This tutorial is part of a set. Find out more about data access with ASP.NET in the Working with Data in ASP.NET 2.0 section of the ASP.NET site at http://www.asp.net/learn/dataaccess/default.aspx.

# Working with Data in ASP.NET 2.0 :: Displaying Data With the ObjectDataSource

## Introduction

With our application architecture and website page layout complete, we're ready to start exploring how to accomplish a variety of common data- and reporting-related tasks. In the previous tutorials we've seen how to programmatically bind data from the DAL and BLL to a data Web control in an ASP.NET page. This syntax – assigning the data Web control's DataSource property to the data to display and then calling the control's DataBind() method – was the pattern used in ASP.NET 1.x applications, and can continue to be used in your 2.0 applications. However, ASP.NET 2.0's new data source controls offer a declarative way to work with data. Using these controls you can bind data retrieved from the BLL created in the previous tutorial without having to write a line of code!

ASP.NET 2.0 ships with five built-in data source controls – <u>SqlDataSource</u>, <u>AccessDataSource</u>, <u>ObjectDataSource</u>, <u>XmlDataSource</u>, and <u>SiteMapDataSource</u> – although you can build your own <u>custom data</u> <u>source controls</u>, if needed. Since we have developed an architecture for our tutorial application, we'll be using the ObjectDataSource against our BLL classes.



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

The ObjectDataSource serves as a proxy for working with some other object. To configure the ObjectDataSource we specify this underlying object and how its methods map to the ObjectDataSource's select, Insert, Update, and Delete methods. Once this underlying object has been specified and its methods mapped to the ObjectDataSource's, we can then bind the ObjectDataSource to a data Web control. ASP.NET ships with many data Web controls, including the GridView, DetailsView, RadioButtonList, and DropDownList, among others. During the page lifecycle, the data Web control may need to access the data it's bound to, which it will accomplish by invoking its ObjectDataSource's select method; if the data Web control

supports inserting, updating, or deleting, calls may be made to its ObjectDataSource's Insert, Update, or Delete methods. These calls are then routed by the ObjectDataSource to the appropriate underlying object's methods as the following diagram illustrates.



Figure 2: The ObjectDataSource Serves as a Proxy

While the ObjectDataSource can be used to invoke methods for inserting, updating, or deleting data, let's just focus on returning data; future tutorials will explore using the ObjectDataSource and data Web controls that modify data.

# **Step 1: Adding and Configuring the ObjectDataSource Control**

Start by opening the SimpleDisplay.aspx page in the BasicReporting folder, switch to Design view, and then drag an ObjectDataSource control from the Toolbox onto the page's design surface. The ObjectDataSource appears as a gray box on the design surface because it does not produce any markup; it simply accesses data by invoking a method from a specified object. The data returned by an ObjectDataSource can be displayed by a data Web control, such as the GridView, DetailsView, FormView, and so on.

**Note:** Alternatively, you may first add the data Web control to the page and then, from its smart tag, choose the <New data source> option from the drop-down list.

To specify the ObjectDataSource's underlying object and how that object's methods map to the ObjectDataSource's, click on the Configure Data Source link from the ObjectDataSource's smart tag.

Code - Microsoft Visual Studio File Edit View Webgte Build Debug	Figmat Layout Iools Window Community Help Addins
BasicReportingIsplay.aspx.cs	- B Z U A Z = - E E S
Working with D	ata Tutorials
Home Basic Reporting	Content - Content1 (Custom)     ObjectDataSource Tasks
Filtering Reports	Configure Data Source

Figure 3: Click the Configure Data Source Link from the Smart Tag

This brings up the Configure Data Source wizard. First, we must specify the object the ObjectDataSource is to work with. If the "Show only data components" checkbox is checked, the drop-down list on this screen lists only those objects that have been decorated with the DataObject attribute. Currently our list includes the TableAdapters in the Typed DataSet and the BLL classes we created in the previous tutorial. If you forgot to add the DataObject attribute to the Business Logic Layer classes you won't see them in this list. In that case, uncheck the "Show only data components" checkbox to view all objects, which should include the BLL classes (along with the other classes in the Typed DataSet – the DataTables, DataRows, and so on).

From this first screen choose the ProductsBLL class from the drop-down list and click Next.

Configure Data Source - ObjectDataSource1	? 🛛
Choose a Business Object	
Select a business object that can be used to retrieve or updat or App_Code directory for this application). Choose your business object:	e data (for example, an object defined in the Bin
ProductsBLL	Show only data components
CategoriesBLL EmployeesBLL NorthwindTableAdapters.CategoriesTableAdapter NorthwindTableAdapters.EmployeesTableAdapter NorthwindTableAdapters.ProductsTableAdapter NorthwindTableAdapters.SuppliersTableAdapter ProductsBLL	
SuppliersBLL	
< Previous	Next > Einish Cancel

## Figure 4: Specify the Object to Use with the ObjectDataSource Control

The next screen in the wizard prompts you to select what method the ObjectDataSource should invoke. The drop-down lists those methods that return data in the object selected from the previous screen. Here we see GetProductsByProductID, GetProducts, GetProductsByCategoryID, and GetProductsBySupplierID. Select the GetProducts method from the drop-down list and click Finish (if you added the DataObjectMethodAttribute to the ProductBLL's methods as shown in the previous tutorial, this option will be selected by default).

Configure	e Data So	urce - O	bjectData	Source1				? 🛛
Ļ	Define	Data N	lethods					
SELECT	UPDATE	INSERT	DELETE					
Choos metho Exam <u>C</u> hoos	se a methor od can retu ple: GetPro se a methor	d of the bu m a Datas ducts(Int3 d:	isiness object iet, DataReax i2 categoryId	t that returns o der, or strong! (), returns a De	lata to asso /-typed coli ataSet.	ociate with t ection.	he SELECT o	peration, The
GetPr	oducts(), r	eturns Pro	ductsDataTal	ble	~			
GetPr	oductByPro	ductID(In	t32 productII	0), returns Pro	ductsDataT	able		
GetPr GetPr	oducts8yCa oducts8ySa	ategoryID pplierID(I	(Int32 catego nt32 supplier)	rryID), returns ID), returns Pr	ProductsDa oductsData	staTable Table		×
				< Previor	as No	ext > [	Einish	Cancel

Figure 5: Choose the Method for Returning Data from the SELECT Tab

## **Configure the ObjectDataSource Manually**

The ObjectDataSource's Configure Data Source wizard offers a quick way to specify the object it uses and to associate what methods of the object are invoked. You can, however, configure the ObjectDataSource through its properties, either through the Properties window or directly in the declarative markup. Simply set the TypeName property to the type of the underlying object to be used, and the SelectMethod to the method to invoke when retrieving data.

```
<asp:ObjectDataSource ID="ObjectDataSource1" runat="server"
SelectMethod="GetProducts"
TypeName="ProductsBLL">
</asp:ObjectDataSource>
```

Even if you prefer the Configure Data Source wizard there may be times when you need to manually configure the ObjectDataSource, as the wizard only lists developer-created classes. If you want to bind the ObjectDataSource to a class in the .NET Framework – such as the <u>Membership class</u>, to access user account information, or the <u>Directory class</u> to work with file system information – you'll need to manually set the ObjectDataSource's properties.

# **Step 2: Adding a Data Web Control and Binding it to the ObjectDataSource**

Once the ObjectDataSource has been added to the page and configured, we're ready to add data Web controls to the page to display the data returned by the ObjectDataSource's select method. Any data Web control can be bound to an ObjectDataSource; let's look at displaying the ObjectDataSource's data in a GridView, DetailsView, and FormView.

# Binding a GridView to the ObjectDataSource

Add a GridView control from the Toolbox to SimpleDisplay.aspx's design surface. From the GridView's smart tag, choose the ObjectDataSource control we added in Step 1. This will automatically create a BoundField in the GridView for each property returned by the data from the ObjectDataSource's Select method (namely, the properties defined by the Products DataTable).

Republic tales are a Rest	BIU	A.2	12,121				
Working with Da	ita Tu	torials		Want Mode	• Enouthing	- Carrent No.	
Home	Contr	anit - Content1 (C	Custom)				
Basic Reporting  titigertDataSource - OtjectDataSource 1 Filtering Reports Customized Formatting FB							
SteMapDataSource - SteMapDataSour	Produ	ctID ProductN	tame Suppl	ierID Catego	oryID QuantityP	erUnit UnitPr	ice Uni
	1	apc	1	1	abc	0.1	0
	2	abc	2	2	abc	0.2	2
	3	abc	3	3	abo	0.3	3
	4	abc	4	4	abc	0,4	4
	22						

## Figure 6: A GridView Has Been Added to the Page and Bound to the ObjectDataSource

You can then customize, rearrange, or remove the GridView's BoundFields by clicking the Edit Columns option from the smart tag.

Fields				?	
Available fields:	Во	undField properties:			
(All Fields)	00	<b>]</b> 2↓ 🔤			
BoundField	🗆 Accessibility				
Productio		AccessibleHeaderTe			
		Appearance			
CategoryID		FooterText			
🔄 🔄 QuantityPerUnit		HeaderImageUrl			
		HeaderText	ProductID		
Add	Ξ	Behavior			
Selected fields:		ApplyFormatInEditM	False		
		ConvertEmptyString	True		
ProductName		HtmlEncode	True		
I SupplierID		InsertVisible	False	~	
CategoryID	He Th	eaderText he text within the hea	ader of this field.		
Auto-generate fields	Co	nvert this field into a	TemplateField		
Refresh Schema			ОК Саг	ncel	

## Figure 7: Manage the GridView's BoundFields Through the Edit Columns Dialog Box

Take a moment to modify the GridView's BoundFields, removing the ProductID, SupplierID, CategoryID, QuantityPerUnit, UnitsInStock, UnitsOnOrder, and ReorderLevel BoundFields. Simply select the BoundField from the list in the bottom left and click the delete button (the red X) to remove them. Next, Rearrange the BoundFields so that the CategoryName and SupplierName BoundFields precede the UnitPrice BoundField by selecting these BoundFields and clicking the up arrow. Set the HeaderText properties of the remaining BoundFields to Products, Category, Supplier, and Price, respectively. Next, have the Price BoundField formatted as a currency by setting the BoundField's HtmlEncode property to False and its DataFormatString property to {0:c}. Finally, horizontally align the Price to the right and the Discontinued checkbox in the center via the ItemStyle/HorizontalAlign property.

```
<asp:GridView ID="GridView1" runat="server"</pre>
  AutoGenerateColumns="False"
                                  DataKeyNames="ProductID"
  DataSourceID="ObjectDataSource1" EnableViewState="False">
    <Columns>
        <asp:BoundField DataField="ProductName"</pre>
         HeaderText="Product" SortExpression="ProductName" />
        <asp:BoundField DataField="CategoryName"</pre>
          HeaderText="Category" ReadOnly="True"
          SortExpression="CategoryName" />
        <asp:BoundField DataField="SupplierName"</pre>
          HeaderText="Supplier" ReadOnly="True"
          SortExpression="SupplierName" />
        <asp:BoundField DataField="UnitPrice"</pre>
          DataFormatString="{0:c}" HeaderText="Price"
            HtmlEncode="False" SortExpression="UnitPrice">
            <ItemStyle HorizontalAlign="Right" />
        </asp:BoundField>
        <asp:CheckBoxField DataField="Discontinued"</pre>
          HeaderText="Discontinued" SortExpression="Discontinued">
            <ItemStyle HorizontalAlign="Center" />
        </asp:CheckBoxField>
    </Columns>
</asp:GridView>
```



### Figure 8: The GridView's BoundFields Have Been Customized

## Using Themes for a Consistent Look

These tutorials strive to remove any control-level style settings, instead using cascading style sheets defined in an external file whenever possible. The <code>Styles.css</code> file contains <code>DataWebControlStyle</code>, <code>HeaderStyle</code>, <code>RowStyle</code>, and <code>AlternatingRowStyle</code> CSS classes that should be used to dictate the appearance of the data Web controls used in these tutorials. To accomplish this, we could set the GridView's <code>Cssclass</code> property to <code>DataWebControlStyle</code>, and its <code>HeaderStyle</code>, <code>RowStyle</code>, and <code>AlternatingRowStyle</code>, <code>RowStyle</code>, and <code>AlternatingRowStyle</code> properties' <code>Cssclass</code> properties accordingly.

If we set these CSSClass properties at the Web control we'd need to remember to explicitly set these property values for each and every data Web control added to our tutorials. A more manageable approach is to define the default CSS-related properties for the GridView, DetailsView, and FormView controls using a Theme. A Theme is a collection of control-level property settings, images, and CSS classes that can be applied to pages across a site to enforce a common look and feel.

Our Theme won't include any images or CSS files (we'll leave the stylesheet Styles.css as-is, defined in the root folder of the web application), but will include two Skins. A Skin is a file that defines the default properties for a Web control. Specifically, we'll have a Skin file for the GridView and DetailsView controls, indicating the default cssclass-related properties.

Start by adding a new Skin File to your project named GridView.skin by right-clicking on the project name in the Solution Explorer and choosing Add New Item.

Add New Item	- C: Wy Proj	ects\Writings	Microsoft	MSDN Article	sWSDN Onli	ne Articles\D	ataTutori	2
Templates:								
Web Form	Master Page	Web User Control	HTML Page	Web Service	Class	Style Sheet	Global Applicati	2
	-							
Web Configurati	XML File	XML Schema	Text File	Resource File	SQL Database	Data5et	Generic Handler	
22		<u></u>	==	3		B	E Z	
Site Map	Mobile Web Form	VBScript File	Report	3Script File	Mobile Web User Control	Mobile Web Configurati	XSLT File	
	•	B						
Skin File	Browser File	Class Diagram						*
A file used to de	fine an ASP.MET	theme						
Name:	GridView.sk	in .						
Language:	Visual C#			Elace code in Select master	separato file page			
						6	dd 🔍	ancel

Figure 9: Add a Skin File Named GridView.skin

Skin files need to be placed in a Theme, which are located in the App\_Themes folder. Since we don't yet have such a folder, Visual Studio will kindly offer to create one for us when adding our first Skin. Click Yes to create the App\_Theme folder and place the new GridView.skin file there.



### Figure 10: Let Visual Studio Create the App\_Theme Folder

This will create a new Theme in the App Themes folder named GridView with the Skin file GridView.skin.



### Figure 11: The GridView Theme has Been Added to the App\_Theme Folder

Rename the GridView Theme to DataWebControls (right-click on the GridView folder in the App\_Theme folder and choose Rename). Next, enter the following markup into the GridView.skin file:

This defines the default properties for the CssClass-related properties for any GridView in any page that uses the DataWebControls Theme. Let's add another Skin for the DetailsView, a data Web control that we'll be using shortly. Add a new Skin to the DataWebControls Theme named DetailsView.skin and add the following markup:

```
<asp:DetailsView runat="server" CssClass="DataWebControlStyle">
<AlternatingRowStyle CssClass="AlternatingRowStyle" />
<RowStyle CssClass="RowStyle" />
<FieldHeaderStyle CssClass="HeaderStyle" />
</asp:DetailsView>
```

With our Theme defined, the last step is to apply the Theme to our ASP.NET page. A Theme can be applied

on a page-by-page basis or for all pages in a website. Let's use this theme for all pages in the website. To accomplish this, add the following markup to Web.config's <system.web> section:

#### <pages styleSheetTheme="DataWebControls" />

That's all there is to it! The styleSheetTheme setting indicates that the properties specified in the Theme should *not* override the properties specified at the control level. To specify that Theme settings should trump control settings, use the theme attribute in place of styleSheetTheme; unfortunately, Theme settings specified via the theme attribute do not appear in the Visual Studio Design view. Refer to <u>ASP.NET Themes</u> and <u>Skins Overview</u> and <u>Server-Side Styles Using Themes</u> for more information on Themes and Skins; see <u>How To: Apply ASP.NET Themes</u> for more on configuring a page to use a theme.

le falt Yew Fyrantas Jools	Gele Search 🏫 Favorites 🕐 💭+	& @ • 🗆	0 × 11 10		
Vorking with Da	ata Tutorials		Home > Basic R	eporting > Sir	nple Display
Home Basic Reporting	GridView Exan	nple			
Simple Display	Product	Category	Supplier	Price Dis	scontinued
Declarative	chai	Beverages	Exotic Liquids	\$18.00	
Parameters	chang	Deverages	Exotic Liquids	\$19.00	And a start
Setting Paramétér Values	Chef Anton's Cajun	Condiments	New Orleans Cajun	\$10.00	
Filtering Reports	Chef Anton's Gumbo Mix	Condiments	New Orleans Cajun Delights	\$21.35	
List	Grandma's Boysenberry Spread	Condiments	Grandma Kelly's Homestead	\$25.00	(D)
Master-Details- Details	Uncle Bob/s Organic Dried Pears	Produce	Grandma Kelly's Homestead	\$30.00	
Details of Selected Row	Northwoods Cranberry Sauce	Condiments	Grandma Kelly's Homestead	\$40.00	
Customized	Mishi Kobe Niku	Meat/Poultry	Tokyo Traders	\$97.00	E
Formatting	Ikura	Seafood	Tokyo Traders	\$31.00	
Format Colors	Queso Cabrales	Dairy Products	Cooperativa de Quesos 'Las Cabras'	\$21.00	
Custom Content in a Gridvlew	Queso Manchego La Pastora	Dairy Products	Cooperativa de Quesos 'Las Cabras'	\$38.00	
Custom Content in a	Konbu	Seafood	Mayumi's	\$6.00	

Figure 12: The GridView Displays the Product's Name, Category, Supplier, Price, and Discontinued Information

# **Displaying One Record at a Time in the DetailsView**

The GridView displays one row for each record returned by the data source control to which it's bound. There are times, however, when we may want to display a sole record or just one record at a time. The <u>DetailsView</u> <u>control</u> offers this functionality, rendering as an HTML with two columns and one row for each column or property bound to the control. You can think of the DetailsView as a GridView with a single record rotated 90 degrees.

Start by adding a DetailsView control *above* the GridView in SimpleDisplay.aspx. Next, bind it to the same ObjectDataSource control as the GridView. Like with the GridView, a BoundField will be added to the DetailsView for each property in the object returned by the ObjectDataSource's Select method. The only difference is that the DetailsView's BoundFields are laid out horizontally rather than vertically.



Figure 13: Add a DetailsView to the Page and Bind it to the ObjectDataSource

Like the GridView, the DetailsView's BoundFields can be tweaked to provide a more customized display of the data returned by the ObjectDataSource. Figure 14 shows the DetailsView after its BoundFields and Cssclass properties have been configured to make its appearance similar to the GridView example.

Untitled Page - Microsoft Internet	t Explorer	
Elle Edit View Pavorites Iools	Help	1
O Back + O + 🖻 🖬 💰 🔑	Search 💁 Favorites 🔗 🙆 🍇 📝	• 🖸 🌒 🐂 🚳
Address Address Address Address Address Address	isicReporting/SimpleDisplay.aspx	🛩 🄁 Go
Working with D	ata Tutorials Home Simpl	e > <u>Basic Reporting</u> > le Display
Home	DetailsView Examle	e e
Simple Display	Product Chai	
Declarative Parameters	Category Beverages Supplier Exotic Liquid	5
Setting Parameter Values	Price \$18.00 Discontinued	
<	#	>
Done Done		Secol Intranet

Figure 14: The DetailsView Shows a Single Record

Note that the DetailsView only displays the first record returned by its data source. To allow the user to step through all of the records, one at a time, we must enable paging for the DetailsView. To do so, return to Visual Studio and check the Enable Paging checkbox in the DetailsView's smart tag.



Figure 15 Enable Paging in the DetailsView Control



Figure 16: With Paging Enabled, the DetailsView Allows the User to View Any of the Products

We'll talk more about paging in future tutorials.

# A More Flexible Layout for Showing One Record at a Time

The DetailsView is pretty rigid in how it displays each record returned from the ObjectDataSource. We may want a more flexible view of the data. For example, rather than showing the product's name, category, supplier, price, and discontinued information each on a separate row, we may want to show the product name and price in an <h4> heading, with the category and supplier information appearing below the name and price in a smaller font size. And we may not care to show the property names (Product, Category, and so on) next to the values.

The <u>FormView control</u> provides this level of customization. Rather than using fields (like the GridView and DetailsView do), the FormView uses templates, which allow for a mix of Web controls, static HTML, and <u>databinding syntax</u>. If you are familiar with the Repeater control from ASP.NET 1.x, you can think of the FormView as the Repeater for showing a single record.

Add a FormView control to the SimpleDisplay.aspx page's design surface. Initially the FormView displays as a gray block, informing us that we need to provide, at minimum, the control's ItemTemplate.



Figure 17: The FormView Must Include an ItemTemplate

You can bind the FormView directly to a data source control through the FormView's smart tag, which will create a default ItemTemplate automatically (along with an EditItemTemplate and InsertItemTemplate, if the ObjectDatatSource control's InsertMethod and UpdateMethod properties are set). However, for this example let's bind the data to the FormView and specify its ItemTemplate manually. Start by setting the FormView's DataSourceID property to the ID of the ObjectDataSource control, ObjectDataSource1. Next, create the ItemTemplate so that it displays the product's name and price in an <h4> element and the category and shipper names beneath that in a smaller font size.

```
<asp:FormView ID="FormView1" runat="server"
DataSourceID="ObjectDataSource1" EnableViewState="False">
<ItemTemplate>
```

```
<h4><%# Eval("ProductName") %>
(<%# Eval("UnitPrice", "{0:c}") %>)</h4>
Category: <%# Eval("CategoryName") %>;
Supplier: <%# Eval("SupplierName") %>
</ItemTemplate>
</asp:FormView>
```



Figure 18: The First Product (Chai) is Displayed in a Custom Format

The <%# Eval (propertyName) %> is the databinding syntax. The Eval method returns the value of the specified property for the current object being bound to the FormView control. Check out Alex Homer's article <u>Simplified and Extended Data Binding Syntax in ASP.NET 2.0</u> for more information on the ins and outs of databinding.

Like the DetailsView, the FormView only shows the first record returned from the ObjectDataSource. You can enable paging in the FormView to allow visitors to step through the products one at a time.

## Summary

Accessing and displaying data from a Business Logic Layer can be accomplished without writing a line of code thanks to ASP.NET 2.0's ObjectDataSource control. The ObjectDataSource invokes a specified method of a class and returns the results. These results can be displayed in a data Web control that's bound to the ObjectDataSource. In this tutorial we looked at binding the GridView, DetailsView, and FormView controls to the ObjectDataSource.

So far we've only seen how to use the ObjectDataSource to invoke a parameter-less method, but what if we want to invoke a method that expects input parameters, such as the ProductBLL class's GetProductsByCategoryID(*categoryID*)? In order to call a method that expects one or more parameters we must configure the ObjectDataSource to specify the values for these parameters. We'll see how to accomplish this in our next tutorial.

Happy Programming!

# **Further Reading**

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

- <u>Create Your Own Data Source Controls</u>
- <u>GridView Examples for ASP.NET 2.0</u>
- Simplified and Extended Data Binding Syntax in ASP.NET 2.0
- Themes in ASP.NET 2.0
- <u>Server-Side Styles Using Themes</u>
- How To: Apply ASP.NET Themes Programmatically

# **About the Author**

Scott Mitchell, author of six ASP/ASP.NET books and founder of 4GuysFromRolla.com, has been working with Microsoft Web technologies since 1998. Scott works as an independent consultant, trainer, and writer, recently completing his latest book, <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 reviewer for this tutorial was Hilton Giesenow. Interested in reviewing my upcoming MSDN articles? If so, drop me a line at mitchell@4GuysFromRolla.com.