What is ADO.NET?

**ADO.NET** is a part of the Microsoft .NET Framework. This framework provides the set of classes that deal with data communication between various layers of the software architecture and the database. It provides a continious access to different data source types such as SQL Server versions 7, 2000, 2005. It also provides connectivity options to data sources through OLE DB and XML. Connectivity may be established with other databases like Oracle, MySQL etc. as well.  
  
ADO.NET has the ability to separate data access mechanisms, data manipulation mechanisms and data connectivity mechanisms.  
  
ADO.NET introduces along with it the **disconnected** architecture. In a disconncted architecture, data may be stored in a **DataSet**. It contains providers for connecting to databases, commands for execution and retrieval of results.   
  
The classes for ADO.NET are stored in the DLL System.Data.dll.   
  
For connection strings, [Click Here](http://sqlconnect.googlepages.com)

What is a connection object in ADO.NET? How to connect to a database in .Net?

A **Connection object** in ADO.NET is used to establish a connection between a program (the program may be a windows page, a web page, a windows service, a web service etc.) and the database. The connection is open just long enough to get or update data. By quickly opening, then closing a connection, the server resources are used as little as possible. See code below on how to open a connection between UI and database...

'Code below in VB.NET ...   
Dim objectConn as SqlClient.SqlConnection  
Dim strConn as String  
Try  
'First, create a connection object   
objectConn=New SqlClient.SqlConnection()  
  
'Next, build the Connection String  
strConn &="Data Source=(local  
strConn &="Initial Catalog=DatabaseName;"  
strConn &= "User ID=admin;"  
strConn &= "Password=;"  
  
'Note here that the connection string may also be passed as a parameter   
'to the connection string object during instantiation  
  
objectConn.ConnectionString = strConn   
objectConn.Open() 'Open the Connection  
  
'The connection is now open  
'Write your vb.net code here for operations on the database  
objectConn.Close()  
  
Catch Ex As Exception  
MessageBox.Show(Ex.Message)  
End Try

[**CONNECTION STRINGS FOR ACCESS, SQL 2000, SQL 2005, Oracle, MySQL**](http://sqlconnect.googlepages.com) **What are Connection Strings** - A connection string has a group of semi-colon-separated attributes. Every .Net Data Provider connection string looks different, depending on the type of .NET Data Provider you need to use and which attributes are set for each different type of database system. An example, the connection string below is an example of what you use to connect to a local SQL Server. See that every parameter is separated by a semicolon.  
  
Data Source=(local);Initial Catalog=Northwind;User ID=sa;Password=;  
  
The connection string shown below is an example of how to connect to a Microsoft Access 2000 database using the OleDbConnection object in System.Data.OleDb.  
  
Provider=Microsoft.Jet.OleDb.4.0;Data Source=C:\Northwind.mdb   
  
**Parameters in a Connection String** - The parameters depend on the data provider is being used.   
  
**Server** - The name of the SQL Server to which connection has to be established through ADO.NET. This is the name of the system that is running SQL server. We may use "local" or "localhost" for local computer. In case we are using named instances of SQL server, then the parameter would contain the computer name, followed by a backslash, followed by a named instance of the SQL server.   
  
**Database** - The name of the database to which connection is to be established.  
  
**User ID** - A user ID configured in the SQL Server by the SQL Server administrator.   
  
**Password** - As the attribute name suggests, this is the password associated with the user id.   
  
Note that connection string may also contain the Windows NT account security settings. This is done by passing the paramater "integrated security=true".

What is a command object in ADO.NET. How to use a command object in .NET?

**ADO.NET Command Object** - The Command object is similar to the old ADO command object. It is used to store SQL statements that need to be executed against a data source. The Command object can execute SELECT statements, INSERT, UPDATE, or DELETE statements, stored procedures, or any other statement understood by the database. See sample code...

'Code below in VB.NET ...   
Dim ObjCom as SqlClient.SqlCommand  
ObjCom.SqlConnection(strCon)  
ObjCom.Connection.Open()  
ObjCom.CommandText = "Select \* from tblSample"  
ObjCom.ExecuteNonQuery()

SqlCommand objects are not used much when we use datasets and data adapters. Following are some properties of the SqlCommand class...  
  
**Connection Property** - This property contains data about the connection string. It must be set on the SqlCommand object before it is executed. For the command to execute properly, the connection must be open at the time of execution.   
  
**CommandText Property** - This property specifies the SQL string or the Stored Procedure.

objCom.CommandText = "Insert into Employees (empid, empname) values ('EMI0334','Mandy')"

**Paramaters Collection** - If we want to update values in the Employees table above, but we do not know the values at design time, we make use of placeholders. These are variables prefixed with "@" symbol. Our code will look like this...

objCom.CommandText = "Insert into Employees (empid, empname) values (@empid, @empname)

Next, we have to create parameters that will be used to insert values into the placeholders. For this, we need to add parameters to the parameters collection of the SqlCommand object. This is done so that the values added through the parameters collection & placeholders get i ncluded in the SQL statement. Here, parameters mean the parameters to be passed to the SQL statement/Stored Procedures, not the method's parameters.   
  
In order to add parameters to the SqlCommand object, we write the following code...

objCom.CommandText = "Insert into Employees (empid, empname) values (@empid, @empname)"  
  
objCom.Parameters.Add("@empID", txtempid.text)  
  
objCom.Parameters.Add("@empname", txtempname.text)

**ExecuteNonQuery Method** - Once the connection is open, we run the query in the SqlCommand object using the ExecuteNonQuery method. This is very simple as shown below...

objConnection.Open()  
objCom.ExecuteNonQuery()  
objConnection.Close()

What is SelectCommand in ADO.NET?

**Select Command Property** - This property is used to hold the SQL command that is used to retrieve data from the data source. The CommandText and Connection are properties of the Select Command propety. CommandType is also a property of Select Command. See example below...

Dim da as new SqlDataAdapter  
da.SelectCommand = New SqlCommand( )  
With da.SelectCommand   
.Connection = objConnection   
.CommandText = "select \* from employees"  
End With

**What is a SqlCommandBuilder in ADO.NET**

**SqlCommandBuilder** class in ADO.NET provides the feature of reflecting the changes made to a DataSet or an instance of the SQL server data. When an instance of the SqlCommandBuilder class is created, it automatically generates Transact-SQL statements for the single table updates that occur. The object of the SqlCommandBuilder acts as a listener for RowUpdating events, whenever the DataAdapter property is set.   
  
The SqlCommandBuilder object automatically generates the values contained within the SqlDataAdapter's InsertCommand, UpdateCommand and DeleteCommand properties based on the initial SelectCommand. The advantage here is that you will not need to write SqlCommand & SqlParameter Types explicitly.   
  
Basically the command builder object builds these objects on the fly. The command builder object actually reads the metadata of the method called. After the builder object reads the underlying schema of the adapter's method, it autogenerates an underlying insert, update & delete command object. See code example below...

//Code below in C#...   
DataSet ds = new DataSet();   
SqlConnection cn = new SqlConnection("strSomeConnectionString");  
//Autogenerate Insert, Update & Delete commands   
SqlDataAdapter da = new SqlDataAdapter("Select from t\_Something", cn);  
SqlCommandBuilder scb = new SqlCommand(da);  
  
//Fill the dataset  
da.Fill(ds,"t\_Something");

What is a DataView in ADO.NET?

**DataView** - Just like we have Views in SQL (in our backend), we have DataView object in ADO.NET. A dataview object represents bindable, customized view of a DataTable object. Operations like Sorting, Filtering, Searching, Editing and Navigation may be performed on a DataView object. In scenarios like retrieval of a subset of data from a Datatable, we may make use of DataViews to get this data. Note that the **DefaultView** property of a DataTable returns the Default data view for the DataTable. In case a custom view of a DataTable has to be created on a DataView, then the **RowFilter** property of the DataView is set to the DefaultView.   
  
A dataview may also be used to sort data that resides in it in ascending or descending order. Below is code on how to sort data in a dataview in ascending or descending order...

DataView objdv = new DataView();  
objdv.Sort("ColumnName Asc|Desc");

What is DataRelation object in ADO.NET? How to use a DataRelation between two columns in ADO.NET?

In order to set the relationship between two or more than two columns, ADO.NET provides the **DataRelation** class. When a DataRelation object is created, it assists to enforce some constraints on the relationships between columns. The constraint may be like a **Unique constraint** that ensures that a column will have no duplicate value in the table. A **Foreign Key** constraint may be used to enforce **Referential Integrity**. The Unique property may be set by setting the Unique property of a DataColumn to True. This may also be done by adding an instance of the UniqueConstraint class to the DataRelation object. As a part of the foreign key constraint, we may specify referential integrity rules that are applied at 3 places   
  
1) When a parent record is updated  
2) When a parent record is deleted   
3) When a change is rejected or accepted.   
  
A DataRelation object permits to establish a parent-child relationship between two or more tables inside a DataSet object. The easiest way to create a DataRelation between two tables in a DataSet is to setup a primary key - foreign key relationship between the columns of a table.   
  
See code example below, where a DataRelation has been setup between the Employee table and the Salary table...

'Code Below in VB.NET   
Dim Conn As SqlConnection  
Dim da As SqlDataAdapter  
Dim ds As DataSet  
Dim RowParent As DataRow  
Dim RowChild As DataRow  
  
Conn = New \_  
SqlConnection(ConfigurationSettings.Appsettings("SomeConnectionStringWrittenInWeb.Config"))  
da = New SqlDataAdapter("SELECT \* FROM Employees", Conn)  
ds = New DataSet()  
  
Try  
   Conn.Open()  
   da.Fill( ds,"Employees")  
   da.SelectCommand = New SqlCommand("SELECT \* FROM Salary", Conn)  
   da.Fill(ds, "Salary")  
Catch ex As SqlException  
   Response.Write(ex.ToString())  
Finally  
   Conn.Dispose()  
End Try   
  
'Next, Let us create a Data Relationship  
ds.Relations.Add("Employee\_Salary", ds.Tables("Employees").Columns("EmployeeID"), \_  
ds.Tables("Salary").Columns("EmployeeID"))  
'Display the Employee and Child Salary in the Form  
'Say we have a Label in the form  
For each RowParent in ds.Tables("Employees").Rows  
  
   lblRelation.Text &= RowParent("Emp\_Name")   
   For each RowChild in RowParent.GetChildRows("Employee\_Salary")  
     lblRelation.Text &= "<br/>" & RowChild("Sal\_Amount")  
   Next  
Next

What is Diffgram in ADO.NET? When do we use Diffgram?

A **DiffGram** is an XML format. It is used to identify current and original versions of data elements. A **DataSet** may use a DiffGram format to load and persist the contents, and further to serialize its contents for porting across a network connection. Whenever a DataSet is written as a DiffGram, the DataSet populates the DiffGram with all the important information to accurately recreate the contents. Note that schema of the DataSet is not recreated. This includes column values from both the Current and the Original row versions, row error information, and row order.

What is a Typed Dataset in ADO.NET? Why do we use a Typed DataSet? What is the difference between a Typed and an UnTyped DataSet?

**Typed DataSet** - When a created DataSet derives from the DataSet class, that applies the information contained in the XSD to create a Typed class, this DataSet is said to be a Typed Dataset. Information from the schema whic comprises the tables, columns, and rows is created and compiled to a new DataSet derived from the XSD. The Typed DataSet class features all functionality of the DataSet class. This may be used with methods that take an instance of the DataSet class as a parameter.   
  
Note that an **UnTyped DataSet** does not have any schema. It is exposed simply as a mere collection.   
  
**How to create a Typed DataSet?** -   
Write click your project in the Solution Explorer.  
Click Add New Item.  
Select DataSet.  
  
This adds a new XSD to the project. The schema created may be viewed as an XML. When this xsd file is compiled, two files are created by Visual Studio. The first file that contains the .vb or .cs extension contains the information about the proxy class. This class contains methods & properties that are required to access the database data. The second file has an extension **xsx** and this contains information about the layout of the XSD.

How to run a stored procedure from .NET code?

Use the **Command** object and its **ExecuteNonQuery** method to run a DDL or DCL statement of SQL. First, create a connection object and a command object, and configure these objects for the statement you wish to execute. Open the database connection. Call ExecuteNonQuery on the command. Close the connection.

**Example**  
Suppose we have to insert FirstName into a table t\_Employees. Our Store procedure is:   
  
Create Procedure dbo.InsertFirstName   
(  
@FirstName varchar(30)  
)  
as  
Insert Into t\_Employees values (@FirstName)  
  
**VB.NET code below, to pass FirstName to this Stored Procedure**  
  
Dim cmRunProc as New SqlCommand("dbo.InsertFirstName", objCon)  
cmRunProc.CommandType = CommandType.StoredProcedure  
cmRunProc.Parameters.Add("@FirstName", txtFirstName.Text)  
objCon.Open()  
cmdRunProc.ExecuteNonQuery()  
objCon.Close()