Working with Data in ASP.NET 2.0 :: Querying Data with the SqlDataSource Control

Introduction

All of the tutorials we've examined so far have used a tiered architecture consisting of presentation, Business Logic, and Data Access layers. The Data Access Layer (DAL) was crafted in the first tutorial (<u>Creating a Data Access Layer</u>) and the Business Logic Layer in the second (<u>Creating a Business Logic Layer</u>). Starting with the <u>Displaying Data With the ObjectDataSource</u> tutorial, we saw how to use ASP.NET 2.0's new ObjectDataSource control to declaratively interface with the architecture from the presentation layer.

While all of the tutorials so far have used the architecture to work with data, it is also possible to access, insert, update, and delete database data directly from an ASP.NET page, bypassing the architecture. Doing so places the specific database queries and business logic directly in the web page. For sufficiently large or complex applications, designing, implementing, and using a tiered architecture is vitally important for the success, updatability, and maintainability of the application. Developing a robust architecture, however, can be overkill when creating exceedingly simple, one-off applications.

ASP.NET 2.0 provides five built-in data source controls – <u>SqlDataSource</u>, <u>AccessDataSource</u>, <u>ObjectDataSource</u>, <u>XmlDataSource</u>, and <u>SiteMapDataSource</u>. The SqlDataSource can be used to access and modify data directly from a relational database, including Microsoft SQL Server, Microsoft Access, Oracle, MySQL, and others. In this tutorial and the next three, we'll examine how to work with the SqlDataSource control, exploring how to query and filter database data, as well as how to use the SqlDataSource to insert, update, and delete data.



Figure 1: ASP.NET 2.0 Includes Five Built-In Data Source Controls

Comparing the ObjectDataSource and SqlDataSource

Conceptually, both the ObjectDataSource and SqlDataSource controls are simply proxies to data. As discussed in the <u>Displaying Data With the ObjectDataSource</u> tutorial, the ObjectDataSource has properties that indicate the

object type that provides the data and the methods to invoke to select, insert, update, and delete data from the underlying object type. Once the ObjectDataSource's properties have been configured, a data Web control – such as a GridView, DetailsView, or DataList – can be bound to the control, using the ObjectDataSource's Select(), Insert(), Delete(), and Update() methods to interact with the underlying architecture.

The SqlDataSource provides the same functionality, but operates against a relational database rather than an object library. With the SqlDataSource, we must specify the database connection string and the ad-hoc SQL queries or stored procedures to execute to insert, update, delete, and retrieve data. The SqlDataSource's Select(), Insert(), Update(), and Delete() methods, when invoked, connect to the specified database and issue the appropriate SQL query. As the following diagram illustrates, the these methods do the grunt work of connecting to a database, issuing a query, and returning the results.



Figure 2: The SqlDataSource Serves as a Proxy to the Database

Note: In this tutorial we'll focus on retrieving data from the database. In the <u>Inserting, Updating, and Deleteing</u> <u>Data with the SqlDataSource Control</u> tutorial, we'll see how to configure the SqlDataSource to support inserting, updating, and deleting.

The SqlDataSource and AccessDataSource Controls

In addition to the SqlDataSource control, ASP.NET 2.0 also includes an AccessDataSource control. These two different controls lead many developers new to ASP.NET 2.0 to suspect that the AccessDataSource control is designed to work exclusively with Microsoft Access with the SqlDataSource control designed to work exclusively with Microsoft SQL Server. While the AccessDataSource is designed to work specifically with Microsoft Access, the SqlDataSource control works with *any* relational database that can be accessed through .NET. This includes

any OleDb- or ODBC-compliant data stores, such as Microsoft SQL Server, Microsoft Access, Oracle, Informix, MySQL, and PostgreSQL, among many others.

The sole difference between the AccessDataSource and SqlDataSource controls is how the database connection information is specified. The AccessDataSource control needs just the file path to the Access database file. The SqlDataSource, on the other hand, requires a complete connection string.

Step 1: Creating the SqlDataSource Web Pages

Before we start exploring how to work directly with database data using the SqlDataSource control, let's first take a moment to create the ASP.NET pages in our website project that we'll need for this tutorial and the next three. Start by adding a new folder named SqlDataSource. Next, add the following ASP.NET pages to that folder, making sure to associate each page with the Site.master master page:

- Default.aspx
- Querying.aspx
- ParameterizedQueries.aspx
- InsertUpdateDelete.aspx
- OptimisticConcurrency.aspx



Figure 3: Add the ASP.NET Pages for the SqlDataSource-Related Tutorials

Like in the other folders, Default.aspx in the SqlDataSource folder will list the tutorials in its section. Recall that the SectionLevelTutorialListing.ascx User Control provides this functionality. Therefore, add this User Control to Default.aspx by dragging it from the Solution Explorer onto the page's Design view.



Figure 4: Add the SectionLevelTutorialListing.ascx User Control to Default.aspx

Lastly, add these four pages as entries to the Web.sitemap file. Specifically, add the following markup after the "Adding Custom Buttons to the DataList and Repeater" <siteMapNode>:

```
<siteMapNode url="~/SqlDataSource/Default.aspx"</pre>
    title="Using the SqlDataSource Control"
    description="Work directly with database data using the SqlDataSource control.">
    <siteMapNode url="~/SqlDataSource/Querying.aspx" title="Retrieving Database Data"</pre>
        description="Examines how to query data from a database that can then be
                     displayed through a data Web control."/>
    <siteMapNode url="~/SqlDataSource/ParameterizedQueries.aspx"</pre>
        title="Parameterized Queries"
        description="Learn how to specify parameterized WHERE clauses in the
                     SqlDataSource's SELECT statement." />
    <siteMapNode url="~/SqlDataSource/InsertUpdateDelete.aspx"</pre>
        title="Inserting, Updating, and Deleting Database Data"
        description="See how to configure the SqlDataSource to include INSERT, UPDATE,
                      and DELETE statements." />
    <siteMapNode url="~/SqlDataSource/OptimisticConcurrency.aspx"</pre>
        title="Using Optimistic Concurrency"
        description="Explore how to augment the SqlDataSource to include support for
                     optimistic concurrency." />
```

After updating Web.sitemap, take a moment to view the tutorials website through a browser. The menu on the left now includes items for the editing, inserting, and deleting tutorials.

🗿 Untitled Page - 🕖 🕀 💶 🔲	×
Eile Edit <u>V</u> iew Favorites *	Ē.
3 Back - 5 - 🖹 🖹 🏠	»
Address 🛃 http://localhost:4 😪 🛃 Go	ij
Using the SqlDataSource Control	^
Retrieving Database Data	
Parameterized Queries	
Inserting, Updating, and Deleting Database Data	
Using Optimistic Concurrency	*
< · · · · · · · · · · · · · · · · · · ·	
Second Second Second Second	1

Figure 5: The Site Map Now Includes Entries for the SqlDataSource Tutorials

Step 2: Adding and Configuring the SqlDataSource Control

Start by opening the Querying.aspx page in the SqlDataSource folder and switch to Design view. Drag a SqlDataSource control from the Toolbox onto the Designer and set its ID to ProductsDataSource. As with the ObjectDataSource, the SqlDataSource does not produce any rendered output and therefore appears as a gray box on the design surface. To configure the SqlDataSource, click on the Configure Data Source link from the SqlDataSource's smart tag.



Figure 6: Click on the Configure Data Source Link from the SqlDataSource's Smart Tag

This brings up the SqlDataSource control's Configure Data Source wizard. While the wizard's steps differ from the ObjectDataSource control's, the end goal is the same – to provide the details on how to retrieve, insert, update, and delete data through the data source. For the SqlDataSource this entails specifying the underlying database to use and providing the ad-hoc SQL statements or stored procedures.

The first wizard step prompts us for the database. The drop-down list includes those databases found in the web application's App_Data folder and those that have been added to the Data Connections node in the Server Explorer. Since we've already added a connection string for the NORTHWIND.MDF database in the App_Data folder to our project's Web.config file, the drop-down list includes a reference to that connection string, NORTHWINDConnectionString. Choose this item from the drop-down list and click Next.

Configure Data Source - SqlDataSource1		• • • ? 🛛
Choose Your Data Connection		
Which data connection should your application use NORTHWNDConnectionString	to connect to the data	base? New <u>C</u> onnection
Connection string		
Data Source=.\SQLEXPRESS;AttachDbFilename= Data Security=True;User Instance=True	Directory][NORTHWND.MD	F;Integrated
< Previous	Next >	hish Cancel

Figure 7: Choose the NORTHWINDConnectionString from the Drop-Down List

After choosing the database, the wizard asks for the query to return data. We can either specify the columns of a table or view to return or can enter a custom SQL statement or specify a stored procedure. You can toggle between this choice through the "Specify a custom SQL statement or stored procedure" and "Specify columns from a table or view" radio buttons.

Note: For this first example, let's use the "Specify columns from a table or view" option. We'll return to the wizard later in this tutorial and explore the "Specify a custom SQL statement or stored procedure" option.

Figure 8 shows the "Configure the Select Statement" screen when the "Specify columns from a table or view" radio button is selected. The drop-down list contains the set of tables and views in the Northwind database, with the selected table or view's columns displayed in the checkbox list below. For this example, let's return the ProductID, ProductName, and UnitPrice columns from the Products table. As Figure 8 shows, after making these selections the wizard shows the resulting SQL statement – SELECT [ProductID], [ProductName], [UnitPrice] FROM [Products].

Þ		
low would you like to re	trieve data from your database?	
Specify a custom SQL st Specify solutions from a	tatement or stored procedure	
 Speciry columns from a Name: 	Capie or view	
Products		
Columps:		
*		Return only unique rows
ProductID	UnitsInStock	WHERE
ProductName	UnitsOnOrder	
SupplierID	ReorderLevel Discontinued	OBDER BY
QuantityPerUnit	Discontinued	Advanced
SELECT statements		
SELECT [ProductID]. [ProductName], [UnitPrice] FROM [Product	cts]

Figure 8: Return Data from the Products Table

Once you have configured the wizard to return the ProductID, ProductName, and UnitPrice columns from the Products table, click the Next button. This final screen provides an opportunity to examine the results of the query configured from the previous step. Clicking the Test Query button executes the configured SELECT statement and displays the results in a grid.

onfigure D	ata Source - SqlDataSource	e 1	• • • ? 🛽
To preview I	Test Query	rce, click Test Query. To	complete this wizard, click Finish.
ProductID	ProductName	UnitPrice	<u>~</u>
1	Chai Tea	19.9500	
2	Chang	19.0000	
3	Aniseed Syrup	10.0000	
4	Chef Anton's Cajun Seasoning	26.6200	
5	Chef Anton's Gumbo Mix	21.3500	
SELECT stat	ement: oductID], [ProductName], [UnitPric	e] FROM [Products]	Test Query
		< Previous Nex	Einish Cancel

Figure 9: Click the Test Query Button to Review Your SELECT Query

To complete the wizard, click Finish.

Like with the ObjectDataSource, the SqlDataSource's wizard merely assigns values to the control's properties, namely the <u>ConnectionString</u> and <u>SelectCommand</u> properties. After completing the wizard, your SqlDataSource control's declarative markup should look similar to the following:

```
<asp:SqlDataSource ID="ProductsDataSource" runat="server"
ConnectionString="<%$ ConnectionStrings:NORTHWNDConnectionString %>"
SelectCommand="SELECT [ProductID], [ProductName], [UnitPrice] FROM [Products]">
</asp:SqlDataSource>
```

The ConnectionString property provides information on how to connect to the database. This property can be assigned a complete, hard-coded connection string value or can point to a connection string in Web.config. To reference a connection string value in Web.config, use the syntax <%\$ expressionPrefix:expressionValue %>. Typically, expressionPrefix is "ConnectionStrings" and expressionValue is the name of the connect string in the Web.config <<u>connectionStrings</u>> section. However, the syntax can be used to reference <appSettings> elements or content from resource files. See <u>ASP.NET ExpressionS Overview</u> for more on this syntax.

The SelectCommand property specifies the ad-hoc SQL statement or stored procedure to execute to return the data.

Step 3: Adding a Data Web Control and Binding it to the SqlDataSource

Once the SqlDataSource has been configured, it can be bound to a data Web control, such as a GridView or DetailsView. For this tutorial, let's display the data in a GridView. From the Toolbox, drag a GridView onto the page and bind it to the ProductsDataSource SqlDataSource by choosing the data source from the drop-down list

in the GridView's smart tag.

ASPNET_Data_Tutorial_47_CS - N	licrosoft Visual Stu	dio			
Elle Edit Yew Website Build Deb	ug Format Layout	Icols Win	idow <u>C</u> ommur	ity Help Addins	
3.0.3 8 8 8 7 0 0	5.6.21	1	<u>نە</u>		
-	BIU	A.Z. 華	「毎日」	2	
Taobox - + X spx*				Second States and States	* X 2
g Panel	John Tute	viala		Boost Martin & Essent Martin & Compart Martin	A lu
👔 🖂 PlaceHolder	Jata Tuto	mais		Root Hous > Farent Hous > Current Hode	200
U View					Eppi
Substitution	Content	- Content1 ((Custom)		1
Could and the state of the state		-	(
CheckBout istValdator	Quer	ying	Datab	ase Data	Servi
Data					40
Pointer	SqlDataSo	aurce - Product	tsDataSource		Plan
Gid/en	Producti	D ProductN	ame Unit Dr	GridView Tasks	9
DataList	0	abe	0.	Auto Format	200 C
DetailsView		abu	0	Choose Data Source: ProductsDataSource 👻	action of the second se
Fom/Vew no	1	abc	0.1	Configure Data Source Product Data Source	10W
Repeater	2	abc	0.2	Refresh Schema Criew data source	4
g SqDataSource	3	abc	0.3	Edit Columns	Prop
AccessDataSource	4	abc	0.4	Add New Column	entio
Up Objectuscasource with				Enable Paging	3
StellanDataGore				Enable Sorting	
Report Wewer				Enable Selection	
S Validation				Edit Templates	
Pointer ster				A second to the one of the other	100
RequiredFieldValidator					
C RangeValidator					
RegularExpressionY	> <asp:content#conte< th=""><th>enti></th><th><asp:gridview#< th=""><th>gridview1></th><th>(W)</th></asp:gridview#<></th></asp:content#conte<>	enti>	<asp:gridview#< th=""><th>gridview1></th><th>(W)</th></asp:gridview#<>	gridview1>	(W)
Free Life Codeut - Free Dear de	1 Dendon Checker	in the second			
Bank Saud	and the start of the start				
arein(s) seved		and the second second			10

Figure 10: Add a GridView and Bind it to the SqlDataSource Control

Once you've selected the SqlDataSource control from the drop-down list in the GridView's smart tag, Visual Studio will automatically add a BoundField or CheckBoxField to the GridView for each of the columns returned by the data source control. Since the SqlDataSource returns three database columns – ProductID, ProductName, and UnitPrice – there are three fields in the GridView.

Take a moment to configure the GridView's three BoundFields. Change the ProductName field's HeaderText property to "Product Name" and the UnitPrice field's to "Price". Also format the UnitPrice field as a currency. After making these modifications, your GridView's declarative markup should look similar to the following:

```
<asp:GridView ID="GridView1" runat="server" AutoGenerateColumns="False"
DataKeyNames="ProductID" DataSourceID="ProductsDataSource"
EnableViewState="False">
<Columns>
<asp:BoundField DataField="ProductID" HeaderText="ProductID"
InsertVisible="False" ReadOnly="True" SortExpression="ProductID" />
<asp:BoundField DataField="ProductName" HeaderText="Product Name"
SortExpression="ProductName" />
<asp:BoundField DataField="UnitPrice" HeaderText="Price"
SortExpression="UnitPrice" DataFormatString="{0:c}"
HtmlEncode="False" />
</columns>
</asp:GridView>
```

Visit this page through a browser. As Figure 11 shows, the GridView lists each product's ProductID, ProductName, and UnitPrice values.

Dutitled Page - Microsoft Internet	Explorer		
Elle Edit View Favorites Tools t	jelp		100 E
	earch 🥠 Favo	where @ @ - & @ - 0	1 1115
	barch pi rave		46 TEQ2
Address Address http://localhost:4935/ASPNET_I	Data_Tutorial_47	_C5/5qDataSource/Querying.aspx	So 🔁 Go
Working with Da	ita Tut	Orials Home > Using the SqlD: Control > Retrieving Da	taSource tabase Data
Home	Que	rying Database Data	
Basic Reporting	10		
Simple Display	Produc	tID Product Name	Price
Declarative	2	Chang	\$19.00
Parameters	3	Aniseed Syrup	\$10.00
Setting Parameter	4	Chef Anton's Cajun Seasoning	\$26.62
Values	5	Chef Anton's Gumbo Mix	\$21.35
Elitering Reports	6	Grandma's Boysenberry Spread	\$30.25
Filtering Reports	7	Uncle Bob's Organic Dried Pears	\$30.00
Filter by Drop-Down	8	Northwoods Cranberry Sauce	\$36.00
List	9	Mishi Kobe Niku	\$97.00
Master-Details-	10	Ikura	\$31.00
Details	11	Queso Cabrales	\$21.00
Master/Detail Across	12	Queso Manchego La Pastora	\$38.00
Two Pages	13	Konbu	\$6.00
Dutalla of Calastad	14	Tofu	\$23.25
Pow	15	Genen Shouyu	\$15.50
NOW	16	Ravlava	#17 AE
E Done			Local intranet

Figure 11: The GridView Displays Each Product's ProductID, ProductName, and UnitPrice Values

When the page is visited the GridView invokes its data source control's Select() method. When we were using the ObjectDataSource control, this called the ProductsBLL class's GetProducts() method. With the SqlDataSource, however, the Select() method establishes a connection to the specified database and issues the SelectCommand (SELECT [ProductID], [ProductName], [UnitPrice] FROM [Products], in this example). The SqlDataSource returns its results which the GridView then enumerates, creating a row in the GridView for each database record returned.

The Built-In Data Web Control Features and the SqlDataSource Control

In general, the features inherent to the data Web controls – paging, sorting, editing, deleting, inserting, and so on – are specific to the data Web control and are not dependent on the data source control used. That is, the GridView can utilize its built-in paging, sorting, editing, and deleting whether it is bound to an ObjectDataSource or a SqlDataSource. However, certain data Web control features are sensitive to the data source control being used or to the data source control's configuration.

For example, in the <u>Efficiently Paging Through Large Amounts of Data</u> tutorial we discussed how, by default, the paging logic for the data Web controls naively returns <u>all</u> records from the underlying data source and then displays only the appropriate subset of records given the current page index and the number of records to display per page. This model is highly inefficient when paging through sufficiently large result sets. Fortunately, the ObjectDataSource can be configured to support custom paging, which returns only the precise subset of records to display. The SqlDataSource control, however, lacks the properties for implementing custom paging.

Another subtlety with paging and sorting arises with the SqlDataSource. By default, the data returned from a

SqlDataSource can be paged or sorted through the GridView. To demonstrate this, check the Enable Paging and Enable Sorting options in the GridView's smart tag in Querying.aspx and verify that this works as expected.

Sorting and paging works because the SqlDataSource retrieves the database data into a loosely-typed DataSet. The total number of records returned by the query – an essential aspect to implementing paging – can be ascertained from the DataSet. Additionally, the DataSet's results can be sorted through a DataView. These capabilities are automatically used by the SqlDataSource when the GridView requests paged or sorted data.

The SqlDataSource can be configured to return a DataReader instead of a DataSet by changing its <u>DataSourceMode property</u> from DataSet (the default) to DataReader. Using a DataReader might be preferred in situations when passing the SqlDataSource's results to existing code that expects a DataReader. Furthermore, since DataReaders are considerably simpler objects than DataSets, they offer better performance. If you make this change, however, the data Web control can neither sort nor page since the SqlDataSource cannot ascertain how many records are returned by the query, nor does the DataReader offer any techniques for sorting the returned data.

Step 4: Using a Custom SQL Statement or Stored Procedure

When configuring the SqlDataSource control, the query used to return data can be specified in one of two approaches – as a custom SQL statement or stored procedure, or as columns from an existing table or view. In Step 2 we examined selecting columns from the Products table. Let's look at using a custom SQL statement.

Add another GridView control to the Querying.aspx page and choose to create a new data source from the dropdown list in the smart tag. Next, indicate that the data will be pulled from a database – this will create a new SqlDataSource control. Name the control ProductsWithCategoryInfoDataSource.

ata Source Config	uration Wizard			E ? 🛽
Choose	a Data Source Ty	pe		
Where will the app	lication get data from	n?	-	
Access Database	abase Object	Site Map	XML File	
Connect to any SQL	database supported by #	ADO.NET, such a	is Microsoft SQL Ser	ver, Oracle, or OLEDB.
Specify an ID for the	data source:			
ProductsWithCategor	yInfoDataSource			
			1	OK Cancel

Figure 12: Create a New SqlDataSource Control Named ProductsWithCategoryInfoDataSource

The next screen asks us to specify the database. As we did back in Figure 7, select the

NORTHWINDConnectionString from the drop-down list and click Next. In the "Configure the Select Statement" screen, choose the "Specify a custom SQL statement or stored procedure" radio button and click Next. This will bring up the "Define Custom Statements or Stored Procedures" screen, which offers tabs labeled SELECT, UPDATE, INSERT, and DELETE. In each tab you can enter a custom SQL statement into the textbox or choose a stored procedure from the drop-down list. In this tutorial we will look at entering a custom SQL statement; the next tutorial includes an example that uses a stored procedure.

Configure Data Source - ProductsWithCategoryInfoDataSource	• • • ? 🛛
Define Custom Statements or Stored Procedures	
Click a tab to create a SQL statement for that operation.	
SELECT UPDATE INSERT DELETE	
SQL statement:	
	~
Quen	y builder
O Stored procedure:	
Cuscordemsc.	
< <u>Previous</u> <u>N</u> ext > <u>F</u> inish	Cancel

Figure 13: Enter a Custom SQL Statement or Pick a Stored Procedure

The custom SQL statement can be entered by hand into the textbox or can be constructed graphically by clicking the Query Builder button. From either the Query Builder or the textbox, use the following query to return the ProductID and ProductName fields from the Products table using a JOIN to retrieve the product's CategoryName from the Categories table:

```
SELECT Products.ProductID, Products.ProductName, Categories.CategoryName
FROM Categories
INNER JOIN Products ON
Categories.CategoryID = Products.CategoryID
```

Query	Builder					6?🛛
	Categories CategoryID CategoryID CategoryName Description Picture	-	Prod	oducts Il Columns) luctID luctName olierID agoryID	-	
	,		,,			~
	Column	Alias Table	,	Output	Sort Type	Sort Or
•	ProductID	Prod	ucts			
	ProductName	Prod	ucts	V		
	CategoryName	Cate	gories	V		~
<		· · · · · · · · · · · · · · · · · · ·				>
FROM	ProductID	R JOIN ategories.Category ProductName	ID = Proc	ducts.Categ	joryID	~
•	1	Chai Tea	Bever:	anes		
	2	Chang	Bever	anes		
	3	Aniseed Syrup	Condir	nents		
	4	Chef Anton's Ca	Condir	ments		
	5	Chef Anton's Gu	Condir	nents		
	6	Grandma's Boys	Condir	nents		
	7	Uncle Bob's Org	Produc	te		~
14 4	1 of 79	► ► ► (@) Cell is	Read Only		
Execu	te Query			C	ок (Cancel

Figure 14: You can Graphically Construct the Query Using the Query Builder

After specifying the query, click Next to proceed to the "Test Query" screen. Click Finish to complete the SqlDataSource wizard.

After completing the wizard, the GridView will have three BoundFields added to it displaying the ProductID, ProductName, and CategoryName columns returned from the query and resulting in the following declarative markup:

<asp:SqlDataSource ID="ProductsWithCategoryInfoDataSource" runat="server"
 ConnectionString="<%\$ ConnectionStrings:NORTHWNDConnectionString %>"
 SelectCommand="
 SELECT Products Products Products Product Name Categories Categories

SELECT Products.ProductID, Products.ProductName, Categories.CategoryName FROM Categories

```
INNER JOIN Products ON Categories.CategoryID = Products.CategoryID"> </asp:SqlDataSource>
```

🗿 Untitled Page - Microsoft Interne	t Explorer		• • • •	
Ele Edit View Favorites Iools	Fielb			
🗿 Back • 🔘 - 💽 🖹 🐔 🔎	Search 👷 Favo	ntes 🐵 🗿 • 🍒 👿 • 🗔 🛛 🌀	🌭 🕰 🏢	
Address () http://localhost:4935/ASPNET	Data_Tutorial_47	_CS/5qDataSource/Querying.aspx	~	> Go
Timo Pages	Produc	tID ProductName	CategoryName	^
Maser/Detail on One	1	Chai Tea	Beverages	
Page	2	Chang	Beverages	
Editing and Deleting	3	Aniseed Syrup	Condiments	
with the DataList	4	Chef Anton's Cajun Seasoning	Condiments	1
Cosics	5	Chef Anton's Gumbo Mix	Condiments	1.1
Batch Update	6	Grandma's Boysenberry Spread	Condiments	
Adding Data Entry	7	Uncle Bob's Organic Dried Pears	Produce	
Validation	8	Northwoods Cranberry Sauce	Condiments	ALC: N
Interface	9	Mishi Kobe Niku	Meat/Poultry	
Contraction Inc.	10	Ikura	Seafood	
Concurrency	11	Queso Cabrales	Dairy Products	21
concurrency	12	Queso Manchego La Pastora	Dairy Products	1.
Confirm On Delete	13	Konbu	Seafood	
Limit Capabilities	14	Tofu	Produce	101
Based on User	15	Genen Shouyu	Condiments	
Paging and Sorting	16	Pavlova	Confections	1
raging and sorting	17	Alice Mutton	Meat/Doultry	×
ð)			S Local intranet	

Figure 15: The GridView Shows Each Product's ID, Name, and Associated Category Name

Summary

In this tutorial we saw how to query and display data using the SqlDataSource control. Like the ObjectDataSource, the SqlDataSource serves as a proxy, providing a declarative approach to accessing data. Its properties specify the database to connect to and the SQL SELECT query to execute; they can be specified through the Properties window or by using the Configure DataSource wizard.

The SELECT query examples we examined in this tutorial returned all of the records from the specified query. The SqlDataSource control, however, can include a WHERE clause with parameters whose values are assigned programmatically or are automatically pulled from a specified source. We'll examine how to create and use parameterized queries in the next tutorial!

Happy Programming!

Further Reading

For more information on the topics discussed in this tutorial, refer to the following resources:

- Accessing Relational Database Data
- <u>SqlDataSource Control Overview</u>
- <u>ASP.NET Quickstart Tutorials: The SqlDataSource Control</u>
- <u>The Web.config <connectionStrings> Element</u>
- Database Connection String Reference

About the Author

Scott Mitchell, author of seven ASP/ASP.NET books and founder of <u>4GuysFromRolla.com</u>, has been working with Microsoft Web technologies since 1998. Scott works as an independent consultant, trainer, and writer. His latest book is <u>Sams Teach Yourself ASP.NET 2.0 in 24 Hours</u>. He can be reached at <u>mitchell@4GuysFromRolla.com</u>. or via his blog, which can be found at <u>http://ScottOnWriting.NET</u>.

Special Thanks To...

This tutorial series was reviewed by many helpful reviewers. Lead reviewers for this tutorial were Susan Connery, Bernadette Leigh, and David Suru. Interested in reviewing my upcoming MSDN articles? If so, drop me a line at mitchell@4GuysFromRolla.com.