quotes around the expressions used in the <b>Text</b> properties of all the steps that follow, rather than double quotes.</p> <p>a. Add an <b>ItemTemplate</b> element to the <b>FormView</b> control that contains the following items:</p> <table> <tr> <td>Literal text</td><td><b>&lt;b&gt;Product ID: &lt;/b&gt;</b></td></tr> <tr> <td>The markup for an <b>asp:Label</b> control</td><td>           Add the following attributes to the <b>asp:Label</b> element:           <ul style="list-style-type: none"> <li>• ID: <b>lblProductID</b></li> <li>• runat: <b>server</b></li> <li>• Text: <b>&lt;%# Eval("ProductID") %&gt;</b></li> </ul> </td></tr> <tr> <td colspan="2">A <b>&lt;br /&gt;</b> element</td></tr> </table>	Literal text	<b>&lt;b&gt;Product ID: &lt;/b&gt;</b>	The markup for an <b>asp:Label</b> control	Add the following attributes to the <b>asp:Label</b> element: <ul style="list-style-type: none"> <li>• ID: <b>lblProductID</b></li> <li>• runat: <b>server</b></li> <li>• Text: <b>&lt;%# Eval("ProductID") %&gt;</b></li> </ul>	A <b>&lt;br /&gt;</b> element	
Literal text	<b>&lt;b&gt;Product ID: &lt;/b&gt;</b>						
The markup for an <b>asp:Label</b> control	Add the following attributes to the <b>asp:Label</b> element: <ul style="list-style-type: none"> <li>• ID: <b>lblProductID</b></li> <li>• runat: <b>server</b></li> <li>• Text: <b>&lt;%# Eval("ProductID") %&gt;</b></li> </ul>						
A <b>&lt;br /&gt;</b> element							

*(continued)*

Tasks	Supporting information
5. <i>(continued)</i>	<div> <div>Literal text</div> <div>&lt;b&gt;Reviewer: &lt;/b&gt;</div> </div>
	<div> <div>The markup for an <b>asp:Label</b> control</div> <div>           Add the following attributes to the <b>asp:Label</b> element:           <ul style="list-style-type: none"> <li>ID: <b>lblReviewerName</b></li> <li>runat: <b>server</b></li> <li>Text: &lt;%# Eval("ReviewerName") %&gt;</li> </ul> </div> </div>
	A   element
	<div> <div>Literal text</div> <div>&lt;b&gt;Review Date: &lt;/b&gt;</div> </div>
	<div> <div>The markup for an <b>asp:Label</b> control</div> <div>           Add the following attributes to the <b>asp:Label</b> element:           <ul style="list-style-type: none"> <li>ID: <b>lblReviewDate</b></li> <li>runat: <b>server</b></li> <li>Text: &lt;%# Eval("ReviewDate") %&gt;</li> </ul> </div> </div>
	A   element
	<div> <div>Literal text</div> <div>&lt;b&gt;Rating: &lt;/b&gt;</div> </div>
	<div> <div>The markup for an <b>asp:Label</b> control</div> <div>           Add the following attributes to the <b>asp:Label</b> element:           <ul style="list-style-type: none"> <li>ID: <b>lblRating</b></li> <li>runat: <b>server</b></li> <li>Text: &lt;%# Eval("Rating") %&gt;</li> </ul> </div> </div>
	A   element
	<div> <div>Literal text</div> <div>&lt;b&gt;Comments: &lt;/b&gt;</div> </div>
	<div> <div>The markup for an <b>asp:Label</b> control</div> <div>           Add the following attributes to the <b>asp:Label</b> element:           <ul style="list-style-type: none"> <li>ID: <b>lblComments</b></li> <li>runat: <b>server</b></li> <li>Text: &lt;%# Eval("Comments") %&gt;</li> </ul> </div> </div>
	A   element



(continued)

Tasks	Supporting information																
5. (continued)	<p>The markup for an <b>asp:Button</b> control</p> <p>Add the following attributes to the <b>asp:Button</b> element:</p> <ul style="list-style-type: none"> <li>• ID: <b>btnAdd</b></li> <li>• Text: <b>Add review</b></li> <li>• CommandName: <b>New</b></li> <li>• runat: <b>server</b></li> </ul> <p> <b>Note</b> The <b>CommandName</b> property of buttons defined in templates determines what action to take if a user clicks them. In this case, the <b>CommandName</b> property <b>New</b> specifies that the <b>FormView</b> control should switch to the <b>InsertItemTemplate</b> from the <b>ItemTemplate</b>.</p>																
6. Define the <b>InsertItemTemplate</b> element for the <b>FormView</b> control.	<p>a. Switch to Design View.</p> <p>b. Add an <b>InsertItemTemplate</b> element to the <b>FormView</b> control that contains an HTML table with a width of 100%.</p> <p>c. Add seven rows to the table, with two cells in each row.</p> <p>d. Add the following literal text to the specified cells of the table:</p> <table border="1"> <tr> <td>Row 1, cell 1</td><td><b>Product</b></td></tr> <tr> <td>Row 2, cell 1</td><td><b>Your Name</b></td></tr> <tr> <td>Row 3, cell 1</td><td><b>Email Address</b></td></tr> <tr> <td>Row 4, cell 1</td><td><b>Review Date</b></td></tr> <tr> <td>Row 5, cell 1</td><td><b>Rating</b></td></tr> <tr> <td>Row 6, cell 1</td><td><b>Comments</b></td></tr> </table> <p>e. Switch to Source View.</p> <p>f. Add the following controls, text, or code to the specified cells of the table.</p> <p> <b>Note</b> Use single quotes around the expressions used in the <b>Text</b> properties and the <b>SelectedValue</b> properties of all the steps that follow, rather than double quotes.</p> <table border="1"> <tr> <td>Row 1, cell 2</td><td>Code that concatenates the <b>ProductName</b> parameter of the <b>QueryString</b> with the <b>ProductID</b> parameter of the <b>QueryString</b>. Include literal parentheses around the <b>ProductID</b> parameter.</td></tr> <tr> <td>Row 2, cell 2</td><td>An <b>asp:TextBox</b> control with the following attributes: <ul style="list-style-type: none"> <li>• ID: <b>txtReviewerName</b></li> <li>• runat: <b>server</b></li> <li>• Text: <b>&lt;%# Bind("ReviewerName") %&gt;</b></li> </ul> </td></tr> </table>	Row 1, cell 1	<b>Product</b>	Row 2, cell 1	<b>Your Name</b>	Row 3, cell 1	<b>Email Address</b>	Row 4, cell 1	<b>Review Date</b>	Row 5, cell 1	<b>Rating</b>	Row 6, cell 1	<b>Comments</b>	Row 1, cell 2	Code that concatenates the <b>ProductName</b> parameter of the <b>QueryString</b> with the <b>ProductID</b> parameter of the <b>QueryString</b> . Include literal parentheses around the <b>ProductID</b> parameter.	Row 2, cell 2	An <b>asp:TextBox</b> control with the following attributes: <ul style="list-style-type: none"> <li>• ID: <b>txtReviewerName</b></li> <li>• runat: <b>server</b></li> <li>• Text: <b>&lt;%# Bind("ReviewerName") %&gt;</b></li> </ul>
Row 1, cell 1	<b>Product</b>																
Row 2, cell 1	<b>Your Name</b>																
Row 3, cell 1	<b>Email Address</b>																
Row 4, cell 1	<b>Review Date</b>																
Row 5, cell 1	<b>Rating</b>																
Row 6, cell 1	<b>Comments</b>																
Row 1, cell 2	Code that concatenates the <b>ProductName</b> parameter of the <b>QueryString</b> with the <b>ProductID</b> parameter of the <b>QueryString</b> . Include literal parentheses around the <b>ProductID</b> parameter.																
Row 2, cell 2	An <b>asp:TextBox</b> control with the following attributes: <ul style="list-style-type: none"> <li>• ID: <b>txtReviewerName</b></li> <li>• runat: <b>server</b></li> <li>• Text: <b>&lt;%# Bind("ReviewerName") %&gt;</b></li> </ul>																


*(continued)*

Tasks	Supporting information
6. <i>(continued)</i>	<p>Row 3, cell 2</p> <p>An <b>asp:TextBox</b> control with the following attributes:</p> <ul style="list-style-type: none"> <li>• ID: <b>txtEmail</b></li> <li>• runat: <b>server</b></li> <li>• Text: <b>&lt;%# Bind("EmailAddress") %&gt;</b></li> </ul>
	<p>Row 4, cell 2</p> <p>An <b>asp:Label</b> control with the following attributes:</p> <ul style="list-style-type: none"> <li>• ID: <b>lblDate</b></li> <li>• runat: <b>server</b></li> <li>• Text: <b>&lt;%# Bind("ReviewDate") %&gt;</b></li> </ul> <p>Add the following literal text between the opening and closing tags of the <b>asp:Label</b> control you have just added:</p> <ul style="list-style-type: none"> <li>• <b>&lt;% =DateTime.Now.ToLongDateString() %&gt;</b></li> </ul>
	<p>Row 5, cell 2</p> <p>An <b>asp:DropDownList</b> control with the following attributes:</p> <ul style="list-style-type: none"> <li>• ID: <b>ddRating</b></li> <li>• runat: <b>server</b></li> <li>• SelectedValue: <b>&lt;%# Bind("Rating") %&gt;</b></li> </ul> <p>Add five <b>ListItem</b> elements to the <b>DropDownList</b> control, with <b>value</b> attributes starting at 1 for the first <b>ListItem</b> element and increasing by 1 for each of the other <b>ListItem</b> elements.</p>
	<p>Row 6, cell 2</p> <p>An <b>asp:TextBox</b> control with the following attributes:</p> <ul style="list-style-type: none"> <li>• ID: <b>txtComments</b></li> <li>• TextMode: <b>MultiLine</b></li> <li>• Rows: <b>10</b></li> <li>• runat: <b>server</b></li> <li>• Text: <b>&lt;%# Bind("Comments") %&gt;</b></li> </ul>
	<p>g. Add an <b>asp:Button</b> control to the first cell of the seventh row with the following attributes:</p> <ul style="list-style-type: none"> <li>• ID: <b>btnInsert</b></li> <li>• runat: <b>server</b></li> <li>• CommandName: <b>Insert</b></li> <li>• Text: <b>Save Review</b></li> </ul>

*(continued)*

Tasks	Supporting information
6. <i>(continued)</i>	<p>h. Add an <b>asp:Button</b> control to the second cell of the seventh row with the following attributes:</p> <ul style="list-style-type: none"> <li>• ID: <b>btnCancel</b></li> <li>• runat: <b>server</b></li> <li>• CommandName: <b>Cancel</b></li> <li>• Text: <b>Cancel</b></li> </ul> <p> <b>Note</b> The <b>CommandName</b> property <b>Insert</b> specifies that the data contained in the bound controls should be used in the <b>Insert</b> command of the data source. The <b>CommandName</b> property <b>Cancel</b> specifies that the values in the data-bound controls should be cleared and that the <b>FormView</b> control should switch to the <b>ItemTemplate</b> from the <b>InsertItemTemplate</b>.</p>
7. Add code to the <b>Inserting</b> event of the <b>FormView</b> control.	<p>a. Add an <b>ItemInserting</b> event handler for the <b>FormView</b> control.</p> <p>b. If you are working with Visual Basic, add the following code to the event handler:</p> <pre>e.Values("ProductID") = _ Request.QueryString("ProductID")</pre> <p>c. If you are working with Visual C#, add the following code to the event handler:</p> <pre>e.Values["ProductID"] = Request.QueryString["ProductID"];</pre> <p> <b>Note</b> The code you have just added is required so that the <b>INSERT</b> statement contained in the <b>Reviews</b> class does not fail the Foreign Key constraint for referential integrity in the AdventureWorks database.</p>
8. Run and test the Web application.	<p>a. Save all files and then run the Web application.</p> <p>b. Click the <b>Products</b> menu item. The <b>Categories</b> drop-down list is populated with four categories, and the <b>Subcategories</b> GridView displays subcategories from the Accessories category. The <b>Products</b> GridView displays the text <b>Please select a Subcategory</b>. This is the text you specified for the <b>EmptyDataTemplate</b> of the <b>gvwProducts</b> control in Exercise 2.</p> <p>c. Click <b>Bikes</b> in the <b>Categories</b> list. The <b>Subcategories</b> GridView control automatically updates to display the bike subcategories.</p> <p>d. Click [<b>&gt;</b>] for the <b>Mountain Bikes</b> subcategory. The <b>Products</b> GridView displays 10 mountain bike products.</p> <p>e. Click [<b>&gt;</b>] for the <b>Mountain-100 Silver, 44</b> product. The product details are displayed.</p> <p>f. Click [<b>Product Reviews</b>]. The ProductReview.aspx page is displayed, which contains the text and button you defined for the <b>EmptyDataTemplate</b> of the <b>FormView</b> control.</p>

*(continued)*

Tasks	Supporting information
8. <i>(continued)</i>	<p><b>g.</b> Click <b>Add review</b>. The <b>FormView</b> control displays the controls and text you defined for the <b>InsertItemTemplate</b>.</p> <p><b>h.</b> Type your name in the <b>Your Name</b> box.</p> <p><b>i.</b> Type <b>someone@microsoft.com</b> in the <b>Email Address</b> box.</p> <p><b>j.</b> Click <b>4</b> in the <b>Rating</b> list.</p> <p><b>k.</b> Type <b>Great bike, but quite heavy on steep climbs!</b> in the <b>Comments</b> box.</p> <p><b>l.</b> Click <b>Save Review</b>. The data you have just entered is added to the AdventureWorks database. The <b>FormView</b> control now displays that data as the first review of the product, by rendering the data with the text and controls you defined for the <b>ItemTemplate</b>.</p>
	<p> <b>Note</b> Using certain characters such as a single quote (') or a semicolon (;) in your review will cause an error on the page because these are also SQL characters that will affect the insert statement the page builds to submit the review to the database.</p>
	<p><b>m.</b> Add another review for the product. After you have saved it, notice that you can move between the two reviews by using the numbered navigation features of the <b>FormView</b> control.</p> <p><b>n.</b> If an AutoComplete dialog box appears, click <b>No</b>.</p> <p><b>o.</b> Close Internet Explorer.</p>




## Exercise 4


### Accessing XML Data by Using XmlDataSource Controls

In this exercise, you will add an **XmlDataSource** object to the TrailReport.aspx page to handle XML data from the TrailReport Web service. You will also add a **TreeView** control to the page and bind it to the **XmlDataSource** object. You will specify which elements of the XML data are bound to which levels of the **TreeView** control. Then you will add the code to retrieve the XML data from the Web service, and you will use that data as the source for the **XmlDataSource** control. Finally, you will test the functionality of the TrailReport.aspx Web page.

### Scenario

Tasks	Supporting information						
1. Add an <b>XmlDataSource</b> object to the TrailReport.aspx page.	<ol style="list-style-type: none"> <li>Open the TrailReport.aspx page.</li> <li>Add an <b>XmlDataSource</b> object to the lower part of the <b>Content</b> control.</li> <li>Set the <b>ID</b> attribute to <b>xmlTrailsSource</b>.</li> <li>Verify that the <b>runat</b> attribute is set to <b>server</b>.</li> </ol>  See the resource in the Resource Toolkit, “How to: Access XML Data by Using XmlDataSource Controls.”						
2. Add and configure a <b>TreeView</b> control for the TrailReport.aspx page.	<ol style="list-style-type: none"> <li>Add a <b>TreeView</b> control to the empty cell in the upper left of the <b>Content</b> control.</li> <li>Set the following attributes for the <b>TreeView</b> control: <ul style="list-style-type: none"> <li>ID: <b>tvwTrails</b></li> <li>DataSourceID: <b>xmlTrailsSource</b></li> <li>ExpandDepth: <b>1</b></li> <li>BackColor: <b>#0E0E6C</b></li> </ul> </li> <li>Set the <b>BackColor</b> of the <b>SelectedNodeStyle</b> property of the TreeView control to <b>#0E0E9C</b>.</li> <li>Add three <b>TreeNodeBinding</b> elements to the <b>TreeView</b> control, as follows: <table border="1"> <tbody> <tr> <td>First <b>TreeNodeBinding</b> element</td><td> <ul style="list-style-type: none"> <li>Depth: <b>1</b></li> <li>DataMember: <b>Region</b></li> <li>TextField: <b>RegionName</b></li> </ul> </td></tr> <tr> <td>Second <b>TreeNodeBinding</b> element</td><td> <ul style="list-style-type: none"> <li>Depth: <b>2</b></li> <li>DataMember: <b>Area</b></li> <li>TextField: <b>AreaName</b></li> </ul> </td></tr> <tr> <td>Third <b>TreeNodeBinding</b> element</td><td> <ul style="list-style-type: none"> <li>Depth: <b>3</b></li> <li>DataMember: <b>Trail</b></li> <li>TextField: <b>TrailName</b></li> </ul> </td></tr> </tbody> </table> </li> </ol>	First <b>TreeNodeBinding</b> element	<ul style="list-style-type: none"> <li>Depth: <b>1</b></li> <li>DataMember: <b>Region</b></li> <li>TextField: <b>RegionName</b></li> </ul>	Second <b>TreeNodeBinding</b> element	<ul style="list-style-type: none"> <li>Depth: <b>2</b></li> <li>DataMember: <b>Area</b></li> <li>TextField: <b>AreaName</b></li> </ul>	Third <b>TreeNodeBinding</b> element	<ul style="list-style-type: none"> <li>Depth: <b>3</b></li> <li>DataMember: <b>Trail</b></li> <li>TextField: <b>TrailName</b></li> </ul>
First <b>TreeNodeBinding</b> element	<ul style="list-style-type: none"> <li>Depth: <b>1</b></li> <li>DataMember: <b>Region</b></li> <li>TextField: <b>RegionName</b></li> </ul>						
Second <b>TreeNodeBinding</b> element	<ul style="list-style-type: none"> <li>Depth: <b>2</b></li> <li>DataMember: <b>Area</b></li> <li>TextField: <b>AreaName</b></li> </ul>						
Third <b>TreeNodeBinding</b> element	<ul style="list-style-type: none"> <li>Depth: <b>3</b></li> <li>DataMember: <b>Trail</b></li> <li>TextField: <b>TrailName</b></li> </ul>						

(continued)

Tasks	Supporting information
2. (continued)	 <p><b>Note</b> The XML data that will be used to populate the <b>TreeView</b> control consists of the following hierarchical elements:</p> <p style="margin-left: 40px;">Region Area Trail</p> <p>The <i>Region</i> elements each contain an attribute called <b>RegionName</b>, and the <i>Area</i> elements each contain an attribute called <b>AreaName</b>. Similarly, the <i>Trail</i> elements each contain an attribute called <b>TrailName</b>. The bindings you have specified simply bind levels of the <b>TreeView</b> control to corresponding levels of the XML hierarchy and display the attributes as the text of the tree nodes.</p>
3. Add code to an event procedure for the <b>TreeView</b> control.	<p>a. Add an event handler for the <b>SelectedNodeChanged</b> event of the <b>TreeView</b> control.</p> <p>b. In the event handler, examine the <b>ChildNodes.Count</b> property of the <b>tvwTrails.SelectedNode</b> object. If it is equal to zero, perform the following actions:</p> <ul style="list-style-type: none"> <li>• Declare a variable named <b>report</b> of type <b>TrailReportWebService.Report</b> and set it to a new instance of the <b>TrailReportWebService.Report</b> class.</li> <li>• Declare a string variable named <b>trailName</b> and set it to the <b>Text</b> property of the <b>tvwTrails.SelectedNode</b> object.</li> <li>• Declare a string variable named <b>trailWeatherReport</b> and set it to the return value of invoking the <b>report.weatherReport()</b> method, passing in a parameter of the <b>Text</b> property of the <b>tvwTrails.SelectedNode</b> object.</li> <li>• Declare a string variable named <b>trailConditionReport</b> and set it to the return value of invoking the <b>report.trailReport()</b> method, passing in a parameter of the <b>Text</b> property of the <b>tvwTrails.SelectedNode</b> object.</li> <li>• Set the <b>Text</b> property of the <b>lblTrailName</b> label to the <b>trailName</b> variable.</li> <li>• Set the <b>Text</b> property of the <b>lblCondition</b> label to the <b>trailConditionReport</b> variable.</li> <li>• Set the <b>Text</b> property of the <b>lblWeather</b> label to the <b>trailWeatherReport</b> variable.</li> </ul>

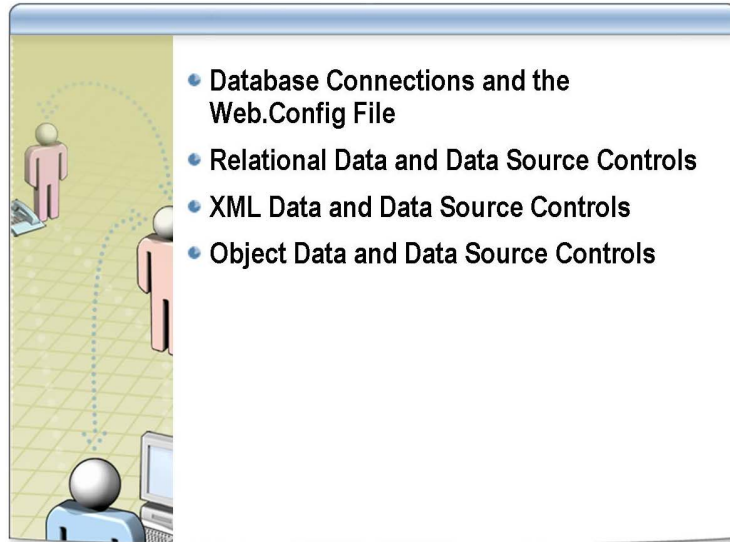
*(continued)*

Tasks	Supporting information
3. <i>(continued)</i>	<p>c. If <b>ChildNodes.Count</b> is <i>not</i> equal to zero, perform the following actions:</p> <ul style="list-style-type: none"> <li>• Invoke the <b>Expand()</b> method of the <b>twvTrails.SelectedNode</b> object.</li> <li>• Set the <b>Text</b> property of the <b>lblTrailName</b>, <b>lblCondition</b>, and <b>lblWeather</b> labels to an empty string.</li> </ul>
4. Add code for the <b>Page_Load</b> event of the <b>TrailReport.aspx</b> page.	<p>a. Locate the <b>Page_Load</b> event handler for the <b>TrailReport.aspx</b> page.</p> <p>b. In the <b>Page_Load</b> event handler method, add code to perform the following actions:</p> <ul style="list-style-type: none"> <li>• Declare a variable named <b>trailReport</b> of type <b>TrailReportWebService.Report</b> and set it to a new instance of the <b>TrailReportWebService.Report</b> class.</li> <li>• Declare a string variable named <b>trails</b> and set it to the return value of invoking the <b>GetTrails()</b> method of the <b>trailReport</b> object.</li> <li>• Set the <b>Data</b> property of the <b>xmlTrailsSource</b> control to the variable <b>trails</b>.</li> </ul>
5. Test the XML data handling of the Web application.	<p>a. In <b>Solution Explorer</b>, expand the <b>TrailReportWebService</b> project.</p> <p>b. Right-click <b>Report.asmx</b>, and then click <b>View in Browser</b> on the shortcut menu. The Web service starts and Internet Explorer displays the default Web service information page.</p> <p>c. Close Internet Explorer. The Web service continues to run in the background.</p> <p>d. Run the Web application.</p> <p>e. Click the <b>Trail Reports</b> menu item. The tree-view is populated with hierarchical data retrieved from the Web service as XML data.</p> <p>f. Expand the <b>Southern Europe</b> node, and then expand the <b>Italy</b> node.</p> <p>g. Click <b>Giro Sicily</b>, and notice that weather and condition reports are returned from the Web service and displayed on the page.</p> <p>h. Expand and click other nodes to view reports for other trails.</p> <p>i. Close Internet Explorer.</p>

## Lab Shutdown

After you complete the lab, you must shut down the **2543B-LON-DEV-01-06** virtual machine and discard any changes.

## Lab Discussion



In the lab, you:

- Created and retrieved database connections from the Web.Config file.
- Accessed relational data by using the **SqlDataSource** control and data controls.
- Accessed XML data by using the **XmlDataSource** controls.
- Accessed objects as data by using the **ObjectDataSource** control.
- Reviewed code that directly accessed ADO.NET 2.0 objects.

The instructor will lead a group discussion of these tasks. You should be prepared to contribute to the discussion, especially if you encountered problems completing the exercises.

