Contents

Using the form view control

The form view control can do anything the details view control can do. You can use it to display, page, edit, insert and delete database records. The form view control is entirely template driven. The form view control provides you with more control over the layout.

Displaying data

To display data you need to link a form view control to a data source control. You then need to create an item template.

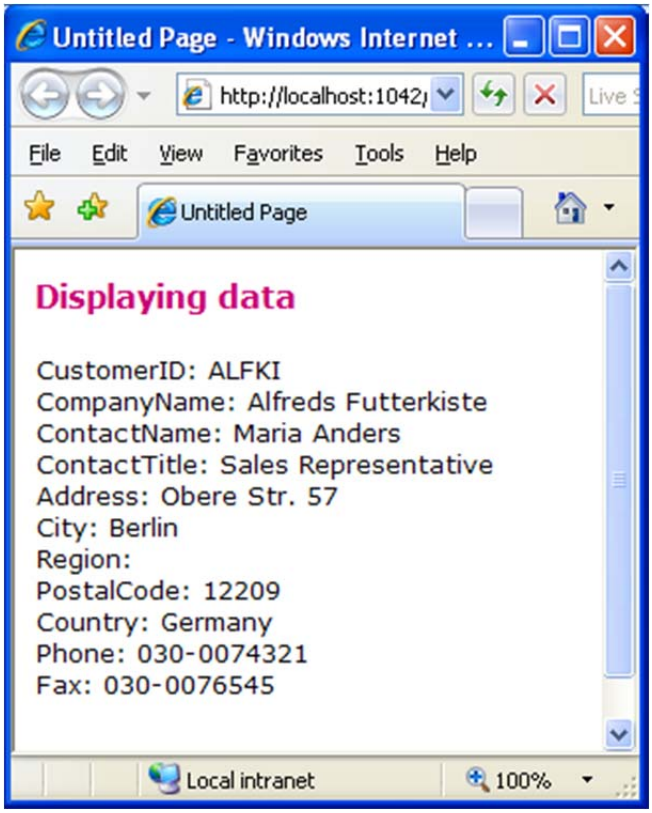
Activity 1. Displaying data

default.aspx

Create a new web page with an sql data source control on it. Configure it to retrieve the customer details for customer ALFKI.

Add a form view control on the page and link to the data source control.

View the page.

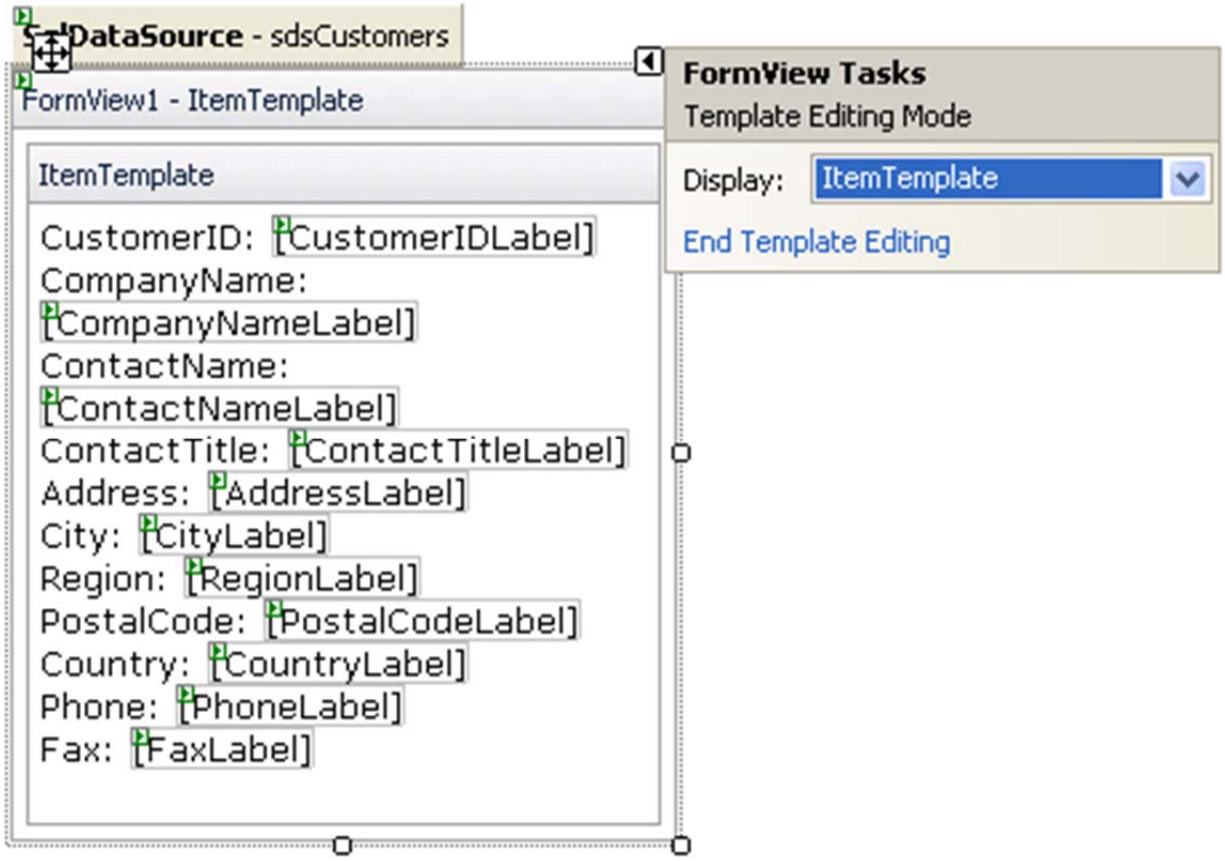


Formatting the data

The data we see above is displayed using the item template. We can add any ASP.NET and HTML control in the item template.

To format the item template select edit templates from the form view smart tag. The available templates are:

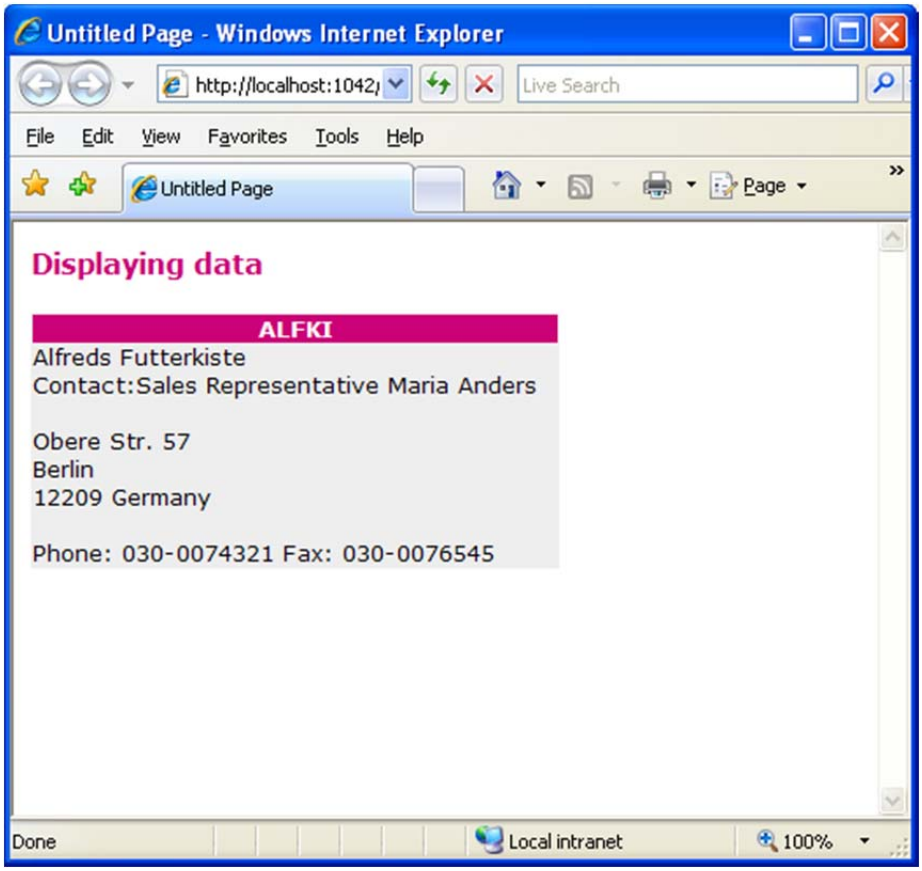
- ItemTemplate
- FooterTemplate
- EditItemTemplate
- InsertItemTemplate
- HeaderTemplate
- EmptyDataTemplate
- PagerTemplate



Activity 2. Formatting the data

displayData2.aspx

Modify the above exercise so that it looks like this:



You will need to use a style sheet:

```
body
{
      font-family: Verdana;
      font-size: small;
}

.bold
{
      font-weight: bold;
}

h1
{
      color: #CC0077;
      font-size: medium;
}

.custID
{
      width:350px;
      background-color: #CC0077;
      color: White;
      font-weight: bold;
      display: block;
      text-align: center;
}

.customer
{
      background-color: #EEEEEE;
      width:350px;
}
```

The custID class will be assigned with the label displaying the customer ID and the customer class will be assigned to the whole form view.

```
<asp:FormView ID="fvCustomers" runat="server" DataKeyNames="CustomerID"
DataSourceID="sdsCustomers" CssClass="customer">
```

The code in the item template looks like this:

```
<ItemTemplate>
      <asp:Label ID="CustomerIDLabel" runat="server"
cssClass="custID" Text='<%# Eval("CustomerID") %>' />
       <asp:Label ID="CompanyNameLabel" runat="server"
Text='<%# Bind("CompanyName") %>' />
<br />
Contact:<asp:Label ID="ContactTitleLabel" runat="server"
Text='<%# Bind("ContactTitle") %>' />
<asp:Label ID="ContactNameLabel" runat="server"
Text='<%# Bind("ContactName") %>' />
<br /><br />
<asp:Label ID="AddressLabel" runat="server"
Text='<%# Bind("Address") %>' />
<br />
<asp:Label ID="CityLabel" runat="server"
Text='<%# Bind("City") %>' />
<asp:Label ID="RegionLabel" runat="server"
Text='<%# Bind("Region") %>' />
<asp:Label ID="PostalCodeLabel" runat="server"
Text='<%# Bind("PostalCode") %>' />
```

```
<asp:Label ID="CountryLabel" runat="server"
Text='<%# Bind("Country") %>' />
<br /> <br />
Phone:<asp:Label ID="PhoneLabel" runat="server"
Text='<%# Bind("Phone") %>' />
 Fax:<asp:Label ID="FaxLabel" runat="server"
Text='<%# Bind("Fax") %>' />
</ItemTemplate>
```

You may have noticed the insert and edit item template have already been created for you, we are not using them at this stage.

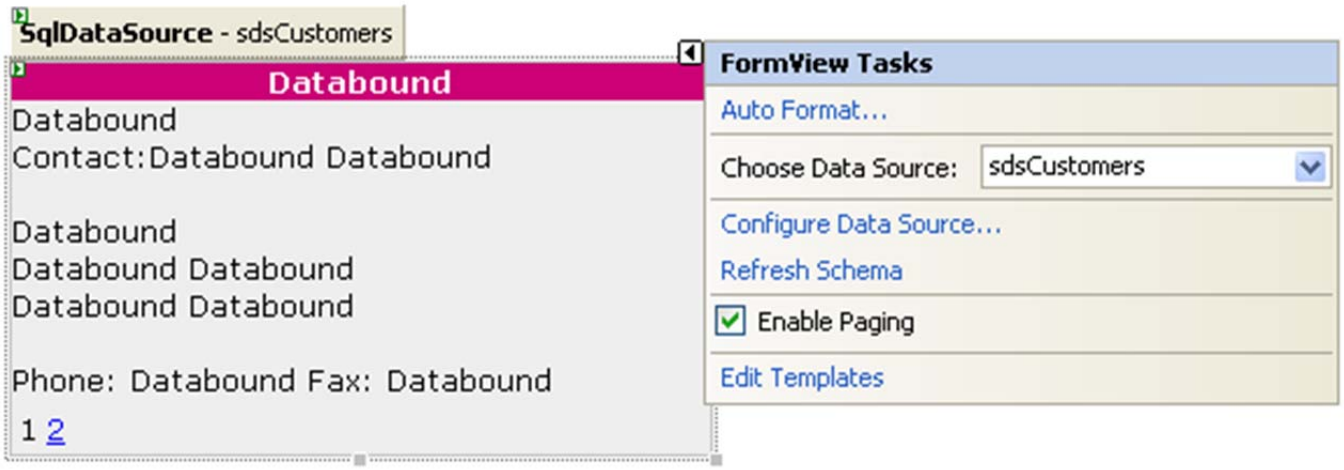
Paging

Just as you could page through the records in a details View control you can also page through the records of a form view control.

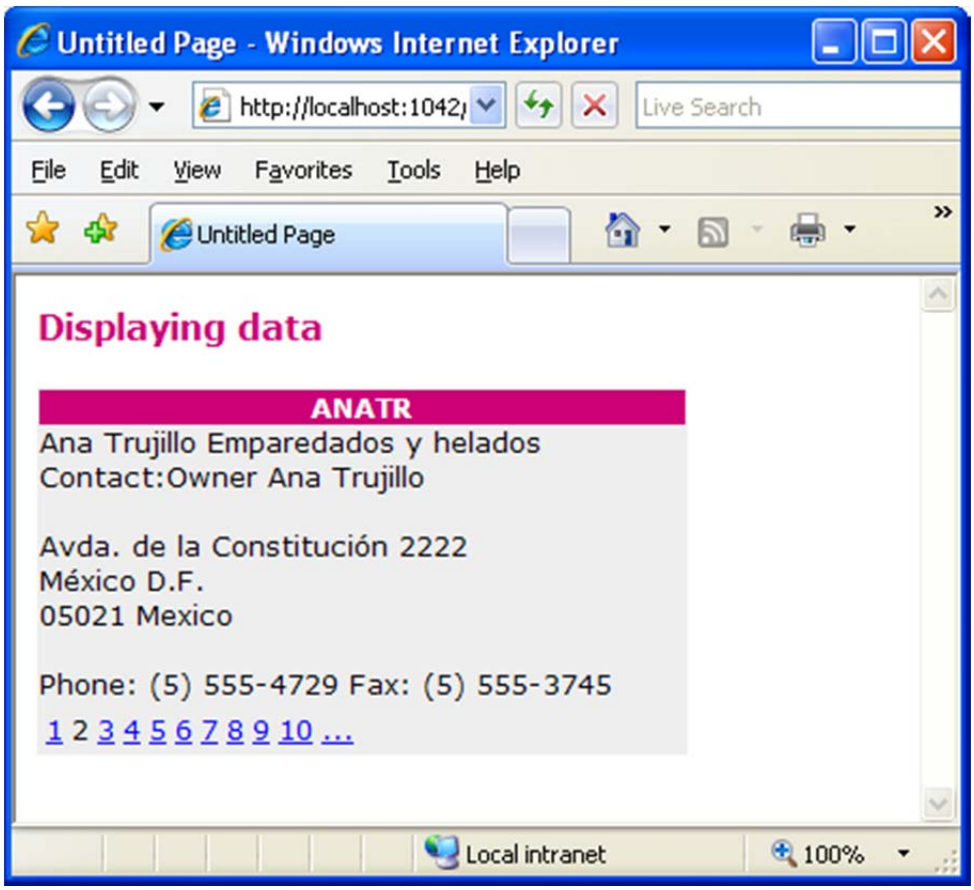
Activity 3. Paging

paging.aspx

Modify the above exercise to allow paging. You will also need to remove the where clause of your SQL statement so that all the customers are retrieved. To allow paging place a tick in the enable paging checkbox from the smart tag of the form view control.



Now view the page you should be able to page through the customer records.



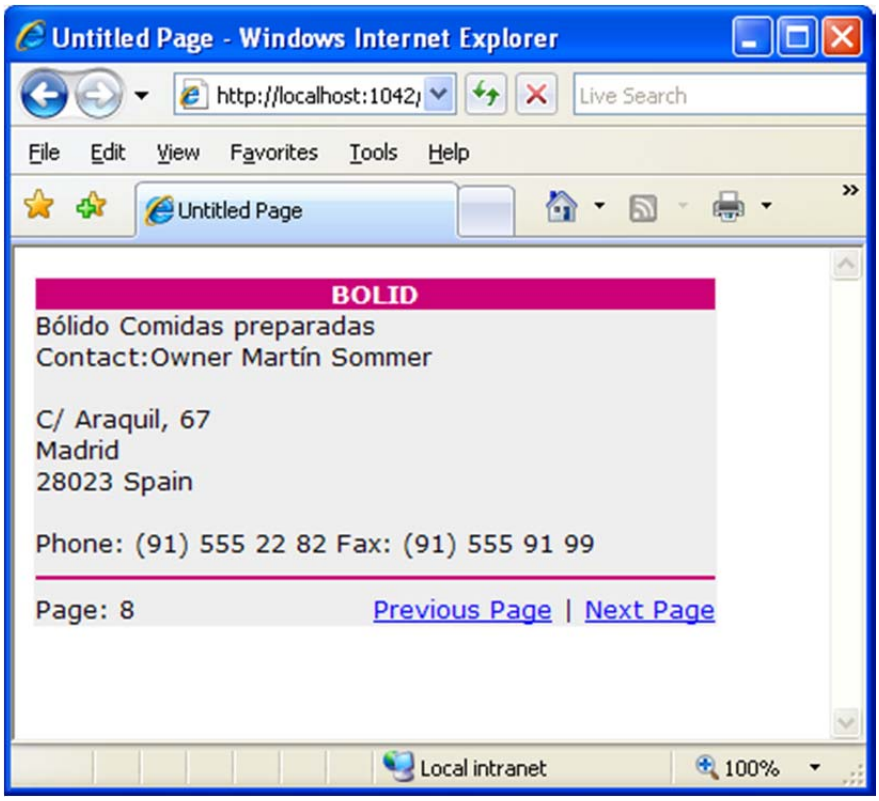
Pager Template

You can use the pager template when you need to customise the appearance of the paging interface.

Activity 4. Pager template

pagerTemplate.aspx

We will use the same exercise as above but this time display a top border, next and previous links and the current page number.



To set this up you need to modify the source code. Switch to source view and add the following code:

```
<PagerTemplate>
<hr />
<div class="pageLeft">
Page:
<asp:Label ID="lblPage" runat="server"
Text="<%# formView1.PageIndex + 1 %>"></asp:Label>
</div>
<div class="pageRight">
<asp:LinkButton ID="lnkPrevious" runat="server"
CommandArgument="Prev" CommandName="Page">
Previous Page</asp:LinkButton>
|
<asp:LinkButton ID="lnkNext" runat="server"
CommandArgument="Next" CommandName="Page">
Next Page</asp:LinkButton>
</div>
</PagerTemplate>
```

The <pagerTemplate> tag goes between the <asp:formView> tags

The first label lblPage displays the current page number. This is done by using the form view pageIndex property. Since the pageIndex starts counting from 0, 1 has to be added to it.

The next and previous links work by assigning the value Page to the commandName and Next or Prev to the command argument. The command name is used to specify the type of operation we will be needing in this case paging. Page is a special keyword which identifies that we will be paging when we click on the link. The command argument specifies how we will be paging. In our example either to the previous page by specifying prev or the next page by specifying next. The available values for commandArgument are:

- First – navigates to the first page
- Last – navigates to the last page
- Prev – navigates to the previous page
- Next – navigates to the next page
- Number - Navigates to a particular page number

You will also need the following changes to your style sheet

```
.pageLeft
{
      float: left;
}

.pageRight
{
      float:right;
      white-space: nowrap;
}
```

Editing data

The edit item template is automatically generated for you when the form view control is bound to the data source control. To allow editing to occur there are a few things you need to do:

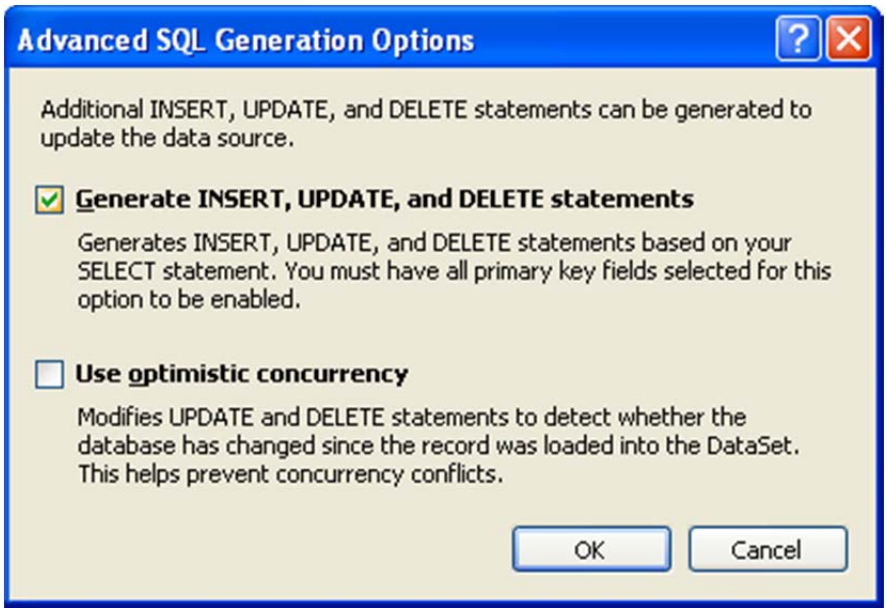
- Make sure the data source is set up to update delete and insert. Remember to generate the insert update and delete statements the SQL select statement must include the primary key.
- The dataKeyNames property of the form view control needs to be set to the primary key.
- An edit link needs to be added to the item template this link needs to have the commandName set to edit.
- The update and cancel links will already be generated for you the update and cancel links need to have the commandName set to update and cancel respectively.

If you would like to have the form view in edit mode by default you can set the defaultMode property to edit for the form view control.

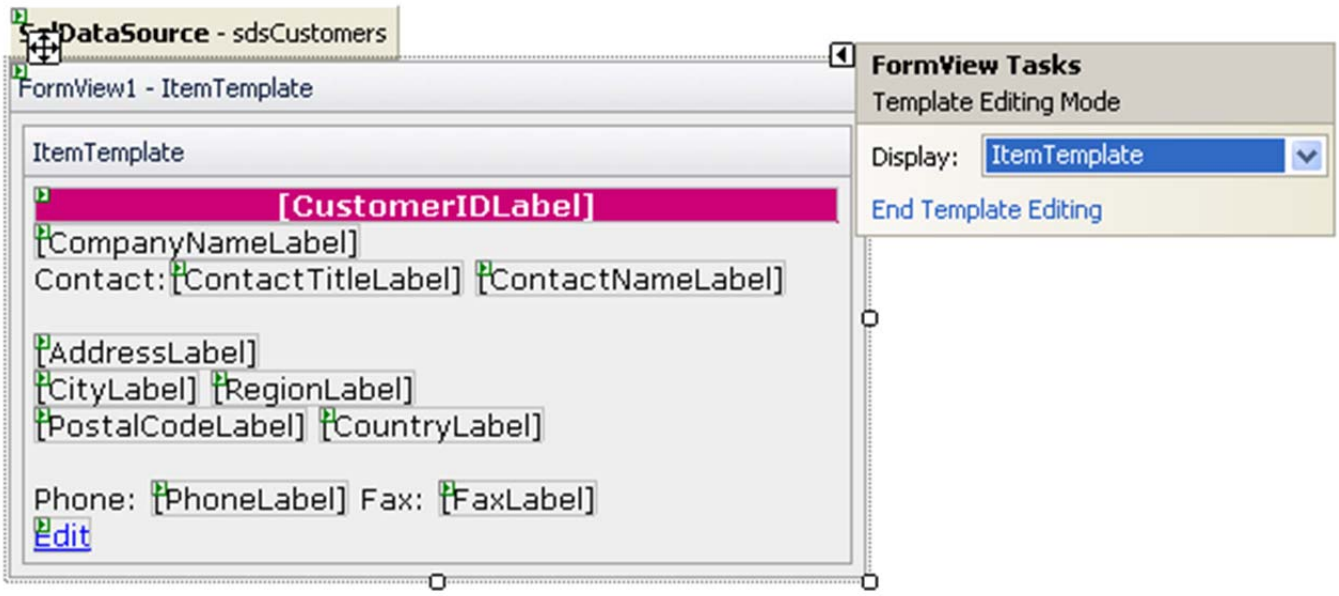
Activity 5. Editing data

editing.aspx

To allow editing of your previous exercise you need to set up the data source control so that you can insert, delete and update.



Next you need to add a link button to the item template. Set it up to display the text edit, set up the commandName to "Edit". The "Edit" command name indicates that when this link is clicked it will put the form view into edit mode and therefore display the edit item template.



Source for link button:

```
<asp:LinkButton ID="lbtEdit" runat="server" CommandName="Edit">Edit </asp:LinkButton>
```

The edit template which is generated for you is not formatted so you will need to make the following changes for each textbox:

```
<asp:Label ID="lblCompName" runat="server"  AssociatedControlID="CompanyNameTextBox"
Text="Company Name:"></asp:Label>

<asp:TextBox ID="CompanyNameTextBox" runat="server"
Text='<%# Bind("CompanyName") %>' />
<br />
```

The associatedControlID property changes the <asp:label> tag into an HTML <label> tag. Since we have added a label we need to set up the style sheet to format it so it floats to the left and has a width of 150px

```
label
{
      float: left;
      width: 150px;
}
```

Inserting data

You can use the form view control to insert new records. The insert template is already generated for you but in order to enable insert there a re a few things that need to be set up.

- Make sure the data source is set up to update delete and insert. Remember to generate the insert update and delete statements the SQL select statement must include the primary key.
- The dataKeyNames property of the form view control needs to be set to the primary key.
- A “New” link needs to be added to the item template the commandName property needs to be set to new.
- The insert and cancel links will already be generated for you the commandName property needs to be set to insert and cancel respectively

If you would like to have the form view in insert mode by default you can set the defaultMode property to insert for the form view control.

Activity 6. Inserting

insertCustomer.aspx

Modify the above exercise to allow inserting. The insertItemTemplate has already been generated but just as the editItemTemplate it is not well formatted. First we need to add a new linkButton to the item template.

```
<asp:LinkButton ID="lbtNew" runat="server" CommandName="New">New</asp:LinkButton> |
<asp:LinkButton ID="lbtEdit" runat="server" CommandName="Edit">Edit</asp:LinkButton>
```

You then need to modify the insertItemTemplate so that it contains a lable for each text box (similar to the edit Item template):

```
<asp:Label ID="CustomerIDLabel1" runat="server"
AssociatedControlID="CustomerIDTextBox" Text="Customer ID" />

<asp:TextBox ID="CustomerIDTextBox" runat="server"
Text='<%# Bind("CustomerID") %>' />
<br />
```

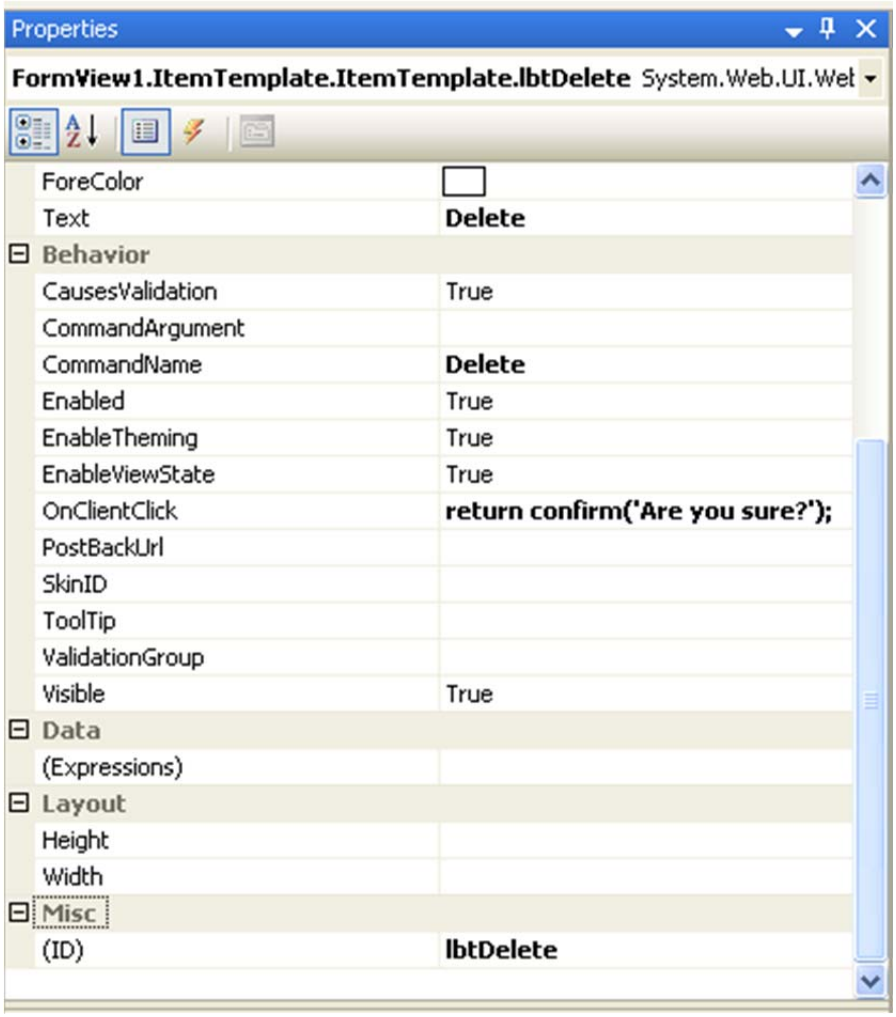
Deleting data

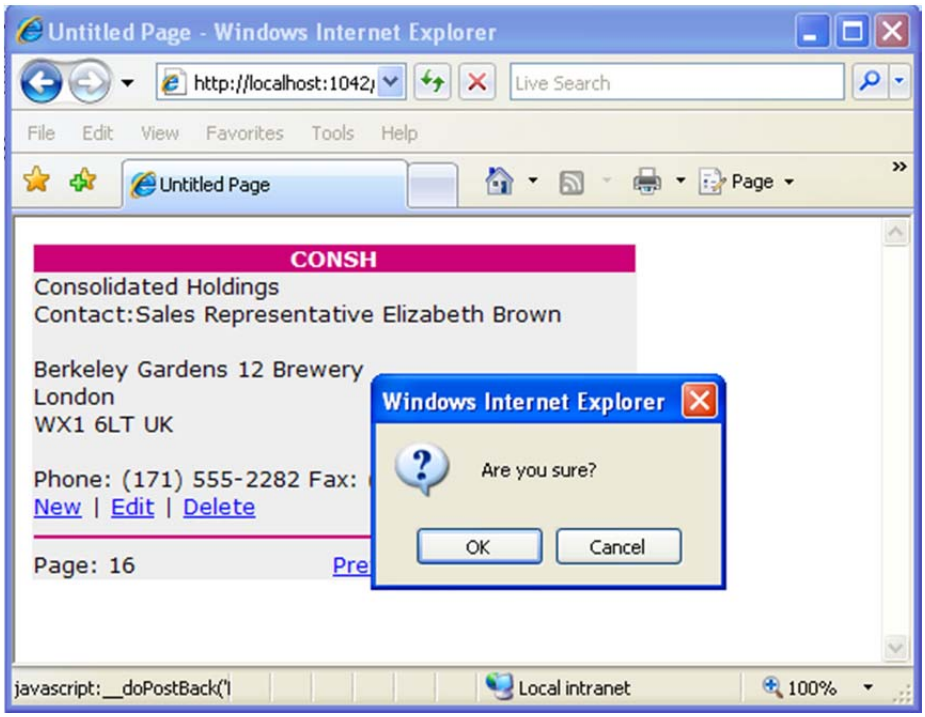
You can use the form view control to delete records. To do this you need to add a link button to the item template. This link button will need to have the commandName set to delete.

Activity 7. Deleting data

delete.aspx

Add a hyperlink button to the item template and set it up so that the text displayed is set to delete, the command name is set to delete and set the onClientClick property to “return confirm('Are you sure?');” this will cause a JavaScript confirm window to be displayed on deletion of this record. If ok is clicked the record will be deleted if cancel is clicked the record will not be deleted.





References

ASP.NET 2.0 unleashed Chapter 12