# *Design pattern*

## ***Design principles***

*Software design principles represent a set of guidelines that helps us to avoid having a bad design.*

## ***VI\*\*\*\*S.O.L.I.D principle***

<https://www.codeproject.com/articles/93369/how-i-explained-ood-to-my-wife>

*SOLID Principle is a set of guidelines that helps us to avoid having a bad design*

### *S:* [*SRP*](https://en.wikipedia.org/wiki/Single_responsibility_principle)*:* [*Single responsibility principle*](https://en.wikipedia.org/wiki/Single_responsibility_principle)

*SRP says that a class should have only one responsibility and not multiple.*

*"****There should never be more than one reason for a class to change.****"*

*SRP seems to be an idea of breaking things into molecular parts so that it becomes reusable and can be managed centrally.*

### *O: OCP:* [*Open/closed principle*](https://en.wikipedia.org/wiki/Open/closed_principle)

*Class entities***(classes, modules, functions, etc.)** *should be open for extension, but closed for modification*

### *L: LSP:* [*Liskov substitution principle*](https://en.wikipedia.org/wiki/Liskov_substitution_principle)

*Objects in a program should be replaceable with instances of their subtypes without altering the correctness of that program.*

### *I: ISP:* [*Interface segregation principle*](https://en.wikipedia.org/wiki/Interface_segregation_principle)

*Many client-specific interfaces are better than one general-purpose interface.*

The Interface Segregation principle ensures that Interfaces are developed so that each of them have their own responsibility and thus they are specific, easily understandable, and re-usable.

### *D: DIP:* [*Dependency inversion principle*](https://en.wikipedia.org/wiki/Dependency_inversion_principle)

*Dependency inversion principle is a software design principle which states*

* *High-level modules should not depend on low-level modules. Both should depend on abstractions.*
* *Abstractions should not depend upon details. Details should depend upon abstractions.*

## ***VI\*\*\*\*D.R.Y Principle***

*This principle states that each small pieces of knowledge (code) may only occur exactly once in the entire system. This helps us to write scalable, maintainable and reusable code.*

*Dryness is achieved by good planning.*

*Example – Asp.Net MVC framework works on this principle.*

## ***VI\*\*\*\*K.I.S.S Principle***

*KISS (keep it small and simple) principle states that most systems work best if they are kept small and simple rather than made large and complicated.*

## ***VI\*\*\*\*Separation of Concerns***

***The process of breaking a computer program into distinct features that overlap in functionality as little as possible.***

## ***VI\*\*\*\*Dependency injection***

***Dependency injection is a process where we inject the dependent objects rather than consumer creating the objects.***

*Dependency injection can be done in three ways.*

***Property injection***

***Constructor injection***

***Method injection***

## ***G.R.A.S.P Pattern***

***General Responsibility Assignment Software Patterns (or Principles)***

### *Creator*

### *Information Expert (or just Expert)*

### *Low Coupling*

### *Controller*

### *High Cohesion*

### *Polymorphism*

### *Pure Fabrication*

### *Indirection*

### *Protected Variations*

## *YAGNI principle*

*It is a principle of Extreme Programming (XP) that states that a* ***programmer should not add functionality until really necessary.***

## *M.I.M.C Principle*

***More is more complex.***

*Having more lines of code, methods, classes, packages, executables, libraries etc. always means also to have more complexity (which is bad).*

## *RoE Principle*

*Explicit is better than implicit.*

## *G.P Principle*

***Generalization Principle***

*A generalized solution, that solves not only one but many problems, is better than a specific one.*

## *Murphy's Law (ML)*

*Whatever can go wrong, will go wrong. So a solution is better the less possibilities there are for something to go wrong.*

## *Anti-patterns*

*The selections of wrong design pattern is defined as “anti-pattern”.*

## *VIQ\*\*\*\*Repository pattern*

***A Repository pattern acts as a bridge between the domain and date mapping layers, acting like an in-memory domain object collection.***

*A Repository represents all objects of a certain type as a conceptual set. It acts like a collection, except with more elaborate querying capability.*

*A Repository mediates between the domain and data mapping layers, acting like an in-memory domain object collection. Client objects construct query specifications declaratively and submit them to Repository for satisfaction. Objects can be added to and removed from the Repository, as they can from a simple collection of objects, and the mapping code encapsulated by the Repository will carry out the appropriate operations behind the scenes*

[***http://www.dotnetcurry.com/aspnet-mvc/1155/aspnet-mvc-repository-pattern-perform-database-operations***](http://www.dotnetcurry.com/aspnet-mvc/1155/aspnet-mvc-repository-pattern-perform-database-operations)

## ***VIQ\*\*\*\*Design pattern***

***The Design Patterns are nothing more than concrete implementations of Design principles, and it is a solution to a general software problem within a particular context. And it is broadly divided into Creational, Structural, and Behavioral patterns.***

## *Types of Design Patterns*

*These can be organized in 4 separate pattern groups depending on the nature of the design problem they intend to solve.*

### *Gang of Four Patterns*

### *Enterprise Patterns*

### *SOA and Messaging Patterns*

### *Model-View Patterns*

## *Main categories of GoF pattern*

### ***Structural Patterns:***

***Structural design patterns focus on relationships/interfaces between entities and objects***

#### \* Adapter

***Adapter pattern acts as a bridge between two incompatible interfaces. This pattern involves a single class called adapter which is responsible for communication between two independent or incompatible interfaces.***

#### \* Bridge

***Bridge pattern is used to separate an abstraction from its implementation so that both can be modified independently.***

#### \* Composite

***Composite pattern is used where we need to treat a group of objects in similar way as a single object. Composite pattern composes objects in term of a tree structure to represent part as well as whole hierarchy.***

*\* Decorator*

***Decorator pattern is used to add new functionality to an existing object without changing its structure.***

#### \* Façade

***Provide a unified interface to a set of interfaces in a subsystem.***

***Facade pattern hides the complexities of the system and provides an interface to the client using which the client can access the system.***

#### \* Flyweight

#### Flyweight pattern is used to reduce the number of objects created, to decrease memory and resource usage. As a result it increase performance.

#### \* Proxy

***Proxy pattern involves a class, called proxy class, which represents functionality of another class.***

***An object representing another object.***

### ***Creational Patterns:***

***The Creational Design Patterns focus on how objects are created and utilized in an application***

#### \* Factory Method

***In this pattern, an interface is used for creating an object, but let subclass decide which class to instantiate. The creation of object is done when it is required. The Factory method allows a class later instantiation to subclasses.***

#### \* *Abstract Factory* –

***In Abstract Factory pattern an interface is responsible for creating a set of related objects, or dependent objects without specifying their concrete classes.***

#### \* Builder

***Builder pattern describes a way to separate an object from its construction. The same construction method can create different representation of the object.***

#### \* Prototype

***Prototype pattern is used to create a duplicate object or clone of the current object to enhance performance. This pattern is used when creation of object is costly or complex.***

#### \* Singleton

***This pattern ensures that a class has only one instance and provides a global point of access to it.***

### ***C) Behavioural Patterns:***

#### **\* Iterator**

***Provide a way to access the elements of an aggregate object sequentially without exposing its underlying representation.***

#### **\* Observer**

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

## *VIQ\*\*\*\*Difference between Factory Method and Abstract Method*

1. *The factory method is a single method, and an abstract factory is an object.*
2. *Factory Methods uses inheritance and the abstract Factory uses object composition.*
3. *Factory Methods lets the subclasses to decide which class to instantiate. Whereas the abstract factory provides and interface for creating families of related or dependent objects*

*With the Factory pattern, you produce implementations (Apple, Banana, Cherry, etc.) of a particular interface -- say, IFruit.*

*With the Abstract Factory pattern, you produce implementations of a particular Factory interface -- e.g., IFruitFactory. Each of those knows how to create different kinds of fruit.*

## *VIQ\*\*\*\*7 Steps to fix a bug*

*Step 1. Identify the error.*

*1. See the error.*

*2. Reproduce the error.*

*3. Understand what the expected behavior should be.*

*4. Validate the identification.*

*Step 2. Find the error.*

* *Logging.*

1. *Event log*
2. *IIS log*
3. *Data base Log*
4. *etc.*

* *Debugging.*
* *Removing code.*

*Step 3. Analyze the error.*

*Step 4. Prove your analysis.*

*Step 5. Cover lateral damage.*

*Step 6. Fix the error.*

*Step 7. Validate the solution.*

## VIQ\*\*\*\*25 Ways to Lead, Inspire and Motivate Your Team to Greatness

[*https://www.entrepreneur.com/article/241883*](https://www.entrepreneur.com/article/241883)

### **1. Find out what they need**

### **2. Ask a lot of questions**

### **24. Hold brainstorming sessions**

### **3. Be accessible**

### **4. Be decisive**

### **5. Be flexible**

### **6. Be personable**

***If your team knows they can come to you at any time with a problem, concern or suggestion and you will hear them out then it is going to help create a well-oiled machine.***

### **7. Be respectful**

***Show respect for their hard work and dedication -- it will create a much more productive team.***

### **8. be transparent**

### **12. Show emotions**

***Nobody wants to be led by an emotionless robot. Let your emotions shine -- smile, laugh and have fun with your team.***

### **11. Do what you say**

### **20. Demonstrate knowledge**

### **9. Lead by example**

### **14. Admit when you don’t have the answer**

### **15. Don’t impose fear**

### **17. Develop future leaders from within**

### **18. Encourage personal growth**

### **13. Push their limits**

### **16. Help overcome weakness**

### **19. Hold your team accountable**

### **21. Understand your purpose**

### **22. Welcome involvement**

### **23. Get to know your team**

### **10. Stress your company’s purpose**

### **25. Stop micromanaging**

***Trust your team members to perform****. --* ***If you made the wrong choice you need to take responsibility and correct it.***

# *C# and OOPs*

## *Class:*

*A class is a blueprint or prototype that defines the variables and the methods (functions) common to all objects of a certain kind.*

*A class describes a group of objects with...*

* *Similar properties (attributes),*
* *Common behavior (operations),*
* *Common relationships to other objects, and*
* *Common meaning (“semantics”).*

## *OBJECT:*

*The Bundle of variables and method (functions) that operates on date is called OBJECT.*

## *Object pool:*

*An object pool is a container of objects that holds a list of other objects that are ready to be used. It keeps track of:*

* *Objects that are currently in use*
* *The number of objects the pool holds*
* *Whether this number should be increased*

## *Application pooling?*

*An Application Pool can contain one or more applications and allows us to configure a level of isolation between different Web applications*

## *VID\*\*\*\*****Attribute***

*An****attribute****is a declarative tag that is used to convey information to runtime about the behaviors of various elements like classes, methods, structures, enumerators, assemblies etc.*

## *Default Access modifiers in C#?*

*An* ***enum*** *has default modifier as* ***public***  *An* ***Class*** *has default modifier as* ***private.*** *A Classe in a namespace are internal, by default.   
An* ***interface*** *has default modifier as* ***public*** *.  
A* ***struct*** *has default modifier as* ***Internal*** *and it can declare its members (methods etc) with following access modifiers: public , internal , private   
A* ***methods, fields, and properties*** *has default access modifier as "Private" if no modifier is specified.*

## *What is sealed modifiers?*

*Sealed modifiers are the access modifiers where it cannot be inherited by the methods. Sealed modifiers can also be applied to properties, events and methods. This modifier cannot be applied to static members.*

## *Constructor:*

*A special method of the class that will be automatically invoked when an instance of the class is created*

* *A constructor is a simple method which has the same name as the class and hence used to create object of a class. C# class can define any number of constructors. Every class contains a default constructor.*
* *Return types of parameters of object of class matches with defined constructor arguments types.*
* *Constructors are never called explicitly.*
* *Constructors are used to initialize the objects. Once the object is declared means, the constructor is also declared by default.*
* *A constructor can be a static constructor*
* *Constructors do not have any return type not even void included in it.*

***Default Constructor:****A constructor without any parameters is called as default constructor.*

***Parameterized Constructor:****A constructor with at least one parameter is called as parameterized constructor.*

***Copy Constructor:******A parameterized constructor that contains a parameter of same class type is called as copy constructor****. Main purpose of copy constructor is to initialize new instance to the values of an existing instance.*

*This is a special constructor for creating a new object as a copy of an existing object. There will be always only on copy constructor that can be either defined by the user or the system.*

***VIQ\*\*\*\*Static Constructor:******Static constructor is used to initialize static fields of the class and to write the code that needs to be executed only once.***

* *A static constructor cannot be called directly.*
* *A constructor doesn’t have any return type even void.*
* *A static constructor cannot be a parameterized constructor.*
* *Within a class you can create only one static constructor.*
* *The static constructor for a class executes before any instance of the class is created.*
* *The static constructor for a class executes before any of the static members for the class are referenced.*
* *The static constructor for a class executes after the static field initializers (if any) for the class.*
* *A static constructor does not take access modifiers or have parameters.  
  6) A static constructor is called automatically to initialize the class before the first instance is created or any static members are referenced.   
  7) A static constructor cannot be called directly.  
  8) The user has no control on when the static constructor is executed in the program.   
  9) 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.*

***VIQ\*\*\*\*Private Constructor:****A private constructor is a special instance constructor. It is commonly used in classes that contain static members only.*

***Private constructor is simple constructor that has private access modifier.***

*Private constructors are used for two purposes.*

*1)* ***To prevent creation of objects outside******the class***

*2) When your application requires a class that has only one instance and you need to provide a global point to access that instance like in the case of singleton.*

## *VIQ\*\*\*\*What is the difference between new and override?*

*The new modifier instructs the compiler to use the new implementation instead of the base class function. Whereas, Override modifier helps to override the base class function.*

## *Encapsulation:*

* *Wrapping up of data and methods in a single unit, Hiding the internal details or mechanics of how an object does something and Exposing the functionality to the world is known as Encapsulation.*
* *Encapsulation is implemented by using access specifiers.*

*Access modifier*

* *Access modifiers determine the scope of the method or variables that can be accessed from other various objects or classes. There are 5 types of access modifiers*
* *An enum has default modifier as public.*
* *A class has default modifiers as private and become internal when used with namespace.*

## *VID\*\*\*\*Association*

***Association represents a relationship between two or more objects where all objects have their own lifecycle and there is no owner.***

## *VID\*\*\*\*Aggregation*

***Aggregation represents a relationship between two or more objects where all objects have their own lifecycle but there is ownership.***

## *Composition*

***Composition is strong type of Aggregation***

***Composition represents a relationship between two or more objects where all objects have their own lifecycle but child objects does not have their lifecycle without Parent object.***

## *Static Class*

*Static Class is a simple class that cannot be instantiated and only allow static members.*

## *Abstract class:*

*Abstract class are the classes that build with implementation gape for sub-class to fill in. A class that can’t be instantiated and can only be derived from.*

1. *An abstract method does not have implementation*
2. *An abstract method can take either static or virtual modifiers*
3. *An abstract method can be declared only in abstract class*
4. *Any subclass of an abstract class must either implement all of the abstract method in the super class or itself be declared abstract.*

## *Interface:*

*Interface is a contract that defines the signature of the functionality that must be implemented by the classes. An interface is a collection of abstract method.*

## *Polymorphism:*

*The ability of the supper class reference to denote object of its own class and its subclass at runtime is called Polymorphism.*

*A class can be used as more than one type; it can be used as its own type, any base types, or any interface type if it implements interfaces. This is called polymorphism.*

***Overloading****and****overriding****are used to implement polymorphism. Polymorphism is classified into compile time polymorphism or early binding or static binding and Runtime polymorphism or late binding or dynamic binding.*

*Assigning behavior or value in a subclass to something that was already declared in the main class.*

*Polymorphism is of two types…*

1. *Compile time Polymorphism -------------Method Overloading*
2. *Run time Polymorphism --------------Method Overriding*

*Method Overloading:*

*The process of creating more than one method in a class with same name or creating a method in derived class with same name as a method in base class is called as method overloading.*

*Method overriding is a feature that allows sub class to provide implementation of a method that is already defined in the main class.*

*Method overloading occurs when a class contains two methods with the same name, but different signatures.*

*Method Overriding: is a process of redefining the supper-class method in the derived class*

*When a virtual method in a base class has its functionality superimposed or overridden by a method in a subclass, which has the same name and signature.*

## *Difference between overloading and overriding?*

*Overloading is static binding whereas Overriding is dynamic binding. Overloading is nothing but the same method with different arguments , and it may or may not return the same value in the same class itself.*

*Overriding is the same method names with same arguments and return types associates with the class and its child class.*

*.*

## *What is inheritance?*

*The process of forming a new class from an existing class is called Inheritance.*

*Use of ‘new’ modifier hides the inherited member i.e it makes only inherited member inaccessible in derived class and hence calls suitable method().*

## *VID\*\*\*\*Extension methods*

***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.***

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

*Extension methods are only in scope when you explicitly import the namespace into your source code with a using directive.*

*namespace ExtensionMethods*

*{*

*public static class MyExtensions*

*{*

*public static int WordCount(this String str)*

*{*

*return str.Split(new char[] { ' ', '.', '?' },*

*StringSplitOptions.RemoveEmptyEntries).Length;*

*}*

*}*

*}*

## *VID\*\*\*\*Delegate:*

*A delegate is a sealed and immutable reference type that contain the reference of a class method or event handler function that matches the delegate signature.*

* *Any method which has the same signature as delegate can be assigned to delegate.*
* *At the time of event handing delegates or used*
* *Delegate types are sealed and immutable type.*

## *VID\*\*\*\*Events:*

*An event is a mechanism via which a class can notify its clients when something happens.*

***An event is a way for a class to allow clients to give it delegates to methods that should be called when the event occurs. When the event occurs, the delegate(s) given to it by its clients are invoked.***

*Events are declared using delegates.*

*An event is an accessor for a delegate object*

*using System;*

*public delegate void EventHandler();*

*class Program*

*{*

*public static* ***event*** *EventHandler \_show;*

*static void Main()*

*{*

*// Add event handlers to Show event.*

*\_show += new EventHandler(Dog);*

*\_show += new EventHandler(Cat);*

*\_show += new EventHandler(Mouse);*

*\_show += new EventHandler(Mouse);*

*// Invoke the event.*

*\_show.Invoke();*

*}*

*static void Cat()*

*{*

*Console.WriteLine("Cat");*

*}*

*static void Dog()*

*{*

*Console.WriteLine("Dog");*

*}*

*static void Mouse()*

*{*

*Console.WriteLine("Mouse");*

*}*

*}*

***Output***

*Dog*

*Cat*

*Mouse*

*Mouse*

## *Linked list:*

***Linked list is a data structure that has a series of connected nodes.***

## *VID\*\*\*\*Collection:*

***A collection is a data structure that has a series of connected nodes that holds 0 or more objects of a given class.***

***Typical collection types include stacks, queues, lists and hash tables***

## *VID\*\*\*\*Generics*

***A Generics are the Classes that have a type parameter <T> representing the data type of the variable of the Classes\*.***

*Methodologies that enable class, delegate, interface, struct types, and methods to be created with type parameters, which are “placeholder” types that can be substituted when the type is known.*

*public class MyGenericArray<T>*

*{*

*private T[] array;*

*public MyGenericArray(int size)*

*{*

*array = new T[size + 1];*

*}*

*public T getItem(int index)*

*{*

*return array[index];*

*}*

*public void setItem(int index, T value)*

*{*

*array[index] = value;*

*}*

*}*

## *Generic method:*

*A method that accepts type parameters*

## *Enum:*

*A value type that contains an enumerator list (****a set of named numeric constants****)).*

*An enum variable cannot have a protected access modifier.*

*An enum can be declared inside a clas.*

*enum element cannot be assigned a value outside the enum declaration.*

## *Serialization:*

*Serialization is the process of converting an object or a graph of objects into a linear sequence of bytes for either storage or transmission to another location.*

*Deserialization is the process of taking in stored information and recreating objects from it*

*NET provides 2 ways for serializtion*

*1) XmlSerializer (*[*Web Services*](http://www.dotnetuncle.com/OOPS/24_serialization.aspx)*)and*

*2) BinaryFormatter/SoapFormatter(Remoting)*

## *Managed code:*

***Managed code means the code which gets complied by CLR and then gets converted into IL***

## *Value type:*

***A data type is a value type if it holds the data within its own memory allocation.***

## *Reference type:*

***A data type is a reference type if it holds a pointer to another memory location that holds the data.***

## *Ref keyword:*

*If you prefix a parameter with the ref keyword,* ***the program becomes an alias for the actual argument rather than copy of the argument. When using a ref parameter anything you do to the parameter, you also do to the original argument.***

## *Out keyword:*

*Out keyword is used for passing a variable for output purpose. It has the same concept of ref keyword. Passing a ref parameter needs variable to be initialized while out parameter is passed without initialized.*

## *Yield Keyword:*

## *Garbage Collector*

*GC is the process by which, dynamically allocated memory is reclaimed that is no longer in Use.*

## *Finalize() and Dispose()*

*Dispose () is called by as an indication for an object to release any unmanaged resources it has held.   
Finalize () is used for the same purpose as dispose however* ***finalize doesn’t assure the garbage collection of an object.****Dispose () operates deterministically due to which it is generally preferred.*

## *Reflection:*

***It is a mechanism discovering class information and read metadata at runtime.*** *Using reflection, it is possible to uncover the methods, Properties, and events of a type, and to invoke them dynamically.*

## *Array:*

*An Array is a collection of contiguous block of memory in which unordered sequence of same type of element live.* ***Array is reference type, regardless of their elements.***

***Array List:*** *A dynamic array class:   
a. it has a default capacity of 16 objects.   
b. It automatically increases in size.   
c. It holds any type of object reference.   
When accessing the objects, they need to be cast to the appropriate type.*

***Jagged array:*** *A multidimensional array where the rows are of varying sizes: an array is comprised of elements that are themselves arrays. It is also known as Array of Arrays.*

***Multidimensional array:*** *An array with multiple dimensions (rows and columns). C# has two: jagged ([][])and rectangular([,]).*

## *Indexer:*

***Indexers are smart arrays which allow an instance of a class or a structure to be indexed just like an array.***

***Indexer is used to access the class instances by means of Index notation.***

## *Wrapper Classes:*

***Wrapped Classes are the classes that allow primitive type to be accessed as an object.***

# *Basic .NET Framework | Asp.net | IIS |Caching | Threading*

## *CLR*

***CLR is an environment in which programs are executed, it activate object, perform security check on them, lay them out in the memory, execute them and garbage collect them.***

## *CAS*

***CAS grants rights to program depending on the security configuration of the machine.*** *Example the program has rights to edit or create a new file but the security configuration of machine does not allow the program to delete a file.* ***AS will take care that the code runs under the environment of machines security configuration.***

## *Assembly*

***An assembly is a collection of DLL and/or EXE files, and is known as basic building blocks of the .NET framework. They are the logical grouping of the functionality in a physical file.***

***An assembly is a partially compiled library for use in deployment, versioning and security.******There are two type:***

* ***process assemblies (***[*EXE*](http://en.wikipedia.org/wiki/EXE)***)***
* ***Library assemblies (***[*DLL*](http://en.wikipedia.org/wiki/Dynamic-link_library)***).***

***There are two kinds of assemblies in .NET****;*

* *private*
* *shared*

***Private assemblies*** *are simple and copied with each calling assemblies in the calling assemblies folder.*

***Shared assemblies*** *(also called strong named assemblies) are copied to a single location (usually the Global assembly cache).*

## *Difference between an EXE and a DLL?*

***DLL:***

***1) It has versioning***

***2) It is not self-executable***

***3) It runs in application process memory***

***4) It has no entry point***

***5) It is reusable***

***Exe:***

***1) It has no versioning***

***2) It is self-executable***

***3) It runs in own memory***

***4) It have main function (Entry point)***

## *Manifest*

***An assembly manifest is a text file containing metadata about .net assemblies. It*** *describes the relationship and dependencies of the components in the assembly, versioning information, scope information and the security permissions required by the assembly.*

## *Satellite assemblies*

***Assemblies which contains culture information are known as satellite assemblies***

## *GAC*

***Global Assembly Cache (GAC) is a common place to share the .NET assemblies across many applications****. GAC caches all strong named assembly references within it. All System assemblies that come with the .NET framework reside in the GAC.*

## *JIT*

***JIT is the process of compiling MSIL code units just when needed at runtime****. The* ***JIT compiler in the Common Language Runtime (CLR) compiles MSIL instructions to native machine code as a .NET application is being executed****. Compilation occurs when a method is called and is not compiled more than once during program execution; because****, JIT-compiled code is cached in memory.***

*JIT compiler is a part of the runtime execution environment.   
In Microsoft .NET there are three types of JIT compilers:  
.Pre-JIT: - Pre-JIT compiles complete source code into native code in a single compilation cycle. This is done at the time of deployment of the application.   
.Econo-JIT: - Econo-JIT compiles only those methods that are called at runtime. However, these compiled methods are removed when they are not required.  
.Normal-JIT: - Normal-JIT compiles only those methods that are called at runtime. These methods are compiled the first time they are called, and then they are stored in cache. When the same methods are called again, the compiled code from cache is used for execution.*

## *Strong Name*

*A strong name consists of the assembly's identity — its simple text name, version number, and culture information (if provided) — plus a public key and a digital signature.*

***There are two ways to sign an assembly with a strong name:*** *1. Using the* ***Assembly Linker (Al.exe)*** *provided by the .NET Framework SDK.*

*To create and sign an assembly with a strong name using the Assembly Linker, at the command prompt, type the following command:****al /out:<assembly name> <module name> /keyfile:<file name>***

*2. Using* ***assembly attributes*** *to insert the strong name information in your code.*

*al /out:MyAssembly.dll MyModule.netmodule /keyfile:sgKey.snk  
To sign an assembly with a strong name using attributes*

## *Delay signing*

***Delay signing is the process of adding strong name to the assembly at the later stage of development. Signing an assembly means adding a strong name to the assembly. As the strong name is added at the later stage due to security reasons it is called as delayed signing.***

## *Side-by-side execution*

***Side-by-side execution******is the ability to run multiple versions of an application or component on the same computer.***

## *E\*\*\*\*What is the difference between a.Equals(b) and a == b?*

*For value types: “==” and Equals() works same way*

*“==” compares reference – returns true if and only if both references point to the SAME object while*

*"Equals" method compares object by VALUE and it will return true if the references refers object which are equivalent*

*Recommendation:*

*For value types: use “==”*

*For reference types: use Equals method.*

## *Page Life cycle of asp.net Page*

|  |  |
| --- | --- |
| **Stage** | **Description** |
| Page request | The page request occurs before the page life cycle begins. When the page is requested by a user, ASP.NET determines whether the page needs to be parsed and compiled (therefore beginning the life of a page), or whether a cached version of the page can be sent in response without running the page. |
| Start | In the start stage, page properties such as [Request](https://msdn.microsoft.com/en-us/library/system.web.ui.page.request.aspx) and [Response](https://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) are set. At this stage, the page also determines whether the request is a postback or a new request and sets the [IsPostBack](https://msdn.microsoft.com/en-us/library/system.web.ui.page.ispostback.aspx) property. The page also sets the [UICulture](https://msdn.microsoft.com/en-us/library/system.web.ui.page.uiculture.aspx) property. |
| Initialization | During page initialization, controls on the page are available and each control's [UniqueID](https://msdn.microsoft.com/en-us/library/system.web.ui.control.uniqueid.aspx) property is set. A master page and themes are also applied to the page if applicable. If the current request is a postback, the postback data has not yet been loaded and control property values have not been restored to the values from view state. |
| Load | During load, if the current request is a postback, control properties are loaded with information recovered from view state and control state. |
| Postback event handling | If the request is a postback, control event handlers are called. After that, the [Validate](https://msdn.microsoft.com/en-us/library/system.web.ui.webcontrols.basevalidator.validate.aspx) method of all validator controls is called, which sets the [IsValid](https://msdn.microsoft.com/en-us/library/system.web.ui.ivalidator.isvalid.aspx) property of individual validator controls and of the page. (There is an exception to this sequence: the handler for the event that caused validation is called after validation.) |
| Rendering | Before rendering, view state is saved for the page and all controls. During the rendering stage, the page calls the [Render](https://msdn.microsoft.com/en-us/library/system.web.ui.control.render.aspx) method for each control, providing a text writer that writes its output to the [OutputStream](https://msdn.microsoft.com/en-us/library/system.web.httpresponse.outputstream.aspx) object of the page's [Response](https://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) property. |
| Unload | The [Unload](https://msdn.microsoft.com/en-us/library/system.web.ui.control.unload.aspx) event is raised after the page has been fully rendered, sent to the client, and is ready to be discarded. At this point, page properties such as [Response](https://msdn.microsoft.com/en-us/library/system.web.ui.page.response.aspx) and [Request](https://msdn.microsoft.com/en-us/library/system.web.ui.page.request.aspx) are unloaded and cleanup is performed. |

1. *PreInit*
2. *Init*
3. *InitComplete*
4. *PreLoad*
5. *Load*
6. *Control events*
7. *LoadComplete*
8. *PreRender*
9. *SaveStateComplete*
10. *Render*
11. *Unload*

## *Application Pool*

*An Application Pool can contain one or more applications and allows us to configure a level of isolation between different Web applications*

## *Validation controls in Asp.net*

***RequiredFieldValidator*** ***RangeValidator***

***RegularExpressionValidator******CompareValidator******CustomValidator******Validation Summary***

## *“GLOBAL.ASAX”*

***The Global.asax file is used to handle application-level and session-level events.***

*1.****Application\_Init****:*

*2.****Application\_Start***

*3.****Application\_***

*4.****Application\_EndRequest***

*5.****Application\_AuthenticateRequest***

*6.****Application\_Error***

*7.****Application\_End***

*8.****Session\_Start***

*9.****Session\_End***

## *Authentication*

*Authentication is the process of identifying who you are.*

1. *Windows*

* *Anonymous*
* *Basic*
* *Digest*
* *Win Integration*

1. *Form*
2. *Passport*

## *Authorization*

*Authorization refers to the process of identifying the resources that you are allowed to access.*

## *Handle Exception in Asp.Net? (Asked in HCL)*

***We can catch and handle errors at three levels***

***At the method level by using try...catch blocks for performing exception handling.***

***At the page level by using Page.Error event. Any error on the page - except for those within try...catch blocks will raise this event.***

***At the application level by:***

***a) Using the Application\_Error event in a global.asax file.***

***b) Using the web.config file .Any error within the application that is not handled by the previous methods will display the error handling page.***

## *Bubbled event*

***The process of listening to the child control in the main or parent control is called as event bubbling.***

## *VID\*\*\*\*HttpModule*

Any class that implements the System.web.IHttpModule interface becomes HttpModule and runs as an assembly that is called on every request made to your application and also acts as a station along the way.

***It's just like a filter. The Modules are called before and after the handler executes.***

***For example: BeginRequest, AuthenticationRequest event, EndRequest event etc. You may intercept , participate and modify each request.***

## *VID\*\*\*\*HTTPHandler*

**An ASP.NET HTTP handler is the *end point* process that runs in response to a request made to an ASP.NET Web application. The most common handler is an ASP.NET page handler that processes .aspx files. When users request an .aspx file, the request is processed by the page via the page handler.**

## *Web Forms*

***Web Forms******where single site hosted on multiple IIS Server and they are running behind the Load Balancer.***

***Web garden***

*An application pool with multiple worker process is called Web Garden.*

## Application Domain

***In which multiple applications can run in same Process without influencing each other is known as application domain.***

## ***VI\*\*\*\*Caching***

***Caching means temporary storage of data in memory that is highly-demanding and frequently used in order to accelerate performance and for quick access to various essential information.***

***1. Output Caching:*** *Output cache stores a copy of the finally rendered HTML pages or part of pages sent to the client*

***2. Fragment Caching/******Partial Page****: the caching of individual user controls within a Web Form.*

***3. Object Caching:*** *Object caching is caching the objects on a page, such as data-bound controls. The cached data is stored in server memory.*

***5. Class Caching:*** *Web pages or web services are compiled into a page class in the assembly, when run for the first time. Then the assembly is cached in the server*

***6. Data Caching:*** *Data caching means caching data from a data source. As long as the cache is not expired.*

***7. Configuration Caching:*** *Configuration caching stores the configuration information in the server memory.*

## *Cache dependency*

***The ability to invalidate a database cache if data in a table is changed****.*

* *File based dependencies*
* *Key based dependencies*
* *Time based dependencies*
* *SQL based dependencies*

## *Cache Call-back in Cache*

***ASP.NET provides capability to execute a call-back method when that item is removed from cache.***

## ***Scavenging***

*A process where items are removed from cache in order to free the memory based on their priority. A property called "CacheItemPriority" is used to figure out the priority of each item inside the cache. This priority property is set when an item is added to the cache.*

## *View State*

***ViewState is a .Net mechanism to store the posted data among post backs.***

***View State data is stored in a hidden field called "\_\_VIEWSTATE" on a Web page.***

***This single hidden field contains all the viewstate values for all the page controls.***

## *Cross page posting*

*The Server.Transfer() method is used to post data from one page to another. In this case, the URL remains the same. However, in* ***cross page posting, data is collected from different Web pages and is displayed on a single page.*** *To do so, you need to set the PostBackUrl property of the control, which specifies the target page. In the target page, you can access the PreviousPage property. For this, you need to use the @PreviousPageType directive. You can access the controls of previous page by using the FindControl()* ***method****.*

## *difference between Cache Object and application object?*

*1) Application object always stores data on the server side RAM****example:*** *Application ["hits"] =1 ;( key-value pair).  
2) Application object maintains its data till the web application is shut down or we release the data manually by assigning null or Clear () method is called.  
3) Application object has no Timeouts or File Dependencies.  
4) Its data can be assigned using Global.asax file  
5) Application object is not used for performance optimization.  
USED in maintaining hit counters, data from readonly files/tables which can then  
be displayed on varrious web pages.****CACHE OBJECT****:  
1)Cache object can store the data on server side RAM as well as client side RAM****example:*** *-- Cache["data"]="asp.net";  
2)Cache object maintains the static data as specified by the Absolute Expiration/ Sliding Expiration or File Dependency. The Time Period for Cache can be defined using the Cache.Insert() overloaded method or Cache.Add() method. It can be from seconds to years.  
3) Cache object can be assigned data from web page and not from Global.asax file.  
4) Cache is used for performance optimization. We retreive the Cache data from  
the Cache without repeating the full cycle again, which is not so in the case  
of the Application object.*

## *Multi-threading*

***Switching between different parts of the same program.***

## *Thread.join() in threading?*

***The thread.join method is useful for determining if a thread has completed before starting another task.***

***Thread.join(integer) ensures that threads do not wait for a long time****.*

## ***How you will improve web application Performance?***

### ***Upgrade Your ASP.NET Framework***

### ***Removing Default HTTP Modules in ASP.NET***

### ***Avoid Round trips***

### ***Always keep CSS and JavaScript External***

### ***Use Caching***

### ***Use File Compression***

### ***Use Bundling and Magnification***

### ***Use CDN (Content Delivery Network)***

### ***Use Control Image Requests***

### ***Script Rendering Order***

### ***Compile in Release Mode***

## Bundling

Bundling is a new feature in ASP.NET 4.5 that makes it easy to combine or bundle multiple files into a single file. You can create CSS, JavaScript and other bundles. Fewer files means fewer HTTP requests and that can improve first page load performance.

# *WCF | Web API*

***Windows Communication Foundation (WCF) is a programming platform (sdk) and runtime system for building, configuring and deploying network-distributed services. WCF is a combined features of Web Service, Remoting, MSMQ and COM+***

## *Main components of WCF*

***1. Service class 2. Hosting environment 3. End point***

## ***Endpoint in WCF***

*The relationship between Address, Contract and Binding (Where, What and How) is called Endpoint.*

*Every service must have Address that defines where the service resides, Contract that defines what the service does and a Binding that defines how to communicate with the service.*

## *Contract in WCF*

*A)* ***Service Contract****: Describes what operations the client can perform.*

*b)* ***Operation Contract****: defines the method inside Interface of Service.*

*c)* ***Data Contract****: Defines what data types are passed*

*d)* ***Message Contract:*** *Defines whether a service can interact directly with messages*

*e)* ***Fault Contract****: Defines which error is raised by the server, and how the service handles and propagates errors to its client.*

## ***Binding (How)***

*A binding defines how an endpoint communicates to the world. A binding defines the transport (such as HTTP or TCP) and the encoding being used (such as text or binary).*

|  |  |
| --- | --- |
| ***Binding*** | ***Description*** |
| *BasicHttpBinding* | *Basic Web service communication. No security by default* |
| *WSHttpBinding* | *Web services with WS-\* support. Supports transactions* |
| *WSDualHttpBinding* | *Web services with duplex contract and transaction support* |
| *WSFederationHttpBinding* | *Web services with federated security. Supports transactions* |
| *MsmqIntegrationBinding* | *Communication directly with MSMQ applications. Supports transactions* |
| *NetMsmqBinding* | *Communication between WCF applications by using queuing. Supports transactions* |
| *NetNamedPipeBinding* | *Communication between WCF applications on same computer. Supports duplex contracts and transactions* |
| *NetPeerTcpBinding* | *Communication between computers across peer-to-peer services. Supports duplex contracts* |
| *NetTcpBinding* | *Communication between WCF applications across computers. Supports duplex contracts and transactions* |

## *Hosting a WCF service*

1. *Hosting in a Managed Application/ Self Hosting*

* *Console Application*
* *Windows Application*
* *Windows Service*

1. *Hosting on Web Server*
   * *IIS 6.0 (ASP.NET Application supports only HTTP)*
   * *Windows Process Activation Service (WAS) i.e. IIS 7.0 supports HTTP, TCP, NamedPipes, MSMQ*

## *Proxy*

***A proxy is a class by which a service client can interact with the service.***

## *Service-Oriented Architecture (SOA)*

*A service-oriented architecture (SOA) is an architectural pattern in which application components provide services to other components via a communications protocol, The communication can involve either simple data passing or it could involve two or more services coordinating some activity*

[*http://www.javaworld.com/article/2071889/soa/what-is-service-oriented-architecture.html*](http://www.javaworld.com/article/2071889/soa/what-is-service-oriented-architecture.html)

## *Web API*

***ASP.NET Web API is a framework that makes it easy to build HTTP services that reach a broad range of clients, including browsers and mobile devices.***

## *Advantages of using ASP.NET Web API?*

1. *Light weight*
2. *Only deal with message*
3. *Less configuration needed as compared to WCF*
4. *Inbuilt support for JSON*
5. *It is preferable for mobile devices as it is less weight and supports JSON message format.*
6. *Easy to access*
7. *Interoperable*
8. *Stateless*
9. *Scalable etc.*
10. *It works the HTTP way using standard HTTP verbs like GET, POST, PUT, DELETE etc for all CRUD operations.*
11. *Complete support for routing.*
12. *Response generated in JSON or XML format using MediaTypeFormatter.*
13. *It has the ability to be hosted in IIS as well as self-host outside of IIS.*
14. *Supports Model binding and Validation.*
15. *Support for OData.*
16. *Stateless communication*

## *New features in Web API 2.0*

*Attribute Routing*

*CORS - Cross Origin Resource Sharing*

*OWIN (Open Web Interface for .NET) self-hosting*

*IHttpActionResult*

*Web API OData*

## *WCF Vs ASP.NET Web API*

### *Web Service*

1. *It is based on SOAP and return data in XML form.*
2. *It support only HTTP protocol.*
3. *It is not open source but can be consumed by any client that understands xml.*
4. *It can be hosted only on IIS.*

### *WCF*

1. *It is also based on SOAP and return data in XML form.*
2. *It is the evolution of the web service(ASMX) and support various protocols like TCP, HTTP, HTTPS, Named Pipes, MSMQ.*
3. *The main issue with WCF is, its tedious and extensive configuration.*
4. *It is not open source but can be consumed by any client that understands xml.*
5. *It can be hosted with in the applicaion or on IIS or using window service.*

### *WCF Rest*

1. *To use WCF as WCF*[*Rest service*](http://kellabyte.com/2011/09/04/clarifying-rest/)*you have to enable webHttpBindings.*
2. *It support HTTP GET and POST verbs by [WebGet] and [WebInvoke] attributes respectively.*
3. *To enable other HTTP verbs you have to do some configuration in IIS to accept request of that particular verb on .svc files*
4. *Passing data through parameters using a WebGet needs configuration. The UriTemplate must be specified*
5. *It support XML, JSON and ATOM data format.*

### *Web API*

1. *This is the new framework for building HTTP services with easy and simple way.*
2. *Web API is open source an ideal platform for building REST-ful services over the .NET Framework.*
3. *Unlike WCF Rest service, it use the full featues of HTTP (like URIs, request/response headers, caching, versioning, various content formats)*
4. *It also supports the MVC features such as routing, controllers, action results, filter, model binders, IOC container or dependency injection, unit testing that makes it more simple and robust.*
5. *It can be hosted with in the application or on IIS.*
6. *It is light weight architecture and good for devices which have limited bandwidth like smart phones.*
7. *Responses are formatted by Web API’s MediaTypeFormatter into JSON, XML or whatever format you want to add as a MediaTypeFormatter.*

## *Action Results in Web API*

* *HttpResponseMessage*
* *IHttpActionResult*
* *Void*
* *Type of Entity that mean some other type*

***Note:****IHttpActionResult is introduced in ASP.NET Web API 2 and the remaining are introduced in earlier versions of Web API.*

## *Explain Web API Routing?*

*Routing is the mechanism of pattern matching as we have in MVC. These routes will get registered in Route Tables. Below is the sample route in Web API –*

*Routes.MapHttpRoute(  
 Name: "MyFirstWebAPIRoute",  
 routeTemplate: “api/{controller}/{id}  
 defaults: new { id = RouteParameter.Optional}  
};*

## *difference between SOAP and REST?*

|  |  |
| --- | --- |
| ***SOAP*** | ***REST*** |
| * *SOAP is a protocol through which two computer communicates by sharing XML document* * *SOAP permits only XML* * *SOAP based reads cannot be cached* * *SOAP is like custom desktop application, closely connected to the server* * *SOAP is slower than REST* * *It runs on HTTP but envelopes the message* | * *Rest is a service architecture and design for network-based software architectures* * *REST supports many different data formats* * *REST reads can be cached* * *A REST client is more like a browser; it knows how to standardized methods and an application has to fit inside it* * *REST is faster than SOAP* * *It uses the HTTP headers to hold meta information* |

## *SOA*

***SOA is architecture for building applications using reusable, interoperable services which have well defined business functionalities*** *and can be orchestrated to achieve a specific functionality by utilizing them together.*

***4 facts you should know about SOA Services***

*1.****SOA separates business functions into services (endpoints),*** *which are made accessible over a network in order to allow users to combine and reuse them in their applications.  
2.       The SOA services can be developed in different languages and OS’es as long as they follow the SOA principles.  
3.       Services are unassociated and loosely coupled units that do not directly rely on each other for their full functioning. Rather than services embedding calls to each other in their source code, they use defined protocols that describe how services pass and parse messages using description metadata.  
4.       Orchestration is a process where business functionality from various services are combined in a system fully aware of all available services and the associated metadata that defines these services and their characteristics.*

# *MVC*

## *Restful URL*

*A REST-full URI is a URI that identifies a domain resource rather than an application resource. URLs that do not contain a query string and instead contain only the path of the resource.*

## *VI\*\*\*\*Routing*

***Routing allows us a way to configure our application to accept a requested URL which actually doesn't map to physical files.*** *From the security perspective of the application, it's important because one can easily know the solution structure of the application.*

## ***Enable Attribute Routing***

*Adding a Routes.MapMvcAttributeRoutes() method to the RegisterRoutes() method of the RouteConfig.cs file.*

***MVC 6***

*ASP.NET MVC and Web API has been merged in to one.*

*Dependency injection is inbuilt and part of MVC.*

*Side by side - deploy the runtime and framework with your application*

*Everything packaged with NuGet, Including the .NET runtime itself.*

*New JSON based project structure.*

*No need to recompile for every change. Just hit save and refresh the browser.*

*Compilation done with the new Roslyn real-time compiler.*

*vNext is Open Source via the .NET Foundation and is taking public contributions.*

*vNext (and Rosyln) also runs on Mono, on both Mac and Linux today.*

***MVC 5***

*One ASP.NET*

*Attribute based routing*

*Asp.Net Identity*

*Bootstrap in the MVC template*

*Authentication Filters*

*Filter overrides*

***MVC 4***

*ASP.NET Web API*

*Refreshed and modernized default project templates*

*New mobile project template*

*Many new features to support mobile apps*

*Enhanced support for asynchronous methods*

## *Lifecycle of an MVC Page*

1. ***App Initialization***
2. ***Routing***
3. ***MVC Handler Executes***
4. ***Instantiation and execute controller***
5. ***Locate and invoke controller action***
6. ***View Result***
7. ***Instantiate and render view.***

## [*Execution flow in MVC*](http://stackoverflow.com/questions/1982517/execution-flow-in-mvc)

*Here are the detailed steps:*

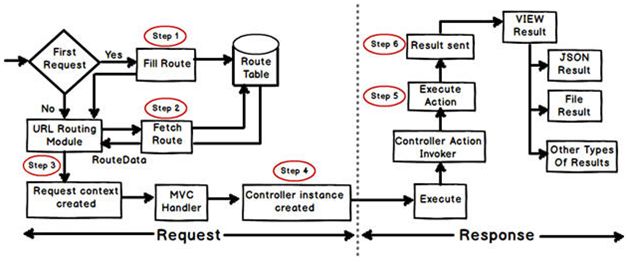
1. *Request comes into ASP.NET*
2. *ASP.NET Routing finds the route match by calling* [*RouteCollection.GetRouteData*](http://msdn.microsoft.com/en-us/library/system.web.routing.routecollection.getroutedata.aspx)
3. *This in turn calls* [*RouteBase.GetRouteData*](http://msdn.microsoft.com/en-us/library/system.web.routing.routebase.getroutedata.aspx) *on each route until it finds a match*
4. *The IRouteHandler for the matching route has its* [*GetHttpHandler*](http://msdn.microsoft.com/en-us/library/system.web.routing.iroutehandler.gethttphandler.aspx) *method called*
5. *The MvcHandler runs ([ProcessRequest](http://msdn.microsoft.com/en-us/library/system.web.mvc.mvchandler.processrequest.aspx) is called)*
6. *The MVC controller factory locates and creates the controller in* [*CreateController*](http://msdn.microsoft.com/en-us/library/system.web.mvc.icontrollerfactory.createcontroller.aspx)
7. *The ControllerActionInvoker determines which action to run in* [*InvokeAction*](http://msdn.microsoft.com/en-us/library/system.web.mvc.controlleractioninvoker.invokeaction.aspx)
8. *The* [*AuthorizationFilter*](http://msdn.microsoft.com/en-us/library/system.web.mvc.iauthorizationfilter.aspx) *stage executes (this includes the authorization method on the controller itself)*
9. *The* [*ActionExecuting*](http://msdn.microsoft.com/en-us/library/system.web.mvc.iactionfilter.aspx) *stage executes*
10. *The requested action method is executed*
11. *The ActionExecuted stage executes*
12. *If there is a result object then the* [*ResultExecuting*](http://msdn.microsoft.com/en-us/library/system.web.mvc.iresultfilter.onresultexecuting.aspx) *stage executes*
13. *If the result wasn't cancelled then the ActionResult's* [*ExecuteResult*](http://msdn.microsoft.com/en-us/library/system.web.mvc.actionresult.executeresult.aspx) *method is executed*
14. *The ResultExecuted stage executes*
15. *If an error occured then the* [*Exception*](http://msdn.microsoft.com/en-us/library/system.web.mvc.iexceptionfilter.aspx) *stage executes*

*I would also like to refer you to the* [*MVC Snake Diagram*](http://weblogs.asp.net/leftslipper/archive/2007/12/10/asp-net-mvc-design-philosophy.aspx) *that I use in many presentations on ASP.NET MVC. Here's the full image:*

*The blog post I linked to describes some of the concepts used in ASP.NET MVC regarding how data flows through the application*

[*http://dotnetmentors.com/mvc/asp-net-mvc-application-request-life-cycle.aspx*](http://dotnetmentors.com/mvc/asp-net-mvc-application-request-life-cycle.aspx)

[*http://www.codeproject.com/Articles/1028156/A-Detailed-Walkthrough-of-ASP-NET-MVC-Request-Life*](http://www.codeproject.com/Articles/1028156/A-Detailed-Walkthrough-of-ASP-NET-MVC-Request-Life)

***Question 3: Explain MVC application life cycle?  
  
Answer:****Any web application has two main execution steps, first understanding the request and depending on the type of the request sending out appropriate response. 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. The following is the detailed explanation of the same.****Step 1: 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.****Step 2: 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.****Step 3: Request context created****The “RouteData” object is used to create the “RequestContext” object.****Step 4: 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.  
  
*

* [*ASP.Net MVC Life Cycle*](http://www.c-sharpcorner.com/UploadFile/00a8b7/Asp-Net-mvc-life-cycle/)

## *Model:*

*Model represents the application data domain. In short the applications business logic is contained within the model.*

## *View:*

***Views represent the user interface, with which the end users interact. In short all the user interface logic is contained within the UI.***

## *Controller:*

***Controller is the component that responds to user actions****. Based on the user actions, the respective controller, work with the model, and selects a view to render that displays the user interface. The user input logic is contained within the controller.*

## *VIQ\*\*\*View model*

***ViewModel is a class that contains the fields which are represented in the strongly-typed view. It is used to pass data from controller to strongly-typed view.***

1. *ViewModel contain fields that are represented in the view (for LabelFor,EditorFor,DisplayFor helpers)*
2. *ViewModel can have specific validation rules using data annotations or IDataErrorInfo.*
3. *ViewModel can have multiple entities or objects from different data models or data source.*

***A view model is a simple class which represents data to be displayed on the view.***

***The key thing to remember is that the view model only represents the data that you want to use***

## *Razor*

***Razor is a general-purpose mark-up syntax templating engine for embedding server based code into web pages. The Razor syntax consists of Razor mark-up, C# and HTML. Files containing Razor generally have a*.cshtml*file extension.***

## *Partial Views*

*A partial view is a view that is rendered within another view.*

*If you want to load the partial view directly inside the main view you could use the Html.Action helper:*

*@Html.Action("Load", "Home")*

*We can call the partial view in a normal view like:*

*Html.RenderPartial("~/Views/Shared/\_Product.cshtml", product);*

*Or*

*@Html.Partial("~/Views/Shared/\_Product.cshtml", product);*

*Html.Partial returns a string, Html.RenderPartial calls Write internally, and returns void. You can store the output of Html.Partial in a variable, or return it from a function. You cannot do this withHtml.RenderPartial because the result will be written to the Response stream during execution. So @html.RenderPartial() has faster execution than @html.Partial() due to RenderPartial giving quick response to the output.*

*// Uses a view in current folder with this name*

*// If none is found, searches the Shared folder*

*@Html.Partial("ViewName")*

*// A view with this name must be in the same folder*

*@Html.Partial("ViewName.cshtml")*

*// Locate the view based on the application root*

*// Paths that start with "/" or "~/" refer to the application root*

*@Html.Partial("~/Views/Folder/ViewName.cshtml")*

*@Html.Partial("/Views/Folder/ViewName.cshtml")*

*// Locate the view using relative paths*

*@Html.Partial("../Account/LoginPartial.cshtml")*

*HTML helper has two methods for rendering the partial view: Partial and RenderPartial.*

1. *<div>*
2. *@Html.Partial("PartialViewExample")*
3. *</div>*
4. *<div>*
5. *@{*
6. *Html.RenderPartial("PartialViewExample");*
7. *}*
8. *</div>*

***@Html.RenderPartial*** *The result of the RenderPartial method is written directly into the HTTP response, it means that this method used the same TextWriter object as used by the current view. This method returns nothing.****@Html.Partial*** *This method renders the view as an HTML-encoded string. We can store the method result in a string variable.  
   
The Html.RenderPartial method writes output directly to the HTTP response stream so it is slightly faster than the Html.Partial method.  
   
Returning a Partial view from the Controller's Action method:*

1. ***public****ActionResult PartialViewExample()*
2. *{*
3. ***return****PartialView();*
4. *}*

***Render Partial View Using jQuery*** *Sometimes we need to load a partial view within a model popup at runtime, in this case we can render the partial view using JQuery element's load method.*

1. *<script type="text/jscript">*
2. *$('#partialView').load('/shared/PartialViewExample’);*
3. *</script>*

***View Vs Partial View***

|  |  |
| --- | --- |
| ***View*** | ***Partial View*** |
| *View contains the layout page* | *Partial view does not contain the layout page* |
| *\_viewstart page is rendered before any view is rendered* | *Partial view does not check for a \_viewstart.cshtml. We cannot place any common code for a partial view within the \_viewStart.cshtml page.* |
| *View may have markup tags like html, body, head, title, meta etc.* | *The Partial view is specially designed to render within the view and as a result it does not contain any mark up.* |
| *Partial view is more lightweight than the view. We can also pass a regular view to the RenderPartial method.* | |
| *If there is no layout page specified in the view, it can be considered as a partial view. In razor, there is no distinction between views and partial views as in the ASPX view engine (aspx and ascx).* | |

## *Scaffolding:*

***Scaffolding is a technique in which the MVC template helps to auto-generate CRUD code.***

## *HTML helpers*

*HTML helpers are methods we can invoke on the Html property of a view. HTML helpers help you to render HTML controls in the view.*

## *Action method:*

*All the public methods of a Controller class are called Action methods. They are like any other normal methods with the following restrictions:*

1. *Action method must be public. It cannot be private or protected*
2. *Action method cannot be overloaded*
3. *Action method cannot be a static method.*

[*http://www.tutorialsteacher.com/mvc/action-method-in-mvc*](http://www.tutorialsteacher.com/mvc/action-method-in-mvc)

## *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.*

## *Types of results*

1. *ContentResult*
2. *EmptyResult*
3. *FileResult*
   1. *FileContentResult*
   2. *FilePathResult*
   3. *FileStreamResult*
4. *HttpUnauthorizedResult*
   1. *HttpUnauthorizedResult*
   2. *HttpNotFoundResult*
5. *JavaScriptResult*
6. *JsonResult*
7. *RedirectResult*
8. *RedirectToRouteResult*
9. *ViewResultBase*
10. *PartialViewResult*
11. *ViewResult*

## *Areas:*

*Areas are logical grouping of Controller, Models and Views and other related folders for a module in MVC applications*

## *Sections:*

*Layout pages, can define sections, which can then be overriden by specific views making use of the layout. Defining and overriding sections is optional.*

## *NonActionAttribute*

*In general, all public methods of a controller class are treated as action methods. If you want prevent this default behavior, just decorate the public method with NonActionAttribute.*

## *\*\*Action Filters:*

***Action Filters are custom attributes that allow us to add pre-action and post-action behaviour to controller action methods***

* *Authentication filters*
* *Authorization filters*
* *Action filters*
* *Response filters*
* *Exception filters*

## *ViewData*

*ViewData is a dictionary, which will contains data to be passed between controller and views. Controller will add items to this dictionary and view reads from it.*

***ViewData VS ViewBag VS TempData***

|  |  |  |
| --- | --- | --- |
| ***ViewData*** | ***ViewBag*** | ***TempData*** |
| *It is Key-Value Dictionary collection* | *It is a type object* | *It is Key-Value Dictionary collection* |
| *ViewData is a dictionary object and it is property of ControllerBase class* | *ViewBag is Dynamic property of ControllerBase class.* | *TempData is a dictionary object and it is property of controllerBase class.* |
| *ViewData is Faster than ViewBag* | *ViewBag is slower than ViewData* | *NA* |
| *ViewData is introduced in MVC 1.0 and available in MVC 1.0 and above* | *ViewBag is introduced in MVC 3.0 and available in MVC 3.0 and above* | *TempData is also introduced in MVC1.0 and available in MVC 1.0 and above.* |
| *ViewData  is also work with .net framework 3.5 and above* | *ViewBag  is only  work with .net framework 4.0 and above* | *TempData  is also work with .net framework 3.5 and above* |
| *Type Conversion code is required while enumerating* | *In depth, ViewBag is used dynamic, so there is no need to type conversion while enumerating.* | *Type Conversion code is required while enumerating* |
| *It value become null if redirection is occurred.* | *Same as ViewData* | *TempData is used to pass data between two consecutive requests.* |
| *It lies only during the current request.* | *Same as ViewData* | *TempData is only work during the current and subsequent request* |

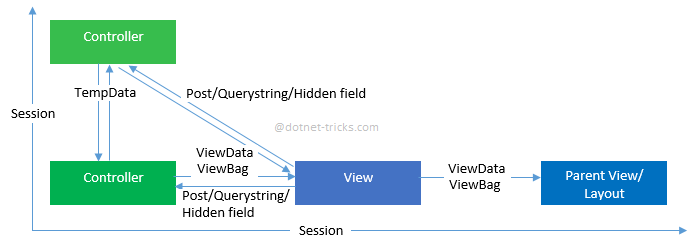
*ASP.NET MVC there are three ways - ViewData, ViewBag and TempData to pass data from controller to view and in next request. Like WebForm, you can also use Session to persist data during a user session. Now question is that when to use ViewData, VieBag, TempData and Session. Each of them has its own importance. In this article, I am trying to explain the differences among these four.*

## ***ViewData***

1. *ViewData is a dictionary object that is derived from ViewDataDictionary class.*
   1. ***public ViewDataDictionary ViewData { get; set; }***
2. *ViewData is a property of ControllerBase class.*
3. *ViewData is used to pass data from controller to corresponding view.*
4. *It’s life lies only during the current request.*
5. *If redirection occurs then it’s value becomes null.*
6. *It’s required typecasting for getting data and check for null values to avoid error.*

## ***ViewBag***

1. *ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0.*
2. *Basically it is a wrapper around the ViewData and also used to pass data from controller to corresponding view.*
   1. ***public Object ViewBag { get; }***
3. *ViewBag is a property of ControllerBase class.*
4. *It’s life also lies only during the current request.*
5. *If redirection occurs then it’s value becomes null.*
6. *It doesn’t required typecasting for getting data.*

**

## ***TempData***

1. *TempData is a dictionary object that is derived from TempDataDictionary class and stored in short lives session.*
   1. ***public TempDataDictionary TempData { get; set; }***
2. *TempData is a property of ControllerBase class.*
3. *TempData is used to pass data from current request to subsequent request (means redirecting from one page to another).*
4. *It’s life is very short and lies only till the target view is fully loaded.*
5. *It’s required typecasting for getting data and check for null values to avoid error.*
6. *It is used to store only one time messages like error messages, validation messages. To persist data with TempData refer this article:*[*Persisting Data with TempData*](http://www.dotnet-tricks.com/Tutorial/mvc/P81S250214-Persisting-Data-with-TempData.html)

*Session*

*1. In ASP.NET MVC, Session is a property of Controller class whose type is HttpSessionStateBase.*

*1. public HttpSessionStateBase Session { get; }*

*2. Session is also used to pass data within the ASP.NET MVC application and Unlike TempData, it persists for its expiration time (by default session expiration time is 20 minutes but it can be increased).*

*3. Session is valid for all requests, not for a single redirect.*

*4. It’s also required typecasting for getting data and check for null values to avoid error.*

## ***Comparison: ViewModel vs ViewData vs ViewBag vs TempData vs Session***

| **#** | **ViewModel** | **ViewData** | **ViewBag** | **TempData** | **Session** |
| --- | --- | --- | --- | --- | --- |
| *1* | *Is a class. It is a model specific for rendering a view.* | *Is a key-value dictionary derived from ViewDataDictionary.* | *Is a Dynamic property. It is a wrapper around ViewData.* | *Is a key-value dictionary derived from TempDataDictionary.* | *Is a key-value dictionary derived from TempDataDictionary.* |
| *2* | *Strongly typed class. So, no need for type-casting.* | *Un-typed. So, needs type-casting for complex data.* | *Type casting is not required.* | *Un-typed: Needs type-casting for complex data type.* | *Un-typed: Needs type-casting and null checking.* |
| *3* | *Represents only the data from a model required for rendering the view.* | *Used to pass data between controller and view.* | *Used to pass data between controller and view.* | *Used to pass data between requests. I.e. it helps to pass data from one controller to another controller.* | *Used to store a small amount of data for the duration of the user visiting the website.* |
| *4* | *The lifespan is only for the current request.* | *The lifespan is only for the current request.* | *The lifespan is only for the current request.* | *Lifespan is for the current and subsequent request. The lifespan of TempData can be increased beyond the first redirection using TempData.Keep() method.* | *lifespan of a session persists till it is forcefully destroyed by the server or the user.* |
| *5* | *On redirection the ViewModel object will be destroyed.* | *On redirection, the value in the ViewData becomes Null.* | *On redirection, the value in the ViewData becomes Null.* | *The data stored in TempData persists only during redirection.* | *The data stored in Session persists during any number of redirection.* |
| *6* | *Provides compile-time error checking and Intellisense support.* | *Does not provide compile-time error checking.* | *Does not provide compile-time error checking.* | *Does not provide compile-time error checking.* | *Does not provide compile-time error checking.* |
| *7* | ***Usage:*** *(a) Display data in a table with master child relationship. (b) Reports with aggregate and summary. (c) Paging data in a table. (d) Flexible with very simple as well as highly complex data from multiple data-source.* | ***Usage:*** *(a) To pass a list of data to render a drop down list. (b) To pass small amount of data to be rendered in the view. (c) Not ideal for complex data involving multiple data sources.* | ***Usage:*** *(a) To pass a list of data to render a drop down list. (b) To pass small amount of data to be rendered in the view. (c) Not ideal for complex data involving multiple data sources.* | ***Usage:*** *(a) Useful for storing one time messages like error message and validation messages. (b) Used in scenarios to pass small data from one action to another action or one controller call to another controller call.* | ***Usage:*** *(a) To check whether the user is logged in to the website. (b) To store the user’s permission information.* |

*Conclusion*

*In my opinion:*

1. *Use****ViewModel****instead of ViewData and ViewBag. As ViewModel provides compile time error checking and intellisense support in Visual Studio IDE, the code will be clean and less error prone.*
2. *Use****TempData****for error handling. On capturing and exception or error, pass the error details using TempData to the error page, so as to render the appropriate error details.*
3. *Use****Session****to check the user’s login status and permission level.*

## *Keep and peek in tempdata*

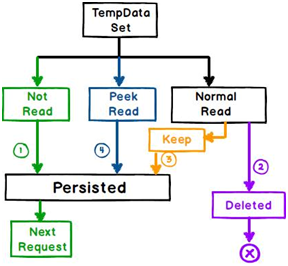
* ***If you set value for TempData and do not read the value then the data will be available for next request.***
* ***If you set value for TempData and read in View then the data will be deleted or will be null.***
* ***If you read TempData in the first request and want to keep the value for the next request then use 'Keep' Method.***

*@TempData[“MyData”];*

*TempData.Keep(“MyData”);*

* ***If you read the TempData using 'Peek' then value persists for the next request also.***

*stringstr = TempData.Peek("Td").ToString();*

**

## *View engine*

*View engine is the one that works between your View and browser to provide valid HTML output to your browser by compiling the code inside your View. There are many view engines available and some of them are following:*

1. *ASPX*
2. *Razor*
3. *Spark*
4. *NHaml*
5. *NDJango*
6. *Hasic*
7. *Brail*
8. *Bellevue*
9. *Sharp Tiles*
10. *String Template*
11. *Wing Beats*
12. *SharpDOM*

# *Angulur Js | JQuery*

*AngularJS is a JavaScript framework which simplifies binding JavaScript objects with HTML UI elements.*

## *Directives*

***Directives are attributes decorated on the HTML elements, specific to your application. They are essentially functions that execute when the Angular compiler finds them in the DOM.*** *All directives start with the word “ng”. As the name says directive it directs Angular what to do.*

* *Element directives*  E
* *Attribute directives* A
* *CSS class directives* C
* *Comment directives* M
* *interpolation directive{{}}*

| **Type** | **Usage** |
| --- | --- |
| A | <div **book**></div> |
| C | <div class="**book**"></div> |
| E | <**book**data="book\_data"></**book**> |
| M | <!--directive:**book** --> |

***So*** it can be A(**A**ttribute),C(**C**lass), E (**E**lement), andM(co**M**ment).

*The ng-bind directive is an alternative to the interpolation directive{{}}*

## *Modules*

*A module is a collection of controllers, directives, services, filters, and configuration information. angular.module is used to configure the $injector.*

*myApp.js. var app = angular.module("myApp", []); The [] parameter in the module definition can be used to define dependent modules.*

## *Controllers*

***Controllers are JavaScript*** *constructor* ***functions*** *attached to the DOM using ng-controller**directive* ***which provide data and logic to HTML UI****. And is acts as a repository for the actions to be the DOM.*

*When the controller is attached with DOM a new injectable scope parameter will be available to the controller’s constructor function of the name $scope****.***

*Controller is a repository for the actions to be exposed to the DOM.*

*The controller() function call is what registers the controller function itself. The first parameter passed to the controller() function is the name of the controller function. This is the name you refer to in the ng-controller attribute of the view. The second parameter is the controller function itself.*

*We only use $scope inside Controllers, where we bind data from the Controller to the View.*

## *Filters?*

*Filters are used to change modify the data and can be clubbed in expression or directives using pipe character.*

## *Dependency injection*

***Dependency injection is a process where we inject the dependent objects rather than consumer creating the objects.***

## *Interceptors*

*The interceptors are service factories that are registered with the $httpProvider For purposes of global error handling, authentication, or any kind of synchronous or asynchronous pre-processing of request or post processing of responses.*

## *Injector*

***An injector is a service locator, used to retrieve object instance as defined by provider****, instantiate types, invoke methods, and load modules.*

## *Provider*

***A provider is an object with a $get() method. The injector calls the $get method to create a new instance of a service. The Provider can have additional methods which would allow for configuration of the provider***

## *Service:*

*.service() is a method on our module that takes a name and a function that defines the service.*

***Angular service is a singleton, substitutable, lazily and only once instantiated JavaScript injectable constructor which contains a set of functions which act as a repository for the code to be shared across the application.*** *You can use services to organize and share code across your app.*

## *Factory:*

***Factory is an injectable function which allows you to add some logic before creating the object. Once created, the value is reused for all services, controllers etc.***

## *Difference between Service and Factory:*

|  |  |
| --- | --- |
| ***Factory*** | ***Services*** |
| *A factory is an injectable function* | *A service is an injectable constructor.* |
| *It returns the created object.* | *It doesn’t return anything.* |
| *When declaring factoryName as an injectable argument****you will be provided the value that is returned by invoking the function reference****passed to module.factory.* | *When declaring serviceName as an injectable argument****you will be provided the actual function reference****passed to module.service* |

[*http://www.dotnet-tricks.com/Tutorial/angularjs/V2YS090914-Understanding-AngularJS-Factory,-Service-and-Provider.html*](http://www.dotnet-tricks.com/Tutorial/angularjs/V2YS090914-Understanding-AngularJS-Factory,-Service-and-Provider.html)

## *$rootscope*

***Every application has a single root scope****. All other scopes are descendant scopes of the root scope. Scopes provide separation between the model and the view, via a mechanism for watching the model for changes.*

***“$rootScope” is a parent object of all “$scope”*** *angular objects created in a web page.*

## *$scope:*

***“$scope” is a model object instance of a controller****. “$scope” object instance gets created when “ng-controller” directive is encountered. It acts like glue between application controller and the view. It can watch expressions and propagate events.*

*Scopes provide separation between the model and the view, via a mechanism for watching the model for changes.*

* *The $scope parameter is the model object to be used by the controller function and the corresponding view. The controller function can insert data and functions into the model object. The view can then use the data and the functions.*

## Isolate scope

isolate scope is a separate scope object tied to the directive.

## *$broadcast or $emit*

*$emit dispatches an event****upwards****through the scope hierarchy.*

*$broadcast dispatches an event****downwards****to all child scopes.*

## *Compile*

*"compilation" means attaching directives to the HTML to make it interactive.*

## *Link*

***Link function is the function that you can use to perform directive-specific operations.***

***In the link phase the data i.e. ($scope) is attached to the template function and executed to get the final HTML output***

The link function is mainly used for attaching event listeners to DOM elements, watching model properties for changes, and updating the DOM.

## *Template*

*The template consists of HTML, CSS, and Angular directives contained in just one HTML file.*

## *Bootstrapping*

***Bootstrapping in AngularJS is nothing but initializing, or starting the Angular app****. AngularJS supports automatic and manual bootstrapping.*

## *Digest cycle:*

***It is a simple loop which updates the model and view.***

## *Watchers:*

***They are listeners which are attached to expression and angular directives and fire when the model data changes.***

*The $scope.watch() function creates a watch of some variable. When you register a watch you pass two functions as parameters to the $watch() function:*

* *A value function*
* *A listener function*

*Here is an example:*

*$scope.$watch(function() {},*

*function() {}*

*);*

*$scope.$watch(function(scope) { return scope.data.myVar },*

*function(newValue, oldValue) {*

*document.getElementById("").innerHTML =*

*"****" + newValue + "****";*

*}*

*);*

## *Dirty check:*

***This is an extra digest loop which runs to check any cascading left over updates due to the first digest cycle.***

## *$apply*

*The $scope.$apply() function is used to execute some code*

*The $apply() function is useful when integrating AngularJS with other code.*

## *$apply()*

*The $scope.$apply() function takes a function as parameter which is executed, and after that$scope.$digest() is called internally. That makes it easier for you to make sure that all watches are checked, and thus all data bindings refreshed. Here is an $apply() example:*

*$scope.$apply(function() {*

*$scope.data.myVar = "Another value";*

*});*

*The function passed to the $apply() function as parameter will change the value of $scope.data.myVar. When the function exits AngularJS will call the $scope.$digest() function so all watches are checked for changes in the watched values.*

* 1. [*$apply()*](https://docs.angularjs.org/api/ng/type/$rootScope.Scope#$apply)
  2. [*$timeout()*](https://docs.angularjs.org/api/ng/service/$timeout)*----$timeout service is used to call another JavaScript function after a given time delay. The $timeout service only schedules a single call to the function. The $timeout() process is the regular JavaScript setTimeout function which by default, at the end of it, triggers $apply(). You can also instruct $timeout() not to invoke the $apply() function.*

*3.*[*$digest()*](https://docs.angularjs.org/api/ng/type/$rootScope.Scope#$digest) *4.*[*$evalAsync()*](https://docs.angularjs.org/api/ng/type/$rootScope.Scope#$evalAsync)*---This function will evaluate the expression during the current cycle or the next.*

* *if code is queued using****$evalAsync from a directive****, it should run after the DOM has been manipulated by Angular, but before the browser renders*
* *if code is queued using****$evalAsync from a controller****, it should run before the DOM has been manipulated by Angular (and before the browser renders) -- rarely do you want this*
* *if code is queued using****$timeout****, it should run after the DOM has been manipulated by Angular, and after the browser renders (which may cause flicker in some cases)*

[***http://www.codingeek.com/angularjs/angular-js-apply-timeout-digest-evalasync/***](http://www.codingeek.com/angularjs/angular-js-apply-timeout-digest-evalasync/)

## *Angular expressions*

*Angular expressions are unit of code which resolves to value. This code is written inside curly braces “{{}} “.*

*The main difference between ng-if and ng-show + ng-hide is that ng-if removes the HTML element completely from the DOM, whereas the ng-show + ng-hide just applies the CSS property display: none; to the elements.*

## *AngularJS Event Listener Directives*

*You attach an event listener to an HTML element using one of the AngularJS event listener directives:*

* *ng-click*
* *ng-dbl-click*
* *ng-mousedown*
* *ng-mouseup*
* *ng-mouseenter*
* *ng-mouseleave*
* *ng-mousemove*
* *ng-mouseover*
* *ng-keydown*
* *ng-keyup*
* *ng-keypress*
* *ng-change*

*The event listener directives have a special variable named $event which you can use as parameter to the event listener function. The $event variable contains the original browser event object. You can also pass other parameters to your event listener functions.*

## ***$q service, deferred and promises?***

*A* ***promise*** *is an assurance that we will get a result from an action at some point in the future. It is a placeholder into which the successful result value or reason for failure will materialize.*

*While* ***deferred*** *helps to control how and when those promise logics will execute.*

*We can think about promises as “WHAT” we want to fire after an operation is completed while deferred controls “WHEN” and “HOW” those promises will execute.*

*For example after an operation is complete you want to a send a mail, log in to log file and so on. So these operations you will define using promise. And these promise logics will be controlled by deferred.*

*So once some action completes deferred gives a signal “Resolve”, “Reject” or “Notify” and depending on what kind of signal is sent the appropriate promise logic chain fires.*

***“$q”*** *is the angular service which provides promises and deferred functionality.*

*Using promises, deferred and “q” service is a 4 step process:-*

*Step 1:- Get the “q” service injected from Angular.*

*Step 2 :- Get deferred object from “q” service object.*

*Step 3 :- Get Promise object from deferred object.*

*Step 4 :- Add logics to the promise object.*

## *Filters*

*Filters format the value of an expression for display to the user. They can be used in view templates, controllers or services. Angular comes with a collection of*[*built-in filters*](https://docs.angularjs.org/api/ng/filter)*, but it is easy to define your own as well.*

## [TypeScript](http://www.typescriptlang.org/)

 is a superset of JavaScript which primarily provides optional static typing, classes and interfaces. One of the big benefits is to enable IDEs to provide a richer environment for spotting common errors as you type the code.

# *SQL Server + LinQ + Entity Framework*

## *Normalization*

*The process of removing redundant information from your records and shifting it to separate table is called Normalization.*

## *M I\*\*\*\*ACID Functionality*

***A transaction must be:  
1.       Atomic - it is one unit of work and does not dependent on previous and following transactions.  
2.       Consistent - data is either committed or roll back, no “in-between” case where something has been updated and something hasn’t.  
3.       Isolated - no transaction sees the intermediate results of the current transaction).  
4.       Durable - the values persist if the data had been committed even if the system crashes right after.***

## *Relation*

*A relation is a table that holds the data we are interested in. It is two-dimensional and has rows and columns.*

## *Referential integrity*

***Referential integrity is a database concept that ensures that relationships between tables remain consistent****. When one table has a foreign key to another table, the concept of referential integrity states that you may not add a record to the table that contains the foreign key unless there is a corresponding record in the linked table.*

## *Wildcard character*

*Wildcard characters Determines whether a given character string matches a specified pattern.*

*There are 2 Wildcard characters i.e. '%' and '\_'.*

## *Transaction*

*Transaction is a process or a set of more than one process. If a transaction is a set of more than one process, then for successful completion of that transaction, it is necessary that all processes must execute successfully in that transaction.*

## *Dataset*

*A dataset is an in-memory representation of a database-like structure. It can have one or more data tables and define relations between them,* [*key field*](http://en.wikipedia.org/wiki/Key_field) *etc.*

## *View*

*A simple view can be thought of as a subset of a table. It can be used for retrieving data, as well as updating or deleting rows.*

## *SQL-Injection*

*SQL injection is a type of security* [*exploit*](http://searchSecurity.techtarget.com/sDefinition/0,,sid14_gci553536,00.html) *in which the attacker adds* [*SQL*](http://searchSQLServer.techtarget.com/sDefinition/0,,sid87_gci214230,00.html) *code to a Web form input box to gain access to resources or make changes to data.*

## *JOIN*

*A JOIN is a relationship operation that produces a table by retrieving data from two tables and matches their rows according to join specification.*

### *Cross Join*

*A cross join that does not have a WHERE clause produces the Cartesian product of the tables involved in the join. The common example is when company wants to combine each product with a pricing table to analyses each product at each price.*

### *Inner Join*

*A join that displays only the rows that have a match in both joined tables is known as inner Join.*

### *Outer Join*

*A join that includes rows even if they do not have related rows in the joined table is an Outer Join.*

#### Left Outer Join:

*In Left Outer Join all rows in the first-named table i.e. “left” table, which appears leftmost in the JOIN clause are included. Unmatched rows in the right table do not appear.*

#### Right Outer Join:

*In Right Outer Join all rows in the second-named table i.e. “right” table, which appears rightmost in the JOIN clause are included. Unmatched rows in the left table are not included.*

#### Full Outer Join:

*In Full Outer Join all rows in all joined tables are included, whether they are matched or not.*

### *Self Join*

*This is a particular case when one table joins to itself, with one or two aliases to avoid confusion. A self join can be of any type, as long as the joined tables are the same.*

*The common example is when company has a hierarchal reporting structure whereby one member of staff reports to another. Self Join can be Outer Join or Inner Join. (*[*Read More Here*](http://blog.sqlauthority.com/2007/06/03/sql-server-2005-explanation-and-example-self-join/)*)*

## *Candidate Key*

***A Candidate Key can be any column or a combination of columns that can qualify as unique key in database***

*A****candidate key****is a column, or set of columns, in a table that can uniquely identify any database record without referring to any other data. Each table may have one or more****candidate keys****, but one****candidate key****is special, and it is called the primary* ***key****.*

## *Unique Key*

*A UNIQUE constraint enforces the uniqueness of the values in a set of columns, so no duplicate values are entered. The unique key constraints are used to enforce entity integrity as the primary key constraints.*

## *Foreign Key*

*A foreign key is an attribute (or group of attributes) that is the primary key to another relation.*

*You cannot truncate a table which has an FK constraint on it.*

## *Composite Key*

***A composite key is a primary key that consists of more than two column****. Composite keys are also known as concatenated or aggregate keys.*

***A key makes by combining at least two or more columns is called composite key.***

## *Alternate Key*

***An alternate key is any candidate key that is not the primary key. Alternate keys are sometimes referred to as secondary keys.***

## *Indexes*

*An index is a physical structure containing pointers to the data. Indices are created in an existing table to locate rows more quickly and efficiently.*

## *Indexing:*

*Arranging all data in a table in a sequential manner and creating indices on each row to locate rows more quickly and efficiently.*

## *Clustered Index*

***Clustered index is an Index that physically arranges all data in a table in a sequential manner.***

## *Non-clustered index*

***A non-clustered index is a special type of index in which the logical order of the index does not match the physical stored order of the rows on disk. The leaf node of a non-clustered index does not consist of the data pages. Instead, the leaf nodes contain index rows.***

## *VIQ\*\*\*\*Covered index*

*A covering index is one which can satisfy all requested columns in a query without performing a further lookup into the clustered index.*

## Hints

* **Query Hints**tell the optimizer to apply this hint throughout the execution of the entire query.
* **Join Hints** tell the optimizer to use a particular join at a particular point in the query
* **Table Hints**control table scans and the use of a particular index for a table

## [**Table Hints**](http://stackoverflow.com/questions/8137431/what-is-the-purpose-of-table-hints-in-sql-server)

Table hints enable you to specifically control how the optimizer “uses” a particular table when generating an execution plan.

## *Cursor*

*A cursor is a set of rows together with a pointer that identifies a current row.*

*In other word, Cursor is a database object used by applications to manipulate data in a set on a row-by-row basis.*

## *Triggers*

*A Trigger is a block of code, fired whenever data in the underlying table is affected by any of the Data Manipulation Language (DML) statements -INSERT,UPDATE,OR DELETE.*

## *Instead of trigger*

*Instead of trigger is used to update the database tables associated with the view instead of updating the view directly.*

*You can define only one “Instead Of” Trigger for one table or one view.*

*INSTEAD OF triggers fire before an insert, update, or delete operation. INSTEAD OF triggers replace the original operation*

*If an INSTEAD OF trigger exists for a table, AFTER triggers that might also exist on the table are never fired.*

## *View*

*A simple view can be thought of as a subset of a table. It can be used for retrieving data, as well as updating or deleting rows.*

## *VIQ\*\*\*\*Difference between Primary Key and Unique Key*

* *Primary key does not allow null value but unique key allows null value.*
* *We can declare only one primary key in a table but a table can have multiple unique key (column assign).*
* *Clustered index is created on Primary key constraint and a non-clustered unique index is created on unique key constraint.*

## *Magic tables*

*There are two virtual tables Inserted and deleted in the SQL Server, which are popularly known as the Magic tables. These tables contain the information about inserted rows, deleted rows and the updated rows.*

## *@@FETCH\_STATUS*

*@@FETCH\_STATUS is a system function that tells you if you’re Fetch was successful. 0 indicates success and -1 indicates failure.*

## *RAISEERROR*

*RAISERROR allows developers to produce our own error message. Using RAISERROR, we can throw our own error message while running our Query or Stored procedure.*

*Raise Error generates an error message and initiates error processing for the session.*

## *Aggregate function*

***Aggregate functions perform a calculation on a set of values and return a single value.***

## *UNION and UNION ALL*

*UNION only selects distinct values, UNION ALL selects all values*

***UNION filter the table before the result set, while Union All does the same but in the end***

*The UNION ALL statement is much faster than UNION, because UNION ALL statement does not look for duplicate rows, and UNION statement does look for duplicate rows, whether or not they exist.*

## *Lambda expression*

*A lambda expression is a function without a name that calculates and returns a single value. All lambda expressions use the lambda operator =>, which read as goes to. The left side of the lambda operator specifies the input parameters and the right side holds the expression or statement block.*

## *VIQ\*\*\*\*Difference between ‘group by’ and ‘Order by’*

***The GROUP BY statement is used in conjunction with the aggregate functions to group the result-set by one or more columns. (Such as SUM, COUNT, AVG, etc.).***

*Group By: Grouping of Similar records in a Column.*

*Order By: Sorting List of Records in a Column(s).*

## *VIQ\*\*\*\*Difference between ‘Having’ and ‘Where’*

***HAVING specifies a search condition for a group or an aggregate function and only used in SELECT statement with GROUP BY clause. When GROUP BY is not used, HAVING behaves like a WHERE clause.***

*HAVING can be used only with the SELECT statement. Where Clause can be used other than Select statement also*

*HAVING applies only to groups as a whole. WHERE clause applies to individual rows.*

*WHERE clause is applied first to the individual rows in the tables. The HAVING clause is then applied to the rows in the result set.*

## *VIQ\*\*\*\*Different between Stored Procedure and Function*

1. ***Stored procedure is precompiled execution plan, whereas functions are not.***
2. ***Function must return a value, but SP may or not returns value.***
3. ***Function returns one value only. Procedure can return multiple values (max 1024).***
4. ***Functions cannot return values of type text, ntext, image & timestamps, whereas procedures can.***
5. ***Function doesn’t support output parameter. But SP have both input and output parameter.***
6. ***Function doesn’t support try-catch block, but SP does.***
7. ***Function doesn’t support Transection, but SP does.***
8. ***Function doesn’t support Temporary Table and allows variable Tables, but SP supports both.***
9. ***Function doesn’t allow to use DML statements, but SP does.***
10. ***Functions are used in Join Clause, but SP can’t be used on Join Clause.***
11. ***Function can be used in the SQL Queries while a procedure cannot be used in SQL queries .that cause a major difference b/w function and procedures.***
12. ***Stored procedure has the security and reduces the network traffic and we can call stored procedure in any no. of applications at a time.***
13. ***Functions are normally used for computations, whereas procedures are normally used for executing business logic.***
14. ***Stored procedure returns always integer value, by default zero. Whereas function returns type could be scalar or table or table values***

## *VIQ\*\*\*\*Difference between IQueryable<T> and IEnumerable<T> interface*

|  |  |
| --- | --- |
| ***IEnumerable*** | ***IQueryable*** |
| ***LINQ to object*** | ***LINQ to SQL*** |
| *System.Collections Namespace* | *System.Linq Namespace* |
| *No base interface* | *Derives from IEnumerable* |
| *While querying data from database, IEnumerable execute select query on server side, load data in-memory on client side and then filter data. Hence does more work and becomes slow.* | *While querying data from database, IQueryable execute select query on server side with all filters. Hence does less work and becomes fast.* |
| *LINQ to Object and LINQ to XML queries.* | *LINQ to SQL queries.* |
| *when querying data from in-memory collections like List, Array etc.* | *When querying data from out-memory (like remote database, service) collections.* |

*IEnumerable: - When we to deal with in process memory object collections, and loop through the collection objects.*

*IQueryable: - When we have to run ad-hoc queries against data source like LINQ to SQL Server, Entity Framework and other sources.*

*IList: - When we have to add, remove or refer to an item by index.*

*IEnumerable: IEnumerable<T> is used to iterate a read only collection. It has only one method GetEnumeartor() which allows you to iterate the read only collection using a foreach loop. It only iterates in the forward direction. It is a read only collection and even to find the number of objects in collection, we need to iterate through entire collection. We can't modify (Add or Remove) data IEnumerable bring ALL data from server to client then filter them, assume that you have a lot of records so IEnumerable puts overhead on your memory.*

*IQueryable<T> constructs the query using an Expression Tree. On the other hand in the IEnumerable <T> the query is constructed using delegates. Both IQueryable<T> and IEnumerable <T> support lazy loading of data from remote database servers.*

[*http://firstcrazydeveloper.com/Blogs/BlogView.html/84/difference-between-iqueryable-and-ienumerable-in-c*](http://firstcrazydeveloper.com/Blogs/BlogView.html/84/difference-between-iqueryable-and-ienumerable-in-c)

*In real life, if you are using a ORM like LINQ-to-SQL*

* *If you create an IQueryable, then the query may be converted to sql and run on the database server*
* *If you create an IEnumerable, then all rows will be pulled into memory as objects before running the query.*

*In both cases if you don't call a ToList() or ToArray() then query will be executed each time it is used, so, say, you have an IQueryable<T> and you fill 4 list boxes from it, then the query will be run against the database 4 times.*

*Also if you extent your query:*

*q.Where(x.name = "a").ToList()*

*Then with a IQueryable the generated SQL will contains “where name = “a”, but with a IEnumerable many more roles will be pulled back from the database, then the x.name = “a” check will be done by .NET.*

## *When to use IEnumerable<T>?*

*1. Working with the read only collection*

*2. Need to read the objects in forward direction only*

*3. Not concerned about thread safety*

*4. Want to iterate the collection’s objects using*

## *VIQ\*\*\*\*Top 10 steps to optimize data access in SQL Server*

1. *Apply proper indexing in the table columns in the database*
   * *Make sure that every table in your database has a primary key.*
   * *Create non-clustered indexes on columns which are:*
     1. *Frequently used in the search criteria*
     2. *Used to join other tables*
     3. *Used as foreign key fields*
     4. *Of having high selectivity (column which returns a low percentage (0-5%) of rows from a total number of rows on a particular value)*
     5. *Used in the ORDER BY clause*
     6. *Of type XML (primary and secondary indexes need to be created; more on this in the coming articles)*
2. *Create the appropriate covering indexes*
   * *Use the Database Tuning Advisor's help while creating covered index*
3. *Defragment indexes if fragmentation occurs*

*You can do this in two ways:*

* + ***Reorganize the fragmented indexes****: execute the following command to do this:*
    1. *ALTER INDEX ALL ON TableName REORGANIZE*
  + ***Rebuild indexes****: execute the following command to do this:*
    1. *ALTER INDEX ALL ON TableName REBUILD WITH (FILLFACTOR=90,ONLINE=ON)*

1. *Move TSQL code from the application into the database server*
2. *Identify inefficient TSQL, re-factor, and apply best practices*
   * *Some TSQL Best Practices*
     1. *Don't use "SELECT\*" in a SQL query*
     2. *Avoid unnecessary columns in the SELECT list and unnecessary tables in join conditions*
     3. *Do not use the COUNT() aggregate in a subquery to do an existence check*
     4. *Try to avoid joining between two types of columns*
     5. *Try to avoid deadlocks*
     6. *Write TSQL using "Set based approach" rather than "Procedural approach"*
     7. *Try to avoid dynamic SQL*
     8. *Try to avoid the use of temporary tables*
     9. *Instead of LIKE search, use full text search for searching textual data*
     10. *Try to use UNION to implement an "OR" operation*
     11. *Implement a lazy loading strategy for large objects*
3. *Apply some advanced indexing techniques*
   * *Implement computed columns and create an index on these*
   * *Create "Indexed Views"*
   * *Create indexes on User Defined Functions (UDF)*
   * *Create indexes on XML columns*
4. *Apply de-normalizations, use history tables and pre-calculated columns*
   * *De-normalization*
   * *History tables*
5. *Diagnose performance problems, and use SQL Profiler and the Performance Monitoring Tool effectively*
   * *Basic use of the SQL Profiler tool*
   * *Effective use of SQL Profiler to troubleshot performance related problems*
   * *Use the Performance Monitoring Tool (Perfmon) to diagnose performance problems*
6. *Organize file groups and files in the database*
7. *Apply partitioning in big fat tables*
   * *Horizontal partitioning*
   * *Vertical partitioning*
8. *(The bonus step): Better-manage DBMS objects, Use TSQL templates*

## VIQ\*\*\*\*Stored Procedures Optimization Tips

<http://solutioncenter.apexsql.com/monitor-sql-server-queries-find-poor-performers-sql-server-profiler/>

1. ***Include the SET NOCOUNT ON statement****.*
2. ***Call stored procedure using its fully qualified name.***
3. ***Consider returning the integer value as an RETURN****.*
4. ***Don't use the prefix "sp\_" in the stored procedure****.*
5. ***Use the sp\_executesql*** *stored procedure instead of the EXECUTE statement.*
6. ***Try to avoid using temporary tables inside your stored procedure****.*
7. ***Try to avoid using Data Definition Language statements inside your stored procedure****.*

## *Entity Framework Code First Migrations*

*Run the following command to configure migrations within your project and for creating new database.*

### *Enable migrations*

*Enable-Migrations*

### *Create migration*

*Add-Migration MigrationsName*

### *Create upgrade/downgrade script*

*Update-Database*

## *Undo/Rollback a Migrations*

*You can also Undo/Rollback a specific migrations by using following commands:*

### *Rollback to a specific migrations*

*Update-Database -TargetMigration:MigrationsName*

### *Rollback all migrations*

*Update-Database -TargetMigration:0*