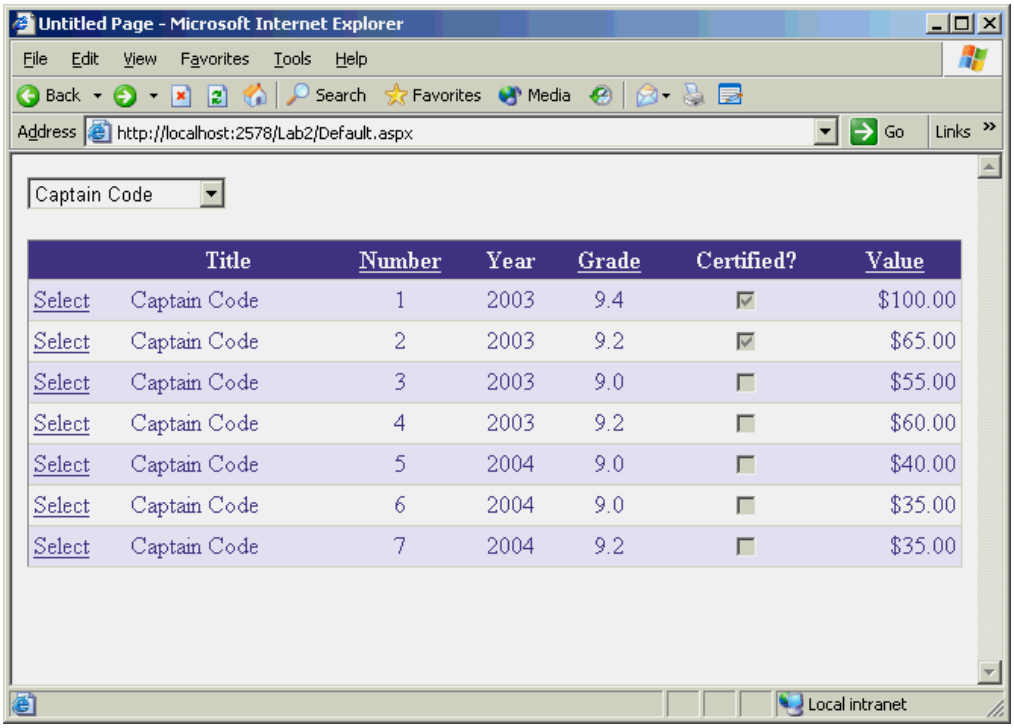
Lab 2: ASP.NET 2.0 Data Access

Estimated time to complete this lab: 60 minutes

Access to back-end databases and other data stores is an important element of data-driven Web applications. ASP.NET 2.0 makes building data-driven pages easier than ever before by providing data controls such as GridView and DetailsView to render data into HTML, and data source controls such as SqlDataSource and ObjectDataSource to declaratively bind data controls to data sources. Thanks to these controls, data access chores that required hundreds of lines of code in ASP.NET can often be accomplished with little or no code in ASP.NET 2.0.

In this lab, you'll begin building a Web site named MyComics that serves as a virtual catalog for comic books. You'll create a Web site to front the MyComics database and build a page that permits users to browse through the comic books in a GridView. Next, you'll add a page that shows comic book details in a DetailsView. Finally, you'll add an admin page that permits comic books to be inserted, updated, and deleted. In addition to learning how to create a Web site with Visual Studio and use the IDE to add pages, controls, and data components, you'll get an in-depth look at SqlDataSource and ObjectDataSource controls and gain first-hand experience with the editing capabilities of GridView and DetailsView controls.

Once the pages are up and running, you'll enable caching in the data source controls to maximize performance by minimizing database I/O. And to top things off, you'll use SQL cache dependencies to ensure that you're always serving up fresh data to your users—even if the underlying data changes.
Here's how the application will look in Internet Explorer at the conclusion of this lab:



Exercise 1
Verify the MyComics Database

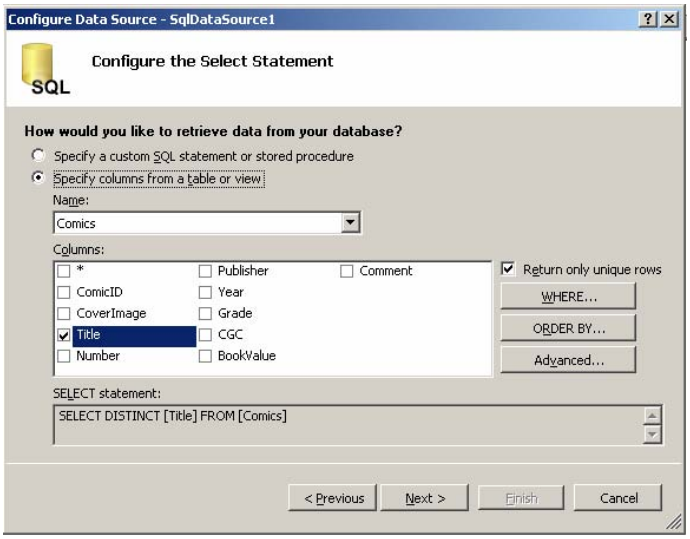
In this exercise, you'll verify a SQL Server database named MyComics exists to serve as the application's primary data store.

Tasks	Detailed Steps
1. Connect to the database	**a.** Start Visual Studio 2005. **b.** In the Server Explorer window right click on **Data Connections** and select **Add Connection** . **c.** Select Microsoft SQL Server and press Next. **d.** Set the Connection Properties as follows: Server Name: localhost Use Windows Authentication Select or enter a database name: MyComics **e.** Press test ! *If unable to connect and test; run the script C:\HOL\Web\LabFiles\Database\MyComics.cmd and try again.*

Exercise 2

Use SqlDataSource to populate a DropDownList

In this exercise, you'll create a new ASP.NET Web site with Visual Studio. Then you'll add a DropDownList control to the page and use a SqlDataSource to populate it with the results of a database query.

Tasks	Detailed Steps
f. Create a new Web site	**a.** Start Microsoft Visual Studio. ▪ Select "New Web Site" from Visual Studio's File menu. ▪ In the New Web Site dialog, choose "Visual C#" or "Visual Basic" as the project type and "ASP.NET Web Site" as the template type. Browse to or Type in "C:\HOL\Web\Starter\<language>\Lab2" into the Location box and click OK to create the Web site.
2. Add a SqlDataSource	**a.** Click the Design button to switch to Design view. **b.** Drag a SqlDataSource control onto the page. **c.** Click "Configure Data Source" in the "SqlDataSource Tasks" menu (see below).  **d.** When the Configure Data Source dialog pops up, click the New button next to "Choose a data connection." **e.** In the Add Connection dialog make sure the Data source is Microsoft SQL Server (SqlClient), then type "localhost" into the "Server name" box and select "Use Windows Authentication." Then select "MyComics" under "Select or enter a database name" and click OK. **f.** Make sure the new connection is selected in the "Choose a data connection" box. Then click Next. **g.** When asked if the connection string should be saved in the application configuration file, answer yes and enter "MyComicsConnectionString" as the connection string name. Then click Next to proceed. **h.** In the ensuing dialog, check "Title" in the Columns box. Also check "Return only unique rows," as shown below. Then click Next to proceed. 

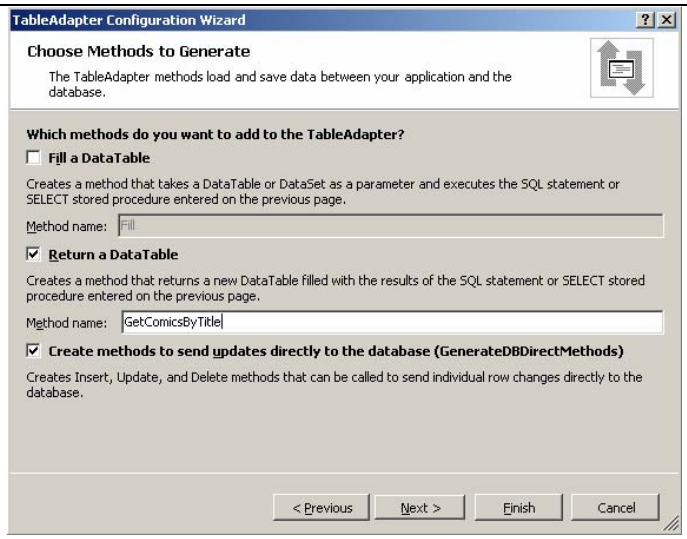
	i. Click Finish to finish configuring the SqlDataSource. **j.** Look in Visual Studio's Solution Explorer window and observe that a Web.config file has been added to the Web site. Double-click Web.config to open it. What do you see in the <connectionStrings> section? **k.** Close Web.config. **l.** Click the Source button to see a Source view of Default.aspx. Find the <asp:SqlDataSource> tag and look at its ConnectionString attribute. What do you see on the right side of the equals sign?
3. Add a DropDownList	**a.** Switch back to Design view and drag a DropDownList control onto the page. **b.** Click "Choose Data Source" in the "DropDownList Tasks" menu. **c.** Select SqlDataSource1 from the list of data sources and click OK. **d.** Check the "Enable AutoPostBack" box in the "DropDownList Tasks" menu. **e.** Select "Start Without Debugging" from Visual Studio's Debug menu (or press Ctrl+F5) to run Default.aspx in your browser. Verify that the page contains a drop-down list containing the items shown below.  **f.** Close the browser and return to Visual Studio.

Exercise 3

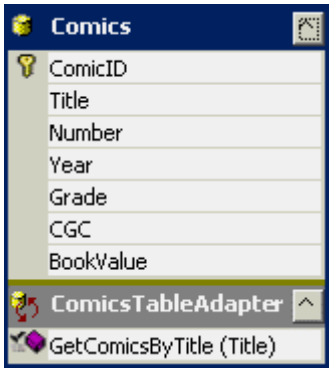
Use ObjectDataSource to Populate a GridView

In this exercise, you'll add a GridView control to the page. Rather than use a SqlDataSource to populate the GridView, you'll create a data component that interacts with the database and bind it to the GridView using an ObjectDataSource. In addition, you'll parameterize the ObjectDataSource so that it passes the title selected in the DropDownList to the data component for use in a WHERE clause.

Tasks	Detailed Steps
1. Add a data component	**a.** Add a folder named "App_Code" to the Web site by right-clicking C:\..\Lab2 in Solution Explorer and selecting **"Add Folder ->App_Code Folder."** **b.** Right-click the App_Code folder and select "Add New Item." **c.** In the ensuing dialog, select "DataSet" as the template type and enter MyComics.xsd as the file name. Then click the Add button. After a brief pause, the TableAdapter Configuration Wizard appears. **d.** If MyComics appears in the list of existing connections, select it. Otherwise, click the New Connection button and create a new MyComics connection, and then select it under "Select or enter a database name." Then click Next. **e.** On the "Choose a Command Type" page, select "Use SQL statements." Then click Next. **f.** On the "Generate the SQL Statements" page, type **"SELECT ComicID, Title, Number, Year, Grade, CGC, BookValue FROM Comics WHERE Title=@Title"** into the box labeled "What data should be loaded into the table?" Do NOT click the Next button just yet. **g.** Click the Advanced Options button. In the ensuing dialog, check the "Generate Insert, Update, and Delete statements" box and verify the other two boxes are unchecked, as shown below. Then click OK followed by Next.  **h.** Fill in the "Choose Methods to Generate" page as shown below so the wizard will include in the data component a GetComicsByTitle method (which uses the query you entered earlier to populate a DataTable) as well as methods for performing INSERTs, UPDATEs, and DELETEs.



i. Click Next, followed by Finish. The TableAdapter Configuration Wizard writes the configuration options you selected to MyComics.xsd. MyComics.xsd contains an XML schema describing a data component. At run-time, ASP.NET autocompiles strongly typed data components from XSD files. You don't see the components since they're not part of the project per se, but you do see a diagram of what's in the XSD file in the Visual Studio designer, as pictured below.



j. Choose "Save All" from Visual Studio's File menu to save changes.

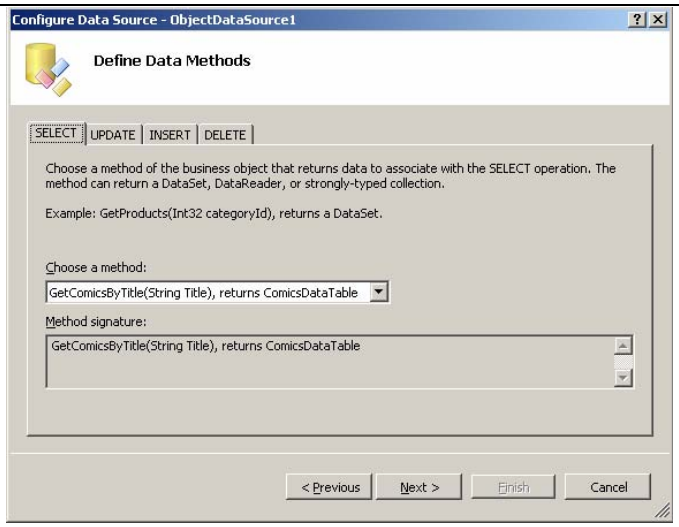
2. Add an ObjectDataSource

a. Return to Default.aspx in the designer and drag an ObjectDataSource control onto the page.

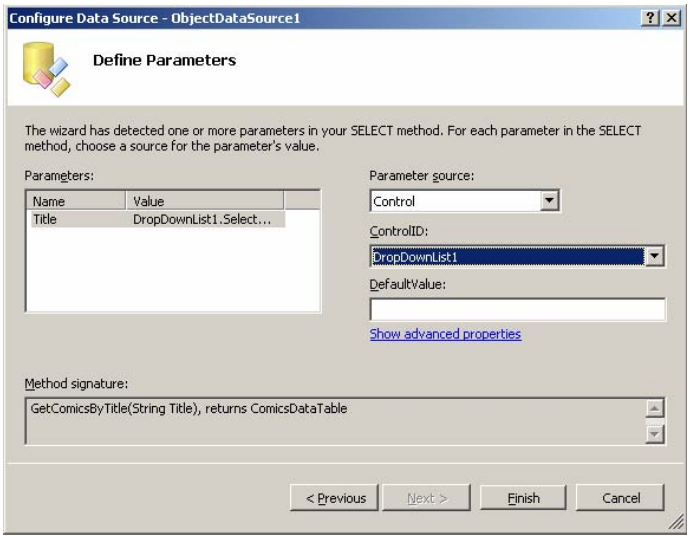
b. Click "Configure Data Source" in the "ObjectDataSource Tasks" menu.

c. Under "Enter the name of your business object," select MyComicsTableAdapters.ComicsTableAdapter (the name of the data component that's autocompiled from the XSD file you generated in the previous exercise). Then click Next.

d. Make sure GetComicsByTitle is selected in the "Choose a method" list, as shown below. One by one, click the UPDATE, INSERT, and DELETE tabs and select "None" from their "Choose a method" lists. This will prevent Visual Studio from configuring the ObjectDataSource to support INSERTs, UPDATEs, and DELETEs. When you're done, click Next.



e. Next you're asked to specify a source for the GetComicsByTitle method's Title parameter. Select Control in the "Parameter source" drop-down and DropDownList1 in the ControlID drop-down, as shown below. Then click Finish.



f. Switch to Source view and examine the <asp:ObjectDataSource> element. What do you see there that creates a connection between the ObjectDataSource and the DropDownList?

3. Add a GridView

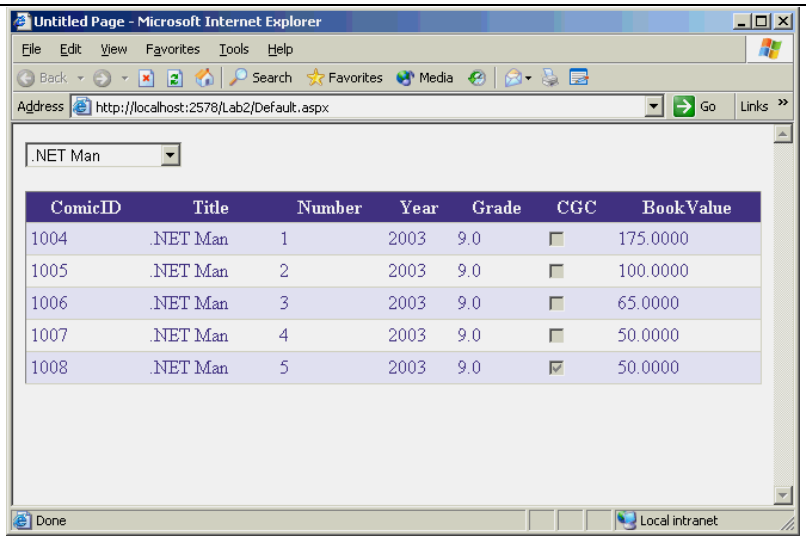
a. Switch back to Design view and drag a GridView control onto Default.aspx. Insert a couple of line breaks to create some space between the GridView and the DropDownList.

b. Use the "GridView Tasks" menu to select ObjectDataSource1 as the GridView's data source.

c. Use the "Auto Format" command in the "Common GridView Tasks" menu to apply the "Slate" format.

d. Select the GridView control in the designer. Then go to the Properties window and set the GridView's Width property to 100%.

e. Press Ctrl+F5 to run Default.aspx. Verify that the output resembles that below, and that selecting a different title in the DropDownList changes the list of comics in the GridView.

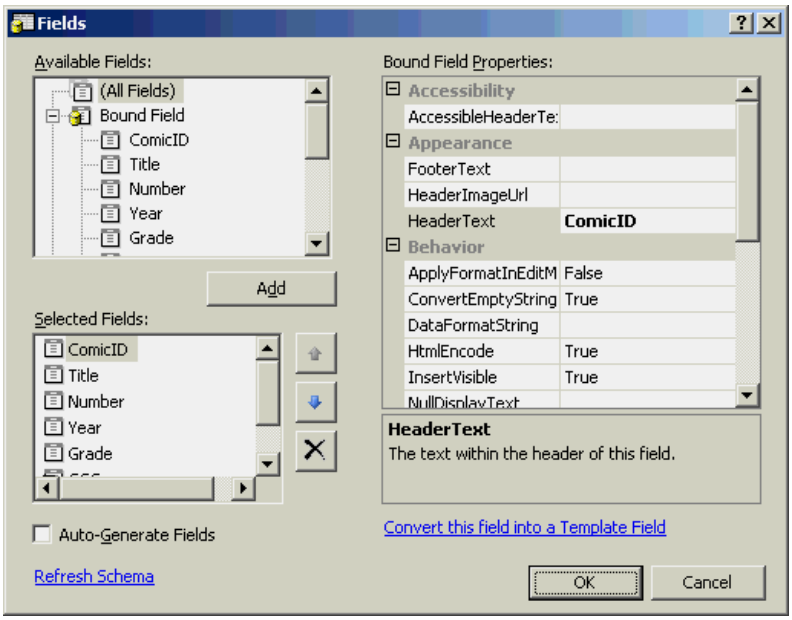


f. Close the browser and return to Visual Studio.

4. Polish the GridView's appearance

a. Open Default.aspx in Design view.

b. Click the arrow in the upper-right corner of the GridView to display the "GridView Tasks" menu, and click "Edit Columns" to display the Fields dialog pictured below.



c. Select "ComicID" in the "Selected Fields" box. In the property grid on the right, set ComicID's HeaderText property to "Comic ID" and its Visible property to false.

d. Select the "Title" field. Set its SortExpression property to an empty string.

e. Select the "Number" field. Set its ItemStyle-HorizontalAlign property to "Center".

f. Select the "Year" field. Set its ItemStyle-HorizontalAlign property to "Center" and its SortExpression property to an empty string.

g. Select the "Grade" field. Set its ItemStyle-HorizontalAlign property to "Center" and its DataFormatString property to "{0:f1}".

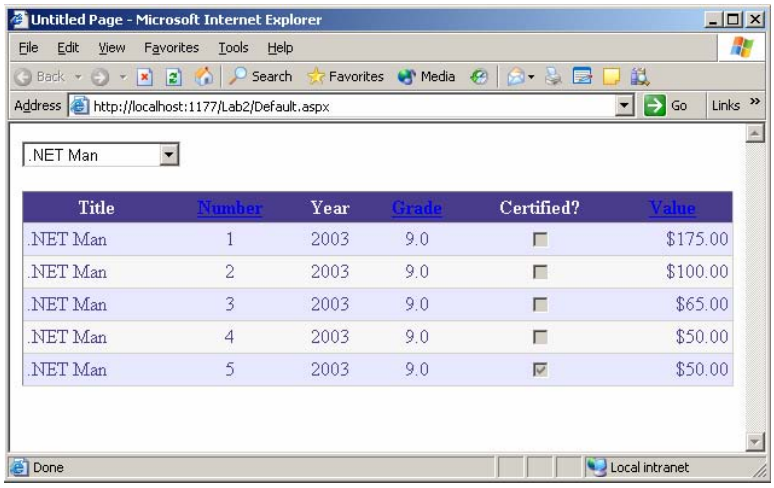
h. Select the "CGC" field. Set its HeaderText property to "Certified?", its ReadOnly property to true, its ItemStyle-HorizontalAlign property to "Center", and its SortExpression property to an empty string.

i. Select the "BookValue" field. Set its HeaderText property to "Value", its DataFormatString property to "{0:c}", and its ItemStyle-HorizontalAlign property to Right.

j. Click OK to commit the changes and dismiss the Fields dialog.

k. In the designer, display the GridView's "GridView Tasks" menu again if it isn't already displayed. Check the menu's "Enable Sorting" box.

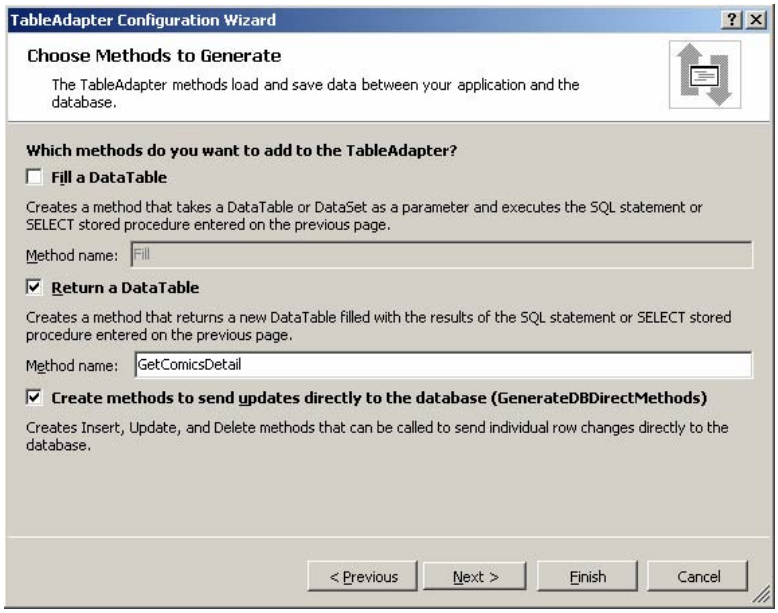
l. Press Ctrl+F5 to Launch Default.aspx in your browser. Verify that the page resembles the one below. Also verify that you can sort by number, grade, and value, and that clicking the same sortable column header several times in a row performs alternating ascending and descending sorts.

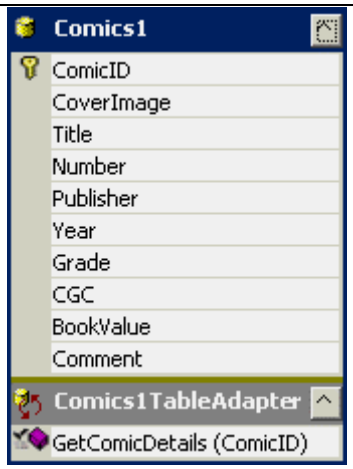


m. Close the browser and return to Visual Studio.

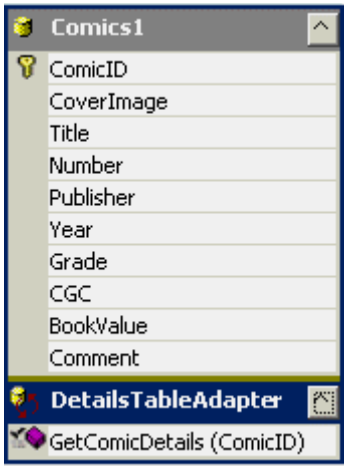
Exercise 4
Create a details page

In this exercise, you'll add a page named Details.aspx to the Web site and embellish it with a DetailsView control. Then you'll connect the two pages so that clicking an item in Default.aspx displays details about that item in Details.aspx.?

Tasks	Detailed Steps
1. Add another page to the Web site	**a.** Right-click C:\..\Lab2 in Solution Explorer and select "Add New Item." **b.** Select "Web Form" and name it Details.aspx. Be sure the "Place code in separate file" box is checked and that C# or Visual Basic is selected in the Language box before clicking the Add button.
2. Add another data component	**a.** Double-click MyComics.xsd in the Solution Explorer window to open it for editing. **b.** Right-click the design surface and select "Add->TableAdapter" to display the TableAdapter Configuration Wizard . **c.** Select "MyComicsConnectionString (Web.config)" in the data connections list and click Next. **d.** On the "Choose a Command Type" page, select "Use SQL statements." Then click Next. **e.** On the "Generate the SQL statements" page, type "SELECT * FROM Comics WHERE ComicID=@ComicID". Then click the Advanced Options button and verify the "Use optimistic concurrency" and "Refresh the data component" boxes are unchecked. Click OK, followed by Next. **f.** Fill in the "Choose Methods to Generate" page as shown below. Note that the name typed into the "Method Name" box is GetComicDetails, not GetComicsByTitle.  **g.** Click Next, followed by Finish. Confirm that the new data component looks like this:



h. Right-click "Comics1TableAdapter" and use the Rename command to change the name to "DetailsTableAdapter":



i. Choose "Save All" from Visual Studio's File menu to save changes.

3. Add an ObjectDataSource

a. Return to the designer and open Details.aspx in Design view.

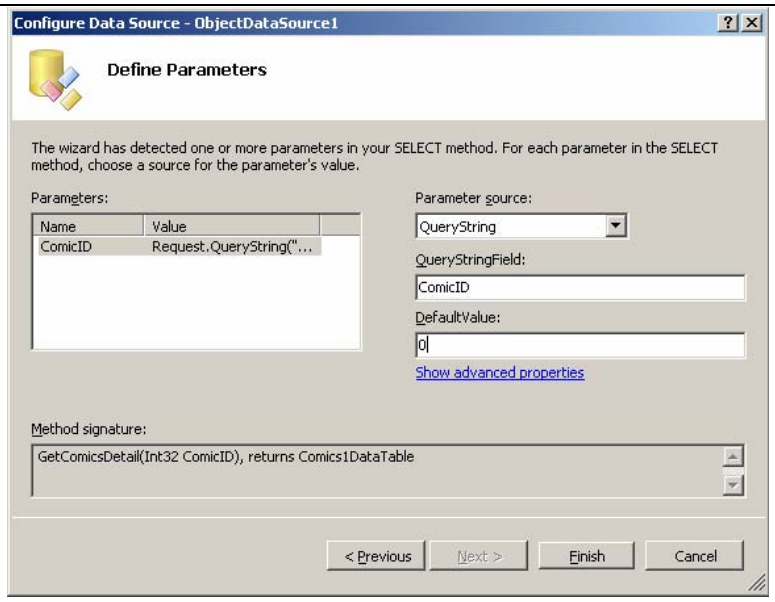
b. Drag an ObjectDataSource control from the Toolbox and drop it onto the page.

c. Click "Configure Data Source" in the "ObjectDataSource Tasks" menu.

d. Under "Enter the name of your business object," select MyComicsTableAdapters.DetailsTableAdapter. Then click Next.

e. Make sure GetComicDetails in selected in the "Choose a method" list on the SELECT page. Select "None" in the "Choose a Method" list on the UPDATE, INSERT, and DELETE pages. Then click Next.

f. Next you're asked to specify a source for GetComicDetails's ComicID parameter. Select QueryString in the "Parameter source" drop-down, type "ComicID" into the QueryStringField box, and type "0" into the DefaultValue box as shown below. Then click Finish.



g. Switch to Source view and inspect the <asp:ObjectDataSource> element. What does the <SelectParameters> element inside it do?

4. Add Select buttons to the GridView

a. Open Default.aspx in Design view.

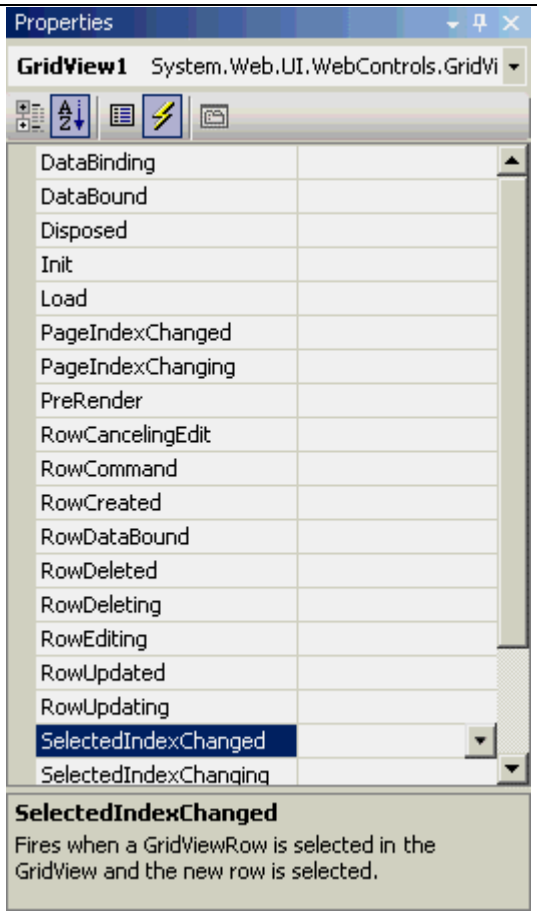
b. Select the GridView control and set its AutoGenerateSelectButton property to true.

c. Press Ctrl+F5 to launch Default.aspx in your browser. What's different about the GridView?

d. Close the browser and return to Visual Studio.

e. Select the GridView control.

f. Click the lightning-bolt icon in the Properties window to display a list of GridView events.



g. Double-click "SelectedIndexChanged" to add a SelectedIndexChanged event handler to Default.aspx.cs or Default.aspx.vb.

h. Add the following code to the body of the handler:

C#

```
Response.Redirect ("Details.aspx?ComicID=" +
GridView1.SelectedValue);
```

VB

```
Response.Redirect("Details.aspx?ComicID=" & _
GridView1.SelectedValue.ToString)
```

i. Return to Default.aspx. In the Properties window, click the button to the left of the lightning bolt to display properties instead of events.

j. Select the GridView control and verify its DataKeyNames property is set to "ComicID" so GridView1.SelectedValue will return the value of the ComicID field in the row that's currently selected.

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

l. Click one of the GridView's Select buttons. What happens? What do you see in the browser's address bar?

m. Close the browser and return to Visual Studio.

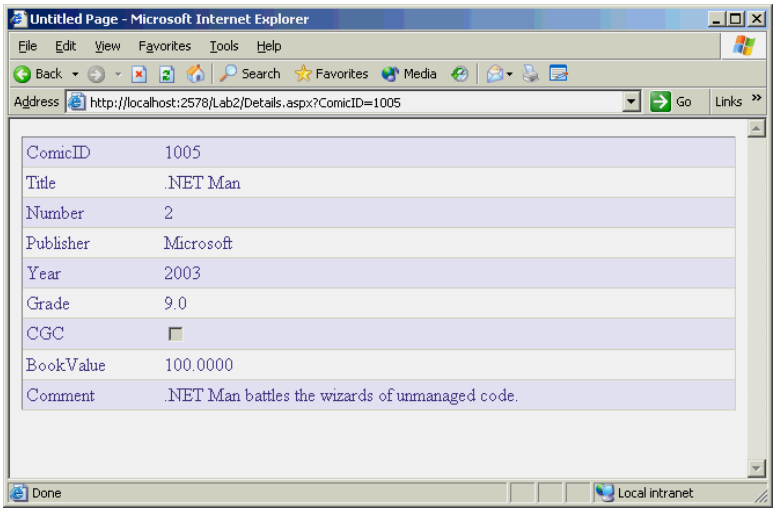
5. Add a DetailsView control

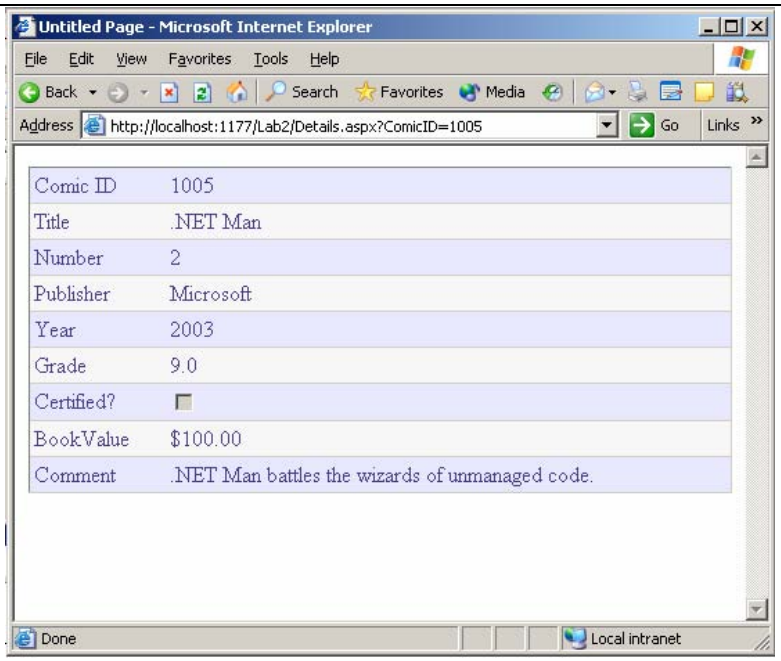
a. Open Details.aspx in Design view.

b. Drag a DetailsView control from the Toolbox and drop it onto the page.

c. Use the "DetailsView Tasks" menu to select ObjectDataSource1 as the DetailsView's data source.

d. Use the "Auto Format" command in the "DetailsView Tasks" menu to apply the format named "Slate."

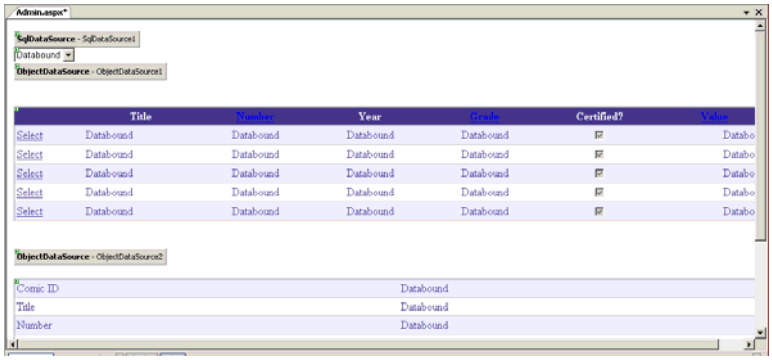
	e. Select the DetailsView control in the designer. Then go to the Properties window and set the DetailsView's Width property to 100%. **f.** Select Default.aspx in the Solution Explorer window. Then press Ctrl+F5 to launch it. **g.** Click one of the GridView's Select buttons and verify that Details.aspx appears showing details about the selected comic, as pictured below.  **h.** Close the browser and return to Visual Studio.
6. Polish the DetailsView's appearance	**a.** Open Details.aspx in Design view. **b.** Click the arrow in the upper-right corner of the DetailsView to display the "DetailsView Tasks" menu, and click "Edit Fields" to display the Fields dialog. **c.** Select "ComicID" in the "Selected Fields" box. Set its HeaderText property to "Comic ID". **d.** Select "Grade" in the "Selected Fields" box. Set its DataFormatString property to "{0:f1}". **e.** Select "CGC" in the "Selected Fields" box. Sets its HeaderText property to "Certified?" and its ReadOnly property to true. **f.** Select "BookValue" in the "Selected Fields" box. Sets its HeaderText property to "Value" and its DataFormatString property to "{0:c}". **g.** Click OK to commit the changes and dismiss the Fields dialog. **h.** Select Default.aspx in the Solution Explorer window. Then press Ctrl+F5 to launch it. **i.** Click one of the GridView's Select buttons and verify that the resulting DetailsView resembles the one below.



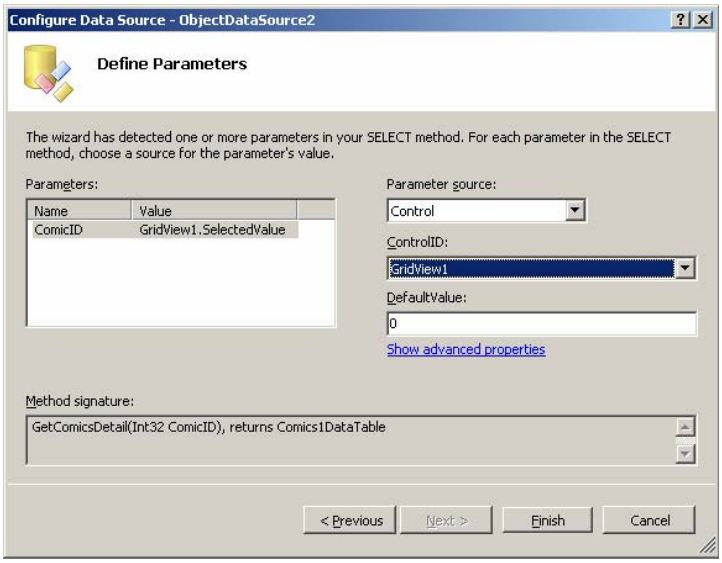
j. Close the browser and return to Visual Studio.

Exercise 5
Create an admin page

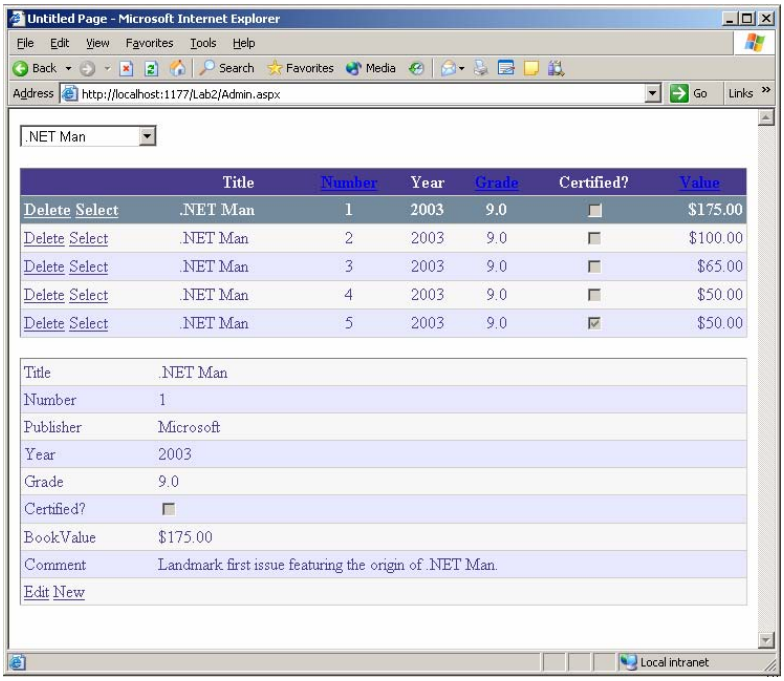
In this exercise, you'll add a page named Admin.aspx to the Web site and populate it with an editable master-detail view for browsing, inserting, updating, and deleting records. You'll use the data components you built in previous exercises to interact with the database, and you'll use the editing features of the GridView and DetailsView controls and the 2-way data-binding support in ObjectDataSource to make short work of updating the database.

Tasks	Detailed Steps
1. Add another page to the Web site	**a.** Right-click C:\..\Lab2 in Solution Explorer and select "Add New Item." **b.** Select "Web Form" and name it Admin.aspx. Be sure the "Place code in separate file" box is checked and that C# or Visual Basic is selected in the Language box before clicking the Add button.
2. Copy controls to the page	**a.** Open Default.aspx in Source view and copy everything between the <form id="form1" runat="server"> tag and the </form> tag to the clipboard. **b.** Open Admin.aspx in Source view and paste the contents of the clipboard between the <form id="form1" runat="server"> and </form> tags. **c.** Open Details.aspx in Source view and copy everything between the <form id="form1" runat="server"> and </form> tags to the clipboard. **d.** Go back to Admin.aspx in Source view and paste the contents of the clipboard immediately below the content you pasted in step 2. **e.** For C# clear the OnSelectedIndexChanged="GridView1_SelectedIndexChanged" portion of the <asp:GridView> tag. **f.** Click the Design button to show Admin.aspx in Design view. Insert a blank line between the GridView and the DetailsView to create some space between them. Here's what Admin.aspx should look like in the designer:  **g.** Use the DetailsView's "DetailsView Tasks" menu to set the DetailsView's data source to ObjectDataSource2. If Visual Studio asks if you'd like the DetailsView's field and keys refreshed, answer no. **h.** Select the GridView control. Then go to the properties window and set the GridView's AutoGenerateDeleteButton property to true. **i.** Select the DetailsView control. Go to the properties window and set the DetailsView's AutoGenerateEditButton and AutoGenerateInsertButton properties to true. **j.** Verify the DetailsView's DataKeyNames property is set to "ComicID". **k.** Display the DetailsView's "DetailsView Tasks" menu and click "Edit Fields." In the Fields dialog, set the "ComicID" field's Visible property to false and the "CGC" field's ReadOnly property to false and Press OK. **l.** Select ObjectDataSource2 and display its "ObjectDataSource Tasks" menu. Click "Configure Data Source" and then click Next until you reach the "Define parameters" page. ObjectDataSource2 is currently configured to acquire the

ComicID parameter that it uses in database queries from a query string. Fill in the dialog as shown below to reconfigure ObjectDataSource2 to acquire the parameter from the GridView. Then click the Finish button. If Visual Studio offers to refresh the DetailsView control, answer no.



m. Press Ctrl+F5 to launch Admin.aspx in your browser. Confirm that clicking one of the GridView's Select buttons displays details about the corresponding comic in a DetailsView, as shown below.

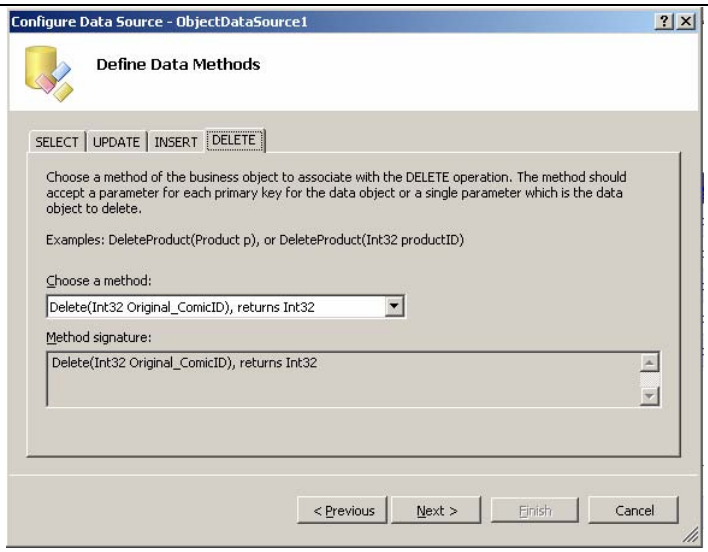


n. Close your browser and return to Visual Studio.

3. Configure the ObjectDataSource controls to support 2-way data binding

a. Return to Admin.aspx in the designer. Select the "Configure Data Source" command from ObjectDataSource1's "ObjectDataSource Tasks" menu. Click Next until you reach the "Define data methods" page.

b. Set ObjectDataSource1's SELECT and DELETE methods to GetComicsByTitle and Delete, respectively. (The DELETE setting is shown below.) Set its UPDATE and INSERT methods to "None" since this ObjectDataSource won't be used to insert or update records. Then click Next, followed by Finish. If Visual Studio offers to refresh the GridView, answer no.



c. Select the "Configure Data Source" command from ObjectDataSource2's "Common ObjectDataSource Tasks" menu. Click Next until you reach the "Define data methods" page.

d. Set the ObjectDataSource2's SELECT, UPDATE, and INSERT methods to GetComicDetails, Update, and Insert, respectively. Set its DELETE method to "None" since this ObjectDataSource won't be used to delete records. Then click Next, followed by Finish. If Visual Studio offers to refresh the DetailsView, answer no.

e. Select ObjectDataSource1 in the designer and click the lightning-bolt icon in the Properties window to display a list of ObjectDataSource events.

f. Double-click "Deleted" in the Properties window to add a handler for ObjectDataSource.Deleted events.

g. Add the following statement to the body of the handler to ensure that the drop-down list of titles is updated if the last title in a category is deleted:

C#

```
DropDownList1.DataBind ();
```

VB

```
DropDownList1.DataBind()
```

h. Return to Admin.aspx and select ObjectDataSource2 in the designer.

i. Go to the Properties window (which currently lists ObjectDataSource events) and double-click "Inserted" to add a handler for ObjectDataSource.Inserted events.

j. Add the following statement to the body of the handler to ensure that the GridView is updated when a comic is added to the database:

C#

```
GridView1.DataBind ();
```

VB

```
GridView1.DataBind()
```

k. Return to Admin.aspx and select ObjectDataSource2 in the designer.

l. Go to the Properties window (which currently lists ObjectDataSource events) and double-click "Updated" to add a handler for ObjectDataSource.Updated events.

m. Add the following statement to the body of the handler to ensure that the GridView is updated when a record is updated:

C#

```
GridView1.DataBind ();
```

VB

```
GridView1.DataBind()
```

4. Test the admin page

a. Press Ctrl+F5 to launch Admin.aspx in your browser.

b. Test the page's record insertion capabilities by selecting one of the Captain Code comics in the GridView and clicking the New button in the DetailsView. Fill in the fields as follows:

Name	Value
Title	Captain Code
Number	8
Publisher	Microsoft
Year	2004
Grade	9.0
Certified?	Unchecked
Value	30
Comment	Introducing CodeGirl and her sidekick, CodeKid

c. Click "Insert" at the bottom of the DetailsView to insert the comic into the database. Verify that Captain Code 8 now appears in the GridView.

d. Test the page's record updating capabilities by selecting Captain Code 8 in the GridView and clicking "Edit" in the DetailsView. Change the grade in the Grade field from 9.0 to 7.0. Then click "Update." Verify that the grade changes to 7.0 in the GridView.

e. Test the page's record deletion capabilities by clicking the Delete button next to Captain Code 8 in the GridView. Verify that the record disappears from view.

f. Close your browser and return to Visual Studio.

Exercise 6
Add caching and cache dependencies

Database queries are expensive—especially if they return images or other large objects. Performance problems are compounded if queries target remote database servers. The solution to the problem of performance-inhibiting database queries is caching, which ASP.NET 2.0 provides robust support for. In this exercise, you'll enable caching in Default.aspx and Details.aspx's data source controls to minimize redundant database accesses. Then you'll use SQL cache dependencies to prevent caching from causing users to see outdated data.

Tasks	Detailed Steps
a. Enable caching in the data sources	**a.** Open Default.aspx in Design view. Select SqlDataSource1 and use the Properties window to set its EnableCaching property to true and its CacheDuration property to 300 (that's 300 seconds, or 5 minutes). **b.** Set ObjectDataSource1's EnableCaching property to true and its CacheDuration property to 300. **c.** Open Details.aspx in Design view. Set ObjectDataSource1's EnableCaching property to true and its CacheDuration property to 300. **d.** Run Default.aspx and make sure it works as before. Leave Default.aspx running in the browser with .NET Man selected in the DropDownList. **e.** Use Server Explorer to change the grade assigned to .NET Man 1 from 9.0 to 2.0. (You can open a table for editing in Server Explorer by drilling down into the MyComics connection to the Comics Table, right click and select "Show Table Data.") **f.** Refresh Default.aspx in the browser. What grade is shown for .NET Man 1? Is it 2.0 or 9.0? Why? **g.** Close the browser and return to Visual Studio.

5. Configure the database for SQL cache dependencies	**a.** Open a Visual Studio command prompt window. You'll find it under "All Programs->Microsoft Visual Studio 2005 ->Visual Studio Tools->Visual Studio Command Prompt."
	b. Type aspnet_regsql -S localhost -E -d MyComics –ed to configure the MyComics database to support SQL cache dependencies.
	c. Type aspnet_regsql -S localhost -E -d MyComics -t Comics –et to configure the Comics table to support SQL cache dependencies.
	d. Go to Server Explorer and refresh the list of tables in the MyComics database. A new table has appeared. What is its name?
	e. Use Server Explorer to view the triggers attached to the Comics table. (You can view a table's triggers by drilling into the table and viewing the objects with a lightning bolt next to them) There should be a trigger named Comics_AspNet_SqlCacheNotification_Trigger. What does the trigger do?
	f. Open Web.config and add the following statements to the <system.web> section:
	`<caching>`
	`<sqlCacheDependency enabled="true" pollTime="5000">`
	`<databases>`
	`<add name="MyComics"`
	`connectionStringName="MyComicsConnectionString" />`
	`</databases>`
	`</sqlCacheDependency>`
	`</caching>`
	g. Close Web.config and return to the designer.

6. Add SQL cache dependencies to the data sources	**a.** In Default.aspx, set SqlDataSource1's SqlCacheDependency property to "MyComics:Comics". **b.** In Default.aspx, set ObjectDataSource1's SqlCacheDependency property to

	"MyComics:Comics". **c.** In Details.aspx, set ObjectDataSource1's SqlCacheDependency property to "MyComics:Comics". **d.** Run the application and make sure Default.aspx and Details.aspx work as they did before. Leave Default.aspx running with .NET Man selected in the DropDownList. **e.** Use Server Explorer to change .NET Man 1's grade from 2.0 to 6.0. **f.** Wait a few seconds, and then refresh Default.aspx in the browser. What grade is shown for .NET Man 1? Why?
7. Monitor database activity on the back end (optional)	**a.** Start the SQL Server Profiler (All Programs->Microsoft SQL Server->Profiler). **b.** Use Profiler's File->New->Trace command to start a new trace. (Click OK in the first dialog and Run in the second one to start the trace running.) **c.** Launch Default.aspx in your browser and refresh the page several times, watching the trace window in Profiler as you do. Is the database queried anew each time the page refreshes?

Summary

ASP.NET 1.x simplified the building of data-driven Web pages by introducing data binding controls such as Repeaters, DataLists, and DataGrids. ASP.NET 2.0 further simplifies the data access story with the introduction of data source controls for declaratively querying and updating data stores and caching query results, data controls for rendering data into HTML, and SQL cache dependencies for refreshing cached query results.

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

- How to use SqlDataSource controls to bind to databases
- How to use ObjectDataSource controls to bind to data components
- How to create and configure data components
- How to parameterize data sources using control values
- How to parameterize data sources using query strings
- How to use GridView and DetailsView controls
- How to customize the columns in a GridView control
- How to customize the rows in a DetailsView control
- How to display images in DetailsView controls using ImageFields
- How to update databases with ObjectDataSources, GridViews, and DetailsViews
- How to configure data source controls to cache query results
- How to use SQL cache dependencies to refresh cached query results

Take a moment to review the application in its current form. As you do, here are some questions to ponder:

- *How many lines of code did you write?*

- *What will happen if an insert, update, or delete performed through the admin page throws an exception? How would you go about handling those exceptions and making the application more robust?*

In the final exercise, SQL Server Profiler showed a stored procedure being executed on the MyComics database approximately every 5 seconds. Who was calling that stored procedure, and why was it being called? Would changing <sqlCacheDepedency>'s pollTime attribute have any effect on the timing?