# C#, OOCP

## Interface vs Abstract class with real time example

Abstract class have parameterized constructor.

An abstract method is by default a virtual

A property can also be abstract in an abstract class

An abstract cannot be inherited from a class

An abstract cannot be inherited from multiple interfaces

An abstract class can never have a sealed modifier

All derived classes must implement all the abstract methods in it

An interface can contain signatures (declarations) of the **Methods, Properties, Indexers and Events.**

**A Delegate is a type that can't be declared in an interface.**

An interface can inherit from one or more base interfaces.

Abstract class can contain abstract members as well as non-abstract members in it

A class can only inherit from one abstract Class.

We cannot create object of an abstract class.

When we have the requirement of a class that contains some common properties or methods with some common properties whose implementation is different for different classes, in that situation, it's better to use Abstract Class then Interface.

We will use abstract class where we want to restrict the user from creating the object of parent class because by creating object of parent class, you can't call child class methods.

So, the developer has to restrict accidental creation of parent class object by defining it as abstract class.

So, I think, in these ways, we can use abstract class in our real time project.

Thus, in C#, multiple inheritance is not supported. When there is a situation like multiple inheritance, use Interface.

Interface contains only abstract methods.

**It is not possible to pass different type of object as a list without the use of Interface or such clean code.**

We cannot create object of an interface.

In this way, we can use abstract class and interface in our project depending on the condition.

**Interface is use to create loosely coupled components, easily maintainable and pluggable components.**

## Difference between Explicit Interface VS Implicit Interface in C#?

Implicit the interface methods are publicly implemented while in explicit the methods are privately implemented.

Explicit interfaces cannot be used by the derived classes.

If you are implementing an interface explicitly the members become private and your child classes do not have the ability to override the members.

The only time it should be used when for reasons you want to implement the interface but the hide some methods and show methods via the class.

```
public interface IDal
{
      void Add();
      void Update();
}


public class Person : IDal
   {
      public string FirstName { get; set; }
      public string LastName { get; set; }

      public void IDal.Add()
      {
      }

      public void IDal.Update()
      {
      }
   }
```

In this particular scenario the “Personal Information” class would like to only see “FirstName” and “LastName” while the data access class would like to only see “Add” and “Update” methods. Putting in other words we have two different kind of abstraction for the same class.

If you create below object you will get Firstname and Lastname property.

Person person = new Person();

If you create below object you will get Add and Update method.

IDal dal = new Person();

**How can I implement static methods on an interface?**

Not Possible. You can't define static members on an interface in C#.

**Can abstract method be declared outside abstract class?**

No

**New keyword in Interface?**

Possible

**Is it possible to static or sealed abstract class and interface in C#?**

Abstract class and interface cannot be sealed or static.

**How can I implement sealed on static class?**

Not Possible. Static class cannot be marked sealed because it is made sealed by compiler by default.

**Can Interface inherit class?**

No

**Is it possible to implement static constructor, static method and static members in abstract class?**

Yes

**Is it possible to use events in interface?**

Yes

**What happen if method have a same name, same parameter and different return type?**

Get compilation error

**What is output of below code?**

```
public class Test
    {
        public int GetData(int a)
        {
            return 1;
        }

        public int GetData(ref int a)
        {
            return 1;
        }
    }
```

Compiled successfully

**What is output of below code?**

```
public class Test
    {
        public int GetData(out int a)
        {
            a = 5;
            return 1;
        }

        public int GetData(ref int a)
        {
            return 1;
        }
    }
```

Compiled not successfully

**What is output of below code?**

```
public class Test
    {
        public int GetData(out int a)
        {
            a = 5;
            return 1;
        }

        public int GetData(int a)
        {
            return 1;
        }
    }
```

Compiled Successfully

**Can Derived Classes Have Greater Accessibility Than Their Base Types?**

No, Derived classes cannot have greater accessibility than their base types. For example the following code is illegal.

```
using System;
internal class InternalBaseClass
{
public void Print()
{
Console.WriteLine("I am a Base Class Method");
}
}
public class PublicDerivedClass : InternalBaseClass
{
public static void Main()
{
Console.WriteLine("I am a Public Derived Class Method");
}
}
```

**Is it possible to use sealed method with virtual keyword?**

Sealed keyword is always used with override keyword. Sealed method is used to define the overriding level of a virtual method.

**Is it possible to use virtual method in sealed class?**

No

**Is possible public static constructor in C#?**

No. Access modifiers not allowed in public static constructor

**If A Child Class Instance Is Created, Which Class Constructor Is Called First - Base Class Or Child Class?**

When an instance of a child class is created, the base class constructor is called before the child class constructor.

**Can You Have Parameters For Static Constructors?**

No, static constructors cannot have parameters.

**Give 2 Scenarios Where Static Constructors Can Be Used?**

A typical use of static constructors is when the class is using a log file and the constructor is used to write entries to this file.
Static constructors are also useful when creating wrapper classes for unmanaged code, when the constructor can call the LoadLibrary method.

**Thread is part of OOCP?**

No

**Does C# Support Multiple Inheritance?**

No

**Protected Class in C#?**

Not Possible

**Is it possible to create virtual property?**

Yes. Virtual properties enable derived classes to override the property behavior by using the override keyword.

**Is it possible to make static property?**

Yes

**How Do You Access A Constant Field Declared In A Class?**

Constant field can be accessed in the method using the class name and not the instance of the class.

**If A Method's Return Type Is Void, Can You Use A Return Keyword In The Method?**

Yes, Even though a method's return type is void, you can use the return keyword to stop the execution of the method

**What is base class in .net?**

System.Object

**Nested class Inheritance concept in C#?**

Possible

**Which is called first? Public constructor or static constructor**

Static constructor

**Static class can be sealed or not?**

Class cannot be both static and sealed

**Can We Override Private Virtual Method?**

No

**Can We Declare The Override Method Static While The Original Method Is Non Static?**

No

**Can We Allow Class To Be Inherited, but Prevent The Method From Being Overridden ?**

Yes, first create class as public and make its method sealed.

**How Do You Create User Defined Data Types in C#?**

Use the struct, class, interface, and enum constructs to create your own custom types.

**Give Examples For Value Types?**

Enum, Struct

**Give Examples For Reference Types?**

Class, Interface, Delegates and array

**Can Fields Inside A Class Be Virtual?**

No, Fields inside a class cannot be virtual. Only methods, properties, events and indexers can be virtual.

## Do Structs Support Inheritance?

No, structs do not support inheritance, **but they can implement interfaces.**

## How to prevent implementation of interface method in abstract class?

Create abstract method

```
public interface ITest
   {
      void GetData();
   }

   public abstract class myClass : ITest
   {
      public abstract void GetData();

   }
```

## Virtual, New and Override Keyword in C#

**Virtual & Override:** The virtual keyword is used to modify a method, property, indexer, or event declared in the base class and allow it to be overridden in the derived class.

Used in polymorphism implementation

Includes same method name and same parameters

Used in method overriding concept

It is also called run time polymorphism

Causes late binding

**New:** The new keyword is used to hide a method, property, indexer, or event of base class into derived class.

It is also used in polymorphism concept

Includes same method name and different parameters

Used in method overloading concept

It is compile time polymorphism

Cause early binding

## SOLID Design Principles in C#

SOLID design principles are design to manage most of the software design problem

- intended to make software design more understandable, flexible and maintainable

Revising SOLID principles

**S stands for SRP (Single responsibility principle):** A class should take care of only one responsibility.

Each class and module should focus on a single task at a time

Everything in a class should be related to that single purpose

```
class Customer
  {
      private FileLogger obj = new FileLogger();
      publicvirtual void Add()
      {
          try
          {
              // Database code goes here
          }
          catch (Exception ex)
          {
              obj.Handle(ex.ToString());
          }
      }
  }
```

**O stands for OCP (Open closed principle):** Extension should be preferred over modification.

Classes, Modules, functions should be open for extension but closed for modification

Any new functionality should be implemented by adding a new classes, attributes and methods, instead of changing the current ones or existing ones

```
class Customer
{
    public virtual double getDiscount(double TotalSales)
    {
        return TotalSales;
    }
}
 class SilverCustomer : Customer
  {
    public override double getDiscount(double TotalSales)
    {
        return base.getDiscount(TotalSales) - 50;
    }
  }
class goldCustomer : SilverCustomer
  {
    public override double getDiscount(double TotalSales)
    {
        return base.getDiscount(TotalSales) - 100;
    }
  }
```

**L stands for LSP (Liskov substitution principle):** Derived types must be completely substitutable for their base types.

- No new exceptions can be thrown by the subtype

- Clients should not know which specific subtype they are calling

- New derived classes just extend without replacing the functionality of old classes

**I stands for ISP (Interface segregation principle):** Client should not be forced to use an interface if it does not need it.

**D stands for DIP (Dependency inversion principle):** High level modules should not depend on low level modules but should depend on abstraction. Abstractions should not depend upon details. Details should depend upon abstractions.

For example, the code below does not comply with above principles

```
public class HighLevelModule
{
  private readonly LowLevelModule _lowLowelModule;

  public HighLevelModule()
  {
    _lowLevelModule = new LowLevelModule();
  }

  public void Call()
  {
    _lowLevelModule.Initiate();
    _lowLevelModule.Send();
  }
}

public class LowLevelModule
{
  public void Initiate()
  {
    //do initiation before sending
  }

  public void Send()
  {
    //perform sending operation
  }
}
```

## List of ways to do DIP in C#

There are a couple ways to do this:

**Using an Interface**

**Using an Abstract Class**

```
public abstract class OperationBase
{
  public abstract void Send();
}

public class LowLevelModule: OperationBase
{
  public LowLevelModule()
  {
    Initiate();
  }

  private void Initiate()
  {
    //do initiation before sending
  }

  public void Send()
  {
    //perform sending operation
  }
}

public class HighLevelModule
{
  private readonly OperationBase _operation;

  public HighLevelModule(OperationBase operation)
  {
    _operation = operation;
  }

  public void Call()
  {
    _operation.Send();
  }
}
```

**Using a Delegate**

**Method hiding in C#**

For hiding the base class method from derived class simply declare the derived class method with the new keyword.
Whereas in C#, for overriding the base class method in a derived class, you need to declare the base class method as virtual and the derived class method as overridden.

**Partial class in C# and When should I use partial Keyword.**

All the parts spread across different files, must have the same access modifiers.

If any of the parts are declared abstract, then the entire type is considered abstract.

If any of the parts are declared sealed then the entire type is considered sealed.

If any of the parts inherit aa class then the entire type inherits that class.

Partial Method return type must be void.

Partial methods are private by default.

Partial methods can have ref but not out parameters.

Partial methods are implicitly private, and therefore they cannot be virtual.

When working on large projects, spreading a class over separate files enables multiple programmers to work on it at the same time.

**Is It Possible To Create Partial Structs, Interfaces And Methods?**

Yes, it is possible to create partial structs, interfaces and methods. We can create partial structs, interfaces and methods the same way as we create partial classes.

**Can You Create Partial Delegates And Enumerations?**

No, you cannot create partial delegates and enumerations.

**Can Different Parts Of A Partial Class Inherit From Different Interfaces?**

Yes, different parts of a partial class can inherit from different interfaces.

## Can You Specify Nested Classes As Partial Classes?

Yes

```
class ContainerClass
{
public partial class Nested
{
void Test1() { }
}
public partial class Nested
{
void Test2() { }
}
}
```

## What is delegates in C#? Advantages of Delegates in C#.

Delegate is a type which holds the method(s) reference in an object. It is also referred to as a type safe function pointer.

**Delegates are used in the following cases:**

Delegates can be used to handle (call/invoke) multiple methods on a single event.

Delegates can be used to define callback (asynchronous) methods.

Delegates can be used for decoupling and implementing generic behaviors.

Delegates can be invoked method at runtime.

Delegates can be used in LINQ for parsing the ExpressionTree.

Delegates can be used in different Design Pattern.

Delegates will be responsible for broadcasting events to the other forms.

## What is Events?

Events the destination can only listen to it.

## Multicast Delegates in C#?

Multicast delegate is an extension of normal delegate. It helps you to point more than one method at a single moment of time.

## Action, Func, Predicate, Converter, Comparison in C#

**Func<TParameter, TOutput>:**

Func is logically similar to base delegate implementation. The difference is in the way we declare.

At the time of declaration, we need to provide the signature parameter & its return type.

Func<string, int, int> tempFuncPointer;

First two parameters are the method input parameters. 3rd parameter (always the last parameter) is the out parameter which should be the output return type of the method.

**Action<TParameter>:**

Action is used when we do not have any return type from method. Method with void signature is being used with Action delegate.

Similar to Func delegate, the first two parameters are the method input parameters. Since we do not have return object or type, all the parameters are considered as input parameters.

**Predicate<in T>:**

Predicate is a function pointer for method which returns Boolean value.

They are commonly used for iterating a collection or to verify if the value does already exist.

**Converter<TInput, TOutput>:**

Convertor delegate is used when you need to migrate / convert one collection into another by using some algorithm. Object A gets converted into Object B.

**Comparison<T>:**

Comparison delegate is used to sort or order the data inside a collection. It takes two parameters as generic input type and return type should always be int.

This is how we can declare Comparison delegate.

## Static class, sealed class and Singleton class in C#

A static constructor does not take access modifiers or have parameters.

The static constructor for a class executes before any instance of the class is created.

The static constructor for a class executes before any of the static members for the class are referenced.

A static constructor cannot be called directly.

The user has no control on when the static constructor is executed in the program

A typical use of static constructors is when the class is using a log file and the constructor is used to write entries to this file.

Static Class - You cannot create the instance of static class.

Singleton pattern - you can create one instance of the object and reuse it.

Static classes- are loaded automatically by the .NET Framework common language runtime (CLR) when the program or namespace containing the class is loaded.

Singleton instance is created for the first time when the user requested.

We can implement the interfaces with the Singleton class while we cannot implement the interfaces with static classes

Singleton objects stored on heap while static class stored in stack.

Singleton class can initialize lazy way while static class initialize when it loaded first

Static classes are sealed class while Single ton classes are not sealed

Singleton Objects can clone but not with static class

We can dispose the objects of a singleton class but not of static class

Sealed class is used to define the inheritance level of a class.

The sealed modifier is used to prevent derivation from a class. An error occurs if a sealed class is specified as the base class of another class.

## Best way to prevent a Singleton class from being Instantiated?

Declare private constructor in singleton pattern

## Access Modifiers in C#?

In general classes, structs, enums, interfaces, delegates are called as types

Fields, properties, constructors, methods etc, that normally reside in type are called as type members

Type members have 5 access modifiers, whereas types can have only 2 (internal, public) of 5 access modifiers

## Hash Table vs Dictionary in C#?

**Dictionary:**

It returns error if we try to find a key which does not exist.

It is faster than a Hashtable because there is no boxing and unboxing.

Only public static members are thread safe.

Dictionary is a generic type which means we can use it with any data type.

**Hashtable:**

It returns null if we try to find a key which does not exist.

It is slower than dictionary because it requires boxing and unboxing.

All the members in a Hashtable are thread safe,

Hashtable is not a generic type,

## What is IL Code, CLR, CTS, CLS and JIT?

**CLR** - CLR is the heart of the .NET framework and it does 4 primary important things:

Garbage collection

CAS (Code Access Security)

CV (Code Verification)

IL to Native translation

**CTS** - CTS ensures that data types defined in two different languages get compiled to a common data type. This is useful because there may be situations when we want code in one language to be called in other language.

We can see a practical demonstration of CTS by creating the same application in C# and VB.NET and then compare the IL code of both applications. Here, the data type of both IL code is same.

**CLS** - CLS is a subset of CTS. CLS is a set of rules or guidelines. When any programming language adheres to these set of rules, it can be consumed by any .NET language.CTS.

**JIT** - JIT compiles the IL code to Machine code just before execution and then saves this transaction in memory.

## Garbage Collection in C#.

Garbage collector manages allocation and reclaim of memory.

GC works on managed heap, which is nothing but a block of memory to store objects.

There is no specific timings for GC to get triggered, GC automatically start operation.

Managed objects are created, managed and under scope of CLR.

Unmanaged objects are wrapped around operating system resources like file streams, database connections, network related instances, handles to different classes, registries, pointers, etc.

Unmanaged resources can be cleaned-up using 'Dispose' method and 'using' statement.

GC works on managed heap, which is nothing but a block of memory to store objects, when garbage collection process is put in motion, it checks for dead objects and the objects which are no longer used, then it compacts the space of live object and tries to free more memory.

**0 Generation (Zero):** This generation holds short-lived objects, e.g., Temporary objects. GC initiates garbage collection process frequently in this generation.

**1 Generation (One):** This generation is the buffer between short-lived and long-lived objects.

**2 Generation (Two):** This generation holds long-lived objects like a static and global variable, that needs to be persisted for a certain amount of time. Objects which are not collected in generation Zero, are then moved to generation 1, such objects are known as survivors, similarly objects which are not collected in generation One, are then moved to generation 2 and from there onwards objects remain in the same generation.

## How to force Garbage Collection to run? What is advantages of it?

It is possible to force garbage collection by calling Collect, but most of the time, this should be avoided because it may create performance issues

For testing purposes, however, you might want to force garbage collection to happen at a particular time.  You can do this by calling the GC.Collect method.  Calling Collect will force a collection across all generations.  You can also specify the highest generation to collect as follows:

GC.Collect() – Collect generations 0, 1, 2

GC.Collect(0) – Collect generation 0 only

GC.Collect(1) – Collect generations 0, 1

## What is Polymorphism, Encapsulation and Abstraction in C#? How to use in your project?

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

How to abstract: - By using Access Specifiers

Abstraction solves the problem at the design level.

Abstraction hides unwanted data and provides relevant data.

Encapsulation solves the problem in the implementation level.

Encapsulation means hiding the code and data into a single unit to protect the data from the outside world.

The word polymorphism means having many forms. Polymorphism can be static or dynamic. In static polymorphism, the response to a function is determined at the compile time. In dynamic polymorphism, it is decided at run-time.

C# provides two techniques to implement static polymorphism. They are Function overloading and Operator overloading.

Dynamic polymorphism is implemented by abstract classes and virtual functions.

## Inheritance in C#? Multiple vs multilevel inheritance in C#?

Acquiring (taking) the properties of one class into another class is called inheritance. Inheritance provides reusability by allowing us to extend an existing class.

**Multiple inheritance:** C# does not support multiple inheritances of classes.

**Multilevel inheritance:** When one class is derived from another derived class then this type of inheritance is called multilevel inheritance

**Hierarchical inheritance:** This is the type of inheritance in which there are multiple classes derived from one base class. This type of inheritance is used when there is a requirement of one class feature that is needed in multiple classes.

**Single inheritance**
it is the type of inheritance in which there is one base class and one derived class.

## 5 OOCP Optimize Technique

Knowing when to use StringBuilder

Comparing Non Case Sensitive string like str1.ToLower() == str2.ToLower()

Use string.Empty like if (str == string.Empty)

Use List<> instead of ArrayList. Because List<> is less cost effective

Use ‘for’ Loop instead of ‘for-each’ Loop

Remove codes and variables whose are not used

Use Try and catch block

## Method Overloading vs Operator Overloading in C#

**Method overloading:** Creating a multiple methods in a class with same name but different parameters and types is called as method overloading. Method overloading is the example of Compile time polymorphism which is done at compile time.

**Operator Overloading:** Overloaded operators are functions with special names the keyword operator followed by the symbol for the operator being defined. Similar to any other function, an overloaded operator has a return type and a parameter list.

```
public static Box operator+ (Box b, Box c) {
  Box box = new Box();
  box.length = b.length + c.length;
  box.breadth = b.breadth + c.breadth;
  box.height = b.height + c.height;
  return box;
}
```

## What is Generics in C#? Prons and Cons of Generics in C#?

**Generics:** Generics allow you to write a class or method that can work with any data type.
It helps you to maximize code reuse, type safety, and performance.

You can create generic collection classes. The .NET Framework class library contains several new generic collection classes in the System.Collections.Generic namespace.
You can create your own generic interfaces, classes, methods, events, and delegates.

```
static void Swap<T>(ref T lhs, ref T rhs) {
         T temp;
         temp = lhs;
         lhs = rhs;
         rhs = temp;
      }
```

**Advantages:** Generics provide type safety without the overhead of multiple implementations.

No boxing and unboxing required for generics

**Disadvantages:** Enumerations cannot have generic type parameters.

Lightweight dynamic methods cannot be generic.

## What is Constraint in Generics?

Constraints are used in Generics to restrict the types that can be substituted for type parameters. Constraints are represented in C# using the where keyword.

```
public void MyMethod< T >()
    where T : class
{
 ...
}
```

Some of the ways we can use constraints are as follows:

Specifying the type to be a reference type:

```
public void MyMethod< T >()
    where T : class
{
 ...
}
```

Specifying the type to be a value type:

```
public void MyMethod< T >()
    where T : struct
{
 ...
}
```

Specifying a constructor as a constraint:

```
public void MyMethod< T >()
       where T : new ()
{
  ...
}
```

Specifying a static base class as a constraint:

Specifying a generic base class as a constraint:

```
public void MyMethod< T, U >()
      where T : U
{
  ...
}
```

**What is Recursive? Give me real time of example of recursive and also give pros and cons of recursive.**

**Advantages:**

Reduce unnecessary calling of function.

Through Recursion one can solve problems in easy way while its iterative solution is very big and complex.

**Disadvantages:**

Recursive solution is always logical and it is very difficult to trace (debug and understand).

In recursive we must have an if statement somewhere to force the function to return without the recursive call being executed, otherwise the function will never return.

Recursion takes a lot of stack space, usually not considerable when the program is small and running on a PC.

Recursion uses more processor time.

**What happens if finally block throws exception?**

Should be handled at higher level

## Properties in C#? Advantages of properties in C#.

Properties are special kind of class member, In Properties we use predefined Set or Get method. They use accessors through which we can read, written or change the values of the private fields. Before allowing a change in data, the properties can validate the data.

Properties can also provide events when data is changed, such as raising an event or changing the value of other fields.

For example, let us take a class named Employee, with private fields for name, age and Employee Id. We cannot access these fields from outside the class, but we can accessing these private fields through properties.

## Out vs Ref in C#

**Ref:** The parameter or argument must be initialized first before it is passed to ref

It is not required to assign or initialize the value of a parameter (which is passed by ref) before returning to the calling method

**Out:** It is not compulsory to initialize a parameter or argument before it is passed to an out.

A called method is required to assign or initialize a value of a parameter (which is passed to an out) before returning to the calling method

## Const, Read Only and Static Read Only in C#?

**Constants:** Constants can be assigned values only at the time of declaration

Constants are known at compile time

**Read Only:** Read only variables can be assigned values either at runtime or at the time of instance initialization via constructor

Read only variables are known at run time

**Static Read Only:** A Static Read only type variable's value can be assigned at runtime or assigned at compile time and changed at runtime

But this variable's value can only be changed in the static constructor. And cannot be changed further

## Class vs Struct in C#?

Struct is value type and class is reference type

Structs are stored on stack and class are stored on heap

Structs can’t have destructors and can’t inherit from another class

## Reflection in C#?

Reflection objects are used for obtaining type information at runtime.

The classes that give access to the metadata of a running program are in the System.Reflection namespace.

## String vs Convert.Tostring in C#?

Convert.ToString() handles null, while ToString() doesn't

## String vs StringBuilder in C#?

**String:** String is immutable. Immutable means once we create string object we cannot modify.

Any operation like insert, replace or append happened to change string simply it will discard the old value and it will create new instance in memory to hold the new value.

**StringBuilder:** String builder is mutable it means once we create string builder object we can perform any operation like insert, replace or append without creating new instance for every time.

## Collection class in C#?

**Non-generic:** ArrayList, HashTable, SortedList, Stack, Queue

**Generic:** List, Dictionary, SortedList, Stack, Queue

## Type of Constructor in C#?

**Private Constructor:** If a class has a private constructor then it can't be instantiated. A class with private constructor cannot inherited.

**Protected Constructor:** If a class has a protected constructor then it can't be instantiated. A class with protected constructor can inherited.

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

employee emp1 = new employee("Vithal", 23);

```
employee emp2 = new employee(emp1);
```

**Parameterized Constructor:** A constructor with at least one parameter is called a parametrized constructor.

**Static constructor:** A static constructor does not take access modifiers or have parameters.

**What is chaining constructor in C#?**

Calling constructor from another constructor

```
public class mySampleClass

{

public mySampleClass(): this(10)

{

 }

 public mySampleClass(int Age)

{

 }

}
```

**What is assembly?**

Assembly is the smallest unit of deployment of a .net application. It can be a dll or an exe.

There are mainly two types to it:

**Private Assembly:** The dll or exe which is sole property of one application only.

It is generally stored in application root folder

**Public/Shared assembly:** It is a dll which can be used by multiple applications at a time.

A shared assembly is stored in GAC i.e Global Assembly Cache.

**Satellite Assembly**: A Satellite Assembly contains only static objects like images and other

non-executable files required by the application

Satellite Assemblies: are the assemblies to provide the support for multiple languages based on different cultures.

These are kept in different modules based on the different categories available

## Boxing vs unboxing in C#?

**Boxing:** Conversion of a value type into a reference type of variable.

**Unboxing:** It is just the opposite of boxing.

Int i=1;

Object o=i;    //boxing

Int j = (int) o; //unboxing

## Extern Alies in C#?

**Extern** eliminates conflicts. Suppose we have 2 class libraries that contain a class that has the same name. For example, ClassLibrary1 and ClassLibrary2 both introduce the same class.

## Abstract vs Non Abstract method in Abstract class?

The abstract methods cannot have implementations (like interface methods). The non-abstract/concrete methods must have implementations (method body).

## Indexers in C#?

An indexer allows an object to be indexed such as an array. When you define an indexer for a class, this class behaves similar to a virtual array.

## Give me real time example of indexes in C#.

Session in web have two indexer Session["Session1"] = "Session 1 Data"

## Yield keyword in C#?

This keyword interacts with the foreach-loop. In this way it can improve performance.

## Tuple in C#?

Tuple (Tuple in C#) is an ordered sequence, immutable, fixed-size and of heterogeneous objects, i.e., each object being of a specific type

## Anonymous methods & Type in C#

Anonymous types allow us to create new type without defining them

```
var anonymousData = new

{

   ForeName = "Jignesh",

   SurName = "Trivedi"

};
```

**Anonymous methods:** Anonymous methods let you declare a method body without giving it a name

## Lambda in C#?

A lambda expression is an anonymous function that you can use to create delegates or expression tree types.

## Dispose vs Finalize vs Destructor in C#?

**Destructor:** It will be called by GC process while collecting the garbage.

**Dispose:** Dispose method can be invoked only by the classes that IDisposable interface.

**Finalize:** Finalize () is called by Garbage Collector implicitly to free unmanaged resources.

## Using in C#

The using statement simplifies the code that you have to write to create and then finally clean up the object. The using statement obtains the resource specified, executes the statements and finally calls the Dispose method of the object to clean up the object.

```
using (TextWriter w = File.CreateText("log.txt"))
{
   w.WriteLine("This is line one");
}
```

## & vs &&, | vs || in C#

In that case, the difference is that && and || short-circuit the evaluation, i.e. && stops if the first operand is false and || stops if the first operand is true, while & and | always evaluate both operands

## Throw vs throw ex in C#

In Throw, the original exception stack trace will be retained. To keep the original

Stack trace information, the correct syntax is 'throw' without specifying an exception.

In Throw ex, the original stack trace information will get override and you will lose the

Original exception stack trace. I.e. 'throw ex' resets the stack trace.

## Debug vs Trace class in C#

**Debug:**

It uses Debug class.

It uses in debug build.

It uses the time of application development.

In Debug mode compiler inserts some debugging code inside the executable.

Debug class works only in debug mode.

Performance analysis cannot be done using Debug.

Debugging uses to find error in program.

For Debug we can use Debug.Write() method.

Debug runs in same thread as main program execute.

**Trace:**

It uses Trace class.

Trace statement includes by default when program compiled into released build.

Trace class is used for testing and optimization even after an application is compiled and released.

Trace class works in both case Debug mode as well as release mode.

Trace runs in different thread form main program execute thread.

For Trace we can use Trace.Write() method.

It uses time of application deployment.

**is vs as in C# ?**

Is operator returns true if an object can be cast to a specific type, otherwise false.

As operator attempts to cast an object to a specific type and return null if it fails.

**Public vs internal in C#**

Public is visible from wherever.

Internal is visible only within an assembly

**string vs String in C#**

String is an alias in C# for System.String. So technically, there is no difference. It's like int vs. System.Int32.

**Base keyword in C#**

The base keyword is used to refer to the base class when chaining constructors or when you want to access a member (method, property, anything) in the base class that has been overridden or hidden in the current class. For example,

```
class A {

  protected virtual void Foo() {

    Console.WriteLine("I'm A");

  }

}


class B : A {

  protected override void Foo() {

    Console.WriteLine("I'm B");

  }
```

```
    public void Bar() {
        Foo();
        base.Foo();
    }
}
```

## Sealed method in C#

Sealed method is used to define the overriding level of a virtual method.

Sealed keyword is always used with override keyword.

## Var, Dynamic and Object in C#

**Var:** Introduced in C# 3.0

Statically typed – This means the type of variable declared is decided by the compiler at compile time

Need to initialize at the time of declaration.

Useful in LINQ and anonymous types

We cannot declare a method parameter or return type as “VAR”

Useful when we don’t have more information about the data type.

It is type safe. It means that the compiler has all information about the type so there is no issue at runtime

e.g., var str=”I am a string”;

**Dynamic:** Introduced in C# 4.0

Dynamically typed - This means the type of variable declared is decided by the compiler at runtime time.

No need to initialize at the time of declaration.

We can declare method parameter and return type as “Dynamic”

It is not type safe. Compiler does not have all information about the type

e.g., dynamic str;

**Object:** Object was introduced with C# 1.0

It can store any kind of value, because object is the base class of all type in .NET framework.

Useful when we don't have more information about the data type.

We can declare method parameter and return type as "Object"

## Nullable in C#

nullable types that allow you to assign null to value type variables. You can declare nullable types using Nullable<t> where T is a type.

```
int? i = null;
double? D = null;
```

## Params in C#

params parameter must last parameter in formal parameter list.

Only one params keyword is permitted in a method declaration.

The params parameter must be a single dimension array.

```
public static int Add(params int[] ListNumbers)
   {
}
```

## Jagged Array in C#?

A jagged array is an array whose elements are arrays. The elements of a jagged array can be of different dimensions and sizes.

A jagged array is sometimes called an "array of arrays."

## What is enum? Possiblle enum in static class?

Enums are enumerations

enums are strongly typed constants

The default underlying type of as enum is int

The default value for first element is zero and gets incremented by 1

Enums are types, not members; there is no concept of a static or non-static enum

**Yes, declare enum in static class**

## Array vs ArrayList in C#?

**Array:** An Array (System.Array) is fixed in size once it is allocated

You can't add items to it or remove items from it.

Also, all the elements must be the same type

**ArrayList:** An ArrayList is a flexible array which contains a list of objects

You can add and remove items from it and it automatically deals with allocating space.

If you store value types in it, they are boxed and unboxed, which can be a bit inefficient. Also, it is not type-safe.

## Is it possible to store mix datatypes such as int, string, float, char, and object all in one array?

Yes, use object type

```
object[] array = new object[3];
array[0] = 101;
array[1] = "C#";
```

## Extensions in C#?

Extension methods enable you to "add" methods to existing types without creating a new derived type, recompiling, or otherwise modifying the original type.

An extension method is a static method of a static class, where the "this" modifier is applied to the first parameter. The type of the first parameter will be the type that is extended.

## SDLC in C#

The following are the typical phases of a Software Development Life Cycle: Project initiation and planning, Feasibility study, System design, Coding, Testing, Implementation, and Maintenance

## Localization vs Globalization in C#

Globalization is the process of designing and developing applications that function for multiple cultures.

Localization is the process of customizing your application for a given culture and locale.

**C# 6.0 features in C#**

**1) string strname = null;**

string[] splitname1 = strName?.Split();

**2) Auto Property Initializer**

```
private readonly Guid _idOld = Guid.NewGuid();//Backing Field

public Guid Id

{

   get { return _idOld; }

}
```

**3) Expression Bodied Function and Members**

```
public string GetNameN(string fullname) => fullname;

public void CalculateN(int x, int y) => Console.WriteLine("Total:" + (x + y));
```

**4) Dictionary Initializer**

```
public Dictionary<string, Customer> clist = new Dictionary<string, Customer>()

{

   {"a", new Customer() { CustomerName = "Ron" }},

   {"b", new Customer() { CustomerName = "Robert" }}

};

public Dictionary<string, Customer> clist = new Dictionary<string, Customer>()

{

   ["a"] = new Customer() { CustomerName = "Ron" },
   ["b"] = new Customer() { CustomerName = "Robert" }

};
```

**5) String Interpolation**

```
Console.WriteLine(string.Format("My fav football player is {0} {1}", firstName, lastName));

Console.WriteLine("My fav football player is \{firstName} \{lastName}");
```

**Synchronization vs Asynchronization in C#?**

Synchronization means two or more operations are running in a same context (thread) so that one may block another.

Synchronization means two or more operations happen sequentially.

Asynchronous means two or more operations are running in different contexts (thread) so that they can run concurrently and do not block each other.

Asynchronous means two or more operations happen asynchronously.

**Thread, Semaphore, Mutex, Lock, Monitor in C#**

**Thread:** Threads are often called lightweight processes. However they are not process.es A Thread is a small set of executable instructions, which can be used to isolate a task from a process.

**Lock:** locking is used to ensure that only one thread can enter particular sections of code at a time.

**Monitor:** Monitor provides a mechanism that synchronizes access to objects. Monitor class has the following methods for the synchronize access to a region of code by taking and releasing a lock:

Monitor.Enter

Monitor.TryEnter

Monitor.Exit

**Mutex:** mutex is locking mechanism used to synchronize access to a resource. Only one task (can be a thread or process based on OS abstraction) can acquire the mutex. It means there is ownership associated with mutex, and only the owner can release the lock (mutex).

**Semaphore:** A Semaphore is used to limit the number of threads that can have access to a shared resource concurrently.

## Explain unit of work design pattern in C#.

Unit of Work design pattern does two important things: first it maintains in-memory updates and second it sends these in-memory updates as one transaction to the database.

So to achieve the above goals it goes through two steps:

- It maintains lists of business objects in-memory which have been changed (inserted, updated, or deleted) during a transaction.
- Once the transaction is completed, all these updates are sent as one **big unit of work** to be persisted physically in a database in one **go**.

This IEntity interface will have an ID property and methods (insert, update, delete, and load) which will help us to do the CRUD operation on the business object. The ID property is a unique number which helps us uniquely identify the record in a database.

```
public interface IEntity
{
   int Id { set; get; }
   void Insert();
   void Update();
   List<IEntity> Load();
}
```

## What is Managed Extensibility Framework in C#?

Managed Extensibility Framework is a new framework from Microsoft Corporation to build Extensible applications. Its basic purpose is to plug-in components to an already running application.

**Assembly:** System.ComponentModel.Composition.dll

**Keyword:** Export, Import

MEF has many advantages as we have seen by this time. However, some of the common advantages are listed below:

MEF breaks the tightly coupled dependencies across the application but respects the type checking of the loosely coupled parts.

Applications can be extended.

Components can be added at runtime.

Dynamic discovery of the components.

Great piece of reusability.

**Is it possible to inject class in C#?**

Yes

**Inherit constructor of base class in derived class, possible?**

Constructors are not inherited. If you would like the derived class constructor to do the same as parent class constructor, call it using base.

**Macros in C#?**

No, C# does not support preprocessor macros like C.

**Class is initialize in heap or stack?**

Heap

**Structure have constructor?**

Structures cannot have default constructor. Structure have a parameterized constructor.

**How many finally blocks can be written for a single try block?**

Only one

Code in finally blocks is always executed

**Task in C#**

A Task represents some asynchronous operation and is part of the Task Parallel Library, a set of APIs for running tasks asynchronously and in parallel.

Task supports cancellation through the use of cancellation tokens.

A task can have multiple processes happening at the same time.

It can be used whenever you want to execute something in parallel. Asynchronous implementation is easy in a task, using' async' and 'await' keywords.

## Difference between Synchronous and Asynchronus programing in C#?

**Synchronous programming:** In Synchronus execution, the program runs all tasks in sequence. When you execute something synchronously, you wait for it to finish before moving on to another task.

It takes longer to finish.

It may stop the user interface (UI) thread.

**Asynchronus programming:** In Asynchronus execution, the program doesn't run all tasks in sequence. When you execute something asynchronously, you can move on to another task before it finishes.

There is no logical sequence anymore. The tasks can end at any time, and you don't have control of which one finishes first.

You must synchronize tasks. For e.g. you run a task that must be executed after the other three have finished. You will have to create a mechanism to wait for all tasks to finish before launching the new task.

You must address concurrency issues. If you have a shared resource, like a list that is written in one task and read in another, make sure that it's kept in a known state.

## Code Review in C#

Code Review is nothing; it is a verifying and finding defects on source code in development phase. It is an improvement of source code quality to provide quality software for the developer's in development phase. It is an available market lot of code review tools (free/paid) like FxCop, StyleCop, NDepend, CodeRush, .NET Compiler Platform and etc.

The important code review areas as following

Provide a proper comments

Proper naming convention

Security

Proper error handling

Thread safety

Performance

Properly use of external/internal components

Architecture design and units of functionality including module integration

Proper unit testing and integration testing

**What is difference between pascal and camel naming conversation?**

**Pascal:** In this the first letter of every word is in capital letter.

Always use **Pascal** for class names, Interfaces, Method names

```
public partial class About : Page
{
    //
}
```

**Note**: Don't use name as all character in CAPS.

**Camel:** In this the first letter of word always in small letter and after that each word with capital letter.

Always use camelCase with Private Member Variable, method arguments and local variables. Don't use Hungarian notation for variables.

```
public string GetPosts(string postId

{

int numberOfPost = 0;

}
```

**Note:** Don't use abbreviations for any words and don't use underscore ( _ ) in between any name.

**Note:** Don't use name with start with numeric character.

**What is difference between build, rebuild and clean?**

**Build:** This will perform an incremental build. In other words it will only build code files which have changed. If they have not changed those files will not be touched.

**Rebuild:** This will delete all currently compiled files (i.e., exe and DLLs) and will build everything from scratch

**Clean:** This menu will delete all compiled files (i.e., EXE's and DLL's) from the *bin/obj* directory

Rebuild = Build + Clean

### What is Dependancy Injection in C#?

Dependency Injection (DI) it is a software design pattern which enables the development of loosely coupled code. Through DI, you can decrease tight coupling between software components. It is also known as Inversion-of-Control.

We have the following ways to implement Dependency Injection.

Constructor Injection

Property Injection

Method Injection

### Which .Net Dependency Injection Framework are worth looking into?

Unity

Autofac

NInject

StructureMap

### What is Serialization?

**The types of Serializations are given bellow:**

**Binary Serialization**: In this process all the public, private, read only members are serialized and convert into stream of bytes. This is used when we want a complete conversion of our objects.

**SOAP Serialization**: In this process only public members are converted into SOAP format. This is used in web services.

**XML Serialization**: In this process only public members are converted into XML. This is a custom serialization. Required namespaces: System.Xml, System.Xml.Serialization.

# Design Pattern

## What is design pattern?

Design patterns are evolved as reusable solutions to the problems that we encounter every day of programming can be applied to the real world problems

**Creational:** This type deals with the object creation and initialization. This pattern gives the program more flexibility in deciding which objects need to be created for a given case

**Structural:** This type deals with class and object composition. This pattern focuses on decoupling interface and implementation of classes and its objects

**Behavioural:** This type deals with the communication between classes and objects

## What is difference between association, Aggregation and Composition?

**Association:** Association is a relationship among the objects. We can define a one-to-one, one-to-many, many-to-one and many-to-many relationship among objects. Association is a more general term to define a relationship among objects. We can define a one-to-one, one-to-many, many-to-one and many-to-many relationship among objects. Association is a more general term to define a relationship among objects. For example Managers and Employees, multiple employees may be associated with a single manager and a single employee may be associated with multiple managers.

**Aggregation:** In Aggregation, the direction specifies which object contains the other object. There are mutual dependencies among objects. For example, departments and employees, a department has many employees but a single employee is not associated with multiple departments.

**Composition:** In this type of Aggregation the child object does not have their own life cycle. The child object's life depends on the parent's life cycle. Only the parent object has an independent life cycle. If we delete the parent object then the child object(s) will also be deleted. For example, the company and company location, a single company has multiple locations. If we delete the company then all the company locations are automatically deleted. The company location does not have their independent life cycle, it depends on the company object's life (parent object).

## What is difference between shallow and deep copy?

**Shallow Copy:** Shallow copying is creating a new object and then copying the *non-static fields* of the current object to the new object. If a field is a **value type** --> a bit-by-bit copy of the field is performed; for a **reference type** --> the reference is copied but the referred object is not; therefore the original object and its clone refer to the same object.

In C# and VB.NET, shallow copy is done by the object method MemberwiseClone().

**Deep Copy:** Deep copy is creating a new object and then copying the *nonstatic fields* of the current object to the new object. If a field is a **value type** --> a bit-by-bit copy of the field is performed. If a field is a **reference type** --> a new copy of the referred object is performed.

**Note:** the classes to be cloned must be flagged as [Serializable].

**What is creational design pattern?**

**Factory Pattern:** Create object through inheritance produce only one product

In this pattern we define an interface which will expose a method which will create objects for us. Return type of that method is never be a concrete type rather it will be some interface (or may be an abstract class)

Define an interface for creating an object, but let subclasses decide which class to instantiate

**Abstract Factory Pattern:** create object through composition produce families of products

In Abstract Factory we define an interface which will create families of related or dependent objects

All this objects will together become the part of some important functionality

**Builder pattern:** separate construction of a complex object from its representation so that the same construction process can create different representation

**Prototype pattern:** create a new object by copying this prototype

**What is structure design pattern?**

**Adapter:** Match interface of different classes allow two incompatible interfaces to work together

**Bridge:** separate object's interface from its implementation

**Composite:** tree structure of simple and composite objects Enable hierarchical tree structures of varying complexity group of objects are treated in the same way as a single instance of an object

**Decorator:** add additional responsibility to object dynamically

**Facade:** a single class represents an entire subsystem defines high level single interface

**Flyweight:** Flyweight pattern focuses on sharing for efficiency allows program to avoid the expense of multiple instances that contain the same information by sharing one instance

**Proxy:** object representing another object

**What is behavior design pattern?**

**Chain of Responsibility:** Chain of responsibility design pattern in useful when you want to handle the request by different handlers sequentially (in the chain) before delivering it to the client

Consider we have to write an application for an organization where employee can apply for the leave. The approval of leave depends upon the days count in applied leave.

If leave days are up to 10 working days, then supervisor can approve the leave.

If leave days are in between 11 to 30 days, then project manager will approve the leave.

If leave days are greater than 30 days, then HR department will approve the leave

**Command Design Pattern:** Command design pattern is useful when you want an object (command) to encapsulate or store all the required information to perform an action and this action can be performed at a later time upon requirement

Simple calculator operation

**Strategy Design Pattern:** Using strategy pattern, client can select their preferred algorithm from a family of algorithms to perform some operation.

Each algorithm will be encapsulated in different classes and can be accessed using a common interface.

Encryption algorithms are RAS Algorithm, DES Algorithm and Blowfish Algorithm

**Template Design Pattern:** Template pattern provides the skeleton to implement an algorithm and the steps to implement the algorithm can be overridden by implementing class i.e. sub-classes can redefine the steps without altering the original structure of the algorithm

Consider a video player which plays different format of videos like mp4, mkv and avi format. To support different formats of video respective video decoder is required.

To play a video first video file will be loaded. This step will be common for all video formats, so it will be default state.

Once the file will be loaded then it will be processed with suitable video decoder. This step will vary for different video formats.

Once the video will be decoded then our media player will starts playing the video. This state is also common for all video formats.

**Observer Design Pattern:**

Consider a scenario of eCommerce website, where user can subscribe to any product. If there will be any discount on the product then the eCommerce website will send a notification message to all of the subscribed customers.

**Mediator Design Pattern:** This real-world code demonstrates the Mediator pattern facilitating loosely coupled communication between different Participants registering with a Chatroom. The Chatroom is the central hub through which all communication takes place. At this point only one-to-one communication is implemented in the Chatroom, but would be trivial to change to one-to-many

**Memento Design Pattern:** Memento pattern allows you to store the internal state of an object and you can restore the object later using stored state.

Let’s consider the design of a game in which you want to store the state of a given object over different check points/ levels and later the player will be able to restore the object using the stored states

**State Pattern:** State pattern is useful when we want to maintain various states for a scenario in our program. State pattern allows an object to change its behavior when it’s internal state changes

Consider a video game where player fights the enemy soldiers. The lifeline of player depends on following condition:

If less than 5 bullets are hit to the player, then it is in healthy state.

If in between 5 and 10 bullets are hit to the player, then it is in hurt state.

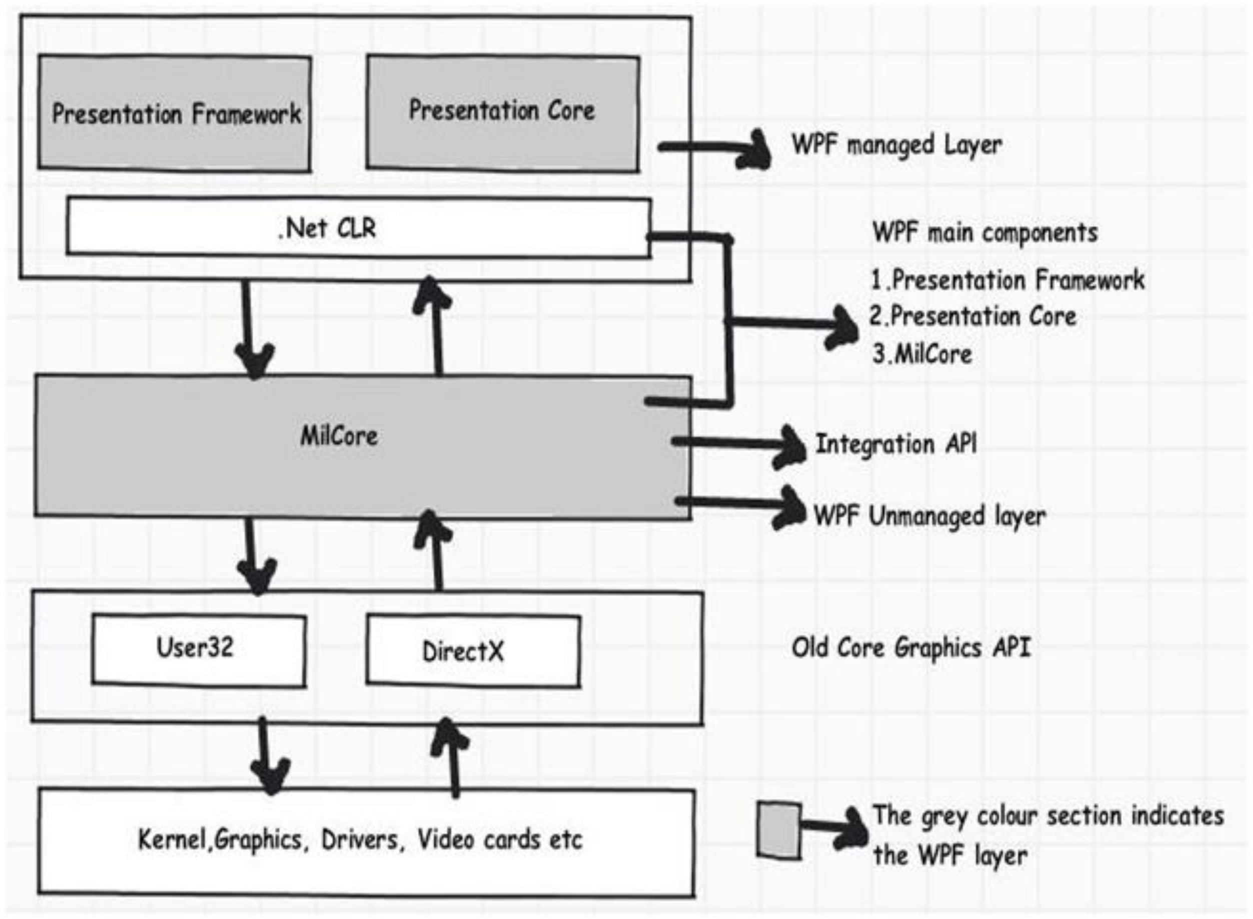
If greater than 10 bullets are hit to the player, then it is in dead state

# WPF

**Explain the difference between static and dynamic resource?**

Resources can be referred statically or dynamically. Static referred resources evaluate the resource only once and after that if the resources change those changes are not reflected in the binding. While dynamic referred resources are evaluated every time the resource is needed.

**What is WPF architecture?**



**User32:** It decides which goes where on the screen.

**DirectX:** As said previously WPF uses directX internally. DirectX talks with drivers and renders the content.

**Milcore:** Mil stands for media integration library. This section is a unmanaged code because it acts like a bridge between WPF managed and DirectX / User32 unmanaged API.

**Presentation core:** This is a low level API exposed by WPF providing features for 2D, 3D, geometry etc.

**Presentation framework:** This section has high level features like application controls, layouts, Content etc which helps you to build up your application.

**What is visual tree and logical tree?**

**The Logical Tree**

The logical tree describes the relations between elements of the user interface. The logical tree is responsible for:

Inherit DependencyProperty values

Resolving DynamicResources references

Looking up element names for bindings

Forwaring RoutedEvents

**The Visual Tree**

The visual tree contains all logical elements including all visual elements of the template of each element. The visual tree is responsible for:

Rendering visual elements

Propagate element opacity

Propagate Layout- and RenderTransforms

Propagate the IsEnabled property.

Do Hit-Testing

RelativeSource (FindAncestor)

**What is control template and data template?**

**Control Template** This template specifies the appearance of a Control; if a control does not have a Control Template

**Data Template** This template specifies a group of characteristics for how data should be displayed. This template is particularly useful when you are binding an ItemsControl such as a ListBox to an entire collection.

**What is dependancy object and dependancy property?**

Dependency object is the base object for all WPF objects. All the UI Elements like Buttons,TextBox etc and the Content Elements like Paragraph, Italic, Span etc all are derived from Dependency Object.

Name the methods present in the DependencyObject

SetValue

ClearValue

GetValue

**Advantages of a Dependency Property**

**Less memory consumption:** The Dependency Property stores the property only when it is altered or modified. Hence a huge amount of memory for fields are free.

**Property value inheritance:** It means that if no value is set for the property then it will return to the inheritance tree up to where it gets the value.

**Change notification and Data Bindings:** Whenever a property changes its value it provides notification in the Dependency Property using INotifyPropertyChange and also helps in data binding.

**Participation in animation, styles and templates:** A Dependency Property can animate, set styles using style setters and even provide templates for the control.

**CallBacks:** Whenever a property is changed you can have a callback invoked.

**Resources:** You can define a Resource for the definition of a Dependency Property in XAML.

**Overriding Metadata:** You can define certain behaviours of a Dependency Property using PropertyMetaData. Thus, overriding a metadata from a derived property will not require you to redefine or re-implement the entire property definition.

**ValidateCallback:** You need to put logic to validate the incoming data as Value argument. True makes it to take the value, false will throw the error.

**CoerceValue:** Can modify or change the value depending on the value passed as argument. It also receives DependencyObject as argument. You can invoke CoerceValueCallback using CoerceValue method associated with DependencyProperty.

**PropertyChanged:** This is the final Messagebox that you see, which will be called after the value is fully modified. You can get the OldValue and NewValue from the DependencyPropertyChangedEventArgs.

There are many inbuilt delegates like Action, Func and Predicate etc. with .NET framework

In C# language there are Normal Properties with Get and Set. We can understand the concept as Dependancy Property = Normal Property + Extra Wpf Property

## Why are dependency properties "static"?

The field you declare as static is only the identifier of a dependency property, not the value of the property. It is shared across all instances of the class, and is used to get/set the value of the property for each instance.

## Explain types of event

**Direct events:** In this case event is raised at the source and handled at the source itself like "MouseEnter" events.

**Bubbling events:** They travel up the visual tree hierarchy. For example "MouseDown" is a bubbling event.

**Tunneling events:** These events travel down the visual tree hierarchy. "PreviewKeyDown" is a tunneling event.

## Explain types of triggers

**Property Trigger:** This Trigger is activated when the UIElements property matches a specified value.

**Multiple Property Trigger:** This is similar to a Trigger but it combines multiple conditions to be met simultaneously. It will execute when all the conditions are satisfied within <MultiTrigger.Conditions>

**Data Trigger:** A Data Trigger is activated when the binding data matches the specified conditions. In the Data trigger we need to specify the binding instead of property name.

**MultiData Trigger:** This is similar to a Trigger but it combines multiple conditions to be met simultaneously. It will execute when all the conditions are satisfied within the <MultiDataTrigger.Conditions>.

**Event Trigger:** An Event Trigger is activated when a RouteEvent of a Framework Element is raised. It is usually used when the animation is to be called in response to an event like colorAnimation, doubleAnimation and many other animations as needed.

## Difference between custom control vs user control

When creating custom controls in Visual Studio, all controls are added to /Themes/Generic.xaml

| **Custom Control** | **User Control** |
|---|---|
| A loosely coupled control w.r.t code and UI | A tightly coupled control w.r.t code and UI |
| Derives from Control | Derives from UserControl |
| Defines UI in a ResourceDictionary | Defines UI as normal XAML |
| UI is skinable | Child controls are skinable |
| Has dynamic layout | has static layout |
| UI can be changed in different projects | UI is fixed and can't have different looks in different project |
| Has full toolbox support | can’t be added to the toolbox |
| Defines a single control | Defines a set of controls |
| More flexible | Not very flexible like a Custom Control |

## What is converter in Wpf

There are two types of Value Converters in WPF

**Value Converters**: A Value Converter is required when a target is bound with one source, For instance you have a text box and a button control. You want to enable or disable the button control when the text of text box is filled or null.

**MultiValue Coverters**: A MultiValue Coverter is required when your target is bound with multiple sources and the source and targets have different data formats or need some conversion.

For Example-1, let's say you have three text box controls and one button control and you want to enable the button control when all the text of the text boxes are filled.

## Advantages and disadvantages of WPF over winforms?

WPF has the ability to separate UI from logic effectively

WPF has an inbuilt storyboarding feature and animation models

Data binding is very much better than with the WinForms application

WPF offers data and control templates that provide flexible modelling of UI on data models

WPF supports 3D graphics to make UIs look really special

**Differene between xmlns vs xmlns:x**

Bothe namespaces helps to define / resolved XAML UI elements.

The first namespace is the default namespace and helps to resolve overall WPF elements.

The second namespace is prefixed by “x:” and helps to resolve XAML language definition.

**What is app.xaml in wpf**

App.xaml is first class loaded in WPF application

**What is stack panel, wrap panell, dock panel, canvas**

**StackPanel** By using StackPanel we can arrange the elements vertically or horizontally

**WrapPanel** In a Wrap Panel the child elements are positioned sequentially, from left to right and top to bottom until there is no more room, where it will wrap to the next line and then continue.

**DockPanel** The DockPanel allows you to dock the child controls to the top, bottom, left or right. By default, the last control, if not given a specific dock position, will fill the remaining space.

**Canvas** Canvas is used to place the control by specifying the position in pixel format. We can use Canvas.Left, Canvas.Right, Canvas.Bottom and Canvas.Top. Its child elements are positioned by explicit coordinates.

**Explain types of binding**

**Two way:** Data can flow from both source to target and from target to source.

**One way:** Data flows only from source to target.

**One way to source:** Data flows only from target to source.

**One time:** Data flows only for the first time from source to target and after that no communication happens.

## What is relative source in WPF?

There are four ways of binding relatively in WPF:

**Self** This relative binding helps to bind to one property of an element to the other property of the same element

<Border BorderBrush="Black" BorderThickness="1" Height="139" Width="{Binding Height, RelativeSource={RelativeSource Self}}"/>

**Ancestor** This relative binding helps to bind properties to the parent element properties

<TextBox Background="{Binding BorderBrush, RelativeSource={RelativeSource FindAncestor, AncestorLevel=1, AncestorType={x:Type Border}}}"/>

Previousdata

Templated parent

## What is full name XAML?

Extensible Application Markup Language

## What is elementname in WPF?

The name of the element to which the binding is to be made.

## Explain difference between data context vs itemsource in wpf

DataContext expects an object type where ItemsSource expects IEnumerable type objects

DataContext does not generate template, it only used to hold common data for other controls to bind. In terms of ItemsSource property, it is mainly used to generate template regardless of you set it in XAML or in the code behind.

DataContext is mainly used to hold common data that other child want to share. Thus it can be inherited by other child elements without problem. But for ItemsSource, it not used to share data in the visual tree. It is only valid for the element that defined. There is still one thing to be noted is that the child element can override the DataContext of the perent DataContext no mater directly or indirectly

**What is xbap in WPF?**

XBAP is stand for XAML Browser Application which is a new Windows technology used for creating Rich Internet Applications.

While windows applications are normally compiled to an .exe file, browser applications are compiled to an extension .xbap and can be run inside Internet Explorer

**How can I enumerate all the descendants of a visual object?**

You can use VisulTreeHelper class

**How do I update the source as I type in a TextBox?**

By setting the "UpdateSourceTrigger=PropertyChanged" in the textbox

**Is MDI Supported in WPF?**

MDI is not supported in WPF. UserControl can be used to give the same functionality as MDI.

**Explain types of XAML?**

A XAML file can be compiled into a .baml (Binary XAML) file, which may be inserted as a resource into a .NET Framework assembly.

At run-time, the framework engine extracts the .baml file from assembly resources, parses it, and creates a corresponding WPF visual tree or workflow.

**WPF XAML:** Encompasses the elements that describe WPF content, such as vector graphics, controls, and documents.

**XPS XAML:** It is the part of WPF XAML that defines an XML representation for formatted electronic documents.

**Silverlight XAML:** A subset of WPF XAML that's intended for Silverlight applications. Silverlight is a cross-platform browser plug-in that allows us to create rich web content.

**WF XAML:** Encompasses the elements that describe Windows Workflow Foundation.

**What are the types of windows in WPF?**

WPF has three types of windows:

Normal Window

Page Window

Navigate Window

**How to get Automation IDs of items in ItemsControl?**

AutomationProperties.AutomationID property

**What is dispatcher Object?**

It is the main of all WPF controls which Take care UI Thread

**What is freezable in WPF?**

A Freezable object  is a special type of object that has two states: unfrozen and frozen. When unfrozen, a Freezable appears to behave like any other object. When frozen, a Freezable can no longer be modified. So the objects dervied from Freezable based class will have only 2 states either Frozen or Unfrozen.

```
Button button = new Button();
SolidColorBrush brush = new SolidColorBrush(Colors.Blue);
if (brush.CanFreeze)
 {
    // Makes the brush unmodifiable.
    brush.Freeze();
 }
 button.Background = brush;
 if (brush.IsFrozen) // Evaluates to true.
 {
    // If the brush is frozen, create a clone and modify the clone.
    SolidColorBrushbrushClone = brush.Clone();
    brushClone.Color = Colors.Red;
    button.Background = brushClone;
 }
 else
 {
    brush.Color = Colors.Yellow; // Change the Property as object is not Frozen
 }
```

**What is Adorners in WPF?**

**What is documents in WPF?**

There are two kinds of document supported by WPF

**Flow format:** Flow format document adjusts as per screen size and resolution

**Fixed Format:** Fixed Format document does not adjust as per screen size and resolution

## What is Virtualization in WPF?

Virtualization technique in WPF improves the rendering performance of UI elements. By applying virtualization, the layout system ensures that only the visible items of a container are rendered on the screen. For example, a list control may have thousands of items but virtualization will reduce the rendering to the visible items only.

The VirtualizingStackPanel control in WPF is used to implement virtualization

```
<VirtualizingStackPanel Width="300" Height="200" />
```

## What is x:static, x:type, x:name, x:class, x:key in WPF?

**x:static** It is a way to insert any static value into XAML

```
namespace A
{
   public class MyConstants
   {
      public static readonly string SomeConstantString = "BAM!";
   }
}
```

I can place it into a WPF UI using XAML like this:

```
<TextBlock Text="{x:Static A:MyConstants.SomeConstantString}" />
```

**x:type**

**x:name** it is used to mention the names of stylesheets as well as the user interface controls

**x:class**

**x:key** it is used mainly with the WPF stylesheet or in ResourceDictionaries

## What is ObservableCollection in WPF?

An ObservableCollection is a dynamic collection of objects of a given type. Objects can be added, removed or be updated with an automatic notification of actions. When an object is added to or removed from an observable collection, the UI is automatically updated. This happens because, when binding to an observable collection, WPF automatically adds a CollectionChanged event handler to the ObservableCollecion's events.

The ObservableCollection class exists in the System.Collections.ObjectModel namespace

## How to implement WPF Performance tuning

**Improve binding** Try to use virtualization in a grid or list box. It makes the application faster because only the visible data is loaded instead of the entire data

You can use a VirtualizingStackPanel instead of a stack panel

**StreamGeometries** Use StreamGeometries instead of PathGeometries if possible to draw complex 2D geometries, because they are much more efficient and consume less memory

**Use freeze object**

**Increase static resource**

Dispatch expensive calls either within the UI thread with a lower DispatcherPriority by calling Dispatcher.BeginInvoke() or to a background thread by

using a BackgroundWorker to keep the UI responsive

## What is default binding mode in WPF?

If you don’t specify any binding mode then default binding will take place.

The default binding mode will choose binding mode depending upon target property. For example,

if you binding property with Label’s Content property then default binding will be OneWay similarly

if you are binding to TextBox’s Text property then default binding will be TwoWay. If you are not sure sometimes

it is best to specify binding mode explicitly

## Difference between ItemsControl vs ItemsPresenter vs ContentControl vs ContentPresenter

Content Presenter in WPF is used inside control templates, as well as inside the root application markup

ContentPresenter is used inside control templates to display content.

**Difference between DataTemplate vs HierarchicalDataTemplate**

**HierarchicalDataTemplates** are used by TreeViews

**What is templatebinding in WPF?**

**What is itemcontrol in WPF?**

**What is prism in WPF?**

PRISM is a framework to develop composite application in WPF and Silverlight. Composite applications are built using composition. In other words rather than building application from scratch we take prebuilt components, assemble them together and create the application.

Take the below example of simple WPF UI. You can see it has lots of sections. Now rather than building the whole UI as one big unit, we can develop all these section as independent unit. Later by using PRISM we can compose WPF UI by taking all these independent units.

**What is different between and purpose of MEF and Unity?**

The main difference is that with unity you will explicitly register each class you want to use in the composition:

```
var container = new UnityContainer();

container.RegisterType<IFoo,Foo>();

container.RegisterType<IBar,Bar>();

...

var program = container.Resolve<Program>();

program.Run();
```

In MEF on the other hand, you mark classes with attributes instead of registering them somewhere else:

```
[Export(typeof(IFoo))]

public Foo

{

   ...

}
```

**What is Full form of MEF?**

MEF full form is Managed Extensibility Framework

**What is dependency injection? Give some other .Net dependency example.**

Dependency injection can be done in three ways.

Constructor injection

Method injection

Property injection

**Dependency Injection Pros & Cons**

**Pros**

Loosely Coupled

Increase Testability

Separate component cleanly

Allow for use of inversion of control container

**Cons**

Increase code complexity

Some Jr.Developer find it difficult to understand at first

Can complicate debugging at First

Complicate following Code Flow

**Example**

Castle Windsor

StructureMap

Autofac

Unity

Ninject

**What is Event Aggregotor in WPF?**

Disadvantages of Event Aggregotor

If there are multiple Subscribers and Publishers then code become hard to read and debug

Its Subscriber responsibility to register and unregister from an event, its seen many practical scenarios the subscriber typically forgets to unregister causing both subscriber and publisher to be in memory causing memory leaks.

**What is ICommand Interface?**

Commands in wpf are basically loosely typed events

ICommand Interface has two methods Execute() and CanExecute()

CanExecute is a method which represents the condition in which your command will be fired, it is a boolean method.

Execute method represents the logic of our command.

# WCF

## Explain WCF architecture

## What is WCF?

Windows Communication Foundation (WCF) is a framework for building service-oriented applications. Using WCF, you can send data as asynchronous messages from one service endpoint to another. A service endpoint can be part of a continuously available service hosted by IIS, or it can be a service hosted in an application. An endpoint can be a client of a service that requests data from a service endpoint. The messages can be as simple as a single character or word sent as XML, or as complex as a stream of binary data.

## Difference between WCF and Web service

**Asp.Net Web Services**

Hosted only in IIS.

Supports only HTTP, HTTPS protocols.

WebService and WebMethod attributes are used for defining web service.

Support security but is less secure as compared to WCF.

Supports XML serializer by using System.Xml.Serialization.

Supports One-Way and Request-Response service operations.

Web Services are slower than WCF

Doesn’t support multi-threading.

**WCF**

Hosted in IIS, WAS (Windows Activation Service), Self-hosting, Windows Service.

Supports various protocols like HTTP, HTTPS, TCP, Named Pipes and MSMQ.

ServiceContract and OperationContract attributes are used for defining WCF service.

Supports security, reliable messaging, transaction and AJAX and REST supports.

Supports DataContract serializer by using System.Runtime.Serialization.

Supports One-Way, Request-Response and Duplex service operations.

WCF are faster than Web Services.

Supports multi-threading by using ServiceBehaviour class.

**What is ABC in WCF service?**

A WCF service endpoint has three basic elements i.e. Address, Binding and Contract.

**Address:** It defines "WHERE". Address is the URL that identifies the location of the service.

**Binding:** It defines "HOW". Binding defines how the service can be accessed.

**Contract:** It defines "WHAT". Contract identifies what is exposed by the service.

**What Message Exchange Patterns (MEPs) supported by WCF? Explain each of them briefly.**

**Request -Reply:**

This is default Message Exchange Pattern

Client send a message to WCF service and then waits for a reply. During this time the client stops processing until response is received from the WCF service.

The client waits for the service call to complete even if the operation return type is void.

All WCF bindings except the MSMQ based bindings support the request-reply message exchange pattern.

In a Request Reply message exchange pattern faults and exceptions get reported to the client immediately if any.

**One Way:**

Only one message exchange between client and service.

Client makes a call to the service method, but does not wait for a response message.

So in short, the receiver of the message does not send a reply message, nor does the sender of the message excepts one.

To make an operation one way set is only = true

[OperationContract(IsOneWay=true)]

void onewayoperation();

As messages exchanged only in one way, faults does not get reported.

Clients are unaware of the server channel faults until a subsequent call is made.

One way calls not same as asynchronous call.

**Duplex:**

Duplex is a Two-Way Communication Process.

Duplex service allows calling back some operation (function) on the client.

Duplex service also knows as Call Backs.

All Binding does not support duplex service.

Duplex communication is not standard. They are absolute Microsoft feature.

wsDualHttpBinding supports duplex communication over HTTP binding.

Duplex communication is possible over netTcpBinding and netNamedPipeBinding

## Contract in WCF?

**Service contract:** This attribute is used to define the Interface. A service contract defines the operations which are exposed by the service to the outside world

**Operation contract:** This attribute used to define the method inside Interface.

An Operation Contract defines the method exposed to the client to exchange the information between the client and server

**Data Contract:** Data Contract in WCF is used to serialize and deserialize the complex data Using DataMemberAttribute you can a) define name, order, and whether if a property or field is required

b) Also, serialize private field and properties

**Fault contract:** A fault contract defines errors raised by the service, and how the service handles and propagates errors to its clients

**Message Contract:** Message Contract is used to control the structure of a message body and serialization process.

It is also used to send / access information in SOAP headers

A message contract defines the elements of the message (like as Message Header, Message Body), as well as the message-related settings, such as the level of message security.

Message contracts give you complete control over the content of the SOAP header, as well as the structure of the SOAP body

Few Example of when Message contracts can be handy

1) Include some custom data in the SOAP header. In general SOAP headers are used to pass user credentials, license keys, session keys etc.

2) Change the name of the wrapper element in the SOAP message or to remove it altogether

## Exception Handling in WCF?

Using **returnUnknownExceptionsAsFaults:** Debugging Mode

Using **FaultException:** Best Option.

Using **IErrorHandler:** Only when the exception can't be handled by Fault

## Instance Mode in WCF?

**PerCall:** A New instance of the service object is created for every request, irrespective of whether the request comes from the same client or difference client.

Better memory usage as service objects are freed immediately after the method call returns.

State is not maintained between calls.

[ServiceBehavior(InstanceContextMode = InstanceContextMode.Percall)]

**PerSession:** A New Instance of the service object is created for each new client session and maintained for duration of that session.

State maintained between calls.

Greater memory consumption as service objects remain in memory until the client session times out.

[ServiceBehavior(InstanceContextMode = InstanceContextMode.PerSession)]

**Single:** A Single instance of the service object is created and handles all requests for the lifetime of the application, irrespective of whether the request comes from the same client or different client.

[ServiceBehavior(InstanceContextMode = InstanceContextMode.Single)]

## What are the possible ways of hosting a WCF service?

Self-Hosting

Windows Service

Internet Information Services (IIS)

Windows Activation Services (WAS)

**What is SOA?**

SOA stands for service oriented architecture. Service Oriented Architecture is an architectural approach in software development where the application is organized as "Services".

Services are a group of methods that contain the business logic to connect a DB or other services.

SOA services should be independent of other services

**ASP.Net Web API vs WCF, which one should I choose in my project?**

We cannot decide which is better than the other. But I want to focus on the two points. Firstly, if you are going to create a service which would be used on different platforms, then go with WCF. Secondly, if you are creating internet service which is going to use external resource, then go with Web API.

Choose WCF when you want to create a service that should support special scenarios such as one way messaging, message queues, duplex communication etc.

Choose WCF when you want to create a service that can use fast transport channels when available, such as TCP, Named Pipes, or maybe even UDP (in WCF 4.5), and you also want to support HTTP when all other transport channels are unavailable.

Choose Web API when you want to create a resource-oriented services over HTTP that can use the full features of HTTP (like URIs, request/response headers, caching, versioning, various content formats).

Choose Web API when you want to expose your service to a broad range of clients including browsers, mobiles, iPhone and tablets.

# SQL SERVER

## Constraint in SQL server

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.

The following constraints are commonly used in SQL:

**NOT NULL** - Ensures that a column cannot have a NULL value

**UNIQUE** - Ensures that all values in a column are different

**PRIMARY KEY** - A combination of a NOT NULL and UNIQUE. Uniquely identifies each row in a table

**FOREIGN KEY** - Uniquely identifies a row/record in another table

**CHECK** - Ensures that all values in a column satisfies a specific condition

**DEFAULT** - Sets a default value for a column when no value is specified

**INDEX** - Used to create and retrieve data from the database very quickly

## How to get last generated column

SCOPE_IDENTITY()

## scope_identity vs @@identity vs ident_current

**@@IDENTITY** it returns the last identity value generated for any table in the current session, across all scopes. Let me explain this... suppose we create an insert trigger on table which inserts a row in another table with generate an identity column, then @@IDENTITY returns that identity record which is created by trigger.

**SCOPE_IDENTITY** It returns the last identity value generated for any table in the current session and the current scope.

**IDENT_CURRENT** It returns the last identity value generated for a specific table in any session and any scope. In other words, we can say it is not affected by scope and session, it only depends on a particular table and returns that table related identity value which is generated in any session or scope.

## Difference between primary key vs unique key

**Primary Key**

Primary key cannot have a NULL value.

Each table can have only one primary key.

By default, Primary key is clustered index, and the data in database table is physically organized in the sequence of clustered index.

Primary key can be related to another tables as a Foreign Key.

We can generate ID automatically with the help of Auto Increment field. Primary key supports Auto Increment value.

We can define Primary key constraint on temporary table and table variable.

We can't delete primary key value from the parent table which is used as a foreign key in child table. To delete we first need to delete that primary key value from the child table.

**Unique Key**

Unique Constraint allows one NULL value.

Each table can have more than one Unique Constraint.

By default, unique key is a unique non-clustered index.

Unique Constraint cannot be related with another table's as a Foreign Key.

**What is difference between CHAR, NCHAR, VARCHAR and NVARCHAR data types?**

**N** stands for National Language Character Set and is used to specify a Unicode string (for example the languages Arabic, German and so on). When using Unicode data types, a column can store any character defined by the Unicode Standard, which includes all of the characters defined in the various character sets.

If your column will store a fixed-length Unicode characters like French, Arabic and so on characters then go for NCHAR. If the data stored in a column is Unicode and can vary in length, then go for NVARCHAR.

CHAR is a fixed length data type

VARCHAR is a variable length data type

**Note:** In VARCHAR, if you put a greater value than 255 it will be converted to a TEXT type

## What are the use cases for selecting CHAR over VARCHAR in SQL?

Generally pick **CHAR** if all rows will have close to the same length. Pick **VARCHAR** when the length varies significantly. CHAR may also be a bit faster because all the rows are of the same length.

## What is difference between TEXT and TINYTEXT data types?

**TINYTEXT:** Holds a string with a maximum length of 255 characters

**TEXT:** Holds a string with a maximum length of 65,535 characters

## Types of join in SQL Server

**Inner Join:** The join that displays only the rows that have a match in both the joined tables is known as inner join

**Left Outer Join:** Left join displays all the rows from first table and matched rows from second table like that

**Right Outer Join:** Right outer join displays all the rows of second table and matched rows from first table like that.

**Full Outer Join:** Full outer join returns all the rows from both tables whether it has been matched or not.

**Cross Join:** A cross join that produces Cartesian product of the tables that are involved in the join. The size of a Cartesian product is the number of the rows in the first table multiplied by the number of rows in the second table like this.

**Self-Join:** Joining the table itself called self-join. Self-join is used to retrieve the records having some relation or similarity with other records in the same table.

## COALESCE() Functions

COALESCE function returns the first non-null expression in the list. If all expressions evaluate to null, then the COALESCE function will return null.

NULLIF(*expr1*, *expr2*)

## NULLIF function

The NULLIF() function returns NULL if two expressions are equal, otherwise it returns the first expression.

## IN vs EXISTS

**IN:** Returns true if a specified value matches any value in a subquery or a list.

**EXISTS:** Returns true if a subquery contains any rows.

## Difference between ROLLUP, CUBE and GROUPING SET in SQL?

**ROLLUP:** The ROLLUP operator is used with the GROUP BY clause. It is used to create subtotals and grand totals for a set of columns.

**CUBE:** The CUBE operators, like the ROLLUP operator produces subtotals and grand totals as well. But unlike the ROLLUP operator it produces subtotals and grand totals for every combination of the columns provided to the CUBE operator.

**GROUPING SET:** GROUPING SETS operator allows you to group your data a number of different ways in a single SELECT statement.

**Note:** You won't see any difference when you use ROLLUP and CUBE on a single column.

## What is CHECK Constraint in SQL?

Allows you to specify a condition on each row in a table.

## Is it possible to use CHECK Constraint in VIEW?

No.

## Difference between store procedure vs function

Function must return a value but in Stored Procedure it is optional (Procedure can return zero or n values).

Functions can have only input parameters for it whereas Procedures can have input/output parameters.

Functions can be called from Procedure whereas Procedures cannot be called from Function

Exception can be handled by try-catch block in a Procedure whereas try-catch block cannot be used in a Function.

We can go for Transaction Management in Procedure whereas we can't go in Function

Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section whereas Function can be

Procedure allows SELECT as well as DML (INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it.

User-defined functions cannot be used to perform actions (INSERT/UPDATE/DELETE) that modify the database state

**User defined function is not "Pre-Compiled". Is that true?**

As far as I know Stored Procedures & Functions both have same behavior in terms of compilation & recompilation.

When you run a stored procedure for the first time it will go through all the above steps and the plan will be cached in-memory. So the next time when the stored procedure is executed it just takes the plan from the cache and executes the same.

Both are compiled when they are executed for the first time. And they could be re-compiled automatically again if there is any change applied to them.

**Difference between Inline query and store procedure?**

Store procedure are pre-compiled

Store procedure prevents SQL Injection Errors.

Stored procedures reduces network traffic. Stored procedures are stored in the server, only the name of stored procedure is required to pass to the server. But in the case of inline queries, the complete query has to be passed to the server. So inline queries will increase network traffic when the queries are very large.

By using stored procedures we can separate all the queries from the Business logic code. Therefore we can create a separate layer.
But while writing inline queries, all the queries have to be written (mixed up) with the business logic code. This create problem while debugging.

**Are execution plan for functions cached in SQL server?**

Yes they do go in the execution plan cache.

**Difference between store procedure output parameter vs return values parameter**

**Output parameter in store procedure:** Any data type, more than value

**Return value in store procedure:** Only Integer data type, only one value

**Advantages of store procedure**

Execution plan retention

Reduce network traffic

Better Security

Avoid SQL Injection attack

## What is GUID in SQL server?

A GUID is a 16 byte binary data type that is globally unique. GUID stands for Global Unique Identifier.

## Deterministic and Nondeterministic

Deterministic function always return same result any time they are called with a specific set of input value and same state of database

Square(), Power(), Sum(), AVG(), Count()

NonDeterministic function may return different result each time they called with a specific set of input value even if database state they access remains the same.

GetDate() and CURRENT_TIMESTAMP

## Types of function

**Scalar function:** scalar function return single value. The returned value can be of any data type, except text, ntext, image, cursor, timestamp

**Inline Table value function:**

We specify Table as return type, instead of any scalar data type

The function body is not enclosed between BEGIN and END block

The structure of the table that get returned, is determined by the SELECT statement with in the function.

**Multi-Statement Table Value Function:**

We specify structure of the table that gets returned

Multi-statement function have BEGIN and END block

## Types of temporary tables

Local temporary table is automatically dropped when connection that has created the it, is closed.

Global temporary table are visible to all the connection of the SQL server, and are only destroyed when the last connection referencing the table is closed.

Global Temporary Table, prefix the name of the table with 2 pound (##) symbols.

## Triggers

A trigger is a special kind of Stored Procedure or stored program that is automatically fired or executed when some event (insert, delete and update) occurs

## Common type expression

A CTE is basically a disposable view. It only persists for a single statement, and then automatically disappears

You can use commas to create multiple CTEs that references the CTEs Above.

You need to put the CTE first and then combine the INSERT INTO with your select statement. Also, the "AS" keyword following the CTE's name is not optional:

```
WITH tab AS (

   bla bla

)

INSERT INTO dbo.prf_BatchItemAdditionalAPartyNos (

BatchID,

AccountNo,

APartyNo,

SourceRowID

)

SELECT * FROM tab
```

## CTE vs Temp table vs Table variable

**Temp Table:** Gets created physically in the tempdb

Available only to the particular session

Primary key, indexes, constraints etc can be created

Table can be altered after creation

DDL operation is allowed

Recursion is not possible

Cannot be used in a view

**Table Variable:** Gets created physically in the tempdb but acts like a variable

Available only to the particular batch in the session

Clustered index or Non-clustered index can be created with the primary key during the time of declaration

Table can NOT be altered after creation

DDL operation is NOT allowed

Recursion is not possible

Cannot be used in a view

**CTE:** Gets created in the memory. In case of low memory it can spill to tempdb also.

Available only to the particular scope in the session

Not Applicable

CTE cannot be modified at all

DDL operation is NOT allowed

Recursion is possible

Can be used in a view

**Can we use update statement with CTE?**

Yes, if a CTE is based on more than one table, and if the Update affects only one base table, then the UPDATE is allowed.

**Normalization - 1, 2 & 3 normal form**

Data normalization is the process of organizing data to minimize data redundancy (data duplication), which in turn ensures data consistency

**First Normal Form:** The data in each column should be atomic. No multiple value, separated by comma, use primary key

**Second Normal Form:** Move redundant data to separate table, use foreign keys

**Third Normal Form:** Does not contain columns that are not fully dependent upon the primary key

## Pivot

Unique values from one column, into multiple columns in the output, there by effectively rotating a table

## Transaction

Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group successfully complete. If any of the tasks fails, the transaction fails. Therefore, a transaction has only two results: success or failure. Incomplete steps result in the failure of the transaction.

Users can group two or more Transact-SQL statements into a single transaction using the following statements:

- Begin Transaction
- Rollback Transaction
- Commit Transaction

If anything goes wrong with any of the grouped statements, all changes need to be aborted. The process of reversing changes is called **rollback** in SQL Server terminology. If everything is in order with all statements within a single transaction, all changes are recorded together in the database. In SQL Server terminology, we say that these changes are **committed** to the database.

## Sub query

A subquery is a SQL query within a query.

Subqueries are nested queries that provide data to the enclosing query.

Subqueries can return individual values or a list of records

Subqueries must be enclosed with parenthesis

## Corelated sub query

Correlated Subquery is a sub-query that uses values from the outer query

## What to choose for performance sub query vs join

Depends upon logic

## Cursors

If there is need to process the rows, on a row by row basis, then cursors are your choice. Cursors are very bad for performance.

## SQL server optimize technique

**Avoid correlated subqueries:** This kind of query tends to run row-by-row, once for each row returned by the outer query, and thus decreases SQL query performance

**Select sparingly:** avoid SELECT * Instead, you should individually include the specific columns that you need

**Does my record exist:** This SQL optimization technique concerns the use of EXISTS(). If you want to check if a record exists, use EXISTS() instead of COUNT()

**Execution Plan Re-use:** The best way to re-use the execution plan is by implementing parameterized stored procedures

Use set nocount on at the top of each stored procedure (and set nocount off) at the bottom

Avoid NOT IN, instead use a left outer join – even though it's often easier to visualize the NOT IN

Avoid temp tables as much as you can, but if you need a temp table, create it explicitly using Create Table #temp

## What is SET NOCOUNT ON and SET NOCOUNT OFF?

SET NOCOUNT ON- It will show "Command(s) completed successfully"

SET NOCOUNT OFF- it will show "(No. Of row(s) affected)"

## View, pros & cons of view

**Pros:** View can be used to reduce the complexity of the database schema

Views can be used as a mechanism to implement row and column level security

Views can be used to present aggregated data and hide detailed data

**Cons:** You cannot pass parameter to a view. Table Valued functions are an excellent replacement for parameterized views.

The Order by clause is invalid in views unless TOP or FOR XML is also specified.

Views cannot be based on temporary tables.

Create View vwOnTempTable

As

Select Id, Name, Gender

From ##TestTempTable

**Where are views stored in SQL Server?**

Physically stored

**Can a View based on another View?**

Yes, A View is based on another View.

**Can we Insert, Update and delete a view?**

View can be update/insert/delete if it contain fields of one table.

If View is based on multiple tables, and if you update the view, it may not update the underlying base tables correctly.

**Can we create primary key on a view?**

Yes, we can create cluster index

**Composite primary key**

Composite key, or composite primary key, refers to cases where more than one column is used to specify the primary key of a table.

In such cases, all foreign keys will also need to include all the columns in the composite key.

Note that the columns that make up a composite key can be of different data types.

## Where, having & group by

FROM & JOINs determine & filter rows

WHERE more filters on the rows

GROUP BY combines those rows into groups

HAVING filters groups

ORDER BY arranges the remaining rows/groups

LIMIT filters on the remaining rows/groups

## Type of indexes, pros and cons of indexes

Indexes are used by queries to find tables quickly.

**Disadvantages of indexes**

**Additional Disk spaces**: No cluster index required additional space as it is stored separately from table

Insert, Update and delete statements can become slow: When DML statements modifies data in table, the data in all the indexes also needs to be updated.

## Rank, dense rank, row number

**Row_Number():** This function will assign a unique id to each row returned from the query

**Rank():** This function will assign a unique number to each distinct row, but it leaves a gap between the groups

**Dense_Rank():** This function is similar to Rank with only difference, this will not leave gaps between groups

## Truncate vs delete vs drop

**DELETE:** The DELETE command is used to remove rows from a table. A WHERE clause can be used to only remove some rows.

If no WHERE condition is specified, all rows will be removed.

After performing a DELETE operation you need to COMMIT or ROLLBACK the transaction to make the change permanent or to undo it

**TRUNCATE:** TRUNCATE removes all rows from a table. The operation cannot be rolled back and no triggers will be fired.

As such, TRUCATE is faster and doesn't use as much undo space as a DELETE

IDENTITY columns are re-seeded on this operation, if no seed was defined then the default value 1 is used.

TRUNCATE is not possible when a table:

a. is reference by a Foreign Key or tables used in replication or with Indexed views.

b. participates in an Indexed/Materialized View.

c. published by using Transactional/Merge replication.

**DROP:** The DROP command removes a table from the database. All the tables' rows, indexes and privileges will also be removed. No DML triggers will be fired.

The operation cannot be rolled back.

**udt in SQL server**

Custom Data Type

**ALL, ANY and SOME in SQL server**

**ALL:** The ALL operator returns TRUE if all of the subqueries values meet the condition, ALL is used with SELECT, WHERE, HAVING statement.

**ANY:** ANY return true if any of the subqueries values meet the condition.

**DDL, DML, DCL, TCL**

**DDL:** DDL is short name of Data Definition Language, which deals with database schemes and descriptions, of how the data should reside in the database

CREATE, ALTER, DROP, TRUNCATE

**DML:** DML is short name of Data Manipulation Language which deals with data manipulation, and includes most common SQL statements such SELECT, INSERT, UPDATE, DELETE etc and it is used to store, modify, delete, retrieve and update data in database.

SELECT, INSERT, UPDATE, DELETE, MERGE

**DCL:** DCL is short name of data control languages which includes commands such as GRANT and mostly concerned with rights, permissions and other controls of the database system.

GRANT, REVOKE

**TCL:** TCL is short name of Transaction Control Language which deals with transaction within a database.

COMMIT, ROLLBACK, SAVEPOINT, SET TRANSACTION

## Insert/Update/Delete with function in SQL Server

User-defined functions cannot be used to perform actions that modify the database state

## Cast vs convert in SQL server

CAST and CONVERT are both used to convert data from one data type to another

Thought their syntax is different, both functions are able to convert values from one formation to another.

Anything you can do with CAST you can do with CONVERT

CAST is part of the ANSI-SQL specification; whereas, CONVERT is not. In fact, CONVERT is SQL implementation specific

CONVERT is more flexible in that you can format dates etc

## Union vs union all in SQL server

**UNION:** UNION combines the result set of two or more queries into a single result set. This result set includes all the rows that belong to all queries in the UNION

**UNION ALL:** UNION ALL is very similar to UNION. It also includes duplicate rows in the result set.

## Intersection vs except in SQL server

**INTERSECT:** gives you the final result set where values in both of the tables match

**EXCEPT:** gives you the final result set where data exists in the first dataset and not in the second dataset

## Cross apply vs outer apply

The CROSS APPLY operator is semantically similar to INNER JOIN operator. It retrieves those records from the table valued function and the table being joined, where it finds matching rows between the two.

On the other hand, OUTER APPLY retrieves all the records from both the table valued function and the table, irrespective of the match.

**Sequence vs identity in SQL server**

**SEQUENCE:** Sequence object is introduced in SQL Server 2012

Sequence object can be shared across multiple tables

Sequence object can be used to generate database-wide sequential number across multiple tables.

A sequence is created independently of the tables by using the CREATE SEQUENCE statement

Sequence is a user-defined database object and as name suggests it generates sequence of numeric values according to the properties with which it is created

Identity Column property is introduced in Sql Server 6.0

Identity property is a table column property. It is also used to generate a sequence of numbers according to the properties with which it is created

Identity property is tied to a Table

Identity property can be used to generate a sequence numbers at a table level

Identity property can be specified for a table column in CREATE TABLE or ALTER TABLE statement

**Insert, Update, Delete on View & CTE**

Not Possible

**Store procedure with select statement**

A procedure can return multiple result sets, each with its own schema. It's not suitable for using in a SELECT statement.

**Is the NOLOCK (SQL Server hint) bad practice?**

With NOLOCK SELECT statement is READ UNCOMMITTED. This means that the query may see dirty and inconsistent data. This is not a good idea to apply as a rule.

**Set Order of this: select, group by, order by, from, where**

Select

From

Where

Group by

Having

Order by

**What are the different authentication modes in SQL Server?**

SQL and windows

**What is an Execution Plan?**

An execution plan is basically a road map that graphically or textually shows the data retrieval methods chosen by the SQL server's query optimizer for a stored procedure or ad hoc query. Execution plans are very useful for helping a developer understand and analyze the performance characteristics of a query or stored procedure, since the plan is used to execute the query or stored procedure.

What is system table in SQL Server?

SQL Server maintains a set of tables that contain information about all the objects, data types, constraints, configuration options, and resources available to the SQL Server.

**How do you copy data from one table to another table?**

```
INSERT INTO table2 (column1, column2, column3, ...)
SELECT column1, column2, column3, ...
FROM table1
WHERE condition;
```

**How to find a Duplicate record?**

```
SELECT name, COUNT(email)
 FROM users
 GROUP BY email
 HAVING COUNT(email) > 1
```

**How do you get the last id without the max fuction?**

select top 1 id from table order by id desc

**Remove duplicate records from a table in SQL Server**

```
WITH TempEmp (Name,duplicateRecCount)

AS

(

SELECT Name,ROW_NUMBER() OVER(PARTITION by Name, Salary ORDER BY
Name)

AS duplicateRecCount

FROM dbo.Employee

)

--Now Delete Duplicate Records

DELETE FROM TempEmp

WHERE duplicateRecCount > 1
```

**How to get second-highest salary employees in a table**

```
;WITH T AS
(
SELECT *,
     DENSE_RANK() OVER (ORDER BY Salary Desc) AS Rnk
FROM Employees
)
SELECT Name
FROM T

WHERE Rnk=2;
```

**Replace null value with previous available value in Row**

```
DECLARE @Table TABLE(
      ID INT,
      Val INT
)

SELECT  *,
       ISNULL(Val, (SELECT TOP 1 Val FROM @Table WHERE ID < t.ID AND Val IS NOT NULL ORDER
BY ID DESC))

FROM    @Table t
```

**How do we get maximum value from table without using Max,Min or Order by Clause in SQL Server**

```
SELECT *

FROM salary s1

WHERE s1.amount > ALL (SELECT s2.amount

FROM salary s2

WHERE s2.id <> s1.id);
```

# MVC

## What are the benefits of using MVC?

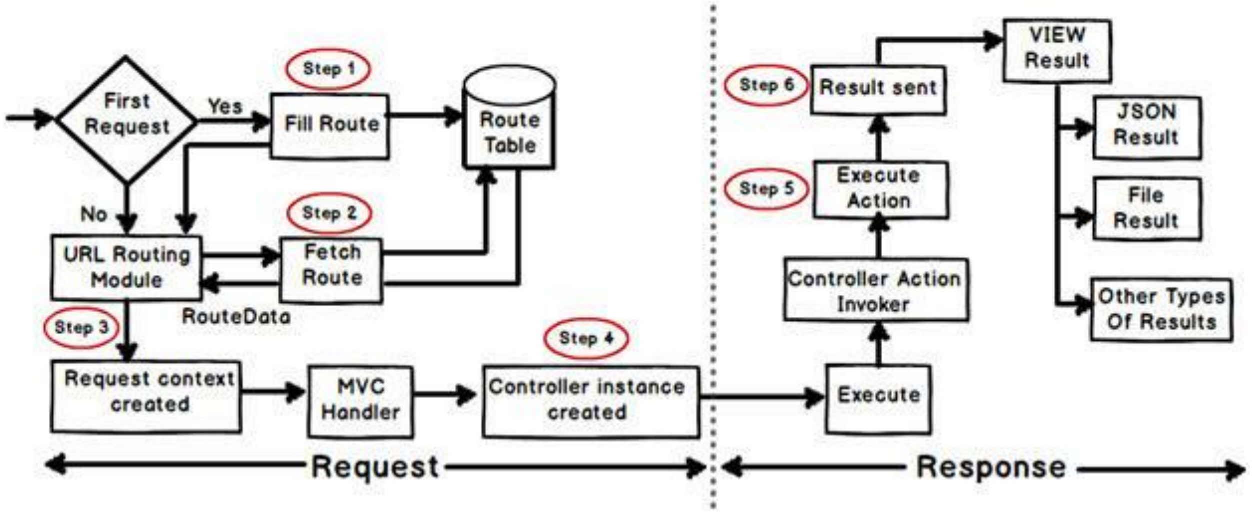
There are two big benefits of MVC:

Separation of concerns is achieved as we are moving the code-behind to a separate class file. By moving the binding code to a separate class file we can reuse the code to a great extent.

Automated UI testing is possible because now the behind code (UI interaction code) has moved to a simple .NET class. This gives us opportunity to write unit tests and automate manual testing.

## Explain MVC application life cycle?

MVC application life cycle is not different it has two main phases first creating the request object and second sending our response to the browser.



**Creating the request object:** The request object creation has four major steps. Below is the detail explanation of the same.

**Fill route:** MVC requests are mapped to route tables which in turn specify which controller and action to be invoked. So if the request is the first request the first thing is to fill the route table with routes collection. This filling of route table happens in the global.asax file.

**Fetch route:** Depending on the URL sent "UrlRoutingModule" searches the route table to create "RouteData" object which has the details of which controller and action to invoke.

**Request context created:** The "RouteData" object is used to create the "RequestContext" object.

**Controller instance created:** This request object is sent to “MvcHandler” instance to create the controller class instance. Once the controller class object is created it calls the “Execute” method of the controller class.

**Creating Response object:** This phase has two steps executing the action and finally sending the response as a result to the view

## What are html helpers in MVC?

HTML helpers help you to render HTML controls in the view. For instance if you want to display a HTML textbox on the view, below is the HTML helper code.

Hide   Copy Code

```
<%= Html.TextBox("LastName") %>
```

## What is the difference between html.TextBox and html.TextBoxFor?

Both of them provide the same HTML output, “HTML.TextBoxFor” is strongly typed while “HTML.TextBox” isn’t.

## What is routing in MVC?

Routing helps you to define a URL structure and map the URL with the controller.

```
routes.MapRoute(
            "View", // Route name
            "View/ViewCustomer/{id}", // URL with parameters
            new { controller = "Customer", action = "DisplayCustomer",
id = UrlParameter.Optional }); // Parameter defaults
```

The route mapping code is written in "RouteConfig.cs" file and registered using "global.asax" application start event.

## What is difference between TempData and ViewData?

ViewBag is a dynamic object, so we can add strongly typed objects, primitive values, etc that we need.
ViewData is a ViewDataDictionary

**How can we maintain sessions in MVC?**

Temp data - Helps to maintain data when you move from one controller to another controller or from one action to another action.
In other words when you redirect, tempdata helps to maintain data between those redirects. It internally uses session variables.

View data - Helps to maintain data when you move from controller to view.

View Bag - It's a dynamic wrapper around view data. When you use Viewbag type, casting is not required. It uses the dynamic keyword internally.

Session variables - By using session variables we can maintain data from any entity to any entity.

Hidden fields and HTML controls - Helps to maintain data from UI to controller only. So you can send data from HTML controls or hidden fields to the controller using POST or GET HTTP methods.

| **Maintains data between** | **ViewData/ViewBag** | **TempData** | **Hidden fields** | **Session** |
|---|---|---|---|---|
| **Controller to Controller** | No | Yes | No | Yes |
| **Controller to View** | Yes | No | No | Yes |
| **View to Controller** | No | No | Yes | Yes |

**What are partial views in MVC?**

Partial view is a reusable view (like a user control) which can be embedded inside other view. For example let's say all your pages of your site have a standard structure with left menu, header, and footer. For every page you would like to reuse the left menu, header, and footer controls. So you can go and create partial views for each of these items and then you call that partial view in the main view.

**Explain Get and Post method in MVC?**

GET is used to request data from a specified resource.

GET requests can be cached

GET requests remain in the browser history

GET requests can be bookmarked

GET requests should never be used when dealing with sensitive data

GET requests have length restrictions

**Note** query string (name/value pairs) is sent in the URL of a GET request

/test/demo_form.php?name1=value1&name2=value2

POST requests are never cached

POST requests do not remain in the browser history

POST requests cannot be bookmarked

POST requests have no restrictions on data length

**How can we do validations in MVC?**

One of the easiest ways of doing validation in MVC is by using data annotations. Data annotations are nothing but attributes which can be applied on model properties.

**Why Razor when we already have ASPX?**

It's a light weight view engine. Till MVC we had only one view type, i.e., ASPX. Razor was introduced in MVC 3. Razor is clean, lightweight, and syntaxes are easy as compared to ASPX.

**What is the difference between ActionResult and ViewResult?**

ActionResult is an abstract class while ViewResult derives from the ActionResult class. ActionResult has several derived classes like ViewResult, JsonResult, FileStreamResult, and so on.

**What are the different types of results in MVC?**

There 12 kinds of results in MVC, at the top is the ActionResult class which is a base class that can have 11 subtypes as listed below:

**ViewResult:** Renders a specified view to the response stream

**PartialViewResult**: Renders a specified partial view to the response stream

**EmptyResult**: An empty response is returned

**RedirectResult:** Performs an HTTP redirection to a specified URL

**RedirectToRouteResult**: Performs an HTTP redirection to a URL that is determined by the routing engine, based on given route data

**JsonResult**: Serializes a given ViewData object to JSON format

**JavaScriptResult**: Returns a piece of JavaScript code that can be executed on the client

**ContentResult**: Writes content to the response stream without requiring a view

**FileContentResult**: Returns a file to the client

**FileStreamResult**: Returns a file to the client, which is provided by a Stream

**FilePathResult**: Returns a file to the client

**What are ActionFilters in MVC?**

ActionFilters help you to perform logic while an MVC action is executing or after an MVC action has executed.

Action filters are useful in the following scenarios:

1. Implement post-processing logic before the action happens.
2. Cancel a current execution.
3. Inspect the returned value.
4. Provide extra data to the action.

You can create action filters by two ways:

- Inline action filter.
- Creating an ActionFilter attribute.

To create an inline action attribute we need to implement the IActionFilter interface. The IActionFilter interface has two methods: OnActionExecuted and OnActionExecuting. We can implement pre-processing logic or cancellation logic in these methods. The problem with the inline action attribute is that it cannot be reused across controllers.

**What is bundling and minification in mvc?**

Bundling and minification helps us improve request load times of a page thus increasing performance. Web projects always need CSS and script files. Bundling helps us combine multiple JavaScript and CSS files in to a single entity thus minimizing multiple requests in to a single request.

Minification reduces the size of script and CSS files by removing blank spaces, comments etc.

**How do we implement minification?**

When you implement bundling, minification is implemented by itself. In other words the steps to implement bundling and minification are the same.

## Explain Areas in MVC?

Areas help you to group functionalities in to independent modules thus making your project more organized.

## Explain difference between MVC and Asp.Net?

**MVC**

It supports the Test Driven Programming model

There is no event in MVC.

It uses the Front Controller pattern that process all requests through a single controller.

MVC does not support view state.

MVC has no specific page life cycle. All individual components (such as Model, View and Controller) have their own life cycle.

Code behind is not tightly coupled with the View (User Interface).

MVC also supports existing markup such as ASP.Net page (.aspx), user control (.ascx) and master page (.master) as view templates.

MVC also supports existing ASP.NET features such as authentication (form and windows), authorization, membership and roles, caching, session state and profile state management.

**Asp.Net**

Provides many events and also is supported in many server controls.

It supports view state on server based forms that can help us to manage state information.

Code behind is tightly coupled with the View (User Interface).

## What is the different between updatemodel vs tryupdatemodel?

UpdateModel() throws an exception, if validation fails, whereas TryUpdateModel() will never throw an exception.

## What is the different between html.partial vs html.renderpartial?

Html.Partial returns a (MvcHtmlString)String. Html.RenderPartial calls write internally and returns void

## Explain State Management Techniques in MVC?

**Client Side:** View Data, View Bag, Temp Data, Hidden Field, Cookies, Query String

Advantage:

The major advantages of having this kind of state management is that it saves a lot of server memory. We relieve server from the burden of keeping the state related information.

Disadvantage:

It takes more bandwidth as considerable amount of data is traveling back and forth. Due to this, web page becomes slow to load.

The main drawback is it creates security issue for sensitive information like passwords, credit card numbers etc.

**Server Side:** Cache, Profile Properties, Session State.

Advantage:

The major advantages of having this kind of state management is that it secure user's confidential and sensitive information.

Disadvantage:

The downside of this is it usage more server memory.

## How can we navigate from one view to another using a hyperlink?

ActionLink method

## How can we restrict MVC actions to be invoked only by GET or POST?

HttpGet or HttpPost attribute

## What is the use of Keep and Peek in “TempData”?

Once “TempData” is read in the current request it’s not available in the subsequent request.

If we want “TempData” to be read and also available in the subsequent request then after reading we need to call “Keep” method as shown in the code below.

## What is authentication and authorization in MVC?

**Authentication:** authentication is the process of checking user credential that is checking of user name and password provided by the user with the system database to allow user in the system to access the resources.

When a user comes to a website for the first time he will register for that website. All his information, like user name, password, email, and so on will be stored in the website database. When a user enters his userID and password, the information will be checked with the database. If the user has entered the same userID and Password as in the database then he or she is a valid user and will be redirected to the website home page. If the user enters a UserID and/or Password that does not match the database then the login page will give a message, something like "Enter valid Name or Password". The entire process of checking whether the user is valid or not for accessing the website is called Authentication.

**Authorization:** authorization start after authentication. Resources that are open for all is acceded by all user. But resources that are restricted and user try to access those, then the system check for user name and password for authentication. If the user is authenticated and are the right person to access, then the system allow the user to access the resources otherwise system redirect the user to login page.

Once the user is authenticated he needs to be redirected to the appropriate page by his role. For example, when an Admin is logged in, then he is to be redirected to the Admin Page. If an Accountant is logged in, then he is to be redirected to his Accounts page. If an End User is logged in, then he is to be redirected to his page.

## What are the types of authentication in asp.net MVC?

Two main type of authentications that are used mostly in ASP.NET applications.

Window Authentication: users are authenticated on their Windows username and password.

Form Authentication: In this type of authentication, the user will explicitly have to provide his credentials and these credentials, once verified by the server, will let the user to log in.

## Difference between HTTP and HTTPS?

**HTTP:** Unsecured

No encryption

No certificates required

Operates at Application Layer

**HTTPS:** Secured

Operates at Transport Layer

Encryption is present

Certificates is required

## What is HTTPS, SSL and TLS?

**SSL:** SSL stands for Secure Sockets Layer and, in short, it's the standard technology for keeping an internet connection secure.

The two systems can be a server and a client (for example, a shopping website and browser) or server to server (for example, an application with personal identifiable information or with payroll information).

It does this by making sure that any data transferred between users and sites, or between two systems remain impossible to read. It uses encryption algorithms to scramble data in transit, preventing hackers from reading it as it is sent over the connection. This information could be anything sensitive or personal which can include credit card numbers and other financial information, names and addresses.

**TLS:** TLS (Transport Layer Security) is just an updated, more secure, version of SSL.

**HTTPS:** HTTPS (Hyper Text Transfer Protocol Secure) appears in the URL when a website is secured by an SSL certificate. The details of the certificate, including the issuing authority and the corporate name of the website owner, can be viewed by clicking on the lock symbol on the browser bar.

## Explain WebConfig.cs in Asp.net MVC?

ASP.NET Web.config file provides you a flexible way to handle all your requirements at the application level.

Web.config files are stored in XML format which makes us easier to work with.

The changes in Web.config don’t require the reboot of the web server.

**Use of webconfig.cs file**

Configuration settings of ASP.NET like connection string

Custom Error Setting

Authentication, Authorization, Membership Provider, Role Provider and Profile Provider Settings

AppSettings like connection strings, file paths, URLs, port numbers, custom key value pairs, etc.

Page Settings like master page and theme for the pages in web application.

## Bind Multiple Models on view in MVC

**ViewModel:** ViewModel is a class which contains the properties which are represented in view or represents the data that you want to display on your view/page.

```
public class BlogCommentViewModel
 {
    public BlogModel Blog { get; set; }

    public List<CommentModel> Comments { get; set; }
 }
```

**ViewBag:** To pass multiple model we will create two view bag in our action method as below.

```
public ActionResult GetBlogComment()
  {
     ViewBag.Blog = GetBlogModel();
     ViewBag.Comments = GetCommentModel();
     return View();
  }
```

**ViewData:** To use the ViewData for multiple models change your action method as below.

```
public ActionResult GetBlogComment()
   {
      ViewData["Blog"] = GetBlogModel();
      ViewData["Comments"] = GetCommentModel();

      return View();
   }
```

**TempData:** TempData is also a dictionary derivative from TempDataDictionary class. TempData stored in short lives session. We can pass multiple models through TempData also.

```
public ActionResult GetBlogComment()
   {
      TempData["Blog"] = GetBlogModel();
      TempData["Comments"] = GetCommentModel();

      return View();
   }
```

**Session:** Session is used to pass data across controllers in MVC Application.Session data never expires.

```
public ActionResult GetBlogComment()
   {
      Session["Blog"] = GetBlogModel();
      Session["Comments"] = GetCommentModel();

      return View();
   }
```

**Tuples:** Tuple is an ordered sequence, immutable, fixed-size and of heterogeneous objects. each object in tuple is being of a specific type.

```
public ActionResult GetBlogComment()
   {
      var BCVM = new Tuple<BlogModel, List<CommentModel>>(GetBlogModel(),
GetCommentModel());
      return View(BCVM);
   }
```

And now change your view as below

```
@model Tuple<BlogModel,List<CommentModel>>
```

**Json:** We can Bind Multiple Models with the help of Json as well. We will retun JsonResult from action Method and on View through JQuery we can pasrse the JSON data and Bind on View.

**Why are there TWO Web.Config files in an ASP.NET MVC v1.0 solution?**

It looks like the inner web.config is for view specifc configutation such as blocking direct access to the views.

# Entity, Linq, MVVM

**What is Entity Framework?**

ADO.NET entity is an ORM (object relational mapping) which creates a higher abstract object model over ADO.NET components.

**What are the benefits of using EF?**

The main and the only benefit of EF is it auto-generates code for the Model (middle layer), Data Access Layer, and mapping code, thus reducing a lot of development time.

**Name of other object relational mapping**

NHibernate

EntitySpaces

LINQ to SQL

**What is the importance of EDMX file in Entity Framework?**

EDMX (Entity Data Model XML) is an XML file which contains all the mapping details of how your objects map with SQL tables. The EDMX file is further divided into three sections: CSDL, SSDL, and MSL.

CSDL (Conceptual Schema definition language) is the conceptual abstraction which is exposed to the application.

SSDL (Storage Schema Definition Language) defines the mapping with your RDBMS data structure.

MSL (Mapping Schema Language) connects the CSDL and SSDL.

**What are T4 templates?**

T4 (Text Template Transformation Toolkit) is a template based code generation engine. You can go and write C# code in T4 templates (*.tt* is the extension) files and those C# codes execute to generate the file as per the written C# logic.

**What is the difference between POCO, Code First and simple EF approach?**

In simple Entity Framework, everything is auto generated and so you need the EDMX XML file as well. POCO is semi-automatic so you have full control on the entity classes but then the context classes are still generated by the EDMX file. In Code First, you have complete control on how you can create the entity and context classes. Because you are going to manually create these classes, you do not have dependency on the EDMX XML file.

| | EDMX | Entity | Context |
|---|---|---|---|
| **Simple entity framework** | Needed | Auto | Auto |
| **POCO approach** | Needed | Manual | Auto |
| **Code First** | Not Needed | Manual | Manual |

## What is the difference between data context and object context?

**ObjectContext** The ObjectContext class is not thread-safe.

ObjectContext can be used by Entity Framework 4.0 and below.

ObjectContext is only useful in Model First and Database First approaches.

ObjectContext class is part of the core Entity Framework API, that allows us to perform queries, change and track updates of the Database by using strongly typed entity classes.

**DbContext** Any public static (C#) or Shared (Visual Basic) members of DbContext are thread-safe.

Any instance members are not guaranteed to be thread safe.

DBContext can be used by Entity Framework 4.1 and above.

DbContext is useful in Model First, Database First approach as well as Code First approach.

The DbContext class can be described as a wrapper of ObjectContext. It exposes the most commonly used features of ObjectContext.

## What is difference between lazy loading and early Loading?

In case of lazy loading, related objects (child objects) are not loaded automatically with its parent object until

They are requested. By default LINQ supports lazy loading.

In case of eager loading, related objects (child objects) are loaded automatically with its parent object.

To use Eager loading you need to use Include() method.

## What is the difference between IEnumerable vs IQueryable?

**IEnumerable**

System.Collections Namespace

No base interface

LINQ to Object and LINQ to XML queries

While querying data from database, IEnumerable executes select query on server side, load data in-memory on client side and then filter data. Hence does more work and becomes slow.

When querying data from in-memory collections like List, Array, etc.

**Best Use:** In-memory traversal

**IQueryable**

System.Linq Namespace

Derives from IEnumerable

While querying data from database, IQueryable executes select query on server side with all filters. Hence does less work and becomes fast.

LINQ to SQL queries

When querying data from out-memory (like remote database, service) collections.

**Best Use:** Paging

## What is MVVM?

**Model:** The model is what I like to refer to as the domain object. The model represents the actual data and/or information we are dealing with.

**View:** The view is what most of us are familiar with and the only thing the end user really interacts with. It is the presentation of the data.

**View Model:** The viewmodel also exposes methods, commands, and other points that help maintain the state of the view, manipulate the model as the result of actions on the view, and trigger events in the view itself.

## What is LINQ? Why we are using it?

LINQ stands for language Integrated Query. LINQ enables us to query any type of data store (SQL Server, XML documents, Objects in memory etc)

LINQ enables us to work with different data source using a similar coding style without having a need to know the syntax specific to the data source. Another benefits of using LINQ is that it provides intellisense and compile time error checking

## What is difference between Single, First, SingleOrDefault and FirstOrDefault?

**Single()** - There is exactly 1 result, an exception is thrown if no result is returned or more than one result.

**SingleOrDefault()** – Same as Single(), but it can handle the null value.

**First()** - There is at least one result, an exception is thrown if no result is returned.

**FirstOrDefault()** - Same as First(), but not thrown any exception or return null when there is no result.

Single() asserts that one and only one element exists in the sequence.

First() simply gives you the first one.

Use Single / SingleOrDefault() when you sure there is only one record present in database or you can say if you querying on database with help of primary key of table.

Developer may use First () / FirstOrDefault() anywhere, when they required single value from collection or database.

## Disadvantages of MVVM pattern?

Some people think that for simple UIs, MVVM can be overkill.

Similarly in bigger cases, it can be hard to design the ViewModel.

Debugging would be bit difficult when we have complex data bindings

## How to get common items from two lists using Linq to objects?

# Angular4

**What is difference between Angular2 and AngularJS?**

Angular2 is 5 times faster compared to AngularJS1

With Angular2 we can build a single application that works across mobile and desktop devices

In angular2 everything is a component. Components are the building blocks of an angular application

The architecture of AngularJS is based on MVC whereas the architecture of Angular 2 is based on service/controller.

One of the biggest advantages of Angular is Dependency Injection. In Angular 2, DI is there but now there is a different way to inject dependencies. As everything is a class in Angular, so DI is achieved via a constructor.

**What is difference between Angular4 and Angular2?**

In Angular 4, Some changes to reduce the size of the AOT(Ahead of time) compiler generation code

In Angular4, Type Script 2.1 and 2.2 compatibility

In Angular 4, Animation features pulled out of @angular/core and are moved into their own package

In Angular 4, new if/else style syntax with *ngif structural directive

**More languages** ECMAScript5, ECMAScript6, TypeScript, Dart, PureScript, Elm

Type script is open source programing languages

Superset of JavaScript

**Benefits:**

Intellisense

Autocompletion

Code navigation

Advanced refactoring

Strong Typing

Support features like classes, interfaces and inheritance

## What is Component?

Component in angular is a class with a template and decorator. Its compose template, class and decorator.

**Template** defines the user interface contains the html, directives and data binding

**Class** contains the code required for the template

**Decorator** add Meta data to the class making it an angular component

## What is pipes in Angular?

Transform data before display

Built in pipes include lowercase, uppercase, decimal, date, percent, currency etc

Use pipe character "|"

## Explain the life cycle hooks of Angular 2 application

Angular 2 component/directive has lifecycle events, managed by @angular/core. It creates the component, renders it, creates and renders its children, processes changes when its data-bound properties change, and then destroys it before removing its template from the DOM.

Some of the events are applicable for both component/directives while few are specific to components.

**ngOnChanges**: Responds when angular sets its data-bound property which receives the current and previous object values.

**ngOnInit**: Initializes the component/directive after first ngOnChange triggers. This is most frequently used method to retrieve the data for the template from a back-end service.

**ngDoCheck**: Detect and act upon changes occuring outside Angular context. It is called when every change detection run.

**ngOnDestroy**: Cleanup just before Angular destroys the directive/component. Unsubscribe observables and detach event handlers to avoid memory leaks.

**Component-specific hooks:**

**ngAfterContentInit**: Component content has been initialized

**ngAfterContentChecked**: After Angular checks the bindings of the external content that it projected into its view.

**ngAfterViewInit**: After Angular creates the component's view.

**ngAfterViewChecked**: After Angular checks the bindings of the component's view.

## What Is Typescript?

TypeScript is a typed super set of JavaScript which has been built and maintained by Microsoft and chosen by the AngularJS team for development.

## List the differences between Angular 2 components vs. directives.

In Angular 2, a component is a directive with a view whereas a directive is a decorator with no view. Components are the specific type of directive that allows us to utilize web component functionality throughout our application. Whereas, Directive is the mechanism by which we attach behavior to elements.

A component is used to break up the application into smaller components. Whereas, Directive is used to design the re-usable components.

Components can be used to define pipes. Whereas, we cannot define pipes using directives.

Components can be present per DOM element. Whereas, Directive is used to add behavior to an existing DOM element.

## Explain Modules in Angular 2

An AngularJS module defines an application.

The module is a container for the different parts of an application.

The module is a container for the application controllers.

```
<script>

var app = angular.module("myApp", []);

</script>
```

The "myApp" parameter refers to an HTML element in which the application will run.

Now you can add controllers, directives, filters, and more, to your AngularJS application.

**What are Angular 2 directives? Explain with examples.**

AngularJS directives are extended HTML attributes with the prefix ng-.

The ng-app directive initializes an AngularJS application.

The ng-init directive initializes application data.

The ng-model directive binds the value of HTML controls (input, select, textarea) to application data.

**How will you handle errors in Angular 2 applications?**

Angular has a built-in ErrorHandler class as part of @angular/core TypeScript module that handles exceptions that are not caught anywhere else within the application. By default, it is not enabled and set up in your Angular app (a bit odd, since the first version of AngularJS had a ready-to-use $log), so you need to import it and set it up in the providers array of your **@NgModule** decorator.

Use HttpErrorResponse

import {HttpErrorResponse} from '@angular/common/http'

Angular 2 applications have the option of error handling. This is done by including the ReactJS catch library and then using the catch function.

```
import { Observable } from 'rxjs/Observable';

import 'rxjs/add/operator/map';
import 'rxjs/add/operator/do';
import 'rxjs/add/operator/catch';
```

**What is routing?**

If you want to navigate to different pages in your application, but you also want the application to be a SPA (Single Page Application), with no page reloading, you can use the ngRoute module.

The ngRoute module routes your application to different pages without reloading the entire application.

**Explain tsconfig.json file**

Typically, you add a TypeScript configuration file called tsconfig.json to your project to guide the compiler as it generates JavaScript files.

The tsconfig.json file specifies the root files and the compiler options required to compile the project.

**Explain systemjs.config.json file**

You need to create a "systemjs.config.js" file and load it from index.html, like a regular script:

```
<script src="systemjs.config.js"></script>
```

In your systemjs.config.js file, you want to map the node package to a path

**Explain app.module.ts file**

Modules are a way of organizing and separating your code. You can have multiple modules and lazy load some modules. You import any other modules into the imports section. You declare any components in your declarations. Any components used in the routing of that module, must be declared in that module. If components are used in another module, then you only list them in that other module. And you provide your services in the providers section.

**How would you optimize the angular 2 application for better performance?**

Consider AOT compilation.

Make sure the application doesn't have un-necessary import statements.

Make sure that any 3rd party library, which is not used, is removed from the application.

Have all dependencies and dev-dependencies are clearly separated.

**What are the core differences between Observables and Promises?**

A Promise handles a **single event** when an async operation completes or fails.

Not cancellable

An Observable is like a **Stream** (in many languages) and allows to pass zero or more events where the callback is called for each event.

Cancellable

Often Observable is preferred over Promise because it provides the features of Promise and more. With Observable it doesn't matter if you want to handle 0, 1, or multiple events. You can utilize the same API in each case.

**What is @outputs In Angular?**

@Output decorator binds a property of a component to send data from one component (child component) to calling component (parent component). This is one way communication from child to parent component. @Output binds a property of the type of angular EventEmitter class. This property name becomes custom event name for calling component. @Output decorator can also alias the property name as @Output(alias) and now this alias name will be used in custom event binding in calling component.

## What is @inputs In Angular 2?

@Input decorator binds a property within one component (child component) to receive a value from another component (parent component). This is one way communication from parent to child. To use alias for the binding property name we need to assign an alias name as @Input(alias).

```
@Input('stdLeader')
myStdLeader : Student;
```

## What are Event Emitters and how it works in Angular 2?

Angular 2 doesn't have bi-directional digest cycle, unlike angular 1. In angular 2, any change occurred in the component always gets propagated from the current component to all its children in hierarchy. If the change from one component needs to be reflected to any of its parent component in hierarchy, we can emit the event by using Event Emitter api.

In short, EventEmitter is class defined in @angular/core module which can be used by components and directives to emit custom events.

```
@output() somethingChanged = new EventEmitter();
```

## What Is Npm?

npm (Node Package Manager) is the world's largest Software Registry. The registry contains over 800,000 code packages. Open-source developers use npm to share software. npm is installed with Node.js. npm is a package manager for Node.js with hundreds of thousands of packages.

## Difference between Constructor and Ngoninit?

**Constructor:** Typescript feature nothing to do with Angular

Constructor is transformed to function with the same name as class created

Called by JavaScript Engine

Constructor is automatically called at the time of creating object of the class

Used for injecting dependencies

Not everything in component is initialized at the time of invocation

**ngOnInit:** One of the Angular life cycle hook method

ngOnInit being added to prototype of the class created

Called by Angular

Invoked by Angular when everything in the component is ready

Actual business logic performed here

Everything is ready at the time of invocation

**What is AOT Compilation? Explain its advantages and disadvantages?**

AOT compilation stands for "Ahead of Time compilation" and it are used to compiles the angular components and templates to native JavaScript and HTML during the build time instead of run-time. The compiled HTML and JavaScript are deployed to the web server so that the compilation and render time can be saved by the browser. It is the big advantage to improve the performance of applications.

**Advantages of AOT**
**Faster download**: The Angular 2 app is already compiled so it is faster.
**Faster Rendering**: If the app is not AOT compiled and the compilation process happens in the browser once the application is fully loaded. This has a wait time for all necessary components to be downloaded and then the time taken by the compiler to compile the app. With AOT compilation, this is optimized.
**Lesser Http Requests**: It is supporting to the lazy loading. Actually, lazy loading is great concepts for sending HTTP request to the server. It is minimize the multiple requests for each associated html and css, there is a separate request goes to the server.
**Detect error at build time**: In Angular 2, the compilation happens beforehand and most of the errors can be detected at the compile time and this process providing us a better application's stability.

**Disadvantages of AOT**
AOT only works only with HTML and CSS and not for other file types. If required other file types that time we will need to follow the previous build step.
We need to maintain AOT version of bootstrap file.

We need to clean-up step before compiling.

## GENERAL

## Difference between Waterfall and Agile Methodology?

**Waterfall:** Waterfall methodology is a sequential design pattern. This mean that as each of the eight stages (Conception, initiation, analysis, design, construction, testing, implementation and maintenance) are completed, the developer move on the next step.

Once a step has been completed, developers can't go back a previous step.

If a requirement error is found, or a change needs to be made, the project has to start from beginning with all new code.

The whole product is only tested at the end.

Use: when there is a clear picture of what the final product should be

When clients won't have the ability to change the scope of the project once it has begun.

Agile: agile methodology follows an incremental approach.

Developers start off with a simplistic project design and then begin to work on small module. The work on these modules is done in weekly or monthly sprints. These sprints allow for bugs to be discovered, the clients feedback to be incorporated into the design before the next sprint is run.

The Agile methodology allows for changes to be made after the initial planning.

Use: When clients will be able to change the scope of the project.

When there isn't a clear picture of what the final product should look like.