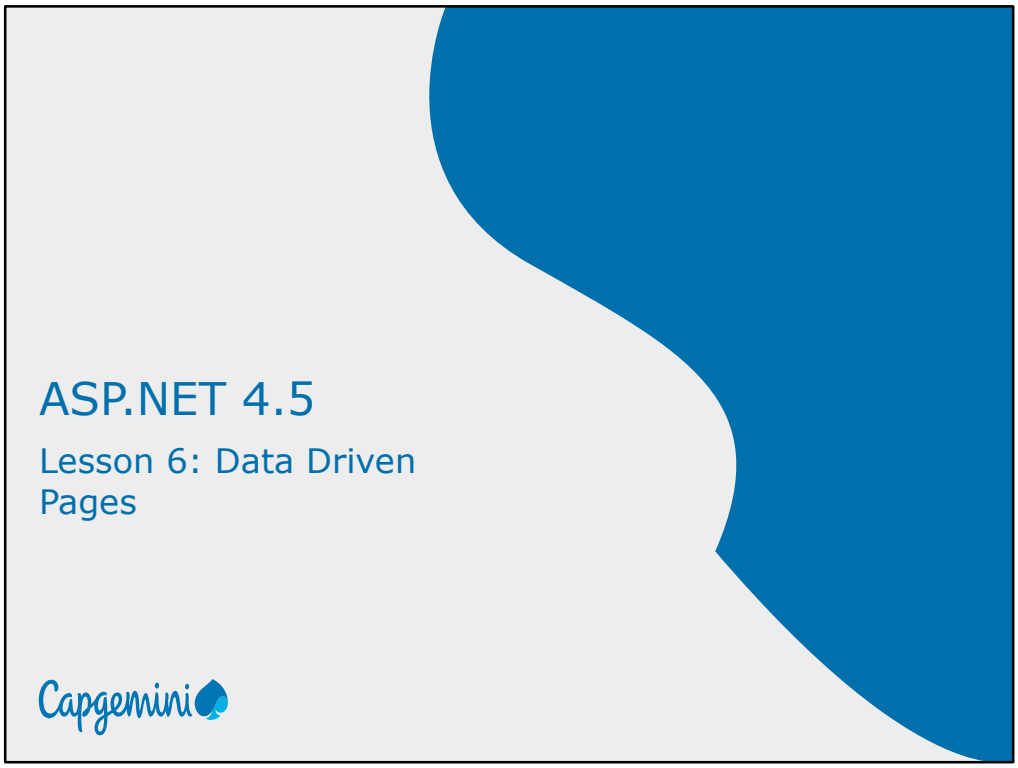

ASP.NET 4.5
Lesson 6: Data Driven Pages
Capgemini

# Lesson Objectives

- After this lesson you will understand:
  - Using ADO.NET in Web Applications
  - Using DataSource controls provided in ASP.NET
  - Using GridView, FormView, Repeater and DetailsView Controls
  - Using Data Bound Controls such as RadioButtonList, DropDownList, CheckBoxList and ListBox

- Slide only with Notes pages

Any web application worth writing involves data access. In ASP.NET, Microsoft has taken the concept of data binding and expanded it to make data binding easier to understand and use.
A new layer of data abstraction called DataSource controls is introduced. In this module we will explore the data source controls and the data binding features in ASP.NET.
The data binding shows how we can use the data source controls to easily and quickly bind data to data bound controls.

In the previous versions, a developer ended up writing data access code to retrieve the Dataset or DataReader. And then the binding with control like DataGrid , DropDownList etc.

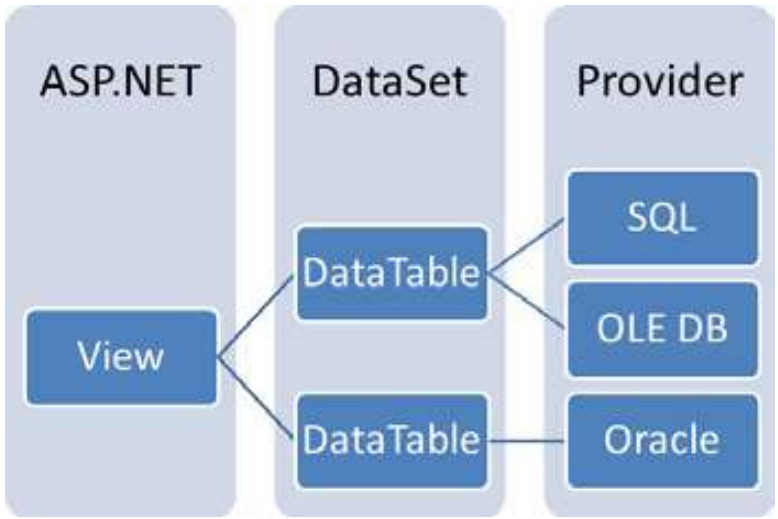
ASP.NET introduces an additional layer of abstraction through the use of datasource controls. These controls abstract the use of an underlying data provider such as SQL or OLEDB data provider. This means that you no longer need to concern yourself with why and how you have to use the data provider. Instead the data source controls does all this intense work for you. You only need to specify where the data is located and at maximum to construct a query.

The data source controls are derived from the Control class, they can be used like any other web server control. You can define the data access without having to write any code either declaratively in your HTML or programmatically.

You use DataBound controls to generate your application's user interface for working with data. The DataBound controls can be used to display and edit database data, XML data, or just any other type of data you can imagine.

6.1.: ASP.NET and ADO.NET

## Asp.Net and ADO.Net



6.1 ASP.NET and ADO.NET

# ADO.NET with ASP.NET

- Understanding Database Access in ASP.NET

6.2: Data source Controls

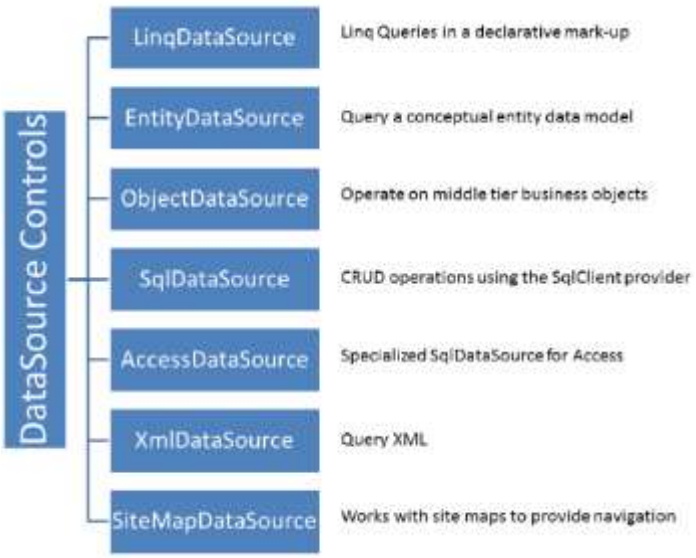
# Data Source Controls

- DataSource controls handle communication between data sources and controls, used for presentation in an abstract and generic way
- ASP.NET provides following DataSource controls:
  - SqlDataSource
  - ObjectDataSource
  - XMLDataSource
  - SiteMapDataSource
  - AccessDataSource

ASP.NET introduces a DataSource controls which handle the communication between data sources and controls, which is used for presentation in an abstract and generic way. All the datasource controls are derived from the DataSourceControl class. The DataSourceControl inturn is derived from Control and implements the IDataSource and IlistSource interfaces. Because the Data Source controls themselves don't have any visualization of their own, they are displayed as a gray box in the IDE.
Following DataSources are provided:
SqlDataSource: Provides access to any Data source that has an ADO.Net Data provider available. For example, SQL Server, OLE DB, ODBC, Oracle, and other database systems.
ObjectDataSource: Provides specialized data access to business objects or other classes that return data.
XMLDataSource: Provides access to XML documents, either physically or in-memory.
SiteMapDataSource: Provides access to site map data for a Web site that is stored by the site map provider.
AccessDataSource: Provides specialized access to Access databases.
In ASP.NET 4.5 you have an option to provide Database access either through Manual coding like ASP.NET 1.1 or use DataSource controls. Since the code in DataSource controls is autogenerated most of the time debugging is tough. Hence many developers would still prefer to use the typical manual coding for Database access.
Note: Though ASP.NET provides these DataSource controls, some of the examples considered in this material use the ASP.NET 1.x method of code writing for data access

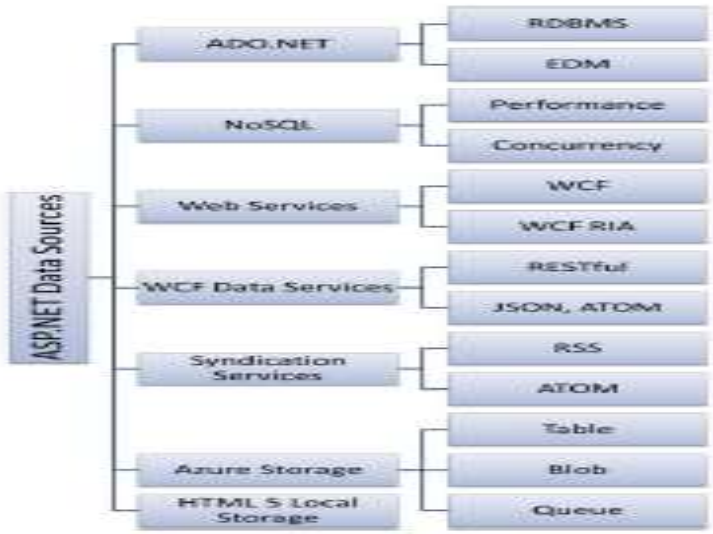
6.2.1 DataSource Controls

## Data Source Controls



6.2.2 Asp.Net Data Sources

## Asp.Net Data Sources




Syndication Services
A natural extension to using services as a source of data is to allow access to Syndication Feeds like RSS and ATOM.
ASP.NET is capable of reading and parsing Syndication Feeds and displaying appropriate results.
Azure Storage
Windows Azure provides storage capabilities in the form of Tables, Blobs, and Queues at no extra cost. ASP.NET applications can leverage Azure Storage as a data store and perform data access operations using the Open Data Protocol.
WCF Data Services
WCF Data Services enables CRUD operations on data using the Open Data Protocol (OData). Microsoft has inserted OData deep into its data access strategy. OData support is now available in all its new-generation platforms including SQL Server 2008 R2, SQL Server 2012, Windows Azure Storage, SharePoint 2013, and PowerPivot.
Data Services will allow REST-style access to data models with flexible querying and association traversal capabilities. This will allow natural integration with web platforms like ASP.NET. Data Services allow data to be represented in well-known formats such as JSON and ATOM.
HTML 5 Local Storage
This is relatively new, and it emphasizes the shift in focus on building applications that utilize the power of clients.
Local Storage is used to store structured data in web clients in a way that is similar to HTTP cookies, but with robust support for storing session data across multiple windows of the client.

6.3: Data Bound Controls

# Data Bound Controls

- DataBound list controls can be bound to database values declaratively i.e. without writing any code
- DataSource controls along with DataBound controls give developers a total ease of working with database application through ASP.NET
- There are new data-bound controls to render common UI for data, such as GridView, DetailsView, and FormView

6.4: GridView Control

# Grid View Control

- Grid View control is a successor to the popular DataGrid control of ASP.NET version 1.x
- New and enhanced GridView makes it easier to work with DataGrid features

ASP.NET 1.1 introduced the DataGrid control. With this control you could display a whole set of records and also perform sorting, paging and editing functions. But still developers were required to write code to take advantage of the functionalities of DataGrid.

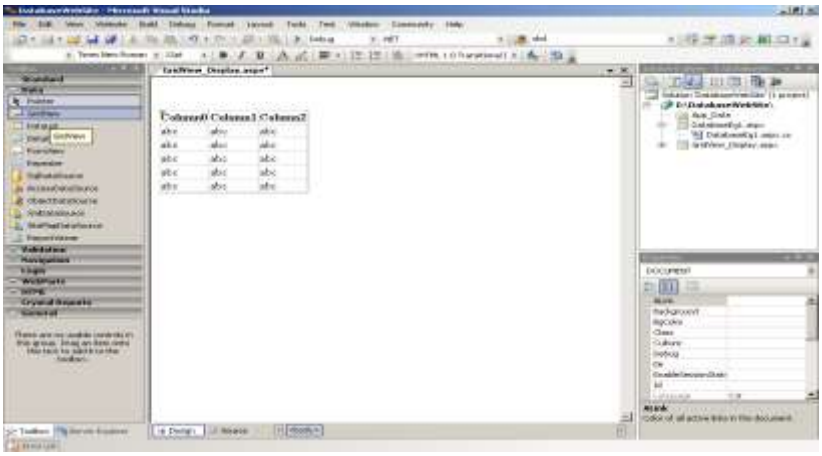
In ASP.NET 2.0 the basic DataGrid has been enhanced creating a new server control called as GridView. We can also say that GridView is a successor to the DataGrid control. This control makes it easier to use all the features provided with DataGrid, without having to write one line of code. But the developers who wish to use coding can still continue to do so.

Let us start with looking and understanding the Gridview control and its features.

6.4: 1:GridView Control

# Using Grid View

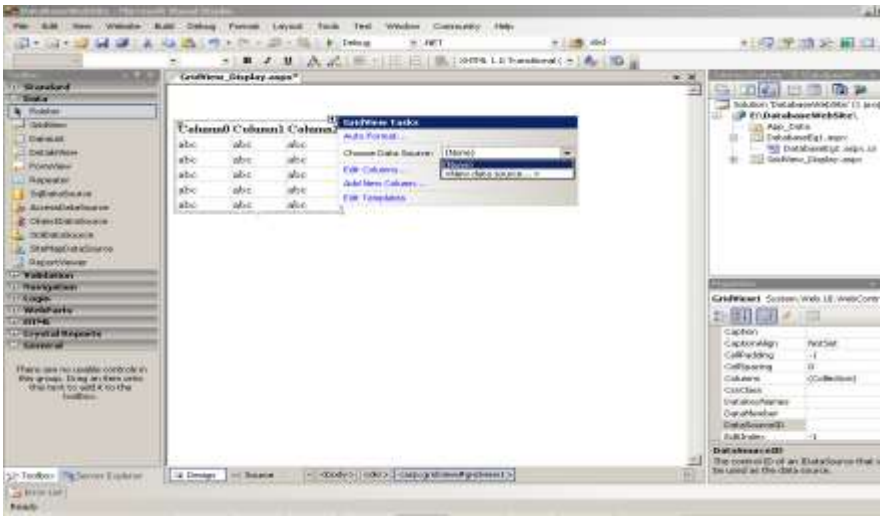
- Use GridView to drag and drop the control on the web page



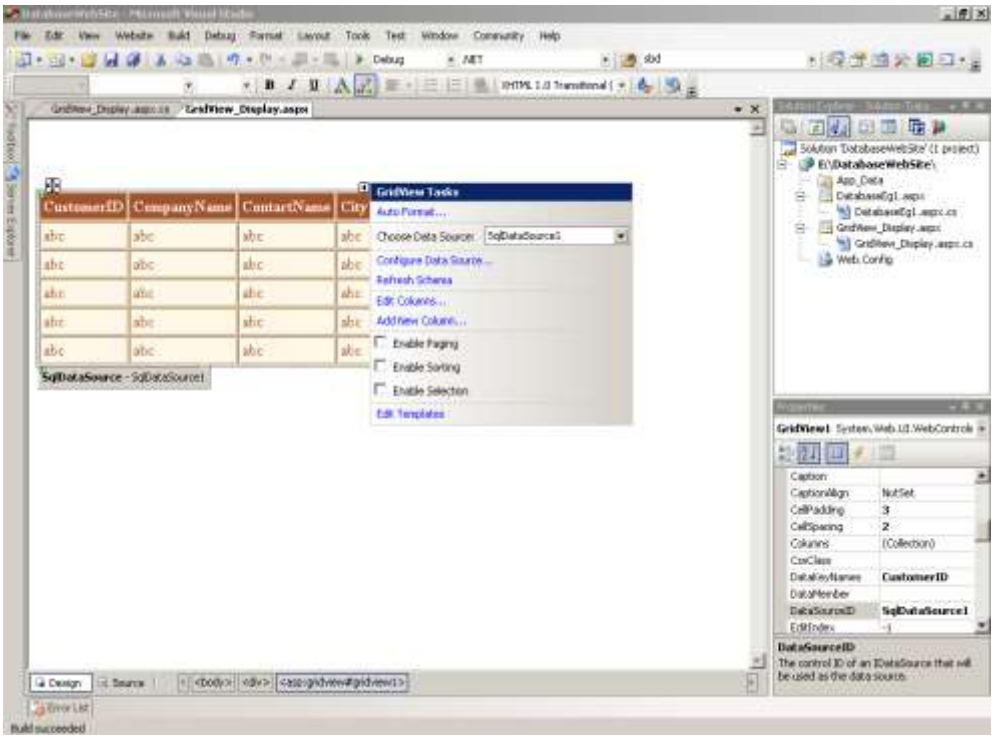
We will now take a step by step look at working with the GridView. This example is for using GridView to display data.
Step 1: Start by dropping the GridView on to your Web Page from the ToolBox-Data tab. (Refer slide)
Step 2: Choose the Data Source for the GridView. We will use the SqlDataSource for this example. Select the NewDataSource from the smart tag, since as of now we do not have any data source. Follow the wizard and provide values for various options as prompted. (Refer next slide)
Choose the datasource as Database. It selects SqlDataSource.
Specify the connection information for the SQL server. The connections string is created with a default name. Optionally you can provide your own name.
The next step is to select the table name and choose the columns which you would like to display in the GridView.
Step 3: click finish. The GridView now reflects the column names chosen for display. You can apply format to the GridView by selecting AutoFormat option.
In case anytime the schema for the GridView does change, you can simply Refresh Schema option the GridView and the changes are applied. Also if you now notice the Smart Tag, you will see other options have been added to it for sorting, paging etc.
Step 4: Execute the application. (We have not written a single line of code).
Once you have executed the program, you can also have a look at the source view of the page to take examine the code that has been generated for the GridView.

6.4.2: Choosing the Data Source

# Choose Data Source



After the datasource has been applied the Smart Tag shows more options as follows:



6.4.3: Sorting and Paging in GridView

## Sorting and Paging in GridView

- In GridView, to enable sorting and paging you simply need to select following options from the smart tag:
  - For Paging: Enable Paging
  - For Sorting: Enable Sorting
- Optionally, you can also select Allow Sorting and Allow Paging from the properties window of the GridView

In an application the capability to sort data is one of the basic requirements. This avoids navigation through a significant number of records at one go.
Also showing a few records on one page makes viewing of the records very convenient. In classic ASP ,the developers would have to write a lot of code to achieve both these functionalities in their web page.
Also with DataGrid control the sorting task had to be achieved through coding.
With the GridView control both of these tasks have become quite simple.

To perform the sorting operation, you can set the AllowSorting property to True . You can also choose the Enable Sorting option from the smart tag. This setting takes care of all the sorting logic fro you internally. As you set the property for sorting the source code for the GridView will reflect it in the code as follows:

```
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False" BackColor="#DEBA84"
        BorderColor="#DEBA84" BorderStyle="None" BorderWidth="1px"
CellPadding="3" CellSpacing="2"
        DataKeyNames="CustomerID" DataSourceID="SqlDataSource1"
AllowPaging="True" AllowSorting="True">
```

The GridView's sorting mechanism has also been enhanced to handle both ascending and descending sorting. You just have to click on the column head and the sorting order can be switched from one mode to another.

- Slide only with Notes pages

The Grid View also improves upon the grid navigation feature i.e. paging.
To enable paging for the GridView you can either set the Enable Paging option from the smart tag of the GridView or set the AllowPaging property to True.

The Enable Paging option in the GridView control defaults to a page size of 10 records and adds the pager to the bottom of the grid.

The source code of the GridView reflects the property change as follows:

```
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False" BackColor="#DEBA84"
        BorderColor="#DEBA84" BorderStyle="None" BorderWidth="1px"
CellPadding="3" CellSpacing="2"
        DataKeyNames="CustomerID" DataSourceID="SqlDataSource1"
AllowPaging="True" AllowSorting="True">
```

6.4.4: Demo on GridView

# Demo: GridView Control

- Understanding GridView control

6.4.5: Customizing Columns in GridView

## Customize Columns in GridView

- To display data in the columns in different formats:
  - You may want to display it as a control
- GridView provides flexibility to display data in a grid
- Edit GridView columns in the following ways:
  - Select 'Edit Columns' in the smart tag.
  - Specify via the code.

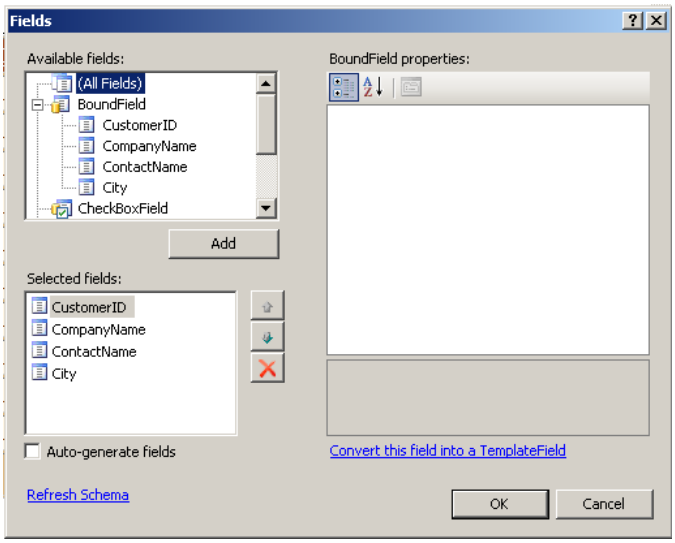
The data displayed in the Grid is by default text content. But sometime we may require the data to be displayed in a different format like a control or hyperlink etc.. The GridView gives you flexibility regarding how you display the data in your grid.

You can edit your GridView columns in two ways. You can either select the Edit Columns option from the GridView smart tag. This link allows you to edit any existing columns in your grid, using the Fields dialog window (Shown on the next slide). Here you can change a columns visibility , header text and other style options. You can also add new columns in the Grid view by selecting the Add New Column in from the smart tag. You can select what field type you want from the drop down list (Shown on the next page). Lets assume that you want a Hyperlink field, choose Hyperlink from the DropDown List. You can then specify the values for the field.

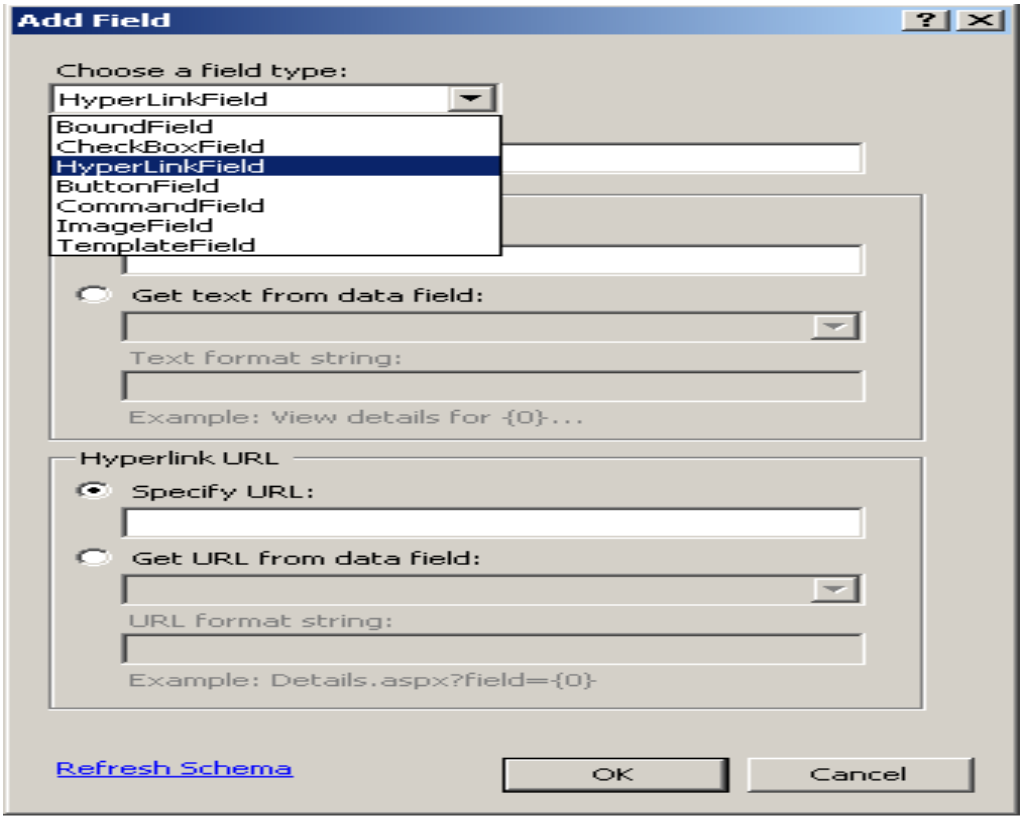
The other way to change the Grid columns is by modifying the Source view.

6.4.6: Selecting Edit Columns options

# Select Edit Columns Options




Select Add New Column.



6.4.7: Adding Field types

# Adding Field Types

- Add New Column option allows you to select from different field types:
  - BoundField
  - CheckBoxField
  - HyperLinkField
  - ButtonField
  - CommandField
  - ImageField
  - TemplateField

The Field types that you can add to your Gridview either through the EditColumns or Add New column option from the Gridview smart tag. You can add:
BoundField: Displays the value of a field in a data source. By default the field type is a Bound Field
CheckBoxField: Displays a check box for each item in the Grid View control. Typically used to display boolena values
HyperlinkField: Displays the field value in the data source as a Hyperlink.
Button Field: Displays a command button for each item in the GridView control.
CommandField: Represents a field that displays command button to perform select, edit, insert or delete operation in a databound control
Image Field: Automatically displays an image when the data in the field represents and image
TemplateField: Displays user-defined content for each item in the Gridview control according to a specified template.

6.4.8: TemplateField Column

# TemplateField Column

- TemplateField column enables you to use templates to completely customize the contents of the column
- Provides following templates:
  - ItemTemplate
  - AltenatingItemTemplate
  - EditItemTemplate
  - InsertItemTemplate
  - HeaderTemplate
  - FooterTemplate

The TemplateField is a very useful column type. It enables you to use templates to completely customize the contents of the column. One common use of a TemplateField is to perform custom logic to massage the output; another common use is to embed other Web controls within a GridView column. The template field provides you with the following templates:
ItemTemplate: Used to display an item in the TemplateField of the data bound control.
AlternatingTemplate: Used to display alternating item of the field.
EditItemTemplate: Template used to display an item in edit mode.
InsertItemTemplate: Template used to display TemplateField item in insert mode.
HeaderTemplate: Used to display the header section of the Template Field.
FooterTemplate: Used to display the footer section of the Template Field

To specify templates for a particular column you can either right-click that column and selecting Edit Templates or select the Edit Templates option from the Smart tag of the GridView.

6.4.9: Editing Grid View Data

# Editing Grid View Data

- GridView Control:
  - Automatically renders UI to modify data through Update and Delete operations
  - Associated data source should be configured to support these capabilities.

- SqlDataSource Control:
  - Supports Update operations when UpdateCommand property is set.
  - Supports Delete operations when DeleteCommand property is set.
  - A valid update or delete command or stored procedure.

One of the most important shortcomings of the DataGrid—and conversely one of the major strengths of the GridView control—is the ability to of handle updates to the data source. When the bound data source supports updates, the GridView can automatically perform data operations, thus providing a real solution instantly.

Adding editing capabilities to the DataGrid was never easy but it was important enough to have these functionalities in an application. The GridView control makes it easy to edit data contained in the grid.

For a GridView control, editing data means in-place editing and record deletion. As mentioned, in-place editing refers to the grid's ability to support changes to the currently displayed records.

Let us consider the same Customer table contents used in the example GridView_Display.

6.4.9: Editing Grid View Data

# Editing Grid View Data (contd..)

- Add editing operation through code
  - Add UpdateCommand attribute to SqlDataSource.

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$
ConnectionStrings:NorthwindConnectionString %>"
SelectCommand="SELECT
[CustomerID],[CompanyName],[ContactName],
[City] FROM [Customers]"  DataSourceMode="DataSet"
UpdateCommand="UPDATE [Customers] SET [CompanyName] =
@CompanyName, [ContactName] = @ContactName, [City] =
@City WHERE [CustomerID] = @original_CustomerID">
```

To add this operations in the GridView it is quite convenient. We will first take a look at adding the capabilities through code.
Let us take a look at it step by step:
Step 1: Add UpdateCommand attribute to the existing SqlDataSource. The Slide show the modification required in your code. The UpdateCommand has place holders for the respective column values.The placeholder values will come from the selected row in the GridView. In order to use the parameters you must define them using the UpdateParameters element of the SqlDataSource control. The UpdateParameters would appear as follows:

```
<UpdateParameters>
        <asp:Parameter Name="CompanyName" Type="String" />
        <asp:Parameter Name="ContactName" Type="String" />
        <asp:Parameter Name="City" Type="String" />
        <asp:ParameterName="CustomerID"Type="String"/>
</UpdateParameters>
```

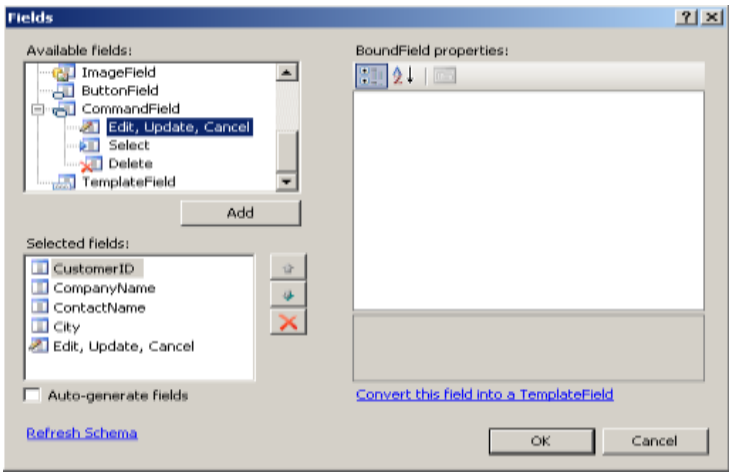
With the UpdateParameters each names parameter is defined using the <asp:parameter> element. It defines the name and the datatype of the parameter. Once the Datasource has been modified, you need to attach this operation to the GridView. From the smart tag, select the Enable Editing check box.

When you execute the application you will get an Edit link in your GridView which allows you to perform editing operation. Note that it does not allow to edit CustomerID.

6.4.9: Editing Grid View Data

# Editing Grid View Data (contd..)

- Another way to allow editing is to add a CommandField column




The second way to enable Editing for GridView data is to add a CommandField column. From the smart tag of the GridView select Edit columns which shows the dialog box seen on the slide. Expand the Command field option and add the Edit, Update and Cancel Field. Ensure that Auto-generate fields option is unchecked. You can control how the command is displayed in the grid using the ButtonType property, which allows you to display the command as a link, a Button or even an image.

The CommandField also has attributes that allow you to control exactly what is shown in the column. You can specify whether the column displays commands like Cancel, Delete, Edit, Insert and Select.

6.4.10: Demo: Editing in Grid View

# Demo

- Understanding Editing in GridView control

6.4.11: Using TemplateField's EditItem Template

# Use TemplateField's EditItem Template

- Create a Bound Column to a TemplateField
  - Select column.
  - Click the "Convert this Field into Template Field" link.

Previously we got introduced to the TemplateFields. One of those templates is the EditItemTemplate which the grid uses to display the TemplateFiled column for a row that has entered into the edit mode.
The default behavior of the Bound control in the Grid allows the user to edit their data only in text boxes. But the TemplateField's EditItemTemplate enables you to customize the way the data editing can be done.
For eg. you may want not want the Primary Key column data to be edited, or you the one of the columns to be presented as DropDown List and so on.
To do this simply change column from Bound Field to a Template field.
Now add an Item Template and an EditItemTemplate.
In EditItemTemplate you can add the control you require for the item to be displayed and provide the proper data binding information.

6.4.12: Demo on Templates in Grid View

# Demo: Templates in Grid View

- Understanding and Working with Templates in Grid View

6.5: Forms View Control

# Form View Control

- Data-bound user interface control that renders a single record at a time from its associated data source
- Provide paging buttons to navigate between records
- Does not use data control fields. Allows you to define the rendering for each item that uses templates

The FormView control is a new control included with the ASP.NET toolbox. The FormView displays a single record at a time from its associated data source. You can provide paging buttons for navigation and it also allows operations for editing, adding and deleting data.

FormView does not use data control fields but allows the user to define the rendering of each item using templates i.e. it does not use data bound controls. This makes the Form View entirely customizable according to user requirements.

Designed to be used mostly as an update and insert interface, the FormView control is unable to validate against a data source schema and does not supply advanced editing features like foreign key field dropdowns.

However, by using templates you can easily provide this functionality.
The FormView control has ItemTemplate, EditItemTemplate, and InsertItemTemplate properties.
The FormView lacks the command row, a toolbar on which available functions are grouped.

6.5.1: Demo on Forms View Control

# Demo: Forms View Control

- Understanding and Working with FormsView Control

6.6. Details View Control

# Details View Control

- Renders a single data item in a table for label-value pairs, similar to the FormsView control
- Often used in combination with a GridView control for master-detail scenarios
- Provides ability to display, edit, insert or delete a single record at a time
- Displays each field of a record on a separate line

DetailsView is a data-bound user interface control that renders a single record at a time from its associated data source, optionally providing paging buttons to navigate between records.
It is similar to the FormView of an Access database, and is typically used for updating and/or inserting new records.
It is often used in a master-details scenario where the selected record of the master control (GridView, for example) determines the DetailsView display record.

The DetailsView control provides an ability to display, edit, insert, or delete a single record at a time from its associated data source. By default, the DetailsView control displays each field of a record on its own line.
The DetailsView control displays only a single data record at a time, even if its data source exposes multiple records.

The DetailsView control relies on the capabilities of the data source control to perform tasks such as updating, inserting, and deleting records.
The DetailsView control does not support sorting.

6:6:1: Demo on Details View Control

# Demo: Details View Control

- Understanding and Working with Details View Control

6.7: Repeater Control

# Repeater Control

- Used where list-like data is to be rendered
- Template-based container control with no basic rendering of its own
- Loops each record in the DataSource and renders the specified template (ItemTemplate) for each record

DataGrid/GridView control is suitable in many scenarios where you wish to display data in a grid like representation for easy understanding. Similarly, if the situation demands for rendering list like data, you can consider using of DataLists and Repeater server controls.

Repeater control is a container control which is template based with no basic rendering of its own. This way, you define layout for the Repeater control by creating different templates based on your needs.
You can create different kinds of lists using Repeater control including Table, Comma-separated list and XML formatted list.

Repeater control is an iterative control ,in the sense it loops each record in the DataSource and renders the specified template (ItemTemplate) for each record in the DataSource collection. In addition, before and after processing the data items, the Repeater emits some markup for the header and the footer of the resulting structure.

6.7.1: Templates in Repeater Control

# Templates in Repeater Control

- Repeater Control supports following templates:
  - ItemTemplate
  - AlternatingItemTemplate
  - HeaderTemplate
  - FooterTemplate
  - SeparatorTemplate

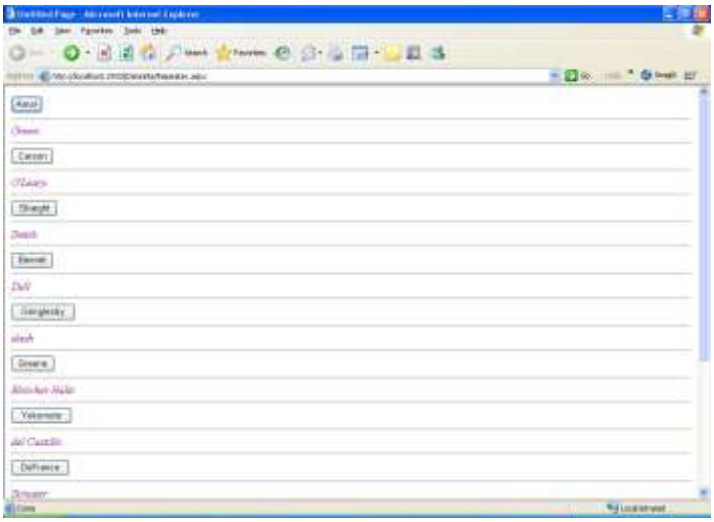
Repeater Control Templates
Repeater controls provides different kinds of templates which helps in determining the layout of control's content. Templates generate markup which determine final layout of content.

Repeater control supports five templates which are as follows:

ItemTemplate: ItemTemplate defines how the each item is rendered from data source collection.
AlternatingItemTemplate: AlternatingItemTemplates define the markup for each Item but for AlternatingItems in DataSource collection like different background color and styles.
HeaderTemplate: HeaderTemplate will emit markup for Header element for DataSource collection
FooterTemplate: FooterTemplate will emit markup for footer element for DataSource collection
SeparatorTemplate: SeparatorTemplate will determine separator element which separates each Item in Item collection. Usually, SeparateTemplate will be <br> html element or <hr> html element.

6.7.2: Example on Repeater Control

# Example: Repeater Control



The above example shows the last name of the authors table in Northwind Database. The authors last names are shown as buttons and the alternating rows are shown as simple data items. The HTML source for Repeater control is as follows:

```
<asp:Repeater ID="rptAuthors" runat="server“
OnItemCommand="rptAuthors_ItemCommand">
        <ItemTemplate>
        <asp:Button runat="server" Text='<%#
DataBinder.Eval(Container.DataItem,"au_lname") %>' />
        </ItemTemplate>
        <AlternatingItemTemplate>
        <font color="purple">
        <i>
         <%# DataBinder.Eval(Container.DataItem,"au_lname") %>
        </i>
        </font>
        </AlternatingItemTemplate>
        <SeparatorTemplate>
        <hr />
        </SeparatorTemplate>
        </asp:Repeater>
        <asp:Repeater ID="Repeater1" runat="server">
</asp:Repeater>
```

6.7.3: Demo on Repeater Control

# Demo: Repeater Control

- Understanding and Working with Repeater Control

6.8: Other Data Bound Controls

# Other Data Bound Controls

- ASP.NET 2.0 provides other DataBound controls such as:
  - DropDownList, ListBox, checkBoxList
- Introduces new RadioButtonList and BulletedList controls

ASP.NET  provided the DropDownList, ListBox, CheckBox list available from the previous versions. Apart from that it has introduced the new RadioButtonList and BulletedList controls.

The RadioButtonList control provides a single-selection checked list. Like other list controls, RadioButtonList has an Items collection with members that correspond to each item in the list. To determine which items are selected, test the Selected property of each item.

The BulletedList control is used to create a list of items formatted with bullets. To specify the individual list items that needs to be appeared in a BulletedList control, place a ListItem object for each entry between the opening and closing tags of the BulletedList control.

6.8.1: Demo on Data Bound Controls

# Demo: Data Bound Controls

- Understanding Data Bound Controls

6.9: SQL server Object Window

# SQL Server Object Window

- VS2012 contains an improved interface to SQL Server that is now very similar to SQL Management Studio interface and provides additional functionality over the integrated tools in previous versions.
- To access this window, select SQL Server object Explorer from the main menu -View

6.10: Chart Control

# Chart Control

- The Chart controls enable you to create ASP.NET pages or Windows Forms applications with simple, intuitive, and visually compelling charts for complex statistical or financial analysis
- Following are the elements of the Chart :-
  - Chart Picture
  - Chart Area
  - Plot Area
  - Title
  - Axis Label
  - Axis Title
  - Legend
  - GridLines
  - Tick marks
  - Label

Chart Picture :The chart picture is the entire image that is produced, and corresponds to the root Chart class.
Series: A related group of data points. Each series has an associated chart type. The number of series that a chart can display and how it displays the series depends on the chart type you specify.
This element corresponds to the Series class.
Chart Area: A rectangular area that is used to draw the series, labels, axes, grid lines, tick marks, and so on. Depending on the chart type, multiple series can be plotted in one chart area. This element corresponds to the ChartArea class.
Plot Area : The rectangular area in a chart area that is used to plot the chart series and grid lines. Labels, tick marks and axes titles are drawn outside of the plotting area, but inside the chart area. The plot area can be set using the InnerPlotPosition property.
Title : A title on the chart picture. You can add any number of titles to a chart picture.
Axis Label :A label on an axis. It is generated automatically if no custom labels are supplied. This element corresponds to the Label class.
Axis Title: The title of an axis, which describes what the axis represents.
Legend: A legend for the chart picture. There can be more than one legend in a chart picture. This element corresponds to an item in the Legend collection.
Grid Lines :The horizontal and vertical grid lines, which usually occur in conjunction with tick marks. This element corresponds to the Grid class.
Tick Marks: The tick marks on the axes, which usually occur in conjunction with grid lines. This element corresponds to the TickMark class.
Label: A label that describes a data point.

6.10.1:Demo

# Demo

- Data Binding a chart to a database

6.11: ASP.Net Strongly Typed Data Controls

## Asp.Net 4.5 Strongly Typed Data Controls

- ASP.NET 4.5 adds the ability to declare the data type of the data that a control is bound to.
- You do this using the new ItemType property. When you set this property, two new typed variables are available in the scope of data-binding expressions:
  - **Item and BindItem.**
- Because the variables are strongly typed, you get the full benefits of the Visual Studio development experience.
- For two-way data-binding expressions, use the BindItem variable

ASP.NET Web Forms introduced the concept of "templates" starting with the very first release. Templates allow you to customize (or override) the markup emitted from server controls, and are typically used with data-binding expressions.
When using data-binding within a template today, you use late-bound expressions to bind to the data. For example, below we are using the Eval() helper method to data-bind the "FirstName" and "LastName" properties from a list of objects data-bound to a repeater control:

```
<ul>
<asp:Repeater runat="server" ID="customers">
<ItemTemplate>
<li>
First Name: <%# Eval("FirstName") %><br />
Last Name: <%# Eval("LastName") %><br />
</li>
</ItemTemplate>
</asp:Repeater>
</ul>When performing 2-way data-binding today, you use the Bind() helper method like so:
<asp:FormView ID="editCustomer" runat="server" >
<EditItemTemplate>
<div>
First Name:
<asp:TextBox ID="firstName" runat="server" Text='<%# Bind("FirstName") %>' />
</div>
<div>
Last Name:
<asp:TextBox ID="lastName" runat="server" Text='<%# Bind("LastName") %>' />
</div>
<asp:Button runat="server" CommandName="Update" />
</EditItemTemplate>
</asp:FormView>One downside with the above approaches is that the calls to Eval() and Bind() are late-bound - meaning you pass strings to represent the property names. This means you don't get Intellisense for the member names, support for code navigation (like Go To Definition), nor compile-time checking support.
```

6.11: ASP.Net Strongly Typed Data controls ….contd

# Strongly Typed Data controls (contd..)

- At run time, these calls use reflection to read the value of the specified member and then display the result in the markup. This approach makes it easy to data bind against arbitrary, unshaped data.
- However, data-binding expressions like this don't support features like IntelliSense for member names, navigation (like Go To Definition), or compile-time checking for these names.

6.12: Two-way data binding

## For two-way data binding, you use Bind:

```
<asp:FormView runat="server" ID="editCustomer">
 <EditItemTemplate>
   <div>
    <asp:Label runat="server" AssociatedControlID="firstName"> First Name:</asp:Label>
    <asp:TextBox ID="firstName" runat="server" Text= '<%#Bind("FirstName") %>' />
   </div>
   <div>
     <asp:Label runat="server" AssociatedControlID="lastName"> First Name:</asp:Label>
     <asp:TextBox ID="lastName" runat="server" Text= '<%# Bind("LastName") %>' />
   </div>
   <asp:Button runat="server" CommandName="Update"/>
 </EditItemTemplate>
</asp:FormView>
```

Previous Versions- One Way Binding

```
<ul> <asp:Repeater runat="server" ID="customers">
 <ItemTemplate>
 <li> First Name: <%# Eval("FirstName")%>
<br /> Last Name: <%# Eval("LastName")%><br /> </li> </ItemTemplate>
</asp:Repeater> </ul>
```

6.13: Model Binding

# Model Binding

- Model binding extends data binding in ASP.NET Web Forms controls to work with code-focused data access.
- It incorporates concepts from the ObjectDataSource control and from model binding in ASP.NET MVC.
- To configure a data control with model binding for selecting data, you set the control's SelectMethod property to the name of a method in the page's code.
- The data control calls the method at the appropriate time in the page life cycle and automatically binds the returned data.
- No need to explicitly call the DataBind method.

If you have used ASP.NET MVC, you will notice the model binding support is similar. Indeed, these features were taken from ASP.NET MVC and moved into the System.Web assembly to be able to use them on Web Forms as well.

6.13: Model Binding contd..

# Model Binding Example

```
<asp:GridView ID="categoriesGrid" runat="server"
ItemType="WebApplication1.Model.Category" SelectMethod="GetCategories"
AutoGenerateColumns="false">
  <Columns>
    <asp:BoundField DataField="CategoryID" HeaderText="ID" />
    <asp:BoundField DataField="CategoryName" HeaderText="Name" />
    <asp:BoundField DataField="Description" HeaderText="Description" />
    <asp:TemplateField HeaderText="# of Products">
      <ItemTemplate>
        <%# Item.Products.Count %>
      </ItemTemplate>
    </asp:TemplateField>
  </Columns>
</asp:GridView>
```

You create the GetCategories method in the page's code. For a simple select operation, the method needs no parameters and should return an IEnumerable or IQueryable object.

6.13:Model Binding Contd..

# Model Binding Example

- The example shows the code for a GetCategories method and uses the Entity Framework Code First model with the Northwind sample database.
- The code makes sure that the query returns details of the related products for each category by the Include method

```
public IQueryable<Category> GetCategories()
{
   var db = new Northwind();
   return db.Categories.Include(c => c.Products);
}
```

6.13: Model Binding Contd..

# Model Binding Example

- When the page runs, the GridView control calls the GetCategories method automatically and renders the returned data using the configured fields:

| ID | Name | Description | # of Products |
|---|---|---|---|
| 1 | Beverages | Soft drinks, coffees, teas, beers, and ales | 13 |
| 2 | Condiments | Sweet and savory sauces, relishes, spreads, and seasonings | 12 |
| 3 | Confections | Desserts, candies, and sweet breads | 13 |
| 4 | Dairy Products | Cheeses | 10 |
| 5 | Grains/Cereals | Breads, crackers, pasta, and cereal | 7 |
| 6 | Meat/Poultry | Prepared meats | 6 |
| 7 | Produce | Dried fruit and bean curd | 5 |
| 8 | Seafood | Seaweed and fish | 12 |

When the page runs, the GridView control calls the GetCategories method automatically and renders the returned data using the configured fields:

- Slide only with Notes Page

ASP.NET provides the ability to enable strongly-typed data templates. Specifically, we've added the ability to declare what type of data a control is going to be bound to, by way of a new "ModelType" property on data controls. Setting this property will cause two new typed variables to be generated in the scope of the data-bound template expressions: Item and BindItem.
For example, set the ModelType on an <asp:Repeater> control to be a "Customer" object. Once we do this we can switch from using Eval("FirstName") to instead use Item.FirstName to reference the property.
We get full Visual Studio code intellisense when we do so:
For 2-way data-binding expressions, we can also now use the BindItem variable and get the same strongly-typed benefits:

```
<asp:FormView ID="editCustomer" runat="server">
  <EditItemTemplate>
   <div>
      First Name:
    <asp:TextBox ID="firstName" Text='<%# BindItem.FirstName %>'
runat="server" />
  </div>
  <div>
   Last Name:
    <asp:TextBox ID="lastName" Text='<%# BindItem.LastName %>'
runat="server" />
   </div>
   <asp:Button runat="server" CommandName="Update" />
 </EditItemTemplate>
</asp:FormView>
```

6.14:SQL credential class

# SQL Credential Class

- It is designed to provide the SQL Server Authentication Credentials with a SqlConnection Instance .
- The credentials can now be specified outside of the connecting string .

```
using (SqlConnection con = new SqlConnection(ConnectionString))
{
    SecureString password = txtPassword.SecurePassword;
    password.MakeReadOnly();
    SqlCredential credential = new SqlCredential(txtUserName.Text,
    password);
    con.Credential = credential;
    con.Open();
}
```

The SqlConnection object supports the Credential property in .NET Framework 4.5.

You can use this property to specify the SQL database authentication credentials.

6:16:Aynchronous Connection

# Asynchronous Connection

- ADO.NET extensively supports asynchronous operations.
- You can asynchronously open a connection to a database,with the new asynchronous features of .NET Framework which also being extended to ADO.NET.
- The SqlConnection object has the OpenAsync method to open a connection asynchronously.

```
readonly string ConnectionString = "Data Source=<server name>;Initial
Catalog=Store;Integrated Security=True";
protected async void ExecuteCommandAsync()
{
    using (SqlConnection con = new SqlConnection(ConnectionString))
    {
    await con.OpenAsync();
    }
}
```

6.16:Provider Factories

# Provider Factories

- The provider factory pattern allows a layer of abstraction when accessing data from different providers
- To create a single API implementation of multiple native providers.
- The System.Data.Common namespace exposes generic provider class like the DbConnection and DbCommand, which are implemented by each native client provider, such as SqlClient.
- The DbProviderFactory class in the System.Data.Common namespace also provides the methods for generic implementation of native providers.

Example:

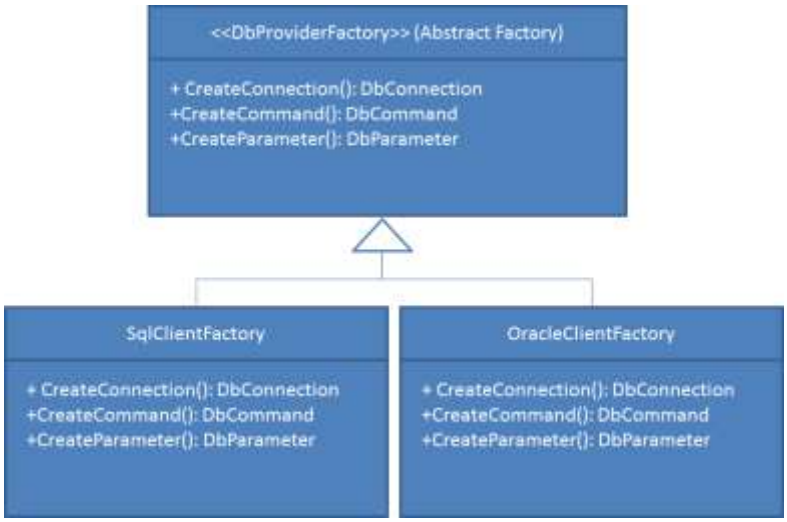
```
DbProviderFactory provider =
DbProviderFactories.GetFactory("System.Data.SqlClient");
using (DbConnection con = provider.CreateConnection())
{
        con.ConnectionString = ConnectionString;
        con.Open();
        DbParameter param = provider.CreateParameter();
        param.ParameterName = "@StoreName";
        param.DbType = DbType.String;

        param.Direction = ParameterDirection.Input;
        param.Value = "Gadgets";
        DbCommand command = provider.CreateCommand();
        command.Connection = con;
        command.CommandText = GetReaderProcedure;
        command.CommandType = CommandType.StoredProcedure;
        command.Parameters.Add(param);
        var result = command.ExecuteReader();
}
```

The methods that allow creating concrete provider types are the CreateConnection and CreateCommand methods
of the DbProviderFactory class. Since SqlClient is the provider used in the example, it would naturally return the
SqlConnection and SqlCommand instances.

6.17:DB Provider Factory

# DB Provider Factory



6.18:Encrypting the Web Configuration

# Encrypting Web Configuration

- ASP.NET allows you to encrypt configuration sections in the Web.config file to strengthen further the security of your application.
- You can use the ASPNET_REGIIS command with the –pe switch to encrypt a configuration section.
- The following code illustrates encrypting the connectionStrings configuration section:

```
aspnet_regiis –pe "connectionStrings" –site "Content Store Site" –app
"/ContentStore" –prov "RsaProtectedConfigurationProvider"
```

You can specify the provider you want to use to encrypt the configuration section using the –prov switch.
It is also possible to write your own providers by inheriting from the custom ones available with the Framework.
Postencryption, your connectionString configuration section in the Web.config file will look like this:

```
<connectionStrings
configProtectionProvider="RsaProtectedConfigurationProvider">
<EncryptedData>
<CipherData>
<CipherValue>HoAwE/Cl+sBAAAAH2... </CipherValue>
</CipherData>
</EncryptedData>
</connectionStrings>
```

You can use the –pd switch to decrypt your configuration section.

# Summary

- In this lesson you have learnt:
  - Using the DataSource controls provided in ASP.NET
  - Using GridView control, FormView Control, DetailsView Control and Repeater Control.
  - Using Data Bound Controls like RadioButtonList, CheckBoxList, ListBox and DropDownList.
  - Model Binding and Strongly Typed Data Controls

# Review Questions

- Question 1: Which templates does GridView have (select two):
  - RowTemplate
  - ItemTemplate
  - ModifyItemTemplate
  - EditItemTemplate
- Question 2: Which of the them is not a Data Control in ASP.NET?
  - GridView
  - Repeater
  - DataList
  - DataStack

# Review Questions

- Question 3: Which of the following is not a Template Field Column?
  - ItemTemplate
  - AltenatingItemTemplate
  - FooterTemplate
  - PageTemplate