

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 :: Adding Client-Side Confirmation When Deleting

Introduction

As we saw back in the [Overview of Editing and Deleting Data in the DataList](#) tutorial, deleting support can be added to the DataList by:

1. Adding a Button, LinkButton, or ImageButton Web control to the DataList's `ItemTemplate`,
2. Setting the Delete button's `CommandName` property to "Delete", and
3. Calling the appropriate BLL delete method from the DataList's `DeleteCommand` event handler (and then rebinding the data so that the just-deleted item is removed from the DataList's items).

From the end user's perspective, clicking an item's Delete button causes a postback and deletes the selected item, removing it from the DataList. This default user interface, however, lacks any sort of confirmation when the user clicks the Delete button. If a user accidentally clicks the Delete button when they meant to click Edit, the record they meant to update will instead be deleted. To help prevent this, we can add a client-side confirmation dialog box that appears when the Delete button is clicked.

The JavaScript `confirm(string)` function displays its string input parameter as the text inside a modal dialog box that comes equipped with two buttons - OK and Cancel (see Figure 1). The `confirm(string)` function returns a Boolean value depending on what button is clicked (`true`, if the user clicks OK, and `false` if they click Cancel).



Figure 1: The JavaScript `confirm(string)` Method Displays a Modal, Client-Side Messagebox

During a form submission, if a value of `false` is returned from a client-side event handler then the browser cancels the form submission. Using this feature, we can have the Delete button's client-side `onclick` event handler return the value of a call to `confirm("Are you certain that you want to delete this product?")`. If the user clicks Cancel, `confirm(string)` will return `false`, thereby causing the form submission to cancel. With no postback, the product whose Delete button was clicked won't be deleted. If, however, the user clicks OK in the confirmation dialog box, the postback will continue unabated and the product will be deleted. Consult [Using JavaScript's `confirm\(\)` Method to Control Form Submission](#) for more information on this technique.

In this tutorial we will see how to add such a client-side confirmation to the Delete button in a DataList.

Note: Using client-side confirmation techniques, like the ones discussed in this tutorial, assumes that your users are visiting with browsers that support JavaScript and that they have JavaScript enabled. If either of these assumptions are not true for a particular user, clicking the Delete button will immediately cause a postback (not displaying a confirm messagebox) and delete the record.

Step 1: Creating a DataList with a Delete Button

Before we can explore how to add client-side confirmation, we first need a DataList from which records can be deleted. Start by opening the `ConfirmationOnDelete.aspx` page in the `EditDeleteDataList` folder and drag a DataList from the Toolbox onto the Designer, setting its `ID` property to `Products`. Next, create a new `ObjectDataSource` named `ProductsDataSource` from the DataList's smart tag. Configure this `ObjectDataSource` so that it uses the `ProductsBLL` class's `GetProducts()` method to retrieve records (see Figure 2). Since we will perform the delete by directly interfacing with the BLL from the DataList's `DeleteCommand` event handler, set the drop-down lists in the Configure Data Source wizard's `INSERT`, `UPDATE`, and `DELETE` tabs to "(None)", as shown in Figure 3.

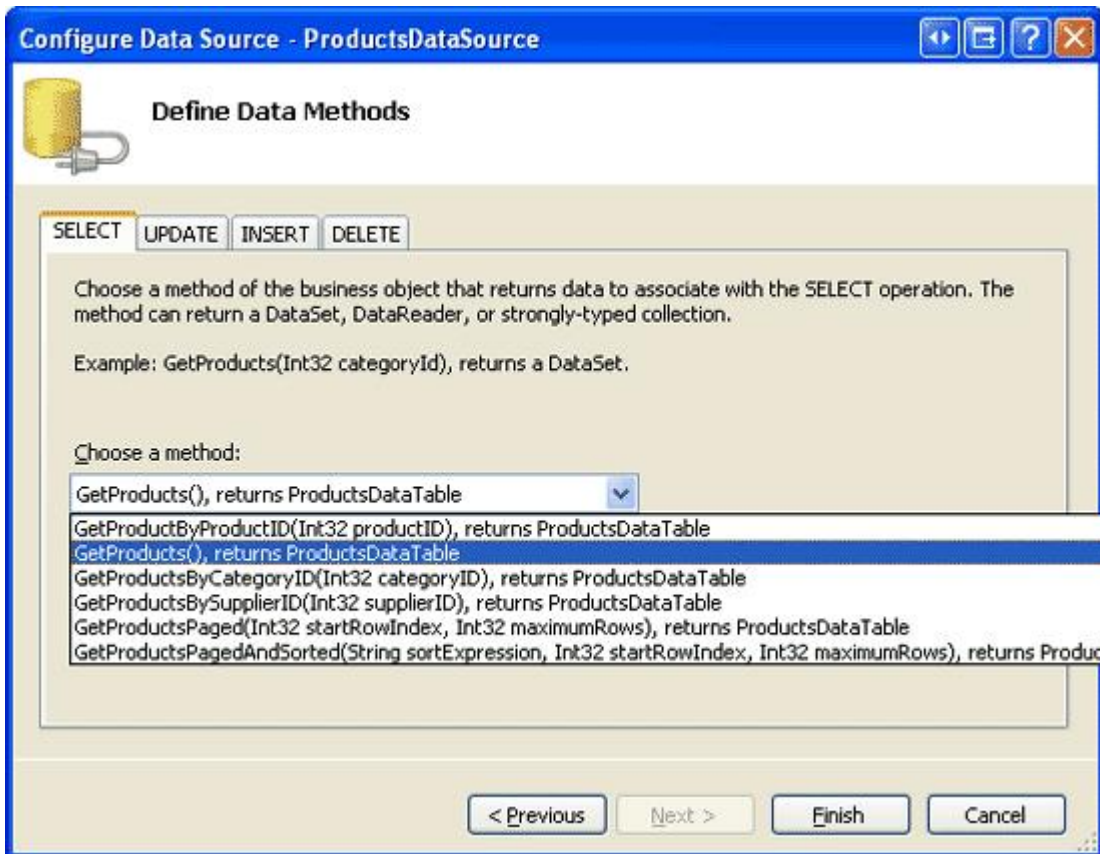


Figure 2: Configure the `ProductsDataSource` to Use the `ProductsBLL` Class's `GetProducts()` Method

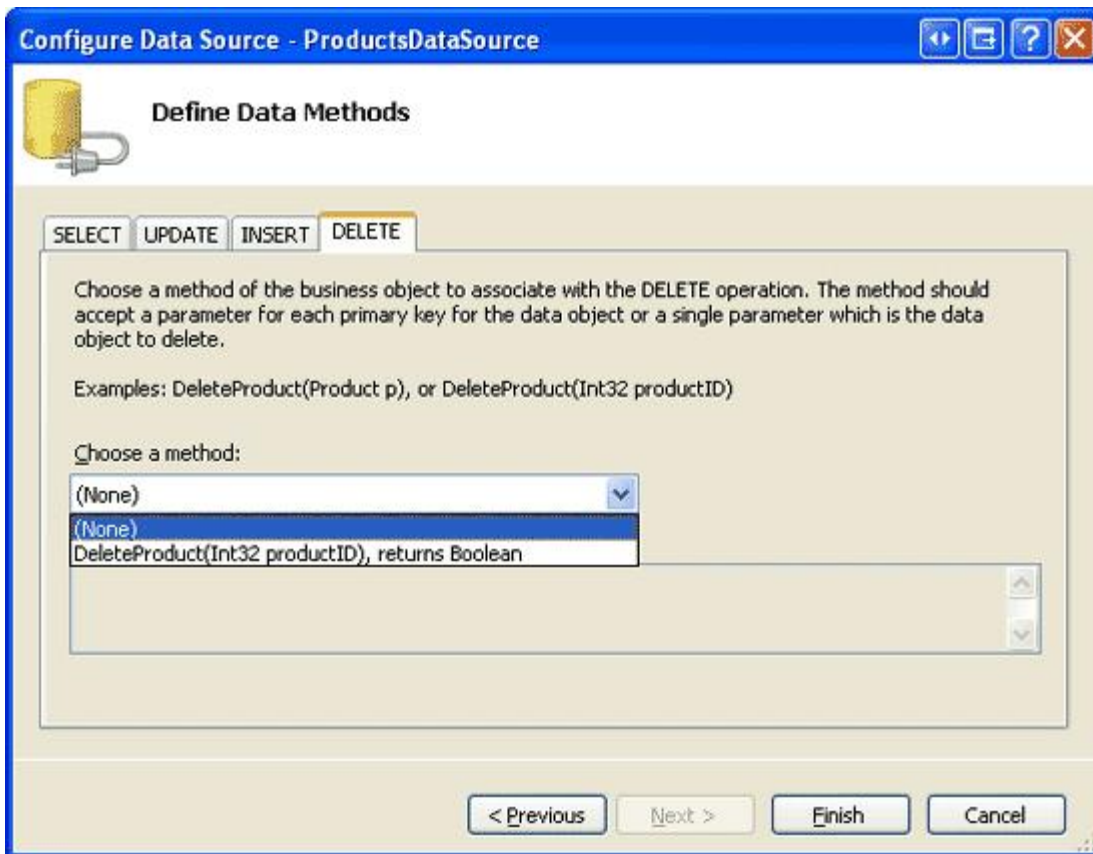


Figure 3: Set the INSERT, UPDATE, and DELETE Tabs' Drop-down Lists to "(None)"

After completing the Configure Data Source wizard, Visual Studio will create a default `ItemTemplate` for the `DataList` — the name of each data field followed by a `Label` Web control displaying the value. Update the `ItemTemplate` so that it only shows the product's name, category, and supplier. Also add a `Delete` button and set its `CommandName` property to "Delete". After making these changes, the `DataList` and `ObjectDataSource`'s declarative syntax should look similar to the following:

```
<asp:DataList ID="Products" runat="server" DataKeyField="ProductID"
DataSourceID="ProductsDataSource">
  <ItemTemplate>
    <h3>
      <asp:Label ID="ProductNameLabel" runat="server"
        Text='<%=# Eval("ProductName") %>' />
    </h3>
    Category:
      <asp:Label ID="CategoryNameLabel" runat="server"
        Text='<%=# Eval("CategoryName") %>' />
    <br />
    Supplier:
      <asp:Label ID="SupplierNameLabel" runat="server"
        Text='<%=# Eval("SupplierName") %>' />
    <br />
    <br />
    <asp:Button runat="server" ID="DeleteButton"
      CommandName="Delete" Text="Delete" />
    <br />
    <br />
  </ItemTemplate>
</asp:DataList>
<asp:ObjectDataSource ID="ProductsDataSource" runat="server"
  OldValuesParameterFormatString="original_{0}"
  SelectMethod="GetProducts" TypeName="ProductsBLL">
</asp:ObjectDataSource>
```

Figure 4 shows this page when viewed through a browser. Since we've yet to create the `DeleteCommand`

event handler, clicking the Delete button just causes a postback, leaving the DataList intact.

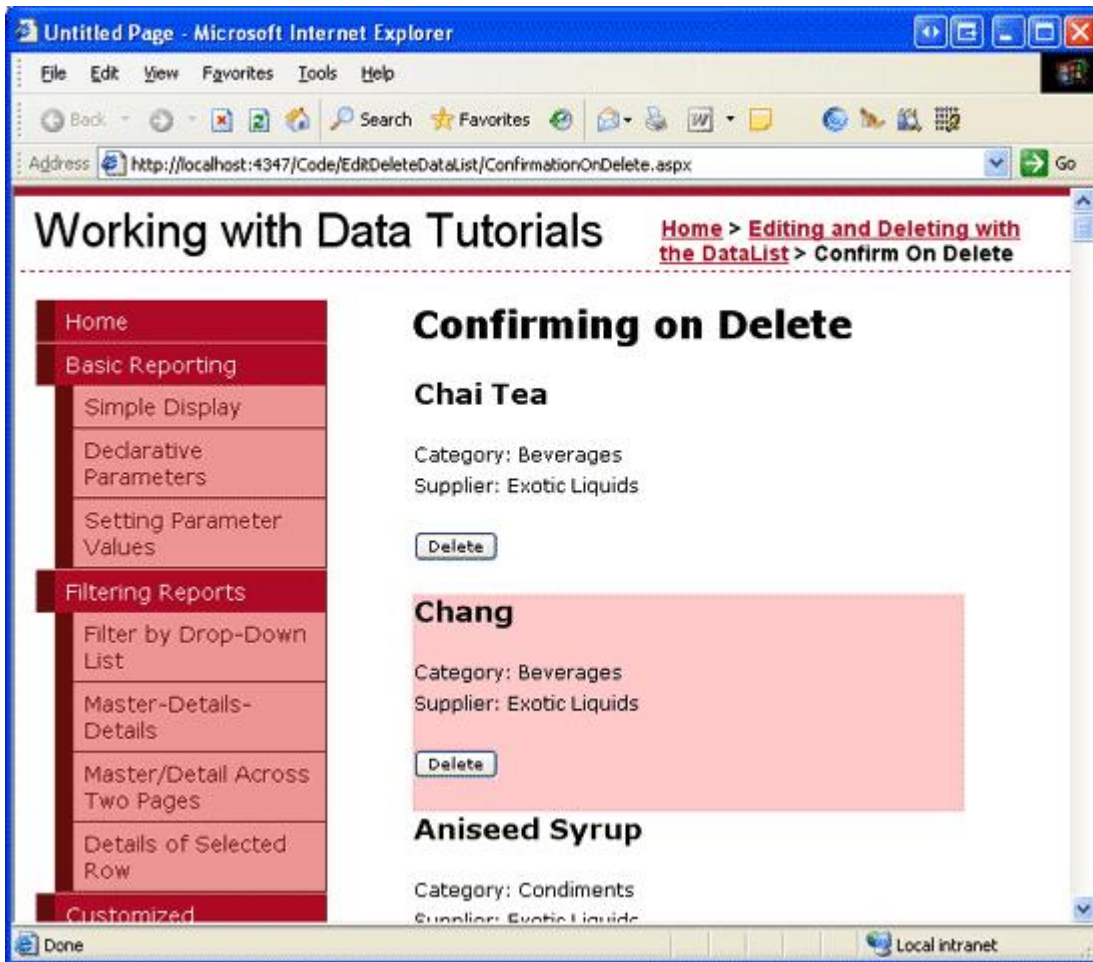


Figure 4: Each Product’s Name, Supplier, and Category is Shown, Along with a Delete Button

Step 2: Adding the DeleteCommand Event Handler

When a user clicks the Delete button, a postback occurs and the DataList’s `ItemCommand` and `DeleteCommand` events fire. The `ItemCommand` event fires anytime a `Command` event is raised from within the DataList; the `DeleteCommand` event is raised in addition because the Button that caused the `Command` event to fire has its `CommandName` property set to “Delete”.

To delete the record in response to the Delete button being clicked, we need to:

- Create an event handler for the DataList’s `DeleteCommand` event,
- Determine the `ProductID` of the item whose Delete button was clicked,
- Invoke the `ProductBLL` class’s `DeleteProduct` method, and
- Rebind the data to the DataList.

The necessary code is shown below; for a thorough explanation of the code, consult the [Overview of Editing and Deleting Data in the DataList](#) tutorial.

```
protected void Products_DeleteCommand(object source, DataListCommandEventArgs e)
{
    // Read in the ProductID from the DataKeys collection
    int productID = Convert.ToInt32(Products.DataKeys[e.Item.ItemIndex]);
```

```

// Delete the data
ProductsBLL productsAPI = new ProductsBLL();
productsAPI.DeleteProduct(productID);

// Rebind the data to the DataList
Products.DataBind();
}

```

Step 3: Adding Client-Side Confirmation to the Delete Button

With the `DeleteCommand` event handler completed, the final task is to add a client-side confirmation to the Delete button. This can be accomplished declaratively or programmatically. The simplest way is to specify the confirmation declaratively via the Button's `OnClickClientClick` property. If, however, the confirmation message or logic depends upon factors that can only be determined at runtime, then we can reference the Delete button in the DataList's `ItemDataBound` event handler and set its `OnClickClientClick` property programmatically.

Let's explore both of these options, starting with the declarative approach. To add the client-side confirmation, simply set the Button's `OnClickClientClick` property to `return confirm(message);`. For example, to have the confirmation message read, "Are you certain that you want to delete this product?", update the Delete button's declarative markup so that it reads as follows:

```

<asp:Button runat="server" ID="DeleteButton" CommandName="Delete" Text="Delete"
  OnClientClick="return confirm('Are you sure you want to delete this product?');" />

```

With this change, clicking the Delete button displays a modal confirmation dialog box (see Figure 5). A postback only occurs — and therefore the product is only deleted — if the user clicks the Ok button.

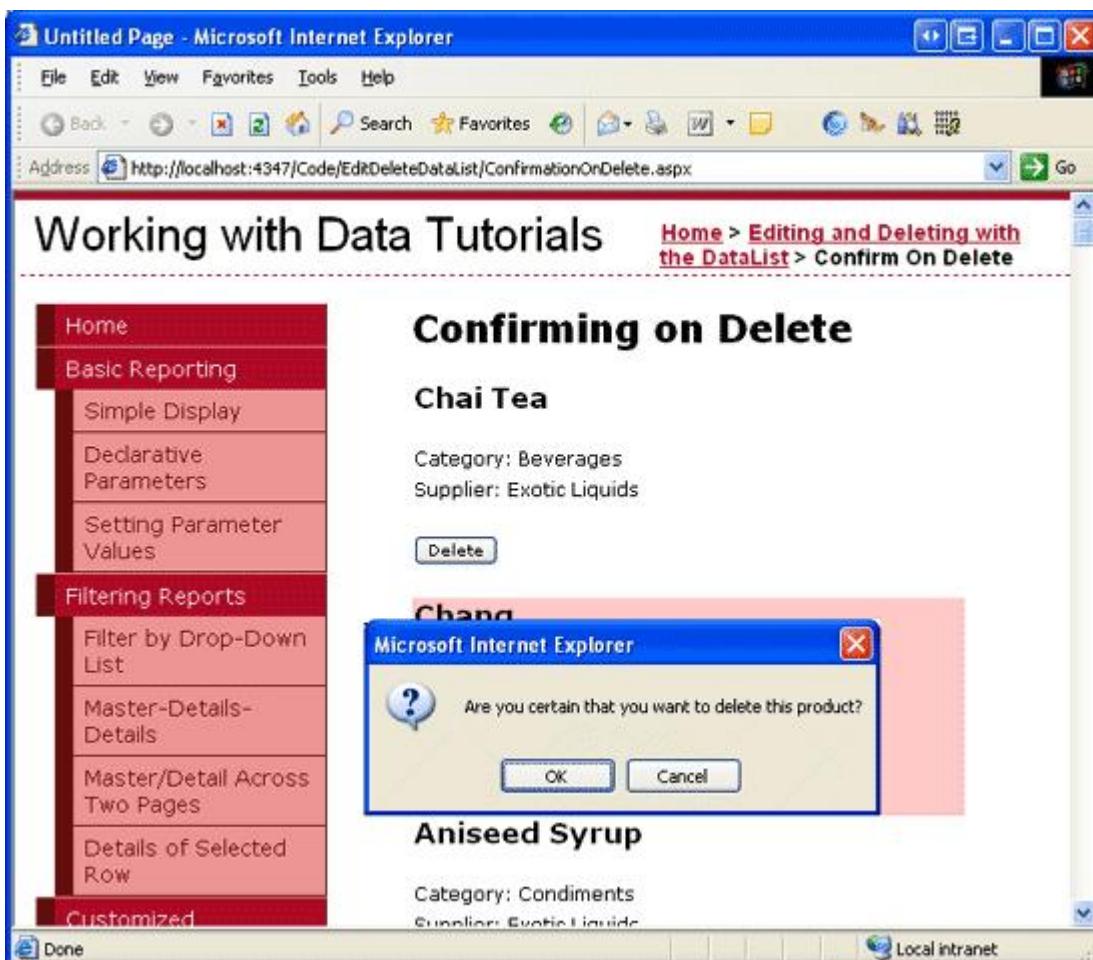


Figure 5: Clicking the Delete Button Displays a Client-Side Confirmation Dialog Box

Note: Since we are using apostrophes to delimit the string passed into the JavaScript `confirm` function, it's imperative that any apostrophes within the input string be escaped using `\'`. That is, if you want to pass the string, “You’re sure you want to do this?” into the `confirm` method, be sure to escape the apostrophe in “You’re” like so: `return confirm('You\'re sure you want to do this?');`

The declarative approach works well when displaying a static confirmation dialog box for all `DataList` items. To customize the confirmation message on an item-by-item basis, however, we need to set this property programmatically. The `DataList`'s `ItemDataBound` event fires after each record has been bound to the `DataList`. By creating an event handler for this event, we can reference the Delete button and set its `OnClickClientClick` property programmatically, based on the data bound to the `DataList` item. The following `ItemDataBound` event handler illustrates how to have the confirmation message include the product's name:

```
protected void Products_ItemDataBound(object sender, DataListItemEventArgs e)
{
    if (e.Item.ItemType == ListItemType.Item ||
        e.Item.ItemType == ListItemType.AlternatingItem)
    {
        // Reference the data bound to the DataListItem
        Northwind.ProductsRow product =
            (Northwind.ProductsRow) ((System.Data.DataRowView)e.Item.DataItem).Row;

        // Reference the Delete button
        Button db = (Button)e.Item.FindControl("DeleteButton");

        // Assign the Delete button's OnClientClick property
        db.OnClientClick =
            string.Format("return confirm('Are you sure that you want to delete" +
                " the product {0}?');", product.ProductName.Replace("'", @"\'"));
    }
}
```

Since the `ItemDataBound` event fires for *all* rows bound to the `DataList` — including header, footer, and separator items — it's important that we ensure that we're dealing with an item or alternating item, as those are the item types that are rendered from the `ItemTemplate`. Next, we access the `DataListItem`'s `DataItem` property, which contains a reference to the `ProductsRow` instance used to create this item. Finally, the Delete button is referenced via the `FindControl("controlID")` method and its `OnClickClientClick` property is assigned the confirmation message, “Are you sure that you want to delete the product *productName*?” Note that any apostrophes in the product's name are escaped.

After replacing the declarative approach with the programmatic one, take a moment to test out the page. When we click the Delete button this time, the confirmation message includes the product's name. Figure 6 shows the output when clicking the Delete button for Chai Tea.

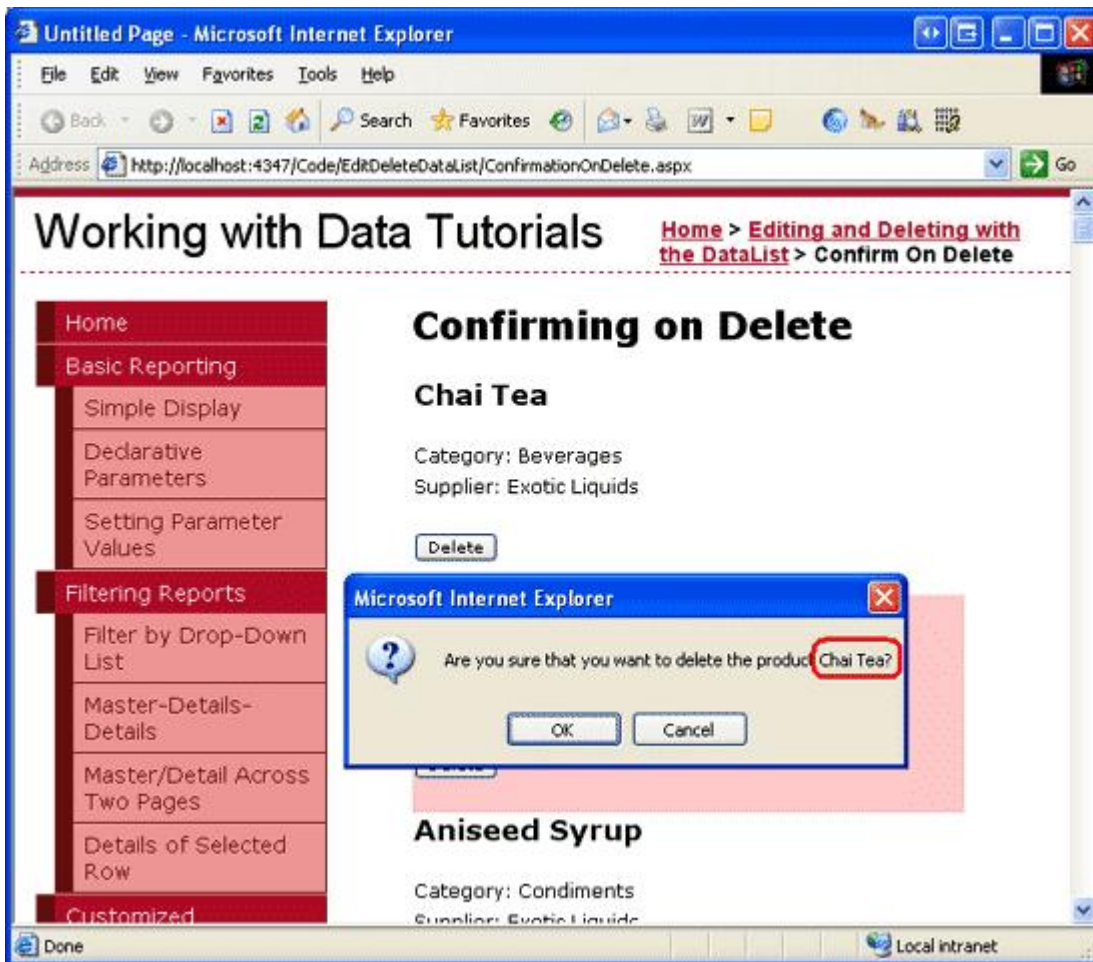


Figure 6: The Confirmation Dialog Box Now Includes the Product's Name

Summary

The JavaScript `confirm(string)` function is a commonly used technique for controlling form submission workflow. When executed, the function displays a modal, client-side dialog box that includes two buttons, OK and Cancel. If the user clicks OK, the `confirm(string)` function returns `true`; clicking Cancel returns `false`. This functionality, coupled with a browser's behavior to cancel a form submission if an event handler during the submission process returns `false`, can be used to display a confirmation message box when deleting a record.

The `confirm(string)` function can be associated with a Button Web control's client-side `onclick` event handler through the control's `OnClickClientClick` property. When working with a Delete button in a DataList this property can be set either declaratively or programmatically, as we saw in this tutorial.

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, [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 ScottOnWriting.NET.