

Contents

Details View Control.....	1
Displaying data with the detailsView control	1
Using the fields with the detailsView	4
Formatting the detailsView control	4
Displaying empty data	7
Paging.....	8
Customising the paging interface.....	8
Updating data	12
Using templates when editing.....	13
Inserting	17
Deleting.....	19
DetailsView control events	20
References	20

The details view and form view controls allow you to work with a single data item at a time. Both these controls allow you to display, edit, delete and insert data. The details view control uses a HTML table to display the data. The form view control uses templates to display the data.

Displaying data with the detailsView control

The details view control displays one row of data at a time. The data is displayed using a HTML table.

Activity 1. Displaying data

default.aspx

To use the details view control you need to add a data source control to the page.

Configure it so that it brings back the employee all the details for employee 1. To set up the where clause you can click on the where button and add the following:

Add one or more conditions to the WHERE clause for the statement. For each condition you can specify either a literal value or a parameterized value. Parameterized values get their values at runtime based on their properties.

Column: EmployeeID

Operator: =

Source: None

SQL Expression: [EmployeeID] = @EmployeeID

Value: 1

Parameter properties

Value: 1

WHERE clause:

SQL Expression	Value
[EmployeeID] = @EmployeeID	1

Buttons: Add, Remove, OK, Cancel

The resulting SQL statement is:

```
SELECT * FROM [Employees] WHERE ([EmployeeID] = @EmployeeID)
```

Now we need to add a details view control and set it up so that it uses the data source control created above.

Have a look at the source code generated:

```
<asp:SqlDataSource ID="sdsEmployee" runat="server" ConnectionString="<%"$
ConnectionString:ConnectionStringNorthWind %">
SelectCommand="SELECT * FROM [Employees] WHERE ([EmployeeID] = @EmployeeID)">
<SelectParameters>
<asp:Parameter DefaultValue="1"
Name="EmployeeID" Type="Int32" />
</SelectParameters>
</asp:SqlDataSource>
```

```
<asp:DetailsView ID="DetailsView1" runat="server" AutoGenerateRows="False"
DataKeyNames="EmployeeID"
DataSourceID="sdsEmployee" Height="50px" Width="125px">
<Fields>
<asp:BoundField DataField="EmployeeID"
HeaderText="EmployeeID" InsertVisible="False"
ReadOnly="True" SortExpression="EmployeeID" />
<asp:BoundField DataField="LastName"
HeaderText="LastName" SortExpression="LastName" />
<asp:BoundField DataField="FirstName"
HeaderText="FirstName" SortExpression="FirstName" />
<asp:BoundField DataField="Title" HeaderText="Title"
SortExpression="Title" />
<asp:BoundField DataField="TitleOfCourtesy"
HeaderText="TitleOfCourtesy"
SortExpression="TitleOfCourtesy" />
<asp:BoundField DataField="BirthDate"
HeaderText="BirthDate" SortExpression="BirthDate" />
<asp:BoundField DataField="HireDate"
```

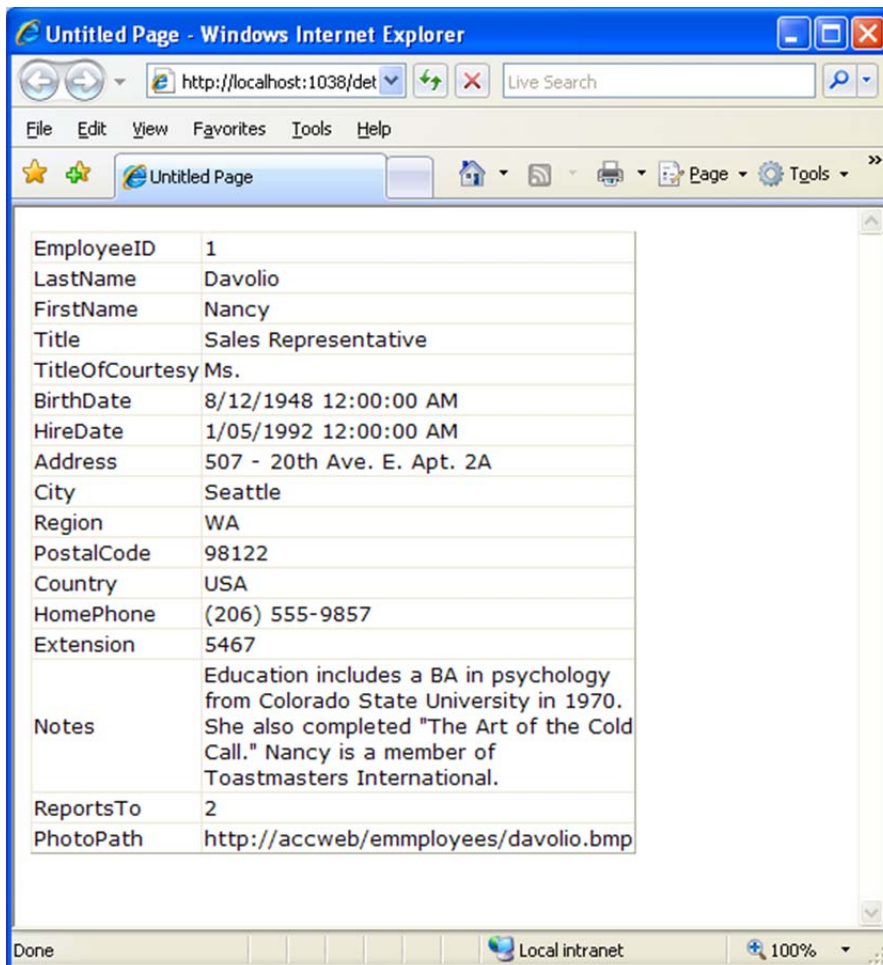
```

HeaderText="HireDate" SortExpression="HireDate" />
<asp:BoundField DataField="Address" HeaderText="Address"
SortExpression="Address" />
<asp:BoundField DataField="City" HeaderText="City"
SortExpression="City" />
<asp:BoundField DataField="Region" HeaderText="Region"
SortExpression="Region" />
<asp:BoundField DataField="PostalCode"
HeaderText="PostalCode" SortExpression="PostalCode" />
<asp:BoundField DataField="Country" HeaderText="Country"
SortExpression="Country" />
<asp:BoundField DataField="HomePhone"
HeaderText="HomePhone" SortExpression="HomePhone" />
<asp:BoundField DataField="Extension"
HeaderText="Extension" SortExpression="Extension" />
<asp:BoundField DataField="Notes" HeaderText="Notes"
SortExpression="Notes" />
<asp:BoundField DataField="ReportsTo"
HeaderText="ReportsTo" SortExpression="ReportsTo" />
<asp:BoundField DataField="PhotoPath"
HeaderText="PhotoPath" SortExpression="PhotoPath" />
</Fields>
</asp:DetailsView>

```

The fields displayed are set up by using the <field> tag. This is similar to the columns tag for the gridview control. Each field to be displayed is set up using a <asp:boundfield> tag. The properties of the boundfield tag are the same as with the gridview control.

View the page in your browser:



Using the fields with the detailsView

The type of fields available are:

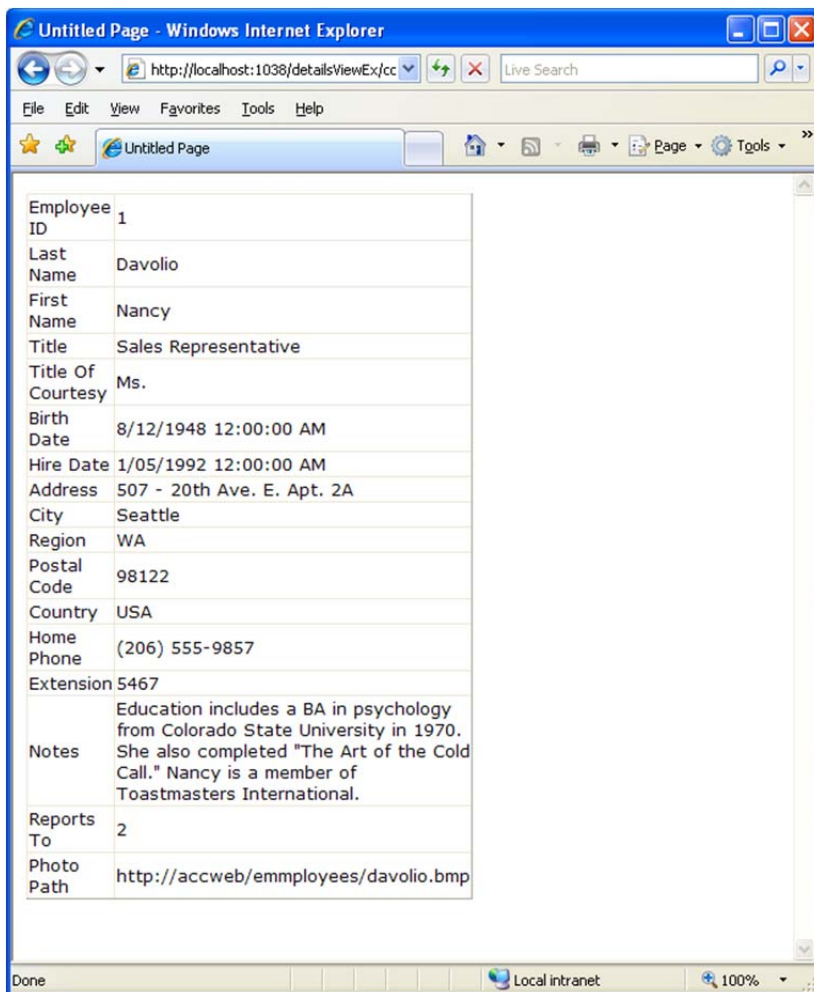
- boundField – Display the data as text
- CheckBoxField – Display the data as a checkbox
- CommandField – Display links for editing, deleting and selecting rows
- buttonField – Displays the data as a button
- hyperlinkField – Displays data as a link
- imageField – Display the data as an image
- TemplateField – Customize the appearance of the data item.

Activity 2. Modify bound field

columnHeadings.aspx

Modify the previous exercise so that the header text is set up to display the headings as 2 separate words.

You can edit the field headings by selecting edit fields from the smart tag menu or through source view.

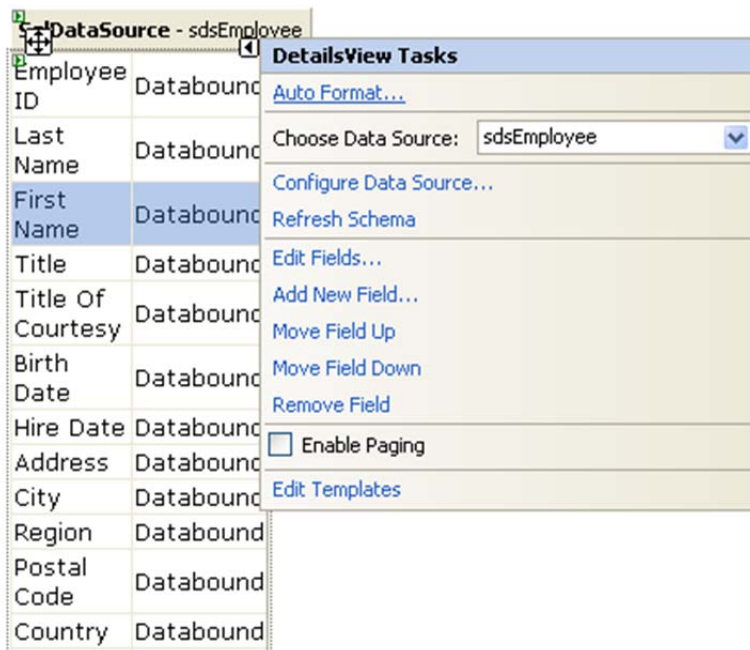


The screenshot shows a web browser window titled 'Untitled Page - Windows Internet Explorer'. The address bar shows 'http://localhost:1038/detailsViewEx/cc'. The page displays a details view for an employee. The fields are as follows:

Employee ID	1
Last Name	Davolio
First Name	Nancy
Title	Sales Representative
Title Of Courtesy	Ms.
Birth Date	8/12/1948 12:00:00 AM
Hire Date	1/05/1992 12:00:00 AM
Address	507 - 20th Ave. E. Apt. 2A
City	Seattle
Region	WA
Postal Code	98122
Country	USA
Home Phone	(206) 555-9857
Extension	5467
Notes	Education includes a BA in psychology from Colorado State University in 1970. She also completed "The Art of the Cold Call." Nancy is a member of Toastmasters International.
Reports To	2
Photo Path	http://accweb/emmployees/davolio.bmp

Formatting the detailsView control

You can format the detailsView control in a number of ways. Firstly you can access the auto format menu item from the smart tag menu.

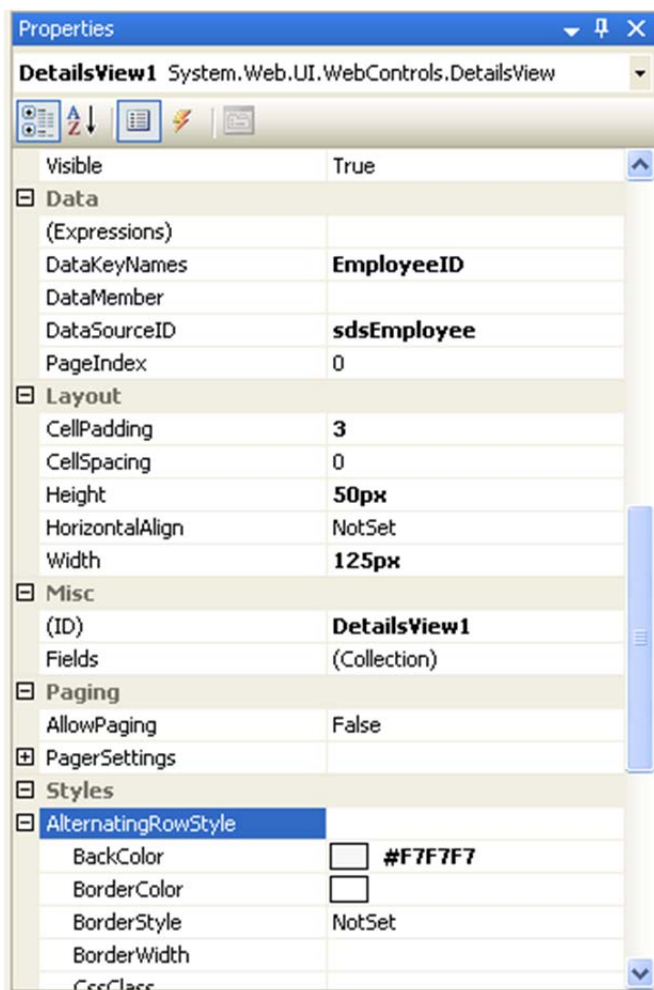


Activity 3. Set the formatting

formatting1.aspx

Change the formatting of your details view control by using the auto format.

You can also change the formatting through modifying the properties of the control.



The best way to set up the formatting is through the use of external style sheets.

The following properties of the details view control contain a cssClass property:

- **cssClass** – sets the style class for the whole detailsView control

- `alternatingRowStyle` – every second row
- `CommandRowStyle` – the row that contains the edit buttons
- `EditRowStyle` – when the details view control is in edit mode
- `EmptyDataRowStyle` – no data is displayed
- `FieldHeaderStyle` – the field labels
- `FooterStyle` – footer row
- `HeaderStyle` – header row
- `InsertRowStyle` – insert mode
- `pagerStyle` – how the paging interface is displayed
- `RowStyle` – each row

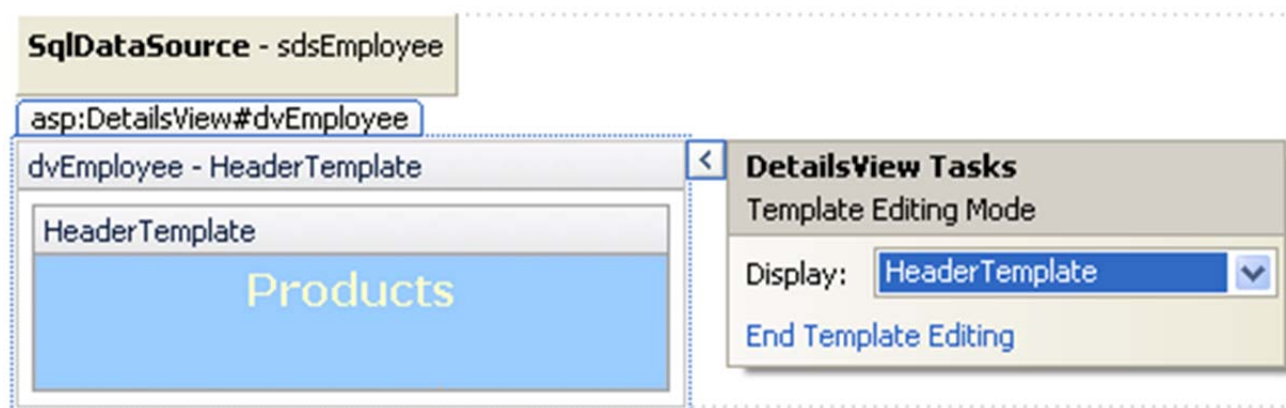
When formatting a details view control you may also want to modify the following properties:

- `Gridlines` – enables you to specify the appearance of rules that appear around the cells of the table. Possible values are none, horizontal, vertical and both.
- `headerText` – Enables you to specify the text for the heading of the details view control
- `footerText` – Enables you to specify the text for the footer row

Activity 4. Formatting

formatting2.aspx

Use a details view control to display the product details for product id 1. Format it so that it has the heading “Products”, it has no gridlines, every second row is a different colour and the field headings are in bold with a different back ground colour. To add a heading you will need to edit the header template

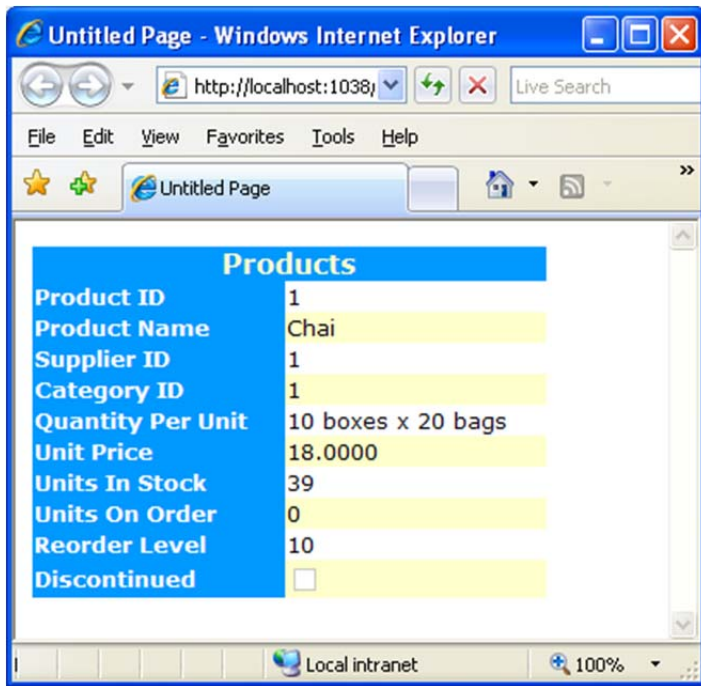


You will also need to add the following style sheet.

```
.alternatingRowStyle
{
    background-color: #FFFFCC;
}

.fieldHeader
{
    background-color: #0099FF;
    font-weight: bold;
    color: White;
}

.heading
{
    font-size: medium;
    background-color: #0099FF;
    text-align: center;
    font-weight: bold;
    color: #FFFFCC;
}
```

Displaying empty data

The details view control includes 2 properties that you can use to display something to the user when no data is returned from the data source. You can use the `emptyDataText` property to display some text or the `emptyDataTemplate` property to display any HTML or ASP.NET control.

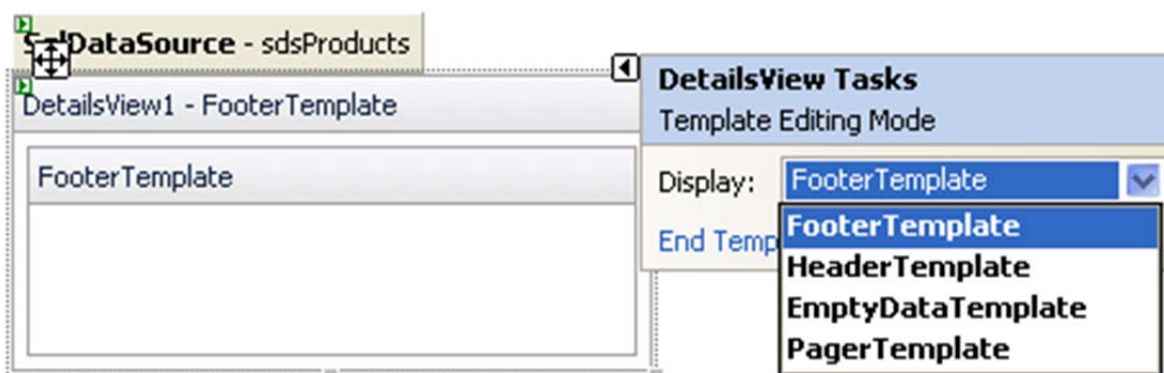
Activity 5. Displaying empty details view

emptyText.aspx

Q1. Display the product details for product number 900 (no such product). Use the `emptyDataText` property and set it to "No product"

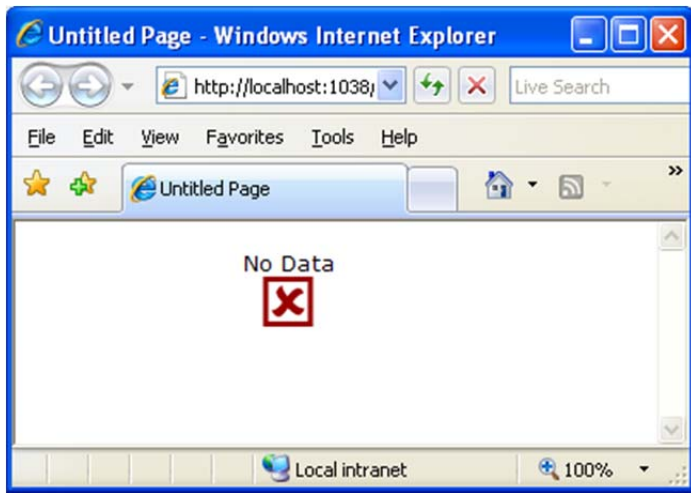
Q2. This time we will set up an `emptyDataTemplate` and set it up to display an image. Select edit template from the detailsView smart tag. Select `emptyDataTemplate` from the list of available templates.

emptyTemplate.aspx



Add the image `noData.gif` and place the text `No Data`.

When you load the page you should see the following:



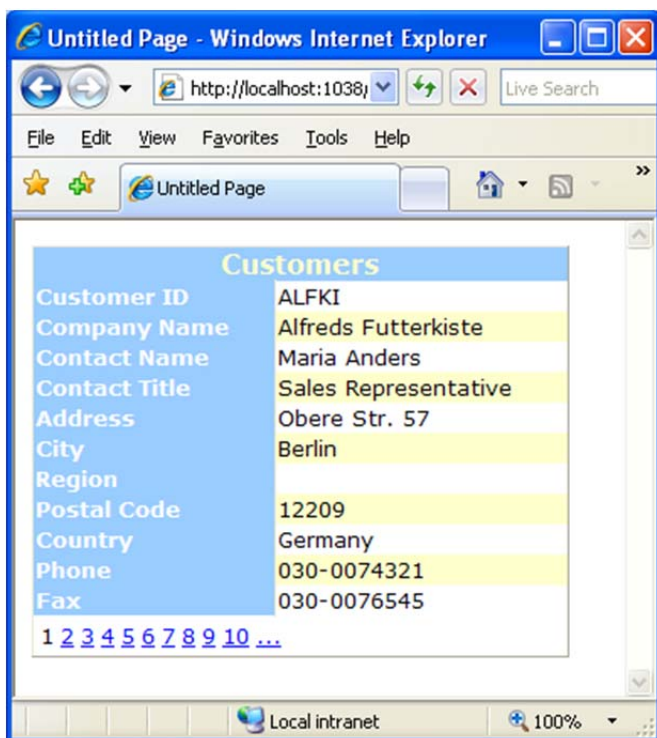
Paging

You can use the detailsView to page through the records. This is achieved by setting the allowPaging property to true.

Activity 6. Page through customers

pagingCustomers.aspx

Create a web page which will display each customer in a details view control. Access the smart tag of the details view control and place a check in the paging checkbox.



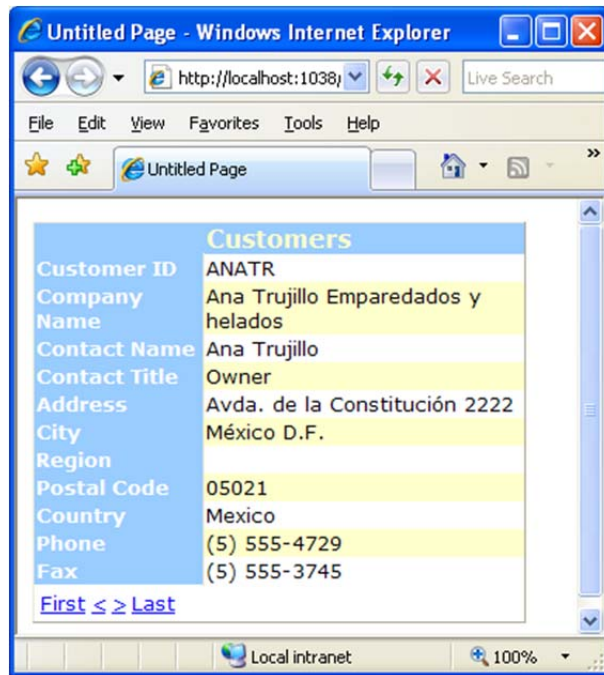
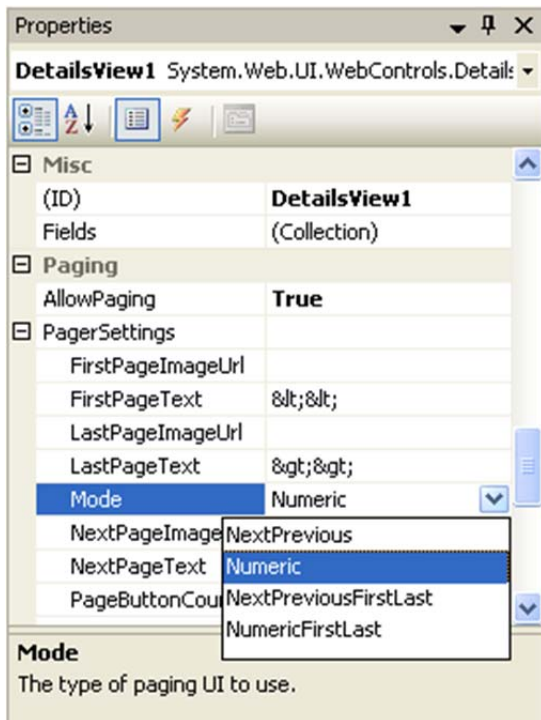
Customising the paging interface

You can customize the paging interface by modifying the pagerSettings property.

Activity 7. Customising paging

pagingCustomers2.aspx

Modify the paging settings so that the links Next, Previous, First and Last are displayed



Activity 8. Customising paging

pagingCustomers3.aspx

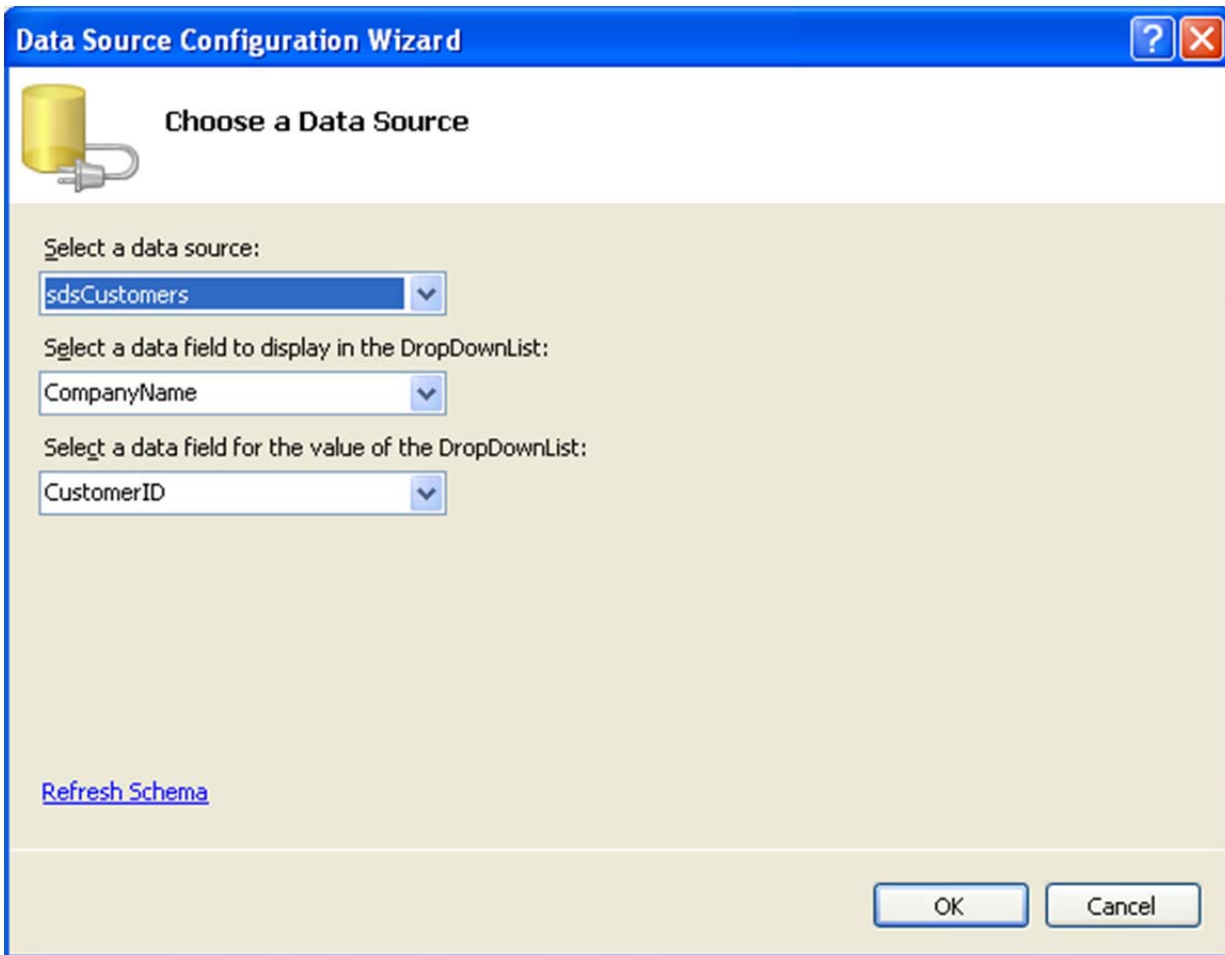
You can customise the paging using the paging template.

In this activity we will display a drop down list listing all the customer IDs when a customer ID is selected the detailsView will display the record of the selected customer.

To start this you need to first add a sqlDataSource control and set it up to retrieve all the customers.

Add a detailsView control and set it up to allow paging. From the details view smart tag select edit template. Select pager template from the list.

We will add a drop down list control in the pager template. Allow autopostback for the drop down list. This means the page is submitted to the server every time a user selects a different list item from the drop down list. Set up the drop down list control to have the same data source as the details view:



Data Source Configuration Wizard

Choose a Data Source

Select a data source:

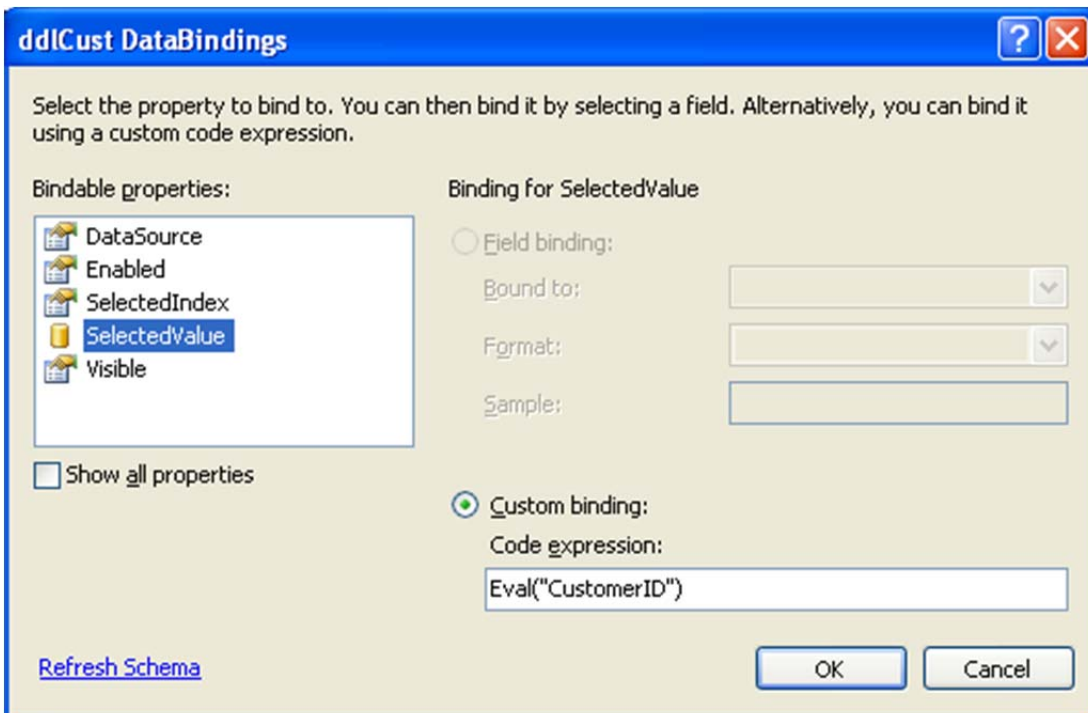
Select a data field to display in the DropDownList:

Select a data field for the value of the DropDownList:

[Refresh Schema](#)

OK Cancel

Set up the following data binding:



ddlCust DataBindings

Select the property to bind to. You can then bind it by selecting a field. Alternatively, you can bind it using a custom code expression.

Bindable properties:

- DataSource
- Enabled
- SelectedIndex
- SelectedValue**
- Visible

☐ Show all properties

Binding for SelectedValue

☐ Field binding:

Bound to:

Format:

Sample:

☒ Custom binding:

Code expression:

[Refresh Schema](#)

OK Cancel

Double click on the dropdown list to create the selectedIndexChanged event handling method. Now you need to add the following code from the code behind:

```
protected void ddlCustomers_SelectedIndexChanged(object sender, EventArgs e)
{
    DropDownList dd1TempCust;

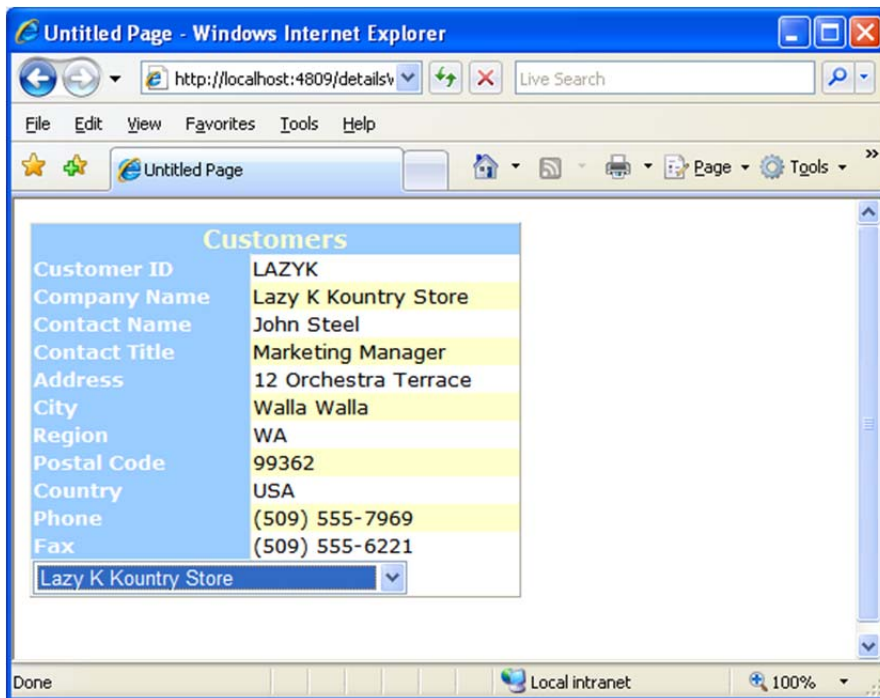
    dd1TempCust = (DropDownList)sender;
```

```

    dvCustomers.PageIndex = ddlTempCust.SelectedIndex;
}

```

The above event is triggered whenever a selection change has occurred in the drop down list. A variable of type drop down list is created. This variable will hold the sender object. The sender object is the object which triggered the event (the drop down list). The details view (dvCustomers) then has its page index changed to the index of the selected item in the drop down list. Making the selected customer ID the displayed customer.



Source code generated:

```

<asp:SqlDataSource ID="sdsCustomers" runat="server" ConnectionString="<%"$
ConnectionString:ConnectionStringNorthWind %>"
SelectCommand="SELECT * FROM [Customers]"></asp:SqlDataSource>
<asp:DetailsView ID="dvCustomers" runat="server" AllowPaging="True"
AutoGenerateRows="False"
DataKeyNames="CustomerID" DataSourceID="sdsCustomers" GridLines="Vertical"
HeaderText="Customers"
Height="50px" Width="319px">
<Fields>
<asp:BoundField DataField="CustomerID"
HeaderText="Customer ID" ReadOnly="True"
SortExpression="CustomerID" />
<asp:BoundField DataField="CompanyName"
HeaderText="Company Name" SortExpression="CompanyName" />
<asp:BoundField DataField="ContactName"
HeaderText="Contact Name" SortExpression="ContactName" />
<asp:BoundField DataField="ContactTitle"
HeaderText="Contact Title"
SortExpression="ContactTitle" />
<asp:BoundField DataField="Address" HeaderText="Address"
SortExpression="Address" />
<asp:BoundField DataField="City" HeaderText="City"
SortExpression="City" />
<asp:BoundField DataField="Region" HeaderText="Region"
SortExpression="Region" />
<asp:BoundField DataField="PostalCode"
HeaderText="Postal Code" SortExpression="PostalCode" />
<asp:BoundField DataField="Country" HeaderText="Country"
SortExpression="Country" />

```

```

<asp:BoundField DataField="Phone" HeaderText="Phone"
SortExpression="Phone" />
<asp:BoundField DataField="Fax" HeaderText="Fax"
SortExpression="Fax" />
</Fields>
<FieldHeaderStyle CssClass="fieldHeader" />
<HeaderStyle CssClass="heading" />
<AlternatingRowStyle CssClass="alternatingRowStyle" />
<PagerSettings FirstPageText="First" LastPageText="Last"
Mode="NextPreviousFirstLast" />
<PagerTemplate>
<asp:DropDownList ID="ddlCust" runat="server"
SelectedValue='<%# Eval("CustomerID") %>'
AutoPostBack="True" DataSourceID="sdsCustomers"
DataTextField="CompanyName" DataValueField="CustomerID"
OnSelectedIndexChanged="ddlCust_SelectedIndexChanged">
</asp:DropDownList>
</PagerTemplate>
</asp:DetailsView>

```

Updating data

You can use the details view control to update existing data in the database. To allow updating you need to set the `autoGenerateEditButton` property to true. You also need to set up your data source control so that it has an update statement generated. When updating in the details view the `datakeynames` property has to be set to the primary key of the table. If you want the details view control to appear in edit mode when it is first displayed you can set the `defaultMode` property to edit.

Activity 9. Updating customers

updateCustomers.aspx

In this exercise we will update the customers.

Add a data source control to your web page. Set it up to select all customers from the customer table.

Click on the advanced button to generate the update delete and insert SQL statements.



Add a details view control to your web page set it up so that it uses the data source control created. Place a tick in the enable editing checkbox. This will add an edit link to the details view. So that you can page through the records you should also place a tick in the enable paging checkbox.

DetailsView Tasks

Auto Format...

Choose Data Source: ▼

Configure Data Source...

Refresh Schema

Edit Fields...

Add New Field...

Move Field Up

Move Field Down

Remove Field

☒ Enable Paging

☐ Enable Inserting

☒ Enable Editing

☐ Enable Deleting

Edit Templates

View your page in the browser click on the edit link make a change and click on update. The changes will be saved to the database:

Untitled Page - Windows Internet Ex...

http://localhost:4809/ Live Search

File Edit View Favorites Tools Help

Untitled Page

CustomerID	ALFKI
CompanyName	Alfreds Futterkiste
ContactName	Maria Anders
ContactTitle	Sales Representative
Address	Obere Str. 57
City	Berlin
Region	
PostalCode	12209
Country	Germany
Phone	030-0074321
Fax	030-0076545

[Update](#) [Cancel](#)

1 2 3 4 5 6 7 8 9 10 ...

Local intranet 100%

Using templates when editing

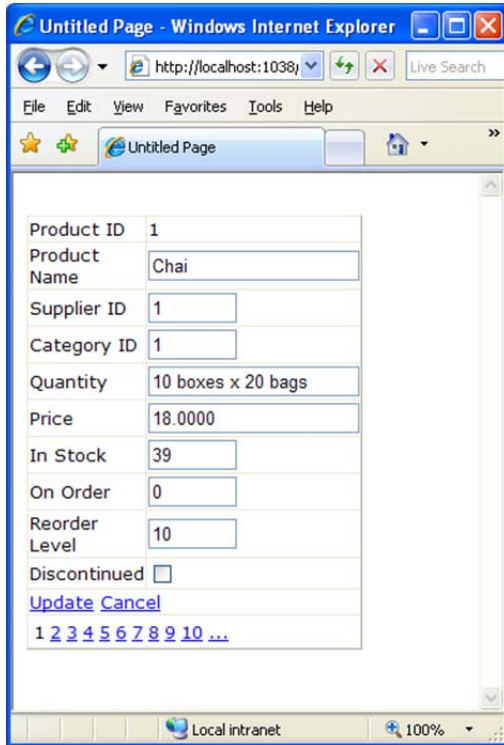
By default when editing records they are displayed in textboxes. For the customer table text boxes are probably adequate for gathering the data to be updated. For the product table however the category and supplier should be displayed as a drop down list to allow the user to select the category and supplier from the available list.

Activity 10. Using templates when editing

editingProducts.aspx

Create a web page which will display the products in a detail view. Allow editing for the details view and paging. Make sure your data source control is set up to generate an update, insert and delete statement.

View your page go into edit mode.



The screenshot shows a web browser window titled 'Untitled Page - Windows Internet Explorer'. The address bar shows 'http://localhost:1038/'. The page displays a form for editing a product. The form fields are as follows:

Product ID	1
Product Name	Chai
Supplier ID	1
Category ID	1
Quantity	10 boxes x 20 bags
Price	18.0000
In Stock	39
On Order	0
Reorder Level	10
Discontinued	<input type="checkbox"/>

Below the form are two links: 'Update' and 'Cancel'. At the bottom of the form, there is a pagination control showing '1 2 3 4 5 6 7 8 9 10 ...'.

The category ID and Supplier ID should really be displayed as drop down lists. To achieve this we need to make these template columns. We also should display the category name and supplier company name when the details view is not in edit mode.

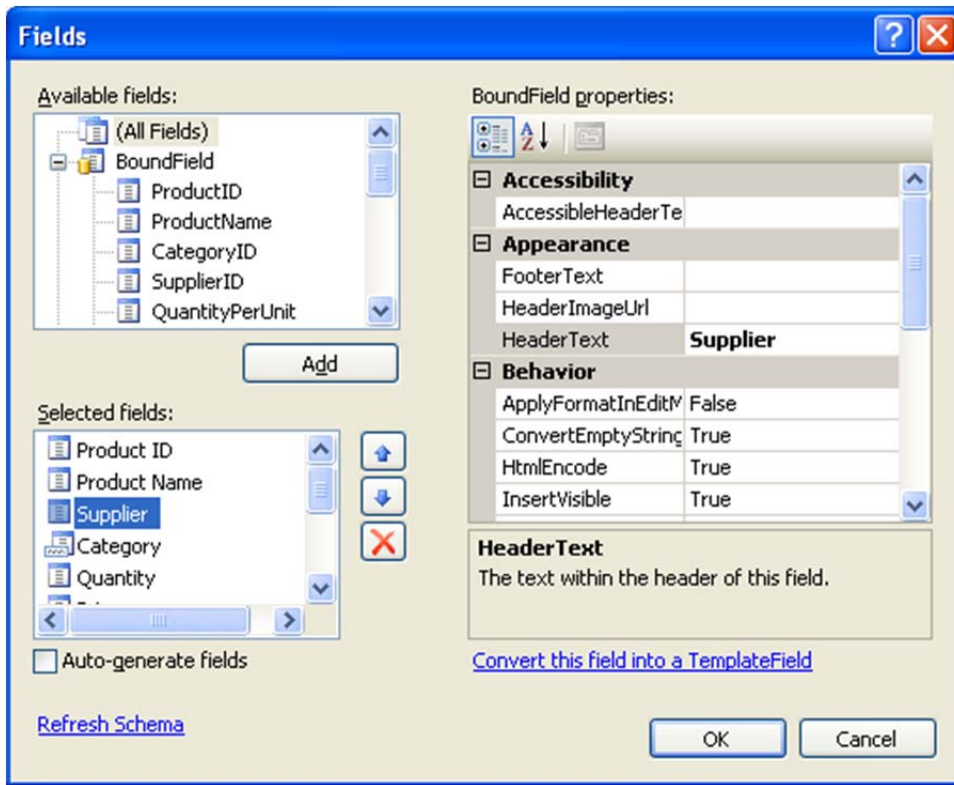
Modify the select statement to retrieve the category name and the supplier company name. The SQL statement should look like this:

```
SELECT [ProductID], [ProductName], products.[CategoryID], suppliers. [SupplierID],  
[QuantityPerUnit], [Discontinued], [ReorderLevel], [UnitsOnOrder], [UnitsInStock],  
[UnitPrice], companyName, categoryName  
FROM [Products]  
inner join categories on products.categoryID = categories.categoryID  
inner join suppliers on suppliers.supplierID = products.supplierID
```

Now the category name and company name will be displayed instead of the IDs.

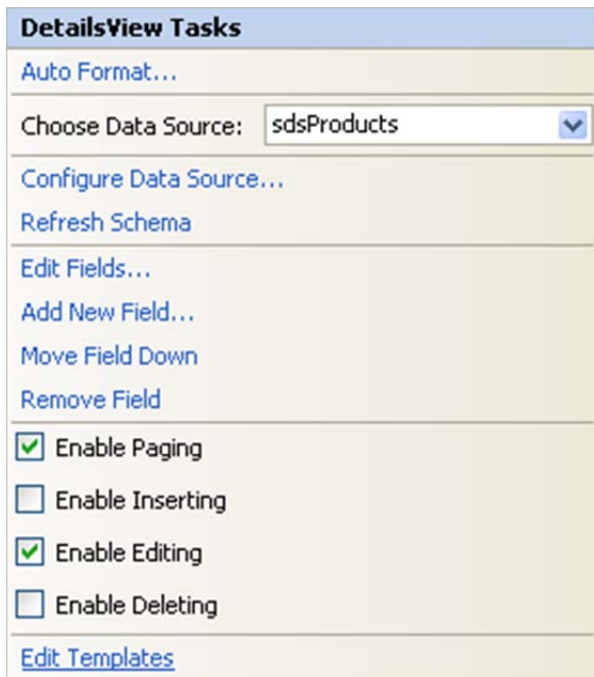
To display the category and supplier as a drop down list we need to add 2 more data source controls. One to retrieve the supplier company name and ID and one to retrieve the category name and ID.

Then you need to convert the category and supplier fields into template columns. To do that select edit fields from the details view smart tag, select the category and click on "convert this field into a template field"



Do that for both the category and supplier field.

Now select "edit template" from the details view smart tag.



This will open the templates editor. The supplier item template will display a label. Rename this label to lblSupplier. Check the data binding it should be set up to display the company name.

lblSupplier DataBindings

Select the property to bind to. You can then bind it by selecting a field. Alternatively, you can bind it using a custom code expression.

Bindable properties:

- Enabled
- Text**
- Visible

☐ Show all properties

Binding for Text

☒ Field binding:

Bound to:

Format:

Sample:

☒ Two-way databinding

☐ Custom binding:


Code expression:

[Refresh Schema](#)

OK Cancel

Now view the edit item template for the supplier. It will contain a textbox. Remove this textbox and place a drop down list there instead. Set the data source to the data source control you set up to retrieve the supplier company name and ID. Display the company name and store the supplier ID for the value.

Data Source Configuration Wizard

 **Choose a Data Source**

Select a data source:

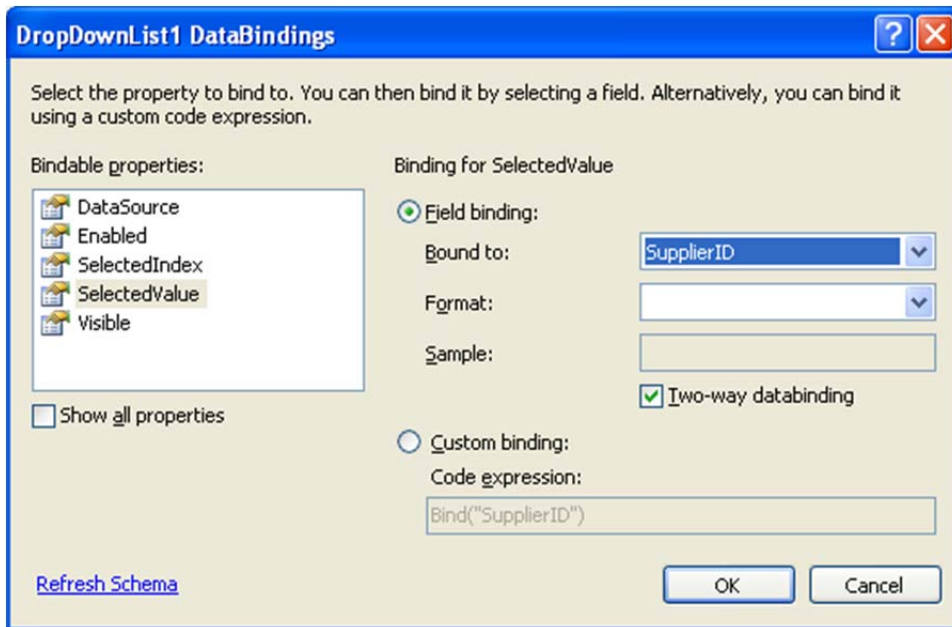
Select a data field to display in the DropDownList:

Select a data field for the value of the DropDownList:

[Refresh Schema](#)

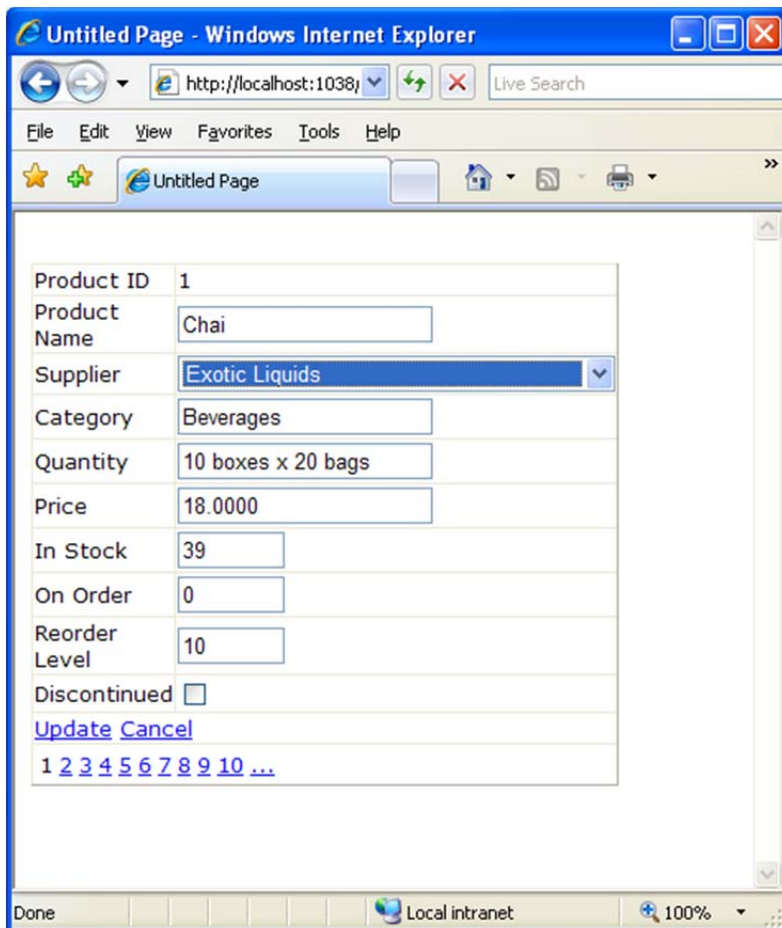
OK Cancel

Click on OK and select "edit data binding" from the dropdown list tasks. Set the supplierID for the selected value.



This will make sure that the selected supplier displayed when entering edit mode is the supplier allocated to that product.

Now view the web page you should see the supplier displayed in a drop down.



Repeat the same thing for the category field.

Inserting

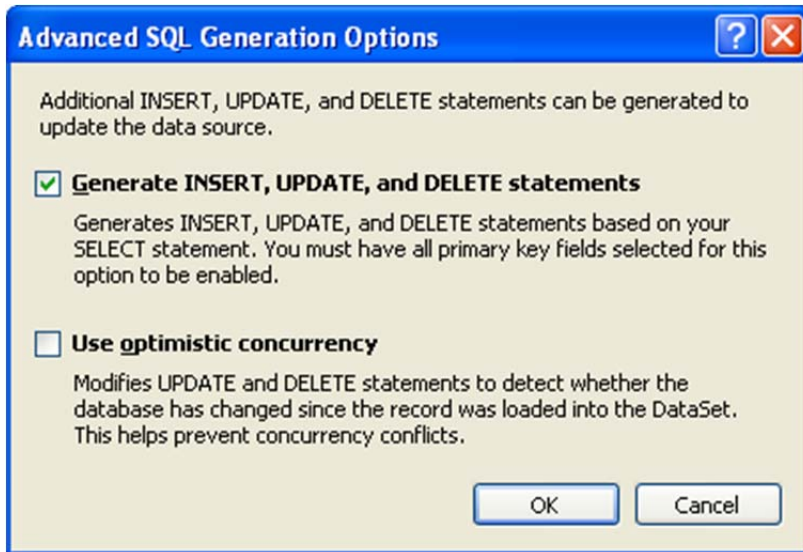
You can use the details view to insert new records in the database. You need to set the `autoGenerateInsertButton` property to true. This property automatically generates the user interface for inserting a new record. If you want the details view control to display an insert form by default you can assign the value `insert` to the details view control's `defaultMode` property.

Activity 11. Inserting

insertCategory.aspx

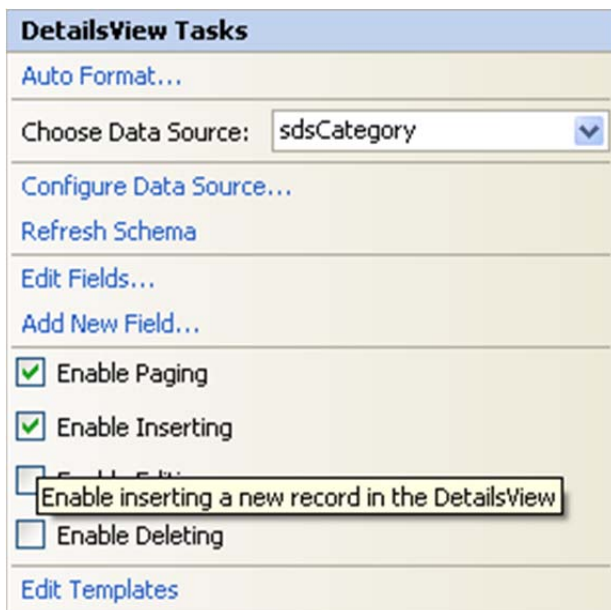
We will insert a new category through the details view control.

Create a new web page which will contain a data source control which will retrieve the category name, description and ID. Make sure you generate the insert update and delete statements for the SQL statement.

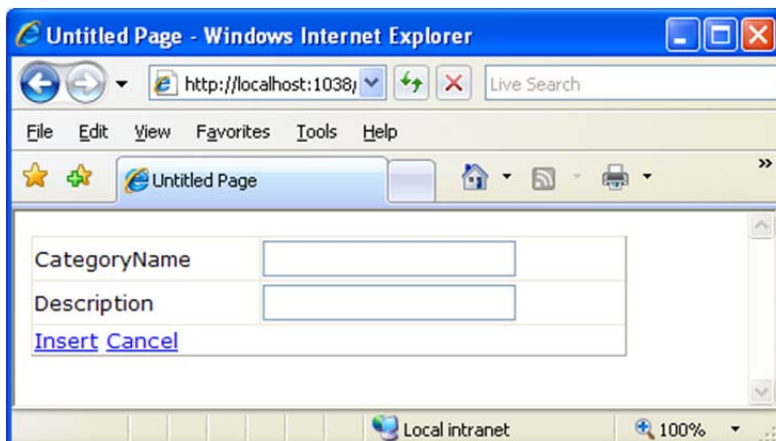


Use a details view control to display the categories. Allow paging.

Enable inserting by placing a tick in the enable inserting checkbox. This will add a new link to your details view.



View your web page and click on the new link.



Add a new category it should now be added to your database.

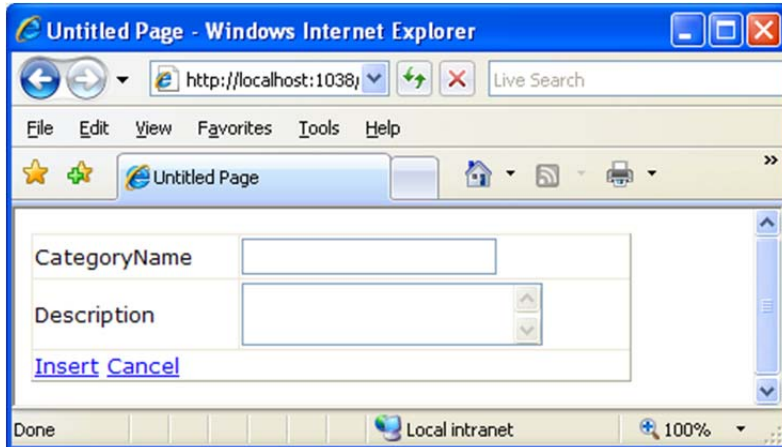
Activity 12. Template field for insert

templateInsert.aspx

In the above exercise the description is displayed as a single line textbox. It would be better if it was displayed as a multiline textbox to allow the user to easily enter a large amount of text.

To make the textbox multiline you need to convert the description field to a template column. Once converted you can edit the insertField template for the field. The description is already displayed as a textbox you just need to change it to multiline.

When in insert mode it should now look like this:



You may have noticed that the categoryID is not displayed when in insert mode. This is because the category ID is the primary key of the category table and automatically generated by the database when a new row is inserted into the table. It does not make sense to allow the user to provide a value for this column. For this reason it has a readonly property set to true. Any field you would not like to allow the user to insert into should be set as readonly.

Deleting

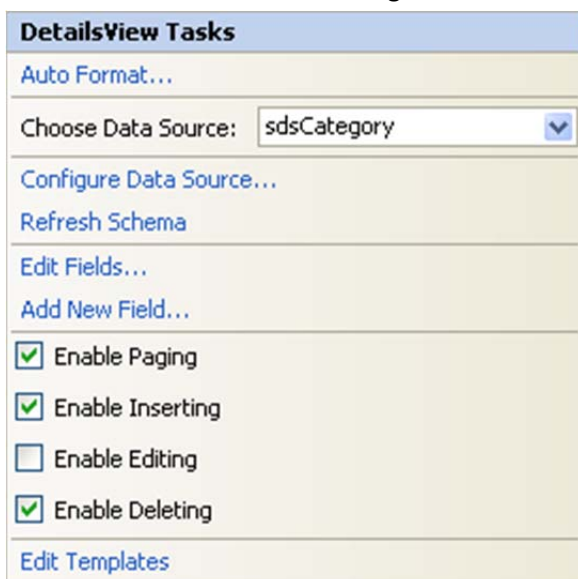
Deleting data with the details view control can be done by setting the autogeneratedDeleteButton property to true.

Activity 13. Deleting categories

deleteCategory.aspx

Now that you have added a few categories to the database lets delete them.

Place a tick in the enable deleting checkbox.



View you web page and make sure you only delete the records you added yourself otherwise you will cause a referential integrity error.

DetailsView control events

The details view control supports the following events:

- **DataBinding** – Raised immediately before the detailsview control is bound to its data source
- **DataBound** – Raised immediately after the detailsView control is bound to its data source
- **ItemCommand** – Raised when any control contained in the details view control raises an event (e.g. a button click)
- **ItemCreated** – Raised when a detailsView renders a data item
- **ItemDeleting** – Raised immediately before the data is deleted
- **ItemDeleted** – Raised immediately after the item is deleted
- **ItemInserting** – Raised immediately before a data item is inserted
- **ItemInserted** – Raised immediately after a data item is inserted
- **ItemUpdating** – Raised immediately before an item is updated
- **ItemUpdated** – Raised immediately after an item is updated
- **ModeChanging** – Raised immediately before the details view control's mode is changed
- **ModeChanged** – Raised immediately after the details view control's mode is changed
- **PageIndexChanging** – Raised immediately before the current page is changed
- **PageIndexChanged** – Raised immediately after the current page is changed

Activity 14. Using details view events

controlEvents.aspx

In this exercise we will use the details view databound event to set focus to the textbox when in insert mode.

To add the event click on the lightning bolt when the details view is selected and double click on the databound event.

```
protected void DetailsView1_DataBound(object sender, EventArgs e)
{
    if (DetailsView1.CurrentMode == DetailsViewMode.Insert)
    {
        DetailsView1.Focus();
    }
}
```

Activity 15. Details view events

Hide the details view control after a successful insert. An insert will only be successful if 1 row was affected. To complete this activity you will need to think about:

- Which event might you use to execute some code after a row has inserted?
- What can you use to check for the number of affected rows?

The answers are not necessarily in these notes you may need to search on Google.

References

ASP.NET 2.0 Unleashed Chapter 12