Salvi College

Written By

Prof. Sonu Raj

TYBsc-IT

Semester-V

Subject-Advanced Web Programming

Syllabus

Unit-I	Introducing .NET C# Language Types, Objects, and Namespaces
Unit-II	Web Form Fundamentals Form Controls
Unit-III	Error Handling, Logging, and Tracing State Management Styles, Themes, and Master Pages
Unit-IV	ADO.NET Fundamentals Data Binding The Data Controls
Unit-V	XML Security Fundamentals ASP.NET AJAX

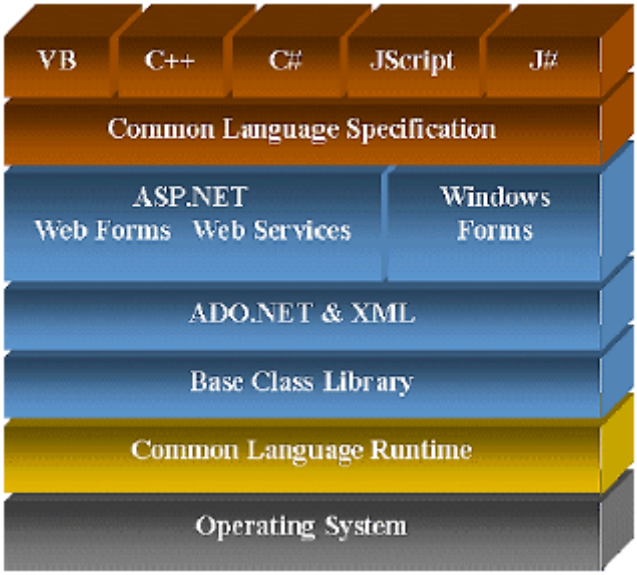
Index

Unit-I

Topics:

.Net Framework | C# Language |Types, Objects, and Namespaces

Q1. What is .NET Framework? Explain its architecture with help of diagram.



.NET Framework is a platform created by Microsoft for building, deploying, and running applications and services that use .NET technologies, such as desktop applications and Web services.

- It is a platform for application developers.
- It is a Framework that supports Multiple Language and Cross language integration.
- It has IDE (Integrated Development Environment).
- Framework is a set of utilities or can say building blocks of our application system.
- .NET Framework provides interoperability between languages i.e. Common Type System (CTS).
- .NET Framework also includes the .NET Common Language Runtime (CLR), which responsible for maintaining the execution of all applications developed using the .NET library.
- The .NET Framework consists primarily of a gigantic library of code.

Components of the .Net Framework:

User Interface and Program:

We can create various type of application using .net framework such as
Console Application
Windows Application
Web application

Base class Library:

- Base class library is one of component of .Net Framework. It supplies a library of base classes that we can use to implement applications quickly.

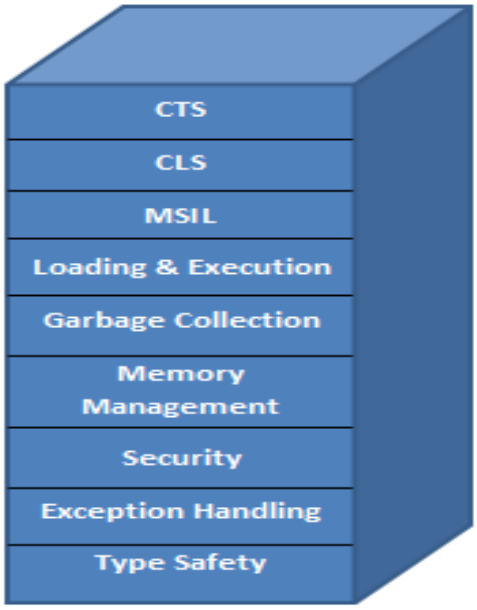
CLR (Common Language Runtime)

- Common Language Runtime (CLR) is the programming (Virtual Machine component) that manages the execution of programs written in any language that uses the .NET Framework, for example C#, VB.Net, F# and so on.
- We can say that it is heart and soul of .Net Framework or backbone.

Q2. Explain CLR in detail.

- Common Language Runtime (CLR) is the programming (Virtual Machine component) that manages the execution of programs written in any language that uses the .NET Framework, for example C#, VB.Net, F# and so on.

- We can say that it is heart and soul of .Net Framework or backbone.
- Programmers write code in any language, including VB.Net, C# and F# then they compile their programs into an intermediate form of code called CLI in a portable execution file (PE) that can be managed and used by the CLR and then the CLR converts it into machine code to be will executed by the processor.
- The information about the environment, programming language, its version and what class libraries will be used for this code are stored in the form of metadata with the compiler that tells the CLR how to handle this code.
- The CLR allows an instance of a class written in one language to call a method of the class written in another language.



Components of the CLR:

CTS

Common Type System (CTS) describes a set of types that can be used in different .Net languages in common. That is, the Common Type System (CTS) ensure that objects written in different .Net languages can interact with each other. For Communicating between programs written in any .NET complaint language, the types must be compatible on the basic level.
These types can be Value Types or Reference Types. The Value Types are passed by values and stored in the stack. The Reference Types are passed by references and stored in the heap.

CLS

CLS stands for Common Language Specification and it is a subset of CTS. It defines a set of rules and restrictions that every language must follow which runs under .NET framework. The languages which follow these set of rules are said to be CLS Compliant. In simple words, CLS enables cross-language integration or Interoperability.

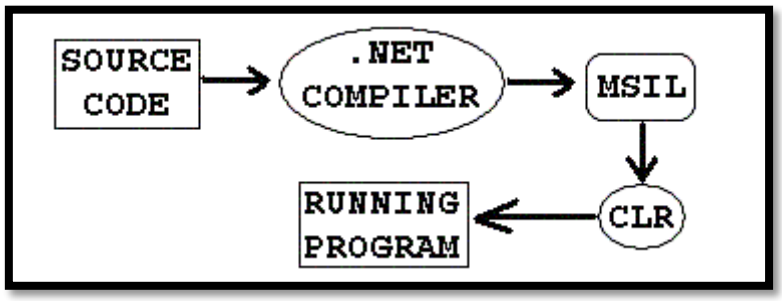
For Example

if we take C-Sharp and VB.net in C# each statement must have to end with a semicolon it is also called a statement Terminator but in VB.NET each statement should not end with a semicolon (;).
So, these syntax rules which we have to follow from language to language differ but CLR can understand all the language Syntax because in.NET each language is converted into MSIL code after compilation and the MSIL code is language specification of CLR.

MSIL

It is language independent code. When you compile code that uses the .NET Framework library, we don't immediately create operating system - specific native code.

Instead, we compile our code into Microsoft Intermediate Language (MSIL) code. The MSIL code is not specific to any operating system or to any language.



Advantages -

- MSIL provide language interoperability as code in any .net language is compiled on MSIL
- Same performance in all .net languages
- support for different runtime environments
- JIT compiler in CLR converts MSIL code into native machine code which is executed by OS

Functions of the CLR

- Garbage Collector
- Exception handling
- Type safety
- Memory management (using the Garbage Collector)
- Security
- Improved performance

Q3. What is namespace? How to create namespace and alias?

A namespace is designed for providing a way to keep one set of names separate from another. The class names declared in one namespace does not conflict with the same class names declared in another.

Defining a Namespace

A namespace definition begins with the keyword namespace followed by the namespace name as follows:

```
namespace namespace_name
{
  // code declarations
}
```

Example:

```
using System;
namespace first_space
{
class namespace_cl
  {
public void func()
    {
Console.WriteLine("Inside first_space");
    }
```

```
  }
}

namespace second_space
{
class namespace_cl
  {
public void func()
    {
Console.WriteLine("Inside second_space");
    }
  }
}

class TestClass
{
static void Main(string[] args)
  {
first_space.namespace_cl fc = new first_space.namespace_cl();
second_space.namespace_cl sc = new second_space.namespace_cl();
fc.func();
sc.func();
Console.ReadKey();
  }
}
```

The using Keyword

The using keyword states that the program is using the names in the given namespace. For example, we are using the System namespace in our programs. The class Console is defined there. We just write:

```
Console.WriteLine ("Hello there");
```

We could have written the fully qualified name as:

```
System.Console.WriteLine("Hello there");
```

Alias of Namespace:

```
using A=System.Console;

class Test
{
static void Main()
  {
A.Write("Craetion of Alias");
A.ReadKey();
  }
}
```

Q4. Explain different types of data types in C#.

OR

Explain primitive and non-primitive data types in C#

The data types in C# are categorized into the following types:

- **Value types**
- **Reference types**
- **Pointer types**

Value Type

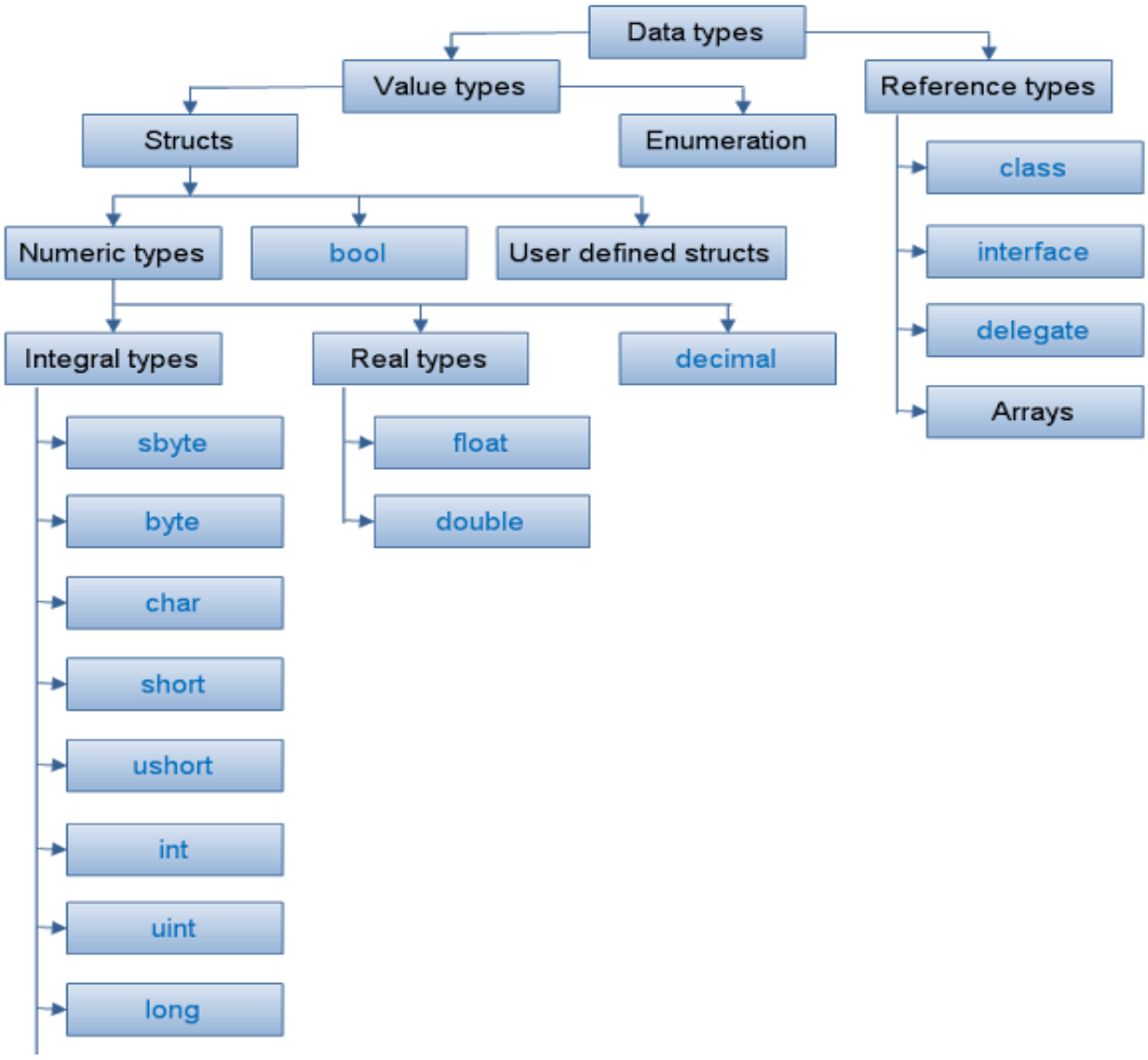
Value type variables can be assigned a value directly. They are derived from the class **System.ValueType.**

The value types directly contain data. Some examples are int, char, and float, which stores numbers, alphabets, and floating-point numbers, respectively. When you declare an int type, the system allocates memory to store the value.

Reference Type

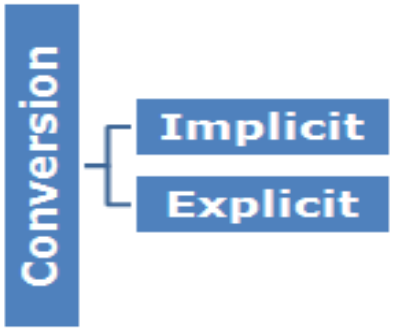
The reference types do not contain the actual data stored in a variable, but they contain a reference to the variables.

In other words, they refer to a memory location. Using multiple variables, the reference types can refer to a memory location. If the data in the memory location is changed by one of the variables, the other variable automatically reflects this change in value. Example of built-in reference types are: object, dynamic, and string.



Q5. Explain the implicit and explicit conversion of data types with examples.

- The process of converting from one data type to another data type is called conversion.
- Conversion can be 2 types as follows:



- When one type of data is assigned to another type of variable, an implicit type conversion will take place automatically if
  - The two types are compatible.
  - The destination type has a range that is greater than the source type.
- When these two conditions are met, a widening conversion takes place. For example, the int type is always large enough to hold all valid byte values, and both int and byte are compatible integer types, so an implicit conversion can be applied.
- For widening conversions, the numeric types, including integer and floating-point types, are compatible with each other.
- For example, the following program is perfectly valid since long to double is a widening conversion that is automatically performed.

```
using System;
class LtoD {
static void Main()
{
long L;
double D;
L = 100123285L;
D = L;
Console.WriteLine("L and D: " + L + " " + D);
}
}
```

Casting Incompatible Types

Although the implicit type conversions are helpful, they will not fulfil all programming needs because they apply only to widening conversions between compatible types. For all other cases we must employ a cast. A cast is an instruction to the compiler to convert the outcome of an expression into a specified type. Thus, it requests an explicit type conversion.

A cast has this general form:

(target-type) expression

Here, target-type specifies the desired type to convert the specified expression to. For

Example

double x, y;
if you want the type of the expression x/y to be int, you can write

(int) (x / y)

Here, even though x and y are of type double, the cast converts the outcome of the expression to int. The parentheses surrounding x / y are necessary. Otherwise, the cast to Int would apply only to the x and not to the outcome of the division.

The cast is necessary here because there is no implicit conversion from double to int. When a cast involves a narrowing conversion, information might be lost.

Q6. Explain Boxing and Unboxing with reference to value type and reference type.

Boxing:
Any type, value or reference can be assigned to an object without explicit conversion. When a compiler fined a value where it needs a reference type, it creates an object 'box' into which it places the value of the value type.

Example: -

int m = 10;
object om = m;
When executed, this code creates a temporary reference _type 'box' for the object on heap. We can also use a C-style cast for boxing.

int m = 10;
object om = (object) m;

Note that the boxing operation creates a copy of the value of the m integer to the object om. Both the variables exist but the value of om resides on the heap. This means that the values are independent of each other.

Example

```
int m =10;
object om = m;
m = 20;
Console.WriteLine(m); // m= 20
Console .WriteLine(om); //om=10
```

Unboxing:

Unboxing is the process of converting the object type back to the value type. Remember that a variable can be unboxed only if it has been previously boxed. In contrast to boxing, unboxing is an explicit operation.

Example:-

```
int m = 100;
object om = m; //boxing
int n = (int) om; //unboxing
```

When performing unboxing, C# checks the value type we request is actually stored in the object under conversion. Only if it is, the value is unboxed.

Q7. List and explain the comparison and logical operators in C#.

Comparison operators:
It is also called as relational operator. They are used to compare at least two operands.
Below is the list of relational operators as follows:

==	Checks if two values are equal to each other.
!=	Checks if two values are not equal
<	Checks if the first value is less than the second.
>	Checks if the first value is greater than the second.
<=	Checks if the first value is less than or equal to the second.
>=	Checks if the first value is greater than or equal to the second.

Logical operators

Logical operators are used to combine two or more condition. Below is list of logical operators used in C#.

&	Returns True when both expressions result in a True value
|	Returns True if at least one expression results in a True value.
!	Reverses the outcome of an expression.
&&	Enables you to short-circuit your logical And condition checks.
||	Enables you to short-circuit your logical Or condition checks.

Q8. Explain flow control, explain what is break and continue statement

The statements inside your source files are generally executed from top to bottom, in the order that they appear. Control flow statements, however, break up the flow of execution by employing decision making, looping, and branching, enabling your program to conditionally execute particular blocks of code.

Break Statement

The break statement is used to exit from loops and switch statements. The break statement terminates the execution control from a loop or a switch statement to the next statement after the loop or switch statement.

The general format is:

Example:-Write a program to generate the following triangle

1
1 2
1 2 3

```
using System;
class demo
{
public static void Main()
{
for(int i=1;i<=5;i++)
{
for(int j=1;j<=5;j++)
{
Console.Write(" "+j);
if(j==i)
{
break;
 }
```

```
}
Console.WriteLine(" ");
}
}
}
```

Continue Statement

The continue statement causes early iteration of a loop. It skips the remaining statements after the continue statement and moves the execution control to the next iteration of the loop.

The general format is:

```
using System;
class demo
{
public static void Main()
{
for(int i=1;i<=25;i++)
{
if(i%2!=0)
{
continue;
}
else
Console.Write(" "+i);
}
}
}
```

Use the break keyword to stop the loop and the continue keyword to skip one loop cycle and continue to the next.

```
using System;

public class Program
{
public static void Main()
{
Console.WriteLine("Demonstrating the use of break.\n");
for (int x = 1; x < 10; x++)
{
if (x == 5)
break;
Console.WriteLine("Number " + x);
}
Console.WriteLine("\nDemonstrating the use of continue.\n");
for (int x = 1; x < 7; x++)
{
if (x == 5)
continue;
Console.WriteLine("Number " + x);
}
}
```

```
}
```

Demonstrating the use of break.

Number 1
Number 2
Number 3
Number 4
Demonstrating the use of continue.
Number 1

Number 2
Number 3
Number 4
Number 6

Q9. Give general form of switch statement. Explain with example.

- The switch statement is similar to if statement in that it executes code conditionally based on the value of a test. However, switch enables us to test for multiple values of a test variable in one go, rather than just a single condition.
- This test is limited to discrete values, rather than clauses such as ‘‘greater than X,’’ so its use is slightly different; but it can be a powerful technique.
- The basic structure of a switch statement is as follows:

```
switch (<testVar>)
{
case<comparisonVal1>:
<code to execute if <testVar> == <comparisonVal1>>
break;
case<comparisonVal2>:
<code to execute if <testVar> == <comparisonVal2>>
break;
. . .
case<comparisonValN>:
<code to execute if <testVar> == <comparisonValN>>
break;
default:
<code to execute if <testVar> != comparisonVals>
break;
}
```

Example:

```
class SwitchEg
{
static void Main(string[] args)
{
const string myName = "karli";
const string sexyName = "angelina";
const string sillyName = "ploppy";
string name;
Console.WriteLine("What is your name?");
name = Console.ReadLine();
switch (name.ToLower())
```

```
{
case myName:
Console.WriteLine("You have the same name as me!");
break;
case sexyName:

Console.WriteLine("My, what a sexy name you have!");
break;
case sillyName:
Console.WriteLine("That's a very silly name.");
break;
}
Console.WriteLine("Hello {0}!", name);

Console.ReadKey();
}
}
```

Q10. Give syntax of foreach loop. Explain with example.

A foreach loop mostly used with array and collection such ArrayList. A foreach loop enables us to address each element in an array using this simple syntax:

```
foreach (<baseType><name> in <array>)
{
// can use <name> for each element
}
```

This loop will cycle through each element, placing it in the variable <name> in turn, without danger of accessing illegal elements. We don't have to worry about how many elements are in the array, and we can be sure that we get to use each one in the loop.

The main difference between using this method and a standard for loop is that foreach gives us read-only access to the array contents, so we can't change the values of any of the elements.

Example

```
using  System;
class Abc
static void Main(string[] args)
{
string[] friendNames = { "Robert Barwell", "Mike Parry", "Jeremy Beacock" };
Console.WriteLine("Here are {0} of my friends:", friendNames.Length);
foreach (string friendName in friendNames)
{
Console.WriteLine(friendName);
}
Console.ReadKey();
}
```

Q11. Difference between for and foreach loop

Afor loop executes a statement or a block of statements repeatedly until a specified expression evaluates to false. There is need to specify the loop bounds (minimum or maximum).

Example:

```
int j = 0;
```

```
for (int i = 1; i <= 5; i++)
{
j= j + i ;
}
```

The foreach statement repeats a group of embedded statements for each element in an array or an object collection. You do not need to specify the loop bounds minimum or maximum.

Example:

```
int j = 0;
int[] tempArr = new int[] { 0, 1, 2, 3, 5, 8, 13 };
foreach (int i in tempArr )
{
j = j + i ;
}
```

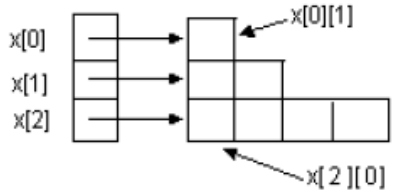
Q12. Explain variable sized array with suitable example.

OR

Explain jagged array with example.

Jagged array is also called Array of Array or variable sized array. In a jagged array, each row may have different number of columns.

It is equivalent to an array of variable sized one-dimensional arrays. The Length property is used to get the length of each one-dimensional array.

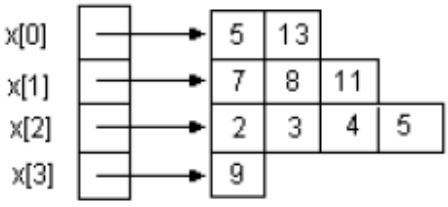


Here the first row is a one-dimensional array of one column, second row is a one-dimensional array of two columns and third row is a one-dimensional array of four columns.

A jagged array can be declared and initialized as follows:

```
datatype[][] arrayname=new datatype[rowsize][];
arrayname[0]=new datatype[]{val1,val2,val3, …};
arrayname[1]=new datatype[]{val1,val2,val3, …};
arrayname[2]=new datatype[]{val1,val2,val3, …};
```

Example




```
class numadd
{
public static void Main()
{
int[][] x=new int[4][];
x[0]=new int[2]{5,13};
```

```
x[1]=new int[3]{7,8,11};
x[2]=new int[4]{2,3,4,5};
x[3]=new int[1]{9};
for(int i=0;i<4;i++)
{
for(int j=0;j<x[i].Length;j++)
{
Console.Write(x[i][j]+"\t");
}
Console.Write("\n");

}

}
```

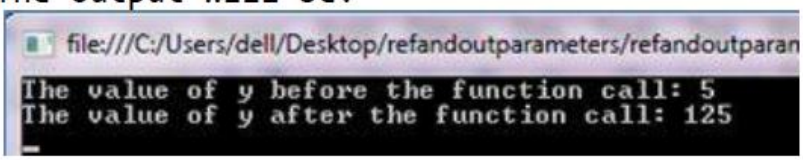
Q13. Explain ref parameter and out parameter

Reference Parameter:

It is used as a call by reference in C#. It is used in both places; when we declare a method and when we call the method. In this example we create a function (cube) and pass a ref parameter in it, now we look at the value before we call the function and after we call the function.

```
classAbc
{
public void cube(ref int x)
{
x= x * x * x;
}
}

class Program
{
static void Main(string[] args)
{ T
ryRef tr = new TryRef();
int y = 5;
Console.WriteLine("The value of y before the function call: " + y);
tr.cube(ref y);
Console.WriteLine("The value of y after the function call: " + y); Console.ReadLine();
}
}
```

The output will be:



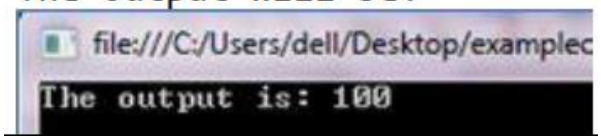
Output Parameter:

Sometimes we do not want to give an initial value to the parameter; in this case we use the out parameter. The declaration of the out parameter is the same as the ref parameter. But it is used to pass the value out of a method. Now we look at the

```
classAbc
```

```
{
public int mul(int a, out int b)
{
b = a * a;
return b;
}
}
class Program
{
static void Main(string[] args)
{
tryout to = new tryout();
int x,y;
x = to.mul(10, out y);
Console.WriteLine("The output is: "+x);

Console.ReadLine();
}
}
```

The Output will be:



Q14. Is Multiple Main () allowed in C#? Justify

We can use more than on Main Method in C# program, But there will only one Main Method which will act as entry point for the program.

```
using System;
class A
{
static void Main(string[] args)
{
Console.WriteLine("I am from Class A");
Console.ReadLine();
}
}
class B
{
static void Main(string[] args)
{
Console.WriteLine("I am from Class B");
Console.ReadLine();
}
}
```

Try to execute Program as given below:

csc filename.cs /main:classname

As in given program there are two classes A and B in which each containing A Main Method. We may write as.

csc filename.cs /main:A [for Class A Main Execution] or,
csc filename.cs /main:B [for Class B Main Execution]

Q15. Difference between Structures and Classes.

Structure	Class
It is value type	It is reference type.
Structure stored in stack memory.	Class stored in heap memory.
Does not consist of parametrized constructor.	Consist of parametrized constructor.
Does not allow inheritance.	Allow inheritance.
Does not consist of destructor.	Consist of destructor.
Structure created with help of struct keyword.	Classes created with the help of class keyword.

Example:-
struct Book
{
}
	Example:-
	class Abc
	{
	}

Q16. Need of enumerator data type in C#

- An enum is a value type with a set of related named constants often referred to as an enumerator list.
- The enum keyword is used to declare an enumeration. It is a primitive data type, which is user defined.
- Enums type can be integer (float, int, byte, double etc.). But if you used beside int it has to be cast.
- The keyword enum is used to create numeric constants in .NET framework.
- There must be a numeric value for each enum type.
- The default underlying type of the enumeration elements is int. By default, the first enumerator has the value 0, and the value of each successive enumerator is increased by 1.

Example:-

```
using System;
enumDays {Sat, Sun, Mon, Tue, Wed, Thu, Fri};

class Abc
{
public static void main()
{

Console.WriteLine((int)Days.Wed);
Console.WriteLine((Days)4);
}
}
```

Q17. Explain inheritance and polymorphism.

Inheritance

Inheritance is the concept we use to build new classes using the existing class definitions. Through inheritance we can modify a class the way we want to create new objects. The original class is known as base or parent class & the modified class is known as derived or subclass or child class. The

concept of inheritance facilitates the reusability of existing code & thus improves the integrity of programs & productivity of programmers.

Polymorphism

Polymorphism is the ability to take more than one form. For example, an operation may exhibit different behavior in different situations. The behavior depends upon the types of data used on the operation. Polymorphism is extensively used while implementing inheritance.

Two types

Compile time polymorphism

Runtime time polymorphism

Q18. What are sealed classes and sealed methods? Why are they used?

The methods declared with a sealed keyword in its header are known as sealed methods. Such method provides a complete implementation to its base class virtual method using the override keyword.

Characteristics

- A method cannot be defined as sealed unless that method is an override of a method in its base class.
- A sealed method cannot be overridden further.
- Sealed methods are useful to avoid problems caused by overriding the existing functionality.
- It prevents the user from changing the internal functionality of a class.

Example:

```
using System;
class A
{
public virtual void F()
{
Console.WriteLine("A.F");
}
public virtual void G()
{
Console.WriteLine("A.G");
}
}
class B: A
{
sealed override public void F()
{
Console.WriteLine("B.F");
}
override public void G()
{
Console.WriteLine("B.G");
}
}
class C: B
```

```
{
override public void G()
{
Console.WriteLine("C.G");
}
}
```

The class B provides two override methods: an F method that has the sealed modifier and a G method that does not. B's use of the sealed modifier prevents C from further overriding F.

Sealed Classes:

- Generally if we create classes we can inherit the properties of that created class in any class without having any restrictions.
- In some situation we will get requirement like we don't want to give permission for the users to derive the classes from it or don't allow users to inherit the properties from particular class.
- For that purpose we have keyword called "sealed" in OOPS.
- When we defined class with keyword "Sealed" then we don't have a chance to derive that particular class and we don't have permission to inherit the properties from that particular class.
- A sealed class cannot be inherited.
- It is an error to use a sealed class as a base class.
- Use the sealed modifier in a class declaration to prevent inheritance of the class.
- It is not permitted to use the abstract modifier with a sealed class.

Example:

```
using System;

sealed class MyClass
{
public int x;
public int y;
}
class MainClass
{
public static void Main()
{
MyClass mC = new MyClass();
mC.x = 110;
mC.y = 150;
Console.WriteLine("x = {0}, y = {1}", mC.x, mC.y);
}
}
```

Q19. What are the rules in defining a constructor? Explain static constructor with example.

Rules for constructor:

- A constructor should have the same name as that of its class.

- Constructors can take any number of arguments.
- Constructor cannot return any value.
- A static constructor is used to initialize static variables of a class. It is declared using the static keyword. A class can have only one static constructor.

Static Constructor:

The general syntax of a static constructor is shown below.

```
static classname()
{
  //static member initializations;
}
```

- Also note that a static constructor is parameter less and no access modifiers are allowed with its declaration. The exact timing of static constructor execution is implementation-dependent, but is subject to the following rules:
- The static constructor for a class executes before any instance of the class is created.
- The static constructor for a class executes before any of the static members for the class are referenced.
- The static constructor for a class executes after the static fields initializes for the class.
- The static constructor for a class executes at most one time during a single program execution.

Example:

```
class A
{
static A()
{
Console.WriteLine("static constructor A is invoked");
}
public static void Main()
{
 }
}
```

Q20. Explain various Types of Constructors in C#

Default and Parameterized

- When constructor does not accept any parameter then it is referred as default constructor or zero parametrized.
- When constructor accepts the parameter then it is referred as parametrized constructor.

Example:

```
using System;
class test
{
double length, breadth;
public test()
```

```
{
Console.WriteLine("default cons");
}
public test(double l, double b)
{
//Console.WriteLine("paramaterized cons");
length = l;
breadth = b;
}
public double area()
{
return (length * breadth);

}
}
class testmain
{
public static void Main()
{
test defaultcon =new test();//calls default cons
test t1 = new test(10.05, 100.45);
double result = t1.area();
Console.WriteLine("area of rectangle-->" + result);
Console.ReadLine();
}
}
```

```
default cons
area of rectangle-->1009.5225
```

Private constructor

- Private constructors are used to restrict the instantiation of object using 'new' operator.
- A private constructor is a special instance constructor. It is commonly used in classes that contain static members only.
- This type of constructors is mainly used for creating singleton object.
- If you don't want the class to be inherited we declare its constructor private.

Example:

```
using System;
class test
{
private test()
{
Console.WriteLine("private cons ");
}
public test(int x)
{
```

```
Console.WriteLine("non private cons--->"+ x);
}
}
class testmain
{
public static void Main()
{
test t1 = new test();//Error'test.test()' is inaccessible due to its protection level
test t2 = new test(10);
Console.ReadLine();
}
}
```

Static Constructor

- There can be only one static constructor in the class.
- The static constructor should be without parameters.
- It can only access the static members of the class.
- There should be no access modifier in static constructor definition.

Example:

```
using System;
class test
{
static int x;
static test()
{
Console.WriteLine("static cons ");
x = 10;//set the values for static member here
}
public static void show()
{
Console.WriteLine("show of x--->" + x);
}
}
class testmain
{
public static void Main()
{
test.show();
Console.ReadLine();
}
}
```

```
static cons
show of x--->10
```

Copy constructor

When constructor accepts the object as parameter then it is referred as copy constructor.

Example:

```
using System;
class test
{
double length, breadth;
public test()
{
Console.WriteLine("default cons");
}
public test(double l, double b)
{
//Console.WriteLine("paramaterized cons");
length = l;
breadth = b;
}
public test(test t1)
{
length = t1.length;
breadth = t1.breadth;
}
public double area()
{
return (length * breadth);
}
}
class testmain
{
public static void Main()
{
test defaultcon = new test();//calls default cons
test t1 = new test(10,10);
double result = t1.area();
Console.WriteLine("area of rectangle t1-->" + result);
test t2 = new test(20, 20);
double result1 = t2.area();
Console.WriteLine("area of rectangle t2-->" + result1);
test t3 = new test(t2);
double result_copy = t3.area();
Console.WriteLine("area of rectangle t3 is copy_con of t2-->" + result_copy);
Console.ReadLine();
}
}
```

```
default cons
area of rectangle t1-->100
area of rectangle t2-->400
area of rectangle t3 is copy_con of t2-->400
```

Q21. How multiple inheritance achieved using interfaces?

C# does not support multiple inheritance. However, you can use interfaces to implement multiple inheritance. The following program demonstrates this:

Example:

```
using System;
class Shape
{
public void setWidth(int w)
{
width = w;
}
public void setHeight(int h)
{
height = h;

}
protected int width;
protected int height;
}

public interface PaintCost
{
int getCost(int area);
}
class Rectangle : Shape, PaintCost
{
public int getArea()
{
return (width * height);
}
public int getCost(int area)
{
return area * 70;
}
}
class RectangleTester
{
static void Main(string[] args)
{
Rectangle Rect = new Rectangle();
int area;
```

```
Rect.setWidth(5);
Rect.setHeight(7);
area = Rect.getArea();
Console.WriteLine("Total area: {0}", Rect.getArea());
Console.WriteLine("Total paint cost: ${0}" , Rect.getCost(area));
Console.ReadKey();
}
}
}
```

Q22. When is explicit interface necessary?

- C# does not support multiple inheritances, but a class has the option of implementing one or more interfaces.
- One challenge with interfaces is that they may include methods that have the same signatures as existing class members or members of other interfaces.
- Explicit interface implementations can be used to disambiguate class and interface methods that would otherwise conflict.
- Explicit interfaces can also be used to hide the details of an interface that the class developer considers private.
- To explicitly implement an interface member, just use its fully qualified name in the declaration.
- A fully qualified interface name takes the form **InterfaceName.MemberName**

Example:

```
interface I1
{
void A();
}
interface I2
{
void A();
}
class C : I1, I2
{
public void I1.A()
{
Console.WriteLine("A() from I");
}
void I2.A()
{
Console.WriteLine("A() from I2");
}
}
```

Q23. Explain the similarities and differences between Interfaces and Abstract classes

Similarities:

- Both abstract classes and interfaces may contain members that can be inherited by a derived class.

- Neither interfaces nor abstract classes may be directly instantiated, but we can declare variables of these types. If we do, we can use polymorphism to assign objects that inherit from these types to variables of these types.
- In both cases, we can then use the members of these types through these variables, although we don't have direct access to the other members of the derived object.

Differences:

- Derived classes may only inherit from a single base class, which means that only a single abstract class can be inherited directly. Conversely, classes can use as many interfaces as they want
- Abstract classes may possess both abstract members and non-abstract members. Interface members conversely, must be implemented on the class that uses the interface they do not possess code bodies.
- Moreover, interface members are by definition public but members of abstract classes may also be private (as long as they aren't abstract), protected, internal, or protected internal. In addition, interfaces can't contain fields, constructors, destructors, static members, or constants.

Q24. Explain method overriding with example

- Method overriding in C# is achieved through inheritance.
- If a class is inherited by a derived class, then the derived class can override the base class method. This concept is termed as Method Overriding.
- Method Overriding is implemented in C# using the keywords virtual and override. The base class method that has to be overridden must be marked as virtual or abstract (if the base class is an abstract class).
- The derived class method that overrides the base class method should be marked with the keyword override.

Example:

```
using System;
class A
{
public virtual void Test() { Console.WriteLine("A::Test()"); }
}

class B : A
{
public override void Test() { Console.WriteLine("B::Test()"); }
}
class C : B
{
public override void Test() { Console.WriteLine("C::Test()"); }
}
class Program
{
static void Main(string[] args)
{
A a = new A();
B b = new B();
C c = new C();
```

```
a.Test(); // output --> "A::Test()"
b.Test(); // output --> "B::Test()"
c.Test(); // output --> "C::Test()"
a = new B();
a.Test(); // output --> "B::Test()"
b = new C();
b.Test(); // output --> "C::Test()"
Console.ReadKey();
}
}
```

Q25. Differentiate Overloading and Overriding

Overloading

Method Overloading means having two or more methods with the same name but with different signature (different parameters list and different type of parameters) in same class or in different classes. Method Overloading forms compile-time polymorphism.

Overriding

Method overriding means having two methods with same name and same signature, one method in base class and other method in derived class.

A subclass inherits methods from a base class. Sometimes, it is necessary for the subclass to modify the methods defined in the base class. This is referred to as method overriding.

This can be achieved by using the virtual and override keywords. We have to use the virtual keyword for the method which in base class and override keyword for the method in subclass.

By default functions are not virtual in C# and so you need to write "virtual" explicitly.

Overloading	Overriding
Having same method name with different Signatures.	Methods name and signatures must be same.
Overloading is the concept of compile time polymorphism	Overriding is the concept of runtime polymorphism
Two functions having same name and return type, but with different type and/or number of arguments is called as Overloading	When a function of base class is re-defined in the derived class called as Overriding
It doesn't need inheritance.	It needs inheritance.
Method can have different data types	Method should have same data type.
Method can be different access specifies	Method should be public.

Q26. Explain method hiding with example.

In a C# program a derived class can have methods with the same signature as that of its base class methods. This hides the base class methods by its derived class counterparts and gives a warning.

This warning can be suppressed by using the **new** keyword with the derive class methods. This means that the members of the base class are made intentionally hidden by the members having the same signature in the derived class. The new function will never be invoked by a base class pointer.

Example:-

class demo

```
{
public void disp()
{
System.Console.WriteLine("From disp method of base class");
}
}

class test:demo
{
new public void disp()
{
System.Console.WriteLine("From disp method of derived class");
}
public static void Main()
{

test t=new test();
t.disp();
}
}
```

Q27. What is the difference between overriding methods and hiding methods?

Overriding methods:

In method overriding we can override a method in base class by creating similar method in derived class this can be achieved by using inheritance principle and using "**virtual & override**" keywords.

If we want to override base class method then we need to declare base class method with "virtual" keyword and the method which we created in derived class to override base class need to declare with "override" keyword like as shown below

Example:-

```
class SampleA
{
public virtual void Show()
{
Console.WriteLine("Sample A Test Method");
}
}
class SampleB:SampleA
{
public override void Show()
{
Console.WriteLine("Sample B Test Method");
}
}
class Program
{
static void Main(string[] args)
```

```
{
SampleA a=new SampleA();
SampleB b=new SampleB();
a.Show();
b.Show();
a = new SampleB();
a.Show();
Console.ReadLine();
}
}
}
```

Hiding methods:

To hide base class methods in derived classes we can declare derived class methods with new keyword. To use new keyword, we need to write the code like as shown below

Example:-

```
class SampleA
{
public void Show()
{

Console.WriteLine("Sample A Test Method");
}
}
class SampleB:SampleA
{
public new void Show()
{
Console.WriteLine("Sample B Test Method");
}
}
class Program
{
static void Main(string[] args)
{
SampleA a=new SampleA();
SampleB b=new SampleB();
a.Show();
b.Show();
a = new SampleB();
a.Show();
Console.ReadLine();
}
}
```

Q28. Define each of the following terms:

Derived class

The functionality of an existing class can be extended by creating a new class from it. The new class is then called a child class or derived class. The derived class inherits the properties of the base class.

Abstract class

An Abstract class is a non-instantiable class which is either partially implemented, or not at all implemented. It can have abstract methods as well as non-abstract methods. Abstract classes require subclasses to provide implementations for the abstract methods. It provides default functionality to its sub classes.

Static class

A static class is class whose objects cannot be created and must contain only static members.

Sealed class

A sealed class is a class which cannot be inherited. In C#, the sealed modifier is used to define a class as sealed.

Partial class

It is possible to split the definition of a class over two or more source files. Each source file contains a section of the type or meth

Q29. Write a short note on Assembly.

An assembly in ASP.NET is a collection of single-file or multiple files. The assembly that has more than one file contains either a dynamic link library (DLL) or an EXE file. The assembly also contains metadata that is known as assembly manifest.

The assembly manifest contains data about the versioning requirements of the assembly, author name of the assembly, the security requirements that the assembly requires to run, and the various files that form part of the assembly.

The biggest advantage of using ASP.NET Assemblies is that developers can create applications without interfering with other applications on the system. When the developer creates an application that requires an assembly that assembly will not affect other applications.

The assembly used for one application is not applied to another application. However, one assembly can be shared with other applications. In this case the assembly has to be placed in the bin directory of the application that uses it.

This is in contrast to DLL in the past. Earlier developers used to share libraries of code through DLL. To use the DLL that is developed by another developer for another application, we must register that DLL in our machine. In ASP.NET, the assembly is created by default whenever we build a DLL. We can check the details of the manifest of the assembly by using classes located in the System.Reflection namespace.

Thus, we can create two types of ASP.NET Assemblies in ASP.NET: private ASP.NET Assemblies and shared assemblies. Private ASP.NET Assemblies are created when you build component files like DLLs that can be applied to one application.

Shared ASP.NET Assemblies are created when you want to share the component files across multiple applications. Shared ASP.NET Assemblies must have a unique name and must be placed in Global Assembly Cache (GAC). The GAC is located in the Assembly directory in WinNT. You can view both the manifest and the IL using ILDisassembler (ildasm.exe).

Q30. List and Explain the Components of assemblies.

An assembly is a fundamental building block of any .NET framework application. It contains the code that is executed by common language runtime. For example, when we build a simple C# application, Visual Studio creates an assembly in the form of a single portable executable (PE) file, specifically an EXE or DLL.

Every assembly has a file called '**manifest'** file that stores all information about that assembly. This information's are known as metadata.

The manifest file contains all the metadata needed to specify the assembly's version requirements, security identity, and all metadata needed to define the scope of the assembly and resolve references to resources and classes.

Components of Assembly

Manifest
It describes the assembly. The manifest file contains all the metadata needed to specify the assembly's version requirements, security identity, and all metadata needed to define the scope of the assembly and resolve references to resources and classes.

Type Metadata
It contains metadata, which describes each type (class, structure, enumeration, and so forth)

MSIL
It contains Intermediate language code.

Resources
It contains bitmaps, icons, audios and other types of resources.

Q31. Difference between public assembly and private assembly

Public Assembly	Private Assembly
Public assembly can be used by multiple applications.	Private assembly can be used by only one application.
Public assembly is stored in GAC (Global Assembly Cache).	Private assembly will be stored in the specific application's directory or sub-directory.
Public assembly is also termed as shared assembly.	There is no other name for private assembly.
Strong name has to be created for public assembly.	Strong name is not required for private assembly.
Public assembly should strictly enforce version constraint.	Private assembly doesn't have any version constraint.
An example to public assembly is the actuate report classes which can be imported in the library and used by any application that prefers to implement actuate reports.	By default, all assemblies you create are examples of private assembly. Only when you associate a strong name to it and store it in GAC, it becomes a public assembly.

Q32. How Garbage collector works.

Every time our application instantiates a reference-type object, the CLR allocates space on the managed heap for that object.

However, we never need to clear this memory manually. As soon as our reference to an object goes out of scope (or our application ends), the object becomes available for garbage collection. The garbage collector runs periodically inside the CLR, automatically reclaiming unused memory for inaccessible objects.

The .NET Framework provides automatic memory management called garbage collection. A .NET program that runs in a managed environment is provided with this facility by .NET CLR (common language runtime).

The purpose of using Garbage Collector is to clean up memory. In .NET all dynamically requested memory is allocated in the heap which is maintained by CLR. The garbage collector continuously looks for heap objects that have references in order to identify which ones are accessible from the code. Any objects without reference will be removed at the time of garbage collection. The Garbage collection is not deterministic.

It is called only when the CLR decides that it is most needed. It happens in a situation such as the heap for the given process is becoming full and requires a clean-up.

In the common language runtime (CLR), the garbage collector serves as an automatic memory manager. It provides the following benefits:

- Enables you to develop your application without having to free memory.
- Allocates objects on the managed heap efficiently.
- Reclaims objects that are no longer being used, clears their memory, and keeps the memory available for future allocations. Managed objects automatically get clean content to start with, so their constructors do not have to initialize every data field.
- Provides memory safety by making sure that an object cannot use the content of another object.

GC.Collect () method:

- This method is used to call garbage collector explicitly. It is used to force a garbage collection to occur at any time.

Q33. What is namespace? Explain System namespace.

Namespaces are C# program elements designed to help you organize our programs. They also provide assistance in avoiding name clashes between two sets of code. Implementing Namespaces in our own code is a good habit because it is likely to save us from problems later when we want to reuse some of our code.

For example, if we created a class named Console, we would need to put it in our own namespace to ensure that there wasn't any confusion about when the **System.Console** class should be used or when our class should be used. Generally, it would be a bad idea to create a class named Console, but in many cases your classes will be named the same as classes in either the .NET Framework Class Library or namespaces help us to avoid the problems that identical class names would cause.

System is fundamental namespace for C# application. It contains all the fundamental classes and base classes which are required in simple C# application. These classes and sub classes defines reference data type, method and interfaces. Some classes provide some other feature like data type conversion, mathematical function.

Some functionality provided by System namespace

- Commonly-used value
- Mathematics
- Remote and local program invocation
- Application environment management
- Reference data types
- Events and event handlers
- Interfaces Attributes Processing exceptions
- Data type conversion
- Method parameter manipulation

Alias of namespace

Below example demonstrate use of alias where A is alias for System.Consolenamespaces.

```
using A=System.Console;

class Abc
{
public static void Main()
{
A.Write("Welcome to C# !!!!");
}
}
```

Q34. List and explain the use of any five namespaces in c#.

- The **System** namespace contains fundamental classes and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.
- The **System.Collections** namespace contains interfaces and classes that define various collections of objects, such as lists, queues, bit arrays, hash tables and dictionaries.
- The **System.Text.RegularExpressions** namespace contains classes like Regex, Match and MatchCollection Class that provide access to the .NET Framework regular expression engine.
- The **System.Data** namespace provides access to classes that represent the ADO.NET architecture. ADO.NET lets you build components that efficiently manage data from multiple data sources.
- The **System.Drawing** parent namespace contains types that support basic GDI+ graphics functionality.

Q35. What are the different types of collections in .NET? Explain.

Collection: Collections are basically group of records which can be treated as a one logical unit.
.NET Collections are divided in to four important categories as follows.

- Indexed based.
- Key Value Pair.
- Prioritized Collection.
- Specialized Collection.

Indexed based:
It helps us to access the value by using generated index number by the collection.

ArrayList: A simple resizable, index-based collection of objects.

Key Value Pair: It helps you to access value by the user defined key.

SortedList: A sorted collection of name/value pairs of objects.
Hashtable: A collection of name/value pairs of objects that allows retrieval by name or index

Prioritized Collection: It helps us to get the element in a particular sequence.

Queue: A first-in, first-out collection of objects
Stack: A last-in, first-out collection of objects

Specialized Collection:
It is very specific collections which are meant for very specific purpose like hybrid dictionary that start as list and become Hashtable.

StringCollection: A simple resizable collection of strings
StringDictionary: A collection of name/values pairs of strings that allows retrieval by name or index
ListDictionary: An efficient collection to store small lists of objects

HybridDictionary: A collection that uses a ListDictionary for storage when the number of items in the collection is small, and then migrate the items to a Hashtable for large collections

Q36. What is ArrayList? Explain Property and Methods of ArrayList.

ArrayList is a non-generic type of collection in C#. It can contain elements of any data types. It is similar to an array, except that it grows automatically as you add items in it. Unlike an array, we don't need to specify the size of ArrayList.

Example: Initialize ArrayList

ArrayList myArryList = new ArrayList();

Properties and Methods of ArrayList

Property	Description
Capacity	Gets or sets the number of elements that the ArrayList can contain.
Count	Gets the number of elements actually contained in the ArrayList.

Method	Description
Add()/AddRange()	Add() method adds single elements at the end of ArrayList. AddRange() method adds all the elements from the specified collection into ArrayList.
Insert()/InsertRange()	Insert() method insert a single elements at the specified index in ArrayList. InsertRange() method insert all the elements of the specified collection starting from specified index in ArrayList.
Remove()/RemoveRange()	Remove() method removes the specified element from the ArrayList. RemoveRange() method removes a range of elements from the ArrayList.
RemoveAt()	Removes the element at the specified index from the ArrayList.
Sort()	Sorts entire elements of the ArrayList.
Reverse()	Reverses the order of the elements in the entire ArrayList.
Contains	Checks whether specified element exists in the ArrayList or not. Returns true if exists otherwise false.

Example: Add elements into ArrayList

```
ArrayList arryList1 = new ArrayList();
arryList1.Add(1);
arryList1.Add("Two");
arryList1.Add(3);
arryList1.Add(4.5);

ArrayList arryList2 = new ArrayList();
arryList2.Add(100);
arryList2.Add(200);

//adding entire arryList2 into arryList1
arryList1.AddRange(arryList2);
```

Example: Insert()

```
ArrayList myArryList = new ArrayList();
myArryList.Add(1);
myArryList.Add("Two");
myArryList.Add(3);
```

```
myArryList.Add(4.5);

myArryList.Insert(1, "Second Item");
myArryList.Insert(2, 100);
```

Example: Remove()

```
ArrayList arryList1 = new ArrayList();
arryList1.Add(100);
arryList1.Add(200);
arryList1.Add(300);
arryList1.Remove(100); //Removes 1 from ArrayList
```

Example: RemoveAt()

```
ArrayList arryList1 = new ArrayList();
arryList1.Add(100);
arryList1.Add(200);
arryList1.Add(300);
arryList1.RemoveAt(1); //Removes the first element from an ArrayList
```

Example: Contains()

```
ArrayList myArryList = new ArrayList();
myArryList.Add(100);
myArryList.Add("Hello World");
myArryList.Add(300);

Console.WriteLine(myArryList.Contains(100));
```

Q37. What is generic class? Explain with example.

Generics introduced in C# 2.0. Generics allow us to define a class with placeholders for the type of its fields, methods, parameters, etc. Generics replace these placeholders with some specific type at compile time.

A generic class can be defined using angle brackets <>. For example, the following is a simple generic class with a generic member variable, generic method and property.

Example:

```
class MyGenericClass<T>
{
   private T genericMemberVariable;

   public MyGenericClass(T value)
   {
      genericMemberVariable = value;
   }

   public T genericMethod(T genericParameter)
   {
      Console.WriteLine("Parameter type: {0}, value: {1}", typeof(T).ToString(),genericParameter);
      Console.WriteLine("Return type: {0}, value: {1}", typeof(T).ToString(), genericMemberVariable);

      return genericMemberVariable;
   }

   public T genericProperty { get; set; }
}
```

MyGenericClass<int> intGenericClass = new MyGenericClass<int>(10);

int val = intGenericClass.genericMethod(200);

As we can see in the above code, MyGenericClass is defined with <T>. <> indicates that MyGenericClass is generic and the underlying type would be defined later, for now consider it as T. We can take any character or word instead of T.

Output:

Parameter type: int, value: 200

Return type: int, value: 10

Q38. How to create and user property in C#.

In C#, properties are nothing but natural extension of data fields. They are usually known as '**smart fields**' in C# community. We know that data encapsulation and hiding are the two fundamental characteristics of any object-oriented programming language.

In C#, data encapsulation is possible through either classes or structures. By using various access modifiers like private, public, protected, internal etc it is possible to control the accessibility of the class members.

Usually inside a class, we declare a data field as private and will provide a set of public SET and GET methods to access the data fields.

In C#, properties are defined using the property declaration syntax. The general form of declaring a property is as follows.

```
<acces_modifier> <return_type> <property_name>
{
get
{
}
set
{
}
}
```

Example:

```
using System;
class MyClass
{
private int x;
public int X
{
get
{
return x;
}
set
{
x = value;
}
}
}
class MyClient
{
public static void Main()
{
MyClass mc = new MyClass();
```

```
mc.X = 10;
int xVal = mc.X;
Console.WriteLine(xVal);//Displays 10
}
}
```

Q39. What is delegate? Explain the steps to implement delegate in C#.NET.

Delegate:

The dictionary meaning of 'Delegates' is "A person acting for another person". A delegate in C# is a class type object & is used to invoke a method that has been encapsulated into it at the time of its creation.

Following steps are used to create delegate:

Delegate declaration

modifier delegate return_type delegate_name(parameters)

Delegate methods definition

```
modifier function_name(parameters)
{
//statement(s)
}
```

Delegate instantiation

new delegate_type(expression);

Delegate invocation

Delegate_object (parameters);

Note: signature of delegate and function must be same

Example:

```
public delegate void MyDelegate(int , int );

class Abc
{

public void Addition(int a, int b)
{
Console.WriteLine ("Addition :"+(a+b));
}

public void Subtraction(int a, int b)
{
Console.WriteLine ("Subtraction :"+(a-b));
}


}
class Xyz
{
public static void Main()
{

Abc a1=new Abc();
MyDelegate m1=new MyDelegate(a1.Addition);
MyDelegate m2=new MyDelegate(a1.Subtraction);
m1(10,20);
m2(40,5);
Console.Read();
```

```
}
}
```

Q40. What is delegate? Explain multiple delegate with example.

Delegate:

A delegate in C# is similar to a function pointer in C or C++. It is a reference type object that allows the programmer to encapsulate a reference to a method in it. It defines the return type and signature for a method and can refer to any methods that are of the same format in the signature.

When same delegate is used to call method multiple time then it is referred as Multicast delegate or Multiple delegate.

Following are condition for multicast delegate:

- void is used as return type
- Out parameter is not allowed as arguments

Example

```
public delegate void MyDelegate();

class Abc
{

public void Show()
{
Console.WriteLine ("New Delhi");
}

public void Display()
{
Console.WriteLine ("New York");
}


}
class Xyz
{
public static void Main()
{

Abc a1=new Abc();
MyDelegate m1=new MyDelegate(a1.Show);
MyDelegate m2=new MyDelegate(a1.Display);
m1=m1+m2+m1+m2-m1;
m1();
Console.Read();

}
```

Q41. Delegates in C# are used for Event Handling. Justify this statement with a relevant example program.

Events:

An important C# feature is built upon the foundation of delegates: the event. An event is, essentially, an automatic notification that some action has occurred. Events work like this: An object that has an interest in an event registers an event handler for that event. When the event occurs, all registered handlers are called. Event handlers are represented by delegates. Events are members of a class and are declared using the event keyword.

Its most commonly used form is shown here:

event event-delegate event-name;

Here, **event-delegate** is the name of the delegate used to support the event, and event-name is the name of the specific event object being declared.

Example:

```
using System;

delegate void MyEventHandler();

class MyEvent
{
public event MyEventHandler SomeEvent;
public void OnSomeEvent() {
if(SomeEvent != null)
SomeEvent();
}
}
class EventDemo
{
static void Handler() {
Console.WriteLine("Event occurred");
}
static void Main()
{
MyEvent evt = new MyEvent();
evt.SomeEvent += Handler;
evt.OnSomeEvent();
}
}
```

Unit-II

Topics:

Web Form Fundamentals |Form Controls

Q1. List and explain different files and folder in ASP.NET web application.

ASP.NET File Types: - ASP.NET applications can include many types of files. Below table show list of files.

File Name	Description
.aspx	These are ASP.NET web pages. They contain the user interface and, optionally, the underlying application code. Users request or navigate directly to one of these pages to start our web application
.ascx	These are ASP.NET user controls. User controls are similar to web pages, except that the user can't access these files directly. Instead, they must be hosted inside an ASP.NET web page. User controls allow you to develop a small piece of user interface and reuse it in as many web forms as you want without repetitive code.
web.config	This is the configuration file for your ASP.NET application. It includes settings for customizing security, state management, memory management, and much more.
global.asax	This is the global application file. You can use this file to define global variables (variables that can be accessed from any web page in the web application) and react to global events (such as when a web application first starts).
.cs	These are code-behind files that contain C# code. They allow you to separate the application logic from the user interface of a web page.

ASP.NET Web Folders.

Every web application starts with a single location, called the root folder. However, in a large, well-planned web application.

Directory	Description
App_Code	Contains source code files that are dynamically compiled for use in our application
App_GlobalResources	Stores global resources that are accessible to every page in the web application. This directory is used in localization scenarios, when we need to have a website in more than one language.
App_Data	Stores data, including SQL Server Express database files
App_Themes	Stores the themes that are used to standardize and reuse formatting in our web application.

Q2. Explain the three layer architecture of ASP.NET.

Asp.net architecture consists of 3 layers as follows:

Presentation Layer

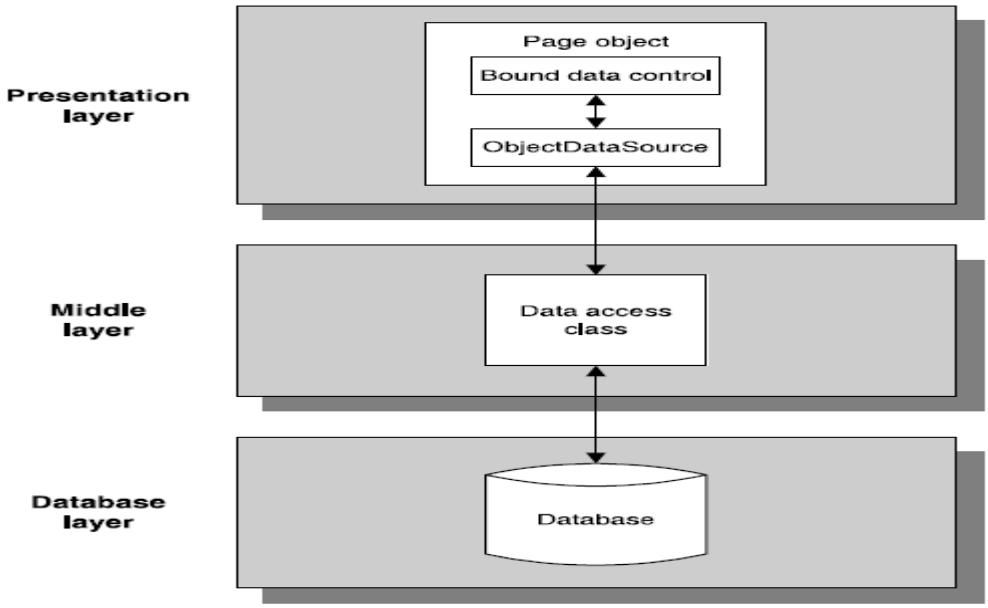
The presentation layer consists of the Asp.net page that manages the appearance of application. This layer can include bound data controls and ObjectDataSource objects that bind data controls to the data.

Middle Layer

The middle layer contains the data access classes that manage the data access for the application. This layer can also contain business objects that represent business entities such as customers, products or employee and that implement business rules such as credit and discount policies.

Database Layer

This layer consists of the database that contains the data for the application. Ideally the SQL statement that do the database access should be saved in stored procedure within the database, nut the SQL statement are often stored in the data access classes.

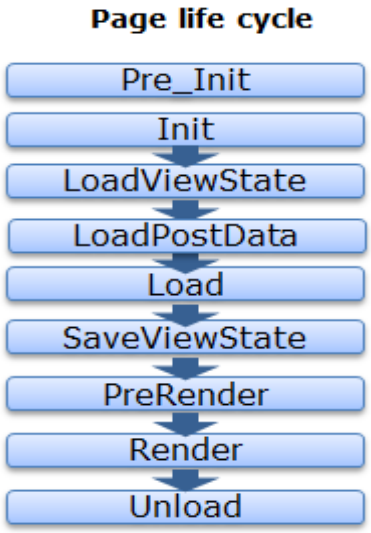


Q3. Explain ASP.NET Life Cycle Page.

ASP.NET life cycle specifies, how:

- ASP.NET processes pages to produce dynamic output
- The application and its pages are instantiated and processed
- ASP.NET compiles the pages dynamically

When a page is requested, it is loaded into the server memory, processed and sent to the browser. Then it is unloaded from the memory. At each of this steps, methods and events are available, which could be overridden according to the need of the application.



Following are the different stages of an ASP.Net page:

Page request
When ASP.Net gets a page request, it decides whether to parse and compile the page or there would be a cached version of the page; accordingly the response is sent

Starting of page life cycle:
At this stage, the Request and Response objects are set. If the request is an old request or post back, the IsPostBack property of the page is set to true.

Page initialization
At this stage, the controls on the page are assigned unique ID by setting the UniqueID property and themes are applied.

Page load
At this stage, control properties are set using the view state and control state values.

Validation
Validate method of the validation control is called and if it runs successfully, the IsValid property of the page is set to true.

Postback event handling
If the request is a postback (old request), the related event handler is called.

Page rendering
At this stage, view state for the page and all controls are saved.

Unload
The rendered page is sent to the client and page properties, such as Response and Request are unloaded and all cleanup done.

Following are the page life cycle events:

PreInit()
Init()
InitComplete()
LoadViewState()
LoadPostData()
PreLoad()
Load()
LoadComplete()
PreRender()
PreRenderComplete()
SaveStateComplete()
UnLoad()

Q4. Differentiate between inline code and code behind

Code Behind	Inline code
The HTML and controls are in the *.aspx* file, and the code is in a separate *.aspx.vb* or *.aspx.cs* file.	The code is in <script> blocks in the same *.aspx* file that contains the HTML and controls.
The code for the page is compiled into a separate class from which the *.aspx* file derives.	The *.aspx* file derives from the Page class.
All project class files (without the *.aspx* file itself) are compiled into a *.dll* file, which is deployed to the server without any source code. When a request for the page is received, then an instance of the project *.dll* file is created and executed.	When the page is deployed, the source code is deployed along with the Web Form page, because it is physically in the *.aspx* file. However, you do not see the code, only the results are rendered when the page runs.
The program logic is separated from user interface design code. So it is easy to understand.	Both program logic and user interface design code are placed in the same .aspx file, which makes it cumbersome.

Q.5Difference between HTML control and Web Server control.

HTML control	Web Server control
HTML control runs at **client side.**	ASP.Net controls run at **server side**.
We can run HTML controls at server side by adding attribute **runat="server"**.	Wecannot run ASP.Net Controls on client side as these controls have this attribute **runat="server"** by default**.**
HTML controls are client side controls, so it does not provide **STATE management.**	ASP.Net Controls are Server side controls, provides STATE management.
HTML control does not require rendering.	HTML control does not require rendering.
As HTML controls runs on client side, execution is **fast**.	As ASP.Net controls run on server side, execution is **slow**.
HTML controls does not support **Object Oriented paradigm**.	With ASP.Net controls, you have full support of Object oriented paradigm.
HTML control have **limited** set of properties and/or methods.	ASP.Net controls have **rich** set of properties and/or methods.
<input type="text" ID="txtName">	<asp:TextBoxId="txtName"runat="server"> </asp:TextBox>

Q.6 Short note on Page class.

When an ASP.NET page is requested and renders markup to a browser, ASP.NET creates an instance of a class that represents our page. That class is composed not only of the code that we wrote for the page, but also code that is generated by ASP.NET.

Page Properties

Property	Description
IsPostBack	This Boolean property indicates whether this is the first time the page is being run (false) or whether the page is being resubmitted in response to a controlevent, typically with stored view state information (true).
EnableViewState	When set to false, this overrides the EnableViewState property of the contained controls, thereby ensuring that no controls will maintain state information.
Application	This collection holds information that's shared between all users in our website. For example, we can use the Application collection to count the number of times a page has been visited.
Session	This collection holds information for a single user, so it can be used in different pages. For example, we can use the Session collection to store the items in the current user's shopping basket on an e-commerce website.
Request	This refers to an HttpRequest object that contains information about the current web request. We can use the HttpRequest object to get information about the user's browser.
Response	This refers to an HttpResponse object that represents the response ASP.NET will send to the user's browser

Q7. Explain the use of global.asx and web.config files in ASP.NET application.

OR

Explain the files used to configure an ASP.NET Application.

- The Global.asax file, also known as the ASP.NET application file, is an optional file that contains code for responding to application-level events raised by ASP.NET or by HttpModules.
- The Global.asax file resides in the root directory of an ASP.NET-based application. The Global.asax file is parsed and dynamically compiled by ASP.NET.

- The Global.asax file itself is configured so that any direct URL request for it is automatically rejected; external users cannot download or view the code written within it.
- The Global.asax file does not need recompilation if no changes have been made to it. There can be only one Global.asax file per application and it should be located in the application's root directory only.

The Global.asax contains two types of events those are

- Events which are fired for every request
- Events which are not fired for every request

Events which are fired for every request:-

Application_BeginRequest() – This event raised at the start of every request for the web application.
Application_EndRequest() – This event raised at the end of each request right before the objects released.
Application_PreRequestHandlerExecute() – This event called before the appropriate HTTP handler executes the request.
Application_PostRequestHandlerExecute() – This event called just after the request is handled by its appropriate HTTP handler.

Events which are not fired for every request:-

Application_Start() – This event raised when the application starts up and application domain is created.
Session_Start() – This event raised for each time a new session begins, This is a good place to put code that is session-specific.
Application_Error() – This event raised whenever an unhandled exception occurs in the application. This provides an opportunity to implement generic application-wide error handling.
Session_End() – This event called when session of user ends.
Application_End() – This event raised just before when web application ends.
Application_Disposed() – This event fired after the web application is destroyed and this event is used to reclaim the memory it occupies.

Web.Config

- Configuration file is used to manage various settings that define a website. The settings are stored in XML files that are separate from our application code. In this way we can configure settings independently from our code.
- Generally a website contains a single Web.config file stored inside the application root directory. However there can be many configuration files that manage settings at various levels within an application.
- There are number of important settings that can be stored in the configuration file. Some of the most frequently used configurations, stored conveniently inside Web.config file are:
  - Database connections
  - Caching settings
  - Session States
  - Error Handling
  - Security

Benefits of XML-based Configuration files

- ASP.NET Configuration system is extensible and application specific information can be stored and retrieved easily. It is human readable.
- We need not restart the web server when the settings are changed in configuration file. ASP.NET automatically detects the changes and applies them to the running ASP.NET application.

Configuration file looks like this

```
1. <configuration>
2. <connectionStrings>
3. <add name="myCon" connectionString="server=MyServer;database=puran;uid=p
      uranmehra;pwd=mydata1223" />
4. </connectionStrings>
5. </configuration/>
```

Q8. Explain TextBox control in ASP.NET with any five properties.

This is an input control which is used to take user input. To create **TextBox** either we can write code or use the drag and drop facility of visual studio IDE.

This is server side control, asp provides own tag to create it. The example is given below.

< asp:TextBoxID="TextBox1" runat="server" ></asp:TextBox>

This control has its own properties that are tabled below.

Property	Description
Text	It is used to set text to be shown for the control.
MaxLength	It is used to set maximum number of characters that can be entered.
Readonly	It is used to make control readonly.
TextMode	It used to determine behavior of control. Possible value are Single / Multiline / Password
CausesValidation	true/false – It is used to enable or disable validation effect on control

Example:

Assign a text to TextBox control when Button click event fires using c#

C# code for above TextBox ExampleC#

```
protected void btndisplay_Click(object sender, EventArgs e)
{
TextBox1.Text = "Asp.net";
}
protected void btnclear_Click(object sender, EventArgs e)
{
TextBox1.Text = "";
}
```

Q9. What is the difference between buttons, Link Buttons and Image Buttons? Explain any three common button attributes.

These controls differ only in how they appear to the user. A button displays text within a rectangular area. A Link button displays text that look like a hyperlink and an image button displays an image.

LinkButton control is a control just like a Button control along with the flexibility to make it look like a Hyperlink. It implements an anchor <a/> tag that uses only ASP.NET postback mechanism to post the data on the server. Despite being a hyperlink, we can't specify the target URL.

ImageButton control is used to post the form or fire an event either client side or server side. It's like a asp:Button control, the only difference is, we have the ability to place our own image as a

button. ImageButton control is generally used to post the form or fire an event either client side or server side.

Button control is used to post the form or fire an event either client side or server side. Button control is generally used to post the form or fire an event either client side or server side. When it is rendered on the page, it is generally implemented through <input type=submit> HTML tag.

Common button attributes

Attribute	Description
Text	(Button and LinkButton controls only) The text displayed by the button. For a LinkButton control, the text can be coded as content between the start and end tags or as the value of the Text attribute.
ImageUrl	(ImageButton control only) The image to be displayed for the button.
AlternateText	(ImageButton control only) The text to be displayed if the browser can't display the image.
CausesValidation	Determines whether page validation occurs when you click the button. The default is True.
CommandName	A string value that's passed to the Command event when a user clicks the button.
CommandArgument	A string value that's passed to the Command event when a user clicks the button.
PostBackUrl	The URL of the page that should be requested when the user clicks the button.

Q10. Explain CheckBox and RadioButton web server controls in ASP.NET.

CheckBox:
CheckBox control is an asp.net web server control. CheckBox control visually as square on web forms. The Checkbox control allow user to check square or uncheck square.
In CheckBox control check and uncheck checkbox specify by the checked property of check box true or false.
Basic syntax for check box:

```
<asp:CheckBox ID= "chkoption" runat= "Server"></asp:CheckBox>
```

CheckBox properties:

- **AutoPostBack:**Specifies whether the form should be posted immediately after the Checked property has changed or not. Default is false
- **CausesValidation:**Specifies if a page is validated when a Button control is clicked
- **Checked:**Specifies whether the check box is checked or not
- **Id:**A unique id for the control
- **Text: The** text next to the check box

Event:

OnCheckedChanged
The name of the function to be executed when the checked property has changed

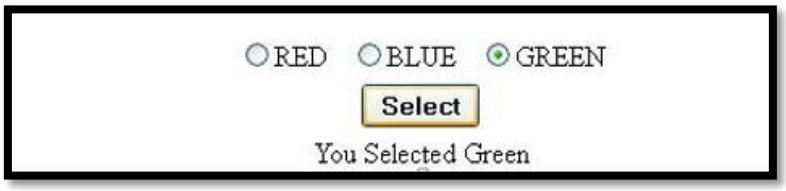
RadioButton
Radiobutton isasp.net web server control. Radiobutton is used to allow user to select a single radiobutton from group of radiobutton. In asp.net generally we use more than one radiobutton and select only one radiobutton at a time from all the radiobutton control. On other hand in checkbox control we can check and uncheck multiple check box at a time.

Basic syntax for radio button:

```
<asp:RadioButton ID= "rdboption" runat= "Server"></asp: RadioButton>
```

RadioButton properties:

- **AutoPostBack:**A Boolean value that specifies whether the form should be posted immediately after the Checked property has changed or not. Default is false
- **Checked** : A Boolean value that specifies whether the radio button is checked or not **id** a unique id for the control
- **GroupName** : The name of the group to which this radio button belongs
- **Text** : The text next to the radio button



Q11. Explain Listbox with properties and methods.

ListBox control is an asp.net web server control. ListBox control used to store the multiple items and allow user to select multiple item from ListBox control. ListBox control is same as dropdownlist control.

The dropdownlist control allow user to select maximum only one item at a time, on other hand ListBox control allow user to select multiple items same time. So we can also say ListBox is multi-row selection box control.

In a ListBox control there is a SelectionMode property to change the mode of section from single to multiple. By default ListBox control selection mode is single if we want to select multiple items from ListBox, then just change the SelectionMode property to multiple.

```
<asp:ListBox ID="dropStatic" runat="server">
<asp:ListItem Text="Red" Value="red"></asp:ListItem>
<asp:ListItem Text="Blue" Value="blue"></asp:ListItem>
<asp:ListItem Text="Green" Value="green"></asp:ListItem>
</asp:ListBox>
```

Some properties are

Rows	No. of rows (items) can be set to display in the List.
SelectionMode	Single or Multiple. If multiple, it allows user to select multiple items from the list by holding Ctrl or Shift key.
SelectedValue	Get the value of the Selected item from the dropdown box.
SelectedIndex	Gets or Sets the index of the selected item in the dropdown box.
SelectedItem	Gets the selected item from the list.
Items	Gets the collection of items from the dropdown box.

Following are the important methods and events:

GetSelectedIndices

- Gets the array of index values for currently selected items in the ListBox control.
- public virtual int[] GetSelectedIndices()
- Say for example, if ID of the list box is 'ListBox1' then the following code displays all selected items from the list box to a label with ID 'Label1'.

```
int[] x = ListBox1.GetSelectedIndices();
Label1.Text = "";
for (int i = 0; i < x.Length; i++)
Label1.Text += ListBox1.Items[x[i]].Text + "<br>";
```

SelectedIndexChanged

SelectedIndexChanged event occurs when the selection from the list control changes between posts to the server.

OnSelectedIndexChanged

- OnSelectedIndexChanged method raises the SelectedIndexChanged event.
- This allows you to provide a custom handler for the event.

Q12. What is the difference between List Box and Drop-Down Lists? List and explain any three common properties of these controls.

- List boxes are used in cases where there are small numbers of items to be selected. In contrast, drop-down lists are typically used with larger list so that they don't take up much space on the page.
- A list box lets a user select one or more items from the list of items. A drop-down list lets a user choose an item from the drop-down list of items.

Following are common properties of ListBox and DropDownList:

Properties	Description
Items	Gets the collection of items from the dropdown box.
SelectedValue	Get the value of the Selected item from the dropdown box.
SelectedIndex	Gets or Sets the index of the selected item in the dropdown box.
SelectedItem	Gets the selected item from the list.

Q13. Explain any five Methods/Property of List Item collection objects.

Following is list of Methods/Property of List Item collection objects:

Property	Description
`Count`	The number of items in the collection.
Indexer	**Description**
`[integer]`	A ListItem object that represents the item at the specified index.
Method	**Description**
`Add(string)`	Adds a new item to the end of the collection, and assigns the specified string value to both the Text and Value properties of the item.
`Add(ListItem)`	Adds the specified list item to the end of the collection.
`Insert(integer, string)`	Inserts an item at the specified index location in the collection, and assigns the specified string value to the Text property of the item.
`Insert(integer, ListItem)`	Inserts the specified list item at the specified index location in the collection.
`Remove(string)`	Removes the item from the collection whose Text property is equal to the specified string value.
`Remove(ListItem)`	Removes the specified list item from the collection.
`RemoveAt(integer)`	Removes the item at the specified index location from the collection.
`Clear()`	Removes all the items from the collection.
`FindByValue(string)`	Returns the list item whose Value property has the specified value.
`FindByText(string)`	Returns the list item whose Text property has the specified value.

Q14. Explain Sorted, SelectedMode, MultiColumn, SelectedItem and SelectedIndex properties of ListBox control.

Sorted

The Sorted property set to true, the ListBox items are sorted. The following code snippet sorts the ListBox items.

```
listBox1.Sorted = true;
```

Selected Mode:

SelectionMode property defines how items are selected in a ListBox. The SelectionMode value can be one of the following four SelectionMode enumeration values. **None** - No item can be selected. **One** - Only one item can be selected. **MultiSimple** - Multiple items can be selected. **MultiExtended** - Multiple items can be selected, and the user can use the SHIFT, CTRL, and arrow keys to make selections.

```
listBox1.SelectionMode = SelectionMode.MultiSimple;
```

```
listBox1.SetSelected(1, true);
listBox1.SetSelected(2, true);
```

MultiColumn:

A multicolumn ListBox places items into as many columns as are needed to make vertical scrolling unnecessary. The user can use the keyboard to navigate to columns that are not currently visible. Set the HorizontalScrollbar property to true to display a horizontal scroll bar that enables the user to scroll to columns that are not currently shown in the visible region of the ListBox. The value of the ColumnWidth property determines the width of each column.

```
listBox1.MultiColumn = true;
```

SelectedItem

Gets or sets the currently selected item in the ListBox. We can get text associated with currently selected item by using Items property.

```
string selectedItem = listBox1.Items[listBox1.SelectedIndex].ToString();
```

SelectedIndex

Gets or sets the zero-based index of the currently selected item in a ListBox.

```
listBox1.SelectedIndex = 1;
```

Q15. Explain Table control with example in ASP.NET

Table control is used to structure a web pages. In other words to divide a page into several rows and columns to arrange the information or images. When it is rendered on the page, it is implemented through <table> HTML tag.

We can simply use HTML <table> control instead of using **<asp:Table>** control. However many of one benefits of using <**asp:Table>** control is we can dynamically add rows or columns at the runtime or change the appearance of the table.
Following are some important **properties** that are very useful.

BackImageUrl

Used to set background image of the table

Caption

Used to write the caption of the table.

Example:

ASP.NET code for a table control

```
<asp:Table ID="Table1" runat="server" Height="123px" Width="567px">
    <asp:TableRow runat="server">
        <asp:TableCell runat="server"></asp:TableCell>
        <asp:TableCell runat="server"></asp:TableCell>
        <asp:TableCell runat="server"></asp:TableCell>
    </asp:TableRow>
    <asp:TableRow runat="server">
        <asp:TableCell runat="server"></asp:TableCell>
        <asp:TableCell runat="server"></asp:TableCell>
        <asp:TableCell runat="server"></asp:TableCell>
    </asp:TableRow>
```

```
        <asp:TableRow runat="server">
            <asp:TableCell runat="server"></asp:TableCell>
            <asp:TableCell runat="server"></asp:TableCell>
            <asp:TableCell runat="server"></asp:TableCell>
        </asp:TableRow>
    </asp:Table>
```

Q16. Explain Multiview and View control with example in ASP.NET

MultiView Control is an asp.net web server control. Multiview Control is a same as Tab control. If we want make a complex designing on one web forms then you have to use multiview control.

MltiView control is a container of several view control. In a multiview control there are many view control for designing separation of view for user.

The View control is a container of several web server controls. Using the multiview control we can feel the multiple page view design on a single web page.

We can also say multiview control allow user to create different view on single web page. There are many view control in a multiview control, each view control has some web server control for design web page. We can see single view at a time on web page by specify the **ActiveViewIndex** property of multiview control.

All view control assign automatically index to all it, the index always start from zero. The first view1 index is zero, second is one and so on, If we want to display first view on web page, then we need to write **MultiView1.ActiveViewIndex=0**.

- Prefix for MultiView control is mv
- Prefix for View control is v

The MultiView control has the following important properties:

Properties	Description
Views	Collection of View controls within the MultiView.
ActiveViewIndex	A zero based index that denotes the active view. If no view is active, then the index is -1.

MultiView control

```
01. <asp:MultView ID= "mvEntryForm" runat= "server">
02.    <%--[View control codes come inside of MultiView]--%>
03. </asp:MultiView>
```

View Control

```
01. <asp:View ID= "vAddressDetail" runat= "server">
02. </asp:View>
```

Note

View control is used inside of MultiView contro and looks like this way:

```
01. <asp:MultView ID= "mvEntryForm" runat= "server">
02.    <asp:View ID= "vAddressDetail" runat= "server"> </asp:View>
03. </asp:MultiView>
```

Q17. Explain Adrotator control with example in ASP.NET

The AdRotator is one of the rich web server control of asp.net. AdRotator control is used to display a sequence of advertisement images as per given priority of image.

Adrotator control display the sequence of images, which is specified in the external XML file. In xml file we indicate the images to display with some other attributes, like image impressions, NavigateUrl, ImageUrl, AlternateText.

In Adrotator control images will be changed each time while refreshing the web page.

AdRotator Control Example in ASP.Net

Step 1 – Open the Visual Studio –> Create a new empty Web application.

Step 2 – Create a New web page for display AdRotator control.

step 3 – Drag and drop AdRotator Control on web page from toolbox.

step 4 – go to **Add New Item** –> Add New **XML File** in project for write advertisement detail.

step 5 – Assign XML File to **AdvertisementFile** Property of AdRotator control.

The basic syntax of adding an AdRotator is as follows:

```
<asp:AdRotator  runat = "server" AdvertisementFile = "adfile.xml"  />
```

The advertisement file is an XML file, which contains the information about the advertisements to be displayed.

There are the following standard XML elements that are commonly used in the advertisement file:

Element	Description
Advertisements	Encloses the advertisement file.
Ad	Delineates separate ad.
ImageUrl	The path of image that will be displayed.
NavigateUrl	The link that will be followed when the user clicks the ad.
AlternateText	The text that will be displayed instead of the picture if it cannot be displayed.
Keyword	Keyword identifying a group of advertisements. This is used for filtering.
Impressions	The number indicating how often an advertisement will appear.
Height	Height of the image to be displayed.
Width	Width of the image to be displayed.

The following code illustrates an advertisement file ads.xml:

```
<Advertisements>
<Ad>
<ImageUrl>rose1.jpg</ImageUrl>
<NavigateUrl>http://www.1800flowers.com</NavigateUrl>
<AlternateText>
      Order flowers, roses, gifts and more
</AlternateText>
<Impressions>20</Impressions>
<Keyword>flowers</Keyword>
</Ad>
<Ad>
<ImageUrl>rose2.jpg</ImageUrl>
<NavigateUrl>http://www.babybouquets.com.au</NavigateUrl>
<AlternateText>Order roses and flowers</AlternateText>
<Impressions>20</Impressions>
<Keyword>gifts</Keyword>
</Ad>
<Ad>
<ImageUrl>rose3.jpg</ImageUrl>
<NavigateUrl>http://www.flowers2moscow.com</NavigateUrl>
<AlternateText>Send flowers to Russia</AlternateText>
<Impressions>20</Impressions>
<Keyword>russia</Keyword>
</Ad>
</Advertisements>
```

Create a new web page and place an AdRotator control on it.

```
<form id="form1" runat="server">
<div>
<asp:AdRotator ID="AdRotator1" runat="server" AdvertisementFile  ="~/ads.xml"
onadcreated="AdRotator1_AdCreated" />
</div>
</form>
```

Q18. Explain Calendar control with example in ASP.NET

ASP.NET provides a Calendar control that is used to display a calendar on the Web page. This control displays a one-month calendar that allows the user to select dates and move to the next and previous months.

By default, this control displays the name of the current month, day headings for the days of the weeks, days of the month and arrow characters for navigation to the previous or next month. The class hierarchy for this control is as follows

Object->Control->WebControl->Calendar

The Calendar is complex, powerful Web server control that you can use to add calendar feature to our web page. We can use calendar control display any date between 0 A.D. and 9999A.D.The Calendar control is represented as:

```
<asp:Calendar ID="Calendar1" runat="server"></asp:Calendar>
```

The Calendar control when rendered to a user browser, it generates an HTML <table> element and a set of associated JavaScript. The Calendar control can be used to select a single date or multiple dates. The **SelectionMode** property is used for this. The **SelectionMode** properties are as:

Property	Description
Day	Allow selection of a single date.
DayWeek	Allows the selection of a single date or a complete week.
DayWeekMonth	Allow selection of single date, complete week or complete month.
None	Doesn't allow you to select any date.

Q19. Explain any five common properties of web server controls.

Following are common properties of web server controls:

Property	Description
AccessKey	Enables you to set a key with which a control can be accessed at the client by pressing the associated letter.
BackColor, ForeColor	Enables you to change background color and text color of a control.
BorderColor	This property is used to change the border color of a control.
BorderStyle	Using this property border Style can be set to none, dotted, dashed, solid double, groove etc.
BorderWidth	Enables you to change border width of a control.
CssClass	Enables you to set the style sheet class for this control.
Enabled	Determines whether the control is enabled or not. If the control is disabled user cannot interact with it.
Font	Enables you to change the Font settings.
Height	Enables you to set the height of the control.
Width	Enables you to set the width of the control.
ToolTip	This property enables you to set a tooltip for the control in the browser and is rendered as a title attribute in the HTML, is shown when the user hovers the mouse over the control.
Visible	Determines whether the control is visible or not.

Q20. What is use of autopostback and runat properties?

- AutoPostback or Postback is nothing but submitting page to server. AutoPostback is webpage going to server, Server processes the values and sends back to same page or redirects to different page.
- HTML elements in ASP.NET files are, by default, treated as text. To make these elements programmable, add a runat="server" attribute to the HTML element. This attribute indicates that the element should be treated as a server control.
- The runat="server" attribute indicates that the form should be processed on the server. It also indicates that the enclosed controls can be accessed by server scripts.
- AutoPostBack means that if any change happens on a control by the client, it should postback to the server to handle that change at server side. If this property is set to TRUE the automatic post back is enabled, otherwise FALSE. Default is FALSE.

Example

<asp:TextBox id= " t1" AutoPostBack="true" runat="server" />

Web server controls are created on the server and they require a runat="server" attribute to work. This attribute indicates that the control should be treated as a server control.

Q21. Explain UrlEncode() and UrlDecode() methods in ASP.NET.

We cannot send special characters through query string. All special characters should be encoded when you pass them through the query string. The encoded string must be decoded at the receiver. There are two methods to achieve this: **UrlEncode and UrlDecode()**:

The main purpose of these two methods is to encode and decode the **URL**respectively. We need to encode the URL because some of the characters are not safe sending those across browser.
Some characters are being misunderstood by the browser and that leads to data mismatch on the receiving end. Characters such as a question mark (?), ampersand (&), slash mark (/), and spaces might be truncated or corrupted by some browsers.

UrlEncode()

This method is used to encode a string to be passed through URL to another web page. URLencode replaces unsafe ASCII characters with a "%" followed by two hexadecimal digits.

Syntax:

UrlEncode (string str)

UrlDecode()

This method is used to decode the encoded URL string. Decodes any %## encoding in the given string.

Syntax:

UrlDecode (string str)

Q22.What is user control? How to create and use user control in ASP.NET Page.

A UserControl is a separate, reusable part of a page. We can put a piece of a page in a UserControl, and then reuse it from a different location. A notable difference is that User Controls can be included on multiple pages, while a page can't.

UserControls are used much like regular server controls, and they can be added to a page declaratively, just like server controls can.

A big advantage of the UserControl is that it can be cached, using the OutputCache functionality described in a previous chapter, so instead of caching an entire page, we may cache only the UserControl, so that the rest of the page is still re-loaded on each request.

Creation of UserControl:

Following steps are used to create UserControl.

1. Open Visual Studio.
2. "File" -> "New" -> "Project..." then select ASP.NET Webform Application.
3. Add a new web form.

To create a new UserControl, in Solution Expolrer, Add New Item, provide your File Name and click Add

Design UserControl as per our requirement. Next step to use UserControl in .aspx page. Add the following line below the standard page declaration:

```
<%@RegisterTagPrefix="My"TagName="UserInfoBoxControl"Src="~/UserInfoBoxCont
rol.ascx" %>
```

Make sure that the src value matches the path to your UserControl file. Now you may use the UserControl in your page, like any other control. For instance, like this:

```
<My:UserInfoBoxControlrunat="server"ID="MyUserInfoBoxControl"/>
```

Q23.Why we use validation controls? List various types of controls used in asp.net

Validation is important part of any web application. User's input must always be validated before sending across different layers of the application.

Validation controls are used to:

- Implement presentation logic.
- To validate user input data.
- Data format, data type and data range is used for validation.
- Validation Controls in ASP.NET

An important aspect of creating ASP.NET Web pages for user input is to be able to check that the information users enter is valid. ASP.NET provides a set of validation controls that provide an easy-to-use but powerful way to check for errors and, if necessary, display messages to the user.

There are six types of validation controls in ASP.NET

- RequiredFieldValidation Control
- CompareValidator Control
- RangeValidator Control
- RegularExpressionValidator Control
- CustomValidator Control
- ValidationSummar

The below table describes the controls and their work:

Validation Control	Description
RequiredFieldValidation	Makes an input control a required field
CompareValidator	Compares the value of one input control to the value of another input control or to a fixed value
RangeValidator	Checks that the user enters a value that falls between two values
RegularExpressionValidator	Ensures that the value of an input control matches a specified pattern
CustomValidator	Allows you to write a method to handle the validation of the value entered
ValidationSummary	Displays a report of all validation errors occurred in a Web page

Q24.Explain following validation controls.

RequiredFieldValidator Control

The RequiredFieldValidator control is simple validation control, which checks to see if the data is entered for the input control. We can have a RequiredFieldValidator control for each form element on which you wish to enforce Mandatory Field rule.

The syntax of the control is as given:

<asp:RequiredFieldValidator ID="rfvcandidate" runat="server" ControlToValidate="txtName" ErrorMessage="Please enter name !!"></asp:RequiredFieldValidator>

RangeValidator Control

The RangeValidator Server Control is another validator control, which checks to see if a control value is within a valid range. The attributes that are necessary to this control are: MaximumValue, MinimumValue, and Type.

It has three specific properties:

Properties	Description
Type	It defines the type of the data. The available values are: Currency, Date, Double, Integer, and String.
MinimumValue	It specifies the minimum value of the range.
MaximumValue	It specifies the maximum value of the range.

The syntax of the control is as given:

<asp:RangeValidator ID="rvclass" runat="server" ControlToValidate="txtclass"

```
  ErrorMessage="Enter your class (6 - 12)" MaximumValue="12"
  MinimumValue="6" Type="Integer">
</asp:RangeValidator>
```

RegularExpressionValidator

The RegularExpressionValidator allows validating the input text by matching against a pattern of a regular expression. The regular expression is set in the ValidationExpression property.

The syntax of the control is as given:

```
<asp:RegularExpressionValidator ID="string" runat="server" ErrorMessage="string"
  ValidationExpression="string" ValidationGroup="string">
</asp:RegularExpressionValidator
```

Q25. What is the use of Compare Validator? Explain it along with its properties.

CompareValidator Control

- The CompareValidator control allows you to make comparison to compare data entered in an input control with a constant value or a value in a different control.
- It can most commonly be used when you need to confirm password entered by the user at the registration time. The data is always case sensitive.
- It has the following specific properties:

Properties	Description
Type	It specifies the data type.
ControlToCompare	It specifies the value of the input control to compare with.
ValueToCompare	It specifies the constant value to compare with.
Operator	It specifies the comparison operator, the available values are: Equal, NotEqual, GreaterThan, GreaterThanEqual, LessThan, LessThanEqual, and DataTypeCheck.

The basic syntax of the control is as follows:

```
<asp:CompareValidator ID="CompareValidator1" runat="server"
  ErrorMessage="CompareValidator">
</asp:CompareValidator>
```

Example:

```
Small number:<br />
<asp:TextBox runat="server" id="txtSmallNumber" /><br /><br />
Big number:<br />
<asp:TextBox runat="server" id="txtBigNumber" /><br />
<asp:CompareValidator runat="server" id="cmpNumbers" controltovalidate="txtSmallNumber"
controltocompare="txtBigNumber" operator="LessThan" type="Integer" errormessage="The first
number should be smaller than the second number!" /><br />
```

Q26. Explain the CustomValidator control with code

The CustomValidator control allows writing application specific custom validation routines for both the client side and the server-side validation.

The client-side validation is accomplished through the ClientValidationFunction property. The client-side validation routine should be written in a scripting language, such as JavaScript or VBScript, which the browser can understand.

The server-side validation routine must be called from the control's ServerValidate event handler. The server-side validation routine should be written in any .Net language, like C# or VB.Net.

Below table shows the properties of CustomValidator.

Property	Description
ControlToValidate	The id of the control to validate
Display	• None (the control is not displayed and error message is shown in the ValidationSummary control) • Static (in case of failure error message is shown in the space reserved). • Dynamic (In case of failure error message is shown. Space is not reserved.)
Enabled	A boolean value that specifies whether the validation control is enabled or not
Id	A unique id for the control
IsValid	A Boolean value that indicates whether the control specified by ControlToValidate is determined to be valid
OnServerValidate	Specifies the name of the server-side validation script function to be executed
Text	The message to display when validation fails

Example:

In this below example we will simply check the length of the string in the TextBox.

```
Custom Text:<br />
<asp:TextBox runat="server" id="txtCustom" />
<asp:CustomValidator runat="server" id="cusCustom" controltovalidate="txtCustom"
onservervalidate="cusCustom_ServerValidate" errormessage="The text must be exactly 8 characters
long!" />
<br /><br />
```

```
protected void cusCustom_ServerValidate(object sender, ServerValidateEventArgs e)
{
if(e.Value.Length == 8)
      e.IsValid = true;
else
      e.IsValid = false;
}
```

Q27. Explain Menu control in ASP.NET

The Menu control is used to create a menu of hierarchical data that can be used to navigate through the pages. The Menu control conceptually contains two types of items. First is StaticMenu that is always displayed on the page, second is DynamicMenu that appears when opens the parent item.

Following steps are used to create Menu Control:
Toolbox > Navigation > Menu or add the following few lines of code snippet.

```
<asp:Menu ID="Menu1" runat="server" Orientation="Horizontal">
<Items>
<asp:MenuItem Text="HOME" NavigateUrl="~/Default.aspx" Selected="true" />
<asp:MenuItem Text="HTML" NavigateUrl="~/HTML.aspx">
<asp:MenuItem Text="HtmlTagList" NavigateUrl="~/HtmlTagList.aspx" />
</asp:MenuItem>
```

```
<asp:MenuItem Text="CSS" NavigateUrl="~/CSS.aspx">
<asp:MenuItem Text="CssSelectors" NavigateUrl="~/CssSelectors.aspx" />
</asp:MenuItem>
</Items>
</asp:Menu>
```

Below are the various Properties of Menu Control

Properties of Menu Control	
DataSourceID	Indicates the data source to be used (You can use .sitemap file as datasource).
Text	Indicates the text to display in the menu.
Tooltip	Indicates the tooltip of the menu item when you mouse over.
Value	Indicates the nondisplayed value (usually unique id to use in server side events)
NavigateUrl	Indicates the target location to send the user when menu item is clicked. If not set you can handle MenuItemClick event to decide what to do.
Target	If NavigationUrl property is set, it indicates where to open the target location (in new window or same window).
Selectable	true/false. If false, this item can't be selected. Usually in case of this item has some child.
ImageUrl	Indicates the image that appears next to the menu item.
ImageToolTip	Indicates the tooltip text to display for image next to the item.
PopOutImageUrl	Inidcates the image that is displayed right to the menu item when it has some subitems.
Target	If NavigationUrl property is set, it indicates where to open the target location (in new window or same window).

Q28. When to use TreeView control in ASP.NET

- ASP.NET TreeView Web control is used to display hierarchical data (such as a table of contents) in a tree structure in a Web page.
- The TreeView control is made up of TreeNode objects. The TreeView control can be bound to data.

It supports the following features:

- Automatic data binding, this allows the nodes of the control to be bound to hierarchical data, such as an XML document.
- Site navigation support through integration with the SiteMapDataSource control.
- Node text that can be displayed as either selectable text or hyperlinks.
- Customizable appearance through themes, user-defined images, and styles.
- Programmatic access to the TreeView object model, which allows you to dynamically create trees, populates nodes, set properties, and so on.
- Node population through client-side callbacks to the server (on supported browsers).
- The ability to display a check box next to each node.

```
<asp:TreeView ExpandDepth="1" runat="server">
<Nodes>
<asp:TreeNode Text="Employees">
<asp:TreeNode Text="Bradley" Value="ID-1234" />
<asp:TreeNode Text="Whitney" Value="ID-5678" />
<asp:TreeNode Text="Barbara" Value="ID-9101" />
</asp:TreeNode>
```

```
</Nodes>
</asp:TreeView>
```

- Each node in the Tree is represented by a name/value pair (not necessarily unique), defined by the Text and Value properties of TreeNode, respectively. The text of a node is rendered, whereas the value of a node is not rendered and is typically used as additional data for handling postback events.
- This example also uses the ExpandDepth property of TreeView to automatically expand the tree 1 level deep when it is first rendered.
- The TreeView control is made up of one or more nodes. Each entry in the tree is called a node and is represented by a TreeNode

Q29. Explain SiteMapPath control in ASP.NET

- The SiteMapPath control basically is used to access web pages of the website from one webpage to another. It is a navigation control and displays the map of the site related to its web pages.
- This map includes the pages in the particular website and displays the name of those pages. We can click on that particular page in the Site Map to navigate to that page. We can say that the SiteMapPath control displays links for connecting to URLs of other pages.
- The SiteMapPath control uses a property called SiteMapProvider for accessing data from databases and it stores the information in a data source.

The representation of the SiteMapPath control is as follows:

Root Node->Child Node

Following steps are used to create Menu Control:

Toolbox > Navigation >SiteMapPath

Following are some important **properties** that are very useful.

ParentLevelsDisplayed : It specifies the number of levels of parent nodes and then displays the control accordingly related to the currently displayed node.

RenderCurrentNodeAsLink : It specifies whether or not the site navigation node that represents the currently displayed page is rendered as a hyperlink.

PathSeperator : It specifies the string that displays the SiteMapPath nodes in the rendered navigation path.

Style properties of the SiteMapPath class

CurrentNodeStyle : It specifies the style used for the display text for the current node.

RootNodeStyle : It specifies the style for the root node style text.

NodeStyle : It specifies the style used for the display text for all nodes in the site navigation path.

Sitemap file has been included in our project and we can see it in the Solution Explorer. And now we have to set the URL and title attributes in the sitemap file.

```
<?xml version="1.0" encoding="utf-8" ?>
<siteMap xmlns="http://schemas.microsoft.com/AspNet/SiteMap-File-1.0" >
      <siteMapNode url="Default.aspx" title="myhomepage"  description="">
          <siteMapNode url="myweb1.aspx" title="myfirstpage"  description="" />
          <siteMapNode url="myweb2.aspx" title="mysecondpage"  description="" />
      </siteMapNode>
</siteMap>
```

Unit-III

Topics:

Error Handling, Logging, and Tracing |State Management |Styles, Themes, and Master Pages

Q1. Explain Error Handling in ASP.NET

Error handling in ASP.NET has three aspects:

Tracing - Tracing the program execution at page level or application level.

Error handling - Handling standard errors or custom errors at page level or application level.

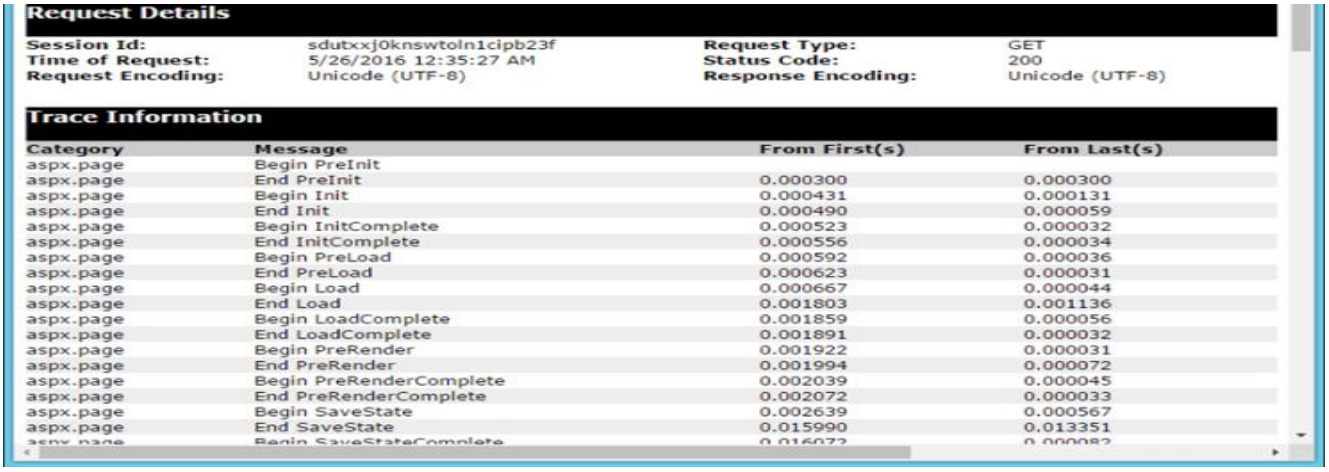
Debugging - Stepping through the program, setting break points to analyze the code.

Tracing

To enable page level tracing, you need to modify the Page directive and add a Trace attribute as shown:

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

Now when we execute the file, we get the tracing information:



Request Details

Session Id:	sdutxxj0knswtoln1cipb23f	**Request Type:**	GET
Time of Request:	5/26/2016 12:35:27 AM	**Status Code:**	200
Request Encoding:	Unicode (UTF-8)	**Response Encoding:**	Unicode (UTF-8)

Trace Information

Category	Message	From First(s)	From Last(s)
aspx.page	Begin PreInit		
aspx.page	End PreInit	0.000300	0.000300
aspx.page	Begin Init	0.000431	0.000131
aspx.page	End Init	0.000490	0.000059
aspx.page	Begin InitComplete	0.000523	0.000032
aspx.page	End InitComplete	0.000556	0.000034
aspx.page	Begin PreLoad	0.000592	0.000036
aspx.page	End PreLoad	0.000623	0.000031
aspx.page	Begin Load	0.000667	0.000044
aspx.page	End Load	0.001803	0.001136
aspx.page	Begin LoadComplete	0.001859	0.000056
aspx.page	End LoadComplete	0.001891	0.000032
aspx.page	Begin PreRender	0.001922	0.000031
aspx.page	End PreRender	0.001994	0.000072
aspx.page	Begin PreRenderComplete	0.002039	0.000045
aspx.page	End PreRenderComplete	0.002072	0.000033
aspx.page	Begin SaveState	0.002639	0.000567
aspx.page	End SaveState	0.015990	0.013351

It provides the following information at the top:

- Session ID
- Status Code
- Time of Request
- Type of Request
- Request and Response Encoding

The status code sent from the server, each time the page is requested shows the name and time of error if any. The following table shows the common HTTP status codes:

The class of a status code can be quickly identified by its first digit:

1xx: Informational

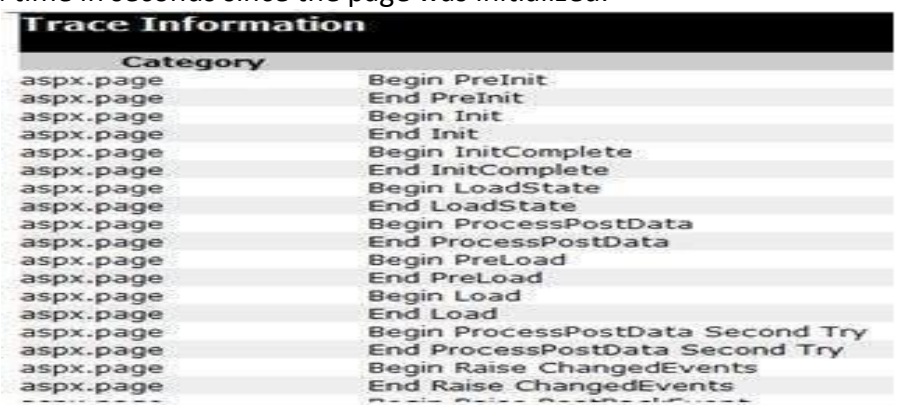
2xx: Success

3xx: Redirection

4xx: Client Error

5xx: Server Error

Under the top level information, there is Trace log, which provides details of page life cycle. It provides elapsed time in seconds since the page was initialized.



Trace Information

Category	
aspx.page	Begin PreInit
aspx.page	End PreInit
aspx.page	Begin Init
aspx.page	End Init
aspx.page	Begin InitComplete
aspx.page	End InitComplete
aspx.page	Begin LoadState
aspx.page	End LoadState
aspx.page	Begin ProcessPostData
aspx.page	End ProcessPostData
aspx.page	Begin PreLoad
aspx.page	End PreLoad
aspx.page	Begin Load
aspx.page	End Load
aspx.page	Begin ProcessPostData Second Try
aspx.page	End ProcessPostData Second Try
aspx.page	Begin Raise ChangedEvents
aspx.page	End Raise ChangedEvents

To check the Warn method, let us forcibly enter some erroneous code in the selected index changed event handler:

```
try
{
int a = 0;
int b = 9 / a;
}
catch (Exception e)
{
Trace.Warn("UserAction", "processing 9/a", e);
}
```

Try-Catch is a C# programming construct. The try block holds any code that may or may not produce error and the catch block catches the error. When the program is run, it sends the warning in the trace log.

Application level tracing applies to all the pages in the web site. It is implemented by putting the following code lines in the web.config file:

```
<system.web>
<trace enabled="true" />
</system.web>
```

Error Handling

Although ASP.NET can detect all runtime errors, still some subtle errors may still be there. Observing the errors by tracing is meant for the developers, not for the users.

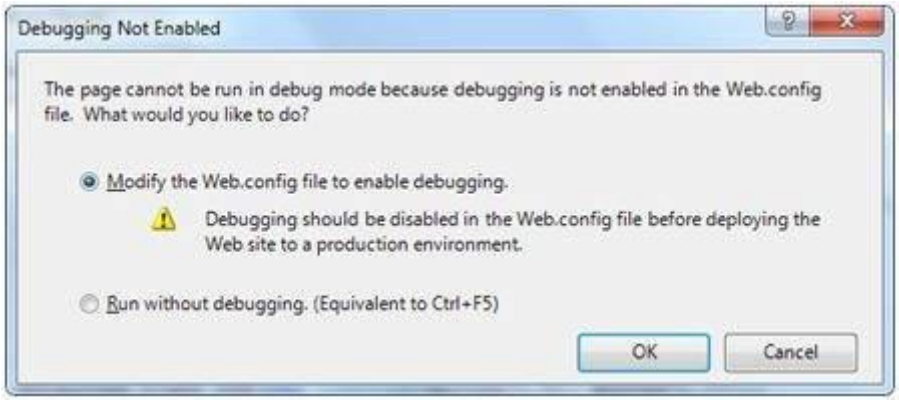
Hence, to intercept such occurrence, we can add error handing settings in the web.config file of the application. It is application-wide error handling. For example, we can add the following lines in the web.config file:

```
<configuration>
<system.web>

<customErrors mode="RemoteOnly" defaultRedirect="GenericErrorPage.htm">
<error statusCode="403" redirect="NoAccess.htm"     />
<error statusCode="404" redirect="FileNotFound.htm" />
</customErrors>

</system.web>
<configuration>
```

Q2. What is Debugging? Explain in ASP.NET

Debugging allows the developers to see how the code works in a step-by-step manner, how the values of the variables change, how the objects are created and destroyed, etc.

When the site is executed for the first time, Visual Studio displays a prompt asking whether it should be enabled for debugging:



When debugging is enabled, the following lines of codes are shown in the web.config:

```
<system.web>
<compilation debug="true">
<assemblies>
    ..............
</assemblies>
</compilation>
</system.web>
```

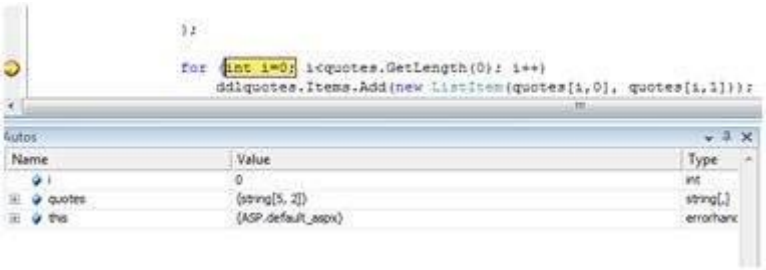
The Debug toolbar provides all the tools available for debugging:

Breakpoints

Breakpoints specifies the runtime to run a specific line of code and then stop execution so that the code could be examined and perform various debugging jobs such as, changing the value of the variables, step through the codes, moving in and out of functions and methods etc.

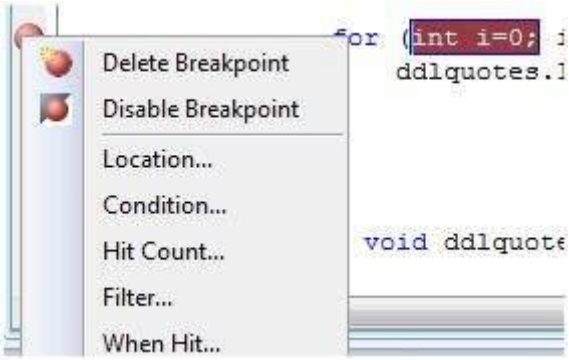
To set a breakpoint, right click on the code and choose insert break point. A red dot appears on the left margin and the line of code is highlighted as shown:

Next when you execute the code, you can observe its behavior.



At this stage, you can step through the code, observe the execution flow and examine the value of the variables, properties, objects, etc.

We can modify the properties of the breakpoint from the Properties menu obtained by right clicking the breakpoint glyph:



Q3. Why exception handling is required? Write syntax for user define exception?

Exception handling:

The mechanism of Exception Handling is throwing an exception and catching it C# uses try-catch block. Code which may give rise to exceptions is enclosed in a try block, and

Catch block catches that exception and handles it appropriately. The try block is followed by one or more catch blocks.

Basic syntax:

```
try
{
//programming logic(code which may give rise to exceptions)
}
catch (Exception e)
{
//message on exception
}
finally
{
// always executes
}
```

Try:A try block identifies a block of code for which particular exceptions will be activated. It's followed by one or more catch blocks.

Catch:A program catches an exception with an exception handler at the place in a program where you want to handle the problem. The catch keyword indicates the catching of an exception.

Finally: The finally block is used to execute a given set of statements, whether an exception is thrown or not thrown. For example, if you open a file, it must be closed whether an exception is raised or not.

Example:

```
using System;
class tryCatch
{
public static void Main()
{
int k=0;
try
{
int n= 10/k;
Console.WriteLine(”n=” + n);
}
catch(Exception e)
{
Console .WriteLine (“Division By zero exception”);
}
Console.WriteLtne(”Statement executed after Exception because of try catch”);
}
```

Q4. What is user-defined exception? Explain with example.

We have seen built-in exception classes however, we often like to raise an exception when the business rule of our application gets violated. So, for this we can create a custom exception class by deriving Exception or ApplicationException class.

For example, create InvalidStudentNameException class in a school application, which does not allow any special character or numeric value in a name of any of the students.

```
class Student
{
   public int StudentID { get; set; }
   public string StudentName { get; set; }
}

class InvalidStudentNameException : Exception
{
   public InvalidStudentNameException() {}

   public InvalidStudentNameException(string name)
      : base(String.Format("Invalid Student Name: {0}", name)){ }
  }
class Program
{
   static void Main(string[] args)
   {
      Student newStudent = null;

      try
      {
         newStudent = new Student();
         newStudent.StudentName = "James007";

         ValidateStudent(newStudent);
      }
      catch(InvalidStudentNameException ex)
      {
         Console.WriteLine(ex.Message );
      }

      Console.ReadKey();
   }

   private static void ValidateStudent(Student std)
   {
      Regex regex = new Regex("^[a-zA-Z]+$");

      if (!regex.IsMatch(std.StudentName))
          throw new InvalidStudentNameException(std.StudentName);
   }
}
```

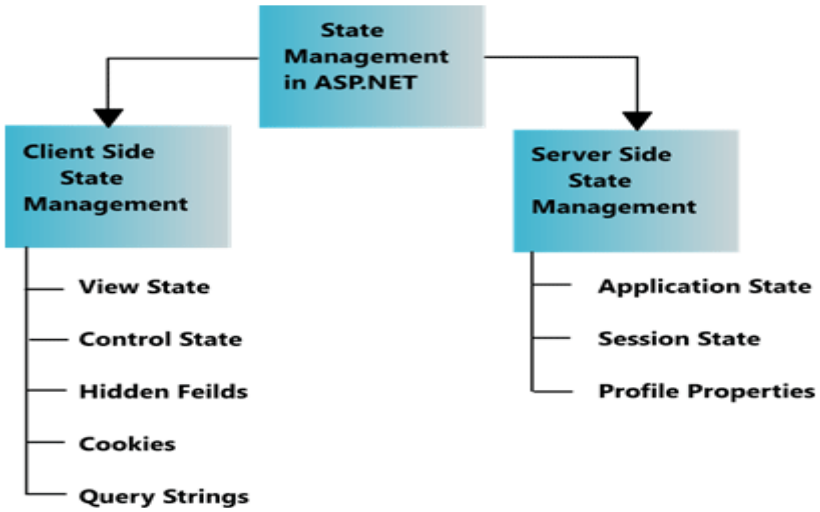
Q5. What is state management? Explain Application state management

A web application is stateless. That means that a new instance of a page is created every time when we make a request to the server to get the page and after the round trip our page has been lost immediately.

It only happens because of one server, all the controls of the Web Page are created and after the round trip the server destroys all the instances. So, to retain the values of the controls we use state management techniques.

State Management Techniques

They are classified into the following 2 categories:



Application State:

- If the information that we want to be accessed or stored globally throughout the application, even if multiple users access the site or application at the same time, then we can use an Application Object for such purposes.
- It stores information as a Dictionary Collection in key - value pairs. This value is accessible across the pages of the application / website.
- There are 3 events of the Application which are as follows

Application_Start
Application_Error
Application_End

Example - Just for an example, I am setting the Page title in the Application Start event of the Global.asax file.
Code for setting value to the Application Object - "PageTitle" is the Key and "Welcome to State Management Application" is the value.

```
void Application_Start(object sender, EventArgs e)
{
    this.Application["PageTitle"] = "Welcome to State Management Application";
}
```

Code for reading value from the Application Object

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!Page.IsPostBack)
    {
        this.Page.Title = Convert.ToString(this.Application["PageName"]);

    }
}
```

Q6. What is Query String? Explain encoding and decoding of Query string

- Query String is the most simple and efficient way of maintaining information across requests.
- The information we want to maintain will be sent along with the URL. A typical URL with a query string looks like **www.somewebsite.com/search.aspx?query=foo**
- The URL part which comes after the? Symbol is called a QueryString.

- QueryString has two parts, a key and a value. In the above example, **query** is the key and **foo** is its value.
- We can send multiple values through querystring, separated by the & symbol. The following code shows sending multiple values to the foo.aspx page.

  Response.Redirect("foo.aspx?id=1&name=foo");

- The following code shows reading the QueryString values in foo.aspx

  String id = Request.QueryString["id"];
  String name = Request.QueryString["name"];

- The **HtmlEncode()** method is particularly useful if you're retrieving values from a database and you aren't sure if the text is valid HTML.
- We can use the **HtmlDecode()** method to revert the text to its normal form if we need to perform additional operations or comparisons with it in your code.
- The **UrlEncode()** method changes text into a form that can be used in a URL, escaping spaces and other special characters. This step is usually performed with information we want to add to the query string.

  Label1.Text = Server.HtmlEncode("To bold text use the <b> tag.");

  Advantages

- Query string is lightweight and will not consume any server resources.
- It is very easy to use, and it is the most efficient state management technique. However, it has many

  Disadvantages

- We can pass information only as a string.
- URL length has limitations. So, we can't send much information through URL.
- Information passed is clearly visible to everyone and can be easily altered.

Q7. Role of Cookies in ASP.NET

- A cookie is a small piece of information stored on the client machine. This file is located on client machines "C:\Document and Settings\Currently_Login user\Cookie" path.
- It is used to store user preference information like Username, Password, City and Phone No etc. on client machines. We need to import namespace called Systen.Web.HttpCookie before we use cookie.

  Types of Cookies:

  Persist Cookie - A cookie has not had expired time which is called as Persist Cookie

  Non-Persist Cookie - A cookie has expired time which is called as Non-Persist Cookie

  Creation of cookies:

  It's really easy to create a cookie in the Asp.Net with help of Response object or HttpCookie

  Example 1:

```
HttpCookie userInfo = new HttpCookie("userInfo");
userInfo["UserName"] = "Annathurai";
userInfo["UserColor"] = "Black";
userInfo.Expires.Add(new TimeSpan(0, 1, 0));
Response.Cookies.Add(userInfo);
```

  Example 2:

```
Response.Cookies["userName"].Value = "Annathurai";
```

```
Response.Cookies["userColor"].Value = "Black";
```

Retrieve from cookie

Its easy way to retrieve cookie value form cookes by help of Request object.

Example 1:

```
string User_Name = string.Empty;
string User_Color = string.Empty;
User_Name = Request.Cookies["userName"].Value;
User_Color = Request.Cookies["userColor"].Value;
```

Example 2:

```
string User_name = string.Empty;
string User_color = string.Empty;
      HttpCookie reqCookies = Request.Cookies["userInfo"];
if (reqCookies != null)
      {
         User_name = reqCookies["UserName"].ToString();
         User_color = reqCookies["UserColor"].ToString();
      }
```

When we make request from client to web server, the web server process the request and give the lot of information with big pockets which will have Header information, Metadata, cookies etc., Then repose object can do all the things with browser.

Cookie's common property:

Domain: This is used to associate cookies to domain.
Secure: We can enable secure cookie to set true (HTTPs).
Value: We can manipulate individual cookie.
Values: We can manipulate cookies with key/value pair.
Expires: Which is used to set expire date for the cookies.

Advantages of Cookie:

- Its clear text so user can able to read it.
- We can store user preference information on the client machine.
- Its easy way to maintain.
- Fast accessing.

Disadvantages of Cookie

- If user clears cookie information we can't get it back.
- No security.
- Each request will have cookie information with page.

Q8. Write a program to create a new cookie with the name "Username" and add it to the HttpResponse object on the click of a button. Set the expiry date of the cookie to One year from Now.

```
protected void Button1_Click(object sender, EventArgs e)
{
HttpCookie Username = new HttpCookie("UserName", "WELCOME");
Username.Expires=DateTime.Now.AddYears(1);
Response.Cookies.Add(Username);
}
```

Q9. Significant of ViewState in ASP.NET

View State

- View State is the method to preserve the Value of the Page and Controls between round trips. It is a Page-Level State Management technique.
- View State is turned on by default and normally serializes the data in every control on the page regardless of whether it is actually used during a post-back.

Features of View State

These are the main features of view state:

- Retains the value of the Control after post-back without using a session.
- Stores the value of Pages and Control Properties defined in the page.
- Creates a custom View State Provider that lets you store View State Information in a SQL Server Database or in another data store.

Advantages of View State

- Easy to Implement.
- No server resources are required: The View State is contained in a structure within the page load.
- Enhanced security features: It can be encoded and compressed or Unicode implementation.

Disadvantages of View State

- **Security Risk:** The Information of View State can be seen in the page output source directly. We can manually encrypt and decrypt the contents of a Hidden Field, but It requires extra coding. If security is a concern, then consider using a Server-Based State Mechanism so that no sensitive information is sent to the client.
- **Performance:** Performance is not good if we use a large amount of data because View State is stored in the page itself and storing a large value can cause the page to be slow.
- **Device limitation:** Mobile Devices might not have the memory capacity to store a large amount of View State data.
- It can store values for the same page only.

Example:

If we want to add one variable in View State,
ViewState["Var"]=Count;
For Retrieving information from View State
string Test=ViewState["TestVal"];

Q10. Difference between Application state and Session state

- ASP.NET implements application state using the System.Web.HttpApplicationState class.
- It provides methods for storing information which can be accessed globally.
- Information stored on application state will be available for all the users using the website.
- Usage of application state is the same as sessions.

The following code shows storing a value in an application variable and reading from it.

```
Application["pageTitle"] = "Welcome to my website - ";
String pageTitle;
if (Application["pageTitle"] != null)
pageTitle = Application["pageTitle"].ToString();
```

We should get a basic knowledge about the events associated with application and session.

These events can be seen in the **global.asax** file.

- **Application_Start** This event executes when application initializes. This will execute when ASP.NET worker process recycles and starts again.
- **Application_End** Executes when the application ends.
- **Session_Start** Executes when a new session starts.
- **Session_End** Executes when session ends. Note: this event will be fired only if you are using InProc as session mode.

Example

The most common usage of application variables is to count the active number of visitors that are browsing currently.

The following code shows how this is done.

```
void Application_Start(object sender, EventArgs e)
{
// Application started - Initializing to 0
Application["activeVisitors"] = 0;
}
void Session_Start(object sender, EventArgs e)
{
if (Application["activeVisitors"] != null)
{
Application.Lock();
int visitorCount = (int)Application["activeVisitors"];
Application["activeVisitors"] = visitorCount++;
Application.UnLock();
}
}
```

- Data stored in session will be kept in server memory and it is protected as it will never get transmitted to a client.
- Every client that uses the application will have separate sessions. Session state is ideal for storing user specific information.

The following code shows storing a string value in session.

```
Session["name"] = "Value";
```

Values stored in sessions can be removed by several methods.
Session.Abandon() : Cancels the session and fires end event. This is used when you are done with the session.
Session.Clear() /Session.RemoveAll() : Clears all contents of the session. This will not end the session
Session.Remove(string) : Removes the session name supplied.

Working of Session:

- ASP.NET maintains a unique id which is called as "session id" for each session. This id is generated using a custom algorithm and it is unique always.
- Session id will be sent to the client as a cookie and the browser resends this upon each request.
- ASP.NET uses this session id to identify the session object.

```
string sessionId = Session.SessionID
```

Session State

- Session state is user and browser specific.

- Session state can be stored in memory on the server as well as client's cookies.
- If client has disabled cookies in his browser then session state will be stored in URL.
- Session state has scope to the current browser only. If we change the browser session id is changed.

Application State

- Application state is application specific.
- Application state is stored only in the memory on the server.
- Application state does not track client's cookies or URL.
- Application state has no scope to the current browser. If we change the browser application id remains same.

Q11. Explain the different types of CSS present in ASP.NET.

There are three types of CSS as follows:

- **External CSS**
- **Internal CSS or Embedded CSS**
- **Inline CSS**

External Style Sheet

The first way to add CSS style sheets to your web pages is through the **<link>** element that points to an external CSS file.

For example the following <link> shows what options you have when embedding a style sheet in your page:

```
<link href="StyleSheet.css" rel="Stylesheet" type="text/css" media="screen" />
```

The href property points to a file within our site when we create links between two pages. The rel and type attributes tell the browser that the linked file is in fact a cascading style sheet. The media attribute enables us to target different devices, including the screen, printer, and handheld devices. The default for the media attribute is screen, so it's OK.

Embedded style sheet

The second way to include style sheets is using embedded <style> elements. The <style> element should be placed at the top of your ASPX or HTML page, between the <head> tags.

For example, to change the appearance of an <h1> element in the current page alone, we can add the following code to the <head> of our page:

```
<head runat="server">
<style type="text/css">
h1
{
color: Blue;
}
</style>
</head>
```

Inline style sheet

The third way to apply CSS to your HTML elements is to use inline styles. Because the style attribute is already applied to a specific HTML element, we don't need a selector and we can write the declaration in the attribute directly:

```
<span style="color: White; background-color: Black ;">
This is white text on a black background.
</span>
```

Q12. Explain the four most important selectors present in CSS.

Universal Selector

The Universal selector, indicated by an asterisk (*), applies to all elements in your page. The Universal selector can be used to set global settings like a font family. The following rule set changes the font for all elements in our page to Arial:

```
*{
font-family: Arial;
 }
```

Type Selector

The Type selector enables us to point to an HTML element of a specific type. With a Type selector all HTML elements of that type will be styled accordingly.

```
h1
{
color: Green;
}
```

This Type selector now applies to all <h1> elements in your code and gives them a green color. Type Selectors are not case sensitive, so you can use both h1 and H1 to refer to the same heading.

ID Selector

The ID selector is always prefixed by a hash symbol (#) and enables us to refer to a single element in the page. Within an HTML or ASPX page, we can give an element a unique ID using the id attribute. With the ID selector, we can change the behavior for that single element, for example:

```
#IntroText
{
font-style: italic;

}
```

Because we can reuse this ID across multiple pages in our site (it only must be unique within a single page), you can use this rule to quickly change the appearance of an element that you use once per page, but more than once in our site, for example with the following HTML code:

I am italic because I have the right ID.

Class Selector

The Class selector enables us to style multiple HTML elements through the class attribute. This is handy when we want to give the same type of formatting to several unrelated HTML elements. The following rule changes the text to red and bold for all HTML elements that have their class attributes set to highlight:

```
.Highlight
{
font-weight: bold;
color: Red;
}
```

The following code snippet uses the Highlight class to make the contents of a element and a link (<a>) appear with a bold typeface:

```
This is normal text but <span class="Highlight">this is Red and Bold.</span>
This is also normal text but
<a href="CssDemo.aspx" class="Highlight">this link is Red and Bold as well.</a>
```

Q13. Explain <LINK> tag with example.

- The HTML <link> tag is used for defining a link to an external document. It is placed in the <head> section of the document.

- The <link> tag defines a link between a document and an external resource.
- The <link> tag is used to link to external style sheets.

Syntax:

<head><link rel="stylesheet" type="text/css" href="theme.css"></head

Example:

```
<html>
<head>
<title>HTML link Tag</title>
<link rel="stylesheet" type="text/css" href="default.css" />
</head>
<body>
<div>
<p>Welcome to our website. We provide tutorials on various subjects.</p>
</div>
</body>
</html>
```

Where,

- **rel**-can be used to specify the relationship of the target of the link to the current page.
- type-This attribute Provides information about the content type of the destination resource, telling wether it's an HTML document, a JPG image, an Excel document, etc.
- **href**(uri)-The "href" attribute specifies the destination resource, which the element is linking to. It may specify a resource in the same website or in an external one.

Q14. What is Theme? How to create and use theme in ASP.NET page.

A theme decides the look and feel of the website. It is a collection of files that define the looks of a page. It can include skin files, CSS files & images.

We define themes in a special **App_Themes** folder. Inside this folder is one or more subfolders named Theme1, Theme2 etc. that define the actual themes. The theme property is applied late in the page's life cycle, effectively overriding any customization we may have for individual controls on our page.

There are 3 different options to apply themes to our website:

1. Setting the theme at the page level: The Theme attribute is added to the page directive of the page.

```
<%@PageLanguage="C#"AutoEventWireup="true"CodeFile="Default.aspx.cs"Inherits="Default"Theme="Theme1"%>
```

2. Setting the theme at the site level: to set the theme for the entire website we can set the theme in the web.config of the website. Open the web.config file and locate the <pages> element and add the theme attribute to it:

```
<pagestheme="Theme1">
....
....
</pages>
```

3. Setting the theme programmatically at runtime: here the theme is set at runtime through coding. It should be applied earlier in the page's life cycle ie. Page_PreInit event should be handled for setting the theme. The better option is to apply this to the Base page class of the site as every page in the site inherits from this class.

```
Page.Theme = Theme1;
```

Uses of Themes

1. Since themes can contain CSS files, images and skins, you can change colors, fonts, positioning and images simply by applying the desired themes.
2. We can have as many themes as we want, and we can switch between them by setting a single attribute in the web.config file or an individual aspx page. Also, we can switch between themes programmatically.
3. Setting the themes programmatically, we are offering our users a quick and easy way to change the page to their likings.
4. Themes allow us to improve the usability of our site by giving users with vision problems the option to select a high contrast theme with a large font size.

Q15. What is Skin file? How to use it.

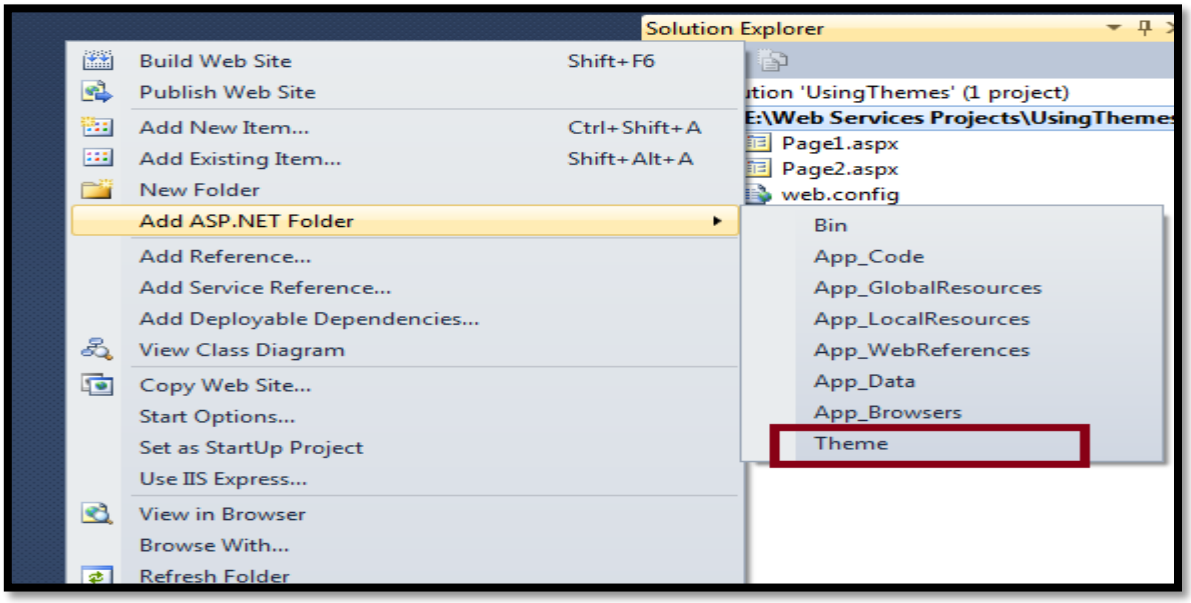
ASP.Net skins can only be used to apply the styles to the ASP.Net controls.so in this article let us see the procedure for using ASP.Net Skins.

First create the web application as in the following:

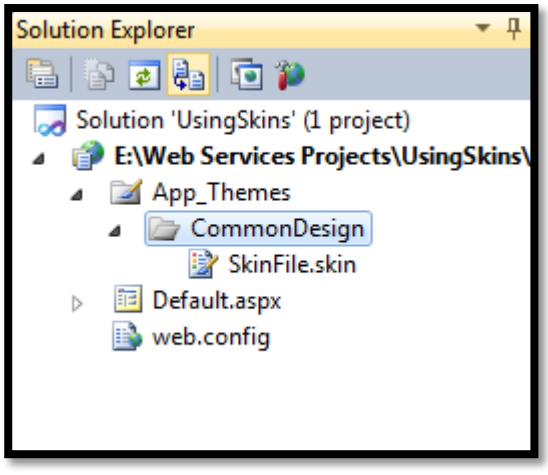
1. "Start" - "All Programs" - "Microsoft Visual Studio 2010"
2. "File" - "New Website" - "C# - Empty website" (to avoid adding a master page)
3. Provide the web site a name, such as UsingSkins or whatever you wish and specify the location
4. Then right-click on the solution in the Solution Explorer then select "Add New Item" - "Default.aspx page" .

Add an ASP.Net Themes Folder

To use the themes in the web site, we need to add an ASP.Net Themes folder by right-clicking on Solution Explorer as in the following:



After adding the theme folder, add the SkinFile.skin file by right-clicking on the ASP.Net theme folder. The Solution Explorer will then look as follows:



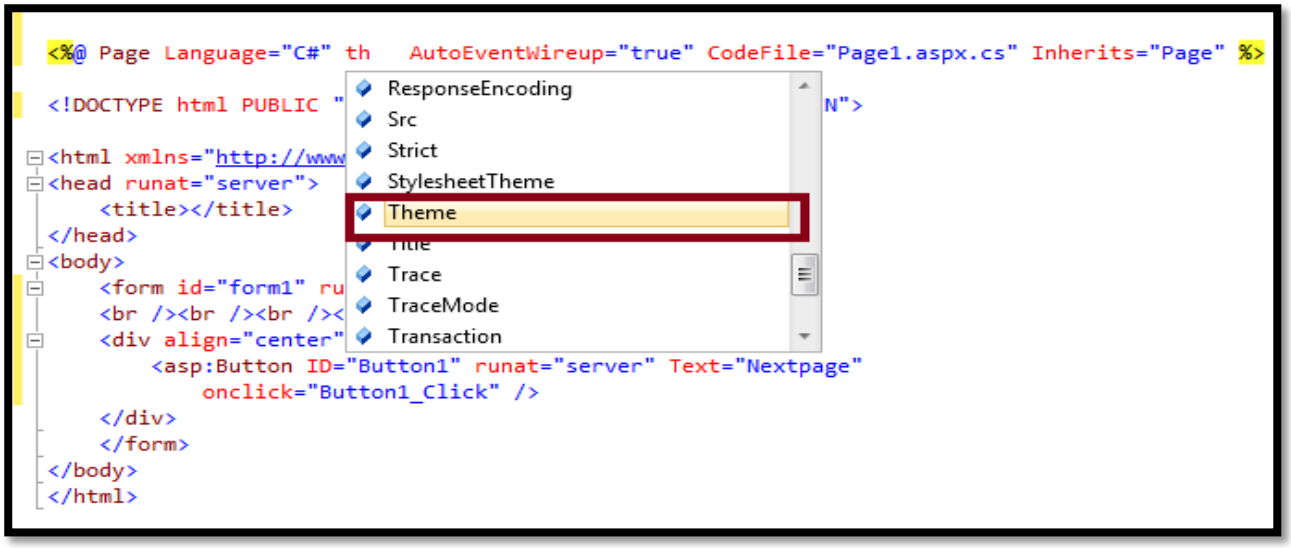
Now add the ASP.Net controls inside the SkinFile.Skin and assign the Style to the controls using

their properties as in the following:

```
<asp:Label  runat="server" ForeColor="Green" SkinID="lbltxt" Text="Label"></asp:Label>
<asp:TextBox  runat="server" ForeColor="DarkBlue" SkinID="txt"></asp:TextBox>
<asp:Button  runat="server" Text="Button"  ForeColor="Chocolate" SkinID="btn" />
```

1. A control Id cannot be assigned to ASP.Net controls inside the SkinFile.skin.
2. SkinId must be assigned to the ASP.Net controls inside the SkinFile.skin.
3. The SkinId should be uniquely defined because duplicate SkinId's per control type are not allowed in the same theme.
4. Only one default control skin per control type is allowed in the same theme.

To use existing ASP.Net Skins in an ASP.Net page we need to assign the existing theme at page level as in the following.



In the preceding source code, we are assigning the existing ASP.Net Skin File at page level, the existing ASP.Net Skin automatically appears in the box after using the Themes property in the page header.

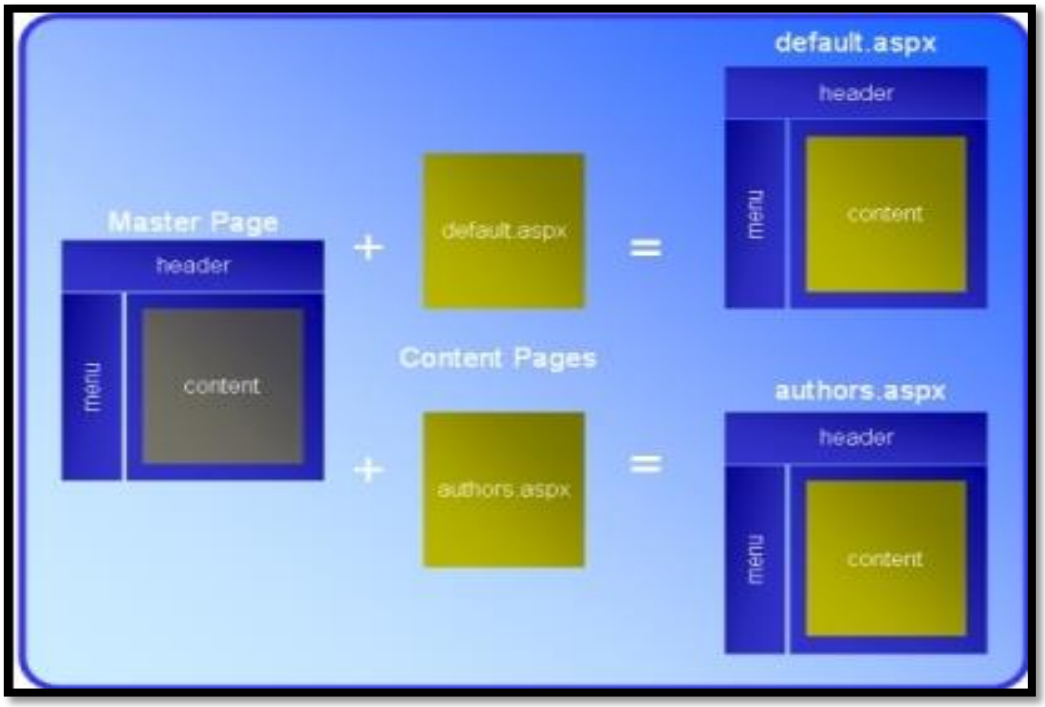
Assigning the Skin to the ASP.Net Controls

to assign the skin to the ASP.Net controls, you need to assign it to the control's SkinId Property as in the following:

```
<asp:Label ID="Label1" runat="server" SkinID="lbltxt" Text="Label"></asp:Label>
<asp:TextBox ID="TextBox1"  runat="server" SkinID="txt"></asp:TextBox>
<asp:Button ID="Button1" runat="server" SkinID="btn" Text="Button"/>
```

Q16. Role of Master Page in ASP.NET

- ASP.NET master pages allow us to create a consistent layout for the pages in our application.
- A single master page defines the look and feel and standard behavior that we want for all of the pages (or a group of pages) in our application.
- We can then create individual content pages that contain the content we want to display.
- When users request the content pages, they merge with the master page to produce output that combines the layout of the master page with the content from the content page.
- Master pages actually consist of two pieces, the master page itself and one or more content pages.



Use of Master Pages

The master pages can be used to accomplish the following:

Creating a set of controls that are common across all the web pages and attaching them to all the web pages.

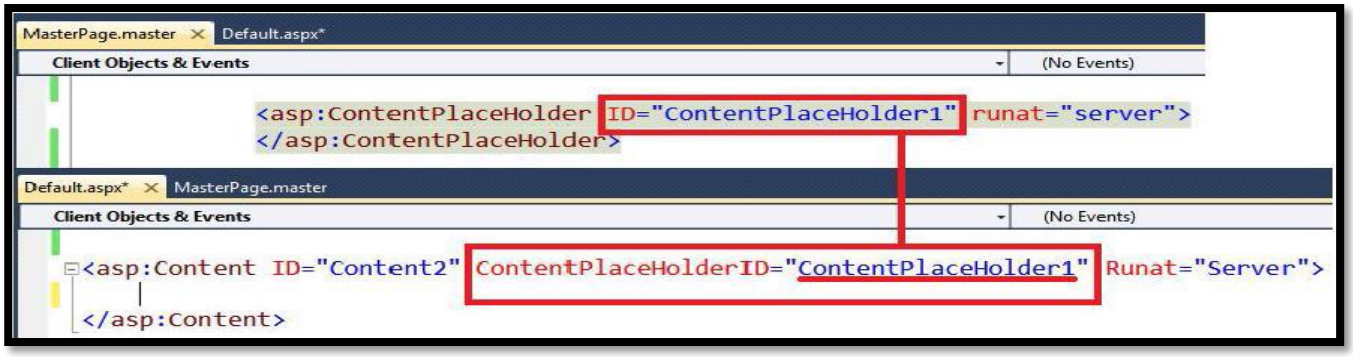
- A centralized way to change the above created set of controls which will effectively change all the web pages.
- To some extent, a master page looks like a normal ASPX page.
- It contains static HTML such as the <html>, <head>, and <body> elements, and it can also contain other HTML and ASP.NET server controls.
- Inside the master page, you set up the markup that you want to repeat on every page, like the general structure of the page and the menu.
- However, a master page is not a true ASPX page and cannot be requested in the browser directly it only serves as the template that real web pages called content pages
- One difference is that while web forms start with the Page directive, a master page starts with a Master directive that specifies the same information, as shown here

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeFile="MasterPage.master.cs" Inherits="MasterPage" %>
```

Q17. Explain relation between content page and master page

- Master page provides a framework (common content as well as the layout) within which the content from other pages can be displayed.
- It provides elements such as headers, footers, style definitions, or navigation bars that are common to all pages in your web site.
- So the Content Pages need not have to duplicate code for shared elements within your Web site.
- It gives a consistent look and feel for all pages in your application.
- The master page layout consists of regions where the content from each content page should be displayed.
- These regions can be set using ContentPlaceHolder server controls.
- These are the regions where you are going to have dynamic content in your page

- A derived page also known as a content page is simply a collection of blocks the runtime will use to fill the regions in the master.
- To provide content for a ContentPlaceHolder, you use another specialized control, called Content.
- The ContentPlaceHolder control and the Content control have a one-to-one relationship.
- For each ContentPlaceHolder in the master page, the content page supplies a matching Content control
- ASP.NET links the Content control to the appropriate ContentPlaceHolder by matching the ID of the ContentPlaceHolder with the Content ContentPlaceHolderID property of the corresponding Content control.



Unit-IV

Topics:

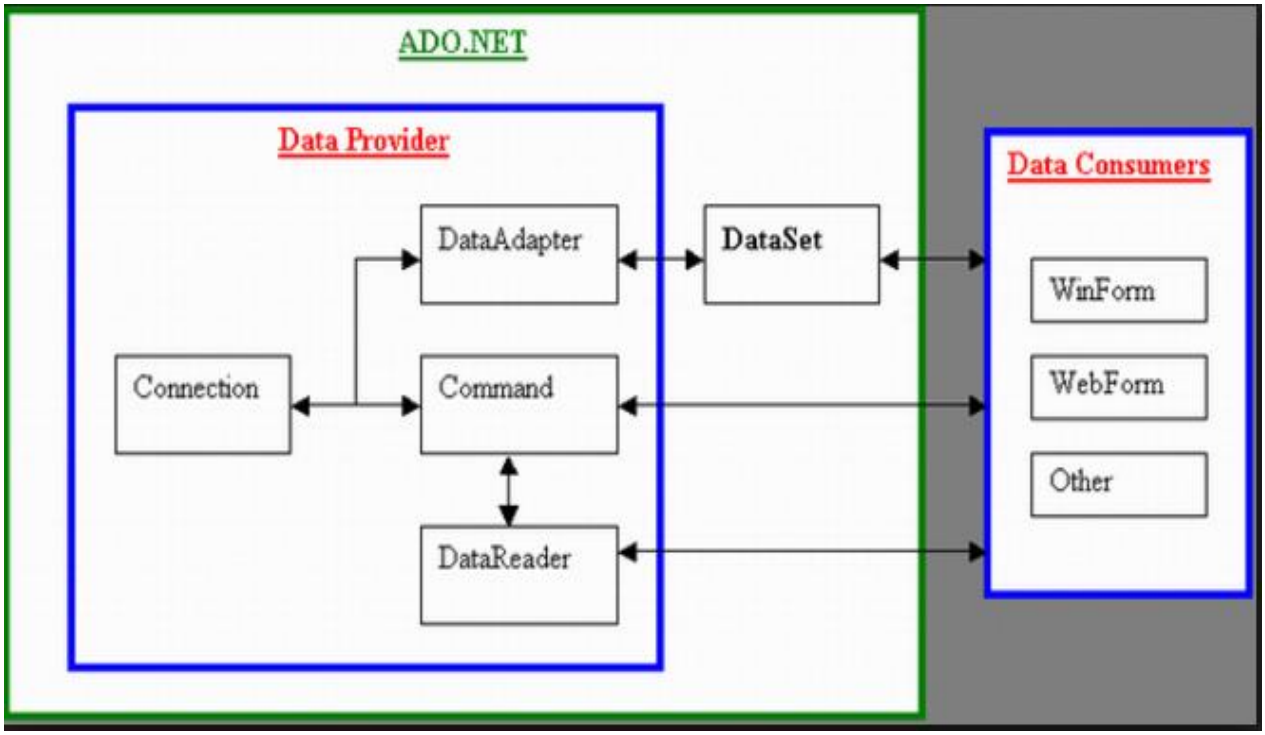
ADO.NET Fundamentals| Data Binding| Data Controls

Q1. What is ADO.Net? Explain its architecture.

ADO.NET provides a bridge between the front-end controls and the back-end database. The ADO.NET objects encapsulate all the data access operations and the controls interact with these objects to display data, thus hiding the details of movement of data.

ADO.NET consist of a set of Objects that expose data access services to the .NET environment. It is a data access technology from Microsoft .Net Framework, which provides communication between relational and non-relational systems through a common set of components.

System.Data namespace is the core of ADO.NET and it contains classes used by all data providers. ADO.NET is designed to be easy to use, and Visual Studio provides several wizards and other features that we can use to generate ADO.NET data access code.



The two key components of ADO.NET are **Data Providers** and **DataSet**. The Data Provider classes are meant to work with different kinds of data sources. They are used to perform all data-management operations on specific databases. DataSet class provides mechanisms for managing data when it is disconnected from the data source.

Data Providers

The .Net Framework includes mainly three Data Providers for ADO.NET. They are the Microsoft SQL Server Data Provider, OLEDB Data Provider and ODBC Data Provider. SQL Server uses the SqlConnection object , OLEDB uses the OleDbConnection Object and ODBC uses OdbcConnection Object respectively

A data provider contains Connection, Command, DataAdapter, and DataReader objects. These four objects provide the functionality of Data Providers in the ADO.NET.

DataSet

DataSet provides a disconnected representation of result sets from the Data Source, and it is completely independent from the Data Source. DataSet provides much greater flexibility when dealing with related Result Sets.

Q2. List and Explain ADO .NET objects.

ADO.NET includes many objects we can use to work with data. Some important objects of ADO.NET are:

Connection

To interact with a database, we must have a connection to it. The connection helps identify the database server, the database name, user name, password, and other parameters that are required for connecting to the data base.
A connection object is used by command objects so they will know which database to execute the command on.

Command

The command object is one of the basic components of ADO .NET. The Command Object uses the connection object to execute SQL queries.

The queries can be in the Form of Inline text, Stored Procedures or direct Table access. An important feature of Command object is that it can be used to execute queries and Stored Procedures with Parameters If a select query is issued, the result set it returns is usually stored in either a DataSet or a DataReader object.

DataReader

Many data operations require that we only get a stream of data for reading. The data reader object allows us to obtain the results of a SELECT statement from a command object. For performance reasons, the data returned from a data reader is a fast forward-only stream of data.

This means that we can only pull the data from the stream in a sequential manner this is good for speed, but if we need to manipulate data, then a DataSet is a better object to work with.

DataSet

DataSet objects are in-memory representations of data. They contain multiple Datatable objects, which contain columns and rows, just like normal database tables. We can even define relations between tables to create parent-child relationships.
The DataSet is specifically designed to help manage data in memory and to support disconnected operations on data, when such a scenario make sense.

DataAdapter

The data adapter fills a DataSet object when reading the data and writes in a single batch when persisting changes back to the database. A data adapter contains a reference to the connection object and opens and closes the connection automatically when reading from or writing to the database.

Q3. Explain Command Class and DataAdapter class with properties and methods

Command class properties and methods

Property	Description
CommandText	Specifies the SQL command or stored procedure or a name of a table.
CommandTimeout	Specifies the time required to wait for the completion of a command before it throws an exception. The default is 30 seconds.
CommandType	Accepts a value from the System.Data.CommandType enumeration that will determine the type of command specified in the CommandText property. It has 3 values, Text, for accepting SQL commands, StoredProcedure for stored procedures,
	and TableDirect to get all the rows and columns of one or multiple tables. Note that by default, Text will be used.
Connection	Specifies the connection that the command is associated to. The Command class must be hooked to an open connection which is the connection where the command is to be executed.
Parameters	A collection of Parameter defined in the CommandText.

Property	Description
Cancel	Tries to cancel the command being executed.
CreateParameter	Creates a new Parameter object that can be added to Command.Parameters collection.
ExecuteReader	Executes the command and returns a forward-only read-only cursor in the form of aDataReader.
ExecuteNonQuery	Executes the command and returns the number of rows that were affected. Often used with record UPDATE, DELETE, or INSERT statements.
ExecuteScalar	Executes the command, and retrieves a single value. Used with aggregate functions and in cases where you want to return the first column of the first row of a result set.

DataAdapter class properties and methods

Property	Description
DeleteCommand	Speicfies the Command object with the SQL command to be used for deleting a record.
FillCommandBehavior	Specifies the behavior of the command used to fill the data adapter.
InsertCommand	Specifies the Command object with the SQL command yused for inserting a record.
SelectCommand	Specifies the Command object with the SQL command to be used for inserting a record.
UpdateBatchSize	Specifies the number of commands that can be executed as a batch.
UpdateCommand	Specifies the Command object with the SQL command to be used for inserting a record.

Method	Description
AddToBatch	Adds a Command object to a batch that will be executed.
ClearBatch	Removes all the Command objects from the batch.
ExecuteBatch	Excutes the batch of commands.
Fill	Executes the SelectCommand property and fills a DataSet that is provided.
InitializeBatching	Initialize the batching process.
GetFillParameters	Gets the paremeters of the SelectCommand property.
TerminateBatching	Terminates the batching process.
Update	Calls the corresponding INSERT, DELETE, or UPDATE command of this DataAdapterand updates the data source using the specified commands.

Q4. Differentiate between DataSet and DataReader.

DataSet	DataReader
The DataSet class in ADO.Net operates in an entirely disconnected nature.	DataReader is a connection oriented service.
DataSet is an in-memory representation of a collection of Database objects including related tables, constraints, and relationships among the tables.	DataReader is designed to retrieve a read-only, forward-only stream of data from data sources.
It fetches entire table or tables at a time so greater network cost.	It fetches one row at a time so very less network cos.
DataSet is not read-only so we can do any transaction on them.	DataReader is readonly so we can't do any transaction on them.
DataAdapter is used to get data in DataSet	DataAdapter is not required
Dataset works with the help of xml technology	DataReader doesn't provide this feature.
Example: Dataset ds=new Dataset ( ); DataAdapter1Fill(ds,"newtablename") // where newtablename is table alias name in dataset	**Example**: Sqlcommand cmd =new sqlcommand (select * from emptable); Data Reader dr= cmd.ExecuteReader ( )

Q5. Explain the difference between DataReader and DataAdapter in ADO.NET.

DataReader

- DataReader is connection oriented architecture.
- DataReader is like a forward only recordset.
- It fetches one row at a time so very less network cost compare to DataSet (Fetches all the rows at a time).
- DataReader is readonly so we can't do any update or transaction on them.
- DataReader will be the best choice where we need to show the data to the user which requires no transaction.
- As DataReader is forward only so we can't fetch data randomly.
- .NET Data Providers optimizes the DataReader to handle huge amount of data.
- Performance is good.

DataAdapter

- DataAdapter acts as a bridge between DataSet and database.
- DataAdapter object is used to read the data from the database and bind that data to dataset.
- DataAdapter is a disconnected oriented architecture.
- DataAdapter resolves the changes made to the DataSet back to the database.

Q6. What is DataReader in ADO.NET? Explain with example.

DataReader provides an easy way for the programmer to read data from a database as if it were coming from a stream. The DataReader is the solution for forward streaming data through ADO.NET.

DataReader is also called a firehose cursor or forward read-only cursor because it moves forward through the data. The DataReader not only allows us to move forward through each record of database, but it also enables us to parse the data from each column. The DataReader class represents a data reader in ADO.NET.

Example:

In below example we read the all records from customer table using DataReader class.

```
string SQL = "SELECT * FROM Customers";
SqlConnection conn = new SqlConnection(ConnectionString);

// create a command object
SqlCommand cmd = new SqlCommand(SQL, conn);
conn.Open();

// Call ExecuteReader to return a DataReader
SqlDataReader reader = cmd.ExecuteReader();
Console.WriteLine("customer ID, Contact Name, " + "Contact Title, Address ");
Console.WriteLine("============================");

while (reader.Read())
{
   Console.Write(reader["CustomerID"].ToString() + ", ");
   Console.Write(reader["ContactName"].ToString() + ", ");
   Console.Write(reader["ContactTitle"].ToString() + ", ");
   Console.WriteLine(reader["Address"].ToString() + ", ");
}

//Release resources
reader.Close();
conn.Close();
```

Q7. What is data binding? Its types.

Data binding in ASP.NET is superficially similar to data binding in the world of desktop or client/server applications, but in truth, it's fundamentally different. In those environments, data binding involves creating a direct connection between a data source and a control in an application window.

If the user modifies the data in a data-bound control, our program can update the corresponding record in the database, but nothing happens automatically.

ASP.NET data binding is much more flexible than old-style data binding. Many of the most powerful data binding controls, such as the GridView and DetailsView, give us unprecedented control over the presentation of our data, allowing us to format it, change its layout, embed it in other ASP.NET controls, and so on.

Types of ASP.NET Data Binding

Two types of ASP.NET data binding exist: **single-value binding** and **repeated-value binding**. Single-value data binding is by far the simpler of the two, whereas repeated-value binding provides the foundation for the most advanced ASP.NET data controls.

Single-Value, or "Simple," Data Binding

We can use single-value data binding to add information anywhere on an ASP.NET page. We can even place information into a control property or as plain text inside an HTML tag.

Single-value data binding doesn't necessarily have anything to do with ADO.NET. Instead, single-value data binding allows us to take a variable, a property, or an expression and insert it dynamically into a page.

Repeated-Value, or "List," Binding

Repeated-value data binding allows us to display an entire table (or just a single field from a table). Unlike single-value data binding, this type of data binding requires a special control that supports it.

Typically, this will be a list control such as CheckBoxList or ListBox, but it can also be a much more sophisticated control such as the GridView.

Q8. What is a Data source? Explain various types of data sources in ASP.NET.

The Data source control connects to and retrieves data from a data source and makes it available for other controls to bind to, without requiring code. ASP.NET allows a variety of data sources such as a database, an XML file, or a middle-tier business object.

The common data source controls are:

- **AccessDataSource** – Enables you to work with a Microsoft Access database.
- **XmlDataSource** – Enables you to work with an XML file.
- **SqlDataSource** – Enables you to work with Microsoft SQL Server, OLE DB, ODBC, or Oracle databases.
- **ObjectDataSource** – Enables you to work with a business object or other class
- **SiteMapDataSource** – Used for ASP.NET site navigation.
- **EntityDataSource** - Enables you to bind to data that is based on the Entity Data Model.
- **LinqDataSource** – Enables you to use Language-Integrated Query (LINQ) in an ASP.NET Web page.

Q9. Explain SqlDataSource in ADO.NET.

The SqlDataSource control represents a connection to a relational database such as SQL Server or Oracle database, or data accessible through OLEDB or Open Database Connectivity (ODBC). Connection to data is made through two important properties ConnectionString and ProviderName.

The following code snippet provides the basic syntax of the control:

```
<asp:SqlDataSource runat="server" ID="MySqlSource"
```

```
ConnectionString='<%$ ConnectionStrings:Empconstr %>'
SelectionCommand= "SELECT * FROM EMPLOYEES" />
```

The following table provides the related sets of properties of the SqlDataSource control, which provides the programming interface of the control:

Property Group	Description
DeleteCommand, DeleteParameters, DeleteCommandType	Gets or sets the SQL statement, parameters, and type for deleting rows in the underlying data.
FilterExpression, FilterParameters	Gets or sets the data filtering string and parameters.
InsertCommand, InsertParameters, InsertCommandType	Gets or sets the SQL statement, parameters, and type for inserting rows in the underlying database.
SelectCommand, SelectParameters, SelectCommandType	Gets or sets the SQL statement, parameters, and type for retrieving rows from the underlying database.
SortParameterName	Gets or sets the name of an input parameter that the command's stored procedure will use to sort data.
UpdateCommand, UpdateParameters, UpdateCommandType	Gets or sets the SQL statement, parameters, and type for updating rows in the underlying data store.

The following code snippet shows a data source control enabled for data manipulation:

```
<asp:SqlDataSource runat="server" ID= "MySqlSource"
  ConnectionString=' <%$ ConnectionStrings:Empconstr %>'
  SelectCommand= "SELECT * FROM EMPLOYEES"
  UpdateCommand= "UPDATE EMPLOYEES SET LASTNAME=@lame"
  DeleteCommand= "DELETE FROM EMPLOYEES WHERE EMPLOYEEID=@eid"
  FilterExpression= "EMPLOYEEID > 10">
  .....
  .....
</asp:SqlDataSource>
```

Q10. What is a GridView control? Explain how to enable row selection, paging and sorting features of GridView.

The GridView control displays the values of a data source in a table. Each column represents a field, while each row represents a record. The GridView control supports the following features:

- Binding to data source controls, such as SqlDataSource.
- Built-in sort capabilities.
- Built-in update and delete capabilities.
- Built-in paging capabilities.
- Built-in row selection capabilities.
- Multiple key fields.
- Multiple data fields for the hyperlink columns.
- Customizable appearance through themes and styles

Sorting allows the user to sort the items in the GridView control with respect to a specific column by clicking on the column's header. To enable sorting, set the **AllowSorting** property to true.

```
AllowSorting="True"
```

Instead of displaying all the records in the data source at the same time, the GridView control can automatically break the records up into pages. To enable paging, set the **AllowPaging** property to true.

AllowPaging="True"

Also we can set how many rows we want to see in a page.

PageSize="4"

Example:

```
<asp:gridview AllowSorting="true" AllowPaging="true" PageSize="5" ID="Gridview1" runat="server"
DataKeyNames="pid" DataSourceID="SqlDS" >
<Columns>
<asp:BoundField DataField="pname" HeaderText="PRODUCT NAME"
SortExpression="pname"></asp:BoundField>
</Columns>
</asp:gridview>
```

Q11. Write a program which display all records from table to GridView

```
string constr = ConfigurationManager.ConnectionStrings["constr"].ConnectionString;
using (SqlConnection con = new SqlConnection(constr))
    {
        using (SqlCommand cmd = new SqlCommand("SELECT * FROM Customers"))
        {
            cmd.Connection = con;
            using (SqlDataAdapter sda = new SqlDataAdapter(cmd))
            {
                DataTable dt = new DataTable();
                sda.Fill(dt);
                GridView1.DataSource = dt;
                GridView1.DataBind();
            }
        }
    }
```

Q12. Explain DetailsView Control.

- DetailsView control is a data-bound control that renders a **single record at a time**. It can provide navigation option also. DetailsView control supports the edit, insert, delete and paging functionality.
- It can insert, update and delete the record also. When it is rendered on the page, generally it is implemented through <table> HTML tag.
- The DetailsView control supports exactly the same fields as the GridView control:

  BoundField: Enables us to display the value of a data item as text.
  CheckBoxField: Enables us to display the value of a data item as a check box.
  CommandField: Enables us to display links for editing, deleting, and selecting rows.
  ButtonField: Enables us to display the value of a data item as a button.
  HyperLinkField: Enables us to display the value of a data item as a link.
  ImageField: Enables us to display the value of a data item as an image.
  TemplateField: Enables us to customize the appearance of a data item.

Properties of DetailsView:

Properties	Description

AllowPaging	Gets or sets a value indicating whether the paging feature is enabled.
DataSource	Gets or sets the object from which the data-bound control retrieves its list of data items.
DataSourceID	Gets or sets the ID of the control from which the data-bound control retrieves its list of data items.
AutoGenerateEditButton	Gets or sets a value indicating whether the built-in controls to edit the current record are displayed in a DetailsView control.
AutoGenerateDeleteButton	Gets or sets a value indicating whether the built-in control to delete the current record is displayed in a DetailsView control.
AutoGenerateRows	Gets or sets a value indicating whether row fields for each field in the data source are automatically generated and displayed in a DetailsView control.
DefaultMode	Get or sets the default data-entry mode of the DetailsView control.

```
SqlConnection conn;
SqlDataAdapter adapter;
DataSet ds;
SqlCommand cmd;
string cs = ConfigurationManager.ConnectionStrings["conString"].ConnectionString;
protected void PopulateDetailView()
   {
      try
      {
         conn = new SqlConnection(cs);
         adapter = new SqlDataAdapter("select * from tblEmps", conn);
         ds = new DataSet();
         adapter.Fill(ds);
         DetailsView1.DataSource = ds;
         DetailsView1.DataBind();
      }
      catch (Exception ex)
      {
        Label1.Text = "ERROR :: " + ex.Message;
      }
   }
   protected void DetailsView1_PageIndexChanging(object sender, DetailsViewPageEventArgs e)
   {
      DetailsView1.PageIndex = e.NewPageIndex;
      PopulateDetailView();
   }
   protected void DetailsView1_ModeChanging(object sender, DetailsViewModeEventArgs e)
   {
      DetailsView1.ChangeMode(e.NewMode);
      PopulateDetailView();
   }
```

EmpID	1
Name	Raj
Gender	Male
Salary	25000
Address	Pune
DEPID	1
Edit Delete New	
1 2 3 4 5 6	

Q13. Briefly explain FormView control. How is it different from DetailsView?

- Like DetailsView, FormView also displays a **single record from the data source at a time**. Both controls can be used to display, edit, insert and delete database records but one at a time. Both have **paging feature** and hence **support backward and forward traversal**.
- FormView is a new data-bound control that is nothing but a **templated version of DetailsView control**. The major difference between DetailsView and FormView is, here user need to define the rendering template for each item.
- The FormView control provides more formatting and layout options than DetailsView.
- The DetailsView control uses <**BoundField**> elements or <**TemplateField**> elements to display bound data whereas FormView can use only templates to display bound data.
- The FormView control renders all fields in a single table row whereas the DetailsView control displays each field as a table row.
- When compare to DetailsView, the FormView control provides more control over the layout.

Following are some important properties that are very useful.

Templates of the FormView Control	
EditItemTemplate	The template that is used when a record is being edited.
InsertItemTemplate	The template that is used when a record is being created.
ItemTemplate	The template that is used to render the record to display only.
Methods of the FormView Control	
ChangeMode	ReadOnly/Insert/Edit. Change the working mode of the control from the current to the defined FormViewMode type.
InsertItem	Used to insert the record into database. This method must be called when the DetailsView control is in insert mode.
UpdateItem	Used to update the current record into database. This method must be called when DetailsView control is in edit mode.
DeleteItem	Used to delete the current record from database.

Q14. What is the difference between ListView and Gridview control? Explain the ListView control.

ListView presents the data in rows and columns just like a GridView control. The main difference between the ListView control and Gridview control is that the ListView control includes an additional row for inserting a new row into the table.

ListView Control:

The ListView control displays columns and rows of data and allows sorting and paging. It is by far the most popular data display control and is ideal for understanding how data display controls interact with data retrieval controls and code.

ListView provides various templates which we can use to display the data. The templates are:

- LayoutTemplate
- ItemTemplate
- ItemSeparatorTemplate
- GroupTemplate
- GroupSeparatorTemplate
- EmptyItemTemplate
- EmptyDataTemplate
- SelectedItemTemplate

- AlternatingItemTemplate
- EditItemTemplate
- InsertItemTemplate

Example:

```
<form id="form1" runat="server">
<asp:ListView ID="ListView1" runat="server" DataSourceID="SqlDataSource1">
</asp:ListView>
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:ConnectionString %>"
SelectCommand="SELECT * FROM [Customer]"></asp:SqlDataSource>
</form>
```

ListView

Sno	Name	Email	Age	Salary	Action
1	Anuj Koundal	Anuj@demo.com	28	321321	
2	Max Payne	Max@Demo.com	28	123123	
3	Tony Stark	Tony@demo.com	32	4564445	
4	Capt. Jack Sparrow	Jack@demo.com	33	44545	
5	Anuj	anuj@gmail.co	28	123456	

Q15. What is the application services provided in ASP.NET? Explain.

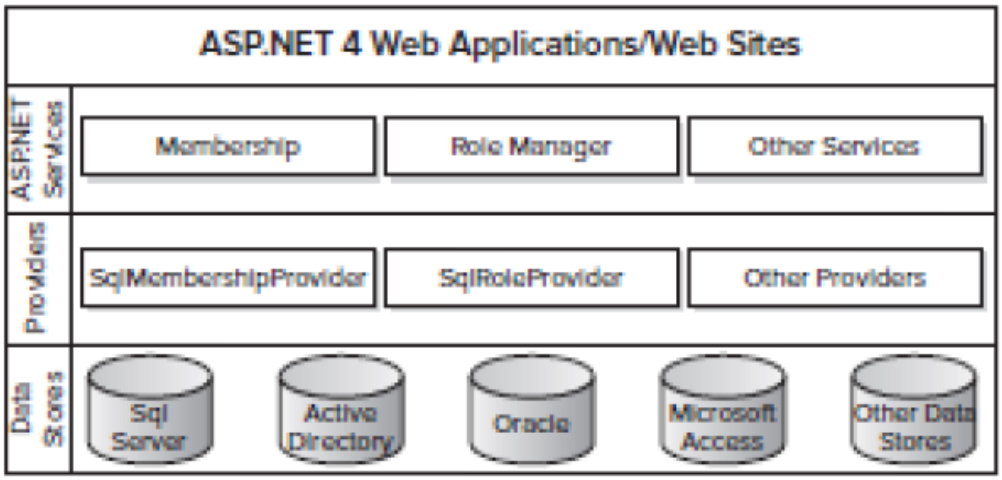
ASP.NET 4 ships with a number of application services, of which the most important ones are:

Membership: Enables us to manage and work with user accounts in our system.

Roles: Enables us to manage the roles that your users can be assigned to.

Profile: Enables us to store user-specific data in a back-end database.

Figure below gives an overview of these services and shows how they are related to our web site and the underlying data stores that the services may use.



A provider is software that provides a standardized interface between a service and a data source. ASP.NET providers are as follows:

- Membership
- Role management

- Site map
- Profile
- Session state etc.

At the top of the diagram you see the ASP.NET 4 web sites and web applications that represent the web sites that you build. These web sites can contain controls like the login controls that in turn can talk to the ASP.NET application services such as membership and profile. To create a flexible solution, these services don't talk to an underlying data source directly, but instead talk to a configured provider.

A provider is an interchangeable piece of software that is designed for a specific task. For example, in the case of the membership services, the membership provider is designed to work with users in the underlying data store. You can configure different providers for the same application service depending on your needs.

Unit-V

Topics:

XML| Security Fundamentals |ASP.NET AJAX

Q1. What is XML? List the various XML classes.

Extensible Markup Language (XML) stores and transports data. If we use a XML file to store the data then we can do operations with the XML file directly without using the database. The XML format is supported for all applications.

It is independent of all software applications and it is accessible by all applications. It is a very widely used format for exchanging data, mainly because it's easy readable for both humans and machines. If we have ever written a website in HTML, XML will look very familiar to us, as it's basically a stricter version of HTML. XML is made up of tags, attributes and values and looks something like this:

```
<?xmlversion="1.0"encoding="utf-8"?>
<EmployeeInformation>
 <Details>
  <Name>Richa</Name>
  <Emp_id>1</Emp_id>
  <Qualification>MCA</Qualification>
 </Details>
</EmployeeInformation>
```

XML Classes:

ASP.NET provides a rich set of classes for XML manipulation in several namespaces that start with **System.Xml**. The classes here allow us to read and write XML files, manipulate XML data in memory, and even validate XML documents.
The following options for dealing with XML data:

XmlTextWriter

The XmlTextWriter class allows us to write XML to a file. This class contains a number of methods and properties that will do a lot of the work for us. To use this class, we create a new XmlTextWriter object.

XmlTextReader

Reading the XML document in our code is just as easy with the corresponding XmlTextReader class. The XmlTextReader moves through our document from top to bottom, one node at a time. We call the **Read()** method to move to the next node. This method returns true if there are more nodes to read or false once it has read the final node.

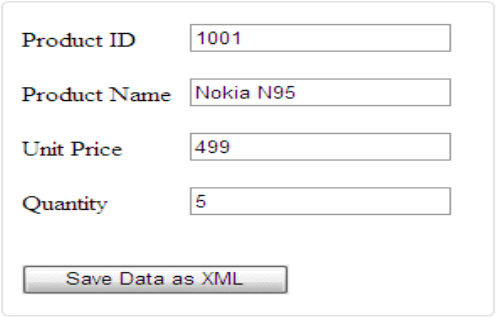
XDocument

The XDocument class contains the information necessary for a valid XML document. This includes an XML declaration, processing instructions, and comments. The XDocument makes it easy to read and navigate XML content. We can use the static **XDocument.Load()** method to read XML documents from a file, URI, or stream.

Q2. What is XmlTextWriter? Explain with example

XmlTextWriter is one of many useful classes in System.Xml namespace, and it provides several useful methods to write a complete XML Document from scratch.

Following form is simple ASP.NET web form asking user to input some data about the product. I will save this data in a separate XML document named **Product.xml** using XmlTextWriter object.



Following C# code is saving all the user input values in XML file.

```
protected void Button1_Click(object sender, EventArgs e)
{
    XmlTextWriter writer = null;
    try
    {
        string filePath = Server.MapPath("~") + "\\Product.xml";
        writer = new XmlTextWriter(filePath, System.Text.Encoding.UTF8);
        writer.Formatting = Formatting.Indented;
        writer.WriteComment("Created On: " + DateTime.Now.ToString("dd-MMM-yyyy"));
        writer.WriteComment("============================");
        writer.WriteStartElement("Product");
        writer.WriteAttributeString("ProductID", TextBox1.Text);
        writer.WriteElementString("ProductName", TextBox2.Text);
        writer.WriteElementString("ProductQuantity", TextBox3.Text);
        writer.WriteElementString("ProductPrice", TextBox4.Text);
        writer.WriteEndElement();
        writer.WriteEndDocument();
        writer.Flush();
        Response.Redirect("Product.xml");
    }
    catch (Exception ex){ }
}
```

Q3. What is XmlTextReader? Explain with example

Reading the XML document in our code is just as easy with the corresponding XmlTextReader class. The XmlTextReader moves through our document from top to bottom, one node at a time.

Example:

TernTenders.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<Tenders>
  <Ravina>
    <BillNo>1</BillNo>
    <PageNo>10</PageNo>
    <Activity>Metals</Activity>
  </Ravina>
  <Ravina>
    <BillNo>2</BillNo>
    <PageNo>20</PageNo>
    <Activity>Formworks</Activity>
  </Ravina>
  < Ravina>
    <BillNo>3</BillNo>
    <PageNo>30</PageNo>
    <Activity>SiteWorks</Activity>
  </ Ravina>
</Tenders>
```

ASPX Page:

```
<asp:GridView ID="GridView1" runat="server"></asp:GridView>
<asp:DropDownList ID="DropDownList1BillNo" runat="server"></asp:DropDownList>
```

```
<asp:DropDownList ID="DropDownList2PageNo" runat="server"></asp:DropDownList>
<asp:DropDownList ID="DropDownList3Activity" runat="server"></asp:DropDownList>
```

Code behind Page

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!IsPostBack)
    {
       BindDataToGridview();
    }
}
protected void BindDataToGridview()
{
  XmlTextReader xmlreader = new
XmlTextReader(Server.MapPath("~/App_Data/TernTenders.xml"));
  DataSet ds = new DataSet();
  ds.ReadXml(xmlreader);
  xmlreader.Close();
  if (ds.Tables.Count != 0)
  {
    GridView1.DataSource = ds;
    GridView1.DataBind();

    DropDownList1BillNo.DataSource = ds;
    DropDownList1BillNo.DataTextField = "BillNo";
    DropDownList1BillNo.DataValueField = "BillNo";
    DropDownList1BillNo.DataBind();

    DropDownList2PageNo.DataSource = ds;
    DropDownList2PageNo.DataTextField = "PageNo";
    DropDownList2PageNo.DataValueField = "PageNo";
    DropDownList2PageNo.DataBind();

    DropDownList3Activity.DataSource = ds;
    DropDownList3Activity.DataTextField = "Activity";
    DropDownList3Activity.DataValueField = "Activity";
    DropDownList3Activity.DataBind();
  }
}
```

Output

BillNo	PageNo	Activity
1	10	Metals
2	20	Formworks
3	30	SiteWorks

1 ▼ 10 ▼ Metals ▼

Q4. What is XDocument? Explain with example

The XDocument class contains the information necessary for a valid XML document. This includes an XML declaration, processing instructions, and comments. The XDocument makes it easy to read and navigate XML content. We can use the static **XDocument.Load()** method to read XML documents from a file, URI, or stream.

Example:

TernTenders.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<Tenders>
   <Ravina>
      <BillNo>1</BillNo>
      <PageNo>10</PageNo>
      <Activity>Metals</Activity>
   </Ravina>
   <Ravina>
      <BillNo>2</BillNo>
      <PageNo>20</PageNo>
      <Activity>Formworks</Activity>
   </Ravina>
   < Ravina>
      <BillNo>3</BillNo>
      <PageNo>30</PageNo>
      <Activity>SiteWorks</Activity>
   </ Ravina>
</Tenders>
```

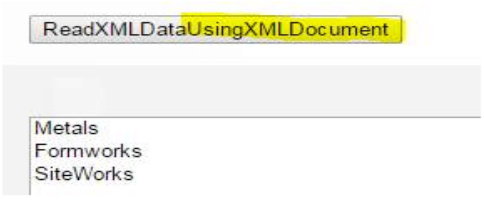
ASPX Page:

```
<asp:Button ID="ReadXmlUsingXMLDocument" runat="server"
Text="ReadXMLDataUsingXMLDocument" OnClick="ReadXmlUsingXMLDocument_Click" />
<asp:ListBox ID="ListBox2" runat="server" Height="500px" Width="500px"></asp:ListBox>
```

Code behind Page

```
protected void ReadXmlUsingXMLDocument_Click(object sender, EventArgs e)
{
   XmlDocument doc = new XmlDocument();
   doc.Load(Server.MapPath("~/App_Data/TernTenders.xml"));
   XmlNodeList elemList = doc.GetElementsByTagName("Activity");
   for (int i = 0; i < elemList.Count; i++)
   {
      ListBox2.Items.Add(elemList[i].InnerXml);
   }
}
```

Output



Q5. What is XElement? Explain with example

The XElement class represents an XML element in **System.Xml.XLinq**. XElement loads and parses XML. It allows us to remove lots of old code and eliminate the possibility of bugs and typos. The XAttribute class represents an attribute of an element. XElement.Save method saves the contents of XElement to a XML file.

Example:

Authors.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<Authors>
  <Author Name="Mahesh Chand">
  <Book>GDI+ Programming</Book>
  <Cost>$49.95</Cost>
  <Publisher>Addison-Wesley</Publisher>
 </Author>
<Author Name="Mike Gold">
   <Book>Programmer's Guide to C#</Book>
   <Cost>$44.95</Cost>
   <Publisher>Microgold Publishing</Publisher>
 </Author>
 <Author Name="Scott Lysle">
   <Book>Custom Controls</Book>
   <Cost>$39.95</Cost>
   <Publisher>C# Corner</Publisher>
 </Author>
</Authors>
```

Code Behind

```
XElement allData = XElement.Load(Server.MapPath("~/App_Data/ Authors.xml")
if (allData != null)
{
      IEnumerable<XElement> authors = allData.Descendants("Author");
      foreach(XElement author in authors)
      Response.Write ((string) author);
}
```

Q6. What is XML Validation?

The validation of an Xml input File could occur at various instances of processing as mentioned below:

- using a schema file
- In the Database Code for validating against business rules.

All Xml code can be considered as categorically correct if they are **well-formed** and **valid xml files**.

Well-formed - The XML code must be syntactically correct or the XML parser will raise an error.

Valid - If the XML file has an associated XML Schema, the elements must appear in the defined structure and the content of the individual elements must conform to the declared data types specified in the schema.

Validation of Xml Files can be achieved through the use of various Technologies ex. Visual Studio Net. Also, there are many on-line Tools available for validating an input .xml File. Few of them are mentioned below:

Xml Schema Validation

An Xml File is generally validated for its conformance to a particular schema. The Xml schema file usually is a XML Schema definition language (XSD). The input Xml File could be even validated against a set of Schema Files. The Schema File is the structural representation of how a given Xml Data File should resemble.

Validation using XSL Technology:

XSLT [Extended Stylesheet Language Transformation] is basically used for transforming the input xml File from one form to another, which could be .html, .xml etc. Here, we use XSLT to transform the input Xml File to a form that we require for further processing

Q7. What is XML Transform using XslCompiledTransform Class? Explain with example

In the .NET Framework a class **XslCompiledTransform** is present in the System.Xml.Xsl namespace that can be used to do the transformation. Then theXslCompiledTransform object calls the Load () method to load the XSLT file content into the object.

Example:

Data.xml

```
<?xml version='1.0'?>
<bookstore>
  <book genre="A"  publicationdate="1981"  ISBN="1-11111-11-0">
    <title>title 1</title>
    <author>
      <first-name>A</first-name>
      <last-name>B</last-name>
    </author>
    <price>8</price>
  </book>
  <book genre="B"  publicationdate="1999"  ISBN="0-222-22222-2">
    <title>title 2</title>
    <author>
      <first-name>C</first-name>
      <last-name>D</last-name>
    </author>
    <price>11.99</price>
  </book>
</bookstore>
```

Data.xsl

```
<?xml version="1.0" encoding="utf-8" ?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:template match="/">
    <xsl:element name="Authors">
      <xsl:apply-templates select="//book"/>
    </xsl:element>
  </xsl:template>
  <xsl:template match="book">
    <xsl:element name="Author">
      <xsl:value-of select="author/first-name"/>
      <xsl:text> </xsl:text>
      <xsl:value-of select="author/last-name"/>
    </xsl:element>
  </xsl:template>
</xsl:stylesheet>
```

```
protected void Page_Load(object sender, EventArgs e)
  {
    Response.ContentType = "text/xml";

    string xsltFile = Path.Combine(Request.PhysicalApplicationPath, "Data.xslt");
    string xmlFile = Path.Combine(Request.PhysicalApplicationPath, "Data.xml");

    XslCompiledTransform xslt = new XslCompiledTransform();
```

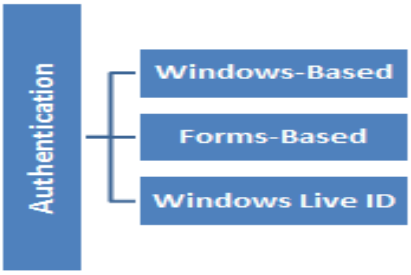
```
    xslt.Load(xsltFile);

    XPathDocument doc = new XPathDocument(xmlFile);
    xslt.Transform(doc, new XmlTextWriter(Response.Output));
  }
```

Q8. What do you mean by authentication? Explain its types.

Authentication is process of validating the identity of a user so the user can be granted access to an application. A user must typically supply a user name and password to be authenticated.

After a user authenticated, the user must still be authorized to use the required application. The process of granting user access to an application is called authorization.

There are 3 types of authentication as follows:



Windows-based authentication:

- It causes the browser to display a login dialog box when the user attempts to access restricted page.
- It is supported by most browsers.
- It is configured through the IIS management console.
- It uses windows user accounts and directory rights to grant access to restricted pages.

```
<authentication mode="Windows">
  <forms name=" AuthenticationDemo" loginUrl="logon.aspx" protection="All" path="/"
timeout="30" />
</authentication>
```

Forms-based authentication:

- Developer codes a login form that gets the user name and password.
- The username and password entered by user are encrypted if the login page uses a secure connection.
- It doesn't reply on windows user account.

```
<authentication mode="Forms">
   <forms name=" AuthenticationDemo" loginUrl="logon.aspx" protection="All" path="/"
timeout="30" />
</authentication>
```

Deny access to the anonymous user in the <authorization> section as follows:

```
<authorization>
   <deny users ="?" />
</authorization>
```

Windows Live ID authentication:

- It is centralized authentication service offered by Microsoft.
- The advantage is that the user only has one maintain one username and password.

Q9. What do you mean by Impersonation in ASP.NET?

ASP.NET impersonation is used to control the execution of the code in authenticated and authorized client. It is not a default process in the ASP.NET application. It is used for executing the local thread in an application. If the code changes the thread, the new thread executes the process identity by default.

The impersonation can be enabled in **two ways** as mentioned below:

Impersonation enabled: ASP.NET impersonates the token passed to it by IIS, can be an authenticated user or an internet user account. The syntax for enabling is as shown below:

```
<identity impersonate=”true” />
```

Impersonation enabled for a specific identity: ASP.NET impersonates the token generated using the identity specified in the Web.config file.

```
<identity impersonate=”true”
  userName=”domain\user”
 password=”password” />
```

Impersonation disabled: It is the default setting for the ASP.NET application. The process identity of the application worker process is the ASP.NET account.

```
<identity impersonate=”false” />
```

Q10. Steps to use Windows authentication with sample code

When we configure our ASP.NET application as windows authentication it will use local windows user and groups to do authentication and authorization for our ASP.NET pages.
In ‘web.config’ file set the authentication mode to ‘Windows’ as shown in the below code snippets.

```
<authentication mode="Windows"/>
```

We also need to ensure that all users are denied except authorized users. The below code snippet inside the authorization tag that all users are denied. ‘?’ indicates any

```
<authorization>
<deny users="?"/>
</authorization>
```

We also need to specify the authorization part. We need to insert the below snippet in the ‘**web.config’** file .stating that only ‘Administrator’ users will have access to

```
<location path="Admin.aspx">
<system.web>
<authorization>
<allow roles="questpon-srize2\Administrator"/>
<deny users="*"/>
</authorization>
</system.web>
</location>
```

The next step is to compile the project and upload the same on an IIS virtual directory. On the IIS virtual directory we need to ensure to remove anonymous access and check the integrated windows authentication.

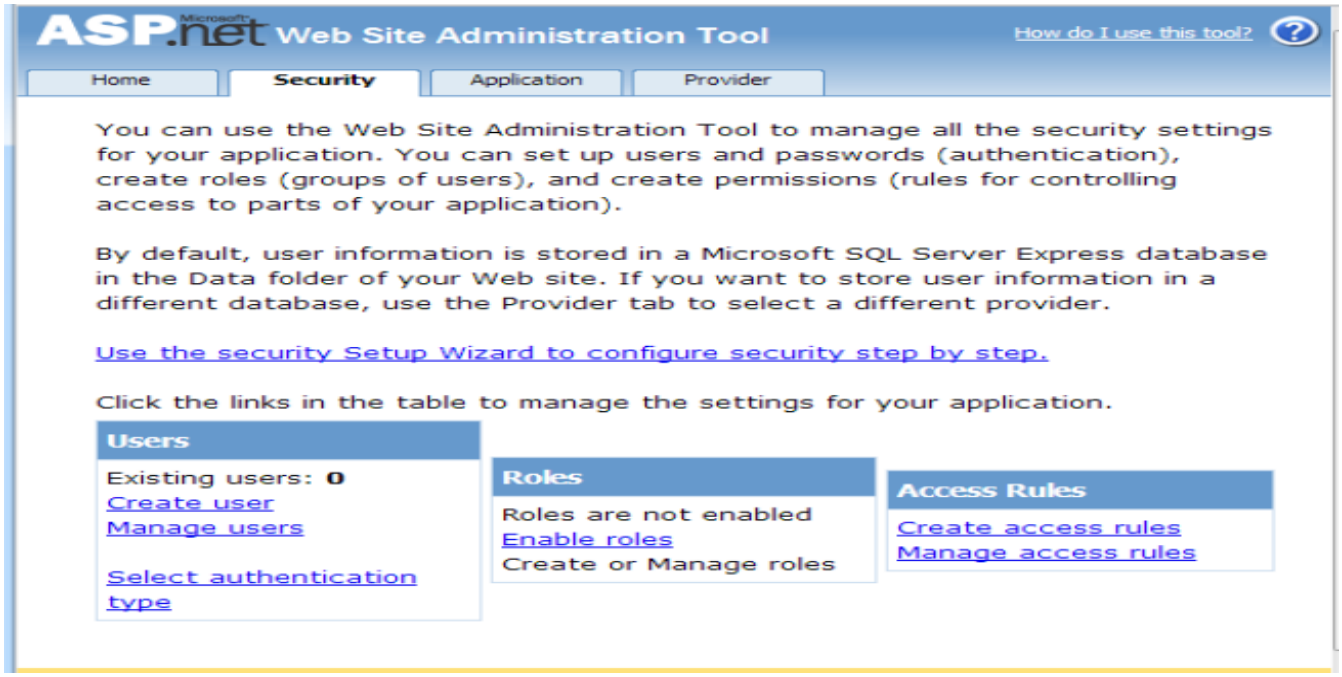
Now if we run the web application we will be popped with a userid and password box.

Q11. Short note on Web Site Administration Tool (WAT) in ASP.NET.

Another way to set up our authentication and authorization rules. Rather than edit the **web.config** file by hand, we can use the WAT from inside Visual Studio. It’s also often quicker to enter a list of authorization rules by hand than to use the WAT.

To use the WAT for this type of configuration**, select Website ➤ ASP.NET Configuration** from the menu. Next, click the Security tab.

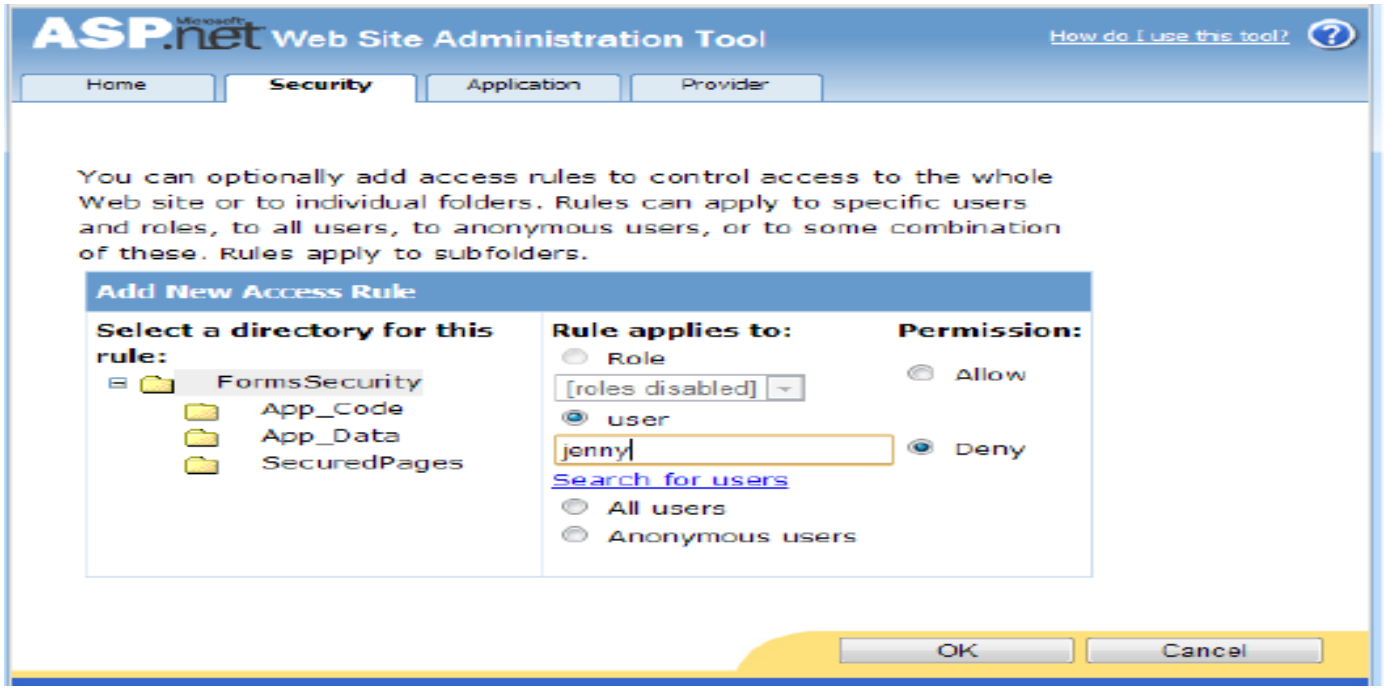
We will see the window shown in figure, which gives us links to set the authentication type, define authorization rules (using the Access Rules section), and enable role-based security. (Role-based security is an optional higher-level feature we can use with forms authentication.



To set an application to use forms authentication, follow these steps:

1. Click Select Authentication Type.
2. Choose the From the Internet option.
3. Click Done. The appropriate <authorization> tag will be created in the web.config file.

Next, it's time to define the authorization rules. To do so, click the Create Access Rules link.



Q12. What is AJAX? Its Advantages and Disadvantages

Asynchronous JavaScript and XML (AJAX) is a development technique used to create interactive web applications or rich internet applications.

AJAX uses several existing technologies together, including: XHTML, CSS, JavaScript, Document Object Model, XML, XSLT, and the XMLHttpRequest object.

With AJAX, web applications can retrieve data from the server asynchronously, in the background, without reloading the entire browser page. The use of AJAX has led to an increase in interactive animation on web pages

Advantages

- Reduces the traffic travels between the client and the server.
- No cross browser pains.
- Better interactivity and responsiveness.
- With AJAX, several multipurpose applications and features can be handled using a single web page(SPA).
- API's are good because those work with HTTP method and JavaScript.

Disadvantages

- Search engines like Google would not be able to index an AJAX application.
- It is totally built-in JavaScript code. If any user disables JS in the browser, it won't work.
- The server information cannot be accessed within AJAX.
- Security is less in AJAX applications as all the files are downloaded at client side.
- The data of all requests is URL-encoded, which increases the size of the request.

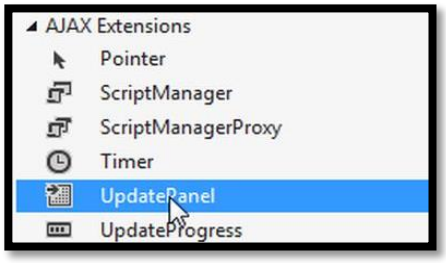
Q13. What are the benefits using Ajax? Explain UpdatePanel and ScriptManager.

The major benefit of Ajax is partial page rendering. The partial update of a page does not necessitate full reload of the page and hence leads to flicker-free page rendering.

UpdatePanel

- We can refresh the selected part of the web page by using UpdatePanel control, Ajax UpdatePanel control contains a two child tags that is ContentTemplate and Triggers. In a ContentTemplate tag we used to place the user controls and the Trigger tag allows us to define certain triggers which will make the panel update its content.

```
<asp:UpdatePanel ID="updatepnl" runat="server">
<ContentTemplate>
```

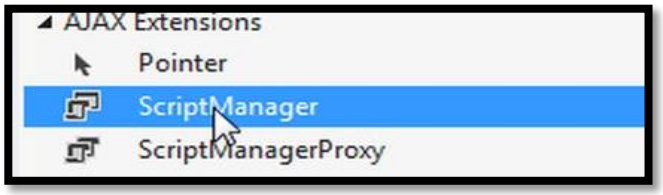


- All the contents that must be updated asynchronously (only ContentTemplate parts are updated and rest of the web page part is untouched) are placed here. It allows us to send request or post data to server without submit the whole page so that is called asynchronous.
- UpdatePanel is a container control. A page can have multiple update panels.
- The UpdatePanel control enables you to create a flicker free page by providing partial-page update support to it.
- It identifies a set of server controls to be updated using an asynchronous post back.
- If a control within the UpdatePanel causes a post back to the server, only the content within that UpdatePanel is refreshed.

```
<asp:UpdatePanel ID="UpdatePanel1" runat="server">
<ContentTemplate>
</ContentTemplate>
</asp:UpdatePanel>
```

ScriptManager

- The ScripManager Control manages the partial page updates for UpdatPanel controls that are on the ASP.NET web page or inside a user control on the web page.
- This control manages the client script for AJAX-enabled ASP.NET web page and ScripManager control support the feature as partial-page rendering and web-service calls.
- The ScriptManager control manages client script for AJAX-enabled ASP.NET Web pages.
- Although this control is not visible at runtime, it is one of the most important controls for an Ajax enabled web page.
- There can be only one ScriptManager in an Ajax enabled web page.



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

Q14. Explain the use of UpdateProgress control in AJAX.

The UpdateProgress provides status information about partial page updates in UpdatePanel controls. Despite the visual problems that post backs usually cause, they have one big advantage: the user can see something is happening.

The UpdatePanel makes this a little more difficult. Users have no visual cue that something is happening until it has happened. To tell users to hold on for a few seconds while their request is being processed, we can use the UpdateProgress control.

We usually put text such as "**Please wait**" or an animated image in this template to let the user know something is happening, although any other markup is acceptable as well. We can connect the UpdateProgress control to an UpdatePanel using the **AssociatedUpdatePanelID** property.

Its contents, defined in the <ProgressTemplate> element, are then displayed whenever the associated UpdatePanel is busy refreshing.

Example:

In the example below, we use the same .gif to display progress while the UpdatePanel is updating its content. For understanding purposes, we have emulated a time-consuming operation by setting a delay of 3 seconds by using **Thread.Sleep(3000)** on the button click.

```
protected void btnInvoke_Click(object sender, EventArgs e)
{
        System.Threading.Thread.Sleep(3000);
        lblText.Text = "Processing completed";
}
```

```
<asp:ScriptManager ID="ScriptManager1" runat="server">
        </asp:ScriptManager>
        <asp:UpdateProgress ID="updProgress"
        AssociatedUpdatePanelID="UpdatePanel1"
        runat="server">
            <ProgressTemplate>
            <img alt="progress" src="images/progress.gif"/>
               Processing...
            </ProgressTemplate>
        </asp:UpdateProgress>

        <asp:UpdatePanel ID="UpdatePanel1" runat="server">
            <ContentTemplate>
                <asp:Label ID="lblText" runat="server" Text=""></asp:Label>
                <br />
                <asp:Button ID="btnInvoke" runat="server" Text="Click"
                    onclick="btnInvoke_Click" />
```

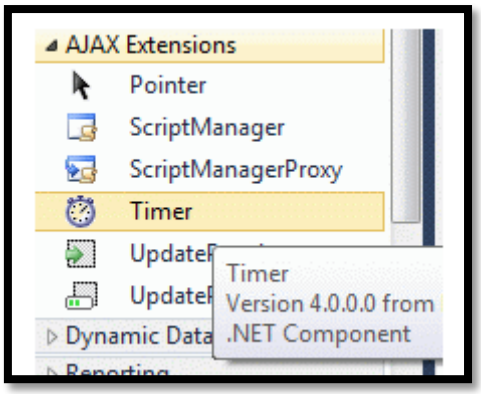
```
            </ContentTemplate>
        </asp:UpdatePanel>
```

In the code shown above, we use the **AssociatedUpdatePanelID** property of the UpdateProgress control to associate it with an UpdatePanel control. The **ProgressTemplate** property that can contain HTML, is used to specify the message displayed by an UpdateProgress control. In our case, we are displaying a .gif image progress as shown below.

Q15. Explain the use of Timer control in AJAX.

Timer controls allow us to do postbacks at certain intervals. If used together with UpdatePanel, which is the most common approach, it allows for timed partial updates of our page, but it can be used for posting back the entire page as well.

The Timer control uses the interval attribute to define the number of milliseconds to occur before firing the Tick event.




```
<asp:Timer ID="Timer1" runat="server" Interval="2000" OnTick="Timer1_Tick">
</asp:Timer>
```

Example:

Here is a small example of using the Timer control. It simply updates a timestamp every 5 seconds.

```
<asp:ScriptManager ID="ScriptManager1" runat="server" />
    <asp:Timer runat="server" id="UpdateTimer" interval="5000" ontick="UpdateTimer_Tick" />
    <asp:UpdatePanel runat="server" id="TimedPanel" updatemode="Conditional">
      <Triggers>
        <asp:AsyncPostBackTrigger controlid="UpdateTimer" eventname="Tick" />
      </Triggers>
      <ContentTemplate>
        <asp:Label runat="server" id="DateStampLabel" />
      </ContentTemplate>
</asp:UpdatePanel>
```

We only have a single CodeBehind function, which we should add to our CodeBehind file:

```
protected void UpdateTimer_Tick(object sender, EventArgs e)
{
  DateStampLabel.Text = DateTime.Now.ToString();
}
```

Q16. Explain ASP.NET AJAX Control Toolkit.

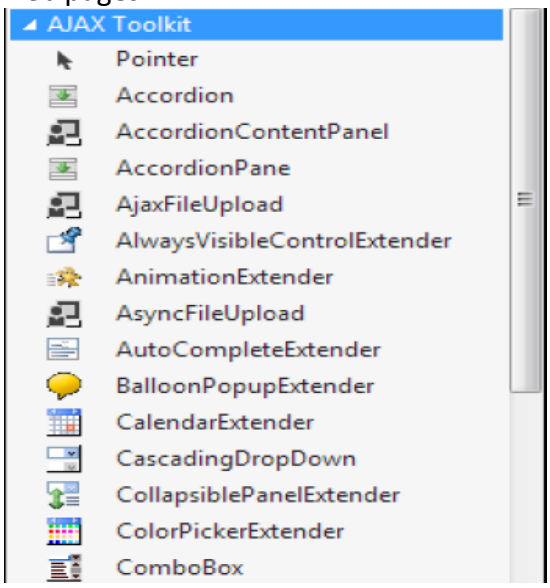
The Ajax Control Toolkit is an open source library for web development. The ASP.net Ajax Control toolkit contains highly rich web development controls for creating responsive and interactive AJAX enabled web applications.

ASP.Net Ajax Control Toolkit contains 40 + ready controls which is easy to use for fast productivity. Controls are available in the Visual Studio Toolbox for easy drag and drop integration with our web application. Some of the controls are like AutoComplete, Color Picker, Calendar, Watermark, Modal Popup Extender, Slideshow Extender and more of the useful controls.

The ASP.Net AJAX Control toolkit is now maintained by **DevExpress** Team. The Current Version of ASP.Net AJAX Toolkit is v16.1.0.0. There are lot of new enhancement in the current version from new controls to bug fixes in all controls.

The ASP.NET AJAX Control Toolkit has a lot going for it:

- It's completely free.
- It includes full source code, which is helpful if we're ambitious enough to want to create our own custom controls that use ASP.NET AJAX features.
- It uses extenders that enhance the standard ASP.NET web controls. That way, we don't have to replace all the controls on our web pages.



Q17. Explain Accordion in ASP.NET AJAX Control Toolkit.

Accordion control provides us multiple collapsible panes or panels where only one can be expanded at a time. Each AccordionPane control has a template for its Header and its content.

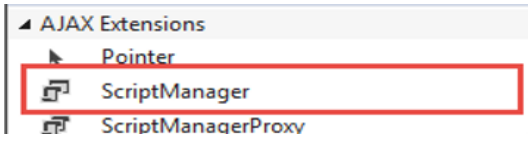
Accordion Web Control contains multiple panes and shows one at a time. The panes contain content of images and text having wide range of supported properties from formatting of the panes and text, like header and content templates, also it formats the selected header template.

Accordion contains multiple collapsible panes while one can be shown at a time and the extended features of accordion also has support data source which points to the DataSourceID and binds through .DataBind() method.

Example we have lot of paragraphs in a single page and we don't have much space and want to summarize in small space then we need to embed to the div or some container which can hold all the information in accordion collapsible pans.

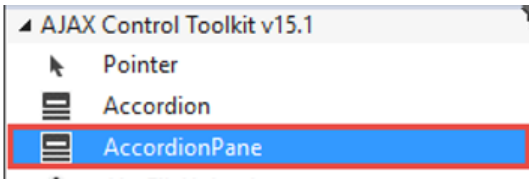
1. Go to File, New, then Website and Create an Empty Website. Add a webform (Default.aspx) in it.

2. Add ScriptManager from AJAX Extensions (from v15.1 of AJAX Control Toolkit, ToolScriptManager is removed. Use Standard Script Manager).

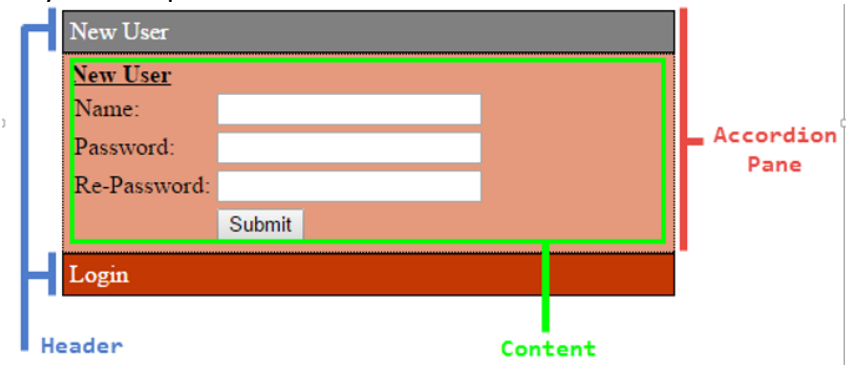


3. Go to AJAX Control Toolkit in Toolbox and drop Accordion Control on Default.aspx.

4. Add AccordionPane in Panes collection of the Accordion.



AccordionPane contains two parts i.e. Header and Content. When AccordionPane is collapsed, only Header part is visible to us.



Accordion control have several other properties like:

- **FadeTransitions**: FadeTransitions is used for adding the transition effect.
- **SelectedIndex**: The AccordionPane to be initially visible.
- **TransitionDuration**: Number of milliseconds to animate the transitions.
- **Panes**: Collection of AccordionPane controls.

Q18. Explain AutoCompleteExtender in ASP.NET AJAX Control Toolkit.

AutoComplete is an ASP.NET AJAX extender that can be attached to any TextBox control and will associate that control with a Popup panel to display a result returned from web services or retrieved from a database on the basis of text in a TextBox.

The Dropdown with name retrieved from the database or web service is positioned on the bottom-left of the TextBox.

Properties of AutoCompleteExtender:

- **TargetControlID**: The TextBox control Id where the user types text to be automatically completed.
- **EnableCaching**: Whether client-side caching is enabled.
- **CompletionSetCount**: Number of suggestions to be retrieved from the web service.
- **MinimumPrefixLength**: Minimum number of characters that must be entered before getting suggestions from a web service.
- **CompletionInerval**: Time in milliseconds when the timer will kick in to get a suggestion using the web service.
- **ServiceMethod**: The web service method to be called.
- **FirstRowSelected**: Whether the first row is selected from the suggestion.

Example:

```
<asp:ScriptManager ID="ScriptManager1" runat="server" EnablePageMethods="true">
</asp:ScriptManager>
<asp:AutoCompleteExtender ServiceMethod="GetSearch" MinimumPrefixLength="1"
CompletionInterval="10"  EnableCaching="false" CompletionSetCount="10"
TargetControlID="TextBox1" ID="AutoCompleteExtender1"  runat="server"
FirstRowSelected="false">
</asp:AutoCompleteExtender>
<asp:Label ID="Label1" runat="server" Text="Search Name"></asp:Label>
```

```
<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
```

Source Code:

```
[System.Web.Script.Services.ScriptMethod()]
[System.Web.Services.WebMethod]
 public static List<string> GetSearch(string prefixText)
{
      SqlConnection con = new SqlConnection("connectionString"); //Create a Sql connection
      SqlDataAdapter da;
      DataTable dt;
      DataTable Result = new DataTable();
      string str = "select nvName from Friend where nvName like '" + prefixText + "%'";
      da = new SqlDataAdapter(str, con);
      dt = new DataTable();
      da.Fill(dt);
      List<string> Output = new List<string>();
      for (int i = 0; i < dt.Rows.Count; i++)
         Output.Add(dt.Rows[i][0].ToString());
      return Output;
}
```