Unit-6 Database Programming with ADO.NET

ADO .NET:

The full form of ADO is **ActiveX Data Object**. ADO.NET is a very important feature of .NET Framework, which is used to work with data that is stored in structured data sources, such as databases and XML files.

ADO.NET is an object-oriented set of libraries that allows you to interact with data sources.

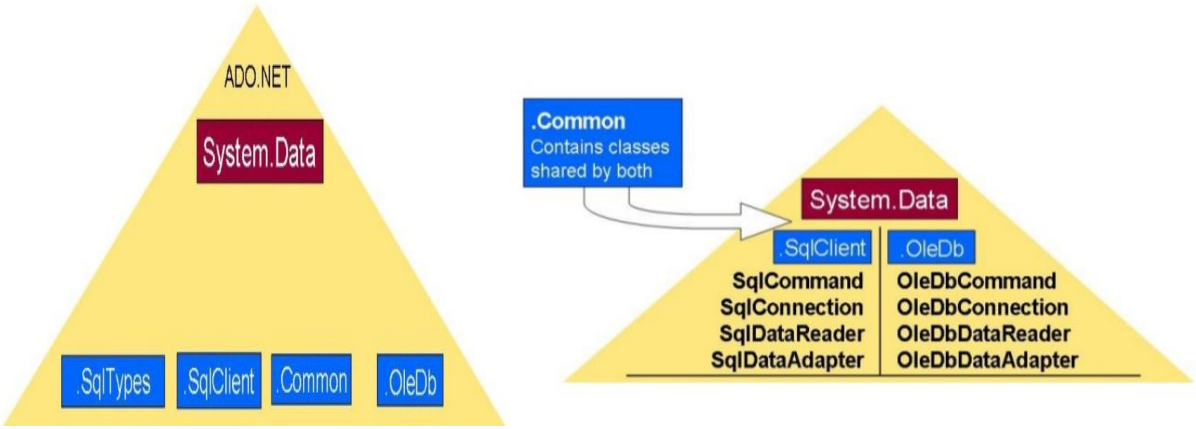
Commonly, the data source is a database, but it could also be a text file, an Excel spreadsheet, or an XML file.

ADO.NET provides a relatively common way to interact with data sources, but comes in different sets of libraries for each way you can talk to a data source. These libraries are called Data Providers and are usually named for the protocol or data source type they allow you to interact with.

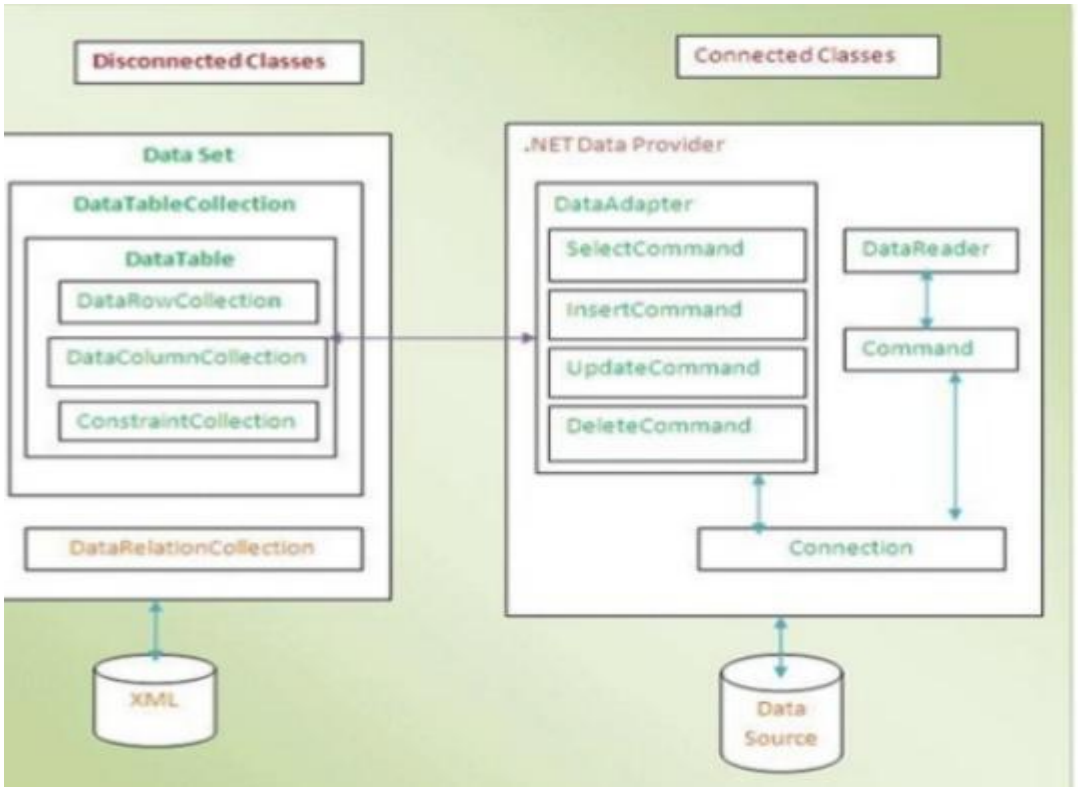
ADO.NET Data Providers

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

ADO.NET-related Namespaces



ADO.Net Architecture



ADO.NET Objects

• **Connection** Object: The connection helps identify the database server, the database name, user name, password, and other parameters that are required for connecting to the data base. A connection object is used by command objects so they will know which database to execute the command on.

• **Command** Object: Use a command object to send SQL statements to the database. A command object uses a connection object to figure out which database to communicate with.

• **DataReader** Object : The data reader object allows you to obtain the results of a SELECT statement from a command object. For performance reasons, the data returned from a data reader is a fast forward-only stream of data.

DataSet Object: DataSet objects are in-memory representations of data. They contain multiple Datatable objects, which contain columns and rows, just like normal database tables. You can even define relations between tables to create parent-child relationships. The DataSet is an object that is used by all of the Data Providers, which is why it does not have a Data Provider specific prefix.•

DataAdapter Object : The data adapter fills a DataSet object when reading the data and writes in a single batch when persisting changes back to the database. A data adapter contains a reference to the connection object and opens and closes the connection automatically when reading from or writing to the database.

• **CommandBuilder** : By default dataadapter contains only the select command and it doesn'tcontain insert, update and delete commands. To create insert, update and delete commands for the dataadapter, commandbuilder is used. It is used only to create these commands for the dataadapter and has no other purpose.

SqlConnection Object

• The connection tells the rest of the ADO.NET code which database it is talking to. It manages all of the low-level logic associated with the specific database protocols.

• Declare and instantiate the SqlConnection as shown below:

SqlConnection conn = new SqlConnection("Data Source=(local); InitialCatalog=Northwind;Integrated Security=SSPI");

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

- **ADO.NET Connection String Parameters**

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

SqlCommand&SqlDataReader Object

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

• Creating a SqlCommand Object:

SqlCommandcmd = new SqlCommand("select CategoryName from Categories", conn);

• A **SqlDataReader** is a type that is good for reading data in the most efficient manner possible. You cannot use it for writing data. SqlDataReaders are often described as fast- forward firehose-like streams of data.

• You can read from SqlDataReader objects in a forward-only sequential manner. Once youve read some data, you must save it because you will not be able to go back and read it again.

• Creating a SqlDataReader Object:-

Getting an instance of a SqlDataReader is a little different than the way you instantiate other ADO.NET objects. You must call ExecuteReader on a command object, like this:

SqlDataReaderrdr = cmd.ExecuteReader();

1) **ExecuteReader -Querying Data**
   • Used to retrieve data records for viewing. It returns a SqlDataReader object.
   Example:
   1. Instantiate a new command with a queryand connection
      SqlCommandcmd = new SqlCommand("select CategoryName from Categories", conn);
   2. Call Exucute reader to get query results
      SqlDataReaderrdr = cmd.ExecuteReader();

2) **ExecuteNonQuery (Insert/Update/Delete)**

To insert data into a database, use the ExecuteNonQuery method of the SqlCommand object.

//prepare command string

String insertString = @" insert into Categories (CategoryName, Description) values ('ABC','XZY')";

Instantiate a new command with a query and connectionSqlCommandcmd = new SqlCommand (insertString,conn);

Call ExecuteNonQuery to send command

cmd.ExecuteNonQuery();

Similarly,ExecuteNonQuery can be used to update a record using below query string

//prepare command string

```
SqlCommandcmd = new SqlCommand("update tblLogin set UserName = 'user1' where UserName='test'", con);
cmd.ExecuteNonQuery();
```

Similarly ExecuteNonQuery can be used to delete a record using below query string

//prepare command string

```
SqlCommandcmd = new SqlCommand("delete from tblLogin where UserName='xyz'", con);
cmd.ExecuteNonQuery();
```

3) **ExecureScalar-Getting single values**

Used to ret count, sum, average, or other aggregated value from a database.

1. Instantiate a new command
   SqlCommandcmd = new SqlCommand("select count(*) from Categories",conn);
2. Call ExecuteScalarto send command
   Int count = (int)cmd.ExecuteScalar();

The query in the SqlCommand constructor contains the count of all records from the Categories table. This query will only return a single value.

Working with Disconnected Data

- **DataSet and SqlDataAdapter**

- A DataSet is an in-memory data store that can hold numerous tables. DataSets only hold data and do not interact with a data source.

- It is the SqlDataAdapter that manages connections with the data source and gives us disconnected behavior. The SqlDataAdapter opens a connection only when required and closes it as soon as it has performed its task.

- For example, the SqlDataAdapter performs the following tasks when filling a DataSet with data:
- Open connection
- Retrieve data into DataSet
- Close connection

- and performs the following actions when updating data source with DataSet changes:
- Open connection
- Write changes from DataSet to data source
- Close connection

- In between the Fill and Update operations, data source connections are closed and you are free to read and write data with the DataSet as you need.

Adding Parameters to Commands

- Using parameterized queries is a three step process:

1) Construct the SqlCommand command string with parameters.
   SqlCommand cmd = new SqlCommand("select * from Customers where city = @City", conn);

1) Declare a SqlParameter object, assigning values as appropriate.
   SqlParameter param = new SqlParameter(); param.ParameterName = "@City"; param.Value = inputCity;

3) Assign the SqlParameter object to the SqlCommand object's Parameters property.
   cmd.Parameters.Add(param);

4) Below is complete console application example double click to open.

Using Stored Procedures

- A stored procedures is a pre-defined, reusable routine that is stored in a database.
- SQL Server compiles stored procedures, which makes them more efficient to use.
- Therefore, rather than dynamically building queries in your code, you can take advantage of the reuse and performance benefits of stored procedures.

- **Executing a Stored Procedure**

```
// 1. create a command object identifying the stored procedure
SqlCommand cmd = new SqlCommand( "<<StoredProcName>>", conn);
// 2. set the command object so it knows to execute a stored procedure
cmd.CommandType = CommandType.StoredProcedure;
```

- **Sending Parameters to Stored Procedures**

```
SqlCommand cmd = new SqlCommand( "CustOrderHist", conn);
// 2. set the command object so it knows to execute a stored procedure
cmd.CommandType = CommandType.StoredProcedure;
// 3. add parameter to command, which will be passed to the stored procedure
cmd.Parameters.Add( new SqlParameter("@CustomerID", custId));
```

- The name of the parameter passed as the first parameter to the SqlParameter constructor must be spelled exactly the same as the stored procedure parameter.

DataGridView

The DataGridView control is designed to be a complete solution for displaying tabular data with Windows Forms. The DataGridView control is highly configurable and extensible, and it provides many properties, methods, and events to customize its appearance and behaviour.

The DataGridView control makes it easy to define the basic appearance of cells and the display formatting of cell values. The cell is the fundamental unit of interaction for the DataGridView. All cells derive from the DataGridViewCell base class. Each cell within the DataGridView control can have its own style, such as text format, background color, foreground color, and font. Typically, however, multiple cells will share particular style characteristics. The data type for the cell's Value property by default is of type Object.

The GridView control displays the values of a data source in a table. Each column represents a field, while each row represents a record. The GridView control supports the following features:

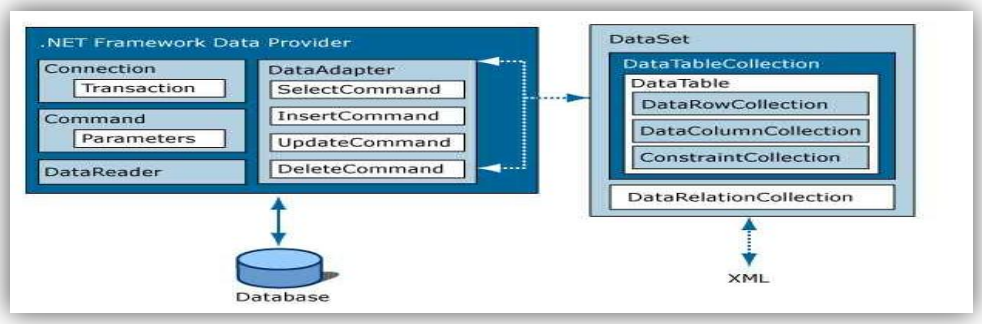
- Binding to data source controls, such as SqlDataSource.
- Built-in sort capabilities.
- Built-in update and delete capabilities.
- Built-in paging capabilities.
- Built-in row selection capabilities.
- Programmatic access to the GridView object model to dynamically set properties, handle events, and so on.
- Multiple key fields.
- Multiple data fields for the hyperlink columns.
- Customizable appearance through themes and styles.

Difference between connected and disconnected architecture

1. In disconnected architecture a **DataSet** is used for retrieving data from the database. Then no need for maintaining the connection also. All the operations can be performed with the data once retrieved. It won't cause traffic problems while working with the data.
   In connected architecture a **DataReader** is used for retrieving data from the database. Here a connection is always maintained. Update, Delete, Read and Select operations can be performed as the data is accessed in the database, so that a connection must be be maintained. This can cause traffic problems.
2. Connected data needs connection to be created to access hence slower while disconnected is in memory data that's faster access.
3. Connected you need to use a read only forward only data reader, disconnected you cannot.

Difference between connected and disconnected architecture

CONNECTED	DISCONNECTED
It is connection oriented.	It is dis_connection oriented.
Connected methods gives faster performance	Disconnected get low in speed and performance.
Datareader	DataSet
connected can hold the data of single table	disconnected can hold multiple tables of data
connected you need to use a read only forward only data reader	disconnected you cannot
Data Reader can't persist the data	Data Set can persist the data
It is Read only, we can't update the data.	We can update data



```
String StrSQL = "", StrConnection = "";
      protected void Page_Load(object sender, EventArgs e)
      {
         StrSQL = "SELECT * FROM Student";
         StrConnection = "Data
Source=ServerName;InitialCatalog=Database;User
ID=Username;Password=password";
      }

      protected void Connected_Click(object sender, EventArgs e)
      {
         using (SqlConnection objConn
= new SqlConnection(StrConnection))
         {
            SqlCommand objCmd = new SqlCommand(StrSQL, objConn);
            objCmd.CommandType = CommandType.Text;
            objConn.Open();
            SqlDataReader objDr = objCmd.ExecuteReader();
            GridView1.DataSource = objDr;
            GridView1.DataBind();
            objConn.Close();
         }
      }

      protected void Disconnected_Click(object sender, EventArgs e)
      {
         SqlDataAdapter objDa = new SqlDataAdapter();
         DataSet objDs = new DataSet();
         using (SqlConnection objConn
= new SqlConnection(StrConnection))
         {
            SqlCommand objCmd = new SqlCommand(StrSQL, objConn);
            objCmd.CommandType = CommandType.Text;
            objDa.SelectCommand = objCmd;
            objDa.Fill(objDs, "Student");
            GridView1.DataSource = objDs.Tables[0];
            GridView1.DataBind();
         }
      }
```