BCA Semester-5
US05CBCA01-Visual Programming through VB.NET

UNIT- 4 Database Programming with ADO.NET

Introduction to ADO.NET

ADO.NET is a protocol i.e. a set of standards to access data stored in a database. ADO stands for ActiveX Data Objects.

ADO represents a collection of built-in controls which can be directly used for accessing database from Visual Basic applications.

ADO.NET particularly is a revised version of original ADO protocol. ADO can be used with earlier versions of VB, but not with VB.NET.

With the help of ADO.NET, it is possible to make connection to a database and fire SQL queries
- ✓ to retrieve records from table(s).
- ✓ to view records
- ✓ to insert new records
- ✓ to edit existing records
- ✓ to delete records
- ✓ and to perform many database-related operations.

ADO.NET provides support for many popular (R) DBMS like MS SQL Server, Oracle, MS Access.

ADO.NET Data Namespaces

- The core ADO.NET classes exist in the System.Data namespace. This namespace, in turn, contains some child namespaces.
- The most important of these are System.Data.SqlClient and System.Data.OleDb, which provide classes for accessing SQL Server databases and OLE (Object Linking and Embedding) DB–compliant databases, respectively.
- The System.Data.OleDb namespace contains classes like OleDbConnection andOleDbDataAdapter.
- The System.Data.SqlClient contains classes like SqlConnection and SqlDataAdapter.
- Another child namespace also exists in the System.Data namespace: System.Data.Odbc. The System.Data.Odbc namespace provides access to older Open Database Connectivity (ODBC) datasources that do not support the OleDb technology.
- The System.Data.SqlClient, System.Data.OleDb, and System.Data.Odbc namespaces are known as data providers in ADO.NET.

Overview of ADO.NET Objects

Here's a list of the most common ADO.NET objects:

1. Data Connection
   - To start working with a database, you must have a data connection.
   - A data connection represents a session or a link between the VB.NET program and the database.
   - The architecture in which the connection to the database is always kept open is known as “Connected” architecture.
   - The architecture in which the connection to the database is not always kept open, but is opened when needed and closed immediately after the use, is known as “Disconnected” architecture.
   - ADO.NET supports both of these types of architecture.

- In ADO.NET. an OleDbConnection is used to connect to MS Access and similar databases whereas SqlConnection is used to connect to MS SQL Server database.

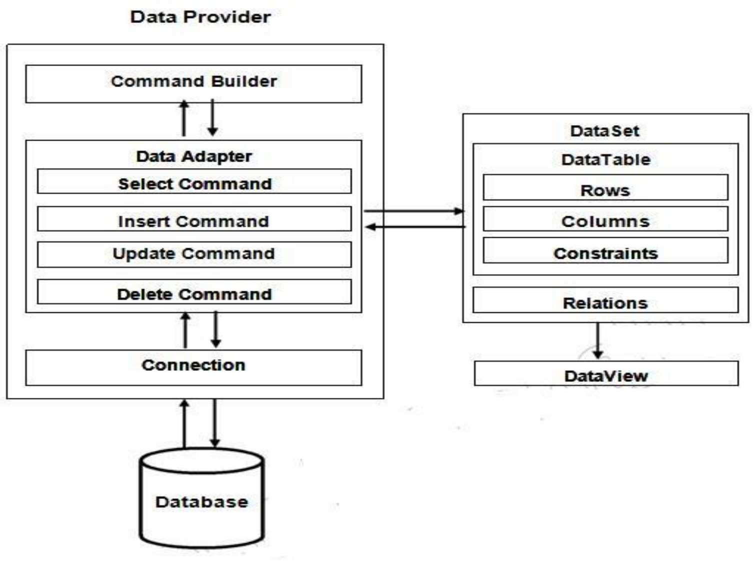
2. Data Adapter
   - Data adapter is a link between the database and the dataset. A dataset is a part of the memory in which the retrieved records are stored.
   - Data adapter is configured with SQL to execute against the data source.
   - It is normally used to execute a select SQL statement to retrieve records and fill the dataset with these records.
   - OleDbDataAdapter is used in case of MS Access database whereas SqlDataAdapter is used with MS SQL Server database.
3. Command
   - Data adapters can read, add, update, and delete records in a data source.
   - To allow you to specify how each of these operations work, a data adapter contains command objects for each of them.
   - Dataadapters support four properties that give you access to these command objects:SelectCommand, InsertCommand, UpdateCommand, and DeleteCommand.
   - Command objects are required to execute DML statements like insert/ update/ delete etc.
4. Dataset
   - Dataset stores data in a disconnected cache.
   - The structure of a dataset is similar to that of a relational database; it gives you access to an object model of tables, rows, and columns, and it contains constraints and relationships defined for the dataset.
   - Datasets are supported with DataSet objects.
5. DataTable
   - DataTable objects hold a data table from a data source.
   - Data tables contain two important properties: Columns, which is a collection of the DataColumn objects that represent the columns of data in a table, and Rows, which is a collection of DataRow objects, representing the rows of data in a table.
6. Data reader
   - DataReader object hold a **read-only, forward-only** (i.e., you can only move to the next record from the current record, not backwards) set of data from a database.
   - Using a data reader can increase speed because only one row of data is in memory at a time.
7. Data view
   - Data views represent a customized view of a single table that can be filtered, searched, orsorted.
   - In other words, a data view, supported by the DataView class, is a data "snapshot" that takes upfew resources.
8. Constraints
   - Datasets support constraints to check data integrity.
   - A constraint, supported by the Constraint class, is a rule that can be used when rows are inserted, updated, or deleted to check the affected table after the operation.
   - There are two types of constraints: unique constraints check that the new values in a column are unique throughout the table, and foreign-key constraints specify how related records should be updated when a record in another table is updated.
9. DataRelation
   - DataRelation objects specify a relationship between parent and child tables based on a key that both tables share.
10. DataRow
    - DataRowobjects correspond to a particular row in a data table.
    - You use the Item property to get or set a value in a particular field in the row.
11. DataColumn
    - DataColumn objects correspond to the columns in a table.
    - Each object has a DataType property that specifies the kind of data each column contains, such as integers or string values.

Difference between DataSet and DataReader

- DataSet allows you to view, insert, update and delete records whereas DataReader allows to only view records, it does not support insert, update or delete operation.
- DataSet allows to go forward as well as backward while navigating through records. DataReader allows to go only forward, not backward.
- DataReader is faster than DataSet.
- DataReader is connected architecture since it keeps the connection open until all rows are fetched one by one. DataSet is disconnected architecture since all the records are brought at once and there is no need to keep the connection alive.

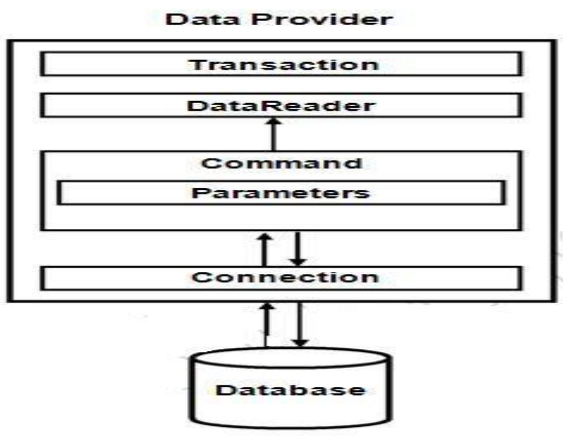
The Disconnected architecture of ADO.NET

- The architecture in which the connection to the database is not always kept open, but is opened when needed and closed immediately after the use, is known as “Disconnected” architecture.
- In disconnected architecture, the connection is opened, an SQL statement is executed, the records are fetched and these records are stored locally. Then, the connection is closed.
- Since the records are stored locally, they can be accessed even after the connection is closed. However, any change made to such records will not be stored in the database.
- Disconnected architecture of ADO.net uses Connection, Data adapter, Command builder and dataset.



The Connected architecture of ADO.NET

- The architecture in which the connection to the database is always kept open is known as “Connected” architecture.
- In disconnected architecture, the connection is kept opened as long as the interaction with the database is not over.
- Since the connection has to be kept open all the time, it uses considerable amount of resources on client-side and network.
- Connected architecture of ADO.net uses Connection, Command and DataReader.



Steps to bind the application with the database in ADO.NET

Step-1 Import required namespace.

- A namespace is a named collection of classes, interfaces, modules and constants. A namespace is imported with the help of Imports statement.
- Syntax: Imports *namespace*
- To access MS SQL Server database, System.Data.SqlClient namespace is imported as follows:

Imports System.Data.SqlClient

Step-2 Create a connection object.

- A connection object must be declared as follows: **Dim cn As SqlConnection**
- Later, it must be created as: **cn = New SqlConnection("connection string")**
- connection string contains information such as type and name of data source, authentication details etc. It can be directly copied from Connection String property of Data Connection object.

Step-3 Open the connection to the database.

- The connection is made by calling the Open( ) method on connection object.

cn.Open()

Step-4 Create a data-adapter object.

- A data-adapter object must be declared as follows: **Dim adp As SqlDataAdapter**
- Later, it must be created as: **adp = New SqlDataAdapter("Select query", cn)**

Step-5 Create a dataset object.

- A dataset object must be declared as follows: **Dim ds As DataSet**
- Later, it must be created as: **ds = New DataSet()**

Step-6 Fill the dataset with records retrieved.

- Records are filled in the dataset object by calling Fill() method on Data adapter object.

adp.Fill(ds)

Connection String

- Connection String is a property associated with a Data Connection.
- It is essential to specify a correct connection string while creating a Connection object.
- Connection String includes following:
  1) Data Source: The computer on which the database is located.
  2) Initial Catalog: Name of the database to access.
  3) User ID: Valid user id to access the database.
  4) Password: Valid password associated with the user id.
  5) Integrated Security: When this property is set to True, Windows user id and password are used instead of database-specific user id and password.
  6) Pooling: Whether connection pooling should be used.

Retrieving data from DataSet

- A DataSet is filled with records by calling Fill() method on SqlDataAdapter object.

```
adp.Fill(ds)
```

Here, adp is an SqlDataAdpater object and ds is a DataSet object.

To access data stored in column no. c of row no. r from the DataSet, the following syntax is used:

ds.Tables(0).Rows(r).Item(c)

Remember, row no. and column no. starts with 0.

You can also use column-name instead of column no. It means

ds.Tables(0).Rows(r).Item("Firstname")

Accessing Data with the Server Explorer

- In Visual Basic, the Server Explorer lets you work with connections to various data sources.
- To display the Server Explorer, if it's not already visible, use the View | Server Explorer menu item, or press Ctrl+Alt+S.
- This tool lets you create and examine data connections, including connections to Web servers.
- You can also see existing connections to various databases in the Server Explorer.

Common SQL statements

(1) SELECT Statement: You can use the SELECT statement to get records from table; The following statement retrieves all the records in the customer table, using wildcard character *:

```
SELECT * FROM Customers
```

You can also use the SELECT statement to select specific fields from a table.

```
SELECT CustomerID, Address, City FROM Customers
```

Remember, when you are reading the data from the database table you need to use SqlDataReader and its Read() method to read the data. And with command you need to use ExecuteReader() method.

(2) DELETE Statement: You can use the DELETE statement to delete records from the database.

```
DELETE * FROM Customers
```

This will delete all records from the table Customers. When you are using DELETE statement, this means you are modifying the actual table of database. So, you have to use ExecuteNonQuery() method with SqlCommand object. This method will return number of rows affected in table.

(3) UPDATE Statement: You can use UPDATE statement to update records in the database.

```
UPDATE Customers SET CustomerID="….", Address="…." WHERE City="…."
```

This will update records in the table Customers. When you are using UPDATE statement, this means you are modifying the actual table of database. So, you have to use ExecuteNonQuery() method with SqlCommand object. This method will return number of rows affected in table.

(4) INSERT Statement: you can use INSERT statement to add new records to the database.

```
INSERT INTO Customers VALUES (……………………….)
```

This will add new records to the table Customers. When you are using INSERT statement, this means you are modifying the actual table of database. So, you have to use ExecuteNonQuery() method with SqlCommand object. This method will return number of rows affected in table.

Using SqlCommand objects

- An SqlCommand object represents a valid SQL statement. It can be a select, insert, update or delete statement.
- It is declared as: **Dim cmd As SqlCommand**
- Later, it can be initialized as:

  cmd = New SqlCommand("SQL command" , cn)
- An SqlCommand object can also be created as: **cmd = New SqlCommand()** In this case, corresponding SQL command and connection must be specified by assigning value to CommandText and Connection property as follows:

  cmd.CommandText = "select * from EMPLOYEE"
  cmd.Connection = cn

Public methods of SqlCommand

Method	**Use**

ExecuteReader()	This method is used to execute a SELECT query. It returns a DataReader object. So, the rows retrieved can be directly stored in a DataReader object.
	
	Dim cmd As SqlCommand
	Dim dr As SqlDataRedaer
	
	cmd = New SqlCommand("SELECT * FROM EMPLOYEE" , cn)
	dr = cmd.ExecuteReader()
	
	dr.read()
	
	TextBox1.Text = dr.Item(0).ToString
	TextBox2.Text = dr.Item(1).ToString
	TextBox3.Text = dr.Item(2).ToString
	TextBox4.Text = dr.Item(3).ToString

ExecuteNonQuery()	This method is used to execute a Transact-SQL i.e. INSERT/DELETE/UPDATE query. It returns an integer – no. of rows affected.
	
	cmd = New SqlCommand("DELETE FROM EMPLOYEE WHERE ID >1009", cn)
	
	Dim n As Integer
	n = cmd.ExecuteNonQuery()

ExecuteScalar()	This method is used to execute a SELECT statement which returns only one row with one column. This mostly involves use of built-in aggregate functions like SUM(), AVG() etc.
	
	cmd = New SqlCommand("SELECT SUM(SALARY) FROM EMPLOYEE ", cn)
	
	Dim Total As Decimal
	
	Total = cmd.ExecuteScalar()

CreateParameter()	Creates a new parameter
Cancel()	Cancels a command's execution
Prepare()	Creates a compiled version of the command.

DataRow

- **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.
- To create a new **DataRow** object, you usually use the **NewRow()** method of a **DataTable** object, and after configuring the row with data, you can use the Add() method to add the new **DataRow** to the table.
- In addition you can call the AcceptChanges() method of the **DataTable** object to make that table treat the new row as it would its original data.

DataRowState	**Description**
Detached	The row has been created but is not part of any DataRowCollection. A DataRow is in this state immediately after it has been created and before it is added to a collection, or if it has been removed from a collection.
Unchanged	The row has not changed since AcceptChanges was last called.
Added	The row has been added to a DataRowCollection, and AcceptChanges has not been called.
Deleted	The row was deleted using the Delete method of the DataRow.
Modified	The row has been modified and AcceptChanges has not been called.

DataGridView Control

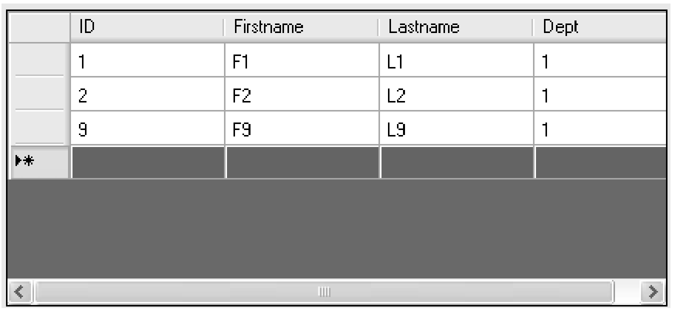
- The DataGridView control is a container that allows you to bind data from your data source and display it in a spreadsheet-like format, displaying the columns of data horizontally and the rows of data vertically.
- The DataGridView also provides many properties that allow you to customize the appearance of the component itself, as well as properties that allow you to customize the column headers and the display of data.
- It is available under Data category in Toolbox.
- The DataGridView displays all the records together, and not one-by-one. Further, the entire record can be seen as a single row in the DataGridView control.
- There is no need to have separate navigation mechanism (i.e First, Previous, Next, Last button) as all the records are displayed within the DataGridView.
- It also support insert, update and delete operations.
- It also allows sorting of records on a particular field.
- It also allows reordering the fields in the view for better understanding.

Important properties of DataGridView

No.	Property	Use
1	DataSource	Source of Data (DataTable) which should be used to fill data in DataGridView. **DataGridView1.DataSource = ds.Tables(0)**
2	AllowUserToAddRows	Whether the option to add row should be displayed to the user.Default value: True
3	AllowUserToDeleteRows	Whether the user is allowed to delete row from the DataGridView. Default value: True
4	AllowUserToOrderColumns	Whether the user is allowed to change the order of columns in display. Default value: False
5	AllowUserToResizeColumns	Whether the user is allowed to change the width of columns in display. Default value: True

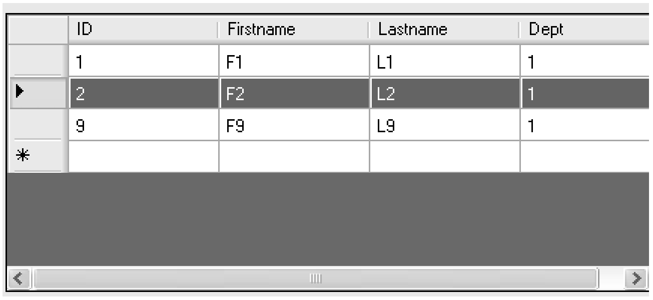
6	AlternatingRowsDefaultCellStyle	Allows to customize display of odd-numbered rows.
7	Dock	Which borders of the control should be bound to thecontainer. It determines position and size of the DataGridView within the form. Possible value: None/ Top/ Bottom/ Center/ Left/ Right
8	EditMode	How cell editing is started.
9	MultiSelect	Whether the user is allowed to select multiple cells, rows or columns. Default value: True
10	ScrollBars	The type of scroll bar to display (None/ Horizontal/ Vertical/ Both) Default value: Both
11	SelectionMode	How the cells can be selected.
12	ReadOnly	Insert, Update and delete are not allowed if this property is True.

To insert a new record, click on the * (asterisk) in the row header and enter a new record.



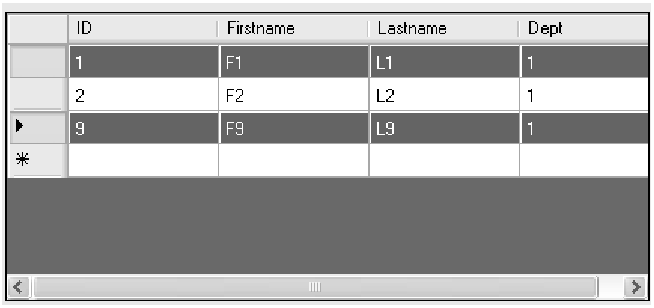
	ID	Firstname	Lastname	Dept
	1	F1	L1	1
	2	F2	L2	1
	9	F9	L9	1
▶*				

To delete a single record, click on the row header before the record and press Delete key on the keyboard.



	ID	Firstname	Lastname	Dept
	1	F1	L1	1
▶	2	F2	L2	1
	9	F9	L9	1
*				

To delete multiple records, control/shift + click on the row headers beforerecords and press Delete key on the keyboard.



	ID	Firstname	Lastname	Dept
	1	F1	L1	1
	2	F2	L2	1
▶	9	F9	L9	1
*				

To update a record, click on the row header before the record and replace old value(s) by new value(s).

To save the changes to the database, the Update() method must be called on Data Adapter object.