ASP.NET using C# Notes

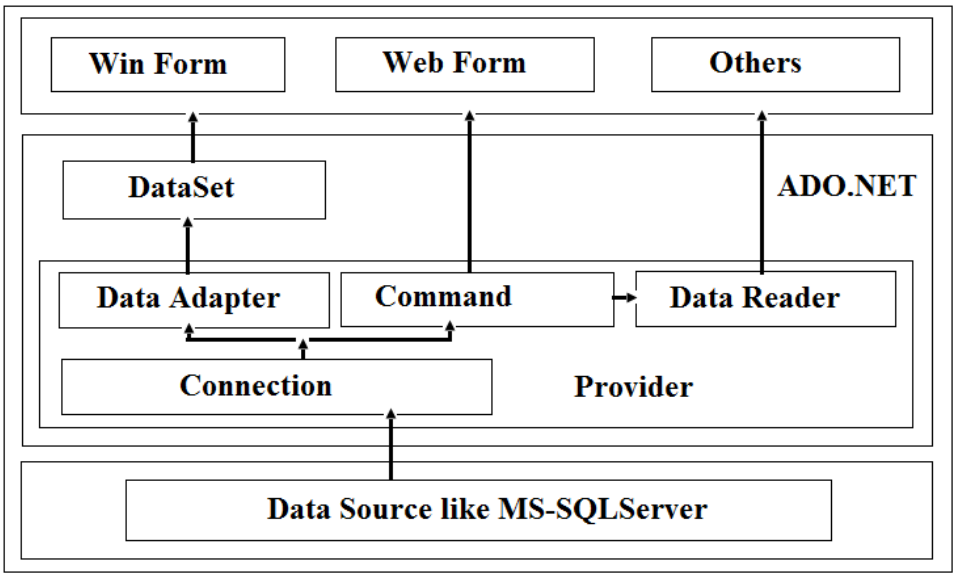
Unit-3

Architecture of ADO.NET:

The full form of ADO.NET is ***ActiveX Data Object.Net***. Basically it is container of all the standard classes which are responsible for database connectivity of .net application to the any third party database software like *Ms Sql Server, Ms Access, MySql, Oracle* or any other database software. All classes belongs to ADO.NET are defined and arrange in *"System.Data"* namespace.

एडीओ.नेट का पूरा नाम "ActiveX डाटा ऑब्जेक्ट.नेट" है। मुख्यत: यह उन सभी क्लाससेस का एक संग्रह है जिनका उपयोग किसी भी थर्ड पार्टी डेटाबेस सॉफ्टवेर के साथ डेटाबेस कनेक्टिविटी करने के लिए करते हैं जैसे *MS SqlServer, Ms Access, MySql, Oracle* या कोई भी अन्य डेटाबेस सॉफ्टवेर। एडीओ.नेट मे आने वाली सभी क्लाससेस *"System.Data"* नेमस्पेस मे परिभाषित होती हैं।

Architecture of ADO.NET



From above diagram it is clear that ADO.NET contain following important components.

उपरोक्त चित्र से स्पस्ट है कि ADO.NET मे निम्न महत्वपूर्ण कम्पोनेंट्स उपलब्ध रहते है।

1. Connection
2. Command
3. DataReader
4. DataAdapter
5. Dataset

1. Connection:

This component of ADO.NET is responsible to connect our .net application with any data source like MS SQL Server database.

इस कोम्पोनेंट का उपयोग किसी भी डाटा सोर्स जैसे एमएस-एसक्यूएल सर्वर के साथ कनैक्ट करने के लिए करते है।

2. Command:

This component contain classes for executing any INSERT, UPDATE, DELETE and SELECT command over database software using active connection.

इसका उपयोग किसी भी एसक्यूएल कमांड को एक्टिव कनैक्शन पर एक्सकुट करने के लिए करते है।

3. DataReader:

This component contains classes that capable to hold reference of records that retrieved by Command class after executing SELECT Query.

इसका उपयोग सिलैक्ट क्यूरी को रन करने पर प्राप्त रेकॉर्ड्स के रिफ्रेन्स को होल्ड करने के लिए करते है।

4. Data Adapter:

This component is capable to perform all tasks like executing INSERT, UPDATE, DELETE and SELECT command on given Connection. The most important use of this component is executing SELECT query for a number of records and fill dataset components.

इसकी सहायता से किसी भी एसक्यूएल कमांड को दिये गए कनैक्शन पर एक्सकुट कर सकते है। इसका उपयोग मुख्य रूप से सिलैक्ट कमांड से प्राप्त रेकॉर्ड्स को डेटासेट मे भरने के लिए करते है।

5. Dataset:

This component is the main source of records for .net control like GridView. It has capability to generate a number of tables to hold records retrieved from DataAdapter or XML.

GridView जैसे कंट्रोल पर डेटाबेस का रेकॉर्ड दिखाने के लिए Dataset सबसे मुख्य क्लास है। इसमे कई सारी tables को generate करने की क्षमता होती हे जिन्हे DataAdapter या XML की सहायता से भरा जाता है।

Connection String:

The first and essential requirement for data communication between .NET application and any database software (like Ms SQLServer) is connection establishments between them.
किसी भी .नेट एप्लिकेशन और डेटाबेस सॉफ्टवेर (जैसे एमएस एसक्यूएल सर्वर) के बीच डाटा का आदान-प्रदान करने के लिए सबसे पहले दोनों के बीच कनैक्शन को जोड़ना अनिवार्य होता है।

The connection will established using *driver name / provider name*, *data source file name* and *security options* that enclosed in string formats called ***connection string***.
कनैक्शन जोड़ने के लिए सबसे पहले एक कनैक्शन स्ट्रिंग बनानी होती है जिसमे डेटाबेस सॉफ्टवेर से संबन्धित सभी आवश्यक जानकरियों को शामिल किया जाता है जैसे ड्राईवर का नाम / प्रोवाइडर का नाम, डाटा सोर्स फ़ाइल, सेक्युर्टी विकल्प आदि।

Connection string example:

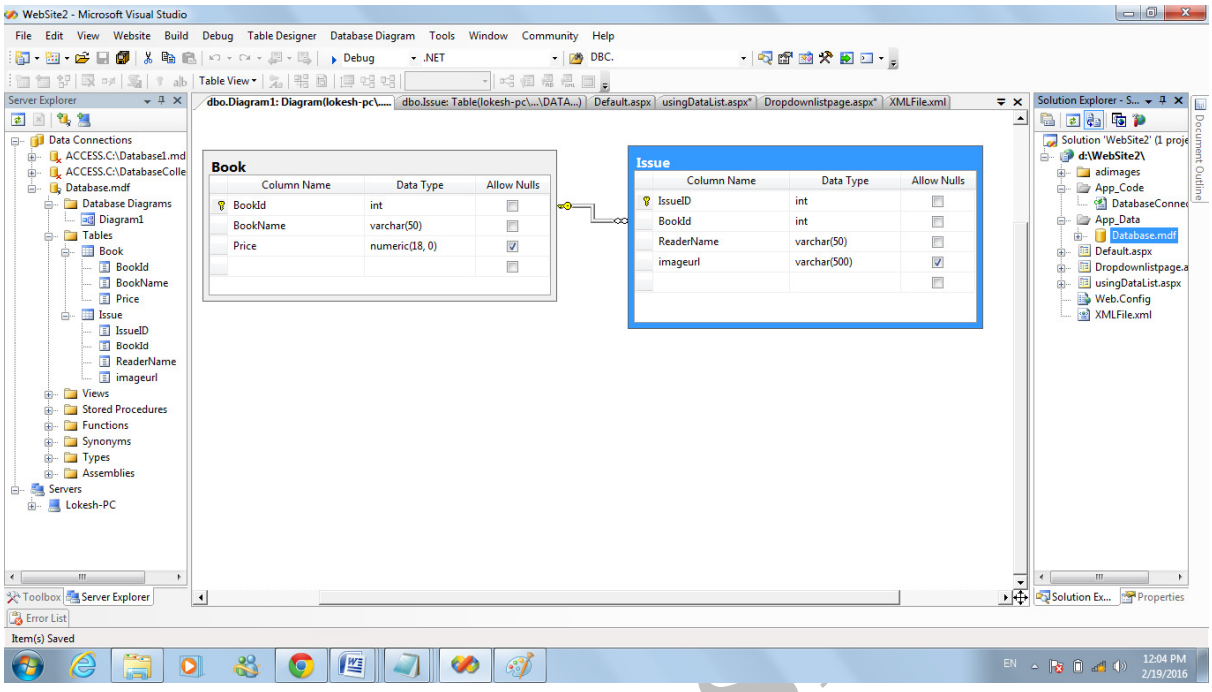
```
"Data Source= .\SQLEXPRESS;
AttachDbFilename=|DataDirectory|\Database.mdf;
Integrated Security=True; User Instance=true"
```

Only accurate information in connection string is responsible for establishing connection with database file.
केवल सही कनैक्शन स्ट्रिंग ही चाहे गए डेटाबेस फ़ाइल के साथ कनैक्शन को स्थापित करने के लिए जिम्मेदार होती है।

ASP.NET using C# Notes

Working with MS-SQL Server:

We have to create following table's structure with relationship.



Creating tables:

1. Create ***App_Data*** system folder in web folder.
2. Right click on App_Data→Add New Item→SQL Database→***Database.mdf*** (might change database filename)→Add.(Wait)
3. From menu bar click on View→Server Explorer→click ***Database.mdf*** (database file)
4. In Server Explorer, Right click on Table→Add New Table→Specify all fields with primary key, not null and data type constraints. (To make auto increment field, Set Identity Specification 'yes')
5. Click Save button to rename table1 as Book.
6. Repeat step4 and step5 for Issue table.

Creating relationship between two tables:

Before relationship we must ensure that primary key field (BookId:int) of master table(Book) also defined in detail table(issue) with same name and type.

1. In Server Explorer, explore database.mdf (database file).
2. Right click on Database Diagrams folder→Add New Diagram→Press Yes→Add all table from list (Book, Issue)→close.
3. Hold Primary key field (BookId) from master table (Book), drag over detail table (issue) then drop on same field name(BookId).
4. Press Ok.

Working with "*web.config*" (Web configuration file):

A web configuration file is a special type of file that can add to configure our web application. This file is an XML file therefore, all the tags of files are case sensitive.

Adding connection string into web.config file:

Select Web Site folder→right click→ Add New Item → select "*web.config*".

In ASP.NET ***connection string*** can be written in ***web.config*** file so that connection string can be available to all code area of web pages. The main advantage of using *web.config* is that we are enable to change in connection string from one place (if required).

एएसपी.नेट मे, web.config फ़ाइल का उपयोग सभी वेब पेजेस के कोड एरिया मे कनैक्शन स्ट्रिंग को उपलब्ध करवाने के लिए कर सकते है। web.config का उपयोग करने का मुख्य लाभ यह है कि कनैक्शन स्ट्रिंग मे किसी भी प्रकार के बदलाव को सिर्फ एक ही स्थान पर करते हैं।

Let ***database.mdf*** is a Ms SQLServer database file created using inbuilt tools of Visual studio 2005, then connection string can be obtain as-

Server Explorer→Select *database.mdf* → property window→copy connection String → paste in <ConnectionString> tag of *web.config* file → replace physical path (like c:\website1\App_data) by |DataDirectory| .

Web.config file look like this-

```
<configuration>
      <connectionStrings>
            <add name="constr1"
                  connectionString="Data Source=.\SQLEXPRESS;
                  AttachDbFilename=|DataDirectory|\Database.mdf;
                  Integrated Security=True ;
                  User Instance=true"/>
      </connectionStrings>
      ------
</configuration>
```

Here "*constr1*" is a variable in which we store connection string. We access connection string in code area using "*constr1*" variable.

Getting connection string in code area:

Following namespace must be included on the top of code window to access connection string.

```
using System.Configuration;
```

After then we declare a string variable and retrieve connection string using "constr" then assign into a string variable.

```
string constr= ConfigurationManager.
ConnectionStrings["constr1"].ConnectionString;
```

Now "*constr*" string variable will enable to hold connection string.

Establishing Connection:

We need to connect our .NET application with required database file before data communication. Before any communication with database file, connection must be open. ADO.NET provides a number of *Connection* classes for every type of database file. Therefore according to database file type, we need to choose that type of connection class to establish connection between application and database.

.नेट एप्लिकेशन और डेटाबेस के मध्य डाटा का आदान प्रदान करने के लिए सबसे पहले दोनों के बीच एक कनैक्शन स्थापित करना होगा। इस कार्य के लिए पहले कनैक्शन को ओपेन करना होता है। और एडीओ.नेट मे यह कार्य Connection क्लास की सहायता से करते है।

Working with Connection Class:

Step1: Create a connection string in web.config file.

Step 2: Get Connection String in required Code area.

Step 3: Include namespace: `System.Data.SqlClient`.(for MS SQL-Server)

Step 4: Create Object of Connection Class and pass connection string to this object.

Step 5: Open connection.

Step 6: Perform data access operation.

Step 7: Close active connection.

Above steps can be implemented as.

Namespace:

If we want to connect with *database.mdf* file that created under MS SQL-Server Software then requied Code area must included following namespace.

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

Connection Class:

This namespace support following Connection Class to establishing connection.

SqlConnection

Before establishing connection, we need to create an connection object and passing connection string to the constructor of Connection class like this-

SqlConnection con= new SqlConnection(constr);

here:

con = Connection object

constr= a string variable that contain Connection String

Opening Connection:

When Connection Object is created then by calling *Open()* method of Connection class, we can create an active connection. Like-

con.Open();

When this method executes then connection will be establised with required database file.

Closing Connection:

When all data communication has been performed over this active connection, then connection must be dis mounted so that connection object can be further used with same database file or different database file. Like-

जब किसी एक्टिव कनैक्शन पर सभी डाटा के आदान प्रदान से संभन्धित कार्य सम्पन्न हो चुके हों तब कनैक्शन को विच्षेद कर दिया जाना चाहिए जिससे की उसी कनैक्शन ऑब्जेक्ट का उपयोग किसी उसी डेटाबेस फ़ाइल या अन्य के साथ कर सकें।

con.Close();

Command Class:

This is one component of ADO.NET. With the help of this componenet we can assign SQL command and execute it on database software using active connection.

एडीओ.नेट का यह एक ऐसा कॉम्पोनेंट है जिसकी सहायता किसी ब्गि एसक्यूएल कुएरी को स्टोर कर, दिये गए एक्टिव कनैक्शन पर एक्सेकूट कर सकते हैं।

SqlCommand is command class for MS-SQL Server Database.

Working process with Command Class:(for MS-SQL Server database)

Step1: Include ADO.NET namespace in code area.

Ex: *using System.Data.SqlClient ;*

Step2: Open Database Connection.

Ex: *SqlConnection con = new SqlConnection(constr);*

con.Open();

constr is Connection String.

Step3: Write Required SQL command like INSERT / UPDATE / DELETE in string format.

Ex: string sql;

sql= “INSERT INTO Book (BookName, Price) VALUES('ASP.NET', 500)”;

or

sql= “UPDATE Book SET BookName= 'C#.NET', Price=' 400' WHERE BookID=1”;

or

sql= “DELETE FROM Book WHERE BookID=1”;

Step4: Create Object of Command Class then Pass “Sql Command and Active Connection” to the Constructor of command class.

Ex: SqlCommand cmd = new SqlCommand(sql, con);

sql= SQL Command in string form.

con= Active Connection.

Step5: Execute sql Command using *ExecuteNonQuery()* method of Command class.

Ex: *cmd.ExecuteNonQuery();*

When all above steps are complete then required sql command will be executed on active connection sucessfully.

DataReader Class:

Data Reader class is one of the component of ADO.NET. The purpose of this class is to hold the reference of records that retived after executing SELECT command using *ExecuteReader()* method of *Command* Class.

SqlDataReader is DataReader class for MS-SQL Server Database.

Read() method of DataReader class return **true** value if records found otherwise false. Using index value(0 to columns-1) of Datareader object we can retieved column value of records.

एडीओ.नेट की DataReader क्लास का उपयोग किसी भी SELECT कुएरी के लिए Command क्लास की ExecuteReader() से प्राप्त रेकॉर्ड के रिफ्रेन्स को होल्ड करने के लिए करते है।

SqlDataReader क्लास का उपयोग एमएस-एसक्यूएल सर्वर के लिए करत्ते है। Read() इस क्लास की एक मेथड होती हे जो रेकॉर्ड्स के उपलब्ध होने पर true वैल्यू रिटर्न करती है अन्यथा false। DataReader क्लास के ओब्जेक्ट को array के समान व्यवहार करने पर इंडेक्स वैल्यू रेकॉर्ड्स की कॉलम वैल्यू को प्राप्त कर सकते है।

Working process with DataReader Class:(for MS-SQL Server database)

Step1: Include ADO.NET namespace in code behind.

Ex: using System.Data.SqlClient ;

Step2: Open Database Connection.

Ex: *SqlConnection con = new SqlConnection(constr);*

con.Open();

constr is Connection String.

Step3: Write Required SELECT SQL command in string formate.

Ex: string sql;

Sql= "SELECT * FROM Book WHERE BookID=1";

Step4: Create Object of Command Class then Pass Sql Command and Active Connection to the Constructor of command class.

Ex: SqlCommand cmd = new SqlCommand(sql, con);

sql= SQL Command in string form.

con= Active Connection.

Step5: Create Object reference of DataReader class.

Ex: ***SqlDataReader dr;***

Step6: Execute SELECT command using *ExecuteReader()* method of Command class and assign reference of retrieved records to the Datareader object reference.

Ex: ***dr=cmd.ExecuteReader();***

Step7: Check availability of retirieved records in to DataReader using ***Read()*** method. If found then retrived columns value into destinating controls using idex value.

Ex:

```
if(dr.Read()==true)
{
        TextBox1.Text=dr[0].ToString(); // First Column Value
        TextBox2.Text=dr[1].ToString(); // Second Column Value
        TextBox3.Text=dr[2].ToString(); // Third Column Value
}
```

When all above steps are complete then required SELECT sql command will be executed on active connection sucessfully and records will be shown on destination control.

Data Adapter and Datset Class:

DataAdapter:

DataAdapter class of ADO.NET can be be used to execute any SQL comaanad on given connection and retrieval records can be filled into given Dataset.

एडीओ.नेट की DataAdapter क्लास का उपयोग दिये गए कनैक्शन पर किसी भी एसक्यूएल कमांड को एक्सेकूट करने के लिए कर सकते है एवं प्राप्त रेकॉर्ड्स को दिये गए Dataset मे भर सकते हैं।

SqlDataAdapter is Data Adapter class for MS-SQL Server.

Fill() method of DataAdapter class that can be used to papulate dataset.

Dataset:

Dataset is a class of ADO.NET which is used to store records that retrieved from any source like MS-SQL server, or XML files. When Dataset is papulated by records from any source then records of Dataset can be shown on any Databound control like GridView, DropDownMenu etc.

एडीओ.नेट की Dataset क्लास का उपयोग किसी भी डाटा सोर्स जैसे एमएस-एसक्यूएल सर्वर अथवा XML फ़ाइल से प्राप्त रेकॉर्ड्स को संग्रहीत करने के लिए करते है जिससे की उन्हे आवश्यकतानुसार बाद मे किसी भी डाटा बाउंड कंट्रोल जैसे GridView पर दिखा सकें।

Working process with DataAdapter and Dataset Class:(for MS-SQL Server database)

Step1: Include ADO.NET namespace in code behind.

Ex: using System.Data.SqlClient ;

Step2: Open Database Connection.

Ex:

```
SqlConnection con = new SqlConnection(constr);
con.Open();
```

Here *constr* is Connection String for required database connectivity.

Step3: Write Required SELECT SQL command in string formate.

Ex:

string sql= "SELECT * FROM Book";

Step4: Create Object of Command Class then Pass Sql Command and Active Connection to the Constructor of command class.

Ex:

SqlCommand cmd = new SqlCommand(sql, con);

Here-

sql= SQL Command in string form.

con= Active Connection.

Step5: Create Object of DataAdapter class then pass object of Command Class to its Constructur.

Ex:

SqlDataAdapter da = new SqlDataAdapter (cmd);

Step6: Create Object of Dataset class.

Ex:

Dataset ds = new Dataset();

Step7: Call Fill() method of DataAdapter class and pass object of Dataset to fill retrived records.

Ex:

da.Fill(ds);

Here-

da= DataReader

ds= Dataset

When all above steps are complete then required SELECT sql command will be executed on active connection sucessfully and records will befillup into Dataset.

Data Bound Control and Data Grid: (GridView Control)

Any control of ASP.NET that can be used to show records of any data source like MS-SQL Server called Data Bound Control. To Bind records on Data Bound control, *DataSource* property and *DataBind()* method is used of that control.

Data Grid or GridView Control is one of the most papular and important Data bound control. Using this control we can show records on table form. The Data source of GridView control is Dataset.

Let GridView1 is one Databound control of GridView Control then to bind Records of Dataset we have to apply DataSource property and DataBind() method.

एएसपी.नेट का ऐसा कोई भी कंट्रोल जिसका उपयोग किसी भी डाटा सोर्स जैसे एमएस-एसक्यूएल सर्वर के रेकॉर्ड्स को दिखाने के लिए कर सकें उन्हे डाटा बाउंड कंट्रोल कहते है। डाटा बाउंड कंट्रोल पर रेकॉर्ड्स को बाइंड करने के लिए उस कंट्रोल की DataSource प्रॉपर्टि एवं DataBind() मेथड का उपयोग करते है।

डाटा ग्रिड अर्थात GridView का उपयोग डाटा बाउंड कंट्रोल की श्रेणी मे सबसे अधिक होता है। इस कंट्रोल की सहायता से हम रेकॉर्ड्स को टेबल के रूप मे दिखा सकते हैं। GridView कंट्रोल का सबसे मुख्य डाटा सोर्स Dataset होता है।

मानाकि GridView श्रेणी का कोई एक डाटा बाउंड कंट्रोल GridView1 वेब पगे पर लिया गया हो तब Dataset के रेकॉर्ड्स को इस कंट्रोल पर बाइंड करने के लिए इसी कंट्रोल की DataSource प्रॉपर्टि एवं DataBind() मेथड को निम्न प्रकार से उपयोग करते है।

```
GridView1.DataSource = ds;
GridView1.DataBind();
```

Here

ds= Dataset with Records.

Working process with DataBound Control (DataGrid / GridView)

First add GridView Control on web page using following asp.net code.

```
<asp: GridView ID= "GridView1" runat= "server" />
```

Now apply following steps on page load event of web page.

Step1 to Step7 are same as for DataAdpter and Dataset.

Step8: Set Data source of GridView control that is Dataset.

Ex:

```
GridView1.DataSource=ds;
```

Step9: Bind GridView control so that data can be shown on control.

Ex:

```
GridView1.DataBind();
```

When all above steps are completed then data of Dataset can be bindup to the data bound control like GridView.

ASP.NET using C# Notes

UNIT-IV

Working with XML Data:

- XML(Extensible Markup Language) has the standard format for information on the web.
  वेब पर किसी जानकारी के लिए XML एक स्टैंडर्ड फ़ारमैट है।
- XML files (or streams of data) are self-describing nature that is each value has a label.
  XML फ़ाइल या डाटा की स्ट्रीम self-describing नेचर की होती है जिसकी प्रत्येक वैल्यू के साथ एक लेबल होता है।
- XML is case-sensitive.
  XML केस-सेनसिटिव होती है।
- XML files can be created, read, and revised using ASP.NET 2.0.
  XML फ़ाइल को ASP.NET मे निर्मित, रीड तथा revised किया जा सकता है।

Example for View of XML format data:

XML फ़ारमैट डाटा के व्यू का उदाहरण:

Let we have following data about any book in tabular form as-

मानाकी हमारे पास किसी book से संबन्धित tabular डाटा निम्न रूप मे उपलब्ध है-

Bid	**BookName**	**Price**
1	ASP.NET	600
2	Java	400
3	AI	500

Now the XML format for above table can be represented as-

उपरोक्त टेबल को XML फ़ारमैट मे निम्न प्रकार से प्रदर्शित कर सकते है-

```
<Books>
    <Book>
        <Bid>1</Bid>
        <BookName>ASP.NET</BookName>
        <Price>600</Price>
    </Book>
    <Book>
        <Bid>2</Bid>
        <BookName>Java</BookName>
        <Price>400</Price>
    </Book>
    <Book>
        <Bid>3</Bid>
        <BookName>AI</BookName>
        <Price>500</Price>
    </Book>
```

</Books>

Here

<Books> Represent name of Database

<Book> Represent Each Row

<Bid>,<BookName> and <Price> Represent Field Names that repeats for each row with different values.

Writing Dataset to XML:

Dataset is the collection of tables in ASP.Net. After getting data into dataset we can write all data of dataset to the XML file in XML format.

ASP.NET मे **dataset** *बहुत सारी* **tables** *का समूह होता है।* **dataset** *मे डाटा को प्राप्त करने के बाद उसके सभी डाटा को* **XML** *फ़ाइल मे* **XML** *फ़ारमैट मे राइट किया जा सकता है।*

In ASP.NET to work with XML format first of all we need to include following name space in code behind-

ASP.NET मे ***XML*** फ़ारमैट के साथ कार्य करने के लिए सबसे पहले निम्न ***namespace*** को ***code behind*** मे शामिल करना होता है-

```
using System.Xml;
```

This namespace contains all the necessary classes that capable to work with XML file.

इस namespace मे वे सभी आवश्यक क्लाससेस होती है जो XML फ़ाइल के साथ कार्य करने के लिए उपयोग मे लाई जा सकती है।

If we want to write data of dataset to the XML format file then we call *WriteXml()* method of Dataset object.

यदि हमे dataset के डाटा को XML फ़ारमैट फ़ाइल मे राइट करना हो तब Dataset ऑब्जेक्ट के द्वारा WriteXml() मेथड को कॉल करना होता है।

```
Dataset ds=new Dataset();
ds.WriteXml(Server.MapPath("xmlfile.xml"));
```

Process of Writing Dataset content to XML:

1. First of all we read table data from database and fill to the Dataset
   सबसे पहले database से किसी table के सभी डाटा को रीड कर dataset को fill कर देते है।
2. Write Dataset content to the XML file.
   Dataset कंटैंट को XML फ़ाइल मे राइट कर देते है।

```
using System.XML;
using System.Data.OleDb;
void btnWrite_Click(object sender, EventArgs e)
{
//Step 1:
    OleDbConnection con = new OleDbConnection("provider=microsoft.jet.oledb.4.0; data source=\datadirectory\database.mdb");//connection string
    con.Open();  //Establish connection
    string sql="select * from student";
    OleDbDataAdapter da = new OleDbDataAdapter(sql, con);
    DataSet ds = new DataSet(); // Create new Dataset
```

```
da.Fill(ds); //dataset has fill by content

//Step 2:
ds.WriteXml(Server.MapPath("xmlfile.xml")); //Write Dataset content to XML
Response.Write("Dataset contents has write to the XML");
}
```

When above code will be implement on any event like button click and event occurs at run time then contents of Dataset will be save in XML form to the xmlfile.xml.This file store dataset content in following form:

जब उपरोक्त code बटन के क्लिक इवैंट पर लिखा जाता है तब रन टाइम पर बटन के क्लिक करने पर dataset के सभी कंटैंट XML फ़ारमैट मे xmlfile.xml मे सेव हो जाते है। यह xml फ़ाइल dataset के कंटैंट को निम्न प्रकार से स्टोर करती है।

```
<?xml version="1.0" standalone="yes"?>
<NewDataSet>
  <Table>
    <ID>1</ID>
    <sname>lokesh</sname>
    <age>35</age>
  </Table>
  <Table>
    <ID>2</ID>
    <sname>jahnavi</sname>
    <age>3</age>
  </Table>
</NewDataSet>
```

From above explanation it is clear that we can write dataset content to the XML file.

उपरोक्त वर्णन से स्पस्त है कि हम dataset कंटैंट को XML फ़ारमैट मे राइट कर सकते है।

Reading Dataset with XML:

As we know Dataset is the collection of tables in ASP.Net and XML is special markup language to represent tabular data. In ASP.NET we can read XML format content and fill into dataset using ReadXml() method of Dataset object.

ASP.NET मे ***Dataset table*** का एक समूह होता है और ***table*** डाटा को अन्य रूप मे प्रदर्शित करने के लिए ***XML*** एक विशेष प्रकार की ***markup language*** होती है। ***XML format*** मे लिखे गए कंटैंट को ***ASP.NET*** मे रीड किया जा सकता है और उसे ***ReadXml()*** मेथड कि सहायता से ***Dataset*** ऑब्जेक्ट को ***fill*** कर सकते है।

Process to Read XML content into dataset:

1. Create XML file as the name “xmlfile.xml” in the root directory of web project that has XML format data like-
   Xmlfile.xml नाम कि फ़ाइल को वेब प्रोजेक्ट के रूट डाइरैक्टरि मे बनाकर इसमे निम्न ***XML*** फ़ारमैट ***data*** को स्टोर करते है।

```
<?xml version="1.0" standalone="yes"?>
<Books>
  <Book>
    <Bid>1</Bid>
    <BookName>ASP.NET</BookName>
    <Price>600</Price>
  </Book>
  <Book>
    <Bid>2</Bid>
    <BookName>Java</BookName>
    <Price>400</Price>
  </Book>
  <Book>
    <Bid>3</Bid>
    <BookName>AI</BookName>
    <Price>500</Price>
  </Book>
</Books>
```

2. *Add data grid into web page and at any event like page_load we write following code to read XML file content into Dataset object and bind it to data grid control.*
   वेब पेज पर एक ग्रिड व्यू को add करते है और पेज लोड इवैंट पर XML फ़ाइल से कंटैंट को dataset ऑब्जेक्ट मे रीड कर ग्रिड व्यू मे bind कर देते है।

```
void Page_Load(object sender, EventArgs e)
{
    DataSet ds = new DataSet(); // create new dataset
    ds.ReadXml(Server.MapPath("xmlfile.xml")); //read XML into dataset

    GridView1.DataSource = ds;  //show dataset content to data grid
    GridView1.DataBind();
}
```

When web page executes then XML content via Dataset into data grid will be shown as

जब वेब पेज **execute** होता है तब **XML** कंटैंट **dataset** कि सहायता से ग्रिड व्यू मे नीम प्रकार से प्रदर्शित हो जाता है-

Bid	**BookName**	**Price**
1	ASP.NET	600
2	Java	400
3	AI	500

From above explanation it is clear that we can read the dataset with XML file content.

उपरोक्त वर्णन से स्पस्त है कि हम XML फ़ाइल के कंटैंट को dataset के द्वारा रीड कर सकते है।

Web application Deployment:

It is required to publish (deploy) a Visual Studio web project to a server where others can access the application over the Internet.

यह एक ऐसी आवश्यकता हे जिसके द्वारा visual studio मे बनाए गई वेब एप्लिकेशन को एक ऐसे सर्वर पर पब्लिश या deploy कराते हे जंहा से कोई भी इंटरनेट यूसर एप्लिकेशन का उपयोग कर सके।

It means Web application deployment is the process of installing web application on the customer's Host machine and making that web application available and accessible to all over the world.

इसका अर्थ यह हे की वेब एप्लिकेशन एक ऐसी प्रोसैस हे जिसमे बनाई गई वेब एप्लिकेशन को कस्टमर के होस्ट मशीन पर इन्स्टाल कर पूरे विश्व मे उपयोग करने लायक बनाया जा सके।

Process to deployment of web application:

1. Before deployment of web application we must ensure that application contains everything that is necessary to run application. It may include- वेब एप्लिकेशन को किसी अन्य मशीन पर डेप्लोयमेंट करने से पहले यह सुनिश्चित करना आवश्यक हे की बनाई गई साइट मे वे सभी कुछ उपलब्ध हे जो अन्य मशीन पर कार्य करने के लिए अनिवार्य हो। इसमे निम्न सामाग्री को शामिल कर चेक कर सकते हे।

- **HTML and CSS files:** Your design and structure.
- **ASPX files:** Your main pages.
- **ASPX.VB or ASPX.CS files:** The code-behind files.
- **Database files (.MDB or .MDF):** The back end of the site.
- **Image files (.JPG, .GIF, .PNG):**
- **XML files:** .XML and .XSD files. Etc

2. After then make sure that web application actually compiles and runs.
   उसके बाद यह चेक करे की बनाई गई वेब एप्लिकेशन वास्तव मे compile और रन भी होती हे या नहीं।
   Main menu: Debug→Start debugging (F5).

3. Convert Web application only in executable mode by following-
   वेब एप्लिकेशन को निम्न तरीके द्वारा केवल executable code मे बदलते हे।
   Main menu: Select Build→Publish web site.
   We get a window. Choose target location and press ok button.
   हमे एक विंडो प्राप्त होती हे। जहा से टार्गेट लोकेशन को चुनकर ओके बटन दबा देते हे।

4. Copy web application from target location to an application folder of remote hosting computer (server).
   वेब एप्लिकेशन को टार्गेट लोकेशन से कॉपी कर उसे एक ऐसे folder मे कॉपी कर देते हे जो रिमोट होस्टिंग सर्वर कम्प्युटर मे एप्लिकेशन को deploy करने के लिए बनाया गया हो।

Web Services:

Basic Means: Web Services allow a *consumer* site (local) to obtain information from a *provider* site.

[Web service की सहायता से कोई भी *local consumer site* उसकी *provider* साइट से *information* प्राप्त कर सकती हे। *]*

For example:

Any local web site can display real-time data using web services provided by the Main site(www.Ford.com), but keep the user on the page of the local site.

[मानाकी *www.ford.com* एक मुख्य साइट हे जिसके द्वारा *web services* को किसी भी लोकल वेब साइट मई उपयोग कर सकते हे। किन्तु यूसर लोकल साइट के उसी पेज पर बना रहता हे।*]*

Features of Web Services:

- ASP.NET 2.0 offers a complete web-services solution.
  [ASP.NET2.0 में वेब सर्विस को प्रदान करने वाले सभी टूल उपलब्ध हे।]

- Web services are a method of making information available that could be accessed by any developer's application over the Web.
  [वेब सर्विसेस एक ऐसी मेथड हे जिसके द्वारा वेब पर किसी भी डेवलपर की एप्लिकेशन को इन्फॉर्मेशन उपलब्ध कारवाई जा सकती हे।]

- Web services can form a library of information that could be anything like a mathematical function calculator.
  [वेब सेरवीक इन्फॉर्मेशन की एक लाइब्ररी से मिल्कर बनती हे जिसे हम उदाहरण के लिए एक गणितीय calculation के लिए उपयोग मे आने वाले function के रूप मे मन सकते हे।]

- A web service is not an local web application and does not rendered as web pages, nor as executable files (.exe); It is just like a user interface.
  [वेब सर्विसेस एक लोकल वेब एप्लिकेशन नहीं होती हे तथा किसी वेब पेज की तरह render भी नहीं होती हे और न ही किसी executable file की तरह। यह केवेल एक यूसर इंटरफ़ेस की तरह होती हे।]

- The information contained in the web service is wrapped up as an XML document (in other words, plain text).
  [वेब सर्विसेस मे रखी जाने वाली इन्फॉर्मेशन को XML document के रूप मे अच्छे से सँजोया जाता हे।]

- Web services communicate using open protocols like SOAP.
  [SOAP जैसे ओपेन प्रोटोकॉल की सहायता से वेब सर्विसेस के साथ जुड़ा जा सकता हे।]

- Web services are self-contained and self-describing mechanism.
  [वेब सर्विसेस self contained तथा self describing नेचर की होती हे।]

- HTTP and XML is the basis for Web services.

  [HTTP एवं XML दोनों ही वेब सेर्किसेस का मुख्य आधार हे।]

- Web services can be published, found, and used on the Web.

[वेब सर्विसेस को वेब पर publish, प्राप्त एवं उपयोग किया जा सकता हे।]

- Web services use XML to code and to decode data, and SOAP to transport it (using open protocols).

  [XML का उपयोग वेब सर्विसेस द्वारा code एवं decode करने के लिए किया जाता हे तथा ओपेन प्रोटोकॉल जैसे SOAP का उपयोग उसे transport करने के लिए करते हे।]

Benefits:

- Web Developer can use easily web services and integrate them into web applications.
  [वेब सर्विसेस का उपयोग वेब डेवलपर द्वारा उसकी एप्लिकेशन मे बहुत ही जोड़कर कर आसानी से कर सकता हे।]
- Web services save the time of developer and effort by reducing code duplication.
  [वेब सर्विसेस के ड्यूप्लिकेट कोड़े को कम कर डेवलपर के टाइम को बचा सकते हे।]

Way of using web services:

They can be used in one of two ways.

[वेब सर्विसेस का उपयोग निम्न दो मे से कर सकते हे।]

1. You can create a web service that is exposed to the web, to share with other developers and other applications. Or
   [वेब सर्विस को बनाकर उसे वेब पर डाल कर अन्य डेवलपर द्वारा किसी अन्य एप्लिकेशन मे उपयोग कर लिया जाए। या]
2. you can search for a web service that can be added to your own application. (They are similar to plug-ins in that respect.)
   [हम वेब सर्विसेस को search कर बनाई जा रही एप्लिकेशन मे उपयोग कर ले।]

Components of Web Services:

Everything to do with web services is standardized:

[वेब सर्विसेस मे जो कुछ भी किया जाता हे प्रत्येक का एक standardized होता हे। जैसे-]

- the method of transmission,
  [वेब सर्विस को ट्रांसमीट करने की method।]
- the method used to wrap the web service up,
  [किस तरह वेब सर्विस को bind up किया जाए।]
- the way the web service is defined,
  [और किस तरह वेब सर्विस को परिभाषित किया जाए।]

All have clear W3C standards associated with the technologies involved. And all these standards are based on XML. So they're quick and easy to download, and even easier to use.

[उपरयुक्त सभी कार्य के साथ W3C standards जुड़ा हुआ होता हे। एवं सभी स्टैंडर्ड XML पर आधारित होते हे। जिससे इन्हे आसानी से download कर सरलता से उपयोग कर सकते हे।]

Web Services Description Language(WSDL):

Basic Mean:

WSDL is a language for describing web services and how to access them.

[WSDL, वेब सर्विसेस को तैयार करने तथा उसे उपयोग करने के तरीके को प्रदान करने वाली एक भाषा हे। *]*

General Features:

- WSDL is written in XML.

  WSDL को XML मे लिखा जाता हे।

- WSDL became a W3C Recommendation 26. June 2007
  WSDL का सुझाव W3C के द्वारा 26. June 2007 मे दिया गया।

- WSDL document is just a simple XML document.

  WSDL डॉकयुमेंट केवेल एक XML डॉकयुमेंट जैसा ही हे।

- It contains set of definitions to describe a web service.

  इसमे वेब सर्विसेस को बताने वाली कई सारी परिभाषाओं का समूह होता हे।

The WSDL Document Structure:

A WSDL document describes a web service using a number of elements. The main structure of a WSDL document looks like this:

[WSDL डॉकयुमेंट मे कई सारे elements होते हे जिनकी सहायता से वेब सर्विसेस को बताया जाता हे। WSDL का main structure निम्न प्रकार से दिखाई देता हे।]

```
< definitions>

< types>
data type definitions.....
< /types>

< message>
definition of the data being communicated....
< /message>

< portType>
set of operations.....
< /portType>

< binding>
protocol and data format specification....
< /binding>

< /definitions>
```

A WSDL document can also contain other elements, like extension elements, and a service element.

[एक WSDL डॉकयुमेंट मे कई और भी अन्य elements होते हे जैसे extension एलिमेंट्स एवं सर्विस element आदि।]

SOAP: (Simple Object Access Protocol)

Basic Concepts:

SOAP provides a way to communicate between applications running on different operating systems, with different technologies and programming languages. It is important for application development to allow Internet communication between programs.

[SOAP एक विशेष प्रकार का protocol या तरीका प्रदान करता हे जिसके द्वारा भिन्न भिन्न operating system पर रन हो रही वेब एप्लिकेशन के मध्य communicate करने के लिए विभिन्न technologies एवं languages का उपयोग कर सके। यह ऐसी एप्लिकेशन के लिए बेहतर साबित होता हे जिसमे programs के बीच internet communication होता हो।]

Need of SOAP:

Today's applications communicate using Remote Procedure Calls (RPC) between objects like DCOM and CORBA, but HTTP was not designed for this. RPC represents a compatibility and security problem; firewalls and proxy servers will normally block this kind of traffic.

[वर्तमान समय मे applications विभिन्न objects जैसे DCOM एवं CORBA आदि के मध्य communicate करने के लिए Remote Procedure Calls (RPC) का उपयोग करती हे। किन्तु इस कार्य के लिए HTTP को नहीं बनाया गया हे। RPC के उपयोग मे कई सारी problems आती हे जैसे- compatibility एवं security की समस्या। इसके अलावा firewalls और proxy servers इस प्रकार के traffic को ब्लॉक भी कर देते हे।]

Solution: SOAP was created to accomplish this for better way to communicate between applications over HTTP, because HTTP is supported by all Internet browsers and servers.

[इस कमी को दूर करने के लिए SOAP को तैयार किया गया जिससे की HTTP पर विभिन्न applications के बीच communicate किया जा सके, क्यूंकी HTTP ही सभी इंटरनेट ब्राउज़र एवं सेर्वेर्स द्वारा सपोर्ट किया जाता हे।]

General Features of SOAP :

- SOAP is a communication protocol
- SOAP is for communication between applications.
- SOAP is a format for sending messages।
- SOAP communicates via Internet.
- SOAP is platform independent.
- SOAP is language independent.
- SOAP is based on XML.
- SOAP is simple and extensible।
- SOAP is a W3C recommendation in 24. June 2003.

SOAP Building Blocks:

A SOAP message is an ordinary XML document containing the following elements:

[SOAP मे लिखा गया मैसेज एक ordinary XML डॉकयुमेंट मे रखा जाता हे जिसमे निम्न element होते हैं।]

Skeleton SOAP Message

```
< ?xml version="1.0"?>
< soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

< soap:Header>
...
< /soap:Header>

< soap:Body>
...
<soap:Fault>
...
</soap:Fault>
< /soap:Body>

< /soap:Envelope>
```

Here-

- An Envelope element : that identifies the XML document as a SOAP message.

  [यह XML document की SOAP मैसेज के रूप मे पहचान करता हे।]

- A Header element: that contains header information।

  [इसमे header information होती हे।]

- A Body element: that contains call and response information

  [यह कॉल एवं response information को रखता हे।]

- A Fault element: containing errors and status information

  [यह errors एवं स्टेटस से संबन्धित इन्फॉर्मेशन को रखता हे।]

All the elements above are declared in the default namespace for the SOAP envelope.

[उपरोक्त सभी elements SOAP envelop के default namespace मे declare रहते हे।]

ASP.NET using C# Notes

Unit-5

C# Language

- C# is Microsoft's programming language for .NET development.
- C# was created at Microsoft late in the 1990s and It was first released in its alpha version in the middle of 2000. C#'s chief architect was Anders Hejlsberg. Different version of C# are 1.0, 1.1, 2.0,3.0.
- The Source code of C# converts into 16 bits MSIL code(Microsoft Intermediate Language) and executed by .NET Framework.
- C# is directly related to C, C++, and Java. From C, C# derives its syntax, many of its keywords, and its operators. **From C++,** C# builds upon and improves the object model.
- C# and Java both descended from C and C++ that shares the C/C++ syntax and object model.
- C# support properties, methods, and events.

C# v/s C, C++ and Java

C was invented by Dennis Ritchie in the 1972 based on the Procedure Oriented *programming. Using C,* large programs were difficult to write.

- C++ was invented by Bjarne Stroustrup beginning in 1979 based on Object oriented Model. Using C++, large programs were easy to handle.

C and C++, always compiled machine dependent executable code and not support internet based programs.

C#	**Java**
C# was created at Microsoft late in the 1990s by Anders Hejlsberg.	Java was invented by James Gosling team in 1991 at Sun Microsystems. Initially called Oak.
C# is a structured, object-oriented language with a syntax and philosophy derived from C and C++.	Java is also a structured, object-oriented language with a syntax and philosophy derived from C and C++.
C# achieved portability by translating a program's source code into an intermediate language called *MSIL code(Microsoft Intermediated language).* This MSIL code was then executed by the .Net Framework.	Java achieved portability by translating a program's source code into an intermediate language called *byte code.* This byte code was then executed by the Java Virtual Machine (JVM).
A C# program could run only in an environment where MS.NET framework is available.	A Java program could run in any environment for which a JVM was available.
C# code is neither upwardly nor downwardly compatible with C or C++, its syntax is sufficiently similar.	Java code is also neither upwardly nor downwardly compatible with C or C++, its syntax is sufficiently similar.
C# has successfully portable in the Internet environment along with ASP.NET.	Java has also successfully portable in the Internet environment.
C# includes features that directly support the constituents of components, such as properties, methods, and events.	Java has also supported.
C#'s has ability to work in a secure, mixed-language Environment.	Java work only in one language environment.

ASP.NET using C# Notes

C# and the .NET Framework

C# is a computer language that has a special relationship to its runtime environment, known as the .NET Framework. It has two reasons.

- First, C# was initially designed by Microsoft to create code for the .NET Framework.
- Second, the libraries used by C# are the ones defined by the .NET Framework.

Because of this, it is important to have the .NET Framework for C# programs.

Structure of C# program

C# program has following structure-

```
// Namespace Declaration
using System;
```

```
// Program start class
class WelcomeCSS
{
        // Main begins program execution.
        static void Main()
         {
            // Write to console
            Console.WriteLine("Welcome to the C# Station Tutorial!");
         }
}
```

```
// Other user define class
```

Remark: C# is case-sensitive. The C# program has 4 primary elements:

1. A namespace declaration
2. A class
3. A *Main* method and
4. A program statement.

Compile C# Code: It can be compiled with the following command line:

csc.exe Welcome.cs

This produces a file named *Welcome.exe*, which can then be executed.

Description:

1. *The namespace declaration*: Namespaces contain groups of code that can be called upon by C# programs.
2. *The class declaration*: It contains the data and method definitions. A *class* is one of a few different types of elements to describe objects, such as *structs*, *interfaces* , *delegates*, and *enums.*
3. *The method name*: *Main*, is reserved for the starting point of a program.

Type System:(Variables, Types and Operators)

"Variables" are simply storage locations for data. You can place data into them and retrieve. Data in a variable is controlled through "Types". C# is a "Strongly Typed" language.

The C# data types: Boolean type, Integrals, Floating Point and String

- **Boolean types**: are declared using the keyword, *bool*. They have two values: *true* or *false*.

  Ex: bool content = true;

- **Integrals:** include sbyte, byte, short, ushort, int, uint, long, ulong, and char.
- **Floating Point**: Include float and double types. Floating point types perform operations requiring fractional representations.
- **The string type**: represents a string of characters.

C# Operators: Results are computed by combining variables and operators together into statements. The following table describes the allowable operators, their precedence, and associativity.

Operators with their precedence and Associativity

Category (by precedence)	Operator(s)	Associativity
Primary	x++ x-- new typeof	left
Unary	+ - ! ~ ++x --x	right
Multiplicative	* / %	left
Additive	+ -	left
Shift	<< >>	left
Relational	< > <= >= is as	left
Equality	== !=	right
Logical AND	&	left
Logical XOR	^	left
Logical OR	|	left
Conditional AND	&&	left
Conditional OR	||	left
Null Coalescing	??	left
Ternary	?:	right
Assignment	= *= /= %= += -= <<= >>= &= ^= |= =>	right

Left associativity means that operations are evaluated from left to right. Right associativity mean all operations occur from right to left, such as assignment operators where everything to the right is evaluated before the result is placed into the variable on the left.

The Array Type: Array can be thought of as a container that has a list of storage locations for a specified type. When declaring an Array, specify the type, name, dimensions, and size.

Array Operations:

```
using System;
class Array
{
```

```
public static void Main()
{
string[] myStrings = new string[3];
myStrings[0] = "Joe";
myStrings[1] = "Matt";
myStrings[2] = "Robert";
……
}
}
```

Flow Controls: Control Statements - Selection

- The *if* statements.
- The *switch* statement with *break*.
- Loop: while, do, for, foreach

The *if* Statement:

An *if* statement depends on a given condition. When the condition evaluates *true*, a block of code for that true condition will execute. You have the option to use optional *else* statement with *if* statement

Ex:

```
using System;
class UseIfElse
{
public static void Main()
{
   // Single Decision and Action with braces
   if (condition)
  {
          Logic Code1
  }
  else
 {
          Logic Code2
  }
}
}
```

When condition evaluates to *true*, the statement in the *if* block are executed; when *false*, the statements in the *else* block are executed.

The *switch* Statement: *switch* statement, executes a set of logic depending on the value of a given parameter.

Ex:

```
using System;
class SwitchSelect
{
public static void Main()
{
switch (Expression)
{
case 1:
        Code1;
        break;

case 2:
```

```
        Code2;
        break;
……
default:
        Optional code;
        break;
}
}
}
```

The *switch* block follows one or more choices. When the result of the *switch* expression matches one of these choices, statements of the matching choice are executed. After then jumps out from switch block using *break*. If none of the other choices match, then the *default* choice is taken and its statements are executed, although the use of *default* label is optional.

The *while* Loop

A *while* loop will check a condition and then continues to execute a block of code as long as the condition evaluates to a boolean value of *true*.

Syntax:

```
while (Condition)
{
        statements
}
```

Once the statements have executed, control returns to the beginning of the *while* loop to check the boolean expression again.When the boolean expression evaluates to *false*, the *while* loop statements are skipped.

The *do* Loop

A *do* loop is similar to the *while* loop, except that it checks its condition at the end of the loop. This means that the *do* loop is guaranteed to execute at least one time. On the other hand, a *while* loop evaluates its boolean expression at the beginning.

Syntax:

```
do
{
        Statements
}while (Condition);
```

The *for* Loop

A *for* loops are appropriate when you know exactly how many times you want to perform the statements within the loop.

Syntax:

The contents within the *for* loop parentheses hold three sections separated by semicolons

```
for (initializer ; condition ; update initialize)
{
        Statements
}
```

Like

```
for (int i=0; i < 20; i++)
{
```

```
Console.Write("{0} ", i);
}
```

Initializer: Evaluated only once during the lifetime of the *for* loop.

Condition: Once the initializer has been evaluated, the *for* loop gives control to condition. When the condition evaluates to *true*, the statements within the curly braces of the *for* loop are executed.

Update Initializer: After executing *for* loop statements, control moves to the top of loop and executes updater, after then control transfer to condition part.

The *foreach* Loop

A *foreach* loop is used to iterate through the items in a list. It operates on arrays or collections such as ArrayList, which can be found in the System.Collections namespace.

Syntax:

foreach (<type> <iteration variable> in <list>)

{

<statements>

}

Introduction to Classes

Classes are declared by using the keyword *class* followed by the *class* name and a set of *class* members surrounded by curly braces.

Every *class* has a constructor, which is called automatically any time an instance of a *class* is created. Constructors do not have return values and always have the same name as the *class*.

सी# मे क्लास को class कीवर्ड के द्वारा declared करते है। इसके साथ क्लास का नाम और क्लास के मेम्बर { कर्ली ब्रेसेस} मे परिभाषित किए जाते हे।

प्रत्येक क्लास का एक constructor भी होता हे जो क्लास के प्रत्येक instance पर स्वत: कॉल होता है तथा यह कोई वैल्यू को रिटर्न नहीं करता। सी# मे constructor और क्लास का नाम एक ही रखते हे।

Example C# Classes: Classes.cs

```
// Namespace Declaration
using System;
class OutputClass
{
 string myString;
 // Constructor
 public OutputClass(string inputString)
 {
  myString = inputString;
 }

 // Instance Method
 public void printString()
{
 Console.WriteLine("{0}", myString);
 }

// Destructor
~OutputClass()
{
// Some resource cleanup routines
```

```
}
}

// Program start class
class ExampleClass
{
// Main begins program execution.
public static void Main()
{
// Instance of OutputClass
OutputClass outCl = new OutputClass("This is printed by the output class.");
// Call Output class' method
outCl.printString();
}
}
```

OutputClass, has a constructor, instance method, and a destructor. It also had a field named *myString*.

इस उदाहरण मे OutputClass मे एक फील्ड, constructor, एक instance की मेथड और एक destructor है।

Interfaces

An *interface* looks like a class, but has no implementation. Interface contains only declarations of *events*, *methods* and *properties*. *Interfaces* only inherited by *classes*, which must provide an implementation for each interface member declared.

एक इंटरफ़ेस क्लास के समान ही होता है, किन्तु इसमे कोई implementation नहीं होता है। इंटरफ़ेस मे केवल events, मेथड एवं properties का declaration ही होता है definition नहीं। इंटरफ़ेस केवल क्लाससेस के द्वारा ही inherit होते है और उसमे प्रत्येक इंटरफ़ेस मेम्बर का implementation होता है।

Defining an Interface:

```
interface MyInterface
{
        void MethodToImplement();
}
```

This method does not have an implementation because the *interface* only specifies the methods that must implement in class.

इस मेथड ka इम्प्लिमैनटेशन नहीं होता है क्यूंकी इंटरफ़ेस केवल मेथड को specify करता है जिसे क्लास मे इम्प्लेमेंट करना आवश्यक होता है।

Using an Interface:

```
class UseInterface : MyInterface
{
    public void MethodToImplement()
    {
        Console.WriteLine("MethodToImplement() called.");
    }
   static void Main()
   {
         UseInterface oi = new UseInterface();
         oi.MethodToImplement();
   }}
```

The UseInterface *class* implements the *MyInterface interface*. Indicating that a *class* inherits an *interface* is the same as inheriting a *class*.

यंहा UseInterface क्लास मे MyInterface नाम की इंटरफ़ेस को इम्प्लेमेंट किया गया है।

Interfaces may also inherit other interfaces.

Boxing and Unboxing

Boxing means the conversion of a value type(int / long / float) on the stack to a **object** type on the heap.

Unboxing means the conversion from an **object** type back to a value type.

बॉक्सिंग से अभिप्राय यह है कि stack मे स्टोर किसी वैल्यू को heap मे किसी object टाइप मे रूपांतरित कर दिया जावे।

उन्बोक्सिंग से अभिप्राय यह है कि object टाइप से पुन: वैल्यू टाइप मे रूपांतरित कर दे।

Boxing occurs automatically whereas u*nboxing* using an explicit cast from the **object** reference to its corresponding value type.

बॉक्सिंग स्वत: होती है जबकि उन्बोक्सिंग के लिए object reference को संगत वैल्यू टाइप मे बाह्य कास्ट करना होता है।

```
// A simple boxing/unboxing example.
using System;
class BoxingUnboxing
 {

        static void Main()
         {
                int x=10;
                object obj=x;   // Boxing
                Console.WriteLine(obj);    // obj=10

                x = (int)obj; // Unboxing
                Console.WriteLine(x);       //x=10

        }
}
```

Value in **x** is boxed simply by assigning it to **obj**, whereas integer value in **obj** is retrieved by casting **obj** to **int**.

X मे स्टोर वैल्यू सामान्य तरीके से object मे assign हो जाती हे(बॉक्सिंग) जबकि object से वैल्यू प्राप्त करने के लिए उसे int से कास्ट किया गया हे।(उन्बोक्सिंग)

Delegates

A delegate is an object that can reference a method just like a function pointer used in C or C++. Therefore, when you create a delegate, you are creating an object that can hold a reference to a method. Furthermore, the method can be called through this reference.

In other words, a delegate can invoke the method to which it refers.

Delegate एक ऐसा ऑब्जेक्ट होता है जिसकी सहायता से किसी भी method के reference को होल्ड कर सकते है जैसे c/c++ में function pointer का उपयोग किया जाता है। इसलिए जब किसी delegate को निर्मित करना हो तब C# मे सबसे पहले एक मेथड के रिफ्रेन्स को होल्ड करने वाला एक ऑब्जेक्ट बनाना होगा। ऐसे ऑब्जेक्ट की सहायता से भी होल्ड मेथड को कॉल किया जा सकता है। ऐसे ऑब्जेक्ट को delegate कहा जाता है।

Creating Delegate:

delegate *ret-type delegateName*(*parameter-list*); //delegate is a keyword

Example: **delegate int Dx(int,int);**

Here Dx is such delegate that can hold reference of any method whose return type is int and has two arguments of int and int type.

यंहा Dx एक ऐसा delegate है जो ऐसे किसी भी मेथड के रिफ्रेन्स को होल्ड कर सकता है जिसका रिटर्न टाइप int है तथा उसके दो अरगुमनेंट int, int हों।

Holding method reference to Delegates:

delegateName objectName=new delegateName(Clsobject.MethodName);

Example: Dx odx=new Dx(oa.getdata);

Here odx is an object of Dx delegate type that hold reference of a method of any object oa of class A.

यंहा Dx delegate टाइप का odx ऑब्जेक्ट है जिसमे किसी class A के ऑब्जेक्ट oa की मेथड getdata का रिफ्रेन्स होल्ड करवाया गया है।

Calling Method using delegate:

delegateObject(list of arguments);

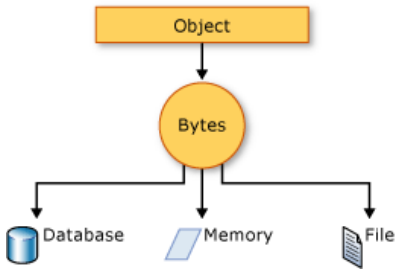
Example: int x=odx(10,20);

Here odx(10,20) is similar to getdata(10,20);

Serialization

Definition: Serialization is the process of converting an object into a stream of bytes in order to store the object or transmit it to memory, a database, or a file. The reverse process is called deserialization.

परिभाषा: Serialization एक ऐसी प्रोसैस हे जिसमे किसी ऑब्जेक्ट को बाइट्स की स्ट्रीम मे परिवर्तित किया जाता है जिससे कि ऑब्जेक्ट को मेमोरी या डेटाबेस या फ़ाइल के रूप मे स्टोर किया जा सके। इसकी विपरीत प्रोसैस को deserialization कहते हैं।



The object is serialized to a stream, which carries not just the data, but information about the object's type, such as its version, culture, and assembly name. From that stream, it can be stored in a database, a file, or memory.

जब किसी ऑब्जेक्ट को स्ट्रीम मे serialized किया जाता है तब ऑब्जेक्ट के डाटा के अलावा उससे संबन्धित जानकारी जैसे version, कल्चर तथा एसम्ब्ली name भी स्टोर होती है। इस स्ट्रीम के द्वारा ही ऑब्जेक्ट को किसी भी डेटाबेस, फ़ाइल या मेमोरी मे स्टोर कर सकते हे।

Uses: Its main purpose is to save the state of an object in order to be able to recreate it when needed. Through serialization, a developer can perform actions like sending the object to a remote application by means of a Web Service, passing an object from one domain to another, etc.

उपयोग: इसका मुख्य उद्देश्य ऑब्जेक्ट कि स्टेट को सुरक्षित करके रखना होता है जिससे कि आवश्यकता होने पर पुनः उसे निर्मित किया जा सके। serialization कि सहायता से कोई भी डेवलपर कुछ कार्यों को अच्छे से संपादित कर सकता है जैसे- वेब सर्विसेस के लिए रेमोट एप्लिकेशन को ऑब्जेक्ट सेंद करना, एक डोमैन से अन्य डोमैन पर ऑब्जेक्ट पास करना आदि।

Making an Object Serializable : To serialize an object, we need an object to be serialized, a stream to contain the serialized object, and *System.Runtime.Serialization* namespace contains the classes necessary for serializing and deserializing objects.

किसी ऑब्जेक्ट को serialize करने के लिए हमे एक ऑब्जेक्ट चाहिए जिसे serialize करना है, एक स्ट्रीम तथा System.Runtime.Serialization namespace जिसमे ऑब्जेक्ट को serialize तथा deserialize करने कि classes होती है।

Reflection

Reflection is the feature that enables you to obtain information about a type. Using this information, you can construct and use objects at runtime. This feature is very powerful because it lets a program add functionality dynamically, during execution.

C# का reflection एक ऐसा गुण है जिसकी सहायता से किसी भी class type की जानकारी प्राप्त की जा सकती है। इस जानकारी के द्वारा runtime पर ही किसी भी ऑब्जेक्ट को बनाया और उपयोग किया जा सकता है। यह गुण सबसे अधिक शक्तिशाली है क्योंकि इसके द्वारा प्रोग्राम के execution के समय ही उसमे और भी नए functions जोड़े जा सकते हैं।

System.Reflection namespace must be included before using classes of reflection.

Reflection से संभन्धित आवश्यक classes का उपयोग करने से पहले *System.Reflection* namespace शामिल करना आवश्यक है।

Ex:

```
// Analyze methods using reflection.
using System;
using System.Reflection;
class MyClass
{
        Class-members
}

class ReflectDemo
```

```
{
        static void Main()
        {
                MyClass obj=New MyClass();

                Type t = typeof(obj); // get a Type of  object obj
                Console.WriteLine("Class Type of obj is:  " + t.Name);
        }
}
```

Output: Class Type of obj is: MyClass

Here **typeof** returns a **Type** object that represents the specified type, which in this case is **MyClass**.