





ASP.NET

CHAPTER 3 : DATABASE ACCESS



AUTHORS



Dr. Ashoksinh V. Solanki
M.C.A., Ph.D.

COLLEGE OF APPLIED SCIENCE
AND PROFESSIONAL STUDIES, CHIKHLI



Mr. Yatin Solanki
M.C.A., Ph.D.(Pursuing)

DOLAT USHA INSTITUTE OF
APPLIED SCIENCES, VALSAD



Mr. Chetan Parmar
M.C.A., NET, SET

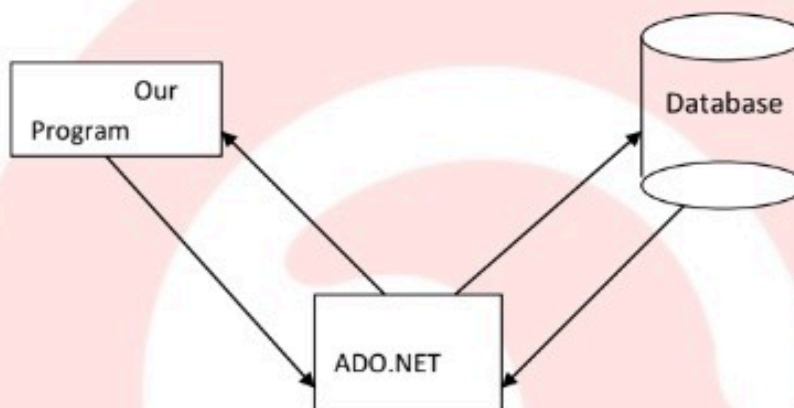
NARMADA COLLEGE OF SCIENCE
& COMMERCE, ZADESHWAR

Website : www.jump2learn.com | Email : info@jump2learn.com | Instagram : www.instagram.com/jump2learn

Facebook : www.facebook.com/Jump2Learn | Whatsapp : +91-909-999-0960 | YouTube : Jump2Learn

❖ ADO.NET

- ADO.NET is a part of the Microsoft .Net Framework.
- ADO.NET consists of a set of classes used to handle data access
- ADO.NET is entirely based on XML
- It's stands for ActiveX® Data Objects.
- ADO.NET has the ability to separate data access mechanisms, data manipulation mechanisms and data connectivity mechanisms.
- ADO.NET is a set of classes that allow applications to read and write information in databases.



- ADO.NET can be used by any .NET language. It's a concept. It's not a programming language.
- We need to add **System.Data** namespace for work with ADO.NET.
- ADO.NET introduces the concept of **disconnected architecture**.

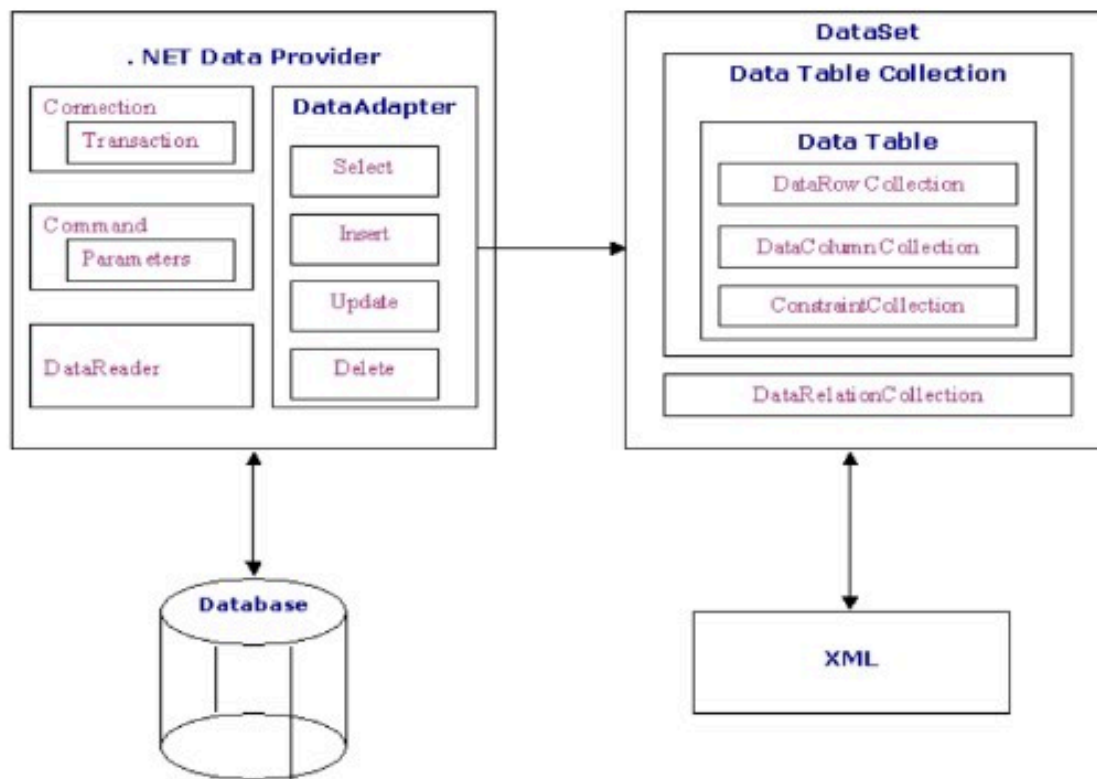
❖ Disconnected environment

- In traditional data access components, we make a connection to the database server and then interact with it. The application stays connected to the database server even when it is not using database service. It is waste the valuable and expensive database resource.
- ADO.NET solves this problem by managing a local buffer of persistent data called **DataSet**. Our application automatically connects to the database server when it needs to pass some query and then disconnects immediately after getting the result back and storing it in dataset.

❖ There are two major components of ADO.NET

- **DataSet:-** It represents either an entire database or a subset of database. It can contain tables and relationships between those tables.
- **Data Provider :-** It is a collection of components like Connection, Command, DataReader, DataAdapter objects and handles communication with a physical data store and the dataset.

❖ ADO.NET Architecture

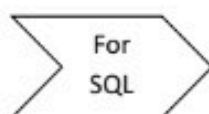


ADO .NET Data Architecture

❖ Data Provider

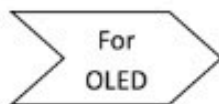
- The data provider is responsible for providing and maintaining the connection to the database.
- It is a set of classes that can be used for communicating with database and manipulating data.
- The data provider connects to the data source on behalf of ADO.NET.
- The data source can be Microsoft SQL server or Oracle database and OLEDB data provider.
- The data provider components are specific to data source.

- **The following list display various data providers:**



Imports System.Data.SqlClient namespace.

It provides data access for Microsoft SQL Server.



Imports System.Data.OleDb namespace.

We can use OLE DB for connect Microsoft Access.



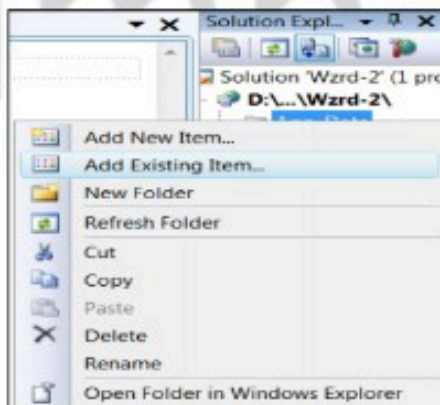
Imports System.Data.OracleClient namespace.

It provides data access for Oracle.

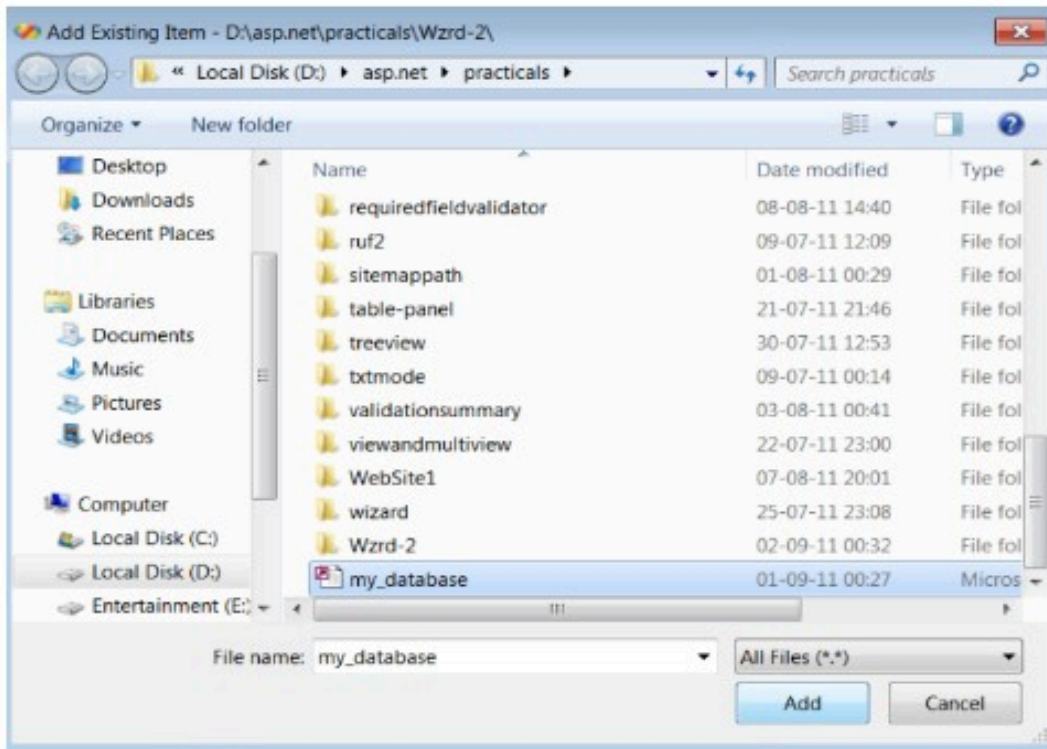
- **Following are the four core objects that make up a .NET Framework data provider.**
 1. **Connection** :- It establishes a connection to the data source. In OleDb the connection can establish using OleDbConnection object.
 2. **Command** :- Fires SQL commands or perform some action on the data source such as insert, update and delete. In OleDb the command can fires using OleDbcommand object.
 3. **DataAdapter** :- It is a bridge between Data source and Dataset object for transferring data. In OleDb the dataadapter can create using OleDbDataAdapter object.
 4. **DataReader** :- It reads records in a read-only, forward-only mode. In OleDb the datareader can create using OleDbDataReader object.

❖ Wizard through Connectivity

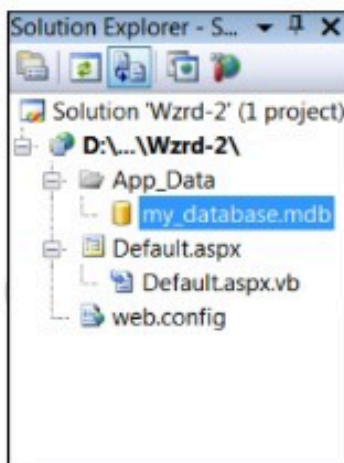
- Create a database in Microsoft access, add table and also insert records in it.
- In solution explorer window, right click on **App_Data** and then click on Add Existing Item...



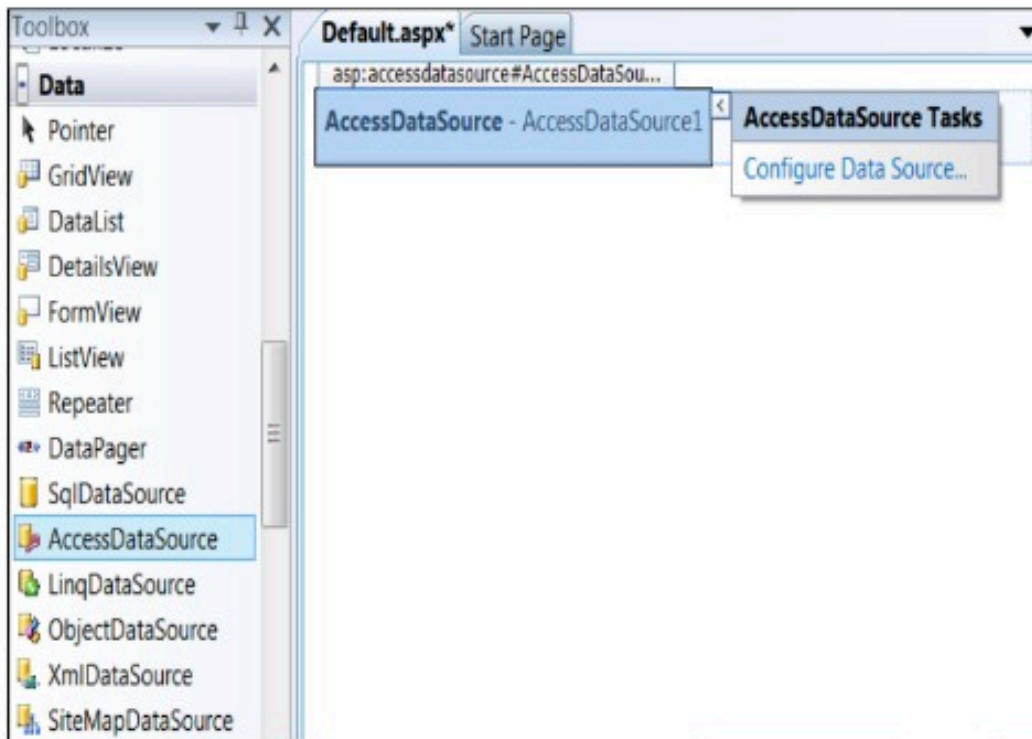
- Now, Add your database from the particular location.



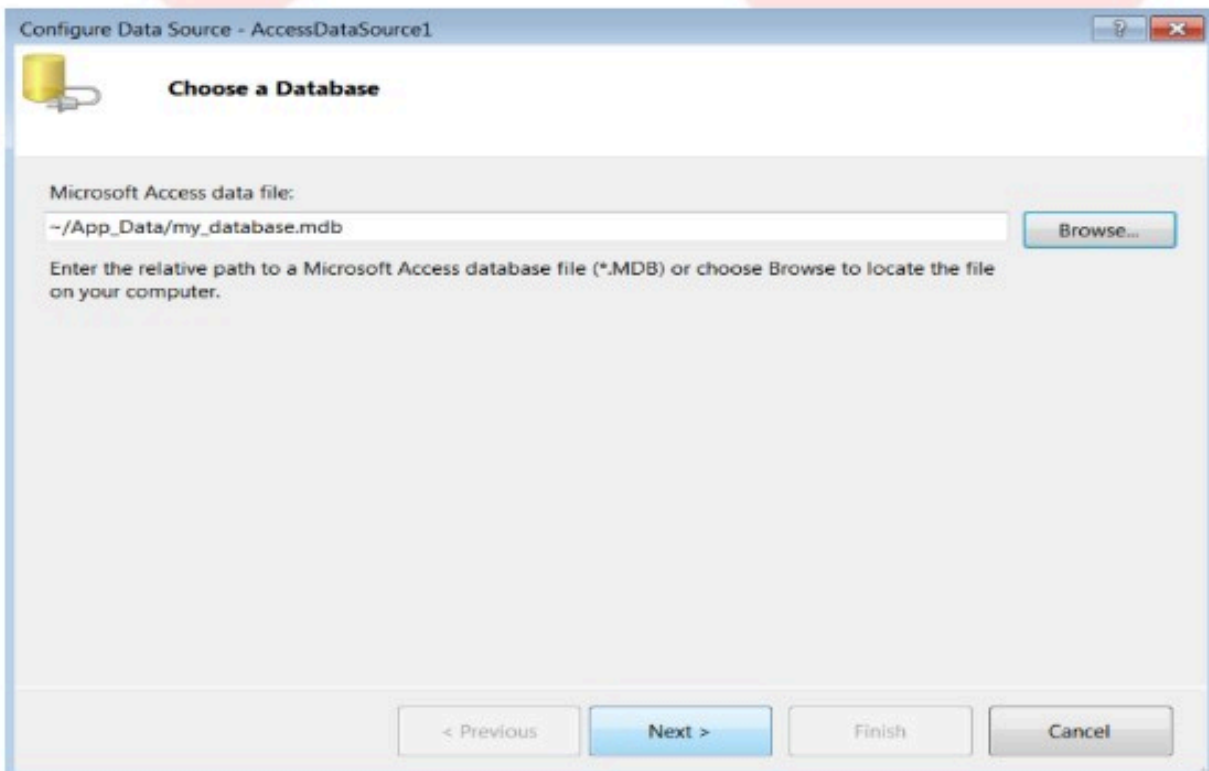
- Your Database automatically stored in APP_DATA folder.



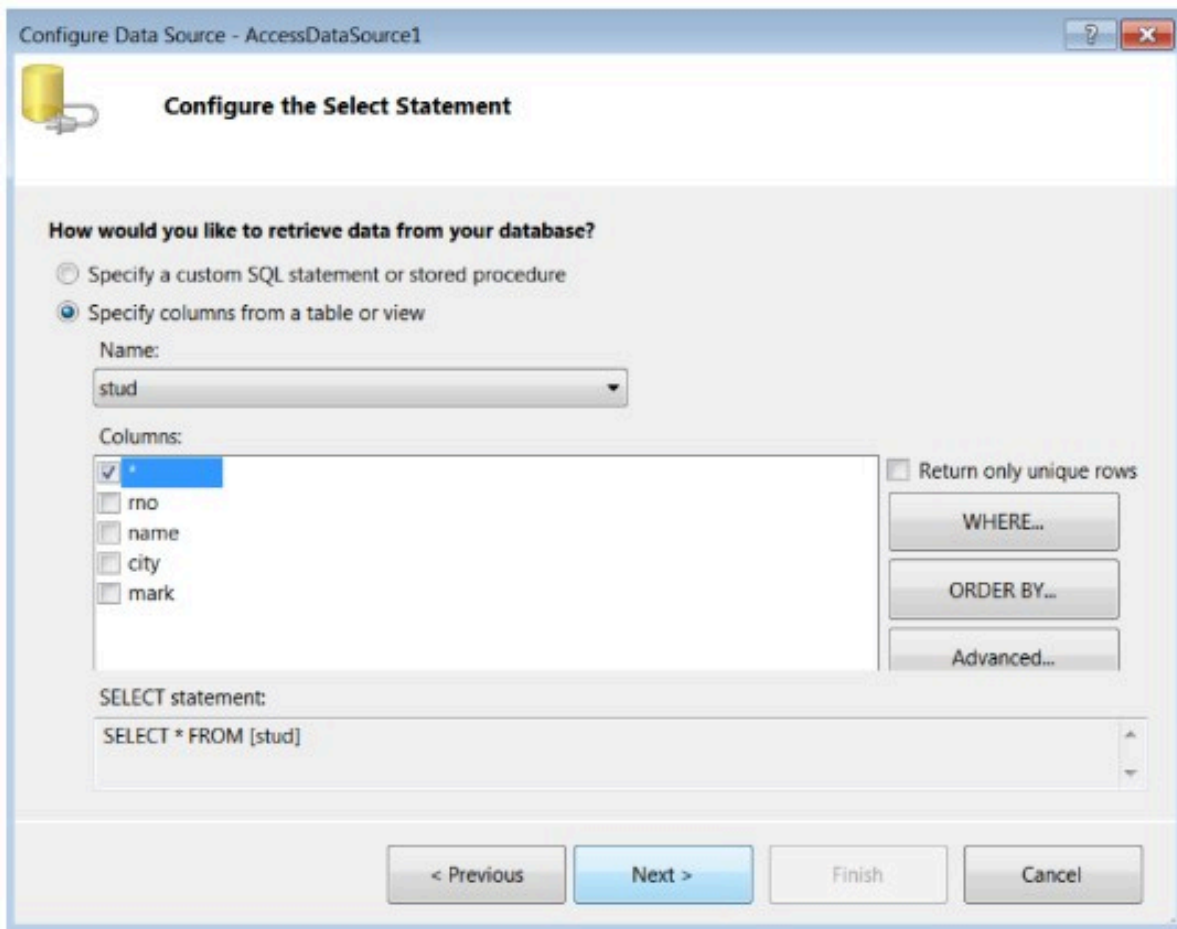
- Now, Add AccessDataSource in our WebPage and then configure your Data Source.



- Click on Browse button and choose your Database and then click on Next.



- Now, configure the Select Statement. We have to select our specific table and the columns to be retrieve from the table. Here, we checked for all columns so all columns will be retrieve from the table.
- Click on Next.



The screenshot shows the 'Configure Data Source - AccessDataSource1' wizard window. The title bar reads 'Configure Data Source - AccessDataSource1'. The main heading is 'Configure the Select Statement'. Below this, the question 'How would you like to retrieve data from your database?' is followed by two radio button options: 'Specify a custom SQL statement or stored procedure' (unselected) and 'Specify columns from a table or view' (selected). Under the selected option, there is a 'Name:' dropdown menu with 'stud' selected. Below that is a 'Columns:' list box containing a checked '*' (all columns) and unchecked 'rno', 'name', 'city', and 'mark'. To the right of the columns list is a checkbox for 'Return only unique rows' (unchecked) and three buttons: 'WHERE...', 'ORDER BY...', and 'Advanced...'. At the bottom, the 'SELECT statement:' text box contains the text 'SELECT * FROM [stud]'. At the very bottom of the window are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

- Click on **Test Query** Button so that the data returned by the Data Source will be displayed.
- To complete this wizard click on Finish.

Configure Data Source - AccessDataSource1

Test Query

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

mo	name	city	mark
101	raj	valsad	67
102	harsh	bombay	87
103	manish	valsad	99
104	nimesh	vapi	66
105	chetan	surat	65

Test Query

SELECT statement:
SELECT * FROM [stud]

< Previous Next > Finish Cancel

- Now, Add a GridView control on our WebPage and Bind it with Database by selecting **AccessDataSource1** from Choose Data Source.

Default.aspx* Start Page

AccessDataSource - AccessDataSource1

asp:gridview#GridView1

Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

GridView Tasks

Auto Format...

Choose Data Source: (None)

Edit Columns... (None)

Add New Column AccessDataSource1

Edit Templates <New data source...>

- Now, you can Run your application.

❖ Connectivity With Coding

- For any of the ADO.NET program with CODING, first we need to Imports namespace called System.Data and then add SQLClient for SQL server and add OLEDB for Access before any class declaration.
- We will use the System.Data.OleDb.

Ex:-

Imports System.Data.OleDb

Partial Class _Default

Inherits System.Web.UI.Page

- The System.Data contains basic enumerations and classes.

❖ Connection object :-

The Connection object creates the connection to the database.

- The Connection object represents the physical connection to a data source.
- The Connection object contains all of the information required to open a connection to the database.
- Basic Steps :-
 1. Create Database.
 2. Create Connection object
 3. Set the Provider.
 4. Specify the data source.
- Connection object have **ConnectionString** property, It has a group of semi-colon-separated attributes.
- Depends on the parameter specified in the **Connection String**, ADO.NET Connection object connect to the specified Database.

Ex:-

Imports System.Data.OleDb

Partial Class _Default

Inherits System.Web.UI.Page



```
Protected Sub Page_Load(ByVal sender As.....
```

```
Dim con As New OleDbConnection("Provider=Microsoft.jet.OLEDB.4.0;  
Data Source=D:\asp.net\practicals\Connectivity\App_Data\my_database.mdb")
```

- **Methods of Connection object.**

- Open :- It opens the connection to our database.
- Close :- It closes the database connection.
- Dispose :- It releases the resources on the connection object.
- State :- Returns state of connection object. Returns 0, if not connected and 1, if connected.

- ❖ **Command object :-**

It is depends on Connection Object.

- Command objects are used to execute commands to a database across a data connection.
- The Command object in ADO.NET executes SQL statements against the data source specified in the Connection object.
- Command object can support SQL statements that return single value; one or more sets of rows or no value at all.
- The Command object has a property called **CommandText**, which contains a String (Query) value that represents the command that will be executed in the Data Source.

- **Properties of Command object**

1. **Connection Property :-**

This property contains data about the connection string. It must be set on the OleDbCommand object before it is executed.

- For the Command to execute properly, the connection must be open at the time of execution.
- We can initialize it as given below:

```
Dim cmd As New OleDbCommand("select * from stud", con)
```

OR

```
Dim str As String
```

```
str = "select * from stud"
```



Dim cmd As New OleDbCommand(str, con)

2. CommandText Property :-

This property specifies the SQL string or stored procedure. (It's specify SQL statement) **Ex :-**

Imports System.Data.OleDb

Partial Class _Default

Inherits System.Web.UI.Page

Protected Sub Page_Load(**ByVal** sender **As** Object,**ByVal** e **As** System.EventArgs) **Handles** Me.Load

```
Dim con As New OleDbConnection ("Provider=
Microsoft.jet.OLEDB.4.0; Data Source= D:\asp.net\practicals\
Connectivity\App_Data\my_database.mdb")
```

Dim cmd **As New** OleDbCommand

cmd.Connection = con

cmd.CommandType = Data.CommandType.Text

cmd.CommandText = "select * from stud"

3. Parameters Collection Property :-

If we want to update values in the student table, but w don't know the values at design time, we make use of placeholders.

- These are variables prefixed with "@" symbol.
- Our code will look like this....

```
cmd.CommandText = "insert into stud(rno,name,city,mark)
values(@rno,@name,@city,@mark)"
```

- Next, we have to create parameters that will be used to insert values.
- For this, we need to add parameters to the parameters collection of the OleDbCommand object. This is done so that the values added through the parameters collection & placeholders get included in the SQL statement.
- To add parameters to the OleDbCommand object, we write the following code:



Ex:-

```
Imports System.Data.OleDb
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub btnAddRec_Click(ByVal sender As Object, ByVal e As System.EventArgs)
        Handles btnAddRec.Click

            Dim rno, mark As Integer
            Dim name, city As String

            Dim con As New OleDbConnection (" Provider= Microsoft.jet.OLEDB.4.0; Data Source=
D:\asp.net\practicals\Connectivity\ App_Data\my_database.mdb")

            Dim cmd As New OleDbCommand

            cmd.Connection = con
            cmd.CommandType = Data.CommandType.Text
            cmd.CommandText = "insert into stud(rno,name,city,mark)
values(@rno,@name,@city,@mark)"

            rno = 101
            name = "raj"
            city = "valsad"
            mark = 55
```

12

'Add parameters

```
cmd.Parameters.Add("@rno", OleDbType.Integer)
cmd.Parameters.Add("@name", OleDbType.Char)
cmd.Parameters.Add("@city", OleDbType.Char)
cmd.Parameters.Add("@mark", OleDbType.Integer)
```

'Initialize parameters

```
cmd.Parameters.Item("@rno").Value = rno
cmd.Parameters.Item("@name").Value = name
```

```
cmd.Parameters.Item("@city").Value = city

cmd.Parameters.Item("@mark").Value = mark

con.Open()

cmd.ExecuteNonQuery()

con.Close()
```

End Sub

End Class

- **Methods of Command object**

- Command object has three important methods for execute queries.
- Connection should be open for below method :
 1. ExecuteScalar
 2. ExecuteNonQuery
 3. ExecuteReader

1. ExecuteScalar:-

- It's use with aggregate functions such as Max, Min, Avg, Sum, Count...
- It returns only one value at a time.
- It fetches the data from the database.

Example:-

First put two textboxes and two buttons in your web page.

First button will count the total number of records available in a **stud** table and second button will find out the maximum **mark**.

Now, write the following code :

Imports System.Data.OleDb

Partial Class methds_cmdobj

Inherits System.Web.UI.Page



```
Dim con As New OleDbConnection("Provider=Microsoft.jet.OLEDB.4.0;Data Source=
D:\asp.net\practicals\connectivity\App_Data\my_database.mdb")

Protected Sub Button2_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Button2.Click

Dim cmd As New OleDbCommand("select max(mark) from stud",con)

con.Open()

TextBox2.Text = cmd.ExecuteScalar()

con.Close()

End Sub

Protected Sub Button1_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Button1.Click

Dim cmd As New OleDbCommand("select count(rno) from stud",con)

con.Open()

TextBox1.Text = cmd.ExecuteScalar()

con.Close()

End Sub

End Class
```

2. ExecuteNonQuery:-

- It's used for fire Insert, Update, Delete queries to the Database.
- It requires open connection.
- It returns number of rows affected. It means if we delete 5 records then it returns 5.

○ Example of ExecuteNonQuery for Delete

Imports System.Data.OleDb

Partial Class methds_cmdobj

Inherits System.Web.UI.Page



```
Dim con As New OleDbConnection("Provider=Microsoft.jet.OLEDB.4.0;Data Source=
D:\asp.net\practicals\connectivity\App_Data\my_database.mdb")
```

```
Protected Sub btnDel_Click(ByVal sender As Object, ByVal e As
System.EventArgs)Handles btnDel.Click
```

```
Dim ans As Integer
```

```
Dim cmd As New OleDbCommand("delete from stud where rno=" & TextBox1.Text, con)
```

```
con.Open()
```

```
ans = cmd.ExecuteNonQuery()
```

```
con.Close()
```

```
MsgBox("Total No. Of Records Deleted :" & ans)
```

```
End Sub
```

```
End Class
```

○ Example of ExecuteNonQuery for Insert

```
Imports System.Data.OleDb
```

```
Partial Class methds_cmdobj
```

```
Inherits System.Web.UI.Page
```

```
Dim con As New OleDbConnection("Provider=Microsoft.jet.OLEDB.4.0;Data Source=
D:\asp.net\practicals\connectivity\App_Data\my_database.mdb")
```

```
Protected Sub btnAdd_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
btnAdd.Click
```

```
Dim str As String
```

```
Dim ans As Integer
```

```
str = "insert into stud values(" & TextBox4.Text & "," & TextBox5.Text & "," & TextBox6.Text
& "," & TextBox7.Text & ")"
```

```
Dim cmd As New OleDbCommand(str, con)
```

```
con.Open()
```




```
ans = cmd.ExecuteNonQuery()

con.Close()

MsgBox("Record Inserted Successfully :-" & ans)

End Sub

End Class
```

○ **Example of ExecuteNonQuery for Update**

```
Imports System.Data.OleDb

Partial Class methds_cmdobj

Inherits System.Web.UI.Page

Dim con As New OleDbConnection("Provider=Microsoft.jet.OLEDB.4.0;Data Source=
D:\asp.net\practicals\connectivity\App_Data\my_database.mdb")

Protected Sub Button5_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles
Button5.Click

Dim str As String

Dim ans As Integer

str = "update stud set name=" & TextBox5.Text & ", city= " & TextBox6.Text & ", mark= " &
TextBox7.Text & " where rno= " & TextBox4.Text

Dim cmd As New OleDbCommand(str, con)

con.Open()

ans = cmd.ExecuteNonQuery()

con.Close()

MsgBox(ans & ": Record Successfully Updated")

End Sub

End Class
```



3. ExecuteReader:-

- When we execute the command object using ExecuteReader method that time it returns DataReader
- DataReader is used to retrieve a read-only, forward-only stream of data from a Database.
- We can not perform any insert, update and delete operations in the DataReader.
- DataReader is opposite of DataSet.
- We can not move previous in the DataReader.
- DataReader requires open connection.
- DataReader can hold one table at a time.
- DataReader can increase application performance by retrieving data as soon as it is available and (by default) storing only one row at a time in memory, reducing system overhead.
- It has item collection.
- For example :- select rno, name, city from student then rno is Item(0), name is Item(1) and city is Item(2).
- DataReader have read method to read next records.
- DataReader is set by command object's ExecuteReader method.
- Example:- display data using DataReader in two dropdown list. (select RollNo and display name according to it).

Imports System.Data.OleDb

Partial Class methds_cmdobj

Inherits System.Web.UI.Page

Dim con As New OleDbConnection("Provider=Microsoft.jet.OLEDB.4.0;Data Source=D:\asp.net\practicals\connectivity\App_Data\my_database.mdb")

Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

If Not Page.IsPostBack Then

Dim str As String

str = "select rno,name from stud"

Dim cmd As New OleDbCommand(str, con)

con.Open()

Dim dr As OleDbDataReader



```
dr = cmd.ExecuteReader

While dr.Read

DropDownList1.Items.Add(dr.Item(0))

DropDownList2.Items.Add(dr.Item(1))

End While

con.Close()

End If

End Sub

Protected Sub DropDownList1_SelectedIndexChanged(ByVal sender As Object, ByVal e As
System.EventArgs)Handles DropDownList1.SelectedIndexChanged

    DropDownList2.SelectedIndex = DropDownList1.SelectedIndex

End Sub

Protected Sub DropDownList2_SelectedIndexChanged(ByVal sender As Object, ByVal e As
System.EventArgs) Handles DropDownList2.SelectedIndexChanged

    DropDownList1.SelectedIndex = DropDownList2.SelectedIndex

End Sub

End Class
```

❖ Differences Between DataReader and DataSet:

DataReader	DataSet
It is directly connected to database system.	It is generally used to disconnected architecture of Ado.Net
It can hold one table at a time.	It can hold multiple tables at a time. It represents a complete set of data including related tables, constraints and relationships among the tables.
Use multiple user updated data every time.	If multiple users are using the database and the database needs to be updated every time, you must not use the DataSet.
It provides Read only data. We cannot update records.	We can make changes in the DataSet.
No local storage is required.	It reads data from database and stores in local system.
Improve application performance.	DataSet requires lot of memory space compare to DataReader.
It holds one row at a time is stored in memory.	It holds multiple records at a time.
No relationships can be maintained.	Relationships can be maintained.

❖ DataAdapter object :-

It is associated Command and Connection objects.

- The DataAdapter is a class at the core of ADO.NET's disconnected data access.
- It is a bridge between the Database and DataSet.
- The DataAdapter provides a set of methods and properties to retrieve and save data between a DataSet and Database.
- The DataAdapter is used **fill** method to fill a DataSet.
- The DataAdapter can commit the changes to the Database by it's **Update** method.
- The DataAdapter provides four properties that represent Database Commands:

1. SelectCommand
2. InsertCommand
3. DeleteCommand
4. UpdateCommand



- When the DataAdapter fills a DataSet, It will create the necessary tables and columns for the returned data.
- There are two ways to initialize it:

```
Dim da As New OleDbDataAdapter(cmd)
```

OR

```
Dim da As New OleDbDataAdapter
```

```
da.SelectCommand = cmd
```

❖ DataSet :

ADO.NET caches data locally on the client and store that data into DataSet.

- The DataSet is a disconnected, in-memory representation of data. It's not exact copy the Database.
 - It can be considered as a local copy of some portions of the Database.
 - The DataSet contains a collection of one or more DataTable objects made up of rows and columns of data.
 - Tables can be identified in DataSet using DataSet's **Tables property**.
 - It also contains primary key, foreign key, constraint and relation information about the data in the DataTable objects.
 - Whatever operations are made by the user it is stored temporary in the DataSet, when the use of this DataSet is finished, changes can be made back to the central database for updating.
 - DataSet doesn't know where the data it contains came from, and in fact it can contain data from multiple sources.
 - Remember that The DataSet is not connected to a data source after it is populated.
 - Inside a DataSet, like in a Database, there are tables, columns, relationships, constraints, view, and so on...
 - The DataSet is populated DataAdapter's **Fill** method.
- Ex:-

```
Dim ds As New Data.DataSet
```

```
da.Fill(ds)
```

▪ **Example:- Display Data in the GridView using DataSet.**

First of all we have to put one **GridView** control on a web page and then write the following code.

```
Imports System.Data.OleDb
```



Partial Class _Default

Inherits System.Web.UI.Page

Protected Sub Page_Load(ByVal sender As Object, ByVal e As System.EventArgs) Handles Me.Load

Dim con As New OleDbConnection("Provider= Microsoft.jet.OLEDB.4.0; Data Source= D:\asp.net\practicals\connectivity\App_Data\my_database.mdb")

Dim cmd As New OleDbCommand("select * from stud", con)

Dim da As New OleDbDataAdapter(cmd)

Dim ds As New Data.DataSet

da.Fill(ds)

GridView1.DataSource = ds

GridView1.DataBind()

End Sub

End Class

❖ Data Controls

- Data control enable you to connect a web page to various sources of data, which includes databases and XML files.
- Data controls also let you display data on the page in tables or in other formats, and enable users to edit data.

❖ Data Source Controls :-

- we always use the **AccessDataSource** control in conjunction with a data-bound control to retrieve data from a relational database and to display, edit and sort data on a web page with little or no code.
- The **AccessDataSource** can support any relational database that can be connected to using an ADO.NET provider, such as the SqlClient, OleDb, OracleClient.



- SqlDataSource
- AccessDataSource
- LinqDataSource
- ObjectDataSource
- XmlDataSource
- SiteMapDataSource

- These Data Source fall within one of two categories:

Represent Tabular Data

- SQLDataSource
- AccessDataSource
- ObjectDataSource

Represent Tabular and hierarchical Data

- XMLDataSource
- SiteMapDataSource

- Databound controls are associated with one of these datasources with its DataSourceId property.

❖ **DataBound Controls :-**

- There are three main DataBound controls.

List Control:

- BulletedList
- CheckboxList
- DropDownList
- Listbox
- RadioButtonList

Hierarchical databound Controls:

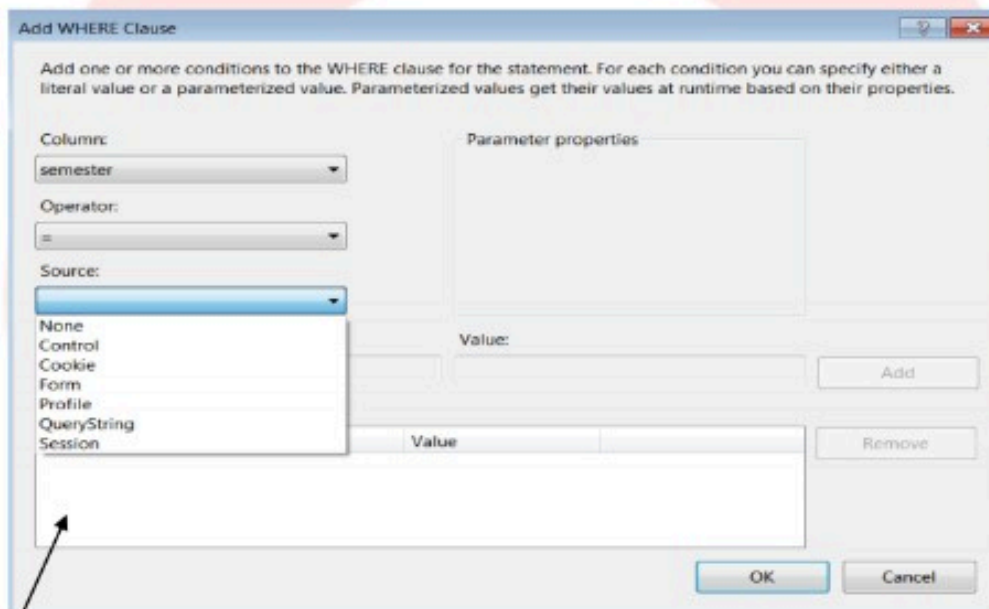
- Menu
- TreeView

Tabular databound Controls:

- Display a set of data
 - GridView
 - DataList
 - Repeater

- Display a single data item at a time
 - DetailsView

❖ Parameters Options in DataSource Controls :-



Parameters

Source

None

ControlParameter

CookieParameter

FormParameter

ProfileParameter

Meaning

Static Value.

Value of a control or page property.

Value of a browser cookie.

Value of an HTML form field.

Value of a Profile property.

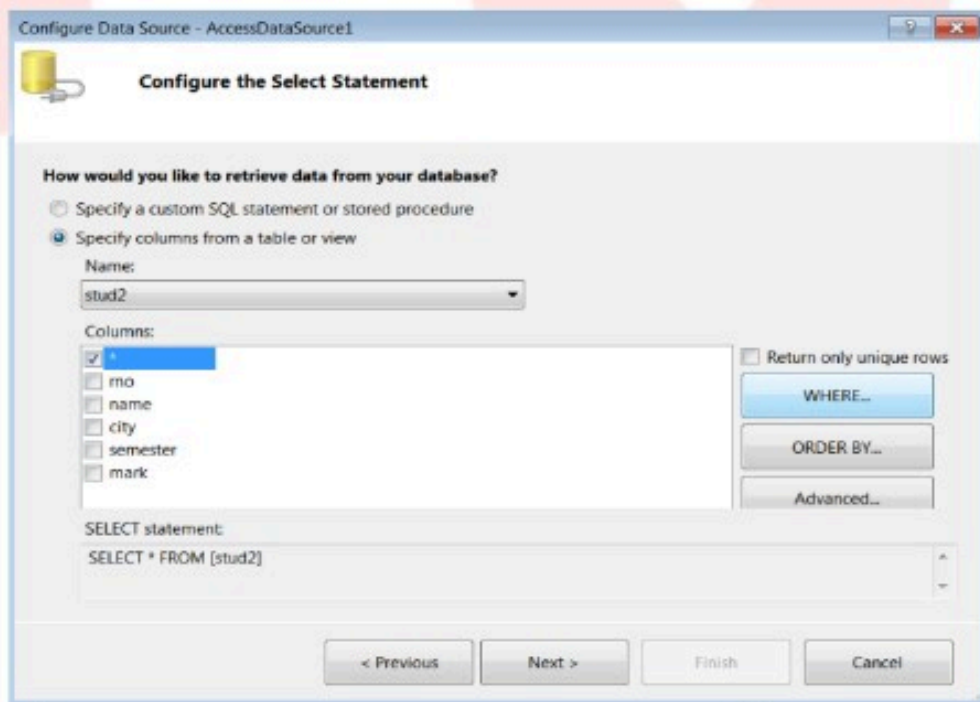
QueryStringParameter	Value of a query string field.
SessionParameter	Value of an item stored in a session.

Adding a WHERE Clause

- In the previous example we saw how to use the **AccessDataSource** control to retrieve data directly from a database. using the configure Data Source wizard, we could choose the database and then select the columns to return from a table; enter a custom SQL statement.
- Whether selecting columns from a table or entering a custom SQL statement, the **AccessDataSource** control's **SelectCommand** property is assigned the resulting ad-hoc SQL SELECT statement and it is this SELECT statement that is executed when the **Select()** method of **AccessDataSource** is invoked.
- Now, create a Parameterized Query:

Ex:- (1) Example of Parameter passing in AccessDataSource. Enter semester and display details of the students.

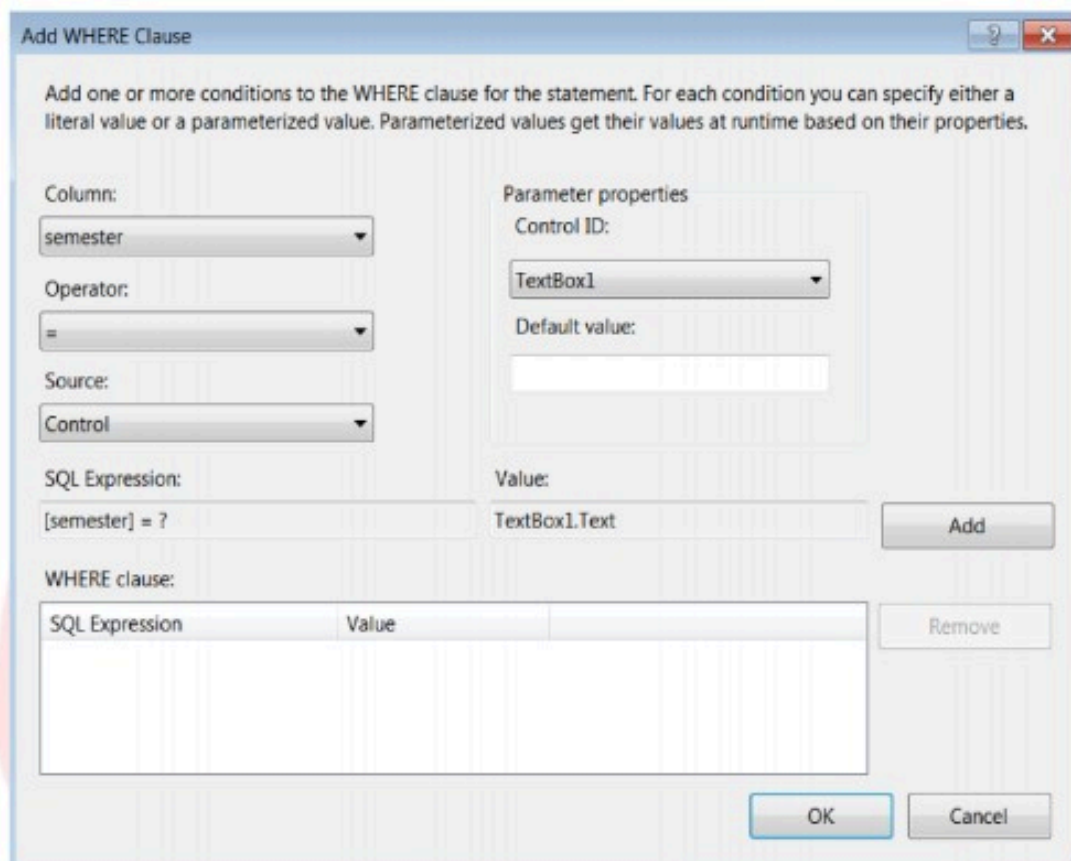
- Click the WHERE button, which brings up the Add WHERE Clause dialog box.



- Now, Add Parameters:
Select "semester" for Column, "=" for Operator, "Control" for Source and "TextBox1" for Control Id.



- Now, Click on Add button and then Click on OK button, which brings back to the Configure DataSource dialog box.



Add WHERE Clause

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

Column: semester

Operator: =

Source: Control

SQL Expression: [semester] = ?

Parameter properties

Control ID: TextBox1

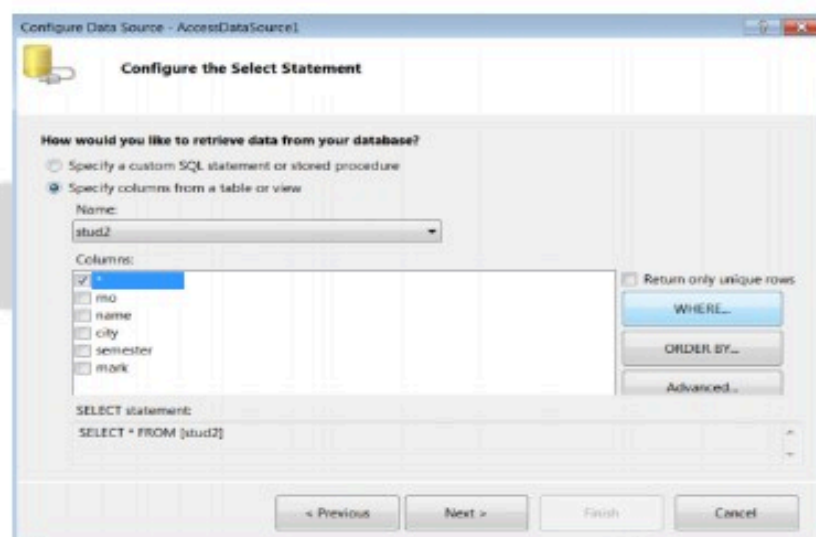
Default value:

Value: TextBox1.Text

WHERE clause:

SQL Expression	Value
----------------	-------

Buttons: Add, Remove, OK, Cancel



Configure Data Source - AccessDataSource1

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: stud2

Columns:

- ☒ id
- ☐ mo
- ☐ name
- ☐ city
- ☐ semester
- ☐ mark

☐ Return only unique rows

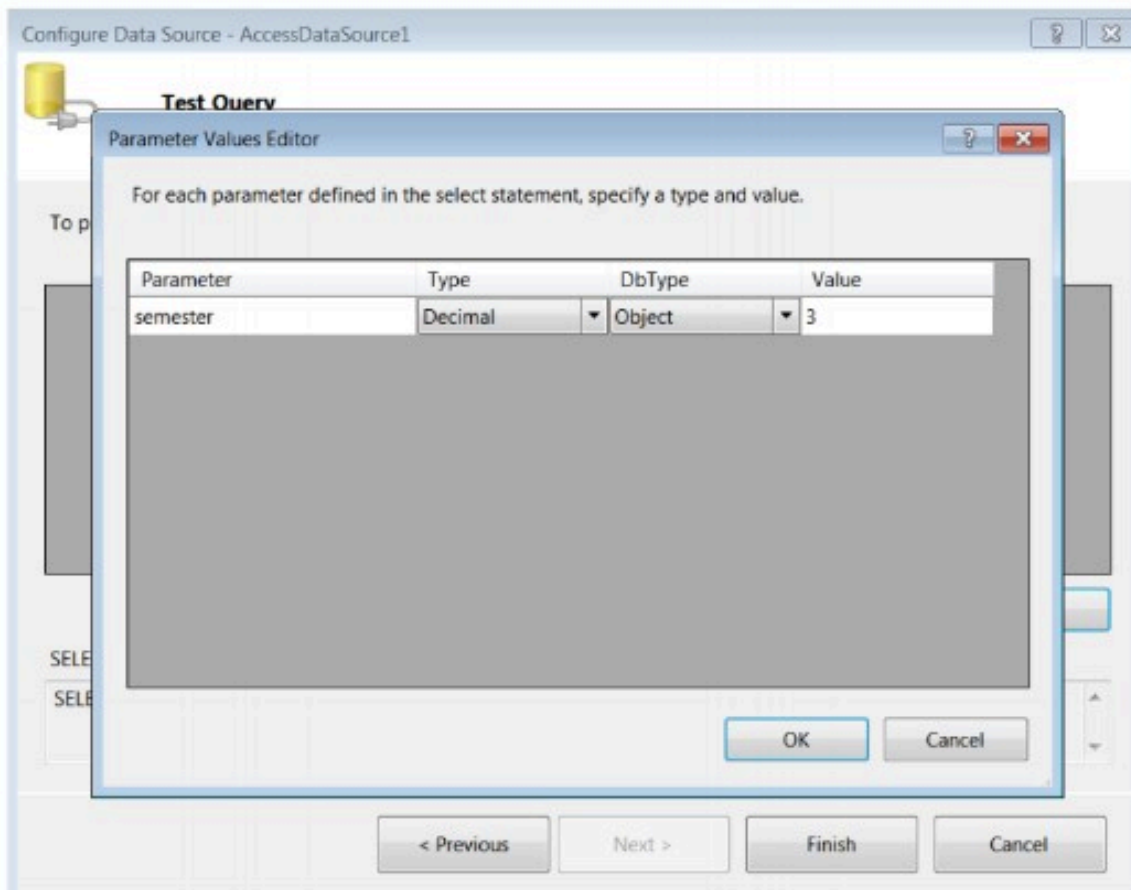
Buttons: WHERE..., ORDER BY..., Advanced...

SELECT statement:
SELECT * FROM [stud2]

Buttons: < Previous, Next >, Finish, Cancel

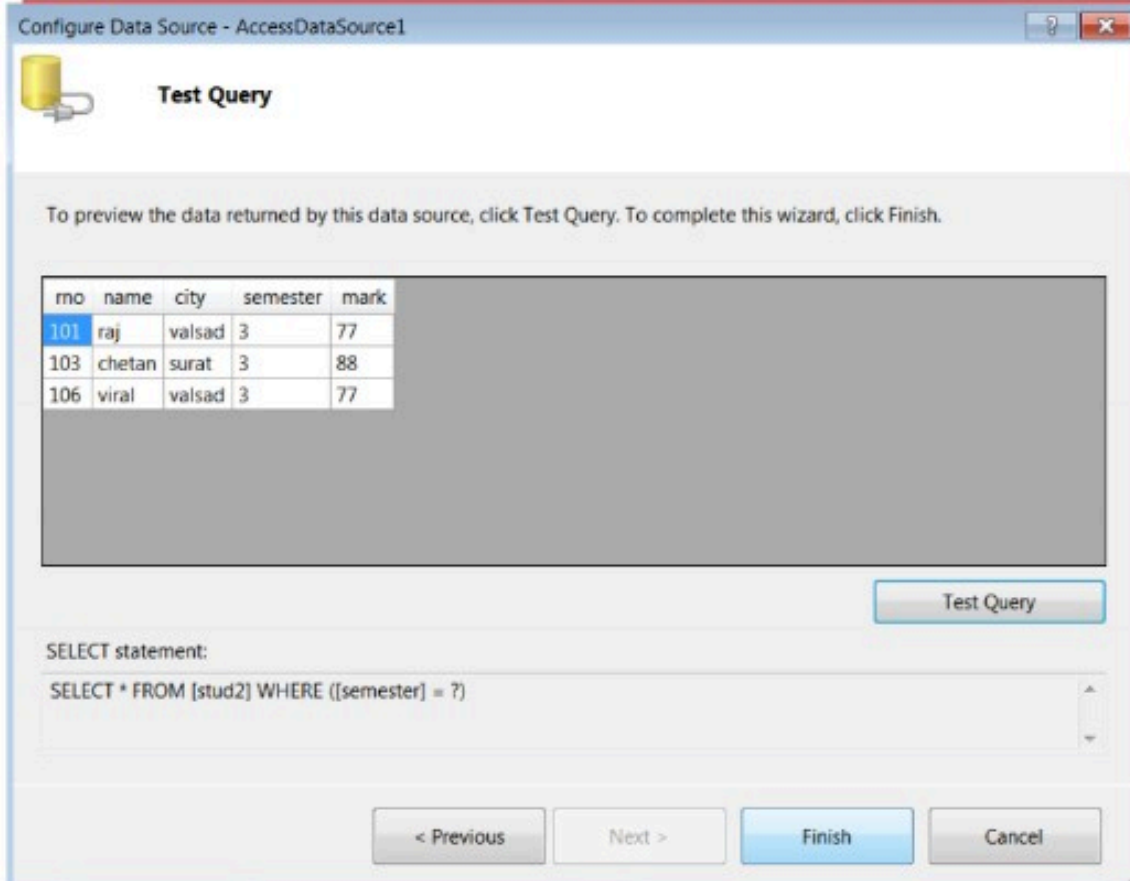


- Now, Click on Next button and then Click on **Test Query** button. It will display the **Parameter Values Editor**.
- In the **Parameter Values Editor** we will specify type and value. Here we choose Decimal as **Type** and enter value 3 for **value** field.
- Click on 'OK' button.

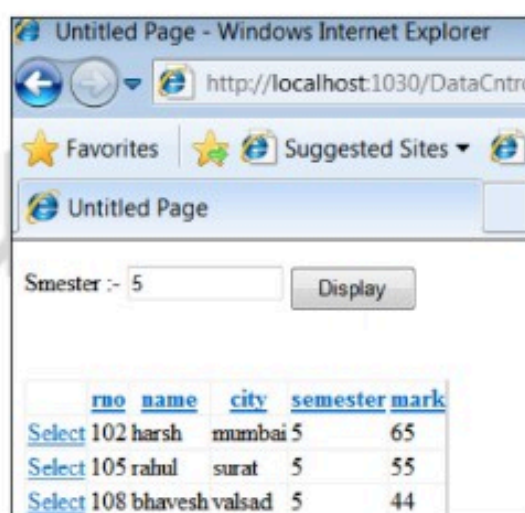


- You can see the data returned by the DataSource.

Jump2Learn



- To complete this wizard, Click 'Finish'.
- Now, you can Run your application.
- If you enter the value-5 in the TextBox1, then display the details of all the students who are in semester-5.



Ex:- (2) Display students details whose mark is in selected Range.



- First of all put Two Label, Two TextBoxes, One Button, AccessDataSource control and GridView control on the web page.
- Configure Data Source. Select table and columns to be retrieve.
- Now, Add **WHERE** criteria.

The 'Add WHERE Clause' dialog box is shown. It contains fields for Column, Operator, Source, SQL Expression, and Value. A table under 'WHERE clause:' lists conditions. The 'Add' button is visible.

SQL Expression	Value
[mark] >= ?	txtMin.Text
[mark] <= ?	txtMax.Text

- Specify the mark range in the Parameter Values Editor Window.

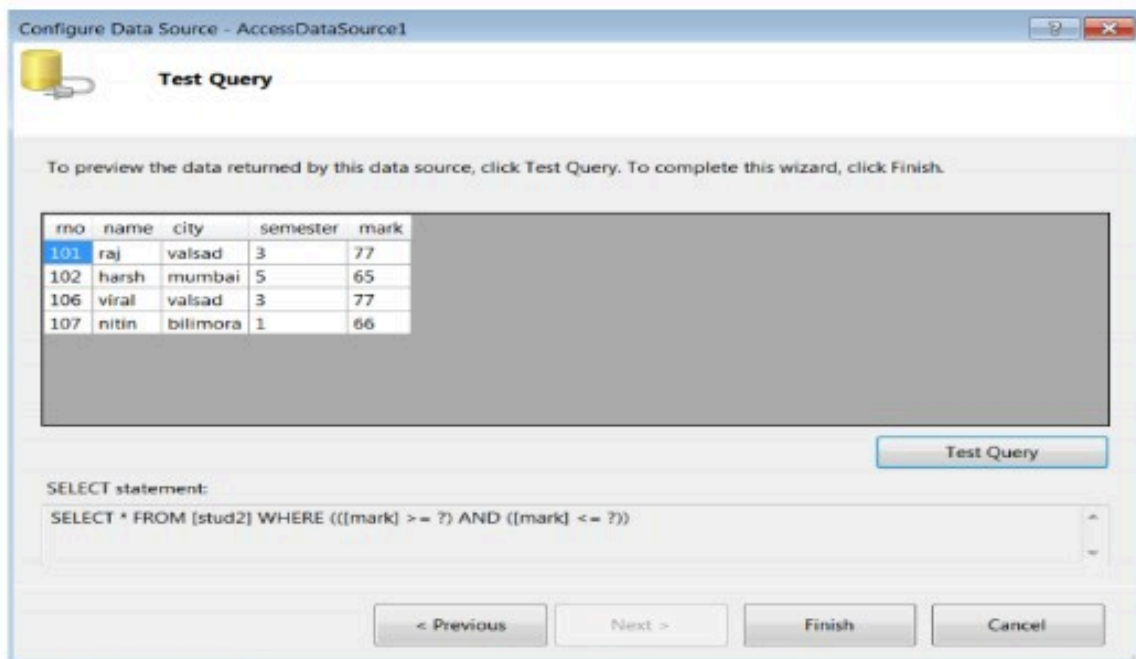
The 'Parameter Values Editor' dialog box is shown. It contains a table for specifying parameter types and values. The 'Test Query' button is visible.

Parameter	Type	DbType	Value
mark	Decimal	Object	60
mark	Decimal	Object	80

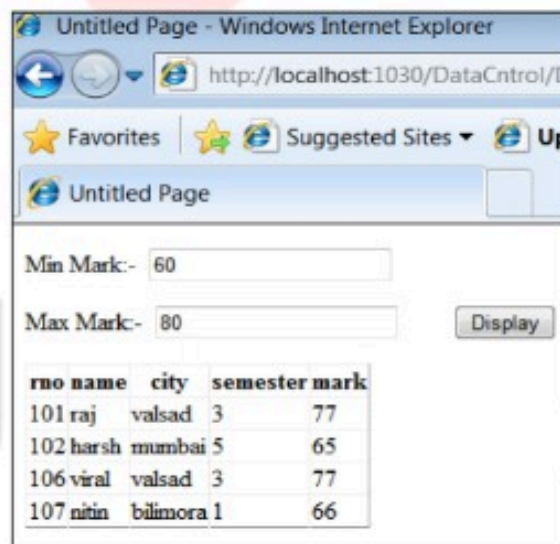
SELECT statement:
SELECT * FROM [stud2] WHERE (([mark] >= ?) AND ([mark] <= ?))



- You can see the data returned by the DataSource.



- To complete this wizard, Click 'Finish'.
- Now, you can Run your application.



Note:- Remember that to display data in the **GridView** we have to Bind it with the **AccessDataSource** by selecting **AccessDataSource1** from **Choose Data Source**.



Ex:- (2) Using session pass the Parameter.

- Create two web pages named page1.aspx and page2.aspx.
- Put the **AccessDataSource**, **DropDownList** and a **Display Button** on the page1.
- Now, configure the Data Source ----> choose the Database. Click on Next.
- Select **semester** column and also tick **Return only unique rows**.
- Click on Next button.

The screenshot shows the 'Configure Data Source - AccessDataSource1' dialog box, specifically the 'Configure the Select Statement' step. The dialog has a title bar with a question mark and a close button. Below the title bar is a yellow database icon and the text 'Configure the Select Statement'. The main area contains the question 'How would you like to retrieve data from your database?' with two radio buttons: 'Specify a custom SQL statement or stored procedure' (unselected) and 'Specify columns from a table or view' (selected). Below this is a 'Name:' dropdown menu showing 'stud2'. To the right of the dropdown is a 'Columns:' list box containing the following items: '*' (unchecked), 'rno' (unchecked), 'name' (unchecked), 'city' (unchecked), 'semester' (checked), and 'mark' (unchecked). To the right of the columns list is a checkbox labeled 'Return only unique rows' which is checked. Below the checkbox are three buttons: 'WHERE...', 'ORDER BY...', and 'Advanced...'. At the bottom of the dialog is a text box labeled 'SELECT statement:' containing the text 'SELECT DISTINCT [semester] FROM [stud2]'. At the very bottom are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

- Click on **Test Query** button so that following dialog box will display, which contain the data returned by the Datasource.



Configure Data Source - AccessDataSource1

Test Query

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

semester
1
3
5

Test Query

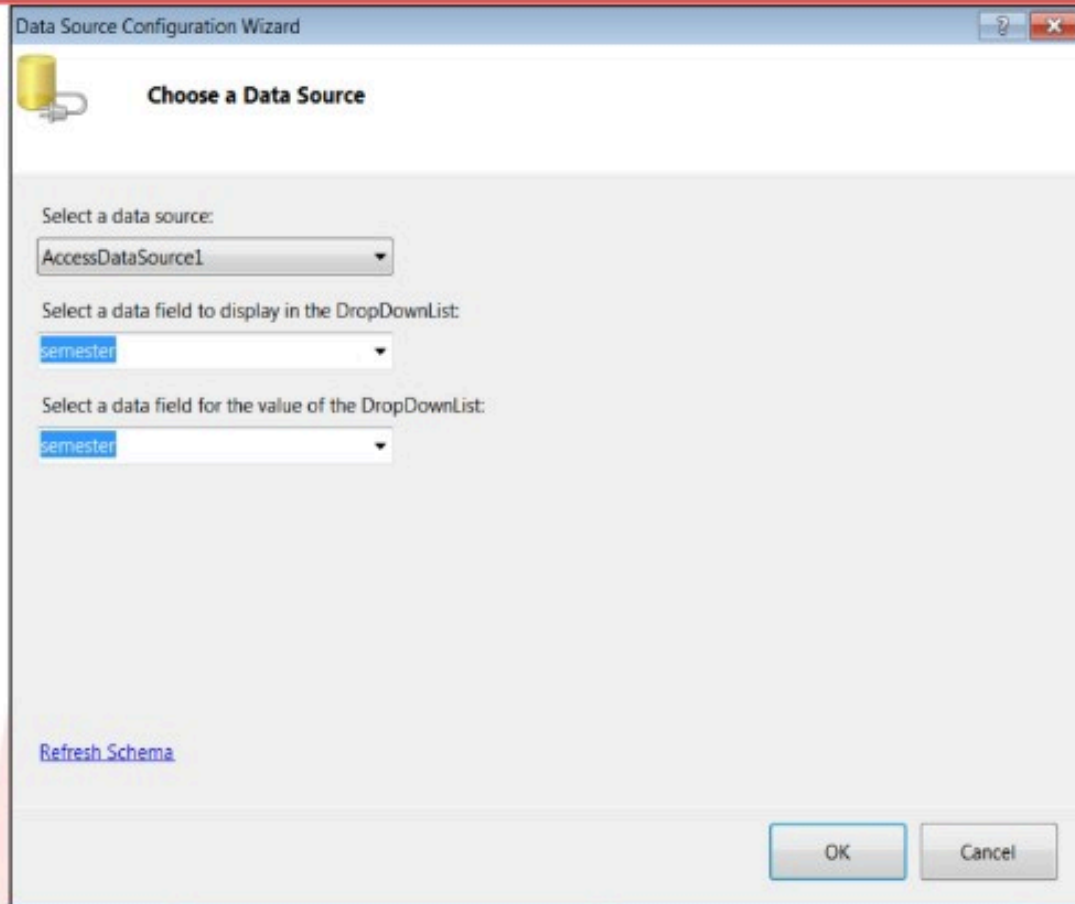
SELECT statement:

SELECT DISTINCT [semester] FROM [stud2]

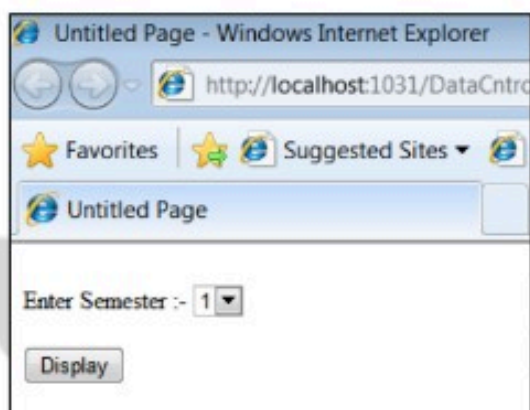
< Previous Next > Finish Cancel

- To complete this wizard, Click Finish.
- Attach **DropDownList1** with **AccessDataSource** control.
- Select **Choose Data Source** from the DropDownList smart tag.
- Select a Data Source and Data Field and then Click on OK button.

Jump2Learn



- **Output View of page1.aspx.**



- **Coding View of page1.**

Partial Class page1

Inherits System.Web.UI.Page

Protected Sub btnDisplay_Click(ByVal sender As Object, ByVal e As System.EventArgs) Handles btnDisplay.Click

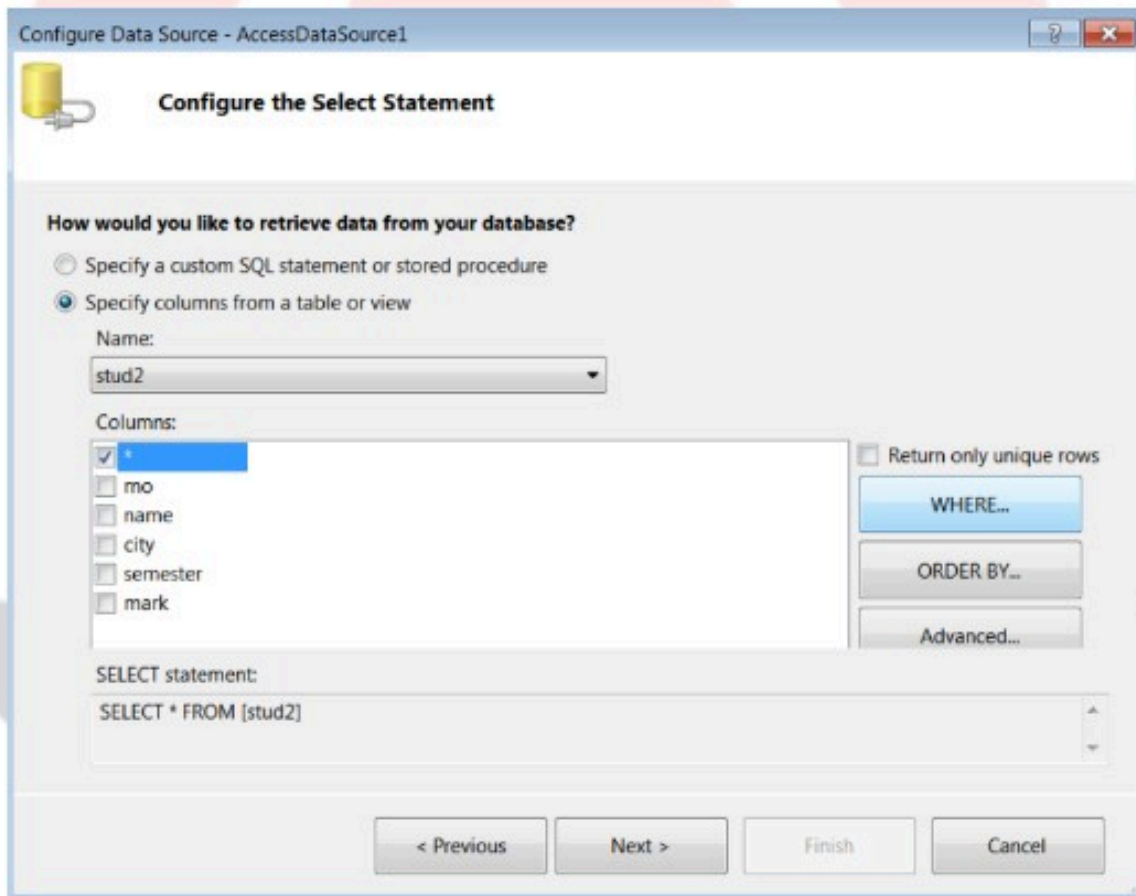
Session("semester") = DropDownList1.Text

Response.Redirect("page2.aspx")

End Sub

End Class

- Now, Put the **AccessDataSource** and **GridView** control on the page2.
- Now, configure AccessDataSource, choose the database and Click on Next button.
- Choose table and columns and then Click on WHERE button to Add where criteria. (here we select * for all columns)



The screenshot shows the 'Configure Data Source - AccessDataSource1' dialog box, specifically the 'Configure the Select Statement' step. The dialog has a title bar with a question mark and a close button. Below the title bar is a yellow database icon and the text 'Configure the Select Statement'. The main area is titled 'How would you like to retrieve data from your database?' and has two radio buttons: 'Specify a custom SQL statement or stored procedure' (unselected) and 'Specify columns from a table or view' (selected). Below the radio buttons is a 'Name:' label and a dropdown menu showing 'stud2'. Under 'Columns:', there is a list box with the following items: '*' (selected), 'mo', 'name', 'city', 'semester', and 'mark'. To the right of the list box is a checkbox labeled 'Return only unique rows' which is unchecked. Below the list box are three buttons: 'WHERE...' (highlighted in blue), 'ORDER BY...', and 'Advanced...'. At the bottom of the dialog is a text area labeled 'SELECT statement:' containing the text 'SELECT * FROM [stud2]'. At the very bottom are four buttons: '< Previous', 'Next >', 'Finish', and 'Cancel'.

- Now, select column:, Operator and Source. Give the name for the session field. Here, we select "semester" for column, "=" for Operator and "Session" for Source. We also specify the session field as "semester"



- Click on Add button and then Click on Ok button.

Add WHERE Clause

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

Column:
semester

Operator:
=

Source:
Session

SQL Expression:
[semester] = ?

Parameter properties

Session field:
semester

Default value:

Value:
Session("semester")

WHERE clause:

SQL Expression	Value

Add

Remove

OK

Cancel

- Click on Next button and then click on **Test Query** button.
- Specify the Type and Value in the Parameter Values Editor and then click on OK.

Jump2Learn



Parameter Values Editor

For each parameter defined in the select statement, specify a type and value.

Parameter	Type	DbType	Value
semester	Decimal	Object	5

OK Cancel

Test Query

SELECT statement:

SELECT * FROM [stud2] WHERE ([semester] = ?)

< Previous Next > Finish Cancel

- Now, Click on Finish to complete this wizard.
- Don't forget to choose Data Source for GridView.
- Same as above we can use Query String.

Jump2Learn

Add WHERE Clause

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

Column:

Operator:

Source:

SQL Expression:

Parameter properties

QueryString field:

Default value:

Value:

WHERE clause:

SQL Expression	Value
[semester] = ?	Request.QueryString("semester")

❖ GridView Control :-

- It is the only DataBound control which has inbuilt edit, update, delete, sort, paging data functionality. We can enable this functionality by setting the properties of GridView without even writing a single line of code.
- It provides more flexibility in displaying and working with data from your database in comparison with any other controls.
- The GridView control enables you to connect to a Datasource and display data in a tabular format.
- Using this control is very easy when compared to other databound controls to display data. You have to just assign the DataSource property to a Data Table and calling **DataBind()** method.
- The main **disadvantage of GridView control is, it does not have the flexibility to display data in any other format except table.**

Some Properties:

- DataSource:- Gets or sets the data source object that contains the data to populate the control.
- AllowPaging:- true/false. Indicate whether the control should support paging.



- AllowSorting:- true/false. Indicate whether the control should support Sorting.
- AutoGenerateEditButton:- true/false. Indicates whether a separate column should be added to edit the record.
- AutoGenerateDeleteButton:- true/false. Indicates whether a separate column should be added to Delete the record.
- AutoGenerateSelectButton:- true/false. Indicates whether a separate column should be added to Select the particular record.
- AlternatingRowStyle:- Define the style properties for every alternate row in the GridView.
- Caption :- Gets or sets the caption of the GridView.
- PageIndex :- Gets or sets the number of records to display in one page of GridView.

Event:

- PageIndexChanging and PageIndexChanged :- Both events occur when the page link is clicked.

Jump2Learn