CS-32: Programming with ASP.NET

UNIT – 3

ADO .NET and Database

Dharmesh Kapadiya
Dhruvita Savaliya

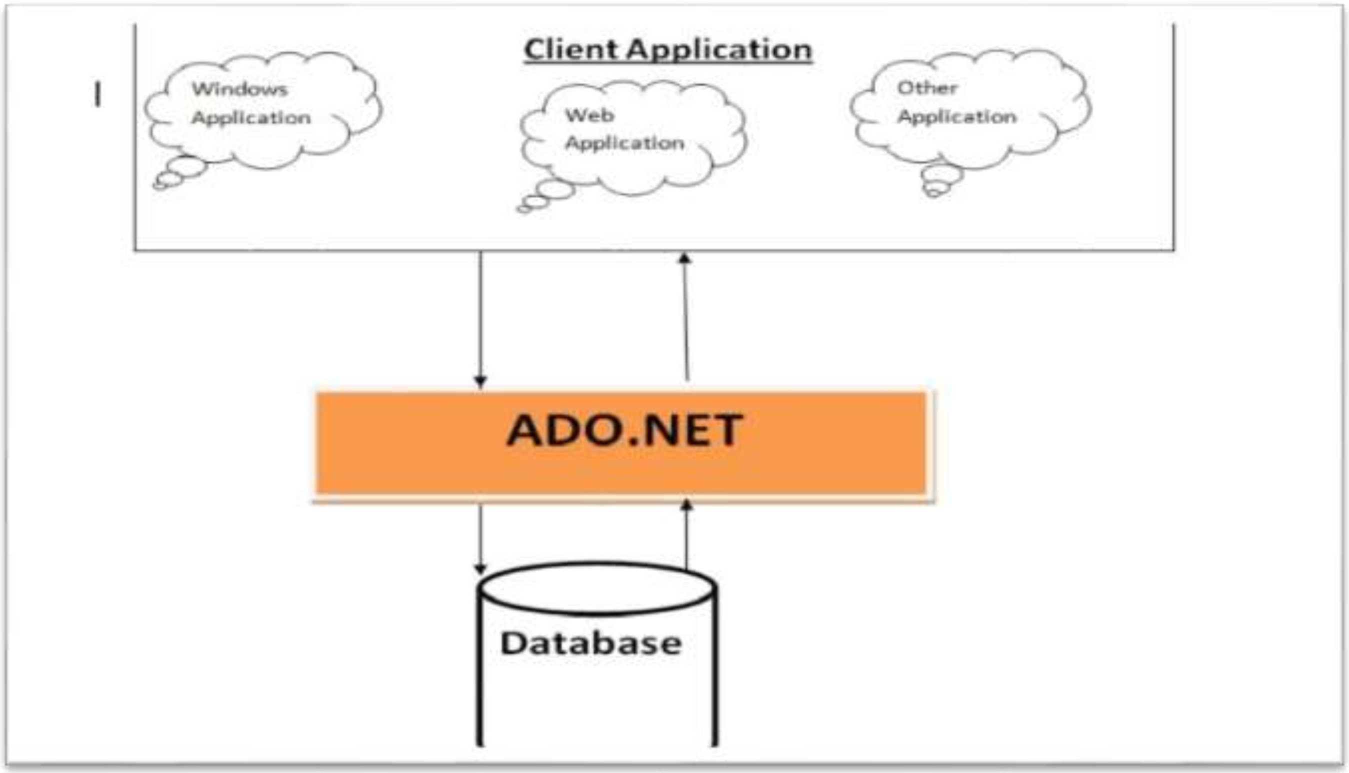
TOPICS :

- Architecture of ADO.NET
- ADO.NET Classes for Connected and Disconnected Architecture
  - Connection,
  - Command,
  - DataReader,
  - DataAdapter,
  - DataSet,
  - DataColumn,
  - DataRow,
  - DataConstraints,
  - DataView etc.
- The Gridview Control, The Repeater Control
- Binding Data to DataBound Controls,
- Diplaying Data in a webpage using SQLDataSource Control
- DataBinding Expressions

Introduction of ADO.net :

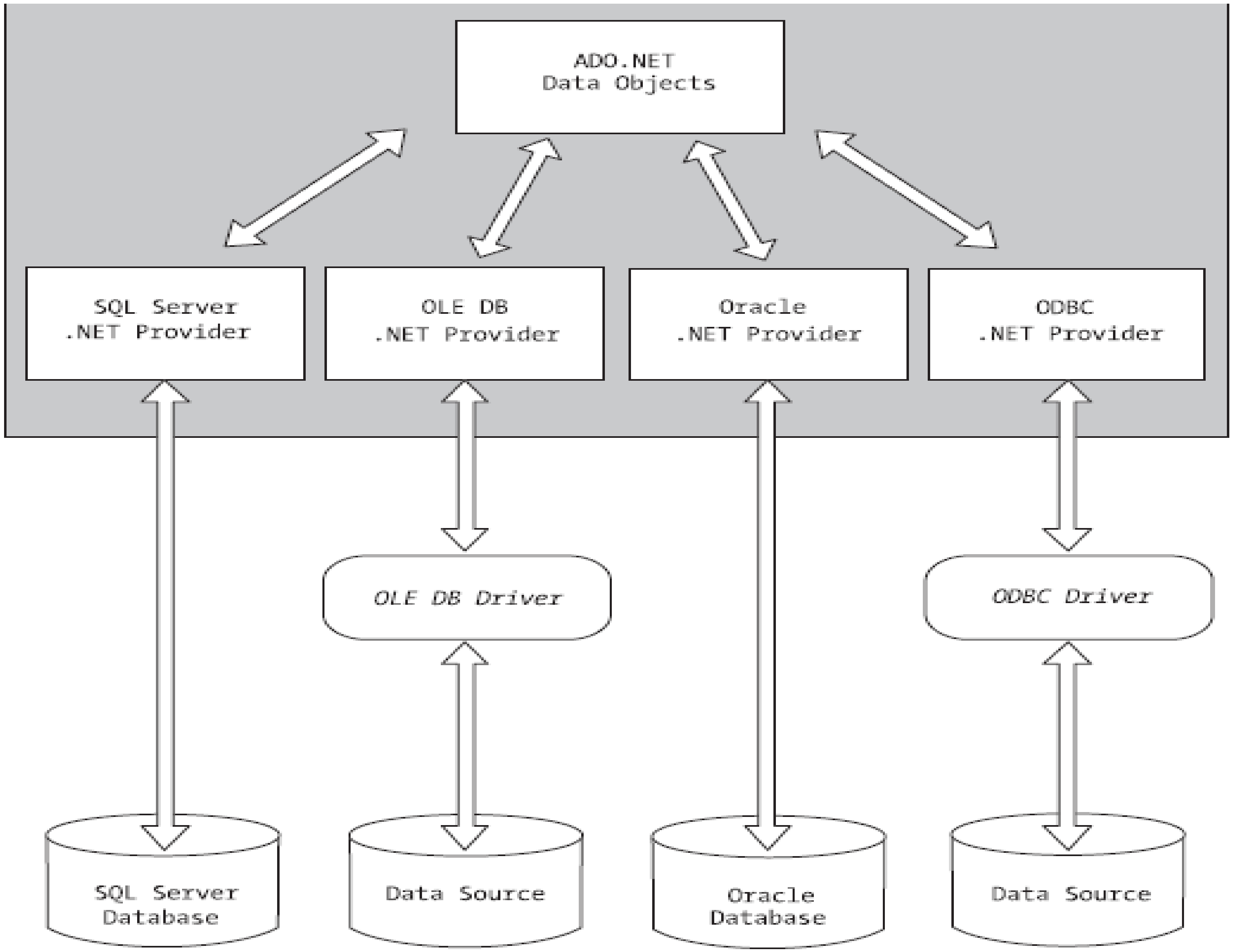
- ADO stands for ActiveX Data Objects. OR Microsoft ActiveX Data Objects.
- ADO.NET is a module of .Net Framework which is used to establish connection between application and data sources.
  Or we can also say that :
  ADO.NET provides a bridge between the front end controls and the back end database.
- Data sources can be such as SQL Server and XML.
- ADO.NET consists of classes that can be used to connect, retrieve, insert and delete data.
- ADO.NET is a set of classes that expose data access services for .NET Framework programmers.
- ADO.NET provides a rich set of components for creating distributed, data-sharing applications.
- It is an integral part of the .NET Framework, providing access to relational, XML, and application data.

- **How ADO.net Work :**



Data Providers in ADO .NET :

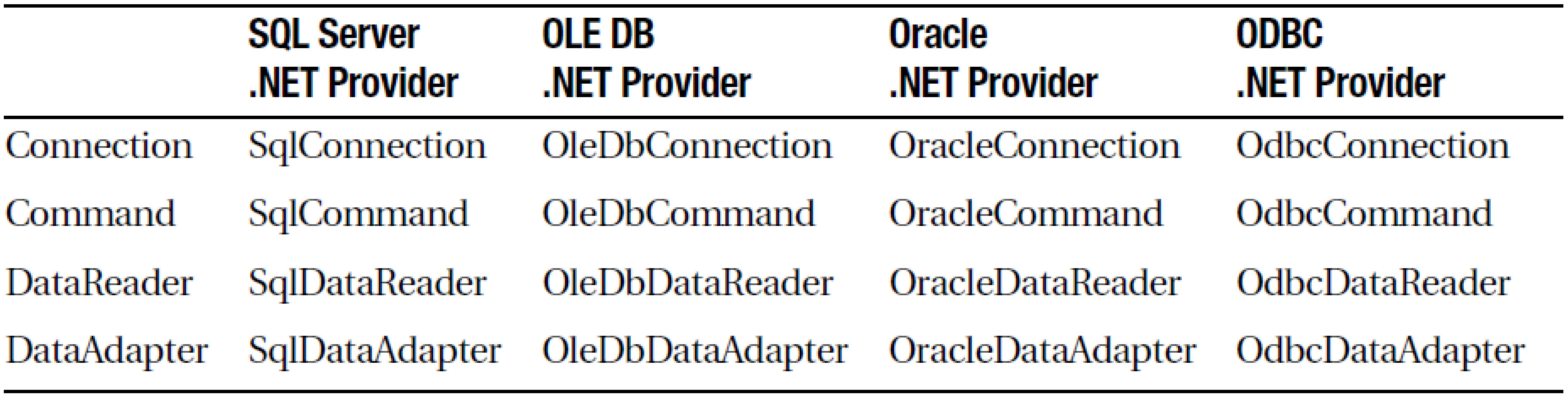
- Data Providers are different set of classes which provide functionality to access to different data sources like Access, Oracle, SQL Server etc… databases.
- There are four types of Data Providers are provided by .Net.


ADO.NET
Data Objects
SQL Server
.NET Provider
OLE DB
.NET Provider
Oracle
.NET Provider
ODBC
.NET Provider
OLE DB Driver
ODBC Driver
SQL Server
Database
Data Source
Oracle
Database
Data Source

- **Namespaces used in ADO.NET**
- Use of Namespaces depends on which provider you choose. Some of the classes are obtained from some common namespaces. But in order to use specific provider you need to import specific namespace.

Namespaces	Description
System.Data	Contains the definition for Columns ,relations, tables, database, rows , views and constraints.
System.Data.SqlClient	Contains the classes to connect to a Microsoft SQL Server database such as SqlCommand, SqlConnection, and SqlDataAdapter.
System.Data.Odbc	Contains classes required to connect to most ODBC drivers. These classes include OdbcCommand and OdbcConnection.
System.Data.OracleClient	Contains classes such as OracleConnection and OracleCommand required to connect to an Oracle database.
System.Data.OLEDB	Contains the classes you use to connect to an OLE DB data source, including OleDbCommand and OleDbConnection.

- Depending on the namespace which you have imported, you can use provider specific classes in order to use ADO.NET. Consider following table for provider specific classes.
- SQL stands for Structured query language.
- OLE DB stands for Object Linking and Embedding Database.
- ODBC stands for Open Database Connectivity.

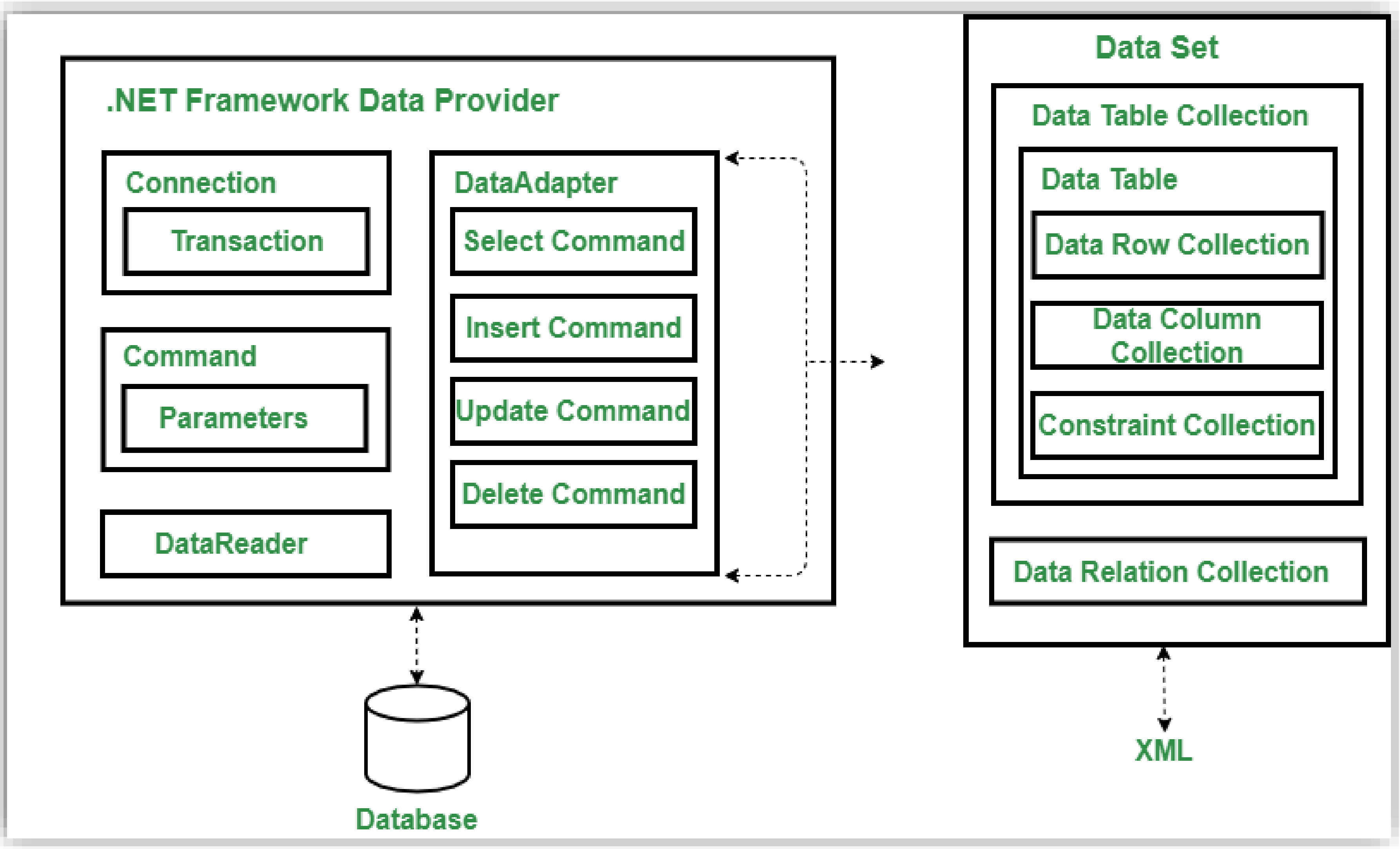


	SQL Server .NET Provider	OLE DB .NET Provider	Oracle .NET Provider	ODBC .NET Provider
Connection	SqlConnection	OleDbConnection	OracleConnection	OdbcConnection
Command	SqlCommand	OleDbCommand	OracleCommand	OdbcCommand
DataReader	SqlDataReader	OleDbDataReader	OracleDataReader	OdbcDataReader
DataAdapter	SqlDataAdapter	OleDbDataAdapter	OracleDataAdapter	OdbcDataAdapter

ADO.NET has several advantages :

- You can use ADO.NET to connect with any type of database.
- ADO.NET supports Connected as well as Disconnected Architecture. Disconnected Architecture is a new invention since ADO.NET is introduced.
- XML is supported by ADO.NET which is widely used for data transformation widely today. This makes your data transformation much more faster.
- You can create Fast, Scalable and Secured applications using ADO.NET
- Cross Language support is provided by ADO.NET as Multilanguage is a prime feature of .NET framework.

Architecture of ADO.net :



- The two main components of ADO.NET for accessing and manipulating data are data provider and dataset.

- **DataSet** represent either an entire database or a subset of database. It can contain tables and relationships between those tables.

- **Data provider** is a collection of components like connection, command, DataReader, DataAdapter objects and handles communication with physical data store and the dataset.

- **There are Two Types of ADO.net Architecture :**

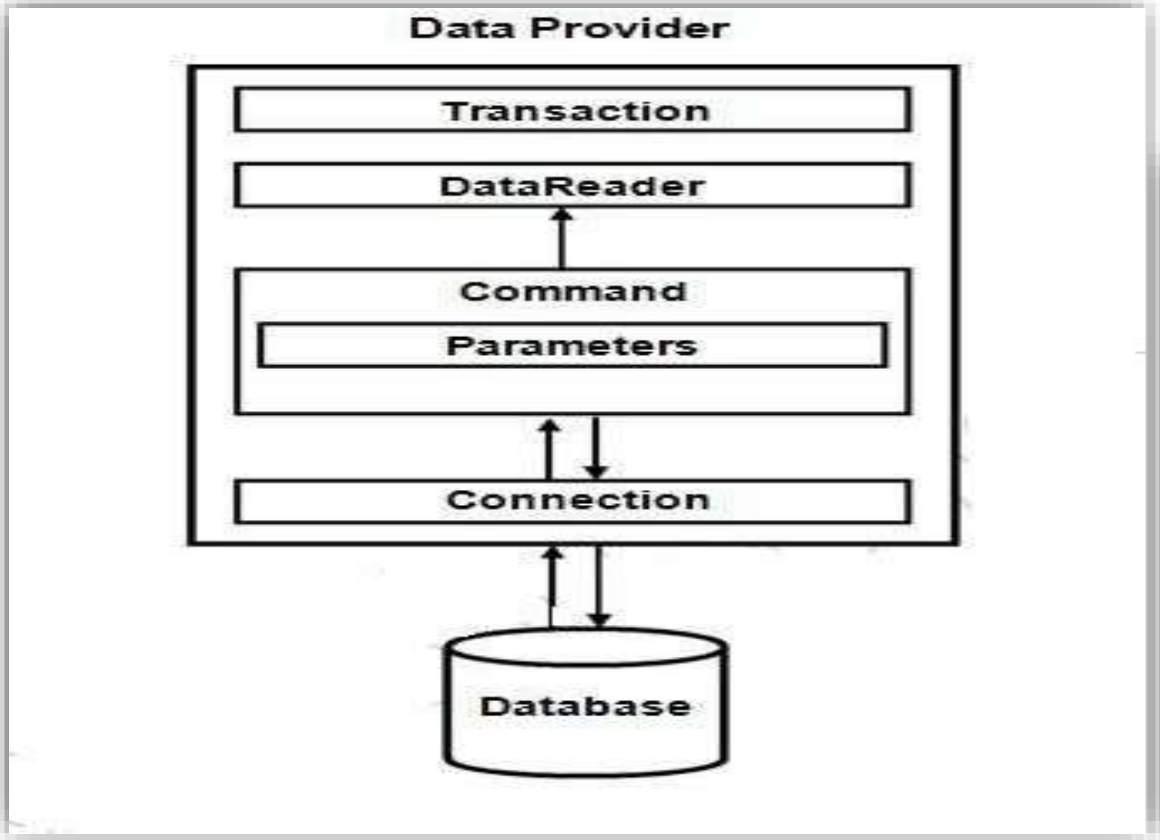
1. Connected Architecture
2. Disconnected Architecture

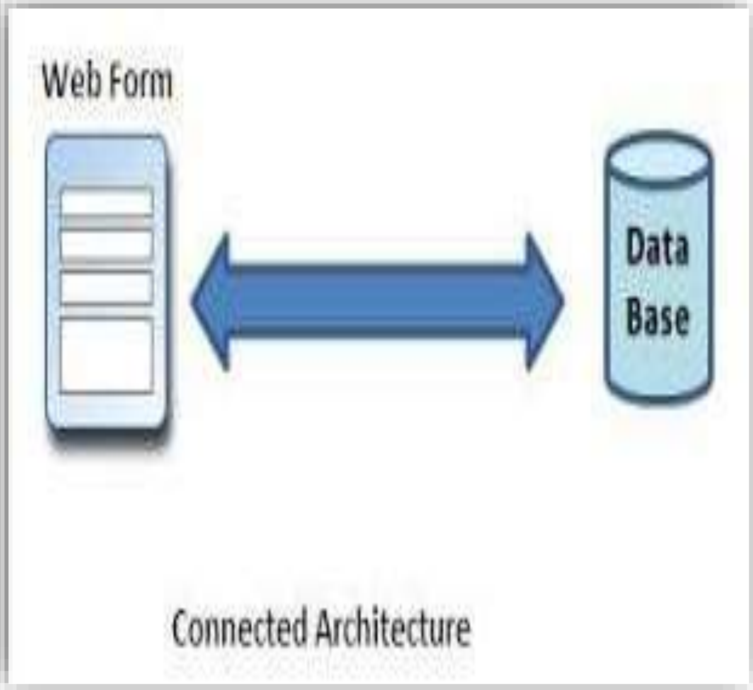
Connected Architecture :

- The architecture of ADO.net, in which connection must be opened to access the data retrieved from database is called as connected architecture.
- We typically employ the DataReader class object for connected architecture.
- In addition to ensuring that the connection is maintained for the entire period of time, DataReader is used to retrieve data from databases.
- Connected Oriented Architecture gives faster performance when dealing with smaller applications, with Select SQL queries and smaller data.
- We can access the data in a **forward-only** and **read-only** manner.
- In connected architecture, the application is **directly linked** to the Database.

- Classes that have been used in Connected Architecture include :

1. Connection
2. Command
3. DataReader
4. Transaction
5. Parameter





- In the connected environment database is not store in a client PC. It means the back up is not on the client PC.
- A data is store directly on server and user wants show a data at a time always we want to connect to the database.

- Following set of classes are used while using Connected Architecture :
- **Connection :** This allows you to connect with database.
- **Command :** This allows you to give commands like Insert, Update, Delete, Select, etc.
- **DataReader :** This allows you to read data directly from the database..
- In connected architecture you have to declare the connection explicitly by using Open(), and close the connection using Close().
- you can execute commands using different methods like.. ExecuteNonQuery, ExecuteScalar, ExecuteReader etc.

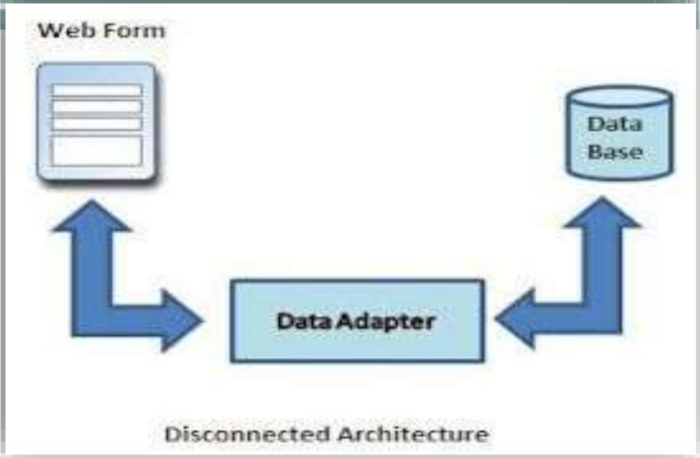
Disconnected Architecture :

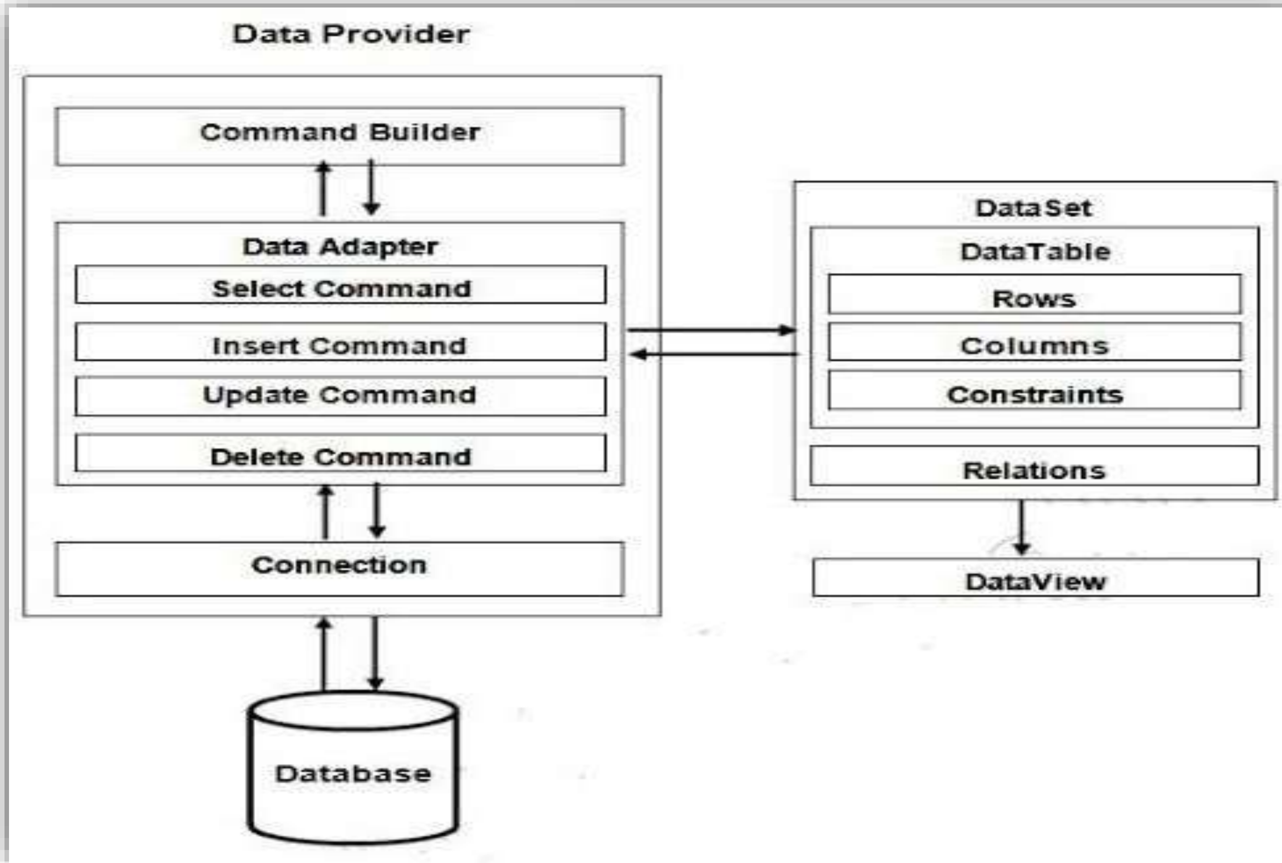
- The architecture of ADO.net in which data retrieved from database can be accessed even when connection to database was closed is called as disconnected architecture.
- It is disconnection-oriented data access architecture. It means always an active and open connection is not required.
- Using DataAdapter and DataSet or DataTable we can implement Dis-Connected Oriented Architecture.
- DataSet is DisConnected Architecture since all the records are brought at once and there is no need to keep the connection alive.
- This indicates that data is fetched from the database and stored in temporary tables because data is required during the processing of the application, then fetched from the temporary tables whenever it is needed.
- Disconnected Architecture can hold the data of multiple tables using dataset and single table data using Datatable.

- You can access the data in **forward and backward directions** and you can also **modify the data**.
- Following set of classes are used while using Disconnected Architecture:
- **Connection :** This allows you to connect with database.
- **DataAdapter :** This plays an important role in Disconnected Architecture. DataAdapter is a mediator which provides (fills) data into DataSet / DataTable and again updates the data back to database.
- **DataSet :** This is used if you want to load some group of tables into local memory. You can also use DataSet to load single table data. DataSet is collection of tables.
- **DataTable :** This is used if you want to load only one table into local memory.
- **DataRow :** This is used incase you want to add one new row into DataSet or DataTable. You can also use DataRow to remove rows from DataSet or DataTable.
- **DataColumn :** This is used incase you want to modify any column or you want to add a new column to DataSet or DataTable.

- Classes that have been used in Disconnected Architecture include :

1. DataAdapter
2. DataSet
3. DataTable
4. DataColumn
5. DataRow
6. DataRelationship
7. DataView





Points	Connected Architecture	Disconnected Architecture
Connection	The application will establish a connection with database server and it will be kept open until unless the task has been completed.	The required data will be fetched into the dataset and then the connection will be closed.
Traffic	For each and every operation we need to communicate with database server, so that traffic to the database server will be **increased**.	The required data manipulations will be done on the tables into the dataset and the changes will be updated back to the database at once, so that traffic to the database sever can be **reduced**.
Transaction	Parallel transactions will be easy.	Parallel transactions will be difficult.
Speed	Fetching the data will be fast.	Fetching the data will be slow.
Memory	No memory management is required.	Memory management is required for dataset.
data read	**DataReader** for read the data.	**Dataset** for read the data.
Way of data access	We can access the data in a **forward-only**	You can access the data in **forward** and **backward** directions
Type of Data	Data is **read only** data	You can **modify the data**.

Connected Architecture :

- Connection
- Command
- DataReader

Connection Class :

- As the name suggest, its main purpose is to establish a connection to database.
- This connection should remain open each time any transaction occurs.
- When developing database applications using .NET, the very first thing that we need is a connection to the database. ADO.NET provides us with connection classes like the SqlConnection class and OleDbConnection class. The SqlConnection class is part of the SQL Server .NET Data Provider. This data provider has been designed for performance optimization with SQL Server 7.0 and later.
- The OleDbConnection is part of the OLEDB .NET Data Provider, which is used to access a data source that has an OLEDB Provider.

• Properties of Connection Object :

Property	Description
CommandTimeout	Sets or returns the number of seconds to wait while attempting to execute a command
ConnectionString	Sets or returns the details used to create a connection to a data source.
ConnectionTimeout	Sets or returns the number of seconds to wait for a connection to open.
DefaultDatabase	Sets or returns the default database name .
Mode	Sets or returns the provider access permission
Provider	Sets or returns the provider name .
State	Returns a value describing if the connection is open or closed .
Version	Returns the ADO version number

- **Methods of Connection Object :**

Method	Description
BeginTransaction	Begins a new transaction
Close	Closes a connection
Open	Opens a connection
ChangeDatabase	Allows you to change the name of database.

- **Events of Connection Object :**

Event	Description
BeginTransComplete	Triggered after the BeginTrans operation
CommitTransComplete	Triggered after the CommitTrans operation
ConnectComplete	Triggered after a connection starts
Disconnect	Triggered after a connection ends
ExecuteComplete	Triggered after a command has finished executing
RollbackTransComplete	Triggered after the RollbackTrans operation
WillConnect	Triggered before a connection starts
WillExecute	Triggered before a command is executed

Example :

```
using System.Data.SqlClient;
public partial class _Default : System.Web.UI.Page
{
    SqlConnection con = new SqlConnection(@"Data Source=(LocalDB)\v11.0;AttachDbFilename=E:\App_Data\Database.mdf;Integrated Security=True");
                                  ||Or||
    //SqlConnection con = new SqlConnection(@"Data Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\Database.mdf;Integrated Security=True");

    con.Open();
    con.Close();
}
```

Command :

- The SqlCommand class is at the heart of the Data Provider’s namespace.
- It is used to execute operations on a database and retrieve data.
- This is specially used under Connected Architecture.
- Command allows you to specify different types of SQL Commands like Insert, Update, Delete, Select, etc.
- It has several methods which help you to manipulate or fetch the data from database.
- While using Command object, it is necessary to open the connection using Open() method before attempting any operation.
- It will not open the connection automatically like Disconnected Architecture.
- The same way after operation is done, you need to close the connection using Close() method.

- **Properties of Command Object :**

Property	Description
CommandText	Contains the text of a SQL query
CommandTimeout	Contains the length of the timeout of a query, in seconds
CommandType	Specifies the type of command to be executed
Connection	Specifies the connection to the database
Parameters	Specifies a collection of parameters for the SQL query
Transaction	Specifies a transaction object, which enables developers to run queries in a transaction

- **Event of Command Object :**

Event	Description
StatementCompleted	This event occurs when the Transactional-SQL command is finished at database end.

- **Methods of Command Object :**

Method	**Description**
ExecuteNonQuery()	Executes the CommandText property against the database and does not return a result set. This method is used incase you have given Transactional Commands like INSERT, UPDATE, DELETE, etc. This method returns integer value as answer. If value is > 0, this means command executed successfully. 0 means it did not have any effect.
ExecuteReader()	Executes the CommandText property and returns data in a DataReader object . This method is used if you have specified SELECT. This allows command object to connect with DataReader which helps you to read the data.
ExecuteScalar()	Executes the CommandText property and returns a single value. This method is also used when you have specified SELECT command. But it returns only one value. It returns value of First Row's First Column. For example your command was "Select * from student", even though entire table, it will give you value of first student's first column.
Cancel()	Cancels the running query
CreateParameter()	Returns a new SQL parameter

Example :

```
con.Open();
string sql = "insert into student values(1,'abc')";
SqlCommand cmd = new SqlCommand(sql, con);
int ans = cmd.ExecuteNonQuery();
if (ans > 0)
        Response.Write("Record Inserted");
else
        Response.Write("Problem in Query");
con.Close();
```

DataReader :

- The DataReader object defines a lightweight yet powerful object that is used to read information from a database.
- This is used under Connected Architecture.
- DataReader is used with Command object in order to read (fetch) the data.
- But remember that DataReader can be used to only read the data in sequential manner (forward only manner).
- You can not go back or go on particular record directly.

- **Properties of DataReader Object :**

Property	Description
FieldCount	Contains the number of fields retrieved from the query
IsClosed	Contains True if the DataReader object is closed
Item	Contains A collection of values that are accessible both by field name and by ordinal number
RecordsAffected	Returns the number of records affected by an executed query
HasRows	Gets a value that indicates whether the DataReader contains one or more rows

- **Methods of DataReader Object :**

Method	Description
Close()	Closes the DataReader object
IsDBNull()	Returns True if the field contains Null
Read()	Reads the next result in the result set into memory . The Read method of the DataReader object is used to obtain a row from the results of the query. Each column of the returned row may be accessed by passing the name or ordinal reference of the column to the DataReader

Example :

```
con.Open();
    SqlCommand cmd = new SqlCommand("select *from
  student",con);
    SqlDataReader rd;
    rd = cmd.ExecuteReader();
    if(rd.HasRows)
    {
      while (rd.Read())
      {
        Response.Write("ID : "+rd[0]);
        Response.Write("<br>Name : "+rd[1]);
        Response.Write("<br>Mobile : " + rd[2]);
      }
    }
    con.Close();
```

Display Data in GridView using Connected Architecture :

File.aspx :

```
<asp:GridView ID="GridView2" runat="server" >
</asp:GridView>
```

File.aspx.xs :

```
public void display()
   {
      con.Open();
      SqlCommand cmd = new SqlCommand("select *from demo", con);
      SqlDataReader rd;
      rd = cmd.ExecuteReader();
      GridView2.DataSource = rd;
      GridView2.DataBind();
      con.Close();
   }
```

Disconnected Architecture :

- DataAdapter
- DataSet
- DataColumn
- DataRow
- DataConstraints
- DataView

DataAdapter :

- DataAdapter is used under Disconnected Architecture. It plays an important role in disconnected architecture. It allows you to fetch the data from database to local memory of client and also to save the updated data back to database.
- In Disconnected Architecture we use DataSet or DataTable to fill the data which resides in local memory. DataAdapter object does two important tasks as follows
- Fill data from Database to DataSet or DataTable
- Update the data back to Database
- DataAdapter fills the specified data into DataSet or DataTable which is in our (client's) local memory. After filling the data connection is automatically terminated (closed) with DataAdapter. But now the data has been loaded into DataSet or DataTable. All types of operations like INSERT, UPDATE, DELETE, SELECT, etc. are performed locally in DataSet or DataTable.

• Properties of DataAdapter Object :

Property	Description
SelectCommand	This property works in background which generates your Select command automatically. OleDbCommandBuilder class works behind this to generate Select command automatically .
DeleteCommand	This property also works in background which generates your Delete command. If you have deleted some rows from DataSet or DataTable, OleDbCommandBuilder automatically generates Delete Command and stores in this.
InsertCommand	This property also works in background which generates Insert Command. If you have insert any new row in DataSet or DataTable, OleDbCommandBuilder automatically generates Insert command and stores in this.
UpdateCommand	This property also works in background which generates Update Command. If you have done some changes in DataSet or DataTable data, OleDbCommandBuilder automatically generates Update command and stores in this.

- **Methods of DataAdapter Object :**

Method	Description
Fill	Most important method of DataAdapter. This method helps you to fill the data into DataSet or DataTable. Remember that in DataAdapter we can give only SELECT command which is filled into DataSet or DataTable. The Fill method is probably the DataAdapter method you will use most frequently. Simply stated, the Fill method adds data from your data source to a dataset. The Fill method accepts a variety of parameters including the DataSet object to fill, a string representing the alias for the newly created DataSet object, an integer representing the lower bound of records to retrieve, and an integer representing the upper bound of records to retrieve from our data source.
Update	Most important method of DataAdapter. This method is used to save the changes back from DataSet / DataTable to actual database table. Before using Update() method you need to use SqlCommandBuilder class so that it can generate related INSERT, UPDATE, DELETE, ALTER, etc. command automatically The Update method calls the respective insert, update, or delete command for each inserted, updated, or deleted row in the DataSet. There are three different ways to call the Update method—you can pass: A DataSet object A DataSet object and a string representing a table name

- **Events of DataAdapter Object :**

Event	Description
FillError	This event is raised when there is a problem in filling data into DataSet or DataTable.
RowUpdating	This event is raised while changes are being made in Row. When you call Update() method, DataAdapter makes changes to actual database table. At that time, while being updating each row this event is raised.
RowUpdated	This event is raised after changes are done in Row. When you call Update() method, DataAdapter makes changes to actual database table. After changes have been done, this event is raised.

- **Example :**

```
protected void Button1_Click(object sender, EventArgs e)
   {
      con.Open();
      SqlDataAdapter ad = new SqlDataAdapter("insert  into demo
         values('"+TextBox1.Text+"','"+TextBox2.Text+"')", con);
      DataSet dt = new DataSet();
      ad.Fill(dt);
     con.Close();
}
```

- **Other Example of SqlDataAdapter Exampe :**

```
protected void savedata_Click(object sender, EventArgs e)
  {
        SqlDataAdapter adapter = new SqlDataAdapter();
        String sql= "Insert into emp(empnm,empcity,empmobile) values('" +
  empnm.Text + "','" + empcity.Text + "','" + empmobile.Text + "')";

         SqlCommand cmd = new SqlCommand(sql,con);
        adapter.InsertCommand = new SqlCommand(sql,con);
         con.Open();
        int a=adapter.InsertCommand.ExecuteNonQuery();
        if(a>0)
        {
              Response.Write("<script>alert('Data save
  successfully');</script>");
        }
        cmd.Dispose();
        con.Close();
    display();
  }
```

• Ways to display table data in gridview :

```
public void display()
   {
string selectSQL = "SELECT ProductID,
   ProductName, UnitPrice FROM
   Products";

 SqlCommand cmd = new
   SqlCommand(selectSQL, con);
SqlDataAdapter adapter = new
   SqlDataAdapter(cmd);

DataSet ds = new DataSet();
   adapter.Fill(ds, "Products");

GridView1.DataSource = ds;
   GridView1.DataBind();
}
```

```
public void display()
   {
      con.Open();
      SqlDataAdapter ad = new
SqlDataAdapter("select *from
   demo", con);

DataSet dt = new DataSet();
      ad.Fill(dt);

Gridview1.DataSource=dt;
Gridview1.DataBind();
}
```

DataSet :

- The DataSet object represents an in-memory group of database tables.
- You can have more then one tables under DataSet.
- It is collection of DataTables.
- The DataSet is the main in disconnected, data-driven application; it is an in-memory representation of a complete set of data, including tables, relationships, and constraints.
- The DataSet does not maintain a connection to a data source, enabling true disconnected data management.
- The data in a DataSet can be accessed, manipulated, updated, or deleted, and then reconciled with the original data source.
- Since the DataSet is disconnected from the data source, there is less contention for valuable resources, such as database connections, and less record locking.

- **Properties of DataSet Object :**

Property	Description
DataSetName	Gets or sets the name of the current DataSet.
HasErrors	Gets a value indicating whether there are errors in any of the DataTable objects within this DataSet.
IsInitialized	Gets a value that indicates whether the DataSet is initialized.
Relations	Get the collection of relations that link tables and allow navigation from parent tables to child tables.
Tables	Gets the collection of tables contained in the DataSet.

- **Events of DataSet Object :**

Event	Description
Initialized	Occurs after the DataSet is initialized .

• Methods of DataSet Object :

Method	Description
AcceptChanges()	Commits all the changes made to this dataset since the last time AcceptChanges was called.
Clear()	Clears all the DataTables of DataSet
ReadXml()	Reads XML schema and data into DataSet using specified Stream or File
ReadXmlSchema()	Reads an XML schema into the DataSet using the specified stream or File
RejectChanges	Rolls back all changes that have been made to the dataset since it was loaded, or the last time AcceptChanges was called.
Select	Allows you to select group of data from dataset. It has different methods to select the data. This is overloaded method which allows you to filter the data in variety of ways.
WriteXml()	Writes the current contents of the DataSet as XML using the specified Stream or File.
WriteXmlSchema()	Writes the current data structure of the DataSet as an XML schema to the specified stream or file.

DataTable :

- The DataTable object represents an in-memory database table.
- You can add rows to a DataTable with a DataAdapter.
- A DataTable is defined in the “System.Data” namespace and represents a single table of memory-resident data.
- It contains a collection of columns represented by the DataColumnCollection, which defines the schema and rows of the table.
- It also contains a collection of rows represented by the DataRowCollection, which contains the data in the table.
- Along with the current state, a DataRow retains its original state and tracks changes that occur to the data.
- The DataTable class is a central class in the ADO.NET architecture; it can be used independently, and in DataSet objects.
- A DataTable consists of a Columns collection, a Rows collection, and a Constraints collection.
- The Columns collection combined with the Constraints collection defines the DataTable schema, while the Rows collection contains the data.

• Properties of DataTable Object :

Property	Description
Columns	The Columns collection is an instance of the DataColumnCollection class, and is a container object for zero or more DataColumn objects. The DataColumn objects define the DataTable column, including the column name, the data type, and any primary key or incremental numbering information.
Constraints	The Constraints collection is an instance of the ConstraintCollection class, and is a container for zero or more ForeignKeyConstraint objects and/or UniqueConstraint objects. The ForeignKeyConstraint object defines the action to be taken on a column in a primary key/foreign key relationship when a row is updated or deleted. The UniqueConstraint is used to force all values in a column to be unique.
DataSet	Gets the DataSet to which this table belongs.
HasErrors	Gets a value indicating whether there are errors in any of the rows in any of the tables of the DataSet to which the table belongs.

- **Properties of DataTable Object :**

Property	Description
IsInitialized	Gets a value that indicates whether the DataTable is initialized.
PrimaryKey	Gets or sets an array of columns that function as primary keys for the data table.
Rows	The Rows collection is an instance of the DataRowCollection class, and is a container for zero or more DataRow objects. The DataRow object contains the data in the DataTable, as defined by the DataTable.Columns collection. Each DataRow has one item per DataColumn in the Columns collection.
TableName	Gets or sets the name of the DataTable.

- **Methods of DataTable Object :**

Method	Description
AcceptChanges()	Commits all the changes made to this DataTable since the last time AcceptChanges was called.
Clear()	Clears all the DataTables of DataTable
NewRow()	Creates a new DataRow with the same schema as the table.
ReadXml()	Reads XML schema and data into DataTable using specified Stream or File
ReadXmlSchema()	Reads an XML schema into the DataTable using the specified stream or File
RejectChanges	Rolls back all changes that have been made to the dataset since it was loaded, or the last time AcceptChanges was called.
Select	Allows you to select group of data from DataTable. It has different methods to select the data. This is overloaded method which allows you to filter the data in variety of ways.
WriteXml()	Writes the current contents of the DataTable as XML using the specified Stream or File.
WriteXmlSchema()	Writes the current data structure of the DataTable as an XML schema to the specified stream or file.

• Events of DataTable Object :

Event	Description
Initialized	Occurs after the DataTable is initialized.
RowChanged	Occurs after a DataRow has been changed successfully.
RowChanging	Occurs when a DataRow is changing.
RowDeleted	Occurs after a row in the table has been deleted.
RowDeleting	Occurs before a row in the table is about to be deleted.
TableCleared	Occurs after the Table is cleared.
TableClearing	Occurs when then Table is being cleared.
TableNewRow	Occurs when you insert a new Row in table.

Example :

```
protected void Page_Load(object sender, EventArgs e)
  {
        DataTable table1 = new DataTable("patients");
        table1.Columns.Add("name");
        table1.Columns.Add("id");
        table1.Rows.Add("Ram", 1);
        table1.Rows.Add("Rahim", 2);

        DataTable table2 = new DataTable("medications");
        table2.Columns.Add("id");
        table2.Columns.Add("medication");
        table2.Rows.Add(1, "abc");
        table2.Rows.Add(2, "xyz");

        DataSet set = new DataSet("office");
        set.Tables.Add(table1);
        set.Tables.Add(table2);
        Response.Write(set.GetXml());
}
```

DataTable with GridView Example :

```
protected void Page_Load(object sender, EventArgs e)
   {
         DataTable table1 = new DataTable("patients");
         table1.Columns.Add("name");
         table1.Columns.Add("id");
         table1.Rows.Add("Ram", 1);
         table1.Rows.Add("Rahim", 2);

        GridView1.DataSource = table1;
         GridView1.DataBind();
}
```

GridView Control :

- The GridView is the most powerful control because it comes equipped with the most ready-made functionality. This functionality includes features for automatic paging, sorting, selecting, and editing.

  Automatically Generating Columns
- The GridView provides a DataSource property for the data object you want to display.Once you’ve set the DataSource property, you call the DataBind() method to perform the data binding and display each record in the GridView.
- Here’s all you need to create a basic grid with one column for each field:
- **<asp:GridView ID="GridView1" runat="server" />**

Display data in GridView using connected & dis-Connected architecture :

Connected Architecture :

```
public void display()
  {
    con.Open();
    SqlCommand cmd = new
   SqlCommand("select *from demo",
   con);
    SqlDataReader rd;
    rd = cmd.ExecuteReader();

    GridView2.DataSource = rd;
    GridView2.DataBind();
    con.Close();
  }
```

Dis-Connected Architecture :

```
public void display()
  {
     con.Open();
     SqlDataAdapter ad = new
SqlDataAdapter("select *from demo",
  con);
DataSet dt = new DataSet();
     ad.Fill(dt);

Gridview1.DataSource=dt;
Gridview1.DataBind();
}
```

Defining Columns :

Class	Description
BoundField	This column displays text from a field in the data source.
ButtonField	This column displays a button in this grid column.
CheckBoxField	This column displays a check box in this grid column. It’s used automatically for true/false fields
CommandField	This column provides selection or editing buttons.
HyperLinkField	This column displays its contents (a field from the data source or static text) as a hyperlink.
ImageField	This column displays image data from a binary field (providing it can be successfully interpreted as a supported image format).
TemplateField	This column allows you to specify multiple fields, custom controls, and arbitrary HTML using a custom template. It gives you the highest degree of control but requires the most work.

- **Example of CommandField for Delete/Update/Edit/Cancel:**

```
<asp:GridView ID="GridView2" runat="server" DataKeyNames="Id"
   DataSourceID="SqlDataSource1"
   OnRowCancelingEdit="GridView2_RowCancelingEdit"
   OnRowDeleting="GridView2_RowDeleting1"
   OnRowEditing="GridView2_RowEditing"
   OnRowUpdating="GridView2_RowUpdating">
         <Columns>
<asp:BoundField DataField="Id" HeaderText="Id" InsertVisible="False"
   ReadOnly="True" SortExpression="Id" />
            <asp:BoundField DataField="name" HeaderText="name"
   SortExpression="name" />
            <asp:BoundField DataField="mobile" HeaderText="mobile"
   SortExpression="mobile" />
            <asp:CommandField ShowDeleteButton="True"
   HeaderText=“Delete"/>
            <asp:CommandField ShowEditButton="True" HeaderText="update" />
         </Columns>
       </asp:GridView>
```

Using Styles :

Style	Description
HeaderStyle	Configures the appearance of the header row that contains column titles, if you've chosen to show it (if ShowHeader is true).
RowStyle	Configures the appearance of every data row.
AlternatingRowStyle	If set, applies additional formatting to every other row. This formatting acts in addition to the RowStyle formatting. For example, if you set a font using RowStyle, it is also applied to alternating rows, unless you explicitly set a different font through AlternatingRowStyle.
SelectedRowStyle	Configures the appearance of the row that's currently selected. This formatting acts in addition to the RowStyle formatting.
EditRowStyle	Configures the appearance of the row that's in edit mode. This formatting acts in addition to the RowStyle formatting.
FooterStyle	Configures the appearance of the footer row at the bottom of the GridView, if you've chosen to show it (if ShowFooter is true).

Example of Styling in GridView :

```
<asp:GridView ID="GridView1" runat="server"
   DataSourceID="sourceProducts"
AutoGenerateColumns="False">
<RowStyle BackColor="#E7E7FF" ForeColor="#4A3C8C" />
<HeaderStyle BackColor="#4A3C8C" Font-Bold="True"
   ForeColor="#F7F7F7" />
<Columns>
<asp:BoundField DataField="ProductID" HeaderText="ID" />
<asp:BoundField DataField="ProductName"
   HeaderText="Product Name" />
</Columns>
</asp:GridView>
```

Properties of GridView :

Property	Description
AllowPaging	It is used to set pagging if table data is too long.
PageSize	If you set AllowPaging to true pagesize works.It breaks the pages as per size specified in PageSize Property.
AllowSorting	It is used to sort data in gridview.
AutoGeneratedColumns	It is used to automatically generate columns which are in bounded table
AutoGenerateDeleteButton	Displays delete link button with records.
AutoGenerateEditButton	Displays edit link button with records.
AutoGenerateSelectButton	Displays select link button with records.
Caption	Sets caption for GridView.
GridLines	Allows you to display gridlines withing GridView.

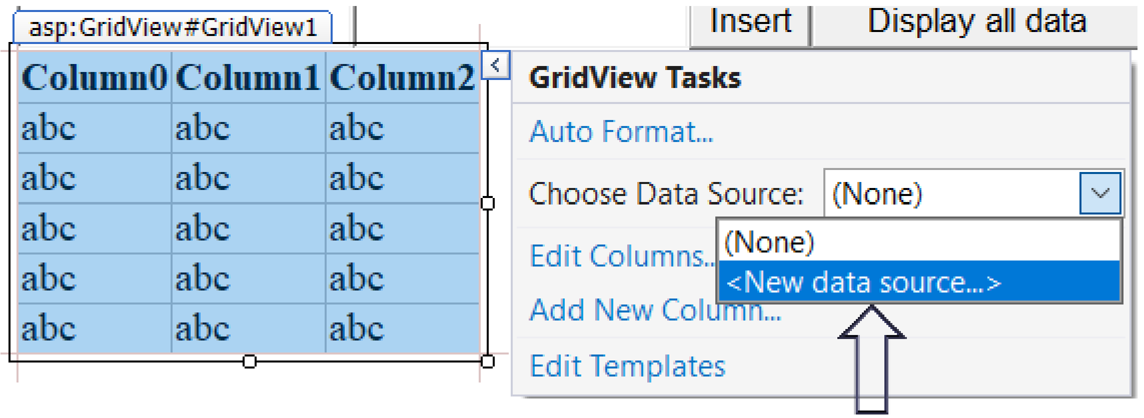
Methods of GridView :

Method	Description
DataBind	Allows you to bind the data with specified data source.
DeleteRow	It is used to delete specified row from gridview.
UpdateRow	It is used to update specified row in gridview.
Sort	Sorts the data in GridView according to two parameter.1)SortExpression 2)SortDirection

Events of GridView :

Event	Description
DataBinding	Raised while data is bounded to GridView.
DataBound	Raised after the data is bounded to GridView.
PageIndexChanged	Raised after page index is changed.This property will work if you set AllowPaging to true.
RowCreated	Raised when you create new row.
RowDeleted	Raised after the row is deleted.
RowUpdated	Raised after the row is updated.

- GridView with SqlDataSource : Diplaying Data in a webpage using SQLDataSource Control:

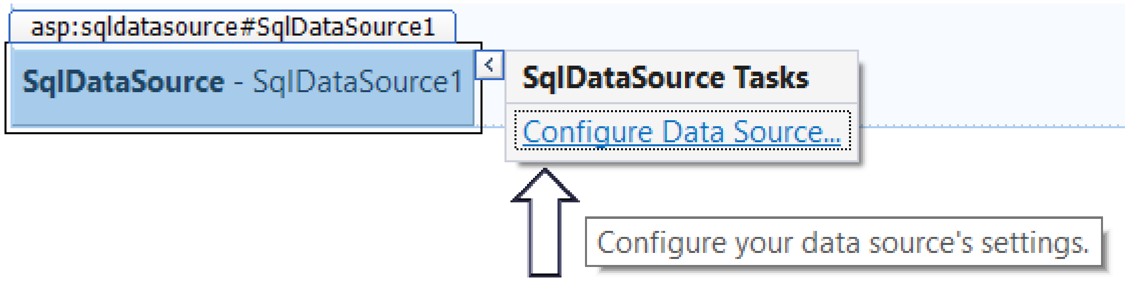


Column0	Column1	Column2
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc
abc	abc	abc

Choose Data Source: (None)

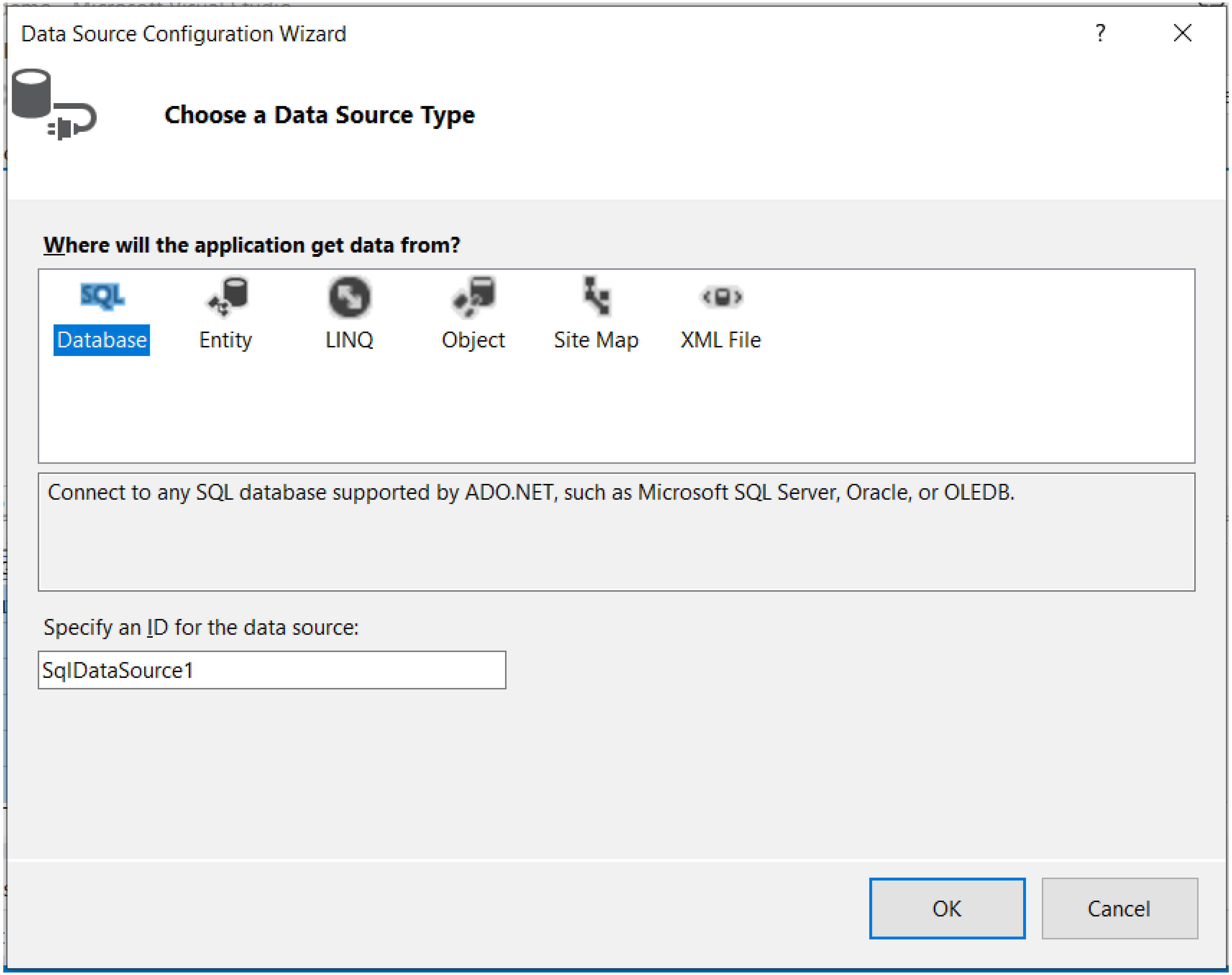
(None)
<New data source...>

← If you are working with the **GridView** then it is step 1 for you


asp:sqldatasource#SqlDataSource1
SqlDataSource - SqlDataSource1
SqlDataSource Tasks
Configure Data Source...

Configure your data source's settings.

← If you are working with the **SqlDataSource** then it is step 1 for you


Data Source Configuration Wizard

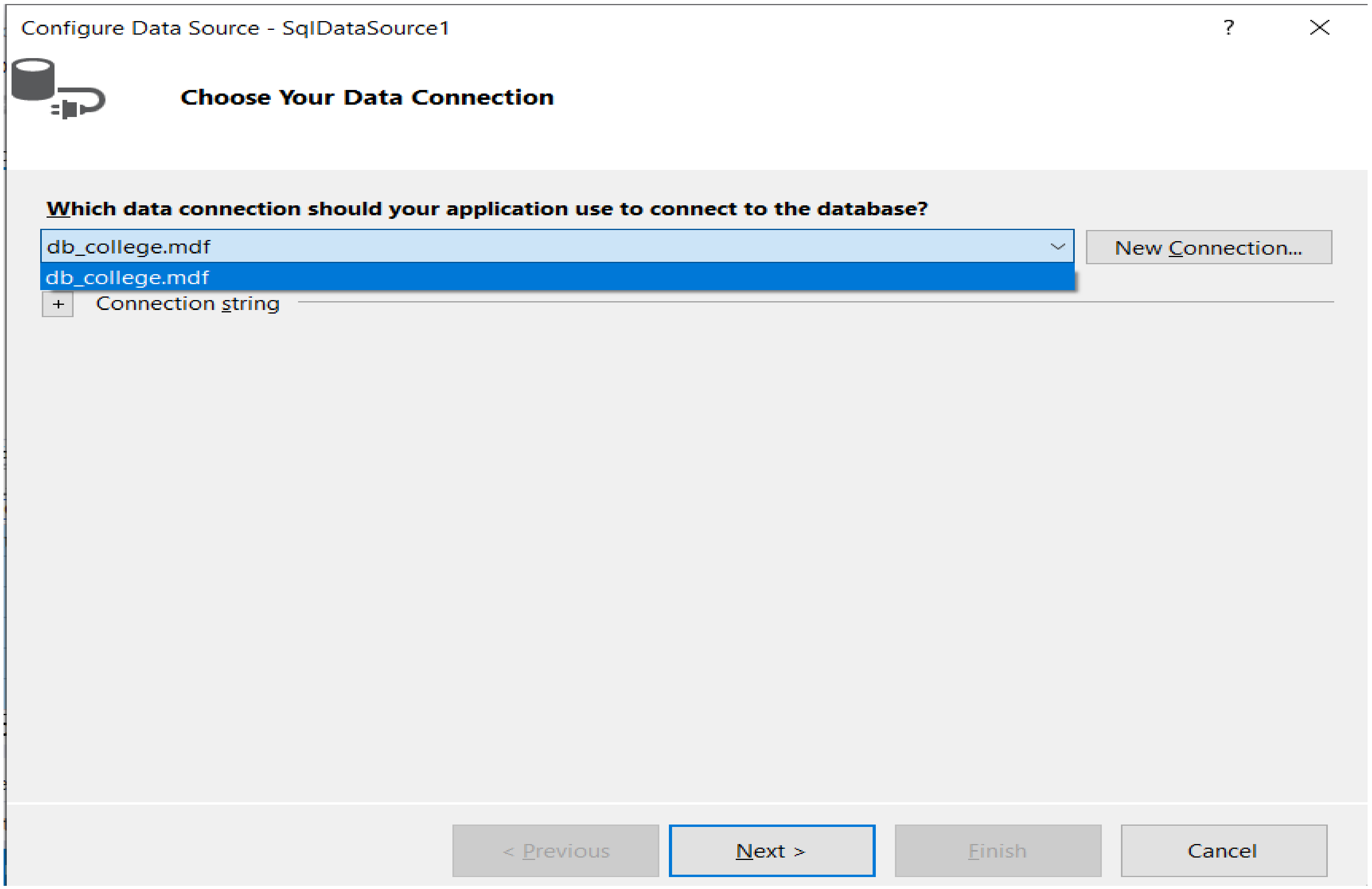
Choose a Data Source Type

Where will the application get data from?

Connect to any SQL database supported by ADO.NET, such as Microsoft SQL Server, Oracle, or OLEDB.

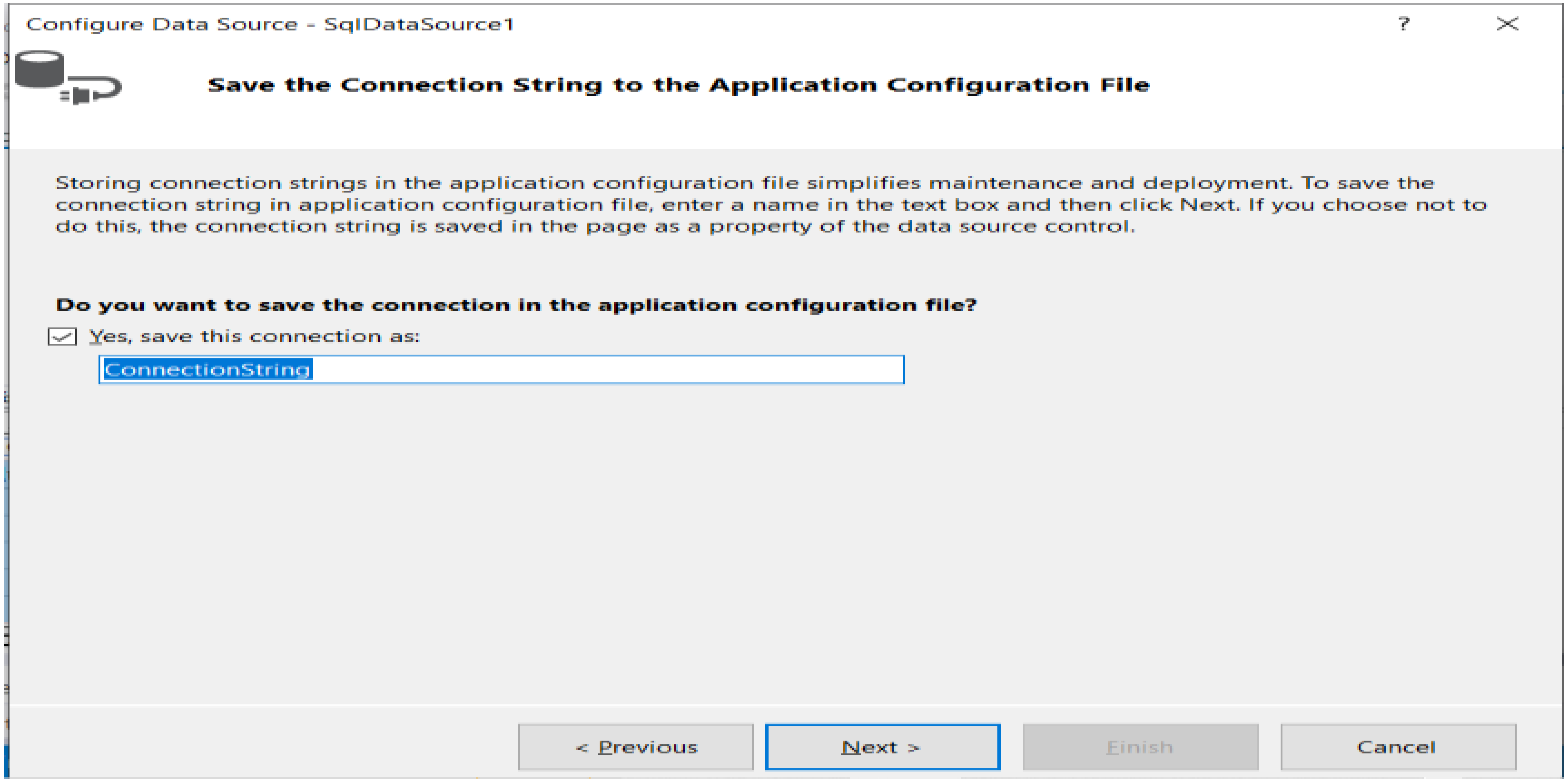
Specify an ID for the data source:

SqlDataSource1


Configure Data Source - SqlDataSource1
Choose Your Data Connection
Which data connection should your application use to connect to the database?
db_college.mdf
db_college.mdf
New Connection...
Connection string
< Previous
Next >
Finish
Cancel

- Click on Connection String (Copy):

DataSource=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\db_college.mdf;Integrated Security=True


Configure Data Source - SqlDataSource1

Save the Connection String to the Application Configuration File

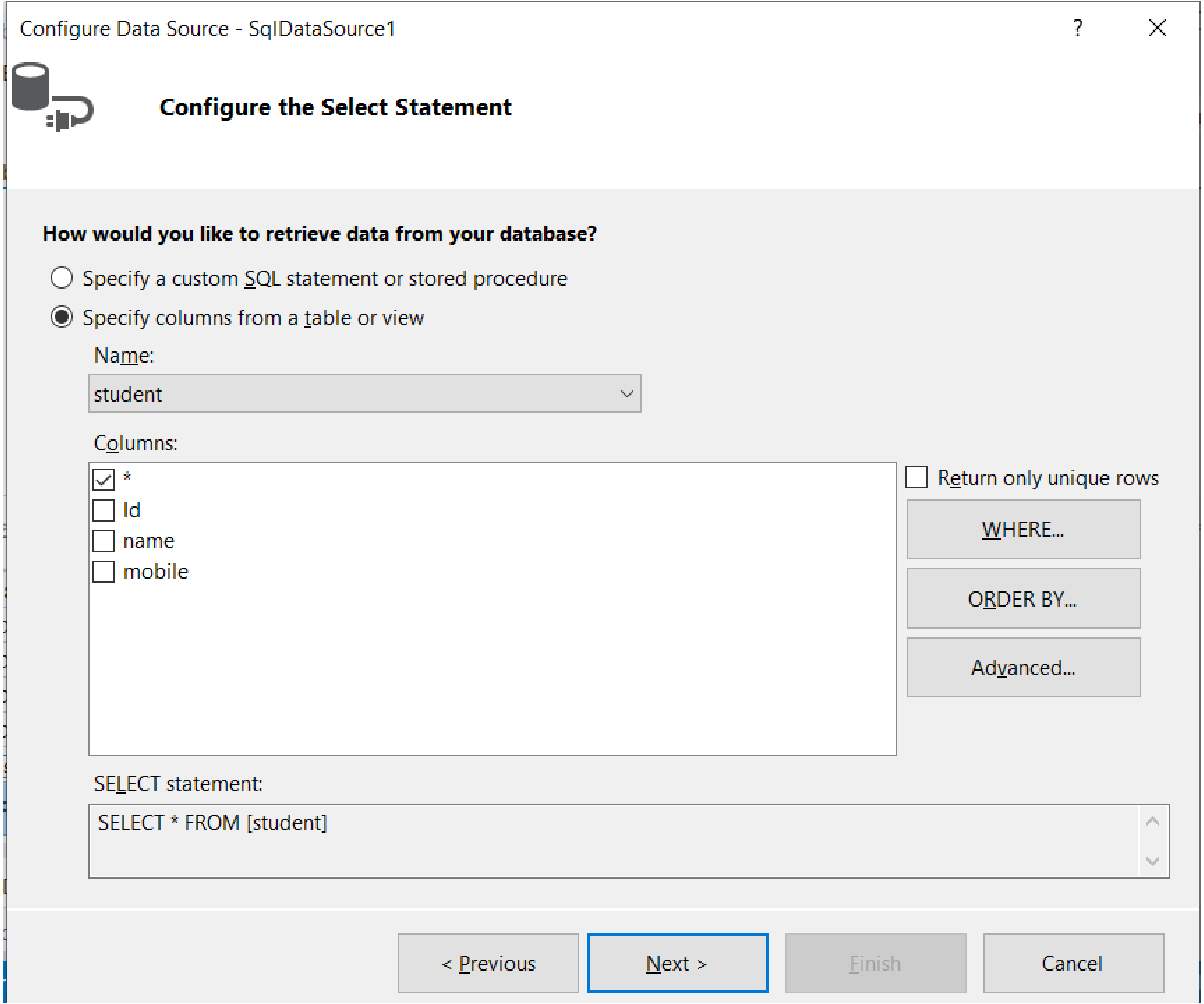
Storing connection strings in the application configuration file simplifies maintenance and deployment. To save the connection string in application configuration file, enter a name in the text box and then click Next. If you choose not to do this, the connection string is saved in the page as a property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

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

✓ Yes, save this connection as:

ConnectionString

- You can change Connection Name like : (college_string)



Configure Data Source - SqlDataSource1

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: student

Columns:

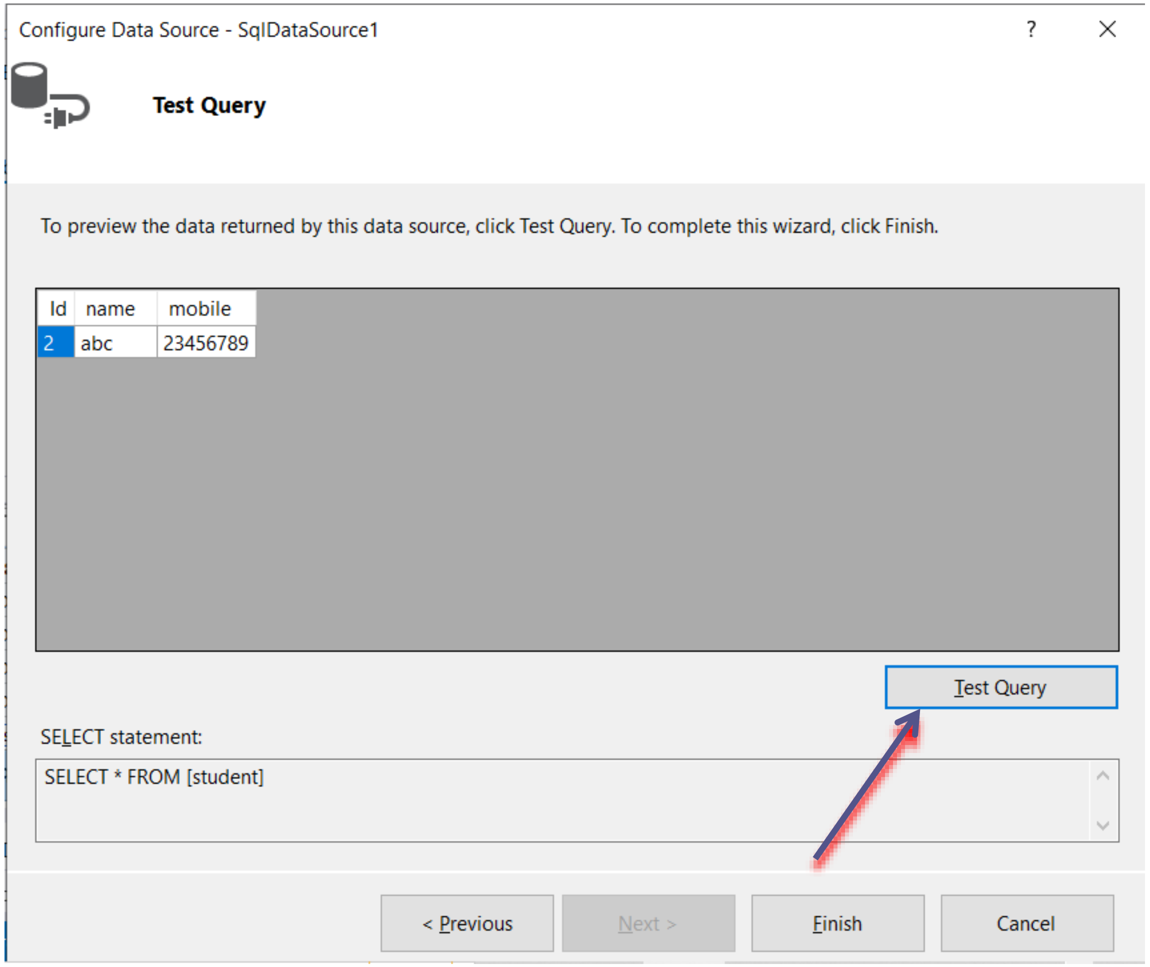
- [x] *
- [ ] Id
- [ ] name
- [ ] mobile

[ ] Return only unique rows

SELECT statement:

```
SELECT * FROM [student]
```

- Now click on test Query to check the result.



Configure Data Source - SqlDataSource1

Test Query

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

Id	name	mobile
2	abc	23456789

SELECT statement:

```
SELECT * FROM [student]
```

- **Now Your Code Like This :**

```
<div>
        <asp:GridView ID="GridView1" runat="server"
    AutoGenerateColumns="False" DataKeyNames="Id"
    DataSourceID="SqlDataSource1">
            <Columns>
                <asp:BoundField DataField="Id" HeaderText="Id"
    InsertVisible="False" ReadOnly="True" SortExpression="Id" />
                <asp:BoundField DataField="name" HeaderText="name"
    SortExpression="name" />
                <asp:BoundField DataField="mobile" HeaderText="mobile"
    SortExpression="mobile" />
            </Columns>
        </asp:GridView>
        <asp:SqlDataSource ID="SqlDataSource1" runat="server"
    ConnectionString="<%$ ConnectionStrings:college_string %>"
    SelectCommand="SELECT * FROM [student]"></asp:SqlDataSource>
      </div>
```

• Web.config file When you add SqlDataSource :

```
<?xml version="1.0"?>
<!--
  For more information on how to configure your ASP.NET application, please visit
  http://go.microsoft.com/fwlink/?LinkId=169433
  -->

<configuration>

    <connectionStrings>
      <add name="college_string" connectionString="Data
   Source=(LocalDB)\v11.0;AttachDbFilename=|DataDirectory|\db_college.
   mdf;Integrated Security=True"
        providerName="System.Data.SqlClient" />
    </connectionStrings>
    <system.web>
      <compilation debug="true" targetFramework="4.5" />
      <httpRuntime targetFramework="4.5" />
    </system.web>

</configuration>
```

•Repeater Control :

- Repeater control is also used to display data in a tabular format. But most important thing to know about Repeater control is **it does not have any view**. It wont display the data in Tabular format like GridView or other tabular controls.
- You need to write customized code to display the data in tabular format in Repeater control. Repeater control gives you the functionality to display the data in a customized format as per your choice. You need to write HTML code of your choice which is adopted by Repeater control.
- When you place Repeater control on form, it looks as follows:

Repeater - Repeater1
Switch to source view to edit the control's templates

- You can read the instruction here "**Switch to source view to edit the control's templates**". This means it has no specific view. You need to create its view by using templates of Repeater control.

- Templates supported by Repeater Control are as follows:

1. **<HeaderTemplate>:** This template is used to specify headings of Repeater control. This template is rendered (executed) only once before displaying any row. You can not bind any data inside this template.
2. **<ItemTemplate>:** This template is used to specify formatting for data being displayed. This is used to format rows which will be displayed under Repeater Control. This template is called each time the new row comes under Repeater Control. If<AlternatingItemTemplate> is not used, all the rows of Repeater Control takes format of this template.
3. **<AlternatingItemTemplate>:** This template is used to specify alter formatting for each alternate row of Repeater control. This is also used to format rows which will be displayed under Repeater Control. If you specify this template, all odd rows i.e. 1,3,5, etc. comes under format of <ItemTemplate> and all even rows i.e. 2,4,6, etc. comes under format of <AlternatingItemTemplate>.
4. **<SeperatorTemplate>:** This template is used to specify format to separate each row of Repeater Control. This format comes between rows of Repeater Control as this is used to separate rows,
5. **<FooterTemplate>:** This template comes as footer in Repeater control. As the name shows it comes after all the data is displayed under Repeater control. This template also can't contain any data bound information.

Example :

```
<form id="form1" runat="server">
  <div>
    <asp:Repeater ID="Repeater1"
  runat="server">
        <HeaderTemplate>
          <table border="1">
            <tr>
              <td>Id</td>
              <td>Name</td>
              <td>Mobile</td>
            </tr>
        </HeaderTemplate>
        <ItemTemplate>
          <tr>
            <td>
              <%#Eval("Id") %>
            </td>
            <td>
              <%#Eval("name") %>
            </td>
            <td>
              <%#Eval("mobile") %>
            </td>
          </tr>
        </ItemTemplate>
        <FooterTemplate>
          </table>
        </FooterTemplate>
    </asp:Repeater>

    <asp:SqlDataSource
ID="SqlDataSource1"
runat="server"
ConnectionString="<%$
ConnectionStrings:ConnectionStri
ng %>" SelectCommand="SELECT
* FROM [demo]">
</asp:SqlDataSource>
</div>
</form>
```

Data Binding :

- Data binding is the process that establishes a connection between the app UI and the data it displays.
- DataBinding is the prime requirement of all the web programmers today.
- Data Binding simply means you are connecting a control with any of the table or table column or data of a particular row.
- In technical words, Data binding is the process of retrieving data from a source and dynamically associating it to a property of a visual element.
- Depending on the context in which the element will be displayed, you can map the element to either an HTML tag or a .NET Web control.
- In ASP.NET when you bind the data with any of the control, it retrieves (gives) you the data from specified data source. Data Source can be anything.

- It can be DataSet, DataTable, DataRow, etc. But in most of the case Data Source is either DataSet or DataTable.
- When you bind any control with some specific DataSet or DataTable, the control is know as Bound Control as it is bounded to particular column or table.
- For example,

  [1] When you connect any table with GridView or DataGrid its called Data Binding.

  [2] When you connect any column with TextBox or Label control, its called Data Binding.
- There are two types of Data Binding methods.

  1. **Simple Data Binding**
  2. **Complex Data Binding**

- **[1] Simple Data Binding:**
- When you are connected to any single piece of information, in other words when you are bounded to a single column of single row, its called Simple Data Binding.
- For example, You have a TextBox or Label which is connected to a single column of table, its called Simple Data Binding.
- **Example :**

```
        <asp:TextBox ID="txt1" runat="server"
        Text="<%#name %>"></asp:TextBox>
```

- **cs file code :**

```
public string name="TYBCA";
protected void Page_Load(object sender, EventArgs e)
   {
        DataBind();
   }
```

- **[2] Complex Data Binding** or **Declarative Data Binding :**
- When you are connected to any single column of a table or with entire table itself is bounded to any control, its called Complex Data Binding.
- For example, you have bounded a Drop Down List with one Column of Table or you have bounded a Grid View or Data Grid with some table, its called **Complex Data Binding** or **declarative data binding**.
- Controls like DropDownList, CheckBoxList, RadioButtonList are known as Complex Data Bind controls which can be bounded to a particular column.
- They can show only single column of a table.
- Controls like Data Grid, Grid View or Repeater are known as Complex Data Bind Controls which can be bounded to entire table. They can show entire table.
- **Examples :**

```
GridView1.DataSource = dt;
GridView1.DataBind();
```

- Above example, uses Grid View control to bind the data.
- By specifying "DataSource" property you have bounded "dt" DataTable with Grid View Control.
- GridView control is capable of showing entire table, no need to specify any "DataTextField" property.
- In this example, entire table inside "dt" DataTable will be displayed after data is bounded.

Example of dropdown list using complex data binding :

```
<asp:DropDownList
   ID="DropDownList1"
   runat="server"
   AutoPostBack="true" >
      </asp:DropDownList>
<asp:Button ID="btn"
   runat="server" Text="click"
   OnClick="btn_Click" />
```

```
protected void btn_Click(object
   sender, EventArgs e)
   {
      SqlDataAdapter ad = new
   SqlDataAdapter("select *from
   demo", con);
      DataSet ds = new DataSet();
      ad.Fill(ds);
   DropDownList1.DataSource
   = ds;
   DropDownList1.DataTextFiel
   d = "name";
      DropDownList1.DataBind();
   }
```

Data Binding Expression :

- Data-binding expressions are contained within <%# and %> delimiters and use the Eval and Bind functions.
- The Eval function is used to define one-way (read-only) binding.
- The Bind function is used for two-way (updatable) binding.
- In addition to calling Eval and Bind methods to perform data binding in a data-binding expression, you can call any publicly scoped code within the <%# and %> delimiters to execute that code and return a value during page processing.

Eval Method :

- The Eval method evaluates late-bound data expressions in the templates of data-bound controls such as the GridView, DetailsView, and FormView controls.
- At run time, the Eval method calls the Eval method of the DataBinder object, referencing the current data item of the naming container.
- The naming container is generally the smallest part of the data-bound control that contains a whole record, such as a row in a GridView control.
- You can therefore use the Eval method only for binding inside templates of a data-bound control.
- **Example :**

```
<table>
 <tr>
     <td align="right"><b>Product ID:</b></td>
     <td><%# Eval("ProductID") %></td>
</tr>
</table>
```

Bind Method :

- The Bind method has some similarities to the Eval method, but there are significant differences.
- Although you can retrieve the values of data-bound fields with the Bind method, as you can with the Eval method, the Bind method is also used when data can be modified.
- **Example :**

```
<EditItemTemplate>
        <table>
            <tr>
               <td align=right> <b>First Name:</b> </td>
               <td> <asp:TextBox ID="EditFirstNameTextBox"
RunAt="Server" Text='<%# Bind("FirstName") %>' /> </td>
           </tr>
        </table>
</EditItemTemplate>
```

Some Questions for Exam :

- Full form of ODBC
- Full form of OLEDB
- Full form of ADO
- Full form of DAO
- Explain Connection string in detail.
- Explain Data Provider in detail.
- What is ADO.net ?
- Explain Connection Class in detail.
- Explain Command Class in detail.
- Explain Adapter Class.
- Write a example of Reading a data from database using DataReader Class.
- Explain GridView in detail.