ADO.NET & Database

UNIT - 4

ADO.NET

- ADO.NET (ActiveX Data Object for .NET) is an object-oriented set of libraries that allows you to interact with data sources.

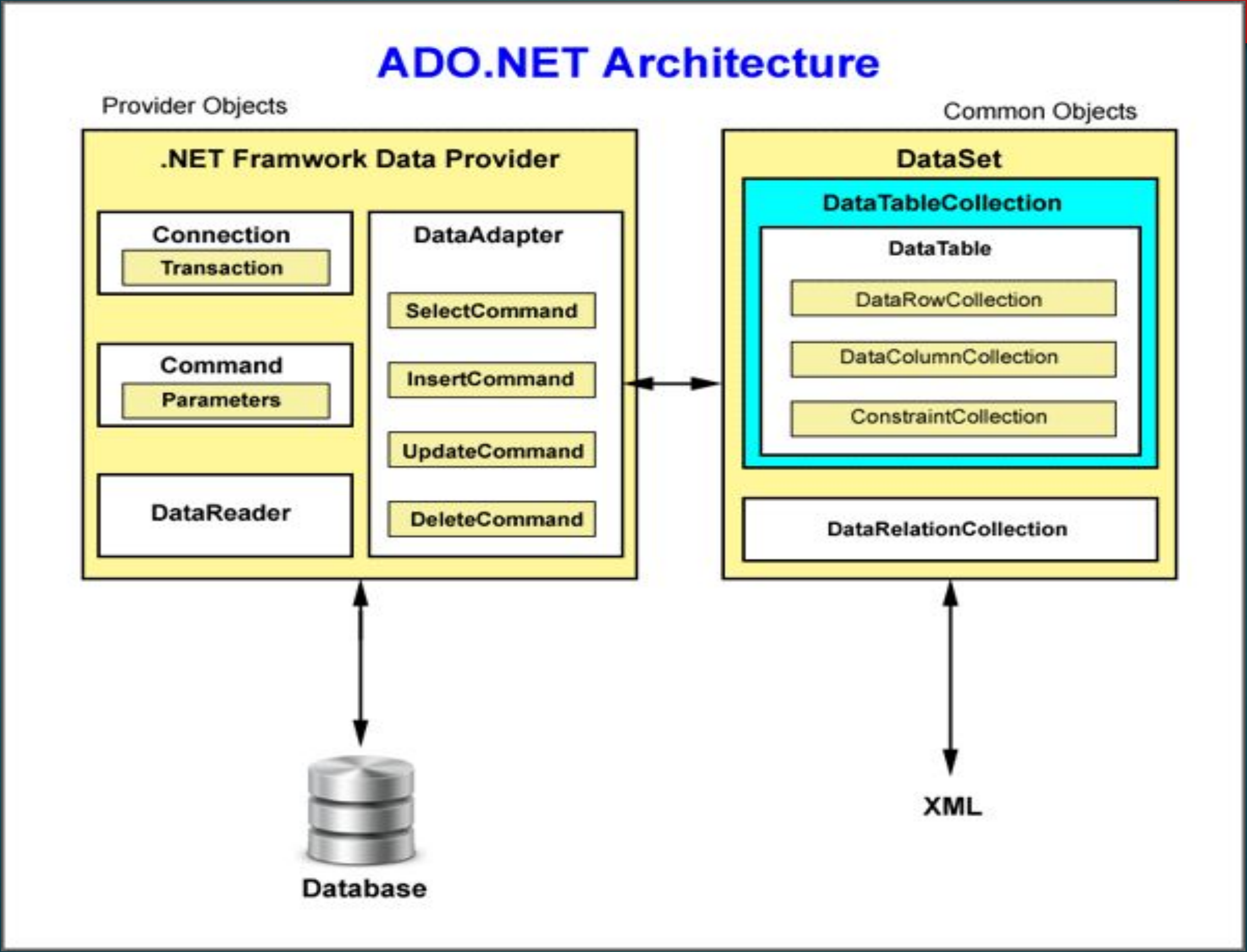
- Commonly, the data source is a database, but it could also be a text file, an Excel spreadsheet, or an XML file.

- It is a part of the base class library that is included with the Microsoft .NET Framework.

- It is commonly used by programmers to access and modify data stored in relational database systems, though it can also access data in non-relational sources.

- ASP.NET Websites / Web Applications are uses ADO.NET to communicate with Database System.

ADO.net Architecture



ADO.net Architecture

- ADO.NET uses a multilayer architecture that mainly has a few concepts, for instance Connection, Reader, Command, Adapter and Dataset objects.
- ADO.NET introduced data providers that are a set of special classes to access a specific database, execute SQL commands and retrieve data.
- The Data providers are extensible. Developers can create their own providers for a proprietary data source.
- There are some examples of data providers such as SQL Server providers, OLE DB and Oracle provider.

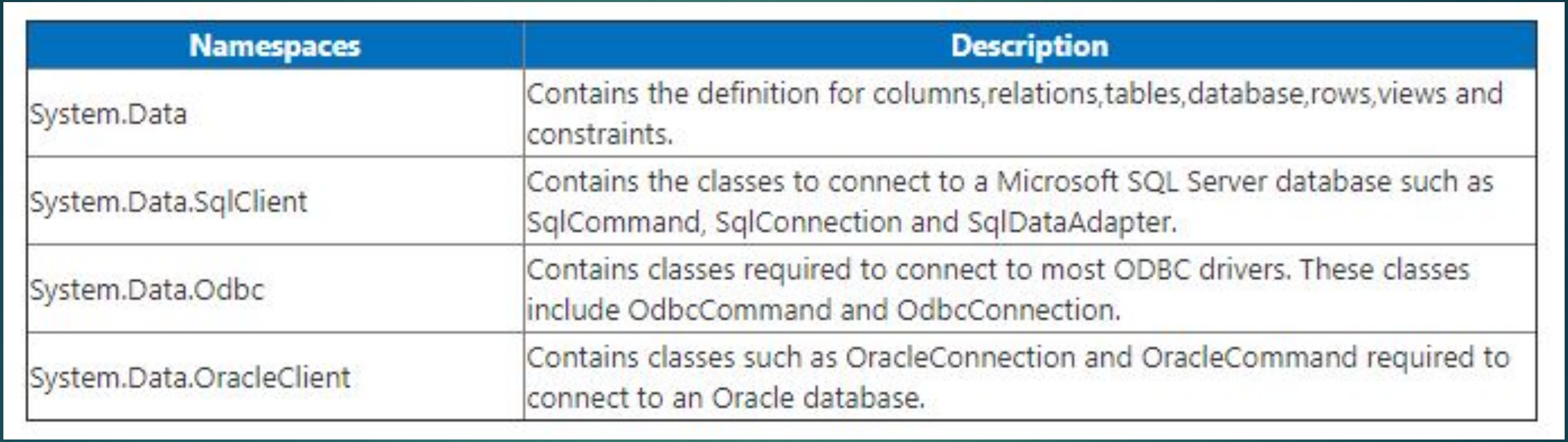
ADO.net Architecture

ADO.NET provides the following two types of classes objects:

- **Connection-based:** They are the data provider objects such as Connection, Command, DataAdapter and DataReader. They execute SQL statements and connect to a database.

- **Content-based:** They are found in the System.Data namespace and includes DataSet, DataColumn, DataRow and DataRelation. They are completely independent of the type of data source.

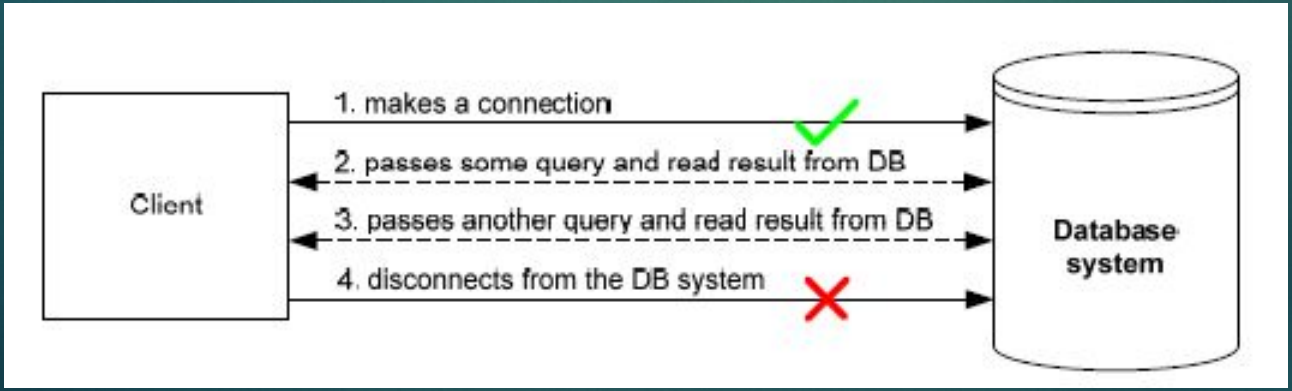
ADO.NET Namespaces



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.

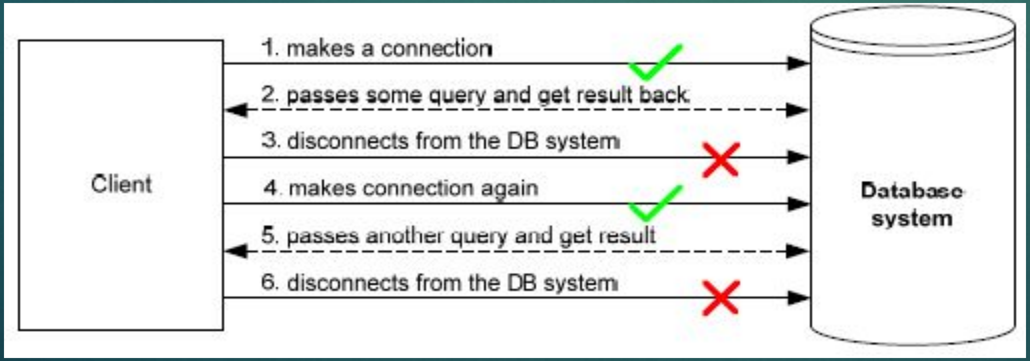
Connected architecture

- In connected architecture ,you made a connection to the database system and then interacted with it through SQL queries using connection.
- The application stays connected to the DB system even when it is not using DB services.
- This commonly wastes valuable and expensive database resources, as most of the time applications only query and view the persistent data.
- We read data from database using DataReader object.
- Connected architecture is read only, we cant update the data.


Client
1. makes a connection
2. passes some query and read result from DB
3. passes another query and read result from DB
4. disconnects from the DB system
Database system

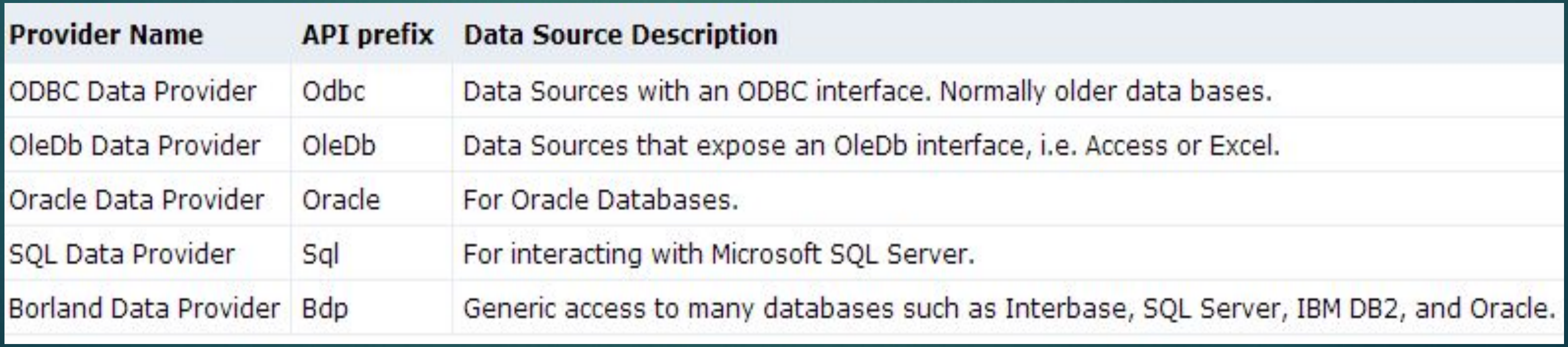
Disconnected architecture

- ADO.NET solves connected architecture problem by managing a local buffer of persistent data called dataset.
- Application automatically connects to the database server when it needs to run a query and then disconnects immediately after getting the result back and storing in dataset.
- This design of ADO.NET is called a disconnected data architecture.
- It maintains a local repository of data in the dataset object.
- User can perform insert, update and delete operations.

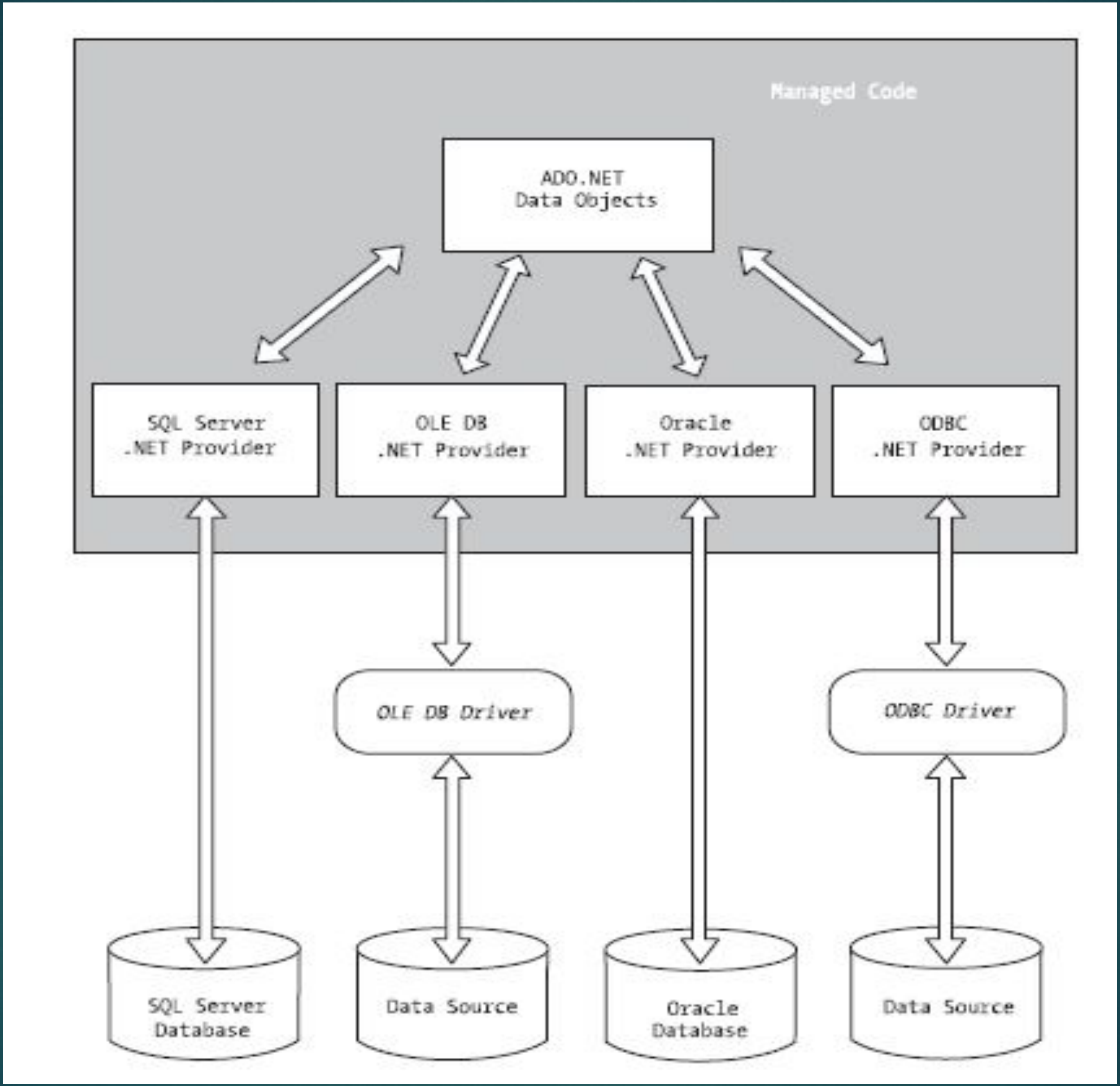

Client
1. makes a connection
2. passes some query and get result back
3. disconnects from the DB system
4. makes connection again
5. passes another query and get result
6. disconnects from the DB system
Database system

Data Providers

- ADO.NET allows us to interact with different types of data sources and different types of databases. However, there isn't a single set of classes that allow you to accomplish this universally.
- Since different data sources expose different protocols, there are more data sources every day that allow you to communicate with them directly through .NET ADO.NET class libraries.
- These libraries are called Data Providers and are usually named for the protocol or data source type they allow you to interact with.

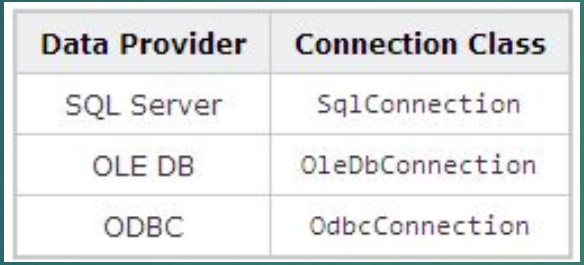


Provider Name	API prefix	Data Source Description
ODBC Data Provider	Odbc	Data Sources with an ODBC interface. Normally older data bases.
OleDb Data Provider	OleDb	Data Sources that expose an OleDb interface, i.e. Access or Excel.
Oracle Data Provider	Oracle	For Oracle Databases.
SQL Data Provider	Sql	For interacting with Microsoft SQL Server.
Borland Data Provider	Bdp	Generic access to many databases such as Interbase, SQL Server, IBM DB2, and Oracle.


Managed Code
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

Connection Object

- Each data provider in ADO.NET contains a Connection class that inherits from the System.Data.Common.DbConnection class.
- The DbConnection serves as the base class for all the Connection classes of different data providers.

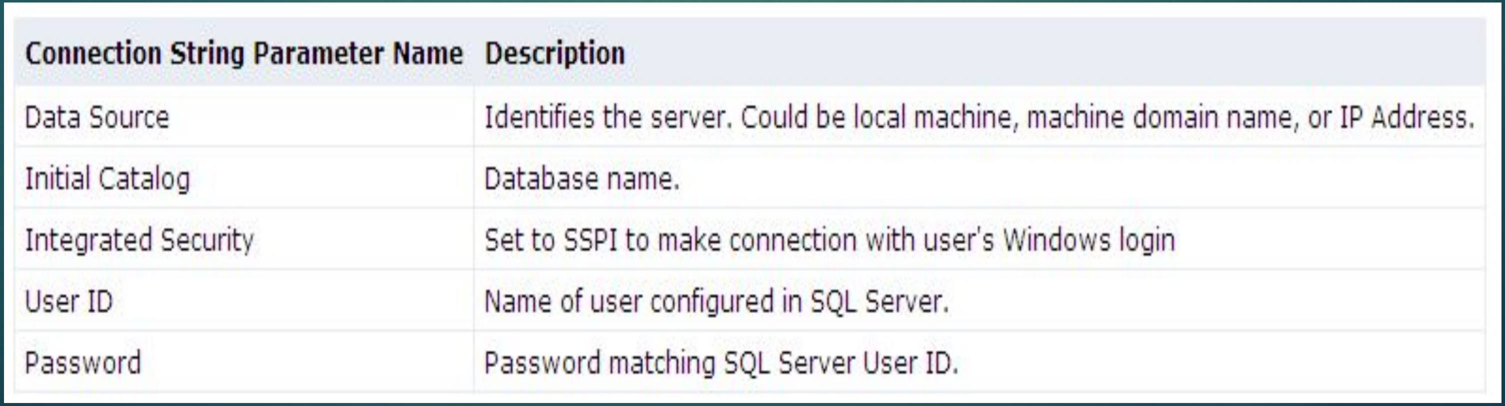


Data Provider	Connection Class
SQL Server	SqlConnection
OLE DB	OleDbConnection
ODBC	OdbcConnection

- To interact with a database, you must have a connection to it. The connection helps identify the database server, the database name, user name, password, and other parameters that are required for connecting to the data base.

Creating a SqlConnection Object

- A SqlConnection is an object, just like any other C# object. Most of the time, you just declare and instantiate the SqlConnection all at the same time, as shown below:
- SqlConnection conn = new SqlConnection("Data Source=(local);Initial Catalog=Northwind;Integrated Security=SSPI");
- The SqlConnection object instantiated above uses a constructor with a single argument of type string This argument is called a connection string.



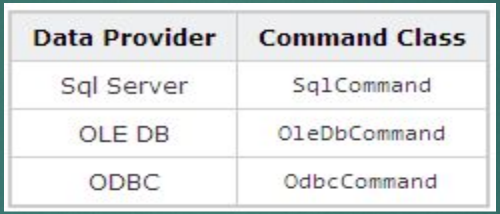
Connection String Parameter Name	Description
Data Source	Identifies the server. Could be local machine, machine domain name, or IP Address.
Initial Catalog	Database name.
Integrated Security	Set to SSPI to make connection with user's Windows login
User ID	Name of user configured in SQL Server.
Password	Password matching SQL Server User ID.

Using a SqlConnection

- The purpose of creating a SqlConnection object is so you can enable other ADO.NET code to work with a database.
- Other ADO.NET objects, such as a SqlCommand and a SqlDataAdapter take a connection object as a parameter. The sequence of operations occurring in the lifetime of a SqlConnection are as follows:
- Instantiate the SqlConnection.
- Open the connection.
- Pass the connection to other ADO.NET objects.
- Perform database operations with the other ADO.NET objects.
- Close the connection.

Command

- Each data provider has their Command class which is use to execute SQL commands or stored procedures to the database.
- Each Command class inherits from the System.Data.Common.DbCommand base class.
- The following are the different flavors of the Command class for each data provider.



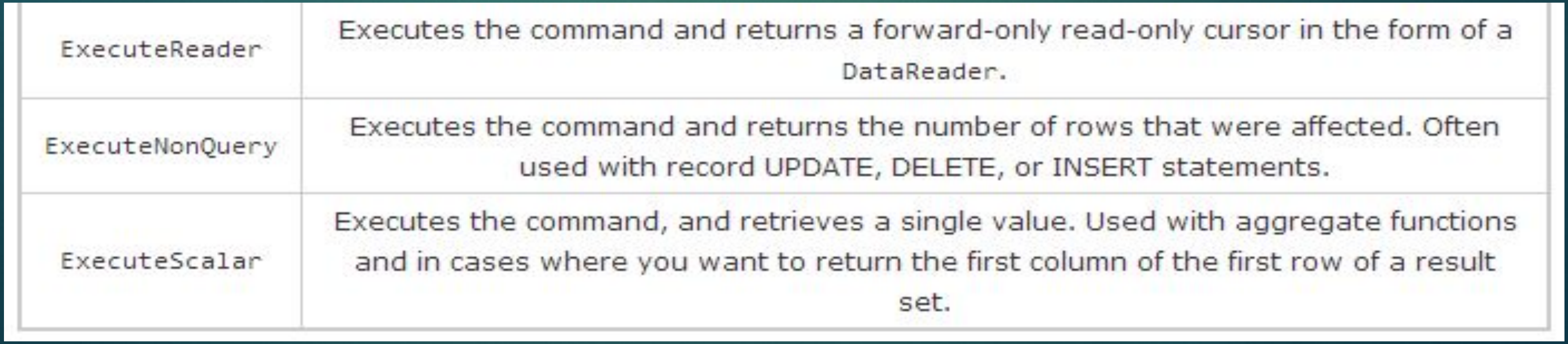
Data Provider	Command Class
Sql Server	SqlCommand
OLE DB	OleDbCommand
ODBC	OdbcCommand

- SqlCommand object allows you to specify what type of interaction you want to perform with a database.
- SqlCommand object can be used to support disconnected architecture.
- For example, you can do select, insert, modify, and delete commands on rows of data in a database table.

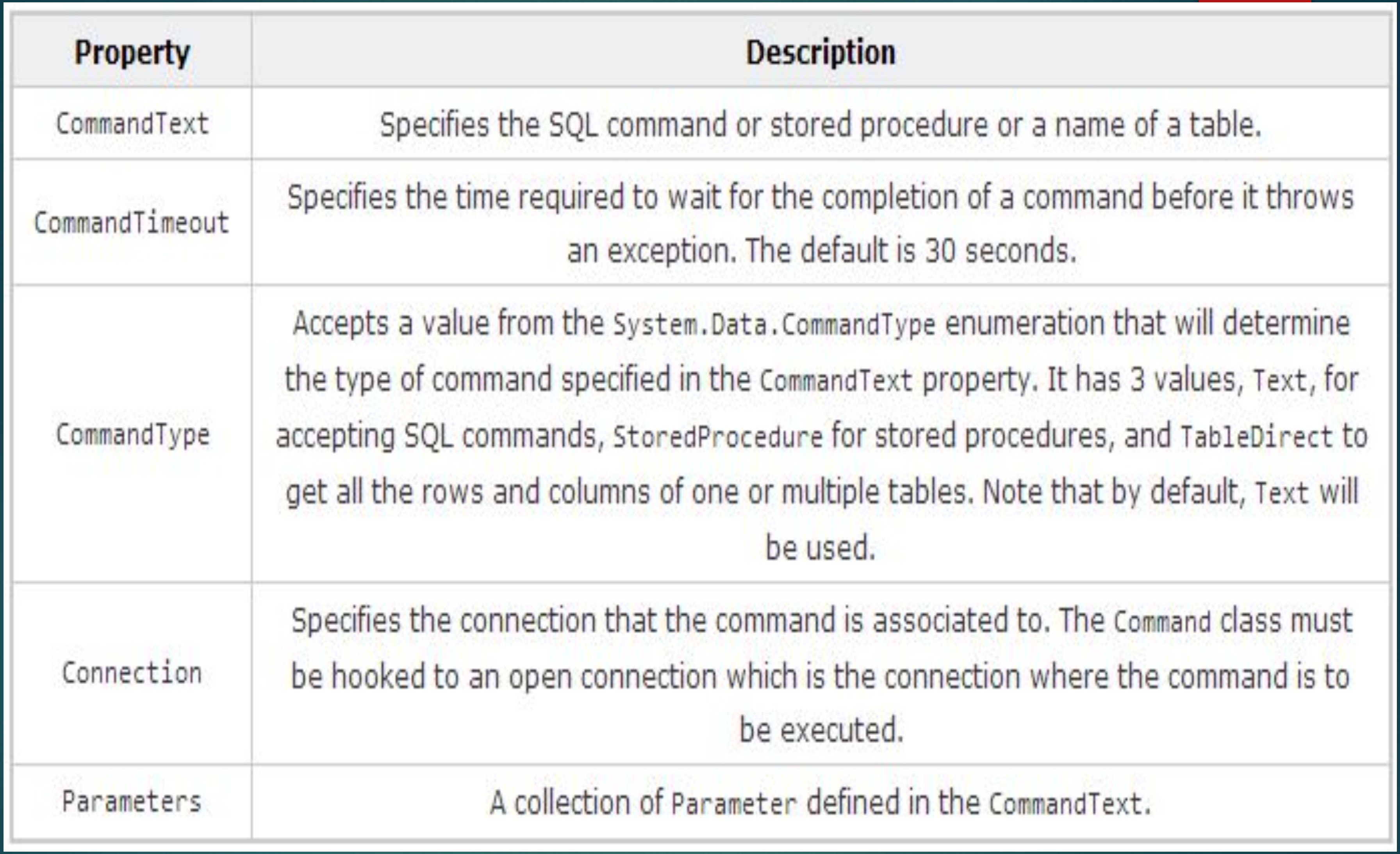
Creating a SqlCommand Object

- Similar to other C# objects, you instantiate a SqlCommand object via the new instance declaration, as follows:
- SqlCommand cmd = new SqlCommand

  ("select CategoryName from Categories", conn);
- for instantiating a SqlCommand object. It takes a string parameter that holds the command you want to execute and a reference to a SqlConnection object.
- The following are important built in methods uses in the Command Object to execute the SQL statements.



ExecuteReader	Executes the command and returns a forward-only read-only cursor in the form of a DataReader.
ExecuteNonQuery	Executes the command and returns the number of rows that were affected. Often used with record UPDATE, DELETE, or INSERT statements.
ExecuteScalar	Executes the command, and retrieves a single value. Used with aggregate functions and in cases where you want to return the first column of the first row of a result set.



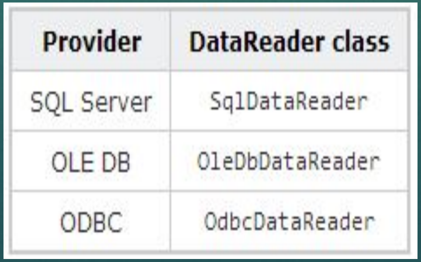
Property	Description
`CommandText`	Specifies the SQL command or stored procedure or a name of a table.
`CommandTimeout`	Specifies the time required to wait for the completion of a command before it throws an exception. The default is 30 seconds.
`CommandType`	Accepts a value from the `System.Data.CommandType` enumeration that will determine the type of command specified in the `CommandText` property. It has 3 values, `Text`, for accepting SQL commands, `StoredProcedure` for stored procedures, and `TableDirect` to get all the rows and columns of one or multiple tables. Note that by default, `Text` will be used.
`Connection`	Specifies the connection that the command is associated to. The `Command` class must be hooked to an open connection which is the connection where the command is to be executed.
`Parameters`	A collection of `Parameter` defined in the `CommandText`.

Data Reader

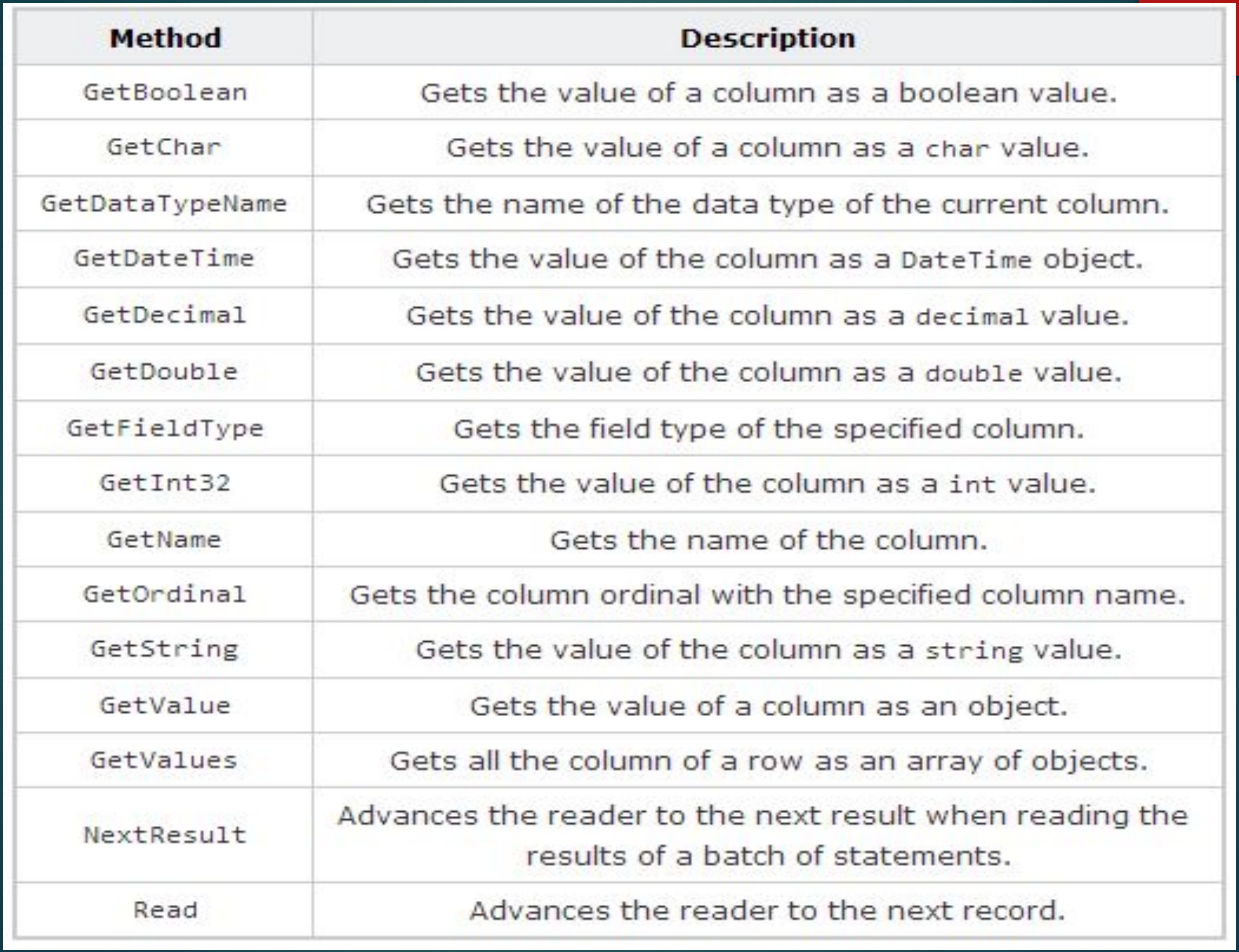
- DataReader object allows forward-only, read-only access to a database.
- Using DataReader is the connected way of accessing data and an open connection must be available first.
- Each provider has its own version of DataReader which inherits to the System.Data.Common.DbDataReader base class.
- DataReader cannot be created directly from code, they can created only by calling the **ExecuteReader** method of a Command Object.

  SqlDataReader sqlReader = sqlCmd.ExecuteReader();
- **Connection Object** can contain only one DataReader at a time and the connection in the DataReader remains open, also it cannot be used for any other purpose while data is being accessed.
- **Read()** method in the DataReader is used to read the rows from DataReader and it always moves forward to a new valid row, if any row exist .

  sqlReader.Read();



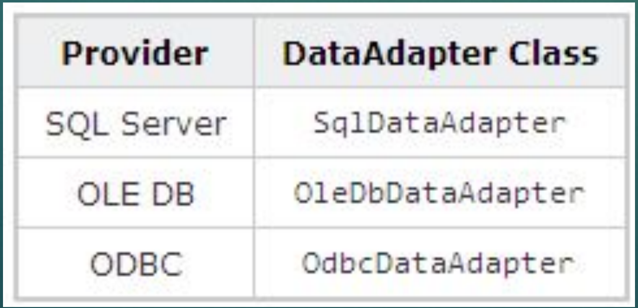
Provider	DataReader class
SQL Server	SqlDataReader
OLE DB	OleDbDataReader
ODBC	OdbcDataReader



Method	Description
GetBoolean	Gets the value of a column as a boolean value.
GetChar	Gets the value of a column as a char value.
GetDataTypeName	Gets the name of the data type of the current column.
GetDateTime	Gets the value of the column as a DateTime object.
GetDecimal	Gets the value of the column as a decimal value.
GetDouble	Gets the value of the column as a double value.
GetFieldType	Gets the field type of the specified column.
GetInt32	Gets the value of the column as a int value.
GetName	Gets the name of the column.
GetOrdinal	Gets the column ordinal with the specified column name.
GetString	Gets the value of the column as a string value.
GetValue	Gets the value of a column as an object.
GetValues	Gets all the column of a row as an array of objects.
NextResult	Advances the reader to the next result when reading the results of a batch of statements.
Read	Advances the reader to the next record.

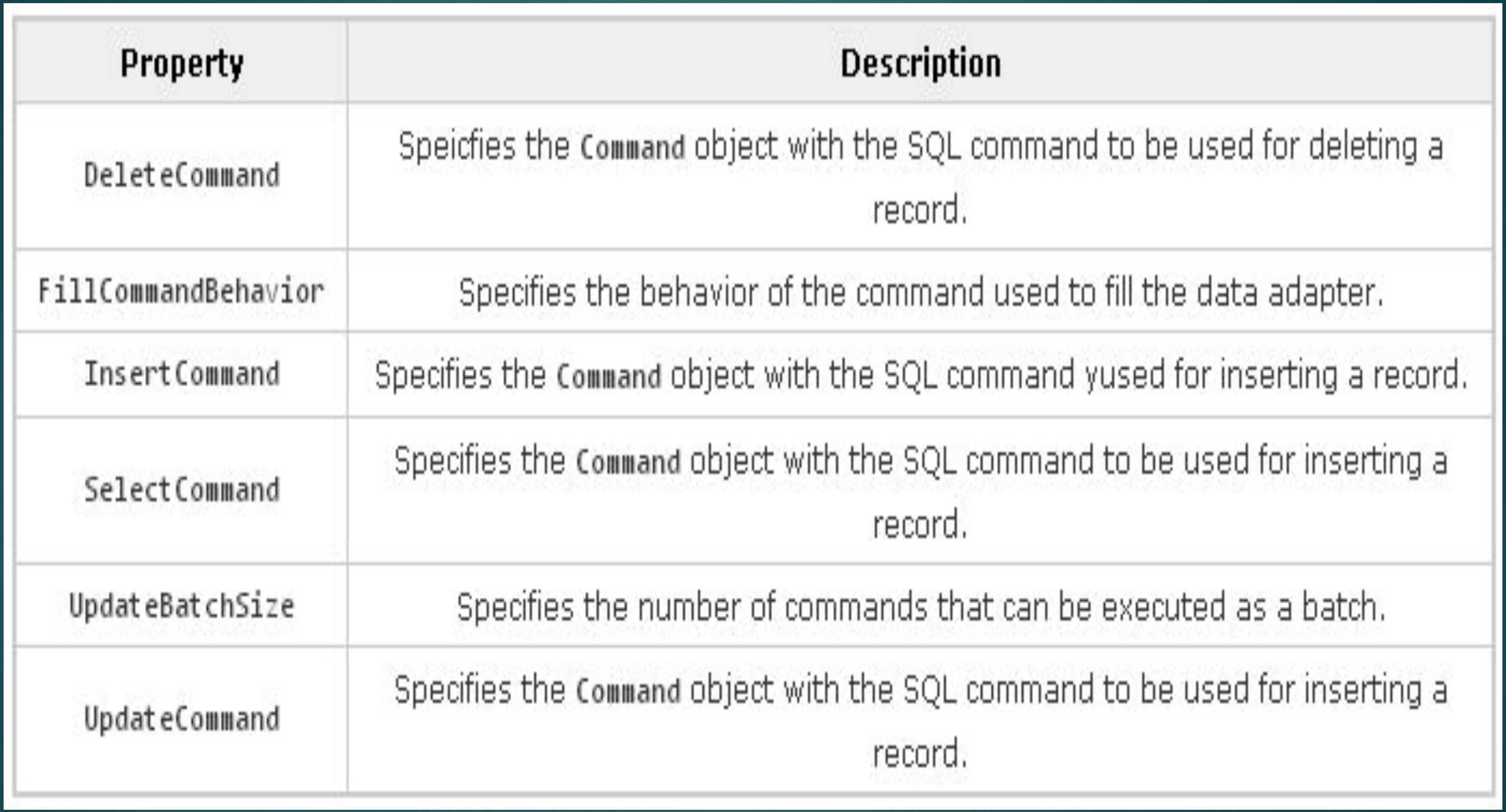
Data Adapter

- DataAdapter can be considered as a bridge between the actual data source to your application.
- It is commonly used together with a DataSet. Using DataAdapter and DataSet is the disconnected way of retrieving data from the data source.
- DataAdapter allows you to fill a DataSet with values from the data source, or execute different commands to the data source.
- DataAdapter class inherits from the System.Data.Common.DbDataAdapter base class.
- Each data provider has its own version of DataAdapter.



Provider	DataAdapter Class
SQL Server	SqlDataAdapter
OLE DB	OleDbDataAdapter
ODBC	OdbcDataAdapter

Data Adapter Properties



Property	Description
DeleteCommand	Speicfies the Command object with the SQL command to be used for deleting a record.
FillCommandBehavior	Specifies the behavior of the command used to fill the data adapter.
InsertCommand	Specifies the Command object with the SQL command yused for inserting a record.
SelectCommand	Specifies the Command object with the SQL command to be used for inserting a record.
UpdateBatchSize	Specifies the number of commands that can be executed as a batch.
UpdateCommand	Specifies the Command object with the SQL command to be used for inserting a record.

- The DataAdapter is the one that actually executes the commands to data source. It has a SelectCommandproperty which accepts a DbCommand object that specifies the SELECT statement used to retrieved data from the data source.
- The following shows you an example of assigning a SelectCommand.
- SqlCommand selectCommand = new SqlCommand("SELECT * FROM Students", connection);
- **SqlDataAdapter adapter = new SqlDataAdapter();** adapter.SelectCommand = selectCommand;
- To execute the command specified by the SelectCommand property, we can use the Fill() method of the DbDataAdapter class.
- The Fill() method requires an instance of the DataSet or DataTable classes. The following shows an example of filling a DataTable instance with values retrieved from the database.
- adapter.Fill(DataSet/DataTable);

Dataset

- System.Data.DataSet class holds data that are retrieved from the database.
- DataSet class allows you to hold disconnected data.
- **DataSet** contains **DataTableCollection** and their**DataRelationCollection** . It represents a complete set of data including the tables that contain, order, and constrain the data, as well as the relationships between the tables.
- Dataset contains more than one Table at a time. We can set up **Data Relations** between these tables within the DataSet. The data set may comprise data for one or more members, corresponding to the number of rows.
- **DataAdapter Object** allows us to populate **DataTables** in a DataSet. We can use Fill method of the DataAdapter for populating data in a Dataset. The DataSet can be filled either from a data source or dynamically.

Data Table

- In ADO.NET, DataTable objects are used to represent the tables in a **DataSet**.
- **DataTable** represents one table of in-memory relational data; the data is local to the .NET-based application in which it resides,
- DataTable is a relational database like table in the memory.
- It has a structural definition and constraints like unique constraints.
- We can create hierarchical relationships among many DataTables dynamically in a DataSet.
- To create DataTable,

```
DataTable myDataTable = new DataTable("Sample_Table");
```

DataColumn

- Stored in collection named columns and represents the schema of a column in a DataTable.

DataRow

- DataRow represents a row of data in a DataTable.
- You can add data to the table using DataRow Object.
- DataRowCollection object represents a collection of data rows of a table.
- Use DataTable's NewRow metgod to return a DataRow object of data table,Add values to the data row and add a row to the data table.

```
DataTable myDataTable = new DataTable("Sample_Table");

DataColumn myDataColumn  new DataColumn();

DataRow my DataRow = myDataTable.NewRow();
```

Data View

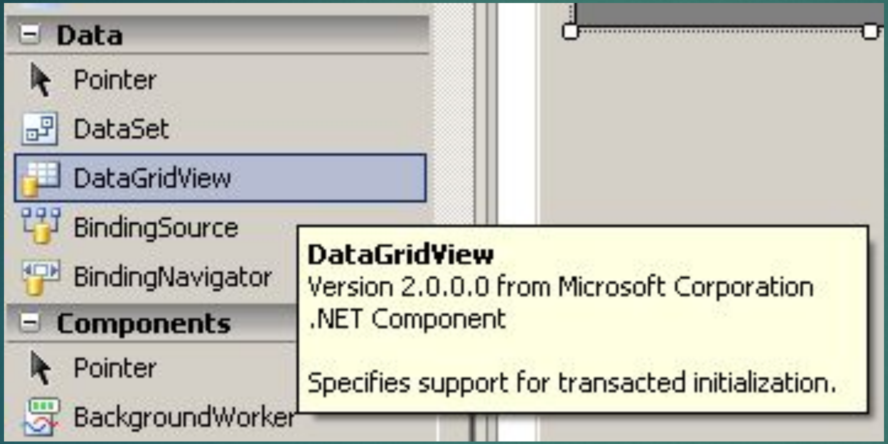
- **DataView** provides different views of the data stored in a DataTable. That is we can customize the views of data from a DataTable.
- DataView can be used to sort, filter, and search the data in a **DataTable** , additionally we can add new rows and modify the content in a DataTable.
- We can create DataView in two different ways. We can use the**DataView Constructor** , or you can create a reference to the **DefaultView Property** of the DataTable.
- The DataView constructor can be empty, or it can take either a DataTable as a single argument, or a DataTable along with filter criteria, sort criteria, and a row state filter.
- dv = new DataView(dt, “Filter“,”Sort”, DataViewRowState.CurrentRows);
- ***dv = dt.DefaultView;***

Data GridView

- DataGridView control is designed to be a complete solution for displaying tabular data with Windows Forms.
- DataGridView control is highly configurable and extensible, and it provides many properties, methods, and events to customize its appearance and behavior.
- DataGridView control makes it easy to define the basic appearance of cells and the display formatting of cell values.
- The cell is the fundamental unit of interaction for the DataGridView. All cells derive from the DataGridViewCell base class. Each cell within the DataGridView control can have its own style, such as text format, background color, foreground color, and font. Typically, however, multiple cells will share particular style characteristics.
- The data type for the cell's Value property by default is of type Object.

Using Data GridView

- Find DataGridView control from ToolBox under Data Tab.



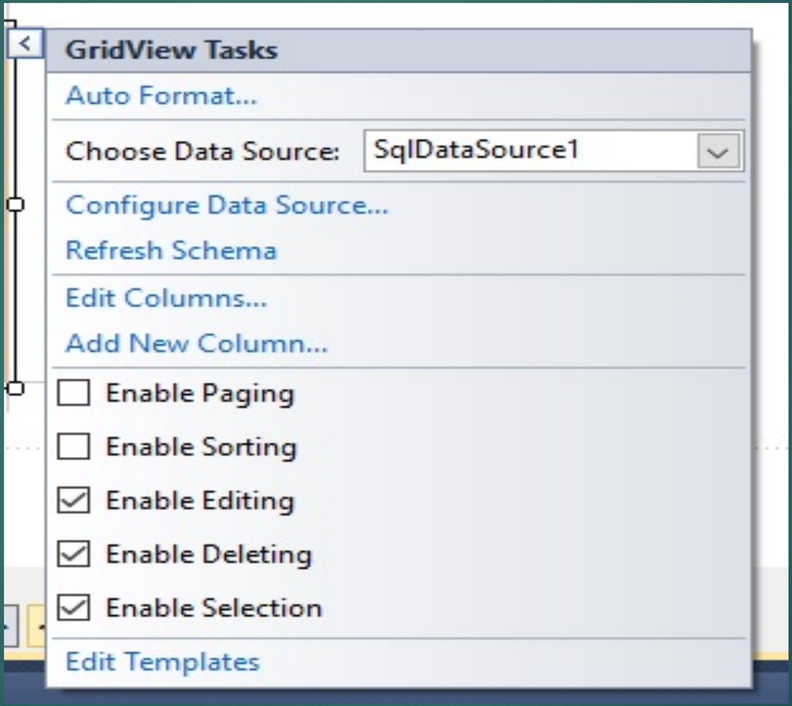
DataGridView
Version 2.0.0.0 from Microsoft Corporation
.NET Component

Specifies support for transacted initialization.

- You can bind data into DataGridView in twop different ways:
  - Using DataGridView Configuration Wizard
  - Dynamically by C# CODE

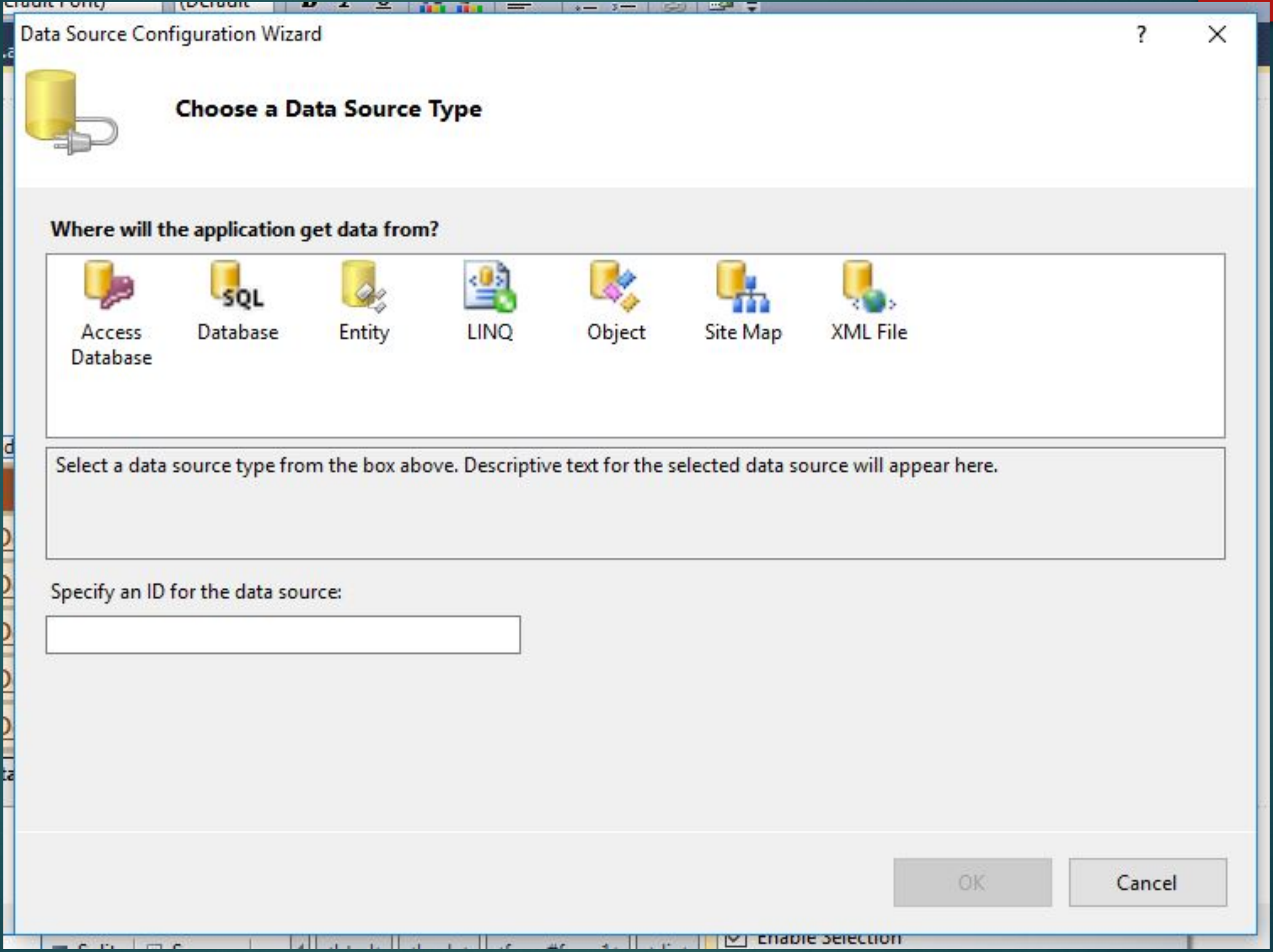
Using DataGridView Configuration Wizard

- Click on the DataSource property of DataGridView Control and click on Add Project DataSource



- You will get DataGridView Config Wizard Dialogbox.

- Select Database and click Next> button.


Data Source Configuration Wizard

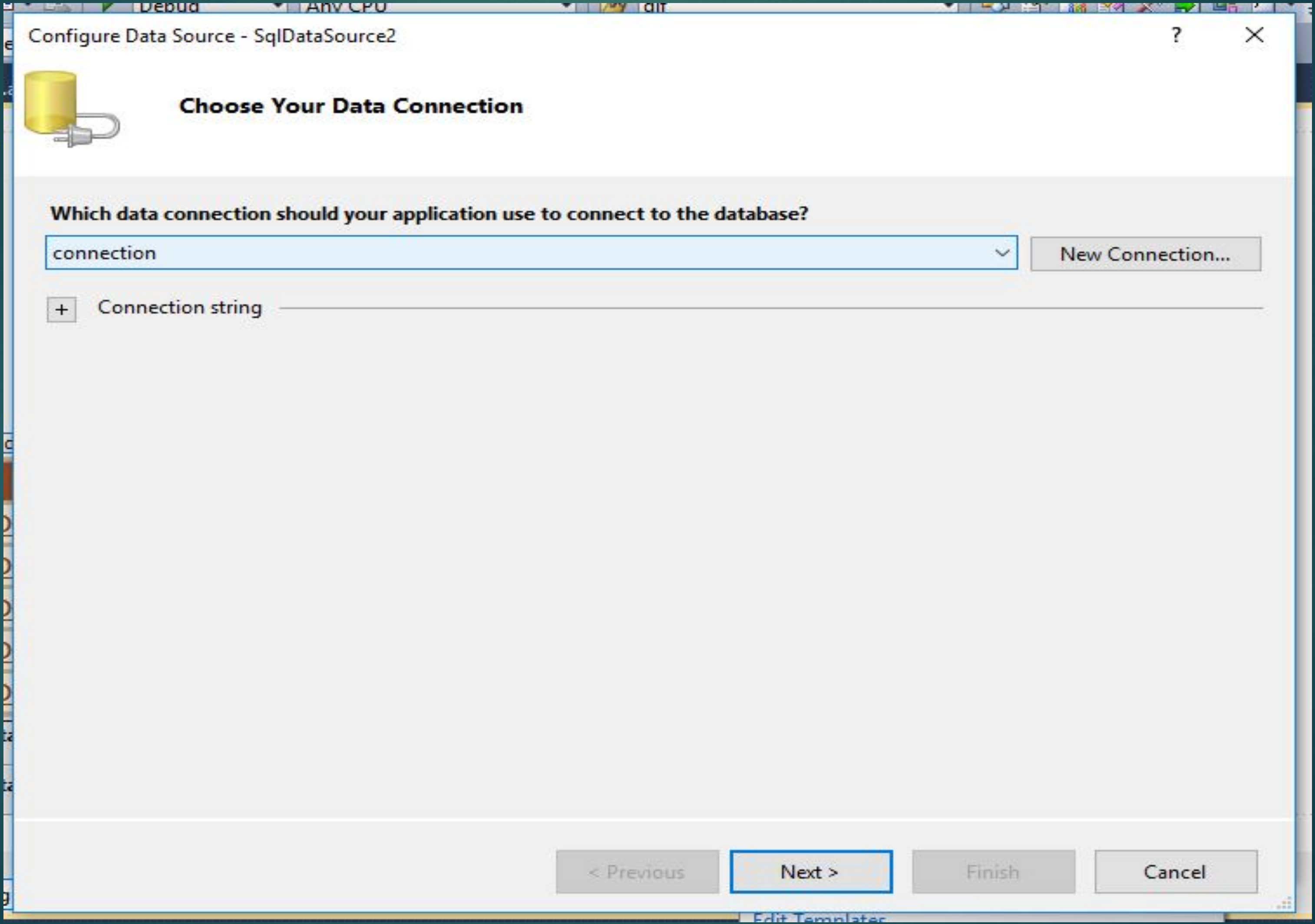
Choose a Data Source Type

Where will the application get data from?

Select a data source type from the box above. Descriptive text for the selected data source will appear here.

Specify an ID for the data source:

- Here you can create new database connection or you can select exsiting database connection. After selecting database conncetion click next.



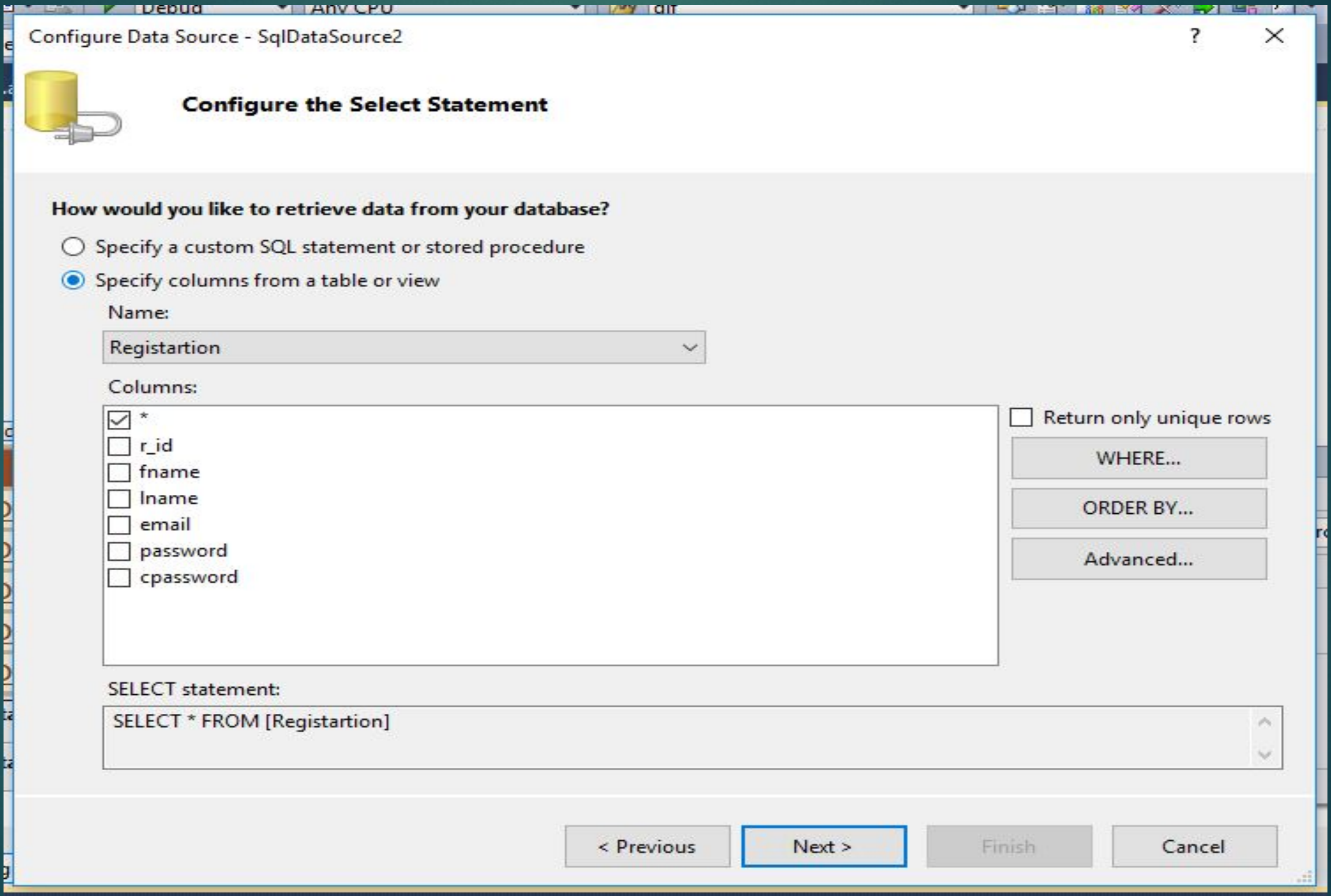
Configure Data Source - SqlDataSource2

Choose Your Data Connection

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

connection

- Then it will ask you to save connection string in application configration file or not.



Configure Data Source - SqlDataSource2

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: Registartion

Columns:

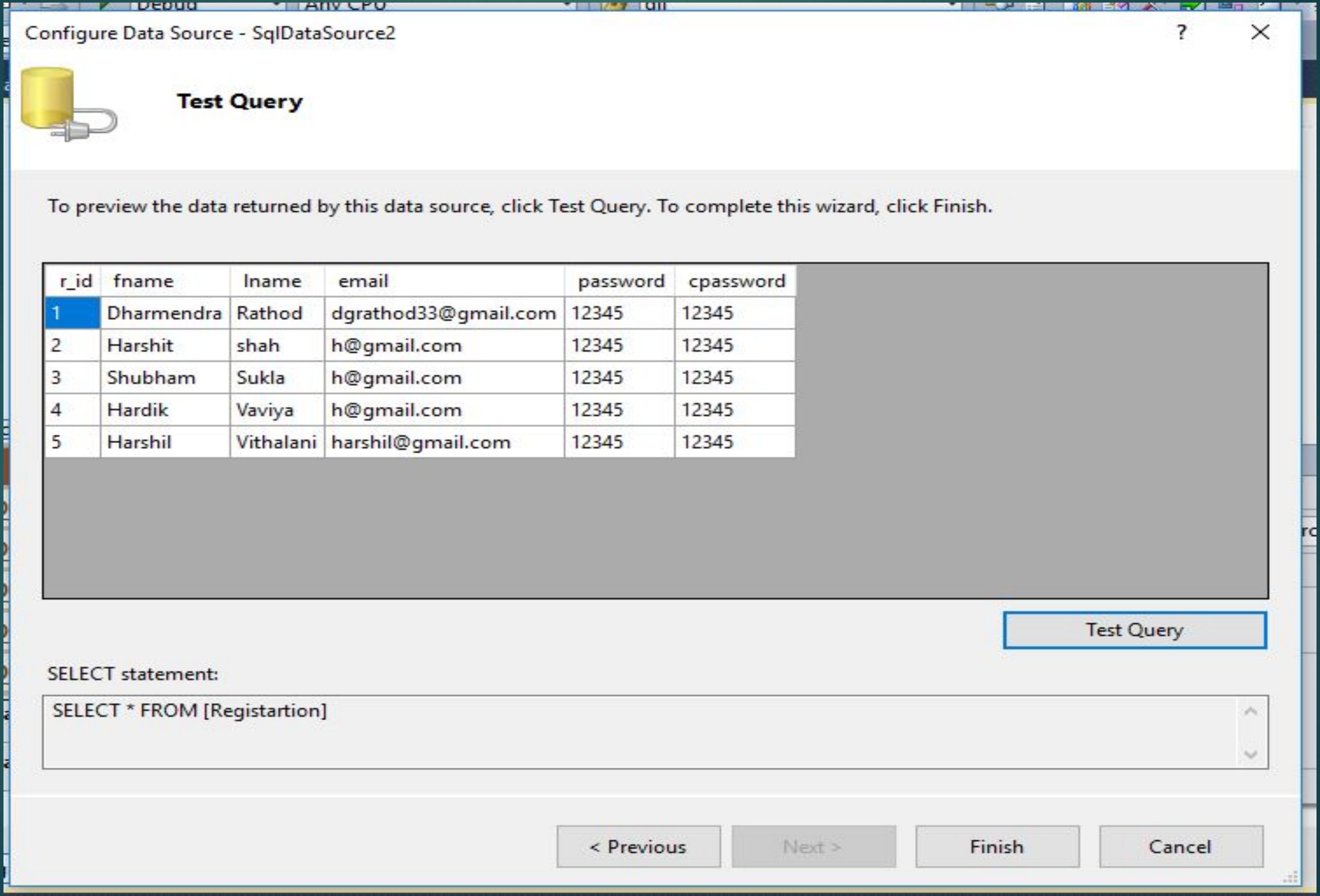
- [✓] *
- [ ] r_id
- [ ] fname
- [ ] lname
- [ ] email
- [ ] password
- [ ] cpassword

[ ] Return only unique rows

SELECT statement:

```
SELECT * FROM [Registartion]
```

- it will ask to choose database objects like tables/Views and it will make DataSet from chosen database objects and click finish.


Configure Data Source - SqlDataSource2

Test Query

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

r_id	fname	lname	email	password	cpassword
1	Dharmendra	Rathod	dgrathod33@gmail.com	12345	12345
2	Harshit	shah	h@gmail.com	12345	12345
3	Shubham	Sukla	h@gmail.com	12345	12345
4	Hardik	Vaviya	h@gmail.com	12345	12345
5	Harshil	Vithalani	harshil@gmail.com	12345	12345

SELECT statement:

```
SELECT * FROM [Registartion]
```

- You can see it will automaticaly add DataSet, DataAdapter and binding source controls into application.
- You can find following automatic generated code in Form_Load Event

```
// TODO: This line of code loads data into the 'db1DataSet.Table1' table. You can move, or remove it, as needed.
this.table1TableAdapter.Fill(this.db1DataSet.Table1);
```

Data GridView Programming

- Use DataSource property and bind DataTable or DataSet to it.

```
OleDbConnection con = new OleDbConnection(@“ConnectionString");

OleDbDataAdapter ad = new OleDbDataAdapter("Select * from Table1",con);

DataTable dt = new DataTable();

ad.Fill(dt);

dataGridView1.DataSource = dt;

dataGridView1.Databind();
```

Repeater Control

- The Repeater control is used to display a repeated list of items that are bound to the control. The Repeater control may be bound to a database table, an XML file, or another list of items.
- Repeater is a Data Bind Control. Data Bind Controls are container controls.
- Data Binding is the process of creating a link between the data source and the presentation UI to display the data.
- ASP.Net provides rich and wide variety of controls, which can be bound to the data.

Repeater Control

- **Repeater has 5 inline template to format it:**

  1. <HeaderTemplate>
  2. <FooterTemplate>
  3. <ItemTemplate>
  4. <AlternatingItemTemplate>
  5. <SeperatorTemplate>

- **HeaderTemplate:** This template is used for elements that you want to render once before your ItemTemplate section.

- **FooterTemplate:** - This template is used for elements that you want to render once after your ItemTemplate section.

Repeater Control

- **ItemTemplate:** This template is used for elements that are rendered once per row of data. It is used to display records

- **AlternatingItemTemplate:** This template is used for elements that are rendered every second row of data. This allows you to alternate background colors. It works on even number of records only.

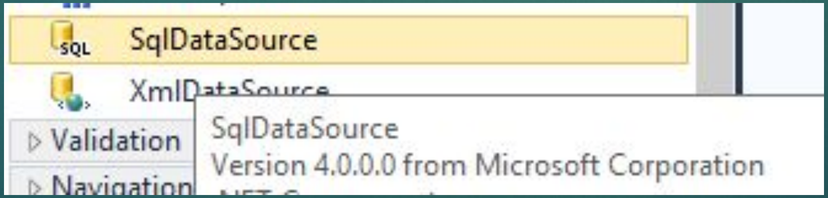
- **SeperatorTemplate:** It is used for elements to render between each row, such as line breaks.

Binding Data to DataBound Controls

- Data-bound controls are Web controls those can easily bind with data components. Microsoft Visual Studio.NET is a rich IDE for ADO.NET data components.
- Data-bound controls have properties, which you can set as a data component and theyre ready to present your data in Web Controls(Gridview,ListBox,ComoboBox).
- DataSource and DisplayMemeber are two important properties.
  - DataSource property of these controls plays a major role. You can set different kind of data components as datasource property of a control. For example, you can set a DefaultViewManager or a DataView as this property.
  - DisplayMember property can be set to a database table field name if you want to bind a particular field to the control.

SQLDataSource

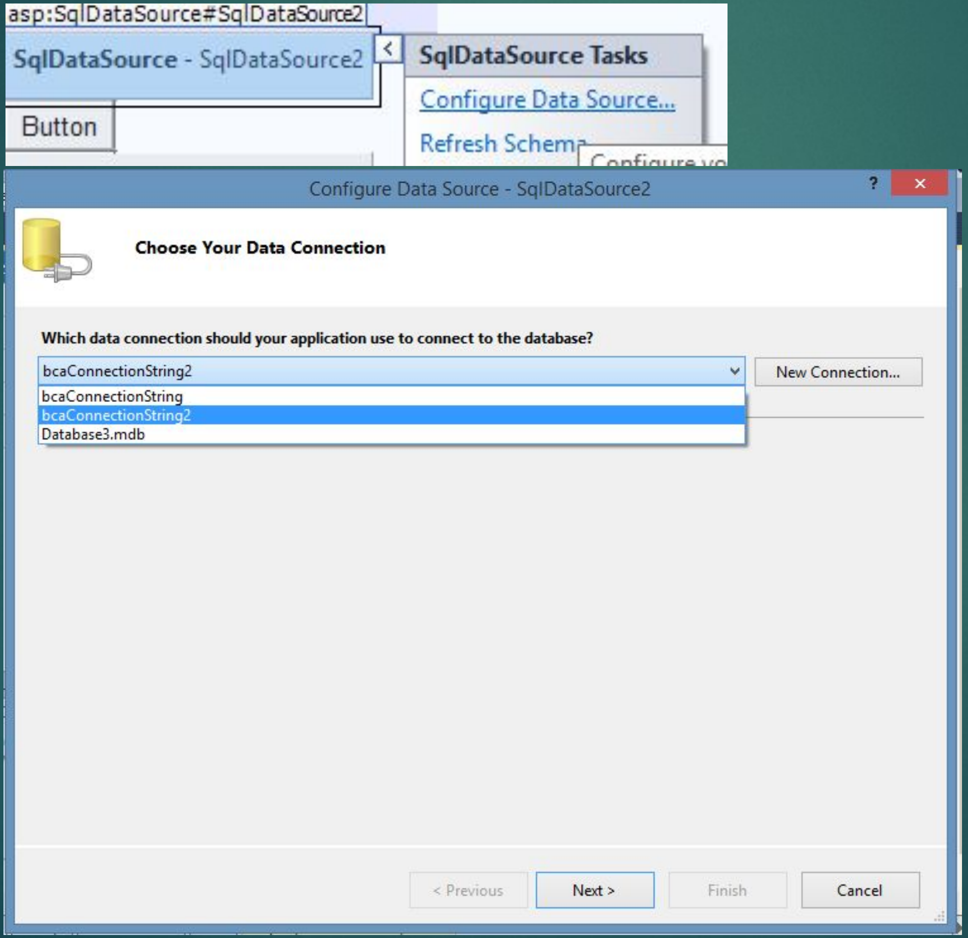
- The SqlDataSource control enables you to use a Web server control to access data that is located in a relational database.
- This can include Microsoft SQL Server and Oracle databases, as well as OLE DB and ODBC data sources.
- You can use the SqlDataSource control with data-bound controls such as the GridView, FormView, and DetailsView controls to display and manipulate data on an ASP.NET Web page, using little or no code.
- To add sqlDataSource goto ToolBox>DataControls



SqlDataSource
Version 4.0.0.0 from Microsoft Corporation

SQLDataSource

- To configure DataSource create db connection string or use exisiting connection string.



asp:SqlDataSource#SqlDataSource2

SqlDataSource - SqlDataSource2

Button

SqlDataSource Tasks

Configure Data Source...

Refresh Schema

Configure Data Source - SqlDataSource2

Choose Your Data Connection

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

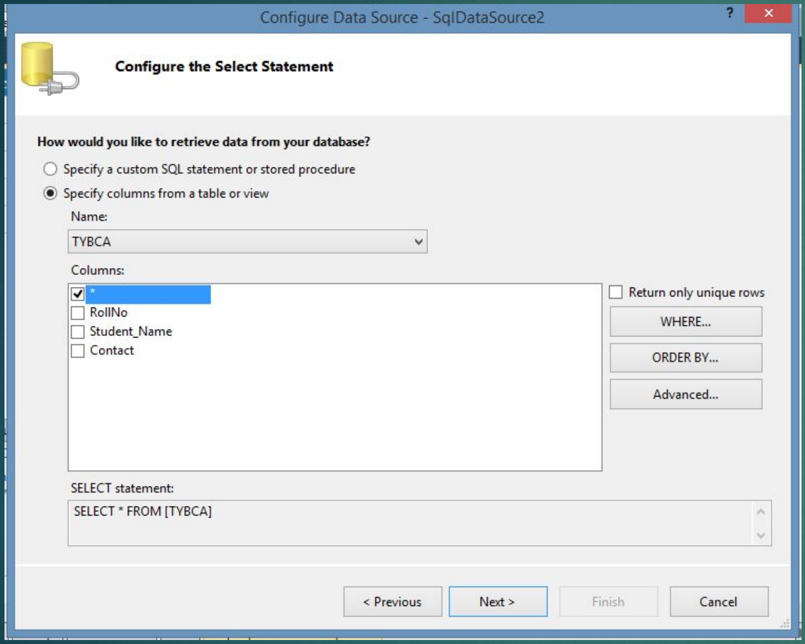
bcaConnectionString2

New Connection...

bcaConnectionString
bcaConnectionString2
Database3.mdb

< Previous | Next > | Finish | Cancel

- You can select fields from interface or you can select first value SPECIFY SQL stmt where you can manually write query or you can use querybulider.



Configure Data Source - SqlDataSource2

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

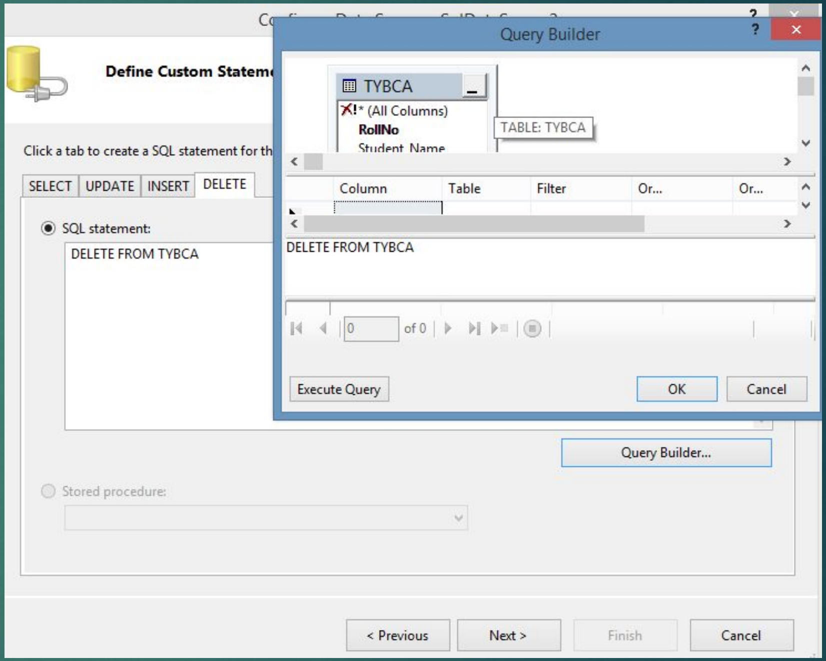
Name: TYBCA

Columns:
- [✓] *
- [ ] RollNo
- [ ] Student_Name
- [ ] Contact

[ ] Return only unique rows

SELECT statement:

```
SELECT * FROM [TYBCA]
```



Define Custom Stateme

Click a tab to create a SQL statement for th

SELECT | UPDATE | INSERT | DELETE

(●) SQL statement:

```
DELETE FROM TYBCA
```

( ) Stored procedure:

Query Builder

TYBCA
- * (All Columns)
- RollNo
- Student_Name

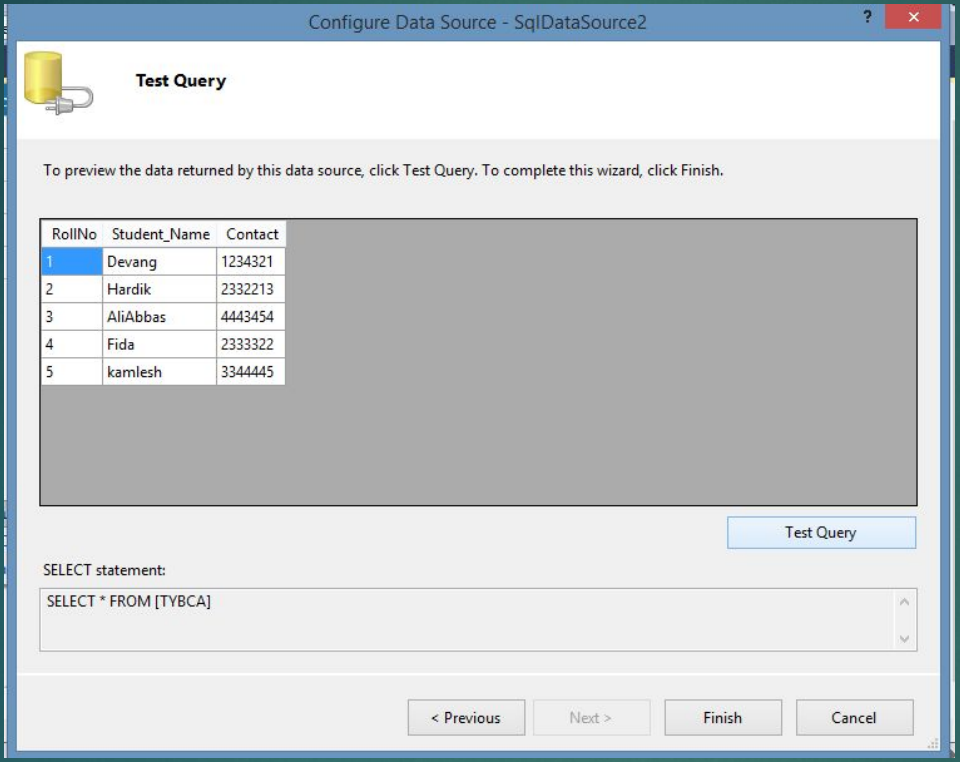
TABLE: TYBCA

Column	Table	Filter	Or...	Or...

```
DELETE FROM TYBCA
```

0 of 0

- You can test your query result.



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

RollNo	Student_Name	Contact
1	Devang	1234321
2	Hardik	2332213
3	AliAbbas	4443454
4	Fida	2333322
5	kamlesh	3344445

SELECT statement:

```
SELECT * FROM [TYBCA]
```

DataBinding Expressions

- Data-binding syntax allows you to bind control property values to data and specify values for retrieving, updating, deleting, and inserting data.
- Data-binding expressions are contained within <%# and %> delimiters and use the **Eval** and **Bind** functions.
- **Eval** function is used to define one-way (read-only) binding.
- **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.
- <%# DataBinder.Eval(Container,"DataItem.RollNo");

DataBinding Expressions

- **Eval** method evalutes the Eval method of 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.
- Data-binding expressions are resolved when the **DataBind** method of a control or of the Page class is called.
- For controls such as the GridView, DetailsView, and FormView controls, data-binding expressions are resolved automatically during the control's **PreRender** event and you are not required to call the **DataBind** method explicitly.