Ado And WebForm

ADO.Net

Q. What is ADO.Net

- ADO.net Stands for ActiveX Data Object.
- It is a database access technology created by Microsoft as part of its .Net Framework that can access any kinds of data source.
- ADO.Net is a Medium which is use to communicate the .Net application to Data Source(like: XML, SQL Server, MS Access, Oracal and etc) by the help of data provider which work like a briz, And this data provider are provide by ADO.Net.

```
Dot.Net Application<--->ADO.Net(Data Provider)<--->Data Source
```

And this data provider is a set of classes in the .NET Framework used to connect, retrieve, manipulate, and update data from data source.

- ADO.Net Provides a bridge bitween frontend controls and the backend database.
- ADO.Net is a Module of .Net Framework which is use to establish the connection bitween application and data source.
- All ADO.Net class are located into `System.Data.dll` and XML class are located into `System.Xml.dll`.
- It's a set of OOP class that provides a rich set of data components to create high-preformance, reliable and scalable database applications.

Q. What are the data provider?

We have multiple data provider because of mutiple type of data souce is there.

Data Provider	Database(data provider)	NameSpace
SQLClient	SQL Server	System.Data.SQLClient.
OracleClient	Oracle	System.Data.OracleClient.
OleDb	MS Access	System.Data.OleDb.
Odbc	Configured Database	System.Data.Odbc.

This Data provide is provie by ADO.Net.

Q. What is the ADO.Net Components? And What is that?

- Components are designed for data manipulation and fast data access.
- conection, Command, DataReader, DataAdaptor, DataSet and DataView are the components of ADO.Net that are used to perform database operations.
- ADO.net has two main components that are used for accessing and manipulating data. Ther are as follow:
    - Data Provider
    - DataSet

Q. What are the comman class in all data provider?

1. **Connection**: Connect to the data base
    - `SQLConnection` for SQLClient.
    - `OracleConnection` for OracleClient.
    - `OleDbConnection` for OleDb.
    - `OdbcConnection` for Odbc.
2. **Command**: Prepare an SQL Command
    - `SQLCommand` for SQLClient.
    - `OracleCommand` for OracleClient.
    - `OleDbCommand` for OleDb.
    - `OdbcCommand` for Odbc.
3. **DataReader**:
    - `SQLDataReader` for SQLClient.
    - `OracleDataReader` for OracleClient.
    - `OleDbDataReader` for OleDb.
    - `OdbcDataReader` for Odbc.
4. **DataAdaptor**:

Q. What are the work for Preforming CRUD Operation with DataBase fro .Net Applications?

To Preform DataBase operations We need some Work

1. Connect to the database.
2. Prepare an SQL Command.
3. Execurte the Command.
4. Retrive the results and display them in the application.

Q. What is the use of SQLConnection class?

- It is uded to establish an open only one connection to the SQL Server database.

- A SQLConnection object represent a unique session to a SQL Server data source.
- ADO.Net connection is an object that provides database connectio and this is the entry point to a database.
- It ia a Sealed class so that cannot be inherited.
- SQLConnection present in inside the `System.Data.SqlClient` and this namepace present in inside the `System.Data.SqlClient.dll`.
- Defination of SQLConnection class:`public sealed SqlConnection: System.Data.Command.DbConnection, ICloneable`.
- SqlConnection inherit from System.Data.Command.DbConnection and it's have interface `ICloneable`.
- Inheritane: Object -> DBConnection -> SqlConnection. All C# class have one parents class name is `Object` class.
- Implements: ICloneable.

Note:

- SqlConection class uses `SqlDataAdapter` and `SqlCommand` class together to `increase performance` when connecting to a Microsoft SQl Server database.
- Connection does not close implictly it's necessary to close Explictly by using Close() method but when you use using block not require to close the connection implictly close by using block.

Q. How to create object of SQLConnection Class of ADO.Net?

```
SqlConnetion con = new SqlConnection(cs);
```

- `SqlConnection(cs)`: this is a Constructor which connection string.

Q. What is connection string?

The string that includes the source, database name, and other parameters needed to establish the initia connection.

If you not pass any things then `SqlConnection(cs)` automaticly inslize with empty string.

Connection string have:

1. `Data Source (Database Server Name)`: This identifies the server name, whic could be the Local machine, machine domain name or IP address.
2. `Initial Catalog (Database Name)`: This identifies the database name.

3. `Integrated Security = True` :For windows auth use true and not write Username and Password another wise use false. If you want to use Sql auth then it's not mendatry to write this directly write Username and password.
4. `User Id`: When use SQL Auth.
5. Password: When use SQL Auth.

Example:`Data Source=MyServrName; Initial Catalog=MyDatabaseName; Integrated Security=true;` or `Data Source=MyServrName; Initial Catalog=MyDatabaseName; Integrated Security=false; User Id=MyUserName; Password=MyPassword` or `Data Source=MyServrName; Initial Catalog=MyDatabaseName; User Id=MyUserName; Password=MyPassword`

Example with SQConnection:

```
string sc= "Data Source=MyServrName; Initial Catalog=MyDatabaseName; Integrated Security=true;";

SqlConnetion con = new SqlConnetion(cs);
con.Open();
//Preform task database queries
con.Close();
```

- When you open the connection you need state property of connection class.

Q. What is .Open() and .Close()? And why we need to close?

- `.Open()`: Use to Open the connection of databse.
- `.Close()`: Use to Close the connection of databse. If you not close the SqlConnetion object then system resources are using continuously mins memory consule continuously.

Q. What is state property?

- When you open the connection you need state property of connection class.
- When you use the `.Open()` and connection is open then state property show open if connection is not open for some reason then it show close.

Q. Give me example to establish connection with SQL Server?

```
using System.Data.SqlClient;

static void Connection()
{
    string cs = "Data Source=DESKTOP-HOOMVQE\\MSSQLSERVER02; Initial Catalog=ADO_Crud; User Id=mk; Password=123";
```

```
    SqlConnection db = new SqlConnection(cs);
    try
    {
        db.Open();
        if (db.State == System.Data.ConnectionState.Open)
        {
            Console.WriteLine("Connection open");
        }
    }
    catch (SqlException ex)
    {
        Console.WriteLine(ex.Message);
    }
    finally
    {
        db.Close();
    }
}
```

Q. When we use Using block?

- To ensure that connection are automaticlly close when code exits the block, then use using block.
- Using block is used to close the connection automatically. We don't need to call close() method explicitly, using block do this for ours implicitly when the code exits the code.

Example:

```
using System.Data.SqlClient;
using System.Data;
static void Connection()
{
    string cs = "Data Source=DESKTOP-HOOMVQE\\MSSQLSERVER02; Initial
Catalog=ADO_Crud; User Id=mk; Password=123";

    SqlConnection db = null;
    try
    {
        using (db = new SqlConnection(cs))//connection automatic close
        {
            db.Open();
            if (db.State == ConnectionState.Open)
            {
                Console.WriteLine("Connection open");
            }
        }
    }
    catch (SqlException ex)
    {
        Console.WriteLine(ex.Message);
```

```
    }
    finally//if connection not close by using in middle any exception come the
finally run
    {
        db.Close();
    }
}
```

Q. What are the two use of Using Keyword?

---

1. Using is used for adding namespace in a file(i.e: Example: Using System.Data.SqlClient).
2. Using is uded for automatically closing of the connection.

Q. Constructors of SQLConnection class of ADO.Net?

---

1. **SqlConnection()**: Insitalize a new instance of the SqlConnection class.
2. **SqlConnection(String)**: Initialize a new instance of the SqlConnection class when give a string that connection string.
3. **SqlConnection(String, SqlCredential)**: Insitalize a new instance of the SqlConnection class given a connection string, that `does not use "Integrated Security = true"(not support windows auth")` and a SqlConnection object that contains the user ID and Password.

Q. Usually where we store the connection string?

A Connection string is usually stored in the `Web.Config` file or `app.config` file of an applicATION.

- If you use the Web from then use `Web.Config` file.
- If you use the anothe like console then use `app.Config` file.
- Use inside the `configuration`. **Syntax**:

```
<connectionString>
    <add name="" connectionString="" providerName=""/>
</connectionString>
```

Syntax:

```
<connectionStrings>
    <add name="dbCon"
         connectionString="Data Source=DESKTOP-HOOMVQE\MSSQLSERVER02; Initial
Catalog=ADO_Crud; User Id=mk; Password=123;"
         providerName="System.Data.SqlClient"/>
```

```
</connectionStrings>
```

Q. How to get the Connection String Value?

- To get the connection string value we need one Refrences `System.Configuration;` first install this in you project.
- To install this refrence write click on Refrences > Add Refrences > Assemblies > select System.Configuration > Ok.

Syntax: `string cs = ConfigurationManager.ConnectionStrings["<Name of add tag>"].ConnectionString;`

Example:

```
using System;
using System.Data;
using System.Configuration;//Add refrence
using System.Data.SqlClient;

static void Main(string[] args)
{
    string cs =
ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;//get connection
string other are same.
    SqlConnection db = null;
    try
    {
        using (db = new SqlConnection(cs))
        {
            db.Open();
            if (db.State == ConnectionState.Open)
                Console.WriteLine("Connection open");
        }
    }
    catch (SqlException ex)
    {
        Console.WriteLine(ex.Message);
    }
    finally
    {
        db.Close();
    }
    Console.ReadLine();
}
```

Q. What are the namespace/provider used for connection class & how to use class with this provider?

ADO.Net provides connection to multiple providers. Each provider has a functionality to connect with diffrent database. Hear is a list of data provider in ADO.Net and this purpose.

- Data Provider for SQL Server (System.Data.SqlClient). Connection object for SQL Server (SqlConnection).
- Data Provider for MS ACCESS (System.Data.OleDb). Connection object for MS ACCESS (OleDbConnection).
- Data Provider for MYSQL (System.Data.Odbc). Connection object for MYSQL (OdbcConnection).
- Data Provider for ORACLE (System.Data.OracleClient). Connection object for ORACLE (OracalConnection).

Q. What is SqlCommand and why we Use?

The ADO.Net SqlCommand class is uded to store and execute the SQL statement against the SQL database.

SqlCommand class is uded to prepare an SQL statement or StoredProcedure that we want to execute on a SQL Server database.

Q. What is the SqlCommand Signature?

- It is a seald class means it cannot be inherited.
- Syntax:

```
public sealed class SqlCommand:System.Data.Command.DbCommand,
ICloneable,IDisposable
```

- `ICloneable`: I represent the Interface alwase.

Q. What are the step follow by ADO.Net for manuplating Database?

1. Connect to the database.
2. Opent the connection.
3. Prepare an SQL Command.
4. Execute the Command.
5. Retrive the results and display in the aplication.
6. Close the connection.

This flow are normal flow which is use in all .Net application to preforme any type of operations with database.

Q. How many Overloaded constructors method in SqlCommand class & Explain it?

---

1. **public SqlCommand();**
2. **public SqlCommand(string cmdText);**
3. **public SqlCommand(string cmdText, SqlConection);**
4. **public SqlCommand(string cmdText, SqlConection, SqlTransaction);**
5. **public SqlCommand(string cmdText, SqlConection, SqlTransaction, SqlCommandColumnEncryptionSetting ColumnEncrypted Setting);**

Explanation:

- `All constructor` is used to initializes a new instance of the `System.Data.SqlClient` class.
- `cmdText` means text of the query. Here, the cmdText is the text of the query that we want to execute.
- `SqlConection` Take connection object `System.Data.SqlClient.SqlConnection`. Connection of an instance of SQL Server.
- `SqlTransaction` is a instance, use when you want to SqlTransaction. It provide the feature of rolback or commit. If you use 3 parameter SqlTransaction.
- `SqlCommandColumnEncryptionSetting`: is provide the encription feature of column.

Q. What are the most commanly method of SqlCommand class?

---

1. ExecuteReader()
2. ExecuteNonQuery()
3. ExecuteScalar()

Q. What are the ExecuteReader() method, SqlDataReader, and Read()? Provide an example using the SqlCommand with a simple query and a stored procedure. Demonstrate both the empty constructor and the 2-parameter constructor of SqlCommand.

- By the help of this method we can execute the sql select query (T-SQL) and read the data and return the SQLDataReader object and display in .Net aplication.
- The ExecuteReader() method is used to execute a SQL SELECT query and retrieve the results. It returns a SqlDataReader object, which is used to read the data row-by-row from the result set. This method is commonly used for reading data from a database in .NET applications.

Example:

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;

namespace ADO
{
    internal class Program
```

```
    {
        static void Main(string[] args)
        {
            // Get the connection string from the config file
            string cs =
ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;

            try
            {
                using (SqlConnection db = new SqlConnection(cs))
                {
                    db.Open(); //Mendatry to open after SqlCommand

                    if (db.State == ConnectionState.Open)
                    {
                        // Example 1: Using 2-parameter constructor
'SqlCommand(string cmdText, SqlConnection)'
                        SqlCommand cmd = new SqlCommand("SELECT * FROM Student",
db);

                        // Example 2: Using empty constructor 'SqlCommand()' and
setting CommandText and Connection separately
                        SqlCommand cmdEmpty = new SqlCommand();
                        cmdEmpty.CommandText = "SELECT * FROM Student";
                        cmdEmpty.Connection = db;

                        // Example 3: Using Stored Procedure with 2-parameter
constructor
                        SqlCommand cmdStoredProcedure = new
SqlCommand("spGetData", db);
                        cmdStoredProcedure.CommandType =
CommandType.StoredProcedure;  // Indicating it's a stored procedure

                        // Execute the command and read the data using
SqlDataReader
                        SqlDataReader dr = cmdStoredProcedure.ExecuteReader();

                        // Read data until the reader has rows
                        while (dr.Read())
                        {
                            Console.WriteLine(
                                "ID = " + dr["ID"] +
                                ", FName = " + dr["FName"] +
                                ", LName = " + dr["LName"] +
                                ", Age = " + dr["Age"] +
                                ", Status = " + dr["Status"]
                            );
                        }
                    }
                }
            }
            catch (SqlException ex)
            {
                Console.WriteLine("SQL Error: " + ex.Message);
```

```
            }
            catch (Exception ex)
            {
                Console.WriteLine("Error: " + ex.Message);
            }

            Console.ReadLine();
        }
    }
}
```

Q. What are the ExecuteNonQuery() method and AddWithValue() in ADO.NET? Provide an example where a user inserts values into a database.

- `ExecuteNonQuery() Method`: This method is used to execute SQL statements such as INSERT, UPDATE, and DELETE that do not return data but affect the rows in the database. It returns an integer value, which indicates the number of rows affected by the query. If no rows are affected, it returns 0.
- It return 0 and 1, 2 ,3 ... because it return the on the basis of affected rows.

Example:

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;

string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
SqlConnection db = null;
try
{
    using (db = new SqlConnection(cs))
    {
        //Get the value from  user
        Console.Write("Enter ID: ");
        int iD = Convert.ToInt16(Console.ReadLine());
        Console.Write("Enter FName: ");
        string fName = Console.ReadLine();
        Console.Write("Enter LName: ");
        string lName = Console.ReadLine();
        Console.Write("Enter Age: ");
        int age = Convert.ToInt16(Console.ReadLine());
        Console.Write("Enter Gender: ");
        char gender = Convert.ToChar(Console.ReadLine());

        //Initializes SqlCommand instance
        string query = "Insert into student(ID, FName, LName, Age, Gender) values
(@Id, @FName, @LName , @Age, @Gender)";
        SqlCommand cmd = new SqlCommand(query, db);
```

```
        //Add value in query which get by user
        cmd.Parameters.AddWithValue("@Id", iD);
        cmd.Parameters.AddWithValue("@FName", fName);
        cmd.Parameters.AddWithValue("@LName", lName);
        cmd.Parameters.AddWithValue("@Age", age);
        cmd.Parameters.AddWithValue("@Gender", gender);

        db.Open(); //Mendatry to open after SqlCommand
        if (db.State == ConnectionState.Open)
        {
            //Execute the command:-
            int result = cmd.ExecuteNonQuery(); //Its return integer value

            //Validate data is inserted of not
            if(result != 0)
                Console.WriteLine("Data inserted Successfully!");
            else
                Console.WriteLine("Come Error at the time of data inserted!.");
        }
    }
}
catch (SqlException ex)
{
    Console.WriteLine(ex.Message);
}
finally
{
    db.Close();
}
Console.ReadLine();
```

- `AddWithValue() Method`: This method is used to add parameters to the SQL query. It binds the parameter to the SQL statement by assigning a value to it, allowing for dynamic data to be passed into queries securely, which helps prevent SQL injection.
- You can also preform Update(`Update Student set FName = @FName where Id = @Id;`) and Delete(`Delete from student where id = @Id;`) operations. You need to replace the query string and insert the value in query string using `cmd.Parameters.AddWithValue("<Variable>", <Value>);` and etc according to recquirement.

Q. What are the ExecuteScalar() method in ADO.NET? Provide an example where a user find Max Age values into a database.

- By the help of `ExecuteScalar()` you can use select query for Aggregate function (like:Count(), Min(), Max(), Avg(), Sum()) but `ExecuteScalar()` return single value.

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;

string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
SqlConnection db = null;
try
{
    using (db = new SqlConnection(cs))
    {
        //Initializes SqlCommand instance
        string query = "Select Max(Age) from student"; //You can use Count() and
etc Aggrigation function
        SqlCommand cmd = new SqlCommand(query, db);

        db.Open();//Mendatry to open after SqlCommand
        if (db.State == ConnectionState.Open)
        {
            //Execute the command:-
            int result = Convert.ToInt32(cmd.ExecuteScalar());//Return sing int
value

            Console.WriteLine(result); //Output
        }
    }
}
catch (SqlException ex)
{
    Console.WriteLine(ex.Message);
}
finally
{
    db.Close();
}
Console.ReadLine();
```

Q. What is SQLDataReader class?

- ADO.NET DataReader object is used for accessing data from a data source in a read-only(you can only read the data) and forward-only manner (you can only move to the next record; you cannot move backward to previous records).
- Syntax: `SqlDataReader dr = cmd.ExecuteReader();`.
- `ExecuteReader()` is executes the Sql query and return an object, which is an instance of SqlDatReader. The Sql Command can be a Select query or Store procudure.

- Each data provide has its own DataReader class:

Data Provider	DataReader Class
SQlClient	SqlDataReader
OracalClient	OracalDataReader
OleDb	OleDbDataReader
Odbc	OdbcDataReader

- Signuture:

```
public class SqlDatReader:DbDataReader, IDataReader, IDisposable, IDataRecord{}
```

- IDataReader, IDisposable, IDataRecord this are interface because prefix with `I` and SqlDatReader Implement this interface.
- `DbDataReader` is the base class, and `SqlDataReader` inherits from it.
- This means that while you can read and display data with `SqlDataReader`, you cannot update or delete the data. If you need to modify the data, you should use a `DataAdapter` instead of a `DataReader`.
- The connection to the database must be open when using `SqlDataReader`. You must explicitly close the connection after reading the data or you can use using block to close it.
- You also need to open and close the `SqlDataReader`. You must explicitly close the `SqlDataReader` or you can use using block to close it.
- You don't use the `new` keyword when creating an instance of `SqlDataReader`. Instead, you call the `ExecuteReader()` method on the `SqlCommand` object.

```
SqlDataReader dr = cmd.ExecuteReader();//Correct
//SqlDataReader dr = new SqlDataReader();// Wrong: You don't instantiate it
directly.
```

- ExecuteReader() is execute the SqlCommand and return an object of SqlDataeader to read data..
- The retrieved data is `stored` in the client network buffer memory `and then` the clent can read date using Read method whic is an method of `SqlDataReader`. because of data stored in client network buffer memory the performance get increase. No need to call database to read data again and again.
- Read method read one row data at a time.

Q. GIve me the Exampp=le of SqlDataReader?

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;
```

```
string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
SqlConnection db = null;
using (db = new SqlConnection(cs))
{
    string query = "Select * from student";
    SqlCommand cmd = new SqlCommand(query, db);

    db.Open();//Mendatry to open
    if (db.State == ConnectionState.Open)
    {
        //Execute the command:-
        using(SqlDataReader dr = cmd.ExecuteReader()) //Open & close the
SqlDataReader
        {
            while (dr.Read())
            {
                Console.WriteLine(dr[0] +" "+ dr[1] + " " + dr[2] + " " + dr[3] +
" " + dr[4]);
                //or
                Console.WriteLine(
                  "ID = " + dr["ID"] +
                  ", FName = " + dr["FName"] +
                  ", LName = " + dr["LName"] +
                  ", Age = " + dr["Age"] +
                  ", Status = " + dr["Status"]
                );

            }
        }
    }
}
Console.ReadLine();
```

Q. What are the properties of SqlDataReader class?

- `FieldCount`(return int): It give the number of columns in the current row.

```
using(SqlDataReader dr = cmd.ExecuteReader())
{
    Console.WriteLine(dr.FieldCount);
}
```

- `HasRows`(return boolean): If any row available in table then return true another wize false.

```
using(SqlDataReader dr = cmd.ExecuteReader())
{
```

```
        Console.WriteLine(dr.HasRows);
    }
```

- `IsClosed`: It retrieves a Boolean value whether the instance has been closed or not `Console.WriteLine(dr.IsClosed);`.
- `Item[strint]`: Get the particulr value of row by passing column name `dr["ID"]`.

3:20

Q. What are the method of SqlDataReader class?

- `Read()`: If next record is available then it's return true another wise return false.
- `Getname(int i)`: It get the name of the specified column. Hear, parameter `int i` is the column number.

```
Console.WriteLine(Getname(2))//Return the position of column name
```

- `NextResult()`: If you want to execute multiple Sql query then it's important. Return multiple results like at one time if you want to two table data using seperat seperat sql query.

```
string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
SqlConnection db = null;
using (db = new SqlConnection(cs))
{
    //I have two table one is student and another one is students.
    string query = "Select * from student; select * from students;";
    SqlCommand cmd = new SqlCommand(query, db);

    db.Open();
    if (db.State == ConnectionState.Open)
    {
        using(SqlDataReader dr = cmd.ExecuteReader())
        {
            while (dr.Read())
            {
                Console.WriteLine($"{dr[0]} | {dr[1]} | {dr[2]} | {dr[3]}");
            }
            Console.WriteLine("-----------------------------------------");
            if (dr.NextResult()) //If you not use this line then only one while
loop work and only one first query execut by default.
            {
                while (dr.Read())
                {
                    Console.WriteLine($"{dr[0]} | {dr[1]} | {dr[2]}");
                }
            }
        }
```

```
    }
}
```

Q. What are the two articture when we accessing the data in ADO.Net?

---

1. **Connected Data Access** in this we articture `SqlDataReader`. In Connected Data Access it's mendatory to open the database connection and close.
2. **Dis-Connected Data Access** in this we articture `SqlDataAdaptor`. `In Dis-Connected Data Access` we not need to open and close the database connection, it's happen automaticlly.

Q. What is SqlDataAdaptor class?

- No need to open and close the database connection, it's happen automataclly. It Store table and Relationship.
- The `SqlDataAdaptor` class is belongs to `System.Data.SqlClient` namespace.
- In `SqlDataAdaptor` we need to use two class with SqlDataAdaptor first one is `DataSet` another one is `DataTable`.
- When `SqlDataAdaptor` read the data we have two option to store the data `DataSet` and `DataTable`.
- It work as a bridge between DataSet and Database(geting the data from database using SqlDataAdaptor and stored in DataSet).
- **Fill method** is used to open and cloase the database, execute the query and store the data in dataset or datatable.
- `SqlDataAdaptor` we can reprent sql query and db connection.
- It can be used to fill the dataSet and Update the data source.

Q. What is DataSet and DataTable?

- Both `DataSet` and `DataTable` use to store the data just like database.
- Normaly database tables are stored in Hardisk. But DataSet tables are stored in Web Server, That way we call the DataSet as a `In-Memory Representation` of the databse.`In-Memory Representation` means table store in web server.
- If you working with multiple tabe then you need use `DataSet`.
- If you working with single tabe then you need use `DataTable`.
- It is indepewndent data provider means it is **not available in any other data provider class** like SqlDataReader, OracalDataReader, OleDbDataReader, OdbcDataReader , it is in indepewndent class.

- It is not in the any dataprovider namespaces like System.Data.SqlClient and etc . **It available in seperate namespace** `System.Data`.
- `System.Data` is indepewndent, **it is common to all dataprovider**.
- Both DataSet can store the multiple table. And to access the table we use index like for first table `Table[0]` and for `n`th table `Table[n]`.
- A **DataRow** is an object that represents a single row of data from a table from a DataTable and DataSet, allowing you to access and work with the data in each column of that row.

Q. What we pass in SqlDataAdapter constructor?

We pass as same as SqlCommand like in SqlCommand constructor we pass sql command and Connection object same as SqlDataAdapter constructor we pass two parameter sql command and Connection object.

Q. Give me the example, using DataSet?

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;

string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
SqlConnection db = null;
using (db = new SqlConnection(cs))
{
    SqlDataAdapter cmd = new SqlDataAdapter("Select * from student", db);
    DataSet ds = new DataSet();
    cmd.Fill(ds);// Data get data from SqlDataAdapter and stored in ds(dataSet).

    foreach (DataRow row in ds.Tables[0].Rows)//or Tables[""<TabelName>]
    {
        Console.WriteLine($"{row[0]} | {row[1]} | {row[2]} | {row[3]} |
{row[4]}");
    }

}
```

- **ds.Tables[0].Rows**: We get the data from `ds` which is an object of DataSet, from ds we target the `Tables[0]`(first table) and from first table we get all Rows.

Q. Give me the example using DataTable?

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;

string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
SqlConnection db = null;
using (db = new SqlConnection(cs))
{
    SqlDataAdapter cmd = new SqlDataAdapter("Select * from student", db);
    DataTable dt = new DataTable();
    cmd.Fill(dt);// Data get data from SqlDataAdapter and stored in ds(datatable).

    foreach (DataRow row in dt.Rows)//Get all rows from student table
    {
        Console.WriteLine($"{row[0]} | {row[1]} | {row[2]} | {row[3]} |
{row[4]}");
    }
}
```

- **ds.Tables[0].Rows**: We get the data from `ds` which is an object of DataSet, from ds we target the `Tables[0]`(first table) and from first table we get all Rows.

Q. What if Fill method?

Fill method belongs to SqlDataAdapter. It have multiple word:

- Open the connection.
- Execute the sql command.
- Fill the data in DataSet or DataTable.
- close the Connection automaticlly.

Because of Fill method SqlDataAdapter called as a Dis-Connected Data Access.

The connection is kept open only as long as it is needed. That means once the fill method completets it's execution, then the connection close automatically.

Use DataRow to loop through each record and print the data on the console.

Once the DataSet or DataTable is filled, then no active connection is required to read the data.

Q. What is the signature of SqlDataAdaptor?

```
public sealed class SqlDataAdapter : DbDataAdapter, IDbDataAdapter, IDataAdapter,
ICloneable{}
```

- SqlDataAdapter inherit DbDataAdapter class and 3 interface IDbDataAdapter, IDataAdapter, ICloneable.

Q. How to call Store procedure using SqlDataAdapter?

- In order to execute a stored procedure using SqlDataAdapter we need just specify the name of the stored procedure instead of the in-line SQL statement like `SqlDataAdapter cmd = new SqlDataAdapter("spGetData", db);` then we have to specify the command type as stored procedure using the command type property of the SqlDataAdapter object`cmd.SelectCommand.CommandType = CommandType.StoredProcedure;` for bect practice it's not mendatary to use without using this we also get same output.

```
string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
SqlConnection db = null;
using (db = new SqlConnection(cs))
{
    SqlDataAdapter cmd = new SqlDataAdapter("spGetData", db);//Pass Store
procedure directly but it's not best practice.

    //For Best practice told him this is a store procedure but it's not mendatory.
    cmd.SelectCommand.CommandType = CommandType.StoredProcedure;//StoredProcedure
is a enum

    DataTable dt = new DataTable();
    cmd.Fill(dt);

    foreach (DataRow row in dt.Rows)
    {
        Console.WriteLine($"{row[0]} | {row[1]} | {row[2]} | {row[3]} |
{row[4]}");
    }
}
```

OR use SqlCommand with SqlDataAdapter

```
string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
SqlConnection db = null;
using (db = new SqlConnection(cs))
{
    SqlDataAdapter cmd = new SqlDataAdapter();//Pass Store procedure
    cmd.SelectCommand = new SqlCommand("spGetData", db); //use SqlCommand

    cmd.SelectCommand.CommandType = CommandType.StoredProcedure;//Not mendatory

    DataTable dt = new DataTable();
    cmd.Fill(dt);

    foreach (DataRow row in dt.Rows)
```

```
    {
        Console.WriteLine($"{row[0]} | {row[1]} | {row[2]} | {row[3]} |
{row[4]}");
    }
}
```

Q. How to call Stored Procedure by geting input parameters?

- Use `cmd.SelectCommand.Parameters.AddWithValue("@id", id);`:- **Example**

```
Console.WriteLine("Enter id:");
int id = Convert.ToInt32(Console.ReadLine());

string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
SqlConnection db = null;
using (db = new SqlConnection(cs))
{
    SqlDataAdapter cmd = new SqlDataAdapter();
    cmd.SelectCommand = new SqlCommand("spGetData", db);
    cmd.SelectCommand.Parameters.AddWithValue("@id", id);//id available in stored
procedure

    cmd.SelectCommand.CommandType = CommandType.StoredProcedure;//StoredProcedure
is a enum

    DataTable dt = new DataTable();
    cmd.Fill(dt);

    foreach (DataRow row in dt.Rows)
    {
        Console.WriteLine($"{row[0]} | {row[1]} | {row[2]} | {row[3]} |
{row[4]}");
    }
}
```

```
--Update the Stored procedure
alter procedure spGetData @id int
as
begin
select * from Student where ID = @id
end

--Get the row(Execute)
execute spGetData 3
```

Q. What is DataTable?

- DataTable represents relation data into Tabular form.
- DataTable is similar to the Tables in SQL.
- "ADO.Net provides a DataTable class to create and use data table independently" or "we can get data from database table in DataTable", in both case DataTable create by independently or database we can preform every type of operations like CRUD.
- DataTable can also be use with DataSet. Both comes under Dis-Connected articture.
- In Starting, when we creat DataTable, it does not have table schema(table structure). We need to explicitly define the Schema like columns, datatype etc.
- We can create table schema by adding columns and constraints(rools like primary, not null, default value and etc) to the table.
- After defining the table schema, we can add rows to the table.
- DataTable is a combination of DataColumn and DataRow.
- We must include `System.Data` namespace before creating DataTable.
- DataTable Represent single table.

Q. Give me 2 exampele add data in DataTable using Database and Without using Database?

- Without using Database

```
using System;
using System.Data;

try
{
    DataTable dt = new DataTable("Employees");//Employees is an DataTable
name(table name).

    //Add Columns
    //ID Columns create
    DataColumn id = new DataColumn("id");//Create Columns name
    id.Caption = "ID";//define Columns name
    id.DataType = typeof(int);//define data type
    id.AllowDBNull = false;//ID Never null
    //Make id is an a Autoincrements(take two property `AutoIcrementSeed`(starting
value) and `AutoIcrementStep`(increment vale mins gap between two id value))
    id.AutoIncrement = true;
    id.AutoIncrementSeed = 101;
    id.AutoIncrementStep = 100;
    dt.Columns.Add(id);
```

```
//Or you can directly do like that
//Name Columns create
DataColumn name = new DataColumn("name")
{
    Caption = "Name",//define Columns name
    DataType = typeof(string),//define data type
    AllowDBNull = false,//name Never null
    MaxLength = 50,//Maximum length of string
    DefaultValue = "Anonymous",
    Unique = true,
};
dt.Columns.Add(name);

//Gender Columns create
DataColumn gender = new DataColumn("gender")
{
    Caption = "Gender",//define Columns name
    DataType = typeof(char),//define data type
    AllowDBNull = false,//name Never null
};
dt.Columns.Add(gender);

//Make id is as a primary key
dt.PrimaryKey = new DataColumn[] { id };



//Add Rows
DataRow r1 = dt.NewRow();
//r1["id"] = 101;
r1["Name"] = "Mritunay";
r1["gender"] = 'M';
dt.Rows.Add(r1);

DataRow r2 = dt.NewRow();
r2["gender"] = 'M';
dt.Rows.Add(r2);

//Or you can add rows like that:
dt.Rows.Add(null, "Ankit", 'M');
dt.Rows.Add(null, "Awnish", 'M');
dt.Rows.Add(null, "Puja", 'F');
//dt.Rows.Add(null, null, 'F'); //Unique

//Remove rows:
dt.Rows.Remove(r2 as DataRow);

//Update Fields:
dt.Rows[2]["Name"] = "Basanti";

// Find the row
DataRow[] foundRows = dt.Select("Name = 'Puja'");
if (foundRows.Length > 0)// Check if a row is found and access it
```

```
    {
        DataRow row = foundRows[0];
        Console.WriteLine($"{row["id"]} | {row["Name"]} | {row["gender"]}");
    }

    //Get the data
    foreach (DataRow r in dt.Rows)
    {
        Console.WriteLine($"{r[0]} | {r[1]} | {r[2]}");
    }
}
catch (Exception ex)
{
    Console.WriteLine(ex.Message);
}
```

- Using Database

Using Database

Q. What is Copy() and Clone() method use with DataTable?

- `DataTable.Copy()` return a DataTable with the structure and data of the DataTable.
- **Example**:

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;

string cs = ConfigurationManager.ConnectionStrings["dbCon"].
ConnectionString;
SqlConnection db = new SqlConnection(cs);
string sqlQuery = "Select * from student";
SqlDataAdapter sda = new SqlDataAdapter(sqlQuery, db);
DataTable std = new DataTable();
sda.Fill(std);

//Printing Original Data
Console.WriteLine("Original Data Table:");
foreach (DataRow r in std.Rows)
{
    Console.WriteLine($"{r[0]} | {r[1]} | {r[2]}");
}

DataTable copyDT = std.Copy();//Creating copy

//Printing Copy Data
```

```
Console.WriteLine("Copy Data Table:");
foreach (DataRow r in copyDT.Rows)
{
    Console.WriteLine($"{r[0]} | {r[1]} | {r[2]}");
}

//both data Printing data  is same
```

- `DataTable.clone()` only return the structure of the DataTable, not the rows or data of the DataTable.
- **Example**:

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;

string cs = ConfigurationManager.ConnectionStrings["dbCon"].
ConnectionString;
SqlConnection db = new SqlConnection(cs);
string sqlQuery = "Select * from student";
SqlDataAdapter sda = new SqlDataAdapter(sqlQuery, db);
DataTable std = new DataTable();
sda.Fill(std);

//Printing Original Data
Console.WriteLine("Original Data Table:");
foreach (DataRow r in std.Rows)
{
    Console.WriteLine($"{r[0]} | {r[1]} | {r[2]} | {r[3]}");
}

//Creating Clone
DataTable cloneDT = std.Clone();

//Add Rows:
cloneDT.Rows.Add(1, "Mritunjay","Kumar", 25);
cloneDT.Rows.Add(2, "Amit","Kumar", 24);

//Printing Clone Data
Console.WriteLine("Copy Data Table:");
if(cloneDT.Rows.Count > 0)
{
    foreach (DataRow r in cloneDT.Rows)
    {
        Console.WriteLine($"{r[0]} | {r[1]} | {r[2]} | {r[3]}");
    }
}
else
{
    Console.WriteLine("Rows not found!");
```

```
}
```

Q. What is DataSet?

- It is collection of DataTable. DataSet belongs to `System.Data` namespace.
- DataTable represent single table, But DataSet represent multiple table means we can stor multiple table in DataSet.
- DataSet represent multiple table & each table is represent DataTable object means DataSet treat each table is an DataTable Object.
- DataSet is tabular representation of data.
- Tabular representation means table arrang in row & coloum formate.
- We can use Dataset, combination with DataAdaptor class.
- The DataSet contain the copy of the data we requested measn inside DataSet copy of data is stored, whic we requested the database.
- Contain multiple table at a time.
- DataSet is a local copy of your Database Table that gets populated in client PC means data come in C# memory from Database.
- It is independent of Data Source and because it exists in the local system, it makes application faster and reliable. Not necessary data come only from Database data come from any data source.
- DataSet can retrive those table which having Relations between tables like foreign key.
- DataAdapter object allows us to populate (store) DataTable in a DataSet. You can use fill method of the DataAdapter for populating data in a DataSet.
- DataSet is an in-memory representation of a collection of database object including related table, constraints and relationship among the tables. Means `DataSet and DataTable table stored in client memory(application memory area)` which is `small database`.
-

Q. How to get two table using DataSet with Store procuder?

- Create Store procudere:

```
create procedure spGetStd
as
begin
```

```
select * from Student
select * from Students
end

execute spGetStd
```

- Code to get two table using DataSet

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;

try
{
    string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
    SqlConnection db = new SqlConnection(cs);

    string query = "spGetStd";
    SqlDataAdapter sda = new SqlDataAdapter(query, db);
    sda.SelectCommand.CommandType = CommandType.StoredProcedure;
    DataSet ds = new DataSet();
    sda.Fill(ds);

    foreach (DataRow r in ds.Tables[0].Rows) //get data from 1st table
    {
        Console.WriteLine($"{r[0]} | {r[1]} | {r[2]} | {r[3]}");
    }

    Console.WriteLine("-----------------------------------------");

    foreach (DataRow r in ds.Tables[1].Rows) //get data from 2nd table
    {
        Console.WriteLine($"{r[0]} | {r[1]} | {r[2]} | {r[3]}");
    }
}
catch (Exception ex)
{
    Console.Write(ex.Message);
}
```

- Why not we use like theat `foreach (DataRow r in ds.Tables["Student"].Rows)`

Q. What happen when we try to access the table like that `foreach (DataRow r in ds.Tables["Student"].Rows)`

- Not work in Console app directly, it give's you error "Null refrence error". But you still nedd to access like that hear is all code:

```
string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
SqlConnection db = new SqlConnection(cs);

string query = "spGetStd";
SqlDataAdapter sda = new SqlDataAdapter(query, db);
sda.SelectCommand.CommandType = CommandType.StoredProcedure;
DataSet ds = new DataSet();
sda.Fill(ds);

ds.Tables[0].TableName = "Student"; //Use this line
ds.Tables[1].TableName = "Students"; //Use this line
foreach (DataRow r in ds.Tables["Student"].Rows)
{
    Console.WriteLine($"{r[0]} | {r[1]} | {r[2]} | {r[3]}");
}

Console.WriteLine("-----------------------------------------------");

foreach (DataRow r in ds.Tables["Students"].Rows)
{
    Console.WriteLine($"{r[0]} | {r[1]} | {r[2]} | {r[3]}");
}
```

- But `foreach (DataRow r in ds.Tables["Student"].Rows)` work on another .Net application like `Web form` and etc.

Q. How to work DataSet and DataTable together, without store procedure use seperate seperate query?

- Hear is the code:

```
using System;
using System.Data;
using System.Configuration;
using System.Data.SqlClient;


string cs = ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;
SqlConnection db = new SqlConnection(cs);

//Write both table query
string query1 = "Select * from Student";
string query2 = "Select * from Students";
```

```
//Object of both Data Adapter
SqlDataAdapter sda1 = new SqlDataAdapter(query1, db);
SqlDataAdapter sda2 = new SqlDataAdapter(query2, db);

//Object of both Data Table
DataTable std1 = new DataTable();
DataTable std2 = new DataTable();

//Fill the both Data Adapter in Data Table
sda1.Fill(std1);
sda2.Fill(std2);

//Create DataSet table object
DataSet ds = new DataSet();

//Store table in Data set (Add Data Table in Data set ) because DataSet store only
table.
ds.Tables.Add(std1);
ds.Tables.Add(std2);

//Show the table data
foreach (DataRow r in ds.Tables[0].Rows)
{
    Console.WriteLine($"{r[0]} | {r[1]} | {r[2]} | {r[3]}");
}

Console.WriteLine("-----------------------------------------------");

foreach (DataRow r in ds.Tables[1].Rows)
{
    Console.WriteLine($"{r[0]} | {r[1]} | {r[2]} | {r[3]}");
}
```

Here's the corrected comparison table between **DataSet** and **DataReader** with grammar and spelling improvements:

DataSet	**DataReader**
DataSet operates in a **disconnected** manner (automatically handles opening and closing the database connection).	DataReader is **connection-oriented** (requires manual opening and closing of the database connection).
DataSet is a collection of **in-memory tables** (tables are stored in client memory).	DataReader retrieves **read-only** and **forward-only** data from a database (cannot update or move backward in the data).
DataSet loads and holds all the data in the application memory after fetching it from the database.	DataReader fetches records **one by one** from the database and stores them in a network buffer, providing the data as requested.

DataSet	DataReader
Fetches **all rows at once**, storing all data from the data source in the application memory.	Fetches **one row at a time**, minimizing network traffic and reducing memory usage compared to DataSet.
Heavier compared to DataReader, as it loads all data into memory at once.	Lighter than DataSet, as it processes data row by row.
Allows **forward and backward** navigation, and random access to the records.	Only supports **forward-only** navigation, meaning you cannot fetch records in reverse or randomly.
Can fetch data from **multiple tables**.	Can only fetch data from **a single table** at a time.
Can maintain **relationships** between multiple tables.	No relationship management, as it works with a single table or data source.
Supports **transactions**.	Does **not** support transactions.
Works in a disconnected environment, automatically opening, fetching, and closing the connection when done.	Works in a connected environment, requiring the connection to stay open while reading data.
Requires **more memory** as it stores all data in memory.	Requires **less memory** since it stores data row by row.
Can be **serialized** and represented in XML, allowing it to be passed between different application layers (e.g., from one application to another).	Cannot be serialized.
Suitable for **data manipulation**, including updates, inserts, and deletes.	Suitable for **data display** (read-only usage).
Can be used in **Web** and **WCF services** because it can be serialized.	Cannot be used in Web and WCF services, as it is not serializable.
Best choice when you need to **navigate through data multiple times** or for operations like binding data to multiple controls.	Can only be **read once**, so it is suitable for binding to a single control and fetching data only once per read.

ADO.Net Using MVC

5:51

Q. How to preform CRUD operation from database using ADO.Net in MVC?

1. Create Database & Table in SQL Server Database.
2. Create MVC Web Application project, Empty & MVC, add connection string in Web.Config file.

```
<connectionStrings>
    <add name="dbCon"
         connectionString="Data Source=DESKTOP-HOOMVQE\MSSQLSERVER02; Initial
Catalog=ADO_Crud; User Id=mk; Password=123;"
         providerName="System.Data.SqlClient"/>
</connectionStrings>
```

3. Create model class(i.e:Student.cs) in side the model folder, which math the fields with Database table for create the object. According to your database tabe create property inside the class i.e:

```
using System;

namespace CRUDAppUsingADOWithWebApplication.Models
{
    public class Student
    {
        public int ID { get; set; }
        public string FName { get; set; }
        public string LName { get; set; }
        public int Age { get; set; }
        public string Gender { get; set; }
        public DateTime JoinDate { get; set; }
        public string UserName { get; set; }
        public string Password { get; set; }
        public bool Status { get; set; }
    }
}
```

4. Create new model(context) class(i.e:StudentDbContext.cs) in side the model folder, which preform operation(CRUD) with Database i.e:

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Configuration;


namespace CRUDAppUsingADOWithWebApplication.Models
{
    public class StudentDbContext
    {
        string cs =
ConfigurationManager.ConnectionStrings["dbCon"].ConnectionString;

        #region All Student
        public List<Student> GetStudents()
        {
            List<Student> students = new List<Student>();
```

```
            SqlConnection conn = new SqlConnection(cs);
            SqlCommand cmd = new SqlCommand("Select * from Student", conn);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                Student s = new Student();
                s.ID = (int)dr.GetValue(0);
                s.FName = dr.GetValue(1).ToString();
                s.LName = dr.GetValue(2).ToString();
                s.Age = (int)dr.GetValue(3);
                s.Gender = dr.GetValue(4).ToString();
                s.JoinDate = Convert.ToString(dr.GetValue(5));
                s.UserName = dr.GetValue(6).ToString();
                s.Password = dr.GetValue(7).ToString();
                s.Status = dr.GetValue(8).ToString();
                students.Add(s);
            }
            conn.Close();
            return students;
        }
        #endregion

        #region Single Student
        public Student GetStudents(int id)
        {
            Student s = null;
            SqlConnection conn = new SqlConnection(cs);
            SqlCommand cmd = new SqlCommand("Select * from Student where Id =
@id", conn);
            cmd.Parameters.AddWithValue("id", id);
            conn.Open();
            SqlDataReader dr = cmd.ExecuteReader();
            if (dr.Read())
            {
                //s = new Student
                //{
                //    ID = Convert.ToInt32(dr["Id"]),
                //    FName = dr["FName"].ToString(),
                //    LName = dr["LName"].ToString(),
                //    Age = Convert.ToInt32(dr["Age"]),
                //    Gender = dr["Gender"].ToString(),
                //    JoinDate = dr["JoinDate"].ToString(),
                //    UserName = dr["UserName"].ToString(),
                //    Password = dr["Password"].ToString(),
                //    Status = dr["Status"].ToString()
                //};

                //OR

                //s = new Student
                //{
                //    ID = Convert.ToInt32(dr.GetValue(0)),
                //    FName = dr.GetValue(1).ToString(),
```

```
                //    LName = dr.GetValue(2).ToString(),
                //    Age = Convert.ToInt32(dr.GetValue(3)),
                //    Gender = dr.GetValue(4).ToString(),
                //    JoinDate = Convert.ToString(dr.GetValue(5)),
                //    UserName = dr.GetValue(6).ToString(),
                //    Password = dr.GetValue(7).ToString(),
                //    Status = dr.GetValue(8).ToString()
                //};

                //OR

                s = new Student()
                {
                    ID = Convert.ToInt32(dr.GetValue(0)),
                    FName = dr.GetValue(1).ToString(),
                    LName = dr.GetValue(2).ToString(),
                    Age = Convert.ToInt32(dr.GetValue(3)),
                    Gender = dr.GetValue(4).ToString(),
                    JoinDate = Convert.ToString(dr.GetValue(5)),
                    UserName = dr.GetValue(6).ToString(),
                    Password = dr.GetValue(7).ToString(),
                    Status = dr.GetValue(8).ToString()
                };
            }
            return s;
        }
        #endregion

        #region Add Student
        public void AddStudent(Student s)
        {
            SqlConnection con = new SqlConnection(cs);
            //string que = "Insert into Student (ID, FName, LName, Age, Gender,
JoinDate, UserName, Password, Status) Values(@ID, @FName, @LName, @Age, @Gender,
@JoinDate, @UserName, @Password, @Status)";
            string que = "Insert into Student (ID, FName, LName, Age, Gender,
UserName, Password, Status) Values(@ID, @FName, @LName, @Age, @Gender, @UserName,
@Password, @Status)";
            SqlCommand cmd = new SqlCommand(que,con);
            cmd.Parameters.AddWithValue("@ID", s.ID);
            cmd.Parameters.AddWithValue("@FName", s.FName);
            cmd.Parameters.AddWithValue("@LName", s.LName);
            cmd.Parameters.AddWithValue("@Age", s.Age);
            cmd.Parameters.AddWithValue("@Gender", s.Gender);
            //cmd.Parameters.AddWithValue("@JoinDate", s.JoinDate);
            cmd.Parameters.AddWithValue("@UserName", s.UserName);
            cmd.Parameters.AddWithValue("@Password", s.Password);
            cmd.Parameters.AddWithValue("@Status", s.Status);
            con.Open();
            int rowsAffected = cmd.ExecuteNonQuery();
            if (rowsAffected == 0)
            {
                throw new Exception("Insert failed, no rows affected.");
            }
```

```
            con.Close();
        }
        #endregion

        #region Update Student
        public void UpdateStudent(Student s)
        {
            using(SqlConnection con = new SqlConnection(cs))
            {
                con.Open();
                string que = "Update Student set FName = @FName, LName = @LName,
Age = @Age, Gender = @Gender, " +
                    "UserName = @UserName, Password = @Password, Status = @Status
where ID = @ID";
                SqlCommand cmd = new SqlCommand(que, con);
                cmd.Parameters.AddWithValue("@FName", s.FName);
                cmd.Parameters.AddWithValue("@LName", s.LName);
                cmd.Parameters.AddWithValue("@Age", s.Age);
                cmd.Parameters.AddWithValue("@Gender", s.Gender);
                cmd.Parameters.AddWithValue("@UserName", s.UserName);
                cmd.Parameters.AddWithValue("@Password", s.Password);
                cmd.Parameters.AddWithValue("@Status", s.Status);
                cmd.Parameters.AddWithValue("@ID", s.ID);
                int result = cmd.ExecuteNonQuery();
                if (result == 0)
                    throw new Exception("Update cant not preform");
            }
        }
        #endregion

        #region Delete Student
        public void RemoveStudent(int id)
        {
            using(SqlConnection con = new SqlConnection(cs))
            {
                con.Open();
                string que = "Delete from Student where ID=@Id";
                SqlCommand cmd = new SqlCommand(que, con);
                cmd.Parameters.AddWithValue("@Id", id);
                int result = cmd.ExecuteNonQuery();
                if (result == 0) throw new Exception("Not Remove");
            }
        }
        #endregion
    }
}
```

5. Create controller(i.e:HomeController.cs) & create action method & create object of context class and use it.

```
using CRUDAppUsingADOWithWebApplication.Models;
using System.Collections.Generic;
using System.Web.Mvc;

namespace CRUDAppUsingADOWithWebApplication.Controllers
{
    public class HomeController : Controller
    {
        StudentDbContext dbContext = new StudentDbContext();
        public ActionResult Index()
        {
            List<Student> obj = dbContext.GetStudents();
            return View(obj);
        }
        public ActionResult GetStudent(int id)
        {
            Student student = dbContext.GetStudents(id);
            return View(student);
        }
        [HttpGet]
        public ActionResult AddStudent()
        {
            return View();
        }
        [HttpPost]
        public RedirectToRouteResult AddStudent(Student student)
        {
            dbContext.AddStudent(student);
            return RedirectToAction("Index");
        }

        [HttpGet]
        public ActionResult EditStudent(int id)
        {
            Student student = dbContext.GetStudents(id);
            return View(student);
        }

        [HttpPost]
        public RedirectToRouteResult EditStudent(Student student)
        {
            dbContext.UpdateStudent(student);
            return RedirectToAction("Index");
        }

        public RedirectToRouteResult DeleteStudent(int id)
        {
            dbContext.RemoveStudent(id);
            return RedirectToAction("Index");
        }

    }
}
```

Q. How to get connection string from Visual Studio?

Open any project in VS > View > Server Explorer > Right click on Data Connection > Add Connection > Fill Server Name & Other information > Ok > Go to you connection > right click select property > Rith hand side connection string is ther copp it.

Q.

Q.

Q.

Q.

Q.

Q.

Q.

Q.

Q.

Q.

Q.

Q.

Q.

Q.

Q.

Q.

Q.