UNIT-II

ADO.NET

ADO.NET is a set of computer software components that programmers can use to access data and data services based on disconnected DataSets and XML. It is a part of the base class library that is included with the Microsoft .NET Framework. It is commonly used by programmers to access and modify data stored in relational database systems, though it can also access data in non-relational sources. ADO.NET is sometimes considered an evolution of ActiveX Data Objects (ADO) technology, but was changed so extensively that it can be considered an entirely new product.

ADO.NET is conceptually divided into *consumers* and *data providers*. The consumers are the applications that need access to the data, and the providers are the software components that implement the interface and thereby provide the data to the consumer.

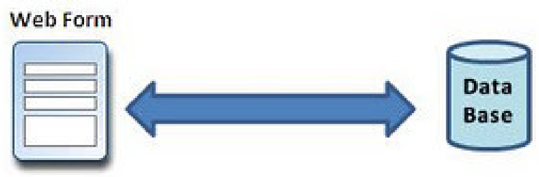
Functionality exists in Visual Studio IDE to create specialized subclasses of the DataSet classes for a particular database schema, allowing convenient access to each field through strongly typed properties. This helps catch more programming errors at compile-time and enhances the IDE's Intellisense feature.

DISCONNECTED DATA ACCESS

Connected Architecture of ADO.NET

The architecture of ADO.net, in which connection must be opened to access the data retrieved from database is called as connected architecture. Connected architecture was built on the classes connection, command, datareader and transaction.

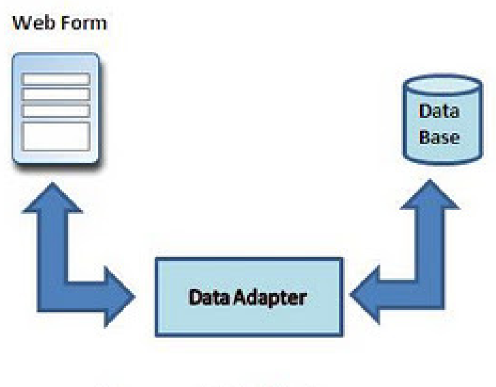
Connected architecture is when you constantly make trips to the database for any CRUD (Create, Read, Update and Delete) operation you wish to do. This creates more traffic to the database but is normally much faster as you should be doing smaller transactions.



Connected Architecture

Disconnected Architecture in ADO.NET

The architecture of ADO.net in which data retrieved from database can be accessed even when connection to database was closed is called as disconnected architecture. Disconnected architecture of ADO.net was built on classes connection, dataadapter, commandbuilder and dataset and dataview.
Disconnected architecture is a method of retrieving a record set from the database and storing it giving you the ability to do many CRUD (Create, Read, Update and Delete) operations on the data in memory, then it can be re-synchronized with the database when reconnecting. A method of using disconnected architecture is using a Dataset.



Disconnected Architecture

DataReader is Connected Architecture since it keeps the connection open until all rows are fetched one by one

DataSet is **DisConnected** Architecture since all the records are brought at once and there is no need to keep the connection alive
Difference between Connected and disconnected architecture

Connected	**Disconnected**
It is connection oriented.	It is dis_connection oriented.
Datareader	DataSet
Connected methods gives faster performance	Disconnected get low in speed and performance.
connected can hold the data of single table	disconnected can hold multiple tables of data
connected you need to use a read only forward only data reader	disconnected you cannot
Data Reader can't persist the data	Data Set can persist the data
It is Read only, we can't update the data.	We can update data

Example

Create Database “Student”

```
CREATE TABLE [dbo].[Student]
(
   [ID] [int] PRIMARY KEY IDENTITY(1,1) NOT NULL,
   [Name] [varchar](255) NULL,
   [Age] [int] NULL,
   [Address] [varchar](255) NULL
)

INSERT INTO Student([Name],[Age],[Address])VALUES('NAME 1','22','PUNE')
INSERT INTO Student([Name],[Age],[Address])VALUES('NAME 2','25','MUMBAI')
INSERT INTO Student([Name],[Age],[Address])VALUES('NAME 3','23','PUNE')
INSERT INTO Student([Name],[Age],[Address])VALUES('NAME 4','21','DELHI')
INSERT INTO Student([Name],[Age],[Address])VALUES('NAME 5','22','PUNE')
```

HTML

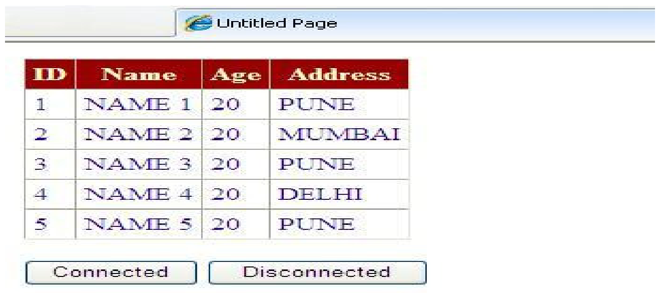
```
<html xmlns="http://www.w3.org/1999/xhtml">
<head id="Head1" runat="server">
   <title>Untitled Pagetitle>
</head>
<body>
   <form id="form1" runat="server">
   <div>
      <asp:GridView ID="GridView1" runat="server" BackColor="White"
BorderColor="#CC9966"
         BorderStyle="None" BorderWidth="1px" CellPadding="4">
         <FooterStyle BackColor="#FFFFCC" ForeColor="#330099" />
         <RowStyle BackColor="White" ForeColor="#330099" />
         <PagerStyle BackColor="#FFFFCC" ForeColor="#330099"
HorizontalAlign="Center" />
         <SelectedRowStyle BackColor="#FFCC66" Font-Bold="True"
ForeColor="#663399" />
         <HeaderStyle BackColor="#990000" Font-Bold="True" ForeColor="#FFFFCC" />
      </asp:GridView>
      <br />
      <asp:Button ID="Connected" runat="server" OnClick="Connected_Click"
Text="Connected" />
      <asp:Button ID="Disconnected" runat="server" EnableTheming="False"
OnClick="Disconnected_Click"
         Text="Disconnected" />
   </div>
   </form></body></html>
```

Code Behind

```
String StrSQL = "", StrConnection = "";
        protected void Page_Load(object sender, EventArgs e)
        {
            StrSQL = "SELECT * FROM Student";
              StrConnection = "Data Source=ServerName;Initial Catalog=Database;User
ID=Username;Password=password";
        }

        protected void Connected_Click(object sender, EventArgs e)
        {
            using (SqlConnection objConn = new SqlConnection(StrConnection))
            {
                SqlCommand objCmd = new SqlCommand(StrSQL, objConn);
                objCmd.CommandType = CommandType.Text;
                objConn.Open();
                SqlDataReader objDr = objCmd.ExecuteReader();
                GridView1.DataSource = objDr;
                GridView1.DataBind();
                objConn.Close();
            }
        }

        protected void Disconnected_Click(object sender, EventArgs e)
        {
            SqlDataAdapter objDa = new SqlDataAdapter();
            DataSet objDs = new DataSet();
            using (SqlConnection objConn = new SqlConnection(StrConnection))
            {
                SqlCommand objCmd = new SqlCommand(StrSQL, objConn);
                objCmd.CommandType = CommandType.Text;
                objDa.SelectCommand = objCmd;
                objDa.Fill(objDs, "Student");
                GridView1.DataSource = objDs.Tables[0];
                GridView1.DataBind();
            }
        }
```



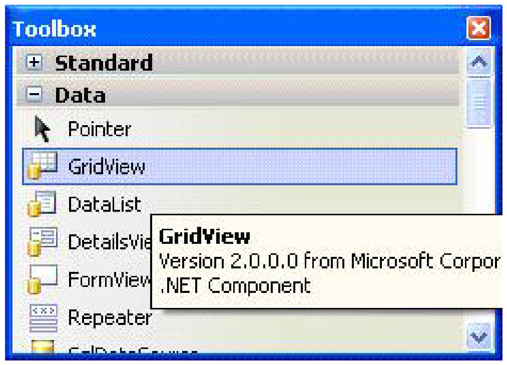
ID	Name	Age	Address
1	NAME 1	20	PUNE
2	NAME 2	20	MUMBAI
3	NAME 3	20	PUNE
4	NAME 4	20	DELHI
5	NAME 5	20	PUNE

GRIDVIEW CONTROL

Shortcomings of DataGrid (ASP.NET 1.x)

- DataGrid requires you to write custom code for handling common operations like sorting, paging and manipulation of data in DataGrid
- DataGrid when bound to DataSource control can only support select operation on DataSource. Updating DataSource through DataGrid can be done only through custom ADO.NET code
- DataGrid supports a restricted event model.
- DataGrid does not support adaptive rendering on different platforms

Above limitations apply only to ASP.NET 1.x. ASP.NET 2.0 enhanced DataGrid Control is available which overcomes some of above mentioned shortcomings



GridView in Toolbox

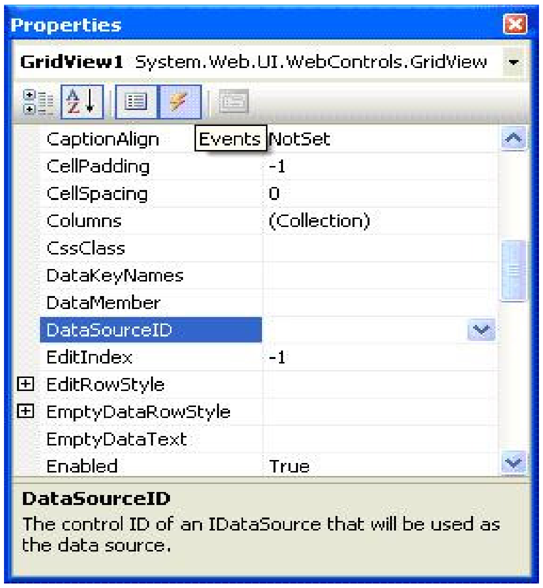
In ASP.NET 2.0, Microsoft has introduced GridView control which too displays tabular data in a grid. With GridView control, you can display, edit, and delete data directly from different kinds of data sources with out writing any single piece of code

GridView Control Features

- Enhanced data source binding capabilities (Direct interaction with DataSource with any writing any ADO.NET code)
- Built-in support for sorting and paging functionalities
- Improved Design time features(Smart Panel Tag)
- Customized pager user interface with PagerTemplate property
- Additional Column types(ImageField)
- New Event model with support for pre event and post event operations

Enhanced Data Source binding capabilities

GridView control automates paging, sorting and database operations like updating, deletion and selection of data with out writing single piece of code. This is possible with new property DataSourceID which allows performing paging, sorting and all database operations directly with the DataSource. For example, GridView can directly update, delete and perform paging, sorting with a sqldatasource with out any custom code. On contrary, DataGrid can declaratively bind a DataSource control but it can do it for only displaying data. All other database operations have to be coded manually.



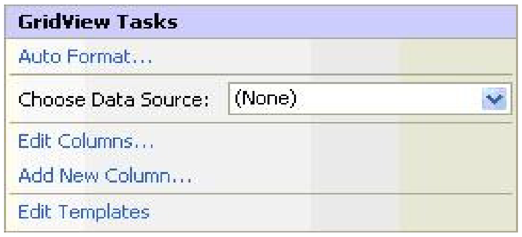
CaptionAlign	NotSet
CellPadding	-1
CellSpacing	0
Columns	(Collection)
CssClass	
DataKeyNames	
DataMember	
DataSourceID	
EditIndex	-1
⊞ EditRowStyle	
⊞ EmptyDataRowStyle	
EmptyDataText	
Enabled	True

DataSourceID
The control ID of an IDataSource that will be used as the data source.

Setting DataSourceID for GridView

Improved Design time features (Smart Panel Tag)

GridView control includes support for Smart tag panel that allows setting the design-time properties easily like auto formatting and adding new template columns.



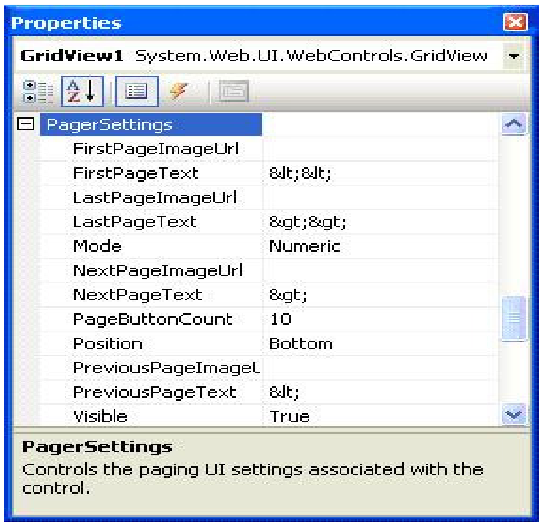
Smart Tag associated with GridView

Customized Pager User Interface

GridView control provides built-in support for paging by using new property called PagerSetting. PagerSetting property supports four modes:

- **Numeric** displays numbered links for pages
- **NextPrevious** displays only next/previous links to traverse pages
- **NumericFirstlast** displays Numbered buttons as well as first/last links
- **NextPreviousFirstlast** displays First, Next, Previous, Last combo links

You can also customize the look and feel of each page by setting various properties like PagerStyle property and using PagerTemplate to change look and style of buttons and pages. You can see properties of GridView1 in below listing



Properties

GridView1 System.Web.UI.WebControls.GridView

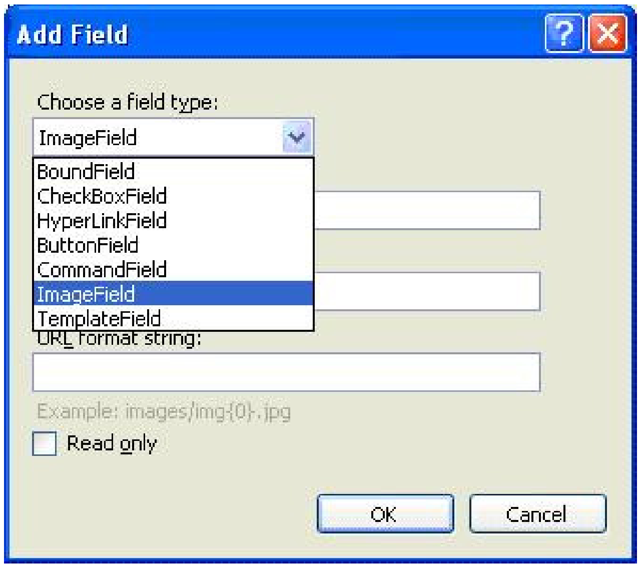
PagerSettings	
FirstPageImageUrl	
FirstPageText	<<
LastPageImageUrl	
LastPageText	>>
Mode	Numeric
NextPageImageUrl	
NextPageText	>
PageButtonCount	10
Position	Bottom
PreviousPageImageL	
PreviousPageText	<
Visible	True

PagerSettings
Controls the paging UI settings associated with the control.

PagerSettings property for GridView

New Column Types

GridView provides supports for new columns like checkbox field and ImageField which allows binding Boolean data and image data to new columns.



Add Field

Choose a field type:

ImageField

- BoundField
- CheckBoxField
- HyperLinkField
- ButtonField
- CommandField
- ImageField
- TemplateField

URL format string:

Example: images/img{0}.jpg

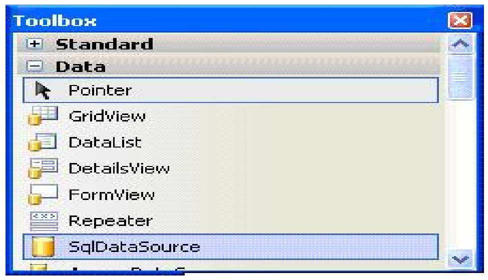
Read only

Adding new column dialog box

Event Model

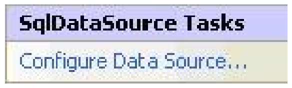
Another striking difference between DataGrid and GridView is in the event model. GridView supports both pre-operation and post-operation events. For example: when ever you update a row RowUpdating is fired before the update and RowUpdated events after the GridView automatically updated a row. This way you can get a chance to inspect the input values just before the event is fired and evaluate the returned results just after the event is fired. Similarly for Delete operation, GridView supports RowDeleting(pre-event) and RowDeleted events(post-event).

Let's consolidate our understanding with an example. We will use SQL Server 2000 database for demonstration purpose

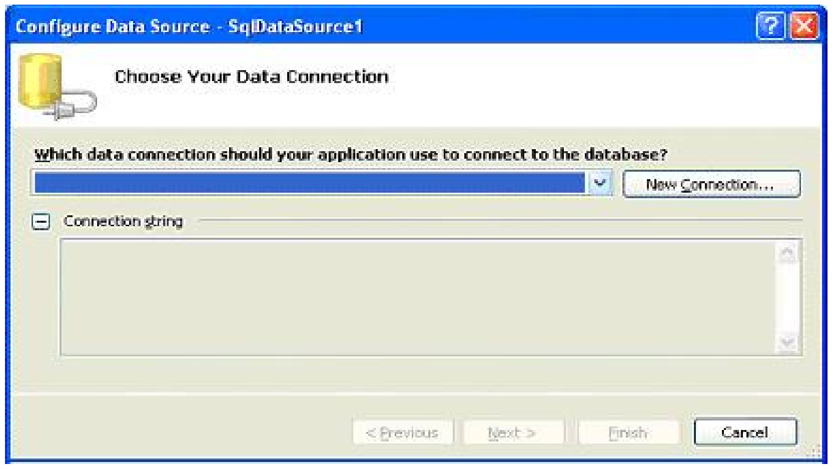


.

Drag and drop the sqlDataSource from toolbox to page. When you drop the sqlDataSource onto page you will be prompted to configure Data Source using below Smart Tag. Click configure Data Source.



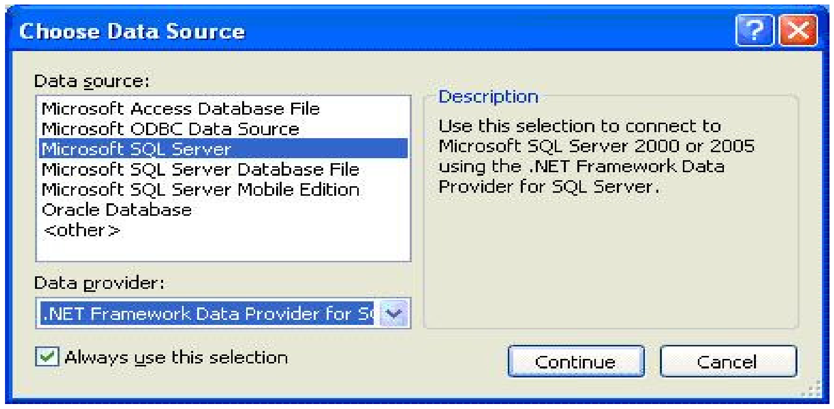
Configure Data Source wizard appears as below:


Configure Data Source - SqlDataSource1

Choose Your Data Connection

Which data connection should your application use to connect to the database?

Click on New connection to establish new connection with data source.


Choose Data Source

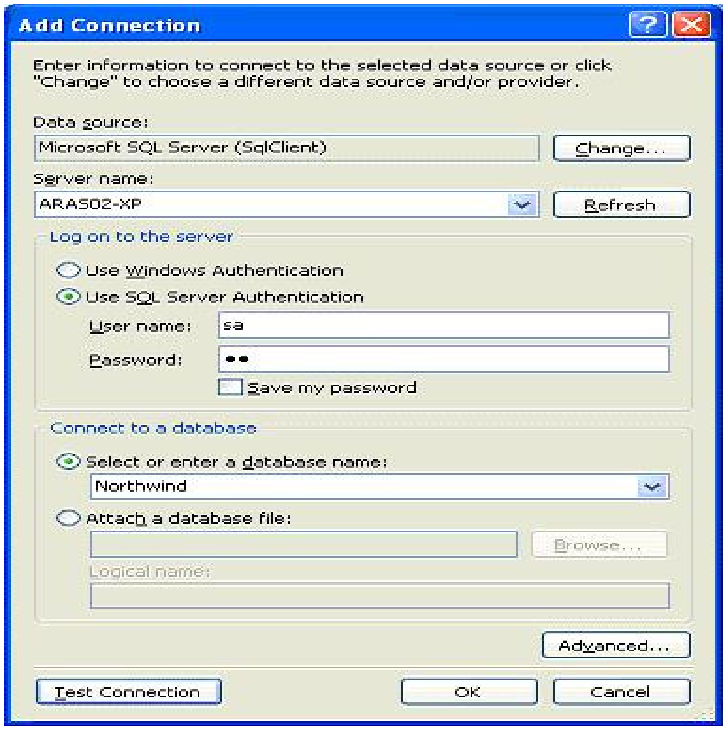
Data source:
Microsoft Access Database File
Microsoft ODBC Data Source
Microsoft SQL Server
Microsoft SQL Server Database File
Microsoft SQL Server Mobile Edition
Oracle Database
<other>

Description
Use this selection to connect to Microsoft SQL Server 2000 or 2005 using the .NET Framework Data Provider for SQL Server.

Data provider: .NET Framework Data Provider for S

Always use this selection

Choose SQL Server Datasource and click continue button.



Add Connection

Enter information to connect to the selected data source or click "Change" to choose a different data source and/or provider.

Data source: Microsoft SQL Server (SqlClient)

Server name: ARAS02-XP

Log on to the server

( ) Use Windows Authentication

(•) Use SQL Server Authentication

User name: sa

Password: ••

[ ] Save my password

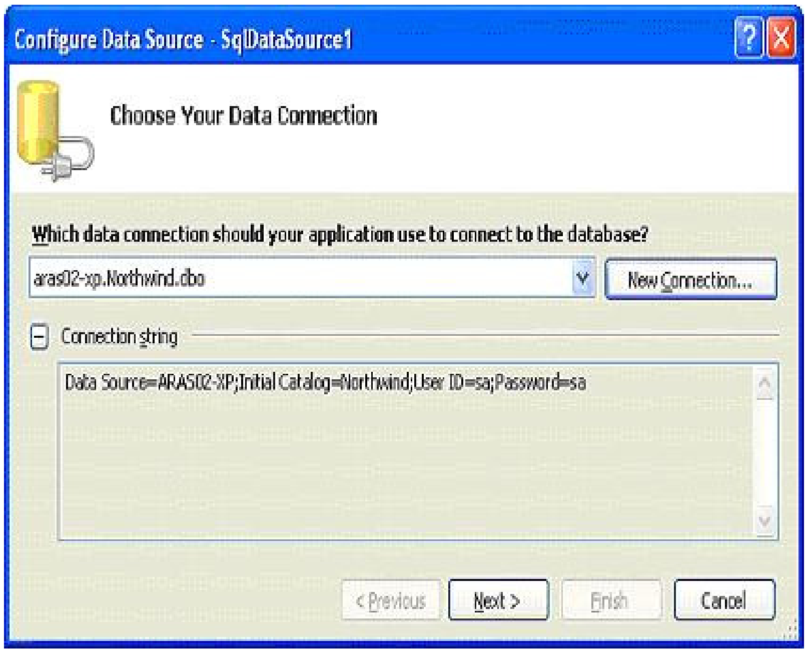
Connect to a database

(•) Select or enter a database name: Northwind

( ) Attach a database file:

Logical name:

Decide whether you would like to go for Windows Authentication or SQL Server authentication. Click test connection button. Once satisfied with connection, click ok button.



Configure Data Source - SqlDataSource1

Choose Your Data Connection

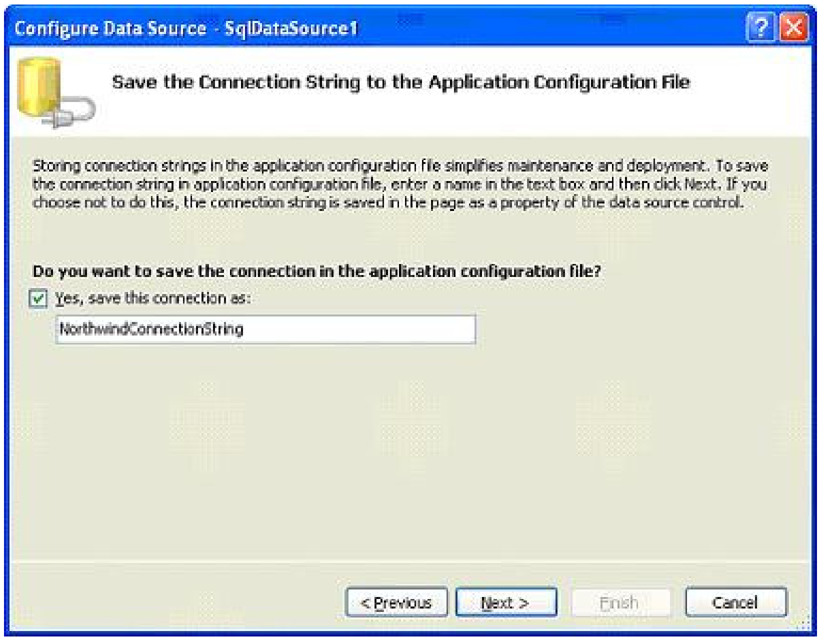
Which data connection should your application use to connect to the database?

aras02-xp.Northwind.dbo

Connection string

Data Source=ARAS02-XP;Initial Catalog=Northwind;User ID=sa;Password=sa

Choose Data connection to connect to database and click next.



Save the Connection String to the Application Configuration File

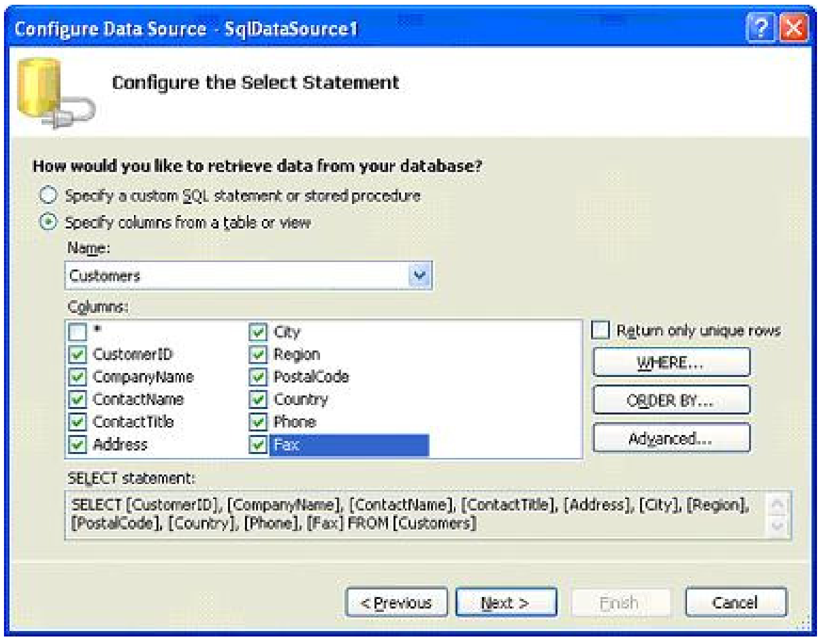
Storing connection strings in the application configuration file simplifies maintenance and deployment. To save the connection string in application configuration file, enter a name in the text box and then click Next. If you choose not to do this, the connection string is saved in the page as a property of the data source control.

Do you want to save the connection in the application configuration file?

☑ Yes, save this connection as:

NorthwindConnectionString

If you want to save new connection info in configuration file, check the Yes, save this connection checkbox and click next.



Configure the Select Statement

How would you like to retrieve data from your database?

○ Specify a custom SQL statement or stored procedure

● Specify columns from a table or view

Name: Customers

Columns:

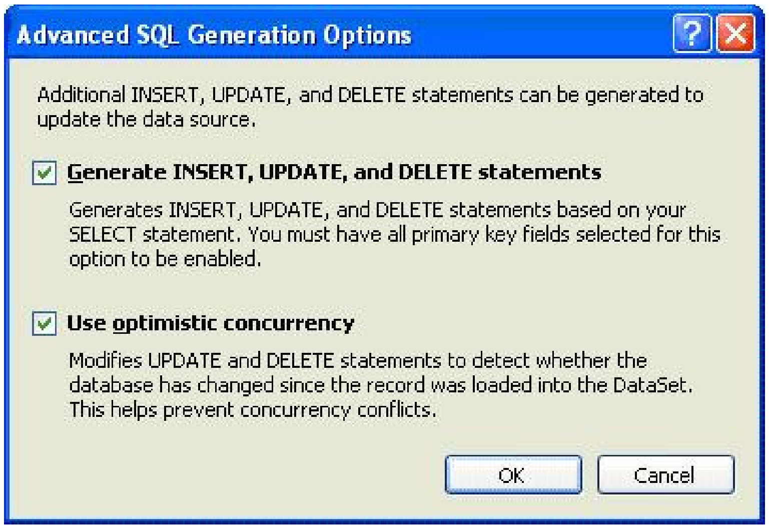
☐ *	☑ City
☑ CustomerID	☑ Region
☑ CompanyName	☑ PostalCode
☑ ContactName	☑ Country
☑ ContactTitle	☑ Phone
☑ Address	☑ Fax

☐ Return only unique rows

SELECT statement:

```
SELECT [CustomerID], [CompanyName], [ContactName], [ContactTitle], [Address], [City], [Region],
[PostalCode], [Country], [Phone], [Fax] FROM [Customers]
```

Specify the columns which would like to bind to GridView control and click next.


Advanced SQL Generation Options

Additional INSERT, UPDATE, and DELETE statements can be generated to update the data source.

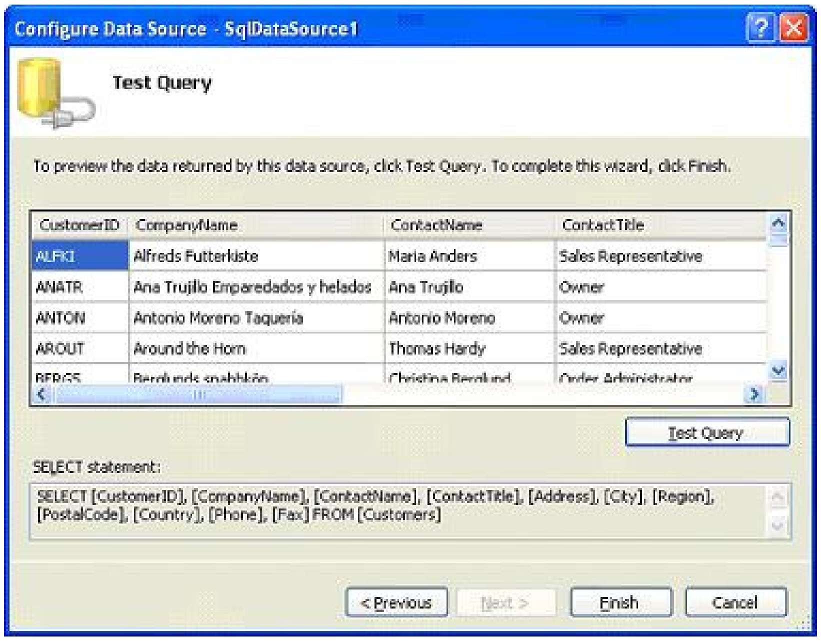
[✓] Generate INSERT, UPDATE, and DELETE statements

Generates INSERT, UPDATE, and DELETE statements based on your SELECT statement. You must have all primary key fields selected for this option to be enabled.

[✓] Use optimistic concurrency

Modifies UPDATE and DELETE statements to detect whether the database has changed since the record was loaded into the DataSet. This helps prevent concurrency conflicts.

If you are interested to add update/delete functionalities to GridView check the options and click ok.


Configure Data Source - SqlDataSource1

Test Query

To preview the data returned by this data source, click Test Query. To complete this wizard, click Finish.

CustomerID	CompanyName	ContactName	ContactTitle
ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative
ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner
ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner
AROUT	Around the Horn	Thomas Hardy	Sales Representative
BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator

SELECT statement:

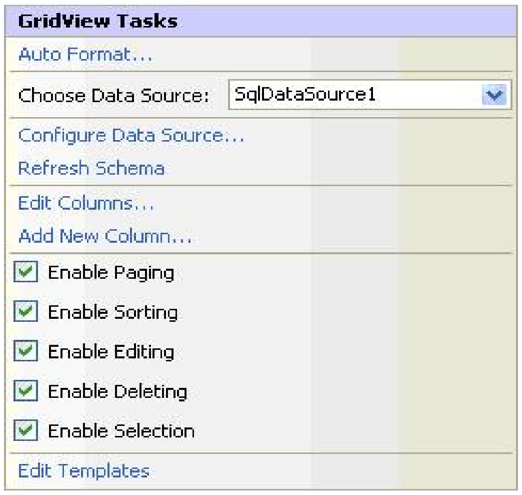
```
SELECT [CustomerID], [CompanyName], [ContactName], [ContactTitle], [Address], [City], [Region],
[PostalCode], [Country], [Phone], [Fax] FROM [Customers]
```

You can preview the data returned by this new data source by clicking Test Query button. Once you are done, click finish button.

After the configuring SqlDataSource using above mentioned steps, drag and drop the GridView control onto the page. Choose previously configured SqlDataSource from Smart tag as seen below. At the same time enable paging, sorting and selection. Also play around with Auto Format feature to apply various formatting features to GridView and Add Edit button and Update button by Clicking Add New Column.

You can enable the built-in edit or delete functionality of the GridView control in any of the following ways:

- Setting the AutoGenerateEditButton property to true to enable updating and the AutoGenerateDeleteButton property to true to enable deleting.
- Adding a CommandField with the ShowEditButton property set to true to enable updating and the ShowDeleteButton property set to true to enable deleting.
- Creating a TemplateField where the ItemTemplate contains command buttons with the CommandName set to "Edit" for updating and "Delete" for deleting.



GridView Tasks

Auto Format...

Choose Data Source: SqlDataSource1

Configure Data Source...

Refresh Schema

Edit Columns...

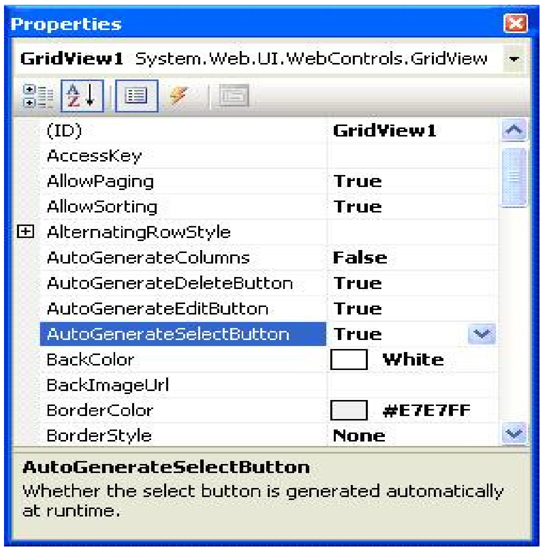
Add New Column...

- [✓] Enable Paging
- [✓] Enable Sorting
- [✓] Enable Editing
- [✓] Enable Deleting
- [✓] Enable Selection

Edit Templates

GridView Smart Tag

Set the AutoGenerateEditButton property to true to enable updating and the AutoGenerateDeleteButton property to true to enable deleting in the properties window Of GridView Control. Below listing shows the properties window.



Properties

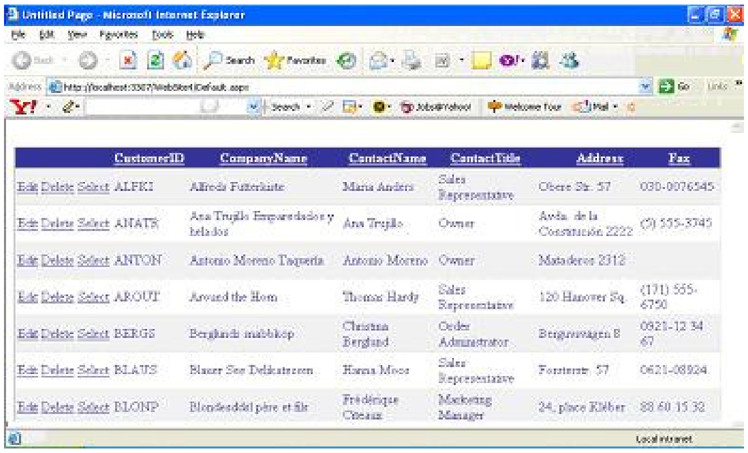
GridView1 System.Web.UI.WebControls.GridView

(ID)	GridView1
AccessKey	
AllowPaging	True
AllowSorting	True
⊞ AlternatingRowStyle	
AutoGenerateColumns	False
AutoGenerateDeleteButton	True
AutoGenerateEditButton	True
AutoGenerateSelectButton	True
BackColor	☐ White
BackImageUrl	
BorderColor	☐ #E7E7FF
BorderStyle	None

AutoGenerateSelectButton
Whether the select button is generated automatically at runtime.

GridView Properties

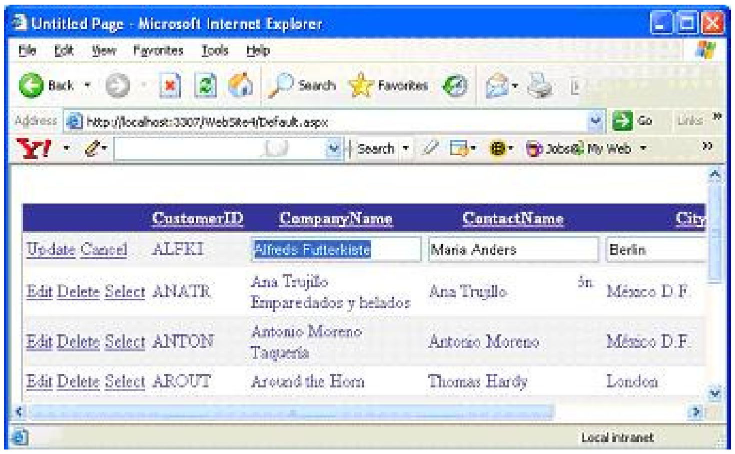
Press CTRL+F5 to load the page into browser and you will see results as shown below.



	CustomerID	CompanyName	ContactName	ContactTitle	Address	Fax
Edit Delete Select	ALFKI	Alfreds Futterkiste	Maria Anders	Sales Representative	Obere Str. 57	030-0076545
Edit Delete Select	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	Owner	Avda. de la Constitución 2222	(5) 555-3745
Edit Delete Select	ANTON	Antonio Moreno Taquería	Antonio Moreno	Owner	Mataderos 2312	
Edit Delete Select	AROUT	Around the Horn	Thomas Hardy	Sales Representative	120 Hanover Sq.	(171) 555-6750
Edit Delete Select	BERGS	Berglunds snabbköp	Christina Berglund	Order Administrator	Berguvsvägen 8	0921-12 34 67
Edit Delete Select	BLAUS	Blauer See Delikatessen	Hanna Moos	Sales Representative	Forsterstr. 57	0621-08924
Edit Delete Select	BLONP	Blondesddsl père et fils	Frédérique Citeaux	Marketing Manager	24, place Kléber	88.60.15.32

GridView output listing

When user clicked Edit link for first row you can observe "select" link disappears and text boxes appear to make changes.



	CustomerID	CompanyName	ContactName		City
Update Cancel	ALFKI	Alfreds Futterkiste	Maria Anders		Berlin
Edit Delete Select	ANATR	Ana Trujillo Emparedados y helados	Ana Trujillo	ón	México D.F.
Edit Delete Select	ANTON	Antonio Moreno Taquería	Antonio Moreno		México D.F.
Edit Delete Select	AROUT	Around the Horn	Thomas Hardy		London

GridView editing

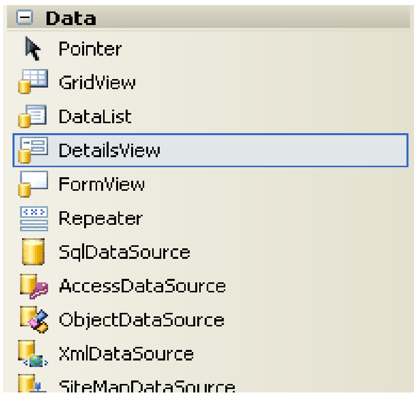
Hence ASP.NET 2.0 GridView control is powerful feature which eliminates the need to write custom code to perform database operation like updating and deleting. It is closely bound with new datasource controls to make this happen. But still, DataGrid is still available in ASP.NET 2.0 with it's own advantages.

GridView is very powerful control. However, in real developer life it is often better to avoid to reinvent the wheel and choose some more professional solution, like APNSoft Datagrid, to reduce your development time and get more stable code.

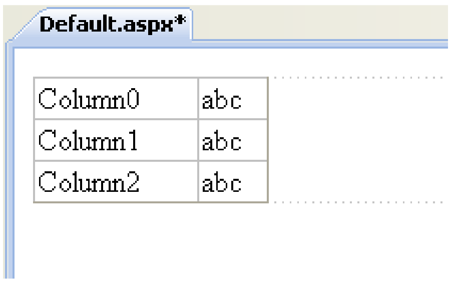
DETAILSVIEW CONTROL

DetailsView server control is used to display, delete, insert or edit a single record from data source. DetailsView control is often used in master/detail scenario with GridView control. To work with individual records in ASP.NET 1.1 you needed additional programming. But, in ASP.NET 2.0 or later we can use specialized DetailsView and FormView controls.

DetailsView control is located in Visual Studio toolbox, under Data tab.



To start work with DetailsView control, grab control from the toolbox and place it on web form. By default, DetailsView control looks like in an image bellow:



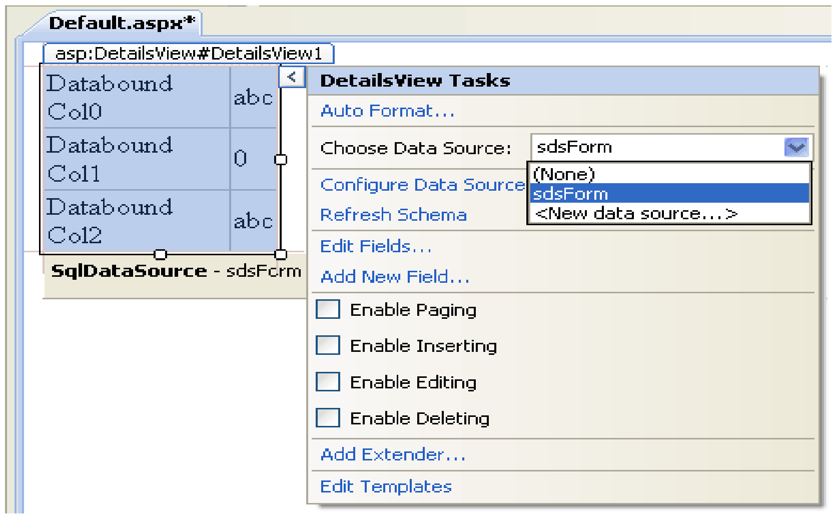
Column0	abc
Column1	abc
Column2	abc

To work with records, first we need to connect DetailsView control with some data source. In this example, I will use SqlDataSource control to bind DetailsView to SQl Server database. Here is a markup code for this SqlDataSource control:

```
<asp:SqlDataSource ID="sdsForm" runat="server"
ConnectionString="<%$ ConnectionStrings:ConnStr %>"
InsertCommand="INSERT INTO Products(ProductName, Description, Image, Category, ProductURL) VALUES (@ProductName, @Description, @Image, @Category, @ProductURL)"
SelectCommand="SELECT Products.ProductID, Products.ProductName, Products.Description, Products.Image, Products.ProductURL, Products.Category, Categories.CategoryName FROM Products INNER JOIN Categories ON Products.Category = Categories.ID ORDER BY ProductID"
UpdateCommand="UPDATE Products SET ProductName = @ProductName, Description = @Description, Image = @Image, Category = @Category, ProductURL = @ProductURL WHERE ProductID = @ProductID"
DeleteCommand="DELETE FROM Products WHERE ProductID = @ProductID">
<UpdateParameters>
<asp:Parameter Name="@ProductName" Type="String" />
<asp:Parameter Name="@Description" Type="String" />
<asp:Parameter Name="@Image" Type="String" />
<asp:Parameter Name="@Category" Type="Int32" />
<asp:Parameter Name="@ProductURL" Type="String" />
</UpdateParameters>
```

```
<InsertParameters>
<asp:Parameter Name="@ProductName" Type="String" />
<asp:Parameter Name="@Description" Type="String" />
<asp:Parameter Name="@Image" Type="String" />
<asp:Parameter Name="@Category" Type="Int32" />
<asp:Parameter Name="@ProductURL" Type="String" />
</InsertParameters>
</asp:SqlDataSource>
```

This data source contains commands for all four basic operations: Select, Update, Insert and Delete. If you only need to show record in read-only mode, your data source control needs only Select command. After we created SqlDataSource control, we need to associate it with DetailsView control. You can do that in design view, like on next image:



Also, you can do this if you set DataSourceID property of DetailsView control to the name of your data source control.

To start an example, you need a database. SqlDataSource control above uses a database with Products and Categories tables. You can create these tables in your database with SQL code like this:

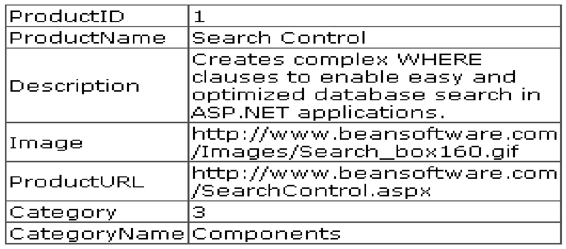
For table Products

```
CREATE TABLE Products(
[ID] [int]  IDENTITY(1,1) NOT NULL,
[ProductName] [nvarchar](50) NOT NULL,
[Description] [nvarchar](250) NOT NULL,
[Image] [nvarchar](1000) NOT NULL,
[Category] [int] NOT NULL,
[ProductURL] [nvarchar](1000) NOT NULL
)
```

And, very simple table Categories

```
CREATE TABLE Categories([ID] [int] IDENTITY(1,1) NOT NULL,
[CategoryName] [nvarchar](50)
)
```

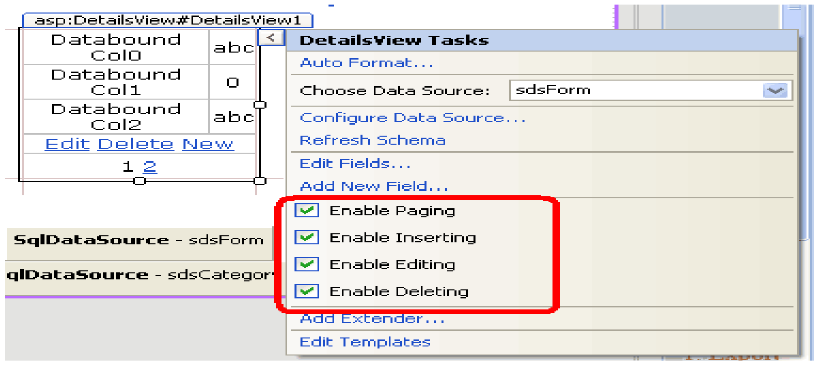
Fill tables with some data and run the example to see how it works. Of course, you can use your own existing table, just adopt SqlDataSource commands to suit your table structure. Run the example and DetailsView control will show first record, similar to the next image:



ProductID	1
ProductName	Search Control
Description	Creates complex WHERE clauses to enable easy and optimized database search in ASP.NET applications.
Image	http://www.beansoftware.com/Images/Search_box160.gif
ProductURL	http://www.beansoftware.com/SearchControl.aspx
Category	3
CategoryName	Components

How to update, insert and delete data and enable paging with DetailsView control

Until now, our example will only show one record in read-only mode. To enable paging, updating, deleting or adding new records, check proper checkboxes like in an image bellow:



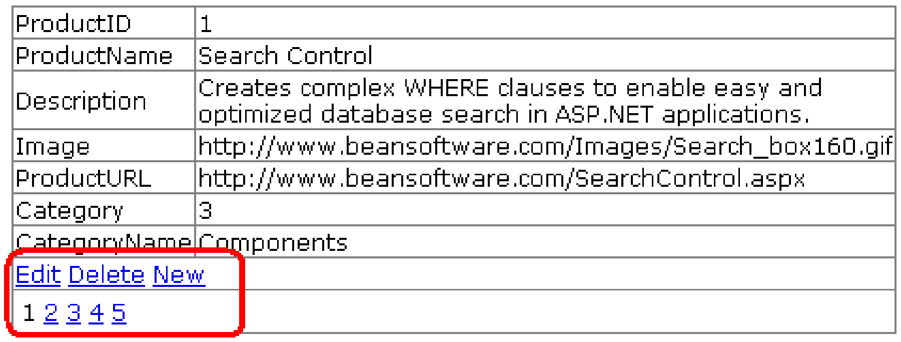
Choose Data Source: sdsForm

✓ Enable Paging
✓ Enable Inserting
✓ Enable Editing
✓ Enable Deleting

This action will produce markup code like this:

```
<asp:DetailsView ID="DetailsView1" runat="server" DataSourceID="sdsForm"AllowPagin
g="True">
<Fields>
 <asp:CommandField ShowDeleteButton="True" ShowEditButton="True"ShowInsertB
utton="True" />
</Fields>
</asp:DetailsView>
```

Before using an example you need to set DataKeyNames property of DetailsView control to the name of table primary key column. In our case table primary key column is named "ProductID". Now, you can start an example again and joy in new features. Test paging, updating, deleting and inserting a new record to see how it works. You'll see new buttons Edit, Delete and New link buttons. Note that we did all this without writing single line of C# or VB.NET code! Output should look similar to image bellow:



ProductID	1
ProductName	Search Control
Description	Creates complex WHERE clauses to enable easy and optimized database search in ASP.NET applications.
Image	http://www.beansoftware.com/Images/Search_box160.gif
ProductURL	http://www.beansoftware.com/SearchControl.aspx
Category	3
CategoryName	Components
Edit Delete New	
1 2 3 4 5	

How to provide better layout with DetailsView control

Now, the example is completely functional, although it looks ugly. For example, instead of showing an URL of image, it would be better to show image itself, instead of product url it will be better to show clickable link etc.By default, DetailsView control has AutoGenerateRows property set to true. That means that all fields will be generated at run time, depending of what fields your data source contains. This is fast solution, but if you want more control over the way how fields will be presented, you need to set AutoGenerateRows property to False. Now you must specify needed fields manually. DetailsView control has seven types of fields, every with different purpose:

BoundField - This is default field, used when AutoGenerateRows is true. You can use it for simple text data type
ButtonField - Showing a Button control
CheckBoxField - Showing a CheckBox control, used for boolean yes/no fields
CommandField - Command buttons, like Edit, Delete or New, used for manipulating data
HyperLinkField - Used to show a hyperlink
ImageField - Used to show an image
TemplateField - Used to show data in custom layout, when other field types are inadequate

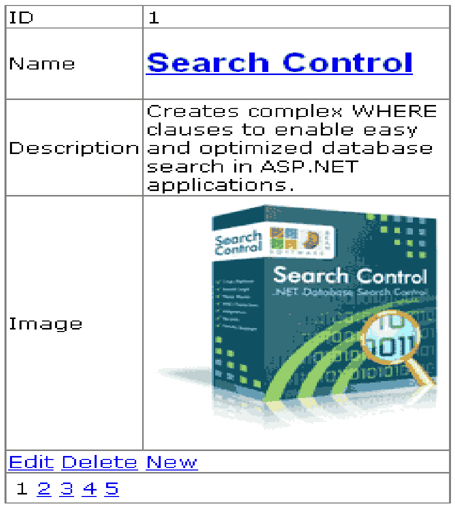
Implemented in our example, markup code with custom fields used could look like this:

```
<asp:DetailsView ID="DetailsView1" runat="server" DataSourceID="sdsForm"Height="50p
x" Width="225px" AllowPaging="True" AutoGenerateRows="false">
<Fields>
 <%--Simple BoundField, we'll make it read-only --%>
 <asp:BoundField DataField="ProductID" ReadOnly="true" HeaderText="ID" />
 <%--TemplateField is used for ProductURL and ProductName fields--%>
 <asp:TemplateField HeaderText="Name">
  <ItemTemplate>
   <h1><a href='<%# Eval("ProductURL") %>'><%# Eval("ProductName") %></a></h1>
  </ItemTemplate>
  <EditItemTemplate>
   <table width="100%"><tr><td>
    Product name: </td><td><asp:TextBox ID="txtProductName" Text='<%#
Bind("ProductName") %>' runat="server"></asp:TextBox></td></tr>
    <tr><td>Image URL: </td><td><asp:TextBox ID="txtImageURL" Text='<%#
 Bind("ProductURL") %>' runat="server"></asp:TextBox></td></tr></table>
  </EditItemTemplate>
  <InsertItemTemplate>
   <table width="100%"><tr><td>
    Product name: </td><td><asp:TextBox ID="txtProductName" Text='<%#
Bind("ProductName") %>' runat="server"></asp:TextBox></td></tr>
    <tr><td>Image URL: </td><td><asp:TextBox ID="txtImageURL" Text='<%#
 Bind("ProductURL") %>' runat="server"></asp:TextBox></td></tr></table>
  </InsertItemTemplate>
 </asp:TemplateField>
 <asp:BoundField DataField="Description" HeaderText="Description" />
<%--This TemplateField show friendly category name from other table--%>
<asp:TemplateField HeaderText="Category">
<ItemTemplate>
 <%# Eval("CategoryName") %>
</ItemTemplate>
<EditItemTemplate>
 <asp:DropDownList ID="ddlCategory"
  DataSourceID="sdsCategory"
  DataValueField="CategoryID"
  DataTextField="CategoryName"
  SelectedValue='<%#  Bind("Category") %>' runat="server" >
 </asp:DropDownList>
</EditItemTemplate>
<InsertItemTemplate>
 <asp:DropDownList ID="ddlCategory"
  DataSourceID="sdsCategory"
  DataValueField="CategoryID"
  DataTextField="CategoryName"
  SelectedValue='<%#  Bind("Category") %>' runat="server" >
 </asp:DropDownList>
</InsertItemTemplate>
</asp:TemplateField>
 <%--Image field is used to show product image--%>
 <asp:ImageField DataImageUrlField="Image" HeaderText="Image"></asp:ImageField>
 <asp:CommandField ShowDeleteButton="True" ShowEditButton="True"ShowInsertButton
="True" />
</Fields>
</asp:DetailsView>
```

Category selection uses data from second table. To bind categories to DropDownList control we need to add another data source control. Markup for second SqlDataSource control looks like this (note that we need just SelectCommand here):

```
<asp:SqlDataSource ID="sdsCategory" runat="server" ConnectionString="<%$
ConnectionStrings:beanConnStr %>" SelectCommand="SELECT ID AS CategoryID,
CategoryName FROM Categories"></asp:SqlDataSource>
```

Output will now look like this:

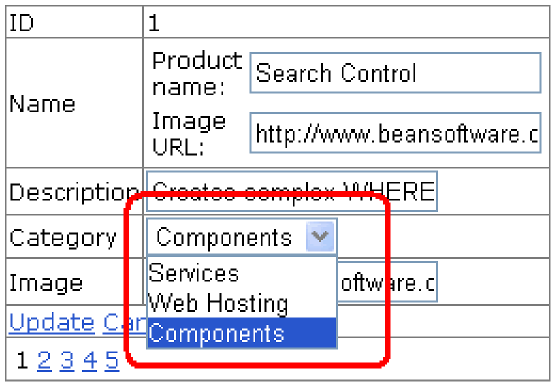


ID	1
Name	Search Control
Description	Creates complex WHERE clauses to enable easy and optimized database search in ASP.NET applications.
Image	
Edit Delete New	
1 2 3 4 5	

This layout looks definitely more user friendly. ImageField is used to show an image, primary key is presented as a read-only field and used TemplateField to enable advanced layout in product name and category selection. TemplateField is a little harder to use compared to other types of fields, but it allows almost unlimited formatting options. Practically, you can place complete all table columns in one TemplateField.

TemplateField contains ItemTemplate, EditItemTemplate and InsertItemTemplate sub tags used to specify different layouts for showing and editing of existing records or adding a new record. In ItemTemplate Eval() method is used since Bind() provides one direction read-only binding. For EditItemTemplate and InsertItemTemplate Bind() method is used. Bind() method enables binding in two directions.

Selection of category is presented in different layouts: as simple text in read-only mode and with DropDownList control in EditItemTemplate and InsertItemTemplate, so users can easily select category from offered category names in DropDownList, instead of writing of category ID to TextBox control.



ID	1
Name	Product name: Search Control Image URL: http://www.beansoftware.c
Description	Creates complex WHERE
Category	Components
Image	oftware.c
Update Car	
1 2 3 4 5	

DetailsView control layout when updating an existing record

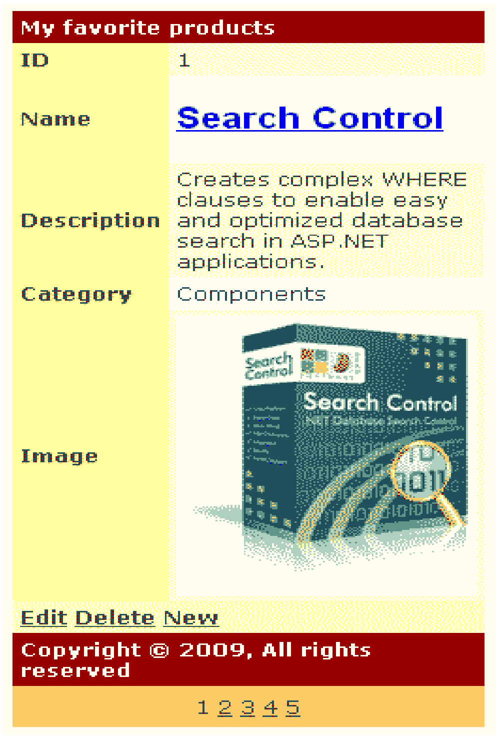
Adding Header and Footer to get more professional look

To add Header and Footer elements to DetailsView control, use HeaderTemplate and FooterTemplate sub tags. Very simple imlementation of header and footer could look like this:

```
<HeaderTemplate>
My favorite products
</HeaderTemplate>
<FooterTemplate>
Copyright &copy; 2009, All rights reserved
</FooterTemplate>
```

Adding styles in DetailsView control

The easiest way to add professional formatting to DetailsView control is to use AutoFormat option. Click on AutoFormat... link button on DetailsView tasks or in Property window and choose one of few predefined styles. I'll choose Colorful auto format that will produce output like in next image:



My favorite products	
ID	1
Name	Search Control
Description	Creates complex WHERE clauses to enable easy and optimized database search in ASP.NET applications.
Category	Components
Image	
Edit Delete New	
Copyright © 2009, All rights reserved	
1 2 3 4 5	

If AutoFormat option can't satisfy your requirements, you can set styles of each element separately. DetailsView control has a number of style properties:

AlternatingRowStyle - Style of every alternate row
EditRowStyle - Style for the row in EditView, when you click to Edit command button control will use this style
RowStyle - Style of rows in display mode
PagerStyle - Defines style of pager (if pager is used, you can remove pager if you set

AllowPaging property to false)
EmptyDataRowStyle - Style used if data source contains zero rows
HeaderStyle - Style used for header element
FooterStyle - Style used for footer element

Also, you can set things like background control image or table cell spacing by using properties from Appearance section.

Custom paging with DetailsView control

Default pager of DetailsView control contains numbers only. You can customize pager more by using PagerSettings property. You can try this line that shows only First, Previous, Next and Last buttons without numbers:

```
<PagerSettings Mode="NextPreviousFirstLast" NextPageText="&gt;"PreviousPageText="&l
t;" LastPageText="&gt;&gt;" FirstPageText="&lt;&lt;" />
```

If existing pager modes are not enough for you, you can try with PagerTemplate to build your own pager layout, like this:

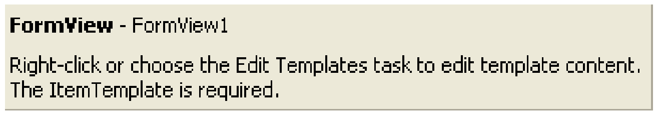
```
<PagerTemplate>
<asp:LinkButton ID="btnFirst" CommandName="Page" CommandArgument="First"Text="
First" RunAt="server"/>  <asp:LinkButton ID="btnPrevious" CommandName="Page" Com
mandArgument="Prev"  Text="Previous" RunAt="server"/> <asp:LinkButton ID="btnNext"
  CommandName="Page" CommandArgument="Next" Text="Next"  RunAt="server"/>  <as
p:LinkButton ID="btnLast"  CommandName="Page"CommandArgument="Last"  Text="Las
t" RunAt="server"/>
</PagerTemplate>
```

FORMVIEW CONTROL

FormView Control is introduced with ASP.NET 2.0. It is often used in master/details scenario with GridView, DataList or Repeater control. FormView Control is similar to DeatilsView control. Both controls enable showing of single record from data source. The main difference between FormView and DeatilsView controls is that FormView control allows using of templates for displaying a record instead of row fields used in DetailsView. Unlike DetailsView, FormView control doesn't have predefined data layout. Developer needs to create a template first and specify controls and formatting to display single record from data source. Because of that developer have full control over layout and it is pretty easy to add validation capabilities.

How to use FormView control

FormView control is by default located in Visual Studio toolbox, inside Data tab. If you place FormView control to web form you'll see that it not appears very user friendly at first look. There is only gray rectangle, like in image bellow:



FormView - FormView1

Right-click or choose the Edit Templates task to edit template content.
The ItemTemplate is required.

Unlike DetailsView control, there is no predefined template in FormView control. To show data, you need to select data source and create your own template to define control's appearance.

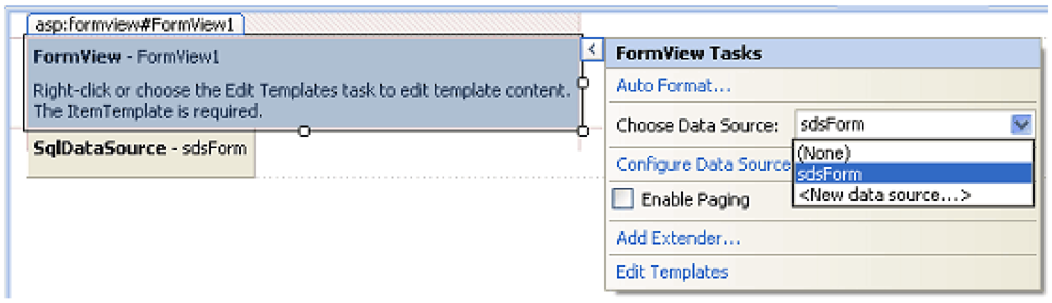
Binding FormView control to data source

There are two ways of data binding of FormView control: using of **DataSourceID property** or by using of **DataSource property**. Using fo DataSourceID property is recommended way because it enables easier paging and updating. When DataSourceID property is used, FormView control supports two way data binding, so in addition to showing the data, control will automatically support edit, insert and delete operations. Set FormView's DataSourceID property to the name of your data source control.

If your FormView control needs to show data in read-only mode only, set just SelectCommand property of data source control. If you also want to modify records, delete or add new items you need UpdateCommand, DeleteCommand and InsertCommand as well. Example of SqlDataSource control that will be used to select, insert, edit and delete data in FormView control could look like code bellow. This example shows typical situation where form includes Category field related to other table:

```
<asp:SqlDataSource ID="sdsForm" runat="server"
 ConnectionString="<%$ ConnectionStrings:ConnStr %>"
 InsertCommand="INSERT INTO Products(ProductName, Description, Image, Category,
ProductURL) VALUES (@ProductName, @Description, @Image, @Category,
@ProductURL)"
 SelectCommand="SELECT Products.ProductID, Products.ProductName,
Products.Description, Products.Image, Products.ProductURL, Products.Category,
Categories.CategoryName FROM Products INNER JOIN Categories ON Products.Category
= Categories.ID ORDER BY ProductID"
 UpdateCommand="UPDATE Products SET ProductName = @ProductName, Description =
@Description, Image = @Image, Category = @Category, ProductURL = @ProductURL
WHERE ProductID = @ProductID"
 DeleteCommand="DELETE FROM Products WHERE ProductID = @ProductID">
  <UpdateParameters>
    <asp:Parameter Name="@ProductName" Type="String" />
    <asp:Parameter Name="@Description" Type="String" />
    <asp:Parameter Name="@Image" Type="String" />
    <asp:Parameter Name="@Category" Type="Int32" />
    <asp:Parameter Name="@ProductURL" Type="String" />
  </UpdateParameters>
  <InsertParameters>
    <asp:Parameter Name="@ProductName" Type="String" />
    <asp:Parameter Name="@Description" Type="String" />
    <asp:Parameter Name="@Image" Type="String" />
    <asp:Parameter Name="@Category" Type="Int32" />
    <asp:Parameter Name="@ProductURL" Type="String" />
  </InsertParameters>
</asp:SqlDataSource>
```

After you create your data source control, you need to associate data source name with DataSourceID property of FormView control. You can do it also in design view, like in image bellow.



Creating FormView control templates

FormView template consists from three parts: for viewing a record, edit a record, and add new record. Also, you can use it for browsing and deleting of records and include formatting for header and footer elements. Templates available in FormView controls are:

HeaderTemplate
ItemTemplate
InsertItemTemplate
EditItemTemplate
EmptyDataTemplate
PagerTemplate
FooterTemplate

How to show data in FormView control

To show data in FormView control in read-only mode, we use ItemTemplate sub tag inside FormView tag, and Eval() method. If we set DataSourceID property to valid data source control, example code bellow will display ProductName and Description:

```
<ItemTemplate>
<h2><%# Eval("ProductName") %></h2>
<p><%# Eval("Description") %></p>
</ItemTemplate>
```

Or, more common example with image, category and Edit/Insert/Delete link buttons, with confirmation message box for deleting a record:

```
<ItemTemplate>
 <h2><a href='<%# Eval("ProductURL") %>'><%# Eval("ProductName") %></a></h2>
 <p><%# Eval("Description") %></p>
 <a href='<%# Eval("ProductURL") %>'><img border="0" src='<%#
Eval("Image") %>'alt='<%# Eval("ProductName") %>' /></a><br />
```

```
 <b>Category:</b> <%# Eval("CategoryName") %><br /><br />
 <asp:LinkButton ID="lbEdit" CausesValidation="false" CommandName="Edit"runat="serv
er">Edit</asp:LinkButton>
 <asp:LinkButton ID="lbAddNew" CausesValidation="false" CommandName="New"runat=
"server">Add New Item</asp:LinkButton>
 <asp:LinkButton ID="lbDelete" CausesValidation="false" CommandName="Delete"runat="
server" OnClientClick="return confirm('Are you sure you want to delete this
item?');">Delete</asp:LinkButton>
</ItemTemplate>
```

Also, with Eval method you can show formatted data. This syntax will format data as currency:

```
<%#Eval("Price","{0:c}")%>
```

How to edit, insert and delete data in FormView

To add new row to data source, use InsertItemTemplate. To edit existing row use EditItemTemplate. Unlike **Eval() method** in ItemTemplate example above, when insert or edit data you need **Bind() method**. Use Eval() method when you need read only data (binding in one direction, most used in ItemTemplate), and use Bind() method when you need binding in two directions, like in EditItemTemplate or InsertItemTemplate. Of course, you can disallow modification of some fields, and for read only fields use Label control and Eval() method.

Bellow is example of EditItemTemplate and InsertItemTemplate:

```
<EditItemTemplate>
 <table><tr><td>
 Product name: </td><td><asp:TextBox ID="txtProductName" Text='<%#
Bind("ProductName") %>' runat="server"></asp:TextBox></td></tr>
  <tr><td>Description: </td><td><asp:TextBox ID="txtDescription" Text='<%#
Bind("Description") %>' runat="server"></asp:TextBox></td></tr>
  <tr><td>Image URL: </td><td><asp:TextBox ID="txtImageURL" Text='<%#
Bind("Image")%>' runat="server"></asp:TextBox></td></tr>
  <tr><td>Product URL: </td><td><asp:TextBox ID="txtProductURL" Text='<%#
Bind("ProductURL") %>' runat="server"></asp:TextBox></td></tr>
  <tr><td>Category: </td><td><asp:DropDownList ID="ddlCategory"
   DataSourceID="sdsCategory"
   DataValueField="CategoryID"
   DataTextField="CategoryName"
   SelectedValue='<%# Bind("Category") %>' runat="server" >
   </asp:DropDownList></td></tr></table>
   <asp:LinkButton ID="lbSave" CommandName="Update"runat="server">Save</asp:LinkB
utton>
   <asp:LinkButton ID="lbCancel" CommandName="Cancel"runat="server">Cancel</asp:Li
nkButton>
</EditItemTemplate>
<InsertItemTemplate>
 <table border="0" width="100%">
  <tr><td>Product            name: </td><td><asp:TextBox ID="txtProductName" Text='<%#
Bind("ProductName") %>' runat="server"></asp:TextBox></td></tr>
  <tr><td>Description: </td><td><asp:TextBox ID="txtDescription" Text='<%#
Bind("Description") %>' runat="server"></asp:TextBox></td></tr>
  <tr><td>Image                 URL: </td><td><asp:TextBox ID="txtImageURL" Text='<%#
Bind("Image")%>' runat="server"></asp:TextBox></td></tr>
  <tr><td>Product              URL: </td><td><asp:TextBox ID="txtProductURL" Text='<%#
Bind("ProductURL") %>' runat="server"></asp:TextBox></td></tr>
  <tr><td>Category: </td><td><asp:DropDownList ID="ddlCategory"
   DataSourceID="sdsCategory"
   DataValueField="CategoryID"
   DataTextField="CategoryName"
   SelectedValue='<%#                              Bind("Category") %>' runat="server">
  </asp:DropDownList></td></tr></table>Â Â
 <asp:LinkButton ID="lbSave" CommandName="Insert"runat="server">Save</asp:LinkButt
on>
 <asp:LinkButton ID="lbCancel" CommandName="Cancel"runat="server">Cancel</asp:Lin
kButton>
</InsertItemTemplate>
```

If Insert, Edit and Delete command buttons will be used, we need to set **DataKeyNames property** to the name of your table primary key (in our example SqlDataSource control, this field is ProductID).

When FormView control is initially displayed on web form, it will show ItemTemplate by default. You can change this behavior by using **DefaultMode property**, and show InsertItemTemplate or EditItemTemplate.

In case that your data source returned zero rows because you deleted all records or simply you didn't insert any record in table yet, FormView have nothing to show. In this case EmptyDataTemplate is shown. You can place any markup and style in this template, or just inform your user that there is no records in data source, like in example code bellow:

```
<EmptyDataTemplate>
There is no records in data source.
</EmptyDataTemplate>
```

Paging in FormView Control

To enable paging in FormView control, set **AllowPaging property** to true and default pager with numbers only will be displayed (if selected data source supports paging). If you are satisfied with good old paging with numbers only, you don't need to do anything else, FormView pager will work well.

If you want more control of layout, you can use PagerSettings property. By changing sub properties of PaggerSettings property, you can change pager look and feel according to your needs. This example will add First and Last buttons to default pager and limit number of visible pages to 5:

```
<PagerSettings Mode="NumericFirstLast" PageButtonCount="5" />
```

If you are still not satisfied with results, you can try PagerTemplate. PagerTemplate is used to create your own custom FormView pager.If PagerTemplate is used, default pager will not be shown. In template you can use any HTML tags, CSS or ASP.NET controls. Very simple example of PagerTemplate with four buttons is shown bellow:

```
<PagerTemplate>
<asp:LinkButton ID="btnFirst" CommandName="Page" CommandArgument="First"Text="
First" RunAt="server"/>
 <asp:LinkButton ID="btnPrevious" CommandName="Page" CommandArgument="Prev" T
ext="Previous" RunAt="server"/>
 <asp:LinkButton ID="btnNext" CommandName="Page" CommandArgument="Next" Text=
"Next" nbsp;RunAt="server"/>
 <asp:LinkButton ID="btnLast" CommandName="Page" CommandArgument="Last" Text="
Last" RunAt="server"/>
</PagerTemplate>
```

How to define FormView control style

FormView control allows using of several self explaining sub tags, used to define style of each element. There are:

- <HeaderStyle />
- <RowStyle />
- <EditRowStyle />
- <InsertRowStyle />
- <EmptyDataRowStyle />
- <PagerStyle />
- <FooterStyle />

For example, to specify style of header and footer elements, we can use FormView sub tags like this:

```
<HeaderStyle BackColor="Blue" ForeColor="White" Font-Names="Arial" />
<FooterStyle BackColor="Khaki" ForeColor="DarkRed" Font-Names="Arial" />
```

If you use HeaderStyle and FooterStyle, then you need to show something in HeaderTemplate and FooterTemplate. This can be any markup code, including ASP.NET server controls. Here is example of pretty simple FormView header and footer that include plain text only:

```
<HeaderTemplate>
My favorite products
</HeaderTemplate>
<FooterTemplate>FormView footer example
</FooterTemplate>
```

CRYSTAL REPORTS IN ASP.NET

Crystal Reports is the standard reporting tool for Visual Studio .NET used to display data of presentation quality. You can display multiple-level totals, charts to analyze data, and much more in Crystal Reports. Creating a Crystal Report requires minimal coding since it is created in Designer interface. It is available as an integrated feature of Microsoft Visual Studio .NET, Borland Delphi, and C#Builder.

Advantages of Crystal Reports

Some of the major advantages of using Crystal Reports are:

1. Rapid report development since the designer interface would ease the coding work for the programmer.

2. Can extend it to complicated reports with interactive charts and enhance the understanding of the business model

3. Exposes a report object model, can interact with other controls on the ASP.NET Web form

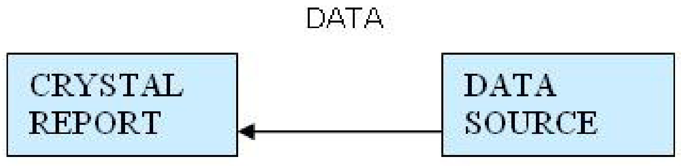
4. Can programmatically export the reports into widely used formats like .pdf, .doc, .xls, .html and .rtf

Implementation Models

Crystal Reports need database drivers to connect to the data source for accessing data. Crystal Reports in .net support two methods to access data from a data source:

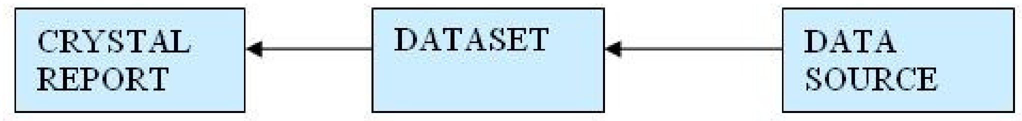
The Pull Method

When this model is used to access data from the data source, the database driver directly retrieves the data from the data source. This model does not require the developer to write code for creating a connection and retrieving data from the data source. It is the Crystal report that manages the SQL commands for connecting by using the specified driver.



The Push Method

When this model is used to access data from data source, the developer writes the code to connect to the data source and retrieve data. The data from the data source is cached in dataset and multiple crystal reports accesses data from the dataset. The performance can be optimized in this manner by using connection sharing and manually limiting the number of records that are passed on to the report.



Crystal Reports Types

Crystal Report Designer can load reports that are included into the project as well as those that are independent of the project.

Strongly-typed Report

When you add a report file into the project, it becomes a "strongly-typed" report. In this case, you will have the advantage of directly creating an instance of the report object, which could reduce a few lines of code, and cache it to improve performance. The related .vb file, which is hidden, can be viewed using the editor's "show all files" icon in the Solution Explorer.

Un-Typed Report

Those reports that are not included into the project are "un-typed" reports. In this case, you will have to create an instance of the Crystal Report Engine's "ReportDocument" object and manually load the report into it.

Creating Crystal Reports
You can create a Crystal Report by using three methods:

1.Manually i.e. from a blank document
2. Using Standard Report Expert
3. From an existing report

Using Pull Method

Creating Crystal Reports Manually.

We would use the following steps to implement Crystal Reports using the Pull Model:

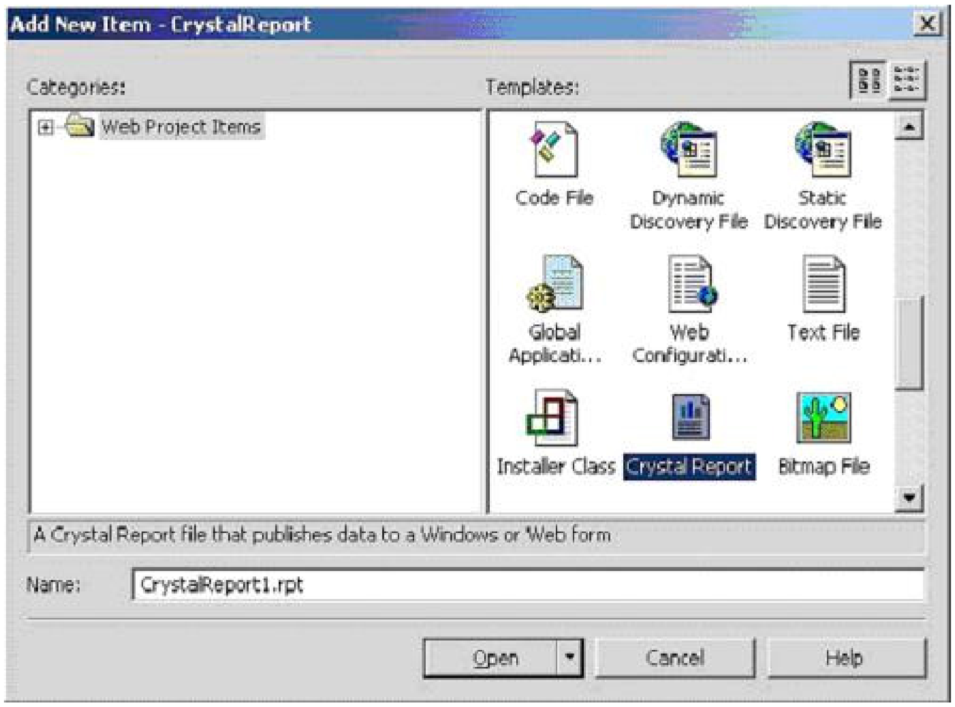
1. **Create the .rpt file** (from scratch) and set the necessary database connections using the Crystal Report Designer interface.

2. **Place a CrystalReportViewer control** from the toolbox on the .aspx page and set its properties to point to the .rpt file that we created in the previous step.

3. Call the databind method from your code behind page.

I. Steps to create the report i.e. the .rpt file

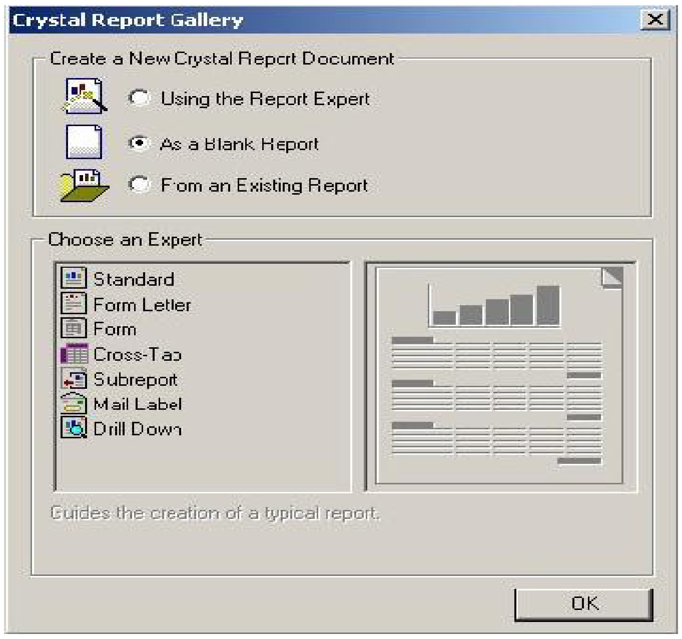
1) Add a new Crystal Report to the web form by right clicking on the "Solution Explorer", selecting "Add" --> "Add New Item" --> "Crystal Report".



A Crystal Report file that publishes data to a Windows or Web form

Name: CrystalReport1.rpt

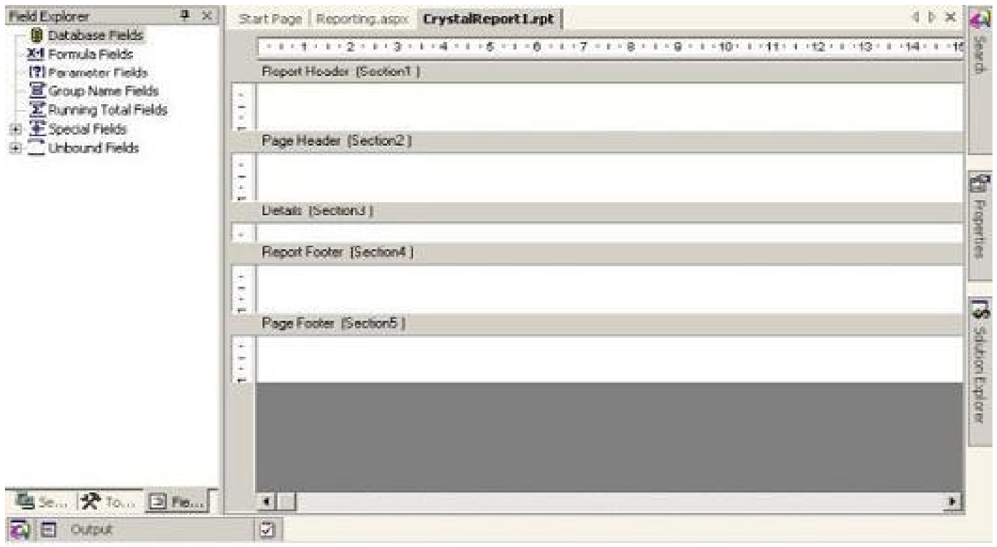
2) On the "Crystal Report Gallery" pop up, select the "As a Blank Report" radio button and click "ok".



( ) Using the Report Expert
(•) As a Blank Report
( ) From an Existing Report

Guides the creation of a typical report.

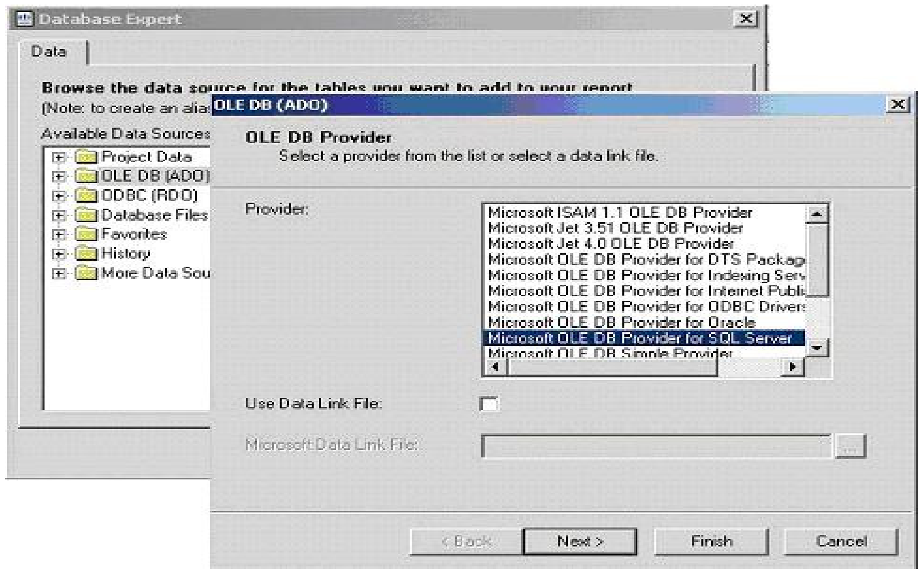
3) This should open up the Report File in the Crystal Report Designer.



4) Right click on the "Details Section" of the report, and select "Database" -> "Add/Remove Database".

5) In the "Database Expert" pop up window, expand the "OLE DB (ADO)" option by clicking the "+" sign, which should bring up another "OLE DB (ADO)" pop up.

6) In the "OLE DB (ADO)" pop up, Select "Microsoft OLE DB Provider for SQL Server" and click Next.


Database Expert

Data

Browse the data source for the tables you want to add to your report

(Note: to create an alias

Available Data Sources

- Project Data
- OLE DB (ADO)
- ODBC (RDO)
- Database Files
- Favorites
- History
- More Data Sou

OLE DB (ADO)

OLE DB Provider

Select a provider from the list or select a data link file.

Provider:

- Microsoft ISAM 1.1 OLE DB Provider
- Microsoft Jet 3.51 OLE DB Provider
- Microsoft Jet 4.0 OLE DB Provider
- Microsoft OLE DB Provider for DTS Packag
- Microsoft OLE DB Provider for Indexing Serv
- Microsoft OLE DB Provider for Internet Publi
- Microsoft OLE DB Provider for ODBC Drivers
- Microsoft OLE DB Provider for Oracle
- Microsoft OLE DB Provider for SQL Server
- Microsoft OLE DB Simple Provider

Use Data Link File: ☐

Microsoft Data Link File:

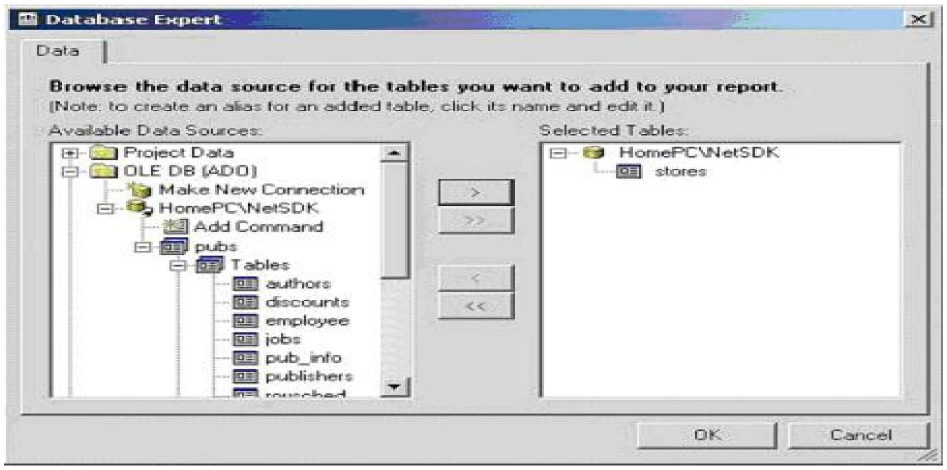
7) Specify the connection information.

8) Click "Next" and then click "Finish"

9) Now you should be able to see the Database Expert showing the table that have been selected

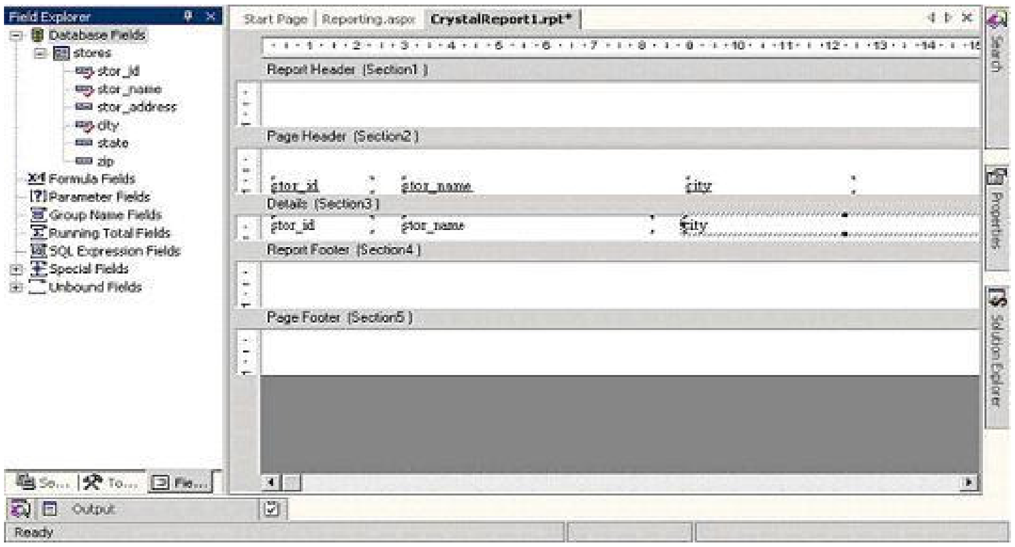
10) Expand the "Pubs" database, expand the "Tables", select the "Stores" table and click on ">" to include it into the "Selected Tables" section.

Note: If you add more than one table in the database Expert and the added tables have matching fields, when you click the OK button after adding the tables, the links between the added tables is displayed under the Links tab. You can remove the link by clicking the Clear Links button.



11) Now the Field Explorer should show you the selected table and its fields under the "Database Fields" section, in the left window.

12) Drag and drop the required fields into the "Details" section of the report. The field names would automatically appear in the "Page Header" section of the report. If you want to modify the header text then right click on the text of the "Page Header" section, select "Edit Text Object" option and edit it.



13) Save it and we are through.

II. Creating a Crystal Report Viewer Control

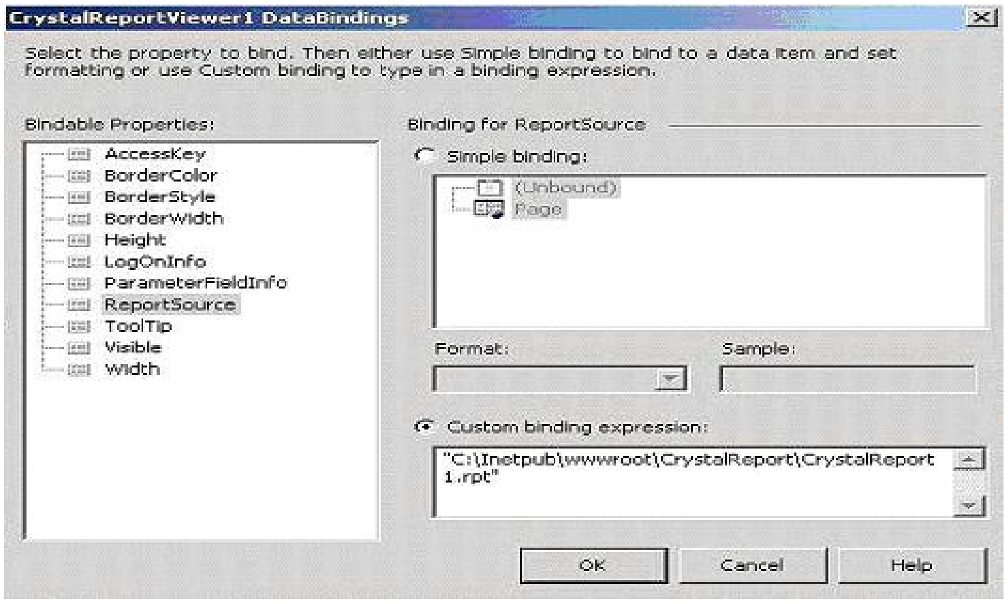
1) Drag and drop the "Crystal Report Viewer>" from the web forms tool box on to the .aspx page

2) Open the properties window for the Crystal Report Viewer control.

3) Click on the [...] next to the "Data Binding" Property and bring up the data binding pop-up window

4) Select "Report Source".

5) Select the "Custom Binding Expression" radio button, on the right side bottom of the window and specify the sample .rpt filename and path as shown in the fig.



CrystalReportViewer1 DataBindings

Select the property to bind. Then either use Simple binding to bind to a data item and set formatting or use Custom binding to type in a binding expression.

Bindable Properties:
- AccessKey
- BorderColor
- BorderStyle
- BorderWidth
- Height
- LogOnInfo
- ParameterFieldInfo
- ReportSource
- ToolTip
- Visible
- Width

Binding for ReportSource

Simple binding:
- (Unbound)
- Page

Format:

Sample:

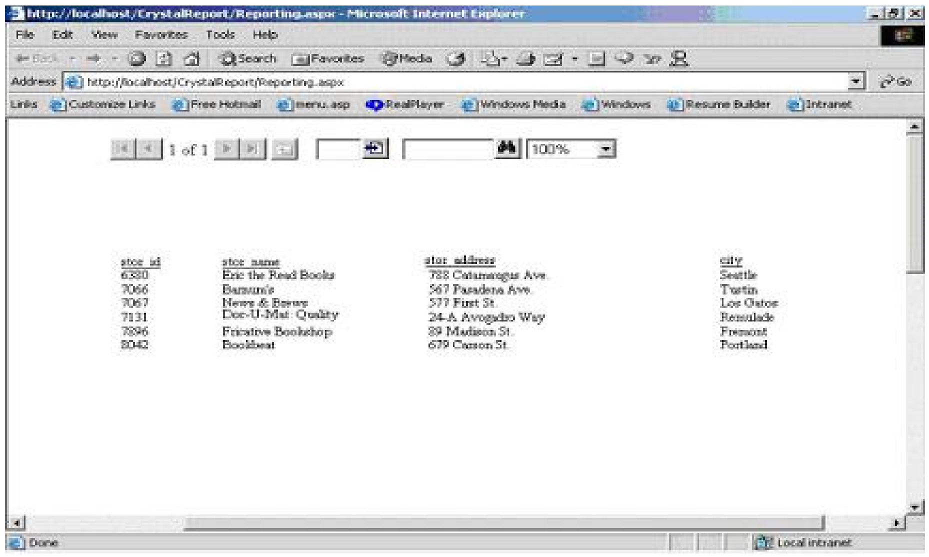
Custom binding expression:

"C:\Inetpub\wwwroot\CrystalReport\CrystalReport1.rpt"

6) You should be able to see the Crystal Report Viewer showing you a preview of actual report file using some dummy data and this completes the inserting of the Crystal Report Viewer controls and setting its properties.

Note: In the previous example, the CrystalReportViewer control was able to directly load the actual data during design time itself as the report was saved with the data. In this case, it will not display the data during design time as it not saved with the data - instead it will show up with dummy data during design time and will fetch the proper data only at run time.

7) Call the Databind method on the Page Load Event of the Code Behind file (.aspx.vb). Build and run your .aspx page. The output would look like this.


http://localhost/CrystalReport/Reporting.aspx - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Address http://localhost/CrystalReport/Reporting.aspx
1 of 1
100%
Done
Local intranet

stor_id	stor_name	stor_address	city
6380	Eric the Read Books	788 Catamaugus Ave.	Seattle
7066	Barnum's	567 Pasadena Ave.	Tustin
7067	News & Brews	577 First St.	Los Gatos
7131	Doc-U-Mat: Quality	24-A Avogadro Way	Remulade
7896	Fricative Bookshop	89 Madison St.	Fremont
8042	Bookbeat	679 Carson St.	Portland

Using a PUSH model

1. Create a Dataset during design time.

2. Create the .rpt file (from scratch) and make it point to the Dataset that we created in the previous step.

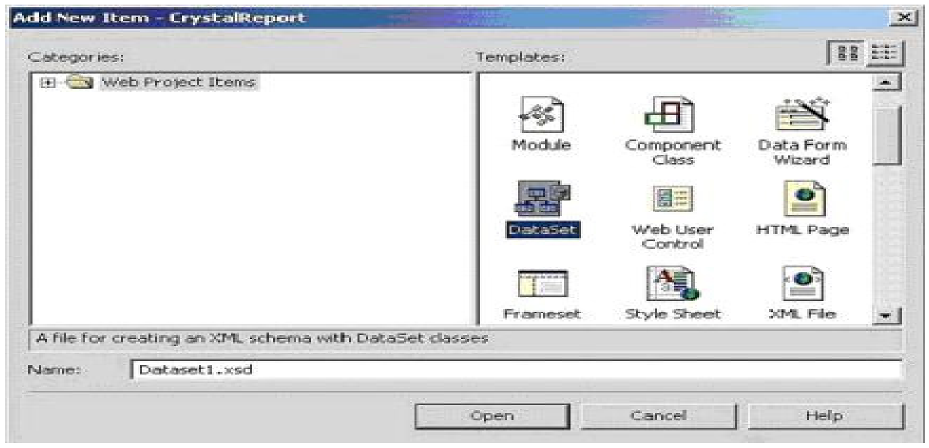
3. Place a CrystalReportViewer control on the .aspx page and set its properties to point to the .rpt file that we created in the previous step.

4. In your code behind page, write the subroutine to make the connections to the database and populate the dataset that we created previously in step one.

5. Call the Databind method from your code behind page.

I. Creating a Dataset during Design Time to Define the Fields of the Reports

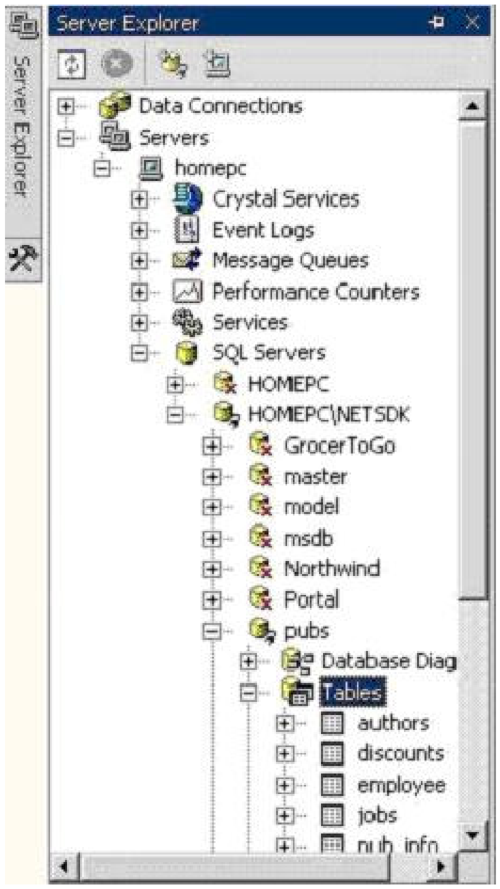
1) Right click on "Solution Explorer", select "Add" --> select "Add New Item" --> Select "DataSet"



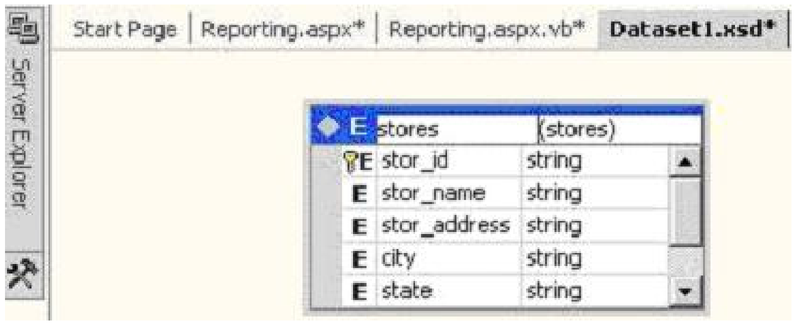
A file for creating an XML schema with DataSet classes

Name: Dataset1.xsd

2) Drag and drop the "Stores" table (within the PUBS database) from the "SQL Server" Item under "Server Explorer".



3) This should create a definition of the "Stores" table within the Dataset



E	stores	(stores)
E	stor_id	string
E	stor_name	string
E	stor_address	string
E	city	string
E	state	string

The .xsd file created this way contains only the field definitions without any data in it. It is up to the developer to create the connection to the database, populate the dataset and feed it to the Crystal Report.

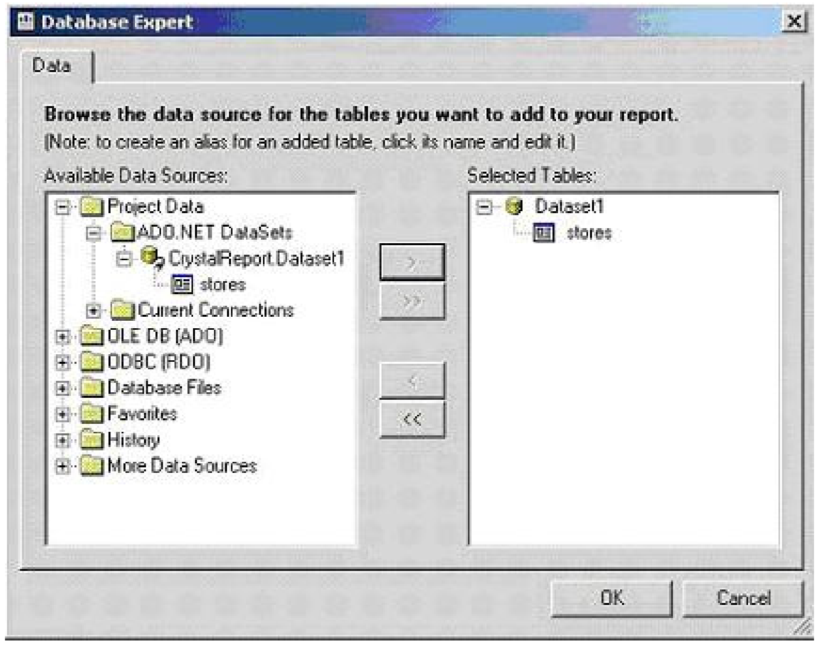
II. Creating the .rpt File

4) Create the report file using the steps mentioned previously. The only difference here is that instead of connecting to the Database thru Crystal Report to get to the Table, we would be using our DataSet that we just created.

5) After creating the .rpt file, right click on the "Details" section of the Report file, select "Add/Remove Database"

6) In the "Database Expert" window, expand "Project Data" (instead of "OLE DB" that was selected in the case of the PULL Model), expand "ADO.NET DataSet", "DataSet1", and select the "Stores"table.

7) Include the "Stores" table into the "Selected Tables" section by clicking on ">" and then Click "ok"



8) Follow the remaining steps to create the report layout as mentioned previously in the PULL Model to complete the .rpt Report file creation

III. Creating a CrystalReportViewer Control

9) Follow the steps mentioned previously in the PULL Model to create a Crystal Report Viewer control and set its properties.

Code Behind Page Modifications: 10) Call this subroutine in your page load event:

```
  Sub BindReport()
     Dim myConnection As New SqlClient.SqlConnection()
     myConnection.ConnectionString = "server=
(local)\NetSDK;database=pubs;Trusted_Connection=yes"
```

```
        Dim MyCommand As New SqlClient.SqlCommand()
        MyCommand.Connection = myConnection
        MyCommand.CommandText = "Select * from Stores"
        MyCommand.CommandType = CommandType.Text
        Dim MyDA As New SqlClient.SqlDataAdapter()
        MyDA.SelectCommand = MyCommand
        Dim myDS As New Dataset1()
        'This is our DataSet created at Design Time
        MyDA.Fill(myDS, "Stores")
        'You have to use the same name as that of your Dataset that you created during design
time
        Dim oRpt As New CrystalReport1()
        ' This is the Crystal Report file created at Design Time
        oRpt.SetDataSource(myDS)
        ' Set the SetDataSource property of the Report to the Dataset
        CrystalReportViewer1.ReportSource = oRpt
        ' Set the Crystal Report Viewer's property to the oRpt Report object that we created
    End Sub
```

Note: In the above code, you would notice that the object oRpt is an instance of the "Strongly Typed" Report file. If we were to use an "UnTyped" Report then we would have to use a ReportDocument object and manually load the report file into it.

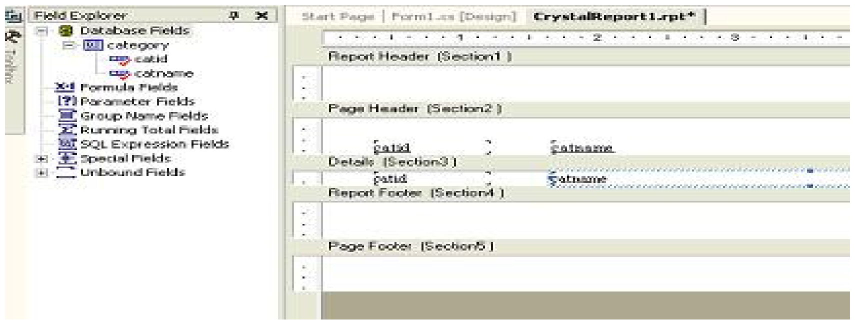
Enhancing Crystal Reports

Accessing filtered data through Crystal reports

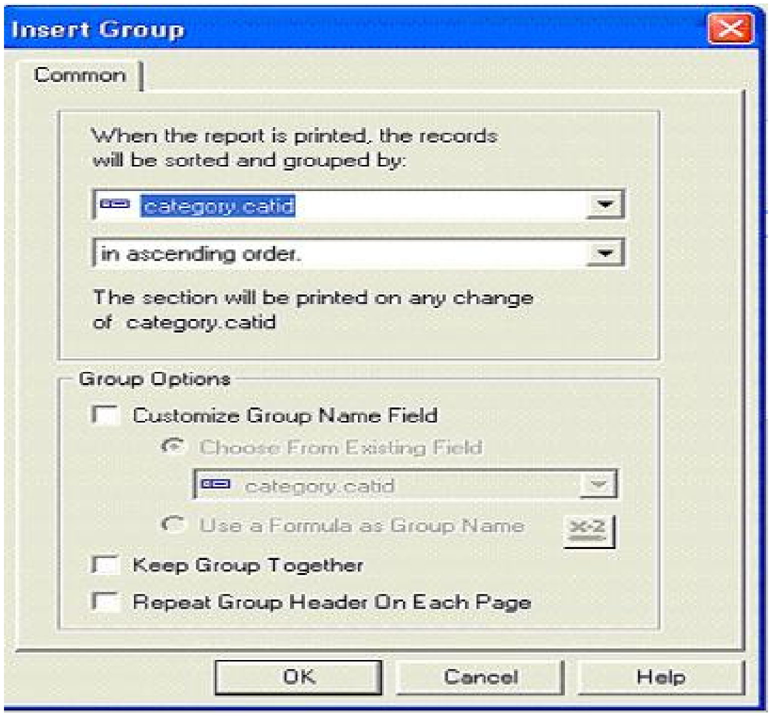
Perform the following steps for the same.

1. Generate a dataset that contains data according to your selection criteria, say "where (cost>1000)".

2. Create the Crystal Report manually. It would look like this.



3. Right Click Group Name fields in the field Explorer window and select insert Group from the shortcut menu. Select the relevant field name from the first list box as shown.


Insert Group

Common

When the report is printed, the records will be sorted and grouped by:

category.catid

in ascending order.

The section will be printed on any change of category.catid

Group Options

Customize Group Name Field

Choose From Existing Field

category.catid

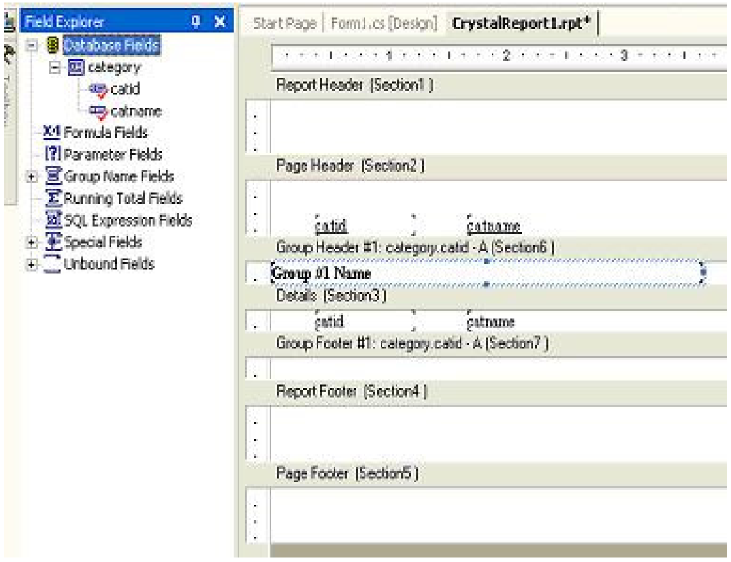
Use a Formula as Group Name

Keep Group Together

Repeat Group Header On Each Page

The group name field is created, since the data needs to be grouped on the basis of the cat id say.

4. A plus sign is added in front of Group Name Filed in the field explorer window. The Group Name Field needs to be added to the Group Header section of the Crystal Report. Notice this is done automatically.



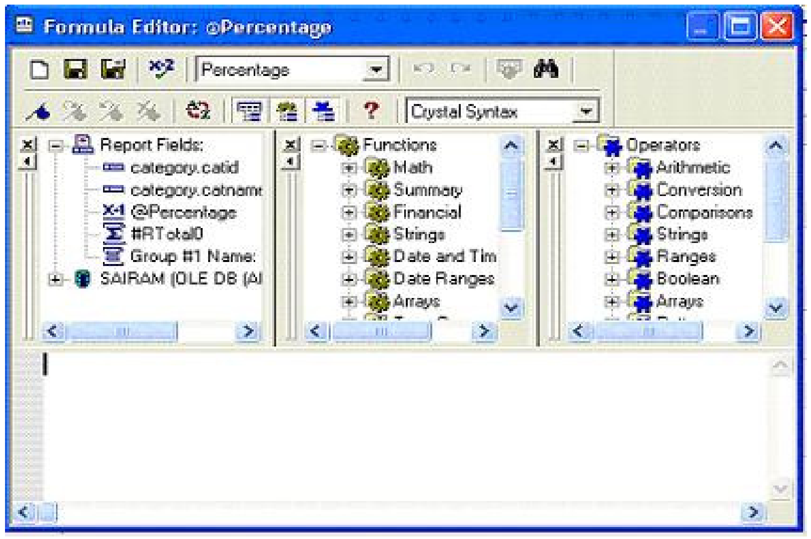
5. Right Click the running total field and select new. Fill the required values through > and the drop down list.

6. Since the count of number of categories is to be displayed for the total categories, drag RTotal0 to the footer of the report.

Create a formula

Suppose if the report required some Calculations too. Perform the following steps:

1. Right Click the formula Fields in the field explorer window and select new. Enter any relevant name, say percentage.



2. A formula can be created by using the three panes in the dialog box. The first pane contains all the crystal report fields, the second contains all the functions, such as Avg, Sin, Sum etc and the third contains operators such as arithmetic, conversion and comparison operators.

3. Double click any relevant field name from the forst pane, say there's some field like advance from some CustOrder table. Then expand Arithmetic from the third pane and double click Divide operator.

4. Double click another field name from the first which you want to use as divisor of the first field name already selected say it is CustOrder.Cost.

5. Double Click the Multiply from third pane and the type 100.

6. The formula would appear as {CustOrder.Advance}/{ CustOrder.Cost} * 100.

7. Save the formula and close Formula Editor:@Percentage dialog box.

8. Insert the percentage formula field in the details pane.

9. Host the Crystal report.

Exporting Crystal reports

When using Crystal Reports in a Web Form, the CrystalReportViewer control does not have the export or the print buttons unlike the one in Windows Form. Although, we can achieve export and print functionality through coding. If we export to PDF format, Acrobat can handle the printing for us, as well as saving a copy of the report.

You can opt to export your report file into one of the following formats:

- PDF (Portable Document Format)
- DOC (MS Word Document)
- XLS (MS Excel Spreadsheet)
- HTML (Hyper Text Markup Language - 3.2 or 4.0 compliant)
- RTF (Rich Text Format)

To accomplish this you could place a button on your page to trigger the export functionality.

When using Crystal Reports in ASP.NET in a Web Form, the CrystalReportViewer control does not have the export or the print buttons like the one in Windows Form. We can still achieve some degree of export and print functionality by writing our own code to handle exporting. If we export to PDF format, Acrobat can be used to handle the printing for us and saving a copy of the report.

Exporting a Report File Created using the PULL Model

Here the Crystal Report takes care of connecting to the database and fetching the required records, so you would only have to use the below given code in the Event of the button.

```
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e AsSystem.EventArgs) H
andles Button1.Click
     Dim myReport As CrystalReport1 = New CrystalReport1()
     'Note : we are creating an instance of the strongly-typed Crystal Report file here.

     Dim DiskOpts As CrystalDecisions.Shared.DiskFileDestinationOptions
= NewCrystalDecisions.Shared.DiskFileDestinationOptions
     myReport.ExportOptions.ExportDestinationType =
CrystalDecisions.[Shared].ExportDestinationType.DiskFile
     ' You also have the option to export the report to other sources
     ' like Microsoft Exchange, MAPI, etc.

     myReport.ExportOptions.ExportFormatType =
CrystalDecisions.[Shared].ExportFormatType.PortableDocFormat
     'Here we are exporting the report to a .pdf format.  You can
     ' also choose any of the other formats specified above.

     DiskOpts.DiskFileName = "c:\Output.pdf"
     'If you do not specify the exact path here (i.e. including
     ' the drive and Directory),
     'then you would find your output file landing up in the
     'c:\WinNT\System32 directory - atleast in case of a
     ' Windows 2000 System
     myReport.ExportOptions.DestinationOptions = DiskOpts
     'The Reports Export Options does not have a filename property
```

```
        'that can be directly set. Instead, you will have to use
        'the DiskFileDestinationOptions object and set its DiskFileName
        'property to the file name (including the path) of your  choice.
        'Then you would set the Report Export Options
        'DestinationOptions property to point to the
        'DiskFileDestinationOption object.

        myReport.Export()
        'This statement exports the report based on the previously set properties.

    End Sub
```

Exporting a Report File Created Using the PUSH Model

Using Push Model, the first step would be to manually create connections and populate the Dataset, and set the Report's "SetDataSource" property to the required Dataset (in the same manner as shown in the Push Model example before). The next step would be to call the above given Export.

THE ROLE OF ADO.NET IN A DISTRIBUTED SYSTEM

ADO.NET is a critical piece of almost any enterprise application simply because most applications need to access a database at some point. In this case, using the legacy ADO libraries makes little sense. They reduce performance (because execution must travel between your managed .NET code and the unmanaged ADO objects through a COM interop bridge) and add few useful features.

ADO.NET supports more than one model of data access. First of all, it uses a simple connection-based paradigm that enables you to execute individual commands. You'll find that this is the most useful aspect of ADO.NET for designing database components using the "service provider" design pattern introduced in Chapter 2. ADO.NET also provides disconnected data access, which is a quick-and-easy way to send information between tiers. In this case, the DataSet essentially takes the role of the "information package" component. It's an enticing option, but it strips away a layer of indirection (the custom information package class), potentially making your solution less generic and making it more difficult to update the client in response to data source changes.

Finally, ADO.NET can play a special role in a system that incorporates non-.NET clients. Internally, the DataSet stores information in a binary form optimized for XML output. That means you can look at the DataSet as an XML document without the danger of losing information. Clients in other programming languages won't know how to manipulate aDataSet object, but they can usually parse an XML document and schema, which makes it easy to transfer information around without needing to write your own error-prone conversion code.

The ADO.NET Object Family

The ADO.NET types are a part of the .NET Framework class library and are found in the namespaces that begin with System.Data.

- System.Data contains generic classes that are independent of the type of data source or the mechanism used to connect to it. The most important of these is the DataSet, which contains collections of other generic classes such as DataTable, DataRow, and DataColumn. There are also a number of generic interfaces (such as IDbCommand and IDbConnection) that are implemented by all provider-specific objects.

- System.Data.SqlClient contains the classes required to connect to a Microsoft SQL Server (7.0 or later) database using the highly efficient Tabular Data Stream (TDS) interface. In other words, these namespaces bypass several other potential layers, including OLE DB and ODBC. You'll use these classes to execute a direct command on a SQL Server database or to fill a DataSet with the results of a query.
- System.Data.OleDb contains the classes required to connect to most OLE DB providers. (OLE DB providers are included for all kinds of data sources, including Microsoft Access, SQL Server, and Oracle.) You can use these classes to execute a direct command on a data source through an OLE DB provider or to fill aDataSet with the results of a query.
Version 1.0 of the .NET Framework includes only these two providers: one that's generic for all OLE DB providers and one that's optimized for SQL Server. However, version 1.1 of the .NET Framework and Visual Studio .NET 2003 add the following new providers:

- System.Data.OracleClient contains the classes required to connect to an Oracle database (version 8.1.7 or later) using the optimized Oracle Call Interface (OCI).
- System.Data.Odbc contains the classes required to connect to most ODBC drivers' providers. ODBC drivers are included for all kinds of data sources and are configured through the Data Sources icon in Control Panel.
In addition, other vendors are sure to develop their own managed providers. The best strategy for the application programmer is to use the managed provider that's customized for the data source, if it's available, and thereby bypass as many intermediate layers as possible. Keep in mind that if you install a third-party provider, it is installed (generally in the GAC) as a separate DLL assembly, and you must reference this assembly before you can use the contained types. Third-party providers also won't be placed in a System.Datanamespace (because System is reserved for the .NET Framework class library).

The Data Objects

At first, it might seem that Microsoft has broken with its long-standing tradition of cross-data-source compatibility. How else can you explain a separate namespace (and potentially a separate assembly) for every type of provider?

On closer examination, ADO.NET is generic in two ways:

- Although separate providers are required to connect to the data source, the DataSet object is generic. Therefore, a client can navigate and update a DataSet regardless of how it was created or where the information originated. All the details that are specific to the data provider can be encapsulated by a custom database component that you create.
- The provider-specific classes all inherit from a common class (in the System.Data.Common namespace) or implement a common interface (from the System.Data namespace). Therefore, the SqlConnection object exposes virtually the same properties and methods as the OleDbConnection and OdbcConnection objects. If you need to work with these objects generically in code, you can cast them to the appropriate interface (such as IDbConnection).
In some cases, classes or members are not directly inherited or implemented from a common base class or interface. (One example is the SqlCommandBuilder and OleDbCommandBuilder classes.) In this case, you'll still find that the classes included with different providers work essentially the same and almost always include properties and methods with exactly the same names. That means that

examples written with one provider are easily translatable to another. Connection strings are one of the minor discrepancies.
So what are the basic ADO.NET objects? I won't describe them one by one here, but for ADO.NET novices, a quick overview is provided in Table.

Function	**SQL Server Provider**	**OLE DB Provider**	**Oracle Provider**	**ODBC Provider**
Connect to a data source	SqlConnection	OleDbConnecti on	OracleConnecti on	OdbcConnectio n
Execute a SQL statement	SqlCommand	OleDbComman d	OracleComman d	OdbcCommand
Execute an SQL stored procedure	SqlCommand (with any number of referencedSqlP arameterobject s)	OleDbComman d(with any number of referencedOleD bParameterobjec ts)	OracleComman d(with any number of referencedOracl eParameterobjec ts)	OdbcCommand (with any number of referencedOdbc Parameterobject s)
Retrieve data with a fast-forward read-only	SqlDataReader	OleDbDataRead er	OracleDataRead er	OdbcDataRead er
Transfer data to aDataSet	SqlDataAdapte r	OleDbDataAdap ter	OracleDataAdap ter	OdbcDataAdapt er
ApplyDat aSetchang es to a data source	SqlDataAdapte r(and, optionally, theSqlComma ndBuilderhelp er class)	OleDbDataAdap ter(and, optionally, theOleDbComm and- Builder helper class)	OracleDataAdap ter(and, optionally, theOracleComm and- Builder helper class)	OdbcDataAdapt er(and, optionally, the Odbc- CommandBuild erhelper class)
Manually initiate and manage a transactio n	SqlTransaction	OleDbTransacti on	OracleTransacti on	OdbcTransactio n

Function	**SQL Server Provider**	**OLE DB Provider**	**Oracle Provider**	**ODBC Provider**
Catch an error from the data source (anything from invalid SQL to an error connectin g)	SqlException	OleDbException	OracleException	OdbcException

Table :ADO.NET Core Provider Classes

Direct Data Source Interaction

The easiest form of access with ADO.NET is simple, connection-based access. With connection-based access, you have one of the following goals:

- Execute a direct SQL Update, Delete, or Insert command.
  You'll need to use the appropriate IDbConnection and IDbCommand objects.
- Execute a query and retrieve the results as a read-only forward-only stream.
  You'll need to use the appropriate IDbConnection, IDbCommand, and IDataReader objects.
  The drawback of this strategy is that substantial data manipulation can result in substantial connection requests, potentially creating a bottleneck. For modest database usage, however, the simplicity and straightforwardness of this model outweighs most other considerations. Also, most ADO.NET providers support some form of connection pooling, which allows them to efficiently handle a large volume of short-lived requests.

The example in Listing-1shows how the CustomerDB service provider class can use the ADO.NET objects to support record deletions, insertions, and updates.

Listing -1 Adding ADO.NET code to the CustomerDB class

Imports System.Data.SqlClient Imports System.Data Public Class CustomerDB Private ConnectionString As String = _ "Data Source=localhost;Initial Catalog=Store;" & _ "Integrated Security=SSPI" ' (GetCustomer and GetCustomers methods left out.) ' Inserts a customer record into the database. Public Sub AddCustomer(ByVal customer As CustomerDetails) Dim Sql As String = "INSERT INTO Customers " Sql &= "(FullName, EmailAddress, Password) VALUES ('" Sql &= customer.Name & "', '" Sql &= customer.Email & "', '" Sql &= customer.Password & "')" ExecuteNonQuery(Sql) End Sub ' Updates an existing customer record in the database. Public Sub UpdateCustomer(ByVal customer As CustomerDetails) Dim Sql As String = "UPDATE Customers SET " Sql &= "FullName='" & customer.Name Sql &= "', EmailAddress='" & customer.Email Sql &= "', Password='" & customer.Password Sql &= "' WHERE CustomerDELETE FROM Customers

WHERE Customercaller inform" exception handling pattern. Throw New ApplicationException(_ "Exception encountered when executing command.", Err) Finally con.Close() End Try End Sub End Class

The CustomerDB class is organized so that a private ExecuteNonQuery method performs the database operation for all three tasks. The only difference is the supplied SQL text□in this design, the methods of theCustomerDB class just generate a dynamic SQL statement that identifies the field names and values. Using the ADO.NET objects, a new connection is opened, a direct command is executed, and the connection is closed immediately.

If an exception is encountered, the class doesn't try to remedy it because it probably indicates that the database is unavailable or the SQL text is invalid. Instead, an ApplicationException is thrown to the client, with the original SqlException packaged inside. This pattern is called caller inform because it ensures that the caller is notified that an unrecoverable error took place, but it uses a higher-level exception class that the client expects. This higher-level exception class could also be a custom exception class that you've defined.

With the caller inform pattern, the ApplicationException always includes the original Exception object nested inside. This approach can help the client diagnose the problem, but it might not be suitable in a secured environment, in which specific error information could inform a malicious user about potential vulnerabilities. In this case, create the ApplicationException object without nesting the original exception inside:

Throw New ApplicationException(_ "Exception encountered when executing command.")

Connection Strings

Connection strings generally detail three types of information: what server to use, what database to use, and how to authenticate the connection. There also might be any number of additional vendor-specific settings.

In Listing 3-1, we use the localhost alias (which always refers to the current computer), the database named Store, and integrated security (which uses the currently logged-in Windows user to access the database).

Private ConnectionString As String = _ "Data Source=localhost;Initial Catalog=Store;" & _ "Integrated Security=SSPI"

If integrated security isn't supported, the connection must indicate a valid user and password combination. For a newly installed SQL Server database, the sa (system administrator) account is usually present, and often without a password.

Private ConnectionString As String = _ "Data Source=localhost;Initial Catalog=Store;" & _ "user id=sa;password="

If you're using the OLE DB provider, your connection string probably resembles the preceding example, with the addition of a Provider setting that identifies the OLE DB driver (or if you're using the ODBC .NET driver, you add a Driver attribute that identifies the DSN). The following connection string can be used to connect to an Oracle database through the MSDAORA OLE DB provider:

Private ConnectionString As String = _ "Data Source=localhost;Initial Catalog=Store;" & _ "user id=sa;password=;Provider=MSDAORA"

If you're using a third-party data provider, you might need to consult its documentation (or the MSDN class library reference) to determine the supported connection string values. For example, most databases support the Connect Timeout setting, which sets the number of seconds to wait for a connection before throwing an exception. (The SQL Server default is 15 seconds.)

Direct Queries and Data Readers

Queries are performed in a slightly different way. The SQL Select statement is executed through the command's ExecuteReader method, which returns a data reader. You can then step through the rows returned by the data reader one at a time from start to finish, using the Read method. The GetCustomer andGetCustomers methods shown in Listing -2 demonstrate this technique and create CustomerDetails objects to encapsulate the retrieved details. Lookup is based on customer ID, but you could make overloaded versions of this method that enable you to search by other criteria (such as the FullName field).

Listing -2 Adding query methods to CustomerDB

```
 Public Class CustomerDB
    ' (Update, insert, and delete methods omitted.)
    ' Retrieves a single customer record from the database.
    Public Function GetCustomer(ByVal customerID As Integer) _      As CustomerDetails
        Dim Sql As String = "SELECT * FROM Customers Where CustomerCustomerID")
            Customer.Name = Reader("FullName")
            Customer.Email = Reader("EmailAddress")
            Customer.Password = Reader("Password")        Catch Err As Exception
            ' Use caller inform pattern.
        Throw New ApplicationException( _
"Exception encountered when executing command.", Err)
     Finally
   con.Close()
  End Try
 Return Customer
  End Function
  ' Retrieves all customer records into an ArrayList collection.
    ' Note that exposing this method allows client to invoke a
' potentially time-consuming query.
  Public Function GetCustomers() As ArrayList
        Dim Sql As String = "SELECT * FROM Customers"
        Dim con As New SqlConnection(ConnectionString)
        Dim cmd As New SqlCommand(Sql, con)
        Dim Reader As SqlDataReader
        Dim Customers As New ArrayList()
 Try
     con.Open()
       Reader = cmd.ExecuteReader()
         Do While Reader.Read()
                Dim Customer As New CustomerDetails()
                Customer.ID = Reader("CustomerID")
                Customer.Name = Reader("FullName")
                Customer.Email = Reader("EmailAddress")
                Customer.Password = Reader("Password")
                Customers.Add(Customer)
```

```
      Loop
   Catch Err As Exception
     ' Use caller inform pattern.
       Throw New ApplicationException( _
"Exception encountered when executing command.", Err)
        Finally
             con.Close()
        End Try
         Return Customers
    End Function
 End Class
```

Note that the GetCustomer method passes a special CommandBehavior.SingleRow parameter to theExecuteReader method. This isn't required, but it does represent a little-known technique that can improve performance if you know you're retrieving only a single row. In this case, the code selects a record using its unique key number, which ensures that only a single item is returned.
Notice also that the GetCustomers method returns an ArrayList full of CustomerDetails objects. Alternatively, you can create a strongly typed custom collection class, such as the one shown in Listing -3.

Listing -3 A custom collection for CustomerDetails

```
 Public Class CustomerDetailsCollection
   Inherits System.Collections.CollectionBase
    Public Sub Add(ByVal customer As CustomerDetails)
      Me.List.Add(customer)
  End Sub
  Public Sub Remove(ByVal Index As Integer)
        ' Check to see if there is an item at the supplied index.
      If Index > Count - 1 Or Index < 0 Then
            Throw New System.IndexOutOfRangeException()
  Else
  List.RemoveAt(Index)
    End If
  End Sub

   Public ReadOnly Property Item(ByVal Index As Integer) _
   As CustomerDetails
      Get
  ' The appropriate item is retrieved and
       ' explicitly cast to the CustomerDetails type.
            Return CType(List.Item(Index), CustomerDetails)
       End Get
  End Property
 End Class
```

This technique ensures that the collection can only contain CustomerDetails objects, not any other type of object. The remarkable fact is that both the custom collection class and

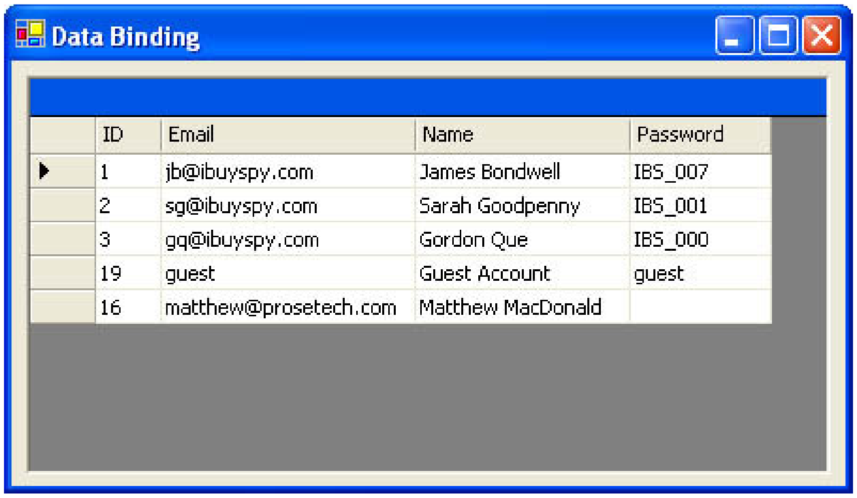
the ArrayList intrinsically support ASP.NET and Windows data binding. That means it takes just a few lines of code to display the results you retrieve without needing to iterate through each item. (As a disadvantage, it's usually harder to configure specific formatting details such as column order and size.)
Therefore, you can create a scalable database component that executes direct commands, returns custom objects, and even allows data binding, without resorting to the DataSet.
Listing -4 shows the code for a simple form that uses the CustomerDB service provider class in this fashion.

Listing -4 A data bound form that uses CustomerDB

```
 Public Class BoundForm
  Inherits System.Windows.Forms.Form
    Friend WithEvents DataGrid1 As System.Windows.Forms.DataGrid
    ' (Designer code omitted.)
    Private Sub BoundForm_Load(ByVal sender As System.Object, _
      ByVal e As System.EventArgs) Handles MyBase.Load
      Dim DB As New CustomerDB()
    DataGrid1.DataSource = DB.GetCustomers()
 End Sub
End Class
```

Figure shows the sample output that displays when this form is used with the CustomerDB component.

Figure : Data binding without a DataSet object



ID	Email	Name	Password
1	jb@ibuyspy.com	James Bondwell	IBS_007
2	sg@ibuyspy.com	Sarah Goodpenny	IBS_001
3	gq@ibuyspy.com	Gordon Que	IBS_000
19	guest	Guest Account	guest
16	matthew@prosetech.com	Matthew MacDonald	

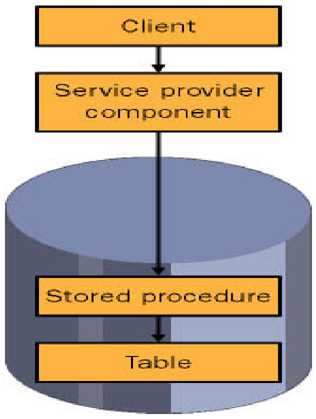
Note

In this discussion, you saw two command methods, ExecuteReader (for running queries and returning a set of rows) and ExecuteNonQuery (which returns only the number of affected rows). You can also use a third method, ExecuteScalar, which returns a single value. ExecuteScalar is ideal for using aggregate SQL functions (such as Sum, Avg, Min, Max, and Count) or for calling a stored procedure that returns a single data value.

Stored Procedures

SQL "programs" that are stored in the database are a key part of any enterprise programming project. If you study Microsoft's platform samples (such as the IBuySpy case studies for ASP.NET), you'll see that they advocate a disciplined multilayered approach. The client Web page code interacts exclusively with a dedicated custom data component, which is similar to the CustomerDB class shown previously. The data component interacts with the data source through a layer of stored procedures, which in turn manipulate the underlying tables and records. This structured approach is shown in Figure -2.

Figure -2. The ideal structured approach to data access



Some key advantages to stored procedures include the following:

- They reduce the amount of data access code.
  With stored procedures, work is offloaded to the server, simplifying your .NET code.
- They improve encapsulation.
  By using stored procedures, you can tighten security (by allowing access to certain specific procedures instead of granting full access to the whole underlying table) and combine a batch of commands into a single logical task. You can also modify and tweak this logic to fine-tuned perfection after the application has been deployed, without needing to recompile the client or data access components.
- They often improve performance.
  Performance improves because the database can create an optimized execution path when stored procedures are created, instead of performing the same work each time they are executed.

The last point depends on the database itself. For example, in SQL Server 2000, dynamic queries are compiled and cached much as stored procedures are. However, in many cases, stored procedures can still improve performance. One reason is that a single stored procedure can often replace dozens of differently structured (but similar) dynamic SQL queries. It's much easier to optimize a small set of stored procedures and create the optimal set of indexes than to sort through dozens of dynamic queries that are embedded into various applications. For more information about database optimization, refer to Chapter 12.

Generally, stored procedures take a little more effort because you need to work with the script-based SQL language. Listing -5 shows an example of a stored procedure that inserts a new customer record and then returns the automatically generated CustomerID.

Listing -5 A stored procedure for adding a customer record

```
 CREATE Procedure CustomerAdd (     @FullName   nvarchar(50),
    @Email      nvarchar(50),     @Password  nvarchar(50),     @CustomerID int OUTPUT )
AS  INSERT INTO Customers (     FullName,     EMailAddress,     Password )  VALUES (
    @FullName,     @Email,     @Password )  SELECT     @CustomerID = @@Identity
```

You can easily rewrite the AddCustomer method to use this stored procedure. As an added frill, this method now returns the unique CustomerID. (See Listing -6.)

Listing 36 Using the stored procedure in CustomerDB

```
 Public Class CustomerDB      ' (Other methods left out.)
    Public Function AddCustomer(ByVal customer As CustomerDetails) _      As Integer
        ' Create the ADO.NET Connection and Command.
        Dim con As New SqlConnection(ConnectionString)
        Dim com As New SqlCommand("CustomerAdd", con)
        com.CommandType = CommandType.StoredProcedure
        Dim Param As SqlParameter         ' Add the three input parameters.
        Param = com.Parameters.Add( _         "@FullName", SqlDbType.NVarChar, 50)
        Param.Value = CustomerDetails.Name
        Param = com.Parameters.Add("@Email", SqlDbType.NVarChar, 50)
        Param.Value = CustomerDetails.Email
        Param = com.Parameters.Add("@Password", SqlDbType.NVarChar, 50)
        Param.Value = CustomerDetails.Password         ' Add the output parameter.
        Param = com.Parameters.Add("@CustomerID", SqlDbType.Int)
        Param.Direction = ParameterDirection.Output
     Try
   con.Open()
     cmd.ExecuteNonQuery()
     Catch Err As Exception
   ' Use "caller inform" exception handling pattern.
        Throw New ApplicationException( _
"Exception encountered when executing command.", Err)
  Finally
 con.Close()
 End Try
  ' Return the unique ID generated by the database.
   Return Param.Value
   End Function
 End Class
```

As you can see, the use of stored procedures forces your code to specify more database details (such as the data types of various parameters). By using a disciplined database component, however, you can still maintain order.

Provider-Agnostic ADO.NET Code

It is possible to create a custom data component that can transparently work with any type of ADO.NET provider, by performing all your database operations through the generic interfaces in the System.Datanamespace. This approach isn't always best□occasionally it might surrender some provider-specific functionality you need□but it can be indispensable in projects where more than one data source stores similar tables with different information or where you plan to migrate from one database product to another.
The basic approach is to start by creating the provider-specific command object that you need

to use, with a small piece of conditional logic. From that point, you can create commands, data readers, and parameters, all by relying on methods and using the generic interfaces.

Listing 3-7 shows how you can apply this technique to the GetCustomers method. Note that the class constructor accepts an enumerated value that indicates the provider type. This allows the client to decide, at run time, which provider to use. The conditional portion of logic that creates the first provider-specific object is placed in a
private CreateConnection function.

Listing -7 CustomerDB with generic ADO.NET code

```
 Imports System.Data.SqlClient
Imports System.Data.OleDb
Imports System.Data
 Public Class CustomerDB
    ' (Other methods omitted.)
   ' Defines the supported providers types.
  Public Enum ProviderType
    SqlServer
   OleDb
 End Enum
 ' The provider this class will use.
  Public Provider As ProviderType
     Private SqlConnectionString As String = _
"Data Source=localhost;Initial Catalog=Store;" & _      "Integrated Security=SSPI"
     Private OleDbConnectionString As String =_
"Data Source=localhost;Initial Catalog=Store;"     & _
"Integrated Security=SSPI;Provider=SQLOLEDB"
    Public Sub New(provider As ProviderType)
    Me.Provider = provider
   End Sub
  Public Function GetCustomers() As ArrayList
     Dim Sql As String = "SELECT * FROM Customers"
        ' The connection object is generated through a private
     ' function.
    Dim con = CreateConnection()
      ' Interact with the command and data reader
        ' using the IDbCommand and IDataReader interfaces.
        Dim cmd As IDbCommand = con.CreateCommand()
       Dim Reader As IDataReader
      Dim Customers As New ArrayList()
    Try
        con.Open()
      Reader = cmd.ExecuteReader()
       Do While Reader.Read()
    Dim Customer As New CustomerDetails()
          Customer.ID = Reader("CustomerID")
           Customer.Name = Reader("FullName")
               Customer.Email = Reader("EmailAddress")
               Customer.Password = Reader("Password")
       Customers.Add(Customer)
   Loop
  Catch Err As Exception
      ' Use caller inform pattern.
       Throw New ApplicationException( _
"Exception encountered when executing command.", Err)
       Finally
   con.Close()
  End Try
  Return Customers
  End Function
 Private Function CreateConnection As IDbConnection
       ' Here we determine which object and connection string to use.
       ' This is the only provider-specific part of the code.
      Select Case Provider
   Case ProviderType.SqlServer
      Return New SqlConnection(SqlConnectionString)
      Case ProviderType.OleDb
     Return New OleDbConnection(SqlConnectionString)
      End Select
   End Function
 End Class
```

A separate connection string is required for each data source. If you're using SQL Server, you can test this sample by connecting to the database through the SQL managed provider or the OLE DB provider for SQL Server (SQLOLEDB).

This approach does have some limitations many of which Microsoft architects are working to remedy in future versions of .NET. Some of the problems include objects that don't use a common interface (like the command builders) and the use of provider-specific exceptions (such as SqlException and OleDbException).

Transactions

When many developers hear the word transaction, they automatically think of COM+ services or its predecessor, the curiously named Microsoft Transaction Service (MTS). That's because one of the early successes of MTS and COM+ was a new model for transaction enlistment that made it possible to create distributed transactions that could bind different data sources into one logical operation. This service is priceless when you need to integrate

legacy systems and multiple database products in an evolving system. (In Chapter 9, we'll explore this service in more detail.)
However, you won't use COM+ distributed transactions when designing a new system. Instead, you'll rely on ordinary SQL transactions that are initiated and governed at the database level. This approach offers better performance and easier programming.

The best way to create a transaction is to encapsulate it entirely within a stored procedure. In that case, you won't see it in your client code. Listing -8 shows the basic structure used to initiate a transaction in SQL Server.

Listing -8 A transaction in stored procedure code

```
 BEGIN TRANSACTION
  < SQL code here >
IF (@@ERROR > 0)
  ROLLBACK TRANSACTION
ELSE
COMMIT TRANSACTION
```

You can also initiate the same type of transaction programmatically through .NET code. The disadvantage with this approach is that it requires the good behavior of the client. In other words, if a client forgets to initiate a transaction, it has the ability to perform an inconsistent update.

ADO.NET has provider-specific IDbTransaction objects that manage client-initiated transactions. You don't create this object directly; instead, you call
the BeginTransaction method of the connection object. You can then enlist multiple commands and commit or roll back the transaction using the IDbTransaction methods when complete. Listing -9 shows a client-initiated transaction.

Listing -9 A client-initiated transaction

```
 Dim Transaction As SqlTransaction = con.BeginTransaction()
' Enlist two commands (cmdA and cmdB) in the transaction.
' They will now succeed or fail as a unit.

 cmdA.Transaction = Transaction

 cmdB.Transaction = Transaction

 ' Execute the commands.

cmdA.ExecuteNonQuery()

cmdB.ExecuteNonQuery()

 ' Commit the transaction.

Transaction.Commit()
```