Sample Question and answer from Chapter 5

1. What is ADO.NET? Advantage of ADO.NET

ADO.NET stands for ActiveX Data Object is a database access technology created by Microsoft as part of its .NET framework that can access any kind of data source. It's a set of object-oriented classes that provides a rich set of data components to create high-performance, reliable and scalable database applications for client- server applications as well as distributed environments over the Internet and intranets.

ADO .NET uses SQL queries and stored procedures to read write update and delete data from a data source. You use SQL queries through ADO.NET Command object, which returns data in the form of DataReader or DataSet objects.

ADO .NET uses SQL queries and stored procedures to read write update and delete data from a data source. You use SQL queries through ADO.NET Command object, which returns data in the form of DataReader or DataSet/DataTable objects.

You access a data source and fill a dataset via data providers. The .NET framework provides three different types of data providers: Sql, OleDb and ODBC. You use a DataAdapter object of a data provider and call its Fill method to fill a dataset.

Advantages of ADO .NET

Single Object- Oriented API

The ADO.NET provides a single object-oriented set of classes. There are different data providers to work with different data sources, but the programming model for all these data providers to work in the same way. So if you know how to work with one data provider, you can easily work with others. It's just a matter of changing class names and connection strings.

Managed Code

The ADO .NET classes are managed classes. They take all the advantages of .NET CLR, such as language independency and automatic resource management. All .NET languages access the same API. So if you know how to use these classes in C#, you'll have no problem using them in VB.NET. Another big advantage is you don't have to worry about memory allocation and freeing it. The CLR will take care of it for you.

Deployment

In real life, writing database application using ODBC, DAO, and other previous technologies and deploying on client machines was a big problem was somewhat taken care in ADO. Installing distributable .NET components will take care of it.

XML Support

Today, XML is an industry standard and the most widely used method of sharing data among applications over the Internet. As discussed earlier in ADO .NET data is cached and transferred in XML format.

Performance and Scalability

Performance and scalability are two major factors when developing web-based applications and services. Transferring data one source to another is a costly affair over the Internet because of connection bandwidth limitations and rapidly increasing traffic. Using disconnected cached data in XML takes care of both of these problems.

Connections and Disconnected data

With ADO .NET you use as few connections as possible and have more disconnected data. Both the ADO and ADO .NET models support disconnected data but ADO'S record set object wasn't actually designed to work with disconnected data. So there are performance problems with that. However, ADO.NET's dataset is specifically designed to work with disconnected data and you can treat a dataset as a local copy of a database. In ADO.NET, you store data in a dataset and close the make final changes to the data source.

DataReader versus DataSet/DataTable

The ADO.NET DataReader is used to retrieve read-only (cannot update data back to a datasource) and forward-only (cannot read backward/random) data from a database. You create a DataReader by calling Command.ExecuteReader after creating an instance of the Command object.

LINQ to DataSet

LINQ to DataSet API provides queries capabilities on a cached DataSet object using LINQ queries. The LINQ queries are written in C#

LINQ to SQL

LINQ to SQL API provides queries against relational databases without using a middle layer database library.

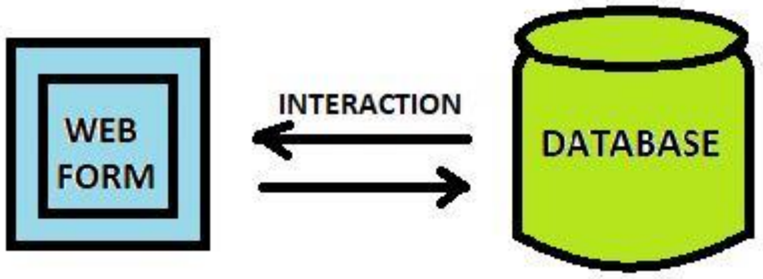
ADO.NET Entity Framework

The ADO.NET Entity Framework is designed to enable developers to create data access applications by programming against a conceptual application model instead of programming directly against a relational storage schema.

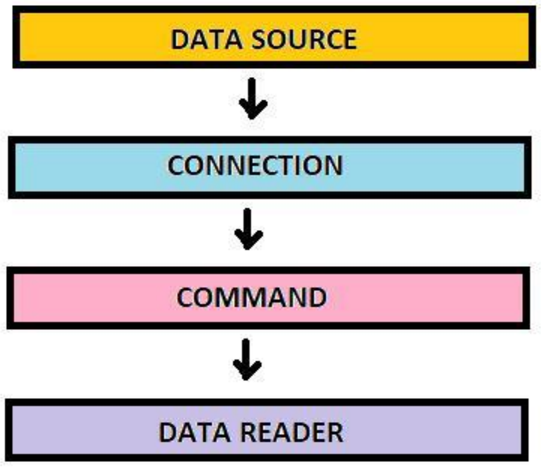
2. What is connected and disconnected architecture of Ado.net/Types of architecture in ado.net

Ado.net is both connection-oriented as well as disconnection oriented. Depending upon the functionality of an application, we can make it connection-oriented or disconnection oriented. We can even use both the modes together in a single application.

1. Connected Architecture



DataReader in Connected architecture



In order to make an object of the DataReader class, we never use the new keyword instead we call the ExecuteReader() of the command object. For e.g.

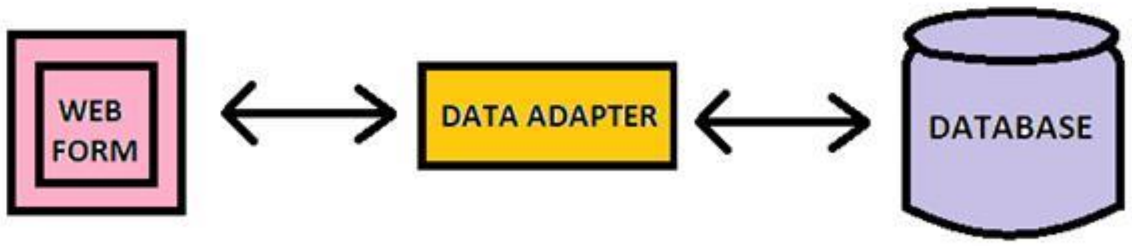
```
SqlCommand cmd= new SqlCommand("Select * from Table");

SqlDatareader rdr=cmd.ExecuteReader(cmd);
```

Here cmd.ExecuteReader() executes the command and creates the instance of DataReader class and loads the instance with data. Therefore, we do not use the 'new' keyword.

2. Disconnected Architecture

DataAdapter in Disconnected architecture



For example:-

```
 public DataTable GetTable(string query)

          {

            SqlDataAdapter adapter = new SqlDataAdapter(query, ConnectionString);

            DataTable Empl = new DataTable();

            adapter.Fill(Empl);

            return Empl;

          }
```

In the above lines of code, the object of the SqlDataAdapter is responsible for establishing the connection. It takes query and ConnectionString as a parameter. The query is issued on the database to fetch the data. ConnectionString allows connectivity with the database. The fill() of the SqlDataAdapter class adds the Table.

3.What is entity Framework core? What are the features of entity framework?

Entity Framework Core (EF Core) is the latest version of the Entity Framework from Microsoft. It has been designed to be lightweight, extensible and to support cross platform development as part of Microsoft's .NET Core framework. It has also been designed to be simpler to use, and to offer performance improvements over previous versions of Entity Framework.

EF Core is an object-relational mapper (ORM). Object-relational mapping is a technique that enables developers to work with data in object-oriented way by performing the work required to **map** between **objects** defined in an application's programming language and data stored in **relational** datasources.

Entity Framework Features

Cross-platform: EF Core is a cross-platform framework which can run on Windows, Linux and Mac.

Modelling: EF (Entity Framework) creates an EDM (Entity Data Model) based on POCO (Plain Old CLR Object) entities with get/set properties of different data types. It uses this model when querying or saving entity data to the underlying database.

Querying: EF allows us to use LINQ queries (C#/VB.NET) to retrieve data from the underlying database. The database provider will translate this LINQ queries to the database-specific query language (e.g. SQL for a relational database). EF also allows us to execute raw SQL queries directly to the database.

Change Tracking: EF keeps track of changes occurred to instances of your entities (Property values) which need to be submitted to the database.

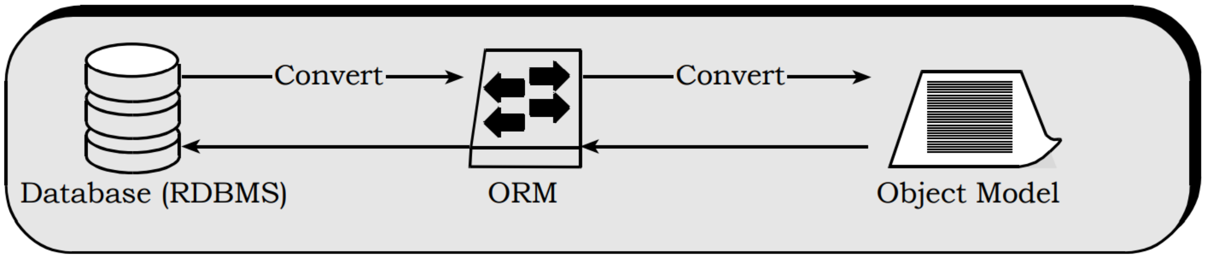
Saving: EF executes INSERT, UPDATE, and DELETE commands to the database based on the changes occurred to your entities when you call the `SaveChanges()` method. EF also provides the asynchronous `SaveChangesAsync()` method.

Transactions: EF performs automatic transaction management while querying or saving data. It also provides options to customize transaction management.

Caching: EF includes first level of caching out of the box. So, repeated querying will return data from the cache instead of hitting the database.

Migrations: EF provides a set of migration commands that can be executed on the NuGet Package Manager Console or the Command Line Interface to create or manage underlying database Schema.

4. Why ORM is used explain?



Most development frameworks include libraries that enable access to data from relational databases via recordset-like data structures. The following code sample illustrates a typical scenario where data is retrieved from a database and stored in an ADO.NET `DataTable` so that it is accessible to the program's code:

```
public DataTable GetAllStudent()
        {
            SqlConnection con = new SqlConnection("Data Source=(LocalDB)\\MSSQLLocalDB;
Integrated Security=True; Initial Catalog=AchsDB");
            SqlCommand cmd = new SqlCommand("select *from tblStudent", con);
            SqlDataAdapter da = new SqlDataAdapter(cmd);
            DataTable dt = new DataTable();
            da.Fill(dt);
            return dt;

        }
```

Strong Typing

When you take a strongly-typed approach to data, you work with properties of predefined classes that form a **domain model** in an object-oriented way:

```
public List<Student> GetAllStudentDataReader()
        {
            List<Student> lst = new List<Student>();
            SqlConnection con = new SqlConnection("Data Source=(LocalDB)\\MSSQLLocalDB;
Integrated Security=True; Initial Catalog=AchsDB");
            SqlCommand cmd = new SqlCommand("select *from tblStudent", con);
            SqlDataReader dr = null;
            con.Open();
            dr = cmd.ExecuteReader();
            while(dr.Read())
            {
                lst.Add(new Student() { StudentId = (int)dr["StudentId"], Fullname =
(string)dr["Fullname"], Email = (string)dr["Email"], Phone = (string)dr["Phone"] });
            }
            dr.Close();
            con.Close();
            return lst;

        }
```

5. CRUD Operation using Entity Framework in Asp.net Core, Console Application.
6. CUD Operation using ADO.Net in asp.net core, Console Application.
7. Create a Student form with Fullname,Email,Address,Phone Field and after submitting display entered record in form in another page in asp.net core.
8. Create a student form with Fullname,Email,Address,Phone validate each field with required validator both in client and server side.