
TOP
200
.NET
INTERVIEW
QUESTIONS
2022
OOPS C#
SQL
ASP.NET
.NET CORE
WEB API
MVC
HAPPY RAWAT

OOPS CONCEPTS

ABSTRACT CLASS & INTERFACE

METHODS & PARAMETERS IN C#

CONSTRUCTOR & ACCESS MODIFIERS

EXCEPTION HANDLING

LOOPS & COLLECTIONS

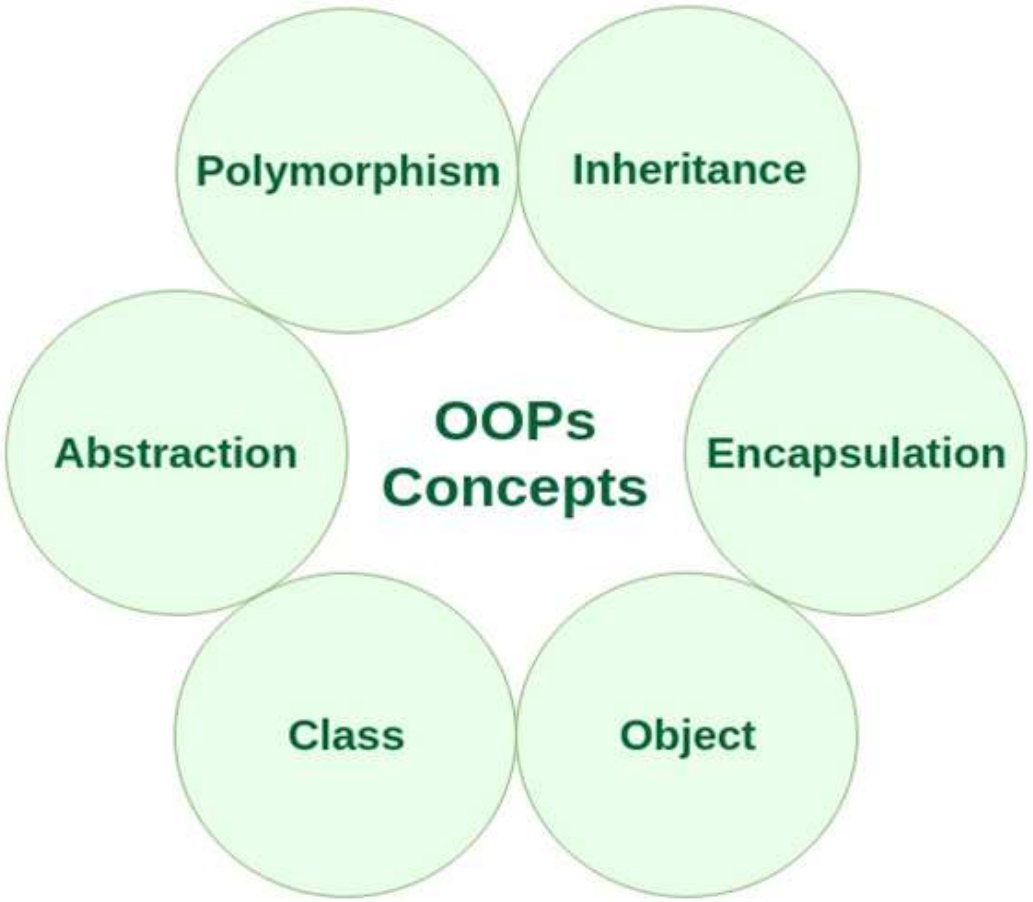
EVENTS & DELGATES

C# IMPORTANT KEYWORDS

Q

What are the main concepts of OOPS? V. IMP.

OOP



What are the main concepts of OOPS?

OOP

CLASS

A class is a **LOGICAL UNIT** that defines the fields, methods, properties.

For example, “USERS” is a class. It can be a group of a particular type, but a single USER is a real-world thing.

OBJECT

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

It’s a real-world entity.

```
public static void Main(string[] args)
{
    Users user = new Users()
    user.Designation = "Softtware Engineer";
    user.Location = "India";
    user.GetUserDetails();
}
```

Object

```
public class Users
{
    public int id = 0;

    public string name = string.Empty;

    public Users()
    {
        //Constructor code
    }

    public void GetUserDetials(int uid, string name)
    {
        id = uid;
        uname = name;
        Console.WriteLine("uid:{0}, name:{1}", id, name);
    }

    public int Designation { get; set; };

    public string Location { get; set; };
}
```

Access Specifier

Class Name

Fields

Constructor

Method

Properties

Q

What are the main concepts of OOPS?

OOP

INHERITANCE

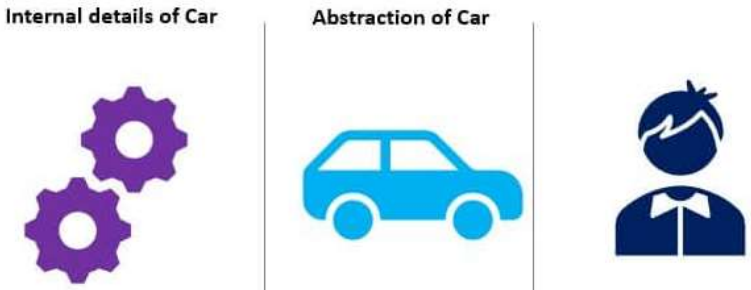
Inheritance is creating a **PARENT-CHILD** relationship between two classes where child automatically get the properties and methods of the parent.

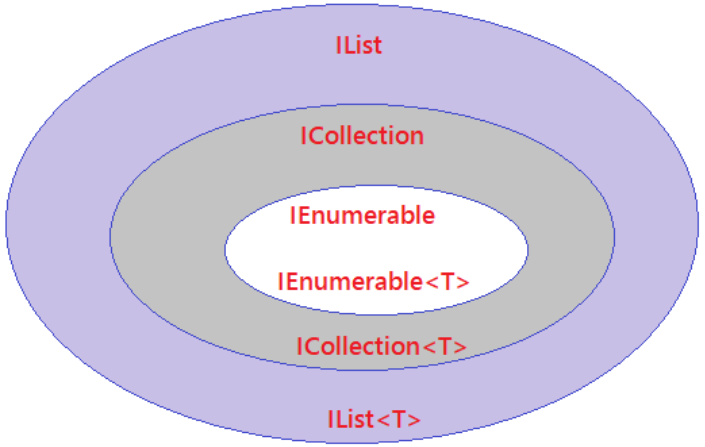
```
class Vehicle                    // Base/ Parent/ Super class
{
    public string brand = "Ford";  // Vehicle field

    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);
  }
}
```

What are the main concepts of OOPS?

ABSTRACTION

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






```
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
```

Background/Hidden methods. Client not aware about them.

What are the main concepts of OOPS?

OOP

ENCAPSULATION

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

```
public class Student
{
    public String StudentName;
}

class ViolateEncapsulation
{
    static public void Main()
    {
        Student obj = new Student();

        obj.StudentName = "Ankita";

        Console.WriteLine("Name: " + obj.StudentName);
    }
}
```

No property or function in this class

Field/Data is accessed without property or function. It will work but it violating encapsulation

```
public class Student
{
    private string StudentName;

    public string Name
    {
        get { return studentName; }

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

This field cannot be accessed from outside without the property

```
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);
    }
}
```

Q

What are the main concepts of OOPS?

OOP

POLYMORPHISM

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

For example, in English, the verb “RUN” has a different meaning if you use it with a laptop, a foot race, and business.

```
public class TestData
{
    public int Add(int a, int b, int c)
    {
        return a + b + c;
    }
    public int Add(int a, int b)
    {
        return a + b;
    }
}
```

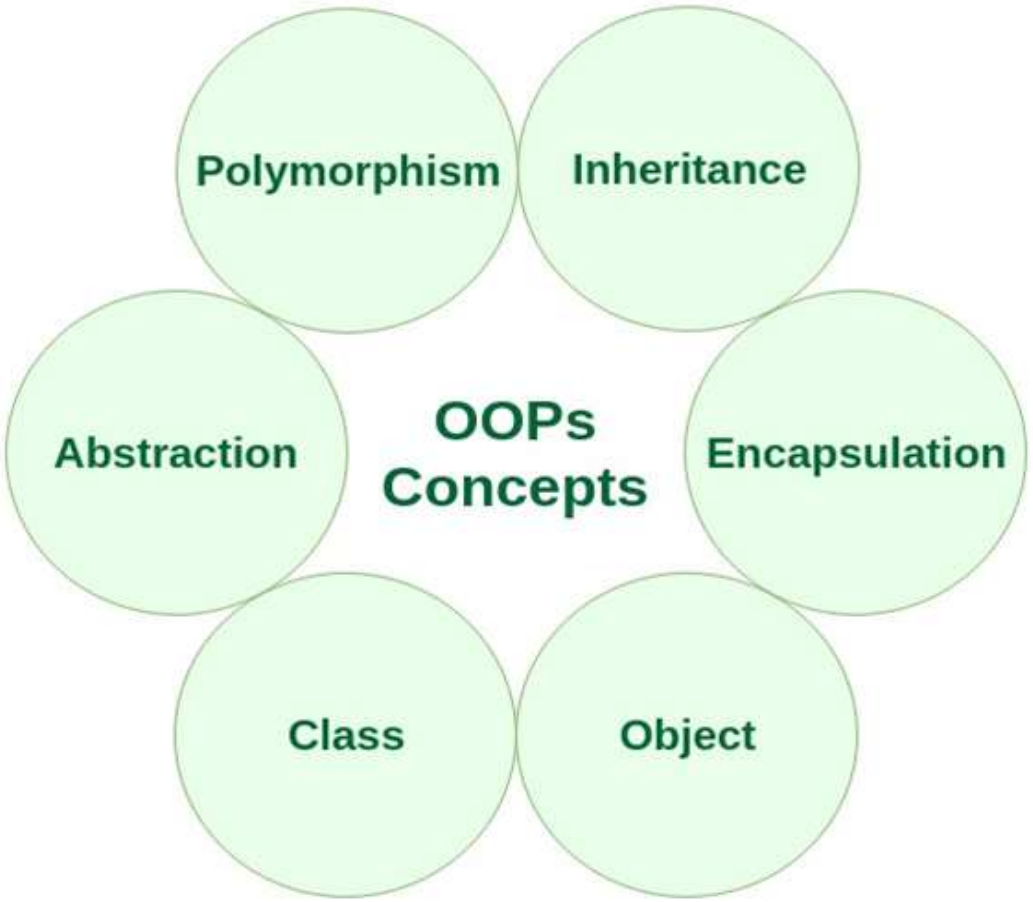
Functions with same name but having multiple forms

```
class Program
{
    static void Main(string[] args)
    {
        TestData dataClass = new TestData();

        int add2 = dataClass.Add(10, 20, 30);

        Console.WriteLine(add2);

        int add1 = dataClass.Add(10, 20);

        Console.WriteLine(add1);
    }
}
//Output: 60 30
```

Q

What are the main concepts of OOPS?

OOP



Q

What is the difference between Object Oriented Programming and Procedural Oriented Programming?

OOP

Book - “Clean Code” by Robert C. Martin

In Procedural Oriented Programming Encapsulation is violated. Data and Functions are not wrapped.

```
public class Student{
    public string name;

}
public class School
{
    public static void Main(string[] args)
    {
        Student stu = new Student();
        stu.name = "Happy";
        Console.WriteLine (stu.name);
    }
}
```

Field is accessible outside the class without a property or function.

Procedural Oriented Programming

In Object Oriented Programming Encapsulation is followed. Data and Functions are wrapped.

```
public class School
{
    public static void Main(string[] args)
    {
        Student stu = new Student();
        stu.Name = "Happy";
        Console.WriteLine (stu.Name);
    }
}
public class Student{
    private string name;
    public string Name
    {
        get {return name;}
        set {name = value;}
    }
}
```

Field in not accessible, only property is

Object Oriented Programming

Q

What are the advantages of OOPS?

OOP

Advantages of OOPS:

1. **Reuse** of code through inheritance
2. **Flexibility** through polymorphism
3. The principle of data hiding/Encapsulation helps the programmer to build **secure** programs.
4. OOP systems can be **easily upgraded** from small to large systems.
5. Modularity for **easier troubleshooting.**

Q

What are the limitations of OOPS?

OOP

Disadvantage of OOPS:

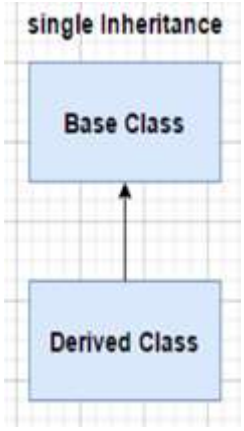
- It is not suitable for **small** applications.

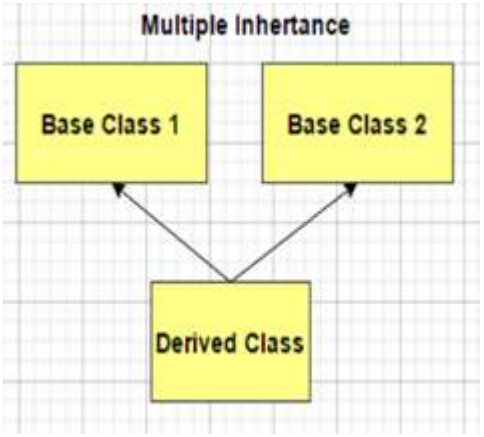
What is Inheritance? Why Inheritance is important?

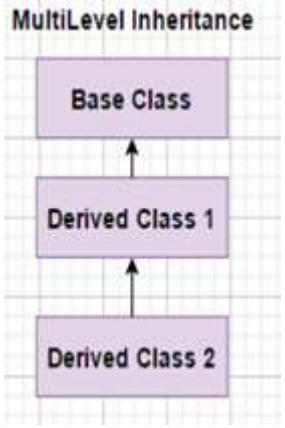
Inheritance is good for: **REUSABILITY** and **ABSTRACTION** of code

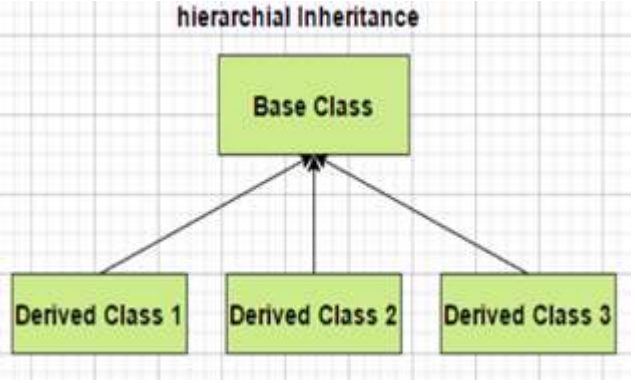
Q What are the different types of Inheritance? V. IMP.

OOP










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

```
class BaseClass1
{
    public void Animal()
    {
        Console.WriteLine("Animal");
    }
}
class Interface1
{
    public void Fly();
}
class ChildClass : BaseClass1, Interface1
{
    public void Eagle()
    {
        Console.WriteLine("Eagle");
    }
}
```

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

```
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");
    }
}
```

Q

How to prevent a class from being Inherited?

OOP

By using **SEALED** keyword in class

```
1 reference
sealed class ABC { }
0 references
class XYZ : ABC { } //Invalid
```

class SealedClassEx.ABC

CS0509: 'XYZ': cannot derive from sealed type 'ABC'

Show potential fixes (Alt+Enter or Ctrl+.)

What is Polymorphism and what are its types?

POLYMORPHISM

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

Q What is Method Overloading? In how many ways a method can be overloaded?

OOP

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) know that these two different methods are different.

**Compile time meaning when you will build the project then you see some build errors. Because the .Net compiler has caught them before running the program. But some errors you will see like "object reference not set to an object" at run time because compiler is not able to catch these errors before running. These are run time errors.*

Method overloading can be done in 3 ways

```
public class Methodoveloading
  {
    public int add(int a, int b)
    {
        return a + b;
    }

    public int add(int a, int b, int c)
    {
        return a + b + c;
    }

    public float add(float a, float b, int c)
    {
        return a + b + c;
    }

    public float add(float a, int c, float b)
    {
        return a + b + c;
    }
  }
```

1. Number of parameters are different

2. Type of parameters are different

2. Order of parameters are different

Q What is the difference between Overloading and Overriding? V. IMP.

OOP

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.

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.

If you will remove Virtual and Override keyword then it will use baseClass Greetings() method and the output will be:

"baseClass Saying Hello!"

```
class baseClass
{
    public virtual void Greetings()
    {
        Console.WriteLine("baseClass Saying Hello!");
    }
}
class subClass : baseClass
{
    public override void Greetings()
    {
        Console.WriteLine("subClass Saying Hello!");
    }
}
class Program
{
    static void Main(string[] args)
    {
        baseClass obj1 = new subClass();

        obj1.Greetings();
    }
}
//Output: subClass Saying Hello!
```

Same method name but one is in base class and another is in derived class

This will call subclass Greetings() method because of overriding.

Q

What is the difference between Overloading and Overriding? V. IMP.

OOP

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

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

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

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

Q

What is the difference between Method Overriding and Method Hiding?

OOP

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.

```
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

```
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

Q What is the difference between an Abstract class and an Interface (atleast 4)? V. IMP.

OOP

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

2. Abstract class keyword: **ABSTRACT**

1. Interface contains **ONLY DECLARATION** of methods.

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

    void Dashboard();
}
```

Only method Declaration is allowed

2. Interface keyword: **INTERFACE**

Q

What is the difference between an Abstract class and an Interface (atleast 4)? V. IMP.

OOP

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

Interface contains **ONLY DECLARATION** of methods.

2. Abstract class keyword: **ABSTRACT**

Interface keyword: **INTERFACE**

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

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

```
// abstract class 'Car'
public abstract class Car {
    //private field
    private string Model;
    // Method with declaration
    public abstract void Engine();

    //Method with definition
    public void Dashboard()
    {
        Console.WriteLine("Dashboard");
    }
}
```

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

    void Dashboard();
}
```

When to use Interface and when Abstract class?

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

Interface contains **ONLY DECLARATION** of methods.

2. Abstract class keyword: **ABSTRACT**

Interface keyword: **INTERFACE**

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

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

4. Abstract class does not Support **MULTIPLE INHERITANCE**

Interface supports multiple inheritance

```
class BaseClass1
{
    public void Animal()
    {
        Console.WriteLine("Animal");
    }
}
class Interface1
{
    public void Fly();
}
class ChildClass : BaseClass1, Interface1
{
    public void Eagle()
    {
        Console.WriteLine("Eagle");
    }
}
```

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.

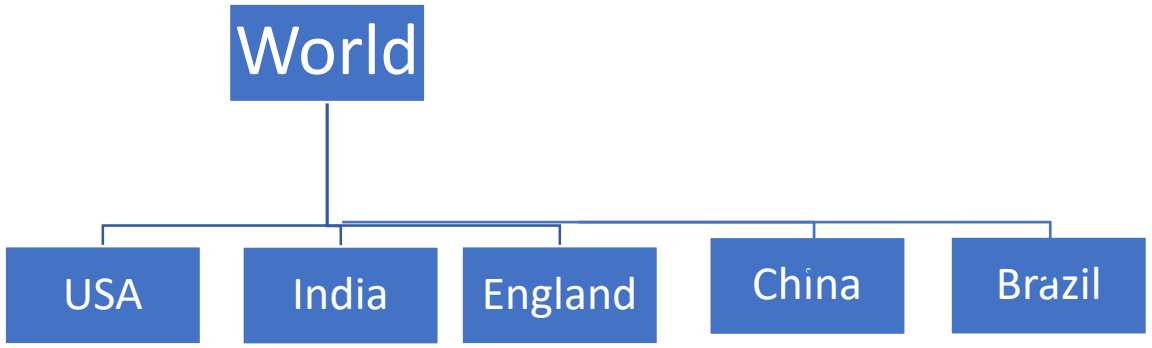
An interface is a good choice when you know a method has to be there, but it can be implemented **DIFFERENTLY** by independent derived classes.

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

```
//Abstract class use - Some mehods declared and some defined
public abstract class USATaxSystem
{
    //Just declared because it is different
    //for an individual and for a company
    //will be defined in derived class
    public abstract int CalculateTax();

    //Always will be in USD(Dollar) therefore it is concrete
    public string TaxCurrency()
    {
        return USD;
    }
}
```

```
//Interfaces use - all methods declared
interface IWorldTaxSystem
{
    //Just declared because it is different
    //for an individual and for a company
    //will be defined in derived class
    public abstract int CalculateTax();

    //Currency will be as per country
    //therefore it declared only
    public string TaxCurrency();
}
```

Why to even create Interfaces?




```
//Interfaces use - all methods declared
interface IWorldTaxSystem
{
    //Just declared because it is different
    //for an individual and for a company
    //will be defined in derived class
    public abstract int CalculateTax();

    //Currency will be as per country
    //therefore it declared only
    public string TaxCurrency();
}
```

Q

Do Interface can have a Constructor?

OOP

NO.

Interface can only be derived from.

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

    void Dashboard();
}
```

Q

Can you create an instance of an Abstract class or an Interface?

OOP

NO.

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

Q

What is the difference between “out” and “ref” parameters?

OOP

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

```
public static void Main(string[] args)
{
    int a = 10;
    int b = 5;

    OutRefExample p = new OutRefExample();

    int c = p.Update( a, b );

    Console.WriteLine(c);
}

public class OutRefExample
{
    public int Update( int c, int d )
    {
        return c + d;
    }
}
```

When to use out and ref parameter?

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

Q What is the difference between “out” and “ref” parameters?

OOP

In short you will use ref when you want to modify a value and out parameter when you want to get a new value from the called method.

```
public static void Main(string[] args)
{
    int a;

    int b = 5;

    OutRefExample p = new OutRefExample();

    p.Update( out a, ref b );

    Console.WriteLine("out value: {0}", a);

    Console.WriteLine("ref value: {0}", b);
}

public class OutRefExample
{
    public void Update( out int a, ref int b )
    {
        a = 10;

        b = 20;
    }
}
```

1. No need to initialize out parameter before passing it.

1. Must initialize ref parameter else error.

2. Out parameter must be initialized before returning.

2. Initialize not necessary, you can comment this line still fine.

Q

What is the purpose of “params” keyword?

OOP

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 used.

```
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.

Q What are Extension Methods in C#? When to use them? V. IMP.

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 main class and you want to add a new method in main class.

Like in the example we added a new 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.

```
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
```

Q What are Access Specifiers? What is the default access modifier in a class?

OOP

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

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

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	✓	✓	✓	✓	✗

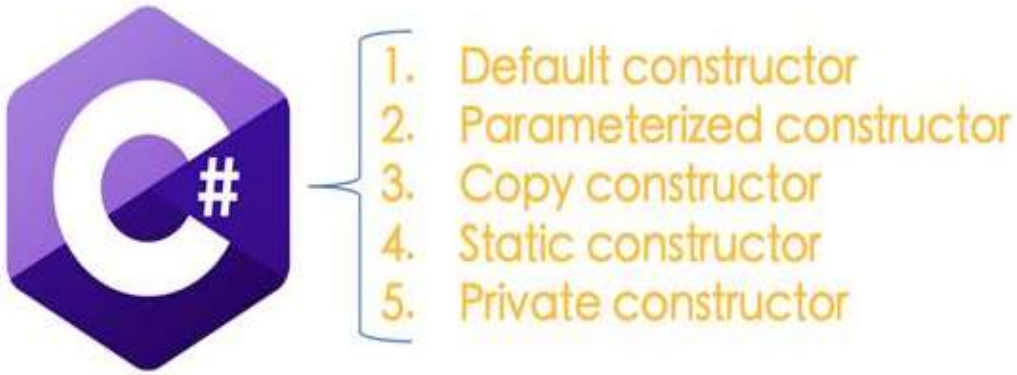
PRIVATE is the default access modifier of a class.

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

         Console.WriteLine(test);
     }
 }
 //Output: HelloWorld
```

What is a Constructor and what are its types? V. IMP.

OOP

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




```
public class Program
{
    public Program()
    {
        //Logic written here is automatically
        //called whenever you will
        //create object of this Program class...

        Console.WriteLine ("Hello Constructor");
    }
    public static void Main(string[] args)
    {
        Program p = new Program();
    }
}

//Output: Hello Constructor
```

Constructor

Q

What is a Constructor and what are its types?

OOP

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();
    }
}
```

Q

What is a Constructor and what are its types?

OOP

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();
    }
}
```

Q

What is a Constructor and what are its types?

OOP

STATIC CONSTRUCTOR

Static constructor is used to be called before any static member of a class is called.

```
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
```

Q

What is a Constructor and what are its types?

OOP

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();
    }
}
```

What is a Constructor and what are its types?

OOP

COPY CONSTRUCTOR

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

```
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();
    }
}
```

When to use Private constructor?

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

Used in singleton class pattern

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

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
}
```

Q

Can we execute multiple Catch blocks?

OOP

NO

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

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

Q

What is a Finally block and give an example when to use it?

OOP

Finally block will be executed IRRESPECTIVE of exception.

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

Q

Can we have only “Try” block without “Catch” block?

OOP

YES.

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

```
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.

Q What is the difference between “throw ex” and “throw”?

V. IMP.

OOP

```
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);
    }
}
```

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

Stack Trace:

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

```
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 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
```

Its a best practice to use *throw* as it **PRESERVE the whole stack trace.**

What are the Loop types in C#?

While Loop

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

Initialize

Condition

Increment

//Output: 0 1 2 3 4

Do while

```
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

What are the Loop types in C#?

OOP

For Loop

```
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

Foreach Loop

```
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

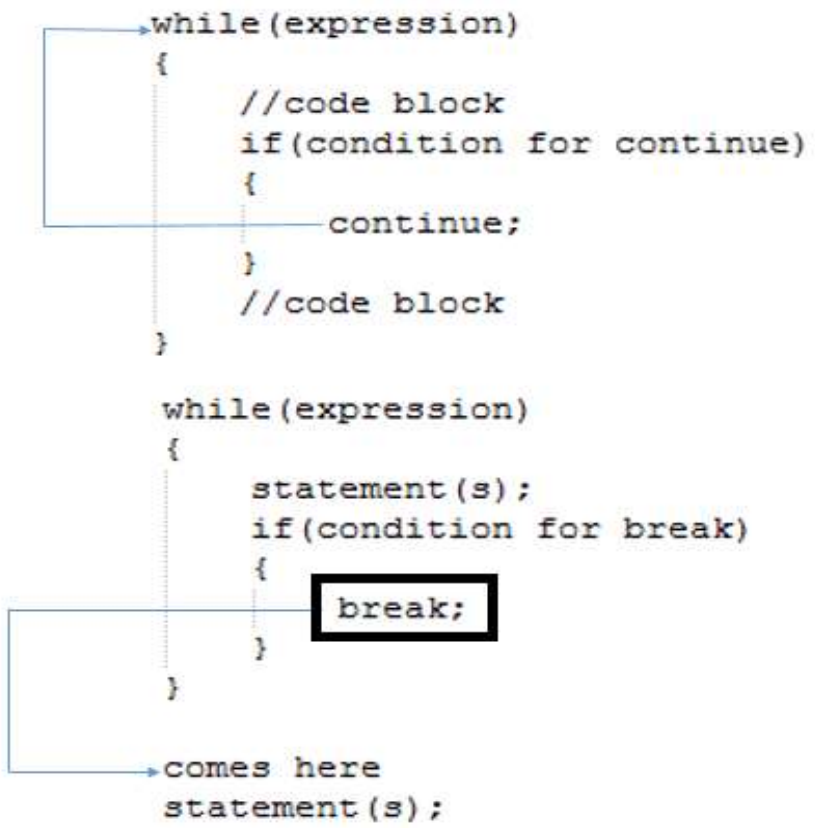
Q

What is the difference between “continue” and “break” statement?

OOP

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

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



Q

What is the difference between Array and ArrayList (atleast 2)?

OOP

1. Array is STRONGLY typed. This means that an array can store only specific type of items\elements. In example the type is custom class Cat.

2. Array can contain FIXED number of items.

Array	ArrayList

Declaration Insertion Access Removal

```
Cat[] catArray;
catArray = new Cat[10];

catArray[0] = moggy1;
catArray[1] = moggy2;

callMethodOn(catArray[1]);

catArray[0] = null;
```

```
ArrayList catAList;
catAList = new ArrayList();

catAList.add(moggy1);
catAList.add(moggy2);

callMethodOn(catAList.get(1));

catAList.remove(moggy1);
```

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

2. ArrayList can store ANY number of items.

Q

What is the difference between Arraylist and Hashtable? V. IMP.

OOP

ArrayList

```
public static void Main(string[] args)
{
    // Example Arraylist:
    ArrayList arrList = new ArrayList();
    arrList.Add(7896);
    arrList.Add("Seven");
}
```

1. In Arraylist we can only add Items/Values to the list.

Hashtable

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

Key Value

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

Q

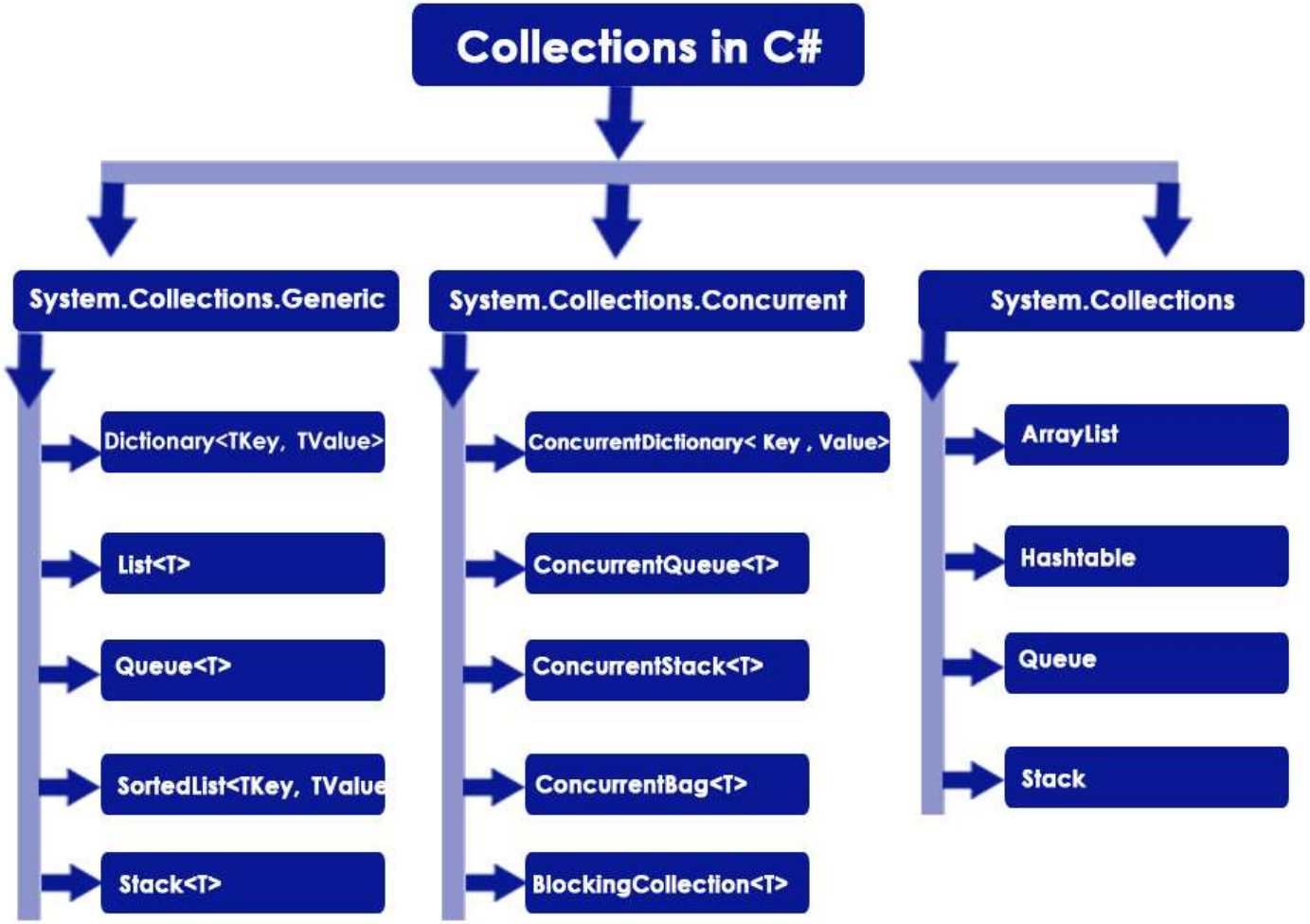
What are Collections in C# and what are their types?

OOP

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.



Q

What is IEnumerable in C#? V. IMP.

OOP

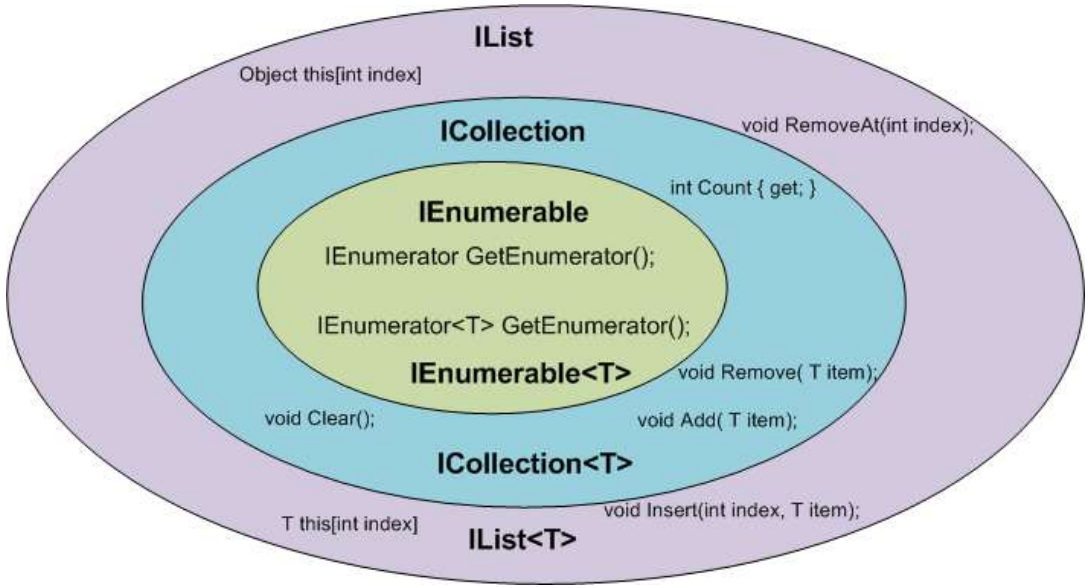
IEnumerable interface is used when we want to ITERATE among our collection classes using a FOREACH loop.

```
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; }
}
```

```
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
```

Q What is the difference between IEnumerable and IEnumerator in C#?

OOP



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

```
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; }
}
```

Q

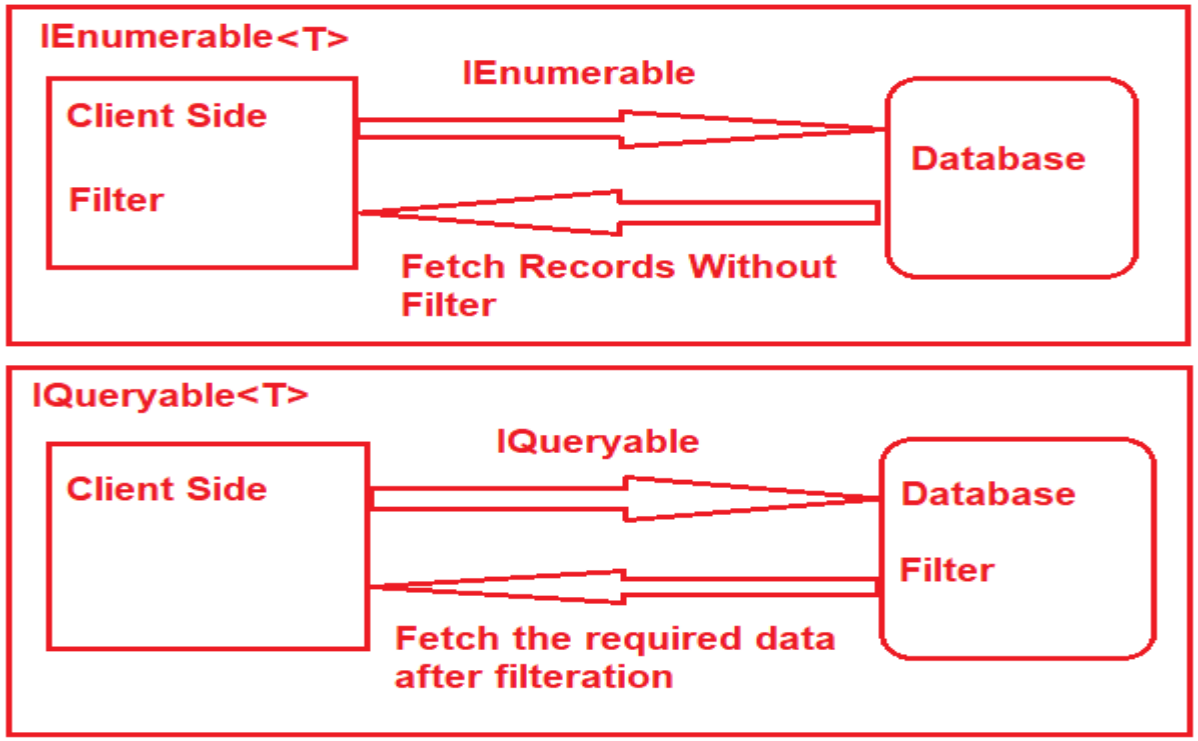
What is the difference between IEnumerable and IQueryable in C#? Why to use IQueryable in sql queries?

OOP

IQueryable is also like IEnumerable and is use to iterate sql query collection from data.

It is under SYSTEM.LINQ namespace.

```
DataClassesDataContext dc = new DataClassesDataContext();
IQueryable<Employee> list = dc.Employees.Where(d => d.Department.Equals("IT"));
list = list.Take<Employee>(3);

foreach (var a in list)
{
    Response.Write(a.Name+"<br/>");
}
```



Q

What you mean by Delegate? When to use them? V. IMP.

OOP

A Delegate is a variable that holds the REFERENCE TO A METHOD.

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

When to use them:
When we need to pass a method as a parameter.

```
public static void Main(string[] args)
{
   int result;

   if(operation = "add")
   {
     result = DelegateExample.add(20, 30);
   }
   else if(operation = "mul")
   {
     result = DelegateExample.mul(20, 30);
   }

   Console.WriteLine(result);
}
```

```
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)
    {
       Calculator c1 = new Calculator(add); //Instantiating delegate

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

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

What are Multicast Delegates?

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

```
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
}
```

What are Multicast Delegates?

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

When we invoke the multicast delegate, then all the functions which are referenced by the delegate are going to be invoked.

```
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
        }
    }
}
```

What are Anonymous Delegates in C#?

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

```
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
```

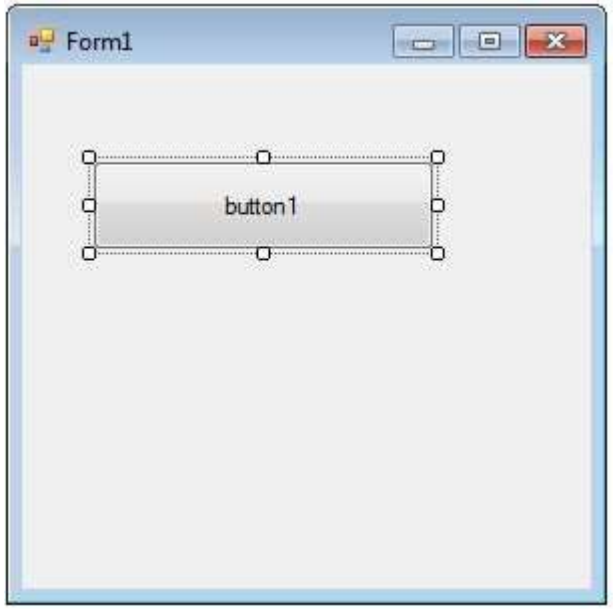
Q

What are the differences between Events and Delegates?

OOP

The event is a notification mechanism that depends on delegates

Functions → Delegate → Event



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.

What is “this” keyword in C#? When to use it?

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

```
class StudentDetails {
   public static void Main(string[] args) {
      Student std1 = new Student(001, "Jack");
      Student std2 = new Student(002, "Harry");

      std1.showInfo();
      std2.showInfo();
   }
}
```

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

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

```
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);
   }
}
```

Q

What is the purpose of “using” keyword in C#? V. IMP.

OOP

There are two purpose of using keyword in C#:

USING DIRECTIVE

using System.IO;
using System.Text;

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

It is mostly used while creating database connections

```
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();
    }
}
```

```
private class SomeDeisposableType : IDisposable
{
    Dispose()
    {
        //Dispose objects here;
    }
}
```

```
SomeDisposableType t = new SomeDisposableType();
try {
    OperateOnType(t);
}
finally {
    if (t != null) {
        ((IDisposable)t).Dispose();
    }
}
```

Equivalent to

```
using (SomeDisposableType u = new SomeDisposableType()) {
    OperateOnType(u);
}
```

Q

What is the difference between “is” and “as” operators?

OOP

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
```

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

Q

What is the difference between “Readonly” and “Constant” variables (atleast 3)? V. IMP.

OOP

Readonly

```
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.

Constant

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

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

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

What is “Static” class? When to use it?

A static class is a class which object can not be created, and which can not be inherited.

Then what is the use of creating Static classes:

Static classes are used as containers for static members like methods, constructors and others.

```
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());
    }
}
```

Q

What is the difference between “var” and “dynamic” in C#?

OOP

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, compile time error

        dynamic b = 10;

        b = "Happy"; //No build error, but run time erro.
    }
}
```

Q

What is Enum keyword used for?

OOP

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
```

Q

What is Boxing and Unboxing? V. IMP.

OOP

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.
It is explicit conversion process.

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

    Object Obj = num;  //Boxing

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

}
```

Q What is the difference between “String” and “StringBuilder”? When to use what?

OOP

String is IMMUTABLE in C#.

That mean you couldn’t modify it after it is created.

Every time you will 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);
    }
}
```

Both these strings are different and occupy different memory in process

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.

```
public static void Main(string[] args)
{
    StringBuilder str1 = new StringBuilder("Interview");

    String str2 = "Happy";

    str1.Append(str2);

    Console.WriteLine(str1);
}
```

Both these strings are occupying same memory in process

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.

What are the basic string operations in C#?

OOP

- **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.");
}
```

Q

What are Nullable types?

OOP

Nullable types

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

```
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;
    }
}
```

Q

Explain Generics in C#? When and why to use them? V. IMP.

OOP

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

```
public class Calculator
{
    public static bool AreEqual(object value1, object value2)
    {
        return value1.Equals(value2);
    }
}
```

The problem with this is, it involves Boxing from converting int (value) to object (reference) type.

This will impact the performance.

```
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

Q

Explain Generics in C#? When and why to use them?

OOP

Here 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);
        }
    }
}
```

Will work

Will work

Q

Explain Generics in C#? When and why to use them?

OOP

Here we are making the **CLASS** independent of type

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

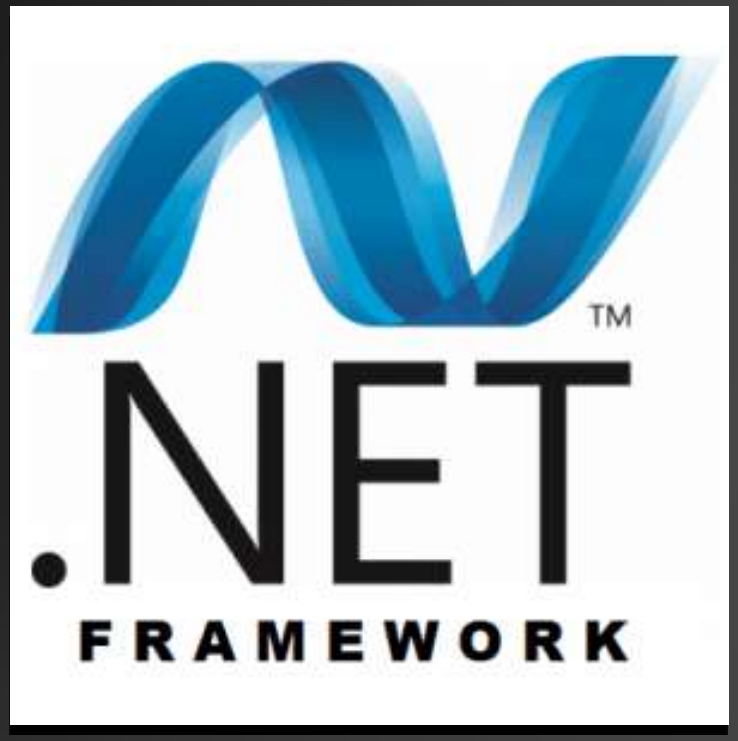
Boxing and unboxing is not required which will impact the performance.

Generics are extensively used by collection classes.

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

Will work

Will work



- BASICS
- GARBAGE COLLECTION
- MULTITHREADING
- MORE

Q

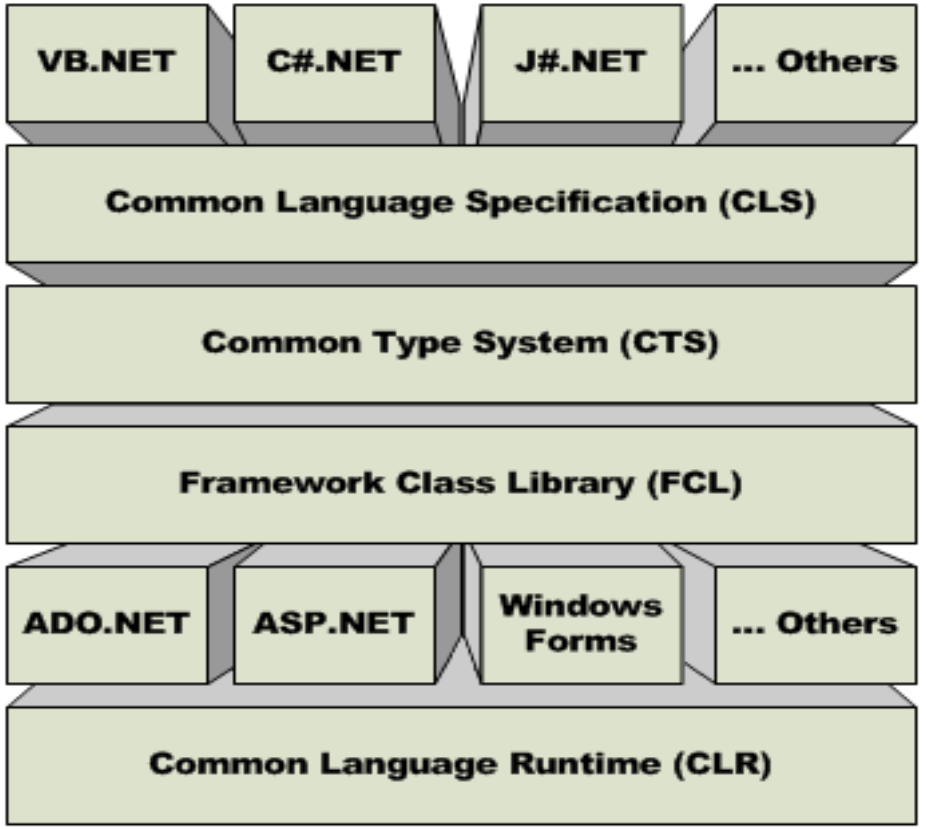
What are the important components of .NET framework? What are their roles?

- 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 differenct .NET languages C#, VB.NET, J# but 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.



What is an Assembly? What are the different types of assembly in .NET?

Assembly is unit of deployment like EXE or a DLL.

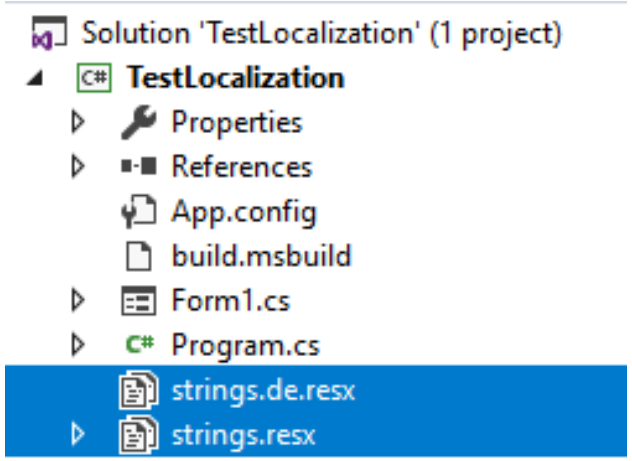
When you create a code and build the solution then the .NET Framework convert it into assembly which you can see inside bin folder.

There are 3 types of assemblies:

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

- **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.

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



Q

What is GAC?

.NET Framework

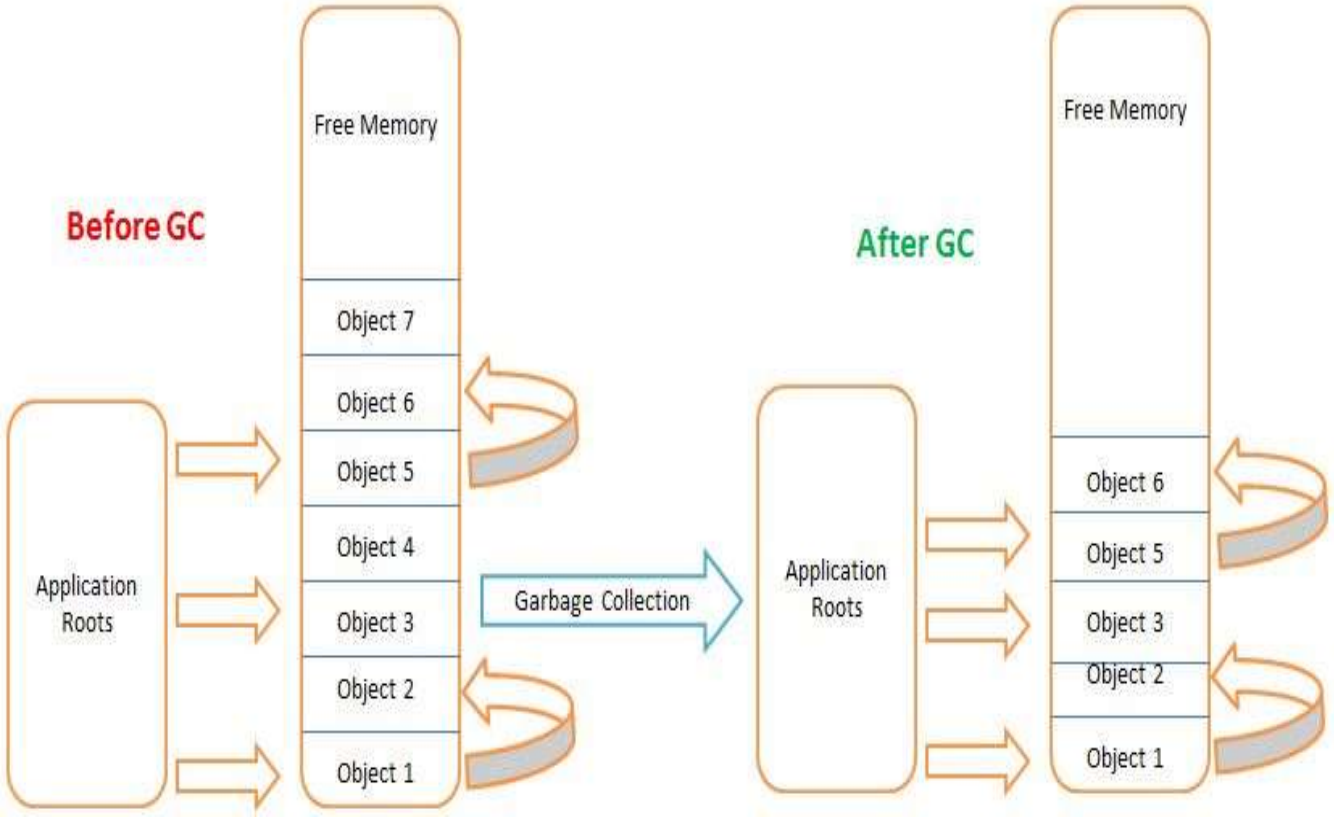
GAC stands for Global Assembly Cache.

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

What is Garbage Collection(GC)? V. IMP.

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

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

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

}
```

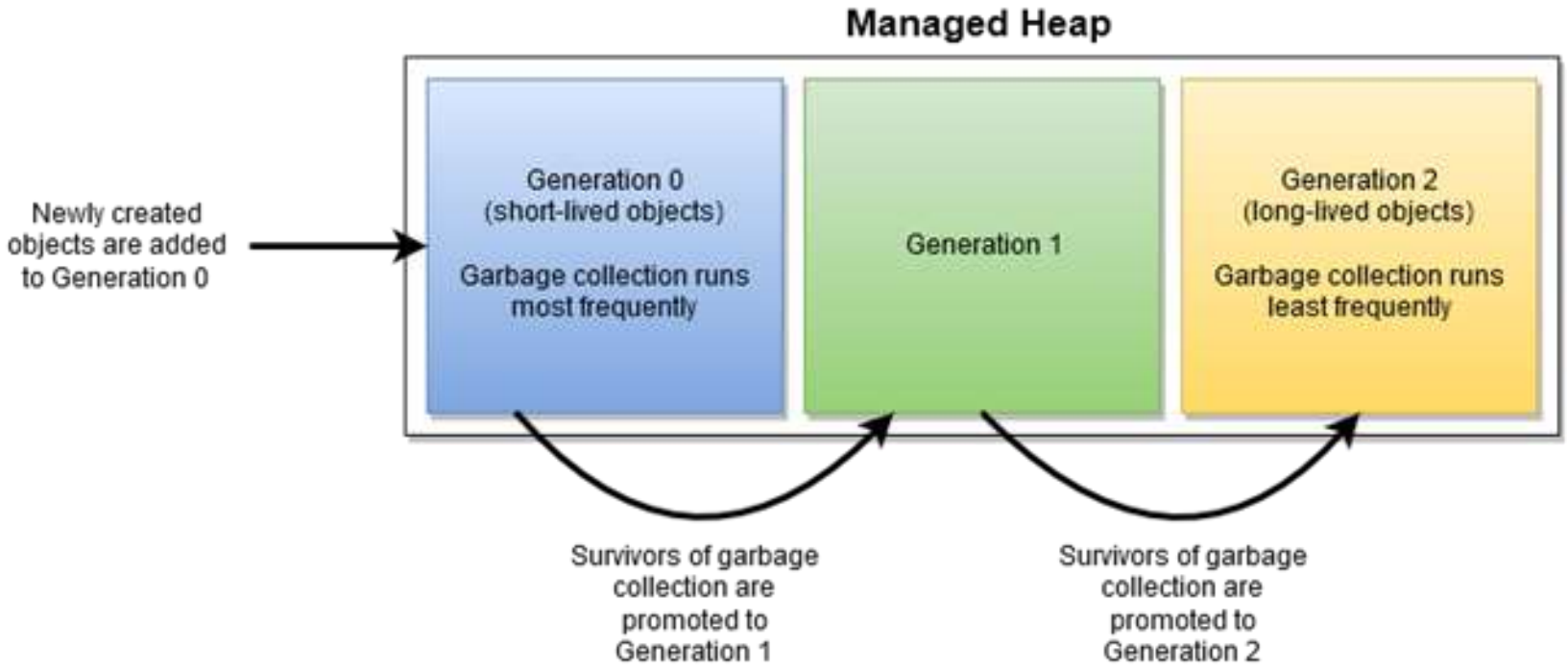


Q

What are Generations in garbage collection?

.NET Framework

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



Q

What is the difference between “Dispose” and “Finalize”? V. IMP.

.NET Framework

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 cannot be called by user code.

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 costs associated with Finalize method. For example, GC is running in very 10 minutes but there is no objects for cleaning in the application then it just wasting the memory which it using for running.

Dispose method

```
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;
         }
      }
}
```

Q 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.

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();
}
```

Q

Can we force Garbage Collector to run?

.NET Framework

YES.

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

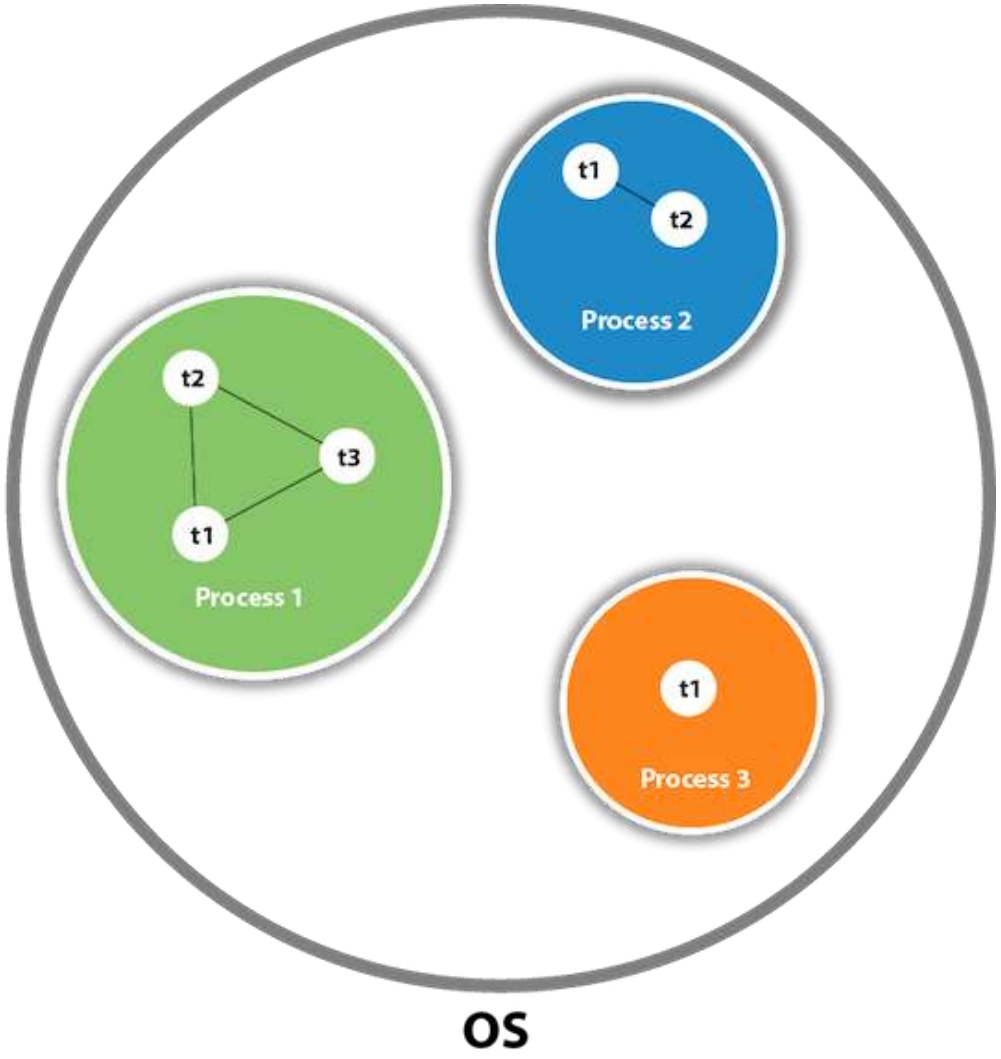
Q

What is the difference between Process and Thread?

.NET Framework

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**.

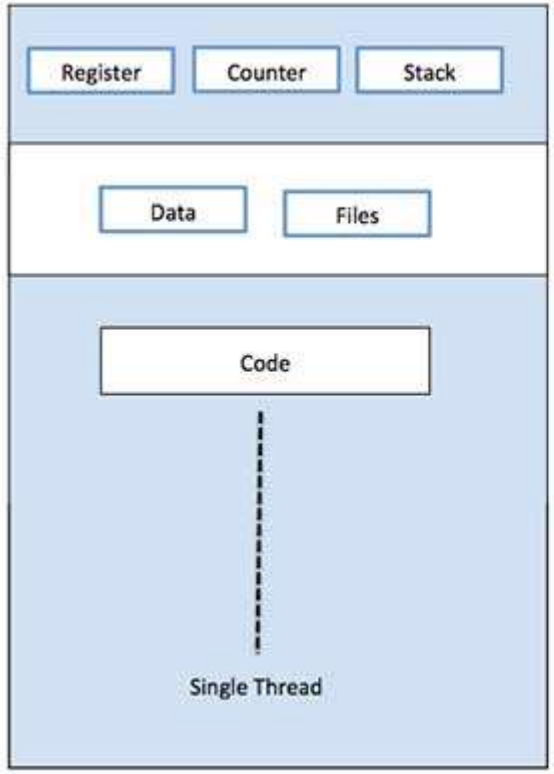


Q

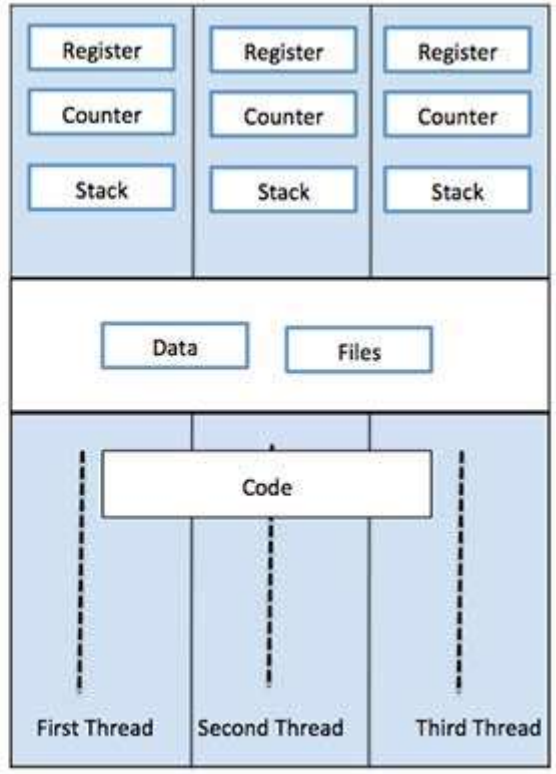
Explain Multithreading?

.NET Framework

- 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

Q What is the difference between Threads and Tasks? (atleast 3) V. IMP.

Task are WRAPPER around Thread classes.

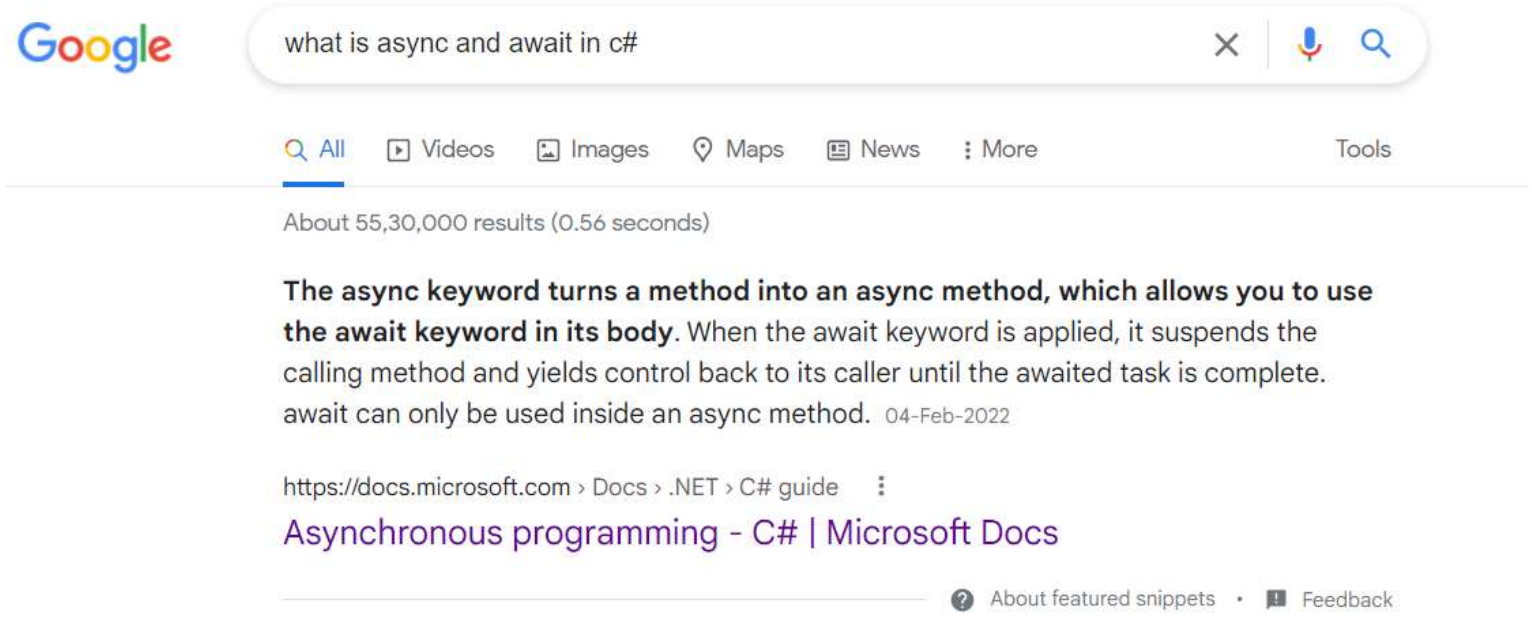
Thread is a computer programming concept, where as Task is not a computer programming concept. Task is created by Microsoft(.NET Framework) in order to work easily with threads.

Task internally used Threads.

Below are some advantages of Tasks over Threads:

1. A Task can RETURN A RESULT but there is no proper way to return a result from Thread.
2. We can apply CHAINING on multiple tasks but we can not in threads.
3. 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.

What is the Async and Await in task? V. IMP.



Q What is the Async and Await in task? V. IMP.

```
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

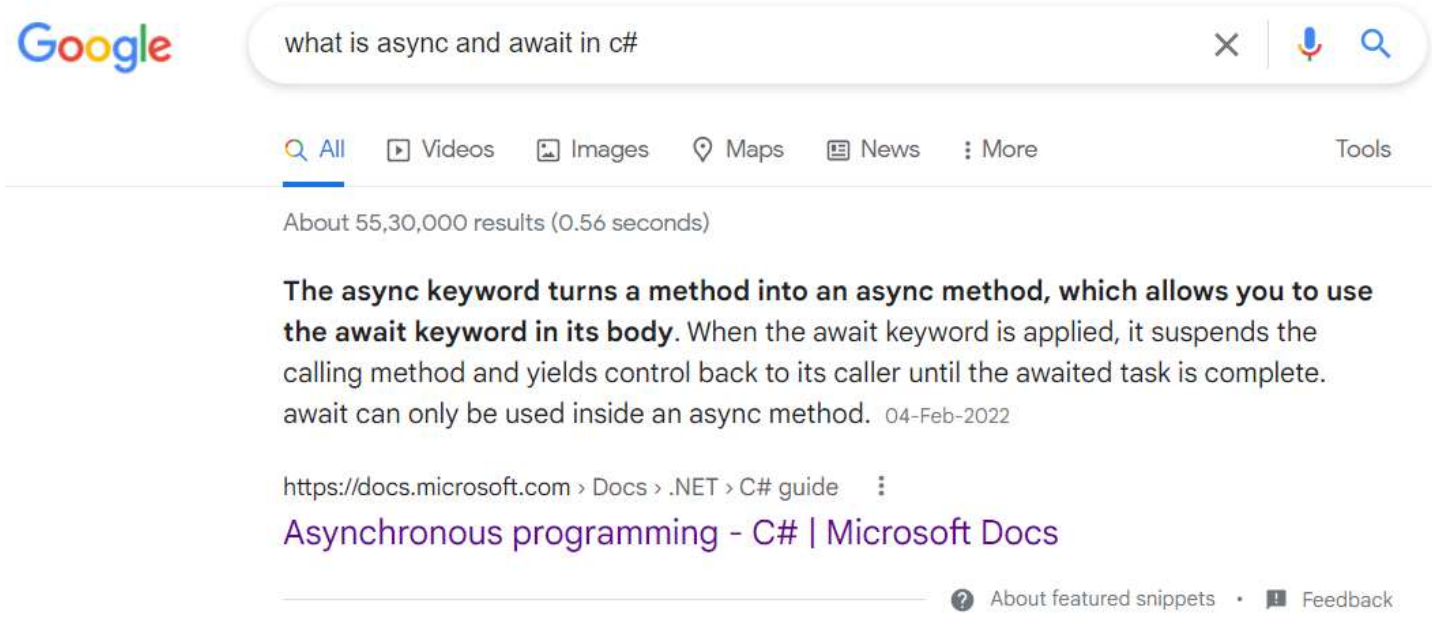
```
public class HelloWorld
{
    public static void Main(string[] args)
    {
        Task task1 = Task.Run(() => {
            int i = Method1();
            Console.WriteLine (i);
        });

        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

What is the Async and Await in task? V. IMP.




```
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

What is Reflection?

Reflection is the ability of a code to access the metadata of the assembly during runtime.

Metadata is information about data.

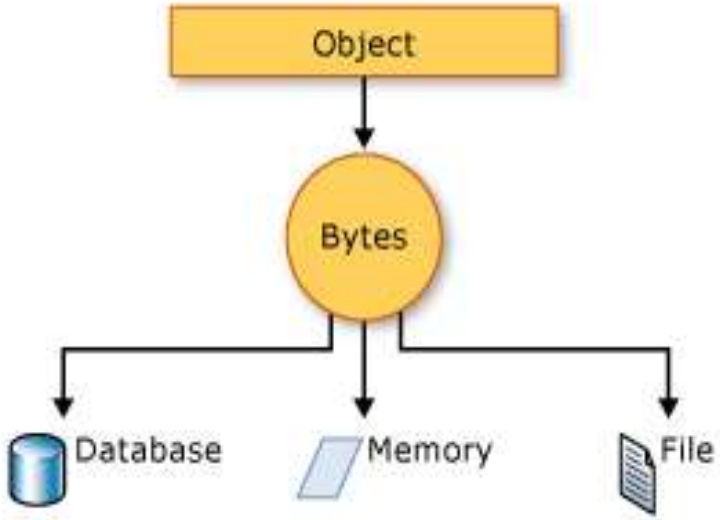
For example, a book is a data, then author name, book name, book size are metadata.

Suppose you have to check the version of the assembly at runtime then you can use reflection.

```
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);

    }
}
```

What is Serialization? V. IMP.

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.

```
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);
}
```

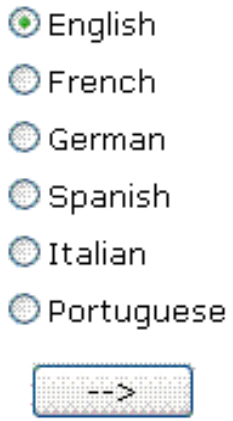
Q

What is meant by Globalization and Localization?

.NET Framework

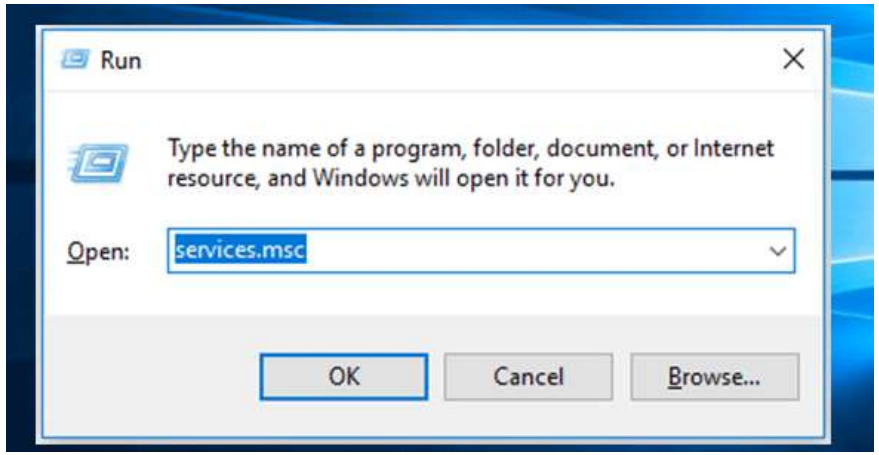
Globalization is the process of designing and developing a software product that functions in **MULTIPLE CULTURES/LOCALES**.

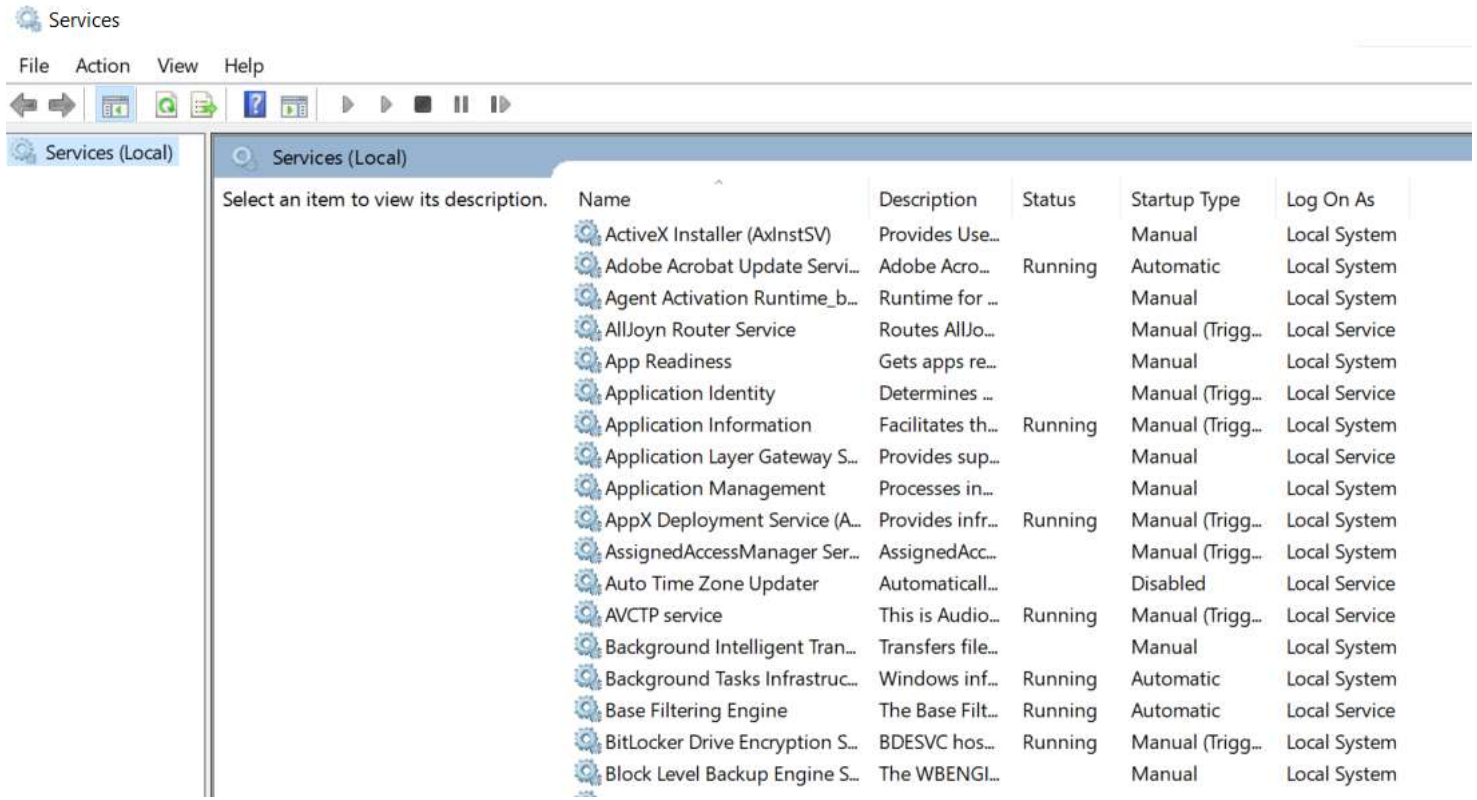
Localization is the process of adapting a globalized application, which you have already processed for localizability, to a particular culture/locale.



What are Window Services?

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





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

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

- JOINS IN SQL
- STORED PROCEDURES & FUNCTIONS
- TRIGGERS
- INDEXES
- SQL QUERIES

Q

What is the difference between DBMS and RDBMS?

Microsoft SQL Server

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, Microsoft Access etc.

Q What is a Constraint is SQL? What are its types. V. IMP.

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.

1. A **PRIMARY KEY** is a field which can uniquely identify each row in a table.

2. **NOT NULL** constraint tells that we cannot store a null value in a column.

3. A **FOREIGN KEY** is a field which can uniquely identify each row in an another table.

4. **CHECK** constraint helps to validate the values of a column to meet a particular condition.

5. **DEFAULT** constraint specifies a default value for the column when no value is specified by the user.

6. **UNIQUE** constraint tells that all the values in the column must be unique.

```
CREATE TABLE Students (
    ID int NOT NULL PRIMARY KEY,
    Name varchar(255) NOT NULL,
    CourseID int FOREIGN KEY REFERENCES Courses(CourseID),
    Age int NOT NULL CHECK (AGE >= 18),
    AdmissionDate date DEFAULT GETDATE(),
    CONSTRAINT UC_Student UNIQUE (ID,Name)
);
```

Q

What is the difference between Primary key and Unique key?

Microsoft SQL Server

	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.

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.

Locations -> LocationHist

Example of After(DML) Trigger

```
CREATE TRIGGER TR_UPD_Locations ON Locations
FOR UPDATE
NOT FOR REPLICATION
AS

BEGIN
    INSERT INTO LocationHist
    SELECT LocationID
        ,getdate()
    FROM inserted
END
```

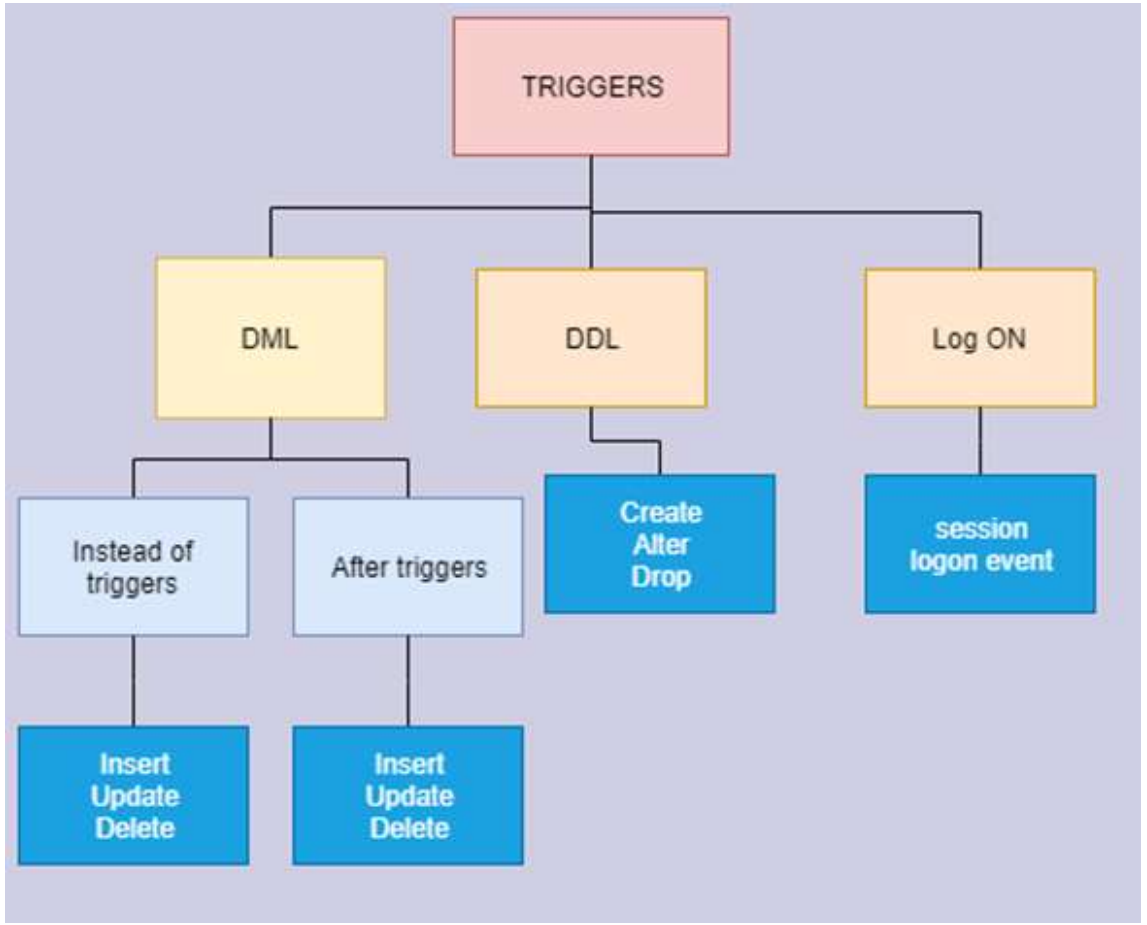
Table Name

Trigger Name

DML Event

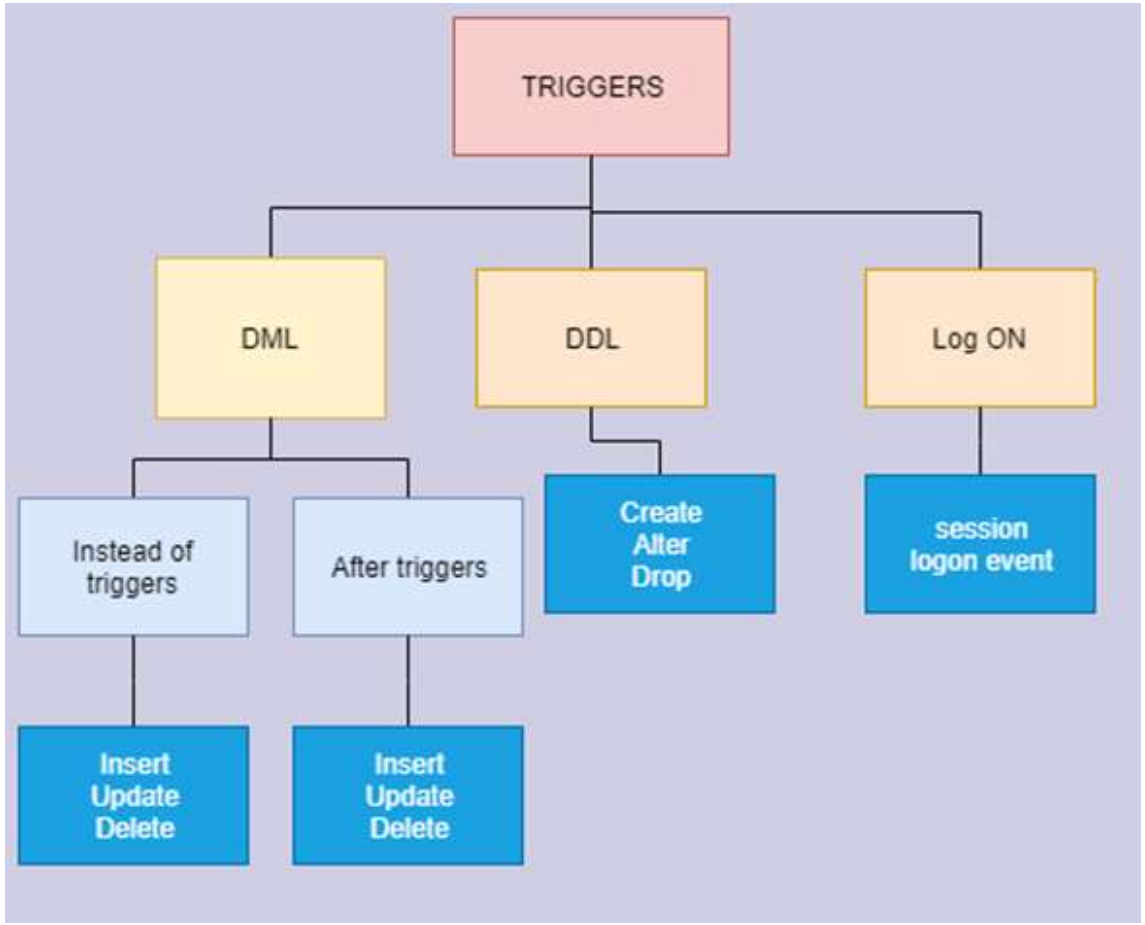
T-SQL block that runs against specified DML Event

In after trigger, Update on the table executed first and then trigger will run. The above example is of after trigger.



What Triggers and types of triggers?

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



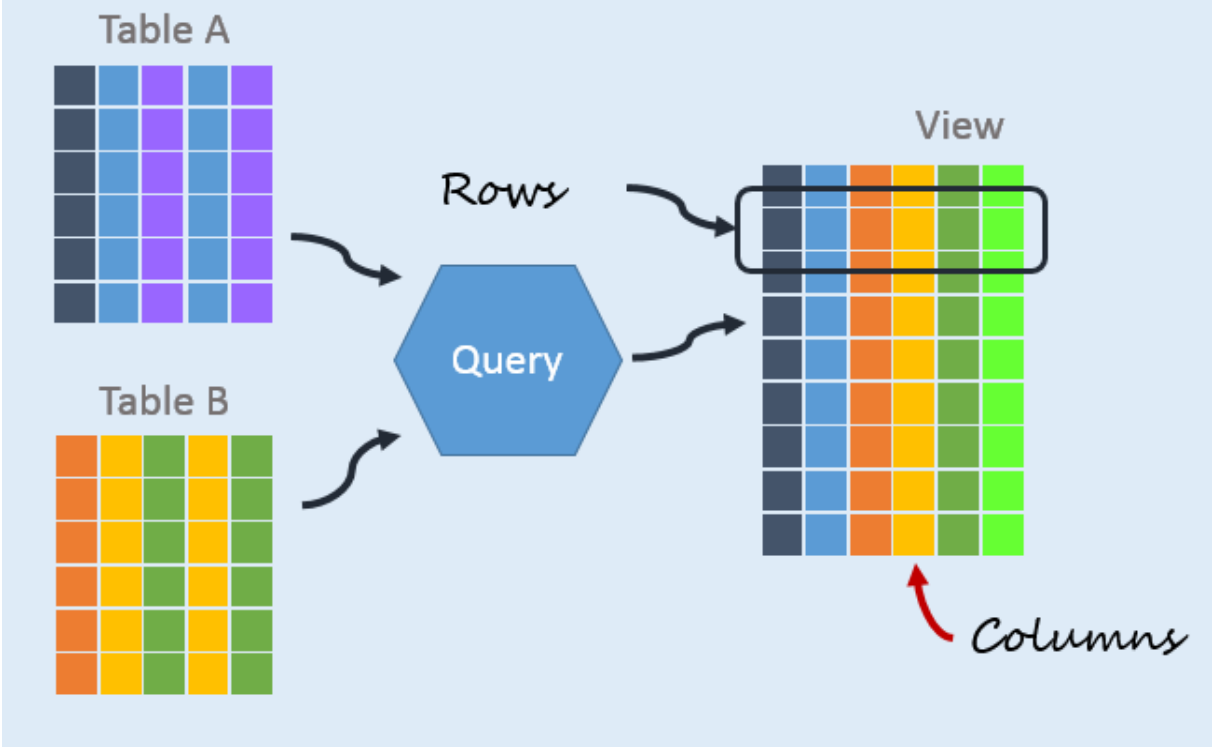
Example of Instead of(DML) Trigger

An INSTEAD OF trigger is a trigger that allows you to skip an INSERT , DELETE , or UPDATE statement to a table or a view and execute other statements defined in the trigger instead.

What is a View? V. IMP.

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

```
CREATE VIEW [India-Customers] AS

SELECT CustomerName, ContactName

FROM Customers

WHERE Country = 'India';
```



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

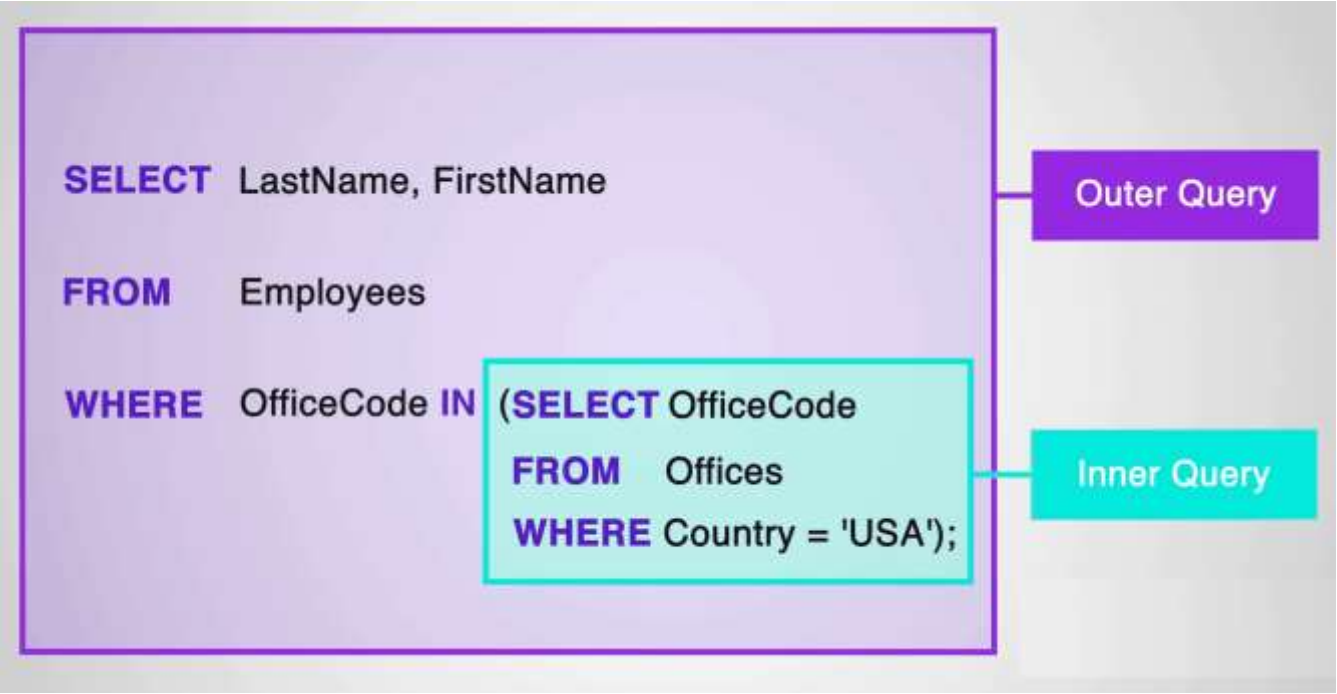
1. Indexed Views to improve the performance.
2. Extra security – DBA can hide the actual table names and expose views for Read operations.

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

Q

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.



Q

What is the difference between Delete, Truncate and Drop commands? V. IMP.

Microsoft SQL Server

DELETE

1. It is a DML.

2. It is used to delete the one or more rows(data) of a table.

3. It can be rollback.

```
DELETE FROM Employees
WHERE Emp_Id = 7;
```

TRUNCATE

1. It is a DDL.

2. It is used to delete all rows from the table.

3. It can be rollback.

4. Trincate will remove all the records from the table Employees but **not** the structure/ schema.

```
TRUNCATE TABLE Employees;
```

DROP

1. It is a DDL.

2. It is used to drop the whole table **with STRUCTURE/SCHEMA**.

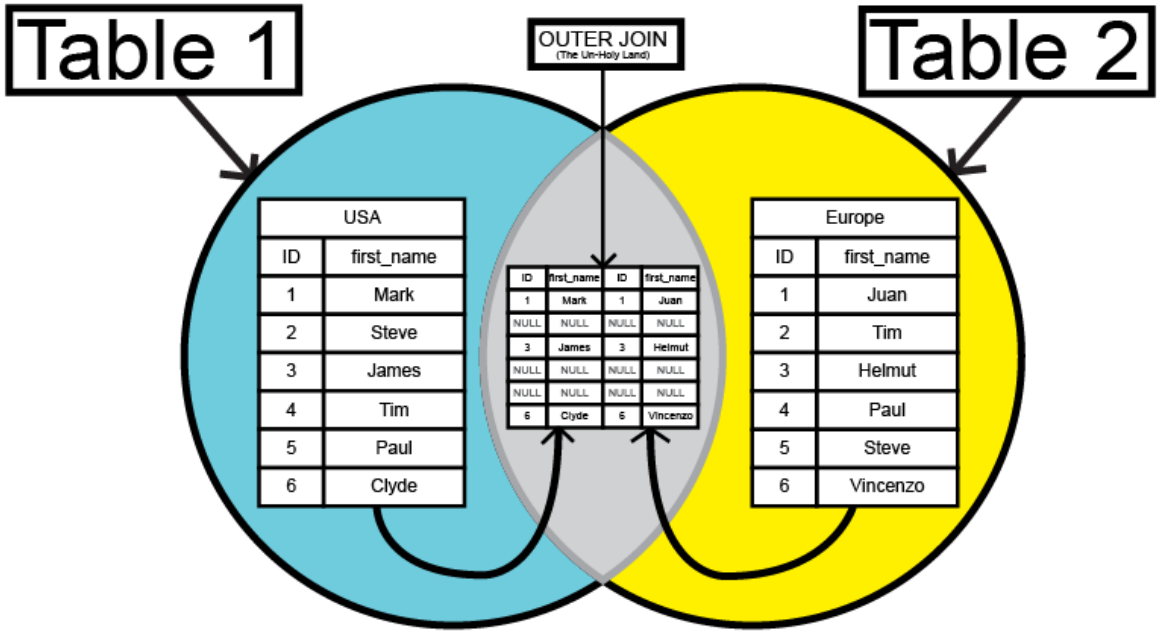
3. It **can not** be rollback.

4. It will remove the structure/ schema also.

```
DROP TABLE Employees;
```

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.

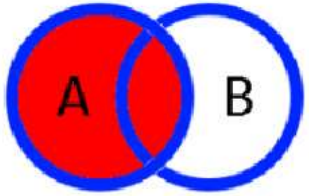


Q

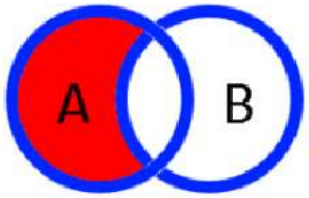
What are the types of Joins in SQL Server? V. IMP.

Microsoft SQL Server

LEFT OUTER JOIN

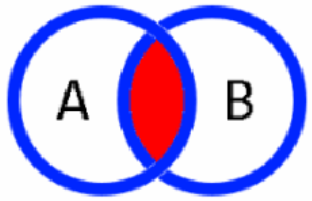



```
SELECT *
FROM TableA a
LEFT JOIN TableB b
    ON a.KEY = b.KEY
```



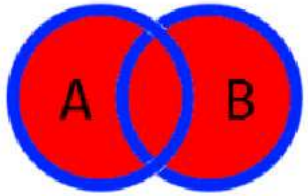

```
SELECT *
FROM TableA a
LEFT JOIN TableB b
    ON a.KEY = b.KEY
WHERE b.KEY IS NULL
```

INNER JOIN

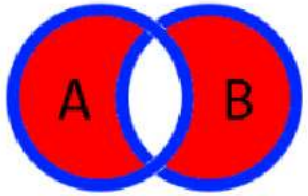



```
SELECT *
FROM TableA a
INNER JOIN TableB b
    ON a.KEY = b.KEY
```

FULL OUTER JOIN

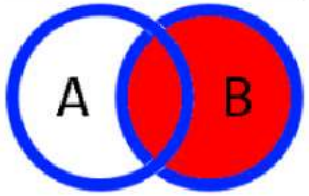



```
SELECT *
FROM TableA a
FULL OUTER JOIN TableB b
    ON a.KEY = b.KEY
```

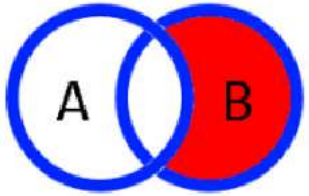



```
SELECT *
FROM TableA a
FULL OUTER JOIN TableB b
    ON a.KEY = b.KEY
WHERE a.KEY IS NULL
    OR b.KEY IS NULL
```

RIGHT OUTER JOIN




```
SELECT *
FROM TableA a
RIGHT JOIN TableB b
    ON a.KEY = b.KEY
```

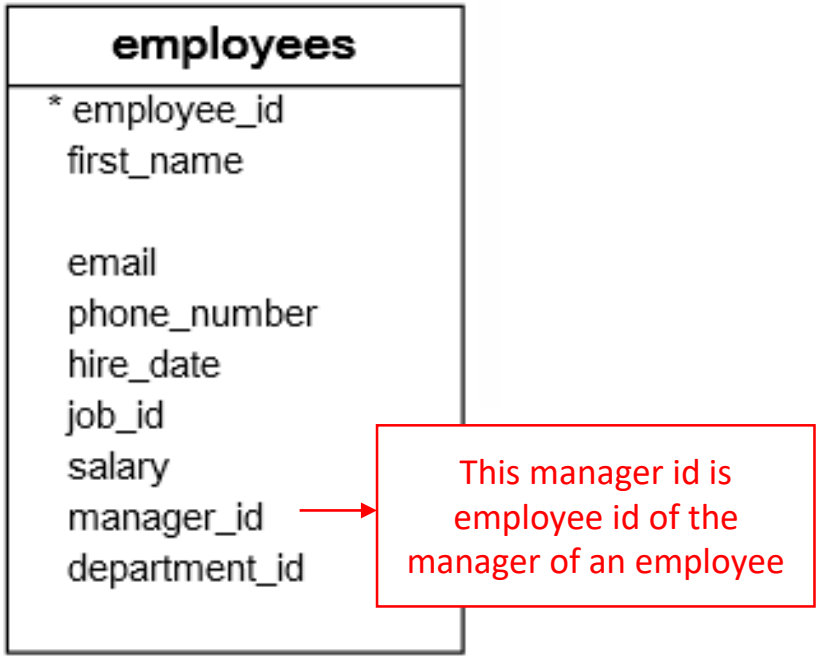



```
SELECT *
FROM TableA a
RIGHT JOIN TableB b
    ON a.KEY = b.KEY
WHERE a.KEY IS NULL
```

What is Self-Join?

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

When to use Self Join



Now your task is to get the employee name with his/her manager name.

```
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;
```

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

Q What is the difference between Stored Procedure and Functions (at least 3)? V. IMP.

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;
```

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.

5 step process:

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

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.

```
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;
```

Q

What is the difference between scope_identity and @@identity?

Microsoft SQL Server

Both are used to get the last value entered in the identity column of the table.

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

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()
```

Q

How to optimize a Stored Procedure or SQL Query?

Microsoft SQL Server

1. Use SET NOCOUNT ON

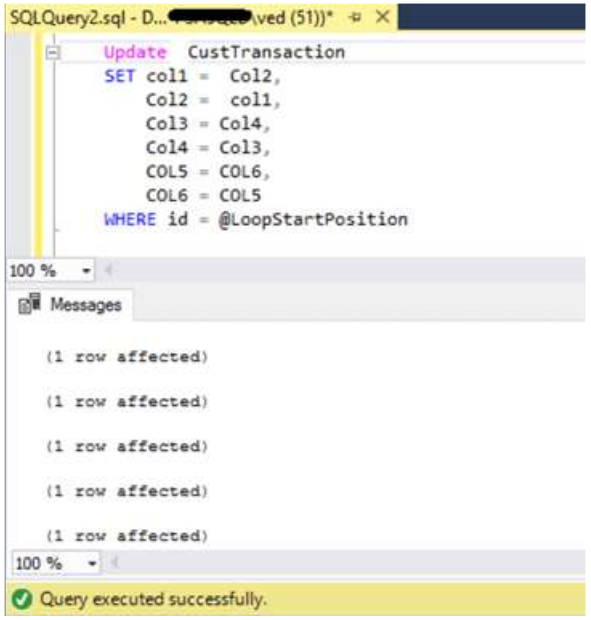
2. Specify column names instead of using * in SELECT statement.

3. Use schema name before objects or tablenames.
   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 :SELECT Count(1) FROM dbo.Employee
Example: IF(EXISTS (SELECT 1 FROM db.Employees))

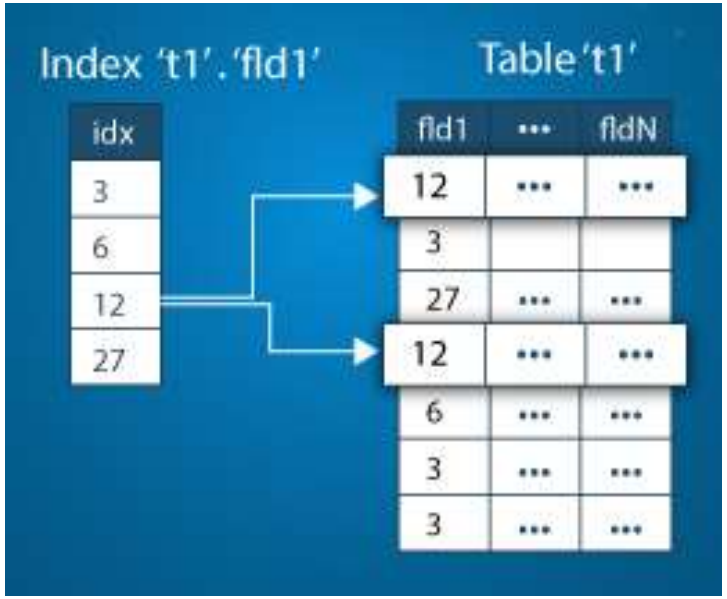
6. Use TRANSACTION when required only



Q

What are Indexes in SQL Server?

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



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

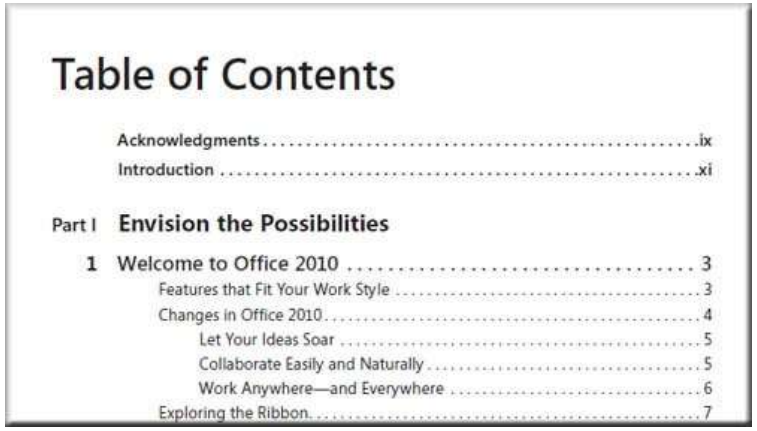


Table of Contents

Acknowledgments ix
Introduction xi

Part I Envision the Possibilities

1 Welcome to Office 2010 3
Features that Fit Your Work Style 3
Changes in Office 2010 4
Let Your Ideas Soar 5
Collaborate Easily and Naturally 5
Work Anywhere—and Everywhere 6
Exploring the Ribbon 7

What is Clustered index?

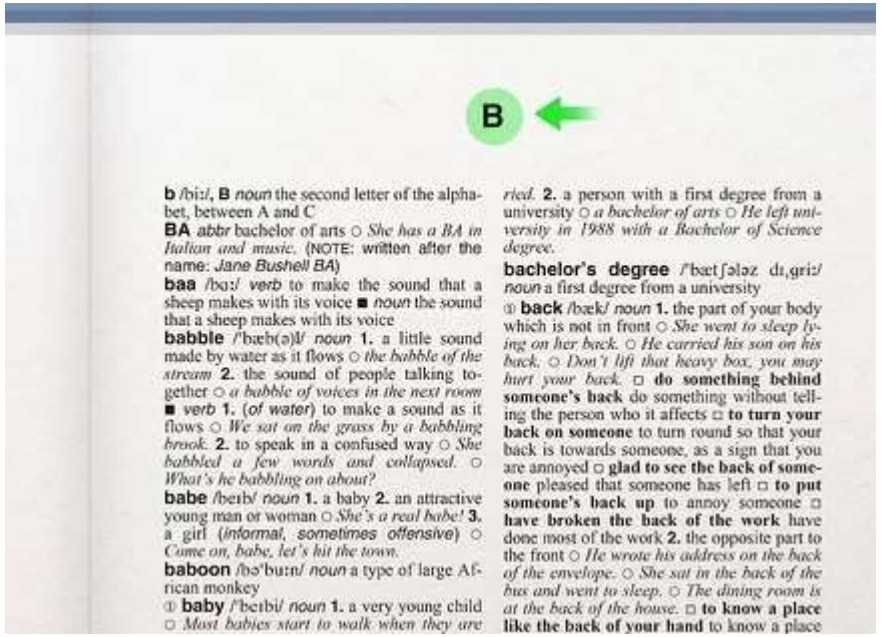
CLUSTERED INDEX

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

Table data can be sorted in only way, therefore, there can be only one clustered index per table.

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

Dictionary



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 /ˈbæb(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 /bəˈbuːn/ *noun* a type of large African monkey
① **baby** /ˈbeɪbi/ *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 /ˈbætʃələz 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

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.

A table can have multiple non-clustered index in a table.

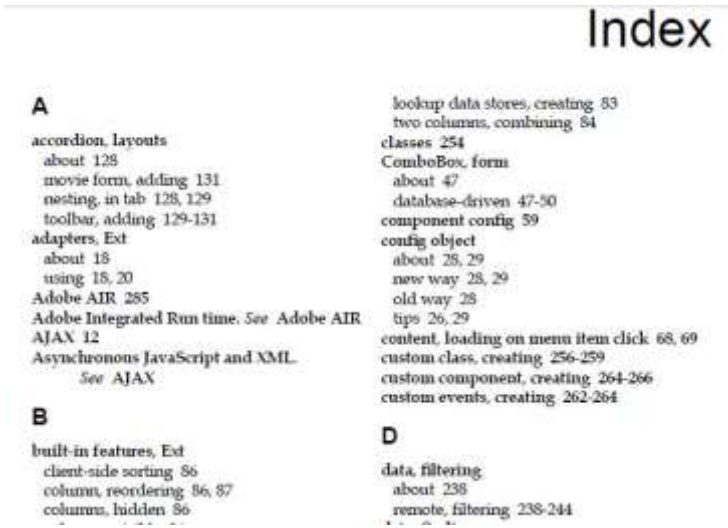
Book Index

Table of Contents

Acknowledgments ix
Introduction xi

Part I **Envision the Possibilities**

1 Welcome to Office 2010 3
Features that Fit Your Work Style 3
Changes in Office 2010 4
Let Your Ideas Soar 5
Collaborate Easily and Naturally 5
Work Anywhere—and Everywhere 6
Exploring the Ribbon 7



Index

A

accordion, layouts
about 128
movie form, adding 131
nesting, in tab 128, 129
toolbar, adding 129-131
adapters, Ext
about 18
using 18, 20
Adobe AIR 285
Adobe Integrated Run time. *See* Adobe AIR
AJAX 12
Asynchronous JavaScript and XML.
See AJAX

B

built-in features, Ext
client-side sorting 86
column, reordering 86, 87
columns, hidden 86
lookup data stores, creating 83
two columns, combining 84
classes 254
ComboBox, form
about 47
database-driven 47-50
component config 59
config object
about 28, 29
new way 28, 29
old way 28
tips 26, 29
content, loading on menu item click 68, 69
custom class, creating 256-259
custom component, creating 264-266
custom events, creating 262-264

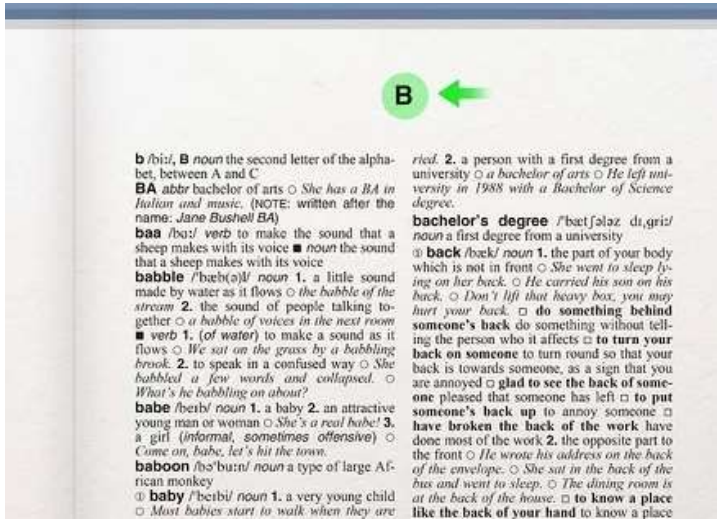
D

data, filtering
about 238
remote, filtering 238-244

Q What is the difference between Clustered and Non-Clustered index? V. IMP.

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.



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

2. A table can have only one clustered index.

A table can have multiple non-clustered index.

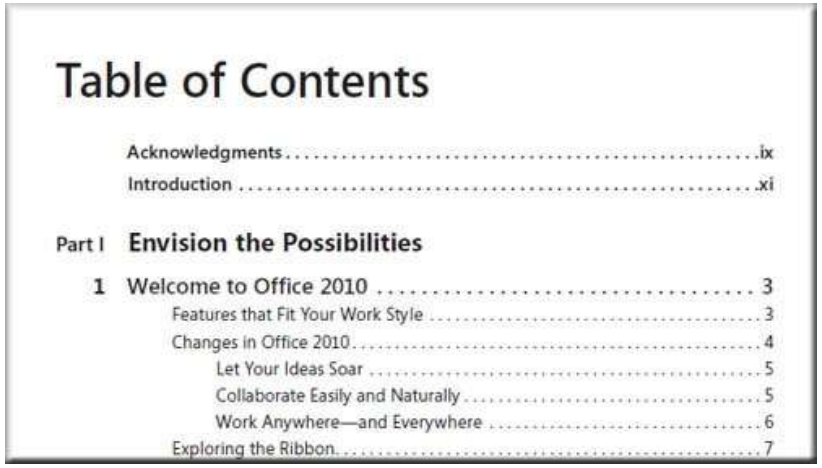


Table of Contents

Acknowledgments ix
Introduction xi

Part I **Envision the Possibilities**

1 Welcome to Office 2010 3
Features that Fit Your Work Style 3
Changes in Office 2010 4
Let Your Ideas Soar 5
Collaborate Easily and Naturally 5
Work Anywhere—and Everywhere 6
Exploring the Ribbon 7

3. Clustered index is faster.

Non-clustered index is slower.

Q

How to create Clustered and Non-Clustered index in a table?

Microsoft SQL Server

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)
```

Q

In which column you will apply the indexing to optimize this query. “select id, class from student where name=”happy””?

Microsoft SQL Server

select id, class from student where name=”happy”

The column **after WHERE** condition, which is “NAME” here.

Q

Write a SQL query to fetch all the Employees who are also Managers.

Microsoft SQL Server

employees
* employee_id
first_name
email
phone_number
hire_date
job_id
salary
manager_id
department_id

```
SELECT

    e.first_name AS employee,
    m.first_name AS manager

FROM

    employees e
        INNER JOIN
    employees m ON m.employee_id = e.manager_id
```

	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

Q How to get the Nth highest salary of an employee? V. IMP.

Salary
5000
10000
6000
4000
2000
7000

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

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

10000
7000
6000

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

    ) RESULT
ORDER BY SALARY
```

Result
6000
7000
10000

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

    ) RESULT
ORDER BY SALARY
```

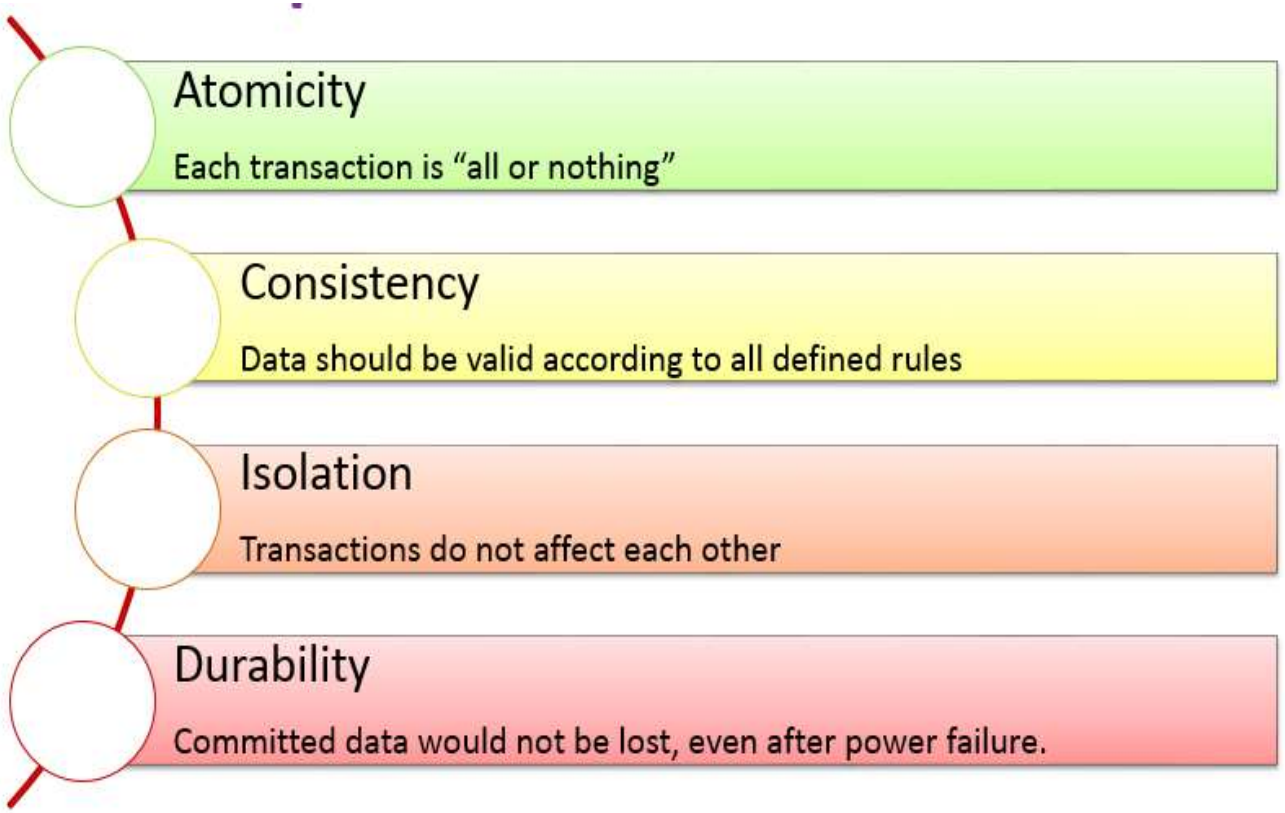
Result
6000

Q

What are ACID properties?

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

For example, multiple inserts are coming at same point of time.



Q

What is Auto Increment/ Identity column in SQL Server?

Microsoft SQL Server

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

Mostly it is the primary key only.

```
CREATE TABLE Persons (
    Personid int IDENTITY(1,1) PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

Q

What is the difference between Having clause and Where clause? V. IMP.

Microsoft SQL Server

WHERE Clause is used before GROUP BY Clause.

HAVING Clause is used after GROUP BY Clause.

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

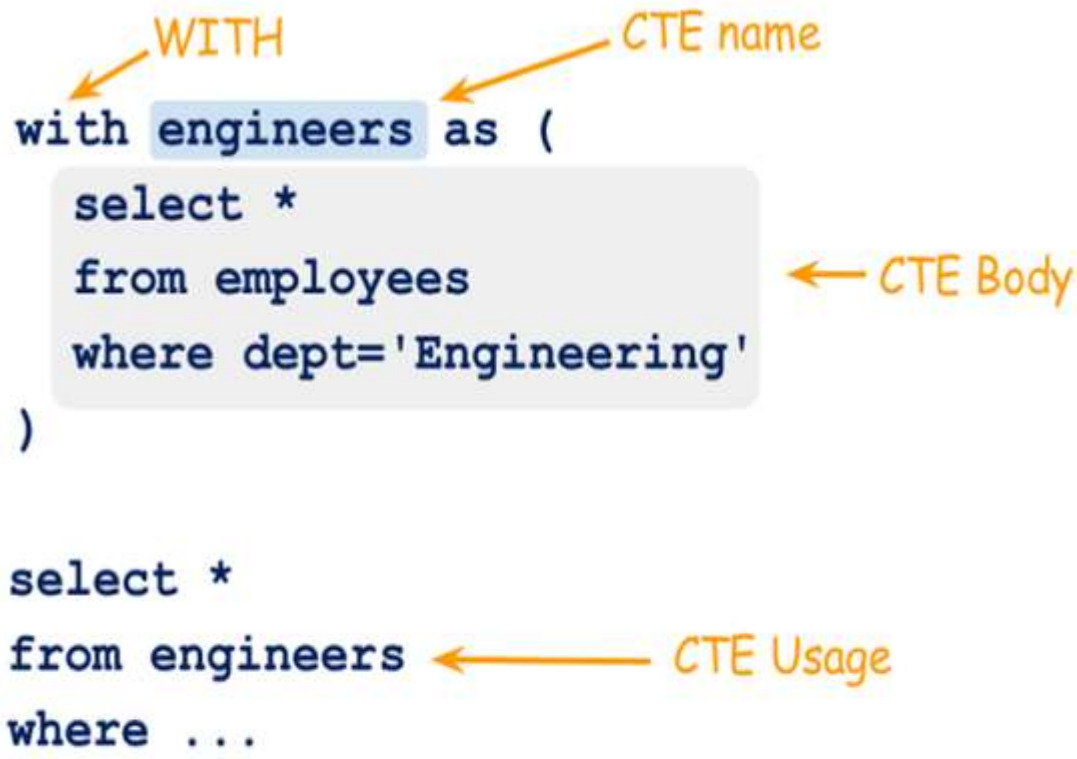
WHERE Clause cannot contain AGGREGATE function.

HAVING Clause can contain aggregate function.

```
select EmpName from Employee
GROUP BY EmpName
HAVING SUM(EmpSalary) <30000
```

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.



Q

What are Magic Tables in SQL Server?

Microsoft 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 updated row gets stored in INSERTED magic table and the old row or previous row gets stored in the DELETED magic table.

The use of magic tables are TRIGGERS.

- MVC BASICS
- ACTION METHODS
- FILTERS
- ROUTING
- MORE

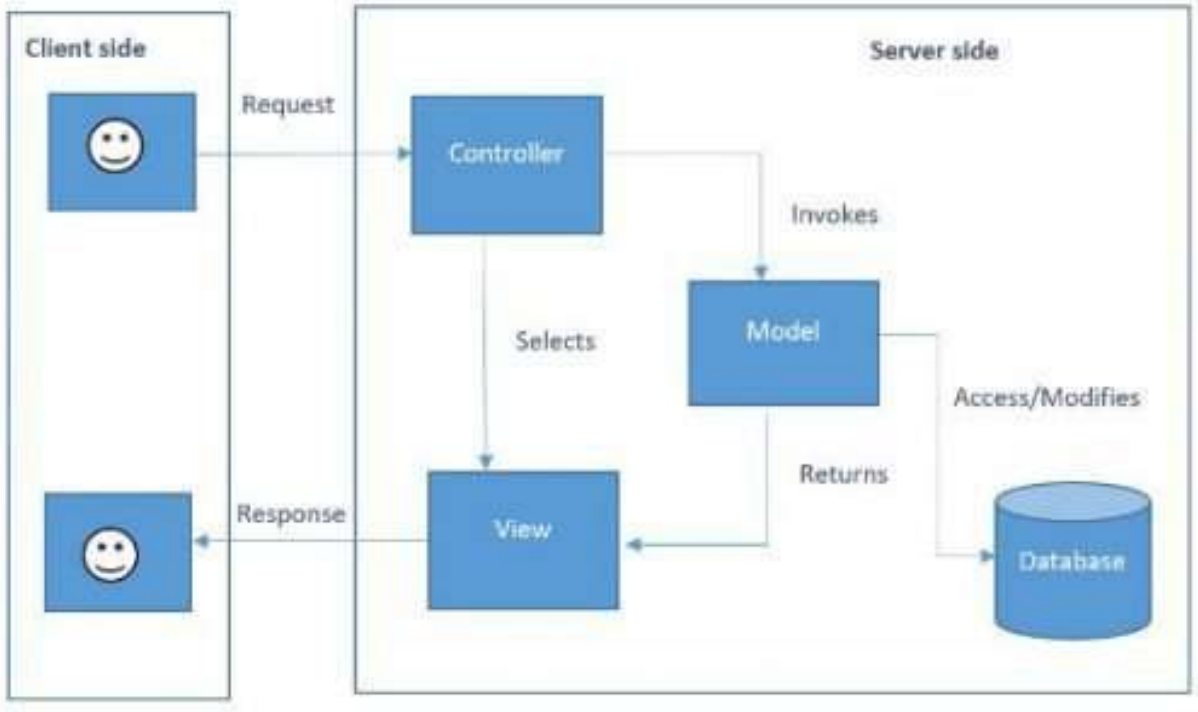
Q

What is MVC (Model View Controller)? Explain MVC Life cycle.

ASP.NET MVC

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.



Q

What are the advantages of MVC over Web Forms (atleast 3)? V. IMP.

ASP.NET 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.

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.

Q

What are the advantages of MVC over web forms (atleast 3)?

ASP.NET MVC

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.

Q

What are the different return types of a controller Action method?

ASP.NET MVC

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.

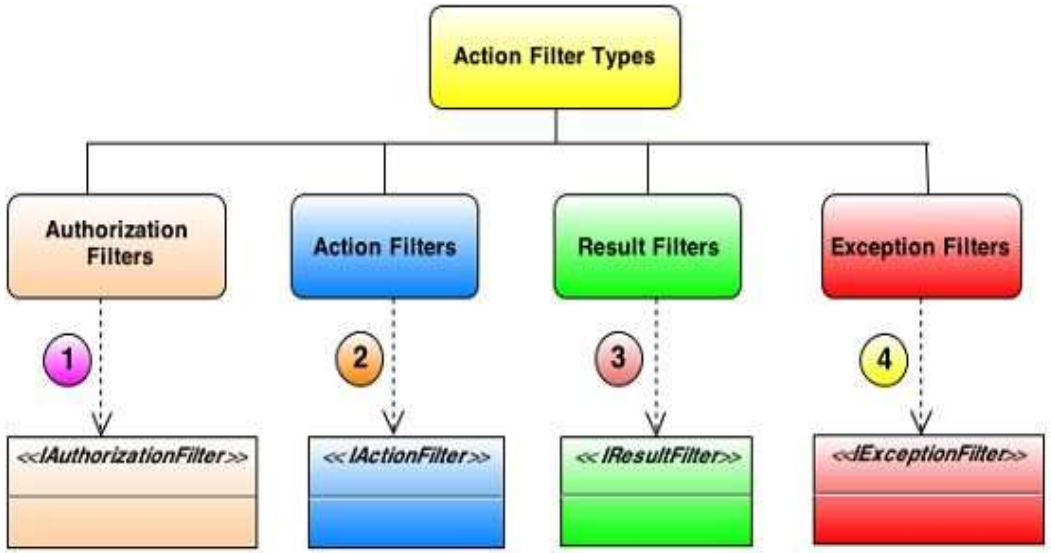
Q

What are Filters and their types in MVC?

ASP.NET MVC

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

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



Q

What are Filters and their types in MVC?

ASP.NET MVC

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

Q

What is Authentication and Authorization in ASP.NET MVC? V. IMP.

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 precedes to Authorization.

What are the types of Authentication in ASP.NET MVC?

ASP.NET MVC

FORM BASED AUTHENTICATION

Form authentication is based on cookies, the authentication and permission settings are stored in cookies. Example: Login page

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

PASSPORT AUTHENTICATION

Passport authentication is a centralized authentication service provided by Microsoft.

WINDOWS AUTHENTICATION

We use windows authentication when we are creating a web application for a limited number of users who already have Windows account.

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

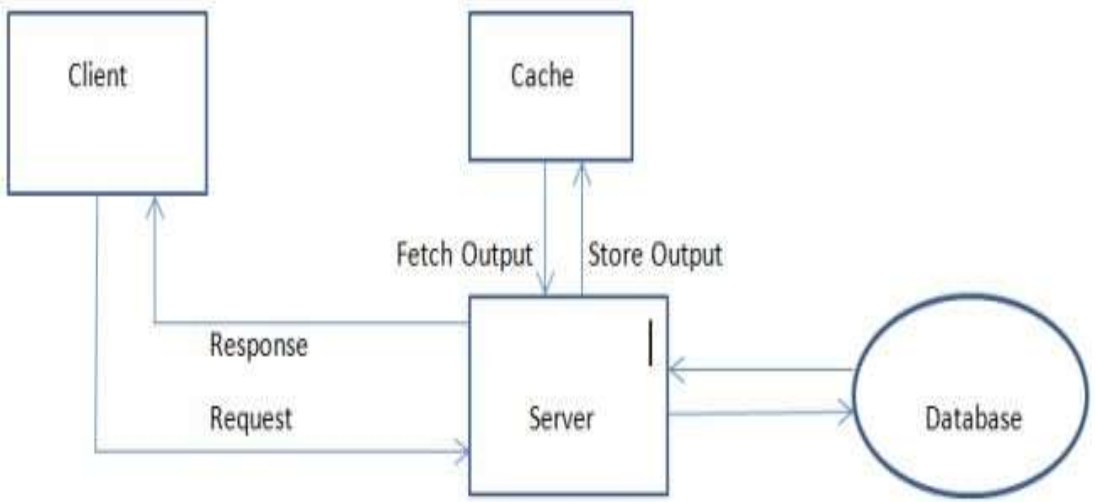
Q

What is Output Caching in MVC? How to implement it?

ASP.NET MVC

The output cache enables you to cache the content returned by a controller action.
This is kind of Result filter

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

    return View();
}
```



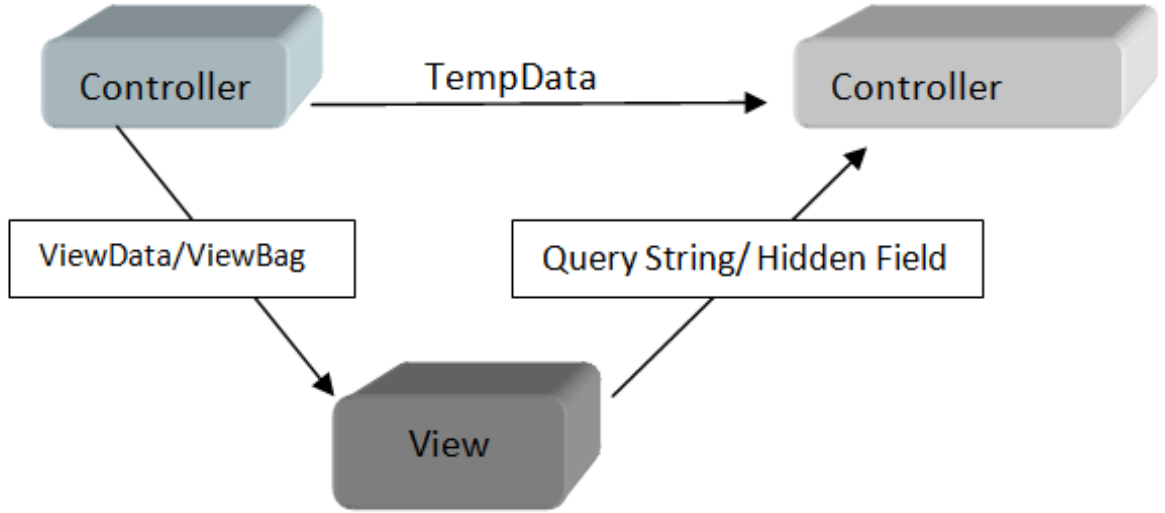
Q

What is the difference between ViewData, ViewBag & TempData? V. IMP.

ASP.NET MVC

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.

ViewBag DOESN'T REQUIRE TYPECASTING for the complex data type.

Q

How can we pass the data from controller to view in MVC?

ASP.NET MVC

By using ViewData and ViewBag

Q

What is Partial View?

ASP.NET MVC

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

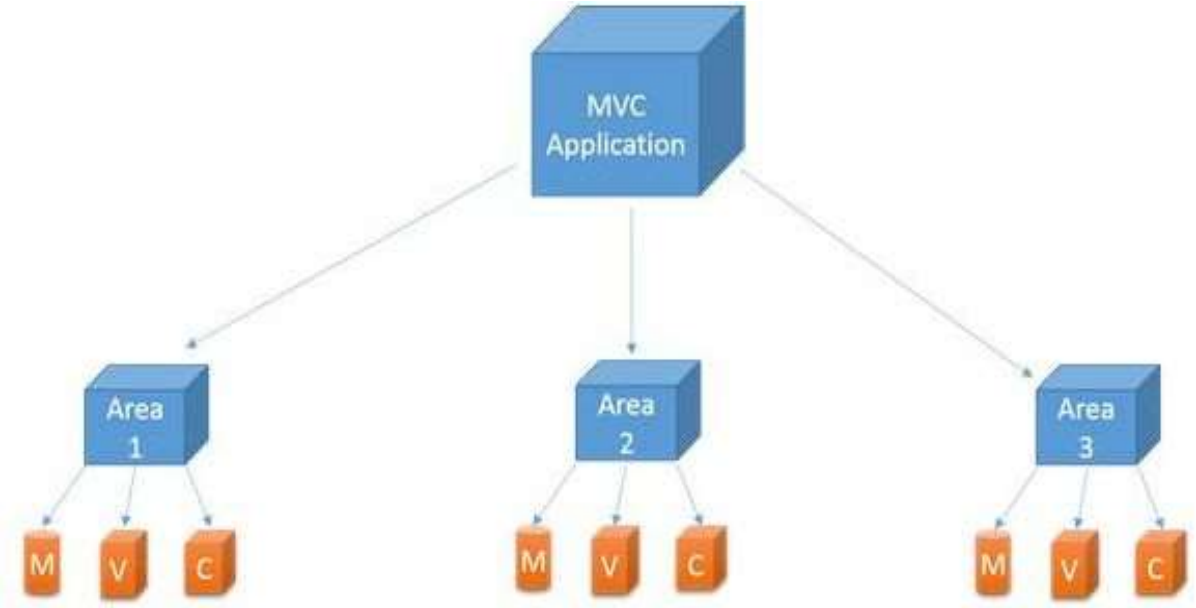
It is just like a user control of a web form application. Partial can be reusable in multiple views. It helps us to reduce code duplication.

Q

What are Areas in MVC?

ASP.NET MVC

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



Q

How Validation works in MVC?

ASP.NET MVC

Validation can be done 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; }
}
```

Q

Explain the concept of MVC Scaffolding?

ASP.NET MVC

Scaffolding is a code generation framework for ASP.NET Web applications. Using scaffolding can reduce the amount of time to develop standard data operations in your project.

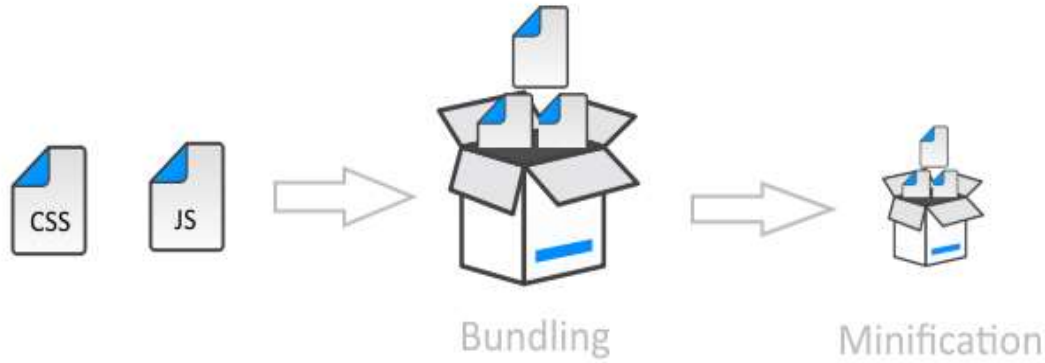
Q

What is Bundling and Minification in MVC?

ASP.NET 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.



Q

How to implement Security in web applications in MVC? V. IMP.

ASP.NET MVC

1) Error Handling Must Setup CUSTOM ERROR PAGE - Intercept too - Burp Suite is a cyber security tool.

2) Cross-Site Request Forgery (CSRF) – AntiForgeryToken

3) Cross-Site Scripting (XSS) attacks

4) Malicious FILE Upload.

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

6) TOKEN based authentication - Web api and application

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

8) Use HTTPS

PAGE LIFE CYCLE

SESSION

CACHING

Q What are the events in Page Life Cycle? In which event are the controls fully loaded? V. IMP.

Microsoft ASP.net Webforms

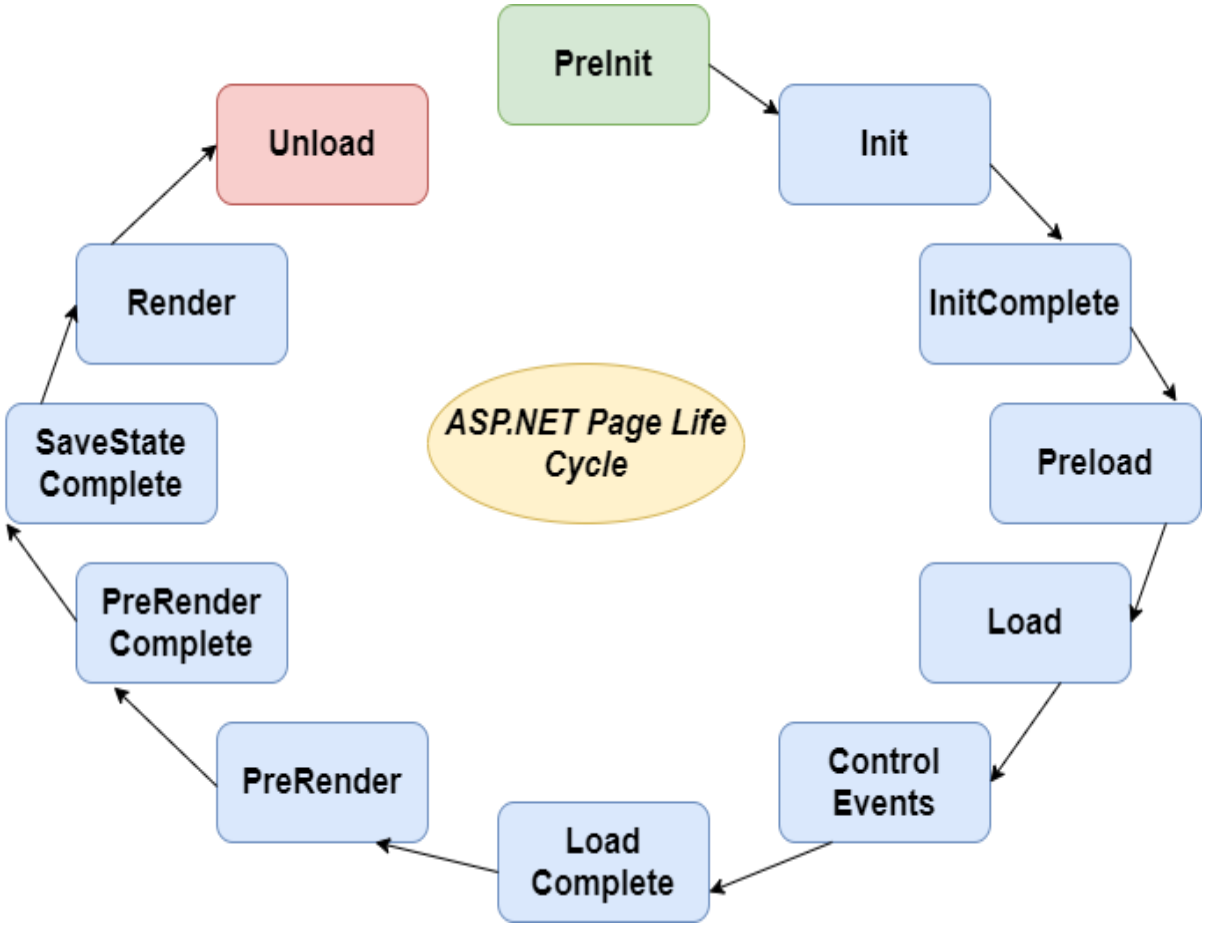
Init - This event fires after each control has been initialized.

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.

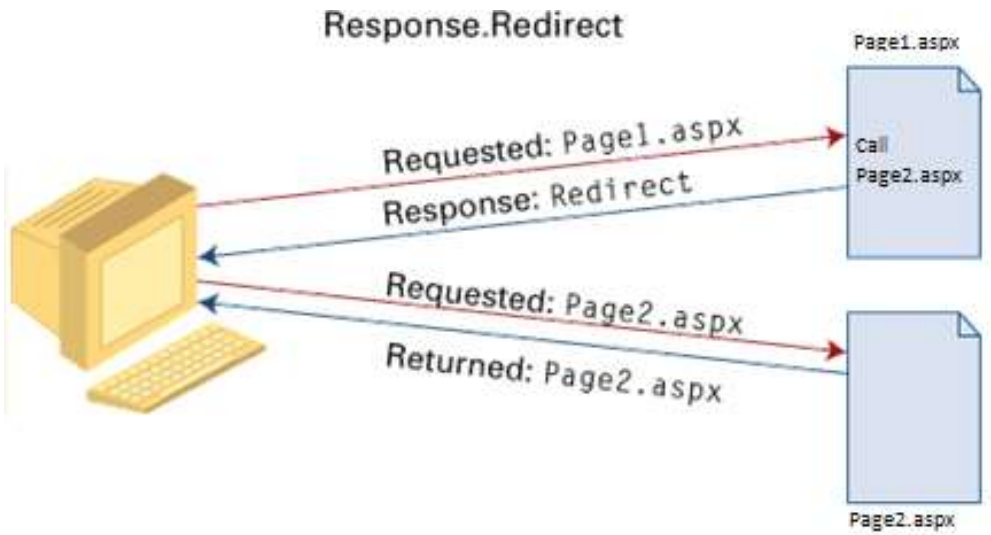
What is the difference between Server.Transfer() and Response.Redirect()?

Microsoft ASP.net Webforms

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

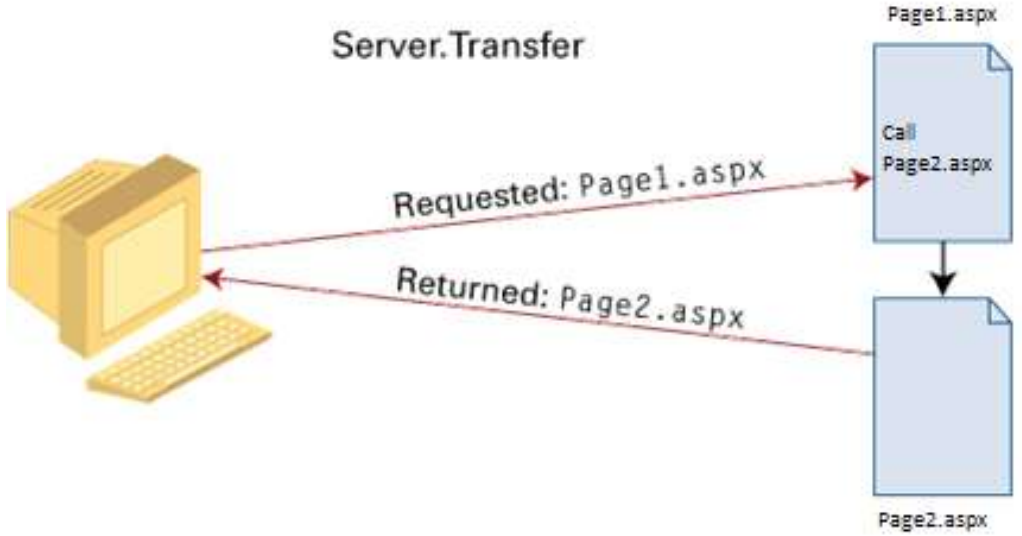
Response.Redirect uses round trip back to the client for redirecting the page.

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



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.



Q

Where the ViewState is stored after the page postback?

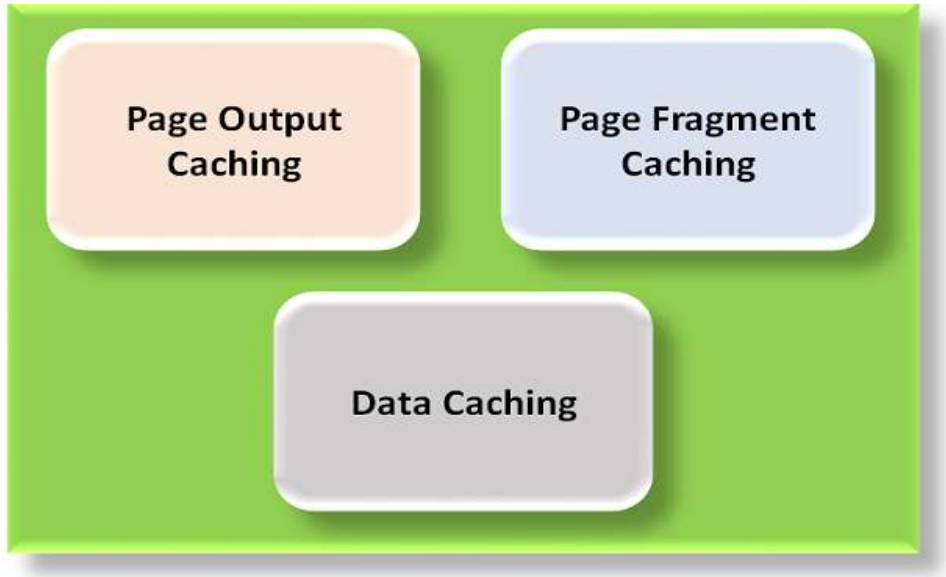
Microsoft ASP.net Webforms

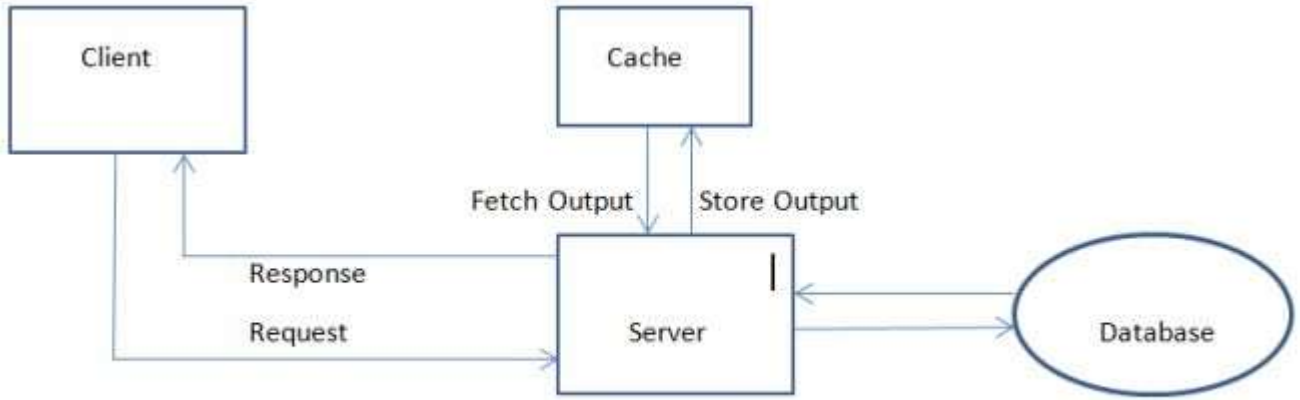
ViewState is stored in a HIDDEN FIELD on the page at client side.

Q

What are the different types of Caching?

Microsoft ASP.net Webforms





Q

What are the different Session state management options available in asp.net?

Microsoft ASP.net Webforms

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

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

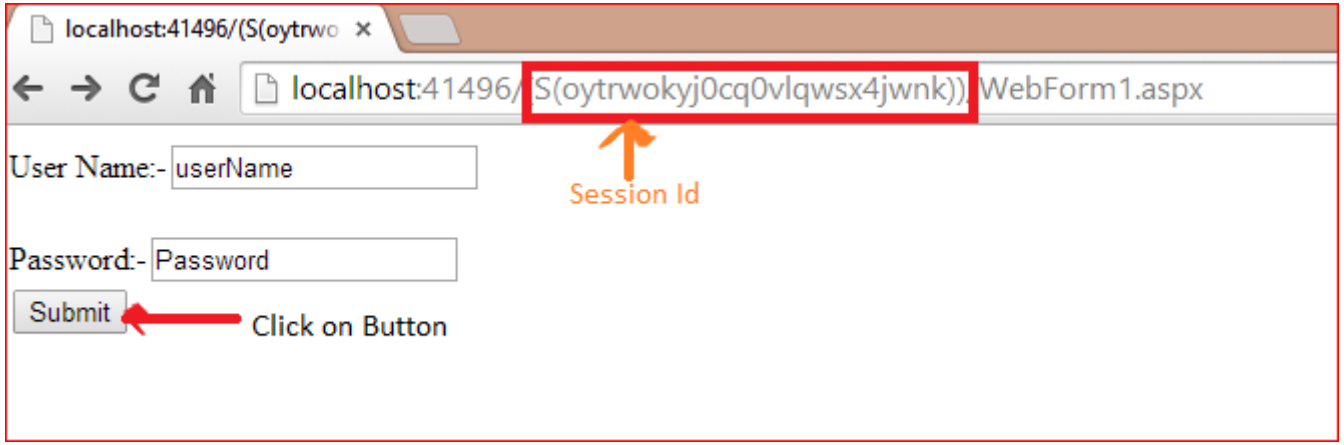
2. Out-of-Process - Stores the seesion data in an **external server**. The external server may be either a SQL Server or a State Server.

What is cookie less session?

Microsoft ASP.net Webforms

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

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



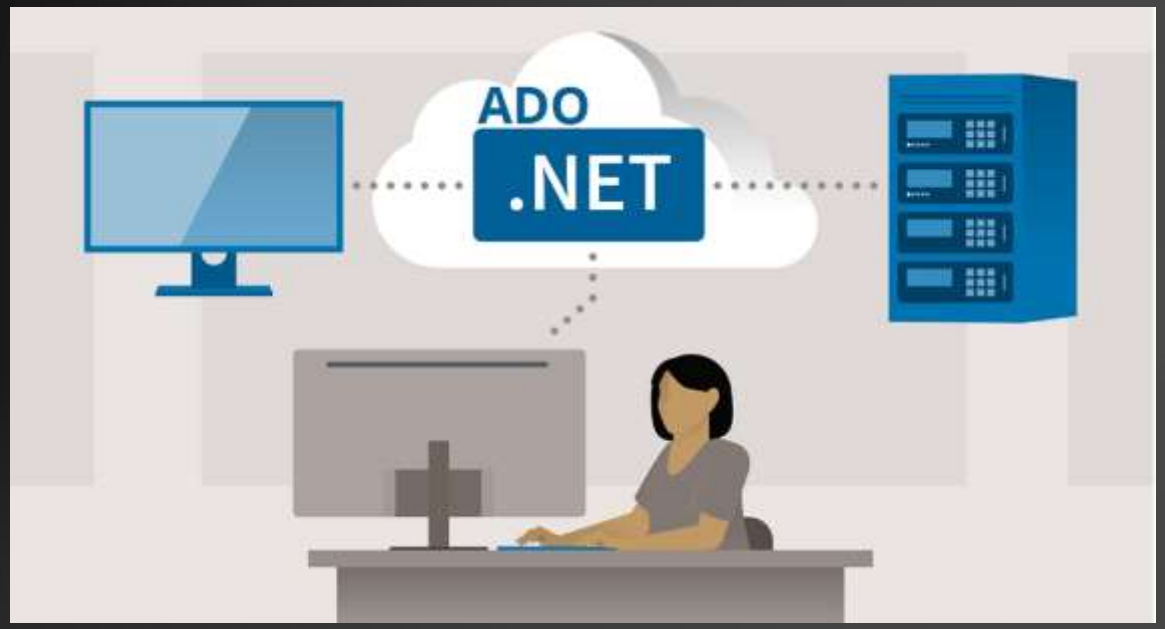
To enable a cookie-less session, we need to change some configuration in the WEB.CONFIG file.

Q

How to force all the validation controls to run in a page in web forms?

Microsoft ASP.net Webforms

PAGE.VALIDATE()



ARCHITECTURE

ADO.NET COMPONENTS

EXECUTE METHODS

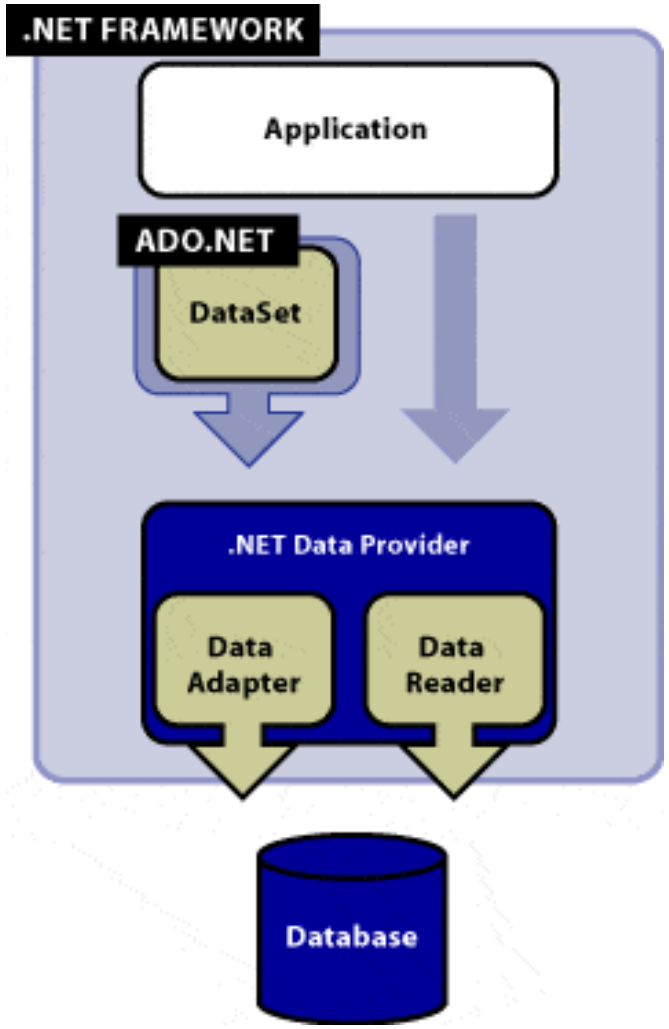
Q

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.



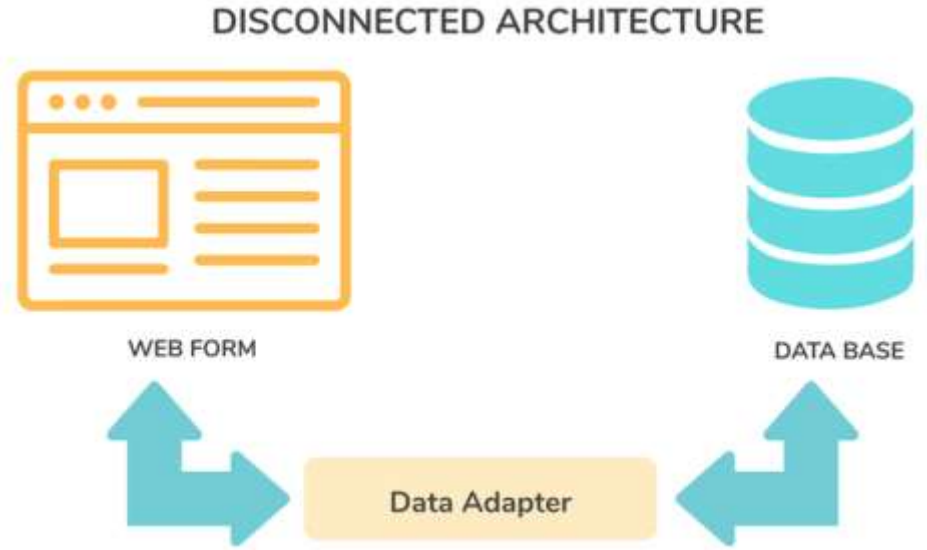
Q

What is Connected architecture and Disconnected architecture?

ADO.NET

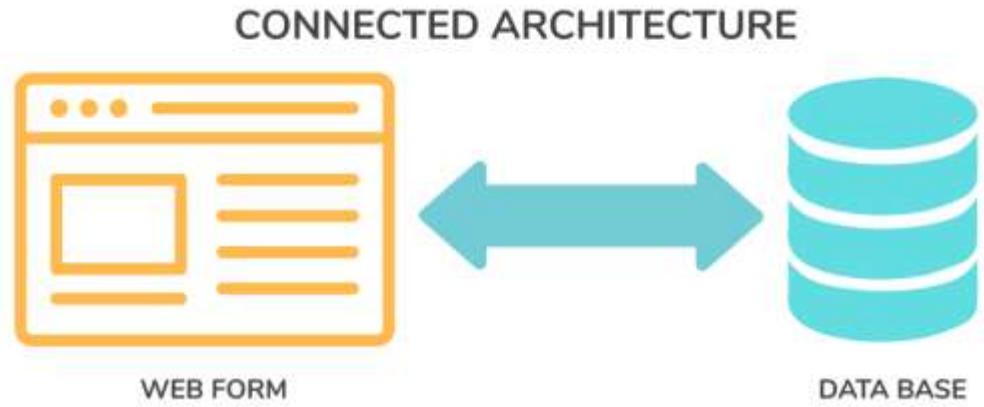
Disconnected Architecture

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.



Q

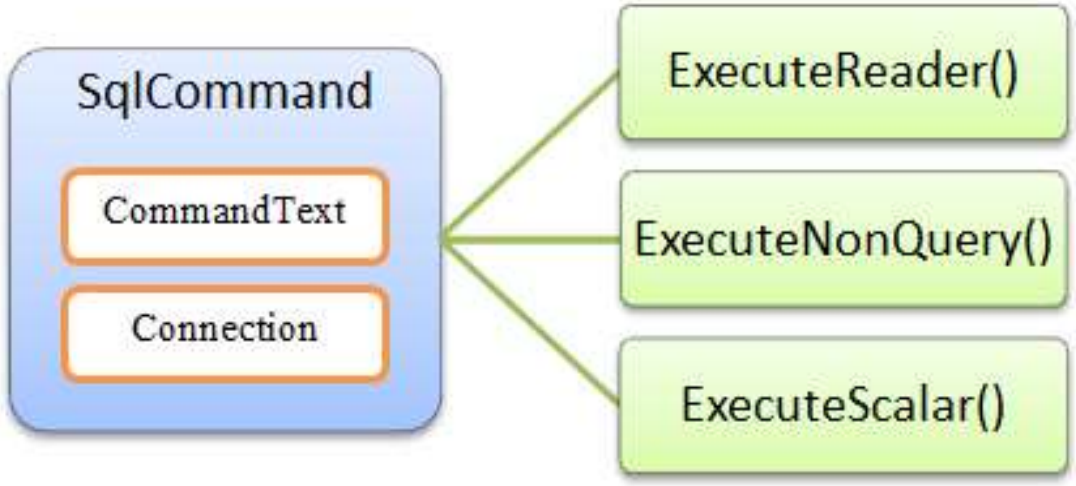
What are the different Execute Methods of ADO.NET?

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.



Q

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.

Entity Framework

- DIFFERENCE ADO.NET & EF
- DBCONTEXT & DBSET
- EF APPROACHES

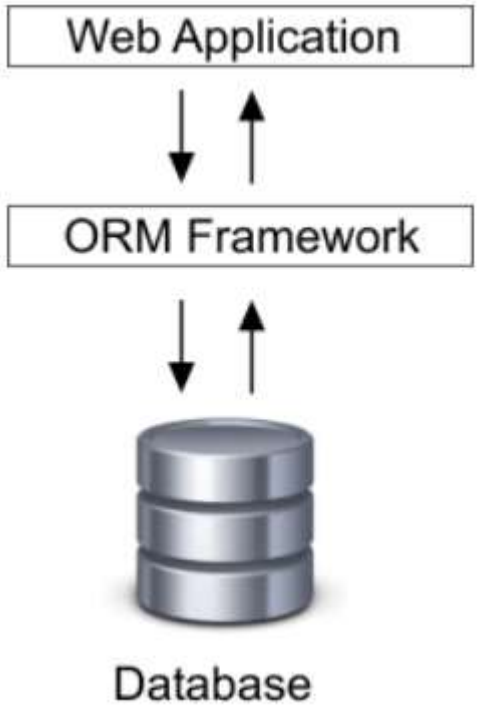
Q

What is ORM? What are the different types of ORM?

EF Core

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

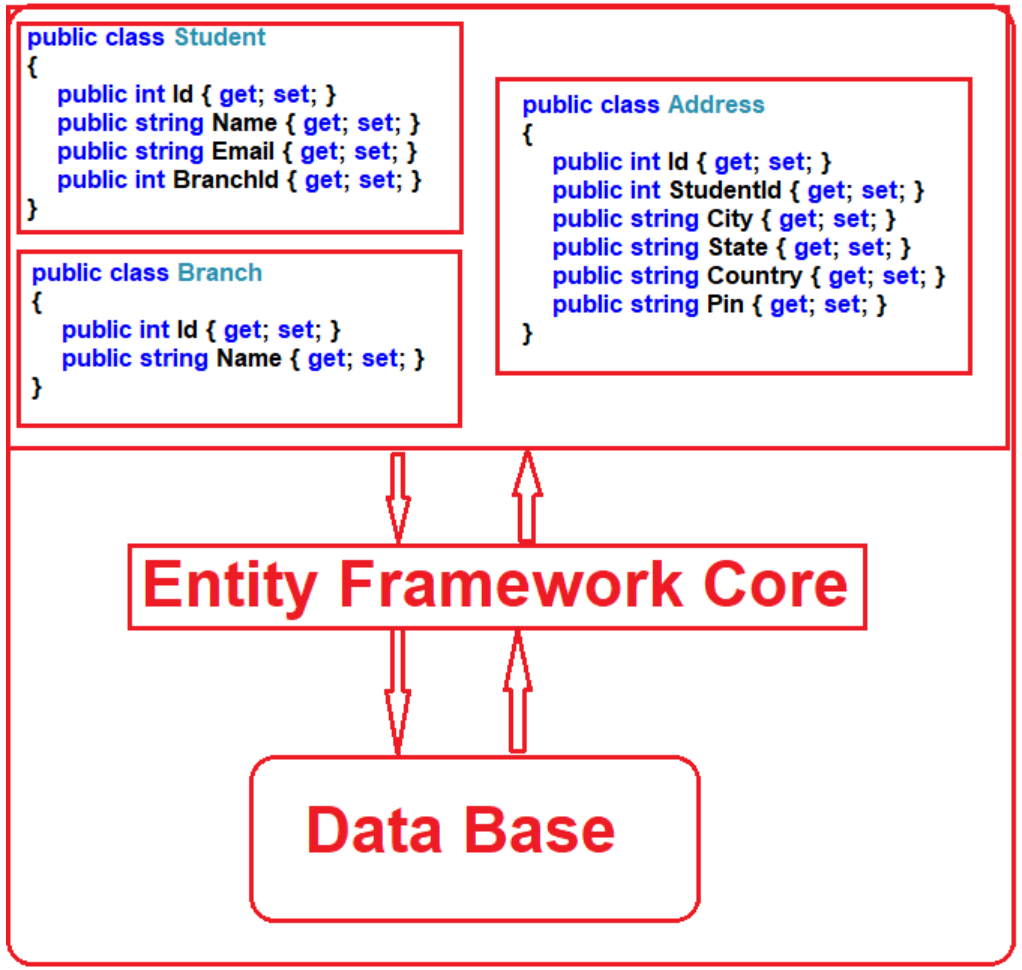
What is Entity Framework?

EF Core

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.



Q

How will you differentiate ADO.NET from Entity Framework?

EF Core

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();
         }
      }
  }
```

Q

How will you differentiate ADO.NET from Entity Framework?

EF Core

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.

How Entity Framework works? OR How to setup EF?

Add records to 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

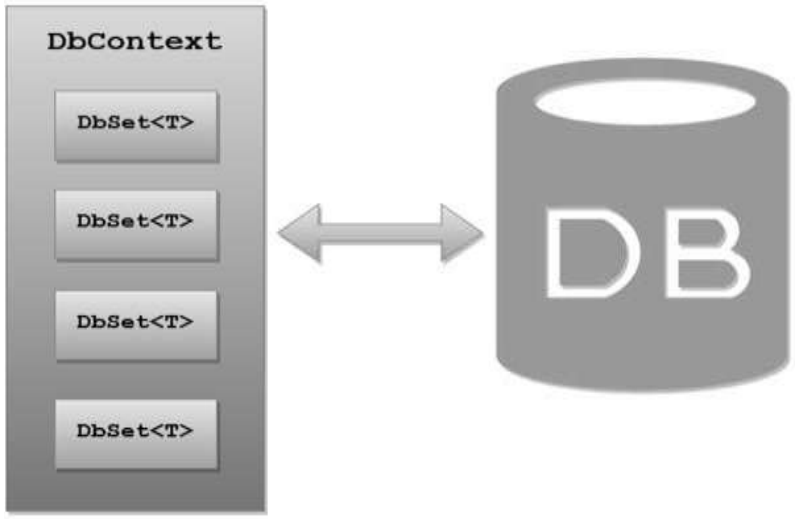
```
public class SchoolContext : DbContext
    {
        public SchoolContext() : base("SchoolContext")
        {
        }

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

3. 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();

        }

    }
```

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.




```
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

The DbSet class represents an **entity set** that can be used for create, read, update, and delete operations.

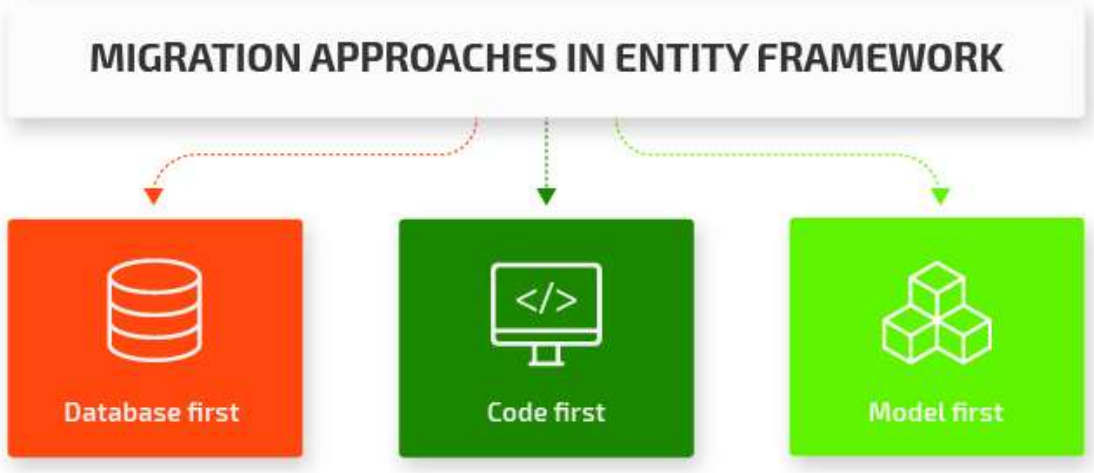
Q

What are the different types of Approaches used in Entity Framework?

EF Core

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.

Q

What is the difference between LINQ to SQL and Entity Framework?

EF Core

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

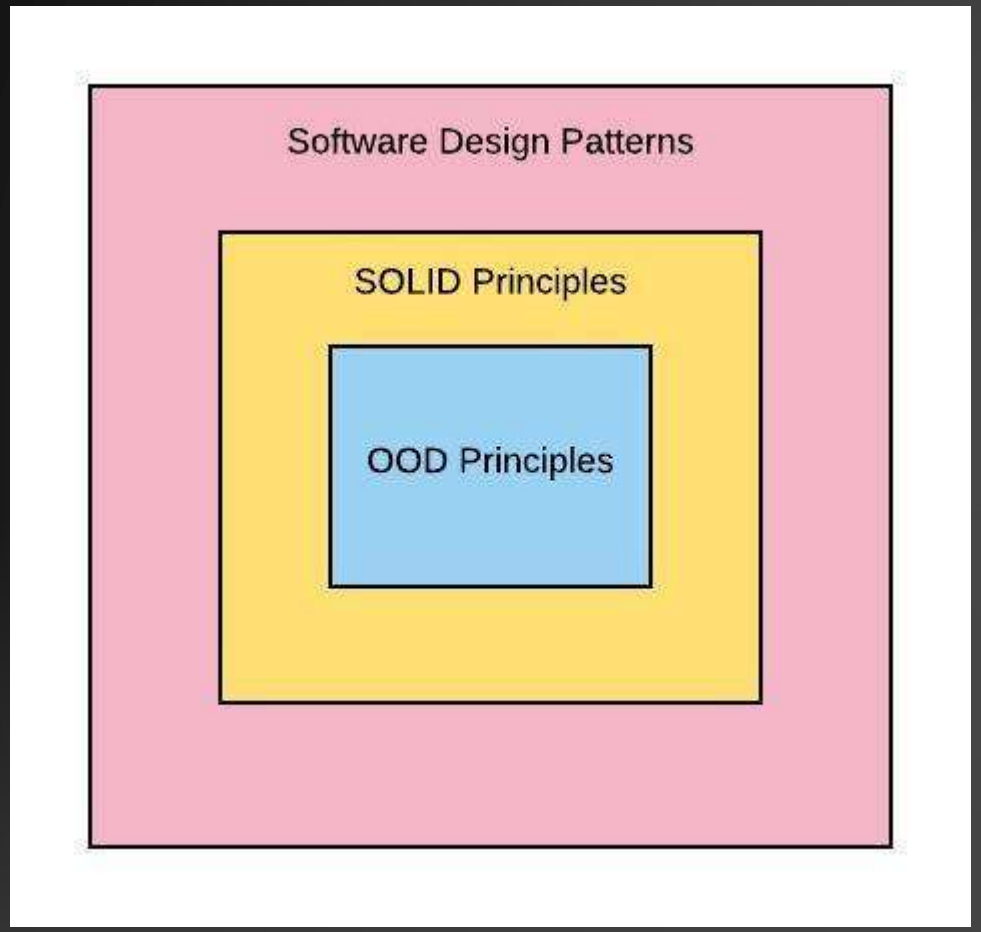
1. Entity framework is a ORM by Microsoft which allow you to query and modify RDBMS like SQL Server, Oracle, DB2 and MySQL etc. by using LINQ syntax.

2. LINQ to SQL will generate the DBML file

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

3. LINQ to SQL is old and slow.

3. EF is mostly used, new and fast.



BASICS

SINGLETON PATTERN

FACTORY PATTERN

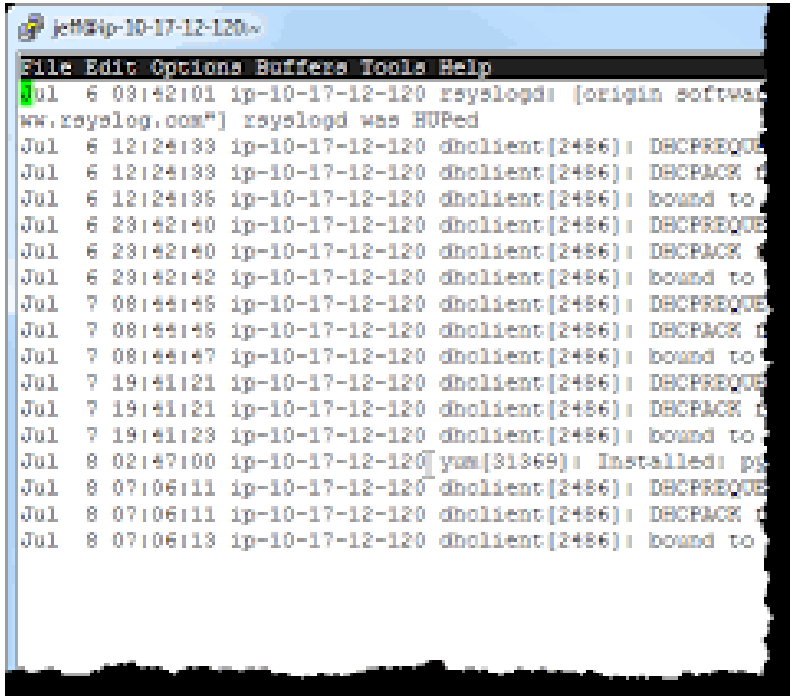
SOLID PRINCIPLES

What are Design Patterns and what problem they solve?

Design Patterns

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

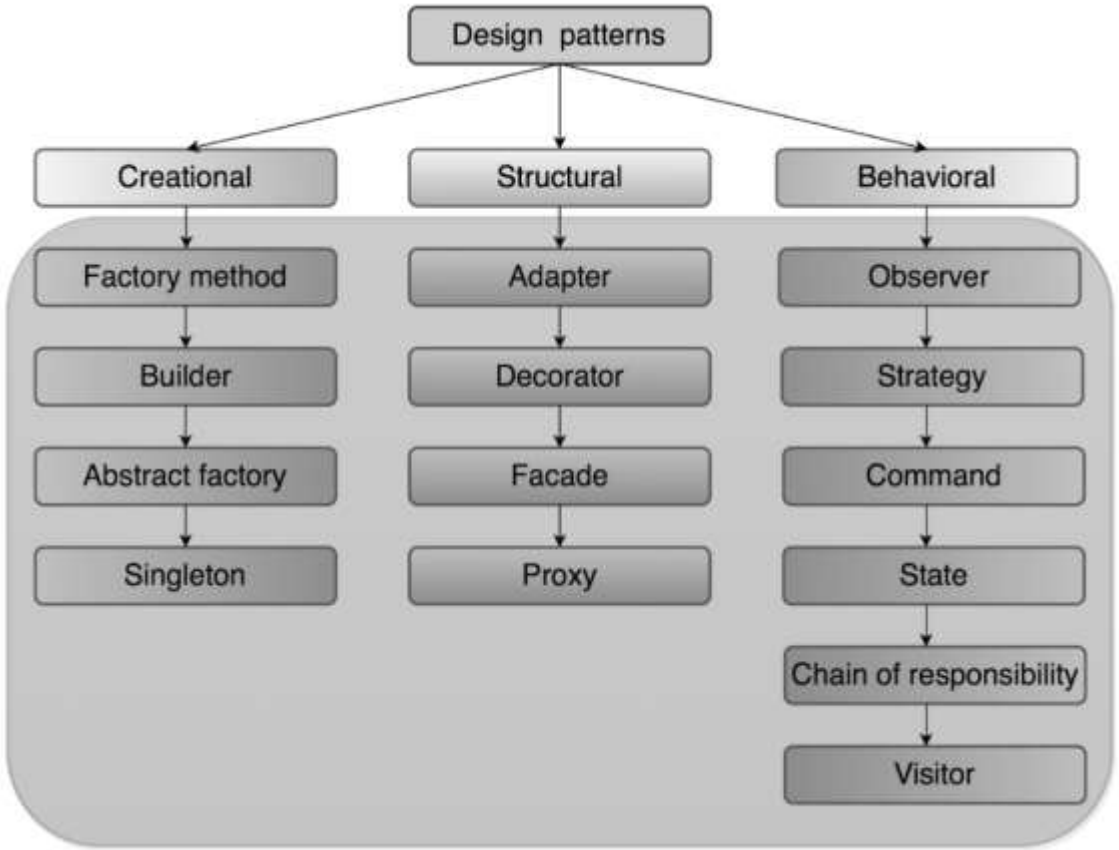
Problem - Suppose you want to implement logging for your application.



Q

What are the types of Design Patterns?

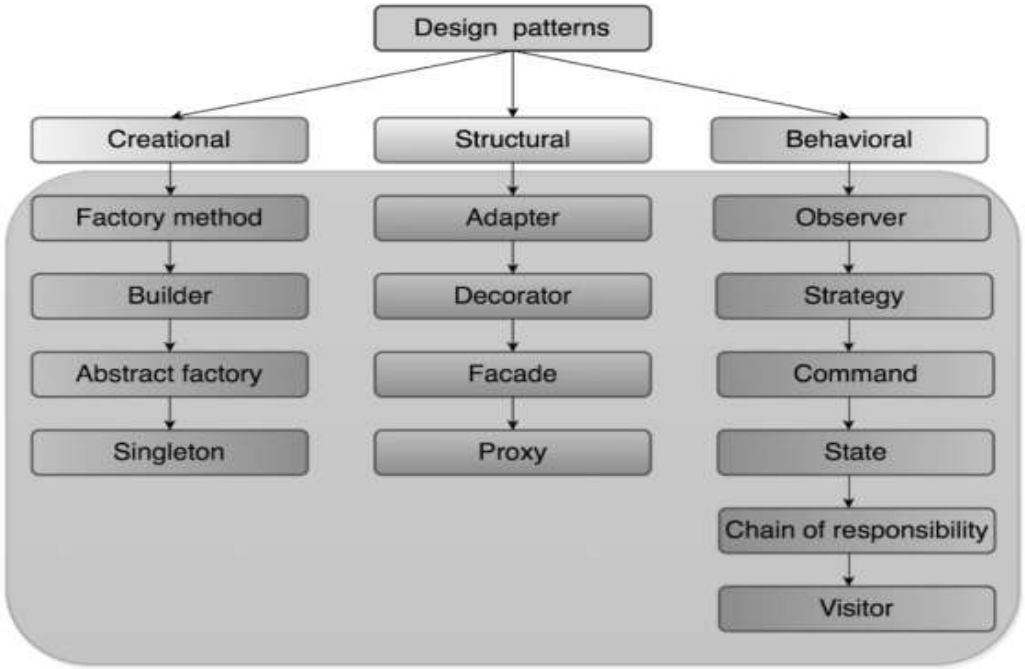
Design Patterns



What are Creational Design Patterns?

Creational Design Patterns

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



Why direct object creation is a problem.

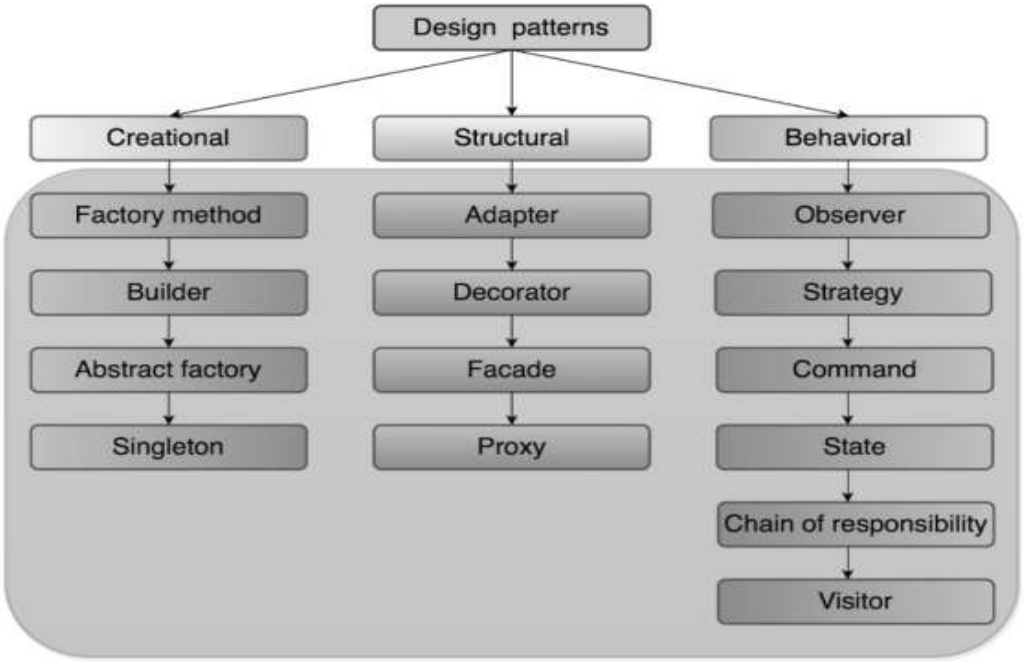
```
public Class Pet
{
        Cat b = new Cat();

        b.CountLegs();
}
```

The reason is in a large/enterprise application, if you want to change this Class from Cat to Dog, then you have to change it everywhere wherever you have used it and then it will be a testing issue. You have to retest all these classes like Pet.

What are Structural Design Patterns?

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.



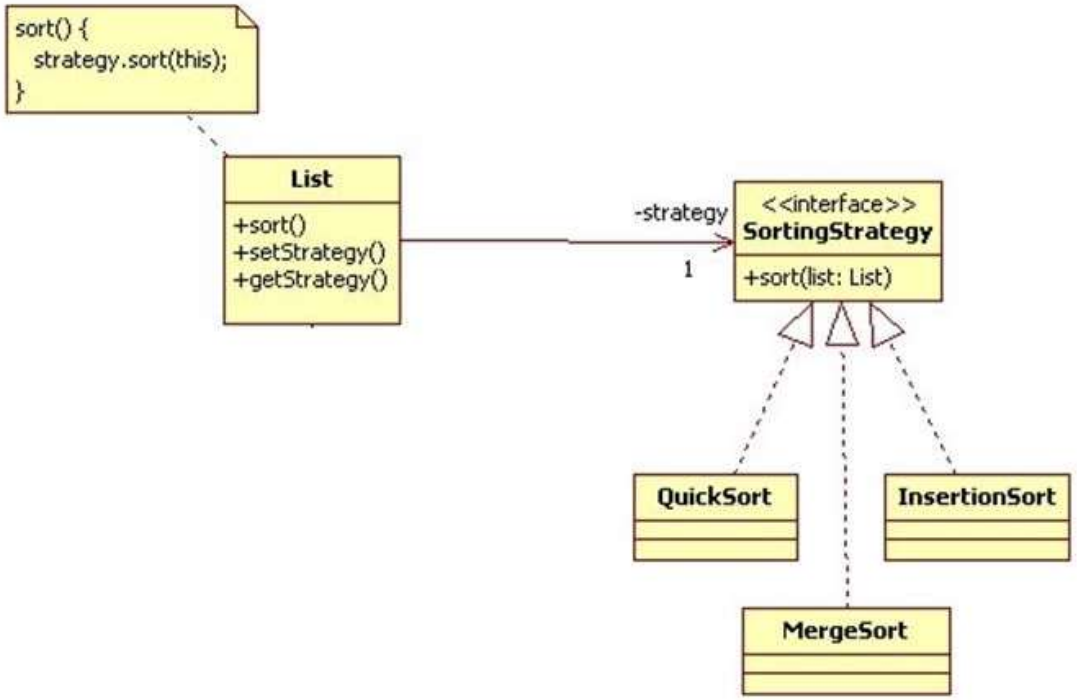
Adapter pattern

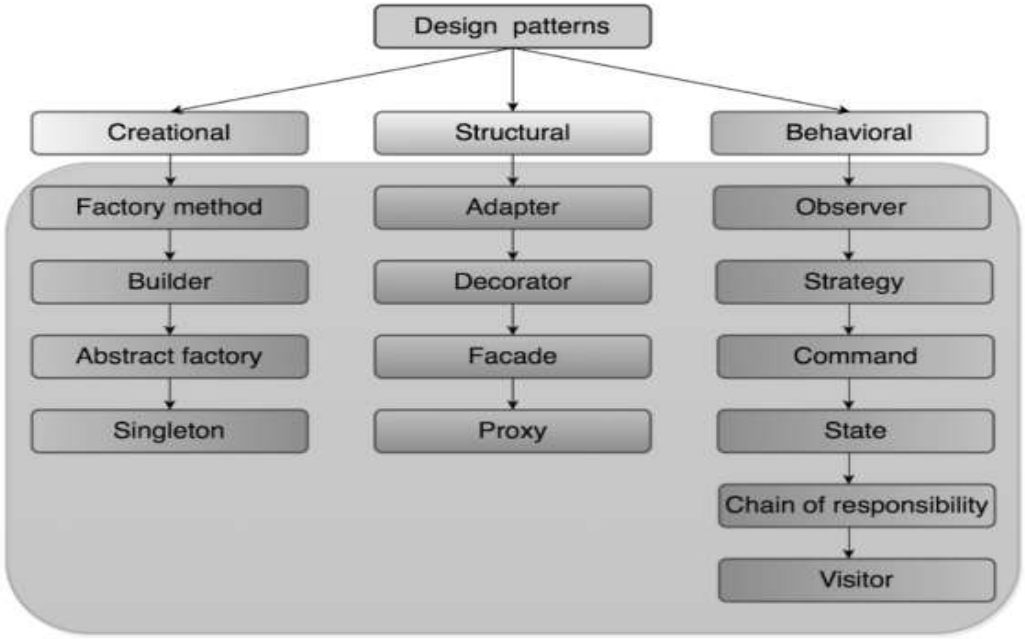
What are Behavioral Design Patterns?

Behavioral Design Patterns

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

For example, in strategy pattern we can choose the best strategy at run time for doing any operation





What is Singleton Design Pattern? V. IMP.

Design Patterns

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.

```
public static void Main(string[] args)
{
    ClassA objClassA = new ClassA();

    objClassA.MethodA();
}
```

1. Create a sealed class so that no one can inherit from this class.

2. Create a private constructor so that no one can create the object of this singleton class.

3. Declare a static instance of this class.

4. Create a method which will check whether the instance is null, if yes then it will create the new instance and if not null then it will return the same instance.

```
public sealed class Singleton
{
    private Singleton()
    {
    }
    private static Singleton instance;

    public static Singleton getInstance()
    {
        if (instance == null)
        {
            instance = new Singleton();
        }
        return instance;
    }
}
```

```
public static void Main(string[] args)
{
    Singleton s1 = Singleton.getInstance();
}
```

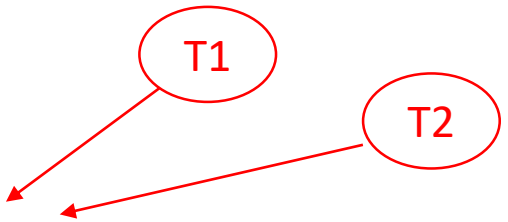
Q

How to make Singleton pattern Thread safe? V. IMP.

Design Patterns

Normal Singleton Pattern

```
public sealed class Singleton
{
    private Singleton() {}
    private static Singleton instance;

    public static Singleton getInstance()
    {
        if (instance == null)
        {
            instance = new Singleton();
        }
        return instance;
    }
}
```



Thread Safe Singleton Pattern

```
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;
         }
    }
}
```

What is Factory pattern? Why to use factory pattern?

Design Patterns

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.

```
// A 'ConcreteProduct' classes
public class DotNet
{
    public string GetSkill()
    {
        return "Core";
    }
}
```

```
// A 'ConcreteProduct' classes
public class Java
{
    public string GetSkill()
    {
        return "Spring";
    }
}
```

```
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();
        }
    }
}
```

1. Lots of **new** keywords

2. Client will know all the concrete classes name

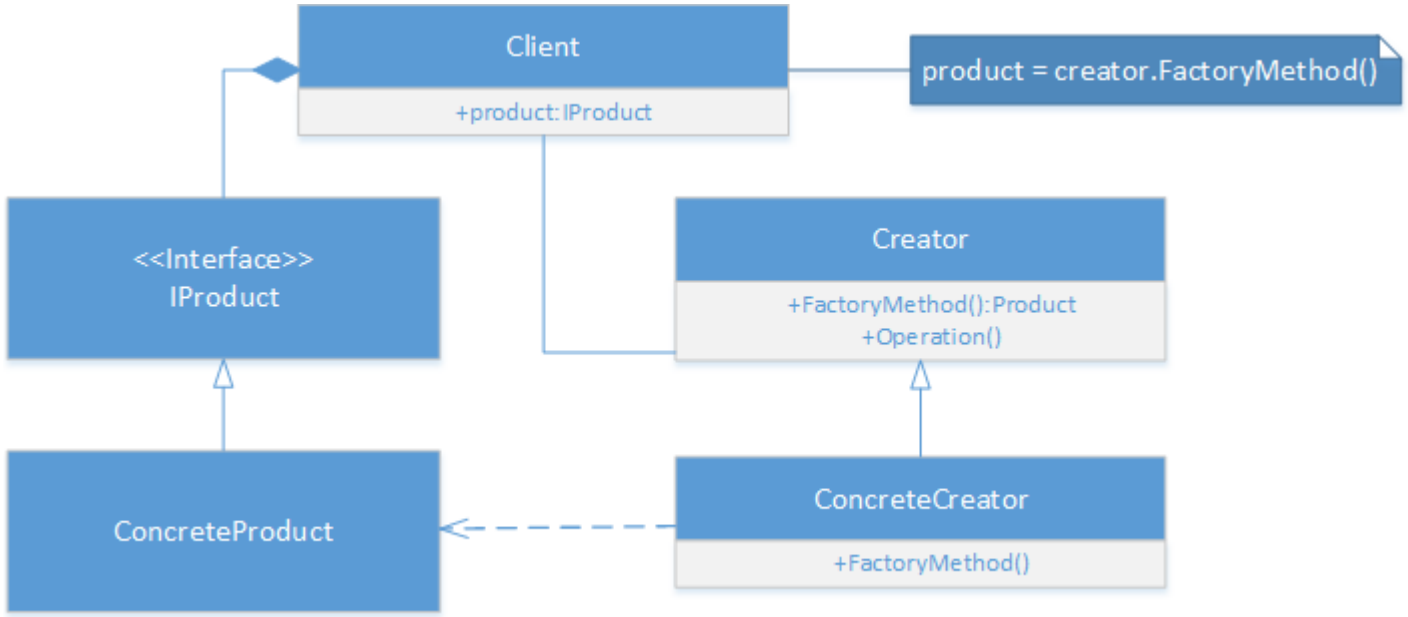
In factory pattern, we create all new objects in a common and separate class.

How to implement Factory method pattern?

Design Patterns

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.




```
//client code
class SoftwareCompanyClient
{
    static void Main(string[] args)
    {
        AgencyCreator factory = new ConcreteAgencyCreator();

        IFactory dotnet = factory.GetSkill("DotNetJobs");
        dotnet.Skill();

        IFactory java = factory.GetSkill("JavaJobs");
        java.Skill();
    }
}
```

ConcreteCreator

Creator

Interface /IProduct

```
/// A 'ConcreteProduct' class
public class DotNet : IFactory
{
    public void Skill()
    {
        Console.WriteLine("core");
    }
}
```

```
/// A 'ConcreteProduct' class
public class Java : IFactory
{
    public void Skill()
    {
        Console.WriteLine("spring");
    }
}
```

ConcreteProduct

How to implement Factory method pattern?

Design Patterns

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.

```
/// The 'Product' interface
 public interface IFactory //Agency or JobConsultancy
{
   void Skill();
}
```

```
//client code
class SoftwareCompanyClient
{
    static void Main(string[] args)
    {
        AgencyCreator factory = new ConcreteAgencyCreator();

        IFactory dotnet = factory.GetSkill("DotNetJobs");
        dotnet.Skill();

        IFactory java = factory.GetSkill("JavaJobs");
        java.Skill();
    }
 }
}
```

ConcreteCreator

Creator

Interface /IProduct

```
// The Creator Abstract Class
public abstract class AgencyCreator
{
   public abstract IFactory GetSkill(string technology);
}

// A 'ConcreteCreator' class
public class ConcreteAgencyCreator : AgencyCreator
{
    public override IFactory GetSkill(string technology)
    {
        switch (technology)
        {
            case "DotNetJobs":
               return new DotNet();
            case "JavaJobs":
               return new Java();
            default:
               return null;
        }
    }
}
```

```
/// A 'ConcreteProduct' class
public class DotNet : IFactory
{
    public void Skill()
    {
       Console.WriteLine("core");
    }
}
```

```
/// A 'ConcreteProduct' class
 public class Java : IFactory
{
    public void Skill()
    {
       Console.WriteLine("spring");
    }
}
```

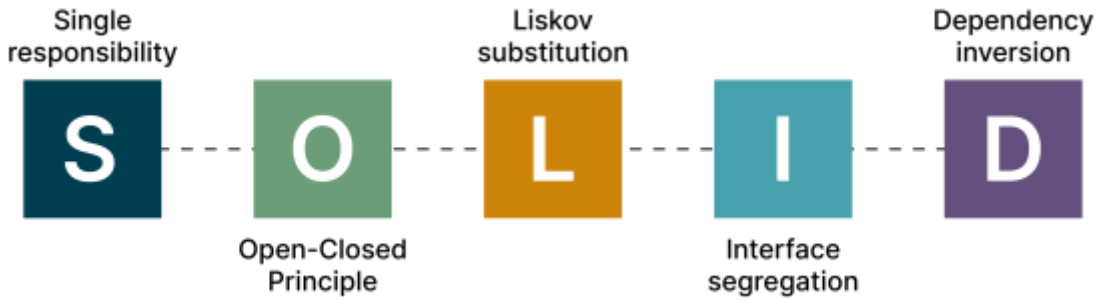
Q

What are SOLID Principles? What is the difference between SOLID Principles and Design Patterns?

S.O.L.I.D. PRINCIPLES

A principle is a general or basic truth, if that is violated then it is not good for the system/ software

SOLID principles aren't concrete - rather abstract.

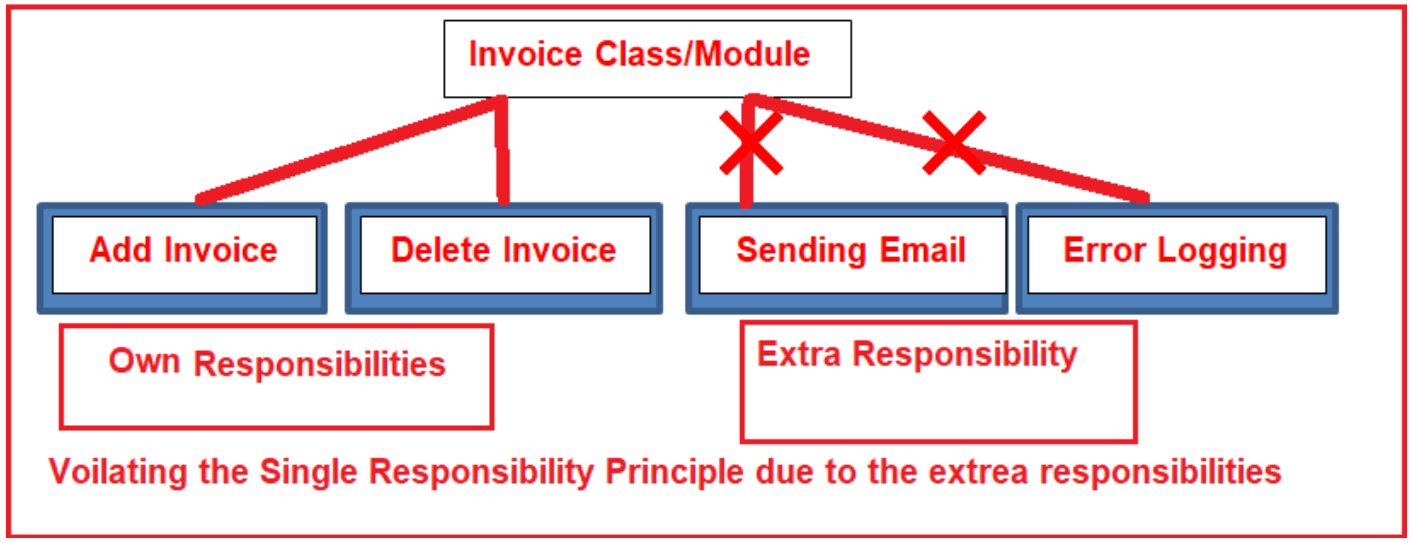


Design patterns are concrete and solve a particular kind of problem.

What is Single Responsibility Principle?

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.



What is Open-closed Principle?

S.O.L.I.D. PRINCIPLES

This 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**.

```
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.

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

```
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;
   }
}
```

Add a new class to extend the SavingAccount functionality which will be then extension not modification.

EASY TESTSTING is one of the top advantage of this principle.
In case of modifying the class, the whole class needs to be tested.

What is Liskov Substitution Principle?

S.O.L.I.D. PRINCIPLES

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

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

```
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
    }
}
```

```
public class LSP
{
    public static void Main(string[] args)
    {
        Employee emp = new Employee();

        Console.WriteLine(emp.CalculateSalary());

        Console.WriteLine(emp.CalculateBonus());
    }
}
//output: 50000 25000
```

Object of Superclass

```
public class LSP
{
    public static void Main(string[] args)
    {
        ContractualEmployee emp = new ContractualEmployee();

        Console.WriteLine(emp.CalculateSalary());

        Console.WriteLine(emp.CalculateBonus());
    }
}
//output:
//50000
//Unhandled Exception:
//System.NotImplementedException: The method or operation is not implemented.
```

Replaced by object of Subclass. Throwing error

Q

What is Liskov Substitution Principle?

S.O.L.I.D.
PRINCIPLES

When you are creating a project and using inheritance, creating base classes and derived then you must ensure that all the methods of base classes are used by derived classes otherwise it will violate LSP.

What is Interface Segregation Principle?

S.O.L.I.D. PRINCIPLES

Do not create big interfaces with many methods declaration and then force them on deriving classes.

```
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();
    }
}
```

What is Interface Segregation Principle?

S.O.L.I.D. PRINCIPLES

Do not create big interfaces with many methods declaration and then force them on deriving classes.

```
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()
    {
        Console.WriteLine("Fly a flying car");
    }
}
```

What is Dependency Inversion Principle?

Design Patterns

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.

```
namespace SOLID_PRINCIPLES.DIP
{
    public class EmployeeBusinessLogic
    {
        EmployeeDataAccess _EmployeeDataAccess;

        public EmployeeBusinessLogic()
        {
            //Getting employee data from database
            _EmployeeDataAccess = DataAccessFactory.GetEmployeeDataAccessObj();
        }

        public Employee GetEmployeeDetails(int id)
        {
            return _EmployeeDataAccess.GetEmployeeDetails(id);
        }
    }
}
```

Higher Level Class

Lower Level Class

Creating object of lower level class inside higher level class, means higher level class is depends on lower level class

Q

What is DRY principle?

Design Patterns

DRY stands for DON’T REPEAT YOURSELF.

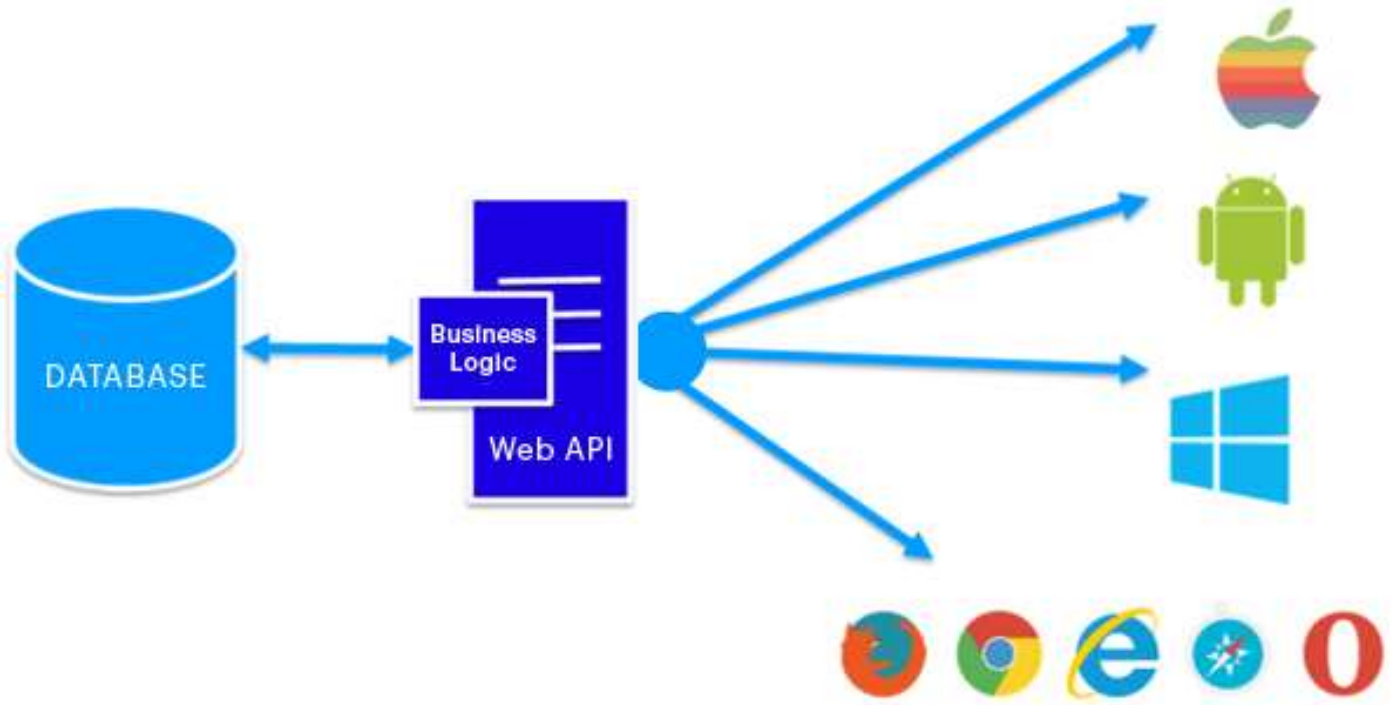
Don’t write the same functionality multiple time.

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

- WEB API ADVANTAGES
- REST & RESTFUL
- CONTENT NEGOTIATION
- AUTHENTICATION TECHNIQUES
- MORE…

What is Web API? What is the purpose of Web API?

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



What are Web API advantages over WCF and web serivces? V. IMP.

Every business like amazon, facebook, twitter needs both mobile app and website and Web API is a better option for that.

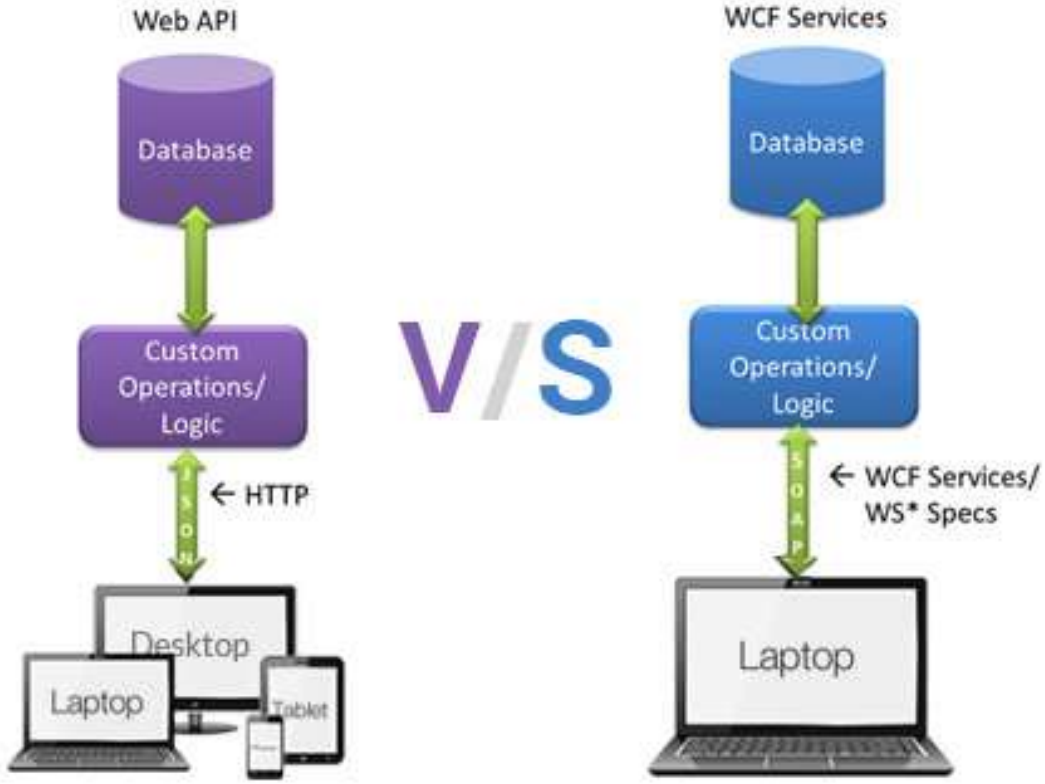
1. It works the way **HTTP** works using standard HTTP verbs like GET, POST, PUT, DELETE for all the crud operation.

2. It is an **OPEN** source.

3. It is a **LIGHTWEIGHT** architecture and is good for devices, which have limited bandwidth(JSON & XML are lightweight) like smart cell phones.

4. Web API Controller pattern is much similar to **MVC** Controller. Thus, it becomes easy to maintain and understand.

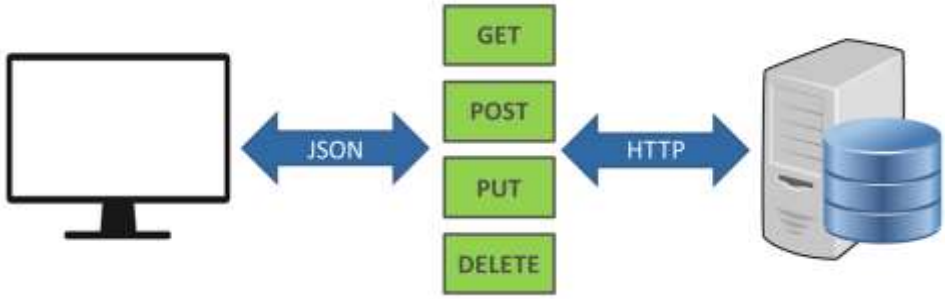
5. Web API has very easy configuration settings as comparison to WCF.



What is Rest and Restful?

REST stands for Representational State Transfer.

REST 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
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.

Q

Is it possible to use WCF as Restful services?

Yes, we can develop RESTful services with WCF.

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)
POST	Create a new resource	http://example.com/api/orders (create a new order, from data provided with the request)
PUT	Update a resource	http://example.com/api/orders/123 (update order #123, from data provided with the request)
DELETE	Delete a resource	http://example.com/api/orders/123 (delete order #123)

How to consume Web API from a .NET MVC application?

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

```
public class HomeController : Controller
{
    string Baseurl = "http://facebook.com/api";

    public async Task<ActionResult> Index()
    {
        List<Employee> EmpInfo = new List<Employee>();

        using (var client = new HttpClient())
        {
            client.BaseAddress = new Uri(Baseurl);
            client.DefaultRequestHeaders.Clear();

            client.DefaultRequestHeaders.Accept.Add(new
                MediaTypeWithQualityHeaderValue("application/json"));

     HttpResponseMessage Res = await client.GetAsync("Employee/GetAllEmployees");

            if (Res.IsSuccessStatusCode)
            {
               var EmpResponse = Res.Content.ReadAsStringAsync().Result;

               EmpInfo = JsonConvert.DeserializeObject<List<Employee>>(EmpResponse
                   );
            }
            return View(EmpInfo);
        }
    }
}
```

1. Create the object of HttpClient class.

2. Set the BaseAddress property of client object.

3. Add media type, the request format of data. Here it is json.

4. GetAsync is the method, which will send request to find web api resource GetAllEmployees. And then store the response in HttpResponseMessage class object.

5. Checking the response is successful or not.

6. If successful then result is saved in a variable

7. Then the variable will be deserialized from json format to Employee object.

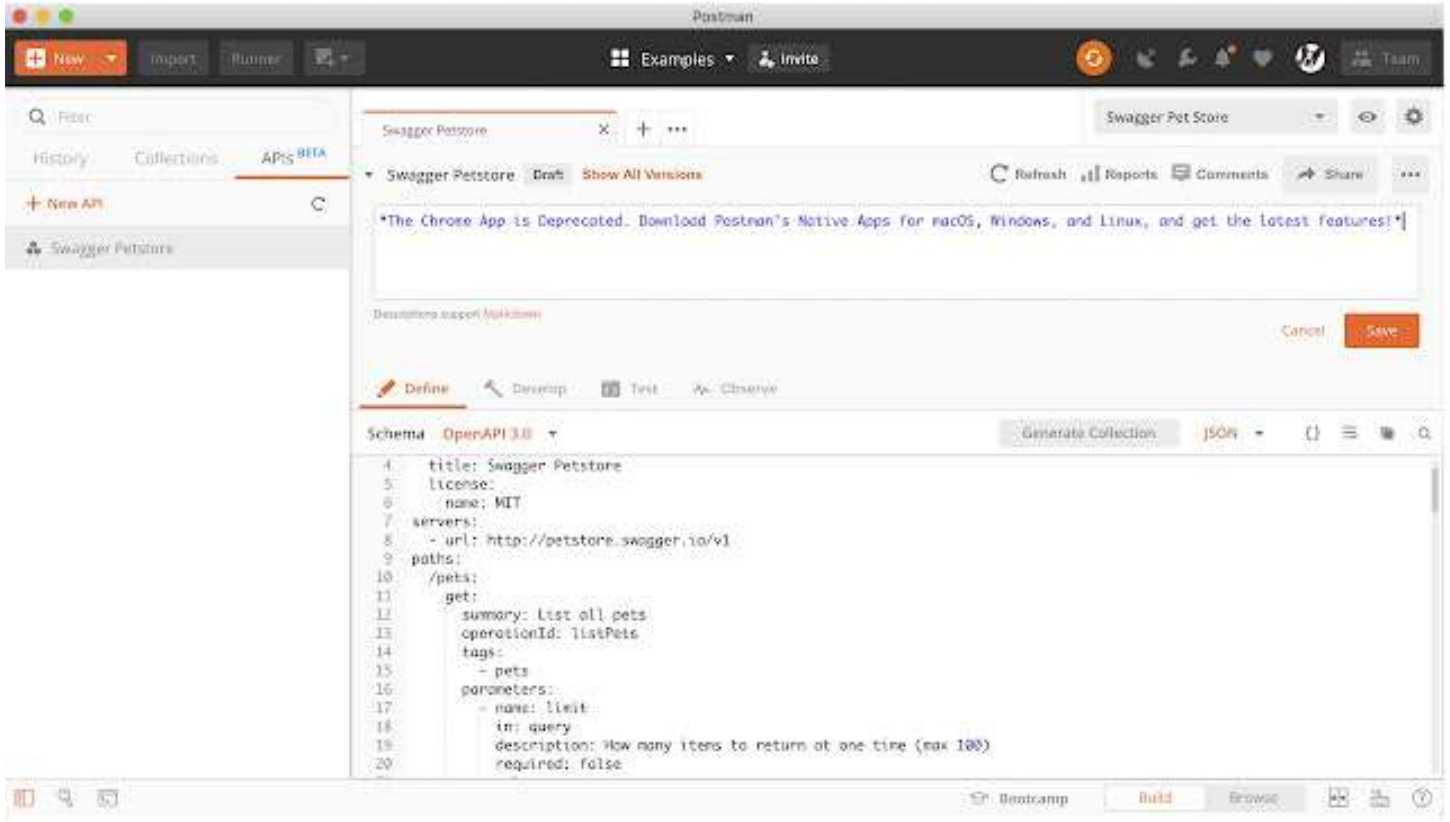
Q

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.

How to test Web API? What are the tools?

FIDDLER, **POSTMAN** & SWAGGER tools



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.

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** introduce which is basically the replacement of HttpResponseMessage. It creates clean code and also simplifies unit testing.

Example of using HttpResponseMessage :

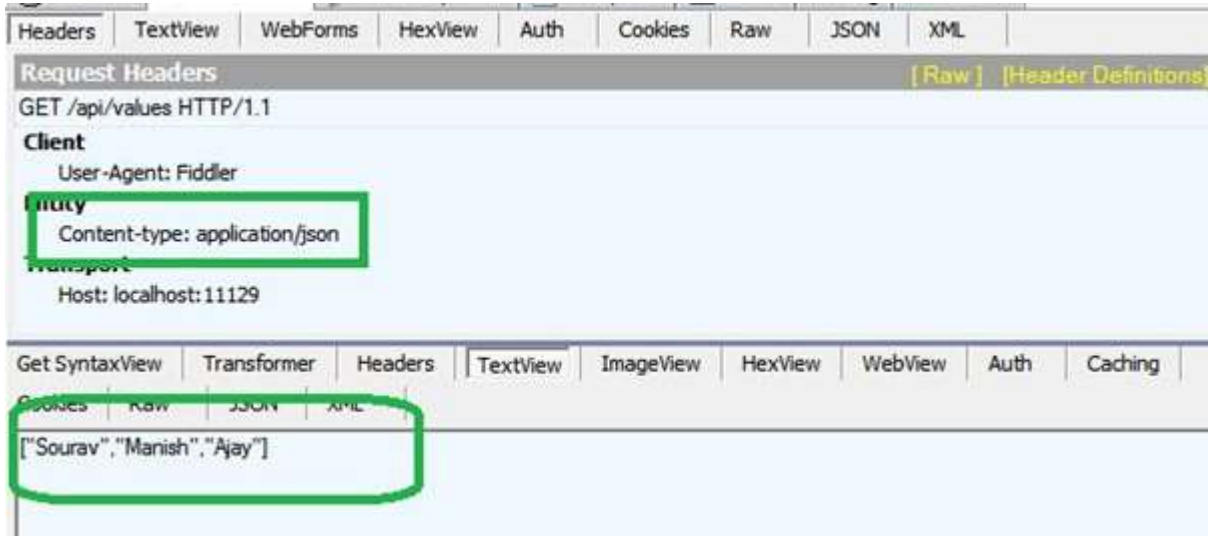
```
public HttpResponseMessage Get(int id)
{
        var product = dbContext.Products.Get(id);

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

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

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

 }
```

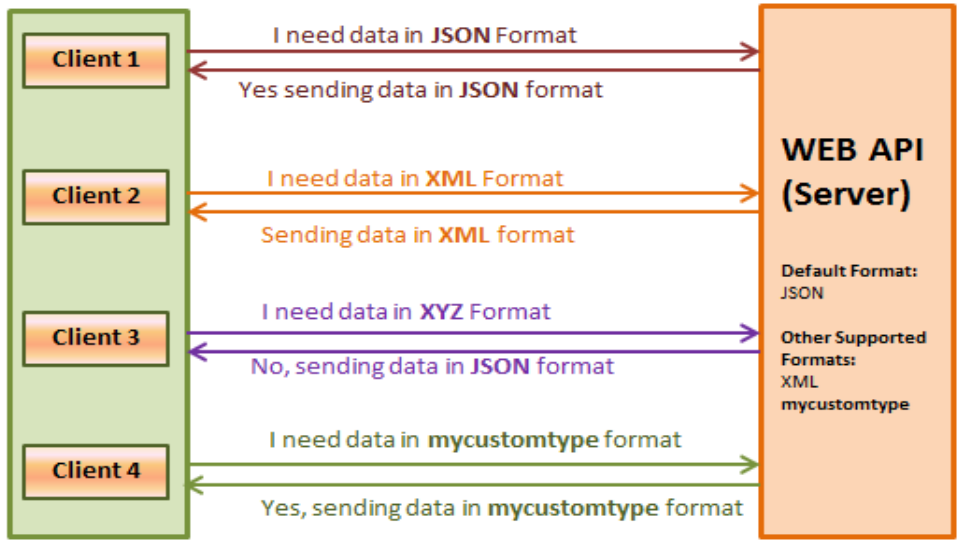
Example of using IHttpActionResult :

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

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

         return Ok(product);
}
```

What is Content Negotiation in Web API? V. IMP.

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

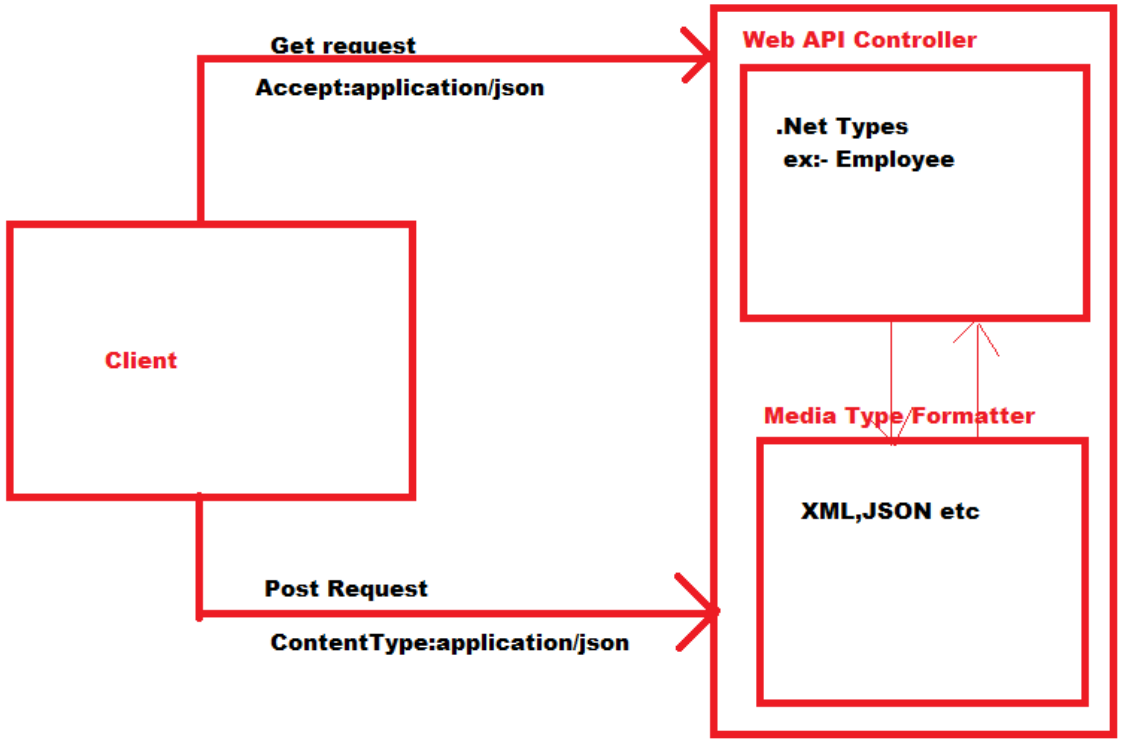




What is MediaTypeFormatter class in Web API?

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

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

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



What are Response Codes in Web API?

1xx: Informational – Communicates transfer protocol-level information.

2xx: Success – Indicates that the client's request was accepted successfully.

3xx: Redirection – This means request is not complete. The client must take some additional action in order to complete their request.

4xx: Client Error – This means there is some error in API code.

5xx: Server Error – This means the error is not due to web api code but due to some environment settings.

What is Basic Authentication in Web API?

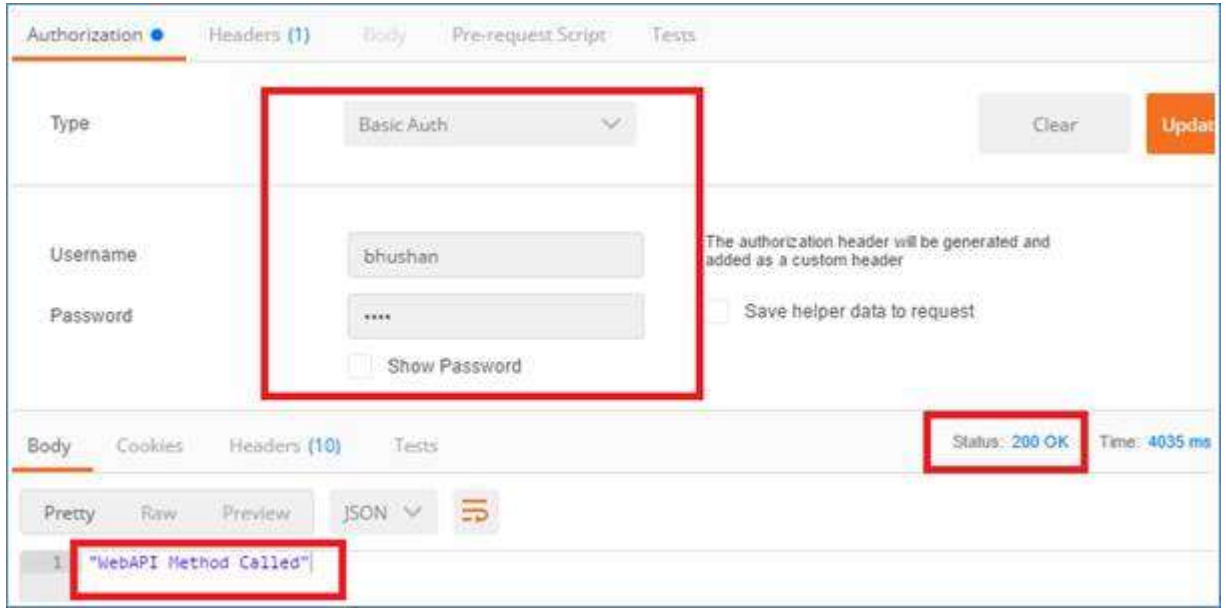
Authentication means verifying a user who is accessing the system or Web API.

Basic Authentication

In Basic Authentication, **the user passes their credentials [user name and password] on a post request**.

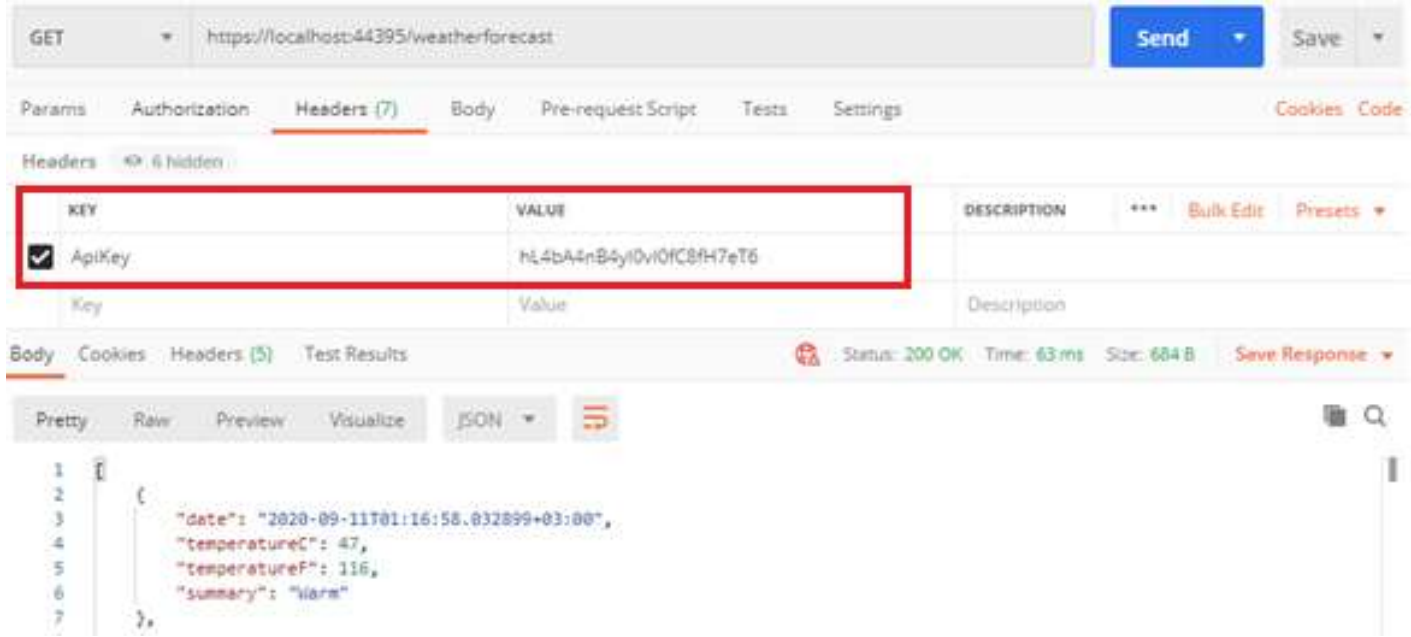
At the WebAPI end, credentials are verified.

If the credentials are valid, then a session will initiate to accept the subsequent requests without validating the user again.



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.



What is Token based authentication? V. IMP.

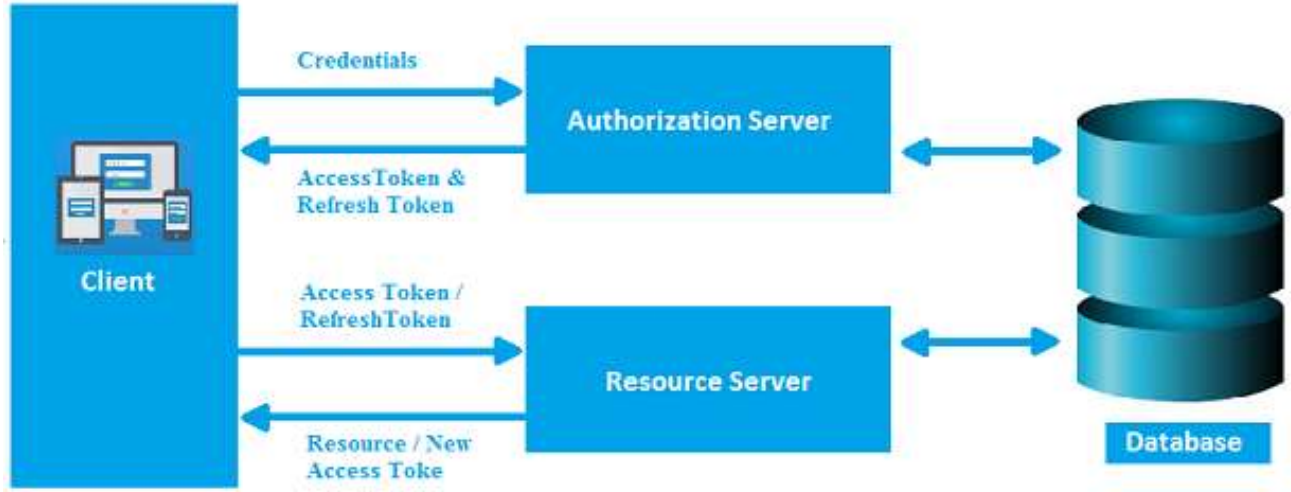
Token-based authentication is a 4 step process:

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.

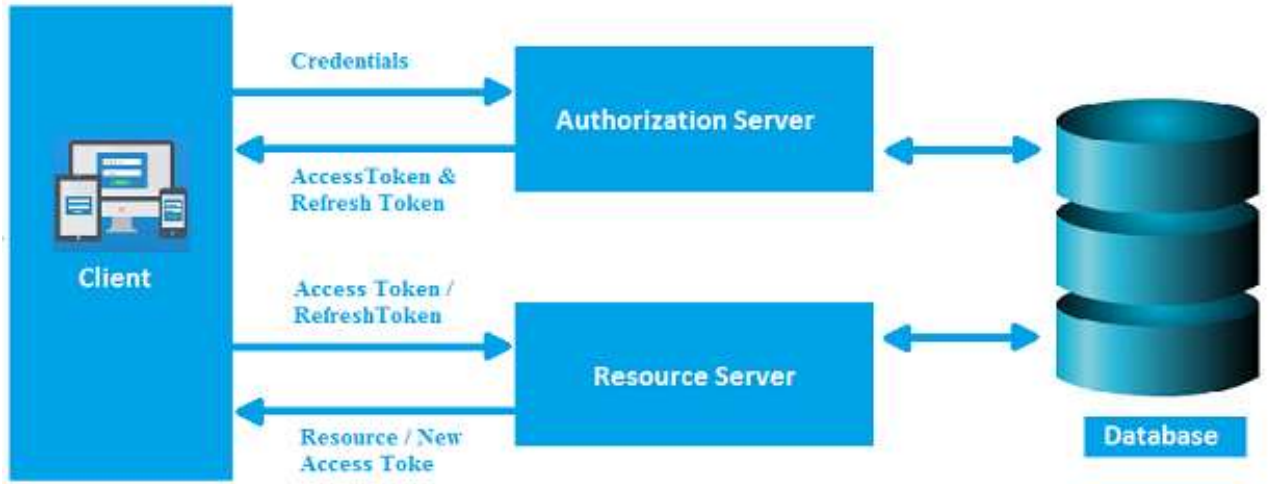
4. If the Access token is expired, then the client application can request for a new access token by using Refresh token.



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

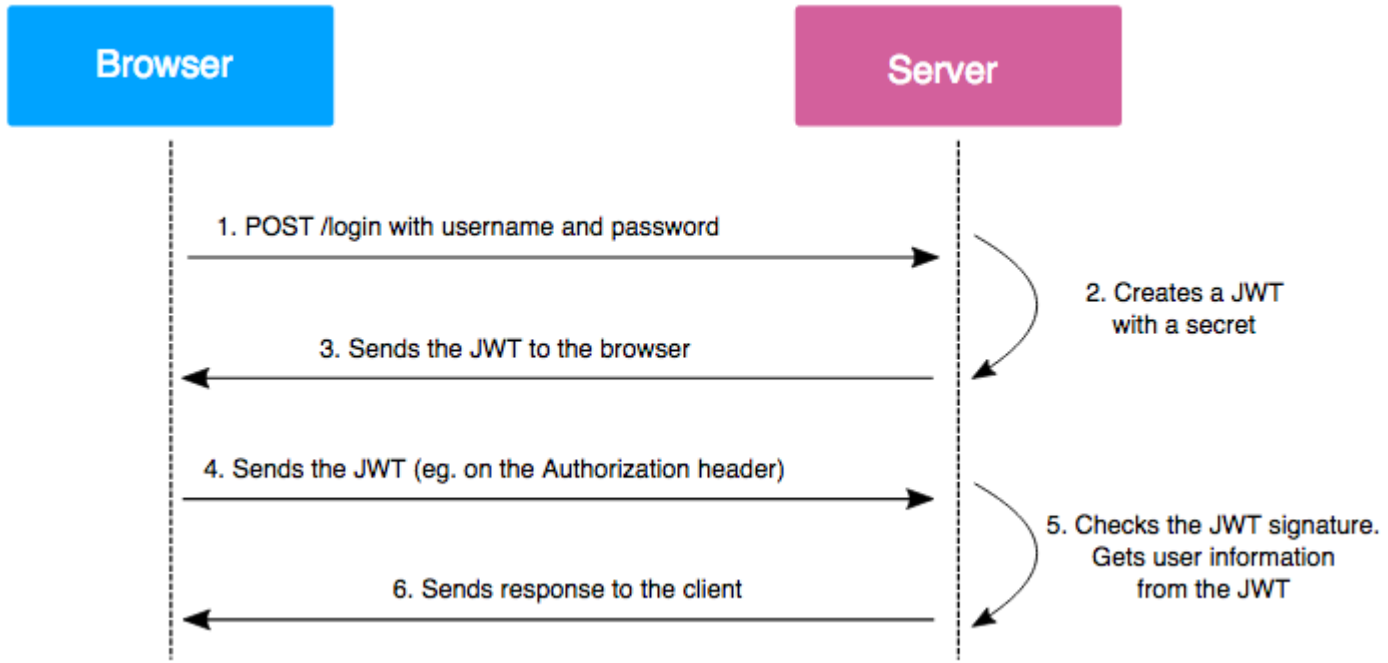
What is OAuth?

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



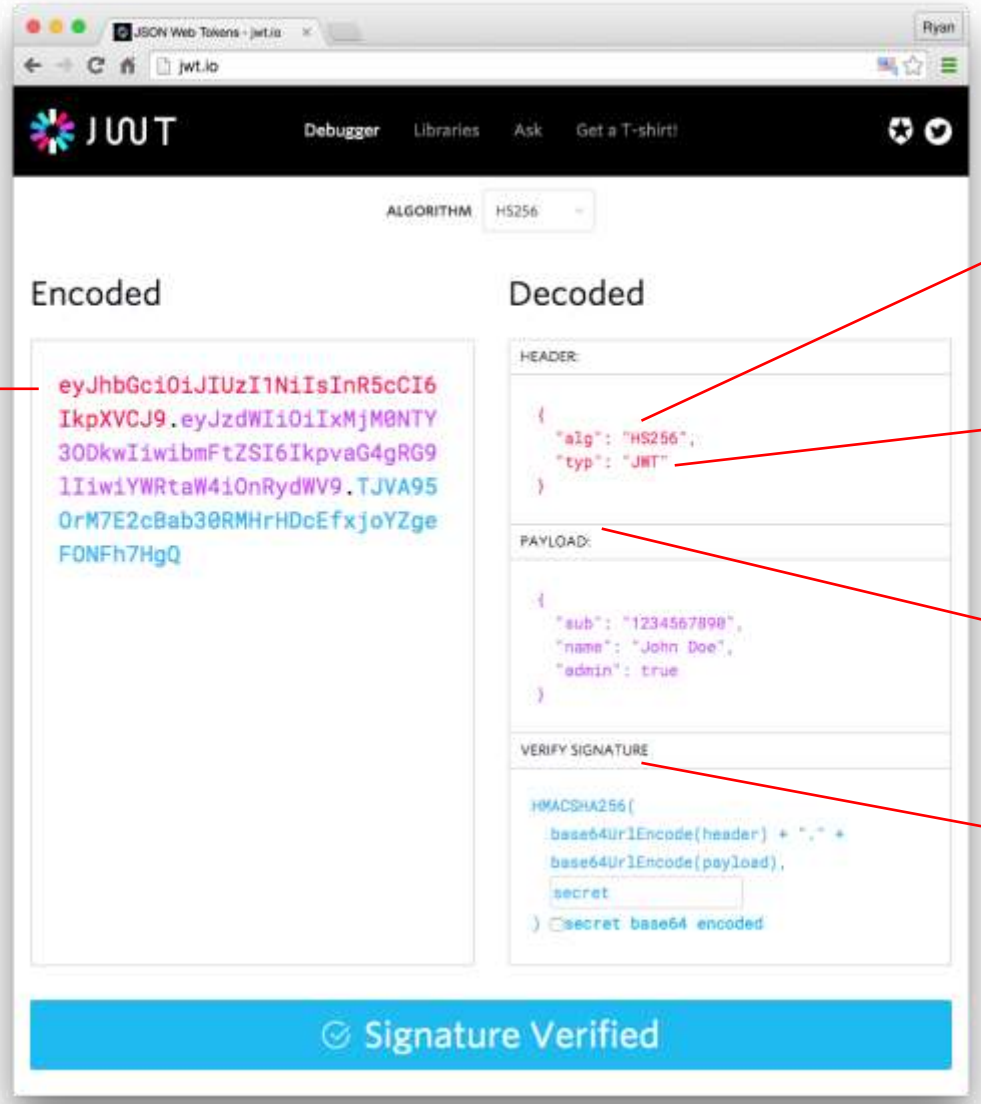
What is JWT Authentication? V. IMP.

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



What are the parts of JWT token?

JWT token JSON Web Tokens



JWT Token

Algorithm used to generate the token.

The type of the token, which is JWT here.

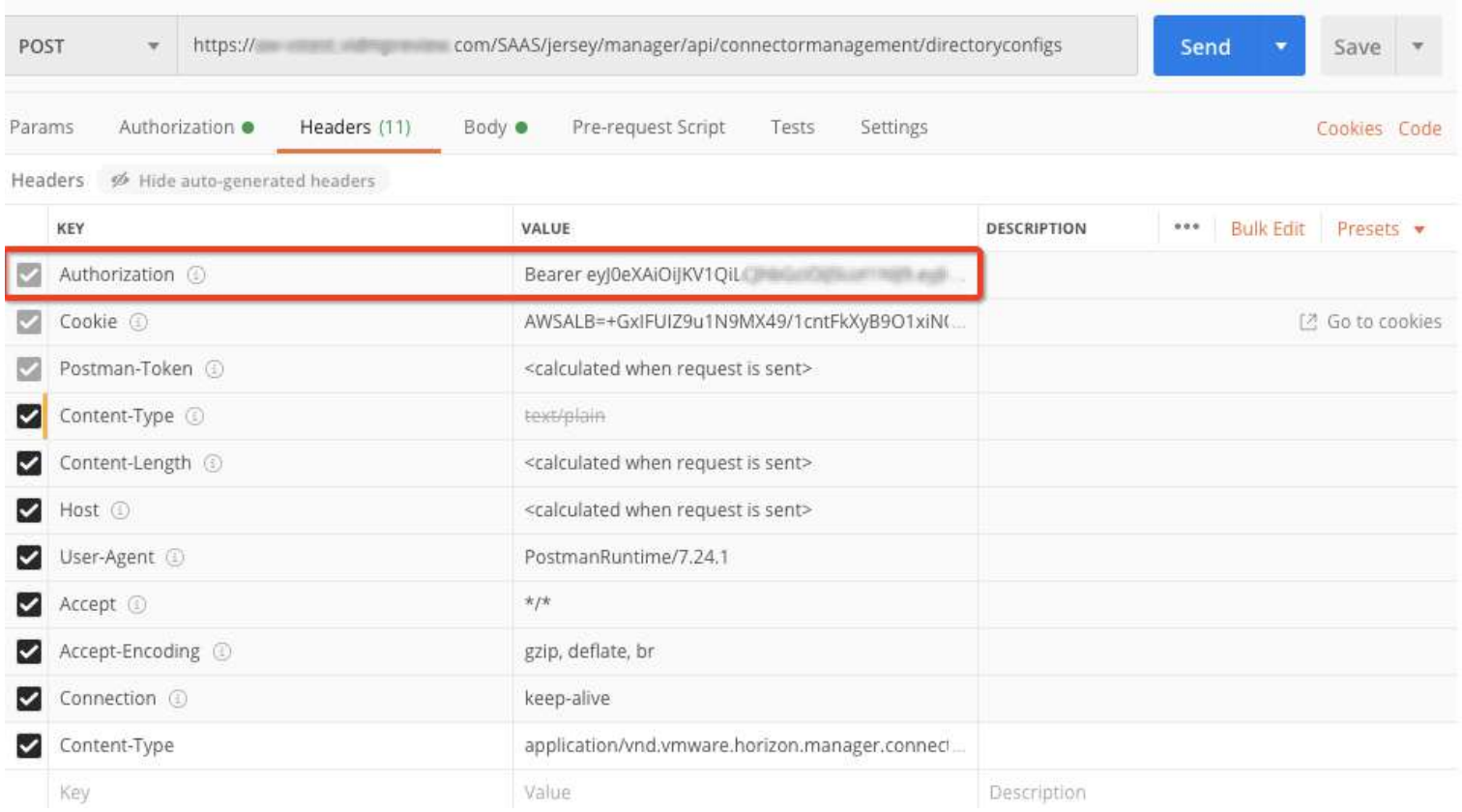
The payload contains the CLAIMS.

The signature is used to verify that the issuer of the JWT correct.

Where JWT token reside in the request?

In REQUEST HEADER.

KEY - Authorization
VALUE - Token



- BASICS
- PROGRAM.CS & STARTUP.CS FILE
- DEPENDENCY INJECTION
- SERVICE LIFETIMES
- MIDDLEWARE

Q1

Why to use Dependency Injection? OR What problems does Dependency Injection solve?

.NET Core

```
0 references
public class FirstController : Controller
{
    0 references
    public int Index()
    {
        //Create object of math class
        MathStudent cls = new MathStudent();
        return cls.GetStudentCount();
    }
}
```


```
4 references
public class MathStudent
{
    2 references
    public int GetStudentCount()
    {
        return 50;
    }
}
```

```
0 references
public class SecondController : Controller
{
    0 references
    public int Index()
    {
        //Create object of math class
        MathStudent cls = new MathStudent();
        return cls.GetStudentCount();
    }
}
```

```
1 reference
public class ScienceStudent
{
    0 references
    public int GetStudentCount()
    {
        return 100;
    }
}
```

100 Controllers
(Big/Enterprise Level Application)

Q2

How can we inject the dependency into the controller?

.NET Core

```
1 reference
public class FirstController : Controller
{
    private IStudent _student;
    0 references
    public FirstController(IStudent student)
    {
        _student = student;
    }
    0 references
    public int Index()
    {
        return _student.GetStudentCount();
    }
}
```


```
4 references
public class MathStudent:IStudent
{
    3 references
    public int GetStudentCount()
    {
        return 50;
    }
}
```

```
1 reference
public interface IStudent
{
    0 references
    public int GetStudentCount();
}
```

```
1 reference
public class SecondController : Controller
{
    private IStudent _student;
    0 references
    public SecondController(IStudent student)
    {
        _student = student;
    }
    0 references
    public int Index()
    {
        return _student.GetStudentCount();
    }
}
```

```
1 reference
public class ScienceStudent: IStudent
{
    1 reference
    public int GetStudentCount()
    {
        return 100;
    }
}
```

```
// This method gets called by the runtime.
// Use this method to add services to the container.
0 references
public void ConfigureServices(IServiceCollection services)
{
    services.AddMvc();
    services.AddSingleton<IStudent, ScienceStudent>();
}
```

100 Controllers
(Big/Enterprise Level Application)

Q3

What is Dependency Injection in ASP.NET Core?
What are the types of Dependency Injection?

.NET Core

Dependency Injection (DI) is a software design pattern that allows us to develop loosely coupled code.

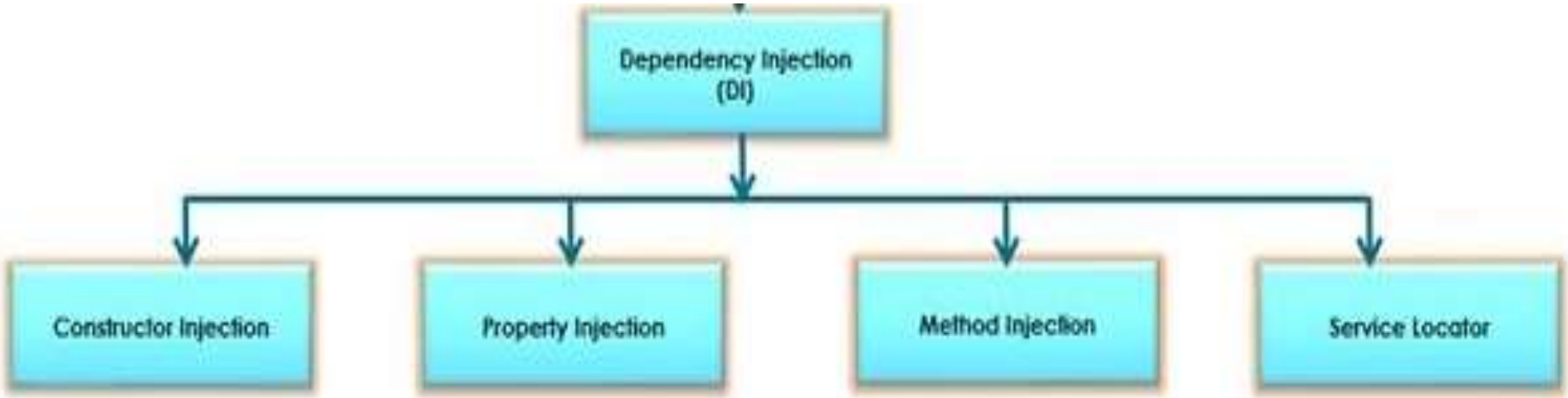
```
public class A
{
    public static void Main(string[] args)
    {
        B b = new B();
        int a = b.MethodB();
        Console.WriteLine(a);
    }
}
```

```
public class B
{
    public int MethodB()
    {
        return 100;
    }

}
```

Q3

What is Dependency Injection in ASP.NET Core?
What are the types of Dependency Injection?

.NET Core



Q4

How to use Dependency Injection in Views in ASP.NET Core?

.NET Core

```
4 references
public interface IStudent
{
    2 references
    public int GetStudentCount();
}
```

```
1 reference
public class MathStudent:IStudent
{
    2 references
    public int GetStudentCount()
    {
        return 100;
    }
}
```

```
0 references
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();

    services.AddScoped<IStudent, MathStudent>();
}
```

```
@{
    ViewData["Title"] = "Home Page";
}
@inject WebApplication1.IStudent _student

<div class="text-center">
    <h1 class="display-4">Welcome</h1>
    Total Student: @_student.GetStudentCount();
</div>
```

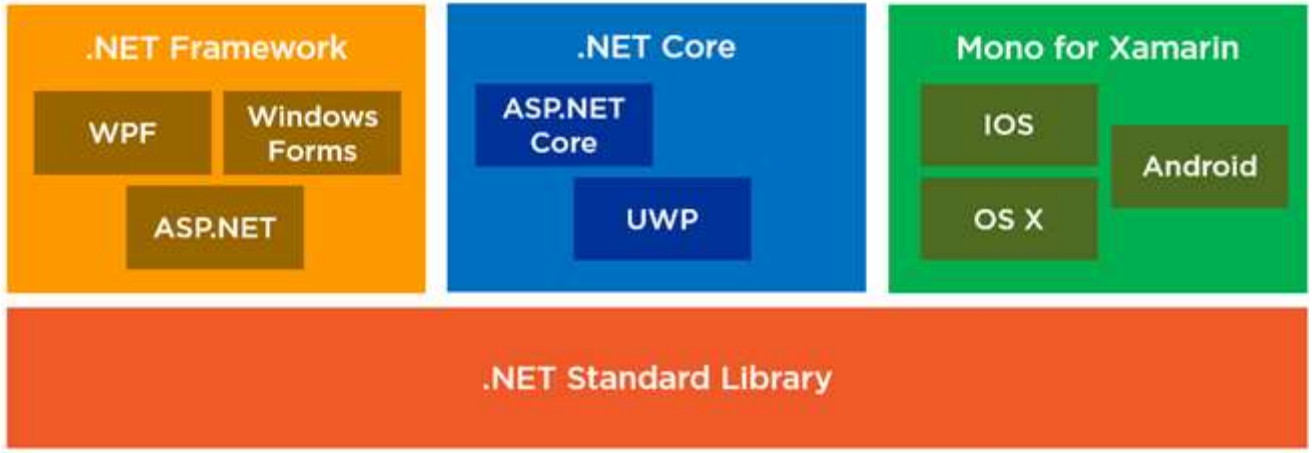
A service can be injected into a view using the @inject directive.

Q5

What is .NET Core?

.NET Core

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

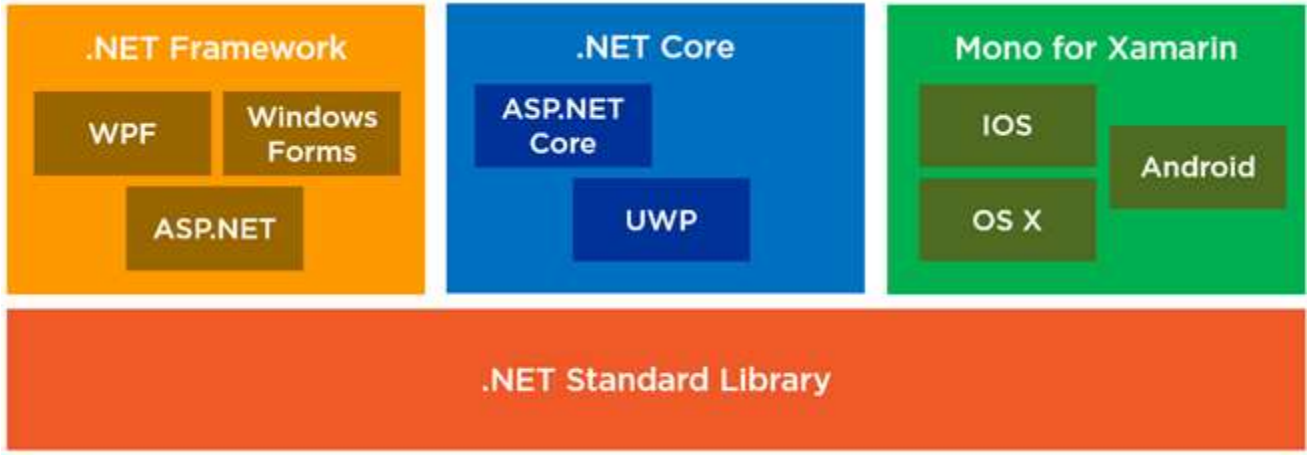


Q6

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.



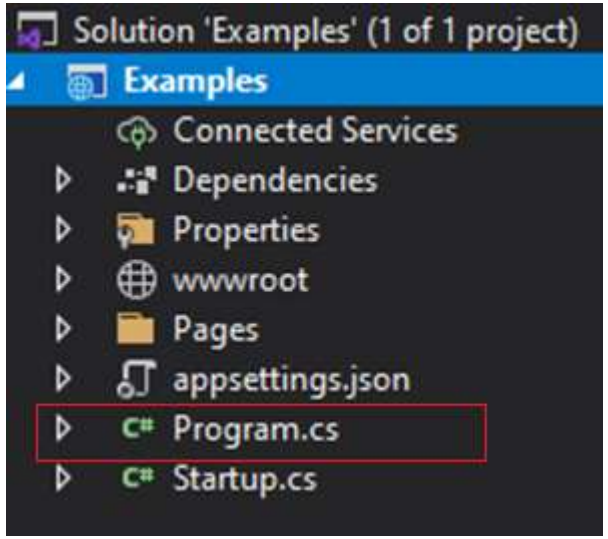
Q7

What are the advantages of .NET Core over .NET framework?

CROSS PLATFORM	OPEN SOURCE	HOSTING	BUILT-IN DEPENDENCY INJECTION	SUPPORT MULTIPLE IDE
• Windows • Linux • MacOS	• Free to use • Modify • Distribute	• Kestrel • IIS • Nginx	• Loosely Coupled Design • Reusability • Testability	• Visual Studio • Visual Studio for Mac • Visual Studio Code
.NET Framework only supports Windows	.NET Framework is paid	.NET Framework only support IIS Hosting	.NET framework don't have built in dependency injection	.NET framework only support Visual Studio IDE

Q8

What is the role of Program.cs file in ASP.NET Core?
OR What is CreateHostBuilder / CreateDefaultBuilder?




```
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>();
            });
}
```

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

2. CreateHosBuilder and CreateDefaultBuilder method prepare the default settings for web server for this application.

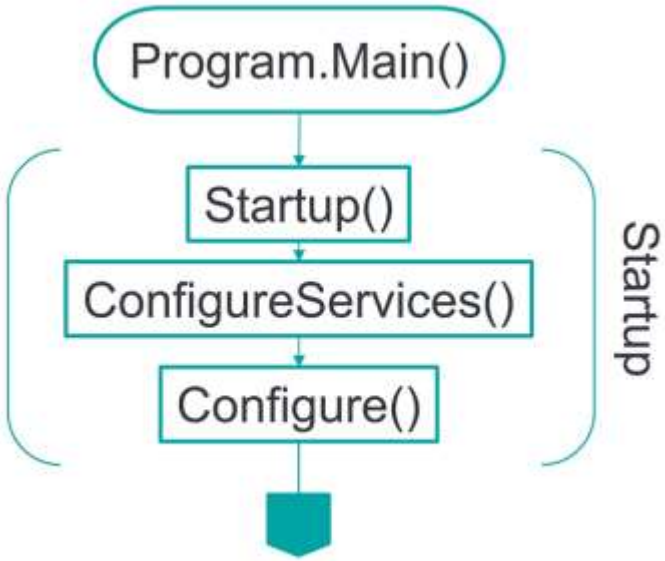
3. Then the Run method will call the Startup class.

Q9

What is the role of Startup.cs file?

.NET Core

```
2 references
public class Startup
{
    0 references
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    1 reference
    public IConfiguration Configuration { get; }

    // This method gets called by the runtime.
    // Use this method to add services to the container.
    0 references
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddControllersWithViews();
    }

    // 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");
            // The default HSTS value is 30 days.
            // You may want to change this for production scenarios, see https://ak
            app.UseHsts();
        }
        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthorization();

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



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

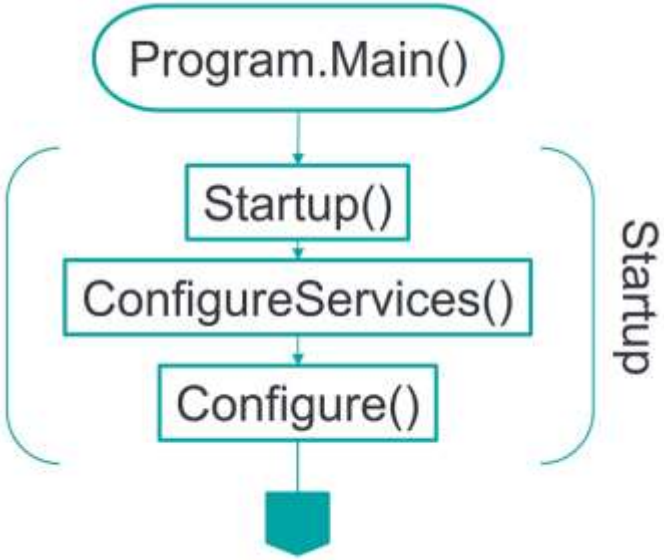
2. It defines the app's REQUEST HANDLING PIPELINE as a series of middleware components by using CONFIGURE method.

Q10

What is the role of ConfigureServices method?

.NET Core

```
// This method gets called by the runtime.
// Use this method to add services to the container.
0 references
public void ConfigureServices(IServiceCollection services)
{
    services.AddControllersWithViews();
}
```



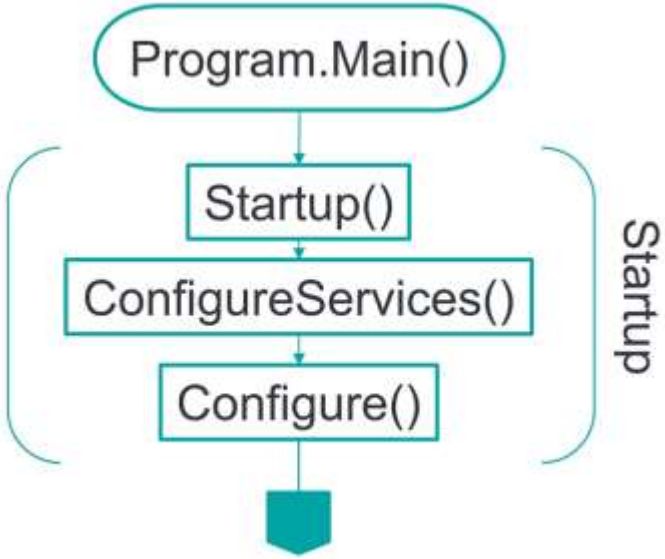
ConfigureServices method is OPTIONAL.
It gets called by the host before the 'Configure' method to configure the APP'S SERVICES.

Q11

What is the role of Configure method?

.NET Core

```
// 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?}");
    });
}
```



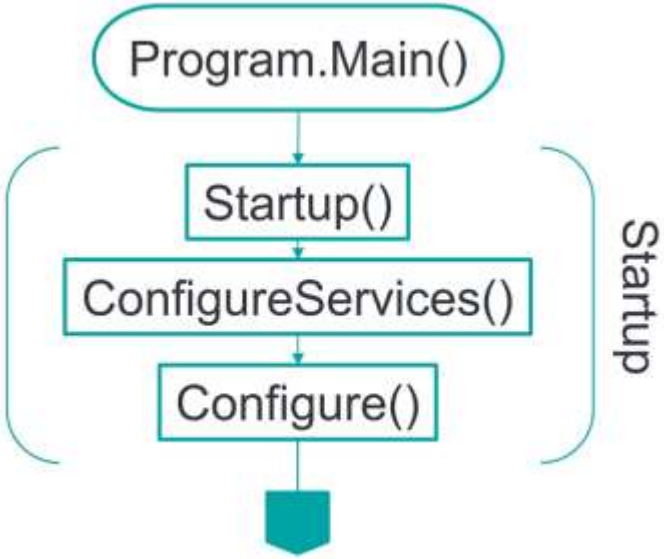
Configure method specifies how the app responds to HTTP request and response.
It is used to setup request pipeline.

Q12

What is the difference between ConfigureServices & Configure method?

.NET Core

```
2 references
public class Startup
{
    0 references
    public Startup(IConfiguration configuration)
    {
        Configuration = configuration;
    }

    1 reference
    public IConfiguration Configuration { get; }

    // This method gets called by the runtime.
    // Use this method to add services to the container.
    0 references
    public void ConfigureServices(IServiceCollection services)
    {
        services.AddControllersWithViews();
    }

    // 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");
            // The default HSTS value is 30 days.
            // You may want to change this for production scenarios, see https://ak
            app.UseHsts();
        }
        app.UseHttpsRedirection();
        app.UseStaticFiles();

        app.UseRouting();

        app.UseAuthorization();

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



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

Q13

What is Middleware in ASP.NET Core?

.NET Core

```
// 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?}");
    });
}
```

A middleware is nothing but a component (class) that is executed on EVERY REQUEST in the ASP.NET Core application.

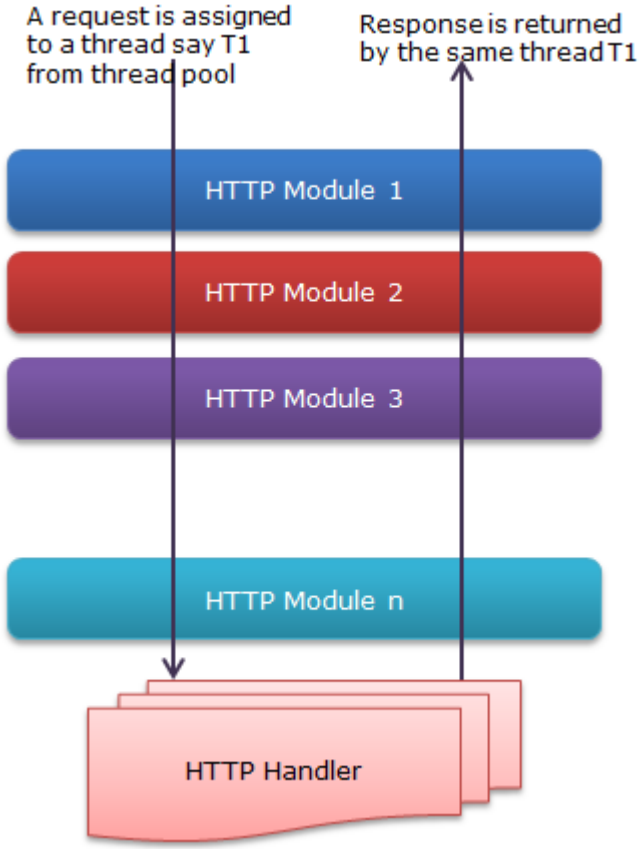
We can set up the middleware in ASP.NET using the CONFIGURE method of our STARTUP class.

Q14

How ASP.NET Core Middleware is different from HttpModule?

.NET Core

```
// 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?}");
    });
}
```



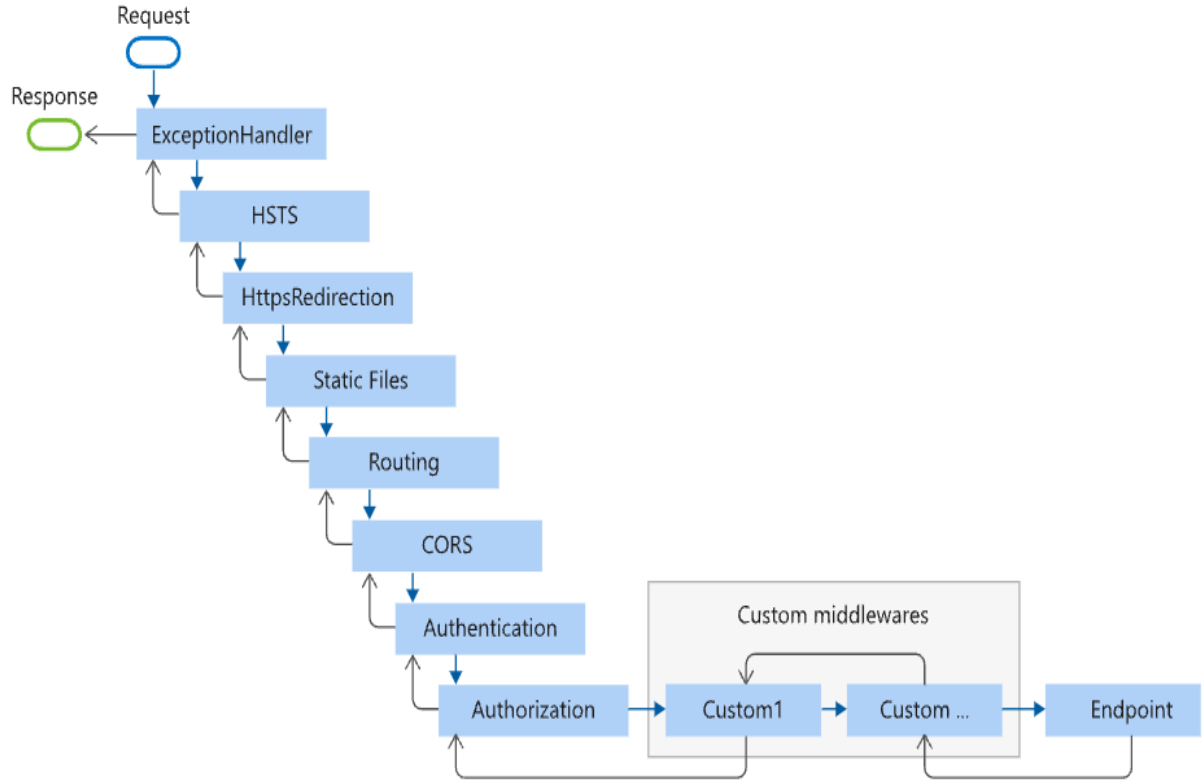
Q15

What is Custom Middleware?

.NET Core

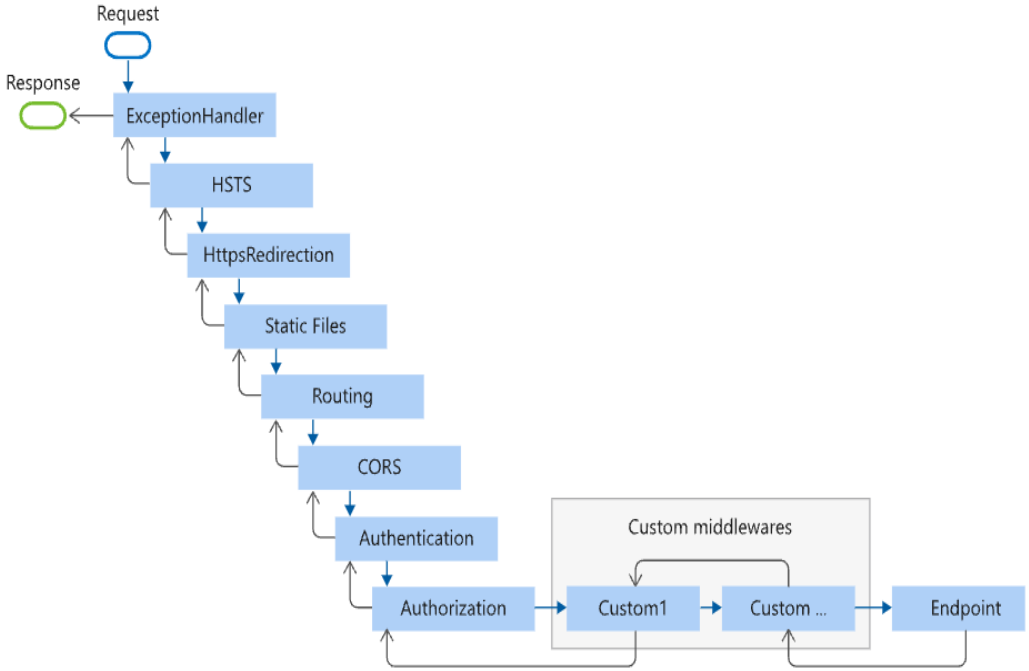
There are middlewares, which are provided by the the .NET Core framework like static file, routing, CORS, Authentication, Authorization.

Similar to that you can also add your own custom middleware and add them into the Configure method of startup.cs class.



Q16

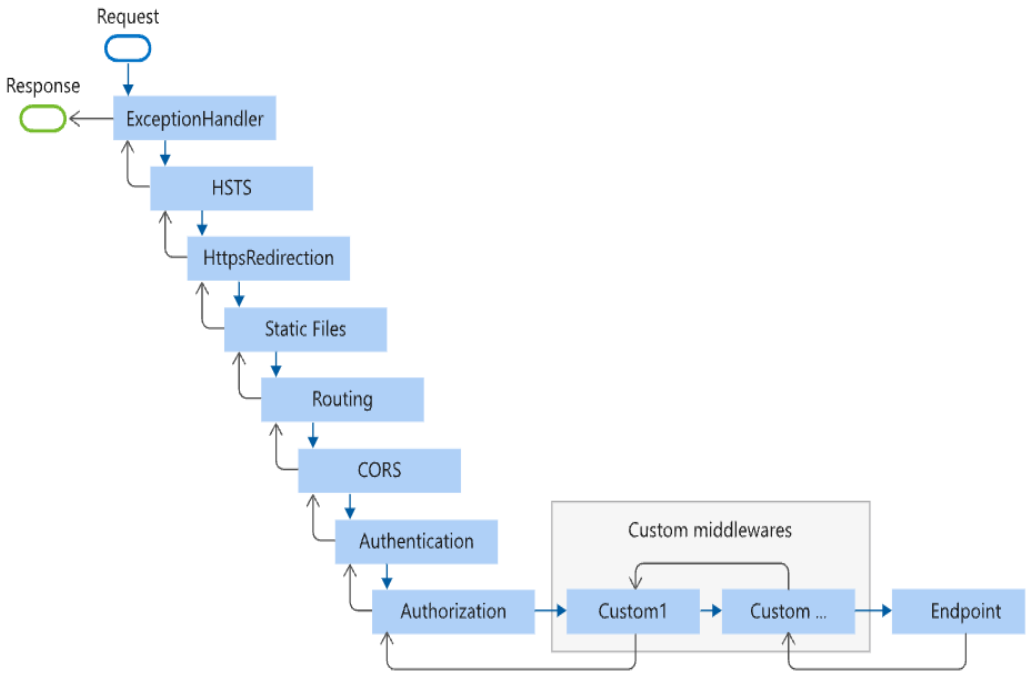
How to add custom middleware in ASP.NET Core?




```
// 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>();
    }
}
```

Q16

How to add custom middleware in ASP.NET Core?



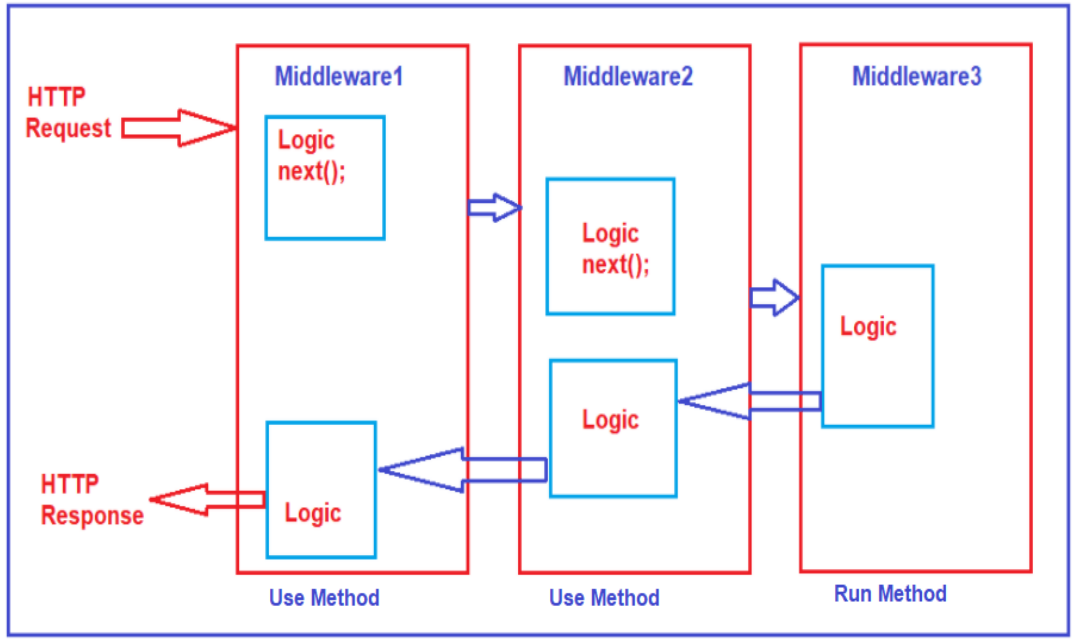

```
// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.
0 references
public void Configure(IApplicationBuilder app, IWebHostEnvironment env)
{
    if (env.IsDevelopment())
    {
        app.UseDeveloperExceptionPage();
    }
    else
    {
        app.UseExceptionHandler("/Error");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

    app.UseStaticFiles();


    app.UseRouting();

    app.UseAuthorization();

    app.UseCustomMiddleware();

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

}
```

Q17

What is Request Delegate in .NET Core?

.NET Core

Request delegates handle each HTTP request and are used to BUILD request pipeline by using RUN, MAP and USE extension methods.



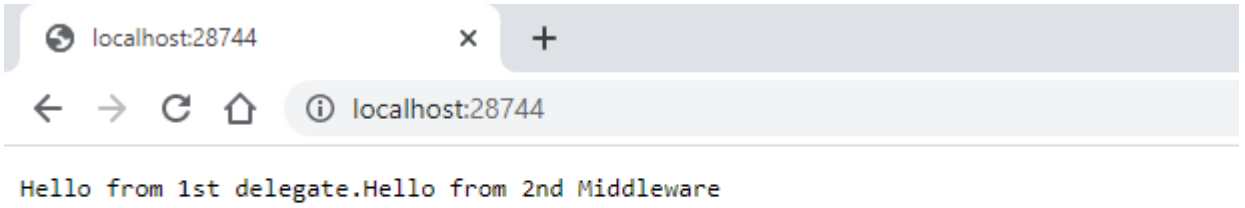
Q18

What is Use() method?

.NET Core

Use method will execute next middleware in sequence

```
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");
            });
        }
```




```
// 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?}");
    });
}
```

Q19

What is Run() method?

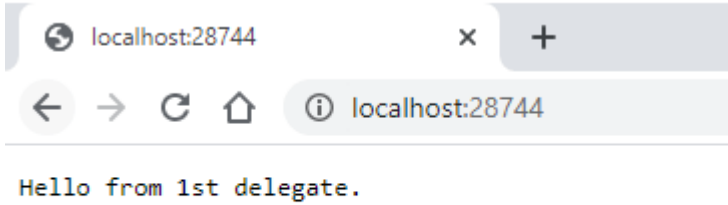
.NET Core

Run method will TERMINATE the chain.

No other middleware method will run after this.

Should be placed at the end of any pipeline.

```
// 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");
            });

        }
```



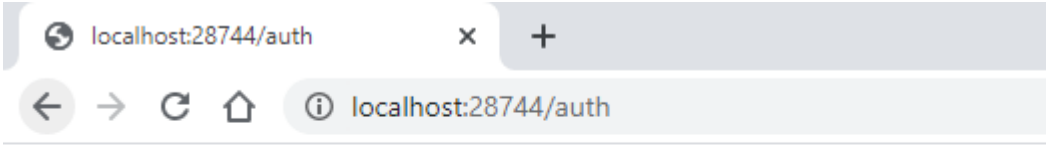
Q20

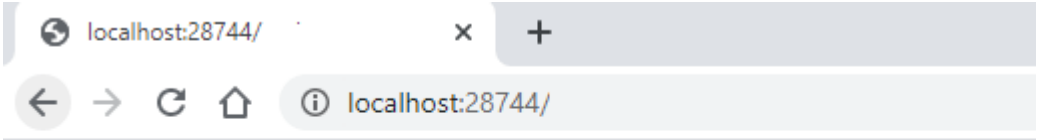
What is Map() method?

.NET Core

Map method will execute middleware if path requested by user equals path provided in parameter.

```
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");
              });
          });
      }
```



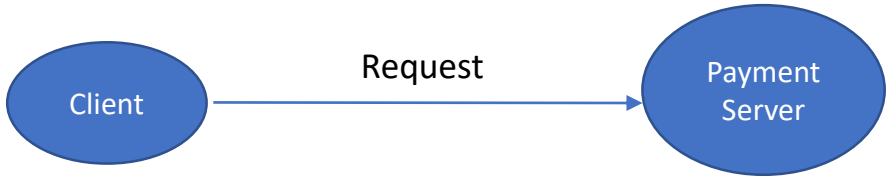


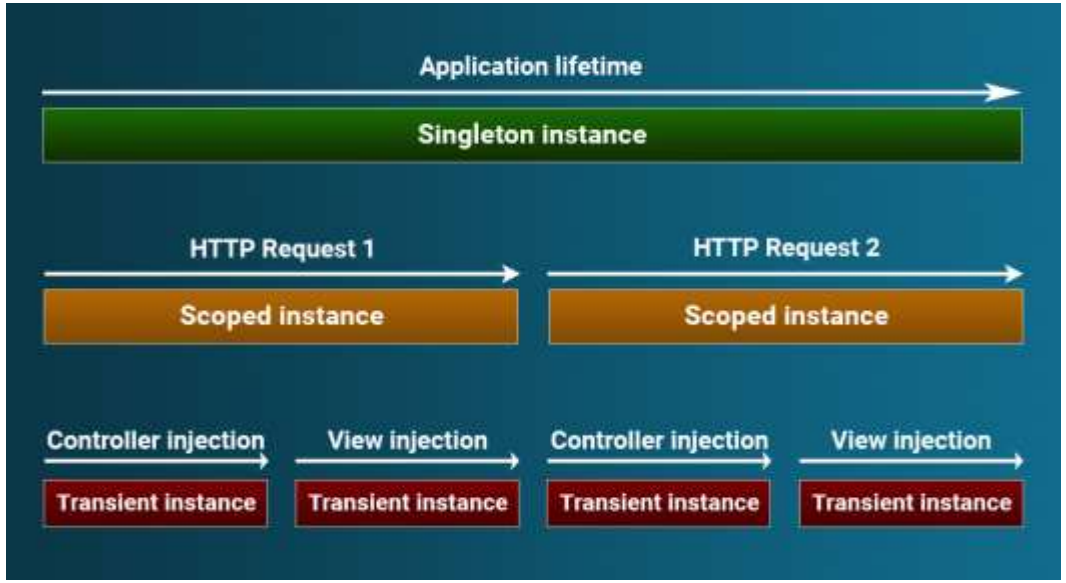
Q21

What are the types of Service Lifetimes of an object/ instance in ASP.NET Core?

.NET Core

The service lifetime controls how long a result object will live for after it has been created by the container.





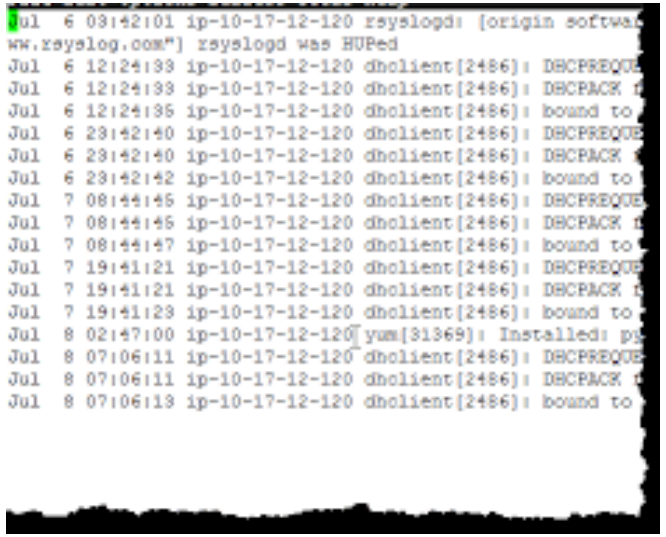
- AddSingleton
- AddScoped
- AddTransient

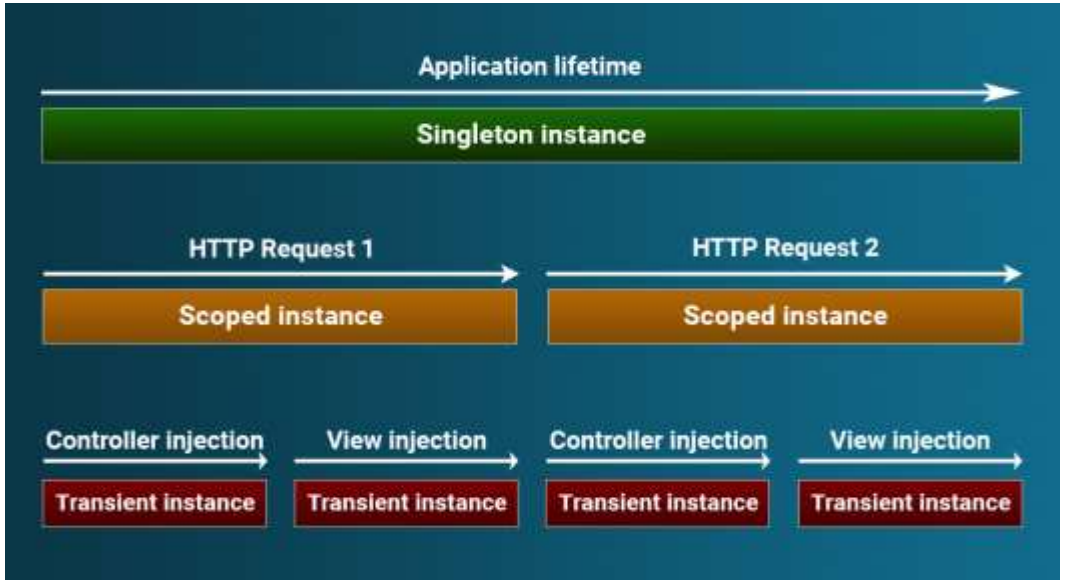
Q22

What is AddSingleton method?

.NET Core

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.



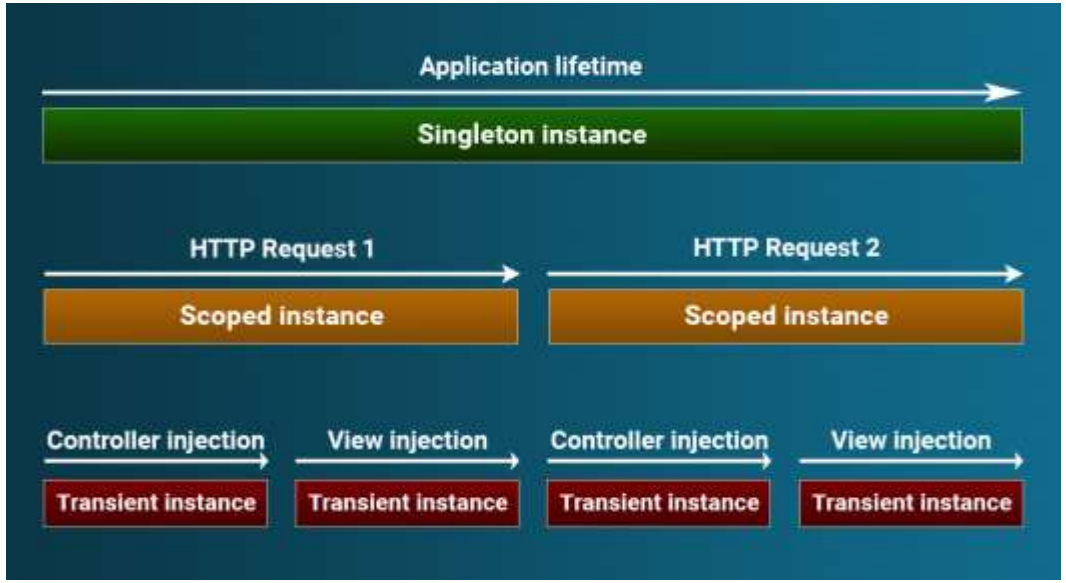


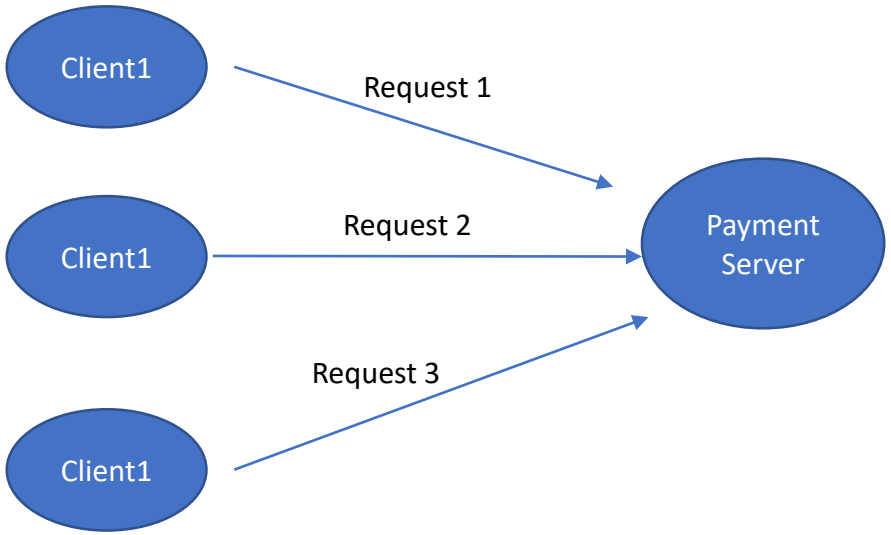

```
// This method gets called by the runtime.
// Use this method to add services to the container.
0 references
public void ConfigureServices(IServiceCollection services)
{
    services.AddSingleton<ILogger, Logger>();
    //services.AddSingleton<IPayment, Payment>();
    //services.AddTransient<IDataAccess, DataAccess>();

}
```

Q23 What is AddScoped method?

AddScoped method create single instance per request.

For every individual request there will be a single instance or object.






```
// 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>();

}
```

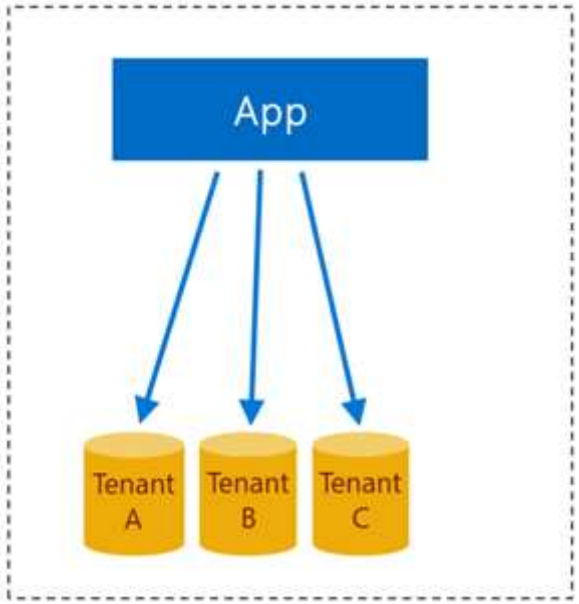
Q24

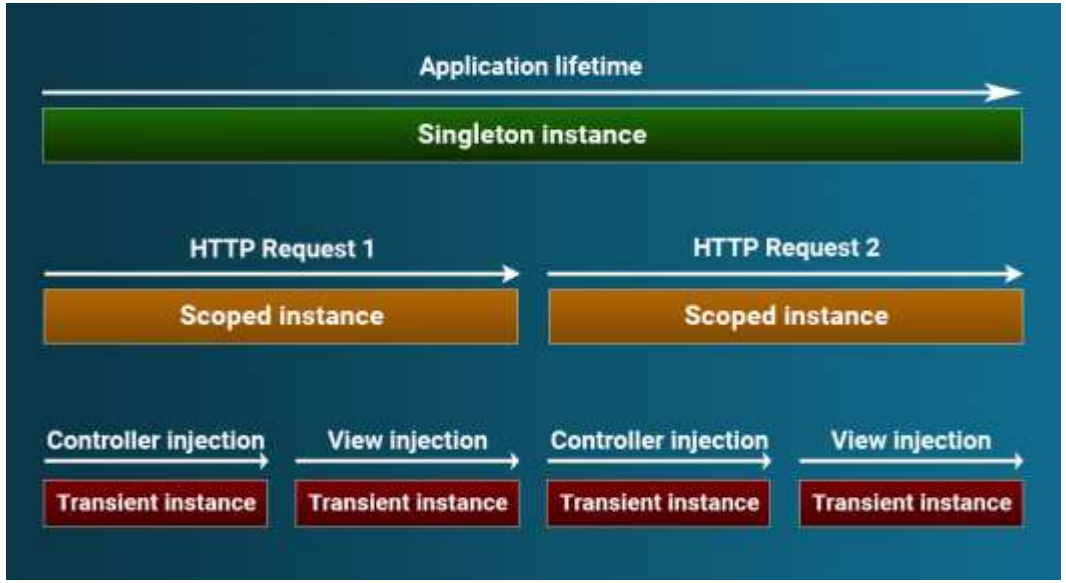
What is AddTransient method?

.NET Core

AddTransient instance will not shared at all even with in the request.

Every time a new instance will be created.





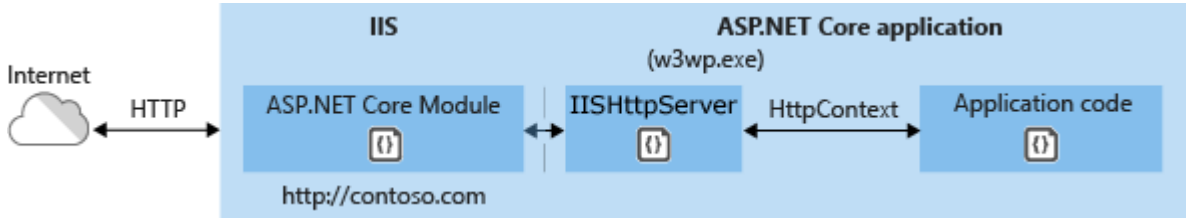

```
// This method gets called by the runtime.
// Use this method to add services to the container.
0 references
public void ConfigureServices(IServiceCollection services)
{
    //services.AddSingleton<ILogger, Logger>();
    //services.AddScoped<IPayment, Payment>();
    services.AddTransient<IDataAccess, DataAccess>();

}
```

Q25

What are the types of Hosting in ASP.NET Core? OR What is in process and out of process hosting?

.NET Core

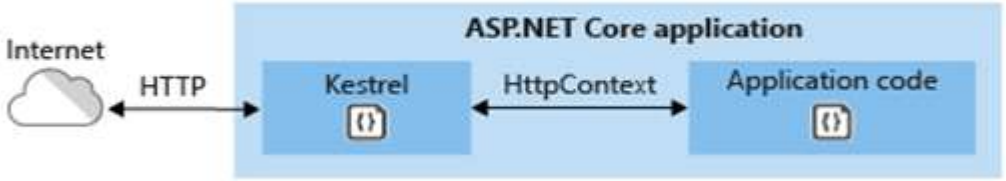
In Process Hosting



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)

Out of Process Hosting

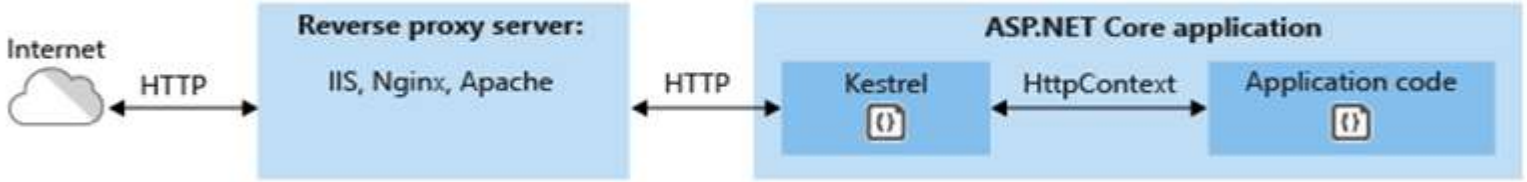
Kestrel used as an edge (Internet-facing) web server:



To configure an app for in-process hosting, set the value of the <AspNetCoreHostingModel> property to InProcess in the project file (.csproj):

```
<PropertyGroup>
<AspNetCoreHostingModel>InProcess</AspNetCoreHostingModel>
</PropertyGroup>
```

Kestrel used in a reverse proxy configuration:



Q26

What is Kestrel?

.NET Core

Kestrel is **a cross-platform web server for ASP.NET Core**.

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.

Q27

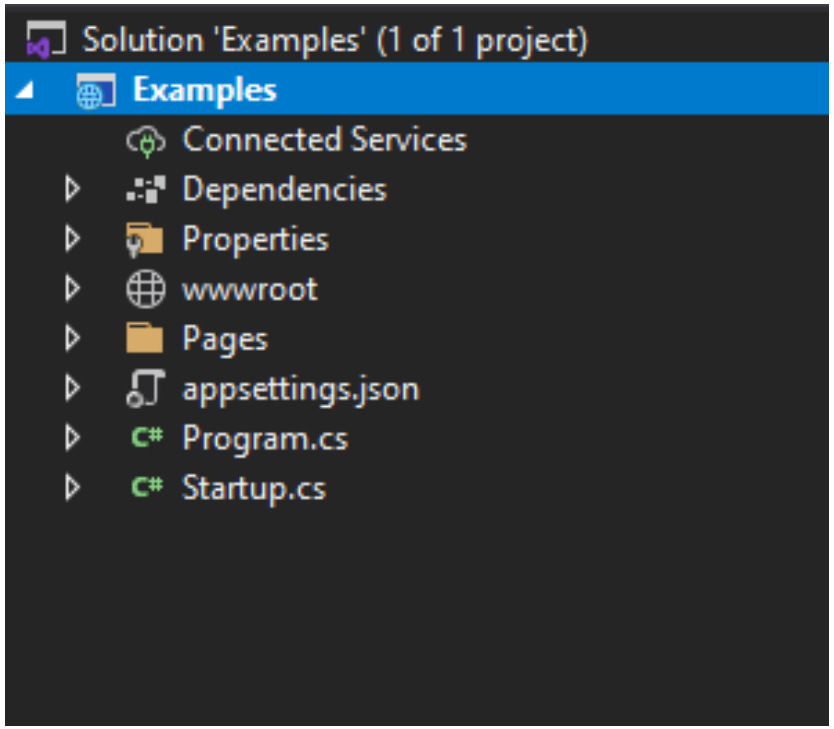
What is the difference between Kestrel and IIS?

.NET Core

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

Q28

Explain default project structure in ASP.NET Core application?



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.

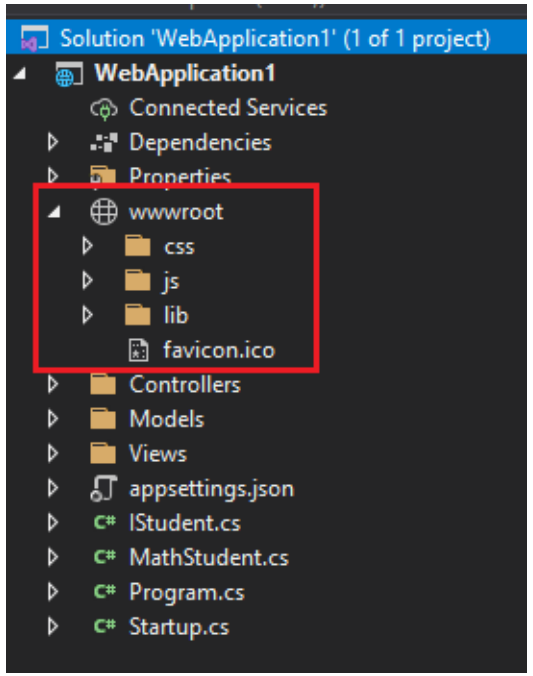
APPSETTINGS.JSON – Configuration settings like database connection strings and other things are saved here.

Q29

How ASP.NET Core serve static files?

.NET Core

- 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?}");
    });
}
```

Q31

What are the main JSON files available in ASP.NET Core?

.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.

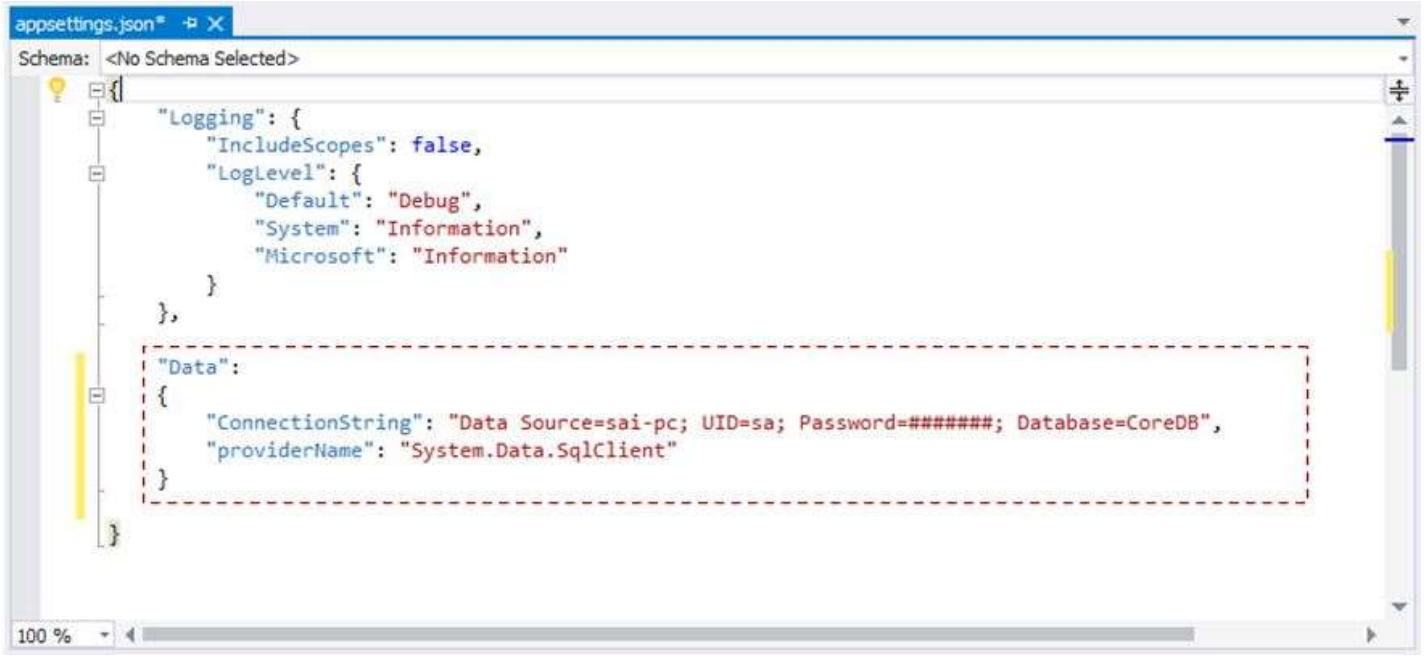
- project.json - ASP.NET Core uses Project.JSON file for storing all project level configuration settings. For example the nugget packages you have installed in the project.

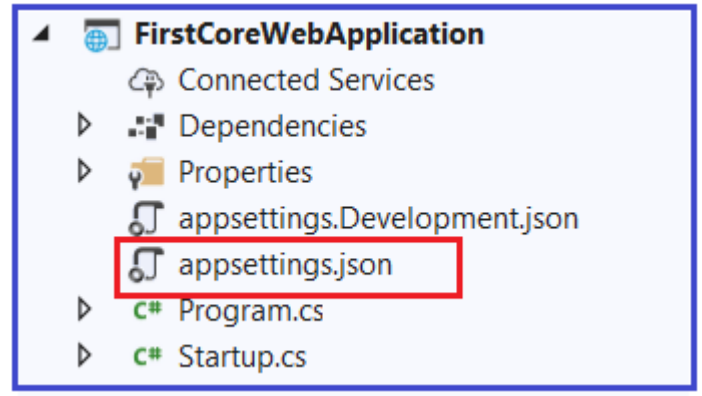
Q32

What is the role Appsettings.Json file in ASP.NET Core?

.NET Core

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.



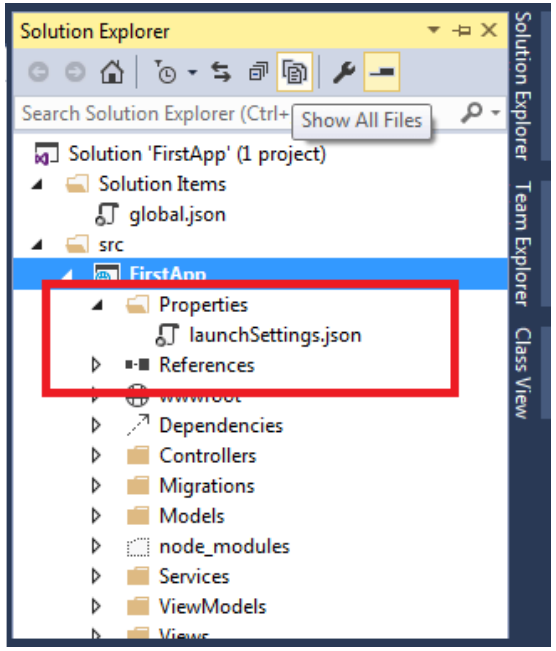


Q33

What is the role of Launchsetting.Json in ASP.NET Core?

.NET Core

The launchSettings. json file is **used to store the configuration information, which describes how to start the ASP.NET Core application**, using Visual Studio.



launchSettings.json

Schema: http://json.schemastore.org/launchsettings.json

```
{
  "iisSettings": {
    "windowsAuthentication": false,
    "anonymousAuthentication": true,
    "iisExpress": {
      "applicationUrl": "http://localhost:50644",
      "sslPort": 44309
    }
  },
  "$schema": "http://json.schemastore.org/launchsettings.json",
  "profiles": {
    "IIS Express": {
      "commandName": "IISExpress",
      "launchBrowser": true,
      "launchUrl": "api/values",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      }
    },
    "HeaderHelper": {
      "commandName": "Project",
      "launchBrowser": true,
      "launchUrl": "api/values",
      "environmentVariables": {
        "ASPNETCORE_ENVIRONMENT": "Development"
      },
      "applicationUrl": "https://localhost:5001;http://localhost:5000"
    },
    "Docker": {
      "commandName": "Docker",
      "launchBrowser": true,
      "launchUrl": "{Scheme}://{ServiceHost}:{ServicePort}/api/values",
      "httpPort": 50645,
      "useSSL": true,
      "sslPort": 44310
    }
  }
}
```

Q34

What is the difference between Appsetting.Json and Launchsetting.Json file?

.NET Core

Remember the Launchsetting.Json is different from appsetting.Json.

Appsetting.Json store the configurations which are required when your application is running.

For example, database connection string is used when application is running and anytime a file can look for connection string.

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.

Q35

What are the various techniques to save configuration settings in ASP.NET Core?

.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

Q36

What is Routing?

.NET Core

Routing is used to handle incoming HTTP requests based on the URL.

http://localhost:52190/Home/Index

```
public class HomeController : Controller
{
    public ViewResult Index()
    {
        return View();
    }
}
```

Q37

What is Attribute routing in ASP.NET Core?

.NET Core

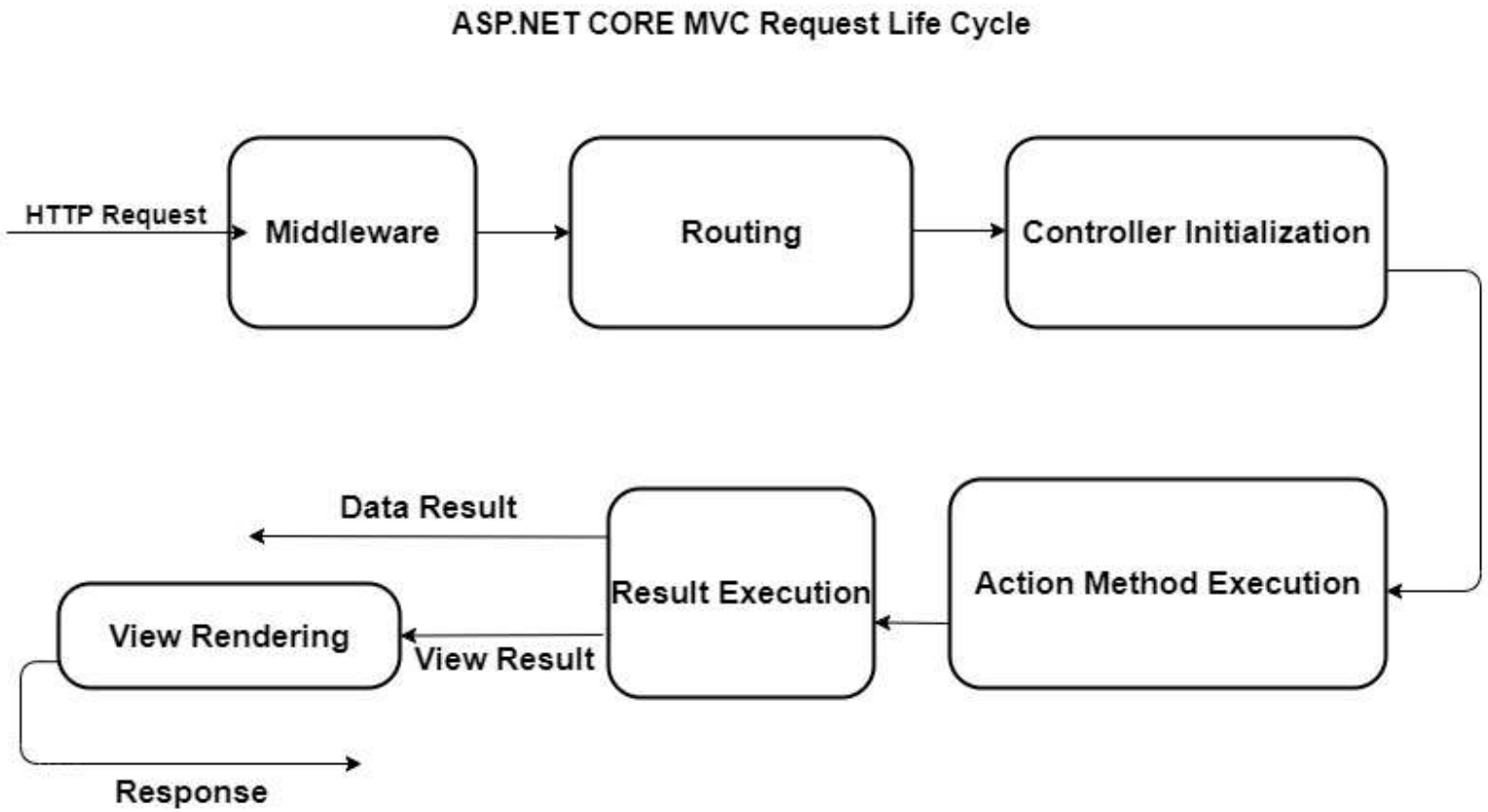
Attribute based routing is the ability to manipulate the behavior of url by Route Attribute

Example
http://localhost:60995/NewIndex

```
[Route("")]
[Route("NewIndex")]
0 references
public IActionResult Index()
{
    return View();
}
```

Q38

Describe the complete Request Processing Pipeline for ASP.NET Core MVC?

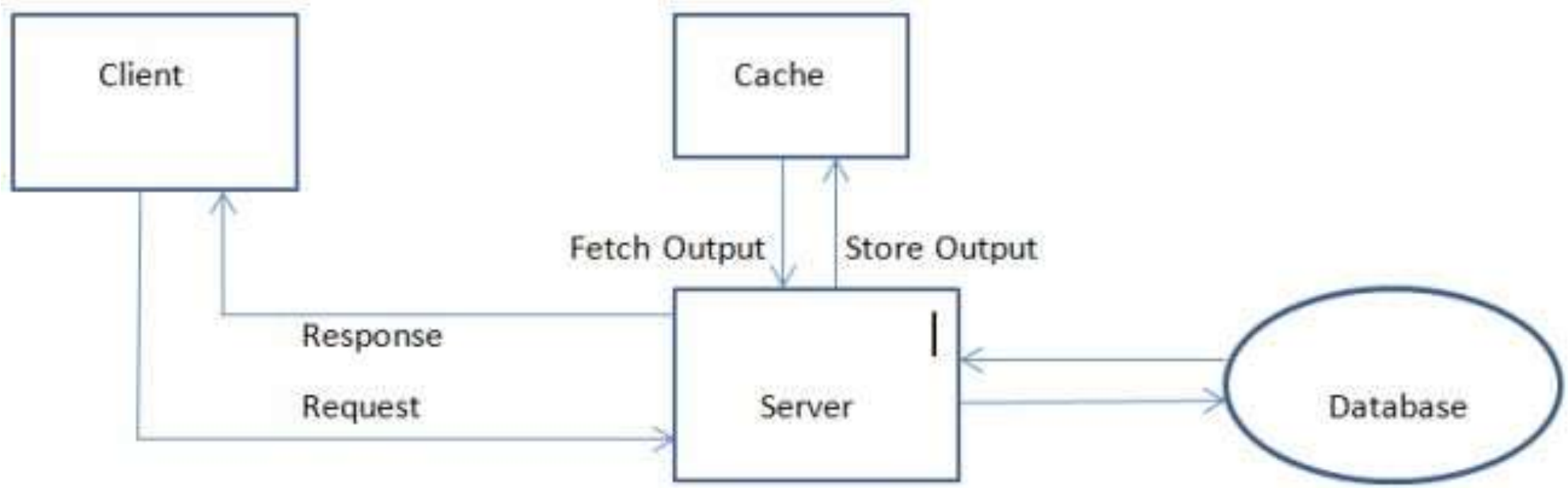


Q39

What is caching in ASP.NET Core?

.NET Core

Caching refers to the process of storing frequently used data so that those data can be served much faster for any **future requests**.



Q40

What is In-Memory caching in .NET Core?

.NET Core

It's the normal way of caching.

An **in-memory** cache is stored in the memory of a single server which is hosting the application.

Basically, the data is cached within the application. This is the easiest way to drastically improve application performance.

Q41

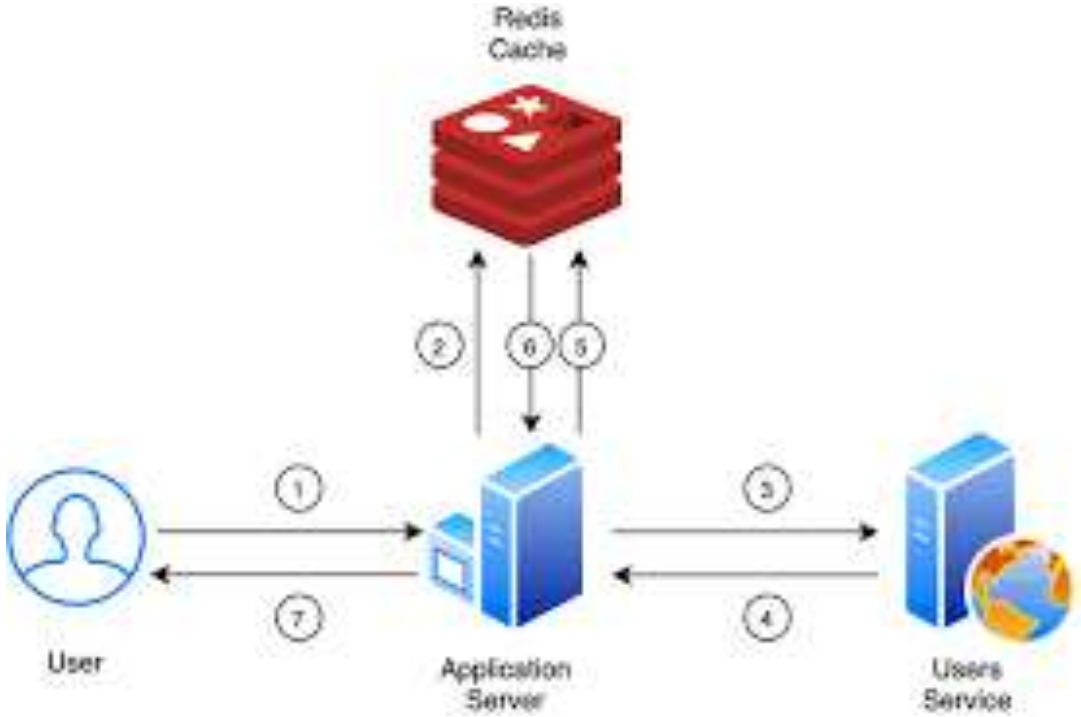
What is Distributed caching?

.NET Core

Distributed caching is when you want to handle caching outside of your application.

redis

Redis is an open-source, highly replicated, performant, non-relational kind of database and caching server.



Q42

When to use In-Memory caching and when to use Distributed caching?

.NET Core

In normal cases where the application size is small, use in-memory cache.

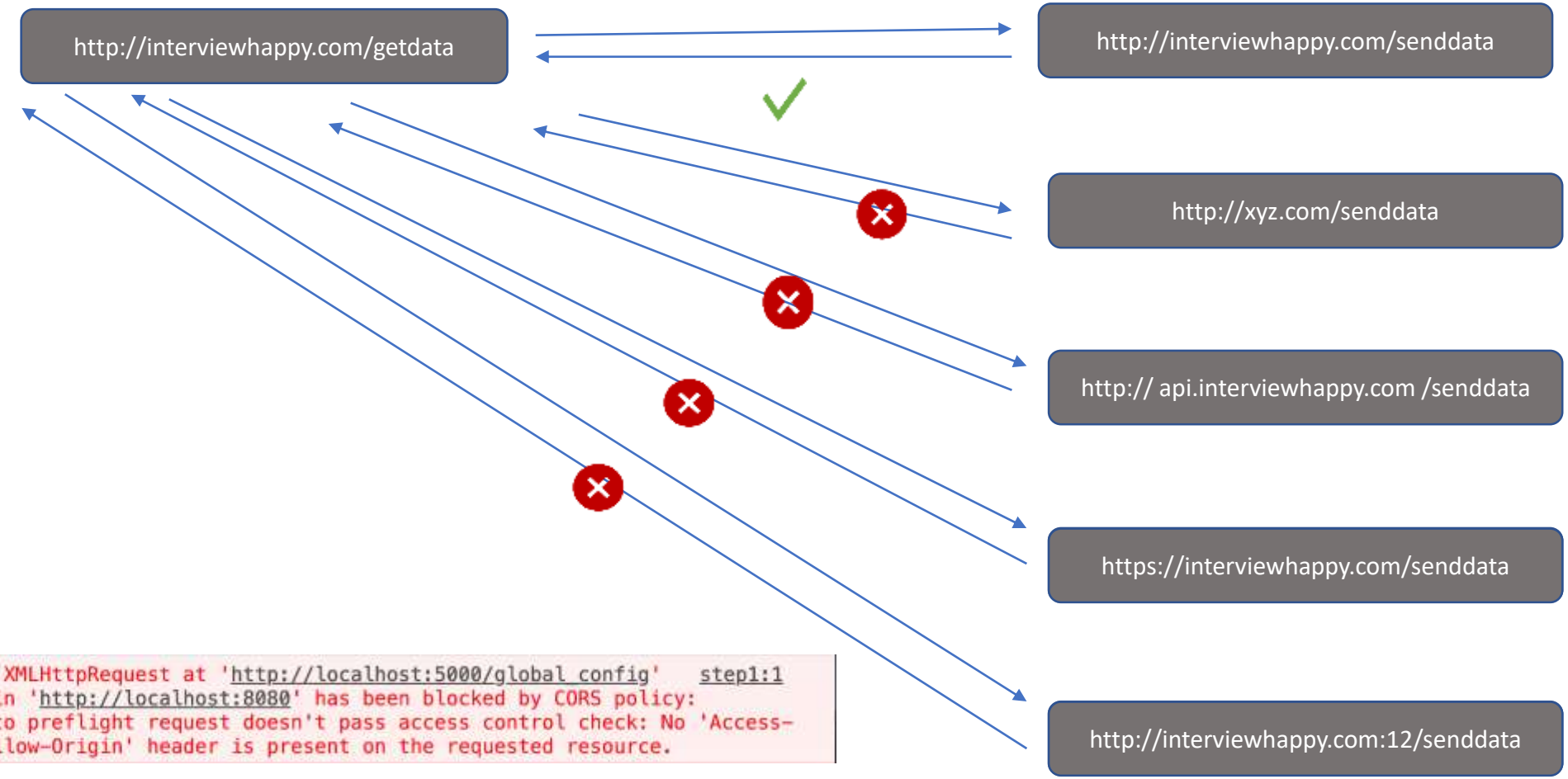
But where application is very big or it's a microservices based architecture, then use distributed caching.

Q43

What is Cross-Origin Requests (CORS) in ASP.NET Core?

.NET Core

The full form of CORS is Cross Origin Resource Sharing.




```
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.
```

Q44

Why CORS restriction is required?

.NET Core

Because of security reason, you can make your api secure by default otherwise any external website can try to access your data.

Q45

How to fix CORS error?

.NET Core

```
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.
```

- Add Microsoft.AspNetCore.Cors nuget package to your project
- In startup class edit the Configure and ConfigureServices method

```
// 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());
}
```

Q46

How to handle errors in ASP.NET Core?

.NET Core

Error handling for development and other environments can be set in CONFIGURE method of Startup.cs class.

```
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.UseEndpoints(endpoints =>
    {
        endpoints.MapRazorPages();
    });
}
```

The IsDevelopment() method will check the ASPNETCORE_ENVIRONMENT value in LaunchSettings.json file.

```
"Example": {
  "commandName": "Project",
  "dotnetRunMessages": "true",
  "launchBrowser": true,
  "applicationUrl": "http://localhost:5000",
  "environmentVariables": {
    "ASPNETCORE_ENVIRONMENT": "Development"
  }
```

Q47

What is Metapackage?

.NET Core

Metapackage is a consolidated package of all the below dependencies.

This is the package present in your application by default, if you are using ASP.NET Core project.

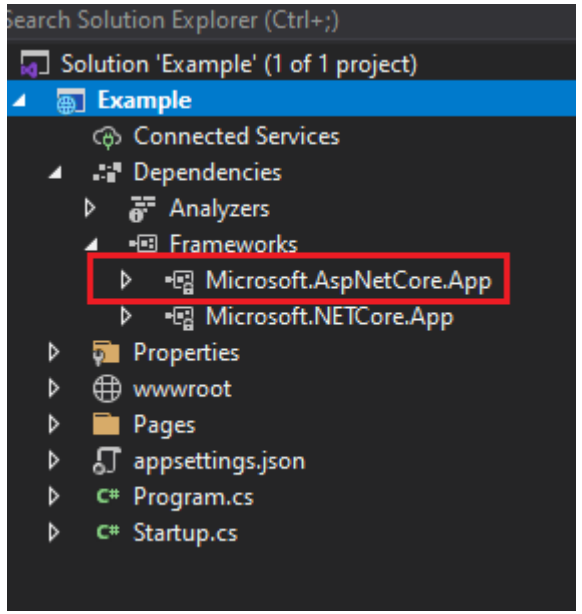
```
Microsoft.AspNetCore.Diagnostics
Microsoft.AspNetCore.Hosting
Microsoft.AspNetCore.Routing
Microsoft.AspNetCore.Server.IISIntegration
Microsoft.AspNetCore.Server.Kestrel
Microsoft.Extensions.Configuration.EnvironmentVariables
Microsoft.Extensions.Configuration.FileExtensions
Microsoft.Extensions.Configuration.Json
Microsoft.Extensions.Logging
Microsoft.Extensions.Logging.Console
Microsoft.Extensions.Options.ConfigurationExtensions
NETStandard.Library
```

Q48

What is the name of Metapackage provided by ASP.NET Core?

.NET Core

Microsoft.AspNetCore.App



Q49

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.

Q50

What are Razor pages in .NET Core?

.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.