Chapter 6 and 7

- **How ADO.NET differs from ADO**

ADO	ADO.NET
ADO is based on COM (Component Object Model) technology	ADO.NET is basically designed by .NET framework for smooth interaction of application and database
ADO works with connected data access architecture .That means when we access the data from data source such as viewing or updating data. ADO recordset is keeping connection with the data source.	ADO.NET works with both connected and disconnected data access architecture.ADO.NET only holds the connection open long enough to either to retrieve the data or to make any requested updates and immediately close the application.
ADO fits all connections to all types of data sources into a single connection object.	ADO.NET can have separate objects that represent connections to different data sources.
ADO recorsdet is a set of rows retrieved from a data source	ADO.NET represents in-memory representation of a database by using DataSet.
ADO recordset can hold data from one data source at a time.	ADO.NET DataSets can hold data from various data sources integrate the data and write back to one or several data sources.
ADO uses OLEDB to access data	ADO.NET uses XML and OLEDB as the format for transmitting data to and from your database and application.
ADO objects communicate in binary mode.	ADO.NET uses XML for passing data

- **Connected and disconnected data architecture:**

The ADO.NET framework supports two models of data access architecture. These are:

1. Connected data access architecture

2. Disconnected data access architecture

In connected data access architecture, **the application makes a connection to the datasource and then interact with it through SQL requests using the same connection**. In these cases, the application stays connected to the database system even when it is not using any database operations.

ADO.NET solves this problem by introducing a new component called Dataset. The Dataset is the central component in the ADO.NET disconnected data access architecture.

Dataset is an in-memory data store that can hold multiple tables at the same time.Datasets only hold data and do not interact with a data source.

In connected data access architecture, when we read data from a database by using a DataReader object, an open connection must be maintained between an application and the data source.

Unlike the DataReader ,the Dataset is not connected directly to a data source through a connection object when we populate it.

It is the DataAdapter that manages connections between data source and Dataset by filling the data from data source to the DataSet and giving a disconnected behavior to the DataSet. The DataAdapter acts as a bridge between the connected and disconnected objects.

By keeping connections open only for a minimum period of time,ADO.NET conserves system resources and provides maximum security for databases.

Creating Connection:

Connection string is a normal string representation which contains database connection information to establish the connection between database and the application.

The connection string includes parameters such as the name of the driver or provider, server name & database name as well as security information such as username and password.

Data providers use a connection string containing a collection of parameters to establish the connection with the database.

The .NET framework provides mainly three data providers:

1. Microsoft SQL server

2. OLEDB(Object Linking and Embedding DataBase)

3. ODBC(Open Database Connection)

Example:

```
OleDbConnection con;
con=new OleDbConnection ("provider=Microsoft.ACE.OLEDB.12.0;data source=e:\\JVM\\database1.accdb");
```

- **Managed data providers or ADO data providers**

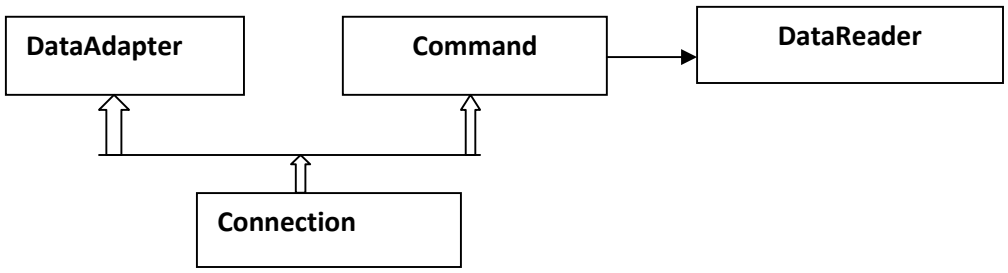
The .NET framework includes mainly three data providers for ADO.NET as follows:

1. Microsoft SQL server

2. OLEDB(Object Linking and Embedding DataBase)

3. ODBC(Open Database Connection)

Following links shows how these data providers making connection to the specified data sources:



The four objects from the .NET framework provide the functionality of data providers in ADO.NET. They are:

i) Connection object

ii) Command object

iii) DataReader object

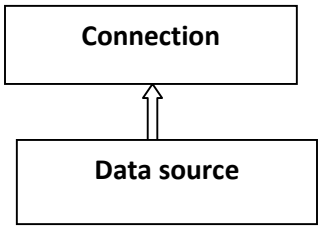
iv) DataAdapter object

i) Connection object

The connection object is handling the part of physical communication between the application and the data source.

The connection object connects to the specified database and open a connection between the application and the database.

When the connection is established, SQL commands may be executed to retrieve or manipulate data in the database. Once the database activity is over, connection should be closed and releases the resources.



The following are commonly used connections in ADO.NET :

i)sqlConnection

ii) OleDbConnection

iii)OdBcConnection

Example:

```
OleDbConnection con;
con=new OleDbConnection ("provider=Microsoft.ACE.OLEDB.12.0;data source=e:\\JVM\\database1.accdb");
```

ii) Command object

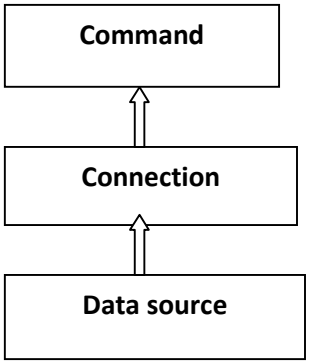
The command object in ADO.NET executes SQL statements and stored procedures against the data source specified in the connection object. The command object requires an instance of a connection object for executing the SQL statements. That is, for retrieving data or executing a SQL statement:

-Create a connection object

-Open a connection to the data source

-Assign the open connection to the connection property of the command object

When the command object return result set, a Datareader is used to retrieve the result set.



The command object has a property called CommandText which contains a string value that represents the command that will be executed in the data source.

Following are some important built-in methods used in the command object to execute the SQL statements.

-ExecuteNonQuery

-ExecuteReader

-ExecuteScalar

Example:

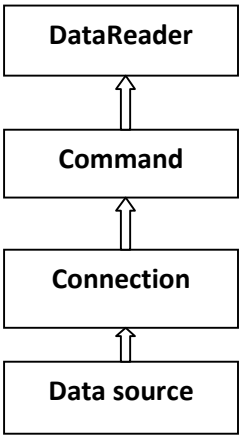
```
OleDbCommand cmd;
 cmd=new OleDbCommand ("insert into student values("+txtsrno.Text+",'"+txtsname.Text+ "', '"+
txtcourse.Text+"')",con);
            con.Open();
          int n=cmd.ExecuteNonQuery ();
          con.Close();
          if(n>0)
          {
             MessageBox.Show("Record inserted");
             loaddata();
          }

           else
             MessageBox.Show("insertion failed");

      }
```

iii) DataReader object

DataReader object in ADO.NET is stream-based forward only, read only retrieval of query results from the data source which do not update the data.

The DataReader cannot be created from code, they created only by calling the ExecuteReader method of a command object.

s



There are to types of DataReader in ADO.NET:

1. sqlDataReader

2. OleDbDataReader

Example:

OleDbDataReader dr=cmd.ExecuteReader();

A connection object can contain only one DataReader at a time & the connection in the DataReader remains open and cannot be used for any other purpose while the data is being accessed.

iv) DataAdapter object:

DataAdapter is a part of the ADO.NET data provider.

Dataadapter provides the communication between the Dataset and the data source. We can use the Dataadapter in combination with the Dataset object. These two objects combine to enable both data access and data manipulation capabilities.

The Dataadapter can perform select,update,insert and delete SQl operations in the data source.

The DataAdapter can perform select statement to fill a Dataset and use the other three SQL commands(insert, update & delete) to transmit changes back to the database.

Following are the types of DataAdapter:

sqlDataAdapter

OdBcDataAdapter

OleDbDataAdapter

Example:

```
OleDbDataAdapter adapter;
adapter=new OleDbDataAdapter ("select * from student",con);
 ds=new DataSet();
 adapter.Fill(ds,"student");
```

- **Advantages of ADO.NET**

ADO.NET is a new object-oriented data handling technology which is very different from the earlier ADO technology.ADO.NET provides greater ease of programming, higher performance, improved scalability, less dependence on the specific nature of each data source and greater ability to interact with other platforms because of XML.

1. Scalability

ADO.NET provides full support for disconnected data access. In this type of data access, most of the time the data we are working with is not connected to a data source. This means that ADO.NET uses database connection only for a short time. In many cases, the availability of less number of live database connections reduces the scalability. The web applications built on ADO.NET are highly scalable, as ADO.NET uses data connections only for a short duration of time.

Data source independence

The fundamental object that holds data in ADO.NET is an object of the class DataSet. This class is present in the namespace System.Data. It is an in-memory copy of whatever part of the database we have retrieved from a data source. This means that if we use a MS-SQL data source instead of a MS-Access data source, we don't have to learn anything new. The code which we write for accessing a MS-SQL data source is much similar to the code we have to write for accessing a MS-Access data source.

Interoperability

ADO.NET uses XML as its standard data transmission format. XML's interoperability feature enables .NET for cross platform, cross internet and flexible communication.

Performance:

ADO uses COM for the transmission of data between the client and the database server, a lot of processing time was needed for converting data between the types recognized in the database and the types that could be worked with COM .Only less processing time is needed for data conversions in the case of ADO.NET, as data transmission occurs via XML.

Firewalls

Usually one point of access is set in local network. All traffic into or out of the local network is checked at such a point. The hardware and software that sits between the Internet and the local network, checking all the data that comes into the network or out, to make it is safe, is called a firewall. Firewalls are mostly configured to reject COM packets, So ADO data cannot get through them. On the other hand, XML data can easily pass through most firewalls.

- **Developing ADO.NET based application:**

  Refer experiment no. 13 in record book

- **Insert,update and delete operations on table**

  Refer experiment no. 14 in record book

- **Filling the dataset**

  Refer experiment no. 13 and 14 in record book