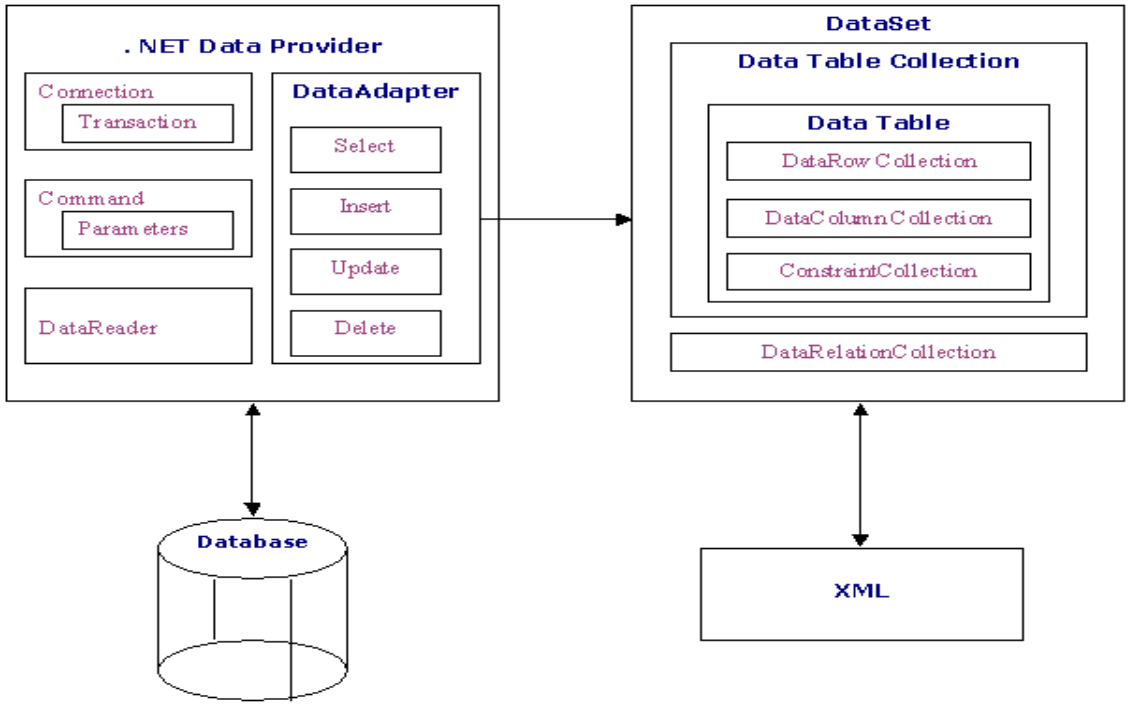
The ADO.NET Data Architecture

The ADO.NET is designed to work with multiple kinds of data sources in same fashion. You can categorize ADO.NET components in two categories: disconnected and the .NET data providers. The disconnected components build the basic ADO.NET architecture. You can use these components (or classes) with or without data providers.

The data provider components are specifically designed to work with different kinds of data sources. For example, ODBC data providers work with ODBC data sources and OleDb data providers work with OLE-DB(object linking and embedding database) data sources and Sql data providers work with MSSQL data sources and OracleDB data providers work with Oracle-DB data sources.

Data Access in ADO.NET relies on two components: DataSet and Data Provider.



ADO .NET Data Architecture

DataSet

The dataset is a disconnected, in-memory representation of data. It can be considered as a local copy of the relevant portions of the database. The DataSet is persisted in memory and the data in it can be manipulated and updated independent of the database. When the use of this DataSet is finished, changes can be made back to the central database for updating.

Data Provider

The Data Provider is responsible for providing and maintaining the connection to the database. A DataProvider is a set of related components that work together to provide data in an efficient and performance driven manner. The .NET Framework currently comes with Four DataProviders: the SQL Data Provider, ODBC Data Provider, OLEDB Data Provider, Oracle Data Provider. Each DataProvider consists of the following component classes:

The Connection object which provides a connection to the database
The Command object which is used to execute a command
The DataReader object which provides a forward-only, read only, connected recordset
The DataAdapter object which populates a disconnected DataSet with data and performs updated using the command object or the DataAdapter.

Component classes that make up the Data Providers

The Connection Object

The Connection object creates the connection to the database. Microsoft Visual Studio .NET provides two types of Connection classes: the SqlConnection object, which is designed specifically to connect to Microsoft SQL Server 7.0 or later, and the OleDbConnection object, which can provide connections to a wide range of database types like Microsoft Access and Oracle. The Connection object contains all of the information required to open a connection to the database.

The Command Object

The Command object is represented by two corresponding classes: SqlCommand and OleDbCommand. Command objects are used to execute commands to a database across a data connection. The Command objects can be used to execute stored procedures on the database, SQL commands, or return complete tables directly. Command objects provide three methods that are used to execute commands on the database:

ExecuteNonQuery: Executes commands that have no return values such as INSERT, UPDATE or DELETE
ExecuteScalar: Returns a single value from a database query
ExecuteReader: Returns a result set by way of a DataReader object

The DataReader Object

The DataReader object provides a forward-only, read-only, connected stream recordset from a database. Unlike other components of the Data Provider, DataReader objects cannot be directly instantiated. Rather, the DataReader is returned as the result of the Command object's ExecuteReader method. The SqlCommand.ExecuteReader method returns a SqlDataReader object, and the OleDbCommand.ExecuteReader method returns an OleDbDataReader object. The DataReader can provide rows of data directly to application logic when you do not

need to keep the data cached in memory. Because only one row is in memory at a time, the DataReader provides the lowest overhead in terms of system performance but requires the exclusive use of an open Connection object for the lifetime of the DataReader.

The DataAdapter Object

The DataAdapter is the class at the core of ADO .NET's disconnected data access. It is essentially the middleman facilitating all communication between the database and a DataSet. The DataAdapter is used either to fill a DataTable or DataSet with data from the database with it's Fill method. After the memory-resident data has been manipulated, the DataAdapter can commit the changes to the database by calling the Update method. The DataAdapter provides four properties that represent database commands:

SelectCommand
InsertCommand
DeleteCommand
UpdateCommand

When the Update method is called, changes in the DataSet are copied back to the database and the appropriate InsertCommand, DeleteCommand, or UpdateCommand is executed.

Use of ADO.NET objects

There are two ways to access & manipulate data of database. First method is visually (with graphical tools) & second method is via coding (using ADO.NET objects directly in coding).

ADO .NET objects are **Connection Object, Command object, DataAdapter object, DataSet objects, DataReader Object, DataTable Object, DataRow Object, DataColumn Object, etc.**

(1) Connection Objects:

The Connection object provides connectivity (physical connection) to a data source (database). Using its method, you can open & close the connection, change the database & manage transactions.

The Connection object is the first component of ADO.NET. A connection sets a link between a data source and ADO.NET. A Connection object sits between a data source and a DataAdapter (via Command).

Connection class exist for **ODBC** (OdbcConnection), **OLE DB** (OledbConnection), **SQL Server** (SqlConnection) & **Oracle** (OracleConnection)

Property :

(1) ConnectionString: It is the string that is used to connect (open) a database when the open method is executed.

ConnectionString property of OledbConnection object has arguments like Provider, Data Source, Database, User ID, Password.

(2) ConnectionTimeout : The maximum time the Connection object attempts to make the connection before throwing an exception. (before terminating the attempt and generating an error). By default is 15 seconds.

(3) DataSource : It is used to specify the server name of the computer on which the database is running. When connecting to an access database, this specifies the path & database name.
(4) State : Gets (returns) the current state of the connection. For example we get Closed, if the connection is closed. & we get Open, if the connection is open.
MsgBox(Con.State.ToString)
(5) ServerVersion : Gets the version of the server.
(6) Provider : This property represents the name of the provider.

Provider parameter specifies the driver that uses to communicate with the database. The most common drivers are **Microsoft.Jet.OLEDB.4.0** for Access, **SQLOLEDB** for SQL server & **MSDAORA** for Oracle.

Method :

(1) Open : Opens a database connection with the property settings specified by the ConnectionString.
(2) Close : Closes the connection to the data source. After the connection is close no transaction can be perform on the database data.

```
Con.Close( )
Con.Dispose( )     'Releases the resources used by the connection object.
Con = Nothing      'Release your reference to the connection object
```

(3) BeginTransaction : Starts (begins) a database transaction.

(2) Command Objects:

The Command object is used to execute SQL statements (Select, Insert, Update & Delete) as well as stored procedure. In addition to the DML statements, you can also execute DDL statements that change the structure of the database.

You can also use the **Parameters** collection in the Command class to pass parameters to stored procedures or SQL statements.

Command object exist for ODBC (OdbcCommand), OLE DB (OledbCommand), SQL Server (SqlCommand) & Oracle (OracleCommand).

Property :

1. **CommandText:** It is the string, contains either SQL statements or name of the stored procedure to be executed.
2. **CommandType:** It represents the type of the **Command** object. Depending upon the command type Command object executes the command. The different command types are as follows.

   StoredProcedure : The name of a stored procedure.
   TableDirect : The name of a table.
   Text : SQL statements. (Default)

For example: Cmd.CommandType = CommandType.**Text**

(3) **Connection:** The name of the active Connection object, through which the command is to be executed.
(4) **Parameters:** The parameters property contains a collection of parameters for the SQL statements or stored procedure.

Methods:

(1) **ExecuteNonQuery** : Executes commands that do not return data rows. But it returns number of rows affected by the commands. (Such as **SQL INSERT**, **DELETE**, **UPDATE**, and **SET** statements).
(2) **ExecuteScalar :** Calculates and returns a single value, such as a sum, min, max from a database. Used for aggregate function.
(3) **ExecuteReader :** Executes SQL commands that return rows. ExecuteReader method is used to create data reader.
(4) **Cancel :** Cancels the execution of the command.

(3) Data Adapter Objects :

The DataAdapter object provides the bridge between the DataSet object and the data source (database) for retrieving and saving data.

The DataAdapter's sole purpose is to retrieves data from the database, then populates (fill) the Datasets & also used to send (propagate) the Datasets changes to the database. (The **DataAdapter** object has **Fill** method to load data from the data source into the dataset, and the **Update** method to send changes you've made in the dataset back to the data source).

The DataAdapter contains four command objects: SelectCommand, InsertCommand, UpdateCommand, and DeleteCommand. The DataAdapter uses the SelectCommand to fill a DataSet & the remaining three commands to transmit changes back to the data source.

Data adapter object exist for ODBC (ODBCDataAdapter), OLE DB (OleDbDataAdapter), SQL Server (SqlDataAdapter) & Oracle (OracleDataAdapter).

Property :

(1) **SelectCommand:** The name of the Command object used to retrieve rows from the data source.
(2) **InsertCommand:** The name of the Command object used to insert rows in the data source.
(3) **UpdateCommand:** The name of the Command object used to update rows in the data source.
(4) **Delete Command:** The name of the Command object used to delete rows in the data source.

Methods :

(1) Fill:- The Fill method which loads data from the data source (database) into the Dataset. If the Connection is closed before Fill is called, it is opened to retrieve data and then closed.

Syntax 1:
OleDbDataAdapter.Fill(DataSet)
Syntax 2:
OleDbDataAdapter.Fill(DataTable)
Syntax 3:

OleDbDataAdapter.Fill(DataSet, TableName)
(2) Update: Update method is used to send changes you've made in the dataset back to the data source (database).

(4) DataSet Objects :

It is the major component of ADO.NET. DataSet is a memory-resident representation of data. **A dataset is a disconnected cache of data, and, that is stored in memory.** DataSet is always disconnected from the data source. It can contain data from multiple sources.

As we have seen in ADO.NET object model, The DataSet composed of two primary objects: the **DataTableCollection**, accessed through Tables property, and **DataRelationCollection** accessed through the Relations property. The DataTableCollection contains zero or more DataTable objects, which are in turn made up of three collections: DataColumnColection, DataRowCollection, and ConstraintCollection. The DataRelationCollection contains zero or more DataRelation objects.

The dataset is a disconnected, in-memory representation of data. An advantage of this is that we do not need to have a continuous connection to the database.

Property :
(1) Relations : It is the collection of DataRelation objects, which defines the relationship of the DataTables within the dataset.
(2) Tables : It is the collection of DataTables contained in the dataset.

Method :
(1) AcceptChanges : Accepts (Commits) all the pending changes made to the dataset.
(2)RejectChanges: Roll back all changes pending in the DateSet. Rolls back the changes made to the dataset since it was created or since the **AcceptChanges** method was called.
(3) Copy : Copies the structure & contents of the DataSet.
(4) Clear : Empties all the tables in the DataSet.
(5) Reset: Returns the DataSet back to its original state.

(5) DataReader Objects:

The DataReader in addition to datasets, there are also datareaders, which are extremely fast, read-only, forward only low-overhead way of retrieving information from the database. You can only move through records with in ascending order means you cannot go backward. ExecuteReader method of a command object is used to create data reader.

DataReader is appropriate when you are processing rows individually and then discarding them. For example, Report generators, you get a performance benefit from **DataReader. DataReader** is best suited for retrieving huge amounts of data, as the data is not cached in the memory.

Property :
(1) FieldCount: Gets the number of columns in the current row.
(2) HasRows: Returns a Boolean value indicating whether the DataReader contains rows of data.
(3) IsClosed: Indicates whether the DataReader is closed.

(4) Item : Gets the value of a column(field). For example If employee table has three fields, EmpNo, EmpName & City, then EmpNo is Item(0), EmpName is Item(1) & City is Item(2).

Method :

(1) Close : Closes the data reader.
(2) GetName : Gets (returns)the name of the specified column.
(3) GetValue : Gets a field's value (column's value) in its native format.
(4) GetValues : Gets all columns in the current row.
(5) IsDBNull : Indicates if a column contains nonexistent (or missing) values.
(6) Read : Read method returns true if there are more rows. It advances the DataReader to the next record.

(6) DataTable Objects :

Datasets are made up of **DataTable** objects. Data in the DataSet is stored in memory in the form of DataTable Objects.

The DataTableCollection contains zero or more DataTable objects, which are in turn made up of three collections: DataColumnColection, DataRowCollection, and ConstraintCollection (used to ensure integrity of data , ForeignKeyConstraint & UniqueConstraint)

(7) DataRow Objects :

DataRow objects represent rows in a **DataTable** object. You use **DataRow** objects to get access to, insert, delete, and update the records in a table.

(8) DataColumn Objects :

DataColumn objects represent the columns, that is, the fields, in a data table.

(9) DataRelation Objects :

The **DataRelation** class supports data relations between data tables. The DataRelationCollection contains zero or more DataRelation objects. It is accessed through the Relations property.

Data Binding :

It is the process of attaching the Data source to a data aware control is called as Data Binding.

.NET supports two types of Data Binding.

(1) Simple Data Binding &
(2) Complex Data Binding.

(1) Simple Data Binding : **In Simple Data Binding, only a single value (one data element) from a DataSet can be bound to control at a time.** In this type of Data Binding, any value from the DataSet can be bound to any property of a control.

Simple Binding at design time :

For example To bind EmpName field of DataSet11, do as follows.

First select the TextBox control, then exapand its **DataBinding** property in the Properties window. Then click on Text property, then click on drop down arrow. The property window display a list of data sources that you can bind to the TextBox.(As shown in figure). For example **DataSet11 - Emp.EmpNo**

Simple Binding in code (at run time)

At run time, we can use a control's **DataBindings** property. It has **Add** method.

```
TextBox1.DataBindings.Add("Text", DataSet11, "Emp.
EmpName")
```

(2) Complex Data Binding : In Complex Data Binding, a control is bound to a complete DataSet (more than one data element) instead of just one column from it.

Most controls support only simple data binding but some controls, such as DataGridView, ComboBox, ListBox & CheckedListBox support complex data binding.

For complex data binding, following properties can be use.

(a) DataSource : It is the name of the data source, for example **DataSet11**.

(b) DataMember : It is the name of the table in a dataset such as the **Employee** table, the DataGridView should display.

(c) DisplayMember : It is the name of the field, that will be display on the control.

(d) ValueMember : It is the name of the field (for example dept no), its corresponding value will be return to the **SelectedValue** property.

Example : To display department names in combo box, we have to set following properties at run time.

```
ComboBox1.DataSource = DataSet11
ComboBox1.DisplayMember = "Dept.DName"
```

Now to get the respective department No, When Department name is selected, you have to set **ValueMember** property to DeptNo.

```
ComboBox1.ValueMember = "Dept.DeptNo"
```

You can access the Department No using the **SelectedValue** property.

```
MsgBox(ComboBox1.SelectedValue)
```