TOP

500 .NET INTERVIEW QUESTIONS 2022

OOPS C# SQL ASP.NET

.NET CORE WEB API MVC

HAPPY RAWAT

ABOUT THE BOOK

This book contains 500 .NET Interview Questions.

This book is based on the research of .NET interview questions asked in top IT and Tech companies like Microsoft, TCS, Accenture, Infosys, Wipro, HCL, IBM, Tech Mahindra, CTS, HP.

ABOUT THE AUTHOR

Happy Rawat have more than 13 years of experience in .NET technologies. He helps candidates in clearing technical interview in tech companies.

This book is divided into two parts:

Part I – Top 200 interview questions

Part II – 300 more questions

Topic	Number of Questions	
	Part I	**Part II**
OOPS/ C#	51	47
.NET FRAMEWORK	16	28
SQL	27	29
ASP.NET MVC	15	53
ASP.NET WEBFORMS	7	35
ADO.NET	4	10
ENTITY FRAMEWORK	7	0
JAVASCRIPT	0	27
DESIGN PATTERNS	16	9
WEB API / WEB SERVICE / WCF	19	38
.NET CORE	38	24
TOTAL	200	300

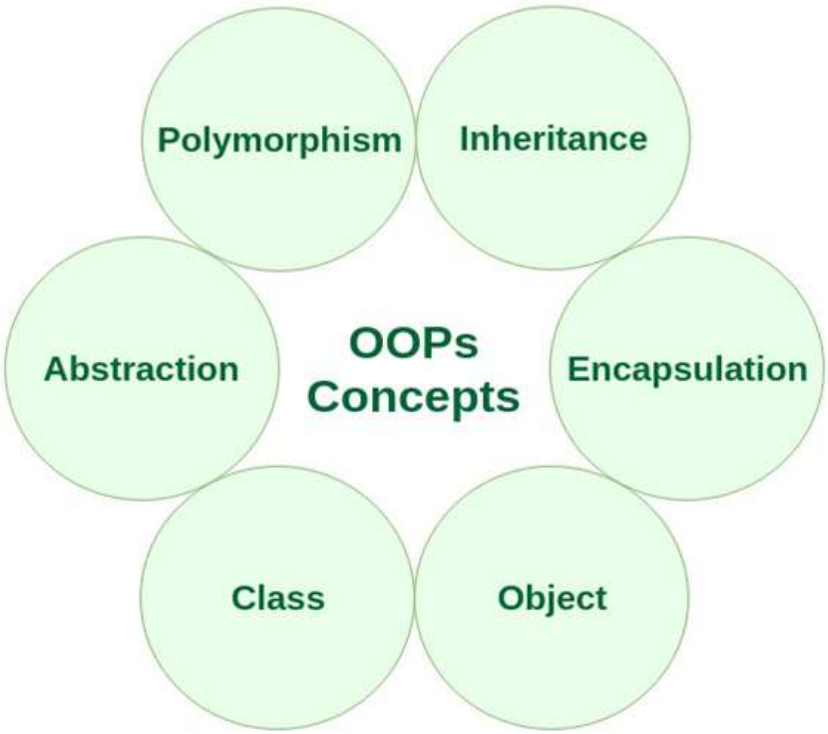
PART I

TOP 200 INTERVIEW QUESTIONS

Chapter 1 – OOPS/ C#

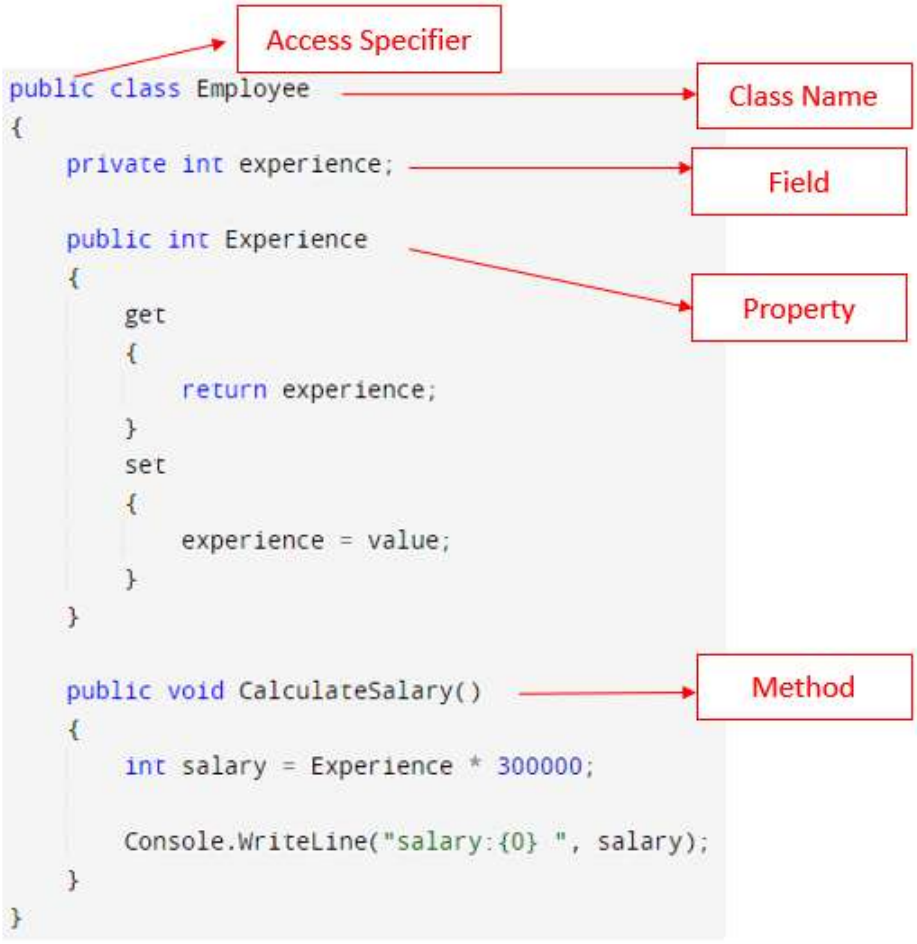
Q1. WHAT ARE THE MAIN CONCEPTS OF OOPS? WHAT ARE CLASSES AND OBJECTS?

Here are the main concepts of OOPS:

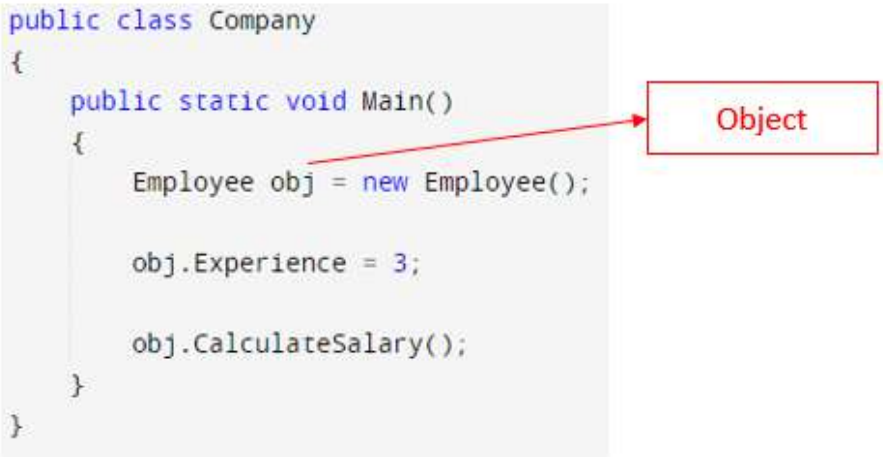


- **CLASS** - A class is a **LOGICAL UNIT or BLUEPRINT** that contains fields, methods and properties.

  Simple class and it members example:



- **OBJECT** - An object is an **INSTANCE** of a class.



Q2. WHAT IS INHERITANCE? WHY INHERITANCE IS IMPORTANT?

Inheritance is creating a **PARENT-CHILD** relationship between two classes where child automatically get the properties and methods of the parent.

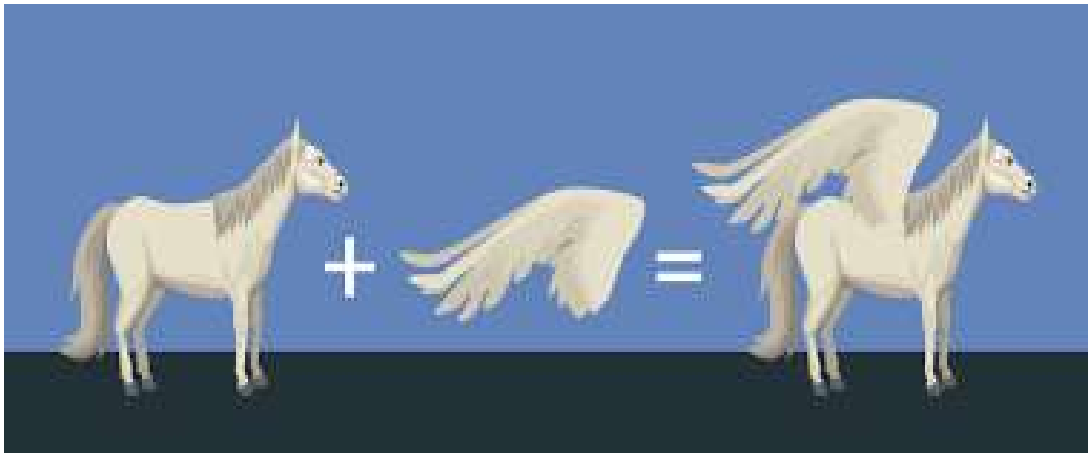
For example, in below code Vehicle is a Parent class and Car is a child class. Now when we will create the object of the Car(derived class) then it will automatically get this Tyre() method, even though this Tyre method is not present in the Car class but it got it from its parent class Vehicle.

```
class Vehicle                          // Base/ Parent/ Super class
{
    public string brand = "Ford";  // Vehicle field

    public void Tyre()             // Vehicle method
    {
        Console.WriteLine("Rubber tyre");
    }
}

class Car : Vehicle                    // Derived/ Child/ Sub class
{
    public string modelName = "Mustang";  // Car field
}

class Program
{
  static void Main(string[] args)
  {
     Car myCar = new Car();

     myCar.Tyre();                     // Tyre() method is not present in Car class, but it will get it automatically from it's parent

     Console.WriteLine(myCar.brand + " " + myCar.modelName);
  }
}
```

Inheritance Use:

Inheritance is good for: **REUSABILITY** and **ABSTRACTION** of code

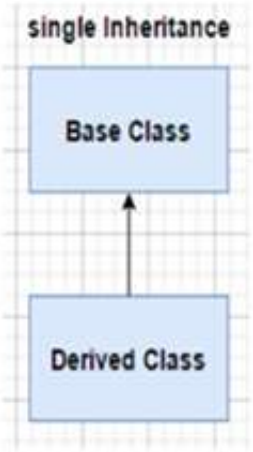
For example, suppose there is a Horse class, now you want to add wings to this horse. Then you will create a wings class as base class for this Horse class. In future if you want to create a Cow class with wings then you can use the same wings base class for Cow class also. That is reusability.



Q3. WHAT ARE THE DIFFERENT TYPES OF INHERITANCE?

Below are the types of inheritance we have in C#:

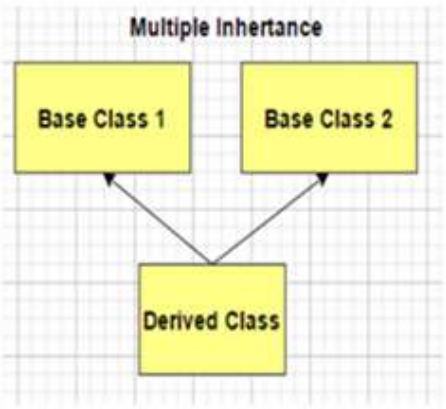
- **Single Inheritance -** One Base class with one Derived class.




```
class BaseClass
{
    public void Animal()
    {
        Console.WriteLine("Animal")
    }
}
class ChildClass : BaseClass
{
    public void Dog()
    {
        Console.WriteLine("Dog");
    }
}
```

- **Multiple Inheritance -** In this case Multiple Base classes can be there for single derived class.

Remember in C#, multiple inheritance can only be achieved with the help of interfaces. Which means only one base or abstract class is allowed, and rest must be interfaces.

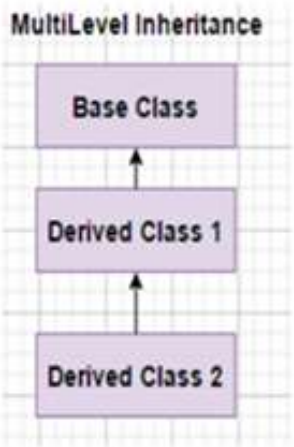



```
1 reference
public class BaseClass{
        0 references
        public void Aninimal(Parameters)
        {
            System.Console.WriteLine("Aminmal");
        }
}

1 reference
public interface InterfaceAnimal{
    1 reference
    public void Fly();
}

0 references
public class ChildClass : BaseClass, InterfaceAnimal{

    1 reference
    public void Fly(){
        System.Console.WriteLine("Test in InterFace ");
    }

    0 references
    public void ChildMehod(){
       System.Console.WriteLine("child Method Write Heare");
    }
}
```

- **Multilevel inheritance** – In this there is a GrandParent class then there is a Parent class which is derived from GrandParent class and then there is Child class which is derived from Parent class.

  Grandparent class -> Parent class -> Child class

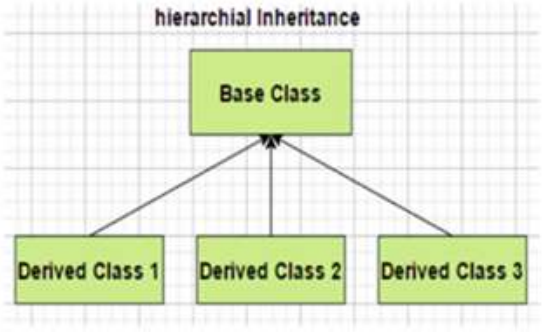
  Here the Child class will get the properties of both Parent class and GrandParent class automatically.




```
class BaseClass
{
    public void Animal()
    {
        Console.WriteLine("Animal");
    }
}
class ChildClass : BaseClass
{
    public void Dog()
    {
        Console.WriteLine("Dog");
    }
}
class SecondChildClass : ChildClass
{
    public void Labrador()
```

- **Hierarchal inheritance -** In this One, child class is derived more from than one base class. This is the most used type of inheritance.

Just to remind you Base/ Parent/ Super classes are same and Child/ Derived/ Subclasses are same.

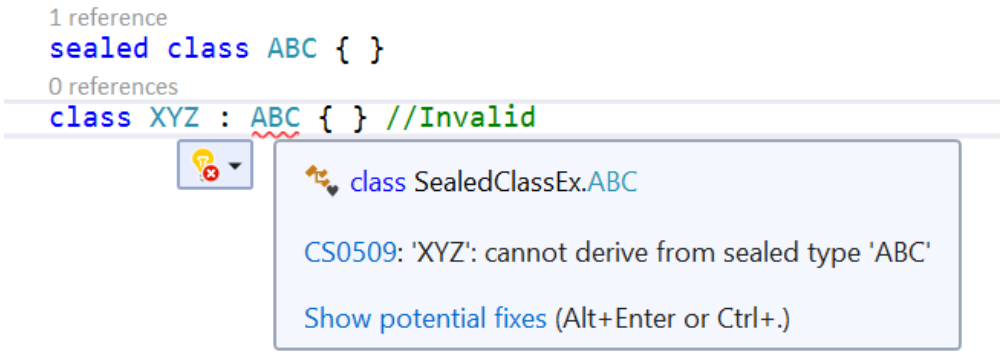



```
class BaseClass
{
    public void Animal()
    {
        Console.WriteLine("Animal");
    }
}
class ChildClass : BaseClass
{
    public void Dog()
    {
        Console.WriteLine("Dog");
    }
}
class SecondChildClass : BaseClass
{
    public void Cat()
    {
        Console.WriteLine("Dog");
    }
}
```

Q4. HOW TO PREVENT A CLASS FROM BEING INHERITED?

By using **SEALED** keyword in class.

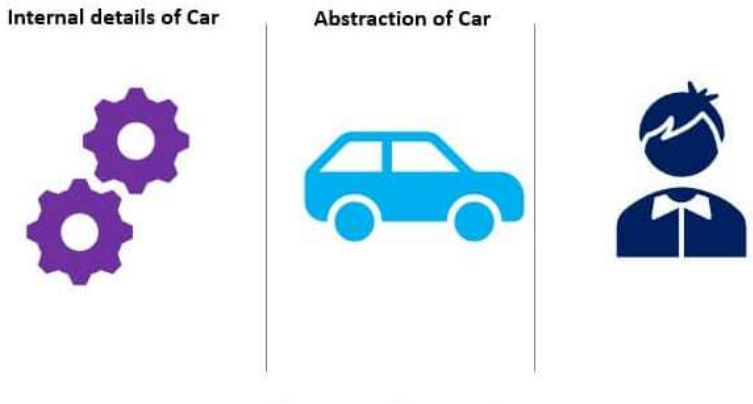
See the screenshot below, you marked the class ABC as sealed. If you will try to derive this class "ABC" in a child class "XYZ" then it will give a compiler error because ABC is marked as Sealed.



Q5. WHAT IS ABSTRACTION?

Abstraction means showing only required things and hide the **BACKGROUND** details.

For example, a car driver only knows about few things like staring, gear and break, but things like working of engine is hidden from him, but still he can drive the car.



For example, see the client code inside main method, in order to run a car, Driver will insert key and call the function RunEngine(). Now the car will start.

But internally inside RunEngine() method, there might be several other methods which are called like InjectFuel(), SparkEngine() and many others.
And they are silently doing their job without making themselves visible to driver.
So that is hiding the details and that is abstraction.

```
public class Program
{
    public static void Main()
    {
        Car carObject = new Car();
        carObject.RunEngine();
    }
}
```

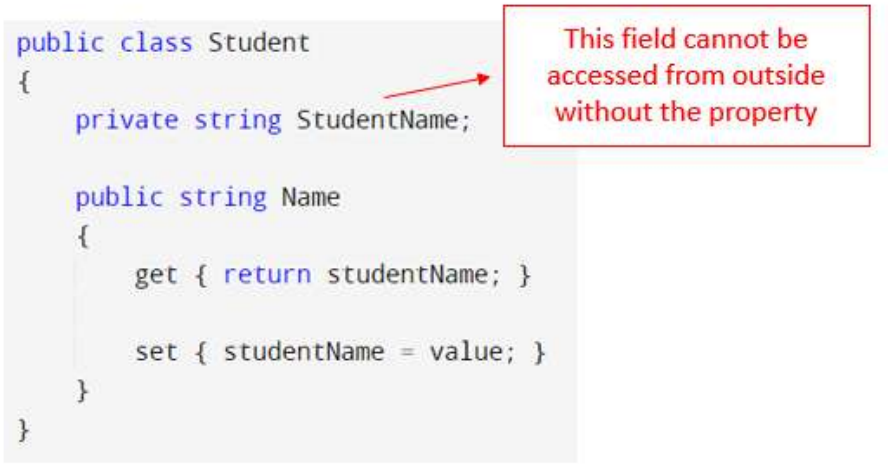
```
public car
{
    public void RunEngine()
    {
        InjectFuel();

        SparkEngine();

        //...Many more things

        Console.WriteLine("Engine Started");
    }
}

//Output: Engine Started
```

Q6. WHAT IS ENCAPSULATION?

Encapsulation means **WRAPPING** of data and methods/properties into a single unit.

For example, in below code the field (field represents data only) can only be accessed by any outside class with the help of property only. So that means they both are wrapped as per encapsulation.

If you will directly access the field in your program without property, then it is violation of encapsulation which is not good for data security.




```
public class Student
{
    private string StudentName;

    public string Name
    {
        get { return studentName; }

        set { studentName = value; }
    }
}
```

```
class DemoEncapsulation {
    // Main Method
    static public void Main()
    {
        // creating object
        Student obj = new Student();
        obj.Name = "Ankita";
        // Displaying values of the variables
        Console.WriteLine("Name: " + obj.Name);
    }
}
```

Q7. WHAT IS POLYMORPHISM AND WHAT ARE ITS TYPES?

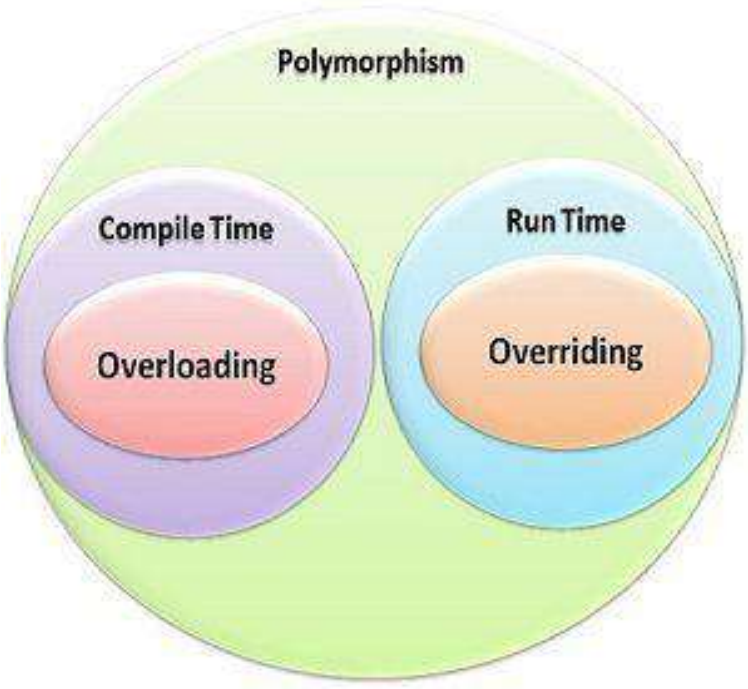
Polymorphism is the ability of a variable, object, or function to take on **MULTIPLE FORMS**.

For example, "Add" method can add integers or also can add strings so it can take multiple forms.

There are two types of polymorphism:

- Compile time
- Run time.

Overloading is a type of Compile time polymorphism and **Overriding** is a type of run time polymorphism.



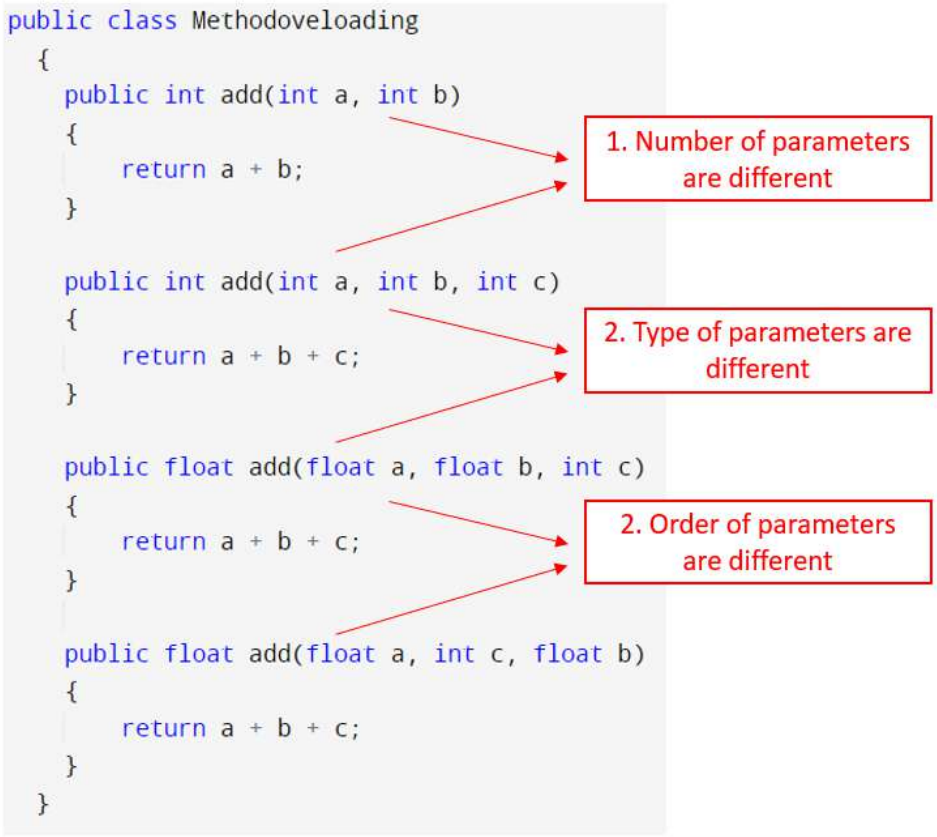
Q8. WHAT IS METHOD OVERLOADING? IN HOW MANY WAYS A METHOD CAN BE OVERLOADED?

Method overloading is a kind of compile time* polymorphism, in which we can create multiple methods of the **same name in the same class**, and all methods work in different ways.

Why it is called compile time polymorphism because .Net compiler(CLR) knows at compile time that these two different methods with the same name are different.

**Compile time is the period when the programming code (such as C#, Java, C, Python) is converted to the machine code (i.e. binary code). Runtime is the period when a program is running and generally occurs after compile time.*

Method overloading can be done in 3 ways:

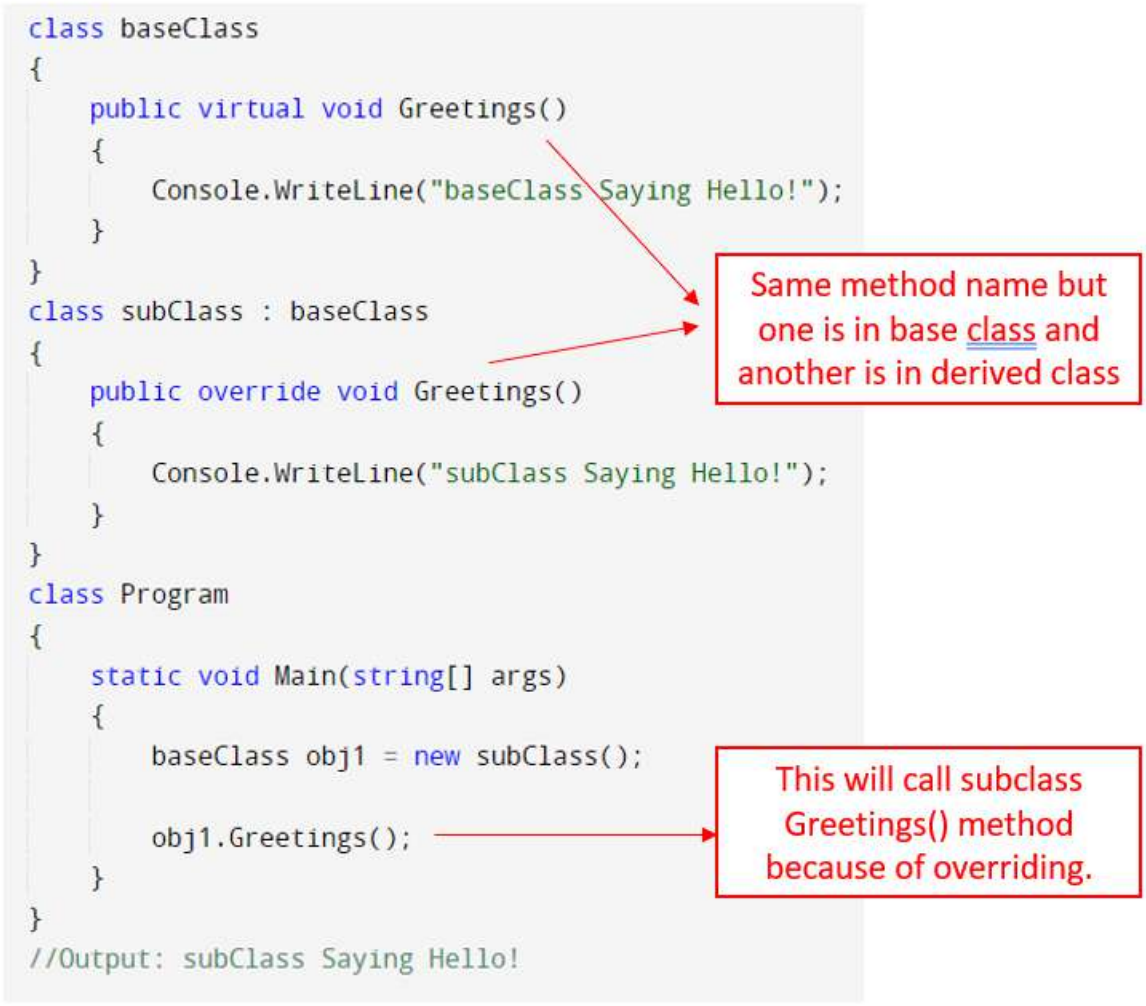


Q9. WHAT IS THE DIFFERENCE BETWEEN OVERLOADING AND OVERRIDING?

Method overriding is creating a method in the **DERIVED** class with the **SAME NAME and SIGNATURE** as a method in the base class.

Overriding uses **VIRTUAL** keyword for base class method and **OVERRIDE** keyword for derived class method.

Below is the code example :



So the difference between Overloading and Overriding are:

1. In Overloading method name is same but signature (number of parameters/ type etc) is different.

But in overriding both method name and signature are same.

2. Overloading is a type of **COMPILE TIME** polymorphism.

Overriding is a type of **RUN TIME** polymorphism.

3. Method overloading doesn't need inheritance. It is possible in same class.

Method overriding **NEEDS INHERITANCE (Base class/ Derived class)**. It is not possible in same class.

Q10. WHAT IS THE DIFFERENCE BETWEEN METHOD OVERRIDING AND METHOD HIDING?

In **Method Hiding**, you can hide the implementation of the methods of a base class from the derived class using the **new keyword**.

In other words, in overriding you are overriding the method but here in method hiding, you are completely **redefining** the method.

Overriding example

```
public class BaseClass
{
    public virtual void Print()
    {
        Console.WriteLine("Base Class Print Method");
    }
}
public class DerivedClass : BaseClass
{
    public override void Print()
    {
        Console.WriteLine("Child Class Print Method");
    }
}
public class Program
{
    public static void Main()
    {
        BaseClass B = new DerivedClass();
        B.Print();
    }
}
```

Output: Child Class Print Method

Method Hiding example

```
public class BaseClass
{
    public virtual void Print()
    {
        Console.WriteLine("Base Class Print Method");
    }
}
public class DerivedClass : BaseClass
{
    public new void Print()
    {
        Console.WriteLine("Child Class Print Method");
    }
}
public class Program
{
    public static void Main()
    {
        BaseClass B = new DerivedClass();
        B.Print();
    }
}
```

Output: Base Class Print Method

Q11. WHAT ARE THE ADVANTAGES AND LIMITATIONS OF OOPS?

Advantages of OOPS:

- **Reuse** of code through inheritance:
  One base class can be used in multiple derived class.

- **Flexibility** through polymorphism:
  Same name methods like ADD can do different operations like adding integers, adding strings etc.

- The principle of data hiding/Encapsulation helps the programmer to build **secure** programs.
  For example, if you are calling str.Contains("one") method then might be in the .NET framework, there are multiple other methods inside the contains() method but that are not visible to you and that's why they are secured.

- OOP systems can be **easily upgraded** from small to large systems.

- Modularity for **easier troubleshooting:**
  When you will divide your application in different folders/ interfaces/ base classes/ modules then it is easy to track things as it is organized.

Limitations of OOPS:

- OOPS is not suitable for **small** applications.

The reason is simple, if you are creating a small application then why to create base classes and derive classes. Why to care about polymorphism and other stuff. Implementation of OOPS needs planning and if the project is very small then why to spend so much time on planning.

Q12. WHAT IS THE DIFFERENCE BETWEEN AN ABSTRACT CLASS AND AN INTERFACE?

1. Abstract class contains both **DECLARATION & DEFINITION** of methods.

```
// abstract class 'Car'
public abstract class Car {
    // abstract method
    public abstract void Engine();

    public void Dashboard()
    {
        Console.WriteLine("Dashboard");
    }
}
```

Method Declared

Method Defined

Interface contains **ONLY DECLARATION** of methods.

```
interface Car {
    // methods having only
    //declaration not definition
    void Engine();

    void Dashboard();
}
```

Only method Declaration is allowed

But C# 8.00 version and after interface can contains property default Method Implementation

```
interface IFile{
 void ReadFile();
 void WriteFile(string text);
 void DisplayName() { Console.WriteLine("IFile");}
}
interface IPoint{ // Property signatures:
 int X { get; set; }
int Y { get; set; }
double Distance { get; }
}
```

https://www.tutorialsteacher.com/csharp/csharp-interface

https://learn.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/interface

2. Abstract class keyword: **ABSTRACT** Interface keyword: **INTERFACE**

3. Abstract class can contain methods, fields, constants, constructor.

Interface can contain **undefined methods only** nothing else.

4. Abstract class does not Support **MULTIPLE INHERITANCE** Interface supports multiple inheritance

---

Q13. WHEN TO USE INTERFACE AND WHEN ABSTRACT CLASS?

Abstract class is a good choice when you are sure some methods are concrete/defined and must be implemented in the **SAME WAY** in all derived classes.

An interface is a good choice when you know a method has to be there, but it can be implemented **DIFFERENTLY** by independent derived classes.

Normally we prefer Interface because it gives use the flexibility to modify the behavior at later stage.

Q14. WHY TO EVEN CREATE INTERFACES?

There are two reasons to create interfaces:

1. One is to achieve Multiple Inheritance in C# as abstract classes do not support multiple inheritance.

2. Second reason is to maintain a consistency between different derived classes.

Q15. DO INTERFACE CAN HAVE A CONSTRUCTOR?

NO.

Interface can only be derived from. Their object creation is not possible, so they don't have any constructor.

Q16. CAN YOU CREATE AN INSTANCE OF AN ABSTRACT CLASS OR AN INTERFACE?

NO.

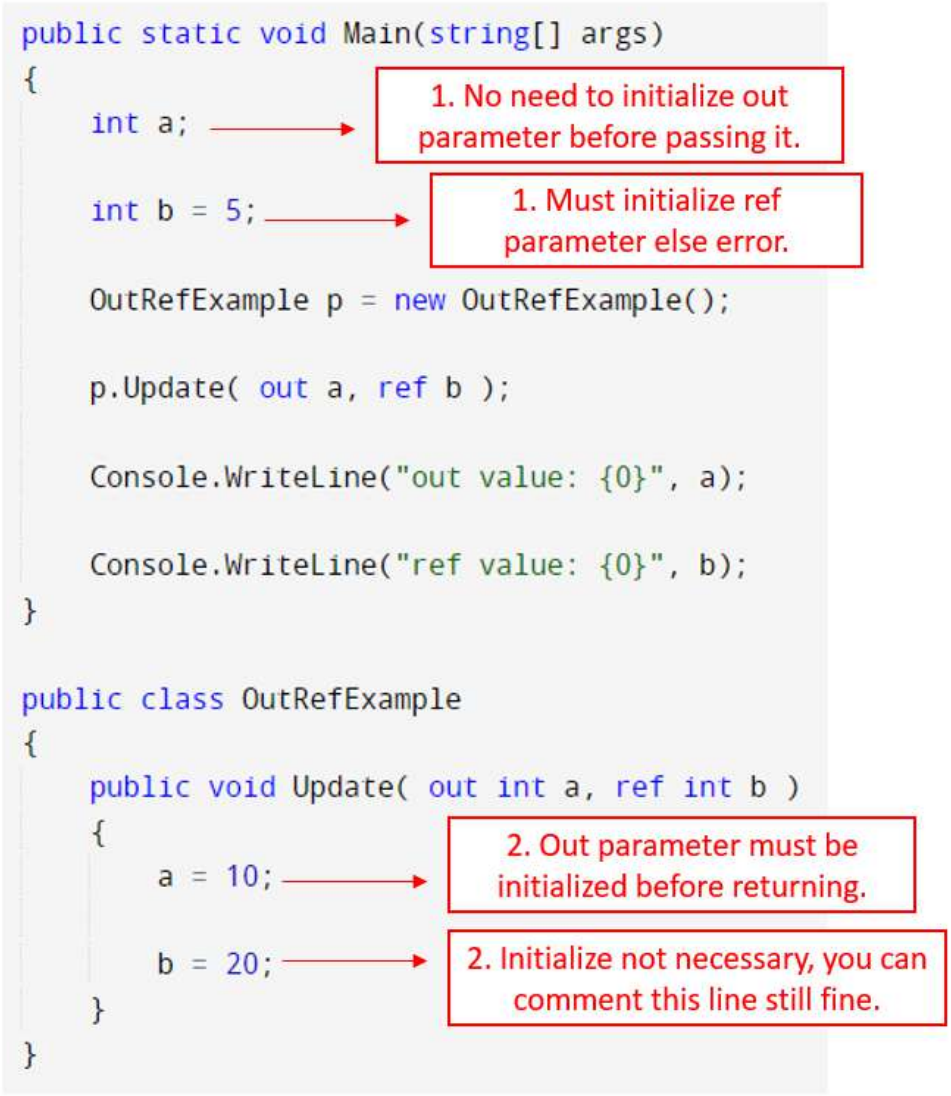
Abstract class and Interface can only be used for inheritance not for object creation.

Q17. WHAT IS THE DIFFERENCE BETWEEN "OUT" AND "REF" PARAMETERS?

By using **ref** and **out** keywords we can pass parameters by reference.

When you want to return more than one values from a method then you can use **out** and **ref** parameters.

Below in the code sample you can see the differences between out("a") and ref("b") parameter



3. Third difference is, one will use ref parameter when you want to **modify** a value and out parameter when you want to get a **new value** from the called method.

Q18. WHAT IS THE PURPOSE OF "PARAMS" KEYWORD?

Params is used as a parameter which can take the **VARIABLE NUMBER OF ARGUMENTS**.

It is useful when programmer don't have any prior knowledge about the number of parameters to be passed.

In below code one can pass any number of parameters in the Add() method and the params keyword will handle it.

```
class InterviewHappy
{
    static void Main(string[] args)
    {
        // Calling function by passing 5
        // arguments as follows
        int y = Add(12,13,10,15,56);

        // Displaying result
        Console.WriteLine(y);
    }
    // function containing params parameters
    public static int Add(params int[] ListNumbers)
    {
        int total = 0;
        // foreach loop
        foreach(int i in ListNumbers)
        {
            total += i;
        }
        return total;
    }
}
```

You can pass any number of parameters here.

Q19. WHAT ARE EXTENSION METHODS IN C#? WHEN TO USE THEM?

Extension method concept allows you to add new methods in the existing class **WITHOUT MODIFYING** the source code of the original class.

USE - Use them when you don't have CODE of the class and you want to add a new method in that class.

*For example in below code, we added a new "Right()" method in the .Net framework **String** class. Note that we don't have the code of String class, still we added a new method in it.*
For that we created a static StringExtensions class first and added the method with the help of "this" keyword.

```
public class Program
{
    public static void Main(string[] args)
    {
        string test  = "HellowWorld";

        Console.WriteLine( test.Substring(0, 5) );

        Console.WriteLine(test.Right(5));
    }
}
```

Right() method is not present in String class. But we can add it by using extension method

```
public static class StringExtensions
{
   public static string Right(this string s, int count)
   {
        return s.Substring(s.Length - count, count);
   }
}

//Output: Hello, World
```

Q20. WHAT ARE ACCESS SPECIFIERS? WHAT IS THE DEFAULT ACCESS MODIFIER IN A CLASS?

Access specifiers are keywords to specify the accessibility of a class, method, property and field.

The keywords are – Public, Private, Protected, Internal and ProtectedInternal.

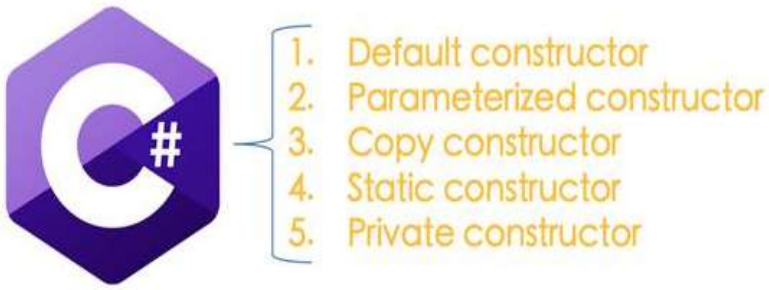
Below is the table which will define what keyword will provide accessibility at what level.

Access Specifier	Inside Same Assembly where member is declared			Other Assembly where containing Assembly is referenced	
	Inside Same Class	Inside Derived Class	Other Code	Inside Derived Class	Other Code
Public	✓	✓	✓	✓	✓
Private	✓	✗	✗	✗	✗
Internal	✓	✓	✓	✗	✗
Protected	✓	✓	✗	✓	✗
ProtectedInternal	✓	✓	✓	✓	✗

Internal is the default access modifier of a class.

Q21. WHAT IS A CONSTRUCTOR AND WHAT ARE ITS TYPES?

Constructor is a special METHOD of the class that is **AUTOMATICALLY** invoked when an instance of the class is created.



- **DEFAULT CONSTRUCTOR -** A constructor without any parameters is a default constructor.

```
class addition
{
    int a, b;
    public addition()   //default contructor
    {
        a = 100;
        b = 175;
    }

    public static void Main()
    {
        //an object is created , constructor is called
        addition obj = new addition();
        Console.WriteLine(obj.a);
        Console.WriteLine(obj.b);
        Console.Read();
    }
}
```

- **PARAMETERIZED CONSTRUCTOR -** A constructor with at least one parameter is a parameterized constructor.

```
class paraconstrctor
{
  public  int a, b;
  // decalaring Paremetrized Constructor with ing x,y parameter
  public paraconstrctor(int x, int y)
    {
        a = x;
        b = y;
    }
}
class MainClass
{
    static void Main()
    {
        paraconstrctor v = new paraconstrctor(100, 175);   // Creating
            object of Parameterized Constructor and ing values
        Console.WriteLine("-----------parameterized constructor example by
            vithal wadje---------------");
        Console.WriteLine("\t");
        Console.WriteLine("value of a=" + v.a );
        Console.WriteLine("value of b=" + v.b);
        Console.Read();
    }
}
```

- **STATIC CONSTRUCTOR -** Static constructor is used to be called before any static member of a class is called.

If you have static methods in your class then static constructor can be used like this.

```
class Test1
{
    //Static constructor
    static Test1()
    {
        Console.WriteLine("Static Constructor Called");
    }

    public static void print()
    {
        Console.WriteLine("Print Method Called");
    }

    public static void Main(string[] args)
    {
        Test1.print();
    }
}

//OUTPUT:
//Static Constructor Called
//Print Method Called
```

- **PRIVATE CONSTRUCTOR -** When a constructor is created with a private specifier, it is not possible for other classes to derive from this class, neither is it possible to create an instance of this class.

```
public class Counter
{
    private Counter()  //private constrctor declaration
    {
    }
    public static int currentview;
    public static int visitedCount()
    {
        return ++ currentview;
    }
}
class viewCountedetails
{
    static void Main()
    {
        // Counter aCounter = new Counter();   // Error
        Console.WriteLine("-------Private constructor ---");
        Console.WriteLine();
        Counter.currentview = 500;
        Counter.visitedCount();
        Console.WriteLine("view count is: {0}", Counter.currentview);
        Console.ReadLine();
    }
}
```

- **COPY CONSTRUCTOR -** The constructor which creates an object by copying variables from another object is called a copy constructor.

Figure: In the below code, you can see we are passing an object inside the constructor and then setting the fields.

```
class employee
{
    private string name;
    private int age;
    public employee(employee emp)   // declaring Copy constructor.
    {
        name = emp.name;
        age = emp.age;
    }
    public employee(string name, int age)  // Instance constructor.
    {
        this.name = name;
        this.age = age;
    }
    public string Details     // Get deatils of employee
    {
        get
        {
            return  " The age of " + name +" is "+ age.ToString();
        }
    }
}
class empdetail
{
    static void Main()
    {
        employee emp1 = new employee("Vithal", 23);  // Create a new employee object.
        employee emp2 = new employee(emp1);          // here is emp1 details is copied to emp2.
        Console.WriteLine(emp2.Details);
        Console.ReadLine();
    }
}
```

Q22. WHEN TO USE PRIVATE CONSTRUCTOR?

Private constructor is a special instance constructor which is used in a class that contains only **static** members.

Below is the code example

```
public class Counter
{
    private Counter()  //private constrctor declaration
    {
    }
    public static int currentview;
    public static int visitedCount()
    {
        return ++ currentview;
    }
}
```

Q23. HOW TO IMPLEMENT EXCEPTION HANDLING IN C#?

Exception handling in Object-Oriented Programming is used to **MANAGE ERRORS.**

- **TRY –** A try block is a block of code inside which any error can occur.
- **CATCH –** When any error occur in TRY block then it is passed to catch block to handle it.
- **FINALLY –** The finally block is used to execute a given set of statements, whether an exception is thrown or not thrown.

- **THROW −** Inside the catch block you can use throw keyword to pass the stack trace to one level up.

```
try
{
   // statements causing exception
}
catch(ExceptionName e1 ) {
   throw ex;
}
catch (ExceptionName e2)
{
    throw;
}
finally
{
    // statements executed in any case
}
```

Q24. CAN WE EXECUTE MULTIPLE CATCH BLOCKS?

NO

We can write multiple catch blocks but **only one** out of them will executed.

For example, in below code in case of any error either e1 will be executed or e2 not both in any case.

Q25. WHAT IS A FINALLY BLOCK AND GIVE AN EXAMPLE WHEN TO USE IT?

Finally block will be executed IRRESPECTIVE of exception.

For example, closing database connection can be kept in "finally" block because whether any error occur or not, we have to make sure the connection must be closed.
So, we will keep connection closing code inside finally block.

```
try
{
   SqlConnection con = new SqlConnection(connectionString);
   //Some logic
   //Error occured
}
catch(ExceptionName ex) {
    //Error handled
}
finally
{
   //Connection closed
   con.Close();
}
```

Q26. CAN WE HAVE ONLY "TRY" BLOCK WITHOUT "CATCH" BLOCK?

YES.

We can have only try block without catch block but we have to have **finally** block then.

Below is the code example. There is no catch block but finally block is there. If we will remove finally then it will give compile time error.

```
public void ExceptionHandling()
{
    try
    {
        SqlConnection con = new SqlConnection(connectionString);
        if(a==b)
        {
            return;
        }
        else
        {
        //Some logic
        }
    }
    finally
    {
        //Connection closed
        con.Close();
    }
}
```

Finally will execute even this function return from here.

Q27. WHAT IS THE DIFFERENCE BETWEEN "THROW EX" AND "THROW"?

Throw keyword preserve the whole **stack trace** and give more information about the error. But throw ex will give limited information about the error.

For example, in the below code the DivideByZero() method will surely give the error and see the information you received using "throw ex" and "throw" keyword. Rest of the code is same in both side.

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            DivideByZero(10);
        }
        catch (Exception ex)
        {
            throw;
        }
    }

    public  static void DivideByZero(int i)
    {
       int j = 0;
       int k = i/j;
       Console.WriteLine(k);
    }
}
```

```
Run-time exception (line 13): Attempted to divide by zero.

Stack Trace:

[System.DivideByZeroException: Attempted to divide by zero.]
   at Program.Main() :line 13
```

```
Run-time exception (line 21): Attempted to divide by zero.

Stack Trace:

[System.DivideByZeroException: Attempted to divide by zero.]
   at Program.DivideByZero(Int32 i) :line 21
   at Program.Main() :line 13
```

```
class Program
{
    static void Main(string[] args)
    {
        try
        {
            DivideByZero(10);
        }
        catch (Exception ex)
        {
            throw ex;
        }
    }

    public  static void DivideByZero(int i)
    {
       int j = 0;
       int k = i/j;
       Console.WriteLine(k);
    }
}
```

See the second one giving more information as comparison to throw ex. So, it's a best practice to use *throw* as it **PRESERVE the whole stack trace.**

Q28. WHAT ARE THE LOOP TYPES IN C#?

There are 4 types of loops present in C#:

1. While Loop

- Initialization happens before the loop starts.
- Condition is set with the while keyword.
- Increment has to be done inside the loop at the end.

```
public class HelloWorld
{
    public static void Main(string[] args)
    {
        int i = 0;          // Initialize
        while (i < 5)       // Condition
        {
          Console.WriteLine(i);
          i++;              // Increment
        }
    }
}
```

//Output: 0 1 2 3 4

2. Do while

The only difference between while and do while is the statement inside do, will **must execute first time** irrespective of while condition whether it is true or not.

See in below code, the output is 10 even if the condition is i<10 is failing first time.

```
public class HelloWorld
{
    public static void Main(string[] args)
    {
        int i = 10;
        do
        {
            Console.WriteLine("i = {0}", i);
            i++;

        } while (i < 10);
    }
}

//output is i = 10
```

This statement executes at least one time irrespective of while condition

3. For Loop

Initialization, condition and Increment all will happen along with the for keyword.

The advantage of it is, the number of lines of code are lesser than while.

```
public class HelloWorld
{
    public static void Main(string[] args)
    {
        for (int i = 0; i < 5; i++)
        {
            Console.WriteLine(i);
        }
    }
}
```

Initialize

Increment

Condition

Output

0 1 2 3 4

4. Foreach Loop

This is used to iterate any collection of array, arraylist, List, Hashtable etc.

This is the most used loop in C#.

```
public static void Main(string[] args)
{
    // creating an array
    int[] a_array = new int[] { 10, 20, 30, 40 };

    // foreach loop begin
    // it will run till the
    // last element of the array
    foreach(int items in a_array)
    {
        Console.WriteLine(items);
    }
}
```

Output

10 20 30 40

Q29. WHAT IS THE DIFFERENCE BETWEEN "CONTINUE" AND "BREAK" STATEMENT?

- Continue statement is used to **skip** the remaining statements inside the loop and transfers the control to the **beginning** of the loop.

```
while(expression)
{
    //code block
    if(condition for continue)
    {
        continue;
    }
    //code block
}
```

- Break statement breaks the loop. It makes the control of the program to **exit** the loop.

```
while(expression)
{
    statement(s);
    if(condition for break)
    {
        break;
    }
}

comes here
statement(s);
```

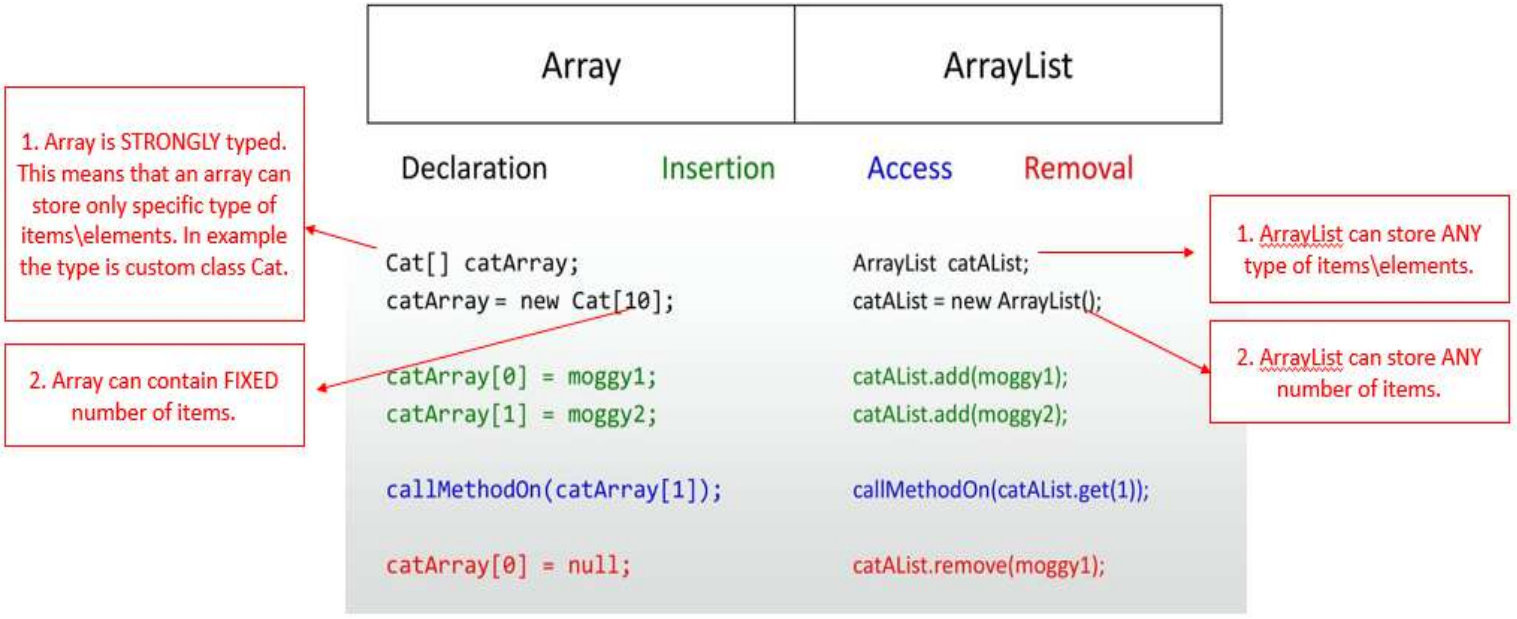
Q30. WHAT IS THE DIFFERENCE BETWEEN ARRAY AND ARRAYLIST?

1. Array is **STRONGLY** typed. This means that an array can store only specific type of items\elements.

ArrayList can store **ANY** type of items\elements.

2. Second difference is, Array can contain **FIXED** number of items.

ArrayList can store **ANY** number of items.



Q31. WHAT IS THE DIFFERENCE BETWEEN ARRAYLIST AND HASHTABLE?

ArrayList - ArrayList can only add items or values in it.

Hashtable - Hashtable add items with keys in it.

```
public static void Main(string[] args)
{
    // Example Arraylist:
    Hashtable hashtbl = new Hashtable();
    hashtbl.Add("Number",1);
    hashtbl.Add("Car", "Ferrari");
}
```

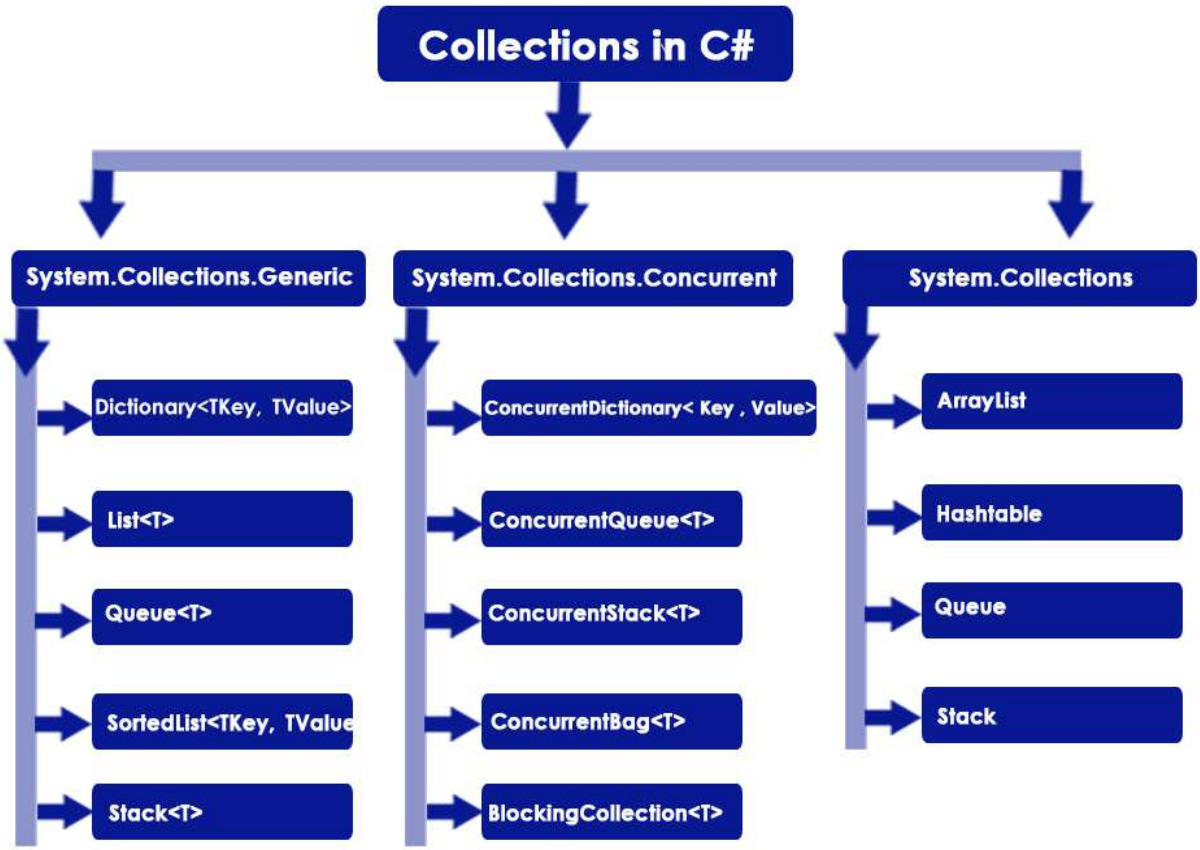
1. In Hashtable we can add Items/Values with the Keys.

Q32. WHAT ARE COLLECTIONS IN C# AND WHAT ARE THEIR TYPES?

C# collection are designed to **store, manage and manipulate** similar DATA more efficiently.

For example ArrayList, Dictionary, List, Hashtable etc.
Data manipulation meaning adding, removing, finding, and inserting data in the collection.

The types of collections are here:



Q33. WHAT IS IENUMERABLE IN C#?

IEnumerable interface is used when we want to **ITERATE** among collection classes using a **FOREACH** loop.

For example, in below code we have a list of Students. Internally this List<Student> collection is using IEnumerable.

```
public class Program
{
    public static void Main()
    {
        var students = new List<Student>() {
                new Student(){ Id = 1, Name="Bill" },
                new Student(){ Id = 2, Name="Steve" }
        };

        foreach(var student in students)
        {
            Console.WriteLine(student.Id + ", " +student.Name);
        }
    }
}
public class Student
{
    public int Id { get; set; }
    public string Name { get; set; }
}
```

If you right click List and see definition then you can see something like this:

```
public class List<T> : System.Collections.Generic.ICollection<T>,
System.Collections.Generic.IEnumerable<T>, System.Collections.Generic.IList<T>,
System.Collections.Generic.IReadOnlyCollection<T>,
System.Collections.Generic.IReadOnlyList<T>, System.Collections.IList
```

If you remove IEnumerable from here then foreach loop on Students will not work.

Q34. WHAT IS THE DIFFERENCE BETWEEN IENUMERABLE AND IENUMERATOR IN C#?

List is using IEnumerable internally and similarly IEnumberable is using IEnumerator internally. So IEnumerator is like a part of IEnumerable.

Suppose you don't want to use .NET Framework List collection and want to create your own custom collection. Then you have to explicitly implement first IEnumerable interface and then IEnumerator interface for iterating in for loop.

For simplicity, just say that IEnumerable is a box that contains IEnumerator functionality inside it.

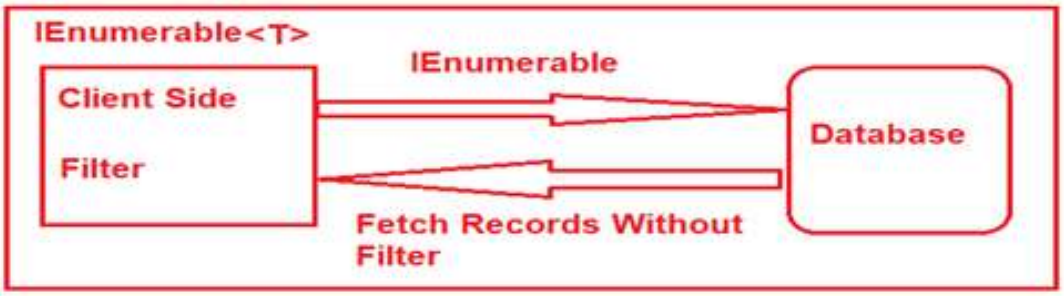
Q35. WHAT IS THE DIFFERENCE BETWEEN IENUMERABLE AND IQUERYABLE IN C#? WHY TO USE IQUERYABLE IN SQL QUERIES?

IQueryable is also like IEnumerable and is used to iterate **SQL** query collection from data.

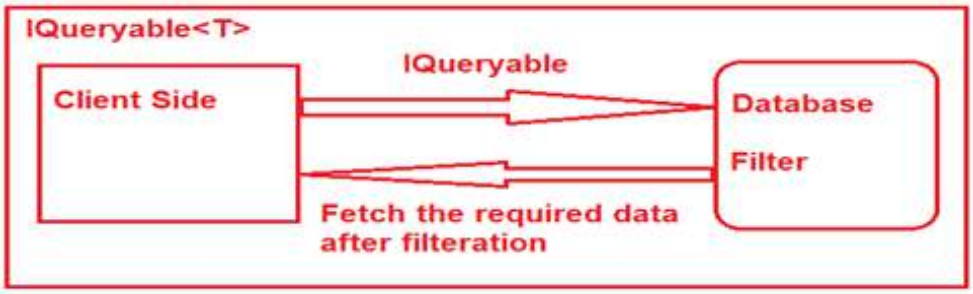
It is under SYSTEM.LINQ namespace.

Why to use IQUeryable when we already have IEnumerable?

IQueryable is better for sql queries because IEnumerable go to the database and fetch the whole data from database without filtering and then FILTER it at server side. So, fetching whole data from the network impact the PERFORMANCE in negative way.



On the other hand IQueryable go to the database, filter the data there only and fetch the required data after filter only which is good for performance.



Q36. WHAT YOU MEAN BY DELEGATE? WHEN TO USE THEM?

A Delegate is a variable that holds the **REFERENCE TO A METHOD**. Or you can say it's a pointer to a function.

A delegate can refer to more than one methods of same return type and parameters.

See here inside the DelegateExample class we have two static methods add and mul. These methods have same return type and same number of parameters. Then we can declare a delegate Calculator for them.

```
delegate int Calculator(int x, int y);//declaring delegate

public class DelegateExample
{
    public static int add(int a, int b)
    {
        return a + b;
    }
    public static int mul(int a, int b)
    {
       return a * b;
    }
    public static void Main(string[] args)
    {
```

Then in the Main method we can instantiate the delegate by passing the method name "add" in the parameter of Delegate.
And then calling the delegate "c1" by passing the other parameters.

```
    public static void Main(string[] args)
    {
        Calculator c1 = new Calculator(add); //Instantiating delegate

        int result = c1(20, 30); //calling method using delegate

        Console.WriteLine(result);
    }
    //Output: 50
}
```

When and why to use delegate?

When we need to pass a method as a parameter.

Q37. WHAT ARE MULTICAST DELEGATES?

A Multicast Delegate in C# is a delegate that holds the references of more than one function.

For example, here is a normal Rectangle class which have two methods GetArea and Getperimeter. The requirement is call GetArea() method first and then Getperimeter() after that. Now in normal scenario if you want to call both these methods then inside the main method you will create the object of rectangle class and then will call both these methods one by one.
If you use normal delegate, then you know you can pass only one method as a parameter and only one will execute.
So, the solution for this is use multicast delegate which will invoke both these methods one by one.

```
namespace MulticastDelegateDemo
{
    public class Rectangle
    {
        public void GetArea(double Width, double Height)
        {
            Console.WriteLine(@"Area is {0}", (Width * Height));
        }
        public void GetPerimeter(double Width, double Height)
        {
            Console.WriteLine(@"Perimeter is {0}", (2 * (Width + Height)))
        }
        static void Main(string[] args)
        {
            Rectangle rect = new Rectangle();

            rect.GetArea(10, 20);

            rect.GetPerimeter(10, 20);
        }
    }
    //output
    //Area is 200
    //Perimeter is 60
}
```

When we invoke the multicast delegate, then all the functions which are referenced by the delegate are going to be invoked one after another like this.

```
namespace MulticastDelegateDemo
{
    public delegate void RectangleDelegate(double Width, double Height);
    public class Rectangle
    {
        public void GetArea(double Width, double Height)
        {
            Console.WriteLine(@"Area is {0}", (Width * Height));
        }
        public void GetPerimeter(double Width, double Height)
        {
            Console.WriteLine(@"Perimeter is {0}", (2 * (Width + Height)));
        }
        static void Main(string[] args)
        {
            Rectangle rect = new Rectangle();

            RectangleDelegate rectDel = new RectangleDelegate(rect.GetArea);

            //Chaining of delegates
            rectDel += rect.GetPerimeter;

            rectDel.Invoke(20, 30); //output: Areas is 600 Perimeter is 100
        }
    }
}
```

Q38. WHAT ARE ANONYMOUS DELEGATES IN C#?

In Anonymous Delegates, You can create a delegate, but there is no need to declare the method associated with it.

For example, see the code below there is no function or method name in here. The line just after the delegate() has the inline logic which is anonymous and that's why it is called anonymous delegate.

```
public delegate void AnonymousDelegate();

public class Program
{
    static void Main()
    {
        AnonymousDelegate delObject = delegate()
        {
            //Inline content of the method;
            Console.WriteLine("Anonymous Delegate method");
        };

        delObject();
    }
}
//Output: Anonymous Delegate method
```

Q39. WHAT ARE THE DIFFERENCES BETWEEN EVENTS AND DELEGATES?

The event is a notification mechanism that depends on delegates

An event is dependent on a delegate and cannot be created without delegates.

Event is like a wrapper over the delegate to improve its security.

Delegate is a wrapper over functions and Event is a wrapper over Delegates.

Q40. WHAT IS "THIS" KEYWORD IN C#? WHEN TO USE IT?

this keyword is used to refer to the **CURRENT INSTANCE** of the class like this.

```
public Student(int id, String name) {
    this.id = id;
    this.name = name;
}
```

this keyword avoids the naming confusion between class fields and constructor parameters.

Without this keyword the code will be like this. Now how to know which id is the parameter and which one is the field.

```
class Student {
    public int id;
    public String name;

    public Student(int id, String name) {
        id = id;
        name = name;
    }

    public void showInfo() {
        Console.WriteLine(id + " " + name);
    }
}
```

Q41. WHAT IS THE PURPOSE OF "USING" KEYWORD IN C#?

There are two purpose of using keyword in C#:

- **USING DIRECTIVE**

  using System.IO;
  using System.Text;

- **USING STATEMENT** - The using statement ensures that the **DISPOSE()** method is called even if an exception occurs.

Below is the code example while making database connections.
*See no where the **connection.Close()** is written but using keyword will do it even if an exception occurs.*

```
using(var connection = new SqlConnection("{your-connection-string}"))
{
    var query = "UPDATE YourTable SET Property = Value WHERE Foo = @foo";
    using(var command = new SqlCommand(query,connection))
    {
        connection.Open();
        // Perform your update
        command.ExecuteNonQuery();
    }
}
```

Q42. WHAT IS THE DIFFERENCE BETWEEN "IS" AND "AS" OPERATORS?

1. The **IS** operator is **USED TO CHECK** the type of an object.

```
P o1 = new P();
P1 o2 = new P1();
Console.WriteLine(o1 is P);

//output: true
```

AS operator is used to **PERFORM CONVERSION** between compatible reference type.

```
object[] o = new object[1];
o[0] = "Hello";
string str1 = o[0] as string;
Console.Write(str1);

//output: Hello
```

2. The ***IS*** operator is of boolean type.

whereas ***AS*** operator is not of boolean type.

Q43. WHAT IS THE DIFFERENCE BETWEEN "READONLY" AND "CONSTANT" VARIABLES ?

1. Using readonly fields, we can assign values in **DECLARATION** as well as in the **CONSTRUCTOR** PART.

```
class Example {
    // readonly variables
    public readonly int myvar1;
    public readonly int myvar2 = 200;

    public Example(int b)
    {
        myvar1 = b;
        Console.WriteLine(myvar1);
        Console.WriteLine(myvar2);
    }

    static public void Main()
    {
        Example obj1 = new Example(100);
    }
}

//Output: 100 200
```

1. Using readonly fields, we can assign values in DECLARATION as well as in the CONSTRUCTOR PART.

Using const fields, we can assign values in **DECLARATION PART ONLY**.

```
class InterviewHappy {
    // Constant fields
    public const int myvar = 10;

    static public void Main()
    {
        Console.WriteLine(myvar);
    }
}
//Output: 10
```

1. Using const fields, we can assign values in DECLARATION PART ONLY.

2. The value of the readonly field can be changed but the value of the const field can not be changed.

3. Readonly fields can be created using ***readonly*** keywords and Constant fields are created using **const** keywords.

4. ReadOnly is a ***RUNTIME*** constant and Constant is a **COMPILE** time constant.

Q44. WHAT IS "STATIC" CLASS? WHEN TO USE IT?

A static class is a class whose object cannot be created, and which cannot be inherited.

What is the use of creating Static classes?
Static classes are used as containers for static members like methods, constructors and others.

Below is the example of Static class which is containing static members only

```
public static class MyCollege
{
    //static fields
    public static string CollegeName;
    public static string Address;

    //static constructor
    static MyCollege()
    {
        CollegeName = "XYZ College of Technology";
    }

    // static method
    public static string CollegeBranch()
    {
        return "Computers";
    }
}
class Program
{
    static void Main(string[] args)
    {
        Console.WriteLine(MyCollege.CollegeName);

        Console.WriteLine(MyCollege.CollegeBranch());
    }
}
```

Q45. WHAT IS THE DIFFERENCE BETWEEN “VAR” AND “DYNAMIC” IN C#?

VAR - The type of the variable is decided by the compiler at compile time.

DYNAMIC - The type of the variable is decided by the compiler at run time.

```
public class HelloWorld
{
    public static void Main(string[] args)
    {
        var a = 10;

        a = "Happy"; //Build error,
        //error CS0029: Cannot implicitly convert type 'string' to 'int'

        dynamic b = 10;

        b = "Happy"; //No Build error
    }
}
```

Q46. WHAT IS ENUM KEYWORD USED FOR?

An enum is a special "class" that represents a group of constants.

For example, you can use Enum class in below cases also
Weekdays – Sunday, Monday, Tuesday….
Fruits – Apple, Orange, Grapes…

```
enum Level
{
  Low,
  Medium,
  High
}
class Program
{
  static void Main(string[] args)
  {
    Level myVar = Level.Medium;

    Console.WriteLine(myVar);
  }
}
}

//Output: Medium
```

Q47. WHAT IS BOXING AND UNBOXING?

Boxing

Boxing is the process of converting from value type to reference type.

Unboxing

Unboxing is the process of converting reference type to value type.

Unboxing is explicit conversion process.

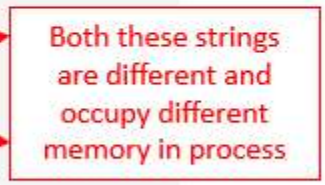
```
public static void Main(string[] args)
{
    int num = 23;

    Object Obj = num;   //Boxing

    int i = (int)Obj; //Unboxing - Explicit

}
```

Q48. WHAT IS THE DIFFERENCE BETWEEN “STRING” AND “STRINGBUILDER”? WHEN TO USE WHAT?

String is **IMMUTABLE** in C#. That mean you couldn’t modify it after it is created.

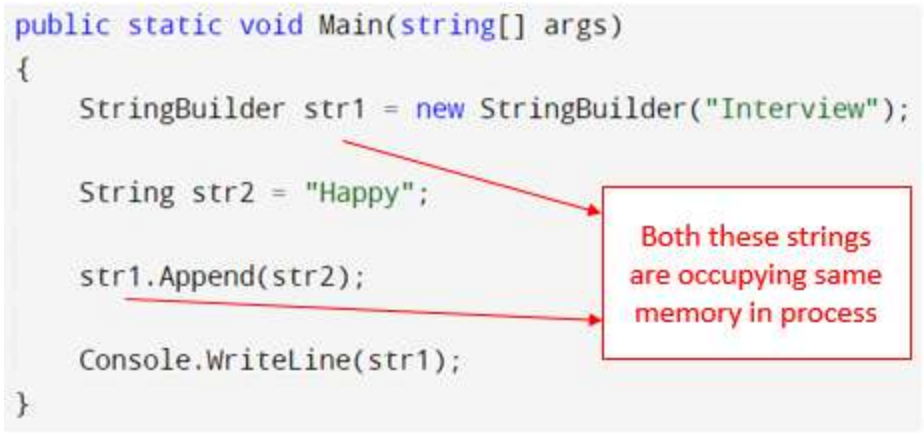
Every time you assign some value to it, it will create a new string.

```
public class HelloWorld
{
    public static void Main(string[] args)
    {
        String str1 = "Interview";

        String str2 = "Happy";

        str1 = str1 + str2;

        Console.WriteLine(str1);
    }
}
```



StringBuilder is **MUTABLE** in C#.

This means that if an operation is performed on the string, it will not create a new instance every time. And the same string is updated.



When to use what?

If you want to **change a string multiple times**, then StringBuilder is a better option from **performance** point of view because it will not require new memory every time.

Q49. WHAT ARE THE BASIC STRING OPERATIONS IN C#?

Concatenate:

Two strings can be concatenated either by using a System.String.Concat or by using + operator.

```
string str1 = "This is one";
string str2 = "This is two";
string str2 = str1 + str2;
```

Modify:

Replace(a,b) is used to replace a string with another string.

```
string str1 = "This is one";
string str2 = str1.Replace("one", "two");
```

Trim:

Trim() is used to trim the white spaces in a string at the end.

Contains:

Check if a string contains a pattern of substring or not.

```
string str = "This is test";
if (str.Contains("test"))
Console.WriteLine("The 'test' was found.");
```

Q50. WHAT ARE NULLABLE TYPES?

Variable types does not hold **null values** so to hold the null values we have to use nullable types.

See the code example below:

```
public class InterviewHappy
{
    public static void Main(string[] args)
    {
        // this will give compile time error
        int j = null;

        // Valid declaration
        Nullable<int> j = null;

        // Valid declaration
        int? j = null;
    }
}
```

Q51. EXPLAIN GENERICS IN C#? WHEN AND WHY TO USE THEM?

Generics allows us to make classes and methods - **type independent or type safe.**

*See the code below Equal will work but **strEqual** will not because the **AreEqual** method is type dependent and the type here is integer which will accept only integer values not strings. To make it type independent we will use Generics.*

```
namespace InterviewHappy
{
    public class GenericExample
    {
        public static void Main(string[] args)
        {
            bool Equal = Calculator.AreEqual(4, 4);
            bool strEqual = Calculator.AreEqual("Interview", "Happy");
            Console.WriteLine(Equal);
            Console.WriteLine(strEqual);
        }
    }
    public class Calculator
    {
        public static bool AreEqual(int value1, int value2)
        {
            return value1.Equals(value2);
        }
    }
}
```

Will work

Will not work

*Now in the below code with the help of generics we are making the **METHOD** independent of type. T is the type here.*
You can use any other alphabet other than "T", but "T" somehow relates to type that's why it is mostly used.

```
namespace InterviewHappy
{
    public class GenericExample
    {
        public static void Main(string[] args)
        {
            bool Equal = Calculator.AreEqual<int>(4, 4);
            bool strEqual = Calculator.AreEqual<string>("Interview", "Happy");
            Console.WriteLine(Equal);
            Console.WriteLine(strEqual);
        }
    }
    public class Calculator
    {
        public static bool AreEqual<T>(T value1, T value2)
        {
            return value1.Equals(value2);
        }
    }
}
```

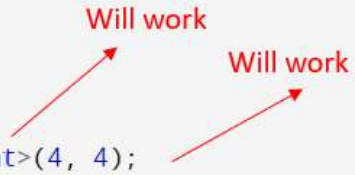
Generics allows us to make classes and method type independent or type safe.

Boxing and unboxing is not required which will impact the performance.

Generics are extensively used by collection classes.

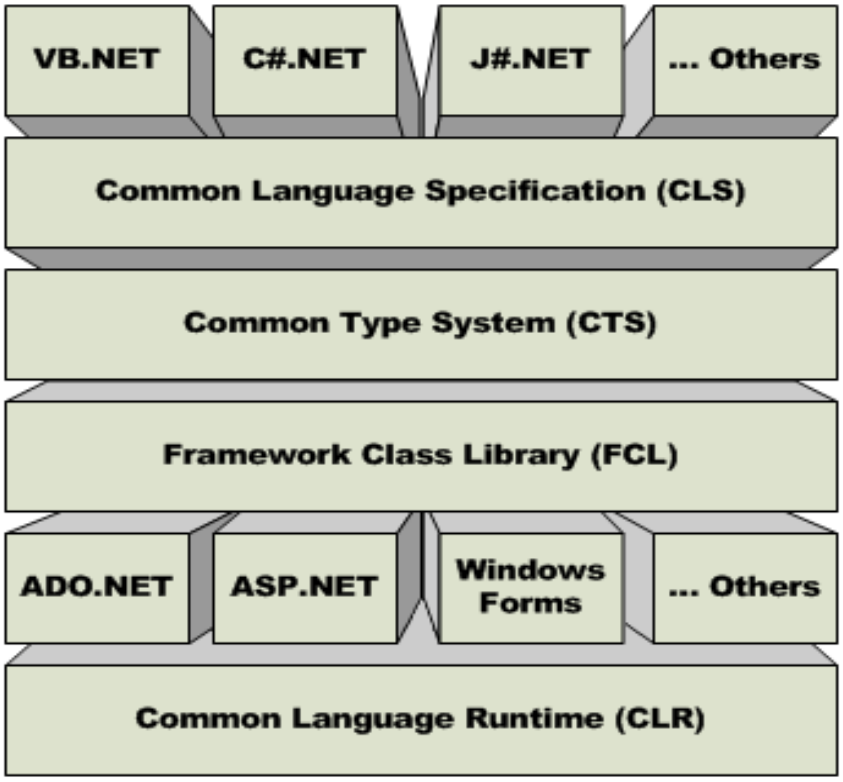
*In the below code with the help of generics we are making the **CLASS** independent of type.*

```
namespace InterviewHappy
{
    public class GenericExample
    {
        public static void Main(string[] args)
        {
            bool Equal = Calculator.AreEqual<int>(4, 4);
            bool strEqual = Calculator.AreEqual<string>("Interview", "Happy");
            Console.WriteLine(Equal);
            Console.WriteLine(strEqual);
        }
    }
    public class Calculator
    {
        public static bool AreEqual<T>(T value1, T value2)
        {
            return value1.Equals(value2);
        }
    }
}
```



Chapter 2 - .NET FRAMEWORK

Q52. WHAT ARE THE IMPORTANT COMPONENTS OF .NET FRAMEWORK? WHAT ARE THEIR ROLES?



- **CLR** – Common Language Runtime (CLR) manages the **execution of programs** written in any language that uses the .NET Framework, for example C#, VB.Net, F# and so on.
  *CLR does various operations like **memory management, security** checks etc.*

- **CTS** – CTS stands for Common Type System. It has a set of rules which state how a **data type** should be declared, defined and used in the program.
  For example, int, string, double, all are types managed by CTS.

- **CLS** – CLS stands for Common Language Specification and it is a **subset of CTS**. It defines a set of rules and restrictions that every language must follow which runs under .NET framework.
  For example, you write a program in different .NET languages C#, VB.NET, J# but if their logic and output is same, then the compiled output assembly will be same for all of them.

- **FCL or BCL** – Framework Class Library is the collection of classes, namespaces, interfaces and value types that are used for .NET applications.
  For example, String class and its methods like Replace, Substring is provided by .NET framework only and it resides in Base or Framework class library.

Q53. WHAT IS AN ASSEMBLY? WHAT ARE THE DIFFERENT TYPES OF ASSEMBLY IN .NET?

Assembly is unit of deployment like EXE or a DLL.

When you create a code and build the solution then the .NET Framework convert it into assembly which you can see inside bin folder.

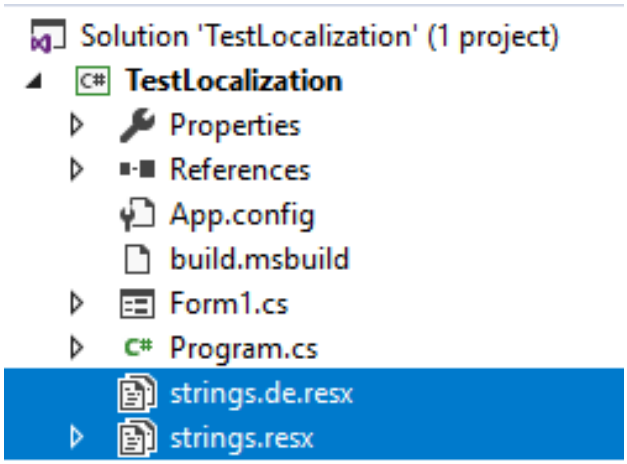
There are 3 types of assemblies:

1. **PRIVATE ASSEMBLEY** - A private assembly is normally used by a single application only. It is not accessible outside.

2. **SHARED ASSEMBLY** - Shared assemblies are usually libraries of code, which multiple applications will use. A shared assembly is normally stored in the global assembly cache.

3. **SATELLITE ASSEMBLY -** A satellite Assembly is defined as an assembly with resources only, no executable code.

Below resx files are the example of satellite assembly.



Q54. WHAT IS GAC?

GAC stands for Global Assembly Cache.

GAC is the **place where shared assemblies get stored in a system**.

Q55. WHAT IS GARBAGE COLLECTION(GC)?

The garbage collector (GC) manages the allocation and release of memory.

For example, in below code you will create object object1 and object2 of classes Employee and Manager in your application. Now you will use these objects for calling any method of that class or assigning some property.

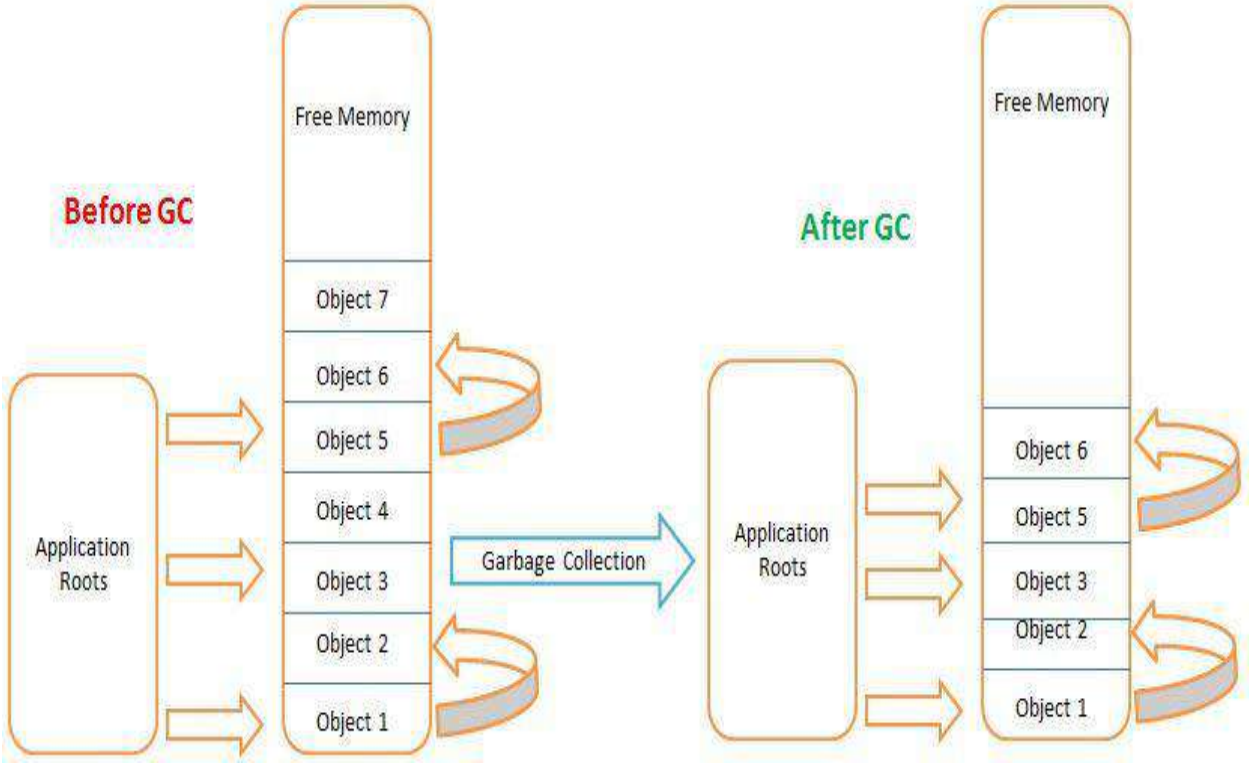
```
public static void Main(string[] args
{
    Employee object1 = new Employee()
    object1.GetSalary();

    Manager object2  = new Manager();
    object2.GiveSalary();

}
```

But after the use you did not destroy these objects. Destroying or disposing the objects is necessary, because it is occupying memory which has to be released.

So that is what garbage collector do. Release the objects when their work is done. It is done AUTOMATICALLY, no need to code for it.

Here is the diagram, see how there are so many objects present in the memory and so less free memory space is there before Garbage Collector run.

And after Garbage collector run many unused objects or resources are released and the free memory level has increase which is good for the performance.



Q56. WHAT ARE GENERATIONS IN GARBAGE COLLECTION?

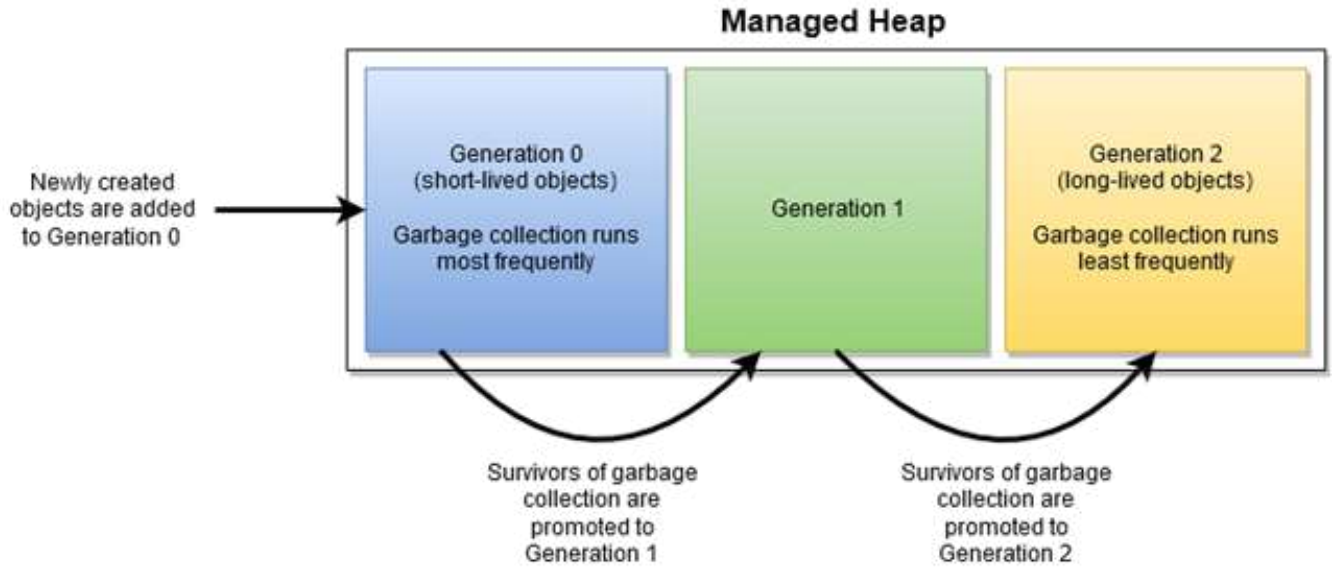
Generation is a mechanism used by Garbage Collection to collect the short-lived objects more frequently than the longer-lived object.

There are 3 types of generations generation 0 ,1 and 2.

See in the below diagram, when garbage collection will run first time, it will put the newly created objects in Generation 0. Then when the garbage collector will run second time then it will destroy the objects from Generation 0 which are no more used and moves the object to Generation 1 which are frequently used.

Garbage collector will do the same process with Generation 1 also.

Now the Garbage collector will more frequently visit Generation 0 to dispose the less used objects and then less frequently visit the Generation 1 and 2 because it have object which are more in use.



Q57. WHAT IS THE DIFFERENCE BETWEEN "DISPOSE" AND "FINALIZE"?

- Dispose is a method of IDisposable interface. Inside this method developer has to write the code to clean or destroy the objects which are no more required.

- Finalize is called by **GARBAGE COLLECTOR** automatically and need not to be called by the user code to run.

There is no **performance** cost associated with Dispose method, as the developers know when the objects will be created and where to clean up them.

There is performance cost associated with Finalize method.

For example, if GC is running in every 10 minutes but there are no objects for cleaning then it is just wasting the memory which it is using for running.

Dispose method Example

```
public class Demo : IDisposable
{
      private bool disposed = false;

      public void Dispose(){

         Dispose(true);

         GC.SuppressFinalize(this);

      }
      protected virtual void Dispose(bool disposing)
      {

         if (!disposed){

            if (disposing){
               //clean up managed objects
            }

            //clean up unmanaged objects
            disposed = true;
         }
      }
}
```

Q58. WHAT IS THE DIFFERENCE BETWEEN "FINALIZE" AND "FINALLY" METHODS?

- **Finalize** – This method is used for **garbage collection**.
  So before destroying an object this method is called as part of clean up activity by GC.

```
class Car
{
    ~Car()  // finalizer
    {
        // cleanup statements...
    }
}
```

- **Finally** – This method is used for executing the code irrespective of exception occurred or not. It is a part of **exception handling**.

```
try
{
   SqlConnection con = new SqlConnection(connectionString);
   //Some logic
   //Error occured
}
catch(ExceptionName ex) {
    //Error handled
}
finally
{
    //Connection closed
    con.Close();
}
```

Q59. CAN WE FORCE GARBAGE COLLECTOR TO RUN?

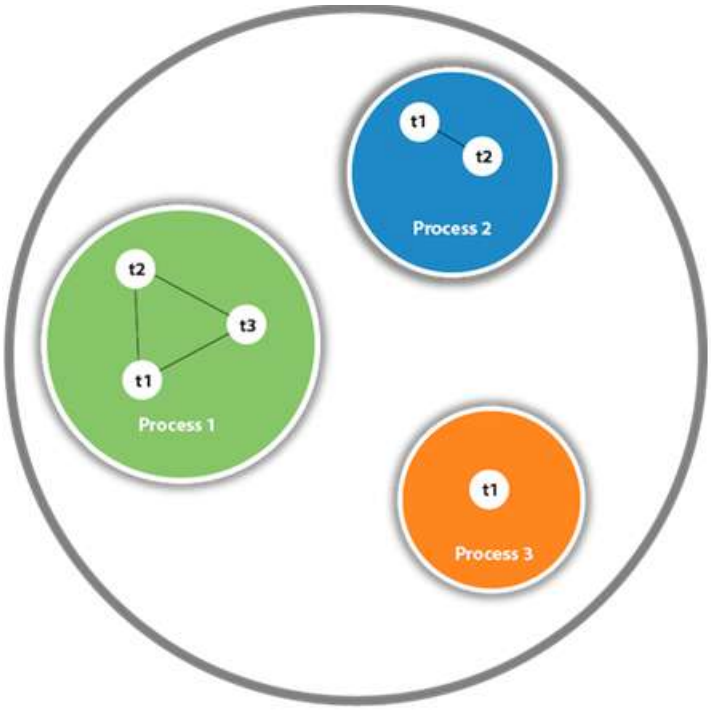
YES. By this method GC.COLLECT() but this is not recommended, instead use Dispose method.

Q60. WHAT IS THE DIFFERENCE BETWEEN PROCESS AND THREAD?

- A Process is an **execution** of a specific program/application.
- A thread is **the smallest unit of process that can be performed in an OS**.

See the diagram below, how t1, t2, t3 threads are linked inside Process1, 2 and 3.

But Process 1,2,3 are not linked. They are isolated.

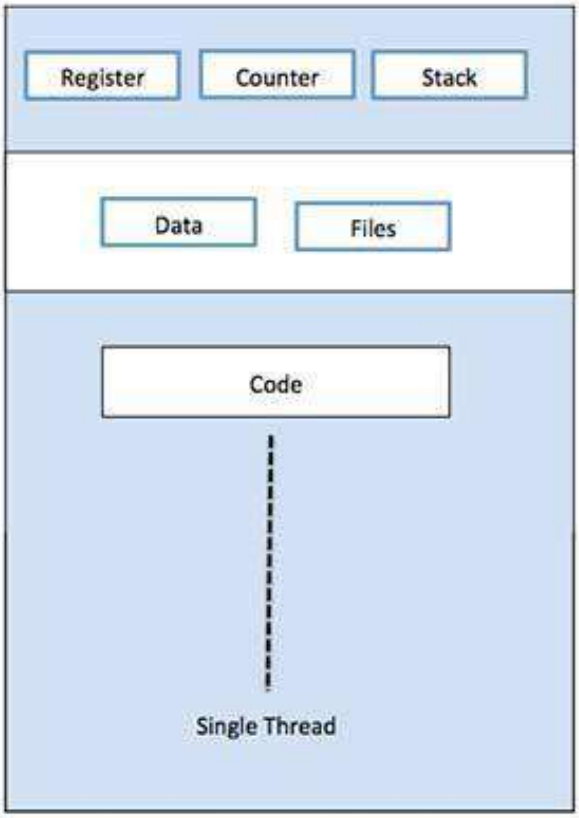


Q61. EXPLAIN MULTITHREADING?

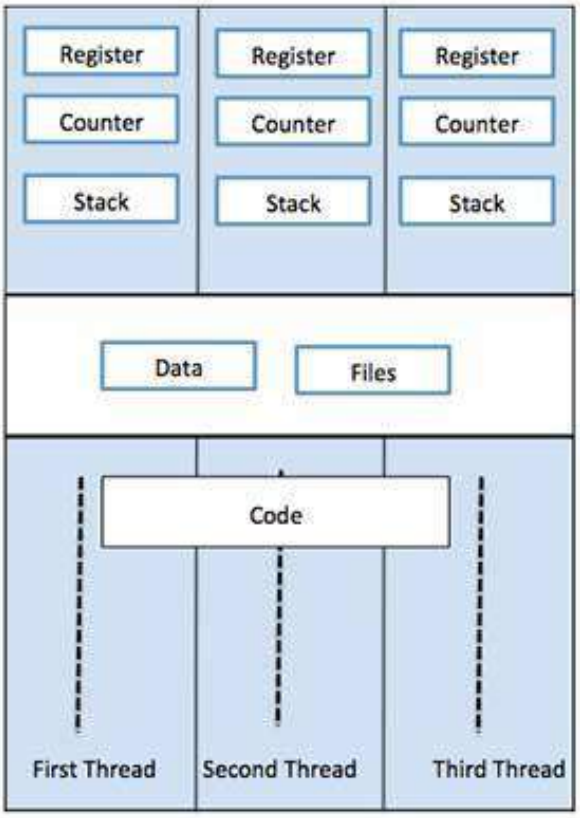
- Multithreading in C# is a process in which multiple threads work **SIMULTANEOUSLY**.

It is a way to achieve **MULTITASKING**. It saves time because multiple tasks are being executed at a time.

To create multithreaded application in C#, we need to use **SYSTEM.THREDING** namespace.



Single Process P with single thread



Single Process P with three threads

Q62. WHAT IS THE DIFFERENCE BETWEEN THREADS AND TASKS?

Task are **WRAPPER** around Thread classes.

Thread is a computer programming concept, whereas Task is not a computer programming concept. Task is created by Microsoft(.NET Framework) in order to work easily with threads.

Task internally uses Threads only.

You can also say that Thread is like a bicycle with which you can go anywhere, but it will need hard work and time. Whereas Task is like a motorcycle, it will also take you to anywhere but with speed and easiness.

Below are some advantages of Tasks over Threads:

- A Task can **RETURN A RESULT** but there is no proper way to return a result from Thread.
- We can apply **CHAINING** on multiple tasks but we can not in threads.
- We can apply **PARENT/CHILD** relationship in Tasks. A Task at one time becomes parent of multiple tasks. Parent Task does not complete until it's child tasks are completed. We do not have any such mechanism in Thread class.

Q63. WHAT IS THE ASYNC AND AWAIT IN TASK?

This is the Microsoft definition:

The async keyword turns a method into an async method, which allows you to use the await keyword in its body. When the await keyword is applied, it suspends the calling method and yields control back to its caller until the awaited task is complete. await can only be used inside an async method.

I have explained the below example in my lecture video, so please visit my video for understanding this:

```
public class HelloWorld
{
    public static void Main(string[] args)
    {
        int i = Method1();
        Console.WriteLine (i);

        int j = Method2();
        Console.WriteLine (j);

        int k = Method3();
        Console.WriteLine (k);

        Console.Read();
    }

    public static int Method1() {
        Thread.Sleep(500);
        return 10;
    }

    public static int Method2() {
        return 20;
    }

    public static int Method3() {
        return 30;
    }
}
//Output 10 20 30
```

Synchronous Programming

```
public class HelloWorld
{
    public static void Main(string[] args)
    {
       Task task1 = Task.Run(() => {
           int i = Method1();
           Console.WriteLine (i);
       });

       Task task2 = Task.Run(() => {
           int j = Method2();
           Console.WriteLine (j);
       });

       int k = Method3();
       Console.WriteLine (k);
       Console.Read();
    }
    public static int Method1() {
        Thread.Sleep(500);
        return 10;
    }
    public static int Method2() {
        return 20;
    }
    public static int Method3() {
        return 30;
    }
}
//Output: 30 20 10
```

Asynchronous Programming

```
public class HelloWorld
{
    public static void Main(string[] args)
    {
        Method1_2();
        int k = Method3();
        Console.WriteLine (k);
        Console.Read();
    }
    static async void Method1_2()
    {
        Console.WriteLine("Test");
        var i = await Task.Run(() =>
        {
            return Method1();
        });

        Console.WriteLine(i);
        int j = Method2(i);
        Console.WriteLine (j);
    }

    public static int Method1()  {
        Thread.Sleep(500);
        return 10;
    }

    public static int Method2(int i)  {
        return 20 * i;
    }
    public static int Method3()  {
        return 30;
    }
}
//Output: Test 30  10  200
```

Asynchronous using Async & Await

Q64. WHAT IS REFLECTION?

Reflection is the ability of a code to access the metadata of the assembly during runtime. Metadata is information about data.

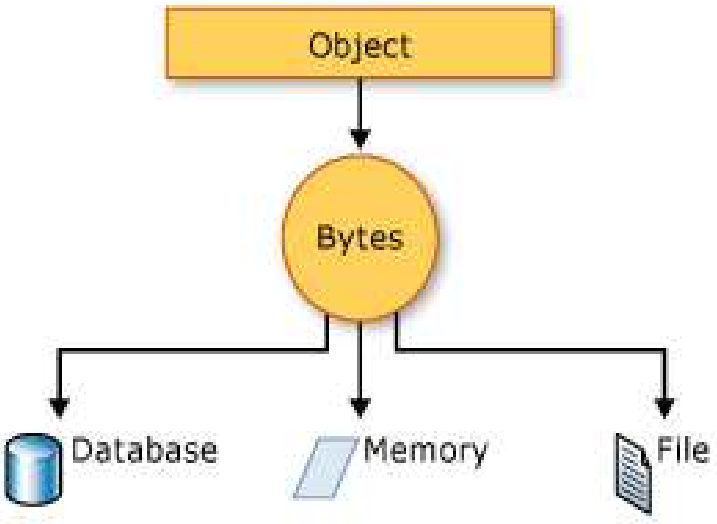
For example, if a book is a data, then author name, book name, book size are metadata as it gives information about the book.

Real time use is - Suppose you have to check the version of the assembly at runtime then you can use reflection like this.

```
using System;
using System.Reflection;
using System.Diagnostics;

public class HelloWorld
{
    public static void Main(string[] args)
    {

        Assembly assembly = Assembly.GetExecutingAssembly();

        FileVersionInfo fvi = FileVersionInfo.GetVersionInfo(assembly.Location);

        string version = fvi.FileVersion;

        Console.WriteLine(version);

    }
}
```

Q65. WHAT IS SERIALIZATION?

Serialization is a process of converting object to its BINARY FORMAT (BYTES)



Once it is converted to bytes, it can be easily stored and written to a disk or any such storage devices.

When to use it?

It is mostly used in Web API to convert class objects into JSON string like this

```
private void JSONSerilaize()
{
    // Serializaion
    Employee empObj = new Employee();
    empObj.ID = 1;
    empObj.Name = "Manas";
    empObj.Address = "India";

    // Convert Employee object to JOSN string format
    string jsonData = JsonConvert.SerializeObject(empObj);
                      jsonData "{\"ID\":1,\"Name\":\"Manas\",\"Address\":\"India\"}"
    Response.Write(jsonData);
}
```

Q66. WHAT IS MEANT BY GLOBALIZATION AND LOCALIZATION?

Globalization is the process of designing and developing a software product that functions in **MULTIPLE CULTURES/LOCALES**.

Localization is the process of adapting a globalized application, which you have already processed for localizability, to a **particular** culture/locale.

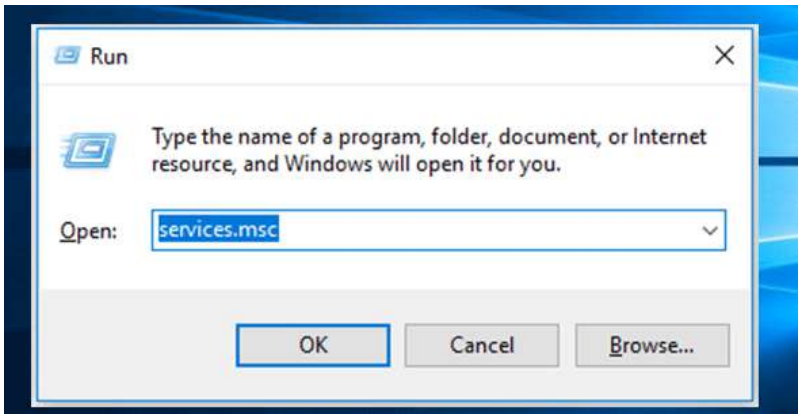
For example, you just started developing an application which supports multiple languages like English, German, French then you will architect your project as per globalization.

After you have created this application, one year later your manager asks you to add one more language like Japanese then you are doing localization for a particular language.

Q67. WHAT ARE WINDOW SERVICES?

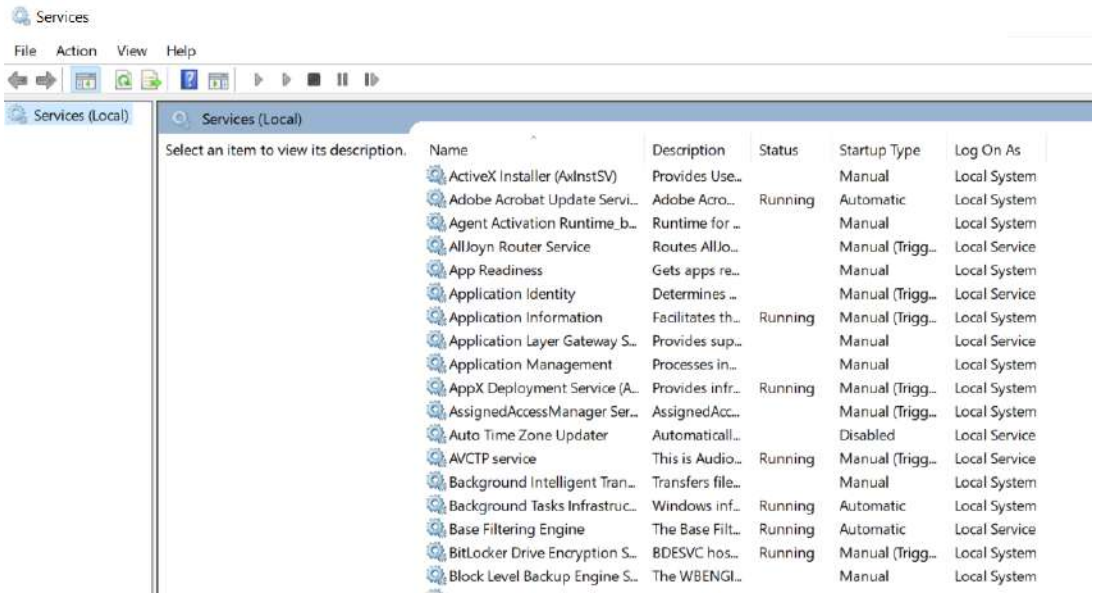
Windows service is a computer program that runs in the **BACKGROUND** to execute some tasks.

You can see the window services in your system by running services.msc command.



Windows services can be started **AUTOMATICALLY** or manually.

You can also manually pause, stop and restart Windows services.



Chapter 3 - SQL

Q68. WHAT IS THE DIFFERENCE BETWEEN DBMS AND RDBMS?

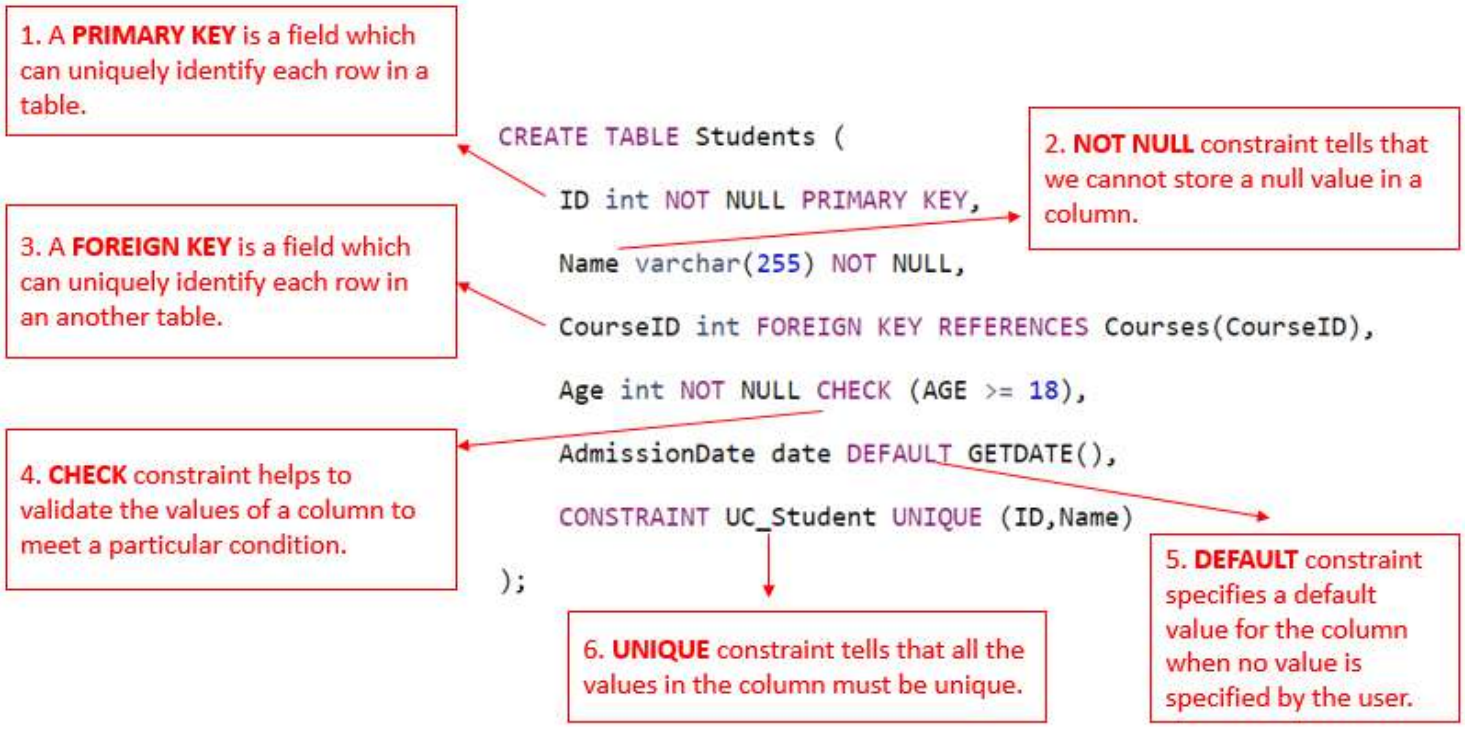
DBMS	RDBMS
1. DBMS stores data as file.	RDBMS stores data in **TABULAR** form.
2. No relationship between data.	Data is stored in the form of tables which are RELATED to each other. Eg: Foreign key relationship.
3. Normalization is not present.	**NORMALIZATION** is present.
4. It deals with small quantity of data.	It deals with **LARGE** amount of data.
5. Examples: XML	Examples: MySQL, PostgreSQL, SQL Server, Oracle etc.

Q69. WHAT IS A CONSTRAINT IN SQL? WHAT ARE ITS TYPES.

SQL constraints are **used to specify rules for the data in a table**.

Constraints are used to limit the type of data that can go into a table.

Below in the diagram you can see 6 constraints and their role in SQL Server:



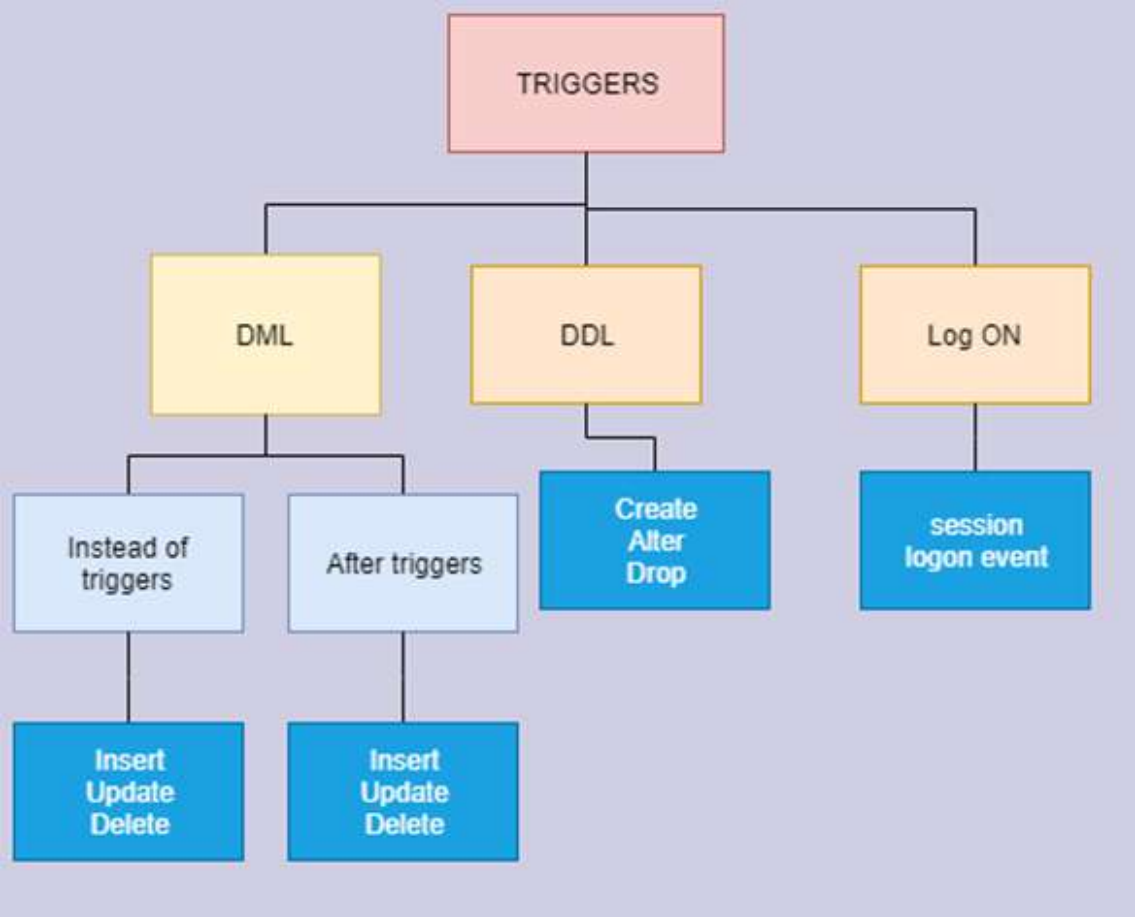
Q70. WHAT IS THE DIFFERENCE BETWEEN PRIMARY KEY AND UNIQUE KEY?

	Primary Key	Unique Key
1	Primary Key Can't Accept Null Values.	Unique Key Can Accept Only One Null Value
2	Creates Clustered Index	Creates Non-Clustered Index
3	Only One Primary key in a Table	More than One Unique Key in a Table.

Q71. WHAT ARE TRIGGERS AND TYPES OF TRIGGERS?

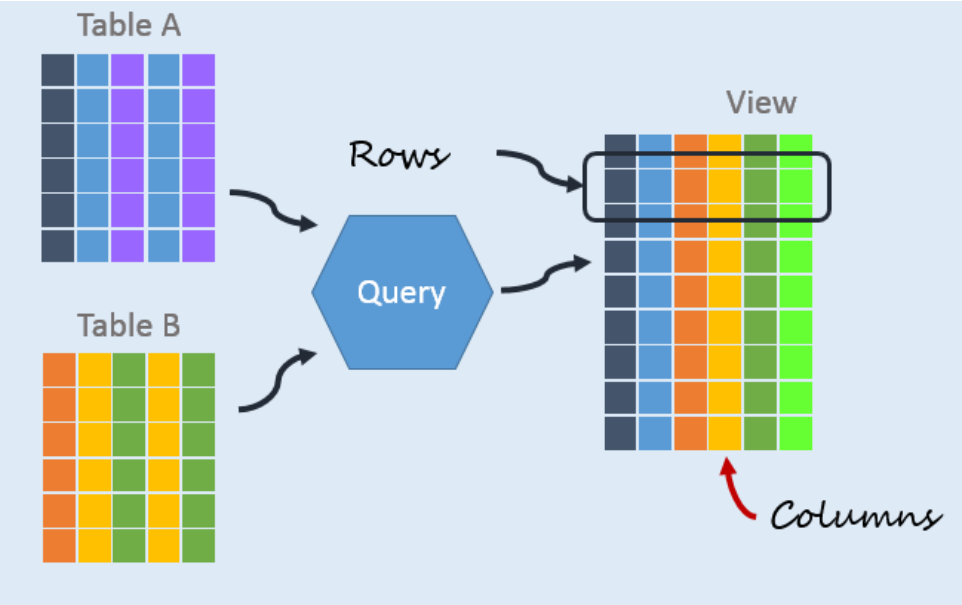
Triggers are stored programs, which are **AUTOMATICALLY** executed or fired when some events (insert, delete and update) occur.

Types of triggers:



Q72. WHAT IS A VIEW?

A view is a **VIRTUAL** table which consists of a subset of data contained in a table or more than one table.



Views are not stored in memory like tables then why to use views?

Because of 2 reasons:

1. Views can be easily Indexed to improve the **performance**.

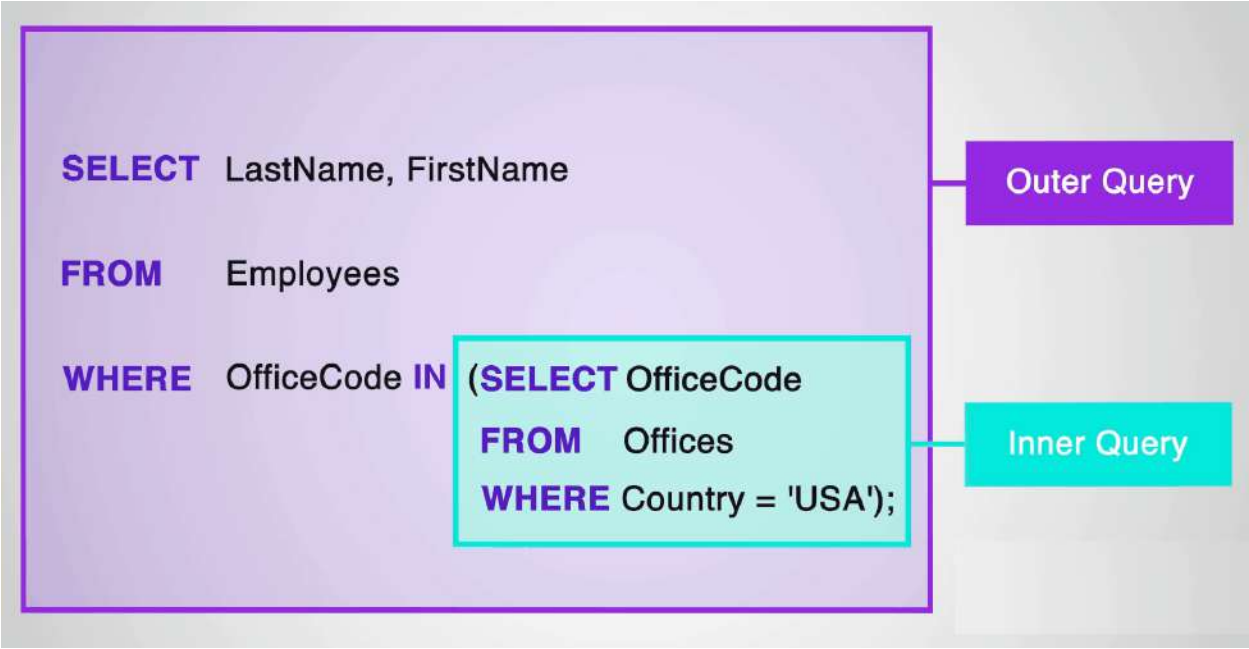
2. **Extra security** – DBA can hide the actual table names and expose views for Read only operations.

Remember, in a view query is stored but the data is never stored like a table.

Q73. WHAT IS SUB QUERY OR NESTED QUERY OR INNER QUERY IN SQL?

A Subquery or Inner query or a Nested query is a query within another SQL query and **embedded** within the **WHERE** clause.

Like this. See there is an outer query and an inner query is embedded in where clause:

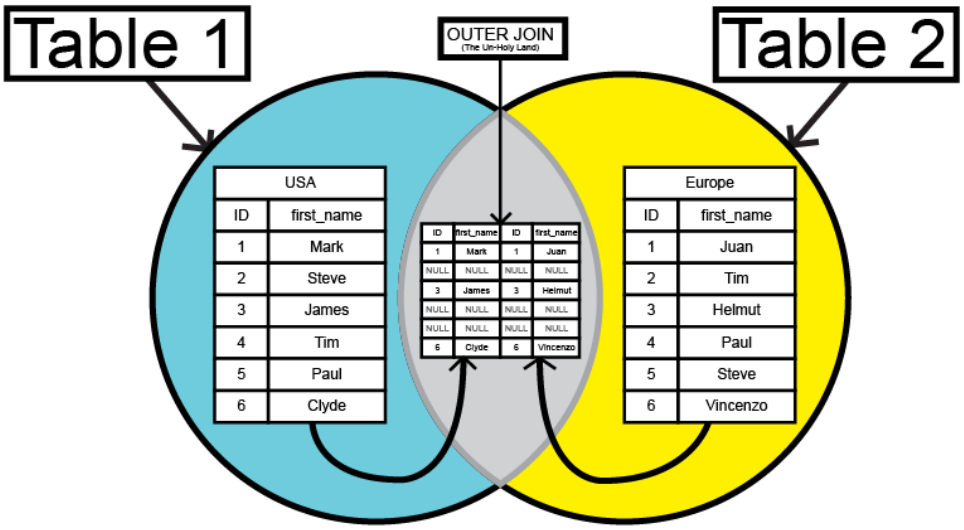


Q74. WHAT IS THE DIFFERENCE BETWEEN DELETE, TRUNCATE AND DROP COMMANDS?

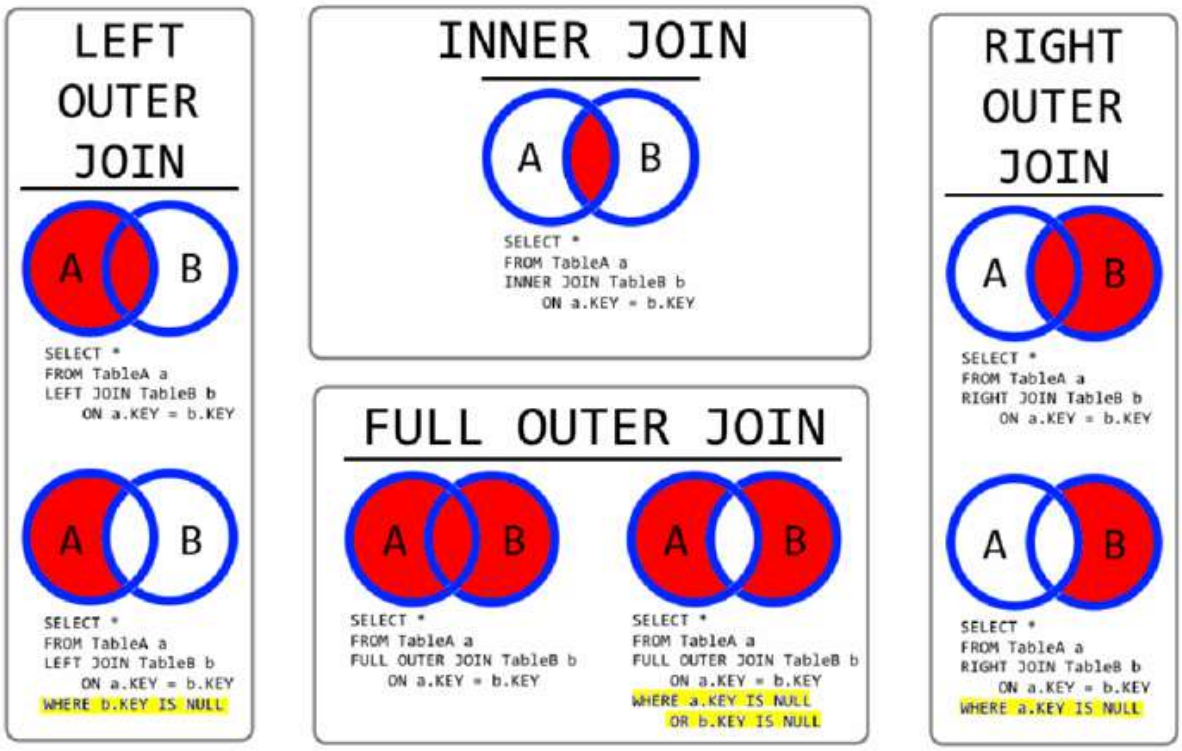
DELETE	TRUNCATE	DROP
1. It is a DML.	1. It is a DDL.	1. It is a DDL.

Q75. WHAT ARE JOINS IN SQL?

A join clause is used to **COMBINE** rows from two or more tables, based on a related column between them.



Q76. WHAT ARE THE TYPES OF JOINS IN SQL SERVER?



- **LEFT OUTER JOIN**: As you can see in the diagram – It returns all records from the left table, and the matched records from the right table.

- **RIGHT (OUTER) JOIN**: Returns all records from the right table, and the matched records from the left table.

- **FULL (OUTER) JOIN**: Returns all records when there is a match in either left or right table.

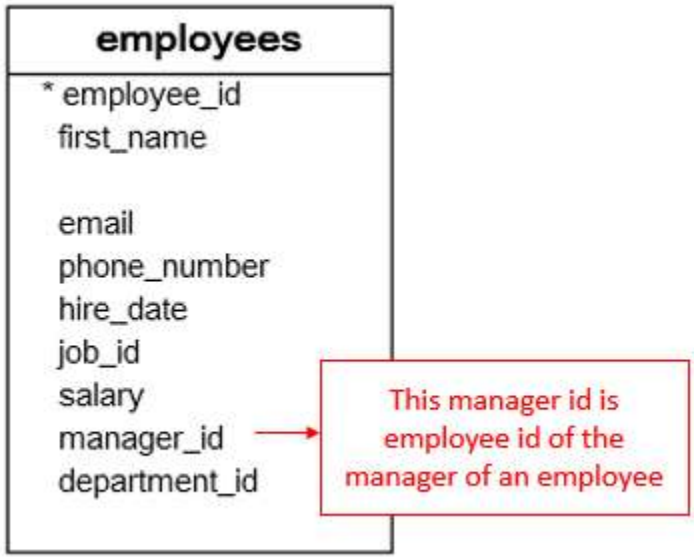
- **(INNER) JOIN**: Returns records that have matching values in both tables. Inner join is the mostly used join in all joins.

Q77. WHAT IS SELF-JOIN?

A self-join is **a join of a table to itself**.

When to use Self Join?

Suppose you have an employee table, here all the employees with their employee id are present. Managers are also employees, so they are also present with their employee id.
Now most of the employee will have a manager, therefore in the same table there is a manager id will be there which is nothing but employee id of the manager of that employee.
Now your task is to get the employee name with his/her manager name.
So, this will be solved by using self join because we don't have any other table here and there is no such column like manager name in this table.



This is the query and the example we will use to get the employee first name and his manager first name
See employees table joining employees table and that is self join.

```
SELECT

    e.first_name AS employee,
    m.first_name AS manager

FROM

    employees e LEFT JOIN
    employees m

    ON m.employee_id = e.manager_id

ORDER BY manager;
```

This is the result we will get:

	employee	manager
▶	Steven King	NULL
	Bruce Ernst	Alexander Hunold
	David Austin	Alexander Hunold
	Valli Pataballa	Alexander Hunold
	Diana Lorentz	Alexander Hunold
	Alexander Khoo	Den Raphaely

Q78. WHAT IS THE DIFFERENCE BETWEEN STORED PROCEDURE AND FUNCTIONS?

1. SP may or may not **return** a value, but function must return a value.

2. SP can have input/**output** parameters, but function only has input parameters.

3. We can c**all** function inside SP, but cannot call SP from a function.

4. We cannot use SP in **SQL statements** like SELECT, INSERT, UPDATE, DELETE, MERGE, etc, but we can use them with function.

*SELECT *, dbo.fnCountry(city.long) FROM city;*

5. We can use try-catch exception handling in SP, but we cannot do that in function.

6. We can use transactions inside SP, but it is not possible in function.

```
--Stored Procedure
CREATE PROCEDURE proc_name
(@Ename varchar(50),
@EId int output)
AS
BEGIN
    INSERT INTO Employee (EmpName) VALUES
(@Ename)
    SELECT @EId= SCOPE_IDENTITY()
END
```

```
--UDF – User Defined Functions
CREATE FUNCTION function_name
(parameters) --only input parameter
RETURNS data_type AS
BEGIN
    SQL statements
    RETURN value
END;
```

Q79. WHAT IS A CURSOR? WHY TO AVOID THEM?

A database Cursor is a control which enables **traversal/ iteration over the rows** or records in the table.

It’s a 5-step process:

1. Declare 2. Open 3. Fetch using while loop 4. Close 5. Deallocate

```
DECLARE
    @product_name VARCHAR(MAX),
    @list_price   DECIMAL;

DECLARE cursor_product CURSOR
FOR SELECT
        product_name,
        list_price
    FROM
        production.products;

OPEN cursor_product;

FETCH NEXT FROM cursor_product INTO
    @product_name,
    @list_price;

WHILE @@FETCH_STATUS = 0
    BEGIN
        PRINT @product_name + CAST(@list_price AS varchar);
        FETCH NEXT FROM cursor_product INTO
            @product_name,
            @list_price;
    END;

CLOSE cursor_product;

DEALLOCATE cursor_product;
```

LIMITATION

A cursor is a **MEMORY** resident set of pointers -- meaning it occupies lots of memory from your system which is **not good for performance**.

Q80. WHAT IS THE DIFFERENCE BETWEEN SCOPE_IDENTITY AND @@IDENTITY?

scope_identity and @@identity, both are used to get the **last value entered in the identity column** of the table.

The @@identity returns the last identity created **in the same session**. The session is the database connection.

```
SELECT @@IDENTITY
```

The scope_identity() function returns the last identity created **in the same session and the same scope**. The scope is the current query or the current stored procedure.

```
SELECT SCOPE_IDENTITY()
```

Normally we have to use scope_identity() function inside stored procedures.

Q81. HOW TO OPTIMIZE A STORED PROCEDURE OR SQL QUERY?

Below are some techniques to optimize a stored procedure:

1. Use SET NOCOUNT ON

2. Specify column names instead of using * in SELECT statement.

3. Use schema name before objects or table names.

   Example: SELECT EmpID, Name FROM dbo.Employee

4. Do not use DYNAMIC QUERIES. They are vulnerable to SQL Injections.

5. Use EXISTS () instead of COUNT ().
   Example: IF(EXISTS (SELECT 1 FROM db.Employees)) is better than SELECT Count(1) FROM dbo. Employee

6. Use TRANSACTION when required only

Q82. WHAT ARE INDEXES IN SQL SERVER?

SQL Indexes are used in relational databases to retrieve data **VERY FAST**.

They are like indexes at the start of the BOOKS, which purpose is to find a topic quickly.

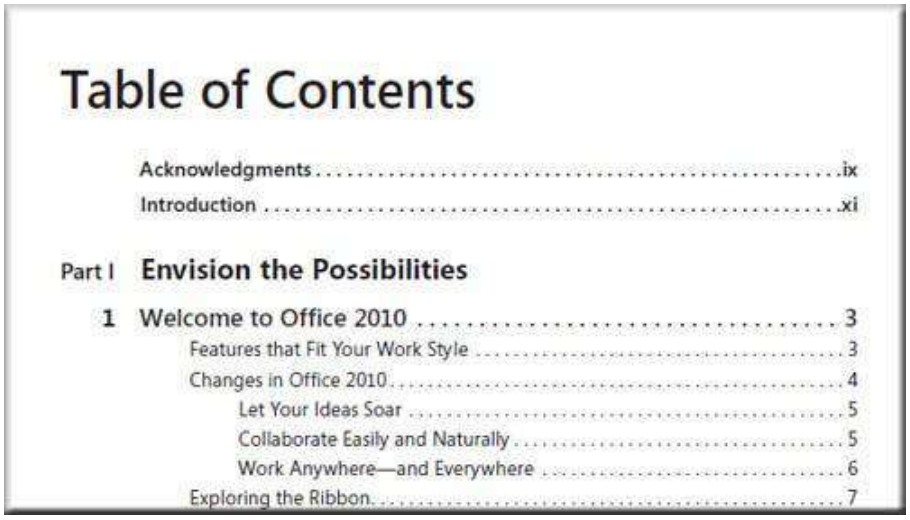


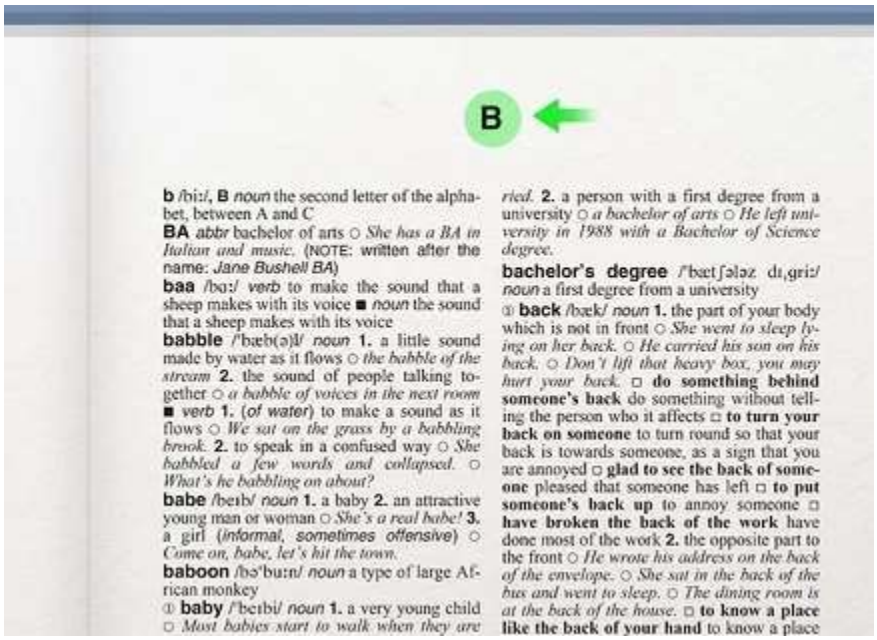
Table of Contents

Q83. WHAT IS CLUSTERED INDEX?

CLUSTERED INDEX

A clustered index defines the order in which data is **physically** stored in a table.

Clustered index are similar to the Dictionary. See how you will find any word. There is only one way to find a word so similarly there can be only one clustered index per table possible.



B

b /biː/, **B** *noun* the second letter of the alphabet, between A and C
BA *abbr* bachelor of arts ○ *She has a BA in Italian and music.* (NOTE: written after the name: *Jane Bushell BA*)
baa /bɑː/ *verb* to make the sound that a sheep makes with its voice ■ *noun* the sound that a sheep makes with its voice
babble /'bæb(ə)l/ *noun* **1.** a little sound made by water as it flows ○ *the babble of the stream* **2.** the sound of people talking together ○ *a babble of voices in the next room* ■ *verb* **1.** (*of water*) to make a sound as it flows ○ *We sat on the grass by a babbling brook.* **2.** to speak in a confused way ○ *She babbled a few words and collapsed.* ○ *What's he babbling on about?*
babe /beɪb/ *noun* **1.** a baby **2.** an attractive young man or woman ○ *She's a real babe!* **3.** a girl (*informal, sometimes offensive*) ○ *Come on, babe, let's hit the town.*
baboon /bə'buːn/ *noun* a type of large African monkey
① **baby** /'beɪbi/ *noun* **1.** a very young child □ *Most babies start to walk when they are ried.* **2.** a person with a first degree from a university ○ *a bachelor of arts* ○ *He left university in 1988 with a Bachelor of Science degree.*
bachelor's degree /'bætʃələz dɪˌgriː/ *noun* a first degree from a university
① **back** /bæk/ *noun* **1.** the part of your body which is not in front ○ *She went to sleep lying on her back.* ○ *He carried his son on his back.* ○ *Don't lift that heavy box, you may hurt your back.* □ **do something behind someone's back** do something without telling the person who it affects □ **to turn your back on someone** to turn round so that your back is towards someone, as a sign that you are annoyed □ **glad to see the back of someone** pleased that someone has left □ **to put someone's back up** to annoy someone □ **have broken the back of the work** have done most of the work **2.** the opposite part to the front ○ *He wrote his address on the back of the envelope.* ○ *She sat in the back of the bus and went to sleep.* ○ *The dining room is at the back of the house.* □ **to know a place like the back of your hand** to know a place

In SQL Server, if you set a primary key on a column then it will automatically create a clustered index on that column.

Q84. WHAT IS NON-CLUSTERED INDEX?

NON-CLUSTERED INDEX

A non-clustered index is stored at one place and table data is stored in another place. So, this index is not physically stored.

It is like the index of a BOOK.
A book can have multiple indexes like one at the start and one at the end like shown below:

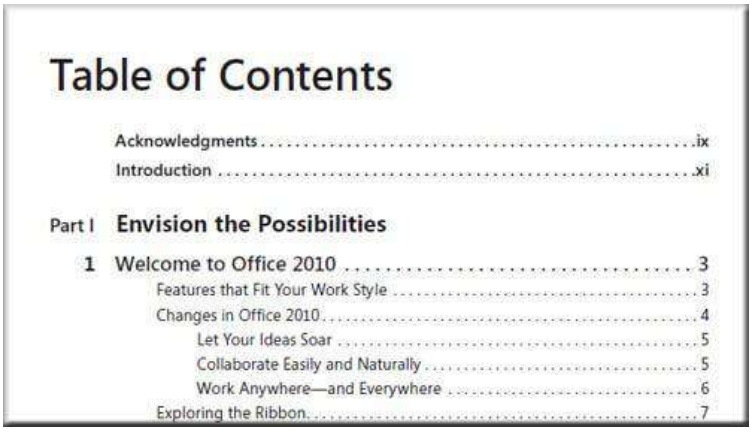
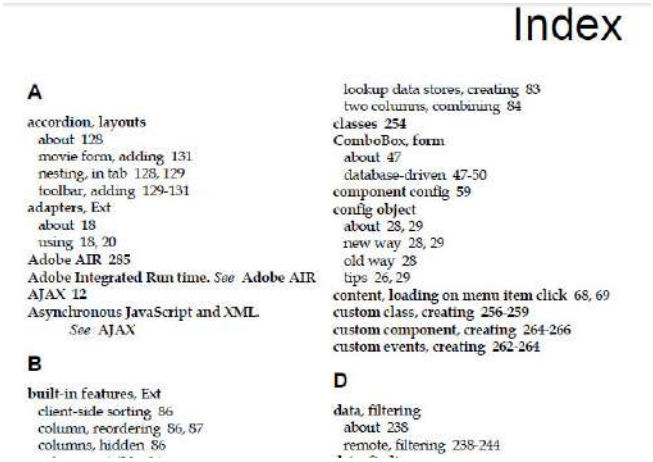


Table of Contents



Index

Similarly, a table can have multiple non-clustered indexes in a table.

Q85. WHAT IS THE DIFFERENCE BETWEEN CLUSTERED AND NON-CLUSTERED INDEX?

1. A clustered index defines the order in which data is **physically** stored in a table.
For example, Dictionary.

A non-clustered index is stored at one place and table data is stored in another place.

For example, Book Index.

2. A table can have only one clustered index.

A table can have multiple non-clustered index.

3. Clustered index is faster.

Non-clustered index is slower.

Q86. HOW TO CREATE CLUSTERED AND NON-CLUSTERED INDEX IN A TABLE?

CLUSTERED INDEX

When you create a PRIMARY KEY constraint, a clustered index on the column or columns is automatically created.

```
CREATE CLUSTERED INDEX <index_name>
ON <table_name>(<column_name> ASC/DESC)
```

NON-CLUSTERED INDEX

```
CREATE NONCLUSTERED INDEX <index_name>
ON <table_name>(<column_name> ASC/DESC)
```

Q87. IN WHICH COLUMN YOU WILL APPLY THE INDEXING TO OPTIMIZE THIS QUERY.

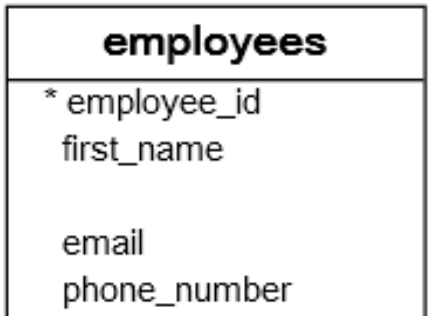
"select id, class from student where name="happy""?

The column **after WHERE** condition, which is "name" here.

Q88. WRITE A SQL QUERY TO FETCH ALL THE EMPLOYEES WHO ARE ALSO MANAGERS?

Use self-join here

Table



SQL Query using self join

```
SELECT

    e.first_name AS employee,
    m.first_name AS manager

FROM
```

This will be the output:

	employee	manager
▶	Steven King	NULL
	Bruce Ernst	Alexander Hunold
	David Austin	Alexander Hunold
	Valli Pataballa	Alexander Hunold
	Diana Lorentz	Alexander Hunold
	Alexander Khoo	Den Raphaely

Q89. HOW TO GET THE NTH HIGHEST SALARY OF AN EMPLOYEE?

Salary
5000
10000
6000
4000
2000
7000

With the below logic one can find 2nd, 3rd, 4th, any highest salary. Below I did it for 3rd highest salary:

1. The logic is first select TOP 3 salaries in descending order.

```
SELECT DISTINCT TOP N SALARY
    FROM tbl_Employees
    ORDER BY SALARY DESC
```

2. Put the result in "Result" and then do order by asc

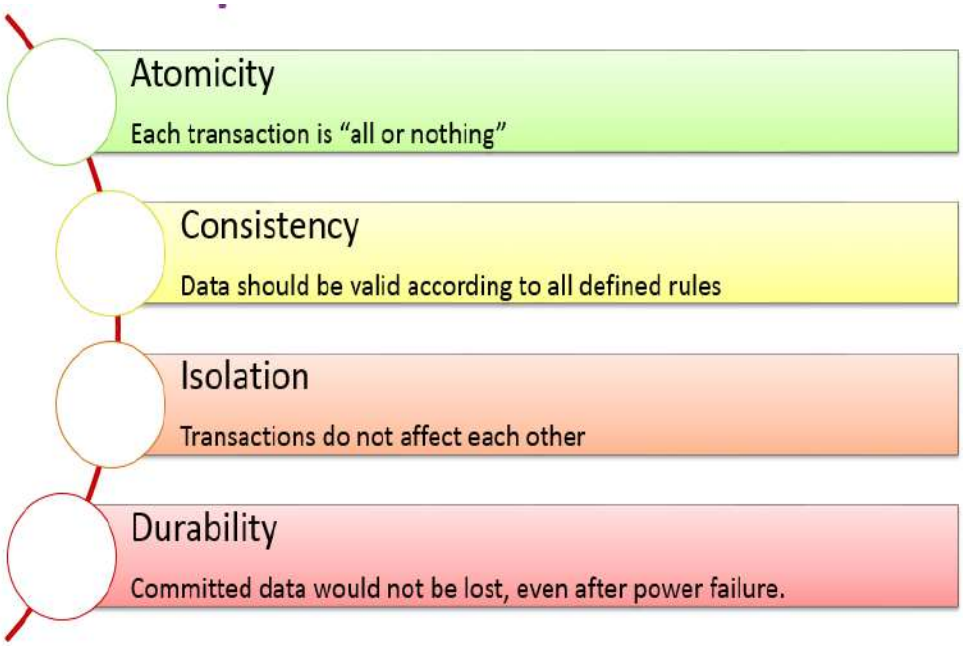
```
SELECT SALARY
FROM (

    SELECT DISTINCT TOP N SALARY
    FROM tbl_Employees
    ORDER BY SALARY DESC
```

3. Select top 1 salary from result set

```
SELECT TOP 1 SALARY
FROM (

    SELECT DISTINCT TOP N SALARY
    FROM tbl_Employees
    ORDER BY SALARY DESC
```

Q90. WHAT ARE ACID PROPERTIES?

ACID properties are used when you are handling **transactions** in SQL.



Q91. WHAT IS AUTO INCREMENT/ IDENTITY COLUMN IN SQL SERVER?

Auto-increment allows a unique number to be **generated automatically** when a new record is inserted into a table.

```
CREATE TABLE Persons (
   Personid int IDENTITY(1,1) PRIMARY KEY,
   FirstName varchar(255),
   Age int
```

);

Q92. WHAT IS THE DIFFERENCE BETWEEN HAVING CLAUSE AND WHERE CLAUSE?

1. WHERE Clause is used before GROUP BY Clause.

HAVING Clause is used after GROUP BY Clause.

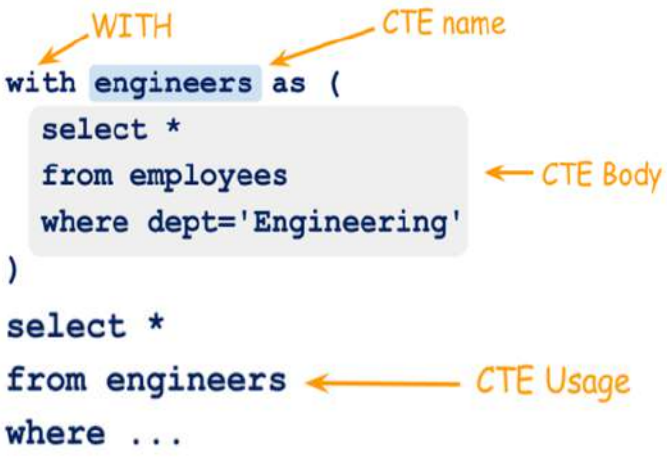
2. WHERE Clause cannot contain AGGREGATE function.

HAVING Clause can contain aggregate function.

```
SELECT COUNT(CustomerID), Country
FROM Customers
WHERE Country = "India"
GROUP BY Country
HAVING COUNT(CustomerID) > 5;
```

Q93. WHAT IS CTE IN SQL SERVER?

A Common Table Expression, is a TEMPORARY named result set, that you can reference within a SELECT, INSERT, UPDATE, or DELETE statement.



Q94. WHAT ARE MAGIC TABLES IN SQL SERVER?

Magic tables are the temporary logical tables that are created by the SQL server whenever there are **insertion or deletion or update**(D.M.L) operations.

Types of magic tables:

INSERTED – The recently inserted row gets added to the INSERTED magic table.
DELETED – The recently deleted row gets added to the DELETED magic table.

The use of magic tables are TRIGGERS.

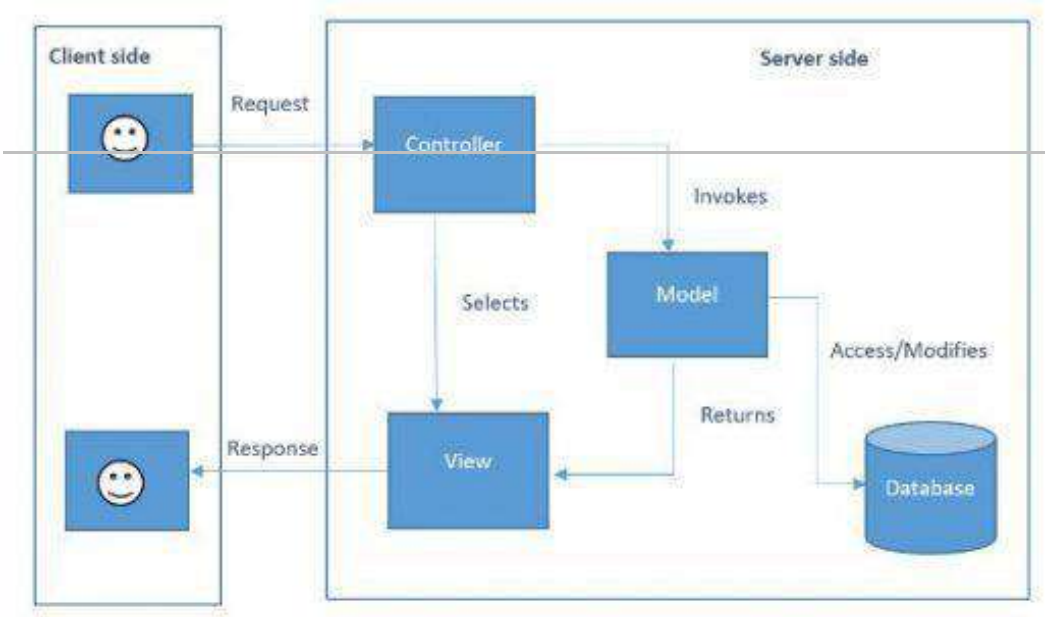
Chapter 4 - ASP.NET MVC

Q95. WHAT IS MVC (MODEL VIEW CONTROLLER)? EXPLAIN MVC LIFE CYCLE.

MVC is a framework for building web applications using an MVC (Model View Controller) design:

- The **Model** represents the data.
- The **View** displays the data.
- **Controllers** act as an interface between Model and View components to process all **the business logic** and then render the final output to view.

Below is the flow of the request processing in MVC:



Q96. WHAT ARE THE ADVANTAGES OF MVC OVER WEB FORMS?

Below are the advantages of MVC:

1. SOC (SEPARATION OF CONCERNS) - Separation of Concerns is one of the core advantages of ASP.NET MVC. The MVC framework provides a clean separation of the UI, Business Logic, Model or Data.

Meaning is view is not dependent on controller and controller is not dependent on view. In Web forms one aspx file will have one aspx.cs file, which means UI(aspx) is TIGHTLY COUPLED with logic (.aspx.cs).

2. MULTIPLE VIEW SUPPORT - Due to the separation of the model from the view, the user interface can display multiple views of the same data at the same time.

3. CHANGE ACCOMMODATION - User interfaces tend to change more frequently than business rules (different colors, fonts, screen layouts, and levels of support for new devices such as cell phones or PDAs) because the model does not depend on the views, adding new types of views to the system generally does not affect the model. As a result, the scope of change is confined to the view.

In Web forms one aspx file will have one aspx.cs file, which means UI(aspx) is TIGHTLY COUPLED with logic (.aspx.cs).

In MVC one view can interact with multiple controllers and one controller can interact with multiple views therefore no TIGHT coupling.

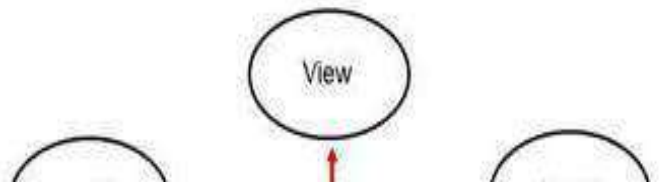
4. MORE CONTROL - The ASP.NET MVC framework provides more control over HTML, JavaScript, and CSS than the traditional Web Forms.

5. TESTABILITY - ASP.NET MVC framework provides better testability of the Web Application and good support for test driven development too.

6. LIGHTWEIGHT - ASP.NET MVC framework doesn't use View State and thus reduces the bandwidth of the requests to an extent.

7. FULL FEATURES OF ASP.NET - One of the key advantages of using ASP.NET MVC is that it is built on top of the ASP.NET framework and hence most of the features of the ASP.NET like membership providers, roles, etc can still be used.

Q97. WHAT ARE THE DIFFERENT RETURN TYPES OF A CONTROLLER ACTION METHOD?



1. VIEWRESULT - This return type is used to return a webpage from an action method.

2. PARTIALVIEWRESULT - This return type is used to send a part of a view that will be rendered in another view.

3. REDIRECTRESULT - This return type is used to redirect to any other controller and action method depending on the URL.

4. REDIRECTTOROUTERESULT - This return type is used when we want to redirect to any other action method.

5. CONTENTRESULT - This return type is used to return HTTP content type like text/plain as the result of the action.

6. JSONRESULT - This return type is used when we want to return a JSON message.

7. JAVASCRIPTRESULT - This return type is used to return JavaScript code that will run in the browser.

8. FILERESULT - This return type is used to send binary output in response.

9. EMPTYRESULT - This return type is used to return nothing (void) in the result.

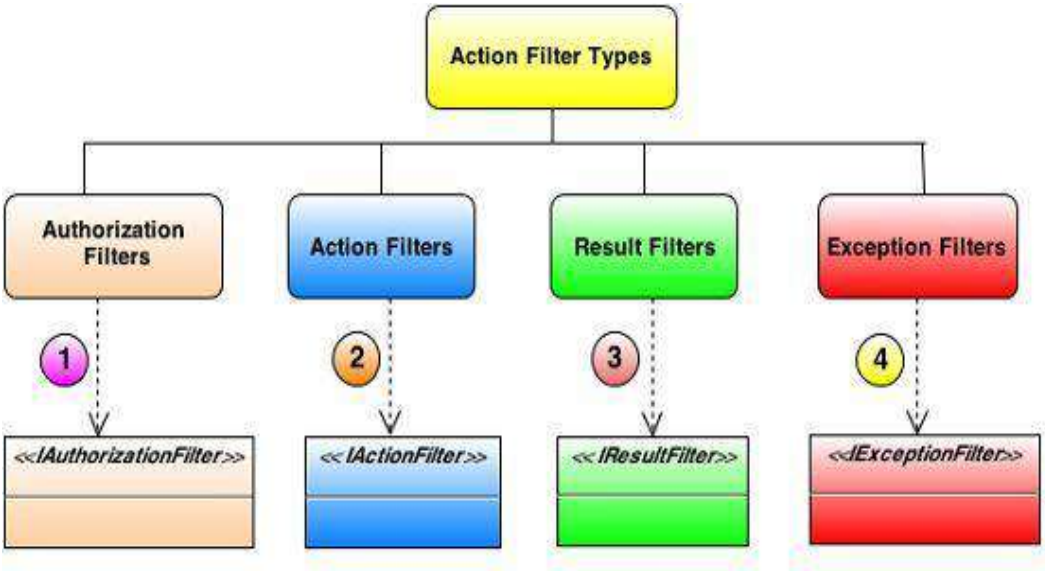
Q98. WHAT ARE FILTERS AND THEIR TYPES IN MVC?

ASP.NET MVC Filter is **a custom class where you can write custom logic to execute before or after an action method executes**.

For example, here is the code of Authorize filter. If any request will try to access this action method DoAdminStuff, first this authorization filter will check that Roles In the request are of Admin, then only it will allow. So, this filter run before the execution of the action method.

```
[Authorize]
0 references
public class AuthorizeController : Controller
{
    [Authorize(Roles = "Admin")]
    0 references
    public ActionResult DoAdminStuff()
    {
        return View();
    }
}
```

Types of Action Filters Are:



Filter Type	Description	Built-in Filter	Interface
Authorization filters	Performs authentication and authorizes before executing an action method.	[Authorize], [RequireHttps]	IAuthorizationFilter
Action filters	Performs some custom operation before and after an action method executes.		IActionFilter
Result filters	Performs some operation before or after the execution of the view. For example if you want to modify a view result right before the view is rendered to the browser.	[OutputCache]	IResultFilter
Exception filters	Performs some operation if there is an unhandled exception thrown during the execution of the ASP.NET MVC pipeline.	[HandleError]	IExceptionFilter

---

Q99. WHAT IS AUTHENTICATION AND AUTHORIZATION IN ASP.NET MVC?

- Authentication is the process of obtaining some sort of credentials (username, password) from the users and using those credentials to verify the **USER'S IDENTITY**.

- Authorization is the process of allowing an authenticated user **ACCESS** to resources.

Authentication is always done before Authorization.

Q100. WHAT ARE THE TYPES OF AUTHENTICATION IN ASP.NET MVC?

1. FORM BASED AUTHENTICATION - It is based on cookies, the authentication and permission settings are stored in cookies.

Example: You can create a Login page for users to get authenticated before viewing content in your application.

```
<authentication mode="Forms">
        <forms loginUrl="Accounts/Login"></forms>
</authentication>
```

2. PASSPORT AUTHENTICATION - Passport authentication is a centralized authentication service provided by Microsoft.

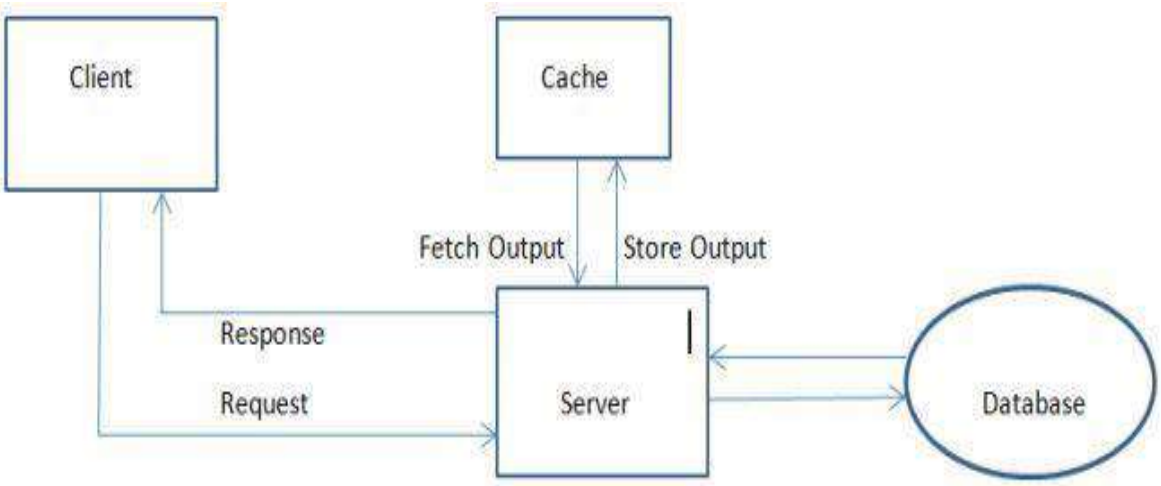
In your application you can apply authentication with the help of Microsoft authentication. This is an enhanced way of securing the authentication in your applications.

3. WINDOWS AUTHENTICATION - We use windows authentication when we are creating a web application for a limited number of users who will use their windows account for authentication.

This type of authentication is quite useful in an intranet environment.

Q101. WHAT IS OUTPUT CACHING IN MVC? HOW TO IMPLEMENT IT?

In normal caching a cache will be setup in the server which will used as a storage to store and transfer data which is requested frequently. And this will improve the performance.

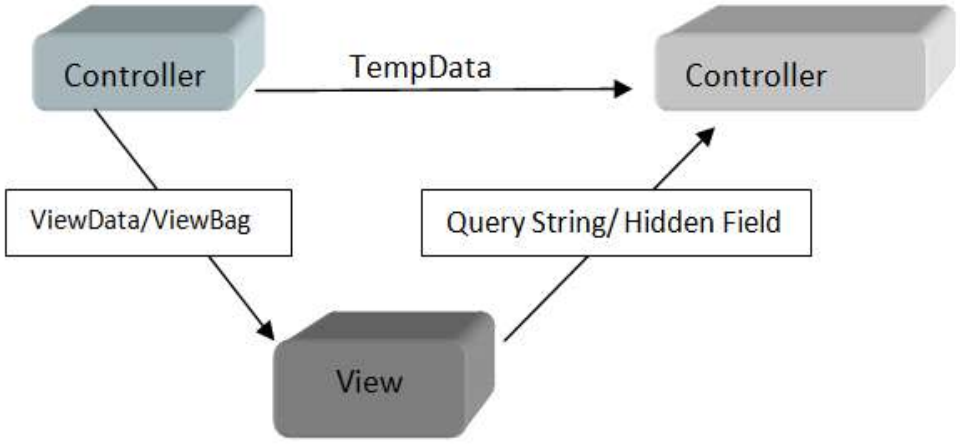


The output cache attribute cache the content returned by a controller action.

See the code here, Output cache will keep the data for 10 second, and then delete it if it is no more required. Because cache has to be light and fast.

```
[OutputCache(Duration=10)]
public ActionResult Index()
{
    ViewBag.Time = DateTime.Now.ToLongTimeString();

    return View();
}
```

Q102. WHAT IS THE DIFFERENCE BETWEEN VIEWDATA, VIEWBAG & TEMPDATA?



- ViewData & ViewBag are used to pass data from **CONTROLLER TO VIEW**.
- TempData is used to pass data from **CONTROLLER TO CONTROLLER**.

ViewData **REQUIRES TYPECASTING** for complex data types whereas ViewBag **DOESN'T REQUIRE TYPECASTING** for the complex data type.

Q103. HOW CAN WE PASS THE DATA FROM CONTROLLER TO VIEW IN MVC?

By using ViewData and ViewBag

Q104. WHAT IS PARTIAL VIEW?

Partial view in ASP.NET MVC is a special view which renders a portion of view content.

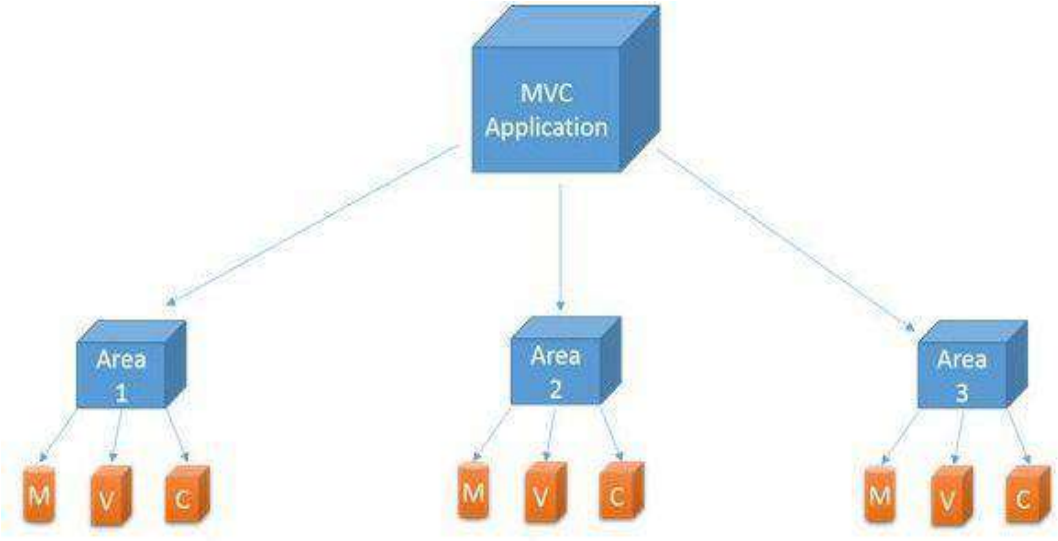
For example, menu can be a partial view for a view or page which can be reused in multiple pages or views. It helps us to reduce code duplication.

It is just like a user control of a web form application.

Q105. WHAT ARE AREAS IN MVC?

Areas are just a way to divide or "isolate" the modules of large applications in multiple or separated MVC projects.

So, if you have a very big MVC application then it can be further divided into small MVC applications with the help of Areas.



Q106. HOW VALIDATION WORKS IN MVC?

Validation can be done in MVC using **DATA ANNOTATION** Attributes.

```
public class Student
{
    public int StudentId { get; set; }

    [Required]
    public string StudentName { get; set; }

    [Range(10, 20)]
    public int Age { get; set; }
}
```

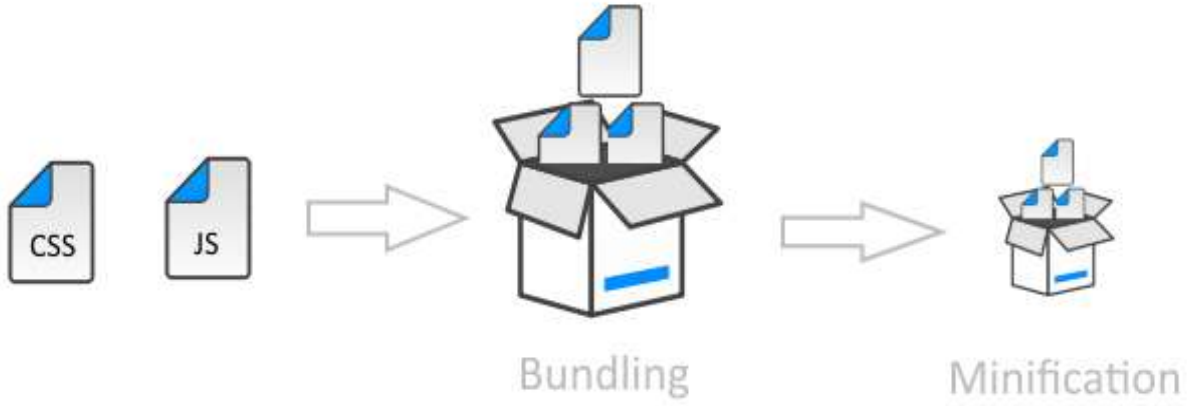
Q107. EXPLAIN THE CONCEPT OF MVC SCAFFOLDING?

Scaffolding is a code generation framework for ASP.NET Web applications.

Q108. WHAT IS BUNDLING AND MINIFICATION IN MVC?

- BUNDLING - It lets us combine multiple JavaScript (.js) files or multiple cascading style sheet (.css) files, so that they can be downloaded as a unit, rather than making individual HTTP requests.

- MINIFICATION - It squeezes out whitespace and performs other types of compression to make the downloaded files as small as possible.



Q109. HOW TO IMPLEMENT SECURITY IN WEB APPLICATIONS IN MVC?

These are the ways your web application security can be compromised and the way to prevent them:

(Not in detail but an overview)

1) Error Handling - Must Setup CUSTOM ERROR PAGE in your application

2) Cross-Site Request Forgery (CSRF) – Ensure use of AntiForgeryToken

3) Cross-Site Scripting (XSS) attacks – Do not allow <script> tag in the inputs of your application.

4) Malicious FILE Upload – Do not allow uploads where file extension is not known.

5) SQL Injection Attack - VALIDATE inputs, Use Parameterized queries, Use ORM (e.g. Dapper , Entity framework), Use Stored Procedures.

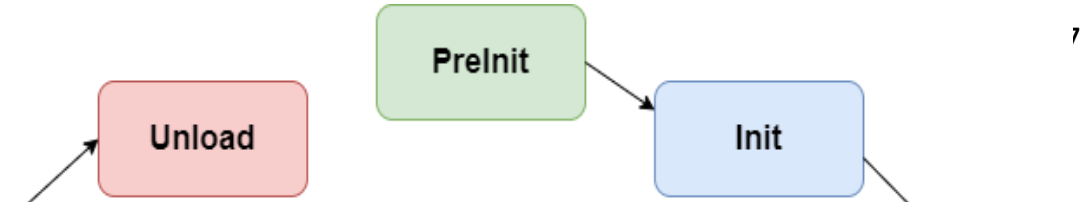
6) Use TOKEN based authentication in Web api and applications.

7) Save the password in ENCRYPTED form so that even developer can't access it.

8) Use HTTPS

Chapter 5 - ASP.NET WEBFORMS

Q110. WHAT ARE THE EVENTS IN PAGE LIFE CYCLE? IN WHICH EVENT ARE THE CONTROLS FULLY LOADED?



All events of the page life cycle. But the main ones are given below:

Init - This event fires after each control has been initialized.

Load – All the controls are ready at this event.

PreRender - Allows final changes to the page or its control.

Render - The Render method generates the client-side HTML

Unload - This event is used for cleanup code.

In PAGE LOAD event controls are fully loaded.

Q111. WHAT IS THE DIFFERENCE BETWEEN SERVER.TRANSFER() AND RESPONSE.REDIRECT()?

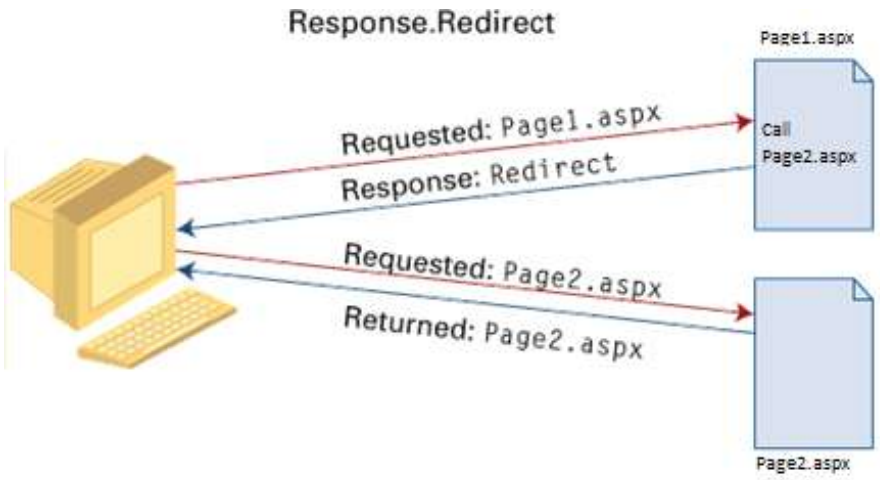
Both Response.Redirect() and Server.Transfer() methods are used to redirect the user's browser from one page to another page.

Response.Redirect uses round trip back to the client for redirecting the page.

It is a slow technique, but it maintains the URL history in the client browser for all pages.

Suppose a client will send a request for Page1.aspx to the server. See this Page1.aspx present on the server. Now inside this Page1.aspx after some logic it

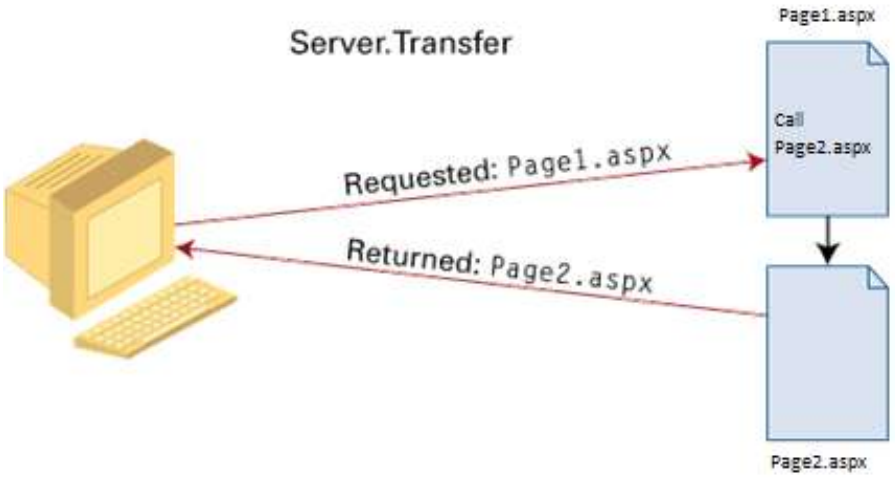
has to be redirected to the Page2.aspx on some condition. So if you are using the response.redirect, then it will send the response back to the client and then it will again request the new page from the server. So, this is what we also call round trip, when you are going from position A to B and then you have to go to position C, for that you are coming back to position A and then going to position C, that is round trip which is not good and slow.



In **Server.Transfer** page processing transfers from one page to the other page WITHOUT MAKING A ROUND-TRIP BACK to the client's browser.

It is a faster technique, but it **does not** maintain the URL history in the client browser for all pages.

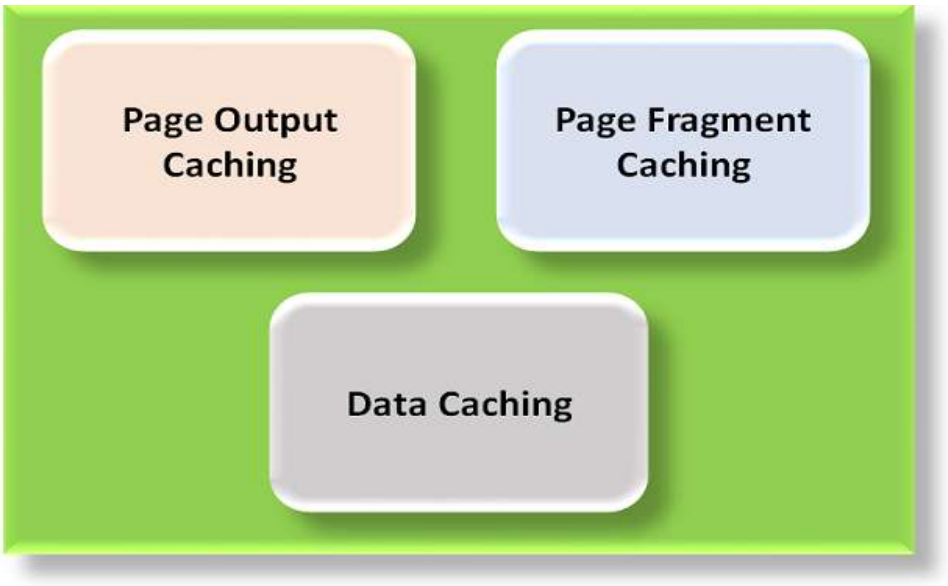
In case of Server.Transfer transfer from Page1 to 2 will be direct without going back to the client like this. So it does not require round trip and hence very fast.



Q112. WHERE THE VIEWSTATE IS STORED AFTER THE PAGE POSTBACK?

ViewState is stored in a HIDDEN FIELD on the page at client side.

Q113. WHAT ARE THE DIFFERENT TYPES OF CACHING?



Q114. WHAT ARE THE DIFFERENT SESSION STATE MANAGEMENT OPTIONS AVAILABLE IN ASP.NET?

Session can be stored in two ways in ASP.NET:

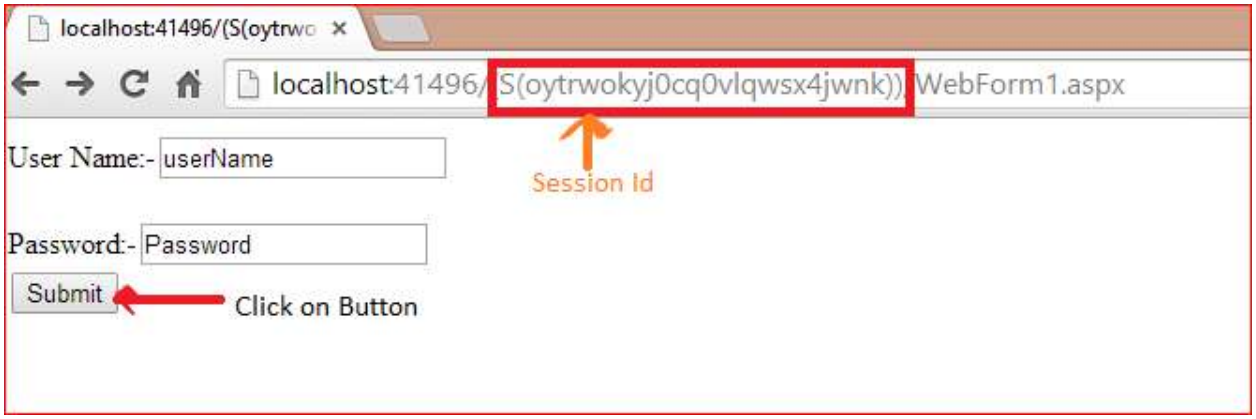
1. In-Process - Stores the session in memory on the same **web server**.

2. Out-of-Process - Stores the seesion data in an **external server**. The external server may be either a SQL Server or a State Server.

Q115. WHAT IS COOKIE LESS SESSION?

By default, a session uses a browser cookie in the background.

In cookie less the session is passed via **URL** INSTEAD OF COOKIE.



To enable a cookie-less session, we need to change some configuration in the WEB.CONFIG file.

Q116. HOW TO FORCE ALL THE VALIDATION CONTROLS TO RUN IN A PAGE IN WEB FORMS?

PAGE.VALIDATE()

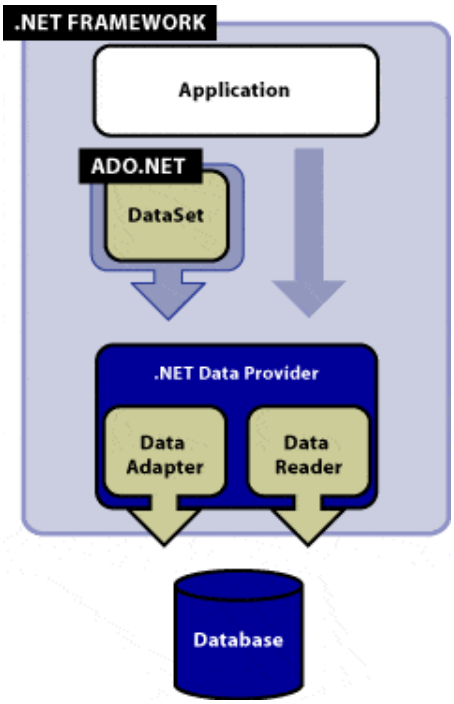
Chapter 6 - ADO.NET

Q117. WHAT ARE THE MAIN COMPONENTS OF ADO.NET?

- **DataSet class** - A DataSet is basically a container which gets the data from one or more then one tables from the database. It follows disconnected architecture.

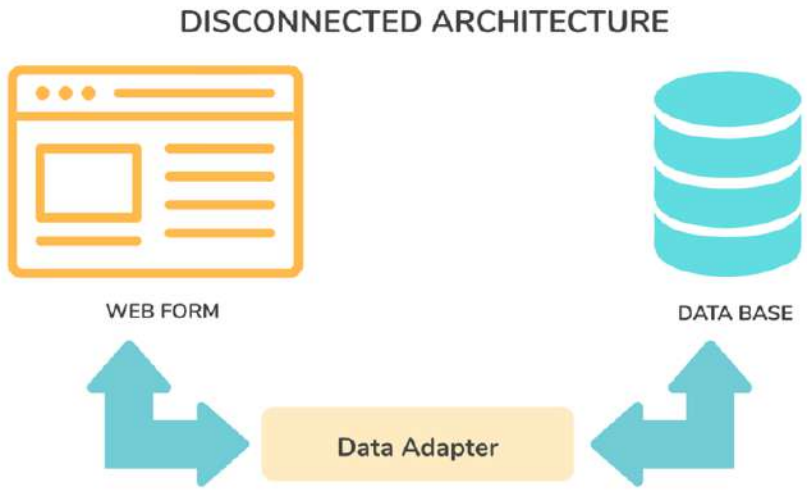
- **DataReader Class -** The DataReader allows you to read the data returned by a SELECT command. It is read only. Unlike dataset we cannot update the database via this. It follows connected architecture.

- **DataAdapter class** - A DataAdapter bridges the gap between the disconnected DataSet/ DataTable objects and the physical database.



Q118. WHAT IS CONNECTED ARCHITECTURE AND DISCONNECTED ARCHITECTURE?

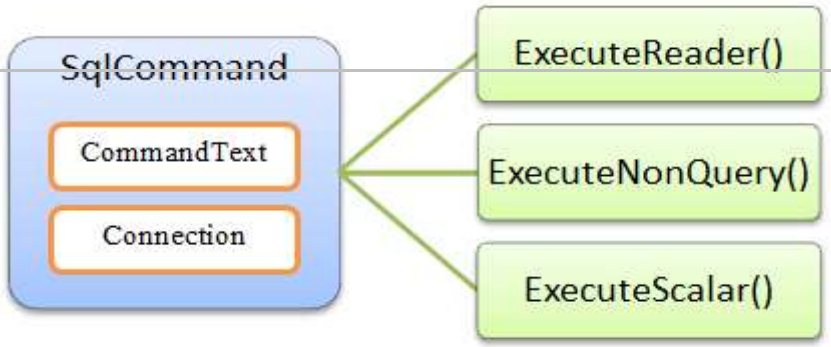
- **Disconnected Architecture -** Disconnected architecture means, you don't need to connect always to get data from the database. You can get data into DataAdapter, disconnect the database, manipulate the DataAdapter and resubmit the data. It is fast and robust(data will not lose in case of any power failure).



- **Connected Architecture -** Connected architecture means you are directly interacting with database but it is less secure and not robust.

Q119. WHAT ARE THE DIFFERENT EXECUTE METHODS OF ADO.NET?

- **EXECUTESCALAR()** - Returns SINGLE value from the dataset
- **EXECUTENONQUERY()** - Returns a RESULTSET from dataset and it has multiple values. It can be used for insert, update and delete.
- **EXECUTEREADER()** - It only retrieves data from database. No update, insert. It is readonly.



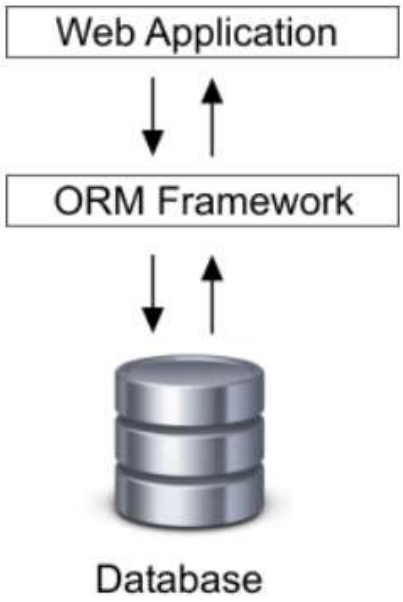
Q120. WHAT ARE THE AUTHENTICATION TECHNIQUES USED TO CONNECT TO SQL SERVER?

- **WINDOWS AUTHENTICATION MODE -** Use authentication using Windows domain accounts only.
- **SQL SERVER AUTHENTICATION MODE** - Authentication provided with the combination of both Windows and SQL Server Authentication.

Chapter 7 – ENTITY FRAMEWORK

Q121. WHAT IS ORM? WHAT ARE THE DIFFERENT TYPES OF ORM?

ORM (Object-Relational Mapper) is for mapping objects in your application with database tables. It is like a wrapper to make database calls simple and easy.



Types of ORM can be used with .NET:

- ENTITY FRAMEWORK 6.X
- ENTITY FRAMEWORK CORE
- N HIBERNATE
- DAPPER

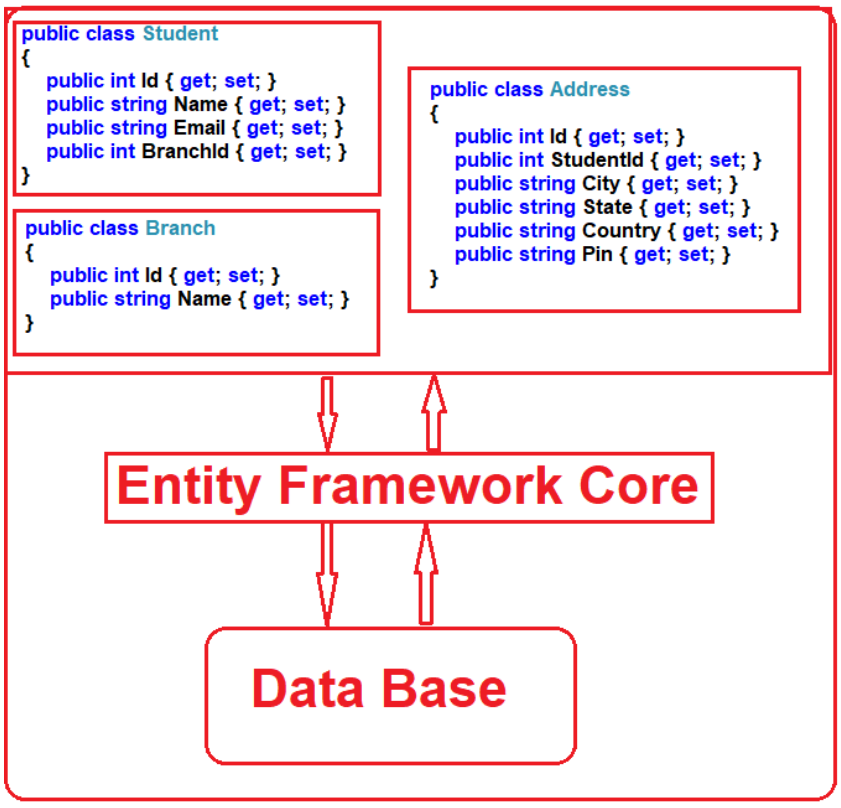
Q122. WHAT IS ENTITY FRAMEWORK?

An Entity Framework (EF) is an open-source **ORM (Object-Relational Mapper)** from Microsoft.

It's like a **WRAPPER** on ADO.NET.

Entity Framework minimizes the coding effort.

For example, here you can see Student clas, Address class and Branch classes.
These classes will be directly matched with database tables.
The properties will be table columns.



Q123. HOW WILL YOU DIFFERENTIATE ADO.NET FROM ENTITY FRAMEWORK?

See the below code. Both EF and ADO.NET are doing same thing but see the number of lines are less in EF and it is simple also.

ADO.NET

```
public class UserRepository
 {
    public DataSet GetAllUsersList()
    {
        DataSet ds = new DataSet(); // Initializes disconnected dataset to fetch results

         //Create and initialize the connection using connection string
        using (SqlConnection sqlConn = SqlConnection ("DataSource=localhost;
                            Initial Catalog=TestDB; User ID=sa;Password=admin;"))
         {

              sqlConn.Open(); //Open connection

              string sql = "select * from tblusers"; // sql query to access user from table
              SqlCommand sqlCmd = new SqlCommand(sql, sqlConn);

              sqlCmd.CommandType = CommandType.Text; // Define Command Type

              SqlDataAdapter sqlAdapter = new SqlDataAdapter(sqlCmd); // Execute command

              sqlAdapter.Fill(ds); //Get the data in disconnected mode
         }

        return ds; // return dataset result
    }
 }
```

EF

```
public class UserRepository
  {
    public static void Main(string[] args)
    {
      // Initializes the dbContext class User Entity
      using (var userContextDB = new UserContext())
      {
        // Create a new user object
        var user = new User() { UserID = "USER-01" };

        // Call Add Command
        userContextDB.User.Add(user);

        // Execute the save command to make changes
        userContextDB.SaveChanges();
      }
    }
  }
```

ADO.NET	Entity Framework
ADO.NET does create codes for the data access layers, intermediate layers, and mapping codes by itself.	EF **automatically** creates codes for the data access layers, intermediate layers, and mapping codes.
ADO.NET is slightly faster.	Entity Framework is comparatively slower.

Q124. HOW ENTITY FRAMEWORK WORKS? OR HOW TO SETUP EF?

Below are the steps to add records in Student table in database using EF:

1. Install-Package EntityFramework

2. Create the data model for the student entity

In the *Models* folder, create a class file named *Student.cs*

```
public class Student   {
      public int ID { get; set; }
      public string LastName { get; set; }
      public string FirstMidName { get; set; }
   }
```

3. Create the database context class – SchoolContext.cs

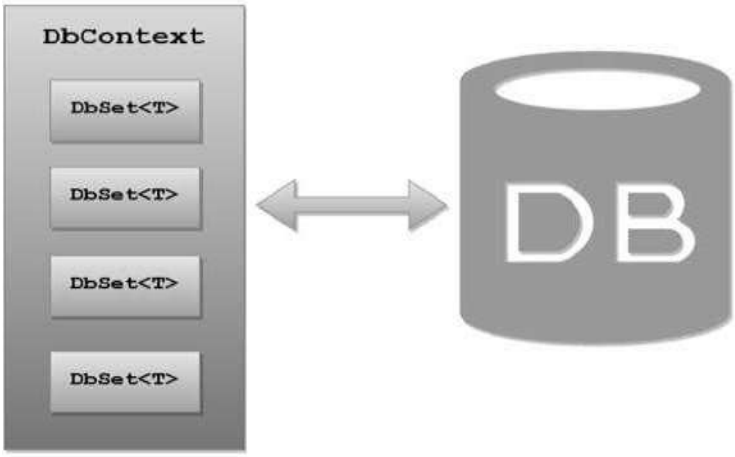
```
public class SchoolContext : DbContext
  {
    public SchoolContext() : base("SchoolContext")
    {
    }
    public DbSet<Student> Students { get; set; }
    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
       modelBuilder.Conventions.Remove<PluralizingTableNameConvention>();
    }
  }
```

3. Finally add data to student table

```
public class SchoolInitializer : System.Data.Entity.
DropCreateDatabaseIfModelChanges<SchoolContext>
{
      protected override void Seed(SchoolContext context)
      {
         var students = new List<Student>
         {
         new Student{FirstMidName="Carson",LastName="Alexander},
         new Student{FirstMidName="Meredith",LastName="Alonso},
         new Student{FirstMidName="Arturo",LastName="Anand},
         };
         students.ForEach(s => context.Students.Add(s));
         context.SaveChanges();
    }
}
```

Q125. WHAT IS MEANT BY DBCONTEXT AND DBSET?

- DbContext is a class in the Entity Framework that helps create a **COMMUNICATION** between the database and the domain/entity class.



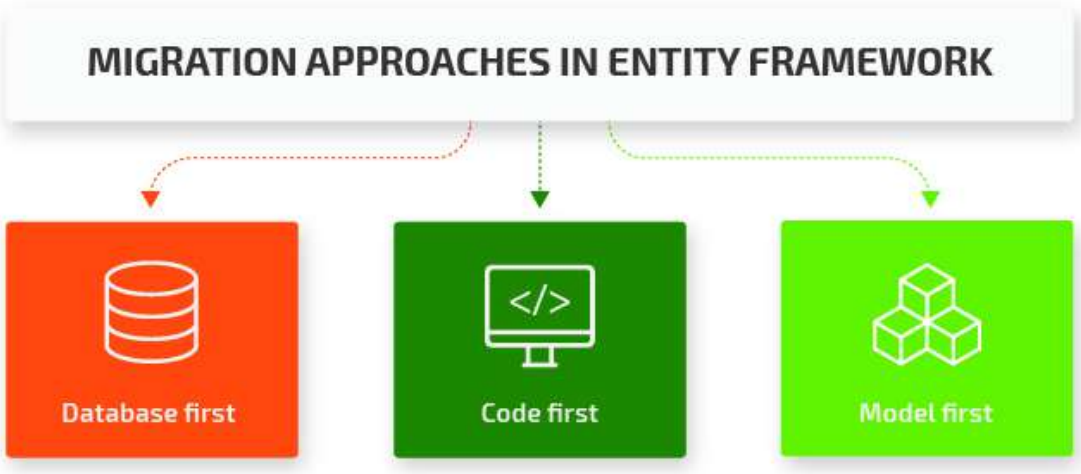
- The DbSet class represents an **entity set** that can be used for create, read, update, and delete operations. You can relate DBSet with the Table in database.

```
public partial class SchoolDBEntities : DbContext
{
    public SchoolDBEntities()
        : base("name=SchoolDBEntities")
    {
    }

    protected override void OnModelCreating(DbModelBuilder modelBuilder)
    {
        throw new UnintentionalCodeFirstException();
    }

    public virtual DbSet<Course> Courses { get; set; }
    public virtual DbSet<Standard> Standards { get; set; }
    public virtual DbSet<Student> Students { get; set; }
    public virtual DbSet<StudentAddress> StudentAddresses { get; set; }
    public virtual DbSet<Teacher> Teachers { get; set; }
```

Fluent API

Entity set

Q126. WHAT ARE THE DIFFERENT TYPES OF APPROACHES USED IN ENTITY FRAMEWORK?



- **Database First -** The Database First approach enables us to create an entity model from the **existing database**.

- **Code First -** The Code First approach enables us to **create a model** and their relation using classes and then create the database from these classes.

- **Model First -** In this approach, **model classes** and their relation is created first using the ORM designer and the physical database will be generated using this model.

Q127. WHAT IS THE DIFFERENCE BETWEEN LINQ TO SQL AND ENTITY FRAMEWORK?

1. LINQ to SQL allow you to query and modify SQL Server database by using LINQ syntax.

Entity framework is a ORM by Microsoft which allow you to query and modify RDBMS like SQL Server, Oracle, DB2 and MySQL etc. by using LINQ syntax.

2. LINQ to SQL only works with SQL Server

EF work with SQL Server, Oracle, DB2 etc.

3. LINQ to SQL will generate the DBML file

EF generates the .EDMX file (Entity Data Model Extension)

4. LINQ to SQL is old and slow.

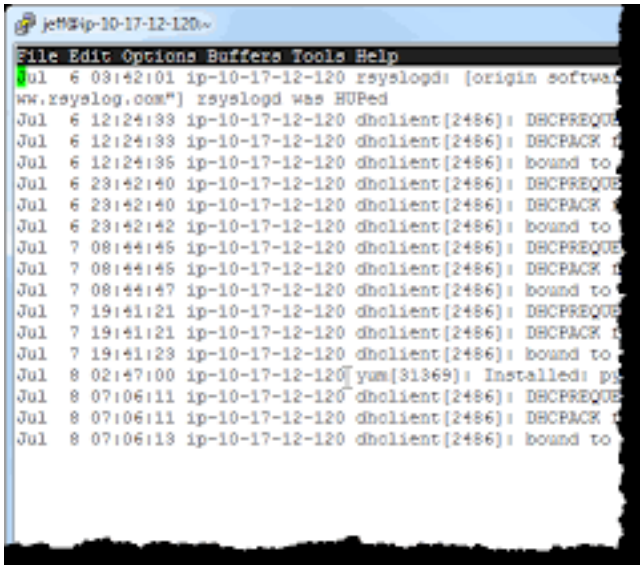
EF is mostly used, new and fast.

Chapter 8 - DESIGN PATTERNS & SOLID PRINCIPLES

Q128. WHAT ARE DESIGN PATTERNS AND WHAT PROBLEM THEY SOLVE?

Design patterns are general **REUSABLE SOLUTIONS** for common problems in **software design**.

What is reusable solutions? Suppose you want to implement logging for your application. Now there are existing good logging techniques already available which you can use and they will be reusable solution for you.



Now you will get many reusable solutions in stackoverflow and github. But are they design patterns?

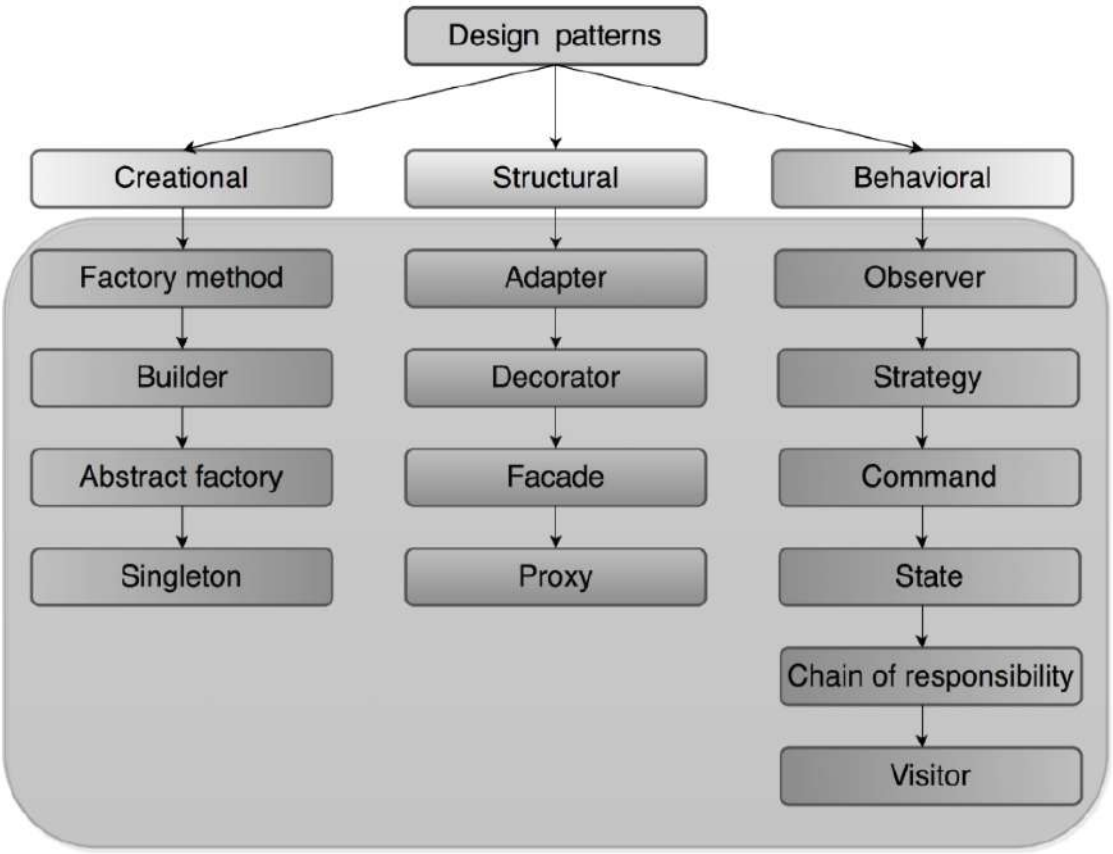
The answer is no. The reusable solutions which solves problem related to designing of the software, only those are design patterns.

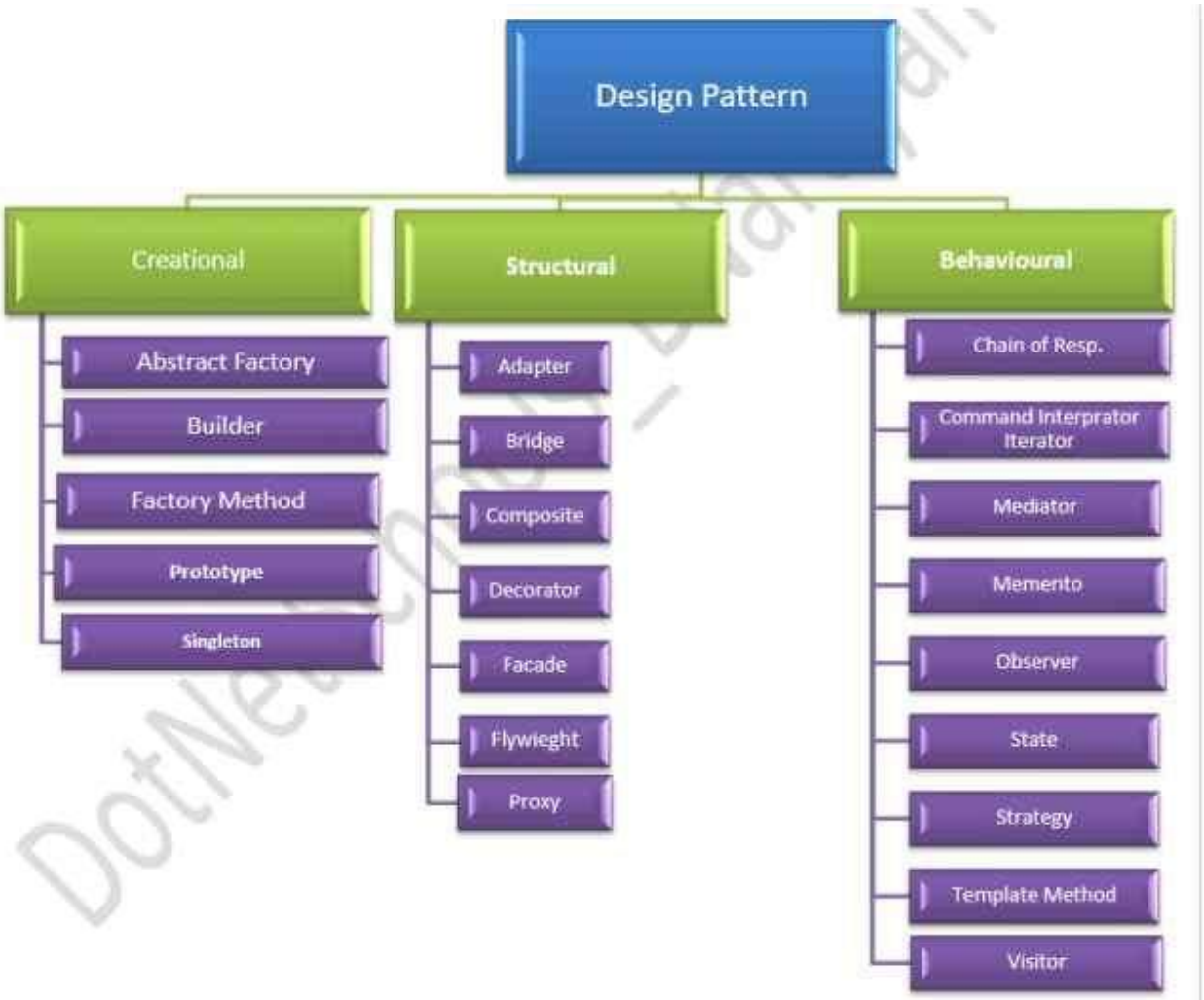
For example, while creating an application from scratch, when you design the classes then how the classes will communicate with each other. Then you can take the help of design patterns.

Q129. WHAT ARE THE TYPES OF DESIGN PATTERNS?

There are three categories of design patterns:

- Creational design patterns
- Structural design patterns
- Behavioral design patterns



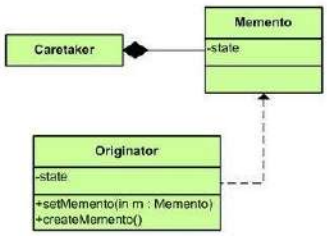

Design Pattern
Creational
Structural
Behavioural
Abstract Factory
Builder
Factory Method
Prototype
Singleton
Adapter
Bridge
Composite
Decorator
Facade
Flywieght
Proxy
Chain of Resp.
Command Interprator Iterator
Mediator
Memento
Observer
State
Strategy
Template Method
Visitor

C Abstract Factory
S Adapter
S Bridge
C Builder
B Chain of Responsibility
B Command
S Composite
S Decorator
S Facade
C Factory Method
S Flyweight
B Interpreter
B Iterator
B Mediator
B Memento
C Prototype
S Proxy
B Observer
C Singleton
B State
B Strategy
B Template Method
B Visitor

Memento

Type: Behavioral

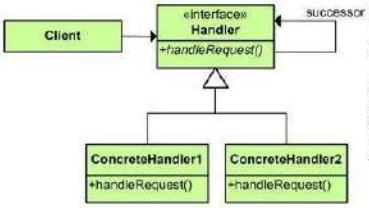
What it is:
Without violating encapsulation, capture and externalize an object's internal state so that the object can be restored to this state later.



Chain of Responsibility

Type: Behavioral

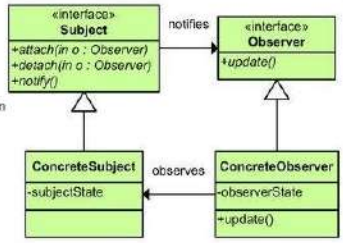
What it is:
Avoid coupling the sender of a request to its receiver by giving more than one object a chance to handle the request. Chain the receiving objects and pass the request along the chain until an object handles it.



Observer

Type: Behavioral

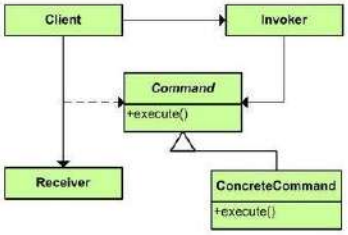
What it is:
Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.



Command

Type: Behavioral

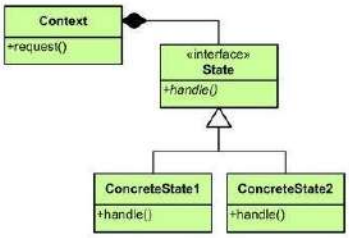
What it is:
Encapsulate a request as an object, thereby letting you parameterize clients with different requests, queue or log requests, and support undoable operations.



State

Type: Behavioral

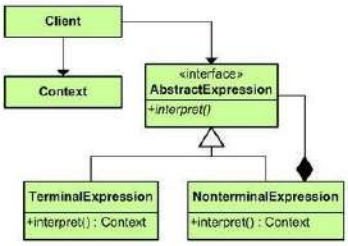
What it is:
Allow an object to alter its behavior when its internal state changes. The object will appear to change its class.



Interpreter

Type: Behavioral

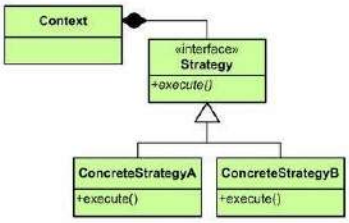
What it is:
Given a language, define a representation for its grammar along with an interpreter that uses the representation to interpret sentences in the language.



Strategy

Type: Behavioral

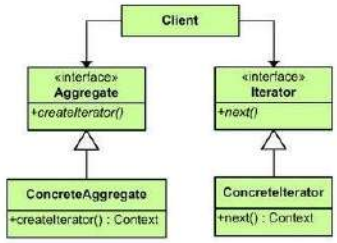
What it is:
Define a family of algorithms, encapsulate each one, and make them interchangeable. Lets the algorithm vary independently from clients that use it.



Iterator

Type: Behavioral

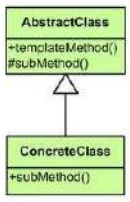
What it is:
Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.



Template Method

Type: Behavioral

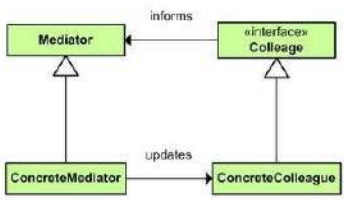
What it is:
Define the skeleton of an algorithm in an operation, deferring some steps to subclasses. Lets subclasses redefine certain steps of an algorithm without changing the algorithm's structure.



Mediator

Type: Behavioral

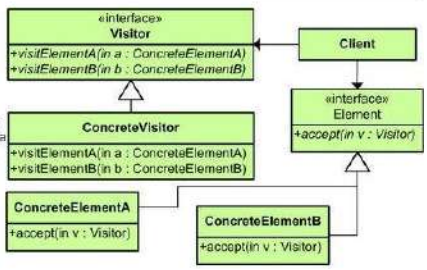
What it is:
Define an object that encapsulates how a set of objects interact. Promotes loose coupling by keeping objects from referring to each other explicitly and it lets you vary their interactions independently.



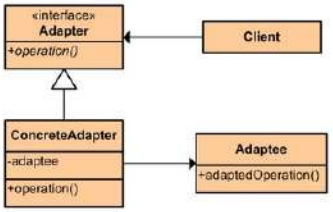
Visitor

Type: Behavioral

What it is:
Represent an operation to be performed on the elements of an object structure. Lets you define a new operation without changing the classes of the elements on which it operates.



Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, Massachusetts: Addison Wesley Longman, Inc.



Adapter

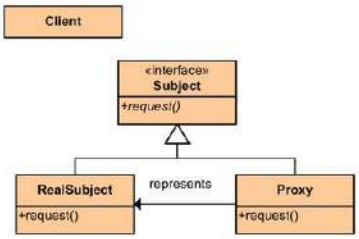
Type: Structural

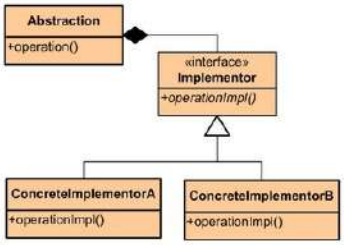
What it is:
Convert the interface of a class into another interface clients expect. Lets classes work together that couldn't otherwise because of incompatible interfaces.

Proxy

Type: Structural

What it is:
Provide a surrogate or placeholder for another object to control access to it.





Bridge

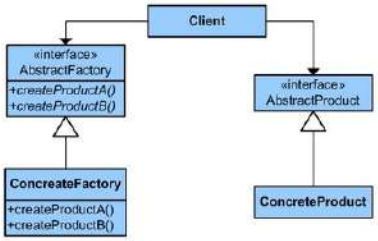
Type: Structural

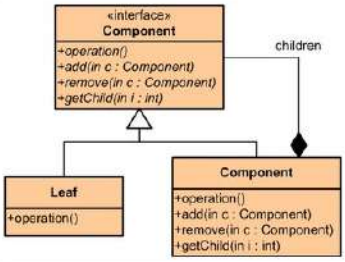
What it is:
Decouple an abstraction from its implementation so that the two can vary independently.

Abstract Factory

Type: Creational

What it is:
Provides an interface for creating families of related or dependent objects without specifying their concrete class.





Composite

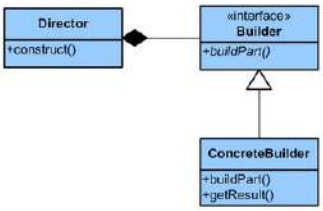
Type: Structural

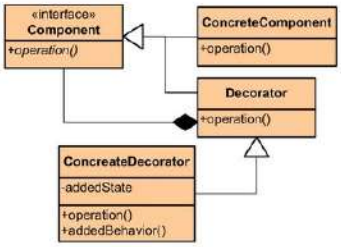
What it is:
Compose objects into tree structures to represent part-whole hierarchies. Lets clients treat individual objects and compositions of objects uniformly.

Builder

Type: Creational

What it is:
Separate the construction of a complex object from its representing so that the same construction process can create different representations.





Decorator

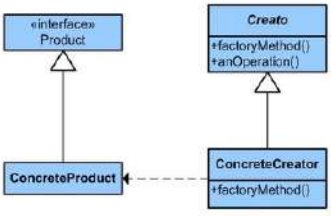
Type: Structural

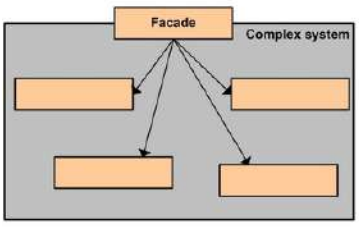
What it is:
Attach additional responsibilities to an object dynamically. Provide a flexible alternative to sub-classing for extending functionality.

Factory Method

Type: Creational

What it is:
Define an interface for creating an object, but let subclasses decide which class to instantiate. Lets a class defer instantiation to subclasses.





Facade

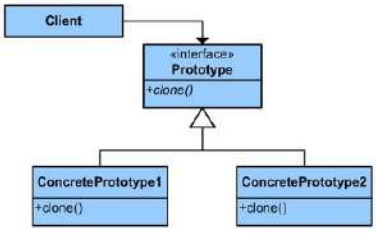
Type: Structural

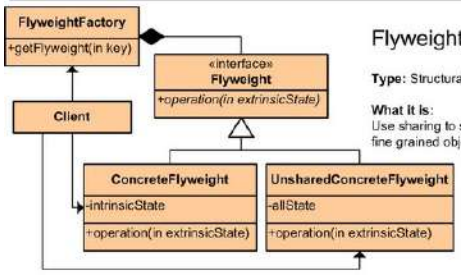
What it is:
Provide a unified interface to a set of interfaces in a subsystem. Defines a high-level interface that makes the subsystem easier to use.

Prototype

Type: Creational

What it is:
Specify the kinds of objects to create using a prototypical instance, and create new objects by copying this prototype.





Flyweight

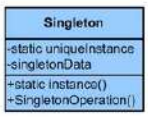
Type: Structural

What it is:
Use sharing to support large numbers of fine grained objects efficiently.

Singleton

Type: Creational

What it is:
Ensure a class only has one instance and provide a global point of access to it.



Gamma, Erich; Helm, Richard; Johnson, Ralph; Vlissides, John (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, Massachusetts: Addison Wesley Longman, Inc.

Q130. WHAT ARE CREATIONAL DESIGN PATTERNS?

Creational design patterns are design patterns that deal with **object creation** mechanisms.

For example, Singleton, Factory Method, Abstract Factory and builder are Creational design patterns.

Q131. WHAT ARE STRUCTURAL DESIGN PATTERNS?

Structural Design Pattern is used to manage the structure of classes and interfaces as well to manage the **relationship** between the classes.

For example, Adapter, Decorator, Façade and Proxy are the example of Structural Design Patterns.

Q132. WHAT ARE BEHAVIORAL DESIGN PATTERNS?

Behavioral design patterns defines the way to **communicate** between classes and object.

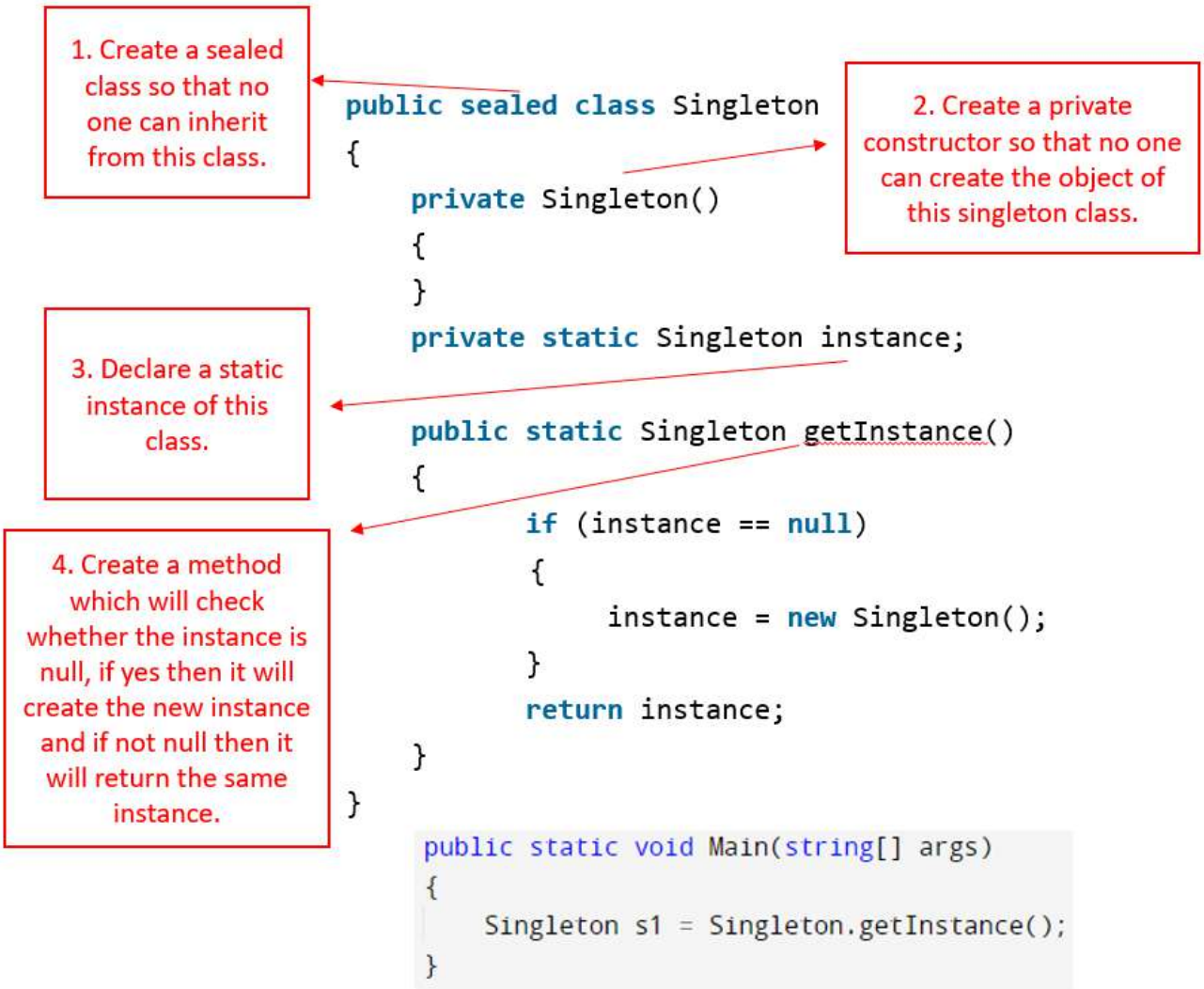
For example, Observer, Strategy, Command, Stage and Visitor are the types of Behavioral Design Patterns.

Q133. WHAT IS SINGLETON DESIGN PATTERN?

It is a creational design pattern because it deals with object creation mechanism.

A singleton is a class that only allows a **SINGLE INSTANCE** of itself to be created.

Below are the steps for creating a singleton class:



Q134. HOW TO MAKE SINGLETON PATTERN THREAD SAFE?

Thread Safe Singleton Pattern can be implemented by using the **LOCK** mechanism. So that multiple threads cannot create multiple instances of the same class.

```
public sealed class Singleton
{
    private Singleton() {}
    private static readonly object lock = new object();
    private static Singleton instance = null;

    public static Singleton getInstance()
    {
          private lock(lock) //lock for multi threads
         {
              if (instance == null)
              {
                   instance = new Singleton();
              }
            return instance;
         }
    }
}
```

Q135. WHAT IS FACTORY PATTERN? WHY TO USE FACTORY PATTERN?

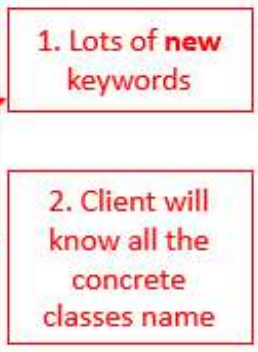
Factory method Or Factory is a creational design pattern which manages object creation. In this pattern, an interface is used for creating an object, but let subclass decide which class to instantiate.

Why to use Factory Pattern? Factory pattern will solve two problems:

1. It will manage the **new** keywords and avoid writing too many new keywords.

2. Second is it will not let the client class know, what is the name of concrete class. It is good for security.

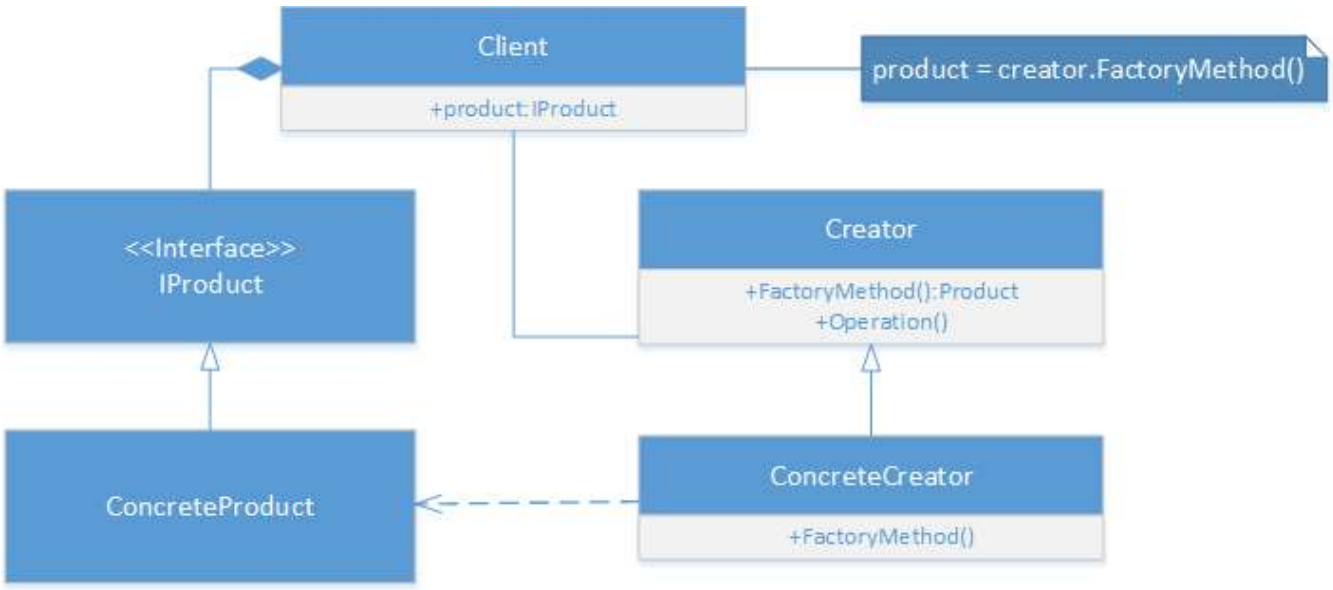
```
public class SoftwareCompanyClient
{
    public static void Main(string[] args)
    {
        string technology = "DotNetJobs";
        string skill;

        if(technology == "DotNetJobs")
        {
            DotNet objDotNet = new DotNet();

            skill = objDotNet.GetSkill();
        }
        else if(technology == "JavaJobs")
        {
            Java objJava = new Java();

            skill = objJava.GetSkill();
        }
    }
}
```



Q136. HOW TO IMPLEMENT FACTORY METHOD PATTERN?

This is the short answer.

In factory pattern, we create all new objects in a common and separate class with the help of Creator, ConcreteCreator and ConcreteProduct class.

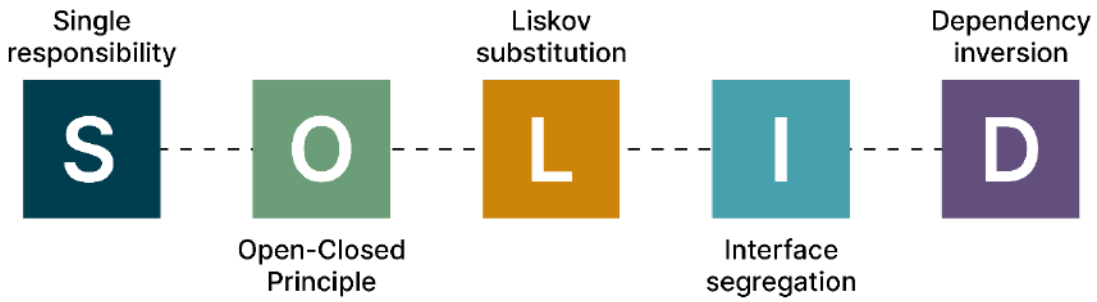


For detail answer see the video lecture.

Q137. WHAT ARE SOLID PRINCIPLES? WHAT IS THE DIFFERENCE BETWEEN SOLID PRINCIPLES AND DESIGN PATTERNS?

SOLID is an acronym that stands for five key design principles:

1. Single responsibility principle
2. Open-closed principle
3. Liskov substitution principle
4. Interface segregation principle
5. Dependency inversion principle.



Difference between SOLID principles and Design Patterns?

SOLID principles aren't concrete - rather abstract. Meaning there can be multiple ways to implement the SOLID principles.

Design patterns are concrete and solve a particular kind of problem in a particular way.

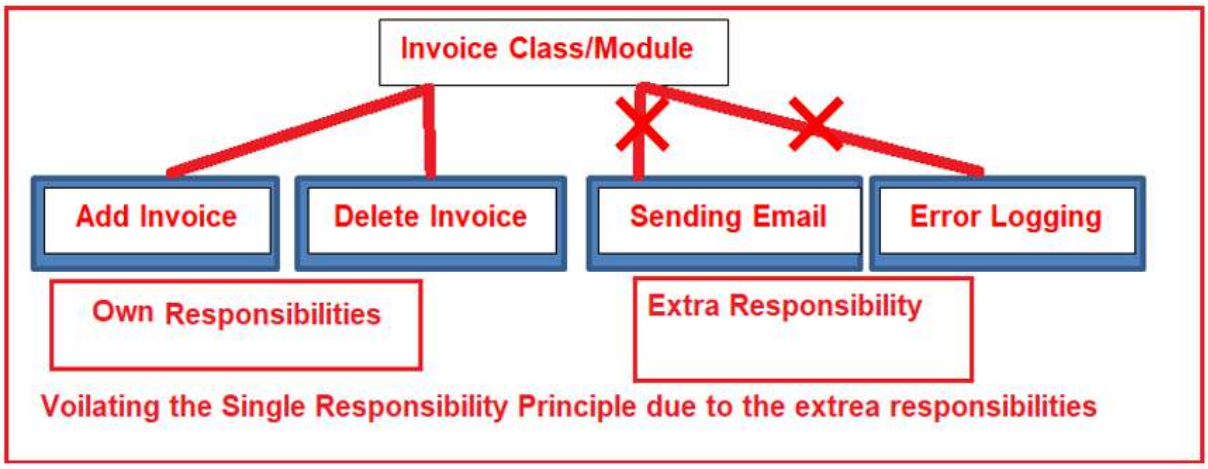
Q138. WHAT IS SINGLE RESPONSIBILITY PRINCIPLE?

- SRP states that, each software module or class should have only one reason to change. Or we can say that each class should have only **one responsibility** to do.

For example, in the below diagram, there is an invoice class which have functionalities like add the invoice and delete the invoice.

Now if this class will also have functionalities of sending invoicing emails or error logging then they are like additional or extra responsibilities. And it is the violation of SRP.

In order to follow SRP, sending email and error logging must not be the part of Invoice class and there should be separate classes for them.



Q139. WHAT IS OPEN-CLOSED PRINCIPLE?

Open-Closed principle states that software entities should be open for **EXTENSION**, but closed for **MODIFICATION**.

In other words, in order to add some functionality in a class don't modify it but extend it using **inheritance**.

For example, suppose we have a SavingAccount class, which has a CalculateInterest() method which calculate interest based on the account type. Right now, we have two account types "Regular" and "Salary", until now it's fine. But suppose tomorrow we get a new account type and that is "Business". What you will do? You will add a new else if condition at the end. Right?

So basically, Inserting one more else if{} will violate the open closed principle because you are modifying the class rather than extending it.

```
Public class SavingAccount
{
   Public decimal CalculateInterest(AccountType accountType)
   {
        If(AccountType=="Regular")
        {
            Interest = balance * 0.4;
        }
        else if(AccountType=="Salary")
        {
            Interest = balance * 0.5;
        }
   }
}
```

Inserting one more else if{} will violate the open closed principle because you are modifying the class rather than extending it.

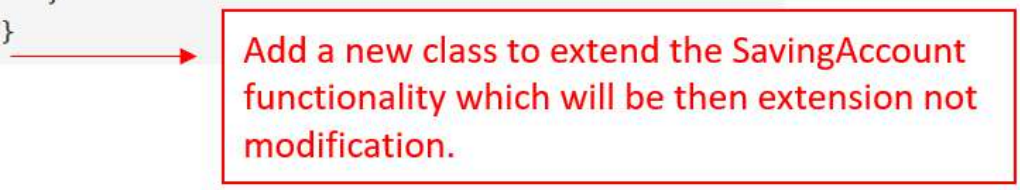
So how to follow Open Closed Principle?

Inheritance is the way of extending the functionality of an existing class.

For that you have to change your design of your application.
First create an interface ISavingAccount in which just declare CalculateInterest() method. After that, create a class for RegularSavingAccount and then inherit it from ISavingAccount class. Now implement CalculateInterest() method inside it. Similarly create a separate class for SalarySvaingAccount and then add a new class for BussinessSavingAccount to extend the SavingAccount functionality which will be then extension not modification.

```
Interface ISavingAccount
{
   decimal CalculateInterest();
}
```

```
Public Class RegularSavingAccount : ISavingAccount
{
   Public decimal CalculateInterest()
   {
      Interest = balance * 0.4;
   }
}

Public Class SalarySavingAccount : ISavingAccount
{
   Public decimal CalculateInterest()
   {
       Interest = balance * 0.5;
   }
}
```



See if you want to add any account then you need not to modify the class but you can add a new class and achieve that.

Q140. WHAT IS LISKOV SUBSTITUTION PRINCIPLE?

LSP states that, Objects of a Superclass shall be replaceable with objects of its Subclasses without **breaking** the application.

Superclass/ Base class/ Parent class are same thing. Similarly, Subclass/ Derived class/ Child class are same thing.

For example, Suppose you have a superclass Employee which have two virtual methods CalculateSalary and CalculateBonus.

```
public class Employee //Superclass or base class
{
    public virtual string CalculateSalary()
    {
        return "50000";
    }
    public virtual string CalculateBonus()
    {
        return "25000";
    }
}
```

Then you have two subclasses from it. One is PermanentEmployee and other is ContractualEmployee. For PermanentEmployee class both methods are good for it. But for contractual employee CalculateBonus is not applicable. So, if someone try to access this CalculateBonus method from ContractualEmployee class then you have to show some error to it.

That is violation of LSP because the application is breaking here.

```
public class PermamentEmployee: Employee //Subclass or derived class
{

}
public class ConntractualEmployee: Employee //Subclass or derived class
{

    public override string CalculateBonus()
    {
        throw new NotImplementedException(); // Not Applicable LSP
    }
}
```

So in order to resolve this or to follow LSP, make sure that base classes have only those methods which must be implemented in the derived classes otherwise it will violate LSP.

Q141. WHAT IS INTERFACE SEGREGATION PRINCIPLE?

ISP states that, do not create big interfaces with many methods declarations and then force them on deriving classes.

For example creating a single interface IVehicle for both Flyingcar and normal car is not good. Because normal car will not implement Fly() method.
Below code is violating ISP:

```
public interface IVehicle
{
    void Drive();
    void Fly();
}
```

```
public class Flyingcar : IVehicle
{
    public void Drive()
    {
        Console.WriteLine("Drive a Flyingcar");
    }

    public void Fly()
    {
        Console.WriteLine("Fly a Flyingcar");
    }
}
```

```
public class Car : IVehicle
{
    public void Drive()
    {
        //actions to drive a car
        Console.WriteLine("Driving a car");
    }

    public void Fly()
    {
        throw new NotImplementedException();
    }
}
```

Rather create separate interfaces IDrive and IFly and use them appropriately.
Below code is following ISP

```
public interface IDrive
{
    void Drive();
}

public interface IFly
{
    void Fly();
}
```

```
public class Car : IDrive
{
    public void Drive()
    {
        Console.WriteLine("Driving a car");
    }
}
```

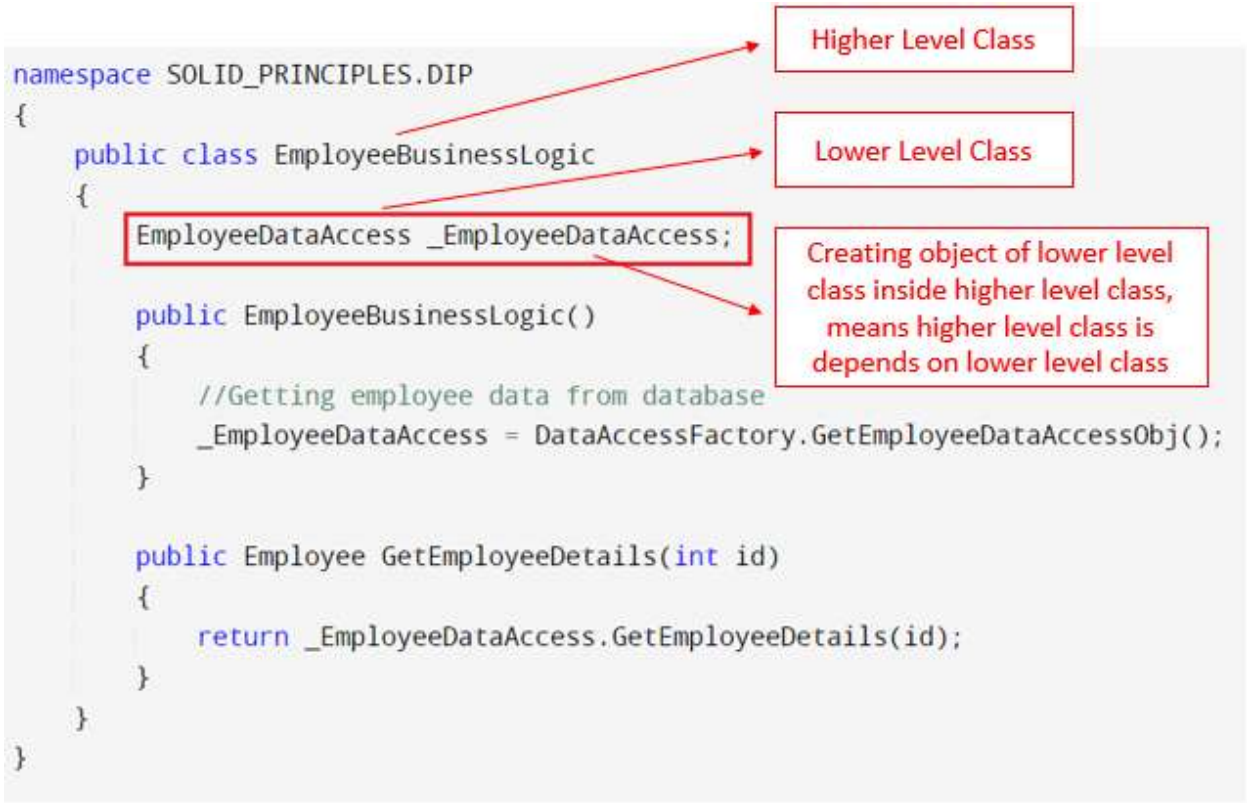
```
public class Flyingcar : IDrive, IFly
{
    public void Drive()
    {
        Console.WriteLine("Drive a flying car");
    }
    public void Fly()
    {
```

Q142. WHAT IS DEPENDENCY INVERSION PRINCIPLE?

The Dependency Inversion Principle (DIP) states that a **high-level class** must not depend upon a **lower level class**. They must both depend upon abstractions.

And, secondly, an abstraction must not depend upon details, but the details must depend upon abstractions.

What is higher level class and what lower level class?
In below code, EmployeeBusinessLogic is higher level class and it is dependent on this EmployeeDataAccess class which is a lower level class.



One way to resolve this is to use **Dependency injection.** Where the lower-level class will be injected in the higher level class.

Q143. WHAT IS DRY PRINCIPLE?

DRY stands for DON'T REPEAT YOURSELF.

Don't write the same functionality multiple time.

Rather use write it once and then use it at multiple places using inheritance.

Chapter 9 – WEB API

Q144. WHAT IS WEB API? WHAT IS THE PURPOSE OF WEB API?

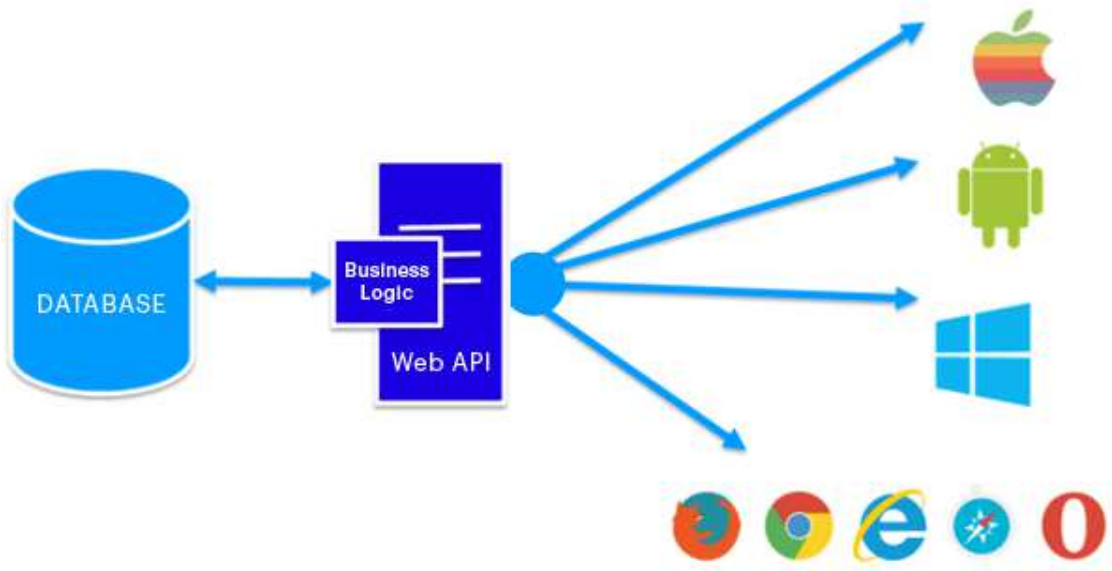
Web API(Application Programming Interface) provides interaction BETWEEN software applications.

For example, if you want to place an order on Amazon via mobile app or website. The UI is different but for both mobile app and website, business logic is written in

a common web API. So that if there is any logic change then it can be done at only one place which is then easy to maintain

.

See the diagram, whether its' an Ipohone app, android app, windows app, or any browser. There UI's are different but there business logic is at some common place which is Web API.

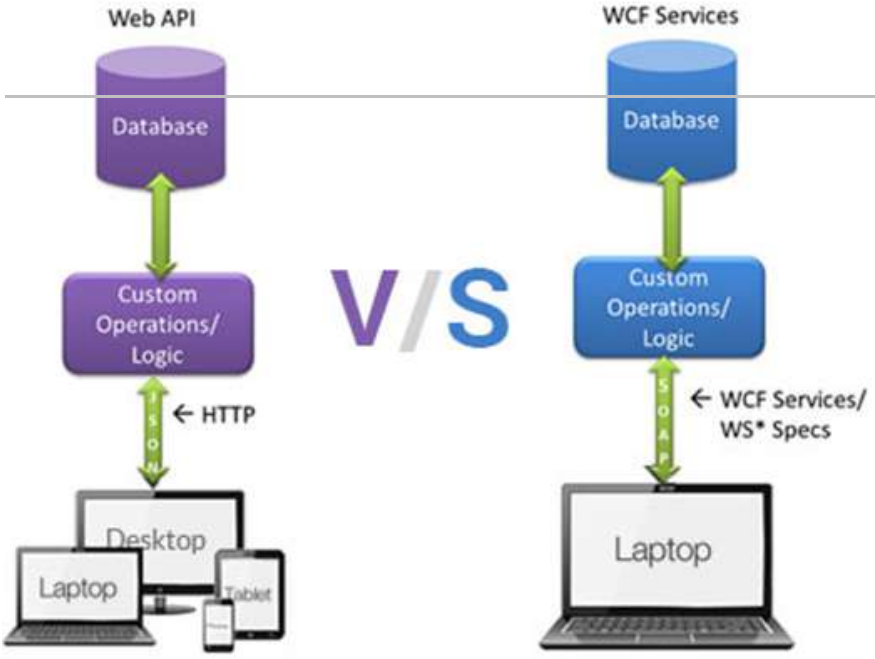


Q145. WHAT ARE WEB API ADVANTAGES OVER WCF AND WEB SERVICES?

Below is the list of Web API advantages:

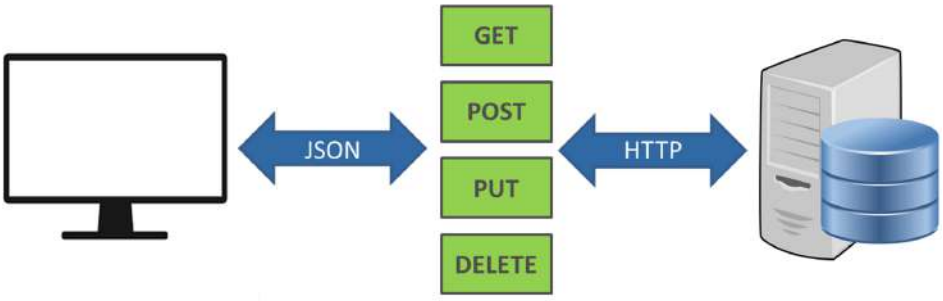
- Web API works the way **HTTP** works using standard HTTP verbs like GET, POST, PUT, DELETE for all the crud operation. So, it's simple as comparison to WCF.
- It is an **OPEN** source but WCF is not.
- It is a **LIGHTWEIGHT** architecture and is good for devices, which have limited bandwidth (JSON & XML are lightweight) like smart cell phones. WCF is not lightweight.
- Web API Controller pattern is much similar to **MVC** Controller. Thus, it becomes easy to maintain and understand.

- Web API has very easy configuration settings as comparison to WCF.



Q146. WHAT IS REST AND RESTFUL?

- REST stands for Representational State Transfer. It is a set of guidelines which helps in creating a system where applications can easily communicate with each other.



REST Guidelines:

1. **Separation of Client and Server** - The implementation of the client and the implementation of the server can be done independently without each knowing about the other.

2. **Stateless** - The server will not store anything about the latest HTTP request the client made. It will treat every request as new. No session, no history.

3. **Uniform interface** – Identify of resources by URL for example: www.abc.com/api/questions

4. **Cacheable** - The cacheable constraint requires that a response should implicitly or explicitly label itself as cacheable or non-cacheable.

5. **Layered system** - The layered system style allows an architecture to be composed of hierarchical layers. For example, MVC is a layered system.

RESTFUL: If a system written by applying REST architectural concepts, it is called RESTful services.

Q147. IS IT POSSIBLE TO USE WCF AS RESTFUL SERVICES?

Yes, we can develop RESTful services with WCF.

Q148. WHAT ARE HTTP VERBS OR HTTP METHODS?

HTTP Method	Action	Examples
GET	Obtain information about a resource	http://example.com/api/orders (retrieve order list)
GET	Obtain information about a resource	http://example.com/api/orders/123 (retrieve order #123)
		http://example.com/api/orders

Q149. HOW TO CONSUME WEB API FROM A .NET MVC APPLICATION?

Web API methods can be consumed with the help of **HttpClient** class.

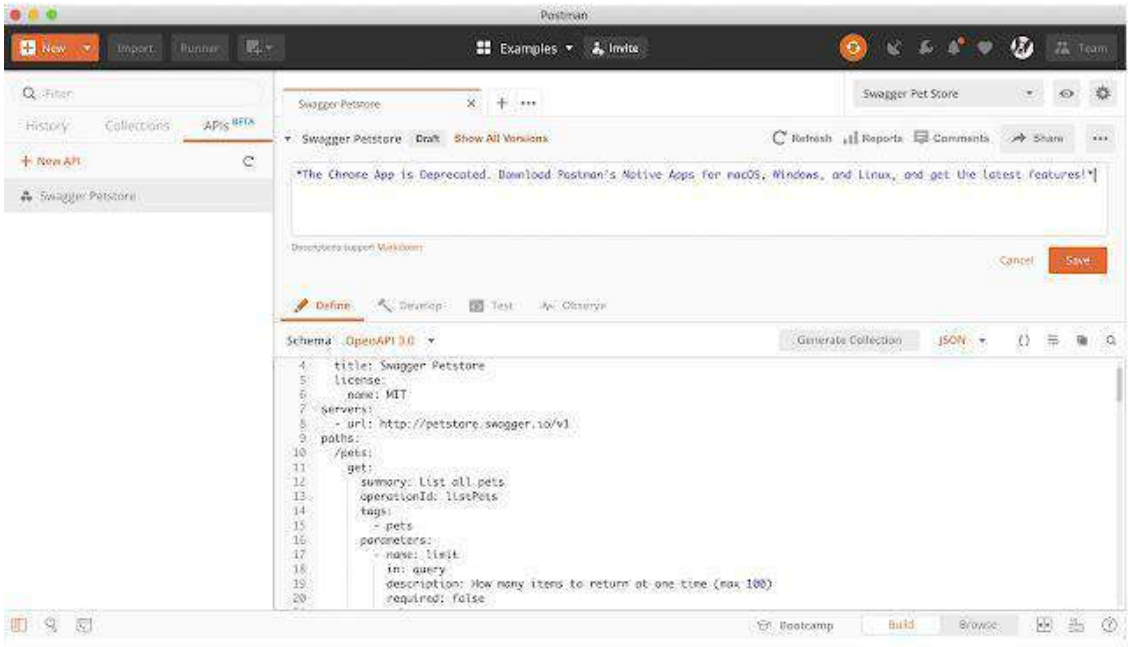
Below are the steps for consuming any Web API in a MVC application:

Q150. WHAT IS THE DIFFERENCE BETWEEN WEB API AND MVC CONTROLLER?

Web API Controller	MVC Controller
1. Web api controller derives from SYSTEM.WEB.HTTP.APICONTROLLER class.	ASP.NET MVC controller derives from SYSTEM.WEB.MVC.CONTROLLER class.
2. Web API controller does not give view support.	ASP.NET MVC gives view support.

Q151. HOW TO TEST WEB API? WHAT ARE THE TOOLS?

FIDDLER, **POSTMAN** & SWAGGER tools. Here is the screenshot of POSTMAN tool:



Q152. WHAT ARE MAIN RETURN TYPES SUPPORTED IN WEB API?

A Web API controller action method can return following types:

1. **Void** – It will return empty content.

2. **HttpResponseMessage** - It will convert the response to an HTTP message.

3. **IHttpActionResult** - internally calls ExecuteAsync to create an HttpResponseMessage.

4. **Other types** – You can create your own custom return type.

Q153. WHAT IS THE DIFFERENCE BETWEEN HTTPRESPONSEMESSAGE AND IHTTPACTIONRESULT?

- In web API version 1.0, We have a type called **HttpResponseMessage** for receiving Http response message from API call.

- In Web API version 2.0, **IHttpActionResult** is introduced which is basically the replacement and advance version of HttpResponseMessage. It creates clean code and also simplifies unit testing.

Example of using HttpResponseMessage :

```
public HttpResponseMessage Get(int id)
{
     var product = dbContext.Products.Get(id);

     if(product == null)
        return Request.CreateResponse(HttpStatusCode.NotFound);

   // could also throw a HttpResponseException(HttpStatusCode.NotFound)

     return Request.CreateResponse(HttpStatusCode.OK, product);

}
```

Example of using IHttpActionResult :

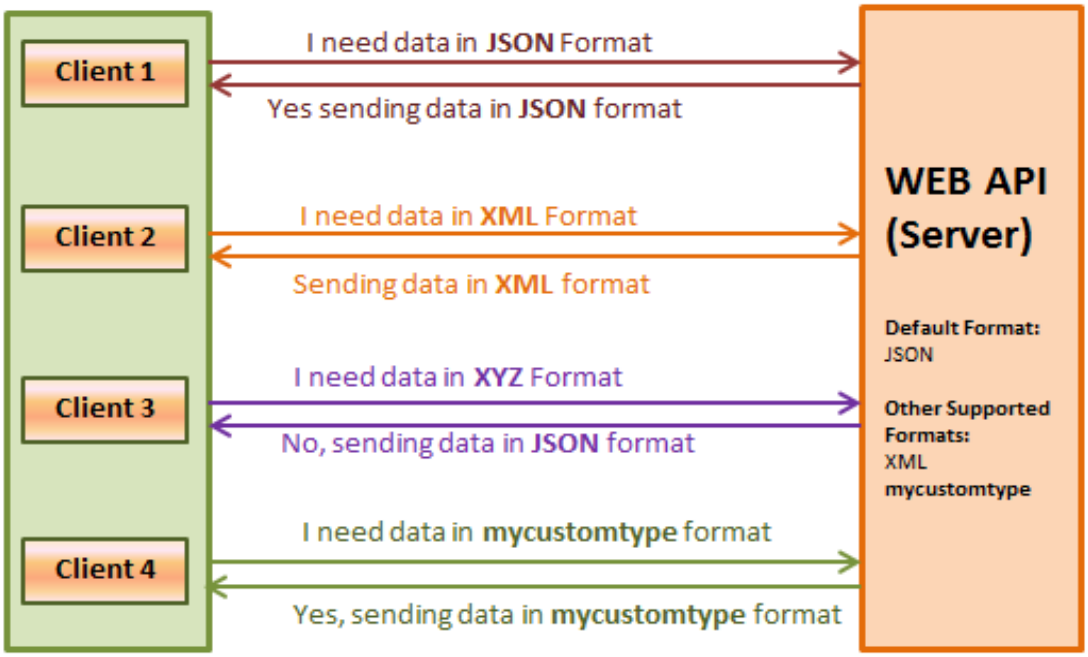
```
public IHttpActionResult Get(int id)
{
     var product = dbContext.Products.Get(id);

     if (product == null)
          return NotFound();

      return Ok(product);
}
```

Q154. WHAT IS CONTENT NEGOTIATION IN WEB API?

Whenever we consume an API, we receive data in either JSON or XML or plain text or your own custom format.

Which means the requester and responder applications are aware of the format in which they will send and receive data. This is nothing but Content Negotiation in Web API 2.

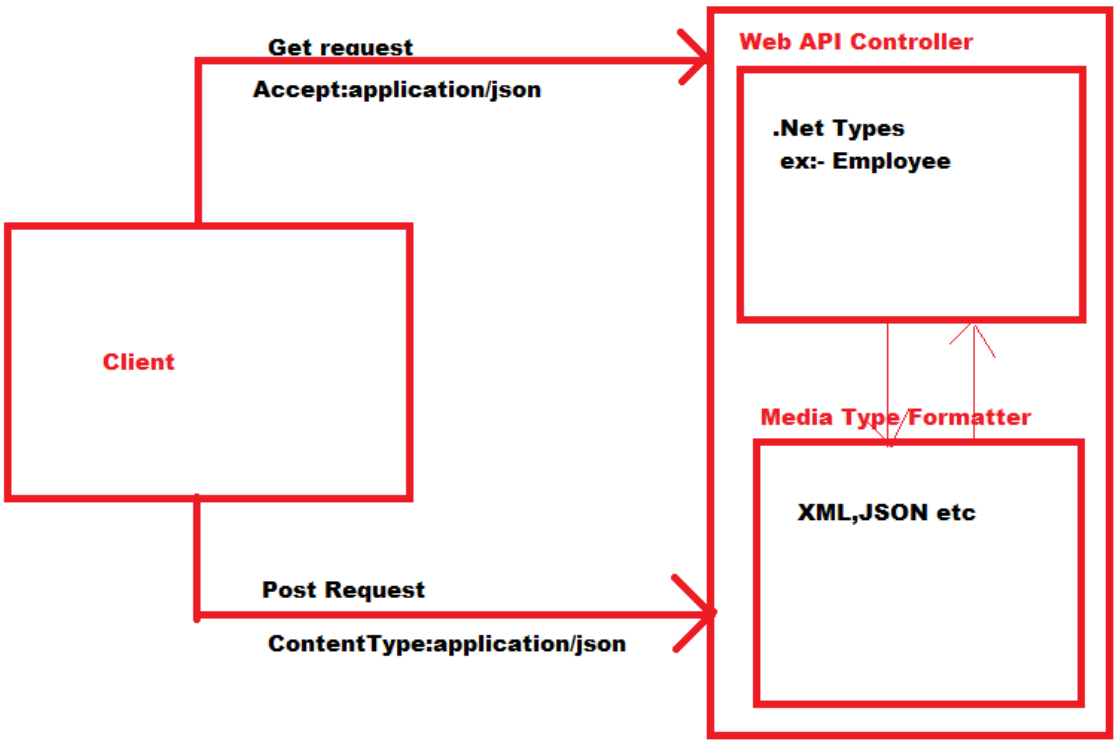
The same you can see in the diagram if the requester client will ask the data in JSON format then web api has to return it in json format.



Q155. WHAT IS MEDIATYPEFORMATTER CLASS IN WEB API?

MediaTypeFormatter is an abstract class from which JsonMediaTypeFormatter and XmlMediaTypeFormatter classes inherits.

Basically it helps in converting the .Net Types like Employees class object to convert in XML or JSON format.




```
namespace System.Net.Http.Formatting
{
    public abstract class BaseJsonMediaTypeFormatter : MediaTypeFormatter
    {
        protected BaseJsonMediaTypeFormatter();
        protected BaseJsonMediaTypeFormatter(BaseJsonMediaTypeFormatter formatter);

        public virtual int MaxDepth { get; set; }
```

Q156. WHAT ARE RESPONSE CODES IN WEB API?

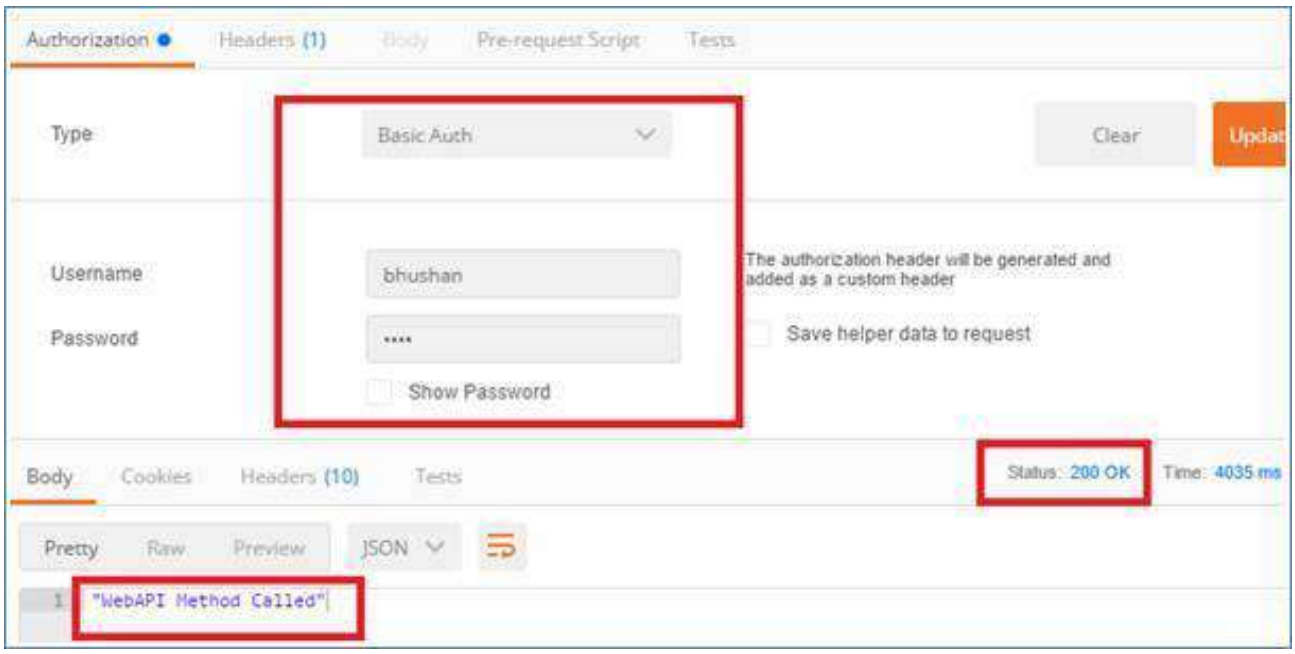
1. **1xx: Informational** – Communicates transfer protocol-level information.
2. **2xx: Success** – Indicates that the client's request was accepted successfully.
3. **3xx: Redirection** – This means request is not complete. The client must take some additional action in order to complete their request.
4. **4xx: Client Error** – This means there is some error in API code.
5. **5xx: Server Error** – This means the error is not due to web api code but due to some environment settings.

Q157. WHAT IS BASIC AUTHENTICATION IN WEB API?

Authentication means verifying a user who is accessing the system or Web API.

- **Basic Authentication -** In Basic Authentication, **the user passes their credentials [user name and password] on a post request**. Then at the WebAPI end, credentials are verified.
  If the credentials are valid, then a session will initiate to accept the subsequent requests without validating the user again.

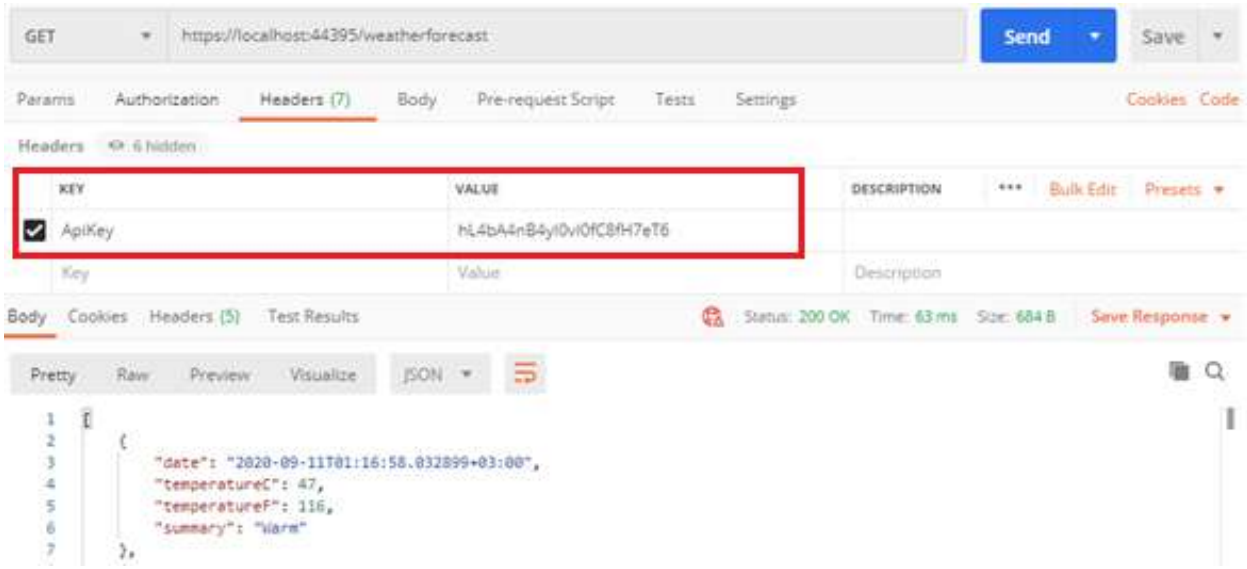
Below is the screenshot of Postman where BasicAuth is used for authentication:



Q158. WHAT IS API KEY AUTHENTICATION IN WEB API?

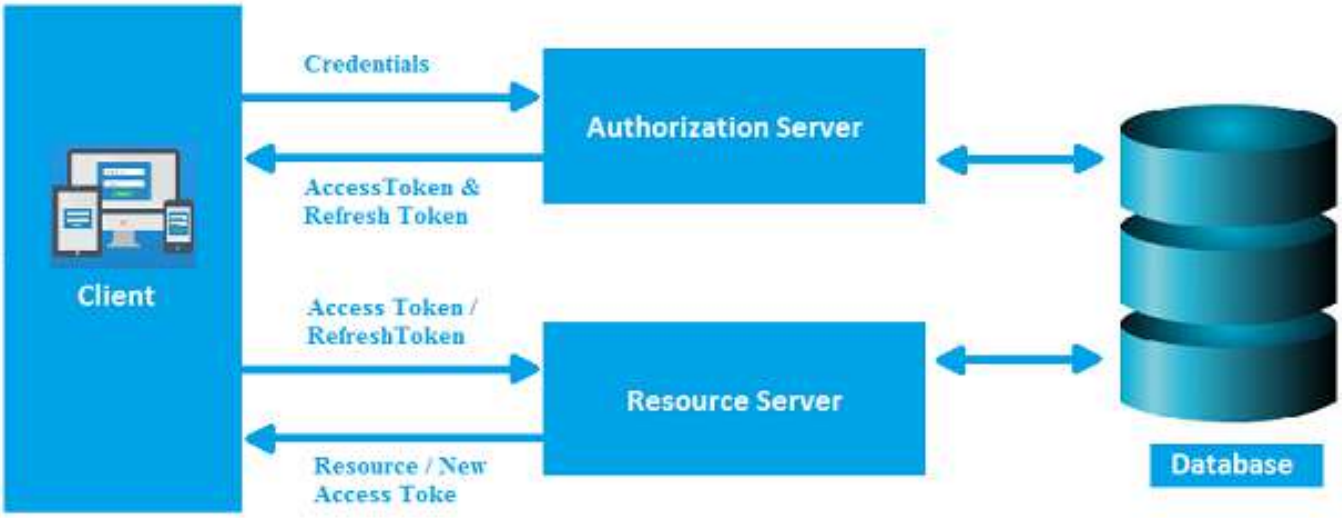
- **API Key Authentication-** In API Key Authentication, the API owner will share an API key with the users and this key will authenticate the users of that API.

Below is the screenshot of Postman where API Key Authentication is used for authentication:



Q159. WHAT IS TOKEN BASED AUTHENTICATION?

Token-based authentication is a 4 step process:



1. Client application first sends a request to Authentication server with a valid credentials.

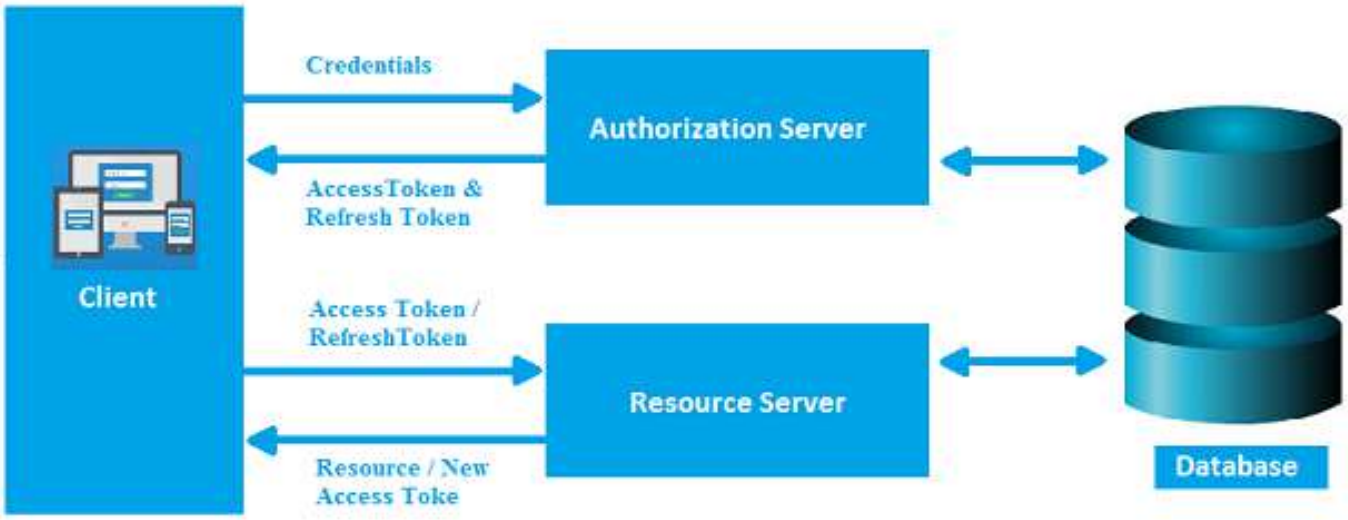
2. The Authentication server/ Web API sends an Access token to the client as a response. This token contains enough data to identify a particular user and it has an expiry time.

3. The client application then uses the token to access the restricted resources in the next requests until the token is valid.

4. If the Access token is expired, then the client application can request for a new access token by using Refresh token.

Q160. WHAT IS OAUTH?

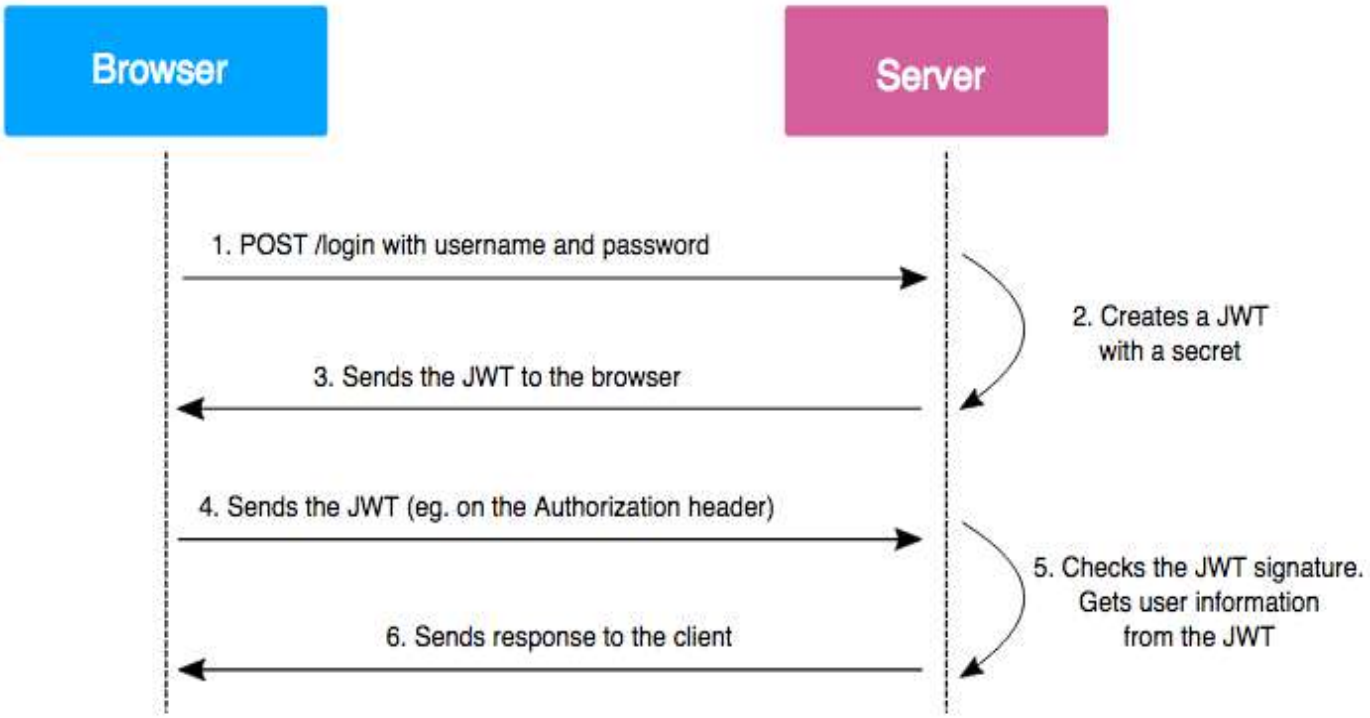
OAuth is a standard for creating token based authentication and authorization.



Q161. WHAT IS JWT AUTHENTICATION?

- JWT authentication is a token base authentication where JWT is a token format.

Below is the flow of the JWT authentication. It is the most popular token based authentication.

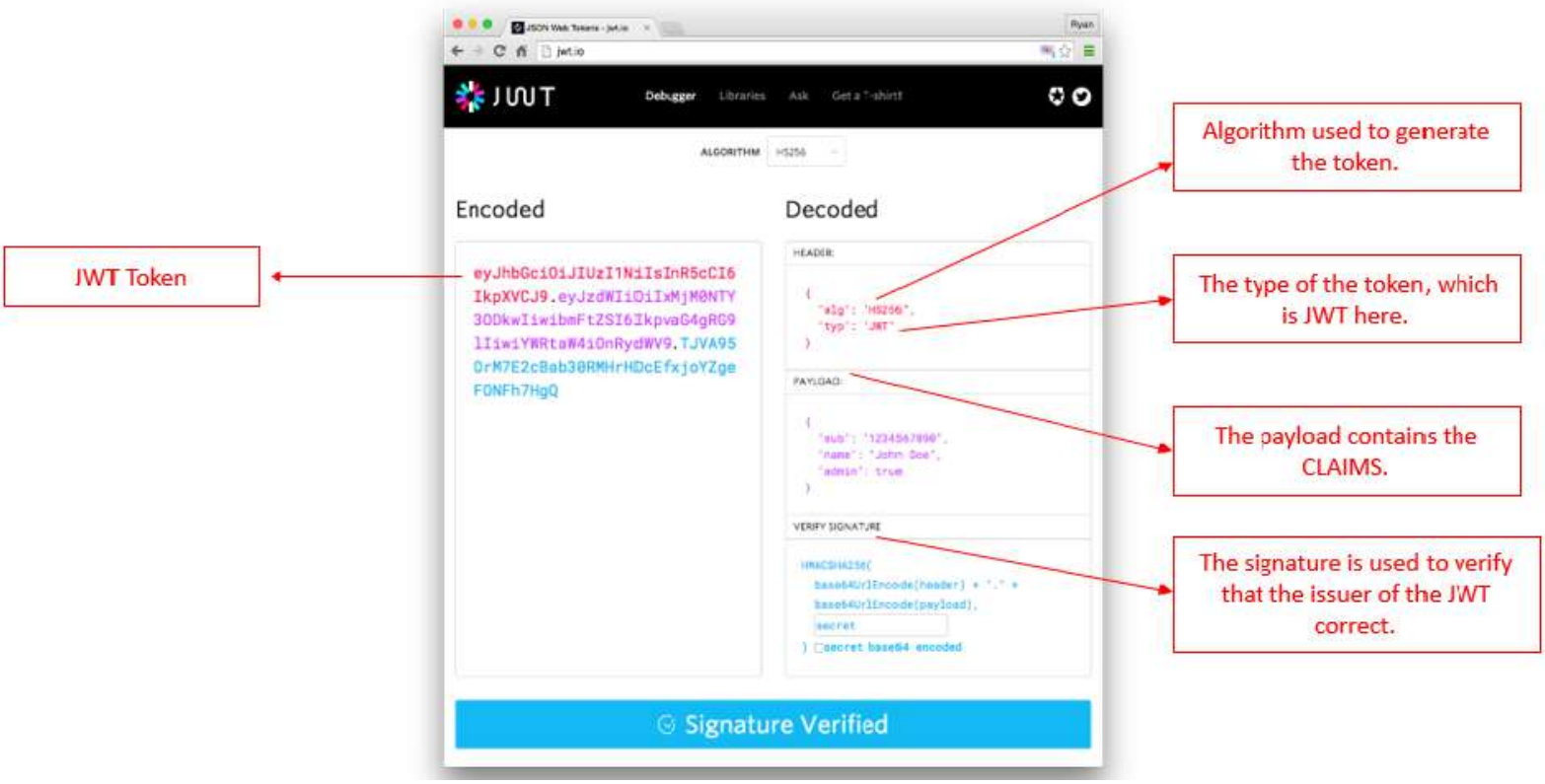


Q162. WHAT ARE THE PARTS OF JWT TOKEN?

JWT token has 3 parts:

1. Header
2. Payload
3. Signature

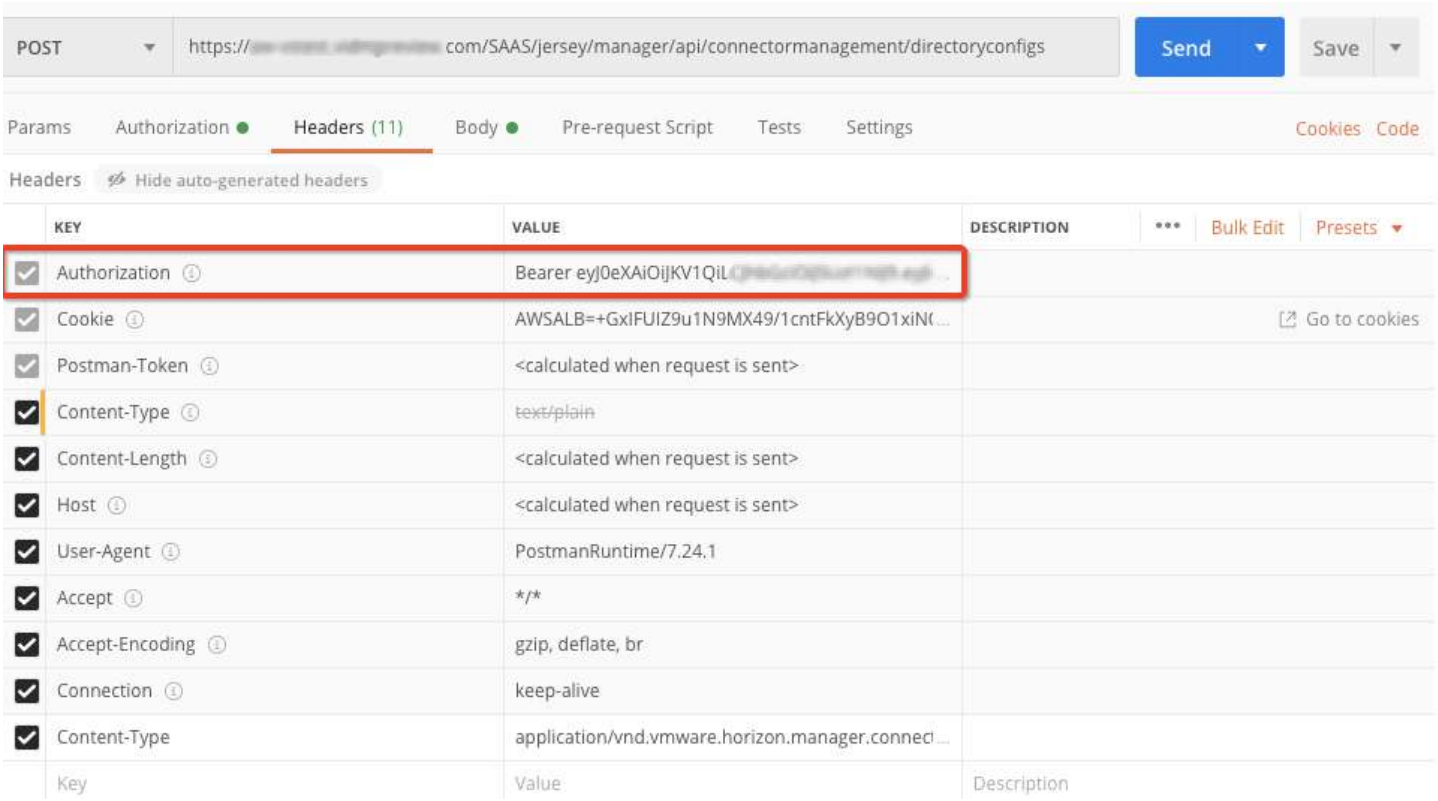
Below are the description of the parts:



Q163. WHERE JWT TOKEN RESIDE IN THE REQUEST?

In **REQUEST HEADER**.

See below in postman under the Header tab the JWT token is present with key and value.

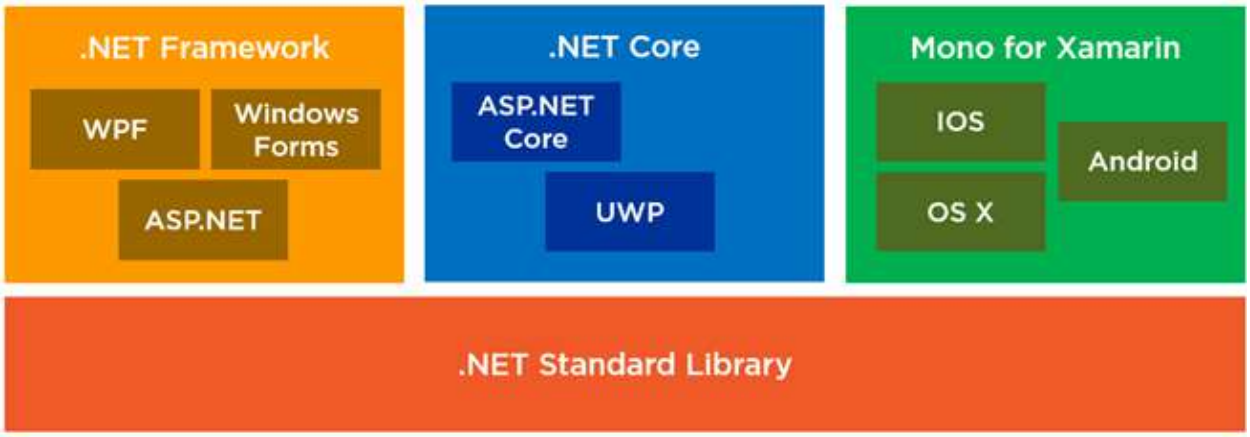


Chapter 10 - .NET CORE

Q164. WHAT IS .NET CORE?

Definition –

.NET Core is completely a **NEW** framework, which is a **FREE** and **OPEN-SOURCE** platform developed and maintained by Microsoft.



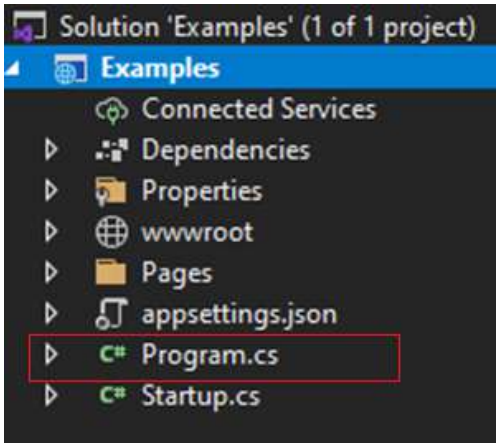
Q165. WHAT IS .NET STANDARD?

.NET Standard is not a framework. .NET Standard defines a set of API's or you can say libraries or set of rules. If any framework will follow these rules that means it is compliant with .NET standard. Right now .NET Framework, .NET Core and Xamarin follow .NET standard.

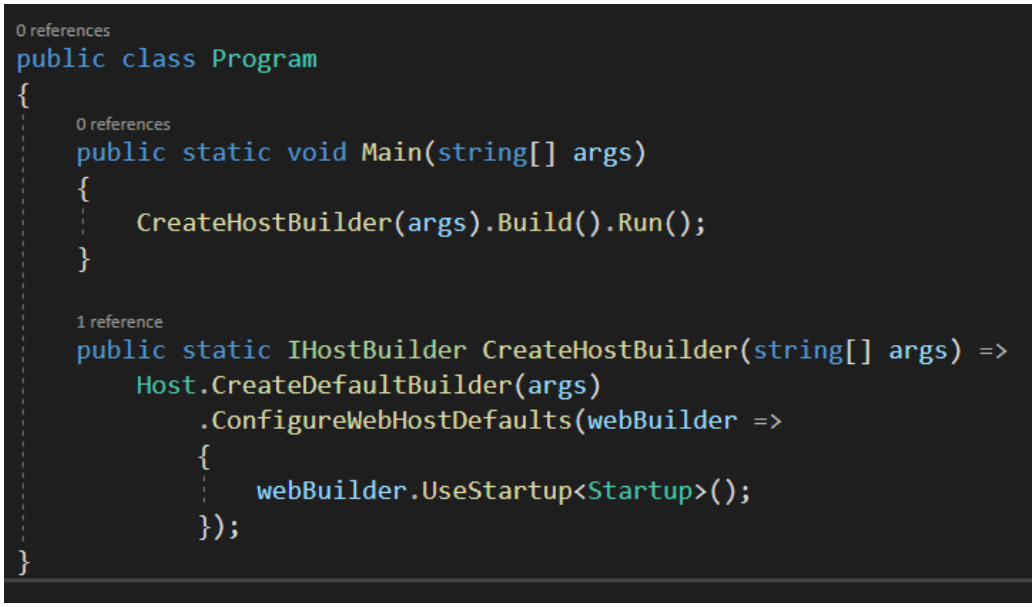
Q166. WHAT ARE THE ADVANTAGES OF .NET CORE OVER .NET FRAMEWORK?

Q167. WHAT IS THE ROLE OF PROGRAM.CS FILE IN ASP.NET CORE? OR WHAT IS CREATEHOSTBUILDER / CREATEDEFAULTBUILDER?

- Program.cs fille is the entry point of application via Main() method.



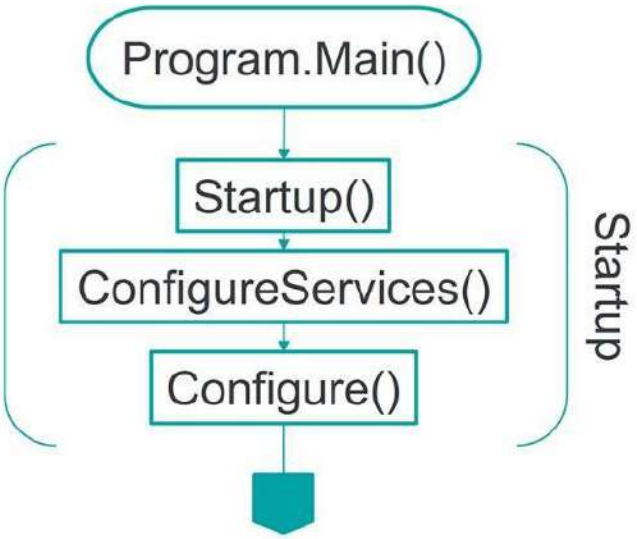
CreateHostBuilder and CreateDefaultBuilder method prepare the default settings for web server for the application.



```
0 references
public class Program
{
    0 references
    public static void Main(string[] args)
    {
        CreateHostBuilder(args).Build().Run();
    }

    1 reference
    public static IHostBuilder CreateHostBuilder(string[] args) =>
        Host.CreateDefaultBuilder(args)
            .ConfigureWebHostDefaults(webBuilder =>
            {
                webBuilder.UseStartup<Startup>();
            });
}
```

Q168. WHAT IS THE ROLE OF STARTUP.CS FILE?

Inside Program.cs file, UseStartup method redirects the request to Startup.cs file.



Startup.cs class does two things:

1. It configures the **SERVICES** which are required by the app by using **ConfigureServices** Method.

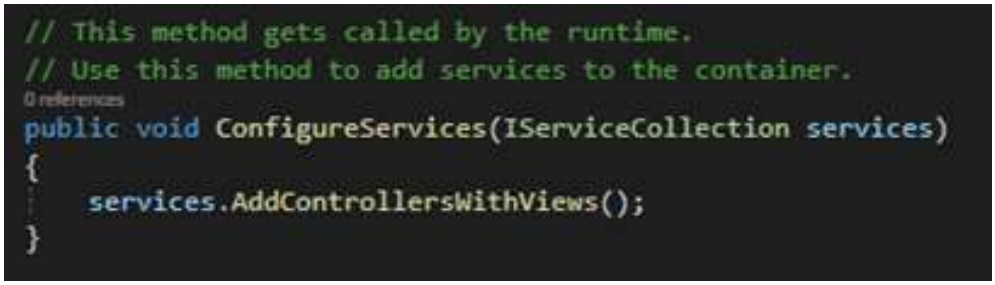
2. It defines the app's **REQUEST HANDLING PIPELINE** as a series of middleware components by using **CONFIGURE** method.

Q169. WHAT IS THE ROLE OF CONFIGURESERVICES METHOD?

- ConfigureServices method called before the 'Configure' method to configure the APP'S SERVICES.

It is an OPTIONAL method.

For example, your application at some point can use Views then the below method services.AddControllersWithViews() will help in configuring that.
Now it is not mandatory that every request will return the output as a view so therefore it is optional. A request may use it or may not.



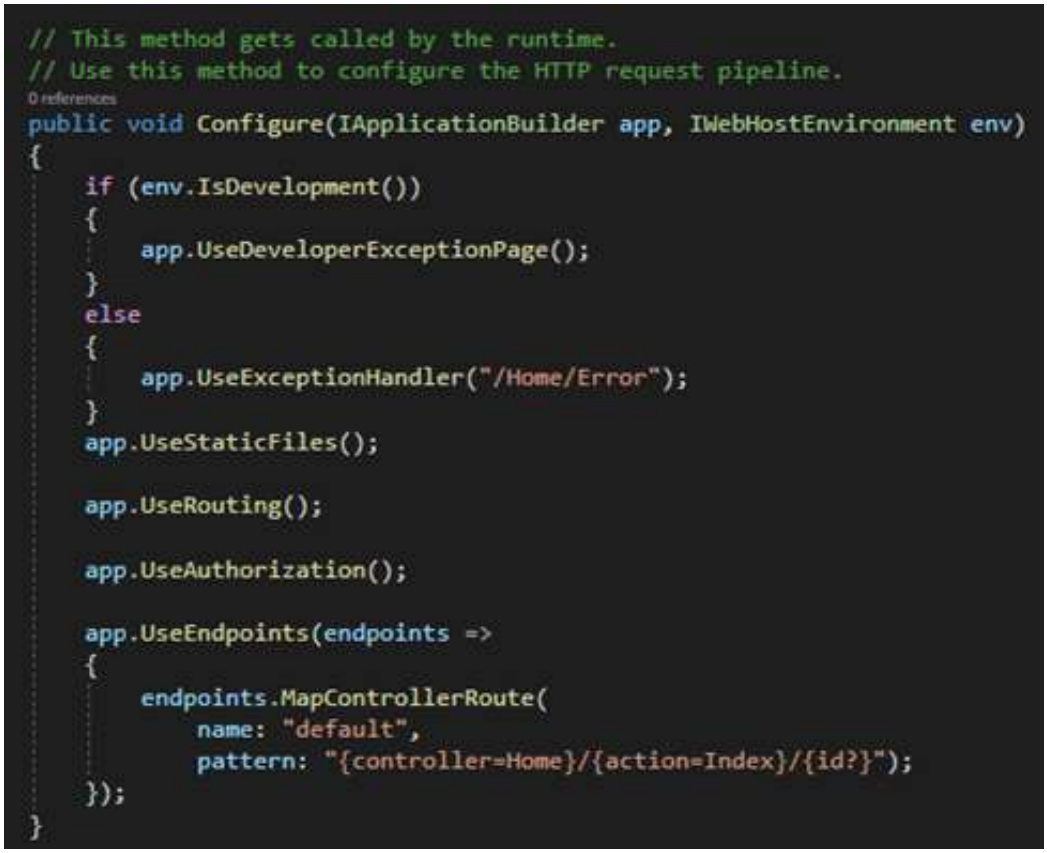
Q170. WHAT IS THE ROLE OF CONFIGURE METHOD?

- Configure method specifies how the app responds to HTTP request and response. It is used to setup request pipeline.

It is not optional.

For example, in below screenshot when the request will arrive, it will go through all the methods present inside Configure method one by one.

Every request will go through all these methods or we also call them middleware so that's' why we say Configure methods are not optional where as ConfigureServices methods are optional.



```
// This method gets called by the runtime.
// Use this method to configure the HTTP request pipeline.
0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
    }
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

Q171. WHAT IS THE DIFFERENCE BETWEEN CONFIGURESERVICES & CONFIGURE METHOD?

- ConfigureServices is an optional method used to register dependent classes/services inside the DI container.

- Configure method specifies how the app responds to HTTP request and respond. It is used to add MIDDLEWARE components or to set request pipeline. It is not optional.

- ConfigureServices method called before the 'Configure' method to configure the APP'S SERVICES.

Q172. WHAT IS DEPENDENCY INJECTION?

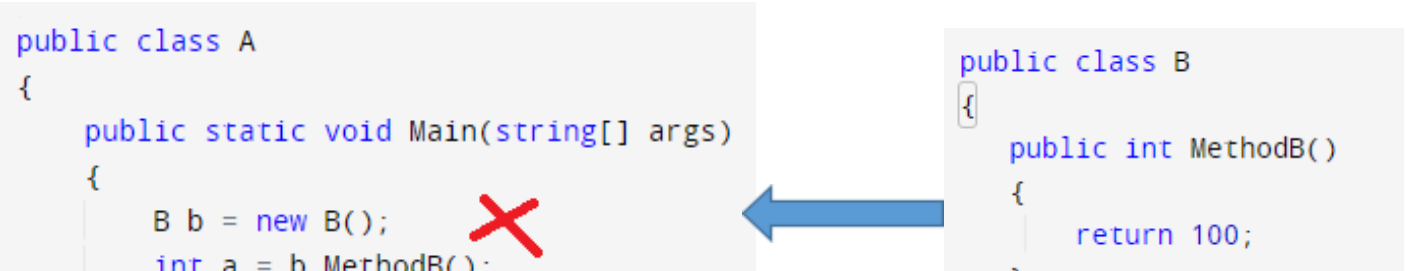
Dependency Injection (DI) is a software design pattern that allows us to develop loosely coupled code.

Now how to implement dependency injection is explained in the video lectures. Please refer them for a better understanding.

Q173. WHY TO USE DEPENDENCY INJECTION? OR WHAT PROBLEMS DOES DEPENDENCY INJECTION SOLVE?

Dependency injection decoubles the two components or classes.

For example, A is a class here and B class is the dependency, so one way to use the dependency is to create the object like shown below:



But now if we want to replace MethodB of B class with MethodC then we have to change the main method also right? Which is a testing impact.

So by using dependency injection we can use interfaces and design the classes in a manner that Main method needs not to be changed for any change in class B.

That will decouple these classes and that is the main agenda of dependecy injection.

Now how to implement dependency injection is explained in the video lectures. Please refer them for a better understanding.

Q174. HOW CAN WE INJECT THE DEPENDENCY INTO THE CONTROLLER?

This is better explained in the video lectures. Please refer them.

Q175. WHAT ARE THE TYPES OF DEPENDENCY INJECTION?

Dependency injection can be done in various ways. Below are the ways dependency injection can be done:

1. Constructor Injection

2. Property Injection

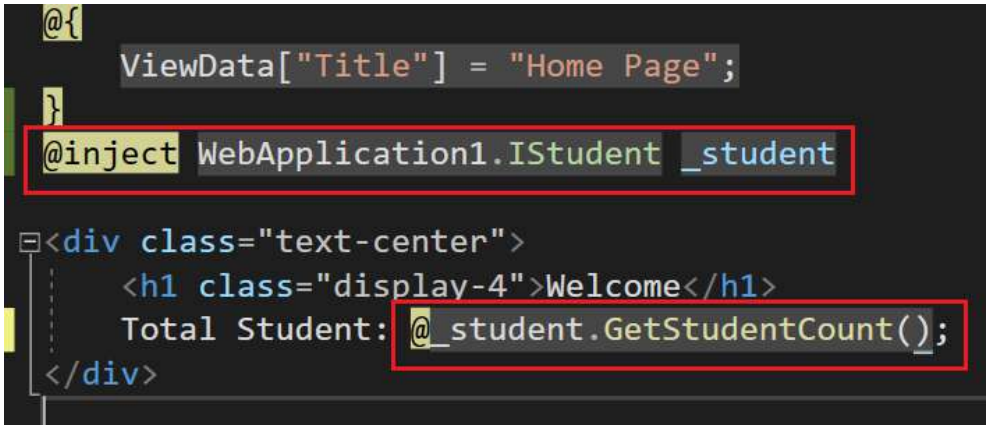
3. Method Injection

Dependency injection can be achieved with the help of any one of these injections.

Q176. HOW TO USE DEPENDENCY INJECTION IN VIEWS IN ASP.NET CORE?

A service can be injected into a view using the **@inject directive**.

For example, in below screenshot IStudent is injected in the view.



Q177. WHAT IS MIDDLEWARE IN ASP.NET CORE?

- A middleware is nothing but a component (class) that is executed on EVERY REQUEST in the ASP.NET Core application.

We can set up the middleware in ASP.NET using the CONFIGURE method of our STARTUP class like shown below.

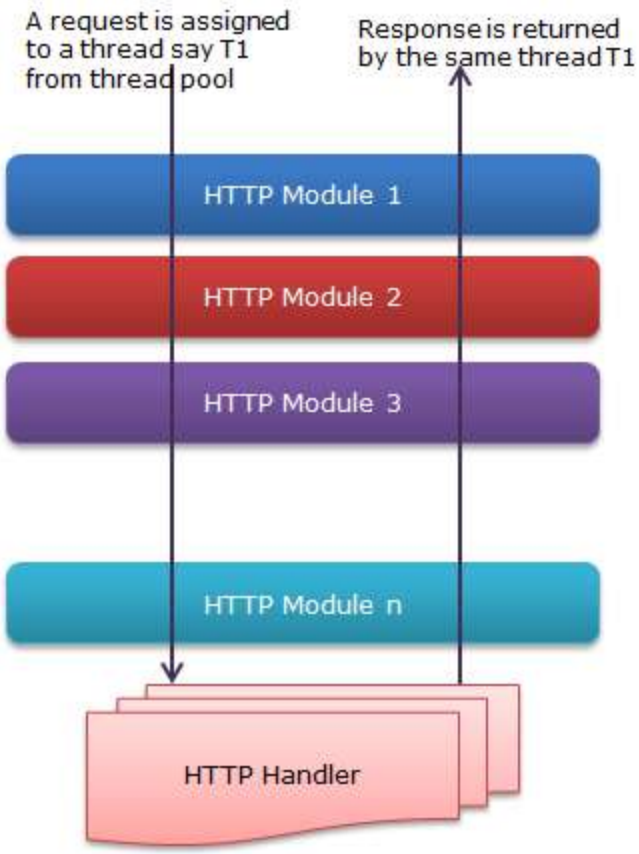
All these methods – UseDeveloperExceptionPage, UseStaticFiles, UseRouting etc are middlewares.

```
// This method gets called by the runtime.
// Use this method to configure the HTTP request pipeline.
0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
    }
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

Q178. HOW ASP.NET CORE MIDDLEWARE IS DIFFERENT FROM HTTPMODULE?

If you have worked in asp.net webforms then you might have heard of HTTPModules.

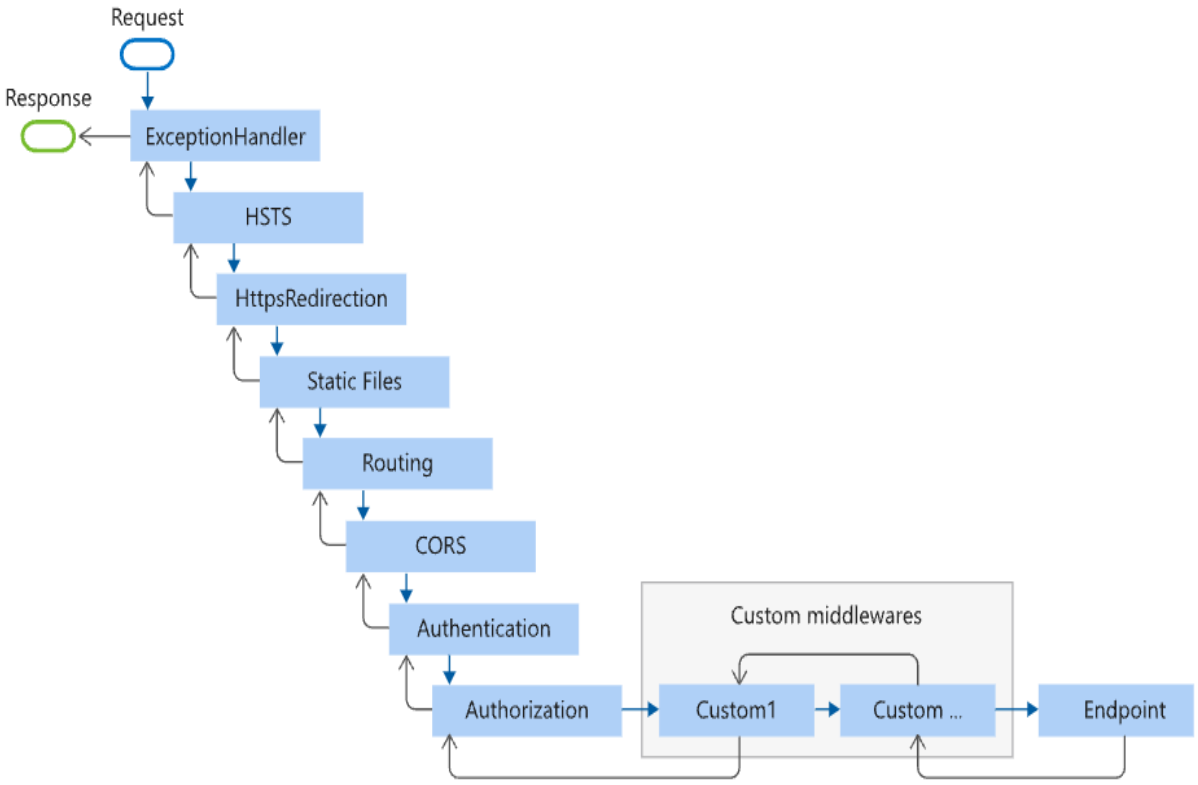
Middleware almost behave similar to HTTPModule, as they both are the part of request pipeline.



Q179. WHAT IS CUSTOM MIDDLEWARE? HOW TO ADD CUSTOM MIDDLEWARE IN ASP.NET CORE?

There are middlewares, which are provided by the .NET Core framework like static file, routing, CORS, Authentication, Authorization etc.

Similar to that you can also add your own custom middleware and add them into the Configure method of startup.cs class like this shown below.



Below are the steps to add custom middleware in your project:

1. Right click on the project and you can see and add middleware.cs class file like any other class file in the project.

Here you can see the code which is automatically generated by the .NET Core.

```
// You may need to install the Microsoft.AspNetCore.Http.Abstractions package into your project
2 references
public class CustomMiddleware
{
    private readonly RequestDelegate _next;

    0 references
    public CustomMiddleware(RequestDelegate next)
    {
        _next = next;
    }

    0 references
    public Task Invoke(HttpContext httpContext)
    {

        return _next(httpContext);
    }
}

// Extension method used to add the middleware to the HTTP request pipeline.
0 references
public static class CustomMiddlewareExtensions
{
    0 references
    public static IApplicationBuilder UseCustomMiddleware(this IApplicationBuilder builder)
    {
        return builder.UseMiddleware<CustomMiddleware>();
    }
}
```

2. Now whatever functionality you want to execute by this middleware, add that in the Invoke method just before return statement.

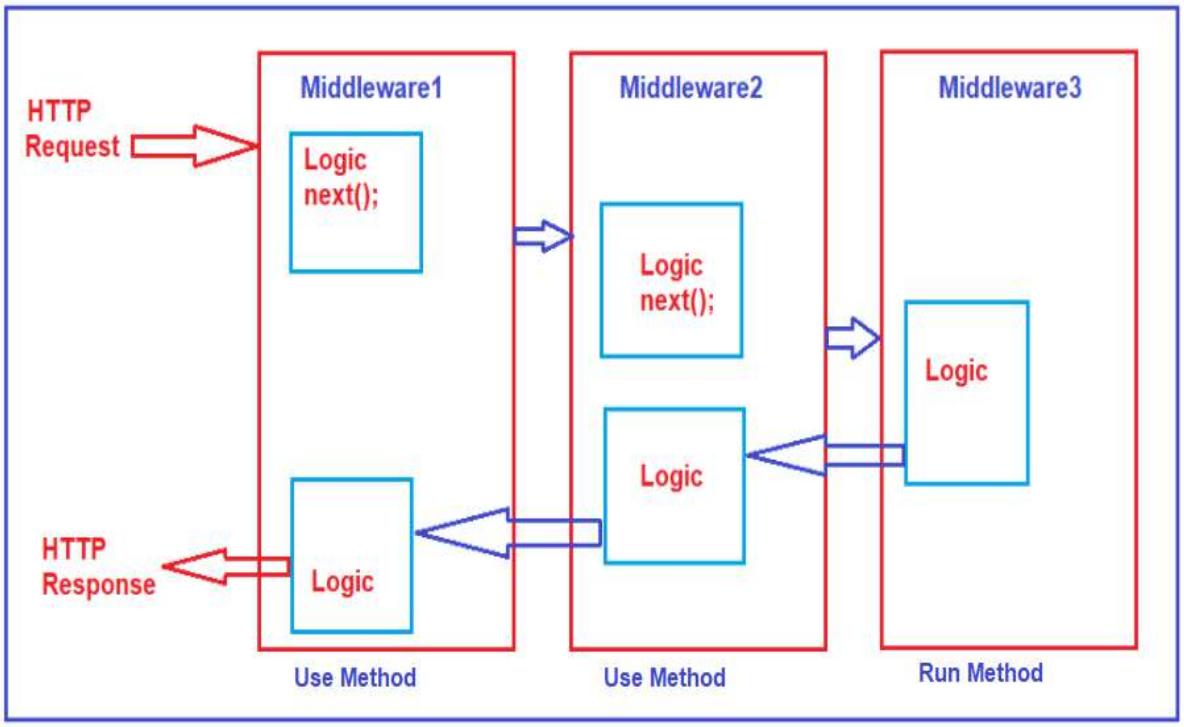
3. And then finally add that method in Configure method of Startup.cs class like this.

```
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");
    }

    app.UseStaticFiles();


    app.UseRouting();

    app.UseAuthorization();

    app.UseCustomMiddleware();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}"
            );
    });

}
```

Q180. WHAT IS REQUEST DELEGATE?

- Request delegates handle each HTTP request and are used to BUILD request pipeline by using RUN, MAP and USE extension methods.

Below is the diagram of a request and the purpose of Use and Run method.



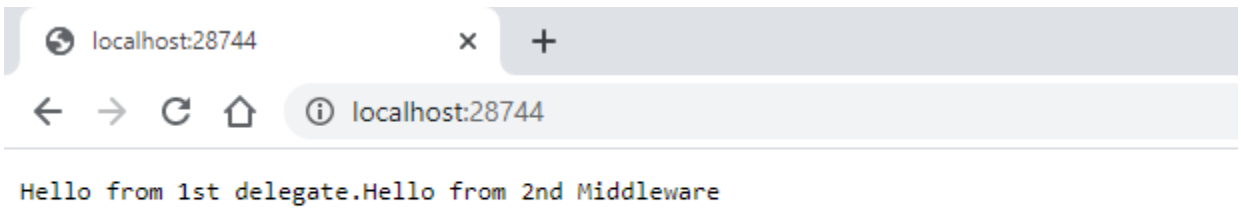
Q181. WHAT IS RUN(), USE() AND MAP() METHOD?

- USE - Use method will execute next middleware in sequence.

For example, the below code will write both lines as Use method will allow the next method to execute.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.Use(async (context,next) =>
            {
                await context.Response.WriteAsync("Hello from 1st delegate.");
                await next();
            });
            app.Run(async (context) =>
            {
                await context.Response.WriteAsync("Hello from 2nd Middleware");
            });
        }
```

See the second line is executed and printed.



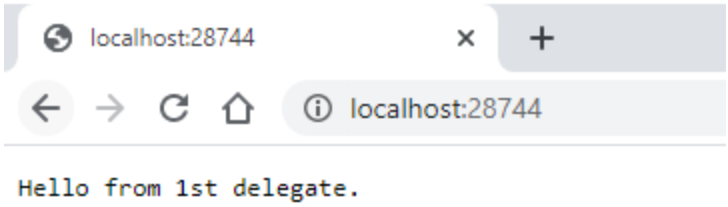
- RUN - Run method will **TERMINATE** the chain. No other middleware method will be allowed to run after this.

It normally placed at the end of any pipeline.

For example, the below code will write only first line because the Run() method will terminate the execution and the next line will not be printed.

```
// This method gets called by the runtime.
        // Use this method to configure the HTTP request pipeline.
        public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {

            app.Run(async context =>
            {
                await context.Response.WriteAsync("Hello from 1st delegate.");
            });
            app.Run(async (context) =>
            {
                await context.Response.WriteAsync("Hello from 2nd Middleware");
            });

        }
```

See the Run has terminated after first line, so second Run or any method will not be executed.

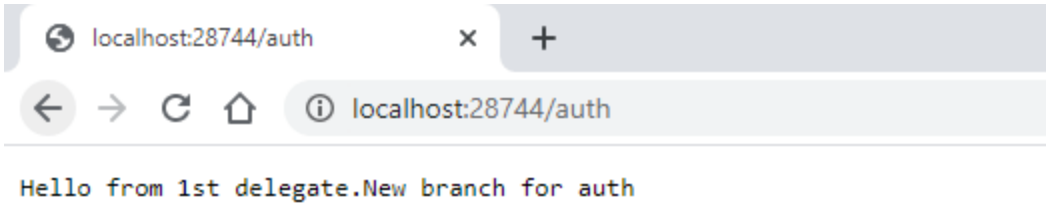


- MAP - Map method will only execute middleware, if path requested by user equals path provided in parameter.

For example, in below code the Map() method will only execute if there is "auth" present in the request URL. If not, then the second statement will not be printed.

```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
        {
            app.Use(async (context, next) =>
            {
                await context.Response.WriteAsync("Hello from 1st delegate.");
                await next();
            });

            app.Map("/auth", a =>
            {
                a.Run(async (context) =>
                {
                    await context.Response.WriteAsync("New branch for auth");
                });
            });

         }
```

See the url has "auth" in it so the second line is printed.



Q182. WHAT ARE THE TYPES OF SERVICE LIFETIMES OF AN OBJECT/ INSTANCE IN ASP.NET CORE?

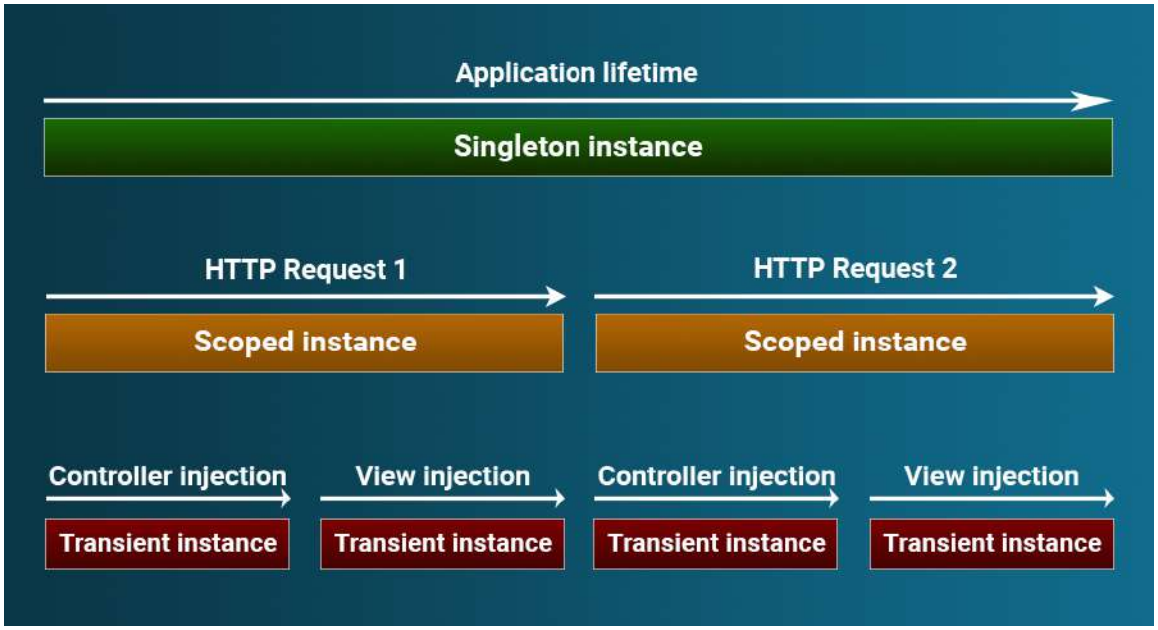
The service lifetime controls how long a result object will live for after it has been created by the container.

Meaning when an instance or object is created by the request then how long this instance will survive and how it will be shared between the multiple requests.

Three types of service lifetimes are there:

1. AddSingleton
2. AddScoped
3. AddTransient

These service lifetimes are added inside the ConfigureServices() method of the startup.cs class.



Q183. WHAT IS ADDSINGLETON, ADDSOPED AND ADDTRANSIENT METHOD?

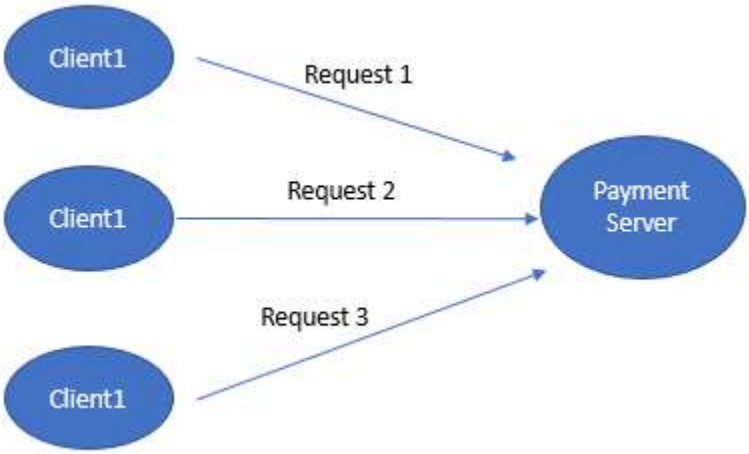
- AddSingleton - AddSingleton method create only **one instance** when the service is requested for first time. And then the same instance will be shared by all different http requests.

For example, in below code the Logger instance will be shared between multiple requests

```
// This method gets called by the runtime.
// Use this method to add services to the container.
0 references
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<ILogger, Logger>();
    //services.AddSingleton<IPayment, Payment>();
    //services.AddTransient<IDataAccess, DataAccess>();

}
```

- AddScoped - AddScoped method create **single instance per request**.

For every individual request there will be a single instance or object.



In below code, AddScoped method will make sure that Payment class instance will remain specific to a specific request and will not be shared between multiple requests.

```
// This method gets called by the runtime.
// Use this method to add services to the container.
0 references
public void ConfigureServices(IServiceCollection services)
{
    //services.AddSingleton<ILogger, Logger>();
    services.AddScoped<IPayment, Payment>();
    //services.AddTransient<IDataAccess, DataAccess>();

}
```

- AddTransient - AddTransient instance will not be shared at all even with in the same request.

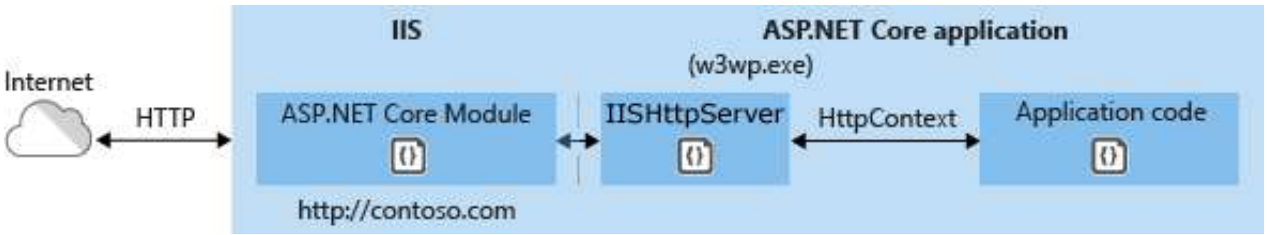
Every time a new instance will be created.

```
// This method gets called by the runtime.
// Use this method to add services to the container.
0 references
public void ConfigureServices(IServiceCollection services)
{
    //services.AddSingleton<ILogger, Logger>();
    //services.AddScoped<IPayment, Payment>();
    services.AddTransient<IDataAccess, DataAccess>();

}
```

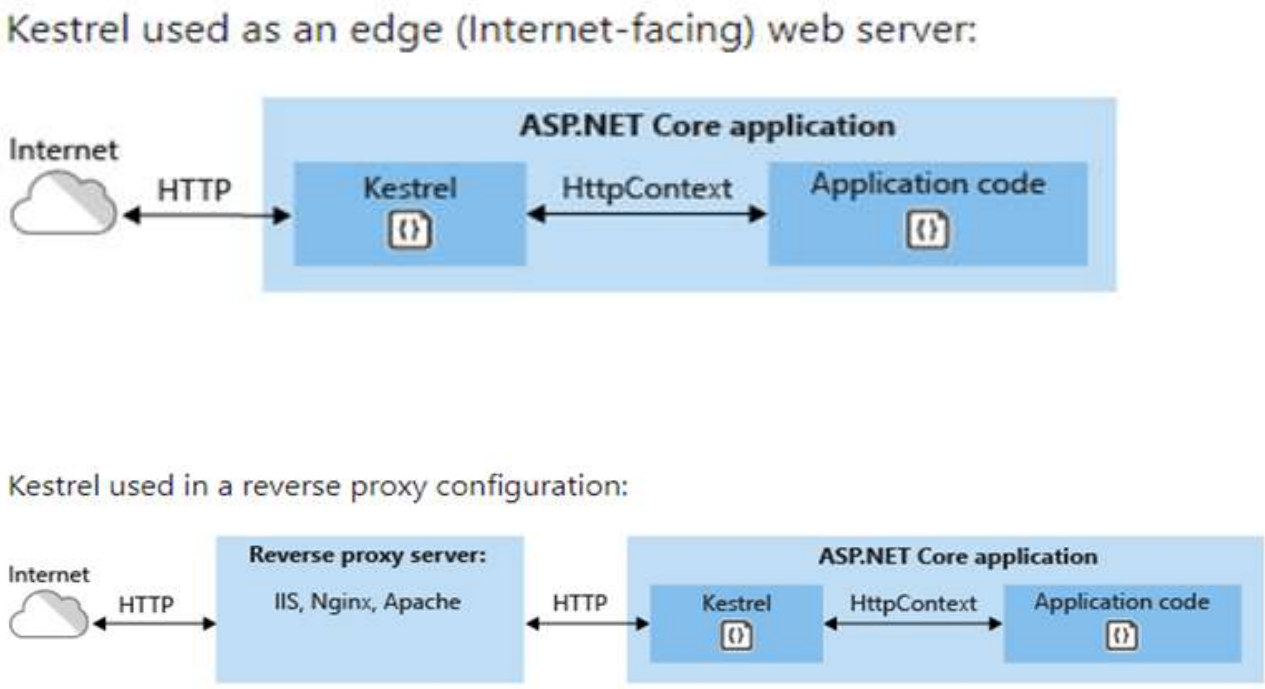
Q184. WHAT ARE THE TYPES OF HOSTING IN ASP.NET CORE? WHAT IS IN PROCESS AND OUT OF PROCESS HOSTING?

There are two type of hosting in ASP.NET Core:

1. In Process Hosting - In process hosting is related to IIS. In this only one web server is used for hosting and that is IIS. And the name of process is w3wp.exe.



2. Out of Process Hosting – This process use Kestrel and sometimes kestrel and IIS both for hosting like shown below.



In-Process	Out of Process
Process name is w3wp.exe	Process name is dotnet.exe
Only one web server (IIS)	Two web server (IIS/Nginx/Apache + Kestrel) One web server (Kestrel)

Q185. WHAT IS KESTREL? WHAT IS THE DIFFERENCE BETWEEN KESTREL AND IIS?

- Kestrel is **a cross-platform web server for ASP.NET Core**.

Cross-platform means it can work with any operating system like windows, lynx etc.

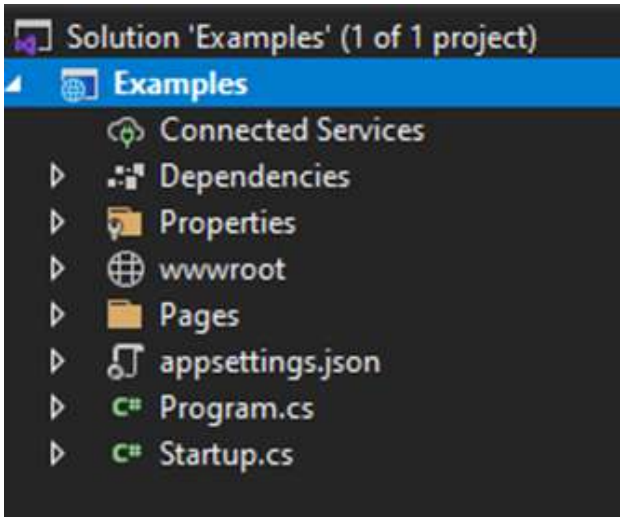
It is a type of out of process hosting. Kestrel can be used alone, or it can be used with IIS or Nginx web servers.

Here are the differences between Kestrel and IIS

Kestrel	IIS
Kestrel is a lightweight web server used for hosting.	IIS is a complete web server which is also used for hosting.
Kestrel is cross platform and can be used with other web servers like IIS, Nginx and Apache.	IIS is not cross platform and can only run-in windows.
Kestrel is open source like .NET Core	IIS is not open source

Q186. EXPLAIN DEFAULT PROJECT STRUCTURE IN ASP.NET CORE APPLICATION?

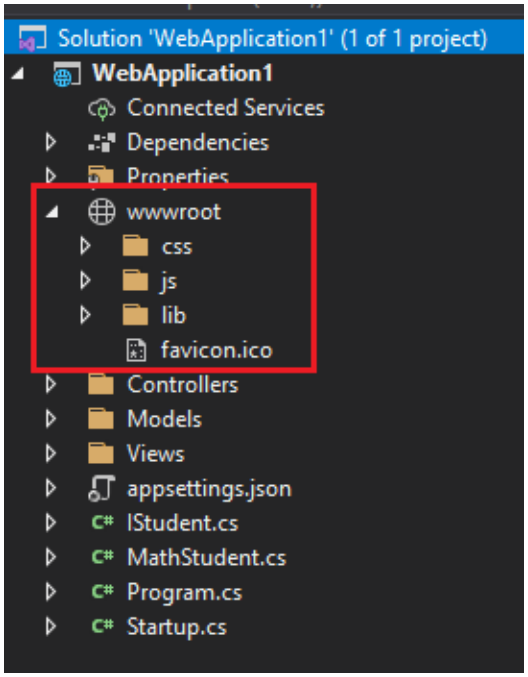
Below are the important files and folders present in asp.net core application by default:



- WWWROOT - To store static files of the application like js/css/images.
- PROGRAM.CS – Entry point of the application.
- STARTUP.CS – Configure services and request pipeline for the application.
- APPSETTINGS.JSON – Configuration settings like database connection strings and other things are saved here.

Q187. HOW ASP.NET CORE SERVE STATIC FILES?

ASP.NET Core template provides a root folder called WWWROOT which contains all these static files.



USESTATICFILES() method inside Startup.Configure enables the static files to be served to client.

```
// This method gets called by the runtime.
// Use this method to configure the HTTP request pipeline.
0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Home/Error");
    }
    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapControllerRoute(
            name: "default",
            pattern: "{controller=Home}/{action=Index}/{id?}");
    });
}
```

Q188. WHAT ARE THE MAIN JSON FILES AVAILABLE IN ASP.NET CORE?

- Global.json - You can define the solution level settings in global.json file for example your application name and version.

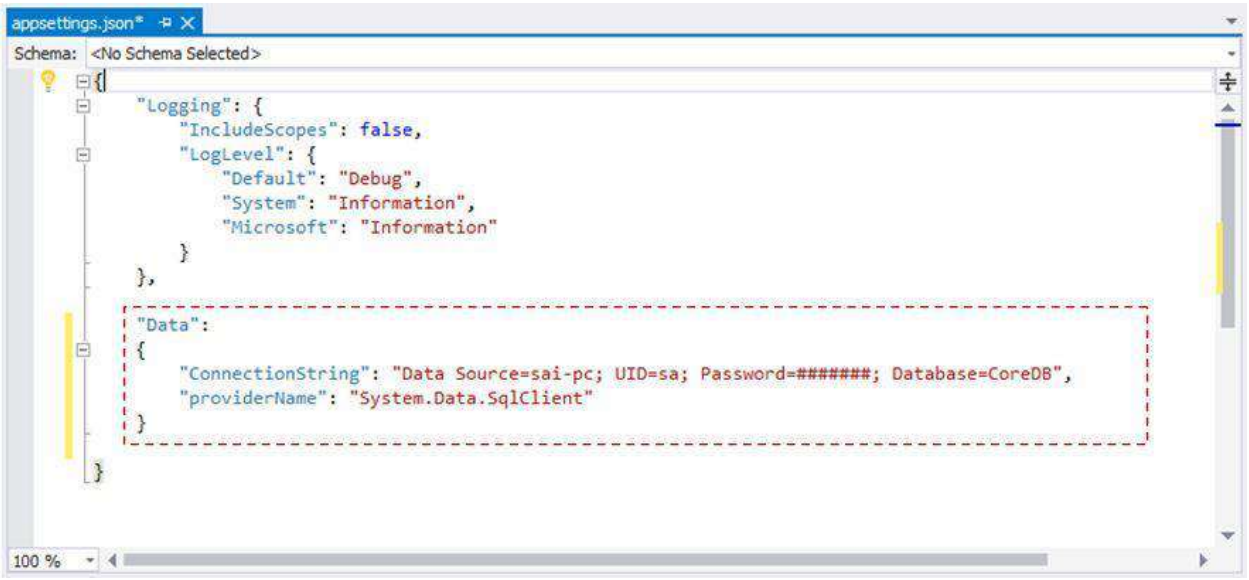
- Launchsettings.json – You can set the environment variables in this file. For example, set development or production environment in this file.

- Appsettings.json – Configuration settings like database connection string can be set in this file. Similar to web.config in ASP.NET.

- Project.json - ASP.NET Core uses Project.JSON file for storing all project level configuration settings. For example the nugget packages you have installed in the project.

Q189. WHAT ARE THE ROLES OF APPSETTINGS.JSON AND LAUNCHSETTING.JSON FILE IN ASP.NET CORE?

- Appsettings. json - The appsettings. json file is an **application configuration file used to store configuration settings** such as database connections strings, any application scope global variables, etc.



- LaunchSettings.json - The launchSettings.json file is **used to store the configuration information, which describes how to start the ASP.NET Core application**, using Visual Studio.

For example the hosting details like IIS etc.

Q190. WHAT IS THE DIFFERENCE BETWEEN APPSETTING.JSON AND LAUNCHSETTING.JSON FILE?

- Appsetting.Json store the configurations which are required when your application is running.

For example, database connection string is used when application is running.

- But Launchsetting.Json store the configuration which are required to start the application.

For example, what will be application url, that can be configure here.

Q191. WHAT ARE THE VARIOUS TECHNIQUES TO SAVE CONFIGURATION SETTINGS IN ASP.NET CORE?

Techniques used to save configuration settings in ASP.NET Core:

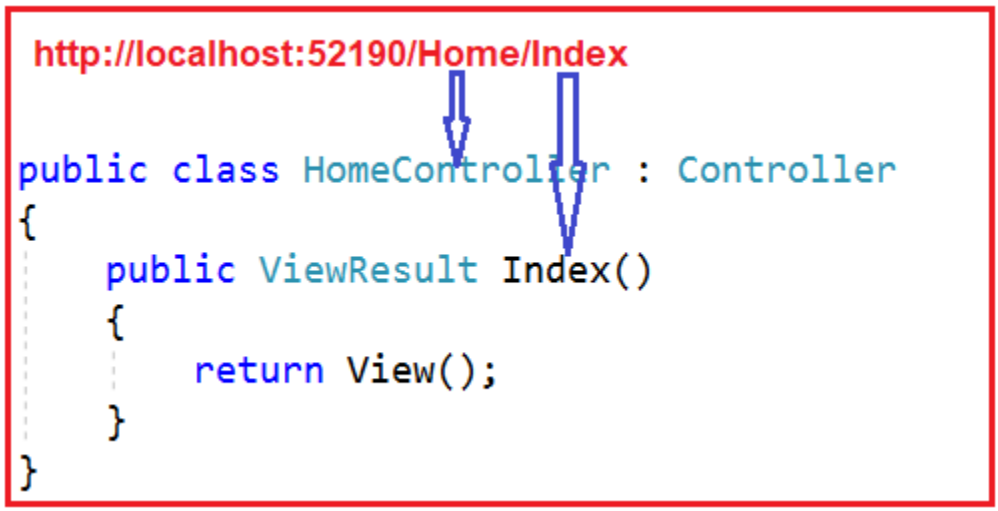
- Appsettings.json (Default) (Mostly Used)
- Azure Key Vault (Mostly used and it's the best if you are using Azure)
- Environment variables
- In-memory .NET objects
- Command Line Arguments
- Custom Providers

Q192. WHAT IS ROUTING? EXPLAIN ATTRIBUTE ROUTING IN ASP.NET CORE?

- Routing is used to handle incoming HTTP requests based on the URL.

This is automatically handled by the MVC framework.

For example, in below URL MVC will automatically detect what is controller, action method and ID parameter.



- Attribute based routing is the ability to manipulate the default behavior of URL by using Route Attribute.

Sometimes you don't want the default way of Routing and you want to map a different action method other than what is in URL.

For example, if this is the url:

http://localhost:60995/NewIndex

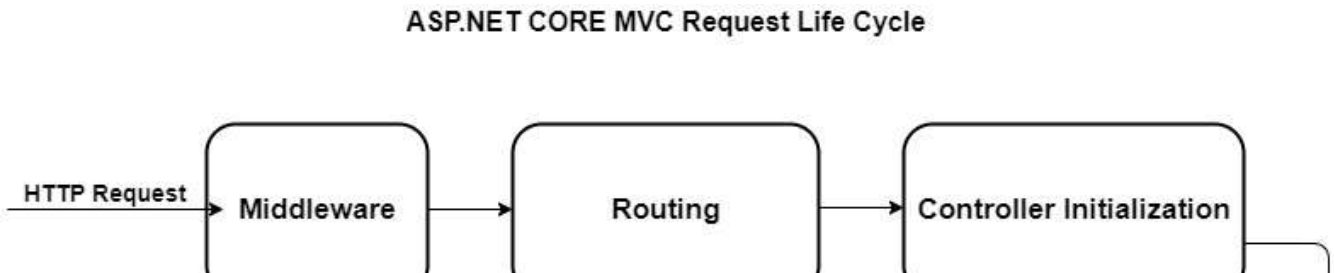
But on this URL you want to call "Index" action method in the controller then you can use Route attribute to map it.

```
[Route("")]
[Route("NewIndex")]
0 references
public IActionResult Index()
{
    return View();
}
```

Above code will check that if the action method is "NewIndex" then it will redirect it to "Index" method.

Q193. DESCRIBE THE COMPLETE REQUEST PROCESSING PIPELINE FOR ASP.NET CORE MVC?

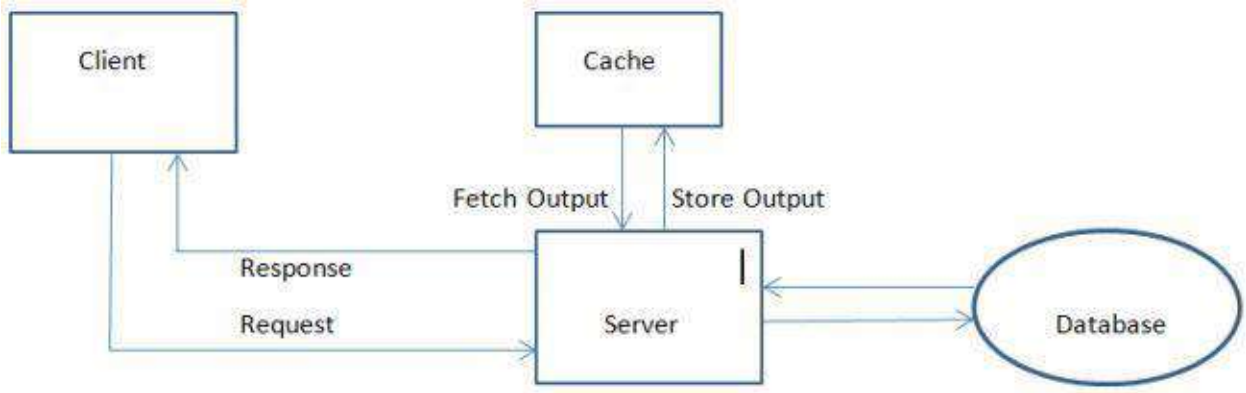
Below is a simple diagram of the flow of a request in the ASP.NET Core MVC application:



Q194. WHAT IS CACHING IN ASP.NET CORE?

- **Caching** refers to the process of storing frequently used data in cache, so that that data can be served much faster for any **future requests**.

Below is the flow of the caching in normal scenarios:

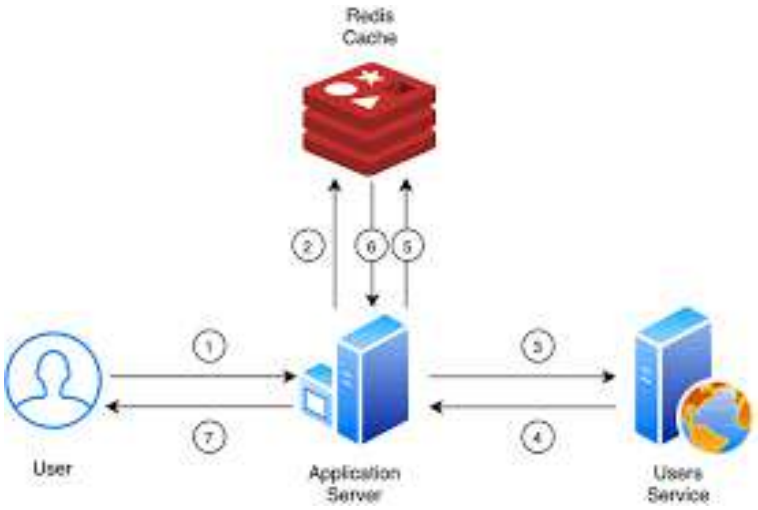


Q195. WHAT IS IN-MEMORY CACHING & DISTRIBUTED CACHING? WHEN TO USE IN-MEMORY CACHING AND WHEN TO USE DISTRIBUTED CACHING?

- **In memory caching –** It is the normal way of caching.
  An **in-memory** cache is stored in the memory of the same server which is hosting the application.
  Basically, the data is cached within the same application server.

- **Distributed caching** - It is used when you want to handle caching outside of your application.

For example, Redis is an open-source database distributed caching server.



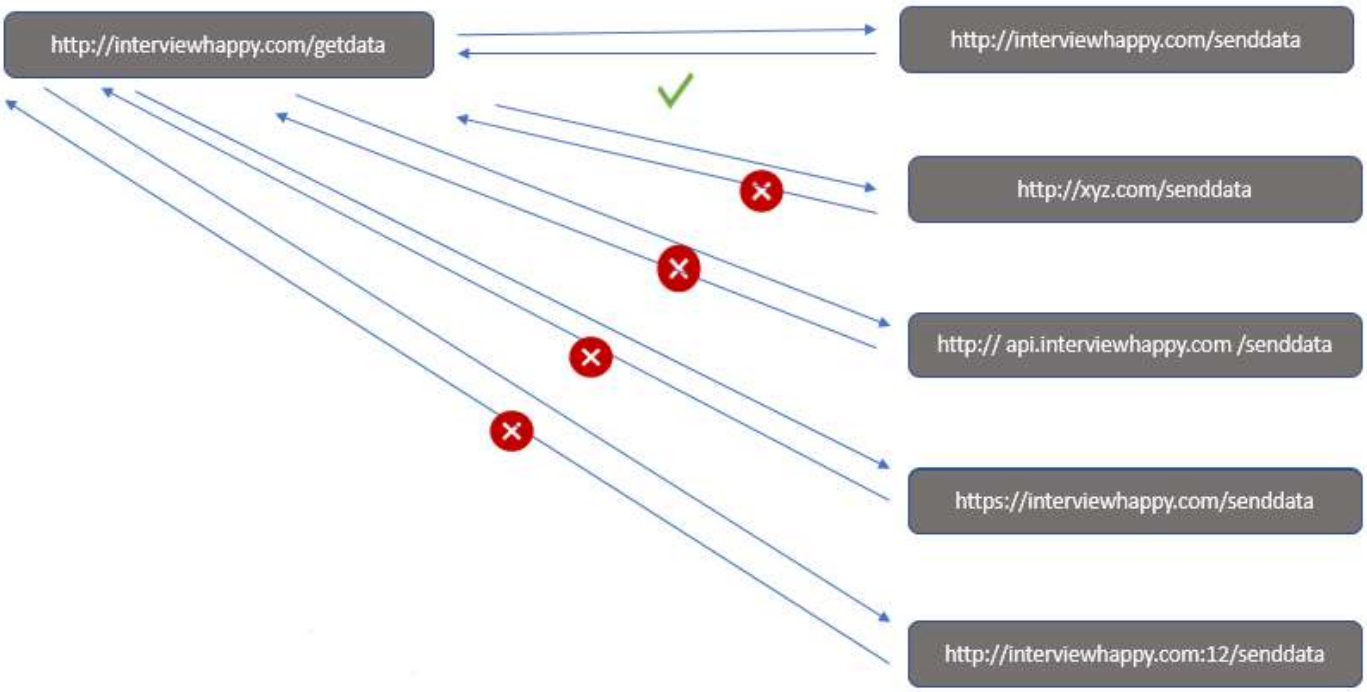
- In normal cases where the application size is small, use in-memory cache.

But where application is very big or it's a microservices based architecture, then use distributed caching.

Q196. WHAT IS CROSS-ORIGIN REQUESTS (CORS) IN ASP.NET CORE? WHY CORS RESTRICTION IS REQUIRED? HOW TO FIX CORS ERROR?

- As the name suggests **Cross Origin Resource Sharing** – meaning the data or resource sharing between different or cross domains is not enabled by default. This is restricted.

Below are the cases when CORS will allow and disallow the response:



Above in first case the response will be fine, but in next four cases the response will not be sent:

1. First case is when you want to get the response from a different domain.

2. Second case is when you want to get the response from a subdomain.

3. Third case is when the domain is same, but you are requesting from http and the sender is https.

4. And in last case, if the port is different.

In all 4 cases above you will receive the below error:

```
Access to XMLHttpRequest at 'http://localhost:5000/global_config'   step1:1
from origin 'http://localhost:8080' has been blocked by CORS policy:
Response to preflight request doesn't pass access control check: No 'Access-
Control-Allow-Origin' header is present on the requested resource.
```

- CORS is good because of security reason. Your API is secure by default otherwise any external website can try to access your data.

Now how to remove this CORS restriction? For that you have to do the two things:

1. Add Microsoft.AspNetCore.Cors nuget package to your project then

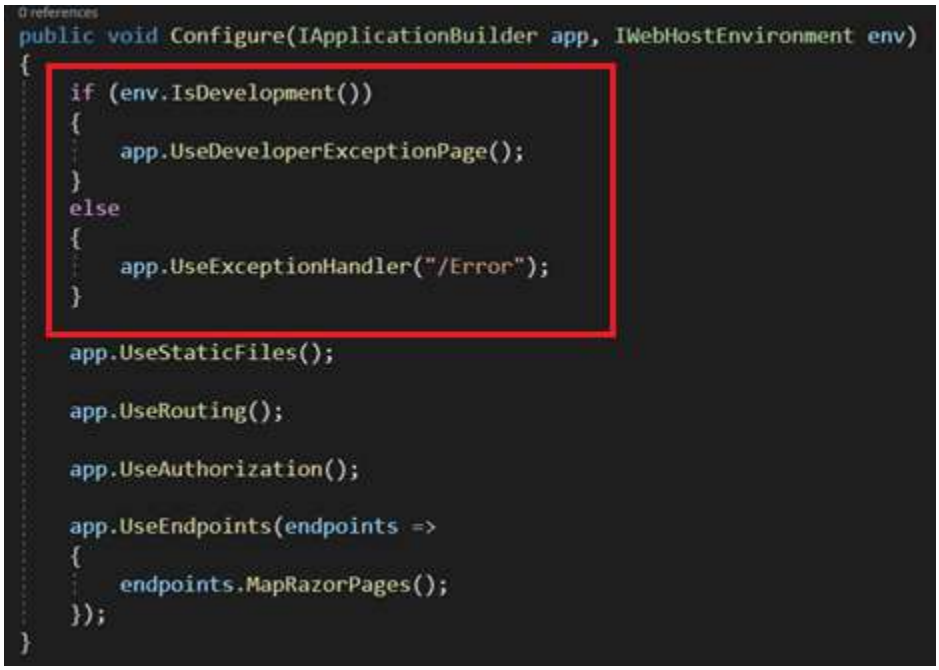
2. In startup class, edit the Configure and ConfigureServices method like this.

```
// This method gets called by the runtime. Use this method to add service
public void ConfigureServices(IServiceCollection services)
{
    services.AddCors();
    services.AddControllers();
}

// This method gets called by the runtime. Use this method to configure t
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    app.UseRouting();

    // global cors policy
    app.UseCors(x => x
        .AllowAnyMethod()
        .AllowAnyHeader()
        .SetIsOriginAllowed(origin => true) // allow any origin
        .AllowCredentials()); // allow credentials

    app.UseAuthentication();
    app.UseAuthorization();

    app.UseEndpoints(x => x.MapControllers());
}
```

Q197. HOW TO HANDLE ERRORS IN ASP.NET CORE?

- Error handling for development and other environments can be set in CONFIGURE method of Startup.cs class.

The IsDevelopment() method will check the ASPNETCORE_ENVIRONMENT value in LaunchSettings.json file. For live or production environment it will be false.



```
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");
    }

    app.UseStaticFiles();

    app.UseRouting();

    app.UseAuthorization();

    app.UseEndpoints(endpoints =>
    {
        endpoints.MapRazorPages();
    });
}
```

```
"Example": {
  "commandName": "Project",
  "dotnetRunMessages": "true",
  "launchBrowser": true,
  "applicationUrl": "http://localhost:5000",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Development"
  }
```

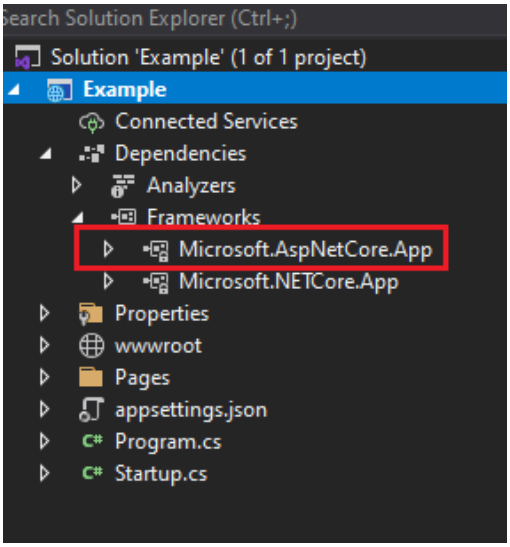
Q198. WHAT IS METAPACKAGE? WHAT IS THE NAME OF METAPACKAGE PROVIDED BY ASP.NET CORE?

- Metapackage is a consolidated package of all the mandatory dependencies for a .NET Core project. This package is present in your application by default.

Below is the list of dependencies in metapackage

```
Microsoft.AspNetCore.Diagnostics
Microsoft.AspNetCore.Hosting
Microsoft.AspNetCore.Routing
Microsoft.AspNetCore.Server.IISIntegration
Microsoft.AspNetCore.Server.Kestrel
Microsoft.Extensions.Configuration.EnvironmentVariables
Microsoft.Extensions.Configuration.FileExtensions
Microsoft.Extensions.Configuration.Json
Microsoft.Extensions.Logging
Microsoft.Extensions.Logging.Console
Microsoft.Extensions.Options.ConfigurationExtensions
NETStandard.Library
```

- Microsoft.AspNetCore.App is the metapackage provided by ASP.NET Core.



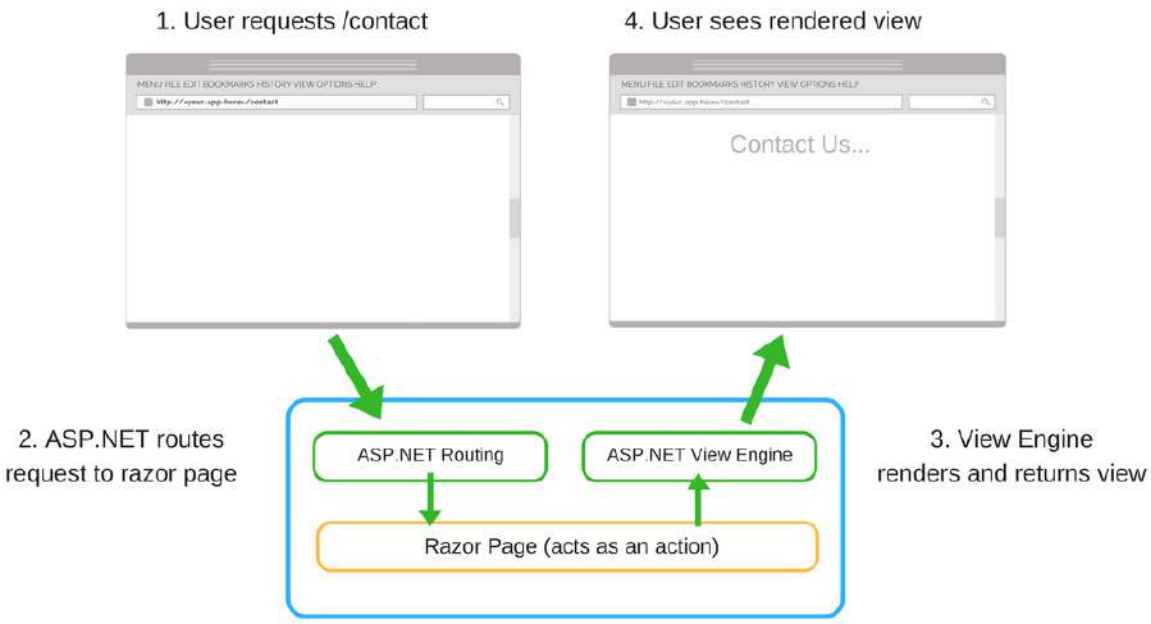
Q199. WHAT IS THE DIFFERENCE BETWEEN .NET CORE AND .NET 5?

- .NET 5 is the next major release after .NET Core 3.1. The word 'Core' is dropped from the name to emphasize that .NET 5 is the future of all earlier versions of .NET Core.

Recently, Microsoft has introduced .NET 6 also.

Q200. WHAT ARE RAZOR PAGES IN .NET CORE?

- Razor pages follows a **page-centric development** model just like ASP.NET web forms. It supports all the feature of ASP.NET Core.



PART II

MORE INTERVIEW QUESTIONS…

OOPS

1. WHAT IS THE DIFFERENCE BETWEEN A CLASS AND A STRUCTURE?

- CLASS: User-defined blueprint from which objects are created. It consists of methods or set of instructions that are to be performed on the objects.

- STRUCTURE: A structure is basically a user-defined collection of variables which are of different data types.

---

2. WHAT IS THE DIFFERENCE BETWEEN A CLASS AND AN OBJECT?

a) Class is a blueprint or template from which objects are created.
b) Object is an instance of a class.

a) Class is a group of similar objects.
b) Object is a real world entity such as pen, laptop, mobile, bed, keyboard, mouse, chair etc.

a) Class is a logical entity.
b) Object is a physical entity.

---

3. WHAT IS POLYMORPHISM? WHAT IS STATIC AND DYNAMIC POLYMORPHISM

Polymorphism refers to the ability to exist in multiple forms. Multiple definitions can be given to a single interface. For example, if you have a class named Vehicle, it can have a method named speed but you cannot define it because different vehicles have different speed. This method will be defined in the subclasses with different definitions for different vehicles.

Static polymorphism (static binding) is a kind of polymorphism that occurs at compile time. An example of compile-time polymorphism is method overloading.

Runtime polymorphism or dynamic polymorphism (dynamic binding) is a type of polymorphism which is resolved during runtime. An example of runtime polymorphism is method overriding.

---

4. WHAT IS OPERATOR OVERLOADING?

Operator overloading refers to implementing operators using user-defined types based on the arguments passed along with it.

---

5. CAN YOU CALL THE BASE CLASS METHOD WITHOUT CREATING AN INSTANCE?

- Yes, you can call the base class without instantiating it if:
- It is a static method.
- The base class is inherited by some other subclass.

---

6. WHAT ARE VIRTUAL FUNCTIONS AND PURE VIRTUAL FUNCTIONS?

Virtual functions are functions that are present in the parent class and are overridden by the subclass. These functions are used to achieve runtime polymorphism.

Pure virtual functions or abstract functions are functions that are only declared in the base class. This means that they do not contain any definition in the base class and need to be redefined in the subclass.

7. WHAT IS A DESTRUCTOR?

A destructor is a method that is automatically invoked when an object is destroyed. The destructor also recovers the heap space that was allocated to the destroyed object, closes the files and database connections of the object, etc.

---

8. WHAT IS A COPY CONSTRUCTOR?

A copy constructor creates objects by copying variables from another object of the same class. The main aim of a copy constructor is to create a new object from an existing one.

---

9. DO WE REQUIRE A PARAMETER FOR CONSTRUCTORS?

No, we do not need a parameter for constructors.

---

10. WHICH OOPS CONCEPT EXPOSES ONLY THE NECESSARY INFORMATION TO THE CALLING FUNCTIONS?

Encapsulation

---

11. IS IT ALWAYS NECESSARY TO CREATE OBJECTS FROM CLASS?

No, it is possible to call a base class method if it is defined as a static method.

12. WHAT IS EARLY AND LATE BINDING?

Early binding refers to the task of values to variables during plan time, whereas late Binding refers to the assignment of values to variables during run time.

---

13. WHICH OOPS CONCEPT IS USED AS A REUSE MECHANISM?

Inheritance is the OOPS concept that can be used as a reuse mechanism.

C#

1. IN TRY BLOCK IF WE ADD RETURN STATEMENT WHETHER FINALLY BLOCK IS EXECUTED?

Yes. Finally block will still be executed in presence of return statement in try block.

---

2. WHAT YOU MEAN BY INNER EXCEPTION?

Inner exception is a property of exception class which will give you a brief insight of the exception i.e, parent exception and child exception details.

---

3. LIST OUT SOME OF THE EXCEPTIONS?

Below are some of the exceptions -

- NullReferenceException
- ArgumentNullException
- DivideByZeroException
- IndexOutOfRangeException
- InvalidOperationException
- StackOverflowException etc.

---

4. EXPLAIN HASHTABLE? HOW TO CHECK WHETHER HASH TABLE CONTAINS SPECIFIC KEY?

It is used to store the key/value pairs based on hash code of the key. Key will be used to access the element in the collection. For example,

```
Hashtable myHashtbl = new Hashtable();
myHashtbl.Add("1", "TestValue1");
myHashtbl.Add("2", "TestValue2");
```

Method – “ContainsKey” can be used to check the key in hash table. Below is the sample code for the same –

Eg: myHashtbl.ContainsKey("1");

---

5. WHAT IS THE DIFFERENCE BETWEEN LIST AND DICTIONARY?

List collection is a generic class and can store any data types to create a list. It only contains value not key.

Dictionary is a collection of keys and values in C#. Dictionary<TKey, TValue> is included in the System.Collection.Generics namespace.

---

6. WHAT IS THE DIFFERENCE BETWEEN STACK AND QUEUE COLLECTIONS?

A stack is a linear data structure in which elements can be inserted and deleted only from one side of the list, called the **top**. A stack follows the **LIFO** (Last In First Out) principle, i.e., the element inserted at the last is the first element to come out.

A queue is a linear data structure in which elements can be inserted only from one side of the list called **rear**, and the elements can be deleted only from the other side called the **front**. The queue data structure follows the **FIFO** (First In First Out) principle, i.e. the element inserted at first in the list, is the first element to be removed from the list.

---

7. EXPLAIN JAGGED ARRAYS?

If the elements of an array is an array then it's called as jagged array. The elements can be of different sizes and dimensions.

---

8. EXPLAIN MULTIDIMENSIONAL ARRAYS?

A multi-dimensional array in C# is an array that contains more than one rows to store the data. The multidimensional array can be declared by adding commas in the square brackets.

---

9. EXPLAIN INDEXERS?

Indexers are used for allowing the classes to be indexed like arrays. Indexers will resemble the property structure but only difference is indexer's accessors will take parameters. For example,

```
class MyCollection<T>
{
 private T[] myArr = new T[100];
 public T this[int t]
 {
 get
 {
 return myArr[t];
 }
 set
 {
 myArr[t] = value;
 }
 }
}
```

10. WHAT IS THE DIFFERENCE BETWEEN METHODS – "SYSTEM.ARRAY.CLONE()" AND "SYSTEM.ARRAY.COPYTO()"?

- "CopyTo()" method can be used to copy the elements of one array to other.
- "Clone()" method is used to create a new array to contain all the elements which are in the original array.

---

11. EXPLAIN NAMESPACES?

Namespaces are containers for the classes. We will use namespaces for grouping the related classes. "Using" keyword can be used for using the namespace in other namespace.

---

12. WHAT ARE VALUE TYPE AND REFERENCE TYPES?

Value types stored in a stack.

Here are the value types:

- Decimal, int, byte, enum, double, long, float

Reference types are stored in a heap.

Below are the list of reference types -

- Class, string, interface, object

---

13. CAN WE OVERRIDE PRIVATE VIRTUAL METHOD?

No. We can’t override private virtual methods as it is not accessible outside the class.

14.EXPLAIN CIRCULAR REFERENCE?

This is a situation where in, multiple resources are dependent on each other and this causes a lock condition and this makes the resource to be unused.

15.EXPLAIN OBJECT POOL?

Object pool is used to track the objects which are being used in the code. So object pool reduces the object creation overhead.

16.WHAT ARE THE TYPES OF DELEGATES?

Below are the uses of delegates -

- Single Delegate
- Multicast Delegate
- Generic Delegate

17.WHAT ARE THE THREE TYPES OF GENERIC DELEGATES?

Below are the three types of generic delegates -

- Func
- Action
- Predicate

18.WHAT ARE THE USES OF DELEGATES?

Below are the list of uses of delegates -

- Callback Mechanism
- Asynchronous Processing
- Abstract and Encapsulate method
- Multicasting

19.CAN WE USE DELEGATES FOR ASYNCHRONOUS METHOD CALLS?

Yes. We can use delegates for asynchronous method calls.

20.WHAT IS AN ESCAPE SEQUENCE? NAME SOME STRING ESCAPE SEQUENCES IN C#.

An Escape sequence is denoted by a backslash (\). The backslash indicates that the character that follows it should be interpreted literally or it is a special character. An escape sequence is considered as a single character.

21.WHAT ARE REGULAR EXPRESSIONS?

Regular expression is a template to match a set of input. The pattern can consist of operators, constructs or character literals. Regex is used for string parsing and replacing the character string.

22.WHY TO USE LOCK STATEMENT?

Lock will make sure one thread will not intercept the other thread which is running the part of code. So lock statement will make the thread wait, block till the object is being released.

23.WHAT IS “EXTERN” KEYWORD?

The `extern` modifier is used to declare a method that is implemented externally. A common use of the `extern` modifier is with the `DllImport` attribute when you are using Interop services to call into unmanaged code.

```
[DllImport("avifil32.dll")]
private static extern void AVIFileInit();
```

24.WHAT IS “SIZEOF” OPERATOR?

The **sizeof operator** returns the number of bytes occupied by a variable of a given type. The argument to the **sizeof operator** must be the name of an unmanaged type or a type parameter that is constrained to be an unmanaged type.

```
Console.WriteLine(sizeof(byte));  // output: 1
```

25.WHAT IS TERNARY OPERATOR?

The conditional operator ?:, also known as the ternary conditional operator, evaluates a Boolean expression and returns the result of one of the two

expressions, depending on whether the Boolean expression evaluates to true or false.

condition ? consequent : alternative

---

26. CAN WE USE "THIS" INSIDE A STATIC METHOD?

No. We can't use "this" in static method.

---

27. WHAT IS THE DIFFERENCE BETWEEN CTYPE AND DIRECTCAST?

- CType is used for conversion between type and the expression.
- Directcast is used for converting the object type which requires run time type to be the same as specified type.

---

28. WHICH STRING METHOD IS USED FOR CONCATENATION OF TWO STRINGS?

"Concat" method of String class is used to concatenate two strings. For example,

string.Concat(firstStr, secStr)

---

29. WHAT IS PARSING? HOW TO PARSE A DATE TIME STRING?

Parsing converts a string into another data type.

For Example:

```
string text = "500";
int num = int.Parse(text);
```

500 is an integer. So, the Parse method converts the string 500 into its own base type, i.e int.

Follow the same method to convert a DateTime string.

```
string dateTime = "Jan 1, 2018";
DateTime parsedValue = DateTime.Parse(dateTime);
```

30.EXPLAIN PARTIAL CLASS?

Partial classes concept added in .Net Framework 2.0 and it allows us to split the business logic in multiple files with the same class name along with "partial" keyword.

31.EXPLAIN ANONYMOUS TYPE?

This is being added 3.0 version. This feature enables us to create an object at compile time. Below is the sample code for the same –

```
Var myTestCategory = new { CategoryId = 1, CategoryName = "Category1"};
```

32.EXPLAIN GET AND SET ACCESSOR PROPERTIES?

Get and Set are called Accessors. These are made use by Properties. The property provides a mechanism to read, write the value of a private field. For accessing that private field, these accessors are used.

Get Property is used to return the value of a property
Set Property accessor is used to set the value.

33.WHAT IS THE DIFFERENCE BETWEEN VAR AND DYNAMIC IN C#?

VAR TYPE - VAR are those variables which are declared without specifying the *.NET type* explicitly. In implicitly typed variable, the type of the variable is automatically deduced at compile time by the compiler from the value used to initialize the variable.

```
var a = 'f';
```

DYNAMIC TYPE - It is used to avoid the compile-time type checking. The compiler does not check the type of the dynamic type variable at compile time, instead of this, the compiler gets the type at the run time. The dynamic type variable is created using dynamic keyword.

```
dynamic val1 = "Interview";
```

34.WHAT IS COVARIANCE IN C#?

Covariance enables you to pass a derived type where a base type is expected.

Small sml = new Bigger();

.NET FRAMEWORK

1. WHAT IS JIT?

JIT stands for **Just In Time**. JIT is a compiler that converts Intermediate Language to a Native code.

The code is converted into Native language during execution. Native code is nothing but hardware specifications that can be read by the CPU. The native code can be stored so that it is accessible for subsequent calls.

---

2. WHAT IS MSIL?

MSIL stands for Microsoft Intermediate Language.

MSIL provides instructions for calling methods, initializing and storing values, operations such as memory handling, exception handling and so on. All .Net codes are first compiled to IL.

---

3. WHAT IS MEANT BY MANAGED AND UNMANAGED CODE?

The code that is managed by the CLR is called **Managed code**. This code runs inside the CLR. Hence, it is necessary to install the .Net framework in order to execute the managed code. CLR manages the memory through garbage collection and also uses the other features like CAS and CTS for efficient management of the code.

Unmanaged code is any code that does not depend on CLR for execution. It means it is developed by any other language independent of .Net framework. It uses its own runtime environment for compiling and execution.

Though it is not running inside the CLR, the unmanaged code will work properly if all the other parameters are correctly followed.

---

4. WHAT IS DIFFERENCE BETWEEN NAMESPACE AND ASSEMBLY?

Following are the differences between namespace and assembly:

• Assembly is physical grouping of logical units, Namespace, logically groups

classes.

• Namespace can span multiple assembly.

---

5. WHAT IS MANIFEST?

Assembly metadata is stored in Manifest. Manifest contains all the metadata needed to do the

following things (See Figure Manifest View for more details):

• Version of assembly.

• Security identity.

• Scope of the assembly.

• Resolve references to resources and classes.

---

6. WHERE IS VERSION INFORMATION STORED OF AN ASSEMBLY?

Version information is stored in assembly inside the manifest.

7. IS VERSIONING APPLICABLE TO PRIVATE ASSEMBLIES?

Versioning concept is only applicable to global assembly cache (GAC) as private assembly lie in

their individual folders. This does not mean versioning is not needed , you can still version it to

have better version control on the project.

---

8. WHAT IS THE APPLICATION DOMAIN?

An Application Domain is a logical container for a set of assemblies in which an executable is hosted. As you have seen, a single process may contain multiple Application Domains, each of which is hosting a .NET executable.

---

9. EXPLAIN CAS (CODE ACCESS SECURITY).

.Net provides a security model that prevents unauthorized access to resources. CAS is a part of that security model. CAS is present in the CLR. It enables the users to set permissions at a granular level for the code.

---

10. WHAT IS THE USE OF 'FINALIZE'?

Finalize as an object method used to free up unmanaged resources and cleanup before Garbage Collection(GC). It performs memory management tasks.

(THREADING)

11. WHAT ARE SYNCHRONOUS AND ASYNCHRONOUS OPERATIONS?

1. Synchronization is a way to create a thread-safe code where only one thread can access the resource at any given time. The asynchronous call waits for the method to complete before continuing with the program flow.
2. Synchronous programming badly affects the UI operations when the user tries to perform time-consuming operations since only one thread will be used. In Asynchronous operation, the method call will immediately return so that the program can perform other operations while the called method completes its work in certain situations.

12.WHAT IS MULTI-TASKING?

It is a feature of modern operating systems with which we can run multiple programs at same

time example Word, Excel etc.

13.NAME SOME PROPERTIES OF THREAD CLASS.

Few Properties of thread class are:

- **IsAlive** – contains value True when a thread is Active.
- **Name** – Can return the name of the thread. Also, can set a name for the thread.
- **Priority** – returns the prioritized value of the task set by the operating system.
- **IsBackground** – gets or sets a value which indicates whether a thread should be a background process or foreground.
- **ThreadState**– describes the thread state.

14. WHAT ARE THE DIFFERENT STATES OF A THREAD?

Different states of a thread are:

- **Unstarted** – Thread is created.
- **Running** – Thread starts execution.
- **WaitSleepJoin** – Thread calls sleep, calls wait on another object and calls join on another thread.
- **Suspended** – Thread has been suspended.
- **Aborted** – Thread is dead but not changed to state stopped.
- **Stopped** – Thread has stopped.

---

15. CAN WE HAVE MULTIPLE THREADS IN ONE APP DOMAIN?

One or more threads run in an AppDomain. An AppDomain is a runtime representation of a

logical process within a physical process. Each AppDomain is started with a single thread, but

can create additional threads from any of its threads.

---

16. WHICH NAMESPACE HAS THREADING?

'Systems.Threading' has all the classes related to implement threading.

---

17. EXPLAIN *LOCK*, *MONITORS*, AND *MUTEX* OBJECT IN THREADING.

Lock keyword ensures that only one thread can enter a particular section of the code at any given time. In the above **Example**, lock(ObjA) means the lock is placed on ObjA until this process releases it, no other thread can access ObjA.

Mutex is also like a lock but it can work across multiple processes at a time. WaitOne() is used to lock and ReleaseMutex() is used to release the lock. But Mutex is slower than lock as it takes time to acquire and release it.

Monitor.Enter and Monitor.Exit implements lock internally. a lock is a shortcut for Monitors. lock(objA) internally calls.

```
Monitor.Enter(ObjA);

try

{

}

Finally {Monitor.Exit(ObjA));}
```

---

18. WHAT IS THREAD.SLEEP () IN THREADING?

Thread's execution can be paused by calling the Thread.Sleep method. This method takes an

integer value that determines how long the thread should sleep. Example

```
Thread.CurrentThread.Sleep(2000).
```

---

19. WHAT IS SUSPEND AND RESUME IN THREADING?

Suspend() method is called to suspend the thread. Resume() method is called to resume the suspended thread. Start() method is used to send a thread into runnable State.

20.WHAT THE WAY TO STOP A LONG RUNNING THREAD?

Thread.Abort() stops the thread execution at that moment itself.

---

21. WHY WE NEED MULTI-THREADING IN OUR PROJECT?

Multi-threading is running the multiple threads simultaneously. Some main advantages are:

- You can do multiple tasks simultaneously. For e.g. saving the details of user to a file while at the same time retrieving something from a web service.
- Threads are much lightweight than process. They don't get their own resources. They used the resources allocated to a process.
- Context-switch between threads takes less time than process.

---

22. HOW TO START A THREAD IN C#?

We have to use the Thread class provided by System.Threading namespace. In the constructor of the class, we have to pass the method name which we want to run in separate thread. After than we have to call the start method of Thread class. Below is the example.

---

23. WHAT IS RACE CONDITION?

A race condition happens when two or more threads want to update shared data at the same time.

---

24. WHAT IS LIVELOCK?

A livelock is very similar to deadlock except involved threads states are continually changing their state but still they cannot complete their work.

A real-world example of livelock occurs when two people meet in a narrow corridor, and each tries to be polite by moving aside to let the other pass, but they end up swaying from side to side without making any progress because they both repeatedly move the same way at the same time.

---

25. WHAT IS IMMUTABLE OBJECT?

An immutable object is an object which states cannot be changed after creation.

Immutable objects are useful in concurrent programming as they are thread-safe.

"String" objects are examples of immutable object because we can cannot change the value of string after it is created.

---

26. HOW MANAGED CODE IS EXECUTED?

Managed code is a code whose execution is managed by Common Language Runtime. It gets the managed code and compiles it into machine code. After that, the code is executed. The runtime here i.e. CLR provides automatic memory management, type safety, etc.

---

27. WHAT IS THE DIFFERENCE BETWEEN SYSTEM EXCEPTIONS AND APPLICATION EXCEPTIONS?

In general System exceptions occurred whenever some non-recoverable or fatal error is encountered, like a database crash, bound errors etc.

While in case of Application level exceptions some error which is recoverable is encountered, for instance, the wrong type of input data, arithmetic exceptions etc.

---

28. WHAT IS XSD?

XSD (XML Schema Definition), a recommendation of the World Wide Web Consortium (W3C), specifies how to formally describe the elements in an Extensible Markup Language (XML) document. It can be used by programmers to verify each piece of item content in a document.

ASP.NET MVC

1. IN WHICH ASSEMBLY IS THE MVC FRAMEWORK DEFINED?

System.Web.Mvc

---

2. IS IT POSSIBLE TO SHARE A VIEW ACROSS MULTIPLE CONTROLLERS?

Yes, put the view into the shared folder. This will automatically make the view available across multiple controllers.

---

3. WHAT IS ROUTE IN MVC? WHAT IS DEFAULT ROUTE IN MVC?

A route is a URL pattern that is mapped to a handler. The handler can be a physical file, such as a .aspx file in a Web Forms application. A handler can also be a class that processes the request, such as a controller in an MVC application. To define a route, you create an instance of the Route class by specifying the URL pattern, the handler, and optionally a name for the route.

You add the route to the application by adding the Route object to the static Routes property of the RouteTable class. The Routesproperty is a RouteCollection object that stores all the routes for the application.

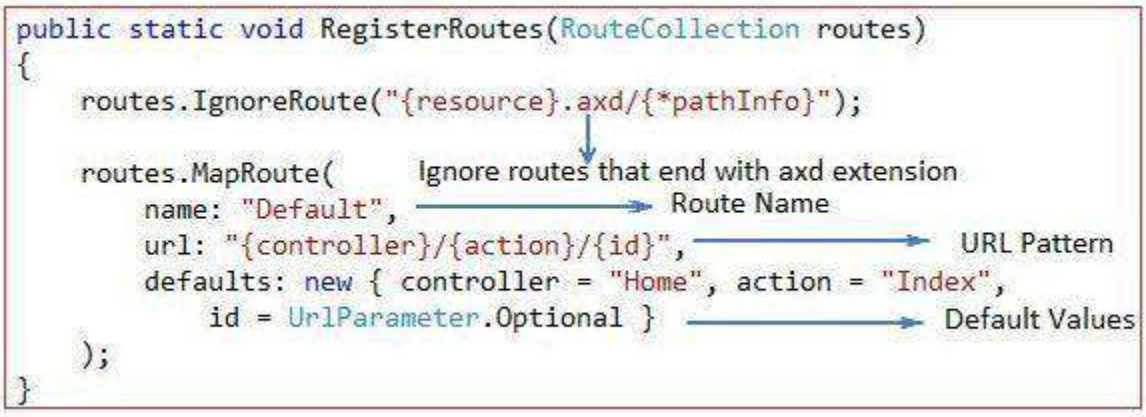
You typically do not have to write code to add routes in an MVC application. Visual Studio project templates for MVC include preconfigured URL routes. These are defined in the MVC Application class, which is defined in the Global.asax file.

Default Route

The default ASP.NET MVC project templates add a generic route that uses the following URL convention to break the URL for a given request into three named segments.

URL: "{controller}/{action}/{id}"

This route pattern is registered via a call to the MapRoute() extension method of RouteCollection.



---

4. WHERE ARE THE ROUTING RULES DEFINED IN AN ASP.NET MVC APPLICATION?

In Application_Start event in Global.asax

---

5. WHAT IS VALIDATION SUMMARY IN MVC?

The ValidationSummary helper method generates an unordered list (ul element) of validation messages that are in the ModelStateDictionary object.

The ValidationSummary can be used to display all the error messages for all the fields. It can also be used to display custom error messages.

---

6. DIFFERENCES BETWEEN RAZOR AND ASPX VIEW ENGINE IN MVC?

Razor View Engine VS ASPX View Engine

The Razor View Engine is a bit slower than the ASPX View Engine.

Conclusion

Razor provides a new view engine with a streamlined code for focused templating. Razor's syntax is very compact and improves the readability of the markup and code. By default, MVC supports ASPX (web forms) and Razor View Engine. MVC also supports third-party view engines like Spark, Nhaml, NDjango, SharpDOM and so on. ASP.NET MVC is open source.

---

7. WHAT ARE THE MAIN RAZOR SYNTAX RULES?

- Razor code blocks are enclosed in @{ ... }
- Inline expressions (variables and functions) start with @
- Code statements end with semicolon
- Variables are declared with the var keyword

- Strings are enclosed with quotation marks
- C# code is case sensitive
- C# files have the extension .cshtml

---

8. HOW DO YOU IMPLEMENT FORMS AUTHENTICATION IN MVC?

Authentication is giving access to the user for a specific service by verifying his/her identity using his/her credentials like username and password or email and password. It assures that the correct user is authenticated or logged in for a specific service and the right service has been provided to the specific user based on their role that is nothing but authorization.

ASP.NET forms authentication occurs after IIS authentication is completed. You can configure forms authentication by using forms element with in web.config file of your application. The default attribute values for forms authentication are shown below,

```
<system.web>
  <authenticationmode="Forms">
    <formsloginUrl="Login.aspx" protection="All" timeout="30" name=
".ASPXAUTH" path="/" requireSSL="false" slidingExpiration="true" defa
ultUrl="default.aspx" cookieless="UseDeviceProfile" enableCrossApp
Redirects="false" />
    </authentication>
</system.web>
```

The FormsAuthentication class creates the authentication cookie automatically when SetAuthCookie() or RedirectFromLoginPage() methods are called. The value of authentication cookie contains a string representation of the encrypted and signed FormsAuthenticationTicket object.

9. HOW WE CAN REGISTER THE AREA IN ASP.NET MVC?

When we have created an area make sure this will be registered in "Application_Start" event in Global.asax. Below is the code snippet where area registration is done :

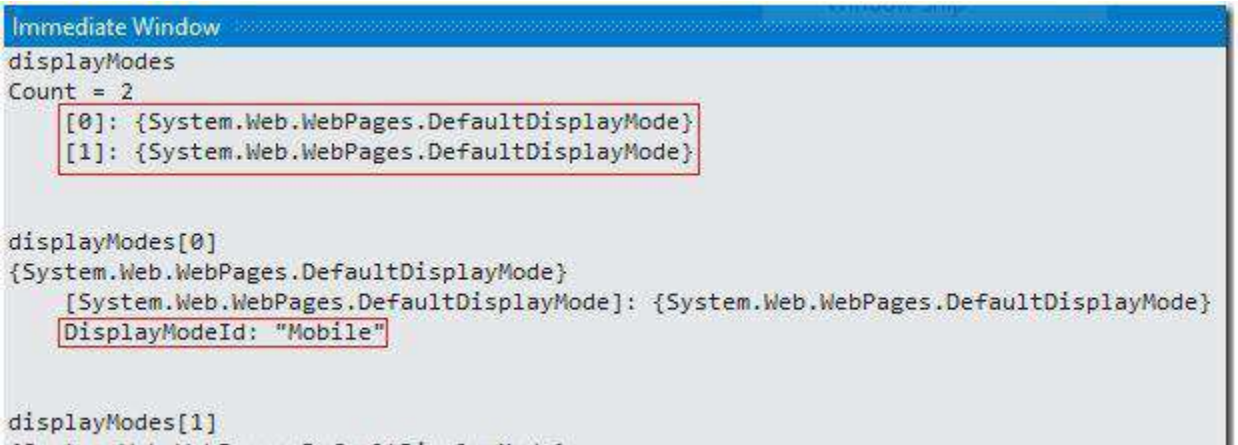
```
protected void Application_Start()
{
AreaRegistration.RegisterAllAreas();
}
```

10.EXPLAIN THE NEED OF DISPLAY MODE IN MVC?

DisplayModes give you another level of flexibility on top of the default capabilities we saw in the last section. DisplayModes can also be used along with the previous feature so we will simply build off of the site we just created.

Using display modes involves in 2 steps

1. We should register Display Mode with a suffix for particular browser using "DefaultDisplayMode"e class inApplication_Start() method in the Global.asax file.
2. View name for particular browser should be appended with suffix mentioned in first step.



1. Desktop browsers (without any suffix. e.g.: Index.cshtml, _Layout.cshtml).
2. Mobile browsers (with a suffix “Mobile”. e.g.: Index.Mobile.cshtml,Layout.Mobile.cshtml)

If you want design different pages for different mobile device browsers (any different browsers) and render them depending on the browser requesting. To handle these requests you can register custom display modes. We can do that using DisplayModeProvider.Instance.Modes.Insert(int index, IDisplayMode item) method.

11.WHAT IS ROUTE CONSTRAINTS IN MVC?

Routing is a great feature of MVC, it provides a REST based URL that is very easy to remember and improves page ranking in search engines.

This article is not an introduction to Routing in MVC, but we will learn a few features of routing and by implementing them we can develop a very flexible and user-friendly application. So, let's start without wasting valuable time.

Add constraint to URL

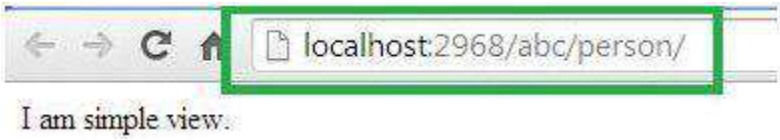
This is very necessary for when we want to add a specific constraint to our URL. Say, for example we want a URL.

So, we want to set some constraint string after our host name. Fine, let's see how to implement it.

It's very simple to implement, just open the RouteConfig.cs file and you will find the routing definition in that. And modify the routing entry as in the following. We will see that we have added “abc” before.

```
routes.MapRoute(
    name: "Default",
    url: "abc/{controller}/{action}/{id}",
    defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }
);
```

Controller name, now when we browse we need to specify the string in the URL, as in the following:



12.WHAT ARE THE FOLDERS IN MVC APPLICATION SOLUTIONS?

Understanding the folders

When you create a project a folder structure gets created by default under the name of your project which can be seen in solution explorer. Below i will give you a brief explanation of what these folders are for.

Model

This folder contains classes that is used to provide data. These classes can contain data that is retrived from the database or data inserted in the form by the user to update the database.

Controllers

These are the classes which will perform the action invoked by the user. These classes contains methods known as "Actions" which responds to the user action accordingly.

Views

These are simple pages which uses the model class data to populate the HTML controls and renders it to the client browser.

App_Start

Contains Classes such as FilterConfig, RoutesConfig, WebApiConfig. As of now we need to understand the RouteConfig class. This class contains the default format of the url that should be supplied in the browser to navigate to a specified page.

13.WHAT ARE THE METHODS OF HANDLING AN ERROR IN MVC?

Exception handling may be required in any application, whether it is a web application or a Windows Forms application.

ASP.Net MVC has an attribute called "HandleError" that provides built-in exception filters. The HandleError attribute in ASP.NET MVC can be applied over the action method as well as Controller or at the global level. The HandleError attribute is the default implementation of IExceptionFilter. When we create a MVC application, the HandleError attribute is added within the Global.asax.cs file and registered in the Application_Start event.

```
1. public static void RegisterGlobalFilters(GlobalFilterCollection filters)
2. {
3.     filters.Add(new HandleErrorAttribute());
4. }
5. protected void Application_Start()
6. {
7.     AreaRegistration.RegisterAllAreas();
8.     RegisterGlobalFilters(GlobalFilters.Filters);
9.     RegisterRoutes(RouteTable.Routes);
10.}
```

Important properties of HandleError attribute

The HandleError Error attribute has a couple for properties that are very useful in handling the exception.

ExceptionType

Type of exception to be catch. If this property is not specified then the HandleError filter handles all exceptions.

View

Name of the view page for displaying the exception information.

Master

Master View for displaying the exception.

Order

Order in which the action filters are executed. The Order property has an integer value and it specifies the priority from 1 to any positive integer value. 1 means highest priority and the greater the value of the integer is, the lower is the priority of the filter.

AllowMultiple

It indicates whether more than one instance of the error filter attribute can be specified.

Example

```
[HandleError(View = "Error")]
public class HomeController: Controller
{
  public ActionResult Index()
  {
```

```
        ViewBag.Message = "Welcome to ASP.NET MVC!";
        int u = Convert.ToInt32(""); // Error line
        return View();
      }
    }
```

HandleError Attribute at Action Method Level,

```
    [HandleError(View = "Error")]
    public ActionResult Index()
    {
      ViewBag.Message = "Welcome to ASP.NET MVC!";
      int u = Convert.ToInt32(""); // Error line
      return View();
    }
```

14.WHAT IS VIEWSTART?

Razor View Engine introduced a new layout named _ViewStart which is applied on all view automatically. Razor View Engine firstly executes the _ViewStart and then start rendering the other view and merges them.

Example of Viewstart

```
    @ {
      Layout = "~/Views/Shared/_v1.cshtml";
    } < !DOCTYPE html >
      < html >
```

```
< head >
< meta name = "viewport"
content = "width=device-width" / >
< title > ViewStart < /title> < /head> < body >
< /body> < /html>
```

15.HOW TO USE VIEWBAG?

ViewBag is dynamic property that takes advantage of new dynamic features in C# 4.0. It's also used to pass data from a controller to a view. In short, The ViewBag property is simply a wrapper around the ViewData that exposes the ViewData dictionary as a dynamic object. Now create an action method "StudentSummary" in the "DisplayDataController" controller that stores a Student class object in ViewBag.

```
1.  public ActionResult StudentSummary()
2.  {
3.     var student = new Student()
4.     {
5.        Name = "Sandeep Singh Shekhawat",
6.           Age = 24,
7.           City = "Jaipur"
8.     };
9.     ViewBag.Student = student;
10.    return View();
11. }
```

Thereafter create a view StudentSummary ("StudentSummary.cshtml") that shows student object data. ViewBag does not require typecasting for complex data type so you can directly access the data from ViewBag.

```
1.  @ {
```

```
    ViewBag.Title = "Student Summary";
    var student = ViewBag.Student;
}
< table >
    < tr >
    < th > Name < /th> < th > Age < /th> < th > City < /th> < /tr> < tr >
    < td > @student.Name < /td> < td > @student.Age < /td> < td > @student
.City < /td> < /tr>
< /table>
```

Here we used one more thing, "ViewBag.Title", that shows the title of the page.

16.HOW CAN WE DONE CUSTOM ERROR PAGE IN MVC?

The HandleErrorAttribute allows you to use a custom page for this error. First you need to update your web.config file to allow your application to handle custom errors.

```
<system.web>
    <customErrors mode="On">
</system.web>
```

Then, your action method needs to be marked with the atttribute.

```
[HandleError]
public class HomeController: Controller
{
    [HandleError]
    publicActionResultThrowException()
    {
        throw new ApplicationException();
    }
}
```

By calling the ThrowException action, this would then redirect the user to the default error page. In our case though, we want to use a custom error page and redirect the user there instead.So, let's create our new custom view page.

```
<%@ Page Language="C#" Inherits="System.Web.Mvc.ViewPage" %>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://
<html xmlns="http://www.w3.org/1999/xhtml">
<head runat="server">
    <title>CustomErrorView</title>
</head>
<body>
    <h2>
        Error</h2>
    <p>
        Controller:
        <%=((HandleErrorInfo)ViewData.Model).ControllerName %>
    </p>
    <p>
        Action:
        <%=((HandleErrorInfo)ViewData.Model).ActionName %>
    </p>
    <p>
        Message:
        <%=((HandleErrorInfo)ViewData.Model).Exception.Message %>
    </p>
    <p>
        Stack Trace:
        <%=((HandleErrorInfo)ViewData.Model).Exception.StackTrace %>
    </p>
</body>
</html>
```

Next, we simply need to update the HandleErrorAttribute on the action method.

```
1. [HandleError]
2. public class HomeController: Controller
3. {
4.     [HandleError(View = "CustomErrorView")]
5.     publicActionResultThrowException()
6.     {
7.         throw new ApplicationException();
```

```
8.     }
9.  }
```

17.WHAT IS SERVER SIDE VALIDATION IN MVC?

The ASP.NET MVC Framework validates any data passed to the controller action that is executing, It populates a ModelState object with any validation failures that it finds and passes that object to the controller. Then the controller actions can query the ModelState to discover whether the request is valid and react accordingly.

I will use two approaches in this article to validate a model data. One is to manually add an error to the ModelState object and another uses the Data Annotation API to validate the model data.

Approach 1 - Manually Add Error to ModelState object

I create a User class under the Models folder. The User class has two properties "Name" and "Email". The "Name" field has required field validations while the "Email" field has Email validation. So let's see the procedure to implement the validation. Create the User Model as in the following,

```
1.  namespace ServerValidation.Models
2.  {
3.     public class User
4.     {
5.        public string Name
6.        {
```

```
        get;
        set;
    }
    public string Email
    {
        get;
        set;
    }
  }
}
```

After that I create a controller action in User Controller (UserController.cs under Controllers folder). That action method has logic for the required validation for Name and Email validation on the Email field. I add an error message on ModelState with a key and that message will be shown on the view whenever the data is not to be validated in the model.

```
using System.Text.RegularExpressions;
using System.Web.Mvc;
namespace ServerValidation.Controllers
{
  public class UserController: Controller
  {
    public ActionResult Index()
    {
        return View();
    }
    [HttpPost]
    public ActionResult Index(ServerValidation.Models.User model)
    {

      if (string.IsNullOrEmpty(model.Name))
      {
        ModelState.AddModelError("Name", "Name is required");
      }
      if (!string.IsNullOrEmpty(model.Email))
      {
```

```
21.          string emailRegex = @ "^([a-zA-Z0-9_\-\.]+)@((\[[0-
   9]{1,3}" +
22.            @ "\.[0-9]{1,3}\.[0-9]{1,3}\.)|(([a-zA-Z0-9\-]
   +\" +
23.          @ ".)+))([a-zA-Z]{2,4}|[0-9]{1,3})(\]?)$";
24.          Regex re = new Regex(emailRegex);
25.          if (!re.IsMatch(model.Email))
26.          {
27.             ModelState.AddModelError("Email", "Email is not val
   id");
28.          }
29.       } else {
30.          ModelState.AddModelError("Email", "Email is required")
   ;
31.       }
32.       if (ModelState.IsValid)
33.       {
34.          ViewBag.Name = model.Name;
35.          ViewBag.Email = model.Email;
36.       }
37.       return View(model);
38.    }
39. }
40.}
```

Thereafter I create a view (Index.cshtml) for the user input under the User folder.

```
1. @model ServerValidation.Models.User
2. @ {
3.    ViewBag.Title = "Index";
4. }
5. @using(Html.BeginForm())
6. {
7.    if (@ViewData.ModelState.IsValid)
8.       {
9.       if (@ViewBag.Name != null)
10.   { < b >
```

```
11.             Name: @ViewBag.Name < br / >
12.             Email: @ViewBag.Email < /b>
13.       }
14.    } < fieldset >
15.       < legend > User < /legend>  < div class = "editor-label" >
16.       @Html.LabelFor(model => model.Name) < /div> < div class = "edi
   tor-field" >
17.       @Html.EditorFor(model => model.Name)
18.    @if(!ViewData.ModelState.IsValid)
19. {
20. < span class = "field-validation-error" > @ViewData.ModelState
   ["Name"].Errors[0].ErrorMessage < /span>
21.
22. }
23. < /div>  < div class = "editor-label" >
24.
25.       @Html.LabelFor(model => model.Email) < /div> < div class = "edi
   tor-field" >
26.       @Html.EditorFor(model => model.Email)
27.    @if(!ViewData.ModelState.IsValid)
28.  {
29. < span class = "field-validation-error" > @ViewData.ModelState
   ["Email"].Errors[0].ErrorMessage < /span>
30.  }
31. < /div> < p >
32.       < input type = "submit"
33.    value = "Create" / >
34.       < /p> < /fieldset>
35. }
```

18.WHAT IS THE USE OF REMOTE VALIDATION IN MVC?

Remote validation is the process where we validate specific data posting data to a server without posting the entire form data to the server. Let's see an actual scenario, in one of my projects I had a requirement to validate an email address, whetehr it already exists in the database. Remote validation was useful for that; without posting all the data we can validate only the email address supplied by the user.

Practical Explanation

Let's create a MVC project and name it accordingly, for me its "TestingRemoteValidation". Once the project is created let's create a model named UserModel that will look like:

```
1.  public class UserModel
2.  {
3.     [Required]
4.     public string UserName
5.     {
6.        get;
7.        set;
8.     }
9.     [Remote("CheckExistingEmail", "Home", ErrorMessage = "Email
   already exists!")]
10.    public string UserEmailAddress
11.    {
12.       get;
13.       set;
14.    }
15. }
```

Let's get some understanding of the remote attribute used, so the very first parameter "CheckExistingEmail" is the the name of the action. The second parameter "Home" is referred to as controller so to validate the input for the UserEmailAddress the "CheckExistingEmail" action of the "Home" controller is

called and the third parameter is the error message. Let's implement the “CheckExistingEmail” action result in our home controller.

```
1. public ActionResult CheckExistingEmail(string UserEmailAddress)
2. {
3.    bool ifEmailExist = false;
4.    try
5.    {
6.       ifEmailExist = UserEmailAddress.Equals("mukeshknayak@gmail.com") ? true : false;
7.       return Json(!ifEmailExist, JsonRequestBehavior.AllowGet);
8.    } catch (Exception ex)
9.    {
10.      return Json(false, JsonRequestBehavior.AllowGet);
11.   }
12.}
```

19.EXPLAIN RENDERSECTION IN MVC?

RenderSection() is a method of the WebPageBase class. Scott wrote at one point, The first parameter to the "RenderSection()" helper method specifies the name of the section we want to render at that location in the layout template. The second parameter is optional, and allows us to define whether the section we are rendering is required or not. If a section is "required", then Razor will throw an error at runtime if that section is not implemented within a view template that is based on the layout file (that can make it easier to track down content errors). It returns the HTML content to render.

```
1. <div id="body">
2.    @RenderSection("featured", required: false)
3.    <section class="content-wrapper main-content clear-fix">

4.       @RenderBody()
```

```
5.      </section>
6.  </div>
```

20.WHAT IS THE SIGNIFICANCE OF NONACTIONATTRIBUTE?

In general, all public methods of a controller class are treated as action methods. If you want prevent this default behavior, just decorate the public method with NonActionAttribute.

21.HOW ROUTE TABLE IS CREATED IN ASP.NET MVC?

When an MVC application first starts, the Application_Start() method is called. This method, in turn, calls the RegisterRoutes() method. The RegisterRoutes() method creates the route table.

22.ASP.NET MVC APPLICATION, MAKES USE OF SETTINGS AT 2 PLACES FOR ROUTING TO WORK CORRECTLY. WHAT ARE THESE 2 PLACES?

1. Web.Config File : ASP.NET routing has to be enabled here.
   2. Global.asax File : The Route table is created in the application Start event handler, of the Global.asax file.

23.WHAT IS THE USE OF THE FOLLOWING DEFAULT ROUTE?

{resource}.axd/{*pathInfo}
This route definition, prevent requests for the Web resource files such as WebResource.axd or ScriptResource.axd from being passed to a controller.

24. WHAT IS THE DIFFERENCE BETWEEN ADDING ROUTES, TO A WEBFORMS APPLICATION AND TO AN MVC APPLICATION?

To add routes to a webforms application, we use MapPageRoute() method of the RouteCollection class, where as to add routes to an MVC application we use MapRoute() method.

---

25. IF I HAVE MULTIPLE FILTERS IMPLEMENTED, WHAT IS THE ORDER IN WHICH THESE FILTERS GET EXECUTED?

1. Authorization filters
2. Action filters
3. Response filters
4. Exception filters

---

26. WHICH FILTER EXECUTES FIRST IN AN ASP.NET MVC APPLICATION?

Authorization filter

---

27. WHAT ARE THE LEVELS AT WHICH FILTERS CAN BE APPLIED IN AN ASP.NET MVC APPLICATION?

1. Action Method
2. Controller
3. Application

28.IS IT POSSIBLE TO CREATE A CUSTOM FILTER?

Yes

29.WHAT FILTERS ARE EXECUTED IN THE END?

Exception Filters

30.IS IT POSSIBLE TO CANCEL FILTER EXECUTION?

Yes

31.WHAT TYPE OF FILTER DOES OUTPUTCACHEATTRIBUTE CLASS REPRESENTS?

Result Filter

32.WHAT ARE THE 2 POPULAR ASP.NET MVC VIEW ENGINES?

1. Razor
2. .aspx

33.WHAT SYMBOL WOULD YOU USE TO DENOTE, THE START OF A CODE BLOCK IN RAZOR VIEWS?

@

34. WHAT SYMBOL WOULD YOU USE TO DENOTE, THE START OF A CODE BLOCK IN ASPX VIEWS?

<%= %>

In razor syntax, what is the escape sequence character for @ symbol?
The escape sequence character for @ symbol, is another @ symbol

---

35. WHAT ARE SECTIONS?

Layout pages, can define sections, which can then be overriden by specific views making use of the layout. Defining and overriding sections is optional.

---

36. WHAT ARE THE FILE EXTENSIONS FOR RAZOR VIEWS?

1. .cshtml – If the programming lanugaue is C#
2. .vbhtml – If the programming lanugaue is VB

---

37. HOW DO YOU SPECIFY COMMENTS USING RAZOR SYNTAX?

Razor syntax makes use of @* to indicate the begining of a comment and *@ to indicate the end.

---

38. IS IT POSSIBLE TO COMBINE ASP.NET WEBFORMS AND ASP.MVC AND DEVELOP A SINGLE WEB APPLICATION?

Yes, it is possible to combine ASP.NET webforms and ASP.MVC and develop a single web application.

---

39. WHAT IS THE USE OF VIEWMODEL IN MVC?

ViewModel is a plain class with properties, which is used to bind it to strongly typed view. ViewModel can have the validation rules defined for its properties using data annotations.

---

40. EXPLAIN BUNDLE.CONFIG IN MVC4?

“BundleConfig.cs” in MVC4 is used to register the bundles by the bundling and minification system. Many bundles are added by default including jQuery libraries like – jquery.validate, Modernizr, and default CSS references.

---

41. HOW WE CAN HANDLE THE EXCEPTION AT CONTROLLER LEVEL IN ASP.NET MVC?

Exception Handling is made simple in ASP.Net MVC and it can be done by just overriding “OnException” and set the result property of the filtercontext object (as shown below) to the view detail, which is to be returned in case of exception.

```
protected overrides void OnException(ExceptionContext filterContext)
{
}
```

---

42. DOES TEMPDATA HOLD THE DATA FOR OTHER REQUEST IN ASP.NET MVC?

If Tempdata is assigned in the current request then it will be available for the current request and the subsequent request and it depends whether data in TempData read or not. If data in Tempdata is read then it would not be available for the subsequent requests.

43.EXPLAIN KEEP METHOD IN TEMPDATA IN ASP.NET MVC?

As explained above in case data in Tempdata has been read in current request only then “Keep” method has been used to make it available for the subsequent request.

```
@TempData[“TestData”];
TempData.Keep(“TestData”);
```

44.EXPLAIN PEEK METHOD IN TEMPDATA IN ASP.NET MVC?

Similar to Keep method we have one more method called “Peek” which is used for the same purpose. This method used to read data in Tempdata and it maintains the data for subsequent request.

```
string A4str = TempData.Peek(“TT”).ToString();
```

45.WHAT ARE CHILD ACTIONS IN ASP.NET MVC?

To create reusable widgets child actions are used and this will be embedded into the parent views. In ASP.Net MVC Partial views are used to have reusability in the application. Child action mainly returns the partial views.

46.EXPLAIN THE TOOLS USED FOR UNIT TESTING IN ASP.NET MVC?

Below are the tools used for unit testing :

NUnit
xUnit.NET
Ninject 2
Moq

47.CAN I USE RAZOR CODE IN JAVASCRIPT IN ASP.NET MVC?

Yes. We can use the razor code in javascript in cshtml by using <text> element.

```
< script type=”text/javascript”>
@foreach (var item in Model) {
< text >
//javascript goes here which uses the server values
< text >
}
< script>
```

48.HOW CAN I RETURN STRING RESULT FROM ACTION IN ASP.NET MVC?

Below is the code snippet to return string from action method :

```
public ActionResult TestAction() {
return Content(“Hello Test !!”);
}
```

49.HOW TO RETURN THE JSON FROM ACTION METHOD IN ASP.NET MVC?

Below is the code snippet to return string from action method :

```
public ActionResult TestAction() {
return JSON(new { prop1 = “Test1”, prop2 = “Test2” });
}
```

50.GIVE AN EXAMPLE FOR AUTHORIZATION FILTERS IN AN ASP.NET MVC APPLICATION?

1. RequireHttpsAttribute
2. AuthorizeAttribute

51.WHAT IS DATA ANNOTATION VALIDATOR ATTRIBUTES IN MVC?

Data Annotations are nothing but certain validations that we put in our models to validate the input from the user. ASP.NET ***MVC*** provides a unique feature in which we can validate the models using the ***Data Annotation attribute***. Import the following namespace to use ***data annotations*** in the application.

Example

```
[Required(ErrorMessage = "Please enter name"), MaxLength(30)]

[Display(Name = "Student Name")]

public string Name { get; set; }
```

52.WHAT ARE THE EXCEPTION FILTERS IN MVC?

Exception filter in MVC provides an ability to handle the exceptions for all the controller methods at a single location. This is by creating a class, which inherits from the FilterAttribute and IExceptionFilter interface.

---

53. WHICH APPROACH PROVIDES BETTER SUPPORT FOR TEST DRIVEN DEVELOPMENT – ASP.NET MVC OR ASP.NET WEBFORMS?

ASP.NET MVC

ASP.NET WEBFORMS

1. **WHAT'S THE USE OF RESPONSE.OUTPUT.WRITE()?**

We can write formatted output using Response.Output.Write().

---

2. **IN WHICH EVENT OF PAGE CYCLE IS THE VIEWSTATE AVAILABLE?**

After the Init() and before the Page_Load().

---

3. **FROM WHICH BASE CLASS ALL WEB FORMS ARE INHERITED?**

Page class.

---

4. **WHICH VALIDATOR CONTROL YOU USE IF YOU NEED TO MAKE SURE THE VALUES IN TWO DIFFERENT CONTROLS MATCHED?**

Compare Validator control.

---

5. WHAT IS VIEWSTATE?

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

---

6. HOW LONG THE ITEMS IN VIEWSTATE EXISTS?

They exist for the life of the current page.

---

7. 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

---

8. HOW YOU CAN ADD AN EVENT HANDLER?

Using the Attributes property of server side control.

e.g.

btnSubmit.Attributes.Add("onMouseOver","JavascriptCode();")

---

9. WHICH TYPE OF 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:

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

---

10. CAN WE HAVE A WEB APPLICATION RUNNING WITHOUT WEB.CONFIG FILE?

Yes

---

11. 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.

---

12. WHAT IS PROTECTED CONFIGURATION?

It is a feature used to secure connection string information.

---

13. WRITE CODE TO SEND E-MAIL FROM AN ASP.NET APPLICATION?

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.

---

14. HOW CAN WE PREVENT BROWSER FROM CACHING AN ASPX PAGE?

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

```
Response.Cache.SetNoStore ();
Response.Write (DateTime.Now.ToLongTimeString ());
```

---

15. 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.

---

16. 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

---

17. WHICH PROTOCOL IS USED TO CALL A WEB SERVICE?

HTTP Protocol

---

18. 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.

<AUTHORIZATION>< authorization >

< allow roles="Domain_Name\Administrators" / > < !-- Allow Administrators in domain. -- >

< deny users="*" / > < !-- Deny anyone else. -- >

< /authorization >

---

19. 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:

<configuration>

```
<system.web>
<pages theme="Windows7" />
</system.web>
</configuration>
```

---

20. 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.

---

21. 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

---

22. 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.

23. 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

---

24. HOW DO YOU REGISTER JAVASCRIPT FOR WEBCONTROLS ?

We can register javascript for controls using <CONTROL -name>Attribtues.Add(scriptname,scripttext) method.

---

25. 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.

---

26. LIST ALL TEMPLATES OF THE REPEATER CONTROL.

- ItemTemplate
- AlternatingltemTemplate
- SeparatorTemplate
- HeaderTemplate
- FooterTemplate

27. LIST THE MAJOR BUILT-IN OBJECTS IN ASP.NET?

- Application
- Request
- Response
- Server
- Session
- Context
- Trace

---

28. 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:

<configuration>

<appSettings>

<add key="ConnectionString" value="server=local; pwd=password; database=default" />

</appSettings>

---

29. WHICH NAMESPACES ARE NECESSARY TO CREATE A LOCALIZED APPLICATION?

System.Globalization

System.Resources

---

30. 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.

---

31. WHAT IS THE FILE EXTENSION OF WEB SERVICE?

Web services have file extension .asmx.

---

32. 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.

System.Web.Mvc

System.Web.Razor

System.ComponentModel.DataAnnotations

---

33. CAN WE HAVE MULTIPLE WEB CONFIG FILES FOR AN ASP.NET APPLICATION?

Yes, one can create different folders and create a config for the specific folder.

---

34. 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.

Web config fields will override machine configurations.

---

35. 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.

ADO.NET

1. WHAT IS OBJECT POOLING?

Object pooling is nothing but a repository of the objects in memory which can be used later. This object pooling reduces the load of object creation when it is needed. Whenever there is a need of object, object pool manager will take the request and serve accordingly.

---

2. WHAT IS CONNECTION POOLING?

Connection pooling consists of database connection so that the connection can be used or reused whenever there is request to the database. This pooling technique enhances the performance of executing the database commands. This pooling definitely reduces our time and effort.

---

3. WHAT IS THE DIFFERENCE BETWEEN DATAREADER AND DATASET?

a) Datareader is FORWARD and READ only
b) Dataset used to UPDATE records.

a) Datareader is CONNECTED architecture

b) Dataset is DISCONNECTED Recordset

a) Datareader contains single table
b) Datareader can contains multiple tables.

a) Datareader Occupies Less Memory
b) Dataset Occupies More memory

---

4. WHAT ARE ALL COMPONENTS OF ADO.NET DATA PROVIDER?

Following are the components of ADO.Net Data provider:

- Connection object – Represents connection to the Database
- Command object – Used to execute stored procedure and command on Database
- ExecuteNonQuery – Executes command but doesn't return any value
- ExecuteScalar – Executes and returns single value
- ExecuteReader – Executes and returns result set
- DataReader – Forward and read only recordset
- DataAdapter – This acts as a bridge between database and a dataset.

---

5. IS IT POSSIBLE TO LOAD MULTIPLE TABLES IN A DATASET?

Yes, it is possible to load multiple tables in a single dataset.**29. Which provider is used to connect MS Access, Oracle, etc…?**

OLEDB Provider and ODBC Provider are used to connect to MS Access and Oracle. Oracle Data Provider is also used to connect exclusively for oracle database.

---

6. WHAT ARE DIFFERENT LAYERS OF ADO.NET?

There are three different layers of ADO.Net:

- Presentation Layer
- Business Logic Layer
- Database Access Layer

---

7. WHAT ARE ALL THE CLASSES THAT ARE AVAILABLE IN SYSTEM.DATA NAMESPACE?

Following are the classes that are available in System.Data Namespace:

- Dataset.
- DataTable.
- DataColumn.
- DataRow.
- DataRelation.
- Constraint.

---

8. WHAT ARE THE DATA PROVIDERS IN ADO.NET?

Following are the Data Providers used in ADO.Net:.

- MS SQL Server.
- OLEDB.
- ODBC.

9. WHAT IS THE DIFFERENCE BETWEEN EXECUTESCALAR AND EXECUTENONQUERY?

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

---

10. WHICH METHOD IS USED BY COMMAND CLASS TO EXECUTE SQL STATEMENTS THAT RETURN SINGLE VALUE?

ExecuteScalar

JAVASCRIPT

1. NAME THE TYPES OF FUNCTIONS IN JAVASCRIPT?

The types of function are:

- Named - These type of functions contains name at the time of definition. For Example:

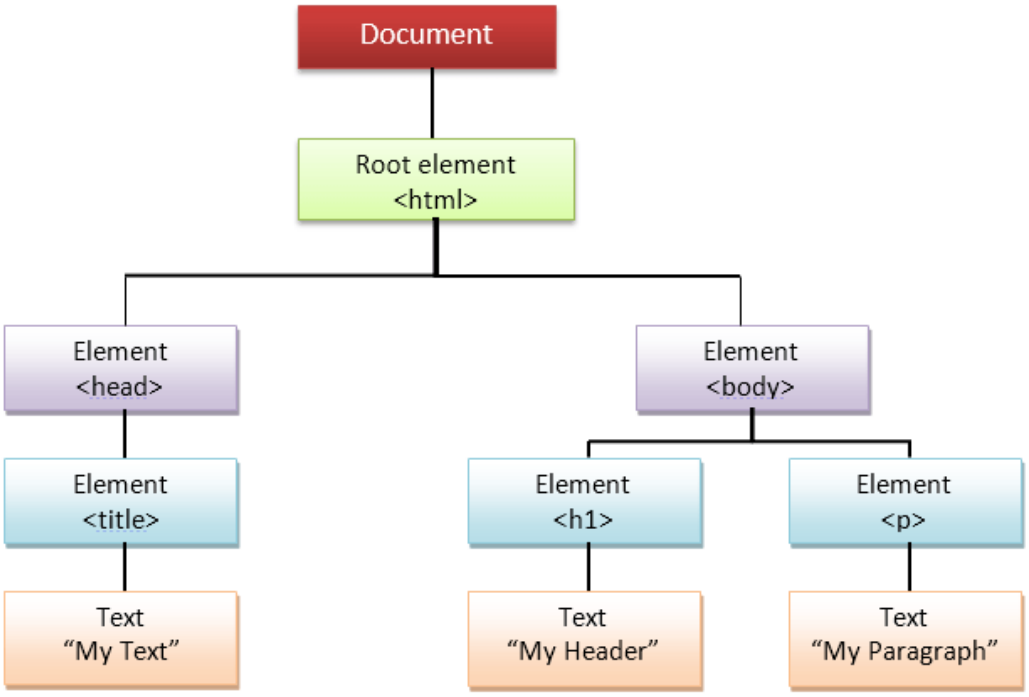
```
function display()
{
  document.writeln("Named Function");
}
display();
```

- Anonymous - These type of functions doesn't contain any name. They are declared dynamically at runtime.

```
var display=function()
{
  document.writeln("Anonymous Function");
}
display();
```

2. WHAT IS DOM? WHAT IS THE USE OF DOCUMENT OBJECT?

DOM stands for *Document Object Model*. A document object represents the HTML document. It can be used to access and change the content of HTML.



---

3. WHAT IS THE DIFFERENCE BETWEEN == AND ===?

The == operator checks equality only whereas === checks equality, and data type, i.e., a value must be of the same type.

---

4. HOW TO WRITE HTML CODE DYNAMICALLY USING JAVASCRIPT?

The innerHTML property is used to write the HTML code using JavaScript dynamically. Let's see a simple example:

```
document.getElementById('mylocation').innerHTML="<h2>This is heading using JavaScript</h2>";
```

5. HOW TO CREATE OBJECTS IN JAVASCRIPT?

There are 3 ways to create an object in JavaScript.

1. By object literal
2. By creating an instance of Object
3. By Object Constructor

Let's see a simple code to create an object using object literal.

emp={id:102,name:"Rahul Kumar",salary:50000}

---

6. WHAT DOES THE ISNAN() FUNCTION?

The isNan() function returns true if the variable value is not a number. For example:

---

7. HOW TO HANDLE EXCEPTIONS IN JAVASCRIPT?

By the help of try/catch block, we can handle exceptions in JavaScript. JavaScript supports try, catch, finally and throw keywords for exception handling.

```
try {
 Block of code to try
}
catch(err) {
 Block of code to handle errors
}
finally {
 Block of code to be executed regardless of the try / catch
result
}
```

8. WHAT IS THIS KEYWORD IN JAVASCRIPT?

The this keyword is a reference variable that refers to the current object.

For example:

```
var person = {
 firstName: "John",
 lastName : "Doe",
 id     : 5566,
 fullName : function() {
  return this.firstName + "  " + this.lastName;
 }
};
```

9. WHAT IS THE USE OF DEBUGGER KEYWORD IN JAVASCRIPT?

JavaScript debugger keyword sets the breakpoint through the code itself. The debugger stops the execution of the program at the position it is applied. Now, we can start the flow of execution manually. If an exception occurs, the execution will stop again on that particular line.. For example:

```
var x = 15 * 5;
debugger;
document.getElementbyId("demo").innerHTML = x;
```

10.WHAT IS THE USE OF MATH OBJECT IN JAVASCRIPT?

The JavaScript math object provides several constants and methods to perform a mathematical operation. Unlike date object, it doesn't have constructors.

For example:

```
Math.abs(-4.7);   // returns 4.7
```

```
Math.ceil(4.4);   // returns 5

Math.random();   // returns a random number
```

11.DEFINE A NAMED FUNCTION IN JAVASCRIPT.

The function which has named at the time of definition is called a named function. For example

```
1. function msg()
2. {
3.   document.writeln("Named Function");
4. }
5. msg();
```

12.WHAT IS THE USE OF WINDOW OBJECT?

The window object is created automatically by the browser that represents a window of a browser. It is not an object of JavaScript. It is a browser object.

The window object is used to display the popup dialog box. Let's see with description.

Method	Description
alert()	displays the alert box containing the message with ok button.
confirm()	displays the confirm dialog box containing the message with ok and cancel but
prompt()	displays a dialog box to get input from the user.

open()	opens the new window.
close()	closes the current window.
setTimeout()	performs the action after specified time like calling function, evaluating expres

---

13.DEFINE ANONYMOUS FUNCTION

It is a function that has no name. These functions are declared dynamically at runtime using the function operator instead of the function declaration. The function operator is more flexible than a function declaration. It can be easily used in the place of an expression. For example:

```
1. var display=function()
2. {
3.   alert("Anonymous Function is invoked");
4. }
5. display();
```

14.CAN AN ANONYMOUS FUNCTION BE ASSIGNED TO A VARIABLE?

Yes, you can assign an anonymous function to a variable.

15.IN JAVASCRIPT WHAT IS AN ARGUMENT OBJECT?

The variables of JavaScript represent the arguments that are passed to a function.

16.DEFINE CLOSURE.

In JavaScript, we need closures when a variable which is defined outside the scope in reference is accessed from some inner scope.

```
1. var num = 10;
2. function sum()
3. {
4. document.writeln(num+num);
5. }
6. sum();
```

17.IF WE WANT TO RETURN THE CHARACTER FROM A SPECIFIC INDEX WHICH METHOD IS USED?

The JavaScript string charAt() method is used to find out a char value present at the specified index. The index number starts from 0 and goes to n-1, where n is the length of the string. The index value can't be a negative, greater than or equal to the length of the string. For example:

```
1. var str="Javatpoint";
2. document.writeln(str.charAt(4));
```

18.WHAT IS THE DIFFERENCE BETWEEN JAVASCRIPT AND JSCRIPT?

Netscape provided the JavaScript language. Microsoft changed the name and called it JScript to avoid the trademark issue. In other words, you can say JScript is the same as JavaScript, but Microsoft provides it.

19.HOW TO USE EXTERNAL JAVASCRIPT FILE?

I am assuming that js file name is message.js, place the following script tag inside the head tag.

```
<script type="text/javascript" src="message.js"></script>
```

20.WHAT IS THE USE OF HISTORY OBJECT?

The history object of a browser can be used to switch to history pages such as back and forward from the current page or another page. There are three methods of history object.

1. history.back() - It loads the previous page.
2. history.forward() - It loads the next page.
3. history.go(number) - The number may be positive for forward, negative for backward. It loads the given page number.

21.HOW TO CREATE A FUNCTION IN JAVASCRIPT?

To create a function in JavaScript, follow the following syntax.

```
function function_name(){
//function body
}
```

22.WHAT ARE THE JAVASCRIPT DATA TYPES?

There are two types of data types in JavaScript:

1. Primitive Data Types - The primitive data types are as follows:

Data Type	Description
String	represents a sequence of characters, e.g., "hello"
Number	represents numeric values, e.g., 100
Boolean	represents boolean value either false or true
Undefined	represents an undefined value
Null	represents null, i.e., no value at all

2. Non-primitive Data Types - The non-primitive data types are as follows:

Data Type	Description
Object	represents an instance through which we can access members
Array	represents a group of similar values
RegExp	represents regular expression

23.HOW TO WRITE NORMAL TEXT CODE USING JAVASCRIPT DYNAMICALLY?

The innerText property is used to write the simple text using JavaScript dynamically. Let's see a simple example:

```
1. document.getElementById('mylocation').innerText="This is text using JavaS
   cript";
```

24.HOW TO CREATE AN ARRAY IN JAVASCRIPT?

There are 3 ways to create an array in JavaScript.

1. By array literal
2. By creating an instance of Array
3. By using an Array constructor

Let's see a simple code to create an array using object literal.

```
1. var emp=["Shyam","Vimal","Ratan"];
```

25.WHAT ARE THE POP-UP BOXES AVAILABLE IN JAVASCRIPT?

- Alert Box
- Confirm Box
- Prompt Box

26.HOW TO SUBMIT A FORM USING JAVASCRIPT BY CLICKING A LINK?

Let's see the JavaScript code to submit the form by clicking the link.

```
1. <form name="myform" action="index.php">
2. Search: <input type='text' name='query' />
3. <a href="javascript: submitform()">Search</a>
4. </form>
5. <script type="text/javascript">
6. function submitform()
7. {
8.   document.myform.submit();
9. }
10.</script>
```

27.WHAT IS THE ROLE OF A STRICT MODE IN JAVASCRIPT?

The JavaScript strict mode is used to generates silent errors. It provides "use strict"; expression to enable the strict mode. This expression can only be placed as the first statement in a script or a function. For example:

```
1. "use strict";
2. x=10;
3. console.log(x);
```

SQL

1. WHAT IS RDBMS?

RDBMS stands for Relational Database Management System. RDBMS store the data into the collection of tables, which is related by common fields between the columns of the table. It also provides relational operators to manipulate the data stored into the tables.

Example: SQL Server.

---

2. WHAT IS DENORMALIZATION.

DeNormalization is a technique used to access the data from higher to lower normal forms of database. It is also process of introducing redundancy into a table by incorporating data from the related tables.

---

3. WHAT ARE ALL THE DIFFERENT NORMALIZATIONS?

The normal forms can be divided into 5 forms, and they are explained below -.

First Normal Form (1NF):.

This should remove all the duplicate columns from the table. Creation of tables for the related data and identification of unique columns.

Second Normal Form (2NF):.

Meeting all requirements of the first normal form. Placing the subsets of data in separate tables and Creation of relationships between the tables using primary keys.

Third Normal Form (3NF):.

This should meet all requirements of 2NF. Removing the columns which are not dependent on primary key constraints.

Fourth Normal Form (4NF):.

Meeting all the requirements of third normal form and it should not have multi-valued dependencies.

---

4. WHAT IS A RELATIONSHIP AND WHAT ARE THEY?

Database Relationship is defined as the connection between the tables in a database. There are various data basing relationships, and they are as follows:.

- One to One Relationship.
- One to Many Relationship.
- Many to One Relationship.
- Self-Referencing Relationship.

---

5. WHAT IS SUBQUERY?

A subquery is a query within another query. The outer query is called as main query, and inner query is called subquery. SubQuery is always executed first, and the result of subquery is passed on to the main query.

---

6. WHAT ARE THE TYPES OF SUBQUERY?

There are two types of subquery – Correlated and Non-Correlated.

A correlated subquery cannot be considered as independent query, but it can refer the column in a table listed in the FROM the list of the main query.

A Non-Correlated sub query can be considered as independent query and the output of subquery are substituted in the main query.

---

7. WHAT IS A TRIGGER?

A DB trigger is a code or programs that automatically execute with response to some event on a table or view in a database. Mainly, trigger helps to maintain the integrity of the database.

Example: When a new student is added to the student database, new records should be created in the related tables like Exam, Score and Attendance tables.

---

8. WHAT ARE LOCAL AND GLOBAL VARIABLES AND THEIR DIFFERENCES?

Local variables are the variables which can be used or exist inside the function. They are not known to the other functions and those variables cannot be referred or used. Variables can be created whenever that function is called.

Global variables are the variables which can be used or exist throughout the program. Same variable declared in global cannot be used in functions. Global variables cannot be created whenever that function is called.

---

9. WHAT IS DATA INTEGRITY?

Data Integrity defines the accuracy and consistency of data stored in a database. It can also define integrity constraints to enforce business rules on the data when it is entered into the application or database.

---

10. WHAT IS AUTO INCREMENT?

Auto increment keyword allows the user to create a unique number to be generated when a new record is inserted into the table. AUTO INCREMENT keyword can be used in Oracle and IDENTITY keyword can be used in SQL SERVER.

Mostly this keyword can be used whenever PRIMARY KEY is used.

---

11. WHAT IS DATAWAREHOUSE?

Datawarehouse is a central repository of data from multiple sources of information. Those data are consolidated, transformed and made available for the mining and online processing. Warehouse data have a subset of data called Data Marts.

---

12. WHAT IS CROSS-JOIN?

Cross join defines as Cartesian product where number of rows in the first table multiplied by number of rows in the second table. If suppose, WHERE clause is used in cross join then the query will work like an INNER JOIN.

---

13. WHAT IS COLLATION?

Collation is defined as set of rules that determine how character data can be sorted and compared. This can be used to compare A and, other language characters and also depends on the width of the characters.

ASCII value can be used to compare these character data.

14. WHAT ARE ALL DIFFERENT TYPES OF COLLATION SENSITIVITY?

Following are different types of collation sensitivity -.

- Case Sensitivity – A and a and B and b.
- Accent Sensitivity.
- Kana Sensitivity – Japanese Kana characters.
- Width Sensitivity – Single byte character and double byte character.

---

15. ADVANTAGES AND DISADVANTAGES OF STORED PROCEDURE?

Stored procedure can be used as a modular programming – means create once, store and call for several times whenever required. This supports faster execution instead of executing multiple queries. This reduces network traffic and provides better security to the data.

Disadvantage is that it can be executed only in the Database and utilizes more memory in the database server.

---

16. WHAT IS ONLINE TRANSACTION PROCESSING (OLTP)?

Online Transaction Processing (OLTP) manages transaction based applications which can be used for data entry, data retrieval and data processing. OLTP makes data management simple and efficient. Unlike OLAP systems goal of OLTP systems is serving real-time transactions.

Example – Bank Transactions on a daily basis.

---

17. WHAT IS CLAUSE?

SQL clause is defined to limit the result set by providing condition to the query. This usually filters some rows from the whole set of records.

Example – Query that has WHERE condition

Query that has HAVING condition.

---

18. WHAT IS RECURSIVE STORED PROCEDURE?

A stored procedure which calls by itself until it reaches some boundary condition. This recursive function or procedure helps programmers to use the same set of code any number of times.

---

19. WHAT IS UNION, MINUS AND INTERACT COMMANDS?

UNION operator is used to combine the results of two tables, and it eliminates duplicate rows from the tables.

MINUS operator is used to return rows from the first query but not from the second query. Matching records of first and second query and other rows from the first query will be displayed as a result set.

INTERSECT operator is used to return rows returned by both the queries.

---

20. WHAT IS AN ALIAS COMMAND?

ALIAS name can be given to a table or column. This alias name can be referred in WHERE clause to identify the table or column.

Example-.

Select st.StudentID, Ex.Result from student st, Exam as Ex where st.studentID = Ex. StudentID

Here, st refers to alias name for student table and Ex refers to alias name for exam table.

---

21. HOW CAN YOU CREATE AN EMPTY TABLE FROM AN EXISTING TABLE?

Example will be -.

Select * into studentcopy from student where 1=2

Here, we are copying student table to another table with the same structure with no rows copied.

---

22. HOW TO FETCH COMMON RECORDS FROM TWO TABLES?

Common records result set can be achieved by -.

Select studentID from student INTERSECT Select StudentID from Exam

---

23. HOW TO FETCH ALTERNATE RECORDS FROM A TABLE?

Records can be fetched for both Odd and Even row numbers -.

To display even numbers-.

Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=0

To display odd numbers-.

Select studentId from (Select rowno, studentId from student) where mod(rowno,2)=1

from (Select rowno, studentId from student) where mod(rowno,2)=1.[/sql]

---

24. HOW TO SELECT UNIQUE RECORDS FROM A TABLE?

Select unique records from a table by using DISTINCT keyword.

Select DISTINCT StudentID, StudentName from Student.

---

25. WHAT IS THE COMMAND USED TO FETCH FIRST 5 CHARACTERS OF THE STRING?

There are many ways to fetch first 5 characters of the string -.

Select SUBSTRING(StudentName,1,5) as studentname from student

Select LEFT(Studentname,5) as studentname from student

---

26. WHAT ARE ALL TYPES OF USER DEFINED FUNCTIONS?

Three types of user defined functions are.

- Scalar Functions.
- Inline Table valued functions.
- Multi statement valued functions.

---

27. WHAT ARE AGGREGATE AND SCALAR FUNCTIONS?

Aggregate functions are used to evaluate mathematical calculation and return single values. This can be calculated from the columns in a table. Scalar functions return a single value based on the input value.

Example -.

Aggregate – max(), count - Calculated with respect to numeric.

Scalar – UCASE(), NOW() – Calculated with respect to strings.

---

28. WHICH OPERATOR IS USED IN QUERY FOR PATTERN MATCHING?

LIKE operator is used for pattern matching, and it can be used as -.

% - Matches zero or more characters.

_(Underscore) – Matching exactly one character.

Example -.

```
Select * from Student where studentname like 'a%'
Select * from Student where studentname like 'ami_'
```

---

29. DEFINE MAGIC TABLES IN SQL SERVER?

A Table which is automatically created and managed by SQL server internally to store the inserted, updated values for any DML (SELECT, DELETE, UPDATE, etc.) operation, is called as Magic tables in SQL server. The triggers preferably use it.

DESIGN PATTERNS

1. WHAT IS PROTOTYPE DESIGN PATTERN?

Prototype Design patterns:

- Prototype pattern specifies the kind of objects to create using a prototypical instance, and create new objects by copying this prototype.
- It is used to create a duplicate object or clone of the current object to enhance performance.

---

2. WHAT IS BUILDER DESIGN PATTERN?

Builder Design patterns:

- Separate the construction of a complex object from its representation so that the same construction process can create different representations.
- In other words,you will have to design the system in such a way that the client application will simply specify the parameters that should be used to create the complex object and the builder will take care of building the complex object.

---

3. WHAT IS ADAPTER DESIGN PATTERN?

Adapter Design patterns:

- The adapter pattern is adapting between classes and objects
- This pattern involves a single class called adapter which is responsible for communication between two independent or incompatible interfaces
- This works like a bridge between two incompatible interfaces

4. WHAT IS BRIDGE DESIGN PATTERN?

Bridge Design patterns:

- Bridge Pattern separates abstraction from its implementation, so that both can be modified Independently

- Bridge Pattern behaves like a bridge between abstraction class and Implementer class.

5. WHAT IS COMPOSITE DESIGN PATTERN?

Composite Design patterns:

- Composite pattern composes objects in term of a tree structure to represent part as well as whole hierarchies.
- Composite pattern creates a class contains group of its own objects. This class provides ways to modify its group of same objects.
- Composite pattern is used when we need to treat a group of objects and a single object in the same way

6. WHAT IS DECORATOR DESIGN PATTERN?

Decorator Design patterns:

- Decorator pattern is used to add new functionality to an existing object without changing its structure.
- Decorators provide a flexible alternative to subclass for extending functionality.
- This pattern creates a decorator class which wraps the original class and add new behaviors/operations to an object at run-time.

7. WHAT IS FACADE DESIGN PATTERN?

Facade Design patterns:

- Facade Design Pattern makes a software library easier to use, understand and test
- Facade Design Pattern make the library more readable
- Facade Design Pattern reduce dependencies of outside code on the inner workings of a library
- Facade Design Pattern wrap a poorly designed collection of APIs with a single well-designed API.

---

8. WHAT IS FLYWEIGHT DESIGN PATTERN?

Flyweight Design patterns:

- Flyweight design pattern is an object that minimizes memory use by sharing as much data as possible with other similar objects
- Flyweight pattern is used to reduce the number of objects created, to decrease memory and resource usage. As a result it increase performance
- Flyweight design pattern provides a way to use objects in large numbers when a simple repeated representation would use an unacceptable amount of memory.
- The flyweight pattern uses the concepts of intrinsic and extrinsic data.Intrinsic datais held in the properties of the shared flyweight objects. This information is stateless and generally remains unchanged, if any change occurs it would be reflected among all of the objects that reference the flyweight.Extrinsic data is computed on the fly means at runtime and it is held outside of a flyweight object. Hence it can be stateful.

---

9. WHAT IS PROXY DESIGN PATTERN?

Proxy Design patterns:

- Proxy Design pattern involves a class, called proxy class, which represents functionality of another class.
- Proxy is a wrapper or agent object that is being called by the client to access the real serving object behind the scenes.

WEB API / WEB SERVICE / WCF

1. IS IT RIGHT THAT ASP.NET WEB API HAS REPLACED WCF?

It's a not at all true that ASP.NET Web API has replaced WCF. In fact, it is another way of building non-SOAP based services, i.e., plain XML or JSON string.

---

2. WEB API SUPPORTS WHICH PROTOCOL?

Web App supports HTTP protocol.

---

3. WHICH .NET FRAMEWORK SUPPORTS WEB API?

NET 4.0 and above version supports web API.

---

4. WEB API USES WHICH OF THE FOLLOWING OPEN-SOURCE LIBRARY FOR JSON SERIALIZATION?

Web API uses Json.NET library for JSON serialization.

---

5. BY DEFAULT, WEB API SENDS HTTP RESPONSE WITH WHICH OF THE FOLLOWING STATUS CODE FOR ALL UNCAUGHT EXCEPTION?

500 - Internal Server Error

---

6. WHAT IS WEB API ROUTING?

Routing is pattern matching like in MVC.

All routes are registered in Route Tables.

For example:

```
Routes.MapHttpRoute(

Name: "ExampleWebAPIRoute",

routeTemplate: "api/{controller}/{id}

defaults: new { id = RouteParameter.Optional}
```

---

7. WHAT IS SOAP?

SOAP is an XML message format used in web service interactions. It allows to send messages over HTTP or JMS, but other transport protocols can be used. It is also an XML-based messaging protocol for exchanging information among computers.

---

8. WHAT IS THE BENEFIT OF USING REST IN WEB API?

REST is used to make fewer data transfers between client and server which make it an ideal for using it in mobile apps. Web API also supports HTTP protocol. Therefore, it reintroduces the traditional way of the HTTP verbs for communication.

---

9. HOW CAN WE USE WEB API WITH ASP.NET WEB FORM?

Web API can be used with ASP.NET Web Form

It can be performed in three simple steps:

1. Create a Web API Controller,

2. Add a routing table to Application_Start method of Global.sax
3. Then you need to make a jQuery AJAX Call to Web API method and get data.

---

10. HOW YOU CAN RETURN VIEW FROM ASP.NET WEB API METHOD?

No, we can't return a view from ASP.NET Web API Method. Web API creates HTTP services that render raw data. However, it's also possible in ASP.NET MVC application.

---

11. HOW TO REGISTER EXCEPTION FILTER GLOBALLY?

It is possible to register exception filter globally using following code-

GlobalConfiguration.Configuration.Filters.Add(new

MyTestCustomerStore.NotImplExceptionFilterAttribute());

---

12. HOW CAN YOU RESTRICT ACCESS METHODS TO SPECIFIC HTTP VERBS IN WEB API?

With the help of Attributes (like HTTP verbs), It is possible to implement access restrictions in Web API.

It is possible to define HTTP verbs as an attribute to restrict access. Example:

```
[HttpPost]

public void Method1(Class obj)
```

```
{

//logic
```

---

13. WHO CAN CONSUME WEBAPI?

WebAPI can be consumed by any client which supports HTTP verbs such as GET, PUT, DELETE, POST. As WebAPI services don't need any configuration, they are very easy to consume by any client. Infract, even portable devices like Mobile devices can easily consume WebAPI which is certainly the biggest advantages of this technology.

---

14. WHAT ARE WEB SERVICES?

Web services are open standard (XML, SOAP, HTTP etc.) based Web applications that interact with other web applications for the purpose of exchanging data. Web Services can convert your existing applications into Web-applications.

---

15. WHAT ARE THE FEATURES OF WEB SERVICES?

Following are the features of Web service −

- It is available over the Internet or private (intranet) networks.
- It uses a standardized XML messaging system.
- It is not tied to any one operating system or programming language.
- It is self-describing via a common XML grammar.
- It is discoverable via a simple find mechanism.

16. WHAT THE COMPONENTS OF A WEB SERVICE?

The basic web services platform is XML + HTTP. All the standard web services work using the following components −

- SOAP (Simple Object Access Protocol)
- UDDI (Universal Description, Discovery and Integration)
- WSDL (Web Services Description Language)

---

17. HOW DOES A WEB SERVICE WORK?

A web service enables communication among various applications by using open standards such as HTML, XML, WSDL, and SOAP.

You can also use C# to build new web services on Windows that can be invoked from your web application that is based on JavaServer Pages (JSP) and runs on Linux.

---

18. WHAT IS THE PURPOSE OF XML IN A WEB SERVICE?

A web services takes the help of XML to tag the data, format the data.

---

19. WHAT ARE THE BENEFITS OF WEB SERVICES?

Following are the benefits of using web services −

- Exposing the Existing Function on the network − Web services allows you to expose the functionality of your existing code over the network. Once it is

exposed on the network, other application can use the functionality of your program.

- Interoperability − Web services allow various applications to talk to each other and share data and services among themselves.
- Standardized Protocol − Web services use standardized industry standard protocol for the communication. All the four layers (Service Transport, XML Messaging, Service Description, and Service Discovery layers) use well-defined protocols in the web services protocol stack.
- Low Cost of Communication − Web services use SOAP over HTTP protocol, so you can use your existing low-cost internet for implementing web services.

---

20. WHAT DO YOU MEAN BY INTEROPERABILITY OF WEB SERVICES?

Web services allow various applications to talk to each other and share data and services among themselves. Other applications can also use the web services. For example, a VB or .NET application can talk to Java web services and vice versa. Web services are used to make the application platform and technology independent.

---

21. WHAT DO YOU MEAN BY LOOSELY COUPLED ARCHITECTURE OF WEB SERVICES?

A consumer of a web service is not tied to that web service directly. The web service interface can change over time without compromising the client's ability to interact with the service. A tightly coupled system implies that the client and server logic are closely tied to one another, implying that if one interface changes, the other must be updated. Adopting a loosely coupled architecture tends to make software systems more manageable and allows simpler integration between different systems.

22.WHAT IS HTTP?

HTTP stands for Hyper Text Transfer Protocol.Currently, HTTP is the most popular option for service transport. HTTP is simple, stable, and widely deployed. Furthermore, most firewalls allow HTTP traffic. This allows XML-RPC or SOAP messages to masquerade as HTTP messages.

23.WHAT IS WSDL?

WSDL is an XML-based language for describing web services and how to access them. WSDL stands for Web Services Description Language.

24.WHAT IS UDDI?

UDDI is an XML-based standard for describing, publishing, and finding web services. UDDI stands for Universal Description, Discovery, and Integration.

25.EXPLAIN WHAT IS WCF?

WCF (Windows Communication Framework) is Microsoft framework to make inter-process communication easier. Through various means, it lets you do the communication like MS messaging Queuing, Services, Remoting and so on. It also allows you talk with other.NET apps, or non-Microsoft technologies (like J2EE).

26.MENTION WHAT ARE THE MAIN COMPONENTS OF WCF?

Main components of WCF are

- Service: The working logic
- Host: The path where the data is saved. E.g., .exe, process, windows service
- Endpoints: The way the service is exposed to the outside world

---

27.EXPLAIN HOW DOES WCF WORKS?

WCF follows the "Software as a Service" model, where all units of functionality are defined as services. For communication, each point is a portal or connection either with the client or other services. It is a program that exposes a collection of endpoints.

28.EXPLAIN WHAT IS THE DIFFERENCE BETWEEN ASMX WEB SERVICES AND WCF?

The difference between WCF and ASMX or ASP.net web service is that ASMX is designed to send and receive messages using SOAP over HTTP only. While the WCF can exchange messages using any format over any transport protocol.

---

26. MENTION WHAT IS THE ENDPOINT IN WCF AND WHAT ARE THE THREE MAJOR POINTS IN WCF?

Every service must have an **address** that determines where the service is located, contract that defines what the service does and **binding** that tells how to communicate with the service.

- Address: It specifies the location of the service which will be like http://Myserver/Myservice. To communicate with our service client it will use this location
- Contract: It specifies the interface between the server and client. It's a simple interface with some attribute

- Binding: It decides how two parties will communicate with each other in terms of transport and encoding and protocols

---

27. EXPLAIN HOW MANY TYPES OF CONTRACT DOES WCF DEFINES?

WCF defines four types of Contracts

- Service Contracts
- Data Contracts
- Fault Contracts
- Message Contracts

28.WHAT ARE THE TRANSPORT SCHEMAS DOES WCF SUPPORTS?

It supports

- HTTP
- TCP
- Peer network
- IPC (Inter Process Communication)
- MSMQ

29.MENTION WHAT ARE THE WAYS OF HOSTING A WCF SERVICE?

The ways of hosting a WCF service are

- IIS
- Self-Hosting
- WAS (Windows Activation Service)

30.MENTION THE ADDRESS SYNTAX AND THE DIFFERENT FORMATS OF WCF TRANSPORT SCHEME?

Address syntax of WCF transport scheme is

[transport]:// [machine or domain] [: optional port] format

31.IN WCF WHAT ARE DUPLEX CONTRACTS?

Duplex messaging or call-back is used in WCF to communicate with the client. Over different transport system Duplex messaging in WCF is done like TCP, Named pipe and even HTTP. Collectively this is known as duplex contracts in WCF.

32.MENTION WHAT ARE THE DIFFERENT INSTANCE MODES IN WCF?

To a particular service instance WCF binds an incoming message request, so the available modes are

- Per Call: This instance is created for each call, efficient in terms of memory but need to maintain session
- Per Session: For a complete session of a user instance are created
- Single: One instance is created which is shared among all the users and shared among all. In terms of memory it is least efficient.

33.EXPLAIN WHAT IS SOA?

SOA (Service Oriented Architectural) is a collection of services that determines how two computing entities will communicate with each other to achieve certain business functionality and also how one entity can work on behalf of another entity.

34. WHAT ARE THE TYPES OF DATA CONTRACTS IN WCF?

There are two types of Data Contracts

- Data Contract: Attribute used to define the class
- Data Member: Attribute used to define the properties

---

35. WHAT ARE THE THREE TYPES OF TRANSACTION MANAGER WCF SUPPORTS?

The types of the transaction manager that WCF supports are

- Light Weight
- WS- Atomic Transaction
- OLE Transaction

---

36. NAME THE NAMESPACE THAT IS USED TO ACCESS WCF SERVICE?

System.ServiceModel is used to access WCF service

---

37. WHAT IS SOA STANDS FOR?

Service Oriented Architecture

---

38. WHAT IS WCF STANDS FOR?

Windows Communication Foundation.

.NET CORE

1. EXPLAIN THE MIDDLEWARE IN ASP.NET CORE?

The Request handling pipeline is a sequence of middleware components where each component performs the operation on request and either call the next middleware component or terminate the request. When a middleware

component terminates the request, it's called **Terminal Middleware** as It prevents next middleware from processing the request. You can add a middleware component to the pipeline by calling **.Use...** extension method as below.

```
app.UseHttpsRedirection();
app.UseStaticFiles();
app.UseRouting();
```

So **Middleware component** is program that's build into an app's pipeline to handle the request and response. Each middleware component can decide whether to pass the request to next component and to perform any operation before or after next component in pipeline.

---

2. WHAT IS REQUEST DELEGATE?

Request delegates handle each HTTP request and are used to build request pipeline. It can configured using Run, Map and Use extension methods. An request delegate can be a in-line as an anonymous method (called in-line middleware) or a reusable class. These classes or in-line methods are called middleware components.

---

3. WHAT IS HOST IN ASP.NET CORE?

Host encapsulates all the resources for the app. On startup, ASP.NET Core application creates the host. The Resources which are encapsulated by the host include:

- HTTP Server implementation
- Dependency Injection
- Configuration
- Logging
- Middleware components

4. DESCRIBE THE GENERIC HOST AND WEB HOST?

The host setup the server, request pipeline and responsible for app startup and lifetime management. There are two hosts:

.NET Generic Host

ASP.NET Core Web Host

.NET Generic Host is recommended and ASP.NET Core template builds a .NET Generic Host on app startup.
ASP.NET Core Web host is only used for backwards compatibility.

```
// Host creation
public class Program
{
    public static void Main(string[] args)
    {
        CreateWebHostBuilder(args).Build().Run();
    }

    public static IWebHostBuilder
CreateWebHostBuilder(string[] args) =>
        WebHost.CreateDefaultBuilder(args)
            .UseStartup();
}
```

5. DESCRIBE THE SERVERS IN ASP.NET CORE?

Server is required to run any application. **ASP.NET Core** provides an in-process HTTP server implementation to run the app. This server implementation listen for HTTP requests and surface them to the application as a set of request features composed into an HttpContext.
ASP.NET Core use the **Kestrel** web server by default. ASP.NET Core comes with:

- Default Kestrel web server that's cross platform HTTP server implementation.
- IIS HTTP Server that's in-process server for IIS.
- HTTP.sys server that's a Windows-only HTTP server and it's based on the HTTP.sys kernel driver and HTTP Server API.

---

6. HOW CONFIGURATION WORKS IN ASP.NET CORE?

In ASP.NET Core, **Configuration** is implemented using various configuration providers. Configuration data is present in the form of key value pairs that can be read by configuration providers as key value from different configuration sources as below.

- appsettings.json - settings file
- Azure Key Vault
- Environment variables
- In-memory .Net objects
- Command Line Arguments
- Custom Providers

By default apps are configured to read the configuration data from appsettings.json, environment variables, command line arguments etc. While reading the data, values from environment variables override appsettings.json data values. 'CreateDefaultBuilder' method provide default configuration.

---

7. HOW TO READ VALUES FROM APPSETTINGS.JSON FILE?

You can read values from appsettings.json using below code.

```
class Test{
// requires using Microsoft.Extensions.Configuration;
 private readonly IConfiguration Configuration;
    public TestModel(IConfiguration configuration)
    {
        Configuration = configuration;
    }
// public void ReadValues(){
var val = Configuration["key"]; // reading direct key values
var name = Configuration["Employee:Name"]; // read complex
values
}
}
```

Default configuration provider first load the values from appsettings.json and then from appsettings.Environment.json file.
Environment specific values override the values from appsettings.json file. In development environment appsettings.Development.json file values override the appsettings.json file values, same apply to production environment.

---

8. WHAT IS THE OPTIONS PATTERN IN ASP.NET CORE?

Options Pattern allow you to access related configuration settings in Strongly typed way using some classes. When you are accessing the configuration settings with the isolated classes, The app should adhere these two principles.

- **Interface Segregation Principle (ISP) or Encapsulation:** The class the depend on the configurations, should depend only on the configuration settings that they use.
- **Separation of Concerns:** Settings for different classes should not be related or dependent on one another.

---

9. HOW TO USE MULTIPLE ENVIRONMENTS IN ASP.NET CORE?

ASP.NET Core use environment variables to configure application behavior based on runtime environment. launchSettings.json file
sets ASPNETCORE_ENVIRONMENT to Development on local Machine.

10.HOW ROUTING WORKS IN ASP.NET CORE?

Routing is used to handle incoming HTTP requests for the app. Routing find matching executable endpoint for incoming requests. These endpoints are registered when app starts. Matching process use values from incoming request url to process the requests. You can configure the routing in middleware pipeline of configure method in startup class.

```
    app.UseRouting(); // It adds route matching to middlware
pipeline


    // It adds endpoints execution to middleware pipeline
    app.UseEndpoints(endpoints =>
    {
    endpoints.MapGet("/", async context =>
    {
    await context.Response.WriteAsync("Hello World!");
```

```
});
});
```

11.HOW TO HANDLE ERRORS IN ASP.NET CORE?

ASP.NET Core provides a better way to handle the errors in Startup class as below.

```
if (env.IsDevelopment())
{
app.UseDeveloperExceptionPage();
}
else
{
app.UseExceptionHandler("/Error");
app.UseHsts();
}
```

For development environment, Developer exception page display detailed information about the exception. You should place this middleware before other middlewares for which you want to catch exceptions. For other environments `UseExceptionHandler` middleware loads the proper Error page. You can configure error code specific pages in Startup class `Configure` method as below.

```
app.Use(async (context, next) =>
{
await next();
if (context.Response.StatusCode == 404)
```

```
    {
    context.Request.Path = "/not-found";
    await next();
    }
    if (context.Response.StatusCode == 403 ||
context.Response.StatusCode == 503 ||
context.Response.StatusCode == 500)
    {
    context.Request.Path = "/Home/Error";
    await next();
    }
    });
```

12.HOW ASP.NET CORE SERVE STATIC FILES?

In ASP.NET Core, **Static files** such as CSS, images, JavaScript files, HTML are the served directly to the clients. ASP.NET Core template provides a root folder called `wwwroot` which contains all these static files. `UseStaticFiles()` method inside `Startup.Configure` enables the static files to be served to client.
You can serve files outside of this webroot folder by configuring Static File Middleware as following.

```
app.UseStaticFiles(new StaticFileOptions
    {
        FileProvider = new PhysicalFileProvider(
            Path.Combine(env.ContentRootPath,
"MyStaticFiles")), // MyStaticFiles is new folder
        RequestPath = "/StaticFiles"  // this is requested
path by client
```

```
    });
// now you can use your file as below
<img src="/StaticFiles/images/profile.jpg" class="img"
alt="A red rose" />
 // profile.jpg is image inside MyStaticFiles/images folder
```

13.EXPLAIN SESSION AND STATE MANAGEMENT IN ASP.NET CORE?

As we know HTTP is a stateless protocol. HTTP requests are independent and does not retain user values. There are different ways to maintain user state between multiple HTTP requests.

- Cookies
- Session State
- TempData
- Query strings
- Hidden fields
- HttpContext.Items
- Cache

14.CAN ASP.NET APPLICATION BE RUN IN DOCKER CONTAINERS?

Yes, you can run an ASP.NET application or .NET Core application in Docker containers.

15.EXPLAIN THE CACHING OR RESPONSE CACHING IN ASP.NET CORE?

Caching significantly improves the performance of an application by reducing the number of calls to actual data source. It also improves the scalability. Response caching is best suited for data that changes infrequently. Caching makes the copy of data and store it instead of generating data from original source.

Response caching headers control the response caching. `ResponseCache` attribute sets these caching headers with additional properties.

16.WHAT IS IN-MEMORY CACHE?

In-memory cache is the simplest way of caching by ASP.NET Core that stores the data in memory on web server.
Apps running on multiple server should ensure that sessions are sticky if they are using in-memory cache. Sticky Sessions responsible to redirect subsequent client requests to same server. In-memory cache can store any object but distributed cache only stores byte[].
IMemoryCache interface instance in the constructor enables the In-memory caching service via ASP.NET Core dependency Injection.

17.WHAT IS DISTRIBUTED CACHING?

Applications running on multiple servers (Web Farm) should ensure that sessions are sticky. For Non-sticky sessions, cache consistency problems can occur. **Distributed caching** is implemented to avoid cache consistency issues. It offloads the memory to an external process. Distributed caching has certain advantages as below.

- Data is consistent across client requests to multiple server
- Data keeps alive during server restarts and deployments.
- Data does not use local memory

`IDistributedCache` interface instance from any constructor enable distributed caching service via Dependency Injection.

---

18. WHAT IS XSRF OR CSRF? HOW TO PREVENT CROSS-SITE REQUEST FORGERY (XSRF/CSRF) ATTACKS IN ASP.NET CORE?

Cross-Site Request Forgery (XSRF/CSRF) is an attack where attacker that acts as a trusted source send some data to a website and perform some action. An attacker is considered a trusted source because it uses the authenticated cookie information stored in browser.

For example a user visits some site 'www.abc.com' then browser performs authentication successfully and stores the user information in cookie and perform some actions, In between user visits some other malicious site 'www.bad-user.com' and this site contains some code to make a request to vulnerable site (www.abc.com). It's called cross site part of CSRF.

How to prevent CSRF?

- In ASP.NET Core 2.0 or later FormTaghelper automatically inject the antiforgery tokens into HTML form element.
- You can add manually antiforgery token in HTML forms by using `@Html.AntiForgeryToken()` and then you can validate it in controller by `ValidateAntiForgeryToken()` method.
- For more you can visit Prevent Cross-Site Request Forgery (XSRF/CSRF)

---

19. DESCRIBE THE DEPENDENCY INJECTION?

Dependency Injection is a Design Pattern that's used as a technique to achieve the Inversion of Control (IoC) between the classes and their dependencies. ASP.NET Core comes with a built-in Dependency Injection framework that makes configured services available throughout the application. You can configure the services inside the ConfigureServices method as below.

```
services.AddScoped();
```

20.EXPLAIN SESSION AND STATE MANAGEMENT IN ASP.NET CORE.

As we know HTTP is a stateless protocol. HTTP requests are independent and does not retain user values. There are different ways to maintain user state between multiple HTTP requests.

- Cookies
- Session State
- TempData
- Query strings
- Hidden fields

21.HOW TO ACCESS HTTPCONTEXT IN ASP.NET CORE?

ASP.NET Core apps access HttpContext through the IHttpContextAccessor interface.

22.EXPLAIN THE CACHING AND RESPONSE CACHING IN ASP.NET CORE.

Caching significantly improves the performance of an application by reducing the number of calls to actual data source. It also improves the scalability.

Three types of caching

- IN-MEMORY CACHING – Good for single server
- DISTRIBUTED CACHE – Good for web farm
- RESPONSECACHE attribute

Response caching is best suited for data that changes infrequently.

Response caching headers control the response caching. ResponseCache attribute sets these caching headers with additional properties.

```
[ResponseCache(Duration = 60)]
public class HomeController : Controller
{
    public IActionResult Index()
    {
        return View(new HomeOutputModel
        {
            LastUpdated = DateTime.Now
        });
    }
}
```

23.HOW TO ENABLE SESSION IN ASP.NET CORE?

The middleware for the session is provided by the package MICROSOFT.ASPNETCORE.SESSION.

To use the session in ASP.NET Core application, we need to add this package to csproj file and add the Session middleware to ASP.NET Core request pipeline.

24.WHAT IS THE META PACKAGE PROVIDED BY ASP.NET CORE.

The assembly MICROSOFT.ASPNETCORE.ALL is a meta package provide by ASP.NET core.

THE END…

BEST OF LUCK FOR INTERVIEWS