UNIT III

CHAPTER 3

ADO .Net

Syllabus Topics

ADO.NET : Data Provider Model, Direct Data Access - Creating a Connection, Select Command, DataReader, Disconnected Data Access.

Data Binding : Introduction, Single-Value Data Binding, Repeated-Value Data Binding, Data Source Controls - SqlDataSource.

Data Controls : GridView, DetailsView, FormView, Working with XML: XML Classes - XMLTextWriter, XMLTextReader.

Caching : When to Use Caching, Output Caching, Data Caching.

LINQ : Understanding LINQ, LINQ Basics.

ASP.NET AJAX : ScriptManager, Partial Refreshes, Progress Notification, Timed Refreshes.

Syllabus Topic : ADO .Net

3.1 Introduction to ADO.NET

3.1.1 Introduction

- Whenever we create a software or website, it is not possible to implement it using only programming languages. Regarding any software or website, there is always important data which we have to store permanently.
- Storing data permanently is not the facility given by any programming language. For this purpose we have to use database. That means software or websites is developed by the combination of programming language and database.
- ADO.Net is a model used by the .Net framework to communicate with database for retrieving and storing data with the help of various built in classes.
- As you know that there are several different types of databases available. Some of them are listed below :
  1. Microsoft SQL Server
  2. Microsoft Access
  3. Oracle
  4. Borland Interbase
  5. IBM DB2, etc

  In this chapter will use SQL Server.

Syllabus Topic : Data Provider Model

3.1.2 Data Provider Model

Q. Explain data provider model in brief. **(4 Marks)**

- The data provider is used for connecting to database, executing queries, and retrieving results.
- Those results are processed and stored in a DataSet in order to be avail to the user as needed.
- The data providers are lightweight components which create a minimal layer between the data source and code and increases performance without sacrificing the functionality.

- Following Data Providers are used in ADO.NET.
  (i) The Microsoft SQL Server
  (ii) OLEDB
- In ADO .Net all the functionalities are implemented with the help of set of core classes. These classes are divided into two groups :
  (i) Classes used to contain and manage data : For example DataSet, DataTable, DataRow, and DataRelation.
  (ii) Classes used to connect to a particular data source : For example Connection, Command, and DataReader.
- The data container classes are in general totally generic. It makes no difference that from where we are extracting the data, it is stored in the similar data container: the specialized DataSet class.
- Dataset is playing similar role just like an array or collection i.e. container or package for the data. But the most important difference is that, the DataSet is specially build for relational data that means it understands concepts such as rows, fields, and table relationships.
- The second group of classes is usually comes in various flavors. All the groups of data interaction classes are known as ADO.NET data providers. Data providers are customized in such a way that all of them use best-performing way for the interaction with their data sources.
- For example, the SQL Server data provider is specially customized to work with SQL Server. Internally, tabular data stream (TDS) protocol of SQL server is used by it for communication purpose. Hence there is guarantee of best performance. For database Oracle the Oracle data provider can be used.
- There is personal prefix of all these data providers for naming the classes. Thus in the SQL Server provider there are classes like SqlConnection and SqlCommand while in the Oracle provider classes named OracleConnection and OracleCommand are present.
- The internal behaviour of these classes is quite different as they have to connect to different databases with the help of different low-level protocols. On the other hand the external display of these classes is very much similar and also provides similar set of basic methods because same common interfaces are implemented by them.
- Because of all these features, the application is protected from the complexity of different standards and SQL Server provider and Oracle provider in the same way. In fact, it is also possible to translate a block of code written for interaction with a SQL Server database into a block of Oracle-specific code just by changing the name of class in the code.
- All the classes regarding database connectivity are categorized under three namespaces

Sr. No.	Namespace	Purpose
1.	System.Data.SqlClient	Provides classes used to connect Microsoft SQL Server database and execute commands like SqlConnection and SqlCommand.
2.	System.Data.SqlTypes	Provides structures for SQL Server–specific data types like SqlMoney and SqlDateTime. These types can be used along with SQL Server data types without need of conversion into standard .NET equivalents like System.Decimal or System.DateTime.
3.	System.Data	Provides fundamental classes with the core ADO.NET functionality. These classes are DataSet and DataRelation which can manipulate structured relational data. These classes are not dependent upon any specific type of database or the connection method.

- Now we are going to see different object of ADO .Net. They are explained below.

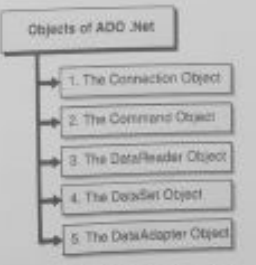


Fig. C3.1 : Objects of ADO .Net

Q. What are the objects of ADO.net? Explain any one. (4 Marks)

1. The Connection Object

- To communicate or interact with a database, you must have a connection with a database.
- The connection helps out to recognize the database server, the database name, user name, password, and other parameters that are required for connecting to the database.
- A connection object is used by command to know on which database to execute the command.

2. The Command Object

- Interacting with a database does not mean only creating as connection; it means you must state the actions that you want to occur. This can be done with the help of command object. The command object can be used to send SQL statements to the database.
- A command object makes use of connection object to tell which database to communicate with. The command object can be used alone, to execute a command directly.

3. The DataReader Object

- In ADO.NET the DataReader Object is a stream-based, read-only, forward-only retrieval of query results from the Data Source, which never update the data.
- A connection oriented data access to the data sources is provided by DataReader. A Connection Object can contain only one DataReader at a time.
- While data is being accessed, the connection in the DataReader remains open and cannot be used for any other purpose. While starting reading through a DataReader it should always be open and positioned price to the first record. The DataReader provides read() method to read the records from it and it always moves forward to a next record if any row exist.

4. The DataSet Object

- The DataSet is used to store the data of database in application. It represents a collection of data retrieved from the Data Source.
- DataSet is tabular representation of data. Tabular representation means it represents data into row and column format.
- We can use Dataset in combination with DataAdapter class. The DataSet object work in disconnected database architecture. It provides a better advantage over DataReader , because the DataReader is working only with the connection oriented database architecture.
- The Dataset contains the copy of the data requested by client. The Dataset can contain more than one tables at a time. We can set up relations between these tables within the DataSet.
- The **DataSet** may contain multiple tables represented by DataTable objects.

5. The DataAdapter Object

- DataAdapter provides the communication between the Dataset and the Datasource.
- DataAdapter can be used in combination with the DataSet Object so that the two objects enable both data retrieval and data manipulation capabilities.
- To retrieve data from a data source the DataAdapter is used and it is also used to manipulates tables which are present in a DataSet.
- The DataAdapter is also used to reflect changes made to the DataSet back to the data source.
- Connection object is used by DataAdapter for connecting to a data source, and Command object is used by DataAdapter to retrieve data and reflect changes to the data source.
- The SelectCommand property of the DataAdapter is used to retrieve data from the data source. The InsertCommand, UpdateCommand, and DeleteCommand properties of the DataAdapter are used to manage updates to the data in the data source according to modifications made to the data in the DataSet.
- The Fill() method of the DataAdapter is used to populate a DataSet with the results of the SelectCommand of the DataAdapter. The arguments of Fill() method are - DataSet to be populated, and a DataTable object, or the name of the DataTable to be filled with the rows returned from the SelectCommand.
- DataReader object is used by Fill() method implicitly for returning the column names and types that are used to create the tables in the DataSet, and the data to manipulate the rows of the tables in the DataSet.
- If Tables and columns are not exist previously then they are created; otherwise existing DataSet schema is used by Fill() method.

- Column types are created as .NET Framework types according to the tables in Data Type Mappings in ADO.NET.
- Primary keys are created if they exist in the data source. If Fill() finds that a primary key exists for a table, it will overwrite data in the DataSet with data from the data source for rows.
- If primary key is not found then the data is appended to the tables in the DataSet. Fill() method uses any mappings that may exist when you manipulate the DataSet.

Syllabus Topic : Direct Data Access

3.1.3 Direct Data Access

Q. Explain Direct Data Access in detail. (10 Marks)

- The Direct Data Access is considered as the most straightforward way to communicate with a database.
- This option gives complete control of building and executing SQL commands to user. All the database operations like insert, update, and delete information can be performed easily.
- While using Direct Data Access, there is no need to store copy of the information in memory. Rather for a brief period of time user work with it and close the connection after finishing the work.
- This option is completely very thus another option of disconnected data access, in which a copy of data is kept in the Dataset object. In disconnected data access, user can work even after closing the connection.
- The Direct Data Access models is basically well suited for ASP.NET web pages in which there is no requirement of keeping a copy of the data in memory for long periods of time.
- We have to keep it in mind that the page is requested and quickly shut down when the response is returned to the user. It indicates that the lifetime of page is only of literally few seconds.
- To access the data with simple data access, we have to follow the given steps :
  1. First Create Connection, Command, and DataReader objects.
  2. Retrieve information from the database with the help of DataReader, and show on a web form.
  3. Disconnect connection by closing it.
  4. Send webpage to user, at this point the database is no longer connected and all the ADO.NET objects are destroyed.
- To add or update information, follow these steps
  1. Create new Connection and Command objects.
  2. Execute the Command by using SQL statement.

Fig. 3.1.1 illustrates how the ADO .NET objects communicate to make direct data access.

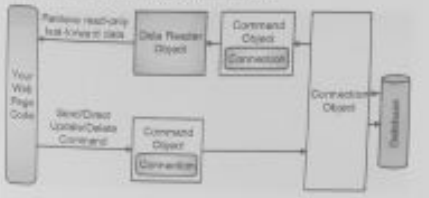


Fig. 3.1.1 : Direct Access with ADO.Net

- For communication with database using ADO.Net, we have to include the important ADO .Net namespaces.
- Here we consider SQL Server provider then we have to import following namespaces :
  - ii) Imports System.Data
  - iii) Imports System.Data.SqlClient

Syllabus Topic : Creating a Connection

3.1.3.1 Creating Connection

Q. Explain how to create a connection in ADO.NET? (6 Marks)

- While interacting with database the first thing is to create a connection. The connection informs the rest of the ADO.NET code which database it is going to be communicated. It handles all of the low level logics related to the particular database protocols.
- Working with connection in ADO.Net is very easy; you just have to understand the connection. When there is single user then you don't have take care of connection but when there are multiple users for a single database you have to take care of it because wrong information makes lot of errors while accessing an application.

Creating a SqlConnection Object

- Like any other C# object SqlConnection is one of the object .You can declare and instantiate the SqlConnection at the same time, as shown below

```
SqlConnection connect = new SqlConnection(
    "Data Source=(local);Initial
Catalog=Northwind;Integrated Security=SSPI");
```

- The above instantiated SqlConnection object uses a constructor which has only one parameter of string data type and this parameter is known as connection string. The following Table 3.1.1 illustrates common parts of a connection string.

Sr. No.	Connection String Parameter Name	Description
1.	Data Source	It is used to Identify the server. It could be local machine, machine domain name, or IP Address.
2.	Initial Catalog	It contains the database name.
3.	Integrated Security	This Parameter is used to set to SSPI for making connection with user's Windows login.
4.	User ID	Name of user configured in SQL Server.
5.	Password	Password matching to SQL Server User ID.

- When you are developing your project using single computer/machine then Integrated Security is secure. You can also provide security based on a SQL Server User ID and the password.
- The following code shows a connection string, using the User ID and Password parameters.

```
SqlConnection connect = new SqlConnection(
        "Data Source=DatabaseServer;Initial
Catalog=Northwind;
                User
ID=YourUserID;Password=YourPassword");
```

- In the above code snippet observe that how the Data Source is set to the DatabaseServer to specify that you can identify a database placed on a different machines, over LAN, or over the Internet. Additionally, User ID and Password replaces the Integrated Security parameter.

Using a SqlConnection

- The aim of creating a SqlConnection object is to allow users to use other ADO.NET code to work with a database. Other ADO.NET objects, such as a SqlCommand and a SqlDataAdapter take a connection object as a parameter.
- The following operations occur during the lifetime of a SqlConnection :

1. Instantiate the SqlConnection.
2. Open the connection.
3. Pass the connection as a parameter to other ADO.NET objects.
4. Perform database operations with the other ADO.NET objects.
5. Close the connection.

- To instantiate the SqlConnection object, the subsequent steps are shown in following code snippet:

Using a SqlConnection

```
using System;
using System.Data;
using System.Data.SqlClient;

class SqlConnectionDemo
{
  static void Main()
  {
    SqlConnection connect = new SqlConnection(
      "Data Source=(local);Initial
Catalog=Database1;Integrated Security=SSPI");

    SqlDataReader datareader = null;

    try
    {

      connect.Open();

SqlCommand command = new SqlCommand("select * from
employee", connect);

      datareader = command.ExecuteReader();

        while (datareader.Read())
        {
          Console.WriteLine(datareader[0]);
        }
    }
    finally
    {
```

Connectivity to database


```
    if datareader != null)
      {
        datareader.Close();
      }

    if (connect != null)
        {
          connect.Close();
        }
    }
}
```

- As shown in the above code snippet, you can open a connection by calling the Open() method of the SqlConnection instance, connect. If you try to perform any operations on a connection that was not opened, the it will produce an exception. So, you have to open the connection before performing any operation on it.
- Before going to use the SqlCommand object, you have to inform the ADO.NET code that which connection it desires. In above code snippet, we have set the second parameter to the SqlCommand object with the SqlConnection object, connect. After passing the SqlConnection object to SqlCommand, the SqlCommand will use that connection for performing the operations.
- The code that uses the connection is a SqlCommand object, which executes a query on the employee table. The result set is returned as a SqlDataReader and the while loop reads the first column from each row of the result set, which is the EmployeeID column.
- When you have done all the operations then you must close the connection. If you do not close the connection then it will affect the performance and scalability of your application. In above code snippet the Close ( ) method is called in a finally block to close the connection.
- Finally block will help you to ensure that a certain part of code will be executed, regardless of whether or not an exception is generated.

Syllabus Topic : Select Command

3.1.3.2 Select Command

- A SqlCommand object permits you to state what type of communication/interaction you want to perform with a database. For example, you can do select, insert, modify, and delete commands on database table.

```
SqlCommand cmd = new SqlCommand("select ename
from employee", connect);
```

- The above line indicates the instantiation of a SqlCommand object. The SqlCommand takes a string parameter which contains the command you desire to execute and a reference to a SqlConnection object.

Querying Data

- To retrieve a data set for viewing to users you make use of SQL select command. The ExecuteReader() method, which returns a SqlDataReader object, which can be used with SqlCommand object to use SQL select command.
- The following example shows how to use the SqlCommand object to acquire a SqlDataReader object

Example

```
SqlCommand command = new SqlCommand("select ename
from employee", connect);
```

instantiate a new command with query and connection

```
SqlDataReader datareader = command.ExecuteReader();
```

Call Execute reader to get query results

Inserting Data

The ExecuteNonQuery () method is used to add data into a database. This method belongs to SqlCommand object. The below code snipper illustrates how to insert data into a database table

Example

```
string instring = @" insert into employee (ename, salary)
values ('Kunal', 30000)";
```

Preparing command string for performing operation

```
SqlCommand command = new SqlCommand(instring,
connect);
```

To send command call ExecuteNonQuery

```
command.ExecuteNonQuery();
```

- In the above code notice that inside the insert command, we explicitly specified the columns ename and salary. The employee table has a primary key field named emp_ID. We are skipping this column from the query because SQL Server will add this field itself.

Trying to add a value to a primary key field, such as emp_ID, will produce an exception.

Updating Data

- To update data in database the ExecuteNonQuery method is used. The below code illustrates how to update data

Example

```
string updated_data = @" update employee set ename =
'Kunal B.' where salary = 30000";
SqlCommand command = new
SqlCommand(updated_data);
command.Connection = connect;
command.ExecuteNonQuery();
```

- Observe that we have used a variable to store the SQL command. Now we have used a different SqlCommand constructor that accepts only the command.

Deleting Data

- The ExecuteNonQuery method is also used to delete data from database. The below example illustrates how to delete a record from a database

Example

```
string delete_data = @" delete from employee where
ename = Kunal B.";
SqlCommand command = new SqlCommand();
command.CommandText = delete_data;
command.Connection = connect();
command.ExecuteNonQuery();
```

Getting Single values

- Sometimes you require just a single value from a database. It could be a count, sum, average, or other aggregated value from a data set. Performing an ExecuteReader and computing the result in your code is not the well-organized way to do this. The best option is to let the database perform the work and return just the single value that you need.
- The below example illustrates how to do this with the ExecuteScalar method

Example

```
SqlCommand command = new SqlCommand("select
count(*) from employee", connect);
int total = (int)command. ExecuteScalar ();
```

- The query given in the SqlCommand constructor gets the count of all records from the employee table. This query will return only one value. The ExecuteScalar method is used to return this single value. Since the return type of ExecuteScalar is type object so we have to convert it. To convert value to int we use a cast operator.

Putting it All Together

- To know each operation in detail we are using the small code snippets. Now we can put it all together to make a single view of all operations. This can be demonstrated as follow

```
using System;
using System.Data;
using System.Data.SqlClient;
class SqlCommandDemo1
{
SqlConnection connect;

  public SqlCommandDemo1()
{
      connect = new SqlConnection(
"Data Source=(local);Initial Catalog=database1;Integrated
Security=SSPI");
  }
   static void Main()
   {
     SqlCommandDemo1 cmddemo = new
SqlCommandDemo1();

     Console.WriteLine("Employee table Before Insertion");

     cmddemo.ReadData();

     cmddemo.Insertdata();
     Console.WriteLine("Employee table After Insertion");

     cmddemo.ReadData();
      cmddemo.UpdateData();
      Console.WriteLine("Employee table After Updation");
      cmddemo.ReadData();
      cmddemo.DeleteData();
```



```
     Console.WriteLine("Categories After Delete");

     cmddemo.ReadData();

     int numrec = cmddemo.GetNumberOfRecords();

        Console.WriteLine("Number of Records: {0}",
numrec);
   }

   public void ReadData()
   {
     SqlDataReader datareader = null;

     try
     {
       connect.Open();

       SqlCommand command = new SqlCommand("select
ename from employee", connect);

       datareader = command.ExecuteReader();
       while (datareader.Read())
       {
         Console.WriteLine(datareader[0]);
       }
     }
     finally
     {

       if (datareader != null)
       {
         datareader.Close();
       }

       if (connect != null)
       {
         connect.Close();
       }
     }
   }
   public void Insertdata()
   {
     try
     {
       conn.Open();

       string instring = @"insert into employee (ename,
salary) values ('Sadashiv patil', 30000)";

       SqlCommand command = new
SqlCommand(instring, connect);

       command.ExecuteNonQuery();
     }
     finally
     {
       if (connect != null)
       {
         connect.Close();
       }
     }
   }
   public void UpdateData()
   {
     try
     {

       connect.Open();

       string updated_data = @" update employee set
ename = Rakesh Jadhav'where salary = 30000";

       SqlCommand command = new
SqlCommand(updated_data);
       command.Connection = connect;
       command.ExecuteNonQuery();
     }
     finally
     {
```

```
    if (connect != null)
    {
      connect.Close();
    }
  }
}

  public void DeleteData()
{
  try
  {
    connect.Open();

    string deleted_data = @"
      delete from employee where ename = 'Rakesh'";

    SqlCommand command = new SqlCommand();

    command.CommandText = deleted_data;

    command.Connection = connect;

      command.ExecuteNonQuery();
  }
  finally
  {
    if (connect != null)
    {
      connect.Close();
    }
  }
}

  public int GetNumberOfRecords()
{
  int count = -1;

  try
  {
```

```
    connect.Open();

    SqlCommand command = new SqlCommand("select
count(*) from employee", connect);

    count = (int) command.ExecuteScalar();
  }
  finally
  {
    if (connect != null)
    {
      connect.Close();
    }
  }
  return count;
}
}
```

Syllabus Topic : Data Reader

3.1.3.3 SqlDataReader

Q. Write note on SQLDAta Reader (4 Marks)

- To read a data in an efficient way a SqlDataReader is useful. SqlDataReader is used for only reading the data you cannot use it for writing a data.
- SqlDataReader can read data in sequential manner i.e. when you read some data and go ahead then you can't come back to again read that data.
- The forward only design of the SqlDataReader is what facilitates it to be quick. It doesn't have overhead related to traversing the data or writing it back to the data source. The SqlDataReader is best option if you want to read collection of data in faster way for only one time.

Creating a SqlDataReader Object

- Creating an object of a SqlDataReader is a quit different than the manner in which you instantiate other objects of ADO.Net. You have to call the ExecuteReader method on a command object. It is shown as follow

```
SqlDataReader datareader = command.ExecuteReader();
```

- The ExecuteReader method of the SqlCommand object, command, returns a SqlDataReader instance

Reading Data

- As discussed earlier the SqlDataReader returns data through an in order stream. That means when you read a single row then you cannot read the previous row. To read that row, you would have to create a new object of the SqlDataReader and read via the data stream.
- One way of reading from the data stream is to use a while loop as given below

```
while (datareader.Read())
{
    string name = (string) datareader["ename"];
    int sal = (int) datareader["salary"];
    Console.Write("{0,-25}", name);
    Console.Write("{0,-20}", sal);
    Console.WriteLine();
}
```

Print out the results

- Observe the call to *Read* on the SqlDataReader, *datareader*, in the while loop condition in the above code. The return value of *Read* is of type Boolean. IT returns true if there are more records to read and returns false when there is no more records available.
- In the above code snippet you extracted two columns from the employee table having name ename, salary. After reading data you can perform any operation on it like printing them to console or modifying them etc.

Finishing Up

- One important thing is that always remember to close your SqlDataReader, in the same way as you close the SqlConnection. Place your data access code in a try block and close operation in the finally block. This can be illustrated as follow

```
try
{
  ...
}
finally
{
    if (datareader!= null)
    {
        datareader.Close();
    }
}
```

- The above code snippet makes sure that the SqlDataReader is not null. After the code knows that a good instance of the SqlDataReader exists, it can close it. Following program combines all the code snippets given above.

Using the SqlDataReader

```
using System;
using System.Data;
using System.Data.SqlClient;

namespace datareed
{
    class ReaderDemo1
    {
        static void Main()
        {
            ReaderDemo1 rd1 = new ReaderDemo1();
            Rd1.SimpledataRead();
        }
        public void SimpledataRead()
        {
            SqlDataReader datareader = null;
                SqlConnection connect = new
SqlConnection(
"Data Source=(local);Initial Catalog=Northwind;Integrated
Security=SSPI");

            SqlCommand command1 = new
SqlCommand(
                "select * from employee", connect);

            try
            {

                connect.Open();

                datareader = command1.ExecuteReader();

                Console.WriteLine("Ename
Salary");

                while (datareader.Read())
                {
```

```
            string name = (string)datareader["ename"];
            int sal = (int) datareader["salary"];
                Console.Write("{0,-25}", name);
                Console.Write("{0,-20}", sal);
                Console.WriteLine();
            }
        }
        finally
        {
            if (datareader != null)
            {
                datareader.Close();
            }
            if (connect != null)
            {
                connect.Close();
            }
        }
    }
}
```

Syllabus Topic : Disconnected Data Access

3.1.3.4 Disconnected Data Access

Q. Explain disconnected data access. (4 Marks)

- In previous section we have seen fully connected mode of operation for interacting with a data source by using the SqlCommand object. This section shows how to do disconnected data access using the DataSet and SqlDataAdapter objects.
- A DataSet is considered as an in-memory data store which can store numerous tables. Th DataSets are used to hold the data.
- They do not interact with data source. The SqlDataAdapter is used to handle the connection with the data source and manage the disconnected behavior.
- The most important thing is that the connection is opened by the SqlDataAdapter when needed and closed as soon as the operation is accomplished.
- For Example, to fill data in DataSet, following steps are performed by SqlDataAdapter
  1. Open connection
  2. Retrieve data into DataSet
  3. Close connection
- To update data source with DataSet changes following steps are performed.
  1. Open connection
  2. Write changes from DataSet to data source
  3. Close connection
- After filling data set the data source connection can be closed before update operation. Even after closing the connection we can read and write data with the DataSet as per requirement. This is the standard mechanism of disconnected data architecture. The application is becomes more scalable as connection is held by application only when required.
- There are number of scenario when this architecture seem to be very useful: for example website making without network connectivity. In another example consider sales persons, who require customer data at the start of the day, they will require to sync up with the main database for getting the latest information available. Throughout the day, they will make necessary changes to existing customer data, include new customers, and enter new orders. This is possible as in the database of their region other people won't be making changes. At the end of the day, the network can be connected by the sales person to update the changes.
- Another scenario is creation of more scalable websites. With a SqlDataReader, we have to go to database whenever we show the page. Hence there is requirement of new connection for each page load. When there is increase in number of users, the scalability gets affected.
- The solution on it is to use DataSet which can be stored in cache once updated. For every request, this data of cache is retrieved which avoid the trip to the database and makes application very efficient.

Creating a DataSet Object

Syntax

```
DataSet CustDS = new DataSet();
```

- The DataSet constructor doesn't require parameters. However there is one overload that accepts a string for the name of the DataSet, which is used if you were to serialize the data to XML. Since that isn't a requirement for this example, I left it out. Right now, the DataSet is empty and you need a SqlDataAdapter to load it.

Creating A SqlDataAdapter

- To read/write data, the SQL commands and connection object are held by the SqlDataAdapter. It is initialized with a SQL select statement and connection object

```
SqlDataAdapter CustDA = new SqlDataAdapter("select CustID, CompName from Customers", conn);
```

- This code will generate a new SqlDataAdapter, CustDA. The SQL select statement is used to specify that what data should be stored in DataSet.
- Here it is considered as the connection object conn is already instantiated. Now SqlDataAdapter is used to open as well as close the connection.
- The connection object, conn, should have already been instantiated, but not opened. It is the SqlDataAdapter's responsibility to open and close the connection during Fill and Update method calls.
- Two option are available to add insert, update, and delete operations : using SqlDataAdapter properties or with a SqlCommandBuilder. Here we are going to use SqlCommandBuilder

```
SqlCommandBuilder cmd_Bldr = new SqlCommandBuilder(CustDA);
```

- The SqlCommandBuilder is instantiated with a single constructor parameter of the SqlDataAdapter.daCustomers, instance.
- To the constructor of SqlCommandBuilder, we have passed CustDA (SqlDataAdapter) instance. The SqlCommandBuilder will read the SQL select statement and infer the insert, update, and delete commands. SqlCommandBuilder is limited to work with single table.

Filling the DataSet

- After generating the DataSet and SqlDataAdapter instances, we have to fill the Dataset. The fill method of SqlDataAdapter is used for this purpose.

```
CustDA.Fill(CustDS, "Customers");
```

- We can see that the fill method accepts two parameters, a DataSet and a table name. The DataSet is filled with data. The table is the one that will be created in the DataSet. It is a customized table name. It is just like nickname to the original table of data source used to refer in the application.

Using the DataSet

- A DataSet can be bind with both ASP.NET and Windows forms DataGrids. Her we ssign the DataSet to a Windows forms DataGrid:

```
dgCustomers.DataSource = CustDS;
dgCustomers.DataMember = "Customers";
```

Updating Changes

- Now after finishing the changes in data, the changes should be written back to the database :

```
CustDA.Update(CustDS, "Customers");
```

- The Update method, here, is invoked on the SqlDataAdapter instance that has filled the CustDSDataSet. The second parameter specifies the name of table from dataset which has modified data.

Putting it All Together

```
using System;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Windows.Forms;
class DisconnectedDataform : Form
{
    private SqlConnection conn;
    private SqlDataAdapter CustDA;

    private DataSet CustDS;
    private DataGrid dgCustomers;
    private const string tableName = "Customers";
    public DisconnectedDataform()
    {
        InitData();
        dgCustomers = new DataGrid();
        dgCustomers.Location = new Point(5, 5);
dgCustomers.Size = new
Size(this.ClientRectangle.Size.Width - 10,
this.ClientRectangle.Height - 50);
        dgCustomers.DataSource = CustDS;
        dgCustomers.DataMember = tableName;
```

Fill dataset

Set up datagrid

```
        Button btnUpdate = new Button();
        btnUpdate.Text = "Update";
        btnUpdate.Location = new Point(
            this.ClientRectangle.Width/2 -
btnUpdate.Width/2,
            this.ClientRectangle.Height -
(btnUpdate.Height + 10));
        btnUpdate.Click += new
EventHandler(btnUpdateClicked);


        Controls.AddRange(new Control[] { dgCustomers,
btnUpdate });
    }

    public void InitData()
    {
        conn = new SqlConnection(

"Server=(local);DataBase=Northwind;Integrated
Security=SSPI");

        CustDS = new DataSet();

        CustDA = new SqlDataAdapter(
            "select CustomerID, CompanyName from
Customers", conn);

        SqlCommandBuilder cmdBldr = new
SqlCommandBuilder(CustDA);

        CustDA.Fill(CustDS, tableName);
    }

    public void btnUpdateClicked(object sender,
EventArgs e)
    {

        CustDA.Update(CustDS, tableName);
    }
    static void Main()
    {
        Application.Run(new DisconnectedDataForm());
    }
}
```

Make sure controls appear on form

Write changes back to DataBase

Syllabus Topic : Data Binding

3.2 Data Binding

Q. Explain data binding in detail. (4 Marks)

- There are number of controls provided by the Web Forms for supporting Data Binding. At design time or run time, these controls can be connected to ADO .Net components like DataView, DataSet, or a DataViewManager.

Data-Bound Controls

- A rich set of data-bound controls is provided by the ASP .Net. All these controls are easy to use. These controls are categorized in two groups: -single-item data-bound control and multi-item data-bound controls.

Fig. 3.2.1: Data-bound control

- In ASP.NET, these controls are created using a <asp:controlName> tag. Table 3.2.1 describes some common data-bound server-side controls.

Table 3.2.1

Control	ASP.NET Code	Description
DataGrid	<asp:DataGrid>	Database is displayed in a scrollable grid format and supports retrieval, add, update, sort, and paging
DataList	<asp:DataList>	Data is displayed in templates and style format

Control	ASP.NET Code	Description
ListBox	<asp:ListBox>	Display data in a list format
DropDownList	<asp:DropDownList>	Displays data in a drop down combo box format
CheckBox	<asp:CheckBox >	It can be connected to an item of the data source
CheckBoxList	<asp:CheckBoxList>	Display list of check boxes, which can be connected to the list of items in data source
Repeater	<asp:Repeater>	Displays a templated data-bound list
TextBox	<asp:TextBox>	Display data using the property

```
<title>Exampl of single value data binding ASP .Net
with variable</title>
<head>
<body>
  <form id="form1" runat="server">
  <div>
    <asp:Label ID="Label1" runat="server"
Font-Size="Large" ForeColor="Blue">
      EmpID: <%# EmpID %><br />
      empName: <%# EmpName %><br />
      City: <%# City %>
    </asp:Label>
  </div>
  </form>
</body>
</html>
```

Program 3.2.1

Write a program to display single value data binding

Solution :

Program to display single value data binding

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;

public partial class DataBindingExample :
System.Web.UI.Page
{
  public int EmpID;
  public string EmpName;
  public string City;
```

Syllabus Topic : Single Value Data Binding

3.2.1 Single Value Data Bound Controls

Q. Explain single value data bound control with example. (4 Marks)

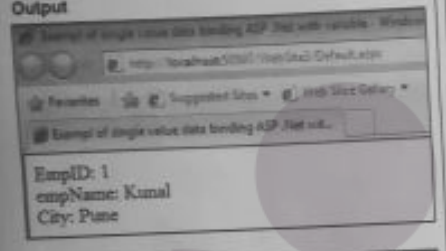
- The single-item data-bound controls as name suggests are used to display the value of a single item of a database table. Direct binding with the data source is not provided by these controls different properties like Text, Caption, or Value are used to show the data field.

Example

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Data_Binding_Example.aspx.cs"
Inherits="DataBindingExample" %>

<!DOCTYPE html>

<html runat="server">
```

```
protected void Page_Load(object sender, EventArgs e)
{
    EmpID = 1;
    EmpName = "Kunal";
    City = "Pune";

    this.DataBind();
}
```

Output



Syllabus Topic : Repeated - Value Data Binding

3.2.2 Multi Value Data Bound Controls

Q. Explain multi value data bound control with example. (4 Marks)

- The multi-item data bound controls are used to display the entire or a partial table. Direct binding to the data source is provided by these controls. The DataSource property of these controls is generally used to bind a database table.

Program 3.2.2

Write a program to display multi item data bound.

Solution :

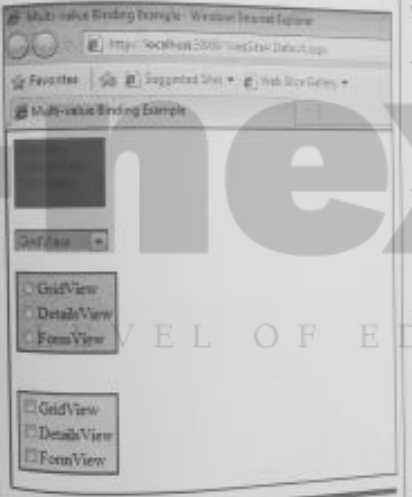
Program to display multi item data bound

```
<%@ Page Language="C#" %>
<%@ Import Namespace="System.Collections.Generic"
%>

<!DOCTYPE html>

<script runat="server">
    protected void Page_Load(object sender,
System.EventArgs e) {
        if (!this.IsPostBack) {
            List<string> DataToolBoxControls = new
List<string>();
            DataToolBoxControls.Add("GridView");
            DataToolBoxControls.Add("DetailsView");
            DataToolBoxControls.Add("FormView");

            List1.DataSource = DataToolBoxControls;
            DropDnList1.DataSource = DataToolBoxControls;
            RadioBList1.DataSource = DataToolBoxControls;
            ChkBoxList1.DataSource = DataToolBoxControls;

            this.DataBind();
        }
    }
</script>

<html>
<head id="Head1" runat="server">
    <title>Multi-value Binding Example</title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ListBox
            ID="List1"
            runat="server"
            BackColor="gray"
            ForeColor="Blue"
            >
        </asp:ListBox>
        <br /><br />
        <asp:DropDownList
            ID="DropDnList1"
            runat="server"
            BackColor="SkyBlue"
            ForeColor="Red"
            >
        </asp:DropDownList>
        <br /><br />
        <asp:RadioButtonList
            ID="RadioBList1"
            runat="server"
            BackColor="Pink"
            ForeColor="Black"
            BorderWidth="2"
            BorderColor="Blue"
            >
        </asp:RadioButtonList>
```



```
        <br /><br />
        <asp:CheckBoxList
            ID="ChkBoxList1"
            runat="server"
            BackColor="AntiqueWhite"
            ForeColor="Blue"
            BorderWidth="2"
            BorderColor="Red"
            >
        </asp:CheckBoxList>
    </div>
    </form>
</body>
</html>
```

Output



Syllabus Topic : Data Source Controls - SqlDataSource

3.2.3 Data Source Controls - SqlDataSource

Q. Explain how to issue commands with sqlDatasource control in brief. (4 Marks)

- The SqlDataSource control helps out to make use of Web server control to get data from data source. There are different data bound controls available such as GridView, FormView and DetailsView which are used to show and manipulate data on an ASP.NET Web page.
- To communicate with the multiple databases which can be supported by ADO.NET the SqlDataSource control takes help of various ADO.NET classes. For example Microsoft SQL Server with the help of the System.Data.SqlClient provider, System.Data.OleDb, System.Data.Odbc, and Oracle with the help of System.Data.OracleClient provider.
- SqlDataSource control can be also used to access and change data in an ASP.NET page without taking the help of ADO.NET classes directly. To connect to your database you can provide a connection string and describe the SQL statements or stored procedures that will work on your database. During run-time, this control by-default opens the database connection, executes the SQL statement or stored procedure that you have given, displays the preferred data (if any), and then closes the connection that has been opened.

Connecting the SqlDataSource Control to a Data Source

- While configuring a SqlDataSource control, you have to set some properties. The ProviderName property is set to know the type of database and the ConnectionString property is set to a particular connection string that contains necessary information needed to connect to the database.
- The parameters of a connection string can be different depending on what type of database the data source control is going to use. For example, the SqlDataSource control requires parameters such as a server name, database (catalog) name, and information about how to authenticate the user when connecting to a SQL Server.
- You can store connection string in the configuration settings as a part of your current application with the help of connectionString's configuration element rather than setting connection strings at design time as property settings in the SqlDataSource control.
- This will makes possible to handle connection strings separately of your ASP.NET code, containing encrypting them using Protected Configuration.
- The code snippet given below illustrates a connection to the SQL Server Northwind sample database using a connection string stored in the connectionStrings configuration element named MyNorthwind.

```
<%@ Page Language="C#" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" >
  <head runat="server">
```

```
<title>SqlDatasource Example</title>
</head>
<body>
  <form id="form1" runat="server">
    <asp:SqlDataSource
      id="SqlDataSource1"
      runat="server"
      DataSourceMode="DataReader"
      ConnectionString="<%$
ConnectionStrings:MyNorthwind%>"
      SelectCommand="SELECT Name FROM
employee">
    </asp:SqlDataSource>

    <asp:ListBox
      id="ListBox1"
      runat="server"
      DataTextField="Name"
      DataSourceID="SqlDataSource1">
    </asp:ListBox>

  </form>
</body>
</html>
```

Issuing Data Commands with the SqlDataSource Control

- The SqlDataSource control can supports four commands i.e. SQL queries. They are listed below
  1. SelectCommand
  2. InsertCommand
  3. UpdateCommand
  4. DeleteCommand
- Every command is a separate property of the data source control and you can define SQL statement to any of these commands for the data source control to execute.
- When you want to select all the columns from the database table you can use the asterisk (*) sign in the Select command, and if you use automatic code generation to perform operation like update or delete then remember that there is no space in columns names.
- You can also generate parameterized commands that contain placeholders for values to be passed at run time. The following example demonstrates a parameterized SQL Select command.

```
Select emp_ID, ename From employee Where salary =
@salary
```

- The DataSource control executes the commands when one of these four methods is called which is equivalent to it.

Returning DataSet or DataReader Objects

- To return data the SqlDataSource control can use one of the two forms :
  1. DataSet object
  2. DataReader object
- You can define which form you want to return by setting the DataSourceMode property of data source control.
- A DataSet object stores all the data in server memory, facilitating you to manipulate the data in a variety of ways after retrieving it.
- A data reader provides you a read-only view that can fetch single record.
- If you want to perform operations on the data such as filtering, sorting, or paging after retrieving it or if you want to maintain a cache then the DataSet is the best choice.
- In contrast, if you want to return the data and are using a control on the page to display that data then DataReader is best one. For example for returning data that you want to show in a ListBox, DropDownList, or GridView control where a list of results is displayed in a read-only format, Datareader is best to use.

Caching with the SqlDataSource Control

- When you retrieve data from the server you can save that data for future use.
- This data is nothing but the cache data. The SqlDataSource can allow this so as to improve the performance of your applications by avoiding costly queries.
- Caching is practical in nearly any situation where the data is not highly volatile and the cached results are small to avoid using the huge system memory.
- By default caching is not enabled, you have to explicitly enable it.
- To enable it set the property EnableCaching to true. You can also provide the cache time by setting the CacheDuration property to the number of seconds you want to cache data.
- The data source control keeps a separate cache entry for each combination of connection, select command, select parameters, and cache settings.

Filtering with the SqlDataSource Control

- To filter the data without re-running the query you have to enable caching for the SqlDataSource control and have to specify a dataset as a form for data returned by a Select query.
- The SqlDataSource control makes the use of FilterExpression property which allows you to give selection criteria that are applied to the data preserved by the data source control.

Sorting with the SqlDataSource Control

- The SqlDataSource control can supports the sort requests coming from the bound control. This can be done when the DataSourceMode is set to DataSet.

Syllabus Topic : Data Controls

3.3 Data Controls

Q. Explain different data controls of ADO.NET.

There are different data controls available in ADO.NET. They are explained briefly in subsequent sections.

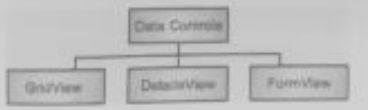


Fig. 3.3.1 : Data control of ADO.NET

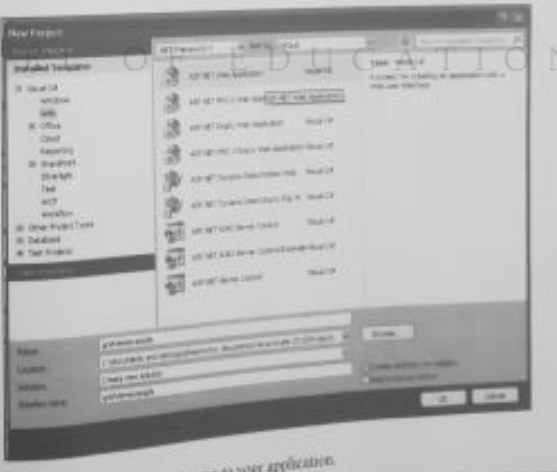
Syllabus Topic : GridView

3.3.1 GridView

Q. Explain GridView data controls of ADO.NET. Write down steps of creating GridView. **(2 Marks)**

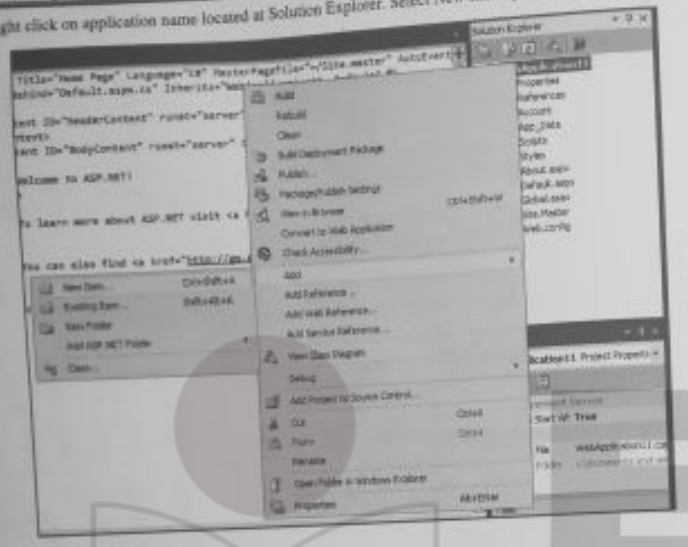
- The GridView is a very flexible grid control which can be used to display a data in tabular format. In grid view every record present inside your data source becomes an individual row and every field becomes an individual column.
- The GridView is considered as very powerful rich data control because it offers various functionalities. This functionalities includes support for automatic paging, sorting of a data, selecting a particular record, and editing a record.
- Following are the steps to create the Gridview control.

Step-1 : Click on File option and click on new project option. It will give the following screen.

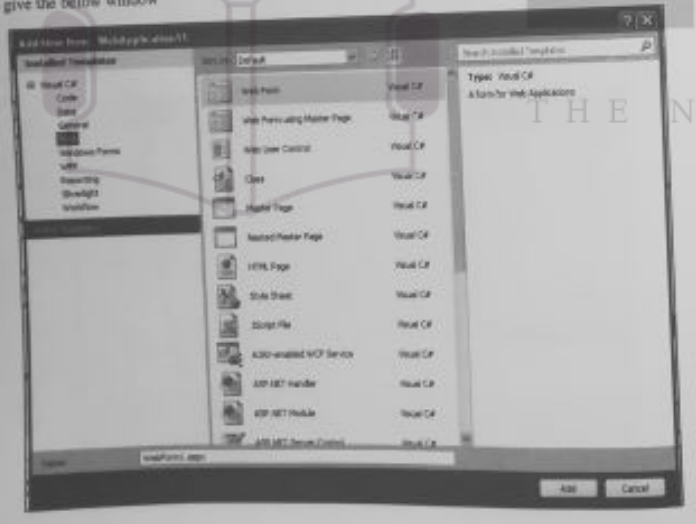


Select ASP.NET web application and give name to your application.

Step 2 : Right click on application name located at Solution Explorer. Select New Item option from Add drop-down list.

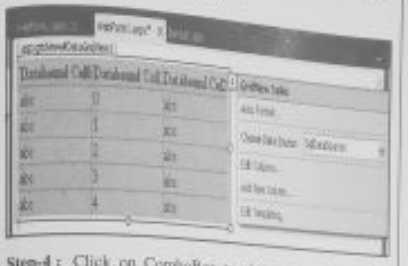


It will give the below window



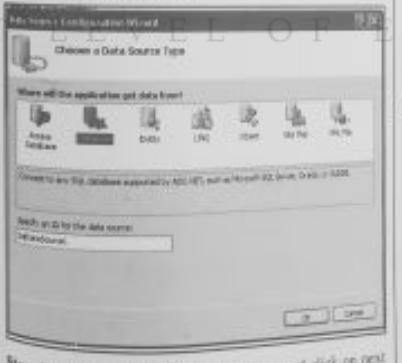
Select Web Form option and click on Add button to add web form to your application.

Step-3 : Take GridView control from the toolbox. Click on right corner arrow symbol. It will open a pop-up window as given below :



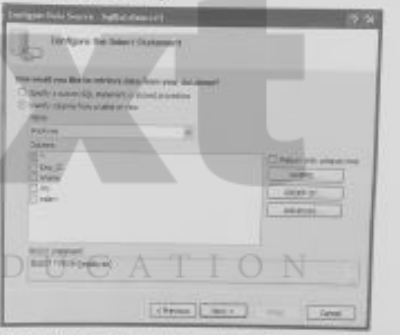
Step-4 : Click on ComboBox to choose the data source. Select new data source option from the list.

Step-5 : Now a new window will appear which will ask you for data source type. Select SQL database and click on OK button.

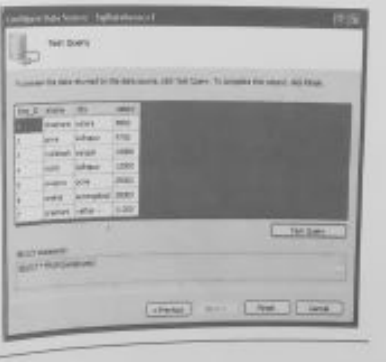


Step-6 : Now select the data connection and click on next button.

Step-7 : Now you can configure the sql statement or stored procedures if any.



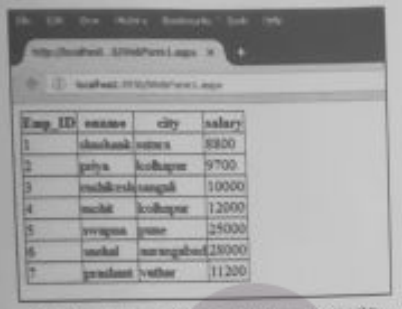
Step-8 : Click on next button and test the query if you want. It looks likes as follow.



Step-9 : Click the Finish button.

Now it's time to run the application.

Output



Emp_ID	ename	city	salary
1	shashank	satara	8800
2	priya	kolhapur	9700
3	sushilesh	sangali	10000
4	mohit	kolhapur	12000
5	swapna	pune	25000
6	snehal	aurangabad	28000
7	prashant	vather	11200

- Another way to attach the database to the GridView is by writing a code. Write the following code on the form load event.

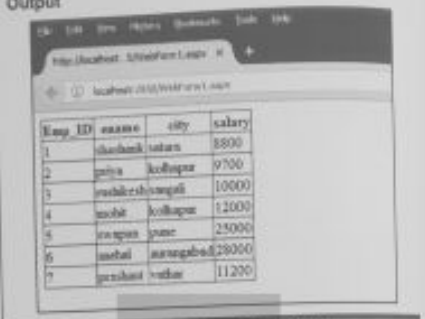
Program 3.3.1

Write a program to attach the database to the GridView.

Solution : Program to attach the database to the GridView

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Data.SqlClient;

namespace DatabindingWithdataGridView
{
    public partial class Form1 : System.Web.UI.Page
    {
        SqlDataAdapter dadapter;
        DataSet dset;
        string connstring =
"server=;database=Database1;user=;password=";
        private void Form1_Load(object sender, EventArgs e)
        {
            dadapter = new SqlDataAdapter("select * from
employee", connstring);
            dset = new System.Data.DataSet();
            dadapter.Fill(dset);
            dataGridView1.DataSource =
dset.Tables[0].DefaultView;
        }
    }
}
```

Output



Emp_ID	ename	city	salary
1	shashank	satara	8800
2	priya	kolhapur	9700
3	sushilesh	sangali	10000
4	mohit	kolhapur	12000
5	swapna	pune	25000
6	snehal	aurangabad	28000
7	prashant	vather	11200

Syllabus Topic : DetailsView

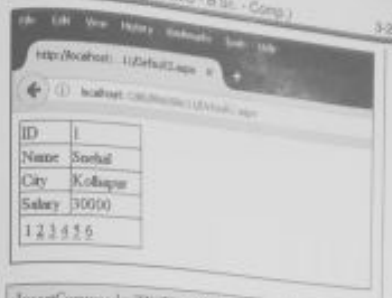
3.3.2 DetailsView

Q. Explain DetailsView data controls of ADO.NET. (4 Marks)

- As you know the GridView control is used to displays all of the records from its data source control at a time, whereas the DetailsView control is used to display single record from a data source at a time in the tabular form.
- The DetailsView control can be used to perform operations on the table data such as updating, inserting, and deleting records from the table.
- Sorting of data cannot be performed with the help of DetailsView control. The GridView control provides an interface for the users to navigate through the records.

☞ **To use the detailsView control perform the following steps :**

1. Take Detailsview control from the toolbox and perform the data binding to it as discussed in Gridview control.
2. When you run the application following output will be produced.
3. If you want to insert a new record to your table then set the AutoGenerateInsertButton to "True" and add the following code in Form tag.

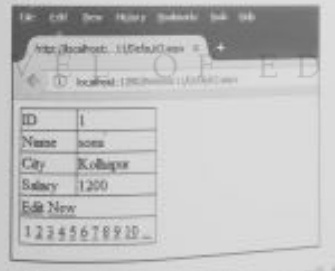


ID	1
Name	Snehal
City	Kolhapur
Salary	30000

1 2 3 4 5 6

```
InsertCommand="INSERT INTO employee(ID, Name, City,
Salary) VALUES (@ID, @name, @city, @salary)"
<InsertParameters>
<asp:Parameter Name="Id" Type="INT32" Size="20" />
                <asp:Parameter Name="name"
Type="string" Size="200" />
                <asp:Parameter Name="City"
Type="string" Size="50" />
                <asp:Parameter Name="Salary"
Type="INT32" Size="20" />
</InsertParameters>
```

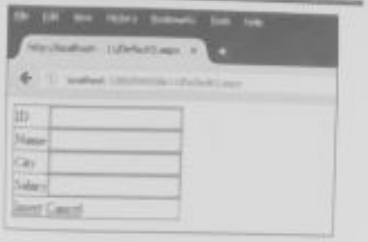
- When you run the application it will give the following window



ID	1
Name	xcxx
City	Kolhapur
Salary	1200
Edit New	

1 2 3 4 5 6 7 8 9 10 ...

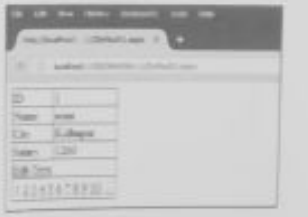
- When you click on new option new page will be loaded. It looks like
- Enter the data and click on Insert option. It will insert the record to your table



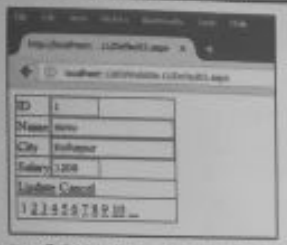
4. Similarly if you want to update a new record to your table then set the AutoGenerateEditButton to "True" and add the following code in Form tag

```
UpdateCommand="UPDATE employee SET ID = @ID,
Name = @name, City = @city, Salary = @salary
WHERE ID = @ID"
    <UpdateParameters>
    <asp:Parameter Name="ID" Type="Int32" />
    <asp:Parameter Name="name" Type="string"
Size="50" />
    <asp:Parameter Name="city" Type="string"
Size="20" />
    <asp:Parameter Name="salary" Type="string"
Size="50" />
    </UpdateParameters>
```

- When you run the application it will give the following window



- When you click on new option new page will be loaded. It looks like
- Edit the data and click on Update option. It will update the record.



- Following Snippet gives the complete code of the DetailsView Control.

```
<%@ Page Language="C#" AutoEventWireup="true"
CodeFile="Default2.aspx.cs" Inherits="Default2" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0
Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-
transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
  <title></title>
</head>
<body>
  <form id="form1" runat="server">
  <div>

    <asp:DetailsView ID="DetailsView1" runat="server"
AllowPaging="True"
      AutoGenerateRows="False"
DataSourceID="SqlDataSource1" Height="50px"
      Width="125px" AutoGenerateInsertButton="True"
      AutoGenerateDeleteButton="True"
AutoGenerateEditButton="True">
      <Fields>
        <asp:BoundField DataField="ID"
HeaderText="ID" SortExpression="ID" />
        <asp:BoundField DataField="Name"
HeaderText="Name" SortExpression="Name" />
        <asp:BoundField DataField="City"
HeaderText="City" SortExpression="City" />
        <asp:BoundField DataField="Salary"
HeaderText="Salary"
          SortExpression="Salary" />
      </Fields>
    </asp:DetailsView>
    <asp:SqlDataSource ID="SqlDataSource1"
runat="server"
      ConnectionString="<%$
ConnectionStrings:ConnectionString %>"
      SelectCommand="SELECT * FROM [employee]"

      InsertCommand="INSERT INTO employee(ID,
Name, City, Salary) VALUES (@ID, @name, @city,
@salary)"
      UpdateCommand="UPDATE employee SET ID =
@ID, Name = @name, City = @city, Salary = @salary
WHERE ID = @ID">
      <UpdateParameters>
        <asp:Parameter Name="ID" Type="Int32" />
        <asp:Parameter Name="name" Type="string"
Size="50" />
        <asp:Parameter Name="city" Type="string"
Size="200" />
        <asp:Parameter Name="salary" Type="string"
Size="50" />
      </UpdateParameters>
      <InsertParameters>
        <asp:Parameter Name="Id"
Type="INT32" Size="50" />
        <asp:Parameter Name="name"
Type="string" Size="200" />
        <asp:Parameter Name="City"
Type="string" Size="50" />
        <asp:Parameter Name="salary"
Type="INT32" Size="20" />

      </InsertParameters>
    </asp:SqlDataSource>

  </div>
  </form>
</body>
</html>
```

Syllabus Topic : FormView

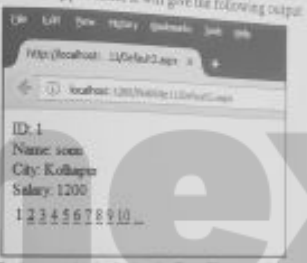
3.3.3 FormView

Q. Explain FormView data controls of ADO.NET. (4 Marks)

- Similar to DetailsView control, the FormView control also provides the facility of displaying single record at a time. The difference between the FormView and DetailsView control is that the DetailsView displays the record in tabular format where as FormView control does not contain any predefined layout for display the record.
- In FormView control you can specify the layout for displaying the records on the browsers.

To add a FormView control to a page

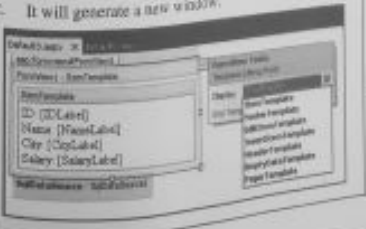
- Select and drag the FormView control from the Toolbox and place where you want.
- To bind the data source perform the data binding to it as discussed in Gridview control.
- Run the application, It will give the following output.



To interactively design the FormView templates

1. In Design view, click on the FormView control. Then on the Common FormView Tasks menu, click Edit Templates.

2. It will generate a new window.



3. From the Display dropdown, select the template you would like to edit. And run the application. Now you have selected the EditItemTemplate.

Syllabus Topic : Working with XML

3.4 Working with XML

- XML stands for eXtensible Markup Language.
- XML is basically designed to store and transport data.
- XML supports both human- and machine-readable data format.
- XML was designed to be self-descriptive.
- XML is a W3C Recommendation.
- XML is a simple text-based format which represents information in structured format like documents, transactions, data, invoices, books etc. This language is derived from comparatively older standard format called SGML (Standard Generalized Markup Language), to make it more suitable for Web use.

Need of XML

- There are several reasons for the need of XML as follows :

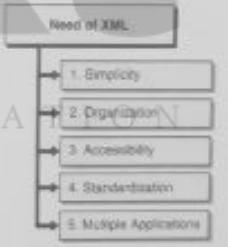


Fig. C3.2 : Need of XML

1. Simplicity

- XML can be easily understood. We can create our own tags and build the application.
- We are free to develop the systems as per our requirements and with our own conventions. This makes the thing very simple for us.

2. Organization

- The design process can be segmented to build the platform. Data can be stored on one page while the formatting rules can be stored on another page.

- It is possible to create the data page to store the content first and later on we can work on design. XML allows us to create the website in stages and stay organized in the entire process.

3. Accessibility

- Data can be divided in XML.
- This makes the access of data easy and fast whenever there is need of making change in the data.

4. Standardization

- XML is an international standard.
- This means XML document can be viewed anywhere in the world.

5. Multiple Applications

- "Write once, use anywhere, any number of times" rule is applied to XML.
- For XML data we can create any number of display pages as we want. XML allows us to create various styles and formats for a single page as per requirement.

XML Key Components

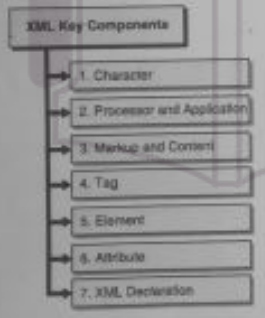


Fig. C3.3 : XML key components

1. Character

- An XML document is a string of characters.
- Almost every legal Unicode character may appear in an XML document.

2. Processor and Application

- The processor analyzes the markup and passes structured information to an application.
- The specification places requirements on what an XML processor must do and not do, but the application is outside its scope.
- The processor (as the specification calls it) is often referred to formally as an XML parser.

3. Markup and Content

- The characters making up an XML document are divided into markup and content, which may be distinguished by the application of simple syntactic rules.
- Generally, strings that constitute markup either begin with the character < and end with a >, or they begin with the character & and end with a ;. Strings of characters that are not markup are content.

4. Tag

- A tag is a markup construct that begins with < and ends with >.
- Tags come in three flavors
  - (i) start-tag, such as <section>
  - (ii) end-tag, such as </section>
  - (iii) empty-element tag, such as <line-break />

5. Element

- An element is a logical document component that either begins with a start-tag and ends with a matching end-tag or consists only of an empty-element tag.
- The characters between the start-tag and end-tag, if any, are the element's content, and may contain markup including other elements which are called child elements.

Example

```
<greeting>Hello, world!</greeting>.
Another is
<line-break />.
```

6. Attribute

- An attribute is a markup construct consisting of a name-value pair that exists within a start-tag or empty-element tag. An example is , where the names of the attributes are "src" and "alt", and their values are " Rose.jpg" and " Rose " respectively.

- Another example is <step number="3">Connect A to B.</step>, where the name of the attribute is "number" and its value is "3".
- An XML attribute can only have a single value and each attribute can appear at most once in each element. In the common situation where a list of multiple values is desired, this must be done by encoding the list into a well-formed XML attribute with some format beyond what XML defines itself. Usually this is either a comma or semi-colon delimited list or, if the individual values are known not to contain spaces, a space-delimited list can be used.
- Welcome! , where the attribute "class" has both the value "inner greeting-box" and also indicates the two CSS class names "inner" and "greeting-box".

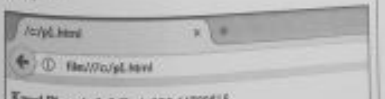
7. XML Declaration

- XML documents may begin with an XML declaration that describes some information about themselves.
- An example is <?xml version="1.0" encoding="UTF-8"?>.

XML Example 1

```
<?xml version="1.0" encoding="UTF-8"?>
<contact-info>
  <name>Kunal</name>
  <company>Phoenix InfoTech</company>
  <phone>020 64700515</phone>
</contact-info>
```

Output



- There are two classes for XML to read and write the data into the document. First one is XMLTextReader and XMLTextWriter respectively.

Syllabus Topic : XML Class - XMLTextWriter

3.4.1 The XMLTextWriter

Q. Explain XMLTextWriter class in detail.

- The XMLTextWriter class is inherited from the XMLWriter class and can be used to create any XML document.

Adding namespace Reference

- The Xml classes are defined in the System.XML namespace, so that our first step is to add this namespace to the project.

```
Imports System.IO
Imports System.Xml
```

Creating an XML Document

- You can create a XML file if it is already not present with the help of constructor of the XmlTextWriter class while writing into the file.
- Consider following example where the new XML file having name xmlf.xml is placed at D:\programs directory.

```
XmlTextWriter writer = new XmlTextWriter
("D:\programs\xmlf.xml", null);
```

- You can also reuse XML document to write data into XML file and can display XML contents on the Console. To do this just pass Console.Out as a parameter of the constructor.

```
XmlTextWriter writer = new XmlTextWriter(Console.Out);
```

Inserting Data to the Document

- To insert data into the document some methods are used. They are listed below.
  1. **WriteStartDocument** : It is used to start the new document.
  2. **WriteString** : It is used to write the string to the document.
  3. **WriteStartElement and WriteEndElement** : It is used in pair to add new elements in to the document.
  4. **WriteComment** : It is used to give comments in the document.
- Consider the following example which can use the XmlTextWriter class and associated methods.

Note : Write code in console application

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml;
```

```
namespace ConsoleApplication1
{
  class Program
  {
      public static void Main()
  {
   XmlTextWriter writer = new
XmlTextWriter("Bookdata.xml", null);

   writer.WriteStartElement("Books");

   writer.WriteElementString("id", "01");
   writer.WriteElementString("title", "Introduction to C");
   writer.WriteElementString("price", "450");

   writer.WriteElementString("id", "02");
   writer.WriteElementString("title", "System
Programming");
   writer.WriteElementString("price", "350");

   writer.WriteElementString("id", "03");
   writer.WriteElementString("title", "DOT.NET");
   writer.WriteElementString("price", "500");

   writer.WriteEndElement();

     writer.Close();
     Console.ReadLine();

    }
  }
}
```

Start with the root element

To write the sub element

Output

- The file named as bookdata.xml will be created at yourapplicationname\bin\Debug directory.

Syllabus Topic : XML Class - XMLTextReader

3.4.2 The XMLTextReader

Q. Explain XMLTextReader class in brief.

- To read data from the XML file the XmlTextReader class can be used. Like an XMLTextWriter class the XmlTextReader class is also comes in sequential manner and forward-only option that means once you have parsed the particular node and moved on then you cannot come back.
- The dynamic searching of node in xml file with the help of XMLTextReader is not possible. You have to traverse each and every node of xml file until the desired node is not encountered or till the end of file.
- Therefore, XmlTextReader class is most useful in circumstances where you're dealing with small files or the application requires the reading of the whole file contents.

Reading and Parsing XML Nodes

- To read the content of xml file you have to create an object of an XmlTextReader class and then inside the loop repeatedly call the XmlTextRead.Read () method until that method returns false.
- Consider the following example which will read the content of xml file which was created through XMLTextWriter class.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Xml;
namespace ConsoleApplication1
{
  class Program
  {
      public static void Main()
  {
    try
    {
      String out1="";
      String format = "XmlNodeType::{0,-12}{1,-
10}{2}";
```



```
XmlTextReader xmlreader = new
XmlTextReader("bookdata.xml");

      while (xmlreader.Read())
      {
        out1
=String.Format(format,(xmlreader.NodeType),xmlreader.Na
me,xmlreader.Value);

        Console.WriteLine(out1);
      }
    }
    catch (Exception ex)
    {
      Console.WriteLine(ex.Message);
        }
      Console.ReadLine();
        }
  }
}
```

Read the data from XML file

Output

Syllabus Topic : Caching - When to use Caching

3.5 Caching

3.5.1 When to use Caching

Q. What is mean by caching ? Explain when to use caching. (4 Marks)

- While developing an application some data is needed repeatedly. If the data is stored at different locations such as on server or on any remote machine then accessing the data every time requires lot of wastage of time as well as system resource to calculate the result that you require.
- Caching is a method mainly used for storing frequently used data/information in memory, so that, when the same data/information is needed next time, it could be directly retrieved from the memory rather than of being generated by the application.
- Caching method is particularly important for improving the performance in ASP.NET, as the pages and controls are generated dynamically. When data related transactions are considered then the caching is best choice because these are costly in terms of response time.
- Data that is used most frequently is placed in random access memory by caching as it can be accessed quickly.
- The ASP.NET runtime consist of a key-value mapping of common language runtime objects called cache. This exists with the application and is accessible via the HttpContext and System.Web.UI.Page.
- The data in cache will not be available in the following scenarios
  - (i) Lifetime of data stored in cache expires.
  - (ii) If the application free its memory.
  - (iii) Due to some reason caching does not take place.
- With the help of indexer you can access items from cache. You may also manage the lifetime of objects in the cache and creates associations between the cached objects and their physical sources.

3.5.2 Types of Caching

Q. Explain the different types of caching. (2 Marks)

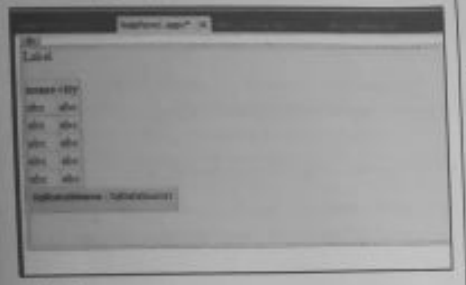
Following are different types of caching

Fig. 3.5.1 : Types of Caching

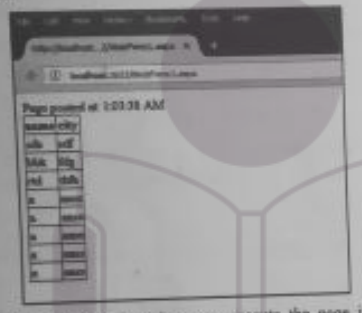
1. Output Caching

- Output cache is used to stores a replica of the final delivered HTML pages or piece of pages sent to the client.

The design of page should look as shown :



Output



- When for the first time you execute the page it will show the time of posted and each time you refresh the page, the page is reloaded and the time shown on the label changes.
- Now we can set the EnableCaching attribute of the data source control to be 'true' and set the Cache duration attribute to '70'. It will implement caching and the cache will expire every 70 seconds.

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
        ConnectionString="<%$
ConnectionStrings:ConnectionString %>"
        SelectCommand="SELECT * FROM [data]"
        EnableCaching="true"
CacheDuration="70"></asp:SqlDataSource>
```

Syllabus Topic : LINQ - Understanding LINQ

3.6 LINQ

Q. What is LINQ?

- LINQ Stands for Language Integrated Query. The LINQ is mainly used to provide consistent access to several data sources like databases and XML.

3.6.1 Introduction to LINQ

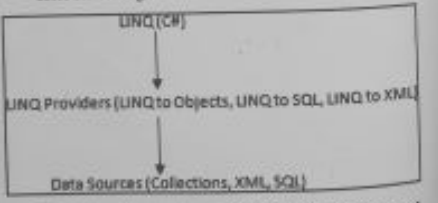
- Now days most of the applications are data-centric, however most of the data repositories are relational databases. Over the years developers have designed applications based on object models.
- The main task of objects is to connect with the data access components, which is also called the Data Access Layer. Consider the following points :

(i) It is not necessary that all the data needed in an application should be stored in the same source. The source could be different such as relational database, some business object, XML file, or a web service.

(ii) Accessing data from particular database or XML file is difficult and costly than accessing in-memory object.

(iii) The data needs to be sorted, ordered, grouped, altered etc. as the accessed data is not used directly.

- LINQ or Language-Integrated Query is a tool that can access all kinds of data sources which permit the combining data from different data sources and perform standard data processing operations.
- ADO.NET is considered as best choice to manipulate the data of a database if you don't want to use the LINQ. But ADO.NET uses tables, relations and other relational database constructs which needs to be mapped to the application objects. This contains more mistakes. Because of this LINQ gets higher priority by users for coding.



- The above diagram shows that the LINQ providers act as a bridge between LINQ and the data sources like SQL and XML, collections etc.
- The three main data providers provided by Microsoft are :
  1. LINQ to objects for querying in-memory collections
  2. LINQ to SQL for querying relational databases
  3. LINQ to XML for querying XML data

- LINQ is included in C# and VB.NET so that we can query any data source in any programming language that you desire. If you don't want to use LINQ and if we wanted to query a database we would use SQL but with LINQ we can do it in the programming language.

Syllabus Topic : LINQ Basics

3.6.2 LINQ Basics

Q. Explain various classes used in LINQ

- LINQ is collection of extensions to the .Net Framework 3.5 and its managed languages that set the query as an object. It describes a common syntax and a programming model to query several types of data with the help of common language.
- For example, querying the employee table in the database, using LINQ query in C#, the code would be

```
var data = from e in dataContext.employee
where e.salary >= 25000
select e;
```

- Where
  - The 'from' keyword can be used to logically loops through the contents of the collection.
  - The expression with the 'where' keyword is estimated for each and every object present inside the collection.
  - The 'select' statement is used to select the estimated object to add to the list being returned.
  - The 'var' keyword is used for variable declaration. You don't know the exact type of the returned object; it indicates that the information will be inferred dynamically.
- LINQ query can be applied to any data-bearing class that is inherited from IEnumerable<T>, here T is any data type, for example, List<Book_data>.

Example

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public class Book_data
{
    public string ID { get; set; }
    public string Title { get; set; }
    public decimal Price { get; set; }

    public static List<Book_data> GetBooks()
    {
        List<Book_data> list = new List<Book_data>();

        list.Add(new Book_data
        {
            ID = "01",
            Title = "System Programming",
            Price = 620 });

        list.Add(new Book_data
        {
            ID = "02",
            Title = "Web Technology",
            Price = 250 });

        list.Add(new Book_data
        {
            ID = "03",
            Title = "Automata Theory",
            Price = 320 });

        list.Add(new Book_data
        {
            ID = "04",
            Title = "Visual Basic",
            Price = 180 });

        list.Add(new Book_data
        {
            ID = "05",
            Title = "Programming in C",
            Price = 210 });

        return list;
    }
}
```

Inserting data to List

- Now take the label control from the toolbox to displays the titles of the books from bookdata. The Page_Load event creates a list of books and returns the titles by using LINQ query.

- When the next client requests for this page, rather than regenerating the page, a cached replica of the page is sent, which will save the time to give response to clients.

2. Data Caching

- Data caching is used to cache data from various data sources. Until the cache is not expired, a request for the data will be accomplished with the help of cache.
- When the cache is expired, the new data is obtained by the data source and the cache is again filled with data.

3. Object Caching

- Object caching is a technique used for caching the objects present on a page.
- For example data-bound controls - gridview, detailsview, etc. The cached data is kept in server memory.

4. Class Caching

- When you run for the first time, the web pages or web services are compiled into a page class in the assembly. Then the assembly is cached in the server.
- Next time when a request is came for the page or service, the cached assembly is referred to. When the source code is modified, the common language runtime recompiles the assembly.

5. Configuration Caching

- The application's configuration information is stored in a configuration file.
- Similarly configuration caching holds the configuration information in the server memory.

Syllabus Topic : Output Caching

3.5.3 Output Caching

Q. Write a short note on output caching. (2 Marks)

- There are some complex processes in page rendering such as database access, rendering complex controls etc. Output cache helps to skip the round trips to server with the help of caching data in memory. It is possible cache even the entire page.
- The output caching is done with the help of OutputCache directive. It enables output caching and provides certain control over its behavior.

Syntax

```
<%@ OutputCache Duration="20" VaryByParam="None" %>
```

- Insert this directive under the page directive. This informs the environment to cache the page for 20 seconds. The following event handler for page load would help in testing that the page was truly cached.

```
protected void Page_Load(object sender, EventArgs e)
{
  Thread.Sleep(20000);
  Response.Write("This page was generated and cache at:"
+ DateTime.Now.ToString());
}
```

- The Thread.Sleep() method is used to stop the process thread for the specific time. In the above example, the thread is stopped for 20 seconds, so when the page is loaded for first time, it takes 20 seconds.
- When next time you refresh the page it does not take any time, because the page is retrieved from the cache.
- The Table 3.5.1 contains the attributes for OutputCache directive, which helps in controlling the behavior of the output cache

Table 3.5.1

Sr. No.	Attribute	Values	Description
1.	DiskCacheable	true/false	This attribute is used to state that output could be written to a disk based cache.
2.	NoStore	true/false	It will be used to state that the "no store" cache control header is sent or not.
3.	CacheProfile	String name	It indicates the name of a cache profile for storing in web.config.
4.	VaryByCustom	Browser Custom string	This attribute is used to inform to vary the output cache by the name of browser and version or by a custom string.

Sr. No.	Attribute	Values	Description
5.	Location	Any	Any : page may be cached anywhere.
		Client	Client : contents of cache will reside at client's machine.
		Downstream	Downstream : contents of cache will reside at downstream and server both.
		Server	Server : contents of cache will reside at server's machine.
		None	None : It will disable the caching.
6.	Duration	Number	It used to state for how much time the page or control is cached.

- Now you can add a text box and a button to the previous example and add this event handler for the button.

```
protected void btnImagic_Click(object sender, EventArgs e)
{
  Response.Write("<br><br>");
  Response.Write("<h3> Welcome, " + txtUname.Text
+ "</h3>");
}
```

- Modify the OutputCache directive :

```
<%@ OutputCache Duration="50"
VaryByParam="txtname" %>
```

- When the program is executed, ASP.NET caches the page on the basis of the name in the text box.

Syllabus Topic : Data Caching

3.5.4 Data Caching

Q. Write a short note on data caching. (2 Marks)

- The key characteristic of data caching is to cache the data through data source controls. As we already know that the data source controls is used to signify the data in a data source.
- The data source controls are inherited from the abstract class named as DataSourceControl. It will have the following properties for implemeneting caching :

Properties for implementing coching

1. **CacheDuration** : This property is used to set the number of seconds for which the data source will cache data.
2. **CacheExpirationPolicy** : This property is used to define the behavior of cache when the data in cache has expired.
3. **CacheKeyDependency** : This property is used to identify a key for the controls that auto-expires the content of its cache when deleted.
4. **EnableCaching** : This property is used to specify whether the data will be cache or not.

Example

To illustrate the data caching perform the following steps :

1. Create a new website in visual studio.
2. Right click on website name present at solution explorer and choose new item from add option. A new window will appear, click on web form option.
3. Add a SqlDataSource control with the database connection.
4. Add a label to the page, which is used to show the response time for the page.

```
<asp:Label ID="labelTime"
runat="server"></asp:Label>
```

5. Add an event handler for the page load event :

```
protected void Page_Load(object sender, EventArgs e)
{
  labelTime.Text = String.Format("Page posted at: {0}",
DateTime.Now.ToLongTimeString());
}
```

```
public partial class _Default : System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
  {
    List<Book_data> books = Book_data.GetBooks();
    var booktitles = from t1 in books select t1.Title;

    foreach (var title in booktitles)
      data.Text += String.Format("{0} <br />", title);
  }
}
```

Retrieving data from the list

- When you execute the above code following window will give the results of the query

LABELSYSTEM PROGRAMMING
WEB TECHNOLOGY
AUTOMATA THEORY
VISUAL BASIC
PROGRAMMING IN C

- The above LINQ expression

```
var booktitles = from t1 in books select t1.Title;
```

Is equivalent to the following SQL query

```
SELECT Title from Book_data;
```

LINQ Clauses

In this section you can see the several clauses of LINQ. They are described in below

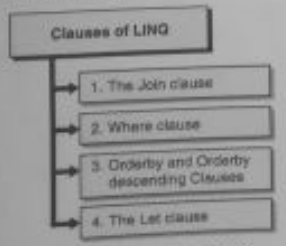


Fig. C3.4 : Clauses of LINQ

1. The Join clause

- In SQL the 'join clause' is used for joining two different data tables and displays a data set containing columns from both the tables.
- LINQ is also able to do this operation. To perform this, add another class named otherbookdata.cs in the previous project.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;

public class other_data
{
  public int total_copies { get; set; }
  public int total_pages { get; set; }
  public string ID { get; set; }

  public static IEnumerable<other_data> getother_data()
  {
    other_data[] data2 =
    {
      new other_data { ID = "01", total_pages=134,
total_copies = 13000},
      new other_data { ID = "02", total_pages=456,
total_copies = 45000},
      new other_data { ID = "03", total_pages=348,
total_copies = 56000},
      new other_data { ID = "04", total_pages=786,
total_copies = 10000},
      new other_data { ID = "05", total_pages=598,
total_copies = 78000},
      new other_data { ID = "06", total_pages=380,
total_copies = 50000},
      new other_data { ID = "07", total_pages=650,
total_copies = 37000},
    };

    return data2.OfType<other_data>();
  }
}
public class Book_data
{
  public string ID { get; set; }
  public string Title { get; set; }
  public decimal Price { get; set; }
```



```
public static List<Book_data> GetBooks()
{
  List<Book_data> list = new List<Book_data>();
  list.Add(new Book_data
  {
    ID = "01",
    Title = "System Programming",
    Price = 620
  });
  list.Add(new Book_data
  {
    ID = "02",
    Title = "Web Technology",
    Price = 250
  });

  list.Add(new Book_data
  {
    ID = "03",
    Title = "Automata Theory",
    Price = 320
  });

  list.Add(new Book_data
  {
    ID = "04",
    Title = "Visual Basic",
    Price = 180
  });

  list.Add(new Book_data
  {
    ID = "05",
    Title = "Programming in C",
    Price = 210
  });

  return list;
}
}

public partial class _Default : System.Web.UI.Page
{
  protected void Page_Load(object sender, EventArgs e)
  {
    IEnumerable<Book_data> books =
Book_data.GetBooks();
    IEnumerable<other_data> total_copies =
other_data.getother_data();

    var booktitles = from b in books
                     join c in total_copies on b.ID equals c.ID
                     select new { Name = b.Title, total_pages =
c.total_pages };

    foreach (var title in booktitles)
      data1.Text += String.Format("{0} <br />", title);
  }
}
```

Output

```
LABEL{ NAME = SYSTEM PROGRAMMING, TOTAL PAGES = 134 }
{ NAME = WEB TECHNOLOGY, TOTAL PAGES = 456 }
{ NAME = AUTOMATA THEORY, TOTAL PAGES = 348 }
{ NAME = VISUAL BASIC, TOTAL PAGES = 786 }
{ NAME = PROGRAMMING IN C, TOTAL PAGES = 598 }
```

2) Where clause

- The 'where clause' is used to add some conditional filters to the query. Consider the following example, which shows the names and the number of pages which are less than 400.
- For doing this make following changes to Page_Load event handler

```
protected void Page_Load(object sender, EventArgs e)
{
  IEnumerable<Book_data> books =
Book_data.GetBooks();
  IEnumerable<other_data> total_copies =
other_data.getother_data();

  var book1 = from x in books
              join y in total_copies on x.ID equals y.ID
              where y.total_pages <= 400
              select new { Name = x.Title, Pages =
y.total_pages };
  foreach (var t2 in book1)
    data1.Text += String.Format("{0} <br />", t2);
```

- The query returns only those rows, where the number of pages is less than 500.

Output

```
LABEL{ NAME = SYSTEM PROGRAMMING, PAGES = 134 }
{ NAME = AUTOMATA THEORY, PAGES = 348 }
{ NAME = DATABASE SYSTEM, PAGES = 380 }
{ NAME = DATA STRUCTURE, PAGES = 380 }
```

3. Orderby and Orderby descending Clauses

- The Orderby clause is used for sorting the query results.
- To display the name of the book and the number of pages, sorted by the price, write the following code in the Page_Load event handler.

```
protected void Page_Load(object sender, EventArgs e)
  {
    IEnumerable<Book_data> books =
Book_data.GetBooks();
    IEnumerable<other_data> total_copies =
other_data.getother_data();

    var book1 = from b in books
                join s in total_copies on b.ID equals s.ID
                where s.total_pages <= 400
                select new { Name = b.Title, Pages =
s.total_pages };

    var book1titles = from b in books
                join s in total_copies on b.ID equals s.ID
                orderby b.Price
                select new { Name = b.Title, Pages =
s.total_pages, Price = b.Price };

    foreach (var t2 in book1)
      data1.Text += String.Format("{0} <br />", t2);
  }
```

Output

```
LABEL{ NAME = SYSTEM PROGRAMMING, PAGES = 134 }
{ NAME = AUTOMATA THEORY, PAGES = 348 }
{ NAME = DATABASE SYSTEM, PAGES = 380 }
{ NAME = DATA STRUCTURE, PAGES = 380 }
```

4. The Let clause

- The let clause is used to define a variable and assigning it a value calculated from the data values. For example, to calculate the total number of copies, you need to calculate :

```
Finalnumberofcopies = Price of the Book data* otherdata
```

- To achieve this, add the following code snippets in the Page_Load event handler

```
protected void Page_Load(object sender, EventArgs e)
  {
    IEnumerable<Book_data> books =
Book_data.GetBooks();
    IEnumerable<other_data> total_copies =
other_data.getother_data();

        var book1 = from b in books
                    join s in total_copies on b.ID equals s.ID
                    let total = (b.Price * s.total_copies)
                    select new { Name = b.Title, TotalSale =
total };
        foreach (var t3 in book1)
          data1.Text += String.Format("{0} <br />", t3);
  }
```

Output

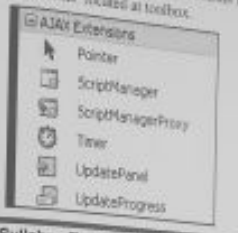
```
LABEL{ NAME = SYSTEM PROGRAMMING, TOTALSALE = 8060000 }
{ NAME = WEB TECHNOLOGY, TOTALSALE = 11250000 }
{ NAME = AUTOMATA THEORY, TOTALSALE = 17920000 }
{ NAME = VISUAL BASIC, TOTALSALE = 1800000 }
{ NAME = PROGRAMMING IN C, TOTALSALE = 16380000 }
{ NAME = DATABASE SYSTEM, TOTALSALE = 25500000 }
{ NAME = DATA STRUCTURE, TOTALSALE = 17500000 }
```

Syllabus Topic : ASP.NET AJAX

3.7 ASP.NET AJAX

- AJAX can be abbreviated as Asynchronous JavaScript and XML. AJAX is a cross platform technology which is used to speed up response time. In ASP.NET the AJAX server controls are used to add script to your page which can be executed and also processed by the browser.

- The AJAX server controls contain methods and event handlers associated with each control similar to other server controls, which are processed on the server side.
- The Visual Studio contains a set of controls named as 'AJAX Extensions' located at toolbox.



Syllabus Topic : ScriptManager

3.7.1 The ScriptManager Control

- The ScriptManager control is considered as the most significant control.
- If other controls which are used on page requires to work properly then the ScriptManager Control have to be present on the page.

Syntax

```
<asp:ScriptManager ID="ScriptManager1"
runat="server">
</asp:ScriptManager>
```

- The ScriptManager control is responsible for taking care of the client-side script.

Syllabus Topic : Partial Refreshes

3.7.2 Partial Refreshes

Q. Write a short note on partial refreshes. (2 Marks)

- The partial refreshes can refreshes only specifies part of the page not the whole page at a time. The partial refreshes can be done with the help of UpdatePanel control.
- The UpdatePanel control is inherited from Control class and it is also called as container control. The UpdatePanel control behaves like a container for the child controls which are placed inside it.
- The UpdatePanel does not contain its own interface. When a control inside it triggers a post back, the UpdatePanel get involved to start the post asynchronously and update only that piece of the page.
- Consider an example, the update panel contains a button control and when it is clicked by user, only the controls which reside within the update panel will have an effect on it, the controls on the outside the panel will not have any effect. This process is known as partial post back or the asynchronous post back.

Example

- Create a new website and add script manager control from AJAX Extension tool. Insert an update panel. Place a button control along with a label control within the update panel control. Place another set of button and label outside the panel.
- The design view looks as follows :

- The source file is as follows

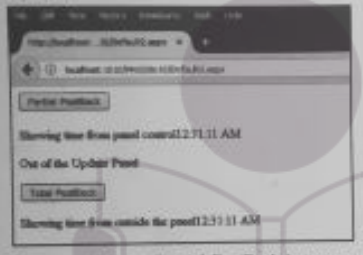
```
<%@ Page Language="C#" AutoEventWireup="true" CodeFile="Default.aspx.cs" Inherits="_Default" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title></title>
</head>
<body>
    <form id="form1" runat="server">
    <div>
        <asp:ScriptManager ID="ScriptManager1" runat="server" />
    </div>

        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Button ID="btnpartial" runat="server"
onclick="btnpartial_Click" Text="Partial PostBack" />
                <br />
                <br />
                <asp:Label ID="labelpartial"
runat="server"></asp:Label>
            </ContentTemplate>
        </asp:UpdatePanel>

        <p> </p>
        <p>Out of the Update Panel</p>
        <p>
            <asp:Button ID="btntotal" runat="server"
onclick="btntotal_Click" Text="Total PostBack" />
```

```
</p>

<asp:Label ID="labeltotal"
runat="server"></asp:Label>
</form>
```

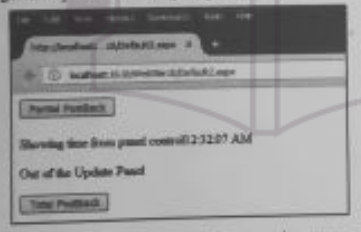
- Write following code on both button controls for the event handler :

```
string t1 = DateTime.Now.ToLongTimeString();
labelpartial.Text = "Showing time from panel control" + t1;
labeltotal.Text = "Showing time from outside the panel" +
t1;
```

- When you run the application and click on Total PostBack button you will get the updated time on both the labels.



- When you click on **Partial PostBack** button you will get the updated time only on partial label.



- A page can hold many update panels, every panel consist other controls such as grid and displaying different part of data.
- When a total post back is occurred, the update panel content is updated by default. You can change this default mode by changing the UpdateMode property of the control.

Syllabus Topic : Progress Notification

3.7.3 Progress Notification

Q. Explain progress notification. **(2 Marks)**

The progress notification can be achieved with the help of UpdateProgress control.

- The UpdateProgress control is useful because it provides a kind of feedback on the browser while one or more update panel controls are being updated.
- For example, while a user logged in or waiting for response from server while doing some database related task, it show "Loading page..." indicating the work is in progress.

☞ **Syntax**

```
<asp:UpdateProgressID="UpdateProgress1"
runat="server" DynamicLayout="true"
AssociatedUpdatePanelID="UpdatePanel1" >

  <ProgressTemplate>
    Loading your page...
  </ProgressTemplate>

</asp:UpdateProgress>
```

- The above code sets a message within the ProgressTemplate tag. You can also specify any image or other control in it.
- The UpdateProgress control is shown for each asynchronous postback unless it is assigned to a single update panel using the AssociatedUpdatePanelID property.

☞ **Properties of the UpdateProgress Control**

- Table 3.7.1 displays the properties of the updateProgress control.

Table 3.7.1

Sr. No.	Properties	Description
1.	AssociatedUpdate PanelID	This property is used to get and set the ID of the update panel which relates it with the UpdateProgress.
2.	Attributes	This property is used to get and set value of cascading style sheet (CSS) attributes of the UpdateProgress control.
3.	DisplayAfter	This property is used to get and set the time in milliseconds after which the progress template is showed. The default value of DisplayAfter property is 500 milliseconds.

Sr. No.	Properties	Description
4.	DynamicLayout	This property is used to state which progress template is rendered dynamically.
5.	ProgressTemplate	This property indicate the template showed at the time of asynchronous post back which takes more time than the DisplayAfter time.

☞ **Methods of the UpdateProgress Control**

- Table 3.7.2 displays the methods of the update progress control.

Table 3.7.2

Methods	Description
GetScriptDescriptors	This property is used to provide a list of components, actions, and client controls that are required for the UpdateProgress control's client functionality.
GetScriptReferences	This property is used to return a list of client script library dependencies for the UpdateProgress control.

Syllabus Topic : Timed Refreshes

3.7.4 Timed Refreshes

Q. Write a short note on timed refreshes.

- The main purpose of timer control is to start the post back automatically.
- This could be done in two way?

(1) Setting the Triggers property of the UpdatePanel control :

```
<Triggers>
  <asp:AsyncPostBackTrigger ControlID="autopanel"
EventName="Click" />
</Triggers>
```

(2) By placing a timer control inside the UpdatePanel we can make it to behave as a child control trigger. One timer can be the trigger for many UpdatePanels.

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server"
UpdateMode="Always">

  <ContentTemplate>
    <asp:Timer ID="Timer1" runat="server"
Interval="1000">
    </asp:Timer>
    <asp:Label ID="Firstlable" runat="server"
Height="203px" style="width:405px" >
    </asp:Label>
  </ContentTemplate>

</asp:UpdatePanel>
```

Review Questions

Q. 1 What are the objects of ADO.net? Explain any one. **(Refer Section 3.2.1)** **(4 Marks)**

Q. 2 Explain Direct Data Access in detail. **(Refer Section 3.1.3)** **(10 Marks)**

Q. 3 Explain data binding in detail. **(Refer Section 3.2)** **(4 Marks)**

Q. 4 Explain different data controls of ADO.NET. **(Refer Section 3.3)** **(10 Marks)**

Q. 5 Explain XMLTextWriter class in detail. **(Refer Section 3.4.1)** **(4 Marks)**

Q. 6 Explain the different types of caching. **(Refer Section 3.5.2)** **(2 Marks)**

Q. 7 Explain various clauses used in LINQ. **(Refer Section 3.6.2)** **(4 Marks)**

Q. 8 Write a short note on timed refreshes. **(Refer Section 3.7.4)** **(4 Marks)**

□□□