**OOPS Concepts**

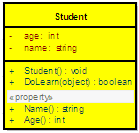
**Class:**

It is a collection of objects.

**Object:**

It is a real time entity.

An object can be considered a "**thing**" that can perform a set of **related**activities. The set of activities that the object performs defines the object's behavior. For example, the hand can grip something or a**Student**(object) can give the name or address. In pure **OOP**terms an object is an instance of a class

[](http://3.bp.blogspot.com/-jk1xzP_jph0/UG3cM0KD95I/AAAAAAAAB5Y/aBkA6f1KBDk/s1600/Student.png)

The above template describe about object **Student**

Class is composed of three things name, attributes, and operations

public class student

{

}

student objstudent=new student ();

According to the above sample we can say that **Student**object, named **objstudent,**has created out of the student class.

In real world you will often find many individual objects all of the same kind. As an example, there may be thousands of other bicycles in existence, all of the same make and model. Each bicycle has built from the same blueprint. In object-oriented terms, we say that the bicycle is an instance of the class of objects known as bicycles. In the software world, though you may not have realized it, you have already used classes. For example, the **Textbox**control, you always used, is made out of the **Textbox**class, which defines its appearance and capabilities. Each time you drag a **Textbox**control, you are actually creating a new instance of the **Textbox**class.

**Encapsulation**:

Encapsulation is a process of binding the data members and member functions into a single unit.

**Example** for encapsulation is **class**. A class can contain data structures and methods.

Consider the following class

public class Aperture

{

public Aperture ()

{

}

protected double height;

protected double width;

protected double thickness;

public double get volume()

{

Double volume=height \* width \* thickness;

if (volume<0)

return 0;

return volume;

}

}

In this example we encapsulate some data such as height, width, thickness and method Get Volume. Other methods or objects can interact with this object through methods that have public access modifier

**Abstraction:**

Abstraction is a process of hiding the implementation details and displaying the essential features.

**Example1**: A Laptop consists of many things such as processor, motherboard, RAM, keyboard, LCD screen, wireless antenna, web camera, usb ports, battery, speakers etc. To use it, you don't need to know how internally LCD screens, keyboard, web camera, battery, wireless antenna, speaker’s works.  You just need to know how to operate the laptop by switching it on. Think about if you would have to call to the engineer who knows all internal details of the laptop before operating it. This would have highly expensive as well as not easy to use everywhere by everyone.

So here the Laptop is an object that is designed to hide its complexity.

How to abstract: - By using **Access Specifiers**

**.Net has five access Specifiers**

**Public** -- Accessible outside the class through object reference.

**Private** -- Accessible inside the class only through member functions.

**Protected** -- Just like private but Accessible in derived classes also through member

functions.

**Internal**-- Visible inside the assembly. Accessible through objects.

**Protected Internal** -- Visible inside the assembly through objects and in derived classes outside the assembly through member functions.

Let’s try to understand by a practical example:-

public class Class1

    {

        int  i;                                         //No Access specifier means private

        public  int j;                                        // Public

        protected int k;                             //Protected data

        internal int m;                        // Internal means visible inside assembly

        protected internal int n;                   //inside assembly as well as to derived classes outside assembly

        static int x;                                 // This is also private

        public static int y;                       //Static means shared across objects

        [DllImport("MyDll.dll")]

        public static extern int MyFoo();       //extern means declared in this assembly defined in some other assembly

        public void myFoo2()

        {

            //Within a class if you create an object of same class then you can access all data members through object reference even private data too

            Class1 obj = new Class1();

            obj.i =10;   //Error can’t access private data through object.But here it is accessible.:)

            obj.j =10;

            obj.k=10;

            obj.m=10;

            obj.n=10;

       //     obj.s =10;  //Errror Static data can be accessed by class names only

            Class1.x = 10;

         //   obj.y = 10; //Errror Static data can be accessed by class names only

            Class1.y = 10;

        }

    }

Now lets try to copy the same code inside Main method and try to compile

[STAThread]

        static void Main()

        {

           //Access specifiers comes into picture only when you create object of class outside the class

            Class1 obj = new Class1();

       //     obj.i =10; //Error can’t access private data through object.

            obj.j =10;

      //      obj.k=10;     //Error can’t access protected data through object.

            obj.m=10;

            obj.n=10;

       //     obj.s =10;  //Errror Static data can be accessed by class names only

            Class1.x = 10;  //Error can’t access private data outside class

         //   obj.y = 10; //Errror Static data can be accessed by class names only

            Class1.y = 10;

        }

What if Main is inside another assembly

[STAThread]

        static void Main()

        {

           //Access specifiers comes into picture only when you create object of class outside the class

            Class1 obj = new Class1();

       //     obj.i =10; //Error can’t access private data through object.

            obj.j =10;

      //      obj.k=10;     //Error can’t access protected data through object.

     //     obj.m=10; // Error can’t access internal data outside assembly

    //      obj.n=10; // Error can’t access internal data outside assembly

       //     obj.s =10;  //Errror Static data can be accessed by class names only

            Class1.x = 10;  //Error can’t access private data outside class

         //   obj.y = 10; //Errror Static data can be accessed by class names only

            Class1.y = 10;

        }

In object-oriented software, complexity is managed by using **abstraction**.

**Abstraction** is a process that involves identifying the critical behavior of an object and eliminating irrelevant and complex details.

**Inheritance:**

Inheritance is a process of deriving the new class from already existing class

**C#** is a complete object oriented programming language. Inheritance is one of the primary concepts of object-oriented programming. It allows you to reuse existing code. Through effective use of inheritance, you can save lot of time in your programming and also reduce errors, which in turn will increase the quality of work and productivity. A simple example to understand inheritance in C#.

Using System;

Public class BaseClass

{

    Public BaseClass ()

    {

        Console.WriteLine ("Base Class Constructor executed");

    }

    Public void Write ()

    {

        Console.WriteLine ("Write method in Base Class executed");

    }

}

Public class ChildClass: BaseClass

{

    Public ChildClass ()

    {

        Console.WriteLine("Child Class Constructor executed");

    }

    Public static void Main ()

    {

        ChildClass CC = new ChildClass ();

        CC.Write ();

    }

}

In the Main () method in ChildClass we create an instance of childclass. Then we call the write () method. If you observe the ChildClass does not have a write() method in it. This write () method has been inherited from the parent BaseClass.

The output of the above program is   
  
**Output:**

  Base Class Constructor executed  
  Child Class Constructor executed  
  Write method in Base Class executed

this output proves that when we create an instance of a child class, the base class constructor will automatically be called before the child class constructor. So in general Base classes are automatically instantiated before derived classes.

In C# the syntax for specifying BaseClass and ChildClass relationship is shown below. The base class is specified by adding a colon, ":", after the derived class identifier and then specifying the base class name.

**Syntax:**  class **ChildClassName**: **BaseClass**  
              {  
                   //Body  
              }

C# supports single class inheritance only. What this means is, your class can inherit from only one base class at a time. In the code snippet below, class C is trying to inherit from Class A and B at the same time. This is not allowed in C#. This will lead to a compile time

error: **Class 'C' cannot have multiple base classes: 'A' and 'B'**.

public class A

{

}

public class B

{

}

public class C : A, B

{

}

In C# Multi-Level inheritance is possible. Code snippet below demonstrates mlti-level inheritance. Class B is derived from Class A. Class C is derived from Class B. So class C, will have access to all members present in both Class A and Class B. As a result of multi-level inheritance Class has access to A\_Method(),B\_Method() and C\_Method().   
  
***Note:*** Classes can inherit from multiple interfaces at the same time. **Interview Question:**How can you implement multiple inheritance in C#? **Ans :**Using Interfaces. We will talk about interfaces in our later article.

Using System;

Public class A

{

    Public void A\_Method ()

    {

        Console.WriteLine ("Class A Method Called");

    }

}

Public class B: A

{

    Public void B\_Method ()

    {

        Console.WriteLine ("Class A Method Called");

    }

}

Public class C: B

{

    Public void C\_Method ()

    {

        Console.WriteLine ("Class A Method Called");

    }

    Public static void Main ()

    {

        C C1 = new C ();

        C1.A\_Method ();

        C1.B\_Method ();

        C1.C\_Method ();

    }

}

When you derive a class from a base class, the derived class will inherit all members of the base class except constructors. In the code snippet below class B will inherit both M1 and M2 from Class A, but you cannot access M2 because of the private access modifier. Class members declared with a private access modifier can be accessed only with in the class. We will talk about access modifiers in our later article.   
  
**Common Interview Question:** Are private class members inherited to the derived class?

**Ans:** Yes, the private members are also inherited in the derived class but we will not be able to access them. Trying to access a private base class member in the derived class will report a compile time error.

Using System;

Public class A

{

Public void M1 ()

{

}

Private void M2 ()

{

}

}

Public class B: A

{

Public static void Main ()

{

B B1 = new B ();

B1.M1 ();

//Error, Cannot access private member M2

//B1.M2 ();

}

}

Method Hiding and Inheritance We will look at an example of how to **hide** a method in C#. The Parent class has a write () method which is available to the child class. In the child class I have created a new write () method. So, now if I create an instance of child class and call the write () method, the child class write () method will be called. The child class is hiding the base class write () method. This is called method hiding.   
  
If we want to call the parent class write () method, we would have to type cast the child object to Parent type and then call the write () method as shown in the code snippet below.

Using System;

Public class Parent

{

    Public void Write ()

    {

        Console.WriteLine ("Parent Class write method");

    }

}

Public class Child: Parent

{

    Public new void Write ()

    {

        Console.WriteLine ("Child Class write method");

    }

    Public static void Main ()

    {

        Child C1 = new Child ();

        C1.Write ();

        //Type caste C1 to be of type Parent and call Write () method

        ((Parent) C1).Write ();

    }

}

**Polymorphism:**

When a message can be processed in different ways is called polymorphism. Polymorphism means many forms.

Polymorphism is one of the fundamental concepts of OOP.

**Polymorphism provides following features:**

* It allows you to invoke methods of derived class through base class reference during runtime.
* It has the ability for classes to provide different implementations of methods that are called through the same name.

**Polymorphism is of two types:**

1. Compile time polymorphism/Overloading
2. Runtime polymorphism/Overriding

**Compile Time Polymorphism**

Compile time polymorphism is method and operators overloading. It is also called early binding.

In method overloading method performs the different task at the different input parameters.

**Runtime Time Polymorphism**

Runtime time polymorphism is done using inheritance and virtual functions. Method overriding is called runtime polymorphism. It is also called late binding.

When **overriding** a method, you change the behavior of the method for the derived class.  **Overloading**a method simply involves having another method with the same prototype.

**Caution:**Don't confused method overloading with method overriding, they are different, unrelated concepts. But they sound similar.

Method overloading has nothing to do with inheritance or virtual methods.

**Following are examples of methods having different overloads:**

void area(int side);

void area(int l, int b);

void area(float radius);

**Practical example of Method Overloading (Compile Time Polymorphism)**

using System;

namespace method\_overloading

{

    class Program

    {

        public class Print

        {

            public void display(string name)

            {

                Console.WriteLine ("Your name is : " + name);

            }

            public void display(int age, float marks)

            {

                Console.WriteLine ("Your age is : " + age);

                Console.WriteLine ("Your marks are :" + marks);

            }

        }

        static void Main(string[] args)

        {

            Print obj = new Print ();

            obj.display ("George");

            obj.display (34, 76.50f);

            Console.ReadLine ();

        }

    }

}

**Note:**In the code if you observe display method is called two times. Display method will work according to the number of parameters and type of parameters.

**When and why to use method overloading**

Use method overloading in situation where you want a class to be able to do something, but there is more than one possibility for what information is supplied to the method that carries out the task.

You should consider overloading a method when you for some reason need a couple of methods that take different parameters, but conceptually do the same thing.

**Method overloading showing many forms.**

using System;

namespace method\_overloading\_polymorphism

{

    Class Program

    {

        Public class Shape

        {

            Public void Area (float r)

            {

                float a = (float)3.14 \* r;

                // here we have used function overload with 1 parameter.

                Console.WriteLine ("Area of a circle: {0}",a);

            }

            Public void Area(float l, float b)

            {

                float x = (float)l\* b;

                // here we have used function overload with 2 parameters.

                Console.WriteLine ("Area of a rectangle: {0}",x);

            }

            public void Area(float a, float b, float c)

            {

                float s = (float)(a\*b\*c)/2;

                // here we have used function overload with 3 parameters.

                Console.WriteLine ("Area of a circle: {0}", s);

            }

        }

        Static void Main (string[] args)

        {

            Shape ob = new Shape ();

            ob.Area(2.0f);

            ob.Area(20.0f,30.0f);

            ob.Area(2.0f,3.0f,4.0f);

            Console.ReadLine ();

        }

    }

}

**Things to keep in mind while method overloading**

If you use overload for method, there are couple of restrictions that the compiler imposes.

The rule is that overloads must be different in their signature, which means the name and the number and type of parameters.

There is no limit to how many overload of a method you can have. You simply declare them in a class, just as if they were different methods that happened to have the same name.

**Method Overriding:**

Whereas **Overriding** means changing the functionality of a method without changing the signature. We can override a function in base class by creating a similar function in derived class. This is done by using virtual/override keywords.  
  
Base class method has to be marked with virtual keyword and we can override it in derived class using override keyword.  
  
Derived class method will completely overrides base class method i.e. when we refer base class object created by casting derived class object a method in derived class will be called.

Example:

// Base class  
public class BaseClass  
{  
public virtual void Method1()  
{  
Console.Write("Base Class Method");  
}  
}  
// Derived class  
public class DerivedClass : BaseClass  
{  
public override void Method1()  
{  
Console.Write("Derived Class Method");  
}  
}  
// Using base and derived class  
public class Sample  
{  
public void TestMethod()  
{  
// calling the overriden method  
DerivedClass objDC = new DerivedClass();   
objDC.Method1();  
 // calling the baesd class method  
BaseClass objBC = (BaseClass)objDC;   
objDC.Method1();  
}  
}

Output  
---------------------

Derived Class Method

Derived Class Method

**Constructors and Destructors:**

Classes have complicated internal structures, including data and functions, object initialization and cleanup for classes is much more complicated than it is for simple data structures. Constructors and destructors are special member functions of classes that are used to construct and destroy class objects. Construction may involve memory allocation and initialization for objects. Destruction may involve cleanup and deallocation of memory for objects.

* Constructors and destructors do not have return types nor can they return values.
* References and pointers cannot be used on constructors and destructors because their addresses cannot be taken.
* Constructors cannot be declared with the keyword virtual.
* Constructors and destructors cannot be declared const, or volatile.
* Unions cannot contain class objects that have constructors or destructors.

Constructors and destructors obey the same access rules as member functions. For example, if you declare a constructor with protected access, only derived classes and friends can use it to create class objects.

The compiler automatically calls constructors when defining class objects and calls destructors when class objects go out of scope. A constructor does not allocate memory for the class object it’s this pointer refers to, but may allocate storage for more objects than its class object refers to. If memory allocation is required for objects, constructors can explicitly call the new operator. During cleanup, a destructor may release objects allocated by the corresponding constructor. To release objects, use the delete operator.

**Example of Constructor**

class C

{

       private int x;

       private int y;

       public C (int i, int j)

       {

                 x = i;

                 y = j;

       }

       public void display ()

       {

               Console.WriteLine(x + "i+" + y);

       }

}  
  
**Example of Destructor**

class D

{

        public D ()

        {

            // constructor

        }

        ~D ()

        {

           // Destructor

        }

}

**Question 1: What is ASP.NET?**  
Answer: ASP.NET was developed in direct response to the problems that developers had with classic ASP. Since ASP is in such wide use, however, Microsoft ensured that ASP scripts execute without modification on a machine with the .NET Framework (the ASP engine, ASP.DLL, is not modified when installing the .NET Framework). Thus, IIS can house both ASP and ASP.NET scripts on the same machine.  
  
**Advantages of ASP.NET**

1. **Separation of Code from HTML:**  
     
   To make a clean sweep, with ASP.NET you have the ability to completely separate layout and business logic. This makes it much easier for teams of programmers and designers to collaborate efficiently.
2. **Support for compiled languages:**  
   Developer can use VB.NET and access features such as strong typing and object-oriented programming. Using compiled languages also means that ASP.NET pages do not suffer the performance penalties associated with interpreted code. ASP.NET pages are precompiled to byte-code and Just In Time (JIT) compiled when first requested. Subsequent requests are directed to the fully compiled code, which is cached until the source changes.
3. **Use services provided by the .NET Framework:**  
   The .NET Framework provides class libraries that can be used by your application. Some of the key classes help you with input/output, access to operating system services, data access, or even debugging. We will go into more detail on some of them in this module.
4. **Graphical Development Environment:**  
   Visual Studio .NET provides a very rich development environment for web developers. You can drag and drop controls and set properties the way you do in Visual Basic 6. And you have full IntelliSense support, not only for your code, but also for HTML and XML.
5. **State management:**  
   To refer to the problems mentioned before, ASP.NET provides solutions for session and application state management. State information can, for example, be kept in memory or stored in a database. It can be shared across web farms, and state information can be recovered, even if the server fails or the connection breaks down.
6. **Update files while the server is running:**  
   Components of your application can be updated while the server is online and clients are connected. The framework will use the new files as soon as they are copied to the application. Removed or old files that are still in use are kept in memory until the clients have finished.
7. **XML-Based Configuration Files:**  
   Configuration settings in ASP.NET are stored in XML files that you can easily read and edit. You can also easily copy these to another server, along with the other files that comprise your application.

**ASP.NET Overview**  
Here are some points that give the quick overview of ASP.NET.

* ASP.NET provides services to allow the creation, deployment, and execution of Web Applications and Web Services.
* Like ASP, ASP.NET is a server-side technology.
* Web Applications are built using Web Forms. ASP.NET comes with built-in Web Forms controls, which are responsible for generating the user interface. They mirror typical HTML widgets like text boxes or buttons. If these controls do not fit your needs, you are free to create your own user controls.
* Web Forms are designed to make building web-based applications as easy as building Visual Basic applications.

**For further information click on the link:**

* [Introduction to ASP.NET](http://www.c-sharpcorner.com/UploadFile/sudhir1/ASPIntro11282005012802AM/ASPIntro.aspx)

**Question 2: What are the different validators in ASP.NET?**  
**Answer:** ASP.NET validation controls define an important role in validating the user input data. Whenever the user gives the input, it must always be validated before sending it across to various layers of an application. If we get the user input with validation, then chances are that we are sending the wrong data. So, validation is a good idea to do whenever we are taking input from the user.  
  
There are the following two types of validation in ASP.NET:

* Client-Side Validation
* Server-Side Validation

**Client-Side Validation:**

When validation is done on the client browser, then it is known as Client-Side Validation. We use JavaScript to do the Client-Side Validation.  
  
**Server-Side Validation:**  
When validation occurs on the server, then it is known as Server-Side Validation. Server-Side Validation is a secure form of validation. The main advantage of Server-Side Validation is if the user somehow bypasses the Client-Side Validation, we can still catch the problem on server-side.  
  
The following are the Validation Controls in ASP.NET:

* RequiredFieldValidator Control
* CompareValidator Control
* RangeValidator Control
* RegularExpressionValidator Control
* CustomFieldValidator Control
* ValidationSummary

**For further information click on the link:**

* [Validation Controls in ASP.Net](http://www.c-sharpcorner.com/UploadFile/ca2535/validation-controls-in-Asp-Net/)

**Question 3: What is View State?**  
**Answer:** 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.   
  
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

**Features of View State**  
These are the main features of view state:

1. Retains the value of the Control after post-back without using a session.
2. Stores the value of Pages and Control Properties defined in the page.
3. 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**

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

**For further information click on the link:**

* [What is View State and How it Works in ASP.NET](http://www.c-sharpcorner.com/UploadFile/225740/what-is-view-state-and-how-it-works-in-Asp-Net53/)

**Question 4: What are the different Session state management options available in ASP.NET?**  
**Answer:**State Management in ASP.NET

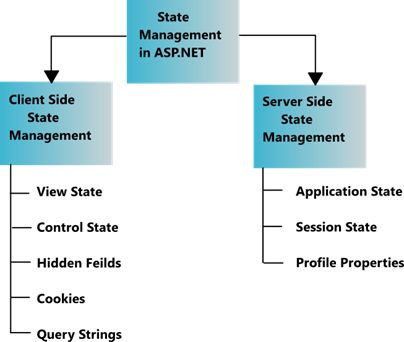
* A new instance of the Web page class is created each time the page is posted to the server.
* In traditional Web programming, all information that is associated with the page, along with the controls on the page, would be lost with each roundtrip.
* The Microsoft ASP.NET framework includes several options to help you preserve data on both a per-page basis and an application-wide basis.   
  These options can be broadly divided into the following two categories:   
    
  + Client-Side State Management Options
  + Server-Side State Management Options

**Client-Side State Management**

* Client-based options involve storing information either in the page or on the client computer.
* Some client-based state management options are:  
    
  + Hidden fields
  + View state
  + Cookies
  + Query strings

**Server-Side State Management**

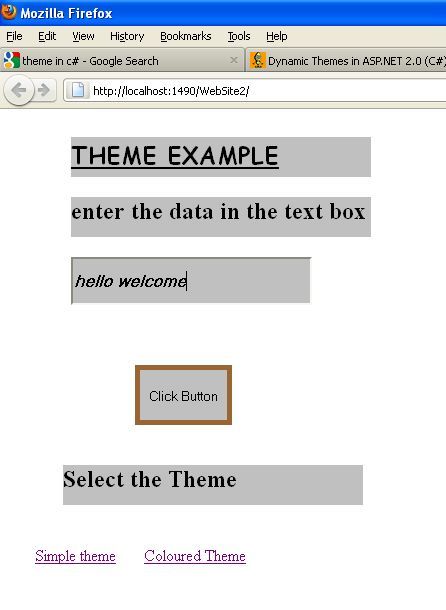
* There are situations where you need to store the state information on the server side.
* Server-side state management enables you to manage application-related and session-related information on the server.
* ASP.NET provides the following options to manage state at the server side:  
    
  + Application state
  + Session state

****For further information click on the link:

* [State Management in ASP.Net](http://www.c-sharpcorner.com/UploadFile/PravinkumarVB/state-management-in-Asp-Net/)

**Question 5: What is caching in ASP.NET?**  
**Answer:** Caching is one of the most interesting concept and operation in ASP.NET. If you can handle it, you can run any web application by applying the caching concept depending on the requirements.  
  
Caching is for providing solutions or the results to the users depending on their request, admin needs to recreate the pages often depending on user requests…STOP!!! "A cache simply stores the output generated by a page in the memory and this saved output (cache) will serve us (users) in the future.".  
  
**Types**  
  
  
  
For further information click on the link:

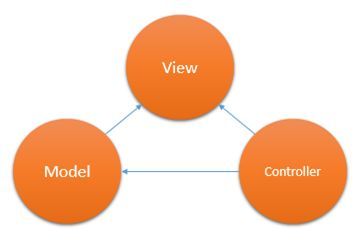
* [Caching in ASP.Net](http://www.c-sharpcorner.com/UploadFile/2072a9/caching-in-Asp-Net/)

**Question 6: How can we apply themes in ASP.NET application?**  
**Answer:** A theme is a collection of settings that define the look of controls and web pages. These themes are applied across all the pages in a web application to maintain a consistent appearance. Themes are included images and skin files; the skin files set the visual properties of ASP.NET controls. Themes are of two types:  
  
**Page Theme**  
A Page theme contains the control skins, style sheets, graphic files, and other resources inside the subfolder of the App\_Theme folder in the Solution Explorer window. A page theme is applied to a single page of the web site.   
  
**Global Theme**  
A Global theme is a theme that is applied to all the web sites on a web server and includes property settings, and graphics. This theme allows us to maintain all the websites on the same web server and define the same style for all the web pages of the web sites.  
  
  
  
For further information click on the link:

* [Creating Web Application Using Themes in ASP.NET](http://www.c-sharpcorner.com/UploadFile/a0a81e/3/)

**Question 7: What is MVC?**  
  
**Answer:** Model-View-Controller (MVC) is a pattern to separate an application into the following three main components:

1. Model
2. View
3. Controller

The ASP.NET MVC framework provides an alternative to the ASP.NET Web Forms pattern for creating web applications. The ASP.NET MVC Framework is a lightweight, highly testable presentation framework that (as with Web Forms-based applications) is integrated with existing ASP.NET features, such as master pages and membership-based authentications. The MVC framework is defined in the **System.Web.Mvc**assembly. It provides full control over HTML, JavaScript and CSS. It's the better as well as a recommended approach for large-scale applications where various teams are working together.  
  
   
  
The ASP.NET MVC framework offers the following advantages:

* It makes it very easy to manage complexity by dividing an application into the Model, View and Controller.
* It does not use view state or server-based forms.
* Full control over HTML, JavaScript and CSS.
* It provides better support for Test-Driven Development (TDD).
* It works well for Web applications that are supported by large teams of developers and for web designers who need a high degree of control over the application behaviour.
* By default support of Facebook and Google Authentication.
* It easy to manage a large application to divide in multiple areas.

For further information click on the link:

* [Getting Started With ASP.Net MVC](http://www.c-sharpcorner.com/UploadFile/8ef97c/getting-started-with-Asp-Net-mvc/)

**Question 8: What are Cookies in ASP.NET?**  
Answer: Cookies are a State Management Technique that can store the values of control after a post-back. Cookies can store user-specific Information on the client's machine like when the user last visited your site. Cookies are also known by many names, such as HTTP Cookies, Browser Cookies, Web Cookies, Session Cookies and so on. Basically cookies are a small text file sent by the web server and saved by the Web Browser on the client's machine.  
  
List of properties containing the HttpCookies Class:

1. **Domain:** Using these properties we can set the domain of the cookie.
2. **Expires:** This property sets the Expiration time of the cookies.
3. **HasKeys:**If the cookies have a subkey then it returns True.
4. **Name:**Contains the name of the Key.
5. **Path:**Contains the Virtual Path to be submitted with the Cookies.
6. **Secured:** If the cookies are to be passed in a secure connection then it only returns True.
7. **Value:**Contains the value of the cookies.

**Limitation of the Cookies**

1. The size of cookies is limited to 4096 bytes.
2. A total of 20 cookies can be used in a single website.

For further info click on the link:

* [Introduction To Cookies in ASP.Net](http://www.c-sharpcorner.com/UploadFile/225740/cookies/)

**Question 9: What is Ajax in ASP.NET?**  
Answer. Ajax stands for Asynchronous JavaScript and XML; in other words Ajax is the combination of various technologies such as a JavaScript, CSS, XHTML, DOM, etc.  
  
AJAX allows web pages to be updated asynchronously by exchanging small amounts of data with the server behind the scenes. This means that it is possible to update parts of a web page, without reloading the entire page.  
  
We can also define Ajax is a combination of client side technologies that provides asynchronous communication between the user interface and the web server so that partial page rendering occurs instead of complete page post back.  
  
Ajax is platform-independent; in other words AJAX is a cross-platform technology that can be used on any Operating System since it is based on XML & JavaScript. It also supports open source implementation of other technology. It partially renders the page to the server instead of complete page post back. We use AJAX for developing faster, better and more interactive web applications. AJAX uses a HTTP request between web server & browser.

* With AJAX, when a user clicks a button, you can use JavaScript and DHTML to immediately update the UI, and spawn an asynchronous request to the server to fetch results.
* When the response is generated, you can then use JavaScript and CSS to update your UI accordingly without refreshing the entire page. While this is happening, the form on the users screen doesn't flash, blink, disappear, or stall.
* The power of AJAX lies in its ability to communicate with the server asynchronously, using a XMLHttpRequest object without requiring a browser refresh.
* Ajax essentially puts JavaScript technology and the XMLHttpRequest object between your Web form and the server.

For further info click on the link:

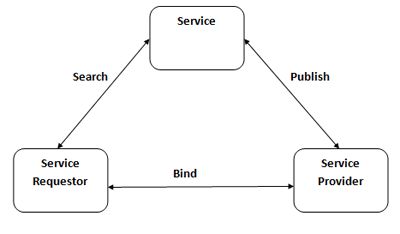
* [Introduction to Ajax and Ajax Control Toolkit](http://www.c-sharpcorner.com/UploadFile/0c1bb2/introduction-to-ajax-and-ajax-control-toolkit-part-1/)

**Question 10: What are Web Services in ASP.NET?**  
**Answer:** A Web Service is a software program that uses XML to exchange information with other software via common internet protocols. In a simple sense, Web Services are a way for interacting with objects over the Internet.  
  
**A web service is:**

* Language Independent.
* Protocol Independent.
* Platform Independent.
* It assumes a stateless service architecture.
* Scalable (e.g. multiplying two numbers together to an entire customer-relationship management system).
* Programmable (encapsulates a task).
* Based on XML (open, text-based standard).
* Self-describing (metadata for access and use).
* Discoverable (search and locate in registries)- ability of applications and developers to search for and locate desired Web services through registries. This is based on UDDI.

Key Web Service Technologies:

* **XML-** Describes only data. So, any application that understands XML-regardless of the application's programming language or platform-has the ability to format XML in a variety of ways (well-formed or valid).
* **SOAP-** Provides a communication mechanism between services and applications.
* **WSDL-**Offers a uniform method of describing web services to other programs.
* **UDDI-** Enables the creation of searchable Web services registries.

  
  
For further info click on the link:

* [ASP.NET Web Service Basics](http://www.c-sharpcorner.com/UploadFile/1d42da/web-service-basics/)

**Question 11: What are the Advantages of ASP.NET?**  
**Answer:**ASP.NET provides services to allow the creation, deployment, and execution of Web Applications and Web Services like ASP, ASP.NET is a server-side technology. Web Applications are built using Web Forms. ASP.NET comes with built-in Web Form controls, which are responsible for generating the user interface. They mirror typical HTML widgets such as text boxes or buttons. If these controls do not fit your needs, you are free to create your own user controls.  
 **Advantages of ASP.NET:**

* Separation of Code from HTML
* Support for compiled languages
* Use services provided by the .NET Framework
* Graphical Development Environment
* Update files while the server is running
* XML-Based Configuration Files

  
  
For further info click on the link:

* [Introduction to ASP.NET](http://www.c-sharpcorner.com/UploadFile/sudhir1/ASPIntro11282005012802AM/ASPIntro.aspx)

**Question 12: What is the concepts of Globalization and Localization in .NET?**  
**Answer:** Localization means "**process of translating resources for a specific culture**", and Globalization means "**process of designing applications that can adapt to different cultures**".

* **Proper Globalization:**Your application should be able to Accept, Verify, and Display all global kind of data. It should well also be able to operate over this data, accordingly. We will discuss more about this "**Accordingly operations over diff. culture data**".
* **Localizability and Localization:** Localizability stands for clearly separating the components of culture based operations regarding the user interface, and other operations from the executable code.

.NET framework has greatly simplified the task of creating the applications targeting the clients of multiple cultures. The namespaces involved in creation of globalize, localizing applications are:

* System.Globalization
* System.Resources
* System.Text

For further info click on the link:

* [Globalization and Localization in .NET: Part I](http://www.c-sharpcorner.com/UploadFile/brijrajsingh/glbal_110012005021049AM/glbal_1.aspx)

**Question 13: What is the Web.config file in ASP?**  
**Answer:** Configuration file is used to manage various settings that define a website. The settings are stored in XML files that are separate from your application code. In this way you can configure settings independently from your 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.  
  
**Usage of configuration file**  
  
ASP.NET Configuration system is used to describe the properties and behaviors of various aspects of ASP.NET applications. Configuration files help you to manage the settings related to your website. Each file is an XML file (with the extension .config) that contains a set of configuration elements. Configuration information is stored in XML-based text files.   
  
**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.
* You 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.
* You can use any standard text editor or XML parser to create and edit ASP.NET configuration files.

For further info click on the link:

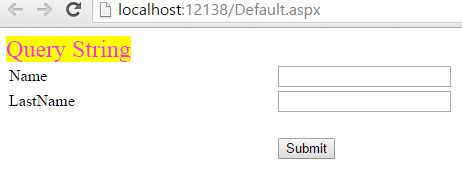
* [ASP.NET Web Configuration File](http://www.c-sharpcorner.com/uploadfile/puranindia/Asp-Net-web-configuration-file/)

**Question 14: What is the App Domain Concept in ASP.NET?**  
**Answer:** ASP.NET introduces the concept of an Application Domain which is shortly known as **AppDomain**. It can be considered as a Lightweight process which is both a container and boundary. The .NET runtime uses an **AppDomain** as a container for code and data, just like the operating system uses a process as a container for code and data. As the operating system uses a process to isolate misbehaving code, the .NET runtime uses an **AppDomain** to isolate code inside a secure boundary.  
  
The CLR can allow the multiple .NET applications to run in a single AppDomain. Mulitple Appdomains can exist in Win32 process.  
  
**How to create AppDomain:** AppDomains are created using the CreateDomain method. AppDomain instances are used to load and execute assemblies (Assembly). When an AppDomain is no longer in use, it can be unloaded.

1. **public** **class** MyAppDomain: MarshalByRefObject
2. {
3. **public** **string** GetInfo()
4. {
5. **return** AppDomain.CurrentDomain.FriendlyName;
6. }
7. }
8. **public** **class** MyApp
9. {
10. **public** **static** **void** Main()
11. {
12. AppDomain apd = AppDomain.CreateDomain("Rajendrs Domain");
13. MyAppDomain apdinfo = (MyAppDomain) apd.CreateInstanceAndUnwrap(Assembly.GetCallingAssembly()
14. .GetName()
15. .Name, "MyAppDomain");
16. Console.WriteLine("Application Name = " + apdinfo.GetInfo());
17. }
18. }

For further info click on the link:

* [AppDomain concept in ASP.Net](http://www.c-sharpcorner.com/UploadFile/nagryum/Appdomain07102007081415AM/Appdomain.aspx)

**Question 15: What is Query String in ASP?**  
**Answer:**A QueryString is a collection of characters input to a computer or web browser. A Query String is helpful when we want to transfer a value from one page to another. When we need to pass content between the HTML pages or aspx Web Forms in the context of ASP.NET, a Query String is Easy to use and the Query String follows a separating character, usually a Question Mark (?). It is basically used for identifying data appearing after this separating symbol. A Query String Collection is used to retrieve the variable values in the HTTP query string. If we want to transfer a large amount of data then we can't use the Request.QueryString. Query Strings are also generated by form submission or can be used by a user typing a query into the address bar of the browsers.  
  
**Syntax of Query String**  
*Request.QueryString(variable)[(index).count]*  
  
  
  
**Advantages:**

* Simple to Implement
* Easy to get information from Query string.
* Used to send or read cross domain (from different domain).

**Disadvantages:**

* Human Readable
* Client browser limit on URL length
* Cross paging functionality makes it redundant
* Easily modified by end user

For further info click on the link:

* [Query Strings in ASP.Net](http://www.c-sharpcorner.com/UploadFile/ca2535/query-string-in-Asp-Net/)

**Question 16: What is master page in ASP.NET?**  
Answer: The extension of MasterPage is '**.master**'. MasterPage cannot be directly accessed from the client because it just acts as a template for the other Content Pages. In a MasterPage we can have content either inside ContentPlaceHolder or outside it. Only content inside the **ContentPlaceHolder**can be customized in the Content Page. We can have multiple masters in one web application.A MasterPage can have another MasterPage as Master to it. The MasterPageFile property of a webform can be set dynamically and it should be done either in or before the Page\_PreInit event of the WebForm. **Page.MasterPageFile = "MasterPage.master".**The dynamically set Master Page must have the ContentPlaceHolder whose content has been customized in the WebForm.  
  
A master page is defined using the following code:  
  
*<%@ master language="C#" %>*  
  
**Adding a MasterPage to the Project**

1. Add a new MasterPage file (MainMaster.master) to the Web Application.
2. Change the Id of ContentPlaceHolder in <Head> to "**cphHead**" and the Id "**ContentPlaceHolder1**" to "**cphFirst**".
3. Add one more ContentPlaceHolder (cphSecond) to Master page.
4. To the master page add some header, footer and some default content for both the content place holders.
   1. <form id="form1" runat="server"> Header...
   2. <br />
   3. <asp:ContentPlaceHolder id="cphFirst" runat="server"> This **is** First Content Place Holder (Default) </asp: ContentPlaceHolder>
   4. <br />
   5. <asp:ContentPlaceHolder ID="cphSecond" runat="server">

This is Second Content Place Holder (Default).

1. </asp:ContentPlaceHolder>
2. <br /> Footer...
3. </form>

For further info click on the link:

* [Master Page in ASP.NET](http://www.c-sharpcorner.com/UploadFile/4ed419/master-page-in-Asp-Net/)

**Question 17: What is tracing in .NET?**  
**Answer:** Tracing helps to see the information of issues at the runtime of the application. By default Tracing is disabled.  
  
Tracing has the following important features:

1. We can see the execution path of the page and application using the debug statement.
2. We can access and manipulate trace messages programmatically.
3. We can see the most recent tracing of the data.

Tracing can be done with the following 2 types.

1. **Page Level:**When the trace output is displayed on the page and for the page-level tracing we need to set the property of tracing at the page level.  
     
   *<%@ Page Trace="true" Language="C#"*
2. **Application:** Level: In Application-Level tracing the information is stored for each request of the application. The default number of requests to store is 10. But if you want to increase the number of requests and discard the older request and display a recent request then you need to set the property in the web.config file.  
     
   *<trace enabled="true"/>*

For further info click on the link:

* [Introduction to Tracing in .Net](http://www.c-sharpcorner.com/UploadFile/225740/introduction-to-tracing/)

**Question 18: What are the data controls available in ASP.NET?**  
**Answer:** The Controls having DataSource Property are called Data Controls in ASP.NET. ASP.NET allows powerful feature of data binding, you can bind any server control to simple properties, collections, expressions and/or methods. When you use data binding, you have more flexibility when you use data from a database or other means.  
  
Data Bind controls are container controls.  
  
*Controls -> Child Control*  
  
Data Binding is binding controls to data from databases. With data binding we can bind a control to a particular column in a table from the database or we can bind the whole table to the data grid.   
  
Data binding provides simple, convenient, and powerful way to create a read/write link between the controls on a form and the data in their application.   
  
Data binding allows you to take the results of properties, collection, method calls, and database queries and integrate them with your ASP.NET code. You can combine data binding with Web control rendering to relieve much of the programming burden surrounding Web control creation. You can also use data binding with ADO.NET and Web controls to populate control contents from SQL select statements or stored procedures.  
  
**Data binding uses a special syntax:**  
  
*<%# %>*  
  
The *<%#,* which instructs ASP.NET to evaluate the expression. The difference between a data binding tags and a regular code insertion tags <% and %> becomes apparent when the expression is evaluated. Expressions within the data binding tags are evaluated only when the DataBind method in the Page objects or Web control is called.  
  
Data Bind Control can display data in connected and disconnected model.  
  
Following are data bind controls in ASP.NET:

* Repeater Control
* DataGrid Control
* DataList Control
* GridView Control
* DetailsView
* FormView
* DropDownList
* ListBox
* RadioButtonList
* CheckBoxList
* BulletList etc.

For further info click on the link:

* [Data Bind Controls in ASP.NET](http://www.c-sharpcorner.com/uploadfile/puranindia/data-bind-controls-in-Asp-Net/)

**Question 19: What are the major events in global.aspx?**  
**Answer:** The Global.asax file, which is derived from the HttpApplication class, maintains a pool of HttpApplication objects, and assigns them to applications as needed. The Global.asax file contains the following events:

* *Application\_Init*
* *Application\_Disposed*
* *Application\_Error*
* *Application\_Start*
* *Application\_End*
* *Application\_BeginReques*

For further info click on the link:

* [Major Events in GLOBAL.ASAX file](http://www.c-sharpcorner.com/uploadfile/aa04e6/major-events-in-global-asax-file/)

**Question 20: Use of CheckBox in .NET?**  
**Answer:**The CheckBox control is a very common control of HTML, unlike radio buttons it can select multiple items on a webpage. The CheckBox control in ASP.NET has many properties and some of them are listed below.

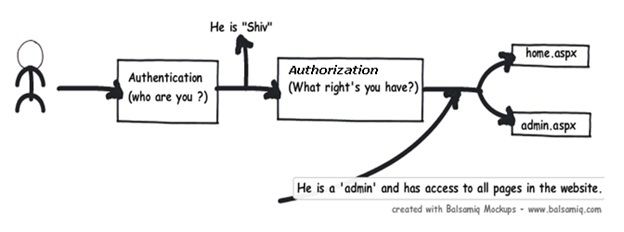
|  |  |
| --- | --- |
| **Property** | **Description** |
| AutoPostBack | Specifies whether the form should be posted immediately after the Checked property has changed or not. The default is false. |
| CausesValidation | Specifies if a page is validated when a Button control is clicked. |
| Checked | Specifies whether the check box is checked or not. |
| InputAttributes | Attribute names and values used for the Input element for the CheckBox control. |
| LabelAttributes | Attribute names and values used for the Label element for the CheckBox control. |
| runat | Specifies that the control is a server control. Must be set to "server". |
| Text | The text next to the check box. |
| TextAlign | On which side of the check box the text should appear (right or left). |
| ValidationGroup | Group of controls for which the Checkbox control causes validation when it posts back to the server. |
| OnCheckedChanged | The name of the function to be executed when the Checked property has changed. |

For further info click on the link:

* [Use CheckBox inside Gridview in ASP.NET](http://www.c-sharpcorner.com/UploadFile/009464/use-checkbox-inside-gridview-in-Asp-Net/)

**Question 21: What is the authentication and authorization in ASP.NET?**  
  
**Answer**

* **Authentication:**Prove genuineness
* **Authorization:**process of granting approval or permission on resources.

In ASP.NET authentication means to identify the user or in other words its nothing but to validate that he exists in your database and he is the proper user.   
  
Authorization means does he have access to a particular resource on the IIS website. A resource can be an ASP.NET web page, media files (MP4, GIF, JPEG etc), compressed file (ZIP, RAR) etc.  
  
  
  
**Types of authentication and authorization in ASP.NET**  
There are three ways of doing authentication and authorization in ASP.NET:

* **Windows authentication:** In this methodology ASP.NET web pages will use local windows users and groups to authenticate and authorize resources.
* **Forms Authentication:**This is a cookie based authentication where username and password are stored on client machines as cookie files or they are sent through URL for every request. Form-based authentication presents the user with an HTML-based Web page that prompts the user for credentials.
* **Passport authentication:** Passport authentication is based on the passport website provided by the Microsoft .So when user logins with credentials it will be reached to the passport website ( i.e. hotmail,devhood,windows live etc) where authentication will happen. If Authentication is successful it will return a token to your website.
* **Anonymous access:** If you do not want any kind of authentication then you will go for Anonymous access.

In 'web.config' file set the authentication mode to 'Windows' as shown in the below code snippets.

1. <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 unknown user.

1. <authorization>
2. <deny users="?"/>
3. </authorization>

For further info click on the link:

* [ASP.NET Authentication and Authorization](http://www.c-sharpcorner.com/uploadfile/shivprasadk/Asp-Net-authentication-and-authorization/)

**Question 22: What are the HTML server control in ASP.NET?**  
**Answer:** The Microsoft.NET Framework provides a rich set of server-side controls for developing Web applications. You can add these controls to WebForms pages just as you add Windows controls to a form. Server-side controls are often called server controls or Web Forms controls. There are four types of Server controls: HTML server controls. Web server controls, validation control, and user controls.

* **HTML Server controls**  
  HTML developers must be familiar with old HTML controls, which they use to write GUI applications in HTML. These controls are the same HTML controls; you can run these controls on the server by defining the runat ="server" attribute. These control names start with Html.

|  |  |
| --- | --- |
| **Controls** | **Description** |
| HtmlForm | Create an HTML form control, used as a place holder of other controls. |
| HtmlInputText | Creates an input text box control used to get input from user. |
| HtmltextArea | Creates multiline text box control. |
| HtmlAnchor | Creates a Web navigation. |
| HtmlButton | Creates a button control. |
| HtmlImage | Creates an image control, which is used to display an image. |
| HtmlInputCheckBox | Creates a check box control. |
| HtmlInputRadioButton | Creates a radio button control. |
| HtmlTable | Creates a table control. |
| HtmlTableRow | Creates a row within a table. |
| HtmlTableCell | Creates a cell with in a row. |

* + Web Server Controls
  + Validation Controls
  + User Controls

For further info click on the link:

* [ASP .NET Server-Side controls](http://www.c-sharpcorner.com/UploadFile/puranindia/asp-net-server-side-controls/)

**Question 23: What are the authentication modes in ASP.NET for security?**  
  
**Answer:**When you begin a program for a customer using ASP.NET, you should consider about security. Security is one of the most important components of any application. Security is even more important when you are making a web application which is exposed to million of users. ASP.NET provides classes and methods that ensure that the application is secure from outside attacks. In this article we will investigate the different types of authentication provided by ASP.NET. In web.config file you can set authentication mode value 'windows' or 'forms'. What's about difference and how to you use them? (Authentication have some other values to, this article does not consider them.).  
  
**How to use mode "Windows"?**  
Windows Authentication mode provides the developer to authenticate a user based on Windows user accounts. This is the default authentication mode provider by ASP.NET. This will return the computer name along with the user name.

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

**How to use mode "Forms"?**  
Insert the*<Forms>* tag, and fill the appropriate attributes.

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

**For further info click on the link:**

* [Authentication Modes in ASP.Net for Security](http://www.c-sharpcorner.com/UploadFile/kaps_deo/Authentication07242007033135AM/Authentication.aspx)

**Question 24: What is the web API in ASP.NET?**  
  
Answer: It is a framework provided by Microsoft for writing HTTP services. There are many frameworks available to build HTTP based services. They follow a common guideline of international standardization but with different flavors.   
  
For example, all framework must adhere to these status codes-

* **1xx** - Informational Message
* **2xx** - Successful
* **3xx** - Redirection
* **4xx** - Client Error
* **5xx** - Server Error

**Features:**

* It is light weight and thus good for small devices also like tablets, smart phones.
* No tedious & extensive configuration like WCF REST is required.
* MediaTypeFormatter makes easy to configure your APIs response type in single line (JSON, XML and so on).
* IIS Hosting dependency is no more and it can be hosted in application too.
* Easy and simple control with HTTP features such as Caching, Versioning, request/response headers and its various content formats.
* It support content-negotiation (deciding the best response data format that client can accept).

**For further info click on the link:**

* [Basic Understanding On ASP.NET Web API](http://www.c-sharpcorner.com/UploadFile/bc1c71/basic-understanding-on-Asp-Net-web-api/)

**Question 25: Describe application state management in ASP.NET?**  
  
**Answer:**Application Level State Management is used to maintain the state of all the users accessing the web forms present within the website.  
  
The value assigned for an application is considered as an object.  
  
Application object will not have any default expiration period.  
  
Whenever the webserver has been restarted or stopped then the information maintained by the application object will be lost.  
  
If any data is stored on the application object then that information will be shared upon all the users accessing the webserver.  
  
Since the information is shared among all the users, it is advisable to lock and unlock the application object as per requirement.  
  
**Global Application Class(Global.asax):**  
  
It is a Class which consists of event handlers which executes the code implicitly whenever a relevant task has been performed on the web server.**Design**:

1. <%@ Application Language="C#" %>
2. <script runat="server">
3. **void** Application\_Start(**object** sender, EventArgs e)
4. {
5. // Code that runs on application startup
6. }
7. **void** Application\_End(**object** sender, EventArgs e)
8. {
9. // Code that runs on application shutdown
10. }
11. **void** Application\_Error(**object** sender, EventArgs e)
12. {
13. // Code that runs when an unhandled error occurs
14. }
15. **void** Session\_Start(**object** sender, EventArgs e)
16. {
17. // Code that runs when a new session is started
18. }
19. **void** Session\_End(**object** sender, EventArgs e)
20. {
21. // Code that runs when a session ends.
22. }
23. </script>

For further info click on the link:

* [Application Level State Management in ASP.NET](http://www.c-sharpcorner.com/uploadfile/e628d9/application-level-state-management-in-Asp-Net/)

**Question 26: What is the code behind and Inline Code?**  
  
**Answer:**  
**Code Behind**  
Code Behind refers to the code for an ASP.NET Web page that is written in a separate class file that can have the extension of .aspx.cs or .aspx.vb depending on the language used. Here the code is compiled into a separate class from which the .aspx file derives. You can write the code in a separate .cs or .vb code file for each .aspx page. One major point of Code Behind is that the code for all the Web pages is compiled into a DLL file that allows the web pages to be hosted free from any Inline Server Code.  
  
**Inline Code**  
Inline Code refers to the code that is written inside an ASP.NET Web Page that has an extension of .aspx. It allows the code to be written along with the HTML source code using a <Script> tag. It's major point is that since it's physically in the .aspx file it's deployed with the Web Form page whenever the Web Page is deployed.  
  
For further info click on the link:

* [Code Behind and Inline Code in ASP.NET](http://www.c-sharpcorner.com/UploadFile/cd7c2e/code-behind-and-inline-code-in-Asp-Net/)

**Question 27: What is the ASP.NET page life Cycle?**  
  
**Answer:** When a page is requested by the user from the browser, the request goes through a series of steps and many things happen in the background to produce the output or send the response back to the client. The periods between the request and response of a page is called the "Page Life Cycle".

* **Request:**Start of the life cycle (sent by the user).
* **Response:**End of the life cycle (sent by the server).

There are four stages that occur during the Page Life Cycle before the HTML Response is returned to the client. Later in this article we"ll study all these stages and their sub events.

1. Initialization
2. Loading
3. Rendering
4. Unloading

|  |  |
| --- | --- |
| **Initialization** | During this stage the IsPostback property is set. The page determines whether the request is a Postback (old request) or if this is the first time the page is being processed (new request). Controls on the page are available and each control's UniqueID property is set. Now if the current request is a postback then the data has not been loaded and the value of the controls have not yet been restored from the view state. |
| **Loading** | At this stage if the request is a Postback then it loads the data from the view state. |
| **Rendering** | Before rendering, the View State is saved for the page and its controls. During this phase, the page calls the render method for each control, providing a text writer that writes its output to the OutputStream of the page's Response property. |
| **Unloading** | Unload is called after the page has been fully rendered, sent to the client and is ready to be discarded. At this point also the page properties such as Response and Request are unloaded. |

For further info click on the link:

* [ASP.Net Page Life Cycle](http://www.c-sharpcorner.com/UploadFile/484ad3/Asp-Net-page-life-cycle/)

**Question 28: What is the ASP.NET page life cycle events?**  
  
**Answer:** We have many events in ASP.NET page life cycle let’s see some most important events:

* **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. The UICulture property of the page is also set.
* **Page initialization**   
    
  At this stage, the controls on the page are assigned unique ID by setting the UniqueID property and themes are applied. For a new request postback data is loaded and the control properties are restored to the view-state values.
* **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. The page calls the Render method for each control and the output of rendering is written to the OutputStream class of the Page's Response property.
* **Unload**  
  The rendered page is sent to the client and page properties, such as Response and Request are unloaded and all cleanup done.

**ASP.NET Page Life Cycle Events**  
  
Following are the page life cycle events:

* **PreInit**  
    
  PreInit is the first event in page life cycle. It checks the IsPostBack property and determines whether the page is a postback. It sets the themes and master pages, creates dynamic controls and gets and sets profile property values. This event can be handled by overloading the OnPreInit method or creating a Page\_PreInit handler.
* **Init**  
    
  Init event initializes the control property and the control tree is built. This event can be handled by overloading the OnInit method or creating a Page\_Init handler.
* **InitComplete**  
    
  InitComplete event allows tracking of view state. All the controls turn on view-state tracking.
* **LoadViewState**  
    
  LoadViewState event allows loading view state information into the controls.
* **LoadPostData**   
    
  During this phase, the contents of all the input fields defined with the <form> tag are processed.
* **PreLoad**  
    
  PreLoad occurs before the post back data is loaded in the controls. This event can be handled by overloading the OnPreLoad method or creating a Page\_PreLoad handler.
* **Load**  
    
  The Load event is raised for the page first and then recursively for all child controls. The controls in the control tree are created. This event can be handled by overloading the OnLoad method or creating a Page\_Load handler.
* **LoadComplete**  
    
  The loading process is completed, control event handlers are run and page validation takes place. This event can be handled by overloading the OnLoadComplete method or creating a Page\_LoadComplete handler.
* **PreRender**  
    
  The PreRender event occurs just before the output is rendered. By handling this event, pages and controls can perform any updates before the output is rendered.
* **PreRenderComplete**  
    
  as the PreRender event is recursively fired for all child controls, this event ensures the completion of the pre-rendering phase.
* **SaveStateComplete**  
    
  State of control on the page is saved. Personalization, control state and view state information is saved. The HTML markup is generated. This stage can be handled by overriding the Render method or creating a Page\_Render handler.
* **UnLoad**  
    
  The UnLoad phase is the last phase of the page life cycle. It raises the UnLoad event for all controls recursively and lastly for the page itself. Final cleanup is done and all resources and references, such as database connections, are freed. This event can be handled by modifying the OnUnLoad method or creating a Page\_UnLoad handler.

For further info click on the link:

* [Page Life Cycle in ASP.Net and its Events](http://www.c-sharpcorner.com/Blogs/13203/page-life-cycle-in-Asp-Net-and-its-events.aspx)

**Question 29: Describe login Controls in ASP?**  
  
**Answer:** The Login control provides the user interface to log a user into a web site. The Login control uses the Membership service to authenticate the user in your membership system. The default Membership service from your configuration file will be used automatically, however you can also set the Membership provider that you would like used as a property on the control.  
  
**The Login Control consists of:**

* **Username Label and Textbox:** Collects the string used to identify the user in the membership system.
* **Password Label and Textbox:**Collects the password for the specified user. The textbox text is always obscured.
* **LoginButton:** The button to submit the users request for authentication.
* **RememberMe:**Configurable to display a checkbox giving the user the option to store a persistent cookie on the user's machine.
* **Title and Instruction:**Text to orient and guide the user through the process.
* **Links:**Configurable links to help, password recovery and user registration information.
* **Validators:** Required field Validators for the username and password textboxes.

**For Example:**

1. <asp:Login ID="Login1" runat="server" BackColor="#FFE0C0" BorderColor="Red" ></asp:Login>

For further info click on the link:

* [How to use Login Control in Visual Studio 2008](http://www.c-sharpcorner.com/uploadfile/anjudidi/how-to-use-login-control-in-visual-studio-2008/)

**Question 30: How to use repeater control in ASP.NET?**  
  
**Answer:** A Repeater is a Data-bound control. Data-bound controls are container controls. It creates a link between the Data Source and the presentation UI to display the data. The repeater control is used to display a repeated list of items.  
  
The main use of Repeater Control is for displaying a repeated list of items bound to the control. A Repeater Control is faster and lightweight for displaying data compared to a GridView or DataGrid. With the Repeater control we can display data in a custom format. The main drawback of a Repeater Control is that it doesn't support paging and sorting.  
  
The Repeater Control has the following types of template fields:

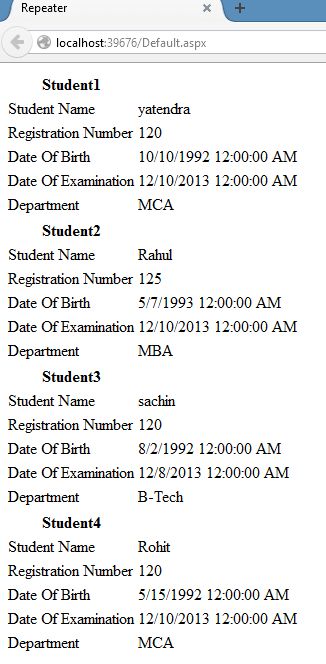
* Item Template
* AlternatingItem Template
* Header Template
* Footer Template
* Separator Template

Write connection code and select command in code bihaind file like:

1. **protected** **void** Page\_Load(**object** sender, EventArgs e)
2. {
3. SqlConnection con = **new** SqlConnection("Data Source=MCNDESKTOP34;Initial Catalog=yatendra;Persist Security Info=True;User ID=sa;
4. Password = Password$2 ");
5. SqlDataAdapter sda = **new** SqlDataAdapter("select \* from Student\_Details1", con); DataTable dt = **new** DataTable(); sda.Fill(dt); Repeater1.DataSource = dt; Repeater1.DataBind();
6. }

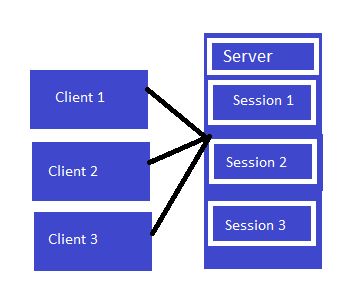
Now use Repeater control object in .aspx file like:

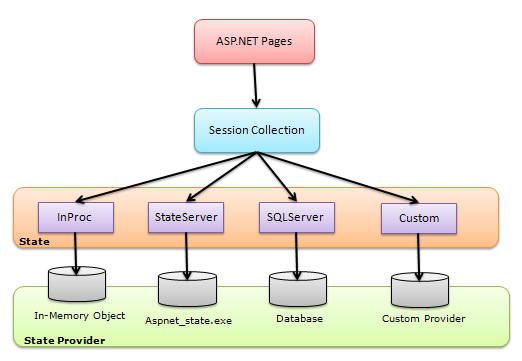
1. <asp:Repeater ID="Repeater1" runat="server">
2. <ItemTemplate>
3. <div>
4. <table>
5. <tr>
6. <th>Student
7. <%#Eval("S\_ID")%>
8. </th>
9. </tr>
10. <tr>
11. <td>Student Name</td>
12. <td>
13. <%#Eval("Student\_Name")%>
14. </td>
15. </tr>
16. <tr>
17. <td>Registration Number</td>
18. <td>
19. <%#Eval("Register\_No")%>
20. </td>
21. </tr>
22. <tr>
23. <td>Date Of Birth</td>
24. <td>
25. <%#Eval("D\_O\_B")%>
26. </td>
27. </tr>
28. <tr>
29. <td>Date Of Examination</td>
30. <td>
31. <%#Eval("D\_O\_E")%>
32. </td>
33. </tr>
34. <tr>
35. <td>Department</td>
36. <td>
37. <%#Eval("Department")%>
38. </td>
39. </tr>
40. </table>
41. </div>
42. </ItemTemplate>
43. </asp:Repeater>

When you run this page so output will look like as:  
  
   
  
For further info click on the link:

* [ASP.Net Repeater Control Using C#](http://www.c-sharpcorner.com/UploadFile/ca2535/Asp-Net-repeater-control-using-C-Sharp/)

**Question 31: What are different methods of session maintenance in ASP.NET?**  
  
**Answer:**Session is a State Management Technique. A Session can store the value on the Server. It can support any type of object to be stored along with our own custom objects. A session is one of the best techniques for State Management because it stores the data as client-based, in other words the data is stored for every user separately and the data is secured also because it is on the server.   
  
We can set the session on one of the following 2 types of configuration files:

1. **Machine Configuration file:** Machine Configuration is applied for all application.
2. **Application Configuration file:**It's applied for only application by application basis.  
     
   

**Session Mode**  
In ASP.NET there are 4 types of Session Mode.  
  
**Off:**We can disable the session mode for the entire application using the off mode.  
  
  
  
According to performance and durability the difference between InProc,State Server and SQL Server is:

|  |  |
| --- | --- |
| **Session mode** | **Performance Durability** |
| InProc | More(1 processor and 1 server) less. |
| State Server | Medium(n processor and 1 server) Medium |
| SQL Server | Less More |

For further info click on the link:

* [ntroduction To ASP.NET Sessions](http://www.c-sharpcorner.com/UploadFile/225740/introduction-of-session-in-Asp-Net/)

**Question 32: What is the Difference between session and caching?**  
**Answer:**The first main difference between session and caching is: a session is per-user based but caching is not per-user based, So what does that mean? Session data is stored at the user level but caching data is stored at the application level and shared by all the users. It means that it is simply session data that will be different for the various users for all the various users, session memory will be allocated differently on the server but for the caching only one memory will be allocated on the server and if one user modifies the data of the cache for all, the user data will be modified.  
  
**For further info click on the link:**

* [Difference Between Session and Caching](http://www.c-sharpcorner.com/UploadFile/75a48f/difference-between-session-and-caching/)

**Question 33: What is the difference between HttpContext.Current.Items and HttpContext.Current.Session in ASP.NET?**  
  
**Answer:**Session state is one of the popular state management techniques in ASP.NET environment. We developer people play with session storage every now and then. It’s pretty simple to manage session if you understand the basic concept. Here is the syntax to do that.

1. Session[“KEY”] =”Value”;

Or

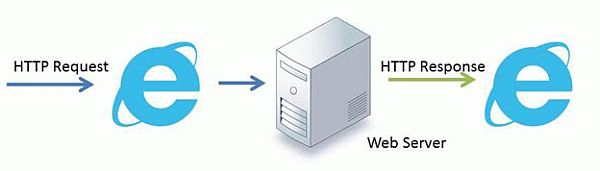
1. Session[index] = ”Value”;
2. Let’ s a have an example: **using** System;
3. **using** System.Collections.Generic;
4. **using** System.Linq;
5. **using** System.Web;
6. **using** System.Web.UI;
7. **using** System.Web.UI.WebControls;
8. **namespace** WebApp
9. {
10. **public** partial **class** WebForm1: System.Web.UI.Page
11. {
12. **protected** **void** Page\_Load(**object** sender, EventArgs e)
13. {
14. **if**(!IsPostBack)
15. {
16. HttpContext.Current.Items["Value"] = "Sourav Kayal in ITEM";
17. HttpContext.Current.Session["Value"] = "Sourav Kayal in SESSION";
18. Response.Write((**string**)(HttpContext.Current.Items["Value"]) + "<br>");
19. Response.Write((**string**)(HttpContext.Current.Session["Value"]));
20. }
21. }
22. **protected** **void** Button1\_Click(**object** sender, EventArgs e)
23. {
24. Response.Write((**string**)(HttpContext.Current.Items["Value"]) + "<br>");
25. Response.Write((**string**)(HttpContext.Current.Session["Value"]));
26. }
27. }
28. }

  
  
For further info click on the link:

* [Difference Between HttpContext.Current.Items and HttpContext.Current.Session in ASP.Net](http://www.c-sharpcorner.com/UploadFile/dacca2/difference-between-httpcontext-current-items-and-httpcontext/)

**Question 34: What is the difference between Server.Transfer and Response.redirect?**  
**Answer:** Both Response.Redirect and Server.Transfer methods are used to transfer a user from one web page to another web page. Both methods are used for the same purpose but still there are some differences as follows.  
  
The Response.Redirect method redirects a request to a new URL and specifies the new URL while the Server.Transfer method for the current request, terminates execution of the current page and starts execution of a new page using the specified URL path of the page.  
  
Both Response.Redirect and Server.Transfer has same syntax like:

1. Response.Redirect("UserDetail.aspx");
2. Server.Transfer("UserDetail.aspx");

Before touching on more points I want to explain some HTTP status codes, these are important for the understanding of the basic differences between these two. The HTTP status codes are the codes that the Web server uses to communicate with the Web browser or user agent.   
  
   
  
For further info click on the link:

* [Difference Between Response.Redirect() and Server.Transfer() Methods in ASP.Net](http://www.c-sharpcorner.com/UploadFile/3d39b4/difference-between-response-redirect-and-server-transfer/)

**Question 35: What is page directives in ASP.NET?**  
**Answer:** Basically Page Directives are commands. These commands are used by the compiler when the page is compiled.  
  
**How to use the directives in an ASP.NET pag**e  
  
It is not difficult to add a directive to an ASP.NET page. It is simple to add directives to an ASP.NET page. You can write directives in the following format:  
  
*<%@[Directive][Attributes]%>*  
  
See the directive format, it starts with "<%@" and ends with "%>". The best way is to put the directive at the top of your page. But you can put a directive anywhere in a page. One more thing, you can put more than one attribute in a single directive.  
  
Here is the full list of directives:

* @Page
* @Master
* @Control
* @Import
* @Implements
* @Register
* @Assembly
* @MasterType
* @Output Cache
* @PreviousPageType
* @Reference

For further info click on the link:

* [ASP.Net Page Directives](http://www.c-sharpcorner.com/UploadFile/azadc/Asp-Net-page-directives/)

**Question 36: What is HTTP Handler?  
  
Answer:**Every request into an ASP.NET application is handled by a specialized component known as an HTTP handler. The HTTP handler is the most important ingredient while handling ASP.NET requests.  
  
**Examples:**ASP.NET uses different HTTP handlers to serve different file types. For example, the handler for web Page creates the page and control objects, runs your code, and renders the final HTML.  
  
ASP.NET default handlers:

1. Page Handler (.aspx) - Handles Web pages.
2. User Control Handler (.ascx) - Handles Web user control pages.
3. Web Service Handler (.asmx) - Handles Web service pages.
4. Trace Handler (trace.axd) - Handles trace functionality.

Why we need to create our own HTTP Handler: Sometime we need to avoid ASP.NET full page processing model, which saves lot of overheads, as ASP.NET web form model has to go through many steps such as creating web page objects, persisting view state etc. What we are interested into is to develop some low level interface that provides access to objects like Request and Response but doesn't use the full control based web form model discussed above.  
  
**Examples:**

1. Dynamic image creator - Use the System.Drawing classes to draw and size your own images.
2. RSS - Create a handler that responds with RSS-formatted XML. This would allow you to add RSS feed capabilities to your sites.
3. Render a custom image,
4. Perform an ad hoc database query,
5. Return some binary data.

All HTTP handlers are defined in the <httpHandlers> section of a configuration file which is nested in the <system.web> element.

1. <httpHandlers>
2. <add verb="\*" path="trace.axd" validate="true" type="System.Web.Handlers.TraceHandler" />
3. <add verb="\*" path="\*.config" validate="true" type="System.Web.HttpForbiddenHandler" />
4. <add verb="\*" path="\*.cs" validate="true" type="System.Web.HttpForbiddenHandler" />
5. <add verb="\*" path="\*.aspx" validate="true" type="System.Web.UI.PageHandlerFactory" /> </httpHandlers>

For further info click on the link:

* [Create your first HTTP Handler in ASP.NET 3.5](http://www.c-sharpcorner.com/uploadfile/37db1d/create-your-first-http-handler-in-Asp-Net-3-5/)

**Question 37: What are Differences between ASP.NET HttpHandler and HttpModule?**  
**Answer:** The user requests for a resource on web server. The web server examines the file name extension of the requested file, and determines which ISAPI extension should handle the request. Then the request is passed to the appropriate ISAPI extension. For example when an .aspx page is requested it is passed to ASP.NET page handler. Then Application domain is created and after that different ASP.NET objects like Httpcontext, HttpRequest, HttpResponse are created. Then instance of HttpApplication is created and also instance of any configured modules. One can register different events of HttpApplication class like BeginRequest, AuthenticateRequest, AuthorizeRequest, ProcessRequest etc.  
  
**HTTP Handler**  
HTTP Handler is the process which runs in response to a HTTP request. So whenever user requests a file it is processed by the handler based on the extension. So, custom http handlers are created when you need to special handling based on the file name extension. Let's consider an example to create RSS for a site. So, create a handler that generates RSS-formatted XML. Now bind the .rss extension to the custom handler.  
  
**HTTP Modules**  
HTTP Modules are plugged into the life cycle of a request. So when a request is processed it is passed through all the modules in the pipeline of the request. So generally http modules are used for:

* **Security:**For authenticating a request before the request is handled.
* **Statistics and Logging:** Since modules are called for every request they can be used for gathering statistics and for logging information.
* **Custom header:** Since response can be modified, one can add custom header information to the response.

For further info click on the link:

* [Difference between ASP.NET HttpHandler and HttpModule](http://www.c-sharpcorner.com/Blogs/7116/difference-between-Asp-Net-httphandler-and-httpmodule.aspx)

**Question 38: Explain the AdRotator Control?**  
Answer: AdRotator control are used to create a dynamic ads. The AdRotator Control presents ad images each time a user enters or refreshes a webpage. When the ads are clicked, it will navigate to a new Web location. The AdRotator control is used to display a sequence of ad images.The AdRotator control to work we need an Advertisement file (XML file) and some sample images.  
  
Adding the AdRotator web server control to your web application. first, select the AdRotator and drag and drop the control to your web form. Map the XML file which contains the details about each and every ad.  
  
The advertisement file is an XML file. The following are some of the elements of this XML file.

1. **<imageUrl>:**Optional. The path to the image file.
2. **<NavigateUrl>:**Optional. The URL to link to if the user clicks the ad.
3. **<AlternateText>:** Optional. An alternate text for the image.
4. **<Impressions>:**Optional. The display rates in percent of the hits.

XML code that has the details about the ads. The file Ads.xml looks like the code below:

1. <Advertisements>
2. <Ad>
3. <ImageUrl>adimages/2.jpg</ImageUrl>
4. <NavigateUrl>http://cat2.com</NavigateUrl>
5. <AlternateText>Cat 2</AlternateText>
6. <Impressions>30</Impressions>
7. </Ad>
8. <Ad>
9. <ImageUrl>adimages/3.jpg</ImageUrl>
10. <NavigateUrl>http://cat3.com</NavigateUrl>
11. <AlternateText>Cat 3</AlternateText>
12. <Impressions>20</Impressions>
13. </Ad>
14. <Ad>
15. <ImageUrl>adimages/4.jpg</ImageUrl>
16. <NavigateUrl>http://cat4.com</NavigateUrl>
17. <AlternateText>Cat 4</AlternateText>
18. <Impressions>10</Impressions>
19. </Ad>
20. </Advertisements>

For further info click on the link:

* [AdRotator control in ASP.NET](http://www.c-sharpcorner.com/Blogs/7249/adrotator-control-in-Asp-Net.aspx)

**Question 39: What is cross-page posting in ASP.NET?**  
**Answer:**ASP.NET 1.1 provides for web forms posting back only to themselves. In many situations, the solution requires posting to a different web page. The traditional workaround alternatives were to use Response.Redirect and/or Server.Transfer to move to a different page and simulate cross page post-back behavior.  
  
ASP.NET 2.0 provides a feature known as Cross Page PostBack for a web form to post-back to a different web form (other than itself)  
  
**How to post to a different page**  
  
To set a web form to post back to a different web form, in the source web form, set the PostBackURL property of a control that implements IButtonControl (eg. Button, ImageButton, LinkButton) to the target web form. When the user clicks on this button control, the web form is cross-posted to the target web form. No other settings or code is required in the source web form.  
  
**Access source page info within the posted page: FindControl Method**  
  
The target web form resulting from the cross-page postback provides a non-null PreviousPage property. This property represents the source page and provides reference to the source web form and its controls.  
  
The controls on the source page can be accessed via the FindControl method on the object returned by the PreviousPage property of the target page.

1. **protected** **void** Page\_Load(**object** sender, EventArgs e)
2. {
3. ...
4. TextBox txtStartDate = (TextBox) PreviousPage.FindControl("txtStartDate ");
5. ...
6. }

At this point the target page does not have any knowledge of the source page. The PreviousPage property is of the type Page. For accessing controls using FindControl, the developer has to presume a certain structure in the source web form. This approach using FindControl has a few limitations. FindControl is dependent on the developer to provide the ids of the controls to access. The code will stop working if the control id is changed in the source web form. The FindControl method can retrieve controls only within the current container. If you need to access a control within another control, you need to first get a reference to the parent control.  
  
**Access source page info within the posted page: @PreviousPageType Directive**  
  
There is another more direct option to get access to the source page controls if the source page is pre-determined. The @PreviousPageType directive can be used in the target page to strongly type the source page. The directive specifies the source page using either the VirtualPath attribute or the TypeName attribute. The PreviousPage property then returns a strongly typed reference to the source page. It allows access to the public properties of the source page.  
  
**SourcePage.aspx:**

1. <form runat="server"> ...
2. <asp:textbox runat="server" id="txtFirstName" />
3. <asp:textbox runat="server" id="txtLastName" />
4. <asp:button runat="server" id="btnViewReport" Text="View Report" PostbackURL="~/targetpage.aspx" /> ... **public** **string** FirstName { **get** { **return** txtFirstName.Text; } } ...

**TargetPage.aspx**

1. <%@ PreviousPageType VirtualPath="sourcepage.aspx" %>
2. **string** strFirstName;
3. strFirstName = PreviousPage.FirstName //Strongly Typed PreviousPage allows direct access to the public properties of the source page

For further info click on the link:

* [Cross Page Postback in ASP.Net 2.0](http://www.c-sharpcorner.com/UploadFile/DipalChoksi/xpgpostbk_asp2_dc08102006235543PM/xpgpostbk_asp2_dc.aspx)

**Question 40: Explain GridView control in ASP.NET?**  
**Answer:**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.
* Programmatic access to the GridView object model to dynamically set properties, handle events, and so on.
* Multiple key fields.
* Multiple data fields for the hyperlink columns.
* Customizable appearance through themes and styles.

**Creating a GridView**

1. <asp:GridView ID="gridService" runat="server">
2. </asp:GridView>

For further info click on the link:

* [GridView Control in ASP.Net](http://www.c-sharpcorner.com/UploadFile/7eb164/gridview-control-in-Asp-Net/)

**Question 41: What is the difference between ASP.NET Web API and WCF?**  
**Answer:** The ASP. NET Web API is a framework that uses the HTTP services and makes it easy to provide the response to the client request. The response depends on the request of the clients. The Web API builds the HTTP services, and handles the request using the HTTP protocols. The request may be GET, POST, DELETE, PUT. We can also say that the ASP. NET Web API:

* Is an HTTP service.
* Is designed for reaching the broad range of clients.
* Uses the HTTP application.

We use the ASP. NET Web API for creating the REST ful (Representational State Transfer) services.  
  
The following are some important points of the ASP. NET Web API:

* The ASP. NET Web API supports the MVC application features that are controller, media formatters, routing etcetera.
* It is a platform for creating the REST services.
* It is a framework for creating the HTTP services.
* Responses can be formatted by the APIs MediaTypeFormatter into the Java Script Object Notation (JSON) and Extencible Markup Language (XML) formats.

For further info click on the link:

* <http://www.c-sharpcorner.com/UploadFile/2b481f/difference-between-Asp-Net-web-api-and-wc>

**Question 42: What is the PostBack property in ASP.NET?**  
Answer: If we create a web Page, which consists of one or more Web Controls that are configured to use AutoPostBack (Every Web controls will have their own AutoPostBack property), the ASP.NET adds a special JavaScipt function to the rendered HTML Page. This function is named \_doPostBack() . When Called, it triggers a PostBack, sending data back to the web Server.  
  
ASP.NET also adds two additional hidden input fields that are used to pass information back to the server. This information consists of ID of the Control that raised the event and any additional information if needed. These fields will empty initially as shown below,

1. <input type="hidden" name="\_\_EVENTTARGET" id="\_\_EVENTTARGET" value="" />
2. <input type="hidden" name="\_\_EVENTARGUMENT" id="\_\_EVENTARGUMENT" value="" />

The following actions will be taken place when a user changes a control that has the AutoPostBack property set to true:

1. On the client side, the JavaScript \_doPostBack function is invoked, and the page is resubmitted to the server.
2. ASP.NET re-creates the Page object using the .aspx file.
3. ASP.NET retrieves state information from the hidden view state field and updates the controls accordingly.
4. The Page.Load event is fired.
5. The appropriate change event is fired for the control. (If more than one control has been changed, the order of change events is undetermined.)
6. The Page.PreRender event fires, and the page is rendered (transformed from a set of objects to an HTML page).
7. Finally, the Page.Unload event is fired.
8. The new page is sent to the client.

For further info click on the link:

* [What is PostBack in ASP.NET](http://www.c-sharpcorner.com/uploadfile/2f73dd/what-is-postback-in-Asp-Net/)

**Question 43: Explain Cookie-less Session in ASP.NET.**  
**Answer:** By default a session uses a cookie in the background. To enable a cookie-less session, we need to change some configuration in the Web.Config file. Follow these steps:

1. Open Web.Config file.
2. Add a <sessionState> tag under <system.web> tag.
3. Add an attribute "cookieless" in the <sessionState> tag and set its value to "AutoDetect" like below:
   1. <sessionState cookieless="AutoDetect" regenerateExpiredSessionId="true"/>

The possible values for "**cookieless**" attribute are:

* **AutoDetect:** Session uses background cookie if cookies are enabled. If cookies are disabled, then the URL is used to store session information.
* **UseCookie:**Session always use background cookie. This is default.
* **UseDeviceProfile:** Session uses background cookie if browser supports cookies else URL is used.
* **UseUri:**Session always use URL.

"**regenerateExpiredSessionId**" is used to ensure that if a cookieless url is expired a new new url is created with a new session. And if the same cookieless url is being used by multiple users an the same time, they all get a new regenerated session url.  
  
For further info click on the link:

* [Using Cookie-less Session in ASP.NET](http://www.c-sharpcorner.com/UploadFile/deepak.sharma00/using-cookie-less-session-in-Asp-Net/)

**Question 44: What is Themes in ASP.NET?**  
**Answer:**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 you may have for individual controls on your page.   
  
How to apply themes  
  
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.
   1. <%@ Page Language="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 you 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:
   1. <pages theme="Theme1">
   2. ....
   3. ....
   4. </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.
   1. 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. You can have as many themes as you want and you can switch between them by setting a single attribute in the web.config file or an individual aspx page. Also you can switch between themes programmatically.
3. Setting the themes programmatically, you are offering your users a quick and easy way to change the page to their likings.
4. Themes allow you to improve the usability of your site by giving users with vision problems the option to select a high contrast theme with a large font size.

For further info click on the link:

* [Themes in ASP.NET](http://www.c-sharpcorner.com/UploadFile/8dd3df/themes-in-Asp-Net/)

**Question 45: What are the Navigations technique in ASP.NET?**  
Answer: Navigation can cause data loss if it not properly handled. We do have many techniques to transfer data from one page to another but every technique has its own importance and benefits.  
  
We will discuss the following techniques in this article.

* Response.Redirect
* Server.Transfer
* Server.Exceute
* Cross page posting

For further info click on the link:

* [Navigation Techniques in ASP.Net](http://www.c-sharpcorner.com/UploadFile/de41d6/navigation-techniques-in-Asp-Net/)

**Question 46: What is WebParts in ASP.NET?**  
**Answer:** ASP.NET 2.0 incorporates the concept of **WEB PARTS** in itself and we can code and explore that as easily as we have done with the other controls in the previous sessions.  
  
We can compose web parts pages from "web parts", which can be web controls, user controls.  
  
**Component of Web Parts:**  
  
The web parts consist of different components like:

* Web Part Manager
* Web Part Zone
* CatLog Part
* CatLog Zone
* Connections Zone
* Editor Part
* Editor Zone

**Web Part Zone**

* Web Part Zone can contain one or more Web Part controls.
* This provides the layout for the Controls it contains. A single ASPX page can contain one or more Web Part Zones.
* A Web Part Control can be any of the controls in the toolbox or even the customized user controls.

For further info click on the link:

* [Understanding WEBPARTS in ASP.NET 2.0: Part I](http://www.c-sharpcorner.com/UploadFile/praveenalwar/WebParts09242007061241AM/WebParts.aspx)

**Question 47: What are master pages?**  
**Answer:**Some points about Master Pages:

1. The extension of MasterPage is '.master'.
2. MasterPage cannot be directly accessed from the client because it just acts as a template for the other Content Pages.
3. In a MasterPage we can have content either inside ContentPlaceHolder or outside it. Only content inside the ContentPlaceHolder can be customized in the Content Page.
4. We can have multiple masters in one web application.
5. A MasterPage can have another MasterPage as Master to it.
6. The content page content can be placed only inside the content tag.
7. Controls of MasterPage can be programmed in the MasterPage and content page but a content page control will never be programmed in MasterPage.
8. A master page of one web application cannot be used in another web application.
9. The **MasterPageFile** property of a webform can be set dynamically and it should be done either in or before the Page\_PreInit event of the WebForm. **Page.MasterPageFile = "MasterPage.master"**. The dynamically set Master Page must have the ContentPlaceHolder whose content has been customized in the WebForm.
10. The order in which events are raised: Load (Page) a Load (Master) a LoadComplete (Page) i.e. if we want to overwrite something already done in Load event handler of Master then it should be coded in the **LoadComplete** event of the page.
11. Page\_Load is the name of method for event handler for Load event of Master. (it's not **Master\_Load**).

For further info click on the link:

* [Master Page in ASP.NET](http://www.c-sharpcorner.com/UploadFile/4ed419/master-page-in-Asp-Net/)

**Question 48: What is Data Cache in ASP.NET and how to use?**  
**Answer:**Data Cache is used to store frequently used data in the Cache memory. It's much efficient to retrieve data from the data cache instead of database or other sources. We need to use System.Web.Caching namespace. The scope of the data caching is within the application domain unlike "session". Every user is able to access this object.  
  
When client request to the server, server execute the stored procedure or function or select statements on the Sql Server database then it returns the response to the browser. If we run again same process will happen on the web server with sql server.  
  
**How to create data cache?**  
*Cache ["Employee"] = "DataSet Name"*  
  
We can create data caching use Cache Keyword. It's located in the System.Web.Caching namespace. It's just like assigning value to the variable.  
  
**How to remove a Data Cache?**  
We can remove Data Cache manually.

1. //We need to specify the cache name
2. Cache.Remove(String key);

For further info click on the link:

* [Data Cache in ASP.NET](http://www.c-sharpcorner.com/uploadfile/annathurai/data-cache-in-Asp-Net/)

**Question 49: Enterprise Library in ASP.NET?**  
**Answer:** Enterprise Library: It is a collection of application blocks and core infrastructure. Enterprise library is the reusable software component designed for assisting the software developers.  
  
We use the Enterprise Library when we want to build application blocks intended for the use of developers who create complex enterprise level application.  
  
**Enterprise Library Application Blocks**

1. **Security Application Block**  
     
   Security Application Block provide developers to incorporate security functionality in the application. This application can use various blocks such as authenticating and authorizing users against the database.
2. **Exception Handling Application Block**  
     
   This block provides the developers to create consistency for processing the error that occur throughout the layers of Enterprise Application.
3. **Cryptography Application Block**  
     
   Cryptography application blocks provides developers to add encryption and hashing functionality in the applications.
4. **Caching Application Block**  
     
   Caching Application Block allows developers to incorporate local cache in the applications.

For further info click on the link:

* [Enterprise Library In ASP.NET](http://www.c-sharpcorner.com/UploadFile/ca2535/enterprise-library-in-Asp-Net/)

**Question 50: How can we improve the Performance of an ASP.NET Web Page?**  
Answer: This is the most common question from ASP.NET forum to any interview. In this post I’m going to point out some of the important points that may help to improve the performance.  
  
Here I used the word “improve performance” in the sense to decrease the loading time of the page. There are various reasons behind. Some of them we look into from the “backend side” (Database side) and rest of them we need to take care in “front-end” ((UI) side.  
  
For illustrative purpose, you have an ASP.NET Web site, one of the aspx page take much time to load. Throughout this article, we are going to see how to decrease the loading time.  
  
**Back End (DB)**

1. Try to check the Query performance that is how much time the query will take to execute and pull the records from DB. Then use SQL Server Profiler and Execution plan for that query so that you can come to a conclusion in which part it took much time.
2. Check in every table (who are all part of the query) Index is created properly.
3. If your query involves a complex stored procedure, which in turn use lot of joins, then you should focus on every table. In some cases, sub-query perform better than the joins.
4. If your web page involves paging concepts, try to move the paging concepts to SQL Server. I meant that based on the page count the SP will return the records, instead of bringing the records as a whole.

**1. What is ASP.Net?**

It is a framework developed by Microsoft on which we can develop new generation web sites using web forms(aspx), MVC, HTML, Javascript, CSS etc. Its successor of Microsoft Active Server Pages(ASP). Currently there is ASP.NET 4.0, which is used to develop web sites. There are various page extensions provided by Microsoft that are being used for web site development. Eg: aspx, asmx, ascx, ashx, cs, vb, html, XML etc.

**2. What’s the use of Response.Output.Write()?** We can write formatted output  using Response.Output.Write().

**3. In which event of page cycle is the ViewState available?**   After the Init() and before the Page\_Load().

**4. What is the difference between Server.Transfer and Response.Redirect?**

In Server.Transfer page processing transfers from one page to the other page without making a round-trip back to the client’s browser.  This provides a faster response with a little less overhead on the server.  The clients url history list or current url Server does not update in case of Server.Transfer.

Response.Redirect is used to redirect the user’s browser to another page or site.  It performs trip back to the client where the client’s browser is redirected to the new page.  The user’s browser history list is updated to reflect the new address.

**5. From which base class all Web Forms are inherited?**

Page class.

**6. What are the different validators in ASP.NET?**

1. Required field Validator
2. Range  Validator
3. Compare Validator
4. Custom Validator
5. Regular expression Validator
6. Summary Validator

**7. Which validator control you use if you need to make sure the values in two different controls matched?**

Compare Validator control.

**8. What is ViewState?**

ViewState is used to retain the state of server-side objects between page post backs.

**9. Where the viewstate is stored after the page postback?**

ViewState is stored in a hidden field on the page at client side.  ViewState is transported to the client and back to the server, and is not stored on the server or any other external source.

**10. How long the items in ViewState exists?**

They exist for the life of the current page.

**11. What are the different Session state management options available in ASP.NET?**

1. In-Process
2. Out-of-Process.

**In-Process** stores the session in memory on the web server.

**Out-of-Process** Session state management stores data in an external server.  The external server may be either a SQL Server or a State Server.  All objects stored in session are required to be serializable for Out-of-Process state management.

**12. How you can add an event handler?**

Using the Attributes property of server side control.

**13. What is caching?**

Caching is a technique used to increase performance by keeping frequently accessed data or files in memory. The request for a cached file/data will be accessed from cache instead of actual location of that file.

**14. What are the different types of caching?**  
ASP.NET has 3 kinds of caching :

1. Output Caching,
2. Fragment Caching,
3. Data Caching.

**15. Which type if caching will be used if we want to cache the portion of a page instead of whole page?**

**Fragment Caching:** It caches the portion of the page generated by the request. For that, we can create user controls with the below code:



|  |  |
| --- | --- |
| 1 | <%@ OutputCache Duration="120" VaryByParam="CategoryID;SelectedID"%> |

**16. List the events in page life cycle.**

1) Page\_PreInit  
2) Page\_Init  
3) Page\_InitComplete  
4) Page\_PreLoad  
5) Page\_Load  
6) Page\_LoadComplete  
7) Page\_PreRender  
8)Render

**17. Can we have a web application running without web.Config file?**

**Yes**

**18. Is it possible to create web application with both webforms and mvc?**

Yes. We have to include below mvc assembly references in the web forms application to create hybrid application.



|  |  |
| --- | --- |
| 1  2  3  4  5 | System.Web.Mvc    System.Web.Razor    System.ComponentModel.DataAnnotations |

**19. Can we add code files of different languages in App\_Code folder?**

 No. The code files must be in same language to be kept in App\_code folder.

**20. What is Protected Configuration?**

It is a feature used to secure connection string information.

**21. Write code to send e-mail from an ASP.NET application?**



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | MailMessage mailMess = new MailMessage ();  mailMess.From = "abc@gmail.com";  mailMess.To = "xyz@gmail.com";  mailMess.Subject = "Test email";  mailMess.Body = "Hi This is a test mail.";  SmtpMail.SmtpServer = "localhost";  SmtpMail.Send (mailMess); |

MailMessage and SmtpMail are classes defined System.Web.Mail namespace.

**22. How can we prevent browser from caching an ASPX page?**

We can SetNoStore on HttpCachePolicy object exposed by the Response object’s Cache property:



|  |  |
| --- | --- |
| 1  2 | Response.Cache.SetNoStore ();  Response.Write (DateTime.Now.ToLongTimeString ()); |

**23. What is the good practice to implement validations in aspx page?**  
Client-side validation is the best way to validate data of a web page. It reduces the network traffic and saves server resources.

**24. What are the event handlers that we can have in Global.asax file?**  
**Application Events:** Application\_Start , Application\_End, Application\_AcquireRequestState, Application\_AuthenticateRequest, Application\_AuthorizeRequest, Application\_BeginRequest, Application\_Disposed,  Application\_EndRequest, Application\_Error, Application\_PostRequestHandlerExecute, Application\_PreRequestHandlerExecute,  
Application\_PreSendRequestContent, Application\_PreSendRequestHeaders, Application\_ReleaseRequestState, Application\_ResolveRequestCache, Application\_UpdateRequestCache

**Session Events:** Session\_Start,Session\_End

**25. Which protocol is used to call a Web service?**  
HTTP Protocol

**26. Can we have multiple web config files for an asp.net application?**  
Yes.

**27. What is the difference between web config and machine config?**  
Web config file is specific to a web application where as machine config is specific to a machine or server. There can be multiple web config files into an application where as we can have only one machine config file on a server.

**28.  Explain role based security ?**

Role Based Security used to implement security based on roles assigned to user groups in the organization.

Then we can allow or deny users based on their role in the organization. Windows defines several built-in groups, including Administrators, Users, and Guests.



|  |  |
| --- | --- |
| 1  2  3  4 | <AUTHORIZATION>< authorization >  < allow roles="Domain\_Name\Administrators" / >   < !-- Allow Administrators in domain. -- >  < deny users="\*"  / >                            < !-- Deny anyone else. -- >  < /authorization > |

**29. What is Cross Page Posting?**

When we click submit button on a web page, the page post the data to the same page. The technique in which we post the data to different pages is called Cross Page posting. This can be achieved by setting POSTBACKURL property of  the button that causes the postback. Findcontrol method of PreviousPage can be used to get the posted values on the page to which the page has been posted.

**30. How can we apply Themes to an asp.net application?**

We can specify the theme in web.config file. Below is the code example to apply theme:



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | <configuration>    <system.web>    <pages theme="Windows7" />    </system.web>    </configuration> |

**31. What is RedirectPermanent in ASP.Net?**

RedirectPermanent Performs a permanent redirection from the requested URL to the specified URL. Once the redirection is done, it also returns 301 Moved Permanently responses.

**32. What is MVC?**

MVC is a framework used to create web applications. The web application base builds on  Model-View-Controller pattern which separates the application logic from UI, and the input and events from the user will be controlled by the Controller.

**33. Explain the working of passport authentication.**

First of all it checks passport authentication cookie. If the cookie is not available then the application redirects the user to Passport Sign on page. Passport service authenticates the user details on sign on page and if valid then stores the authenticated cookie on client machine and then redirect the user to requested page

**34. What are the advantages of Passport authentication?**

All the websites can be accessed using single login credentials. So no need to remember login credentials for each web site.

Users can maintain his/ her information in a single location.

**35. What are the asp.net Security Controls?**

* <asp:Login>: Provides a standard login capability that allows the users to enter their credentials
* <asp:LoginName>: Allows you to display the name of the logged-in user
* <asp:LoginStatus>: Displays whether the user is authenticated or not
* <asp:LoginView>: Provides various login views depending on the selected template
* <asp:PasswordRecovery>:  email the users their lost password

**36. How do you register JavaScript for webcontrols ?**  
We can register javascript for controls using <CONTROL -name>Attribtues.Add(scriptname,scripttext) method.

**37. In which event are the controls fully loaded?**

Page load event.

**38. what is boxing and unboxing?**

Boxing is assigning a value type to reference type variable.

Unboxing is reverse of boxing ie. Assigning reference type variable to value type variable.

**39. Differentiate strong typing and weak typing**

In strong typing, the data types of variable are checked at compile time. On the other hand, in case of weak typing the variable data types are checked at runtime. In case of strong typing, there is no chance of compilation error. Scripts use weak typing and hence issues arises at runtime.

**40. How we can force all the validation controls to run?**

The *Page.Validate()* method is used to force all the validation controls to run and to perform validation.

**41. List all templates of the Repeater control.**

* *ItemTemplate*
* *AlternatingltemTemplate*
* *SeparatorTemplate*
* *HeaderTemplate*
* *FooterTemplate*

**42. List the major built-in objects in ASP.NET?**

* *Application*
* *Request*
* *Response*
* *Server*
* *Session*
* *Context*
* *Trace*

**43. What is the appSettings Section in the *web.config* file?**

The *appSettings* block in web config file sets the user-defined values for the whole application.

For example, in the following code snippet, the specified *ConnectionString* section is used throughout the project for database connection:



|  |  |
| --- | --- |
| 1  2  3  4 | <em><configuration>  <appSettings>  <add key="ConnectionString" value="server=local; pwd=password; database=default" />  </appSettings></em> |

**44. Which data type does the *RangeValidator* control support?**

The data types supported by the *RangeValidator* control are Integer, Double, String, Currency, and Date.

**45. What is the difference between an *HtmlInputCheckBox* control and an *HtmlInputRadioButton* control?**

In *HtmlInputCheckBoxcontrol,*multiple item selection is possible whereas in*HtmlInputRadioButton*controls, we can select only single item from the group of items.

**46. Which namespaces are necessary to create a localized application?**

*System.Globalization*

*System.Resources*

**47. What are the different types of cookies in ASP.NET?**

**Session Cookie** – Resides on the client machine for a single session until the user does not log out.

**Persistent Cookie** – Resides on a user’s machine for a period specified for its expiry, such as 10 days, one month, and never.

**48. What is the file extension of web service?**

Web services have file extension .asmx..

**49. What are the components of ADO.NET?**

The components of ADO.Net are Dataset, Data Reader, Data Adaptor, Command, connection.

**50. What is the difference between ExecuteScalar and ExecuteNonQuery?**

ExecuteScalar returns output value where as ExecuteNonQuery does not return any value but the number of rows affected by the query. ExecuteScalar used for fetching a single value and ExecuteNonQuery used to execute Insert and Update statements.