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 :: Master/Detail Filtering With Two DropDownLists

### Introduction

In the <u>previous tutorial</u> we examined how to display a simple master/details report using a single DropDownList populated with the categories and a GridView showing those products that belong to the selected category. This report pattern works well when displaying records that have a one-to-many relationship and can easily be extended to work for scenarios that include multiple one-to-many relationships. For example, an order entry system would have tables that correspond to customers, orders, and order line items. A given customer may have multiple orders with each order consisting of multiple items. Such data can be presented to the user with two DropDownLists and a GridView. The first DropDownList would have a list item for each customer in the database with the second one's contents being the orders placed by the selected customer. A GridView would list the line items from the selected order.

While the Northwind database include the canonical customer/order/order details information in its Customers, Orders, and Order Details tables, these tables aren't captured in our architecture. Nonetheless, we can still illustrate using two dependent DropDownLists. The first DropDownList will list the categories and the second the products belonging to the selected category. A DetailsView will then list the details of the selected product.

## **Step 1: Creating and Populating the Categories DropDownList**

Our first goal is to add the DropDownList that lists the categories. These steps were examined in detail in the preceding tutorial, but are summarized here for completeness.

Open the MasterDetailsDetails.aspx page in the Filtering folder, add a DropDownList to the page, set its ID property to Categories, and then click the Configure Data Source link in its smart tag. From the Data Source Configuration Wizard choose to add a new data source.

Data Source Configuration Wizard	? 🛛
Choose a Data Source	-1115
Select a data source:	
	OK Cancel

#### Figure 1: Add a New Data Source for the DropDownList

The new data source should, naturally, be an ObjectDataSource. Name this new ObjectDataSource CategoriesDataSource and have it invoke the CategoriesBLL object's GetCategories() method.



Figure 2: Choose to Use the CategoriesBLL Class

Configure	Data So	urce - c	ategoriesDat	Source					?	×
Ļ	Define	Data N	1ethods							
SELECT	UPDATE	INSERT	DELETE							
Choos metho Examp Choos	e a methor id can retu ole: GetPro	d of the bi m a Datas ducts(Int3 d:	usiness object th Set, DataReader, 32 categoryId), r	at returns d or strongly aturns a Da	ata to assi -typed col taSet.	ociate with lection.	the SELEC	T operation.	The	
GetCa	tegories(),	returns (	CategoriesDataTo	ble	~					
GetCa	tegories(),	returns C	ategoriesDataTa VIot32 categorie	ble D) returns	Categorie	CotaTable				
GetC	ategories()	, returns (	CategoriesDataT	able	Carogone	20001000			K. 12	
				< <u>P</u> reviou	n N	ext >	Einist		Iancel	

#### Figure 3: Configure the ObjectDataSource to Use the GetCategories () Method

After configuring the ObjectDataSource we still need to specify which data source field should be displayed in the Categories DropDownList and which one should be configured as the value for the list item. Set the CategoryName field as the display and CategoryID as the value for each list item.

Data Source Configuration Wizard	28
Choose a Data Source	
Select a data source:	
CategoriesDataSource	
Select a data field to display in the DropDownList:	
CategoryName 🗸	
Select a data field for the value of the DropDownList:	
EstegoryID 😽	
CategoryID CategoryName Description	
Refresh Schema	
	OK Cancel

### Figure 4: Have the DropDownList Display the CategoryName Field and Use CategoryID as the Value

At this point we have a DropDownList control (Categories) that's populated with the records from the Categories table. When the user chooses a new category from the DropDownList we'll want a postback to occur in order to refresh the product DropDownList that we're going to create in Step 2. Therefore, check the Enable AutoPostBack option from the categories DropDownList's smart tag.



Figure 5: Enable AutoPostBack for the Categories DropDownList

### **Step 2: Displaying the Selected Category's Products in a Second DropDownList**

With the Categories DropDownList completed, our next step is to display a DropDownList of products belonging to the selected category. To accomplish this, add another DropDownList to the page named ProductsByCategory. As with the Categories DropDownList, create a new ObjectDataSource for the ProductsByCategory DropDownList named ProductsByCategoryDataSource.

Data Source Configuration Wizard	? 🛛
Choose a Data Source	
Select a data source:	
<pre></pre>	
	OK Cancel

Figure 6: Add a New Data Source for the ProductsByCategory DropDownList

ata Source C	onfiguration	Wizard				? 🛛
de la ch	ioose a Data	Source Ty	ре			
Where will th	e application	get data from	n?			
2						
Access Database	Database	Object	Site Map	XML File		
C		22				
ProductsBvCat	egoryDataSourc	e				
- Anna - Anna		-				
					ОК	Cancel
						-

### Figure 7: Create a New ObjectDataSource Named ProductsByCategoryDataSource

Since the ProductsByCategory DropDownList needs to display just those products belonging to the selected category, have the ObjectDataSource invoke the GetProductsByCategoryID(*categoryID*) method from the ProductsBLL object.

Configure Data Source - productsByCategoryDataSour	ce 🛛 🛛 🔀
Choose a Business Object	
Select a business object that can be used to retrieve or update da or App_Code directory for this application). Choose your business object:	ata (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 8: Choose to Use the ProductsBLL Class

Configure Data Source - productsByCategoryDataSource	? 🛛
Define Data Methods	
SELECT UPDATE INSERT DELETE	
Choose a method of the business object that returns data to associate with the SELECT operation method can return a DataSet, DataReader, or strongly-typed collection. Example: GetProducts(Int32 categoryId), returns a DataSet. Choose a method:	n. The
GetProductsByCategoryID(Int32 categoryID), returns Prc 🐱	
GetProductByProductID(Int32 productID), returns ProductsDataTable GetProducts(), returns ProductsDataTable	
GetProductsByCategorvID(Int32 categoryID), returns ProductsDataTable GetProductsBySupplierID(Int32 supplierID), returns ProductsDataTable	< 3
< Previous Next > Enish	Cancel

### Figure 9: Configure the ObjectDataSource to Use the GetProductsByCategoryID (*categoryID*) Method

In the final step of the wizard we need to specify the value of the *categoryID* parameter. Assign this parameter to the selected item from the *categories* DropDownList.

onfigure Data Source - ProductsByCatego	ryDataSource 🛛 🛛 🔀
Define Parameters	
The wizard has detected one or more parameters in method, choose a source for the parameter's value.	your SELECT method. For each parameter in the SELECT
Alama Value	Control
categoryID Categories.SelectedValue	ControlID:
	Categories 🗸
	DefaultValue:
	Show advanced properties
Method signature:	
GetProductsByCategoryID(Int32 categoryID), retu	rns ProductsDataTable
<8	revious Next > Einish Cancel

### Figure 10: Pull the *categoryID* Parameter Value from the Categories DropDownList

With the ObjectDataSource configured, all that remains is to specify what data source fields are used for the display and value of the DropDownList's items. Display the ProductName field and use the ProductID field as the value.

ata Source Configuration W	izard		? 🛛
Choose a Data S	ource		
Select a data source:			
ProductsByCategoryDataSource	*		
Select a data field to display in the	DropDownList:		
ProductName	~		
Select a data field for the value of	the DropDownList:		
ProductID	~		
ProductID ProductName SupplierID CategoryID QuantkyPerUnit UnitPrice UnitSIDStock UnitSOnOrder Refresh Schema	~		
Nonexi Scheme			
		ок	Cancel

## Figure 11: Specify the Data Source Fields Used for the DropDownList's ListItems' Text and Value Properties

With the ObjectDataSource and ProductsByCategory DropDownList configured our page will display two DropDownLists: the first will list all of the categories while the second will list those products belonging to the selected category. When the user selects a new category from the first DropDownList, a postback will ensue and the second DropDownList will be rebound, showing those products that belong to the newly selected category. Figures 12 and 13 show MasterDetails.aspx in action when viewed through a browser.



Figure 12: When First Visiting the Page, the Beverages Category is Selected



Figure 13: Choosing a Different Category Displays the New Category's Products

Currently the productsByCategory DropDownList, when changed, does *not* cause a postback. However, we will want a postback to occur once we add a DetailsView to display the selected product's details (Step 3). Therefore, check the Enable AutoPostBack checkbox from the productsByCategory DropDownList's smart tag.



Figure 14: Enable the AutoPostBack Feature for the productsByCategory DropDownList

# Step 3: Using a DetailsView to Display Details for

## the Selected Product

The final step is to display the details for the selected product in a DetailsView. To accomplish this, add a DetailsView to the page, set its ID property to ProductDetails, and create a new ObjectDataSource for it. Configure this ObjectDataSource to pull its data from the ProductsBLL class's

GetProductByProductID (*productID*) method using the selected value of the ProductsByCategory DropDownList for the value of the *productID* parameter.

Configure Data Source - productsByCategoryDataS	iource 🛛 🛛 🔀
Choose a Business Object	
Select a business object that can be used to retrieve or upda or App_Code directory for this application). Ghoose your business object:	te 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	
< But in the second sec	Next > Enish Cancel

Figure 15: Choose to Use the ProductsBLL Class

Configure	2 Data So	urce - O	bjectDat	aSourc	e1						? 🛛
Ļ	Define	Data M	lethods								
SELECT	UPDATE	INSERT	DELETE								
Choos metho Examp <u>C</u> hoos	e a methor id can retu ole: GetPro e a methor	d of the bu m a Data5 ducts(Int3 d:	usiness obje Set, DataRe 32 category	et that re ader, or Id), retu	eturns dat strongly-t rns a Data	ta to a typed o aSet.	ssociate ( collection.	with the	SELECT	' operatio	on. The
GetPr	oductByPro	ductID(In	it32 produci	tID), retu	urns Produ	ж 🛩	_		19		
GetPri GetPri GetPri	oducts(), re oducts(), re oductsByCa oductsBySc	ductID(In eturns Pro ategoryID applierID(I	t32 product ductsDataT (Int32 cate nt32 suppli	1D), reb able goryID), srID), ret	returns Produ returns Pr turns Prod	roducts JuctsDa	aTable sDataTab staTable	le			*
				<	Previous		<u>N</u> ext >		Einish		Cancel

Figure 16: Configure the ObjectDataSource to Use the GetProductByProductID (productID) Method

onfigure Data Source - ObjectDataSource	1 ?
Define Parameters	
The wizard has detected one or more parameters in method, choose a source for the parameter's value.	your SELECT method. For each parameter in the SELECT
Parangeers:	Parameter source:
Name Value productID ProductsByCategory,SelectedY	Control
	ControlID:
	ProductsByCategory 🛩
	DefaultValue:
< >	
	Show advanced properties
Method signature:	
GetProductByProductID(Int32 productID), returns I	ProductsDataTable
and produced in the produce of receiver	
<	yevious Next > Einish Cancel

#### Figure 17: Pull the productID Parameter Value from the ProductsByCategory DropDownList

You can choose to display any of the available fields in the ProductDetails DetailsView. I've opted to remove the ProductID, SupplierID, and CategoryID fields and reordered and formatted the remaining fields. In addition, I cleared out the DetailsView's Height and Width properties, allowing the DetailsView to expand to the width needed to best display its data rather than having it constrained to a specified size. The full markup appears below:

```
<asp:DetailsView ID="ProductDetails" runat="server"</pre>
  AutoGenerateRows="False" DataKeyNames="ProductID"
    DataSourceID="ObjectDataSource1" EnableViewState="False">
    <Fields>
        <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="QuantityPerUnit"</pre>
           HeaderText="Qty/Unit" SortExpression="QuantityPerUnit" />
        <asp:BoundField DataField="UnitPrice"</pre>
           DataFormatString="{0:c}" HeaderText="Price"
            HtmlEncode="False" SortExpression="UnitPrice" />
        <asp:BoundField DataField="UnitsInStock"</pre>
            HeaderText="UnitsInStock"
            SortExpression="Units In Stock" />
        <asp:BoundField DataField="UnitsOnOrder"</pre>
            HeaderText="UnitsOnOrder"
            SortExpression="Units On Order" />
        <asp:BoundField DataField="ReorderLevel"</pre>
            HeaderText="ReorderLevel" SortExpression="Reorder Level" />
        <asp:CheckBoxField DataField="Discontinued"</pre>
            HeaderText="Discontinued" SortExpression="Discontinued" />
```

#### </Fields> </asp:DetailsView>

Take a moment to try out the MasterDetailsDetails.aspx page in a browser. At first glance it may appear that everything is working as desired, but there's a subtle problem. When you choose a new category the ProductsByCategory DropDownList is updated to include those products for the selected category, but the ProductDetails DetailsView continued to show the previous product information. The DetailsView is updated when choosing a different product for the selected category. Furthermore, if you test thoroughly enough, you'll find that if you continually choose new categories (such as choosing Beverages from the Categories DropDownList, then Condiments, then Confections) every other category selection causes the ProductDetails DetailsView to be refreshed.

To help concretize this problem, let's look at a specific example. When you first visit the page the Beverages category is selected and the related products are loaded in the ProductsByCategory DropDownList. Chai is the selected product and its details are displayed in the ProductDetails DetailsView, as shown in Figure 18.



Figure 18: The Selected Product's Details are Displayed in a DetailsView

If you change the category selection from Beverages to Condiments, a postback occurs and the ProductsByCategory DropDownList is updated accordingly, but the DetailsView still displays details for Chai.



Figure 19: The Previously Selected Product's Details are Still Displayed

Picking a new product from the list refreshes the DetailsView as expected. If you pick a new category after changing the product, the DetailsView again won't refresh. However, if instead of choosing a new product you selected a new category, the DetailsView would refresh. What in the world is going on here?

The problem is a timing issue in the page's lifecycle. Whenever a page is requested it proceeds through a number of steps as its rendering. In one of these steps the ObjectDataSource controls check to see if any of their SelectParameters values have changed. If so, the data Web control bound to the ObjectDataSource knows that it needs to refresh its display. For example, when a new category is selected, the ProductsByCategoryDataSource ObjectDataSource detects that its parameter values have changed and the ProductsByCategory DropDownList rebinds itself, getting the products for the selected category.

The problem that arises in this situation is that the point in the page lifecycle that the ObjectDataSources check for changed parameters occurs *before* the rebinding of the associated data Web controls. Therefore, when selecting a new category the ProductsByCategoryDataSource ObjectDataSource detects a change in its parameter's value. The ObjectDataSource used by the ProductDetails DetailsView, however, doesn't note any such changes because the ProductsByCategory DropDownList has yet to be rebound. Later in the lifecycle the ProductsByCategory DropDownList rebinds to its ObjectDataSource, grabbing the products for the newly selected category. While the ProductsByCategory DropDownList's value has changed, the ProductDetails DetailsView's ObjectDataSource has already done its parameter value check; therefore, the DetailsView displays its previous results. This interaction is depicted in Figure 20.

Producto	View ProductDetails ObjectDataSource	ProductsInCategory DropDownList	ProductsInCategory ObjectDataSource	Categories DropDownList
The ObjectDataSources check to see if their parameter values have changed	Has the value of	parameter banged?	Has the pu value chi Ye	rameter ingeo?
		Parame has ch	tar value langad!	
The data Web controls whose ObjectDataSources's parameters have change are refreshed		Se	Select Results based on new category/ID parameter value	
		Results new ca parame		

#### Figure 20: The ProductsByCategory DropDownList Value Changes After the ProductDetails DetailsView's ObjectDataSource Checks for Changes

To remedy this we need to explicitly rebind the ProductDetails DetailsView after the ProductByCategory DropDownList has been bound. We can accomplish this by calling the ProductDetails DetailsView's DataBind() method when the ProductsByCategory DropDownList's DataBound event fires. Add the following event handler code to the MasterDetailsDetails.aspx page's code-behind class (refer to the "Programmatically Setting the ObjectDataSource's Parameter Values" for a discussion on how to add an event handler):

```
Protected Sub ProductsByCategory_DataBound(ByVal sender As Object, _
ByVal e As System.EventArgs) _
Handles ProductsByCategory.DataBound
ProductDetails.DataBind()
End Sub
```

After this explicit call to the ProductDetails DetailsView's DataBind() method has been added, the tutorial works as expected. Figure 21 highlights how this changed remedied our earlier problem.



Figure 21: The ProductDetails DetailsView is Explicitly Refreshed When the ProductsByCategory

# Summary

The DropDownList serves as an ideal user interface element for master/detail reports where there is a one-to-many relationship between the master and detail records. In the preceding tutorial we saw how to use a single DropDownList to filter the products displayed by the selected category. In this tutorial we replaced the GridView of products with a DropDownList, and used a DetailsView to display the details of the selected product. The concepts discussed in this tutorial can easily be extended to data models involving multiple one-to-many relationships, such as customers, orders, and order items. In general, you can always add a DropDownList for each of the "one" entities in the one-to-many relationships.

Happy Programming!

### **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.