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 Summary Information in the GridView's Footer

Download the code for this sample

Click here for the previous tutorial

Introduction

In addition to seeing each of the products' prices, units in stock, units on order, and reorder levels, a user might also be interested in aggregate information, such as the average price, the total number of units in stock, and so on. Such summary information is often displayed at the bottom of the report in a summary row. The GridView control can include a footer row into whose cells we can programmatically inject aggregate data.

This task presents us with three challenges:

- 1. Configuring the GridView to display its footer row
- 2. Determining the summary data; that is, how do we compute the average price or the total of the units in stock?
- 3. Injecting the summary data into the appropriate cells of the footer row

In this tutorial we'll see how to overcome these challenges. Specifically, we'll create a page that lists the categories in a drop-down list with the selected category's products displayed in a GridView. The GridView will include a footer row that shows the average price and total number of units in stock and on order for products in that category.

Back + O 🗈 🗟 🙆 🔎	Search 🏫 Favorites 🖉 😥 • 🖕 📝 •	D 0×	12. 119	
Vorking with Da	ata Tutorials Hom	e > <u>Customize</u> er	<u>d Formatting</u> > Sum	imary Data in
Home	Displaying Sum	mary Da	ata in the	
Basic Reporting	GridView's Foot	er		
Simple Display		States .		
Dedaration	Choose a category: Beverages			
Parameters	Product	Price Ur	nits In Stock Units	s On Order
and the second se	Chai	\$18.00	39	0
C. March C. March S. M.	una			
Setting Parameter	Chang	\$19.00	17	40
Setting Parameter Values	Chang Guaranà Fantástica	\$19.00 \$4.50	17 20	40 0
Setting Parameter Values Filtering Reports	Chang Guaranà Fantástica Sasquatch Ale	\$19.00 \$4.50 \$14.00	17 20 111	40 0 0
Setting Parameter Values Filtering Reports Filter by Drop-Down	Chang Guaraná Fantástica Sasquatch Ale Steeleye Stout	\$19.00 \$4.50 \$14.00 \$18.00	17 20 111 20	40 0 0
Setting Parameter Values Filtering Reports Filter by Drop-Down List	Chang Guaranà Fantástica Sasquatch Ale Steeleye Stout Côte de Blaye	\$19.00 \$4.50 \$14.00 \$18.00 \$263.50	17 20 111 20 17	40 0 0 0
Setting Parameter Values Filtering Reports Filter by Drop-Down List	Chang Guaranà Fantástica Sasquatch Ale Steeleye Stout Côte de Blaye Chartreuse verte	\$19.00 \$4.50 \$14.00 \$18.00 \$263.50 \$18.00	17 20 111 20 17 69	40 0 0 0 0
Setting Parameter Values Filtering Reports Filter by Drop-Down List Master-Details- Details	Chang Guaraná Fantástica Sasquatch Ale Steeleye Stout Côte de Blaye Chartreuse verte Ipoh Coffee	\$19.00 \$4.50 \$14.00 \$18.00 \$263.50 \$18.00 \$46.00	17 20 111 20 17 69 17	40 0 0 0 0 0 10
Setting Parameter Values Filtering Reports Filter by Drop-Down List Master-Details- Details	Chang Guaraná Fantástica Sasquatch Ale Steeleye Stout Côte de Blaye Chartreuse verte Ipoh Coffee Laughing Lumberjadk Lager	\$19.00 \$4.50 \$14.00 \$18.00 \$263.50 \$18.00 \$46.00 \$14.00	17 20 111 20 17 69 17 52	40 0 0 0 0 10
Setting Parameter Values Filtering Reports Filter by Drop-Down List Master-Details- Details Master/Detail Across	Chang Guaraná Fantástica Sasquatch Ale Steeleye Stout Côte de Blaye Chartreuse verte Ipoh Coffee Laughing Lumberjack Lager Outback Lager	\$19.00 \$4.50 \$14.00 \$18.00 \$263.50 \$18.00 \$46.00 \$14.00 \$15.00	17 20 111 20 17 69 17 52 15	40 0 0 0 0 10 0 10
Setting Parameter Values Filtering Reports Filter by Drop-Down List Master-Details- Details Master/Detail Across Two Pages	Chang Guaraná Fantástica Sasquatch Ale Steeleye Stout Côte de Blaye Chartreuse verte Ipoh Coffee Laughing Lumberjack Lager Outback Lager Rhonbrau Klosterbier	\$19.00 \$4.50 \$14.00 \$263.50 \$18.00 \$46.00 \$14.00 \$15.00 \$7.75	17 20 111 20 17 69 17 52 15 125	40 0 0 0 0 10 0 10 0
Setting Parameter Values Filtering Reports Filter by Drop-Down List Master-Details- Details Master/Detail Across Two Pages Details of Selected	Chang Guaraná Fantástica Sasquatch Ale Steeleye Stout Côte de Blaye Chartreuse verte Ipoh Coffee Laughing Lumberjack Lager Outback Lager Rhonbráu Klosterbier Lakkaliköön	\$19.00 \$4.50 \$14.00 \$263.50 \$18.00 \$46.00 \$14.00 \$15.00 \$7.75 \$18.00	17 20 111 20 17 69 17 52 15 125 57	40 0 0 0 0 10 0 10 0 0 0 0

Figure 1: Summary Information is Displayed in the GridView's Footer Row

This tutorial, with its category to products master/detail interface, builds upon the concepts covered in the earlier <u>Master/Detail Filtering With a DropDownList</u> tutorial. If you've not yet worked through the earlier tutorial, please do so before continuing on with this one.

Step 1: Adding the Categories DropDownList and Products GridView

Before concerning ourselves with adding summary information to the GridView's footer, let's first simply build the master/detail report. Once we've completed this first step, we'll look at how to include summary data.

Start by opening the SummaryDataInFooter.aspx page in the CustomFormatting folder. Add a DropDownList control and set its ID to Categories. Next, click on the Choose Data Source link from the DropDownList's smart tag and opt to add a new ObjectDataSource named CategoriesDataSource that invokes the CategoriesBLL class's GetCategories() method.

	Configuration	Wizard			20
۳ ا	noose a Data	a Source Ty	ре		
Where will th	he application	get data fron	n?		
Access Database	Database	Object	Site Map	XML File	
Connect to a	middle-tier busin	ess object or Da	ataSet in the Bin	or App_Code direc	ctory for the application.
Specify an ID f	for the data sour	ce:			
	aSource				
CategoriesDat					
CategoriesDat					
CategoriesDat					

Figure 2: Add a New ObjectDataSource Named CategoriesDataSource

Configure Data Source - CategoriesDataSource	? 🛛
Define Data Methods	
SELECT UPDATE INSERT DELETE	
Choose a method of the business object that returns data to associate with the method can return a DataSet, DataReader, or strongly-typed collection. Example: GetProducts(Int32 categoryId), returns a DataSet. @hoose a method:	SELECT operation. The
GetCategories(), returns CategoriesDataTable	
GetCategories(), returns CategoriesDataTable GetCategoryByCategoryID(Int32 categoryID), returns CategoriesDataTable	
GetCategories(), returns CategoriesDataTable	N N
< Previous Mext >	Enish Cancel

Figure 3: Have the ObjectDataSource Invoke the CategoriesBLL Class's GetCategories () Method

After configuring the ObjectDataSource, the wizard returns us to the DropDownList's Data Source Configuration wizard from which we need to specify what data field value should be displayed and which one should correspond to the value of the DropDownList's ListItems. Have the CategoryName field displayed and use the CategoryID as the value.

Data Source Configuration Wizard	2 🛛
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:	
EategoryID 💌	
CstegoryID CategoryName Description	
Refresh Schema	
	OK Cancel

Figure 4: Use the CategoryName and CategoryID Fields as the Text and Value for the ListItems, Respectively

At this point we have a DropDownList (Categories) that lists the categories in the system. We now need to add a GridView that lists those products that belong to the selected category. Before we do, though, take a

moment to check the Enable AutoPostBack checkbox in the DropDownList's smart tag. As discussed in the *Master/Detail Filtering With a DropDownList* tutorial, by setting the DropDownList's AutoPostBack property to True the page will be posted back each time the DropDownList value is changed. This will cause the GridView to be refreshed, showing those products for the newly selected category. If the AutoPostBack property is set to False (the default), changing the category won't cause a postback and therefore won't update the listed products.



Figure 5: Check the Enable AutoPostBack Checkbox in the DropDownList's Smart Tag

Add a GridView control to the page in order to display the products for the selected category. Set the GridView's ID to ProductsInCategory and bind it to a new ObjectDataSource named ProductsInCategoryDataSource.

Data Source Co	onfiguration	Wizard				? 🛛
the chi	oose a Data	Source Ty	pe			
₩here will the	e application	get data fron	n?			
Access Database	Database	Object	Site Map	XML File		
Connect to a m	iddle-tier busini	ess object or Da	ataSet in the Bin	or App_Code d	irectory for the applic	ation.
Specify an ID fo	or the data sour	ce:				
ProductsInCate	goryDataSourc	e				
					ок	Cancel

Figure 6: Add a New ObjectDataSource Named ProductsInCategoryDataSource

Configure the ObjectDataSource so that it invokes the ProductsBLL class's

GetProductsByCategoryID(*categoryID*) method.

Configure	e Data Source - ProductsInCategoryDataSource	2 🗙
Ļ	Define Data Methods	
SELECT	UPDATE INSERT DELETE	
Choos metho Exam Choos	se a method of the business object that returns data to associate with the SELECT operation. The od can return a DataSet, DataReader, or strongly-typed collection. ple: GetProducts(Int32 categoryId), returns a DataSet. se a method:	
GetPr	roductsByCategoryID(Int32 categoryID), returns Prc 👻	
GetPr GetPr GetPr	oductByProductID(Int32 productID), returns ProductsDataTable oducts(), returns ProductsDataTable oductsByCategoryID(Int32 categoryID), returns ProductsDataTable oductsBySupplierID(Int32 supplierID), returns ProductsDataTable	
	< Previous Next > Einish Cancel	

Figure 7: Have the ObjectDataSource Invoke the GetProductsByCategoryID (*categoryID*) Method

Since the GetProductsByCategoryID(*categoryID*) method takes in an input parameter, in the final step of the wizard we can specify the source of the parameter value. In order to display those products from the selected category, have the parameter pulled from the Categories DropDownList.

onfigure Data Source - ProductsIn(CategoryDataSource
Define Parameters	
The wizard has detected one or more paran method, choose a source for the parameter	neters in your SELECT method. For each parameter in the SELECT 's value.
Paramgters:	Parameter gource:
Name Value	Control
categoryID Categories.SelectedValu	ControlID:
	Categories
	DefaultValue:
	Show advanced properties
Method signature:	
GetProductsByCategoryID(Int32 category)	D), returns ProductsDataTable
	Carca

Figure 8: Get the *category1D* Parameter Value from the Selected Categories DropDownList

After completing the wizard the GridView will have a BoundField for each of the product properties. Let's clean up these BoundFields so that only the ProductName, UnitPrice, UnitsInStock, and UnitsOnOrder

BoundFields are displayed. Feel free to add any field-level settings to the remaining BoundFields (such as formatting the UnitPrice as a currency). After making these changes, the GridView's declarative markup should look similar to the following:

```
<asp:GridView ID="ProductsInCategory" runat="server"</pre>
    AutoGenerateColumns="False"
    DataKeyNames="ProductID"
    DataSourceID="ProductsInCategoryDataSource"
    EnableViewState="False">
    <Columns>
        <asp:BoundField DataField="ProductName" HeaderText="Product"</pre>
          SortExpression="ProductName" />
        <asp:BoundField DataField="UnitPrice" DataFormatString="{0:c}"
          HeaderText="Price"
            HtmlEncode="False" SortExpression="UnitPrice">
            <ItemStyle HorizontalAlign="Right" />
        </asp:BoundField>
        <asp:BoundField DataField="UnitsInStock"</pre>
          HeaderText="Units In Stock" SortExpression="UnitsInStock">
            <ItemStyle HorizontalAlign="Right" />
        </asp:BoundField>
        <asp:BoundField DataField="UnitsOnOrder"</pre>
          HeaderText="Units On Order" SortExpression="UnitsOnOrder">
            <ItemStyle HorizontalAlign="Right" />
        </asp:BoundField>
    </Columns>
</asp:GridView>
```

At this point we have a fully functioning master/detail report that shows the name, unit price, units in stock, and units on order for those products that belong to the selected category.

Gle Edit yow Fyronitas Iools Othech • O • e e e for P Inner e Http://localhost:2302/Code/Cu	Bib Search 🚖 Parontes 🕘 😒 + 🖕 ston Formatting Summery Ostals Poster as	w • 🗭 🔹	0×11 10	- 8
Norking with Da	ata Tutorials	Home > Cus Data in Foot	tomized Format ter	ting > Summary
Home Basic Reporting Simple Display	Displaying S GridView's F	ummary ooter	y Data in	the
Dedarative Parameters	Choose a category: Beve Product	rages × Price	Units In	Units On
Setting Parameter Values	Chai Chang	\$18.00 \$19.00	39 17	0 40
Filtering Reports	Guaraná Fantástica	\$4.50	20	0
Filter by Drop-Down List	Sasquatch Ale Steeleye Stout	\$14.00 \$18.00	111 20	0
Master-Details- Details	Côte de Blaye Chartreuse verte	\$263.50 \$18.00	17 69	0
Master/Detail Across Two Pages	Laughing Lumberjack Lager	\$14.00	52	0
Details of Selected Row	Outback Lager Rhönbräu Klosterbier	\$15.00 \$7.75	15 125	10
Customized	Lakkalikööri	\$18.00	57	0
Customized	Landervoor	protos		Local intravet

Figure 9: Get the categoryID Parameter Value from the Selected Categories DropDownList

Step 2: Displaying a Footer in the GridView

The GridView control can display both a header and footer row. These rows are displayed depending on the values of the ShowHeader and ShowFooter properties, respectively, with ShowHeader defaulting to True and ShowFooter to False. To include a footer in the GridView simply set its ShowFooter property to True.

	lode - Mic	rosoft Visual	Studio					
Ele	gdit y	ew Website	Build Debug	Fgrmat Layout	Iook	Window Community	Help Addins	
				B I U	12	E - IE IE I		
Sel.	Custom	ormattnFoo	ter.aspx*		• ×	Properties		• 0 ×
100					6	ProductsInCategory :	System.Web.UI	.WebConb +
8	Cho	ose a cate	gory: Databou	und 💌		雄 21 回 🖌 🖂		_
	Ob	ectDataSourc	e - CategoriesData	Source		PageSize	10	^
			Units In	Units On		RowHeaderColumn		1
	Pro	oduct Price	Stock	Order		SelectedIndex	-1	
	abo	\$0.00	0	0		E SelectedRowStyle		_
	abo	\$0.10	1	1	10	ShowFooter	True	~
	abo	\$0.20	2	2		SkinID	High	
	abo	\$0.30	3	3		Tabāndex	0	
	abo	\$0.40	4	4		ToolTip UseAccessibleHeader	True	1.0
						Visible	True	
	E Ob	jectDataSourc	e - ProductsInCate	egoryDataSource		ShowFooter		
						Whether to the show the	control's foote	r.
	<				>			
	🖬 Design	E Source	e casp:gridvie	evv#productsinc>		Soluti Prope	Serve	Eg Class
	👌 Error List	Uutput	Find Results 1					-
Rea	dy							

Figure 10: Set the GridView's showFooter Property to True

The footer row has a cell for each of the fields defined in the GridView; however, these cells are empty by default. Take a moment to view our progress in a browser. With the ShowFooter property now set to True, the GridView includes an empty footer row.

Untitled Page - Alkcrosoft Interne Die Edit View Favorites Tools	t Explorer Help Search		2 1 II	
Address Attp://localhost.2302/Code/O	stomFormatting/SummaryDataInPooter.a	spor	Bel 148	💌 🛃 Go
Working with Da	ata Tutorials	<u>Home > Cu</u> Data in Foo	stomized Format ter	tting > Summary
Home Basic Reporting Simple Display	Displaying S GridView's F	ummar ooter	y Data in	the
Declarative Parameters	Choose a category: Bew Product	Price	Units In Stock	Units On
Setting Parameter Values	Chai Chang	\$18.00 \$19.00	39 17	0 40
Filtering Reports	Guaraná Fantástica	\$4.50	20	o
Filter by Grop-Down List	Sasquatch Ale Steeleye Stout	\$14.00 \$18.00	111 20	0
Master-Details- Details	Cote de Blaye Chartreuse verte	\$263.50 \$18.00	17 69 17	0
Master/Detail Across Two Pages	Laughing Lumberjack	\$14.00	52	0
Details of Selected Row	Outback Lager Rhönbrau Klösterbier	\$15.00 \$7.75	15 125	10 0
Customized Formatting	Lakkalikoon	\$18.00	57	0
Done				S Local Intranet

Figure 11: The GridView Now Includes a Footer Row

The footer row in Figure 11 doesn't stand out, as it has a white background. Let's create a <code>FooterStyle CSS</code> class in <code>Styles.css</code> that specifies a dark red background and then configure the <code>GridView.skin</code> Skin file in the <code>DataWebControls</code> Theme to assign this CSS class to the <code>GridView's FooterStyle's CssClass</code> property. If you need to brush up on Skins and Themes, refer back to the <u>Displaying Data With the ObjectDataSource</u> tutorial.

Start by adding the following CSS class to Styles.css:

```
.FooterStyle
{
    background-color: #a33;
    color: White;
    text-align: right;
}
```

The FooterStyle CSS class is similar in style to the HeaderStyle class, although the HeaderStyle's background color is subtlety darker and its text is displayed in a bold font. Furthermore, the text in the footer is right-aligned whereas the header's text is centered.

Next, to associate this CSS class with every GridView's footer, open the GridView.skin file in the DataWebControls Theme and set the FooterStyle's CssClass property. After this addition the file's markup should look like:

As the screen shot below shows, this change makes the footer stand out more clearly.

Untitled Page - Microsoft Interne Pla Edit View Excepter Tools	t Explorer			
O Back • O • I I I I I I I I I I I I I I I I I	Search 👷 Favorites 🕢 🔯 - 🖕	in • □ •	● > 以 服	
Working with Da	ata Tutorials	Home > Cu: Data in Foo	tomized Format ter	tting > Summary
Home Basic Reporting Simple Display	Displaying S GridView's F	ummary ooter	y Data in	the
Declarative Parameters	Product	Price	Units In	Units On
Setting Parameter Values	Chai Chang	\$18.00 \$19.00	39 17	0 40
Filtering Reports	Guaraná Fantástica	\$4.50	20	0
Filter by Drop-Down List	Sasquatch Ale Steeleye Stout	\$14.00 \$18.00	111 20	0
Master-Details- Details	Côte de Blaye Chartreuse verte	\$263.50 \$18.00	17 69	0
Master/Detail Across Two Pages	Laughing Lumberjack Lager	\$14.00	52	0
Details of Selected Row	Outback Lager Rhönbrau Klosterbier	\$15.00 \$7.75	15 125	10 0
Customized Formatting	Lakkalkööri	\$18.00	57	0
Done				S Local Intranet

Figure 12: The GridView's Footer Row Now Has a Reddish Background Color

Step 3: Computing the Summary Data

With the GridView's footer displayed, the next challenge facing us is how to compute the summary data. There are two ways to compute this aggregate information:

1. Through a SQL query – we could issue an additional query to the database to compute the summary data for a particular category. SQL includes a number of aggregate functions along with a GROUP BY clause to specify the data over which the data should be summarized. The following SQL query would bring back the needed information:

```
SELECT CategoryID, AVG(UnitPrice), SUM(UnitsInStock),
SUM(UnitsOnOrder)
FROM Products
WHERE CategoryID = categoryID
GROUP BY CategoryID
```

Of course you wouldn't want to issue this query directly from the SummaryDataInFooter.aspx page, but rather by creating a method in the ProductsTableAdapter and the ProductsBLL.

2. Compute this information as it's being added to the GridView – as discussed in <u>Custom Formatting</u> <u>Based Upon Data</u> tutorial, the GridView's RowDataBound event handler fires once for each row being added to the GridView after its been databound. By creating an event handler for this event we can keep a running total of the values we want to aggregate. After the last data row has been bound to the GridView we have the totals and the information needed to compute the average.

I typically employ the second approach as it saves a trip to the database and the effort needed to implement the summary functionality in the Data Access Layer and Business Logic Layer, but either approach would suffice. For this tutorial let's use the second option and keep track of the running total using the RowDataBound event handler.

Create a RowDataBound event handler for the GridView by selecting the GridView in the Designer, clicking the lightning bolt icon from the Properties window, and double-clicking the RowDataBound event. Alternatively, you can select the GridView and its RowDataBound event from the drop-down lists at the top of the ASP.NET code-behind class file. This will create a new event handler named ProductsInCategory_RowDataBound in the SummaryDataInFooter.aspx page's code-behind class.

```
Protected Sub ProductsInCategory_RowDataBound(ByVal sender As Object, _
ByVal e As System.Web.UI.WebControls.GridViewRowEventArgs) _
Handles ProductsInCategory.RowDataBound
```

End Sub

In order to maintain a running total we need to define variables outside of the scope of the event handler. Create the following four page-level variables:

- _totalUnitPrice, of type Decimal
- _totalNonNullUnitPriceCount, of type Integer
- _totalUnitsInStock, of type Integer
- _totalUnitsOnOrder, of type Integer

Next, write the code to increment these three variables for each data row encountered in the RowDataBound

event handler.

```
' Class-scope, running total variables...
Dim _totalUnitPrice As Decimal = 0
Dim _totalNonNullUnitPriceCount As Integer = 0
Dim totalUnitsInStock As Integer = 0
Dim totalUnitsOnOrder As Integer = 0
Protected Sub ProductsInCategory RowDataBound(ByVal sender As Object,
  ByVal e As System.Web.UI.WebControls.GridViewRowEventArgs)
  Handles ProductsInCategory.RowDataBound
    If e.Row.RowType = DataControlRowType.DataRow Then
        ' Reference the ProductsRow via the e.Row.DataItem property
        Dim product As Northwind.ProductsRow = ____
          CType(CType(e.Row.DataItem, System.Data.DataRowView).Row, _
          Northwind.ProductsRow)
        ' Increment the running totals (if they're not NULL!)
        If Not product.IsUnitPriceNull() Then
            _totalUnitPrice += product.UnitPrice
            totalNonNullUnitPriceCount += 1
        End If
        If Not product.IsUnitsInStockNull() Then
             totalUnitsInStock += product.UnitsInStock
        End If
        If Not product.IsUnitsOnOrderNull() Then
             totalUnitsOnOrder += product.UnitsOnOrder
        End If
    ElseIf e.Row.RowType = DataControlRowType.Footer Then
        ' Determine the average UnitPrice
        Dim avgUnitPrice As Decimal = totalUnitPrice /
          CType( totalNonNullUnitPriceCount, Decimal)
        ' Display the summary data in the appropriate cells
        e.Row.Cells(1).Text = "Avg.: " & avgUnitPrice.ToString("c")
        e.Row.Cells(2).Text = "Total: " & _totalUnitsInStock.ToString()
        e.Row.Cells(3).Text = "Total: " & totalUnitsOnOrder.ToString()
    End If
End Sub
```

The RowDataBound event handler starts by ensuring that we're dealing with a DataRow. Once that's been established, the Northwind.ProductsRow instance that was just bound to the GridViewRow object in e.Row is stored in the variable product. Next, running total variables are incremented by the current product's corresponding values (assuming that they don't contain a database NULL value). We keep track of both the running UnitPrice total and the number of non-NULL UnitPrice records because the average price is the quotient of these two numbers.

Step 4: Displaying the Summary Data in the Footer

With the summary data totaled, the last step is to display it in the GridView's footer row. This task, too, can be accomplished programmatically through the RowDataBound event handler. Recall that the RowDataBound event handler fires for *every* row that's bound to the GridView, including the footer row. Therefore, we can augment our event handler to display the data in the footer row using the following code:

Protected Sub ProductsInCategory_RowDataBound(ByVal sender As Object, _

```
ByVal e As System.Web.UI.WebControls.GridViewRowEventArgs)
Handles ProductsInCategory.RowDataBound
If e.Row.RowType = DataControlRowType.DataRow Then
... Increment the running totals ...
ElseIf e.Row.RowType = DataControlRowType.Footer
... Display the summary data in the footer ...
End If
End Sub
```

Since the footer row is added to the GridView after all of the data rows have been added, we can be confident that by the time we're ready to display the summary data in the footer the running total calculations will have completed. The last step, then, is to set these values in the footer's cells.

To display text in a particular footer cell, use e.Row.Cells(*index*).Text = *value*, where the Cells indexing starts at 0. The following code computes the average price (the total price divided by the number of products) and displays it along with the total number of units in stock and units on order in the appropriate footer cells of the GridView.

Figure 13 shows the report after this code has been added. Note how the ToString("c") causes the average price summary information to be formatted like a currency.

Untitled Page - Microsoft Interne 54 54 59 Payor Revortes Jools Stack - O R R & P Construction Construction	l Explorer Hilp Saardi 📌 Favorites 🖉 🎒 🎄 🕅 Istoniformatting(SummaryData)nFooter-aspx	• 🗆 💿 🛰	82. 119	
Working with Da	ata Tutorials 🛛 🙀	<u>ne > Customize</u> ter	d Formatting > Su	mmary Data in
Home Basic Reporting Simple Display	Displaying Sum GridView's Foo	imary Da ter	ata in the	
Dedarative	Choose a category: Beverage	s 🕐		
Parameters	Product	Price UI	nits In StockUni	ts on order
Setting Parameter	Chang	\$10.00	17	40
Values	Guaraná Fantástica	\$4.50	20	10
Filtering Reports	Sasquatch Ale	\$14.00	111	0
E-Investment of the second	Steeleve Stout	\$18.00	20	0
List	Côte de Blaye	\$263.50	17	0
and the second se	Chartreuse verte	\$18.00	69	0
Plaster-Detars-	Ipoh Coffee	\$45.00	17	10
Conclusion (1997)	Laughing Lumberjack Lager	\$14.00	52	0
Master/Detail Across	Outback Lager	\$15.00	15	10
Two Pages	Rhönbräu Klosterbier	\$7.75	125	0
Details of Selected	Lakkaliköön	\$18.00	57	0
Plow	A CONTRACTOR OF	4Vg.: \$37.98	Total: 559	Total: 60
Customized				
Done			THE R. LEWIS	Corol Infranet

Figure 13: The GridView's Footer Row Now Has a Reddish Background Color

Summary

Displaying summary data is a common report requirement, and the GridView control makes it easy to include such information in its footer row. The footer row is displayed when the GridView's ShowFooter property is set to True and can have the text in its cells set programmatically through the RowDataBound event handler. Computing the summary data can either be done by re-querying the database or by using code in the ASP.NET page's code-behind class to programmatically compute the summary data.

This tutorial concludes our examination of custom formatting with the GridView, DetailsView, and FormView controls. Our next tutorial kicks off our exploration of inserting, updating, and deleting data using these same controls.

Happy Programming!

Click here for the next tutorial

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>.