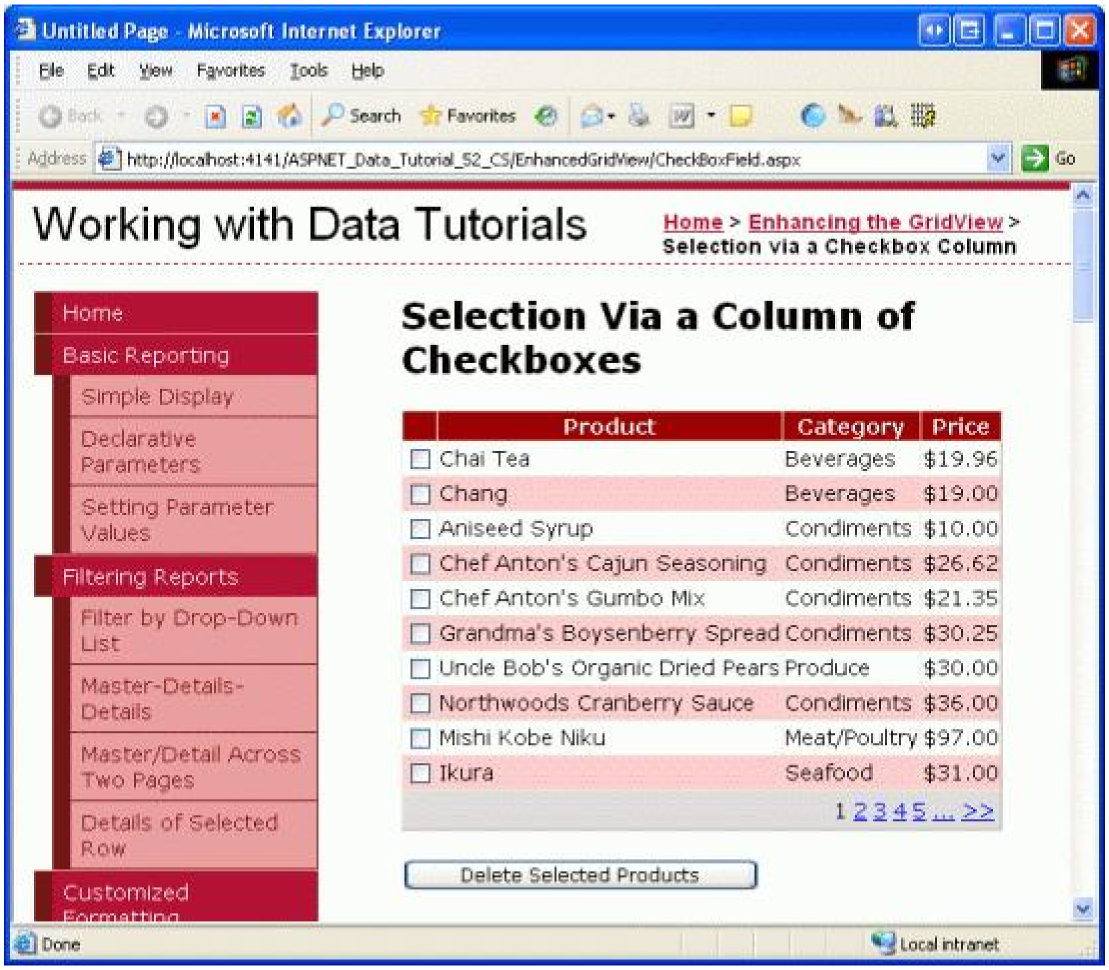
Working with Data in ASP.NET 2.0 :: Adding a GridView Column of Checkboxes

Introduction

In the preceding tutorial we examined how to add a column of radio buttons to the GridView for the purpose of selecting a particular record. A column of radio buttons is a suitable user interface when the user is limited to choosing at most one item from the grid. At times, however, we may want to allow the user to pick an arbitrary number of items from the grid. Web-based email clients, for example, typically display the list of messages with a column of checkboxes. The user can select an arbitrary number of messages and then perform some action, such as moving the emails to another folder or deleting them.

In this tutorial we will see how to add a column of checkboxes and how to determine what checkboxes were checked on postback. In particular, we'll build an example that closely mimics the web-based email client user interface. Our example will include a paged GridView listing the products in the `Products` database table with a checkbox in each row (see Figure 1). A "Delete Selected Products" button, when clicked, will delete those products selected.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:4141/ASPNET_Data_Tutorial_52_CS/EnhancedGridView/CheckBoxField.aspx
Go
Working with Data Tutorials
Home > Enhancing the GridView > Selection via a Checkbox Column
Home
Basic Reporting
Simple Display
Declarative Parameters
Setting Parameter Values
Filtering Reports
Filter by Drop-Down List
Master-Details-Details
Master/Detail Across Two Pages
Details of Selected Row
Customized Formatting
Selection Via a Column of Checkboxes
Delete Selected Products
Done
Local intranet

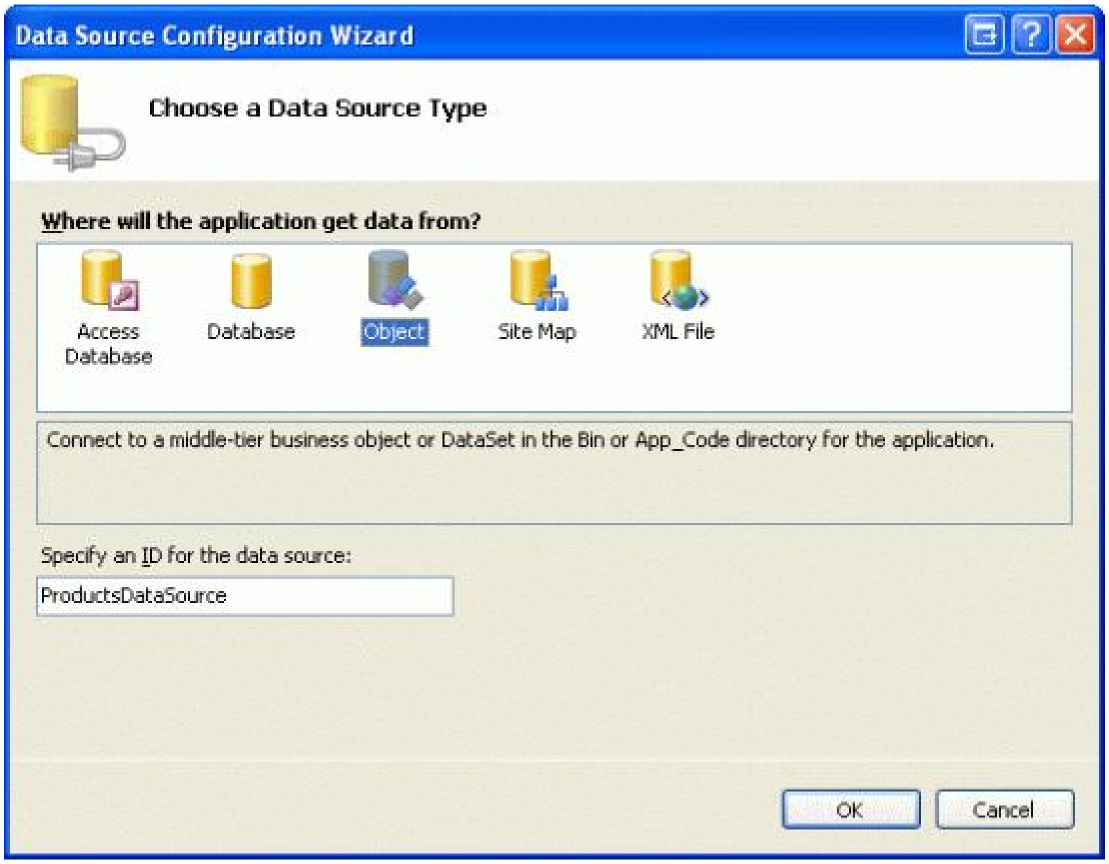
	Product	Category	Price
☐	Chai Tea	Beverages	$19.96
☐	Chang	Beverages	$19.00
☐	Aniseed Syrup	Condiments	$10.00
☐	Chef Anton's Cajun Seasoning	Condiments	$26.62
☐	Chef Anton's Gumbo Mix	Condiments	$21.35
☐	Grandma's Boysenberry Spread	Condiments	$30.25
☐	Uncle Bob's Organic Dried Pears	Produce	$30.00
☐	Northwoods Cranberry Sauce	Condiments	$36.00
☐	Mishi Kobe Niku	Meat/Poultry	$97.00
☐	Ikura	Seafood	$31.00

1 2 3 4 5 ... >>

Figure 1: Each Product Row Includes a Checkbox

Step 1: Adding a Paged GridView that Lists Product Information

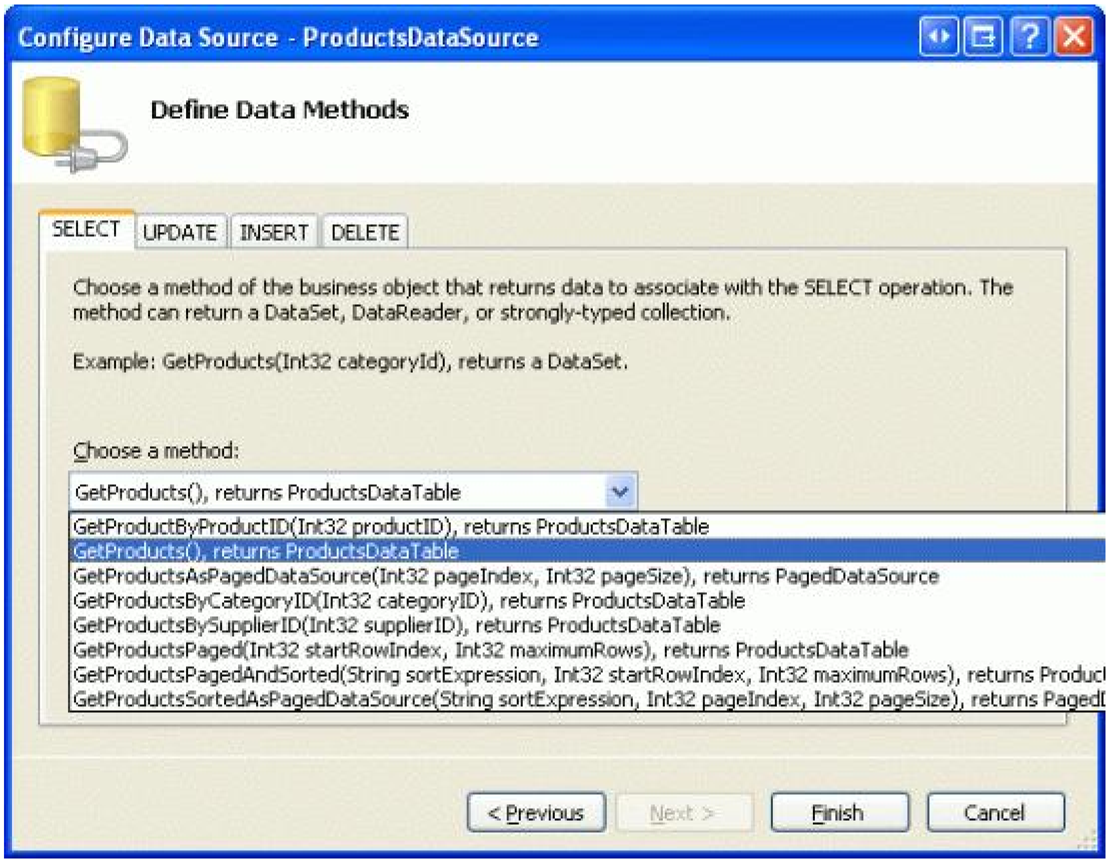
Before we worry about adding a column of checkboxes, let's first focus on listing the products in a GridView that supports paging. Start by opening the `CheckBoxField.aspx` page in the `EnhancedGridView` folder and drag a GridView from the Toolbox onto the Designer, setting its `ID` to `Products`. Next, choose to bind the GridView to a new ObjectDataSource named `ProductsDataSource`. Configure the ObjectDataSource to use the `ProductsBLL` class, calling the `GetProducts()` method to return the data. Since this GridView will be read-only, set the drop-down lists in the UPDATE, INSERT, and DELETE tabs to "(None)".


Data Source Configuration Wizard
Choose a Data Source Type
Where will the application get data from?
Access Database
Database
Object
Site Map
XML File
OK
Cancel

Connect to a middle-tier business object or DataSet in the Bin or App_Code directory for the application.

Specify an ID for the data source: ProductsDataSource

Figure 2: Create a New ObjectDataSource Named `ProductsDataSource`



Choose a method of the business object that returns data to associate with the SELECT operation. The method can return a DataSet, DataReader, or strongly-typed collection.

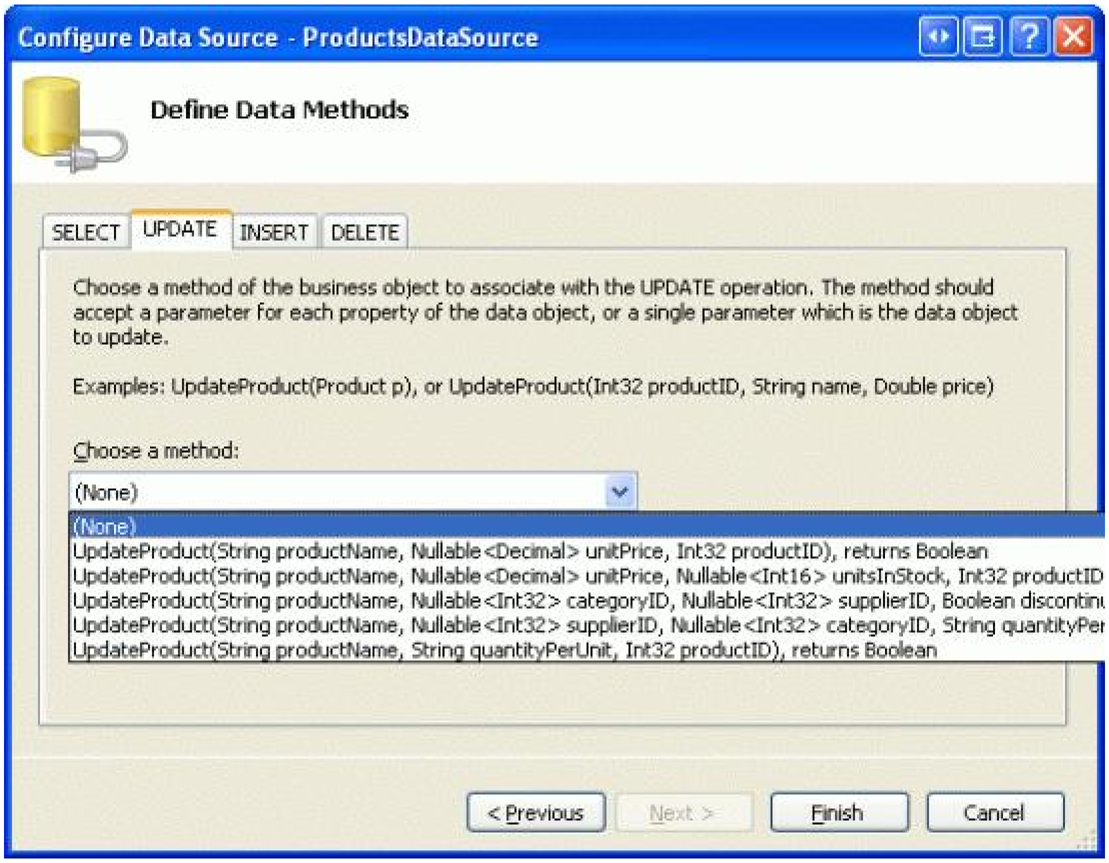
Example: GetProducts(Int32 categoryId), returns a DataSet.

Choose a method:

GetProducts(), returns ProductsDataTable

```
GetProductByProductID(Int32 productID), returns ProductsDataTable
GetProducts(), returns ProductsDataTable
GetProductsAsPagedDataSource(Int32 pageIndex, Int32 pageSize), returns PagedDataSource
GetProductsByCategoryID(Int32 categoryID), returns ProductsDataTable
GetProductsBySupplierID(Int32 supplierID), returns ProductsDataTable
GetProductsPaged(Int32 startRowIndex, Int32 maximumRows), returns ProductsDataTable
GetProductsPagedAndSorted(String sortExpression, Int32 startRowIndex, Int32 maximumRows), returns Produc
GetProductsSortedAsPagedDataSource(String sortExpression, Int32 pageIndex, Int32 pageSize), returns Paged
```

Figure 3: Configure the ObjectDataSource to Retrieve Data Using the `GetProducts()` Method



Choose a method of the business object to associate with the UPDATE operation. The method should accept a parameter for each property of the data object, or a single parameter which is the data object to update.

Examples: UpdateProduct(Product p), or UpdateProduct(Int32 productID, String name, Double price)

Choose a method:

(None)

(None)
UpdateProduct(String productName, Nullable<Decimal> unitPrice, Int32 productID), returns Boolean
UpdateProduct(String productName, Nullable<Decimal> unitPrice, Nullable<Int16> unitsInStock, Int32 productID
UpdateProduct(String productName, Nullable<Int32> categoryID, Nullable<Int32> supplierID, Boolean discontin
UpdateProduct(String productName, Nullable<Int32> supplierID, Nullable<Int32> categoryID, String quantityPer
UpdateProduct(String productName, String quantityPerUnit, Int32 productID), returns Boolean

Figure 4: Set the Drop-Down Lists in the UPDATE, INSERT, and DELETE Tabs to "(None)"

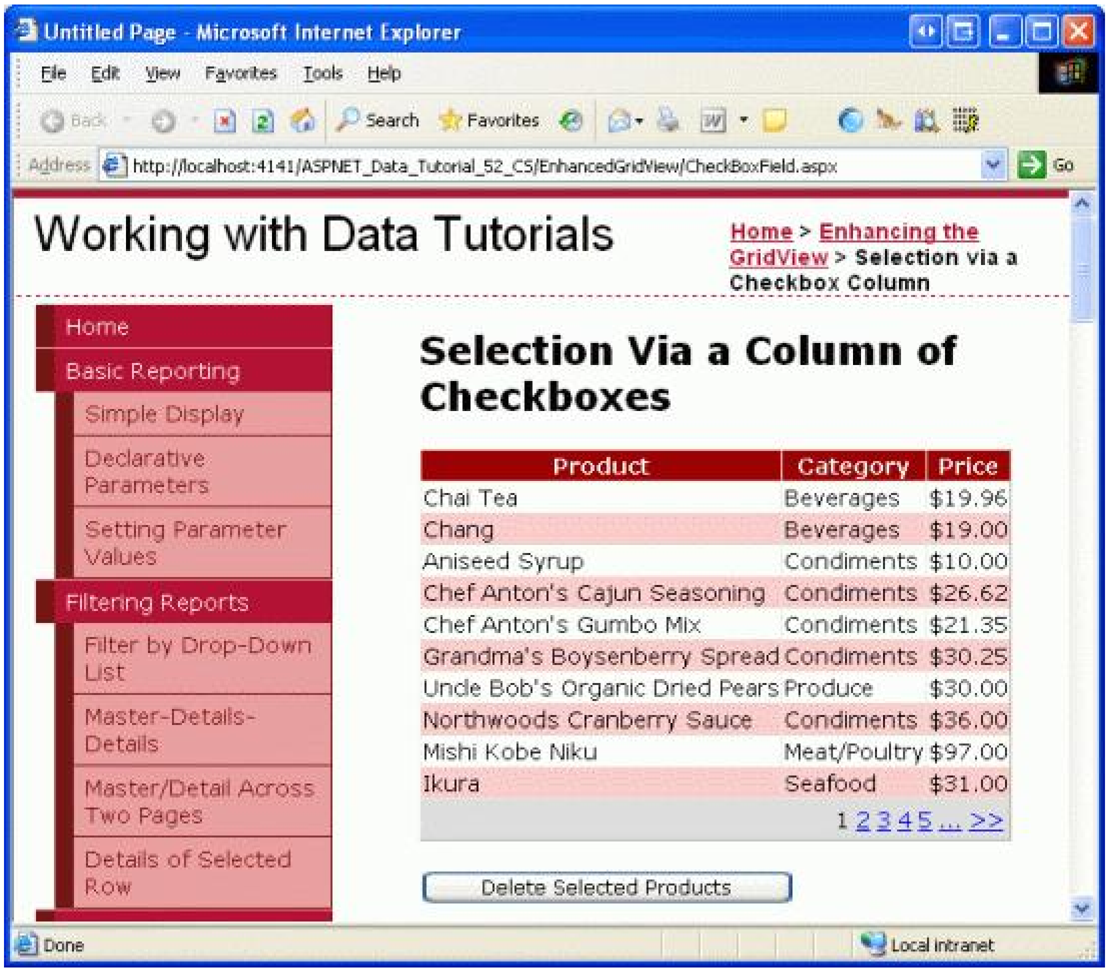
After completing the Configure Data Source wizard, Visual Studio will automatically create BoundColumns and a CheckBoxColumn for the product-related data fields. Like we did in the previous tutorial, remove all but the `ProductName`, `CategoryName`, and `UnitPrice` BoundFields, and change the `HeaderText` properties to "Product", "Category", and "Price". Configure the `UnitPrice` BoundField so that its value is formatted as a currency. Also configure the GridView to support paging by checking the "Enable Paging" checkbox from the smart tag.

Let's also add the user interface for deleting the selected products. Add a Button Web control beneath the GridView, setting its `ID` to `DeleteSelectedProducts` and its `Text` property to "Delete Selected Products". Rather than actually deleting products from the database, for this example we'll just display a message stating the products that would have been deleted. To accommodate this, add a Label Web control beneath the Button. Set its ID to `DeleteResults`, clear out its `Text` property, and set its `Visible` and `EnableViewState` properties to `False`.

After making these changes, the GridView, ObjectDataSource, Button, and Label's declarative markup should similar to the following:

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

    <asp:ObjectDataSource ID="ProductsDataSource" runat="server"
        OldValuesParameterFormatString="original_{0}"
        SelectMethod="GetProducts" TypeName="ProductsBLL">
    </asp:ObjectDataSource>
</p>
<p>
    <asp:Button ID="DeleteSelectedProducts" runat="server"
        Text="Delete Selected Products" />
</p>
<p>
    <asp:Label ID="DeleteResults" runat="server" EnableViewState="False"
        Visible="False"></asp:Label>
</p>
```

Take a moment to view the page in a browser (see Figure 5). At this point you should see the name, category, and price of the first ten products.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:4141/ASPNET_Data_Tutorial_52_CS/EnhancedGridView/CheckBoxField.aspx
Go
Working with Data Tutorials
Home > Enhancing the GridView > Selection via a Checkbox Column
Home
Basic Reporting
Simple Display
Declarative Parameters
Setting Parameter Values
Filtering Reports
Filter by Drop-Down List
Master-Details-Details
Master/Detail Across Two Pages
Details of Selected Row
Selection Via a Column of Checkboxes
Delete Selected Products
Done
Local intranet

Product	Category	Price
Chai Tea	Beverages	$19.96
Chang	Beverages	$19.00
Aniseed Syrup	Condiments	$10.00
Chef Anton's Cajun Seasoning	Condiments	$26.62
Chef Anton's Gumbo Mix	Condiments	$21.35
Grandma's Boysenberry Spread	Condiments	$30.25
Uncle Bob's Organic Dried Pears	Produce	$30.00
Northwoods Cranberry Sauce	Condiments	$36.00
Mishi Kobe Niku	Meat/Poultry	$97.00
Ikura	Seafood	$31.00
		1 2 3 4 5 ... >>

Figure 5: The Name, Category, and Price of the First Ten Products are Listed

Step 2: Adding a Column of Checkboxes

Since ASP.NET 2.0 includes a CheckBoxField, one might think that it could be used to add a column of checkboxes to a GridView. Unfortunately, that is not the case, as the CheckBoxField is designed to work with a Boolean data field. That is, in order to use the CheckBoxField we must specify the underlying data field whose value is consulted to determine whether the rendered checkbox is checked. We cannot use the CheckBoxField to just include a column of unchecked checkboxes.

Instead, we must add a TemplateField and add a CheckBox Web control to its `ItemTemplate`. Go ahead and add a TemplateField to the `Products` GridView and make it the first (far-left) field. From the GridView's smart tag, click on the "Edit Templates" link and then drag a CheckBox Web control from the Toolbox into the `ItemTemplate`. Set this CheckBox's `ID` property to `ProductSelector`.

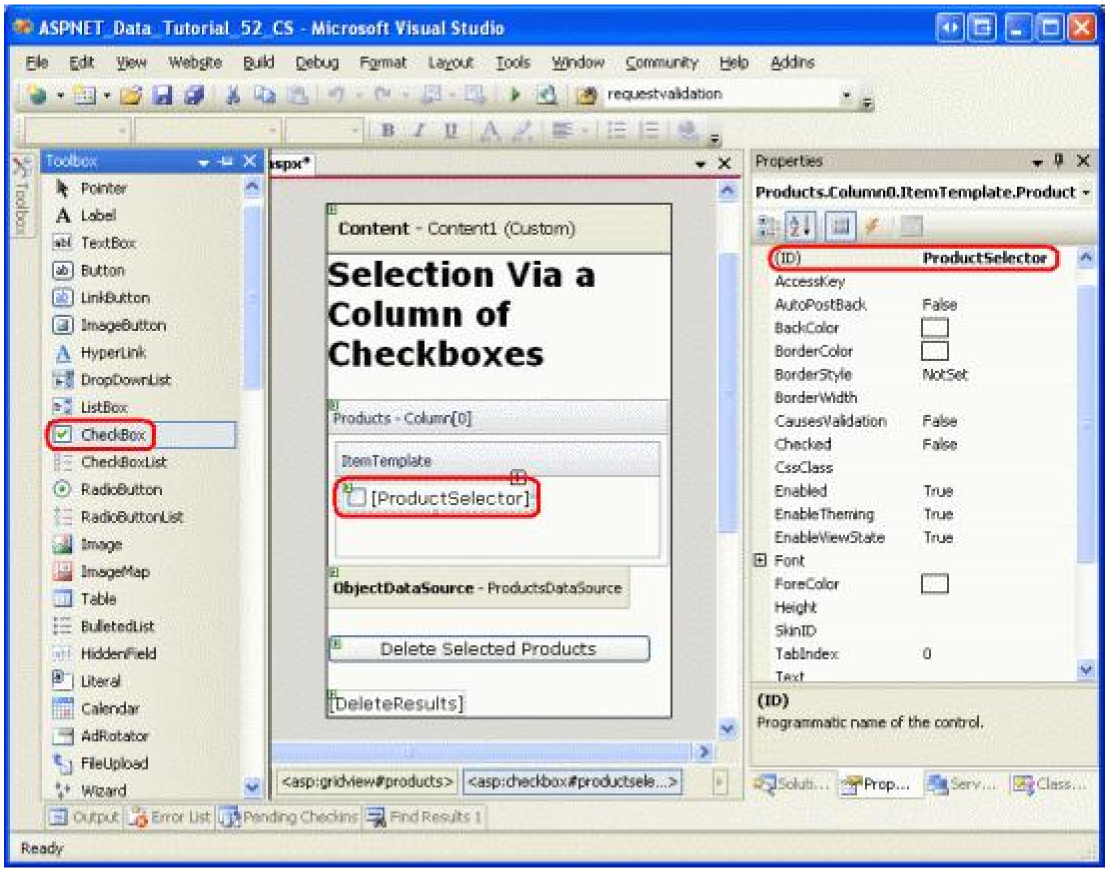
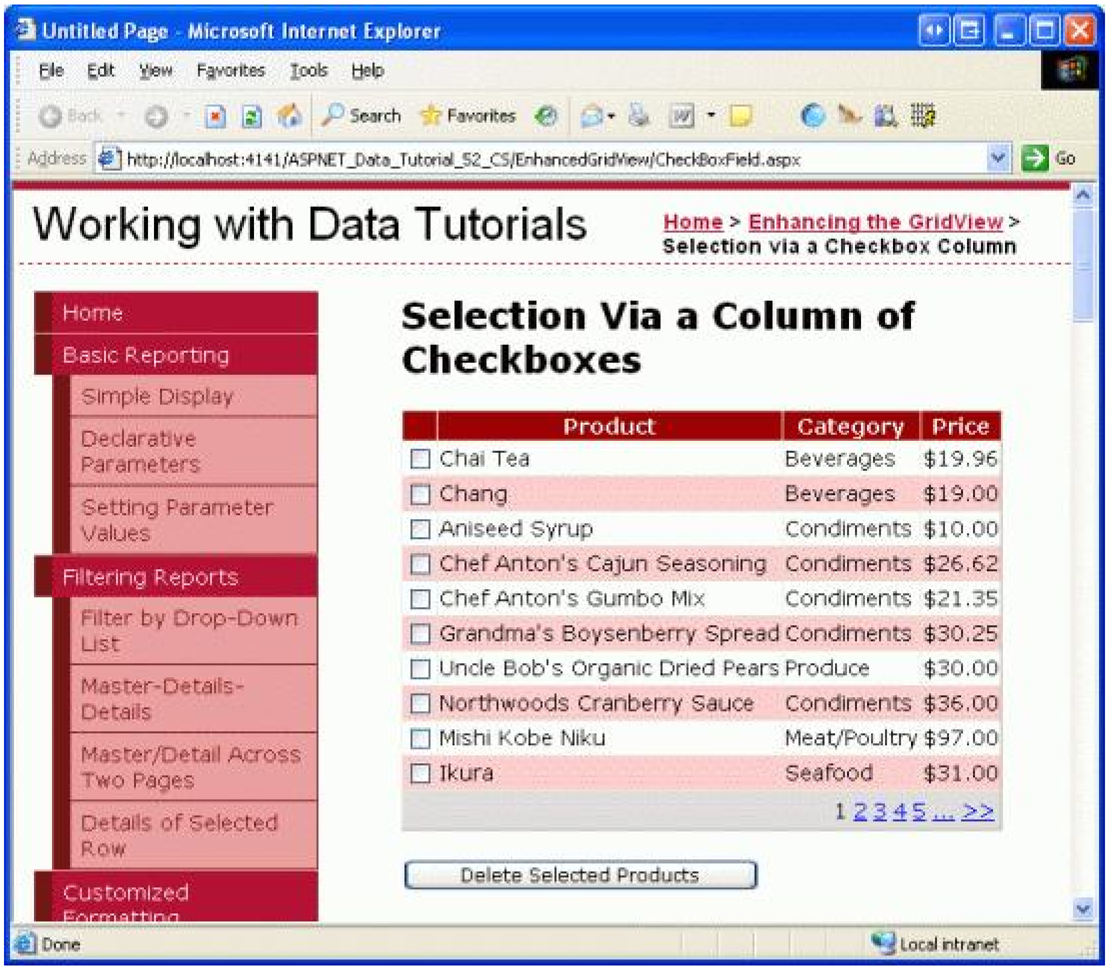

ASPNET_Data_Tutorial_52_CS - Microsoft Visual Studio
Toolbox
CheckBox
Selection Via a Column of Checkboxes
Products - Column[0]
ItemTemplate
[ProductSelector]
ObjectDataSource - ProductsDataSource
Delete Selected Products
[DeleteResults]
Properties
Products.Column0.ItemTemplate.Product
(ID) ProductSelector

Figure 6: Add a CheckBox Web Control Named `ProductSelector` to the TemplateField's `ItemTemplate`

With the TemplateField and CheckBox Web control added, each row now includes a checkbox. Figure 7 shows this page, when viewed through a browser, after the TemplateField and CheckBox have been added.



Working with Data Tutorials

Home > Enhancing the GridView > Selection via a Checkbox Column

Selection Via a Column of Checkboxes

	Product	Category	Price
☐	Chai Tea	Beverages	$19.96
☐	Chang	Beverages	$19.00
☐	Aniseed Syrup	Condiments	$10.00
☐	Chef Anton's Cajun Seasoning	Condiments	$26.62
☐	Chef Anton's Gumbo Mix	Condiments	$21.35
☐	Grandma's Boysenberry Spread	Condiments	$30.25
☐	Uncle Bob's Organic Dried Pears	Produce	$30.00
☐	Northwoods Cranberry Sauce	Condiments	$36.00
☐	Mishi Kobe Niku	Meat/Poultry	$97.00
☐	Ikura	Seafood	$31.00

1 2 3 4 5 ... >>

Delete Selected Products

Figure 7: Each Product Row Now Includes a Checkbox

Step 3: Determining What Checkboxes Were Checked On Postback

At this point we have a column of checkboxes but no way to determine what checkboxes were checked on postback. When the "Delete Selected Products" button is clicked, though, we need to know what checkboxes were checked in order to delete those products.

The GridView's `Rows` property provides access to the data rows in the GridView. We can iterate through these rows, programmatically access the CheckBox control, and then consult its `Checked` property to determine whether the CheckBox has been selected.

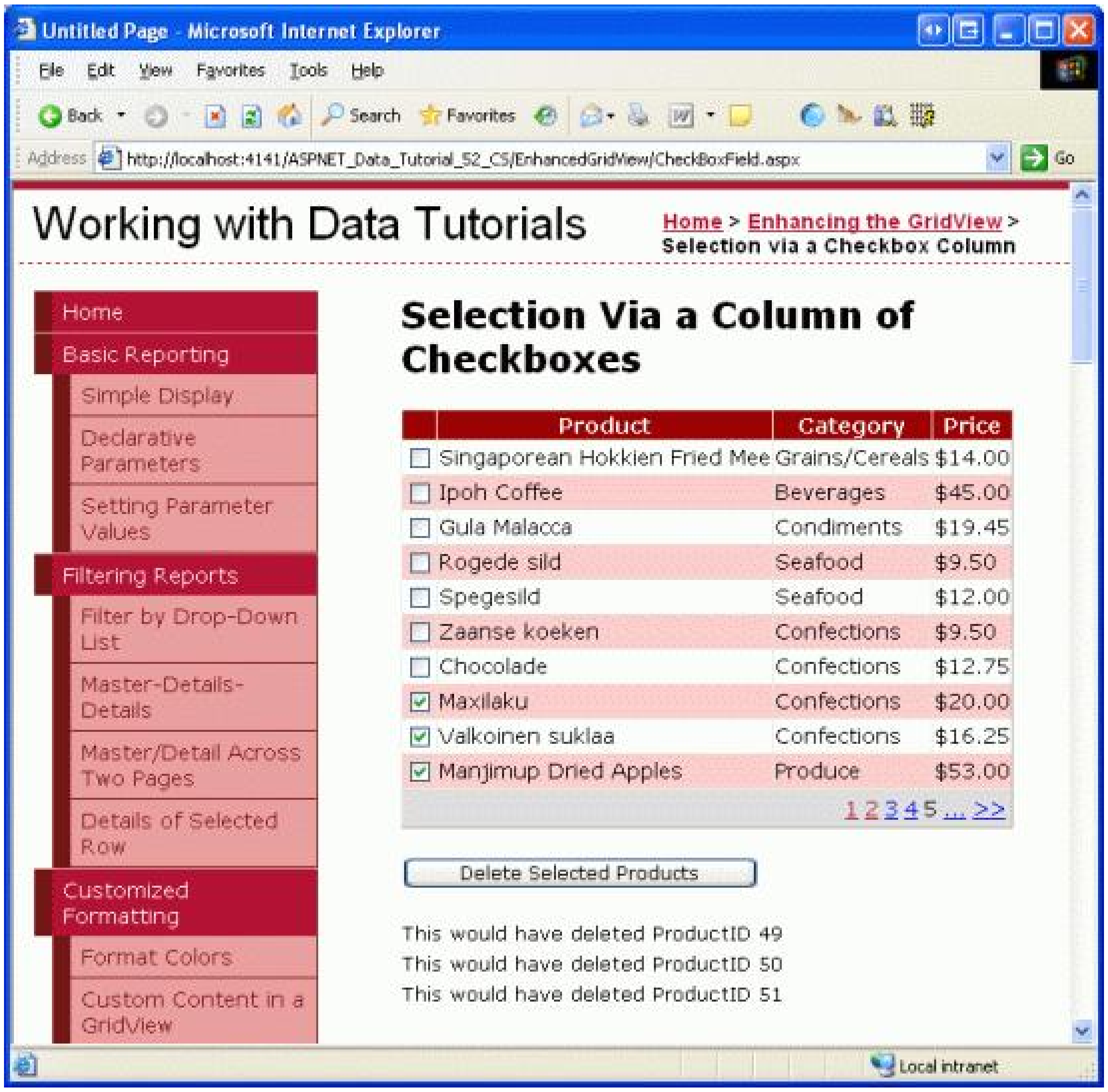
Create an event handler for the `DeleteSelectedProducts` Button Web control's `Click` event and add the following code:

```
Protected Sub DeleteSelectedProducts_Click(sender As Object, e As EventArgs) _
    Handles DeleteSelectedProducts.Click

    Dim atLeastOneRowDeleted As Boolean = False

    ' Iterate through the Products.Rows property
    For Each row As GridViewRow In Products.Rows
        ' Access the CheckBox
```

```
            Dim cb As CheckBox = row.FindControl("ProductSelector")
            If cb IsNot Nothing AndAlso cb.Checked Then
                ' Delete row! (Well, not really...)
                atLeastOneRowDeleted = True

                ' First, get the ProductID for the selected row
                Dim productID As Integer = _
                    Convert.ToInt32(Products.DataKeys(row.RowIndex).Value)

                ' "Delete" the row
                DeleteResults.Text &= String.Format( _
                    "This would have deleted ProductID {0}<br />", productID)

                '... To actually delete the product, use ...
                ' Dim productAPI As New ProductsBLL
                ' productAPI.DeleteProduct(productID)
                '............................................
            End If
        Next

        ' Show the Label if at least one row was deleted...
        DeleteResults.Visible = atLeastOneRowDeleted
    End Sub
```

The `Rows` property returns a collection of `GridViewRow` instances that makeup the GridView's data rows. The `For Each` loop here enumerates this collection. For each `GridViewRow` object, the row's CheckBox is programmatically accessed using `row.FindControl("`*`controlID`*`")`. If the CheckBox is checked, the row's corresponding `ProductID` value is retrieved from the `DataKeys` collection. In this exercise, we simply display an informative message in the `DeleteResults` Label, although in a working application we'd instead make a call to the `ProductsBLL` class's `DeleteProduct(`*`productID`*`)` method.

With the addition of this event handler, clicking the "Delete Selected Products" button now displays the `ProductID`s of the selected products.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:4141/ASPNET_Data_Tutorial_52_CS/EnhancedGridView/CheckBoxField.aspx
Go
Working with Data Tutorials
Home > Enhancing the GridView > Selection via a Checkbox Column
Home
Basic Reporting
Simple Display
Declarative Parameters
Setting Parameter Values
Filtering Reports
Filter by Drop-Down List
Master-Details-Details
Master/Detail Across Two Pages
Details of Selected Row
Customized Formatting
Format Colors
Custom Content in a GridView
Local intranet

Selection Via a Column of Checkboxes

	Product	Category	Price
☐	Singaporean Hokkien Fried Mee	Grains/Cereals	$14.00
☐	Ipoh Coffee	Beverages	$45.00
☐	Gula Malacca	Condiments	$19.45
☐	Rogede sild	Seafood	$9.50
☐	Spegesild	Seafood	$12.00
☐	Zaanse koeken	Confections	$9.50
☐	Chocolade	Confections	$12.75
☑	Maxilaku	Confections	$20.00
☑	Valkoinen suklaa	Confections	$16.25
☑	Manjimup Dried Apples	Produce	$53.00
			1 2 3 4 5 ... >>

Delete Selected Products

This would have deleted ProductID 49
This would have deleted ProductID 50
This would have deleted ProductID 51

Figure 8: When the "Delete Selected Products" Button is Clicked the Selected Products' `ProductIDs` are Listed

Step 4: Adding "Check All" and "Uncheck All" Buttons

If a user wants to delete all products on the current page, they must check each of the ten checkboxes. We can help expedite this process by adding a "Check All" button that, when clicked, selects all of the checkboxes in the grid. An "Uncheck All" button would be equally helpful.

Add two Button Web controls to the page, placing them above the GridView. Set the first one's `ID` to `CheckAll` and its `Text` property to "Check All"; set the second one's `ID` to `UncheckAll` and its `Text` property to "Uncheck All".

```
<asp:Button ID="CheckAll" runat="server" Text="Check All" />
&nbsp;
<asp:Button ID="UncheckAll" runat="server" Text="Uncheck All" />
```

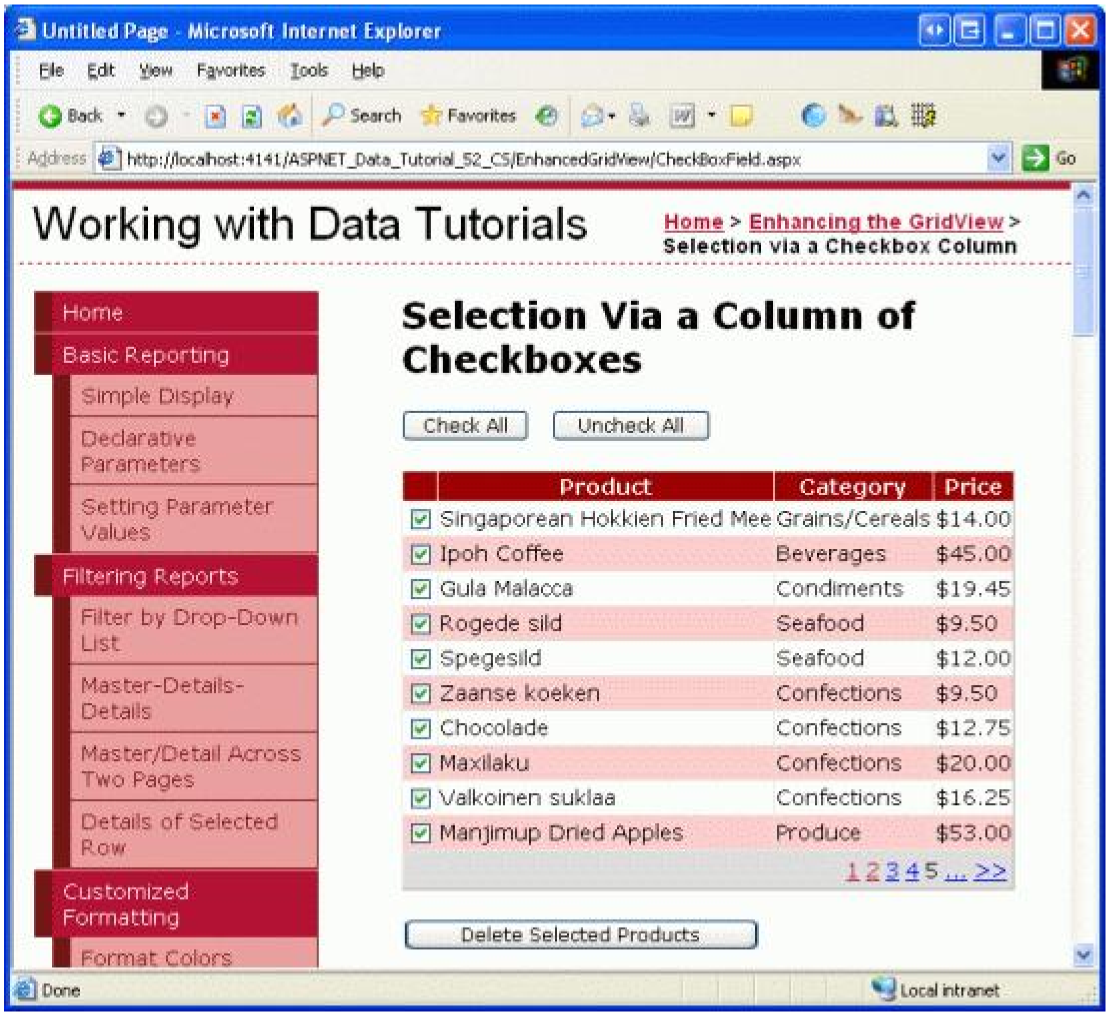
Next, create a method in the code-behind class named `ToggleCheckState(`*`checkState`*`)` that, when invoked, enumerates the `Products` GridView's `Rows` collection and sets each CheckBox's `Checked` property to the value of the passed in *checkState* parameter.

```
Private Sub ToggleCheckState(ByVal checkState As Boolean)
    ' Iterate through the Products.Rows property
    For Each row As GridViewRow In Products.Rows
        ' Access the CheckBox
        Dim cb As CheckBox = row.FindControl("ProductSelector")
        If cb IsNot Nothing Then
            cb.Checked = checkState
        End If
    Next
End Sub
```

Next, create `Click` event handlers for the `CheckAll` and `UncheckAll` buttons. In `CheckAll`'s event handler, simply call `ToggleCheckState(True)`; in `UncheckAll`, call `ToggleCheckState(False)`.

```
Protected Sub CheckAll_Click(sender As Object, e As EventArgs) _
    Handles CheckAll.Click

    ToggleCheckState(True)
End Sub

Protected Sub UncheckAll_Click(sender As Object, e As EventArgs) _
    Handles UncheckAll.Click

    ToggleCheckState(False)
End Sub
```

With this code, clicking the “Check All” button causes a postback and checks all of the checkboxes in the GridView. Likewise, clicking “Uncheck All” unselects all checkboxes. Figure 9 shows the screen after the “Check All” button has been checked.


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://localhost:4141/ASPNET_Data_Tutorial_52_CS/EnhancedGridView/CheckBoxField.aspx
Go
Working with Data Tutorials
Home > Enhancing the GridView > Selection via a Checkbox Column
Home
Basic Reporting
Simple Display
Declarative Parameters
Setting Parameter Values
Filtering Reports
Filter by Drop-Down List
Master-Details-Details
Master/Detail Across Two Pages
Details of Selected Row
Customized Formatting
Format Colors
Check All
Uncheck All
Delete Selected Products
Done
Local intranet

Selection Via a Column of Checkboxes

	Product	Category	Price
✓	Singaporean Hokkien Fried Mee	Grains/Cereals	$14.00
✓	Ipoh Coffee	Beverages	$45.00
✓	Gula Malacca	Condiments	$19.45
✓	Rogede sild	Seafood	$9.50
✓	Spegesild	Seafood	$12.00
✓	Zaanse koeken	Confections	$9.50
✓	Chocolade	Confections	$12.75
✓	Maxilaku	Confections	$20.00
✓	Valkoinen suklaa	Confections	$16.25
✓	Manjimup Dried Apples	Produce	$53.00
			1 2 3 4 5 ... >>

Figure 9: Clicking the "Check All" Button Selects All Checkboxes

Note: When displaying a column of checkboxes, one approach for selecting or unselecting all of the checkboxes is through a checkbox in the header row. Moreover, the current "Check All" / "Uncheck All" implementation requires a postback. The checkboxes can be checked or unchecked, however, entirely through client-side script, thereby providing a snappier user experience. To explore using a header row checkbox for "Check All" and "Uncheck All" in detail, along with a discussion on using client-side techniques, check out Checking All CheckBoxes in a GridView Using Client-Side Script and a Check All CheckBox.

Summary

In cases where you need to let users choose an arbitrary number of rows from a GridView before proceeding, adding a column of checkboxes is one option. As we saw in this tutorial, including a column of checkboxes in the GridView entails adding a TemplateField with a CheckBox Web control. By using a Web control (versus injecting markup directly into the template, as we did in the previous tutorial) ASP.NET automatically remembers what CheckBoxes were and were not checked across postback. We can also programmatically access the CheckBoxes in code to determine whether a given CheckBox is checked, or to chnage the checked state.

This tutorial and the last one looked at adding a row selector column to the GridView. In our next tutorial we'll examine how, with a bit of work, we can add inserting capabilities to the GridView.

Happy Programming!

About the Author

Scott Mitchell, author of seven 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. His latest book is *Sams Teach Yourself ASP.NET 2.0 in 24 Hours*. He can be reached at mitchell@4GuysFromRolla.com. or via his blog, which can be found at http://ScottOnWriting.NET.