ADO.NET (ActiveX Data Object)

- Data access technology from Microsoft .Net Framework.
- Provides communication between relational and non-relational systems.
- A set of computer software components.
- Used by programmers to access data.
- Commonly used by programmers to access and modify data stored in relational database system.
The two key components of ADO.NET are:

- Data Providers and,
- DataSet.

The .Net Framework includes mainly three Data Providers for ADO.NET:

- The Microsoft SQL Server,
- OLEDB and,
- ODBC.
The following **four Objects** from the .Net Framework provide the **functionality of Data Providers** in ADO.NET.

- **Connection Object**: provides **physical connection** to the Data Source.
- **Command Object**: perform **SQL statement** or stored procedure to be executed at the Data Source.
- **DataReader Object**: a stream-based, forward-only, read-only retrieval of query results from the Data Source, which **do not update** the data.
- **DataAdapter Object**: populate a **Dataset Object** with results from a Data Source.
The Connection Object

■ The first step is to create connection, that is to create a connection object.

■ Properties
  - ConnectionString
  - ConnectionTimeOut
    ■ Gets or sets the time to wait while trying to establish a connection before terminating the attempt and generating an error.
  - Database
  - DataSource
  - State
    ■ Gets the current state of the connection.
Chapter 8 – ADO.NET

- Methods
  - Open
  - Close
  - BeginTransaction
  - ChangeDatabase

- Events
  - StateChange
Setting the **connectionString** property:

```
DIM tableconnection AS STRING = "provider = microsoft.jet.oledb.4.0 ;" & "data source= database_path;"
DIM cn AS NEW OleDbConnection()
Cn.connectionstring = tableconnection
Cn.open
```
The Command Object

Properties

- **CommandText** → contain SQL text
- **CommandType** → value which specify the type of query: Text or stored procedure.
- **Connection**
- **CommandTimeOut**
- **parameters**
Methods

- **ExecuteReader** → execute the **select query** specified by the **commandText**.
- **ExecuteNonQuery** → execute the **action query** specified by the **commandText**.
ExecuteReader

- It is a method of the command object.
- This method return a DataReader object, which you can use to read the resultSet one row at a time.
- Send the SQL statements to OleDbConnection Object and populate an OleDbDataReader Object based on the SQL statement.
- When the ExecuteReader method execute, it instantiate an OleDb.OleDbDataReader Object.
- The Read() method in the OleDbDataReader is used to read the rows from OleDbDataReader and it always moves forward to a new valid row, if any row exist.
ExecuteNonQuery

- Used for executing statements that do not return result set.
- Performs Data Definition tasks as well as Data Manipulation tasks also.
- The Data Definition tasks like creating Stored Procedures and Views. Also Data Manipulation tasks like Insert, Update and Delete.
Imports System.Data.OleDb
Public Class Form1
    Dim dataconnect As New OleDbConnection()
    Dim datacmd As OleDbCommand
    Dim dr As OleDbDataReader
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim tableconnection As String = "provider=microsoft.jet.oledb.4.0;" & "data source=c:\practice.mdb;"
        dataconnect.ConnectionString = tableconnection
        dataconnect.Open()
    End Sub
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    datacmd = New OleDbCommand("SELECT * FROM student", dataconnect)
    dr = datacmd.ExecuteReader
    dr.Read()
    TextBox1.Text = dr.Item("name")
    TextBox2.Text = dr.Item("address")
   TextBox3.Text = dr.Item("dateOfBirth")
End Sub
End Class
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Do While dr.Read()
            TextBox1.AppendText(dr.Item("stname") & ControlChars.CrLf)
        Loop
    End Sub
End Class
Dim sql As String = "SELECT * FROM students WHERE stname = 'safi' "
Dim cmd As New OleDbCommand(sql, cn)
Dim dr As OleDbDataReader = cmd.ExecuteReader(CommandBehavior.SingleRow)
Dr.Read()
TextBox1.Text = dr("stname")
Dr.Close()
Chapter 8 – ADO.NET

Imports System.Data.OleDb
Public Class Form1
    Dim connection As New OleDbConnection
    Dim command As OleDbCommand
    Private Sub Form1_Load(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles MyBase.Load
        connection.ConnectionString = "provider=microsoft.jet.oledb.4.0;" & "data source=c:\practice.mdb;"
        connection.Open()
        command = New OleDbCommand("SELECT count(qtype) FROM Exam", connection)
        Dim reccount As Integer = CInt(command.ExecuteScalar())
        TextBox1.Text = reccount
    End Sub
End Class
اگر خواسته باشیم تا تعداد ریکورد های مشخص را معلوم نماییم:

command = New OleDbCommand("SELECT count(*) FROM Exam where qtype = 'جغرافیه', connection")
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    connection.ConnectionString = "provider=microsoft.jet.oledb.4.0;"
    & "data source=c:\BCS4.mdb;"
    connection.Open()
    newcommand = New OleDbCommand("SELECT count(fname) FROM student where address = " & TextBox4.Text & ","
    connection)
    Dim reccount As Integer = CInt(newcommand.ExecuteScalar())
    TextBox3.Text = reccount
    connection.Close()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    connection.ConnectionString = "provider=microsoft.jet.oledb.4.0;"
    & "data source=c:\BCS4.mdb;"
    connection.Open()
    newcommand = New OleDbCommand("SELECT count(fname) FROM student where address = "" & ComboBox1.Text & """,
    connection)
    Dim reccount As Integer = CInt(newcommand.ExecuteScalar())
    TextBox3.Text = reccount
    connection.Close()
End Sub
cn.Open()
Dim cmd As OleDbCommand
cmd = New OleDbCommand("SELECT * FROM exam", cn)
Dim dr As OleDbDataReader = cmd.ExecuteReader
Do While dr.Read
    Qnum = Qnum + 1
Loop
cn.Close()
ExecuteScalar

- Used for get a single value from Database.
- It executes SQL statements or Stored Procedure and returned a scalar value on first column of first row in the Result Set.
- If the Result Set contains more than one columns or rows, it takes only the first column of first row, all other values will ignore.
- It is very useful to use with aggregate functions like Count(*) or Sum() etc.
مثال ذیل استفاده از میتود ExecuteScalar را نشان می‌دهد:

```vbnet
Dim sql As String = "SELECT stname FROM students WHERE stid = '1235' "
Dim cmd As New OleDbCommand(sql, cn)
Dim pubname As String = cmd.ExecuteScalar().ToString

استفاده دیگر این میتود خواندن نتیجه تابع Aggregate می‌باشد.
Dim cmd As New OleDbCommand("SELECT COUNT(*) FROM students", cn)
Dim reccount As Integer = Cint(cmd.ExecuteScalar())
```

هر حالت اولین فیلد سطر اول را برگشت می‌دهد.
DataSet

Contains the copy of the data we requested through the SQL statement.

We can use Dataset in combination with OleDbDataAdapter class.

The OleDbDataAdapter object allows us to populate Data Tables in a DataSet. We can use Fill method in the OleDbDataAdapter for populating data in a Dataset.

DataSet contains DataTableCollection and their DataRelationCollection. It represents a collection of data retrieved from the Data Source.
The **DataSet** object offers a disconnected data source architecture.

The **Dataset** gives a better advantage over DataReader, because the **DataReader** is working only with the connection oriented Data Sources.

The **Dataset** contains more than one Table at a time. We can set up Data Relations between these tables within the **DataSet**.
Some properties of DataSet:

- Datasetname
- Tables
- Relations
- Local
- Prefix
- casesensitive
Some methods of DataSet

- Acceptchanges
- Rejectchanges
- Haschanges
- Merge
- Clone
- Copy
- clear
Data Source

Connection

Command

DataAdapter

DataSet
DataAdapter

A part of the ADO.NET Data provider.

Allows us to populate DataTables in a DataSet.

We can use Fill method of the DataAdapter for populating data in a Dataset. The DataSet can be filled either from a data source or dynamically. A DataSet can be saved to an XML file and then loaded back into memory very easily.

DataAdapter Provides the communication between the Dataset and the DataSource.

We can use the DataAdapter in combination with the DataSet Object. That is these two objects combine to enable both data access and data manipulation capabilities.
- **In charge** of **filling** one or more **DataTable** objects with data and work in a **completely disconnected mode**

- After the **end user** has performed all the **editing**, the application can **reopen** the **connection** and **reuse** the same **DataAdapter** object to send changes to the databases.

- It can fill a **DataTable** with **data** taken from any **data source**, SQL server, Access, text file, or a **mainframe**, and **process** it with the same **routines** regardless of its origin.

- The **architecture** based on the **DataSet** and **DataAdapter** makes it possible to **read data** from **one source** and **send updates to another source**.
The **DataAdapter** can perform:

- Select,
- Insert,
- Update and Delete SQL operations.

The **Insert**, **Update** and **Delete** SQL operations, we are using the *continuation* of the **Select command** perform by the **DataAdapter**. That is the **DataAdapter** uses the Select statements to fill a **DataSet** and use the other three SQL commands (**Insert**, **Update**, **delete**) to transmit changes back to the **Database**.
Imports System.Data.OleDb

Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim connectionString As String
        Dim connection As OleDbConnection
        Dim oleDbAdapter As OleDbDataAdapter
        Dim ds As New DataSet
        Dim sql As String
        Dim i As Integer
        connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=Your mdb filename;"
        sql = "Your SQL Statement Here"
connection = New OleDbConnection(connetionString)
Try
    connection.Open()
    oledbAdapter = New OleDbDataAdapter(sql, connection)
    oledbAdapter.Fill(ds)
    oledbAdapter.Dispose()
    connection.Close()
    For i = 0 To ds.Tables(0).Rows.Count - 1
        MsgBox(ds.Tables(0).Rows(i).Item(0) & "  --  " & ds.Tables(0).Rows(i).Item(1))
    Next
Catch ex As Exception
    MsgBox("Can not open connection ! ")
End Try
End Sub
End Class
In some situations we have to find how many tables inside the Dataset Object contains. The following VB.NET source code shows how to find the tables inside the Dataset.
Imports System.Data.OleDb

Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim connectionString As String
        Dim connection As OleDbConnection
        Dim oleDbAdapter As OleDbDataAdapter
        Dim ds As New DataSet
        Dim sql As String
        Dim i As Integer
        connectionString = "Provider=Microsoft.Jet.OLEDB.4.0;Data Source=Your mdb filename;"
        sql = "Your SQL Statement Here"
connection = New OleDbConnection(connectionString)
    Try
        connection.Open()
        OleDbAdapter = New OleDbDataAdapter(sql, connection)
        OleDbAdapter.Fill(ds, "OLEDB Temp Table")
        OleDbAdapter.Dispose()
        connection.Close()
        For i = 0 To ds.Tables.Count - 1
            MsgBox(ds.Tables(i).TableName)
        Next
    Catch ex As Exception
        MsgBox("Can not open connection ! ")
    End Try
End Sub
End Class
Note: it is important to close the DataReader object to release resources on both the client and the server and make the connection available again for other commands. You can’t issue any other command on a connection while a DataReader object is active on that connection.

The following VB.NET source code shows how to find the number of rows in a table that resides in the Dataset.
Imports System.Data.OleDb
Public Class Form1
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim connectionString As String
        Dim connection As OleDbConnection
        Dim command As OleDbCommand
        Dim adapter As New OleDbDataAdapter
        Dim ds As New DataSet
        Dim sql As String
        connectionString = "provider=microsoft.jet.oledb.4.0;" 
        & "data source=c:\practice.mdb;"
        sql = "Your SQL Statement Here"
        connection = New OleDbConnection(connectionString)
Try
  connection.Open()
  command = New oledbCommand(sql, connection)
  adapter.SelectCommand = command
  adapter.Fill(ds, "SQL Temp Table")
  adapter.Dispose()
  command.Dispose()
  connection.Close()
  MsgBox("Number of row(s) - " & ds.Tables(0).Rows.Count)
Catch ex As Exception
  MsgBox("Can not open connection ! ")
End Try
End Sub
End Class
To Create Table

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim TableCreate As String = "CREATE TABLE thistable(" & "stName varchar(20)," & "lastName varchar(20)," & "score Byte);"

    newcommand = New OleDbCommand(TableCreate, connection)

    connection.Open()
    newcommand.ExecuteNonQuery()
    connection.Close()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim abc As Object
    abc = TextBox3.Text
    Dim TableCreate As String = "CREATE TABLE " & abc & "(") & 
    "stName varchar(20)," & _ 
    "lastName varchar(20)," & "score Byte);"
    newcommand = New OleDbCommand(TableCreate, connection)
    connection.Open()
    newcommand.ExecuteNonQuery()
    connection.Close()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    Dim abc As Object
    abc = InputBox("Enter Table Name :")
    Dim TableCreate As String = "CREATE TABLE " & abc & "(" & 
    "stName varchar(20)," & _ 
    "lastName varchar(20)," & "score Byte);" 
    newcommand = New OleDbCommand(TableCreate, connection)
    connection.Open()
    newcommand.ExecuteNonQuery()
    connection.Close()
End Sub
To Delete a Table

Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    Dim DeleteTable As String = "DROP TABLE thistable;"
    newcommand = New OleDbCommand(DeleteTable, connection)
    connection.Open()
    newcommand.ExecuteNonQuery()
    connection.Close()
End Sub
To Insert data into Table

Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    Dim sql As String = "insert into thistable (stName, lastName, score) values ('Wais', 'Habib', 95)"
    newcommand = New OleDbCommand(sql, connection)
    connection.Open()
    newcommand.ExecuteNonQuery()
    connection.Close()
End Sub
Private Sub Button4_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button4.Click
    Dim sql As String = "insert into thistable (stName, lastName, score) values ('" & TextBox1.Text & ", " & TextBox2.Text & ",95)"
    newcommand = New OleDbCommand(sql, connection)
    connection.Open()
    newcommand.ExecuteNonQuery()
    connection.Close()
End Sub
در مثال ذیل نام و لايت از یک ComboBox گرفته ميشود:

Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    connection.ConnectionString = "provider=microsoft.jet.oledb.4.0;"
 & "data source=c:\BCS4.mdb;"
    connection.Open()
    newcommand = New OleDbCommand("SELECT count(fname) FROM student where address = "" & ComboBox1.Text & """, connection)
    Dim reccount As Integer = CInt(newcommand.ExecuteScalar())
    TextBox3.Text = reccount
    connection.Close()
End Sub
Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
    connection.ConnectionString = "provider=microsoft.jet.oledb.4.0;" & "data source=c:\BCS4.mdb;"
    connection.Open()
    newcommand = New OleDbCommand("SELECT count(fname) FROM student where address = " & TextBox4.Text & "", connection)
    Dim reccount As Integer = CInt(newcommand.ExecuteScalar())
    TextBox3.Text = reccount
    connection.Close()
End Sub

در مثال ذیل نام ولایت از یک textbox گرفته می‌شود:
کنترول `DataGridView` به یک جدول وصل به یک `BindingSource` استفاده می‌شود. این خاصیت دارای مفیدی است که از وصل شدن ارتباط می‌باشد. این کنترول از طریق خاصیت `Data Source` به یک `Data Source` ارتباط می‌زند. بعد از `Bound` شدن ویژه‌ی این کنترول به شکل `Data Source` و اتصال به یک `Method` به‌صورت `Fill` به‌صورت آنالوگ به مثال ذیل توجه نمایید:

-
Imports System.Data.OleDb
Public Class Form1
    Private bindingSource1 As New BindingSource()
    Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
        dgv1.Dock = DockStyle.Fill
        dgv1.AutoGenerateColumns = True
        bindingSource1.DataSource = GetData("Select * From exam")
        dgv1.DataSource = bindingSource1
        dgv1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders
        dgv1.BorderStyle = BorderStyle.Fixed3D
        dgv1EditMode = DataGridViewEditMode.EditOnEnter
    End Sub
Private Shared Function GetData(ByVal sqlCommand As String) As DataTable
    Dim connectionString As String = "provider=microsoft.jet.oledb.4.0;" & "data source= c:\practice.mdb;"
    Dim northwindConnection As OleDbConnection = New OleDbConnection(connectionString)
    Dim command As New OleDbCommand(sqlCommand, northwindConnection)
    Dim adapter As OleDbDataAdapter = New OleDbDataAdapter()
    adapter.SelectCommand = command
    Dim table As New DataTable
    adapter.Fill(table)
    Return table
End Function
End Class
Imports System.Data.OleDb
Public Class Form1
    Private connectionString As String = 
        "provider=microsoft.jet.oledb.4.0;" & "data source= c:\practice.mdb;"
    Private newconnection As OleDbConnection = New OleDbConnection(connectionString)
    Private newcommand As New OleDbCommand
    Private adapter As OleDbDataAdapter = New OleDbDataAdapter()
    Private bindingSource1 As New BindingSource()
    Private newtable As New DataTable
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button3.Click
    dgv1.Dock = DockStyle.Fill
    dgv1.AutoGenerateColumns = True
    newcommand = New OleDbCommand("select * from exam", newconnection)
    Dim adapter As OleDbDataAdapter = New OleDbDataAdapter()
    adapter.SelectCommand = newcommand
    adapter.Fill(newtable)
    bindingSource1.DataSource = newtable
    dgv1.DataSource = bindingSource1
    dgv1.AutoSizeRowsMode = DataGridViewAutoSizeRowsMode.DisplayedCellsExceptHeaders
    dgv1.BorderStyle = BorderStyle.Fixed3D
    dgv1.EditMode = DataGridViewEditMode.EditOnEnter
End Sub
End Class
Reading Data

There are 3 ways to read data from a table:

- Using the `ExecuteReader` Method
- Using `DataReader` Object
- Using `ExecuteScalar` Method
- Using `ExecuteXmlReader` method
- Using `XmlReader` object
DataReaders

A DataReader is a lightweight object that provides read-only, forward-only data in a very fast and efficient way. Using a DataReader is efficient than using a DataAdapter but it is limited. Data access with DataReader is read-only, meaning, we cannot make any changes (update) to data and forward-only, which means we cannot go back to the previous record which was accessed. A DataReader requires the exclusive use of an active connection for the entire time it is in existence. We instantiate a DataReader by making a call to a Command object's ExecuteReader command. When the DataReader is first returned it is positioned before the first record of the result set. To make the first record available we need to call the Read method. If a record is available, the Read method moves the DataReader to next record and returns True. If a record is not available the Read method returns False. We use a While Loop to iterate through the records with the Read method.
Creating a Data Table

Imports System.Data.OleDb
Public Class Form1
    Dim ds As New DataSet("mydataset")
    Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button1.Click
        Dim dtemp As New DataTable("Employee")
        dtemp.MinimumCapacity = 100
        dtemp.CaseSensitive = False
        Dim dcfname As New DataColumn("firstname", GetType(String))
        dtemp.Columns.Add(dcfname)
        Dim dclname As New DataColumn("lastname", GetType(String))
        dtemp.Columns.Add(dclname)
        ds.Tables.Add(dtemp)
    End Sub
End Class
Private Sub Button2_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles Button2.Click
    Dim dtemp As DataTable = ds.Tables("employee")
    Dim dr As DataRow = dtemp.NewRow()
    dr("firstname") = "Adelyar"
    dr("lastname") = "Hassan"
    dtemp.Rows.Add(dr)
    TextBox1.Text = dr("firstname")
End Sub
Private Sub Button3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs)
Handles Button3.Click
    Dim dtemp As DataTable = ds.Tables("employee")
    Dim sr As New System.IO.StreamReader("C:\employee.dat")
    Dim filetext As String = sr.ReadToEnd
    sr.Close()
    Dim re As New System.Text.RegularExpressions.Regex("(?<fname>\[^"\]+);(?<lname>\[^"\]+)"")
    Dim ma As System.Text.RegularExpressions.Match
    dtemp.BeginLoadData()
    For Each ma In re.Matches(filetext)
        Dim values() As Object = {ma.Groups("fname").Value, ma.Groups("lname").Value}
        dtemp.LoadDataRow(values, True)
    Next
    dtemp.EndLoadData()
    Dim dr As DataRow = dtemp.Rows(1)
    TextBox1.Text = dr("firstname")
End Sub
Dim I as integer
For I = 0 to 9
    Dim dr as DataRow = dtemp.Rows(i)
    Dr(“firstname”) = dr(“firstname”).ToString().ToUpper
    Dr(“lastname”) = dr(“lastname”).ToString().ToUpper
Next