



# .NET Interview Questions with Answers

For Face-to-Face classroom/offline Angular training in Mumbai: - [www.stepbystepschools.net](http://www.stepbystepschools.net)

For more step-by-step videos visit: - <https://www.questpond.com>

Also subscribe our YouTube Channel: - <https://youtube.com/questpondvideos>

Join QuestPond Telegram Channel: - <https://tinyurl.com/QuestPondChannel>

Foreword .....	8
Chapter 1: Top 50 technical and Non-technical questions .....	10
Can you explain architecture of your current project?.....	10
What role did you play in your project and company?.....	10
What's your salary expectation?.....	11
Why do you want to leave your current organization?.....	11
How much do you rate yourself between 1 to 10?.....	12
Can you speak about yourself? .....	12
How can we improve performance of .NET? .....	13
What is the difference between .NET 1.X, 2.0, 3.0,3.5,4.0 and 4.5?.....	13
What is ILcode,JIT,CLR, CTS, CLS and CAS?.....	13
What is a garbage collector? .....	14
What is GAC? .....	14
What are stacks, heaps,value types, reference types, boxing and unboxing?.....	14
What is the difference between Out and Ref? .....	15
How are exceptions handled in .NET? .....	17
What are different types of collections in .NET? .....	17
What are generics? .....	17
Explain Abstraction, encapsulation, inheritance and polymorphism?.....	18
How is abstract class different from aninterface?.....	18
What are the different types of polymorphism? .....	19
How does delegate differ from an event? .....	19
What are different access modifiers?.....	19
explain connection, command, datareader and dataset inADO.NET ?.....	20
How does "Dataset" differ from a "Data Reader"? .....	20
How is ASP.NET page life cycle executed?.....	21
What are Httphandlers and HttpModules and difference between them? .....	21
What are different kind of validator controls in ASP.NET ?.....	22
How is 'Server.Transfer' different from 'response.Redirect' ? .....	22
So when to use "Server.Transfer" and when to use "Response.Redirect" ?.....	23
Figure 1.5:- Server Transfer.....	24
What is importance of "preserveForm" flag in "Server.Transfer"? .....	24
Response.Redirect(URL,true) vs Response.Redirect(URL,false) ? .....	24
Can you explain windows, forms and passport authentication? .....	25
What is difference between Grid view, Data list, and repeater?.....	25
Which are the various modes of storing ASP.NET session? .....	25
How can we do caching in ASP.NET? .....	26

What is ViewState?.....	26
What are indexes and what is the difference between clustered and non-clustered? .....	26
How is stored procedure different from functions? .....	26
What's the difference between web services and remoting? .....	26
What's the difference between WCF and Web services? .....	26
What are end point, contract, address, and bindings? .....	27
What is WPF and silverlight? .....	28
What is LINQ and Entity framework?.....	28
What's the difference between LINQ to SQL and Entity framework? .....	28
What are design patterns? .....	29
Which design patterns are you familiar with? .....	29
Can you explain singleton pattern?.....	29
What is MVC, MVP and MVVM pattern? .....	29
What is UML and which are the important diagrams? .....	30
What are different phases in a software life cycle? .....	30
What is Ajax and how does it help?.....	31
How did you do unit testing in your project? .....	31
What is Agile?.....	31
How did you do code reviews? .....	31
How did you convert requirements to technical document?.....	32
Chapter 2: Basic .NET Framework.....	32
What is an IL code? .....	32
Why IL code is not fully compiled? .....	32
Who compiles the IL code and how does it work? .....	32
How does JIT compilation work? .....	32
What are different types of JIT? .....	32
What is Native Image Generator (Ngen.exe)? .....	33
So does it mean that NGEN.EXE will always improve performance?.....	33
Is it possible to view the IL code ? .....	33
What is a CLR? .....	34
What is the difference between managed and unmanaged code? .....	34
What is a garbage collector? .....	34
What are generations in Garbage collector (Gen 0, 1 and 2)? .....	34
Garbage collector cleans managed code, how do we clean unmanaged code? .....	35
But when we create a destructor the performance falls down? .....	35
So how can we clean unmanaged objects and also maintain performance? .....	35
Can we force garbage collector to run? .....	36
What is the difference between finalize and dispose? .....	36
What is CTS? .....	37
What is a CLS (Common Language Specification)? .....	37
What is an Assembly? .....	37
What are the different types of Assembly? .....	37
What is Namespace? .....	37
What is Difference between NameSpace and Assembly? .....	38
What is ILDASM? .....	38

What is Manifest? .....	38
Where is the version information stored of an assembly? .....	38
Is versioning applicable to private assemblies? .....	38
What is the use of strong names? .....	38
What is Delay signing? .....	39
What is GAC? .....	40
How to add and remove an assembly from GAC? .....	40
If we have two versions of the same assembly in GAC how to we make a choice? .....	40
What is Reflection and why we need it? .....	41
How do we implement reflection? .....	41
What are the practical uses of reflection? .....	42
What is the use of Dynamic keyword? .....	42
What are practical uses of Dynamic keyword? .....	44
What is the difference between Reflection and Dynamic? .....	44
Figure 2.7:- Reflection .....	45
Explain the difference between early binding and late binding? .....	45
What is the difference between VAR and Dynamic keyword? .....	46
Explain term type safety and casting in C#? .....	47
Explain casting, implicit conversion and explicit conversion? .....	48
What are stack and heap? .....	48
What are Value types and Reference types? .....	49
What is concept of Boxing and Unboxing? .....	49
How performance is affected due to boxing and unboxing? .....	49
How can we avoid boxing and unboxing? .....	49
If we have a class referencing value type, where is value type stored ? .....	50
Are static variables stored on a heap or stack ? .....	50
How to prevent my .NET DLL to be decompiled? .....	50
What is the difference between Convert.ToString and .ToString () method? .....	50
What is the difference between String vs string? .....	50
So when both mean the same thing why are they different? .....	51
So when to use “String” and “string”? .....	52
How can we handle exceptions in .NET? .....	52
How can I know from which source the exception occurred? .....	53
What if we do not catch the exception? .....	53
What are system level exceptions and application level exceptions? .....	53
Can two catch blocks be executed? .....	54
What are different types of collections in .NET? .....	55
What is the difference between arraylist and list? .....	55
Are Arraylist faster or Arrays? .....	55
What are hashtable collections? .....	55
What are Queues and stack collection? .....	55
Can you explain generics in .NET? .....	56
Explain generic constraints and when should we use them? .....	57
Can you explain the concept of generic collection? .....	58
What is the difference between dictionary and hashtable? .....	58

What are the generic equivalent for array list,stack, queues and hashtable? .....	58
What is the use of IEnumerable, ICollection, IList and IDictionary? .....	59
Differentiate IEnumerable vs IQueryable? .....	59
What is code access security (CAS)? .....	61
So how does CAS actually work? .....	61
Is CAS supported in .NET 4.0? .....	62
What is sandboxing? .....	62
How can we create a windows service using .NET? .....	63
What is serialization and deserialization in .NET? .....	64
Can you mention some scenarios where we can use serialization? .....	65
When should we use binary serialization as compared to XML serialization? .....	66
What is Regular expression? .....	66
What is time out support in regex (regular expression)? .....	67
Can you explain the concept of “Short Circuiting”? .....	68
What is the difference between “Typeof” and “GetType” ? .....	68
Will the following c# code compile? .....	68
Explain the use of Icomparable in c#? .....	69
What is difference between Icomparable and IComparer ? .....	73
Can you explain Lazy Loading? .....	76
So how do we implement “LazyLoading” ? .....	78
Are there any readymade objects in .NET by which we can implement Lazy loading? .....	79
What are the advantages / disadvantages of lazy loading? .....	80
What is the difference between “IS” and “AS” keyword ? .....	81
What is the use of “Yield” keyword? .....	81
What is the difference between “=” and .Equals()? .....	87
What’s the difference between catch with parameter and catch without parameter? .....	89
What are attributes and why do we need it? .....	90
How can we create custom Attributes? .....	92
How can we mark a method as deprecated? .....	93
What is the difference between Build Vs Rebuild Vs Clean solution menu ? .....	93
Figure 2.37: - Clean+Rebuild .....	95
What is the difference between i++ vs ++i ? .....	95
Figure 2.38: - i++ VS ++i .....	95
Figure 2.39: - ++i .....	96
Figure 2.40: - Assign and Increment .....	96
When should we use “??”(NULL Coalescing operator)? .....	97
Explain need of NULLABLE types ? .....	97
In what scenario’s we will use NULLABLE types ? .....	97
What is the benefit of coalescing? .....	97
Chapter 3: OOP .....	98
What is Object Oriented Programming? .....	99
What is a Class and object? .....	99
What are different properties provided by Object-oriented systems? .....	99
How can we implement encapsulation in .NET? .....	99
What’s the difference between abstraction and encapsulation? .....	100

How is inheritance implemented in .NET?.....	100
What are the two different types of polymorphism? .....	101
How can we implement static polymorphism? .....	101
How can we implement dynamic polymorphism?.....	101
What is downcasting and upcasting ? .....	102
What is the difference overriding and overloading?.....	103
What is operator overloading? .....	104
What are abstract classes?.....	105
What are abstract methods? .....	105
What is an Interface? .....	105
Do interface have accessibility modifier ?.....	106
Can we create an object of abstract class or an interface? .....	106
What is difference between abstract classes and interfaces? .....	106
An abstract with only abstract method, how is it different from interfaces? .....	107
If we want to update interface with new methods, what is the best practice? .....	107
What is a delegate? .....	108
How can we create a delegate? .....	109
What is a multicast delegate?.....	109
What are Events? .....	109
What is the difference between delegate and events?.....	110
Do events have return type?.....	110
Can events have access modifiers?.....	110
Can we have shared events? .....	111
Explain Action , Func , Anonymous methods and Lambda expressions?.....	111
What is shadowing? .....	111
What is the difference between Shadowing and Overriding?.....	111
If we inherit a class do the private variables also get inherited? .....	111
How can we stop the class from further inheriting? .....	112
What is the use of “Must inherit” keyword in VB.NET? .....	112
What are similarities between Class and structure?.....	112
What is the difference between Class and structure’s?.....	112
When to use Structures and When to use classes ?.....	113
What does virtual keyword mean?.....	113
What are shared (VB.NET)/Static(C#) variables?.....	114
What is ENUM and what are the benefits of using it? .....	114
What is the use of Flags in ENUM? .....	114
How can we loop through ENUM values? .....	115
What is the use of “ENUM.Parse” method? .....	115
Figure 3.4: - ENUM.Parse .....	116
What is nested Classes? .....	116
If you create the child class object which constructor will fire first? .....	116
In what instances you will declare a constructor to be private? .....	116
Can we have different access modifiers on get/set methods of a property? .....	116
How can you define a property read only for external world and writable in the same assembly?.....	117

Will the finally run in this code?.....	118
What is an Indexer? .....	118
Figure 3.5: - Address Customers .....	120
Figure 3.6: - Address Customers .....	121
How to draw rectangle using Canvas and SVG using HTML 5 ? .....	122
What is the use of column layout in CSS?.....	129
Can you explain CSS box model? .....	132
Can you explain some text effects in CSS 3? .....	133
What are web workers and why do we need them ?.....	134
What are the restrictions of Web Worker thread ? .....	135
So how do we create a worker thread in JavaScript?.....	136
How to terminate a web worker.....	137
Why do we need HTML 5 server-sent events?.....	137
What is local storage concept in HTML 5? .....	139
How can we add and remove data from local storage? .....	139
What is the lifetime of local storage? .....	140
What is the difference between local storage and cookies?.....	140
What is session storage and how can you create one?.....	140
What is difference between session storage and local storage?.....	141
What is WebSQL? .....	141
Is WebSQL a part of HTML 5 specification?.....	141
So how can we use WebSQL ? .....	141
Is Web SQL database still used?.....	142
Explain Indexed DB? .....	142
What is application cache in HTML5? .....	144
So how do we implement application cache in HTML 5 ?.....	144
So how do we refresh the application cache of the browser? .....	145
What is fallback in Application cache? .....	146
What is network in application cache ? .....	146

## Foreword

Changing job is one of the biggest event for any IT professional. When he starts searching he realizes that he needs much more than actual experience. Working on a project is one thing and cracking an interview is a different ball game.

When you work on a project you are doing routine job and you tend to forget the basic fundamentals after some time.

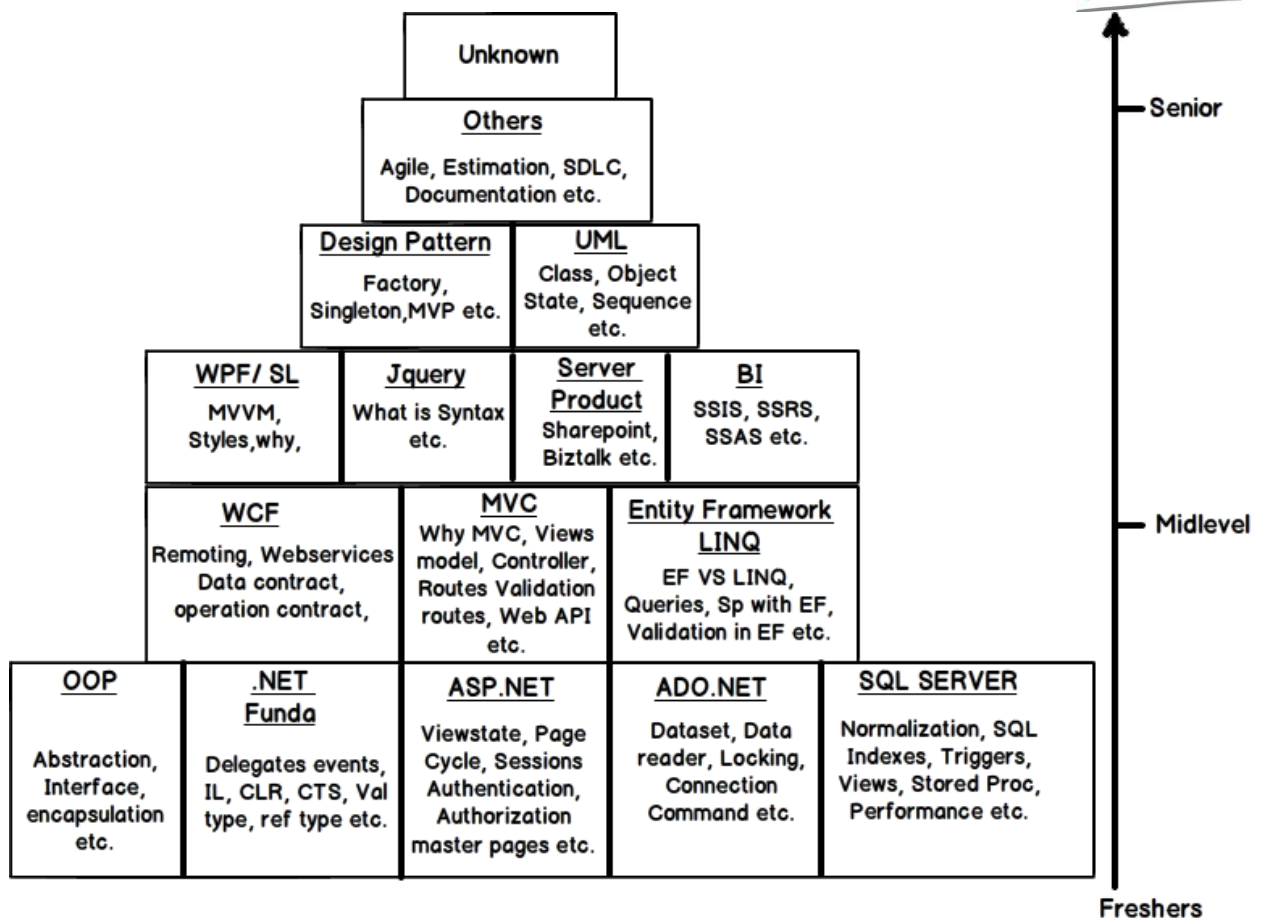
For instance you are working on a highly technical project which uses MVC and Azure majorly, its very much possible that you can fail in simple OOP questions because you are completely out of touch with that topic.

We all know failing in simple OOP questions will not even clear your first round. It does not mean you do not know the fundamentals, it's only that you need to revise that topic.

This book will give you bird eye view of what is needed in .NET interviews. It will help you in doing quick revision so that you can be ready for the interview in day or two. The best way to read this book is not from start to end rather just read the index and then go in details if needed. Before you start preparing for .NET interviews , one golden advice , keep a bigger picture in mind. Do not concentrate on topics which you love rather spread your effort across all topics evenly.

Every company is different , every interviewer is different , so prepare in a holistic way rather than specifics. Please see image (Road map for .net preparation) which shows road map of how to prepare in .NET interviews.





**Figure 0.1 :- Road map for .NET preparation**

It's really good to see emails saying 'We got a job', just makes us feel better, please do write to us on [questpond@questpond.com](mailto:questpond@questpond.com) about your success. I hope this book takes you to a better height and gives you extra confidence boost during interviews.  
 Best of Luck and Happy Job-Hunting.....



Figure: - 0.2: - Professional

## Chapter 1: Top 50 technical and Non-technical questions

Note: - This is a challenge from me , you should atleast get 15 questions from my top 50 most asked question.

### Can you explain architecture of your current project?

Note: - I think 40% of .NET interview starts with this question. Below is a sample answer and will differ from developer to developer.

My current architecture is a 3 layer architecture with UI made in ASP.NET MVC, middle tier having the business logic and the data access layer created by using Entity framework codefirst. It follows MVC architecture and uses Unity DI to decouple the layers.

We also have a common library which is shared in our project and for external customer we have exposed a WCF service.

### What role did you play in your project and company?

Note: - Do not answer in one liners like I am developer , project manager , talk end to end. This will help interviewer to understand much better about you completely. Below is a simple sample answer how professionals reply.

My role was end to end. I was involved right from the requirement phase where I was a part of requirement gathering team and helped the team in requirement documentation.

I was also actively involved in design phase to conceptualize the technical aspect of the project. My main role was in development / coding phase where I was involved in coding and unit testing.

I also played an important role in acceptance testing where I helped the team to fix defects and issues raised by testers and end users.

I am also partly involved with the COE group where I help developers in the company to upgrade to new technologies.

Note: - If you are a part of other activities like training , quality , pre-sales team , estimation etc speak about the same during interview. Companies look out for multitasking capabilities rather than specific capabilities.

---

## **What's your salary expectation?**

This is a very complex question at the end of the book we have a complete chapter at the last which discusses specifically on current .NET developer salary packages and some tips for negotiation.

One golden tip from me, if you really want the job and you do not want salary to be a hurdle, add the word “NEGOTIABLE” after the salary amount. Make the interviewer understand I am ready for a small sacrifice.

## **Why do you want to leave your current organization?**

First and foremost avoid the below three reasons :-

Avoid saying negative about your previous organization because that would send a wrong message to the interviewer. Do not create a pessimistic image before the interviewer. On the contrary if you start with positive things like: - “The current company has no issues and I am extremely happy with the current environment but it's more of a personal problem etc etc...” that would create a very strong and positive image of yours.

Avoid complaining about your current work profile like “It's a maintenance project, it's in 1.1 version and I am not learning anything new etc.” It's very much possible that the interviewer has a maintenance job. So in such kind of situation's I have seen the interviewer rejecting people irrespective even though if they have performed well in technical round.

Avoid saying you are jumping only because of salary. Some of the interviewer's do not like people who just look at monetary benefits. When you are asked about salary expectation these things could be better discussed during that conversation. Do not make it explicit during the interview. Bet nobody likes greedy professionals☺.

Below are some of the good reasons for which the interviewer probably would not question back:-

- For better prospects and growth.
- Office is far away I spend lot of time travelling and would like minimize my travelling time.
- They are telling me to relocate but in my current situation it's not viable.
- I am developer but due to project requirement I am in a testing role and so want to switch.
- Project is getting closed and the company does not have projects.
- Company is not paying salary on time.

## How much do you rate yourself between 1 to 10?

Note:-This questions comes in different forms like, How much do you rate yourself in ASP.NET , How much do you rate yourself in SQL Server etc.

---

Answer for this question is again very subjective but some important tricks here to avoid any traps.

- Even if you are really good in that topic do not rate yourself 10 or 9. It just shows that you are overconfident. If you are extremely good put 8. Let your technical answers talk about your rating.
- If you know the topic averagely, rate yourself 5.
- If you are bad in the topic or have never done anything do not put any rating. Probably it's better to say you have not worked rather than trying to attempt and create more problems.

## Can you speak about yourself?

Note:- Many people become emotional here and talk personal. Most of the time interviewer is expecting to talk about your role you played , technological skills and which path you want to grow. Below is a sample answer and it varies from person to person.

---

I have 8 years of experience, in the first 4 years I worked as a developer in Microsoft technologies. The next 2 years I worked as a team lead guiding the team, helping them resolve problems. Currently I am working in Microsoft server products like SharePoint, BizTalk. My main technical skills revolve around C#, Visual studio, SQL Server, .NET and ASP.NET. I would like to see myself as an technical lead in further coming times.

Note:- The last sentence is very important , let the interviewer know where you want to see yourself in the later stages.

---



## How can we improve performance of .NET?

Note:-This question can have various other forms like How can we improve performance of ASP.NET , How can we improve performance of SQL Server etc. Remember 5 best points of each category. You can pick your favorite ones.

But whatever you pick , should be commonly used by every one.

---

Below are some common points you can remember. List is endless but I do not want to make it long, so that you can remember important ones.

- Use string builder for concatenation rather than string when concatenation data is huge.
- Avoid boxing / unboxing by using generics.
- Avoid writing in line SQL queries use stored procedures.
- Choose your indexes (clustered and non-clustered) properly.
- Use Caching for data which will not change frequently.
- In ASP.NET use output cache directive for page level caching.

## What is the difference between .NET 1.X, 2.0, 3.0,3.5,4.0 and 4.5?

Below is the list of top differences between the framework versions. Please remember the list is much bigger than what I have put down. But for interview perspective I have taken top 5 in each one of them so that we can remember the important ones.

<b>.NET 2.0</b>	<b>.NET 3.0</b>	<b>.NET 3.5</b>	<b>.NET 4.0</b>	<b>.NET 4.5</b>
Support for 64 bit application.	WCF	LINQ	MEF	Async / Await
Generics	WPF	Ajax inbuilt	Parallel computing	ZIP Compression
SQL cache dependency	WWF	ADO Entity framework	DLR dynamic	Profile optimization
Master pages	WCS ( card space)	ADO data services	Code contract	Regex timeout
Membership and roles		Multi targeting	language runtime	Background G
			Lazy initialization	
			Background GC	

## What is ILcode,JIT,CLR, CTS, CLS and CAS?

- IL code is a partially compiled code.
- JIT (Just in time compiler) compiles IL Code to machine language.



- CLR (Common language run time) is the heart of .NET framework and it does 4 primary things Garbage collection, CAS (Code Access security), CV (Code verification) and IL to Native translation.
- CTS (Common types system) ensures that data types defined in two different languages gets compiled to a common data type.
- CLS is a specification or set of rules or guidelines. When any .NET programming language adheres to these set of rules it can be consumed by any language following .NET specifications.
- CAS is the part of .NET security model which determines what kind of rights does a .NET code have.

Note: - Do have a look at the video What is IL code, CLR, CTS, CLS and JIT? There are two parts to this video in the first part we have discussed the theory in the next part we have discussed practically how these terminologies come to life in .NET environment.

---

## What is a garbage collector?

Garbage collector is a feature of CLR which cleans unused managed (it does not clean unmanaged objects) objects and reclaims memory. It's a back ground thread which runs continuously and at specific intervals checks for unused objects whose memory can be claimed.

Note: - Do have a look at the video What is Garbage collector, Gen 0, 1 and 2 , which is in the DVD?

---

## What is GAC?

GAC (Global Assembly Cache) is place where all shared .NET assembly resides. GAC is used when assemblies have to be shared among several applications which are installed in the same computer.

Note :- Do have a look at two classic videos shipped in the DVD which shows how garbage collector works internally.

What is Garbage Collector, Gen 0, 1 & 2 ?

What is IDisposable interface & finalize dispose pattern in GC?

---

## What are stacks, heaps, value types, reference types, boxing and unboxing?

Stack and heap are memory types in an application. Stack memory stores data types like int , double , Boolean etc. While heap stores data types like string and objects.

For instance when the below code runs the first two variables i.e. “i” and “y” are stored in a stack and the last variable “o” is stored in heap.

```
void MyFunction()
{
    int i = 1; // This is stored in stack.
    int y = i; // This is stored in stack.
    object o = null; // This is stored in heap.
} // after this end the stack variable memory space is reclaimed while // the heap memory is reclaimed later
by garbage collector.
```

Value types contain actual data while reference types contain pointers and the pointers point to the actual data.

Value types are stored on stack while reference types are stored on heap. Value types are your normal data types like int , bool , double and reference types are all objects.

When value type is moved to a reference type it’s called as boxing. The vice-versa is termed as unboxing.

Note :- Watch indepth videos given in the CD What is a stack, Heap, Value types and Reference types and What is boxing and unboxing?. These videos example explain step by step concept of boxing , unboxing , value , reference types , stack and heap.

## What is the difference between Out and Ref?

Out and Ref are C# keywords which helps to pass variables in methods and functions as REFERENCE types. Out is one way and Ref is two way.

Let us try to understand the above sentence. Below is a simple function “SomeFunction” which takes a variable “InsideVar”.

```
static void SomeFunction(int InsideVar)
{
    InsideVar = InsideVar + 10;
}
```

Now assume there is a caller who calls the above method as shown in the below code. After the call is made to “SomeFunction” , “OutSideVar” variable is not modified. In other words by default values are passed to methods and function BY VAL.





```
int OutSideVar = 20;
SomeFunction(OutSideVar);
Console.WriteLine(OutSideVar); // Outside var is not modified.
```

Now let us apply REF keyword to the method variable as shown in the below code.

```
static void SomeFunction(ref int InsideVar)
{
    InsideVar = InsideVar + 10;
}
```

If you decorate method variables using REF you also need to provide REF keyword in the caller as shown in the below code. Now in this situation the output of “OutSideVar” will be “30”. In other words the variable is passed by reference. Any modification to the REF variables inside the methods and function will be affected to outside variables as well.

```
int OutSideVar = 20;
SomeFunction(ref OutSideVar); // here outsidevar will be 30
```

OUT is same like REF keyword , it also passes variables by reference but the OUT variables compulsorily needs to be initialized inside the methods and functions.

```
static void SomeFunction(out int InsideVar)
{
    InsideVar=0; // this is mandatory
    InsideVar = InsideVar + 10;
}
```

Which means when the caller passes data to a method which is having OUT variables those data will be discarded. In the below case “20” value is passed from the caller but this data will be discarded as the OUT variables will be initialized again inside the method / function.

```
int OutSideVar = 20;
SomeFunction(Out OutSideVar); // 20 value is not passed
Console.WriteLine(OutSideVar); // here outsidevar will be 10
```

So let's summarize so that interviewer does not take us for a ride:-



- OUT and REF helps to pass variables BY REFERENCE.
- In OUT we need to compulsorily initialize variables inside the function for REF we do not need to do the same.
- REF helps to pass data two ways i.e. from caller to callee and back. While OUT is one way it just passes data from callee to caller.

## How are exceptions handled in .NET?

Exceptions are handled by “System.Exception” base class. If you want raise an error from source you need to create the exception object with below code snippet.

```
throw new Exception("Customer code cannot be more than 10");
```

Once the exception is raised if you want to catch the same you need to use the try catch block as shown below.

```
try
{
    // This section will have the code which
    // which can throw exceptions.
}
catch(Exception e)
{
    // Handle what you want to
    // do with the exception
    label.text = e.Message;
}
```

## What are different types of collections in .NET?

There are five important collections in .NET Arrays, Lists, Hashtable , stacks and queues .

## What are generics?

Generics help to separate logic and data type to increase reusability. In other words you can create a class whose data type can be defined on run time.

Note :- Watch the video shipped in the DVD What is generics ?.

## Explain Abstraction, encapsulation, inheritance and polymorphism?

### Abstraction

Abstraction means show only what is necessary. Example color is abstracted to RGB. By just making the combination of these three colors we can achieve any color in world. It is a model of real world or concept.

### Encapsulation

It is a process of hiding all the complex processing from the outside world and make your objects simple.

### Inheritance

This concept helps to define parent child relationship between classes.

### Polymorphism

It's a property of object to act differently under different conditions. For instance a simple user object depending on conditions can act like a admin or like data entry object.

Note :- Watch the video Can you define OOP and the 4 principles of OOP? for more detail explanation.

## How is abstract class different from an interface?

	<b>Abstract class</b>	<b>Interface</b>
Implementation	Some methods in abstract classes can have implementation.	All methods, function, properties in interfaces are empty. They are simple signatures.
Scenario	Abstract classes are used when we want to share common functionality in parent child relationship.	Interfaces are used to define contract, enforce standardization, decoupling and dynamic polymorphism.
Variable declaration	We can declare variables	In interface we cannot declare variables.
Inheritance VS Implementation	Abstract classes are inherited.	Interfaces are implemented.

Note :- Refer the below 3 videos to understand abstract classes and interfaces in more depth.

What is an abstract class?

Define Interface & What is the diff. between abstract & interface?

Define Interface & Diff. between abstract & interface? - Part 2

## What are the different types of polymorphism?

There are 2 kinds of polymorphism static and dynamic. Many people also call them as runtime or compile time polymorphism. Static polymorphism is implemented by overloading while dynamic polymorphism is implemented by overriding and virtual keyword.

Note: -Do watch the video “What is Polymorphism, overloading, overriding and virtual?” for in depth demonstration of how to implement polymorphism in .NET. The video also talks about two different types of polymorphism in depth.

## How does delegate differ from an event?

Delegate is an abstract strong pointer to a function or method while events are higher level of encapsulation over delegates. Events use delegates internally.

They differ for the below reasons:-

- Actually, events use delegates in bottom. But they add an extra layer on the delegates, thus forming the publisher and subscriber model.
- As delegates are function to pointers, they can move across any clients. So any of the clients can add or remove events, which can be confusing. But events give the extra protection / encapsulation by adding the layer and making it a publisher and subscriber model.

Just imagine one of your clients doing this

```
c.XyzCallback = null
```

---

This will reset all your delegates to nothing and you have to keep searching where the error is.

Note :- Do watch the below three videos given in the CD to get more insight on delegates and events.

What problem does Delegate Solve? - Part1

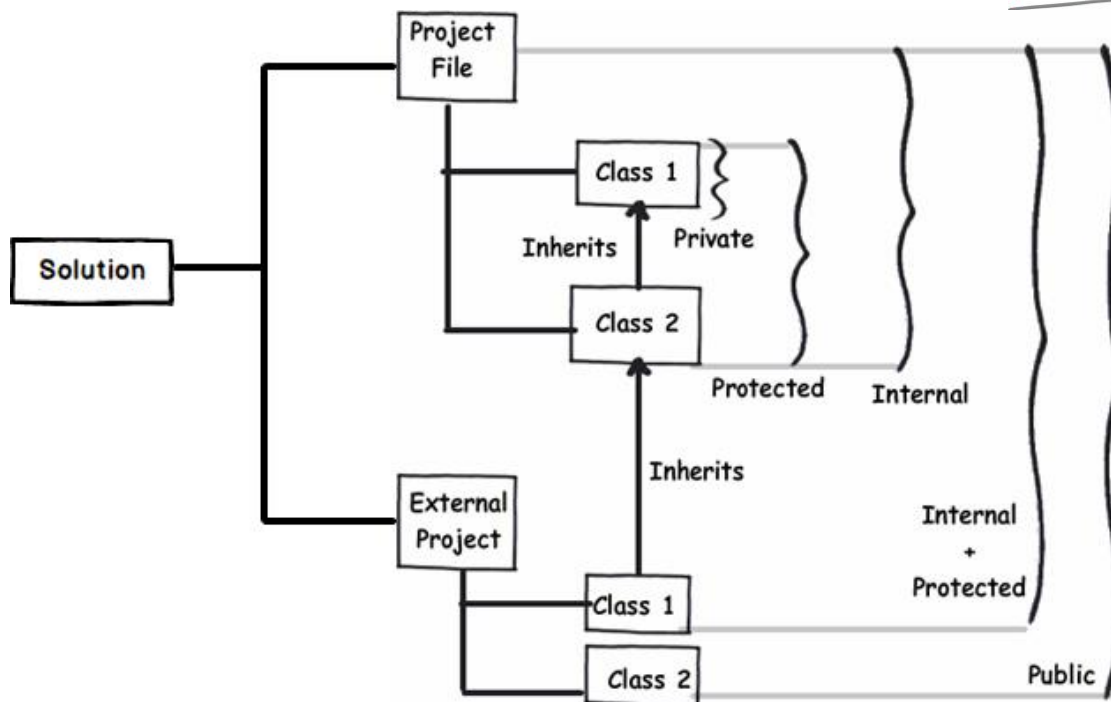
What is a Multicast delegate? - Part2

What are events & what's the difference between delegates & events? - Part3

## What are different access modifiers?

There are 5 access modifiers. Access modifiers define scope for members.

- Private: Accessible only with in the class.
- Protected: -Accessible with in the class and in derived classes.
- Friend (internal in C#):-Accessible anywhere within the current project.
- Protected friend (protected internal in C#):- Accessible with current project and derived classes.
- Public: -Accessible everywhere.



**Figure 1.1:- Different Access Modifiers**

Note: - Watch the video Can you explain encapsulation and abstraction? Where we have explained how encapsulation can be implemented using access modifiers.

### **explain connection, command, datareader and dataset inADO.NET ?**

- Connection: - This object creates a connection to the database. If you want to do any operation on the database you have to first create a connection object.
- Command: - This object helps us to execute SQL queries against database. Using command object we can execute select, insert, update and delete SQL command.
- Data reader: - This provides a recordset which can be browsed only in forward direction. It can only be read but not updated. Data reader is good for large number of records where you want to just browse quickly and display it.
- Dataset object: - This provides a recordset which can be read back and in forward direction. The recordset can also be updated. Dataset is like a in memory database with tables, rows and fields.
- Data Adapter: - This object acts as a bridge between database and dataset; it helps to load the dataset object.

### **How does “Dataset” differ from a “Data Reader”?**

- “Dataset” is a disconnected architecture, while “Data Reader” has live connection while reading data. If we want to cache data and pass to a different tier “Dataset” forms the best choice and it has decent XML support.
- When application needs to access data from more than one table “Dataset” forms the best choice.
- If we need to move back while reading records, “data reader” does not support this functionality.
- However, one of the biggest drawbacks of Dataset is speed. As “Dataset” carry considerable overhead because of relations, multiple table’s etc speed is slower than “Data Reader”. Use “Data Reader” when you want to quickly read and display records on a screen.
- Dataset can manipulate data while data reader is only for reading purpose.

## How is ASP.NET page life cycle executed?

Following is the sequence in which the events occur:-

- Init
- Load
- Validate
- Event
- Render

Remember the word SILVER :- SI ( Init ) L ( Load ) V ( Validate ) E ( Event ) R ( Render ) .

---

Note :- Watch the complete video of Can you explain ASP.NET application and Page life cycle ?

## What are Httphandlers and HttpModules and difference between them?

Handlers and modules helps you inject pre-processing logic before the ASP.NET request reaches the website.

For instance before your request reaches any resource you would like to check if the user has been authenticated or not.

Httphandlers is an extension based processor. In other words the pre-processing logic is invoked depending on file extensions.

Httpmodule is an event based processor. In other words ASP.NET emits lot of event like BeginRequest, AuthenticateRequest etc, we can write logic in those events using Httpmodule.

---

Note :- Watch the complete video of Can you explain ASP.NET application and Page life cycle ? where we have explained handlers and modules in depth.

## What are different kind of validator controls in ASP.NET ?

There are six main types of validation controls:-

### **RequiredFieldValidator**

It checks whether the control have any value. It is used when you want the control should not be empty.

### **RangeValidator**

It checks if the value in validated control is in that specific range. Example TxtCustomerCode should not be more than eight lengths.

### **CompareValidator**

It checks that the value in controls should match some specific value. Example Textbox TxtPie should be equal to 3.14.

### **RegularExpressionValidator**

When we want the control, value should match with a specific regular expression.

### **CustomValidator**

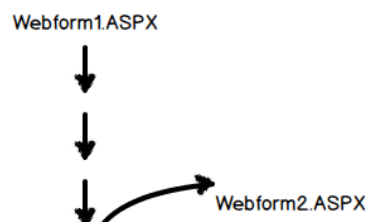
It is used to define User Defined validation.

### **Validation Summary**

It displays summary of all current validation errors on an ASP.NET page.

## How is 'Server.Transfer' different from 'response.Redirect' ?

“response.redirect” and “server.transfer” helps to transfer user from one page to other page while the page is executing.

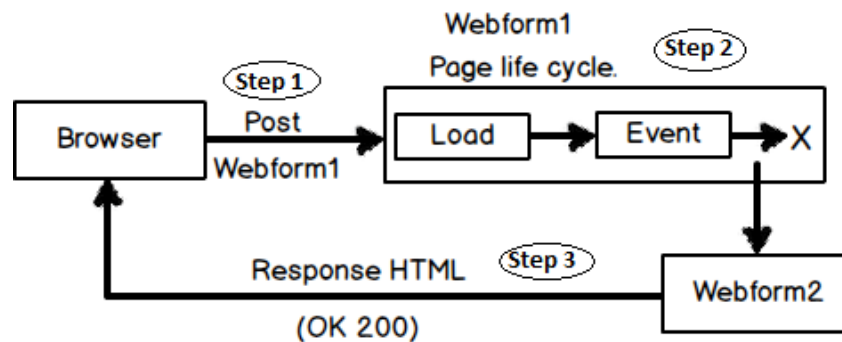


**Figure 1.2:- Response.Redirect**

The main difference between them is who does the transfer. In “response.redirect” the transfer is done by the browser while in “server.transfer” it’s done by the server.

In “Server.Transfer” following is the sequence of how transfer happens:-

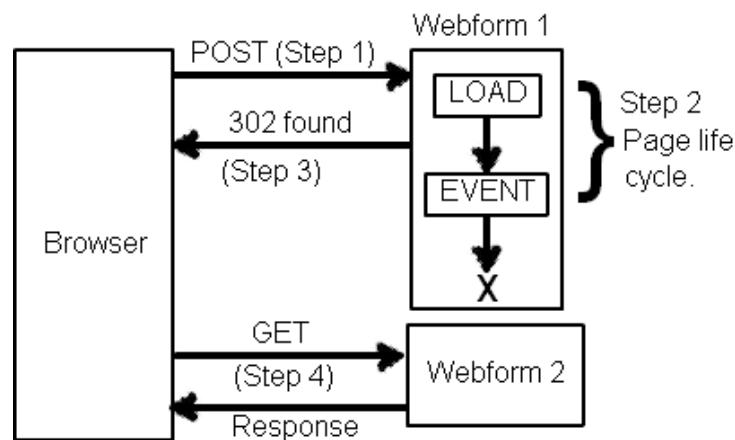
1. User sends a request to an ASP.NET page. In the below figure the request is sent to “WebForm1” and we would like to navigate to “Webform2”.
2. Server starts executing “Webform1” and the life cycle of the page starts. But before the complete life cycle of the page is completed “Server.transfer” happens to “WebForm2”.
3. “Webform2” page object is created, full page life cycle is executed and output HTML response is then sent to the browser.



**Figure 1.3:- Webforms**

In “Response.Redirect” following is the sequence of events for navigation:-

1. Client(browser) sends a request a page.In the below figure the request is sent to “WebForm1” and we would like to navigate to “Webform2”.
2. Life cycle of “Webform1” starts executing. But in between of the life cycle “Response.Redirect” happens.
3. Now rather than server doing a redirect, he sends a HTTP 302 command to the browser. This command tells the browser that he has to initiate a GET request to “Webform2.aspx” page.
4. Browser interprets the 302 command and sends a GET request for “Webform2.aspx”.



**Figure 1.4:- Page Life Cycle**

**So when to use “Server.Transfer” and when to use “Response.Redirect” ?**

Use “Server.Transfer” when you want to navigate pages which reside on the same server, use “Response.Redirect” when you want to navigate between pages which resides on different server and domain.

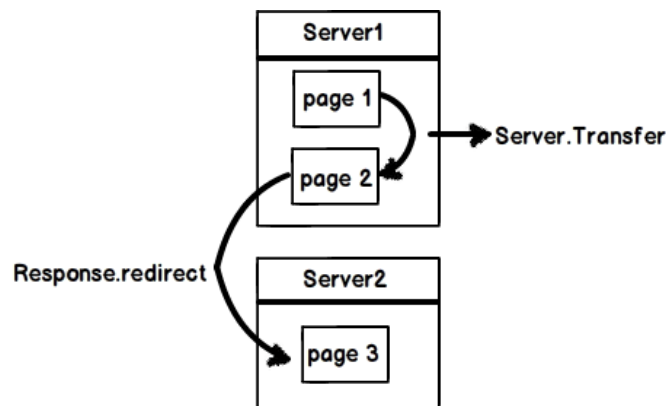


Figure 1.5:- Server Transfer

## What is importance of “preserveForm” flag in “Server.Transfer”?

Note: -This question can also be asked in the form what is the difference between “Server.Transfer(true)” vs “Server.Transfer(false)”.

“Server.Transfer” helps to redirect from one page to other page. If you wish to pass query string and form data of the first page to the target page during this redirection you need to set “preserveForm” to “true” as shown in the below code.

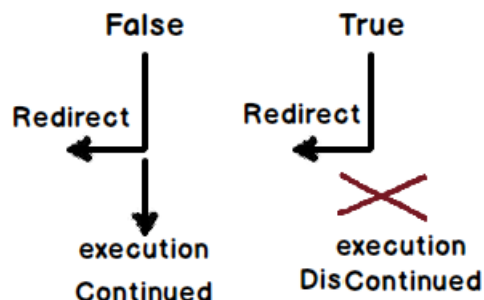
```
Server.Transfer("Webform2.aspx",true);
```

By default the value of “preserveForm” is “true”.

## Response.Redirect(URL,true) vs Response.Redirect(URL,false) ?

Response.Redirect(URL,false) :- Client is redirected to a new page and the current page on the server will keep processing ahead.

Response.Redirect(URL,true) :- Client is redirected to a new page but the processing of the current page is aborted.





**Figure 1.6:- False True**

## **Can you explain windows, forms and passport authentication?**

There are 3 major ways of doing authentication and authorization:

- **Windows:** - In this mode the users are stored in windows local user groups.
- **Forms:** - In this mode we create a login screen and use the forms authentication class to do validations. It's a ticket based authentication.
- **Passport :-** In this mode the users are validated from Microsoft sites like hotmail , devhood , MSN etc , ticket is generated and that ticket can be used to do authentication and authorization in your web application.

Note: - Watch the 6 videos which talks in depth about the above concepts. Its possible that interviewer can get in to details.

What is Authentication, Authorization, Principal & Identity objects? Part-1

ASP.NET Authentication and Authorization Video series Part-2

ASP.NET Authentication and Authorization Video series Part-3

ASP.NET Authentication and Authorization Video series Part-4

ASP.NET Authentication and Authorization Video series Part-5

ASP.NET Authentication and Authorization Video series Part-6

## **What is difference between Grid view, Data list, and repeater?**

Grid view and data grid by default display all the data in tabular format i.e. in table and rows.

Developer has no control to change the table data display of datagrid.

Data list also displays data in a table but gives some flexibility in terms of displaying data row wise and column wise using the repeat direction property.

Repeater control is highly customizable. It does not display data in table by default. So you can customize from scratch the way you want to display data.

## **Which are the various modes of storing ASP.NET session?**

- **InProc:** - In this mode Session, state is stored in the memory space of the Aspnet\_wp.exe process. This is the default setting. If the IIS reboots or web application restarts then session state is lost.
- **StateServer:-** In this mode Session state is serialized and stored in a separate process (Aspnet\_state.exe); therefore, the state can be stored on a separate computer(a state server).
- **SQL SERVER:** - In this mode Session, state is serialized and stored in a SQL Server database.



Session state can be specified in <sessionState> element of application configuration file. Using State Server and SQL SERVER session state can be shared across web farms but note this comes at speed cost as ASP.NET needs to serialize and deserialize data over network repeatedly.

## **How can we do caching in ASP.NET?**

There are two ways of caching output cache directive and using the cache objects.

## **What is ViewState?**

Viewstate is a built-in structure for automatically retaining values amongst the multiple requests for the same page. The viewstate is internally maintained as a hidden field on the page but is hashed, providing greater security than developer-implemented hidden fields do.

## **What are indexes and what is the difference between clustered and non-clustered?**

Index makes your search faster by using B-Tree structure logic. Clustered and non-clustered indexes are two types of indexes. In clustered index the leaf node points to the actual data while in non-clustered index the leaf node uses the clustered index to locate data.

## **How is stored procedure different from functions?**

- Function cannot affect the state of the database which means we cannot perform CRUD operation on the database. Stored Procedure can affect the state of the database by using CRUD operations.
- Store Procedure can return zero or n values whereas Function can return only one value.
- Store Procedure can have input, output parameters for it whereas functions can have only input parameters.
- Function can be called from Stored Procedure whereas Stored Procedure cannot be called from Function.

## **What's the difference between web services and remoting?**

Remoting works only when both the ends i.e. server and client are in .NET technologies. Web services are useful when the client is not .NET like java etc.

## **What's the difference between WCF and Web services?**

- WCF services can be hosted in multiple protocols like http, tcp etc. Web services can only be hosted on Http protocol.



- WCF has COM+ so you can call two different WCF services in a transaction , we can not call two different web services in one transaction.
- WCF integrates with MSMQ easily, but for web services we will need to write code to communicate with the MSMQ pool.

In simple words below equation shows the difference with simple equation.

**WCF = Web services + Remoting + MSMQ + COM+**

**Web service = WCF – (Remoting + MSMQ + COM+ )**

---

Note :- Do watch the below 4 videos of WCF to get more insight in to WCF.

What are basic steps to create a WCF service (Part I, Creating the Service)?

What are basic steps to create a WCF service (Part II, Consuming the Service)?

What are endpoints, address, contracts and bindings?

What are various ways of hosting WCF service?

---

## What are end point, contract, address, and bindings?

When we want to host any WCF service we need to provide where to host it, how to host it and what to host.

- **Contract** (What)

Contract is an agreement between two or more parties. It defines the protocol how client should communicate with your service. Technically, it describes parameters and return values for a method.

- **Address** (Where)

An Address indicates where we can find this service. Address is a URL, which points to the location of the service.

- **Binding** (How)

Bindings determine how the service can be accessed. It determines how communications is done. For instance, you expose your service, which can be accessed using SOAP over HTTP or BINARY over TCP. So for each of these communications medium two bindings will be created.

- **End point**: - It's the combination of contract, address and binding.

In WCF web.config file we can specify end point , address , binding and contract as shown in the below code snippet.

```
<endpoint address="http://www.questpond.com" binding="wsHttpBinding"
contract="WcfService3.IService1">
```

---

Note :- Do watch the below 4 videos of WCF to get more insight in to WCF.

What are basic steps to create a WCF service (Part I, Creating the Service)?

What are basic steps to create a WCF service (Part II, Consuming the Service)?

What are endpoints, address, contracts and bindings?

What are various ways of hosting WCF service?

What is the difference of hosting a WCF service on IIS and Self hosting?

What is the difference between BasicHttpBinding and WsHttpBinding?

---

## What is WPF and silverlight?

WPF (Windows Presentation foundation) is a graphical subsystem for displaying user interfaces, documents, images, movies etc. It uses XAML which is a XML descriptive language to represent UI elements.

Silver light is a web browser plug-in by which we can enable animations, graphics and audio video. You can compare silver light with flash. We can view animations with flash and it's installed as a plug-in in the browser.

Note :- Do have a look at the below videos for WPF and silverlight.

What is the need of WPF when we had GDI, GDI+ and DirectX?

Can you explain how we can make a simple WPF application?

Can you explain the architecture of Silverlight ?

What are the basic things needed to make a silverlight application ?

---

## What is LINQ and Entity framework?

LINQ is a uniform programming model for any kind of data access. It is also an OR mapper which helps us to expedite our business object creation process.

ADO.NET entity is an ORM (object relational mapping) which abstracts data model by providing a simplified object model.

In other words the complete middle tier development is expedited using entity framework.

Note :- Below simple video demonstration in the CD will help your understand LINQ and entity framework in to great depth.

What is LINQ and can you explain same with example?

Can you explain a simple example of LINQ to SQL?

---

## What's the difference between LINQ to SQL and Entity framework?

- LINQ to SQL is good for rapid development with SQL Server. EF is for enterprise scenarios and works with SQL server as well as other databases.

- LINQ maps directly to tables. One LINQ entity class maps to one table. EF has a conceptual model and that conceptual model map to storage model via mappings. So one EF class can map to multiple tables or one table can map to multiple classes.
- LINQ is more targeted towards rapid development while EF is for enterprise level where the need is to develop loosely coupled framework.

If you are starting with fresh development using entity framework, in case you are in older version of framework where entity framework does not exist use linq to sql.

## What are design patterns?

Design patterns are recurring solution to recurring problems in software architecture.

Note :- Do watch the following videos for design pattern which is shipped with the CD :-

Introduction

Factory pattern

Memento Pattern

Singleton Pattern

## Which design patterns are you familiar with?

Left to the readers, pick any three patterns which you have used in your project and talk about it. In chapter “Design patterns, UML, Estimation and Project management” we have explained 3 design patterns.

Note :- In CD we have snipped 3 design pattern videos you can have a look at them.

Factory pattern

Memento Pattern

Singleton Pattern

## Can you explain singleton pattern?

Singleton pattern helps us to create a single instance of an object which can be shared across project. Main use of singleton pattern is for global data sharing and caching.

Note :- The CD contains a video explanation with source code for “Singleton Pattern”.

## What is MVC, MVP and MVVM pattern?

All the above design patterns come in presentation pattern category and help to remove any kind of cluttered code in UI like manipulation of user interfaces and maintaining state. Thus keeping your UI code cleaner and better to maintain.



MVC pattern divides the architecture in to 3 part model, view and controller. The first request comes to the controller and the controller then decides which view to be displayed and ties up the model with the view accordingly.

MVP (Model view presenter) has the same goals as MVC i.e. separating the UI from the model. It does the same by using a presenter class. The UI talks via an interface to the presenter class and the presenter class talks with the model.

MVVM is an architectural pattern with the focus of removing UI cluttered code. It does the same by using an extra class called as view model. MVVM is mostly suitable for Silverlight and WPF projects because of the rich bindings provided by the technologies.

**Note :-** In the CD we have shipped 4 videos which explains MVC , MVP and MVVM patterns. Below are the name of the videos :-

The basic of MVC HttpHandlers

MVC using core ASP.NET and HttpHandler

MVC using MVC ASP.NET

Model View Presenter video

---

## **What is UML and which are the important diagrams?**

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system.

UML provides blue prints for business process, System function, programming language statements, database schemas and reusable components. Some of the important UML diagrams are use case , class diagrams , sequence diagram, activity diagram , object diagrams , collaboration diagrams , state chart diagrams , component diagram and deployment diagram.

**Note :-** We have shipped 5 videos which explains some important UML diagrams in detail. Below is the list.

Introduction

Use Case Diagrams

Class Diagrams

Sequence Diagrams

Collaboration Diagrams

---

## **What are different phases in a software life cycle?**

There are six phases in software development:-

- Requirement
- Design
- Coding and unit testing
- System testing
- Acceptance testing
- Go live

## What is Ajax and how does it help?

Ajax stands for Asynchronous JavaScript and XML. There are two prime benefits of Ajax:-

- It send's only necessary data to the server. For instance let's say you have 4 textboxes and on a submit button you want to only send two text box data, Ajax helps in the same.
- The second benefit is it's asynchronous. In other words when you click on submit button and until the server processes the request you can do other activities on the site. For instance when you click on send email and until the email is sent , you can start composing a new email at the back ground.

## How did you do unit testing in your project?

From unit testing perspective there are two great tools "NUNIT" and "Visual studio unit test template". Talk about which you are comfortable with. Below are two videos in DVD which shows demo of how to use NUNIT and VSTS unit test template.

Note :- In the CD we have shipped four videos which explain unit testing using

What is Unit Testing & can we see an example of the same?

How can we write data driven test using NUNIT & VS Test?

---

## What is Agile?

Agile is a development methodology where we develop in incremental and iteratively. In agile we consider software as the most important entity and accept user changes and deliver them in small releases. There are four important principle of agile:-

- Individuals and interactions over processes and tools.
- Working software over comprehensive documentation.
- Customer collaboration over contract negotiation.
- Responding to change over following a plan.

## How did you do code reviews?

Code reviews are either done manually i.e. peer review or automated review using tools like style cop and fxcop.

Note :- We have shipped two videos in the CD which explains how to use stylecop and fxcop tools for code reviews.

---

## **How did you convert requirements to technical document?**

Note :- This is a subjective answer. Below goes my version.

---

Requirements are normally available in use cases or free text. The first step is to identify the classes, properties for the classes, methods / functions and relationship between the classes. As a general rule nouns become classes and verbs become actions for the classes.

Once the classes are identified we can use sequence or collaboration diagrams to detail out the interactions. The database objects are later created using the class diagrams.

## **Chapter 2: Basic .NET Framework**

### **What is an IL code?**

IL code is a partially compiled code.

Note: - Half compiled means this code is not yet compiled to machine/CPU specific instructions.

---

### **Why IL code is not fully compiled?**

We do not know in what kind of environment .NET code will run. In other words we do not know what can be the end operating system, CPU configuration, machine configuration, security configuration etc. So the IL code is half compiled and on runtime this code is compiled to machine specific using the environmental properties (CPU, OS, machine configuration etc).

### **Who compiles the IL code and how does it work?**

IL code is compiled by JIT (Just in time compiler).

### **How does JIT compilation work?**

JIT compiles the code just before execution and then saves this translation in memory. Just before execution JIT can compile per-file, per function or a code fragment.

### **What are different types of JIT?**

In Microsoft .NET there are 3 types of JIT compilers:-





- **Normal-JIT (Default):** - 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.
- **Econo-JIT:** - Econo-JIT compiles only those methods that are called at runtime. However, these compiled methods are not stored in cache so that RAM memory can be utilized in an optimal manner.
- **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. We can implement Pre-jit by using ngen.exe.

Normal-jit is the default implementation and it produces optimized code. Econo-jit just replaces IL instruction with native counterpart. It does not do any kind of optimization. Econo-jit does not store the compiled code in cache so it requires less memory.

The choice of Normal-JIT and Econo-JIT is decided internally. Econo-JIT is chosen when devices have limitation memory and CPU cycle issues like windows CE powered device. When there is no memory crunch and CPU power is higher than Normal-JIT is used.

Pre-JIT is implemented by using ngen.exe which is explained in the next question.

## What is Native Image Generator (Ngen.exe)?

Ngen stores full compiled .NET native code in to cache. In other words rather than dynamically compiling the code on run time a full image of native compiled code is stored in cache while installing the application. This leads to better performance as the assembly loads and execute faster.

In order to install full compiled native code in cache we can execute the below command line from your visual studio command prompt.

```
ngen.exe install <assemblyname>
```

---

## So does it mean that NGEN.EXE will always improve performance?

No, it's not always necessary that ngen.exe produces optimized code because it uses the current environments parameters which can change over a period of time. For instance a code compiled in windows XP environment will not be the optimized code to run under windows 2008 server. So we need to once test with 'ngen' and without 'ngen' to conclude if really the performance increases.

## Is it possible to view the IL code ?

Yes by using ILDASM simple tool we can view all code of a DLL or EXE. In order to view IL code using ILDASM, go to visual studio command prompt and run "ILDASM.EXE". Once ILDASM is running you view the IL code.

## What is a CLR?

CLR (Common language run time) is the heart of .NET framework and it does 4 primary important things:-

- Garbage collection
- CAS (Code Access security)
- CV (Code verification)
- IL to Native translation.

Note: - There are many other uses of CLR but I have kept it short from the interview point of view. In the further section we will go in depth of these questions.

---

## What is the difference between managed and unmanaged code?

Code that executes under CLR execution environment is called as managed code. Unmanaged code executes outside CLR boundary. Unmanaged code is nothing but code written in C++ , VB6 , VC++ etc. Unmanaged codes have their own environment in which the code runs and it's completely outside the control of CLR.

## What is a garbage collector?

Garbage collector is a feature of CLR which cleans unused managed (it does not clean unmanaged objects) objects and reclaims memory. It's a back ground thread which runs continuously and at specific intervals it checks if there are any unused objects whose memory can be claimed.

Note: - GC does not claim memory of unmanaged objects.

---

Note: - Garbage collector is one of the very important interview topics due to complexity of generations, double GC loop because of destructor and the implementation of finalize and dispose pattern. So please do go through the video of "What is Garbage collection, Generation, Finalize, Dispose and IDisposable?" to ensure that you understand the fundamentals well.

---

## What are generations in Garbage collector (Gen 0, 1 and 2)?

Generations defines age of the object. There are three generations:-

- **Gen 0**:- When application creates fresh objects they are marked as Gen 0.
- **Gen 1**:- When GC is not able to clear the objects from Gen 0 in first round it moves them to Gen 1 bucket.
- **Gen 2**:- When GC visits Gen 1 objects and he is not able to clear them he moves them gen 2.



Generations are created to improve GC performance. Garbage collector will spend more time on Gen 0 objects rather than Gen 1 and Gen 2 thus improving performance.

Note: - More the objects in Gen 0, more your application is stable.

---

## **Garbage collector cleans managed code, how do we clean unmanaged code?**

Garbage collector only claims managed code memory. For unmanaged code you need to put clean up in destructor / finalize.

## **But when we create a destructor the performance falls down?**

Yes, when we define a destructor, garbage collector does not collect these objects in the first round. It moves them to Gen 1 and then reclaims these objects in the next cycle.

As more objects are created in Gen 1 the performance of the application falls down because more memory is consumed.

## **So how can we clean unmanaged objects and also maintain performance?**

We need to follow the below steps:-

- Implement IDisposable interface and implement the dispose function.
- In Dispose function calls the “GC.SuppressFinalize” method.
- At the client side ensure that the “Dispose” function is called when the object is no more required.

Below goes the code, this is also called as “Finalize and Dispose pattern”. This ensures that your objects are created in Generation 0 rather than Generation 1. “GC.SuppressFinalize” tells the garbage collector to not worry about destructor and destroy the objects in the first call itself.

```
class clsMyClass : IDisposable
{
    ~clsMyClass()
    {
        // In case the client forgets to call
        // Dispose , destructor will be invoked for
        Dispose(false);
    }
    protected virtual void Dispose(bool disposing)
```

```
{  
    if (disposing)  
    {  
        // Free managed objects.  
    }  
    // Free unmanaged objects  
}  
  
public void Dispose()  
{  
    Dispose(true);  
    // Ensure that the destructor is not called  
    GC.SuppressFinalize(this);  
}  
}
```

---

Note :- Please do go through the videos of “What is IDisposable interface & finalize dispose pattern in GC?” in which we have actually showed how generation performance increases by using Finalize and Dispose pattern.

---

## Can we force garbage collector to run?

“System.GC.Collect ()” forces garbage collector to run. This is not a recommended practice but can be used if situations arise.

## What is the difference between finalize and dispose?

- Finalize is a destructor and dispose is a function which is implemented via ‘IDisposable’ interface.
- Finalize is nondeterministic, since it’s called by garbage collector. Dispose is a function and needs to be called by the client for clean up. In other finalize is automatically called by garbage collector while dispose needs to be called forcefully.

Note: -As a good practice Finalize and dispose is used collectively because of double garbage collector loop. You can talk about this small note after you talk about the above 2 differences.

---

## What is CTS?

In .NET there are lots of languages like C#, VB.NET, VF.NET etc. There can be situations when we want code in one language to be called in other language. In order to ensure smooth communication between these languages the most important thing is that they should have a common type system. CTS (Common types system) ensures that data types defined in two different languages get compiled to a common data type.

So “Integer” data type in VB6 and “int” data type in C++ will be converted to System.int32, which is data type of CTS.

**Note:** If you know COM programming, you would know how difficult it is to interface VB6 application with VC++ application. As datatype of both languages did not have a common ground where they can come and interface, by having CTS interfacing is smooth.

---

## What is a CLS (Common Language Specification)?

CLS is a subset of CTS. CLS is a specification or set of rules or guidelines. When any programming language adheres to these set of rules it can be consumed by any .NET language. For instance one of the rule which makes your application CLS non-compliant is when you declare your methods members with same name and with only case differences in C#. You can try this create a simple class in C# with same name with only case differences and try to consume the same in VB.NET ,it will not work.

## What is an Assembly?

Assembly is unit of deployment like EXE or a DLL.

## What are the different types of Assembly?

There are two types of assembly Private and Public assembly. A private assembly is normally used by a single application, and is stored in the application's directory, or a sub-directory beneath. A shared assembly is stored in the global assembly cache, which is a repository of assemblies maintained by the .NET runtime.

Shared assemblies are needed when we want the same assembly to be shared by various applications in the same computer.

## What is Namespace?

Namespace does two basic functionalities:-

- It logically groups classes, for instance System.Web.UI namespace logically groups UI related features like textboxes, list control etc.

- In Object Oriented world, many times it is possible that programmers will use the same class name. Qualifying NameSpace with class names avoids this collision.

## **What is Difference between NameSpace and Assembly?**

Following are the differences between namespace and assembly:

- Assembly is physical grouping of logical units, Namespace, logically groups classes.
- Namespace can span multiple assemblies while assembly is a physical unit like EXE , DLL etc.

## **What is ILDASM?**

ILDASM is a simple tool which helps you to view IL code of a DLL or EXE. In order to view IL code using ILDASM , go to visual studio command prompt and run “ILDASM.EXE”. Once ILDASM is running you view the IL code.

## **What is Manifest?**

Assembly metadata is stored in Manifest. Manifest contains metadata which describes the following things -:

- Version of assembly.
- Security identity.
- Scope of the assembly.
- Resolve references to resources and classes.

The assembly manifest is stored in the DLL itself.

## **Where is the version information stored of an assembly?**

Version information is stored in assembly inside the manifest.

## **Is versioning applicable to private assemblies?**

Yes, versioning is applicable to private assemblies also.

## **What is the use of strong names?**

**Note:-** This question can also be asked in two different ways , what are weak references and strong reference or how do you strong name a .NET assembly.

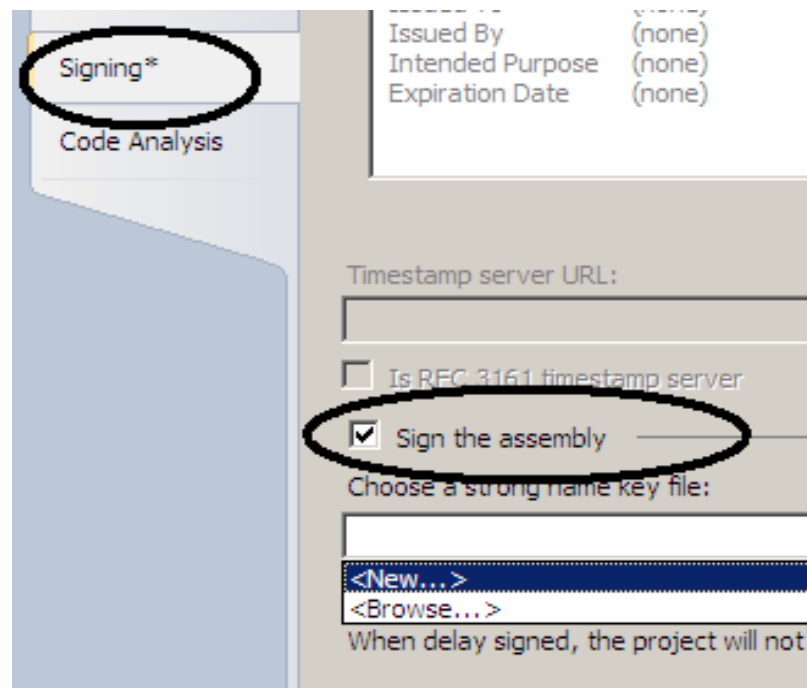
---

When we talk about .NET application it has two parts one is the class library or the DLL and the other the consumer like windows UI etc using this DLL.

If the consumer identifies the DLL library by namespace and class names it's called as weak reference. It's very much possible in deployment environment someone can delete the original class library and fake a similar class library with the same class name and namespace name.

Strong name is a unique name which is produced by the combination of version number, culture information, public key and digital signature. No one can fake this identity or generate the same name.

So your consumer or UI will refer the class library with strong names rather than class and namespace names. In order to create strong name, right click on the class library, click on properties, click on signing tab and click on the new menu to generate strong names as shown in the below figure.



**Figure 2.1: - Strong Names**

## What is Delay signing?

The whole point about strong names is to ensure that the clients (UI , External components etc) who is consuming the DLL knows that the DLL was published from a valid source. This authenticity is verified by using strong names. Strong name protection is good from external hackers but what if your own developers think of doing something mischievous.

That's where delay signing helps. The strong name key has two keys public key and private key. You only share the public key with your developers so that they can work seamlessly. The private key is stored in a secured location and when the DLL is about to be deployed on production the key is injected for further security.



## What is GAC?

GAC (Global Assembly Cache) is where all shared .NET assembly resides. GAC is used in the following situations:-

- If the application has to be shared among several application which is in the same computer.
- If the assembly has some special security, requirements like only administrators can remove the assembly. If the assembly is private then a simple delete of assembly the assembly file will remove the assembly.

## How to add and remove an assembly from GAC?

You can use the 'GacUtil' tool which comes with visual studio. So to register an assembly in to GAC go to "Visual Studio Command Prompt" and type "gacutil -i (assembly name)", where (assembly name) is the DLL name of the project.

Once you have installed the assembly the DLL can be seen in 'c:\windows\assembly\' folder.

When we have many DLL's to be deployed we need to create setup and deployment package using windows installer. So the common way of deploying GAC DLL in production is by using windows installer.

## If we have two versions of the same assembly in GAC how to we make a choice?

When we have two version of the same assembly in GAC we need to use binding redirect tag and specify the version we want to use in the new version property as shown in the below "app.config" file.

```
<configuration>
<runtime>
<assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
<dependentAssembly>
<assemblyIdentity name="ComputerName" publicKeyToken="cfc68d722cd6a164" />
<publisherPolicy apply="yes" />
<bindingRedirect oldVersion="1.1.0.0" newVersion="1.0.0.0" />
</dependentAssembly>
</assemblyBinding>
</runtime>
</configuration>
```



## What is Reflection and why we need it?

Reflection is needed when you want to **determine / inspect contents of an assembly**. For example look at your visual studio editor intellisense, when you type “.” (dot) before any object , it gives you all members of the object. This is possible because of reflection.

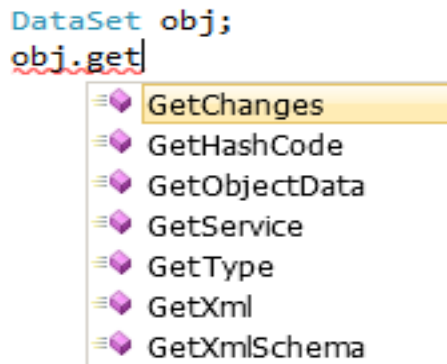


Figure 2.2:- Reflection

Reflection also goes one step further; it can also invoke a member which is inspected. For instance if the reflection detects that there is a method called as “GetChanges” in an object. We can get a reference to that method instance and invoke the same on runtime. In simple words reflection passes through two steps “Inspect” and “Invoke” (optional). “Invoke” process is optional.

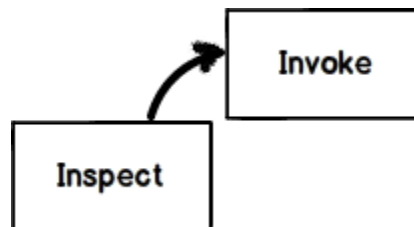


Figure 2.3:- Invoke

## How do we implement reflection?

Implementing reflection in c# is a two step process , 1<sup>st</sup> get the “type” of the object and then use the type to browse members like “methods” , “properties” etc.

**Step 1:-** The first step is to get the type of the object. So for example you have a DLL called as “ClassLibrary1.dll” which has a class called as “Class1”. We can use the “Assembly” (belongs to System.Reflection namespace) class to get a reference to the type of the object. Later we can use “Activator.CreateInstance” to create an instance of the class. The “GetType()” function helps us to get reference to the type of the object.

```
var myAssembly = Assembly.LoadFile(@"C:\ClassLibrary1.dll");  
var myType = myAssembly.GetType("ClassLibrary1.Class1");  
dynamic objMyClass = Activator.CreateInstance(myType);  
// Get the class type  
Type parameterType = objMyClass.GetType();
```

**Step 2 :-** Once we have reference the type of the object we can then call “GetMembers” or “GetProperties” to browse through the methods and properties of the class.

```
// Browse through members  
foreach (MemberInfo objMemberInfo in parameterType.GetMembers())  
{Console.WriteLine(objMemberInfo.Name);}   
  
// Browse through properties.  
foreach (PropertyInfo objPropertyInfo in parameterType.GetProperties())  
{Console.WriteLine(objPropertyInfo.Name);}
```

In case you want to invoke the member which you have inspected you can use “InvokeMember” to invoke the method. Below is the code for the same.

```
parameterType.InvokeMember("Display",BindingFlags.Public | BindingFlags.NonPublic |  
BindingFlags.InvokeMethod | BindingFlags.Instance,null, objMyClass, null);
```

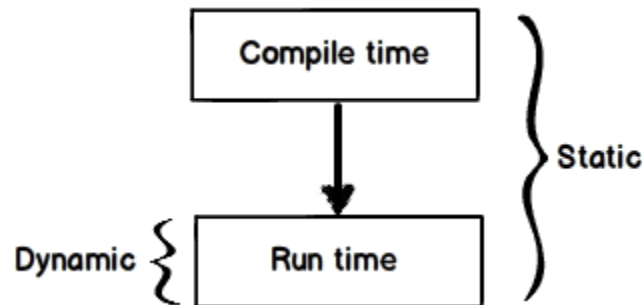
## What are the practical uses of reflection?

- If you are creating application like visual studio editors where you want show internal of an object by using intellisense.
- In unit testing sometimes we need to invoke private methods. That’s a different thing test private members are proper or not.
- Sometimes we would like to dump properties, methods and assembly references to a file or probably show it on a screen.

## What is the use of Dynamic keyword?

Programming languages can be divided in to two categories strongly typed and dynamically typed. Strongly typed languages are those where the checks happen during compile time while dynamic languages are those where type checks are bypassed during compile time. In dynamic

language object types are known only during runtime and type checks are activated only at run time.



**Figure 2.4:- Dynamic Keyword**

So we would like to take advantage of both the world. Because many times we do not know object type until the code is executed. In other words we are looking at something like dynamically statically typed kind of environment. That's what dynamic keyword helps us with. If you create a variable using the "Dynamic" keyword and if you try to see members of that object you will get a message as shown below "will be resolved at runtime".

```
dynamic x = "c#";
```

```
x.|
```

(dynamic expression)  
This operation will be resolved at runtime.

**Figure 2.5:- Dynamic X**

Now try the below code out. In the below code I have created a dynamic variable which is initialized with a string data. And in the second line I am trying to have fun by trying to execute a numeric incremental operation. So what will happen now?.... think.

```
dynamic x = "c#";
```

```
x++;
```

Now this code will compile fine without any complains. But during runtime it will throw an exception complaining that the mathematical operations cannot be executed on the variable as it's a string type. In other words during runtime the dynamic object gets transformed from general data type to specific data type (ex: - string for the below code).



Figure 2.6:- String

## What are practical uses of Dynamic keyword?

One of the biggest practical uses of dynamic keyword is when we operate on MS office components via interop.

So for example if we are accessing Microsoft excel components without dynamic keyword , you can see how complicated the below code is. Lots of casting happening in the below code, right.

```
// Before the introduction of dynamic.
Application excelApplication = new Application();
(Excel.Range)excelApp.Cells[1, 1].Value2 = "Name";
Excel.Range range2008 = (Excel.Range)excelApp.Cells[1, 1];
```

Now look at how simple the code becomes by using the dynamic keyword. No casting needed and during runtime type checking also happens.

```
// After the introduction of dynamic, the access to the Value property and
// the conversion to Excel.Range are handled by the run-time COM binder.
dynamic excelApp = new Application();
excelApp.Cells[1, 1].Value = "Name";
Excel.Range range2010 = excelApp.Cells[1, 1];
```

## What is the difference between Reflection and Dynamic?

- Both reflection and dynamic are used when we want to operate on an object during runtime.
- Reflection is used to inspect meta-data of an object. It also has the ability to invoke members of an object on runtime.

- Dynamic is a keyword which was introduced in .NET 4.0. It evaluates object calls during runtime. So until the method calls are made compiler is least bothered if those methods / properties etc exist or not.
- Dynamic uses reflection internally. It caches the method calls made thus improving performance to a certain extent.
- Reflection can invoke both public and private members of an object while dynamic can only invoke public members.

Below is the detail comparison table which shows in which scenario they are suited.

	<b>Reflection</b>	<b>Dynamic</b>
<b>Inspect ( Meta-data)</b>	Yes	No
<b>Invoke Public members</b>	Yes	Yes
<b>Invoke Private members</b>	Yes	No
<b>Caching</b>	No	Yes

Below is a simple diagram which summarizes visually what reflection can do and what dynamic keyword can do?

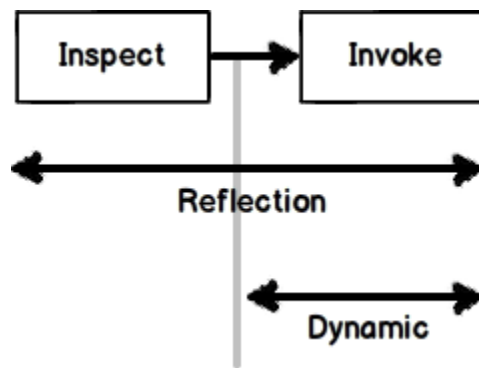


Figure 2.7:- Reflection

## Explain the difference between early binding and late binding?

Early binding or early bound means the target methods, properties and functions are detected and checked during compile time. If the method / function or property does not exist or has a data type issues then the compiler throws an exception during compile time.

Late binding is opposite of early binding. Methods, properties and functions are detected during runtime rather than compile time. So if the method, function or property does not exist during runtime application will throw an exception.

Note :- In simple words everything is early binded until you are using reflection or dynamic keyword.

## What is the difference between VAR and Dynamic keyword?

Note :- Comparing VAR and Dynamic keyword is like comparing Apples and oranges. Many people confuse VAR with the variant datatype of VB6 and there's where interviewer tries to confuse you. But there is absolute no connection of var c# keyword with variant of VB6.

VAR is early binded (statically checked) while DYNAMIC is late binded (dynamically evaluated).

VAR keyword looks at your right hand side data and then during compile time it decides the left hand side data type. In other words VAR keyword just saves you typing lot of things.

```
var x = "IamVar";  
int y = x.Length;  
(local variable) string x
```

Figure 2.8:- VAR

On the other hand dynamic keyword is for completely different purpose. Dynamic objects are evaluated during runtime. For instance in the below code the “Length” property exists or not is evaluated during runtime.

```
dynamic z = "Dynamic";  
y = z.Length; // Is Length present or not checked during runtime
```

If for some reason you mistype the “Length” property to “length” (Small “l”) you would end up in to a runtime exception as shown in the below figure.

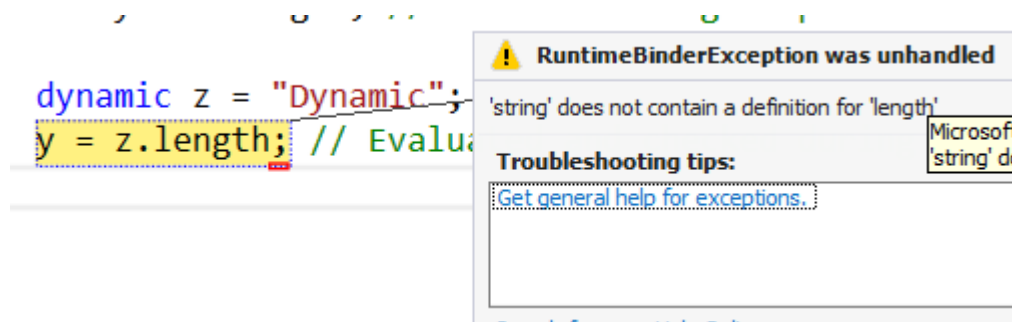


Figure 2.9:- Length

So in short sentence VAR does static check while DYNAMIC evaluates and checks during runtime.

## Explain term type safety and casting in C#?

There are two types of languages one which is type safe and one which is not. Type safety means preventing type errors. Type error occurs when data type of one type is assigned to other type **UNKNOWINGLY** and we get undesirable results.

For instance JavaScript is a **NOT** a type safe language. In the below code “num” is a numeric variable and “str” is string. Javascript allows me to do “num + str”, now GUESS will it do arithmetic or concatenation .

Now for the below code the results are “55” but the important point is the confusion created what kind of operation it will do.

This is happening because javascript is not a type safe language. Its allowing to set one type of data to the other type without restrictions.

```
<script>
var num = 5; // numeric
var str = "5"; // string
var z = num + str; // arithmetic or concat ???
alert(z); // displays "55"
</script>
```

C# is a type safe language. It does not allow one data type to be assigned to other data type. The below code does not allow “+” operator on different data types.

```
int num = 5;
string str = "5";
int total = num + str;
```

(local variable) string str

Error:

Cannot implicitly convert type 'string' to 'int'

Figure 2.10:- Type Safe



But in certain scenarios we would like to convert the data type to achieve the given results that's where we need to do conversion or casting. The next question talks about the same in detail.

## Explain casting, implicit conversion and explicit conversion?

Note :- Some of interviewers term this conversion as casting and some just prefer to call it as conversion, at the end of the day they mean the same thing.

To understand implicit and explicit conversion of data types first let's try to understand type casting concept. Type casting is a mechanism where we convert one type of data to other type.

For example below is simple code where we want to move integer (value without decimals) value to a double (value with decimals). When we try to move double data type to integer data type casting occurs. Casting is also termed as conversion.

```
int i = 100;
double d = 0;
d = i; // casting
```

In the above example there is no loss of data. In other words when we moved 100 integer value to double we get the 100 value as it. This is termed as implicit conversion / casting.

Now consider the below code where we are trying to move a double to an integer. In this scenario in integer only 100 will be received decimals will be removed. You can also see I need to explicitly specify that we need to convert in to integer. This is termed as explicit conversion.

```
double d = 100.23;
int i = 0;
i = (int) d; // this will have 100, 0.23 will truncated.
```

## What are stack and heap?

Stack and heap are memory types in an application. Stack memory stores data types like int, double, Boolean etc. While heap stores data types like string and objects.

For instance when the below code runs the first two variables i.e. "i" and "y" are stored in a stack and the last variable "o" is stored in heap.

```
void MyFunction()
```



```
{  
int i = 1; // This is stored in stack.  
int y = i; // This is stored in stack.  
object o = null; // This is stored in heap.  
} // after this end the stack variable memory space is reclaimed while // the heap memory is reclaimed later  
by garbage collector.
```

## What are Value types and Reference types?

Value types contain actual data while reference types contain pointers and the pointers point to the actual data.

Value types are stored on stack while reference types are stored on heap. Value types are your normal data types like int , bool , double and reference types are all objects.

## What is concept of Boxing and Unboxing?

When value type is moved to a reference type it's called as boxing. The vice-versa is termed as unboxing.

Below is sample code of boxing and unboxing where integer data type is converted in to object and then vice versa.

```
int i = 1;  
object obj = i;           // boxing  
int j = (int) obj; // unboxing
```

## How performance is affected due to boxing and unboxing?

When boxing and unboxing happens the data needs to jump from stack memory to heap and vice-versa which is a bit of memory intensive process. As a good practice avoid boxing and unboxing where ever possible.

## How can we avoid boxing and unboxing?

First thing it's very difficult to avoid boxing and unboxing. For instance most of the time you will moving data from UI objects like text boxes etc to business objects and also vice versa which will demand boxing and unboxing. Below are some key points to remember:-

- First thing is it really necessary to use boxing and unboxing. If it's unavoidable like moving data from UI text boxes to internal c# data types, then go for it.
- Try to see if you can use generics and avoid it.

## **If we have a class referencing value type, where is value type stored ?**

The value type will be stored in a heap.

## **Are static variables stored on a heap or stack ?**

Static variables even if its value type is stored in a heap.

## **How to prevent my .NET DLL to be decompiled?**

By design, .NET embeds rich Meta data inside the executable code using MSIL. Any one can easily decompile your DLL back using tools like ILDASM (owned by Microsoft) or Reflector for .NET which is a third party. Secondly, there are many third party tools, which make this decompiling process a click away. So any one can easily look in to your assemblies and reverse engineer them back in to actual source code and understand some real good logic, which can make it easy to crack your application.

The process by which you can stop this reverse engineering is using “obfuscation”. It is a technique, which will foil the decompilers. Many third parties (XenoCode, Demeanor for .NET) provide .NET obfuscation solution. Microsoft includes one that is Dotfuscator Community Edition with Visual Studio.NET.

**Note:-** We leave this as homework to reader’s compile, a DLL obfuscate it using “Dotfuscator Community Edition” which comes with Visual Studio.NET and try viewing the same using ILDASM.

---

## **What is the difference between Convert.ToString and .ToString () method?**

Just to give an understanding of what the above question means see the below code.

```
int i =0;
MessageBox.Show(i.ToString());
MessageBox.Show(Convert.ToString(i));
```

---

We can convert the integer “i” using “i.ToString()” or “Convert.ToString” so what is the difference. The basic difference between them is “Convert” function handles NULLS while “i.ToString()” does not. It will throw a NULL reference exception error. So as a good coding practice using “convert” is always safe.

## **What is the difference between String vs string?**

“String” is an alias( the same thing called with different names) of “string”. So technically both the below code statements will give the same output.

```
String s = “C# interview questions”;
or
```

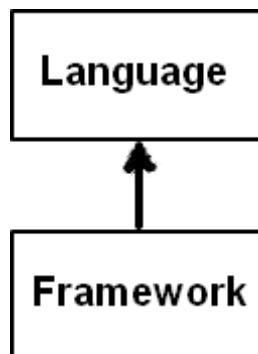
```
string s = "C# interview questions";
```

In the same way there are aliases for other c# data type as shown below:-

- object: System.Object
- string: System.String
- bool: System.Boolean
- byte: System.Byte
- sbyte: System.SByte
- short: System.Int16
- ushort: System.UInt16
- int: System.Int32
- uint: System.UInt32
- long: System.Int64
- ulong: System.UInt64
- float: System.Single
- double: System.Double
- decimal: System.Decimal
- char: System.Char

### So when both mean the same thing why are they different?

When we talk about .NET there are two different things one there is .NET framework and the other there are languages( C# , VB.NET etc) which use that framework.



**Figure 2.11:- Framework**

“System.String” a.k.a “String” ( capital “S”) is a .NET framework data type while “string” is a C# data type.

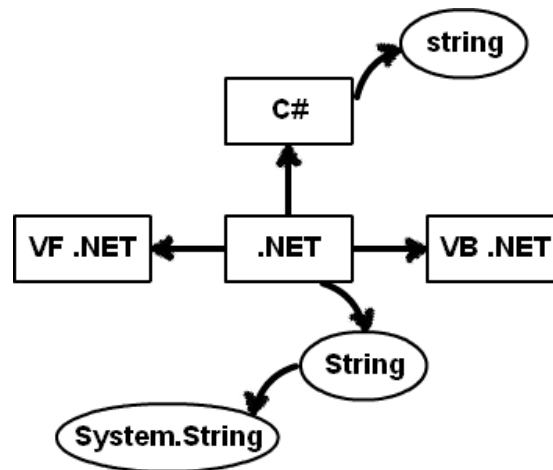


Figure 2.12:- System.String

## So when to use “String” and “string”?

First thing to avoid confusion use one of them consistently. But from best practices perspective when you do variable declaration it's good to use “string” ( small “s”) and when you are using it as a class name then “String” ( capital “S”) is preferred.

In the below code the left hand side is a variable declaration and it declared using “string”. At the right hand side we are calling a method so “String” is more sensible.

```
string s = String.ToUpper() ;
```

## How can we handle exceptions in .NET?

Exceptions are handled by “System.Exception” base class. If you want to raise an error from source you need to create the exception object with below code snippet.

```
throw new Exception("Customer code cannot be more than 10");
```

Once the exception is raised if you want to catch the same you need to use the try catch block as shown below.

```
try
{
// This section will have the code which
// which can throw exceptions.
}
```

```
catch(Exception e)
{
    // Handle what you want to
    // do with the exception
    label.text = e.Message;
}
```

## How can I know from which source the exception occurred?

“System.Exception.StackTrace” allows you to identify series of call which lead to this exception.

## What if we do not catch the exception?

If you do not catch the error, .NET framework will handle the same and you will get a message box as shown in the below figure.

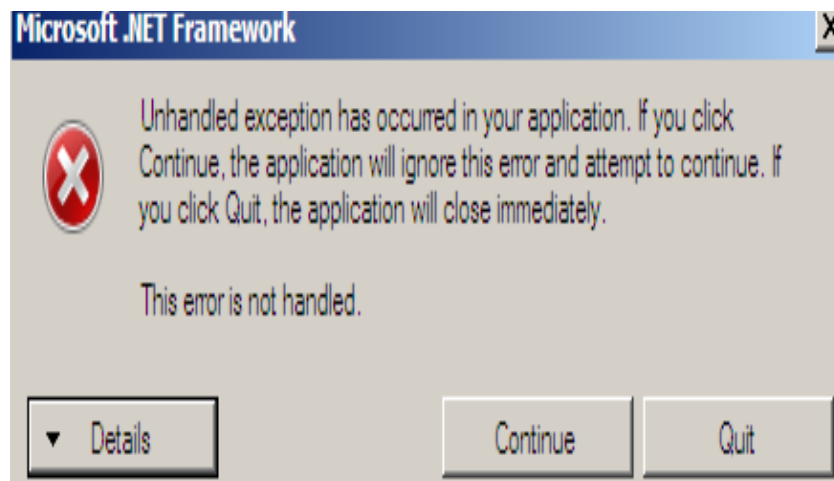


Figure 2.13: - .Net framework handles the errors

## What are system level exceptions and application level exceptions?

Exceptions that are thrown by .NET framework are called as system exceptions. These errors are non-recoverable or fatal errors like `ArgumentOutOfRangeException`, `IndexOutOfRangeException`, `StackOverflowException` etc.

Application exceptions are custom exceptions created for the application. Application exceptions are created by deriving from “ApplicationException” class as shown below. You can then create the object of the below exception and throw the same from the code and catch the same on the client side.

```
public class MyOwnException : ApplicationException
{
    private string messageDetails = String.Empty;
    public DateTime ErrorTimeStamp {get; set;}
    public string CauseOfError {get; set;}

    public MyOwnException() {}
    public MyOwnException(string message,
        string cause, DateTime time)
    {
        messageDetails = message;
        CauseOfError = cause;
        ErrorTimeStamp = time;
    }

    // Override the Exception.Message property.
    public override string Message
    {
        get
        {
            return string.Format("This is my own error message");
        }
    }
}
```

### Can two catch blocks be executed?

No, once the proper catch section is executed the control goes to the ‘finally’ block. So there is no chance by which multiple catch block will ‘execute’.

## What are different types of collections in .NET?

There are five important collections in .NET Arrays, Lists, Hashtable , stacks and queues .

## What is the difference between arraylist and list?

- Arrays are fixed in size while ArrayList is resizable.
- Arrays are strongly typed, in other words when you create an array it can store only one data type data. ArrayList can store any datatype.

## Are ArrayList faster or Arrays?

Array list takes any data type which leads to boxing and unboxing. As arrays are strongly typed they do not do boxing and unboxing. So arrays are faster as compared to array list.

```
// Array definition
```

```
int[] str = new int[10];
```

```
// Array list definition
```

```
ArrayList MyList = new ArrayList();
```

## What are hashtable collections?

In arraylist or array if we have to access any data we need to use the internal index id generated by the array list collection. For instance the below code snippet shows how the internal id is used to fetch data from array list.

In actual scenarios we hardly remember internal id's generated by collection we would like to fetch the data by using some application defined key. There's where hash table comes in to picture.

```
string str = MyList[1].ToString();
```

Hash table helps to locate data using keys as shown below. When we add data to hash table it also has a provision where we can add key with the data. This key will help us to fetch data later using key rather than using internal index id's generated by collections.

```
objHashtable.Add("p001","MyData");
```

This key is converted in to numeric hash value which is mapped with the key for quick lookup.

## What are Queues and stack collection?



Queues are collection which helps us to add object and retrieve them in the manner they were added. In other word queues helps us to achieve the first in first out collection behavior.

Stack collection helps us to achieve first in last out behavior.

## Can you explain generics in .NET?

Generics help to separate logic and data type to increase reusability. In other words you can create a class whose data type can be defined on run time.

For instance below is a simple class “class1” with a “compareme” function created using generics. You can see how the unknown data type is put in greater than and less than symbol. Even the compare me method has the unknown data type attached.

```
public class Class1<UNNKOWDATATYPE>
{

    public bool Compareme(UNNKOWDATATYPE v1, UNNKOWDATATYPE v2)
    {

        if (v1.Equals(v2))
        {
            return true;
        }
        else
        {
            return false;
        }
    }
}
```

During runtime you can define the datatype as shown in the below code snippet.

```
Class1<int> obj = new Class1<int>();
bool b = obj.Compareme(1,2); // This compares numeric
Class1<string> obj1 = new Class1<string>();
bool b1 = obj1.Compareme("shiv","shiv"); // This does string comparison
```



## Explain generic constraints and when should we use them?

Generic's helps to decouple logic from the data type. But many times some logics are very specific to specific data types.

```

public class CompareNumeric<UNKNOWDATATYPE>
{
    public bool Compareme(UNKNOWDATATYPE v1, UNKNOWDATATYPE v2)
    {
        if (v1>v2)
        {return true;}
        else
        {return false;}
    }
}
  
```

For example above is a simple generic class which does comparison if one number is greater than other number. Now the greater and less than comparison is very specific to numeric data types. These kind of comparison's cannot be done on non-numeric types like string.

So if some uses the classes with "int" type its perfectly valid.

```

CompareNumeric<int> obj = new CompareNumeric<int>();
bool boolgreater = obj.Compare(10,20);
  
```

If someone uses it with "double" data type again perfectly valid.

```

CompareNumeric<double> obj = new CompareNumeric<double>();
bool boolgreater = obj.Compare(100.23,20.45);
  
```

But using string data type with this logic will lead undesirable results. So we would like to restrict or put a constraint on what kind of types can be attached to a generic class this is achieved by using "generic constraints".

```

CompareNumeric<string> obj = new CompareNumeric<string>();
bool boolgreater = obj.Compare("interview","interviewer");
  
```

Generic type can be restricted by specifying data type using the "WHERE" keyword after the generic class as shown in the below code. Now if any client tries to attach "string" data type with the below class it will not allow, thus avoiding undesirable results.



```
public class CompareNumeric<UNNKOWDATATYPE> where UNNKOWDATATYPE : int, double
{

}

}
```

## Can you explain the concept of generic collection?

Array List, Stack and Queues provide collections which are not type safe. This leads 2 problems first it's not type safe and second it leads to boxing and unboxing.

By using generics we can have type safety and also we can avoid boxing and unboxing. Below is a simple code snippet which shows a strong typed list of type integer and string.

```
List<int> obj;
obj.add(1); // you can only add integers to this list
List<string> obj;
obj.add("shiv"); // you can only add string to this list
```

## What is the difference between dictionary and hashtable?

Dictionary collection is a generic collection equivalent for hashtable. Hashtable allows you to add key and value of any type (i.e. objects) . This leads to two problems one is boxing and unboxing issues and second it's not strongly typed.

```
// Creates a strongly typed dictionary with integer key and value
// pair
Dictionary<int,int> obj = new Dictionary<int,int>();
// We can only add integer value and key to the dictionary collection
Obj.Add(123,550);
```

## What are the generic equivalent for array list,stack, queues and hashtable?

Below are the listed generic equivalents for each of them:-

- Array list generic equivalent is List<int>.
- Stack generic equivalent is Stack<int>.
- Queue generic equivalent is Queue<int>.

- Hashtable generic equivalent is Dictionary<int,int>.

## What is the use of IEnumerable, ICollection, IList and IDictionary?

The above 4 entities are nothing but interfaces implemented by collections:-

**IEnumerable:** - All collection classes use this interface and it helps to iterate through all the collection elements.

**ICollection:** - This interface has the count property and it's implemented by all collection elements.

**IList:** - This interface allows random access, insertion and deletion in to the collection. Its implemented by array list.

**IDictionary:** - This interface is implemented by hashtable and dictionary.

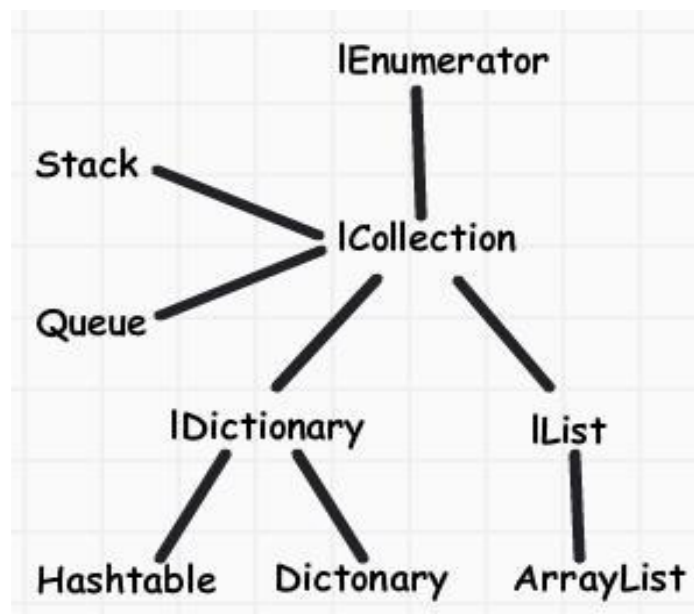
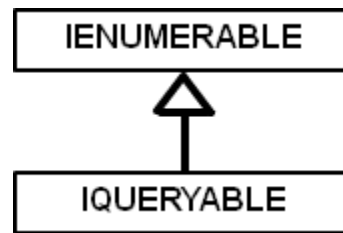


Figure 2.14: - Interface collections

## Differentiate IEnumerable vs IQueryable?

The first important point to remember is “IQueryable” interface inherits from “IEnumerable”, so whatever “IEnumerable” can do, “IQueryable” can also do.



**Figure 2.15: - IEnumerable**

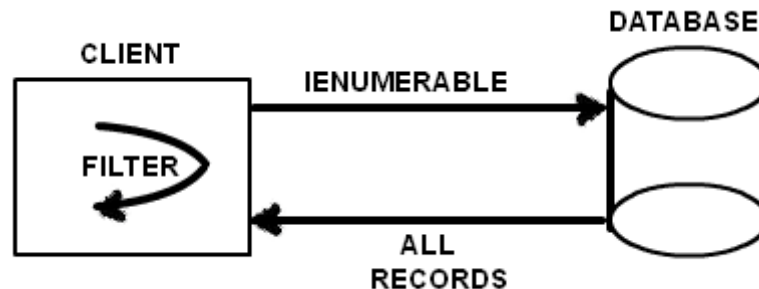
There are many differences but let us discuss about the one big difference which makes the biggest difference. “IQueryable” interface is useful when your collection is loaded using LINQ or Entity framework and you want to apply filter on the collection.

Consider the below simple code which uses “IEnumerable” with entity framework. It’s using a “where” filter to get records whose “EmpId” is “2”.

```

EmpEntities ent = new EmpEntities();
IEnumerable<Employee> emp = ent.Employees;
IEnumerable<Employee> temp = emp.Where(x => x.Empid == 2).ToList<Employee>();
  
```

This where filter is executed on the client side where the “IEnumerable” code is. In other words all the data is fetched from the database and then at the client it scans and gets the record with “EmpId” is “2”.



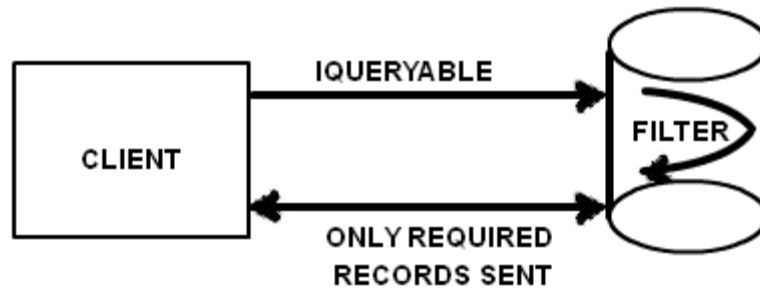
**Figure 2.16: - All Records**

But now see the below code we have changed “IEnumerable” to “IQueryable”.

```

EmpEntities ent = new EmpEntities();
IQueryable<Employee> emp = ent.Employees;
IEnumerable<Employee> temp = emp.Where(x => x.Empid == 2).ToList<Employee>();
  
```

In this case the filter is applied on the database using the “SQL” query. So the client sends a request and on the server side a select query is fired on the database and only necessary data is returned.



**Figure 2.17: - Required Record**

So the difference between “IQueryable” and “IEnumerable” is about where the filter logic is executed. One executes on the client side and the other executes on the database.

So if you working with only in-memory data collection “IEnumerable” is a good choice but if you want to query data collection which is connected with database “IQueryable” is a better choice as it reduces network traffic and uses the power of SQL language.

## **What is code access security (CAS)?**

CAS is the part of .NET security model which determines whether or not a particular code is allowed to run and what kind of resources can the code access.

## **So how does CAS actually work?**

It’s a four step process:-

- First Evidence is gathered about the assembly. In other words from where did this assembly come? , who is the publisher etc.
- Depending on evidences the assembly is assigned to a code group. In other words what rights does the assembly depending on the evidence gathered.
- Depending on code group security rights are allocated.
- Using the security rights the assembly is run with in those rights

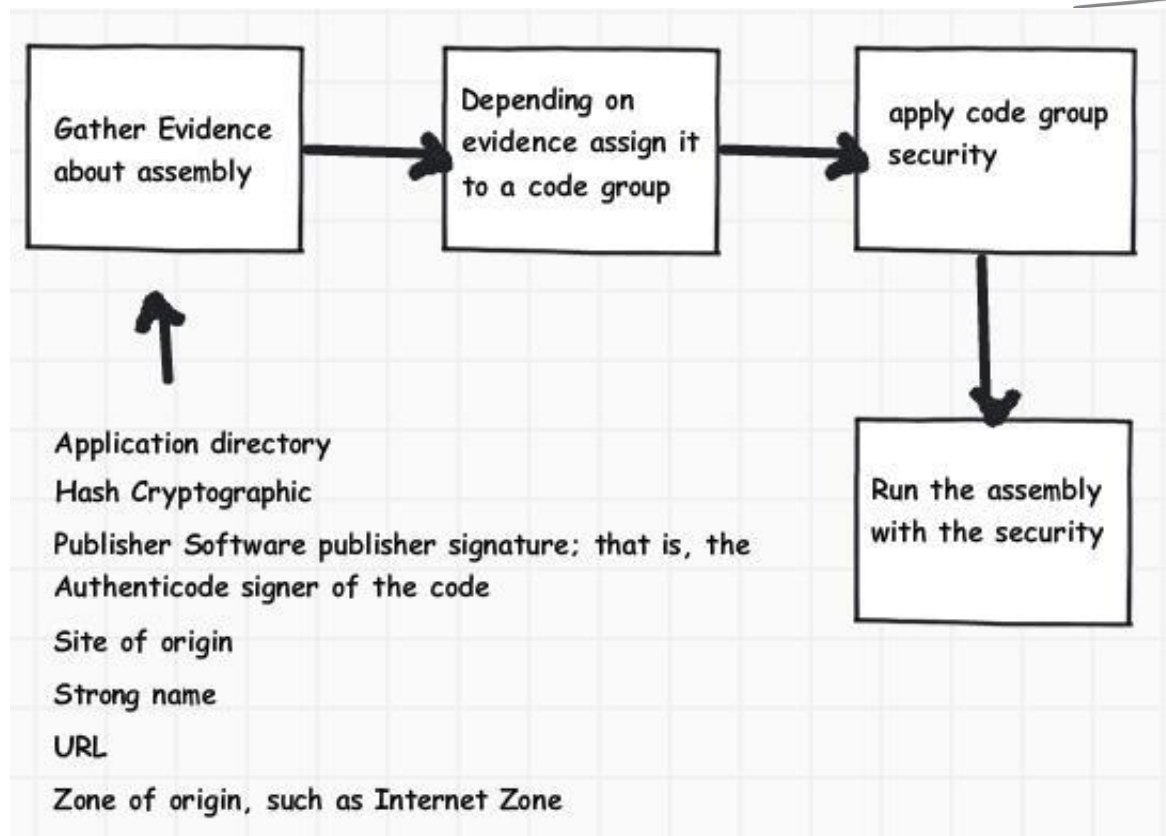


Figure 2.18: - CAS in action

## Is CAS supported in .NET 4.0?

CAS is deprecated in .NET 4.0 and two major changes are brought in:-

- Permission granting is no more the work of CAS; it's now the work of the hosting model. In other words CAS is disabled in .NET 4.0 by default. The host will decide what rights to be given to the .NET assembly.
- A new security model i.e. Security transparent model is introduced. The security transparent model puts code in to separate compartments/ boxes as per the risk associated. If you know a code can do something wrong you can compartmentalize the code as 'Security transparent' and if you have a code which you trust you can box them in to 'Security critical'.

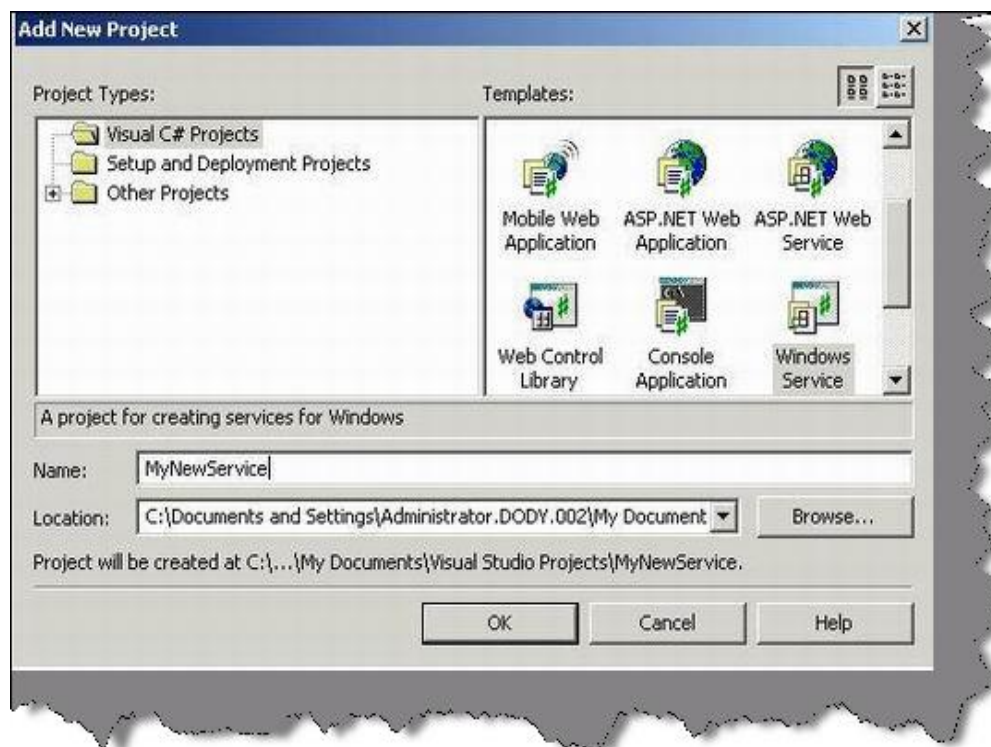
## What is sandboxing?

If you have want to execute an untrusted third party DLL, you can create your own 'appdomain' and assign permission sets so that your 3<sup>rd</sup> party DLL runs under a control environment.

## How can we create a windows service using .NET?

Windows Services are long-running processes that run at the background. It has the ability to start automatically when the computer boots and also can be manually paused, stopped or even restarted.

Following are the steps to create a service:-  
Create a project of type “Windows Service”.



**Figure 2.19:- Windows service**

If you see, the class created it is automatically inheriting from “System.ServiceProcess.ServiceBase”.

You can override the following events provided by service and write your custom code. All the three main events can be used that is Start, stop and continue.

```
protected override void OnStart(string[] args)
```

```
{
}
```

```
protected override void OnStop()
```

```
{
```

```

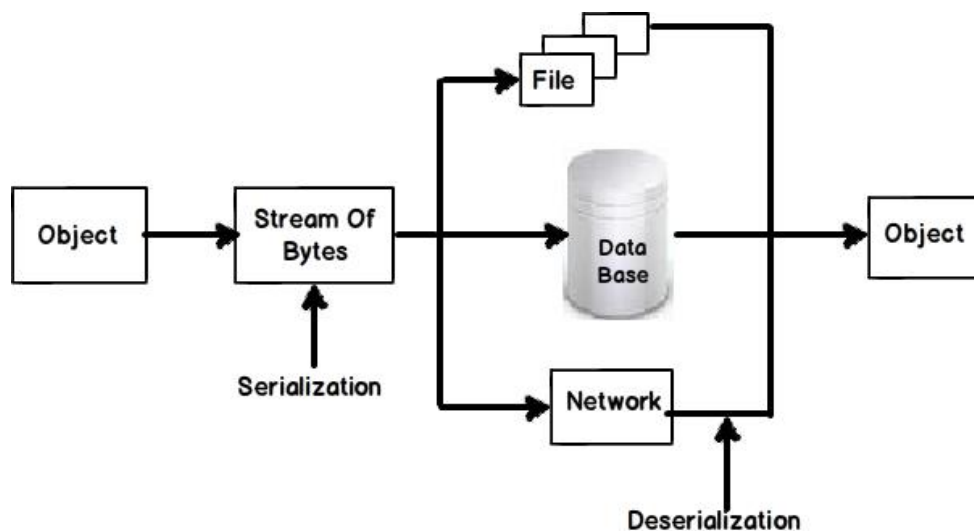
}
protected override void OnContinue()
{
}

```

Now to install the service you need to do run the install util exe.  
 InstallUtil <Project Path>\BIN\MyNewService.exe

## What is serialization and deserialization in .NET?

Serialization is a process where we can convert an object state in to stream of bytes. This stream can then be persisted in a file, database, or sent over a network etc. Deserialization is just vice-versa of serialization where we convert stream of bytes back to the original object.



**Figure 2.20: - serialization and deserialization in .NET**

Below is a simple code of how to serialize and de-serialize an object.

Let's first start with serialization.

### Step 1:- Create the object and put some values

```

// Serialization
Customer obj = new Customer(); // Create object
obj.CustomerCode = 123; // Set some values

```



**Step 2:- Create the file where to save the object.**

```
using System.Runtime.Serialization;  
using System.Runtime.Serialization.Formatters.Binary;  
  
IFormatter i = new BinaryFormatter(); // Use the binary formatter  
Stream stream = new FileStream("MyFile.txt", FileMode.Create, FileAccess.Write, FileShare.None); // Give  
file name
```

---

**Step 3:- Use serialize method to save it to hard disk**

```
i.Serialize(stream, o); // write it to the file  
stream.Close(); // Close the stream
```

---

Let's also see a simple example of de-serialization.

**Step 1:- Read the file**

```
// De-Serialization  
IFormatter formatter = new BinaryFormatter(); // Use binary formatter  
Stream stream = new FileStream("MyFile.txt", FileMode.Open, FileAccess.Read, FileShare.Read); // read the  
file
```

---

**Step 2:- Recreate it back to the original object.**

```
Customer obj = (Customer)formatter.Deserialize(stream); // take data back to object  
stream.Close(); // close the stream
```

---

If you want to save to XML or any other content type use the appropriate formatter.

**Can you mention some scenarios where we can use serialization?**

Below are some scenarios where serialization is needed:-

- Passing .NET objects across network. For example .NET remoting , web services or WCF services use serialization internally.
- Copy paste .NET objects on clip board.
- Saving old state of the object and reverting back when needed. For example when the user hits the cancel button you would like to revert back to the previous state.

## When should we use binary serialization as compared to XML serialization?

- Binary is smaller in size, so faster to send across network and also faster to process.
- XML is more verbose, but easy to understand and human readable. But due to the XML structure its complex to parse and can impact performance.

Note :- Many people answer that XML serialization should be used when you have different platforms and binary serialization should be used when you same platforms. But this answer is actually not right as we have lot of binary serialization methodology like ASN , protobuf which are cross platform and widely accepted. So the important difference is that one is XML which readable by eyes and the binary is not.

## What is Regular expression?

Regular expression is a pattern matching technique. Most of the data follow patterns, some examples of common data patterns are as shown in the below table :-

Email Address	xyz@pqr.com abc@lmn.com	Email address data starts with some characters followed by “@” and then a “.” and finally the domain extension.
Phone number	901-9090-909023 909-999-1202920	Phone number data starts with an international code followed by region code and then the phone number.

By using regular expression you can represent these data patterns and utilize the same for validations and manipulation of data.

In order to use regular expression we need to create the object of "Regex" class and pass the pattern to Regex. For example the below pattern represent's data pattern of 10 character length which can be any small characters from a to z.

```
Regex obj = new Regex("[a-z]{10}");
```

Once you have specified the pattern as described in the previous step you can then use the "IsMatch" function to check if the data matches with the pattern.

```
if(obj.IsMatch("shivkoirala"))
```

```
{
//                                proper                                data
}
else
```

```
{  
//                                improper                                data  
}
```

Note :- In some rare cases we have seen interviewers ask to write down regex patterns. So we would suggest to see the video “Explain Regex ?” provided in the DVD which will help you to write down any regex pattern easily.

## What is time out support in regex (regular expression)?

This is a new feature of .NET 4.5. Some of the regular expressions are very complex and they can take lot of time to evaluate.

For instance below is a simple regular expression.

```
var regex = new Regex(@"^(\\d+)$", RegexOptions.Singleline);
```

---

If someone inputs a huge number as shown in the below code snippet. It will take more than a minute to resolve the expression leading to lot of load on the application. So we would like to give a time out on the expression. So if the validation takes more than a specific interval we would like the application to give up and move ahead.

```
var match = regex.Match("123453109839109283090492309480329489812093809x");
```

---

In .NET 4.5 we can now provide the timeout value as parameter on the constructor. For instance in the below code we have provided 2 seconds timeout on the regex (regular expression). So if it exceeds more than 2 second “regexMatchTimeoutException” will occur.

```
try  
{  
var regex = new Regex(@"^(\\d+)$", RegexOptions.Singleline, TimeSpan.FromSeconds(2));  
var match = regex.Match("123453109839109283090492309480329489812093809x");  
}  
  
catch (RegexMatchTimeoutException ex)  
{  
Console.WriteLine("Regex Timeout");  
}
```

---

## Can you explain the concept of “Short Circuiting”?

Short circuiting occurs when you do logical operations like ‘AND’ and ‘OR’.

“When we use short circuit operators only necessary evaluation is done rather than full evaluation.”

Let us try to understand the above sentence with a proper example. Consider a simple “AND” condition code as shown below. Please note we have only one “&” operator in the below code.

```
if(Condition1 & Condition2)
{
}
```

---

In the above case “Condition2” will be evaluated even if “Condition1” is “false”. Now if you think logically, it does not make sense to evaluate “Condition 2”, if “Condition 1” is false .It’s a AND condition right? , so if the first condition is false it means the complete AND condition is false and it makes no sense to evaluate “Condition2”.

There’s where we can use short circuit operator “&&”. For the below code “Condition 2” will be evaluated only when “Condition 1” is true.

```
if(Condition1 && Condition2)
{
}
```

---

The same applies for “OR” operation. For the below code (please note its only single pipe (“|”).) “Condition2” will be evaluated even if “Condition1” is “true”. If you think logically we do not need to evaluate “Condition2” if “Condition1” is “true”.

```
if(Condition1 | Condition2)
{
}
```

---

So if we change the same to double pipe (“||”) i.e. implement short circuit operators as shown in the below code, “Condition2” will be evaluated only if “Condition1” is “false”.

```
if(Condition1 || Condition2)
{
}
```

---

## What is the difference between “Typeof” and “GetType” ?

Both “TypeOf” and “GetType” help you to get the type. The difference is from where the information is extracted. “TypeOf” gets the type from a class while “GetType” gets type from an object.

## Will the following c# code compile?

```
double          dbl          = 109.22;
int             i             = 109;
```

No, the above code will give an error. Double size is larger than “int” so an implicit conversion will not take place. For that we need to do explicit conversion. So we need to something as shown in the below code. In the below code we have done an explicit conversion.

```
double          dbl          = 109.22;
int             i             = (int) 109; // this is explicit conversion / casting
```

Also note after explicit conversion we will have data loss. In variable “i” we will get “109” value , the decimal values will be eliminated.

## Explain the use of Icomparable in c#?

“IComparable” interface helps to implement custom sorting for collections in c#. Let us try to understand the same with a simple c# example. For instance let’s say you have a list collection of people names as shown in the below code.

```
List<string> Peoples = new List<string>();
Peoples.Add("Shiv");
Peoples.Add("Raju");
Peoples.Add("Sukesh");
Peoples.Add("Ajay");

foreach (string str in Peoples)
{
    Console.WriteLine(str);
}
```

Now if you run the above program you will see that we have not applied sort on the collection. It displays data as they were inserted.

```
Shiv
Raju
Sukesh
Ajay
```

But now if you call “Peoples.Sort()” method on the collection you will get the following sorted output on the “name” value in ascending position.

```
Ajay  
Raju  
Shiv  
Sukesh
```

---

But now let’s say that we need the person’s age also to be added in the list with the person name value. So logically you will create a “Person” class with “name” and “age” property as shown in the code below.

```
class Person  
{  
    private string _Name;  
  
    public string Name  
    {  
        get { return _Name; }  
        set { _Name = value; }  
    }  
  
    private int _Age;  
  
    public int Age  
    {  
        get { return _Age; }  
        set { _Age = value; }  
    }  
}
```

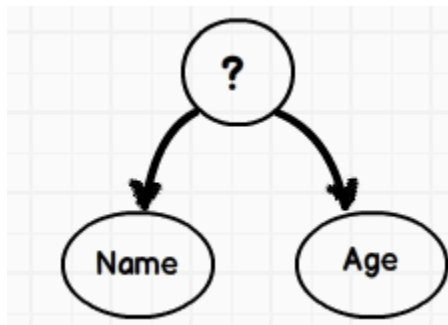
---

Once you create the class you would create an object of the person class and add it to the list collection as shown in the below code snippet.

```
class Program
{
    static void Main(string[] args)
    {
        List<Person> Peoples = new List<Person>();

        Peoples.Add(Createperson("Shiv",20));
        Peoples.Add(Createperson("Raju", 20));
        Peoples.Add(Createperson("Sukesh", 30));
        Peoples.Add(Createperson("Ajay", 40));
        Peoples.Sort();
        foreach (Person obj in Peoples)
        {
            Console.WriteLine(obj.Name + " " + obj.Age);
        }
    }
    static Person Createperson(string Name, int Age)
    {
        Person obj = new Person();
        obj.Name = Name;
        obj.Age = Age;
        return obj;
    }
}
```

But if you now try to call the “Sort” method on the List, he gets confused ?. On which value should he sort on “Name” or “Age” ?.



**Figure 2.21:- IComparable**

If you run the application it will throw up the below exception text. The exception text says that he is confused and he needs specific direction on how the sorting should work. This direction can be provided by using “IComparable” interface.

---

Unhandled Exception: System.InvalidOperationException: Failed to compare two elements in the array. ---> System.ArgumentException: At least one object must implement IComparable.

---

So in order to show a proper direction for the sort logic we need to implement “IComparable” interface as shown in the below. The logic of “Sort” needs to be defined in the “CompareTo” method.

```

class Person : IComparable<Person>
{
    private string _Name;

    public string Name
    {
        get { return _Name; }
        set { _Name = value; }
    }

    private int _Age;
    public int Age
    {
        get { return _Age; }
        set { _Age = value; }
    }
}
  
```



```
public int CompareTo(Person other)
{
    if (other.Age == this.Age)
    {
        return this.Name.CompareTo(other.Name);
    }
    else
    {
        return other.Age.CompareTo(this.Age);
    }
}
```

You can see in the “CompareTo” method we have defined the logic saying if the “age” value is same then sort by using the “name” value or else use “age” value for sorting. If you run the application you would get the following output. You can see for “40” and “30” value he has sorted on the basis “Age” but for the remaining people the age is same so he sorted on the “Name” value.

```
Loki 40
Sukesh 30
Ajay 20
Madan 20
Raju 20
Shiv 20
```

So the use of “IComparable” interface is to define custom sorting logic on a collection.

### **What is difference between Icomparable and IComparer ?**

Pre-requisite:- Please read “Explain the use of Icomparable in c#?” , before reading this answer.

“IComparable” interface helps you to implement a default sort implementation for the collection. But what if we want to sort using multiple criteria’s?.For those instances “IComparable” has a limitation and “IComparer” is the guy.

For example if we want to sort the list by “Name” under some situation or sort by “Age” under some other situations we need to implement “IComparer” interface.

So the first step is to create different classes for each sort criteria. These classes will implement “IComparer” interface and will have sorting logic defined in the “Compare” method. You can see we have two different classes defined “CompareByName” and “CompareByAge”. One compares on the basis of “name” and the other on the basis of “age”.

```
class CompareByName : IComparer<Person>
{
    public int Compare(Person x, Person y)
    {
        return string.Compare(x.Name, y.Name);
    }
}
class CompareByAge : IComparer<Person>
{
    public int Compare(Person x, Person y)
    {
        if (x.Age > y.Age) return 1;
        if (x.Age < y.Age) return -1;
        return 0;
    }
}
```

If you see the logic for “CompareByName” its simple. It uses the string comparison to evaluate the “Name” value sorting. But when we talk about numeric comparison there are 3 possible outputs Greater than , less than or Equal. So if you see “CompareByAge” class it returns 3 values 1 ( Greater than ) , -1 ( Less than ) and 0 ( for Equal to ).

Now that we have created the separate logics we can now invoke the “Peoples’ list by passing the class object. So for example if we want to sort by name, you will use the below code snippet.

```
Peoples.Sort(new CompareByName());
```

The output of the above code is nothing but alphabetically sorted data.

Ajay 20

Loki 40  
 Madan 20  
 Raju 20  
 Shiv 20  
 Sukesh 30

If you invoke the sort method of the list by using the age logic it will throw an output sorted on age value.

```
Peoples.Sort(new CompareByAge());
```

Ajay 20  
 Raju 20  
 Shiv 20  
 Madan 20  
 Sukesh 30  
 Loki 40

So the difference is really default internal implementation or customizable external implementation. When we use “IComparable” we can have only one default sorting logic and that logic goes inside the collection itself. For “IComparer” the logic is outside the collection , in other words more extensible and the collection is not disturbed.

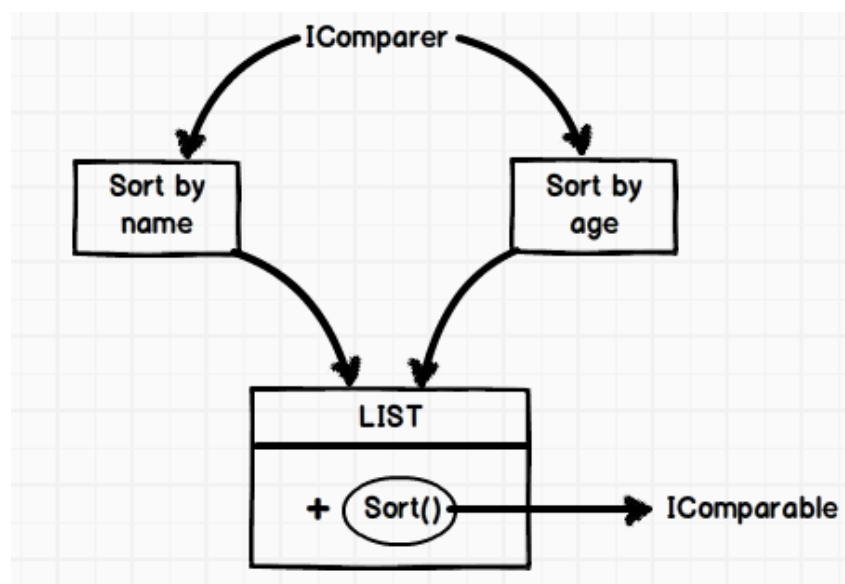


Figure 2.22:- IComparer vs IComparable



## Can you explain Lazy Loading?

Lazy loading is a concept where we delay the loading of the object until the point where we need it. Putting in simple words on demand object loading rather than loading the objects unnecessarily.

For example consider the below example where we have a simple “Customer” class and this “Customer” class has many “Order” objects inside it. Have a close look at the constructor of the “Customer” class. When the “Customer” object is created it also loads the “Order” object at that moment. So even if we need or do not need the address object, it’s still loaded.

But how about just loading the “Customer” object initially and then on demand basis load the “Order” object.

```
public class Customer
{
    private List<Order> _Orders= null;
    ...
    ...
    public Customer()
    {
        _CustomerName = "Shiv";
        _Orders = LoadOrders(); // Loads the address object even though //not needed
    }

    private List<Order> LoadOrders()
    {
        List<Order> temp = new List<Order>();
        Order o = new Order();
        o.OrderNumber = "ord1001";
        temp.Add(o);
        o = new Order();
        o.OrderNumber = "ord1002";
        temp.Add(o);
        return temp;
    }
}
```

So let's consider you have client code which consumes the "Customer" class as shown below. So when the "Customer" object is created no "Order" objects should be loaded at that moment. But as soon as the "foreach" loop runs you would like to load the "Order" object at that point ( on demand object loading).

```
Customer o = new Customer(); // Address object not loaded
Console.WriteLine(o.CustomerName);
foreach (Order o1 in o.Orders) // Load address object only at this moment
```

```
{  
    Console.WriteLine(o1.OrderNumber);  
}
```

---

## So how do we implement “LazyLoading” ?

So for the above example if we want to implement Lazy loading we will need to make the following changes:-

- Remove the “Order” object loading from the constructor.
- In the “Order” get property, load the “Order” object only if it’s not loaded.

```
public class Customer  
{  
    private List<Order> _Orders= null;  
    ...  
    ...  
    public Customer()  
    {  
        _CustomerName = "Shiv";  
    }  
  
    public List<Order> Orders  
    {  
        get  
        {  
            if (_Orders == null)  
            {  
                _Orders = LoadOrders();  
            }  
            return _Orders;  
        }  
    }  
}
```

---

Now if you run the client code and halt your debugger just before the “ForEach” loop runs over the “Orders” object, you can see the “Orders” object is null ( i.e. not loaded). But as soon as the “ForEach” loop runs over the “Order” object it creates the “Order” object collection.

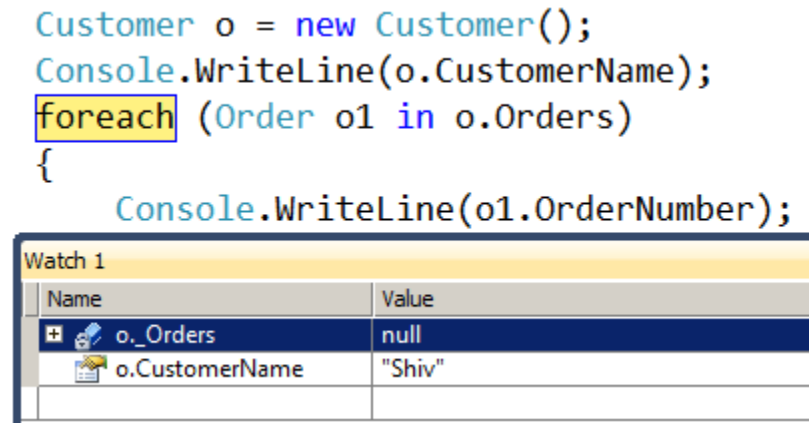


Figure 2.23: - For Each Loop

### Are there any readymade objects in .NET by which we can implement Lazy loading?

In .NET we have “Lazy<T>” class which provides automatic support for lazy loading. So let’s say if you want to implement “Lazy<>” in the above code we need to implement two steps for the same:-

Create the object of orders using the “Lazy” generic class.

```
private Lazy<List<Order>> _Orders = null;
```

Attach this Lazy<> object with the method which will help us load the order’s data.

```
_Orders = new Lazy<List<Order>>(() => LoadOrders());
```

Now as soon as any client makes a call to the “\_Orders” object ,it will call the “LoadOrders” function to load the data.

You will get the “List<Orders>” data in the “Value” property.

```
public List<Order> Orders
{
    get
    {
```

```
        return _Orders.Value;  
    }  
  
}
```

Below goes the full code for the same.

```
public class Customer  
{  
    private Lazy<List<Order>> _Orders= null;  
  
    public List<Order> Orders  
    {  
        get  
        {  
            return _Orders.Value;  
        }  
    }  
  
    public Customer()  
    {  
        // Makes a database trip  
        _CustomerName = "Shiv";  
        _Orders = new Lazy<List<Order>>(() => LoadOrders());  
    }  
}
```

## What are the advantages / disadvantages of lazy loading?

Below are the advantages of lazy loading:-



- Minimizes start up time of the application.
- Application consumes less memory because of on-demand loading.
- Unnecessary database SQL execution is avoided.

The disadvantage is that the code becomes complicated. As we need to do checks if the loading is needed or not. So must be there is a slight decrease in performance.

But the advantages of are far more than the disadvantages.

FYI :- The opposite of Lazy loading is Eager loading. So in eager loading we load the all the objects in memory as soon as the object is created.

### What is the difference between “IS” and “AS” keyword ?

“IS” keyword is useful to check if objects are compatible with a type. For instance in the below code we are checking if “ocust” object is a type of “Customer” class.

```
object ocust = new Customer();  
  
if (ocust is Customer)  
{
```

“AS” keyword helps to do conversion from one type to other type. For instance in the below code we are converting object to a string data type. If the “AS” keyword is not able to type cast it returns NULL.

```
object o = "interview";  
string str = o as string;
```

### What is the use of “Yield” keyword?

“Yield helps you to provide custom stateful iteration over .NET collections.”

There are two scenarios where “yield” keyword is useful:-

- Customized iteration through a collection without creating a temporary collection.
- Stateful iteration.

#### Scenario 1:- Customized iteration through a collection

Let's try to understand what customized iteration means with an example. Consider the below code.

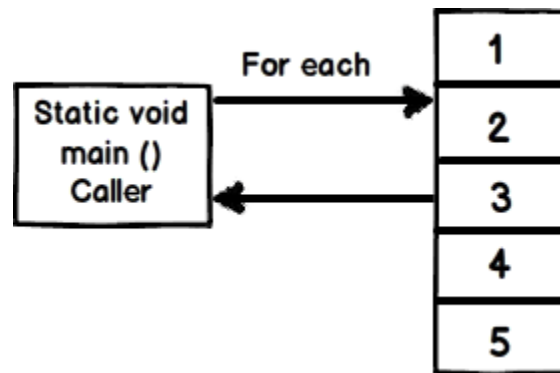
Let say we have a simple list called as "MyList" which has collection of 5 continuous numeric values 1,2,3,4 and 5. This list is browsed/iterated from console application from within static void main method.

For now let's visualize the "main()" method as a caller. So the caller i.e. "main()" method calls the list and displays the items inside it. Simple...till now ;-).

```
static List<int> MyList = new List<int>();

static void FillValues()
{
    MyList.Add(1);
    MyList.Add(2);
    MyList.Add(3);
    MyList.Add(4);
    MyList.Add(5);
}

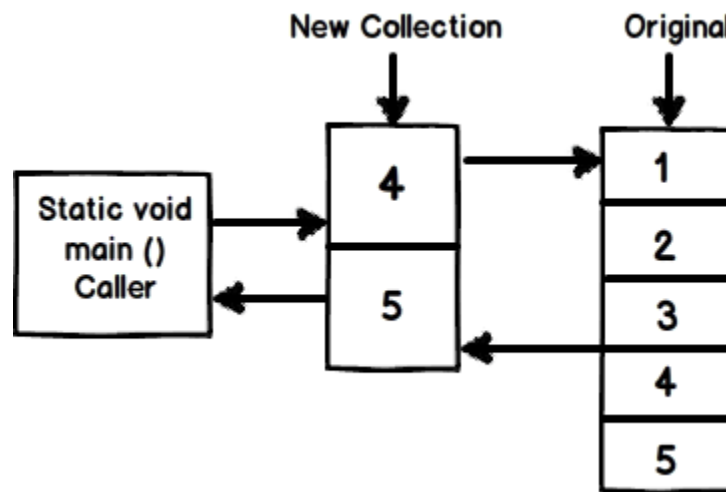
static void Main(string[] args) // Caller
{
    FillValues(); // Fills the list with 5 values
    foreach (int i in MyList) // Browses through the list
    {
        Console.WriteLine(i);
    }
    Console.ReadLine();
}
```



**Figure 2.24: - My List**

Now let me complicate this situation lets say the caller only wants values greater than “3” from the collection. So the obvious thing as a c# developer we will do is create a function as shown below. This function will have temporary collection. In this temporary collection we will first add values which are greater than “3” and return the same to the caller. The caller can then iterate through this collection.

```
static IEnumerable<int> FilterWithoutYield()
{
    List<int> temp = new List<int>();
    foreach (int i in MyList)
    {
        if (i > 3)
        {
            temp.Add(i);
        }
    }
    return temp;
}
```



**Figure 2.25: - New Collection**

Now the above approach is fine but it would be great if we would get rid of the collection, so that our code becomes simple. This where “yield” keyword comes to help. Below is a simple code how we have used yield.

“Yield” keyword will return back the control to the caller, the caller will do his work and re-enter the function from where he had left and continue iteration from that point onwards. In other words “yield” keyword moves control of the program to and fro between caller and the collection.

```

static IEnumerable<int> FilterWithYield()
{
    foreach (int i in MyList)
    {
        if (i > 3) yield return i;
    }
}
  
```

So for the above code following are details steps how the control will flow between caller and collection. You can also see the pictorial representation in the next diagram shown below.

- Step 1:- Caller calls the function to iterate for number’s greater than 3.
- Step 2:- Inside the function the for loop runs from 1 to 2 , from 2 to 3 until it encounters value greater than “3” i.e. “4”. As soon as the condition of value greater than 3 is met the “yield” keyword sends this data back to the caller.

- Step 3:- Caller displays the value on the console and re-enters the function for more data. This time when it reenters, it does not start from first. It remembers the state and starts from “5”. The iteration continues further as usual.

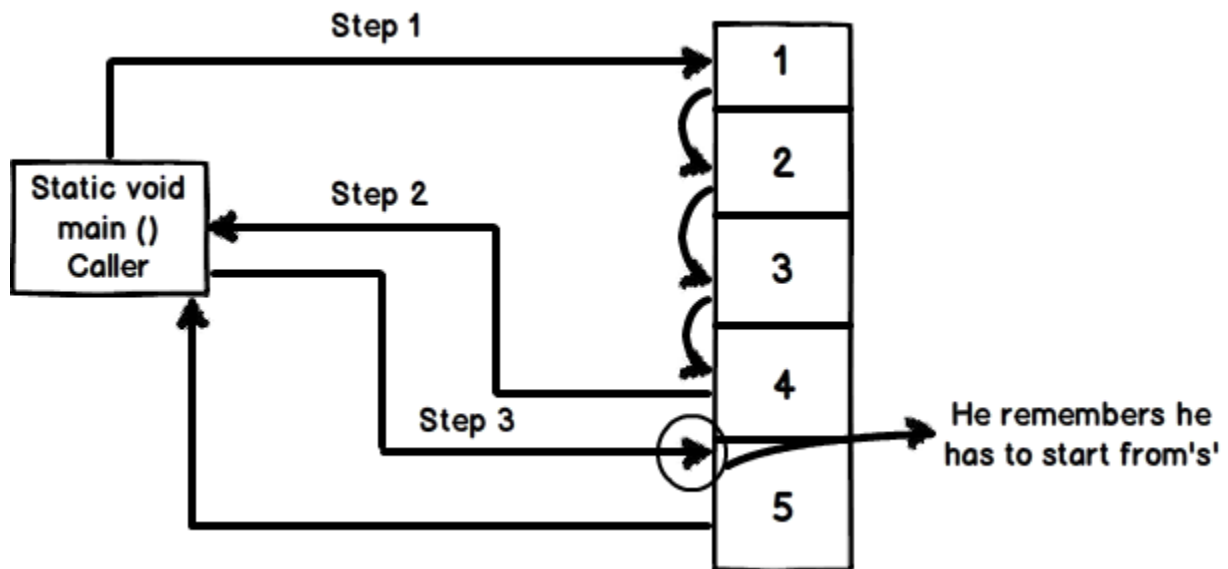


Figure 2.26: - Static Void Main Caller

### Scenario 2:- Stateful iteration

Now let us add more complications to the above scenario. Let's say we want to display running total of the above collection. What do I mean?.

In other words we will browse from 1 to 5 and as we browse we would keep adding the total in variable. So we start with “1” the running total is “1”, we move to value “2” the running total is previous value “1” plus current value “2” i.e. “3” and so on.

Below is the pictorial representation of the running total looks like.

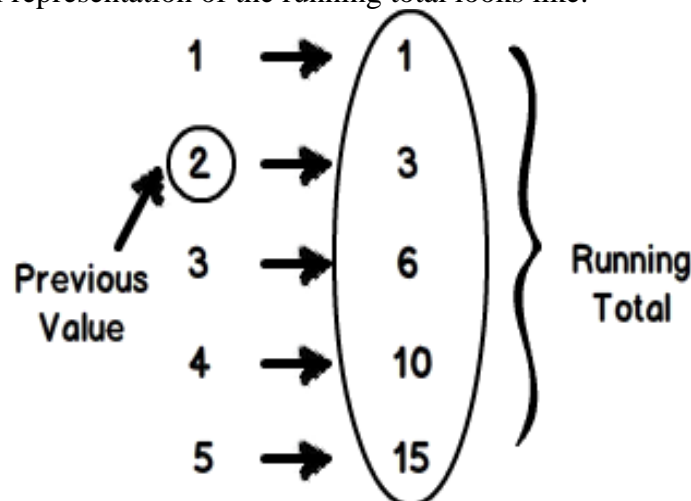


Figure 2.27: - Stateful Iteration

In other words we would like to iterate through the collection and as we iterate would like to maintain running total state and return the value to the caller ( i.e. console application). So the function now becomes something as shown below. The “runningtotal” variable will have the old value every time the caller re-enters the function.

```
static IEnumerable<int> RunningTotal()
{
    int runningtotal=0;
    int index=0;
    foreach(int i in MyList)
    {
        index = index + 1;
        if(index==1)
        {
            runningtotal = i;
        }
        else
        {
            runningtotal = i + runningtotal;
        }
        yield return (runningtotal);
    }
}
```

Below goes the caller code and output.

```
foreach (int i in RunningTotal())
{
    Console.WriteLine(i);
}
Console.ReadLine();
```

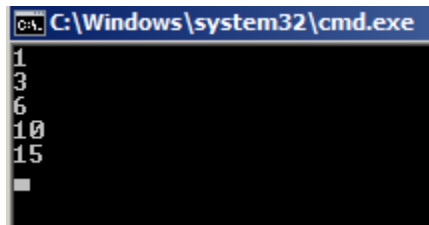


Figure 2.28: - Output

## What is the difference between “==” and .Equals()?

When we create any object there are two parts to the object one is the content and the other is reference to that content.

So for example if you create an object as shown in below code:-

1. “.NET interview questions” is the content.
2. “o” is the reference to that content.

```
object o = ".NET Interview questions";
```

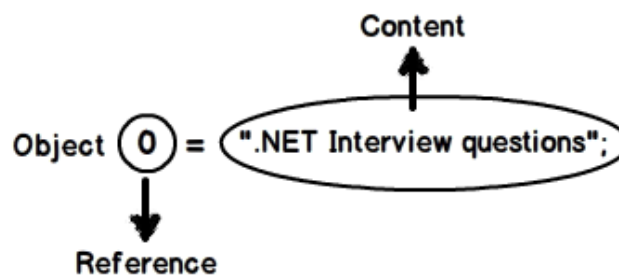


Figure 2.29: - Object

“==” compares if the object references are same while “.Equals()” compares if the contents are same.

So if you run the below code both “==” and “.Equals()” returns true because content as well as references are same.

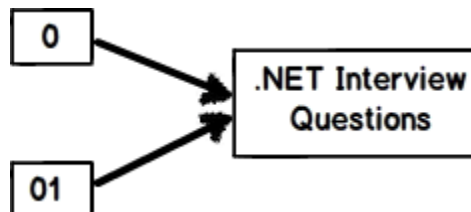


Figure 2.30: - .Equals()

```
object o = ".NET Interview questions";
```

```
object o1 = o;
```

```

Console.WriteLine(o == o1);
Console.WriteLine(o.Equals(o1));
Console.ReadLine();

```

True  
True

Now consider the below code where we have same content but they point towards different instances. So if you run the below code both “==” will return false and “.Equals()” will return true.

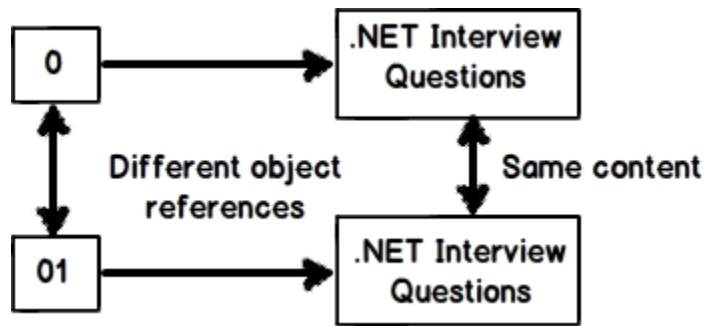


Figure 2.31: - Object References

```

object o = ".NET Interview questions";
object o1 = new string(".NET Interview questions".ToCharArray());

Console.WriteLine(o == o1);
Console.WriteLine(o.Equals(o1));
Console.ReadLine();

```

False  
True

When you are using string data type it always does content comparison. In other words you either use “.Equals()” or “==” it always do content comparison.



## What's the difference between catch with parameter and catch without parameter?

First let's try to understand this question: -

```
catch(Exception e)
{
    ...
}
```

VS

```
Catch
{
    ...
}
```

For this interview question many people answer, “Second one will cause compile error”. But both the codes will work properly. Actually from .NET 2.0 there is no difference. But in the initial versions of .NET i.e. prior 2.0 some of the exceptions thrown by some COM components did not translate to “Exception” compatible object.

From .NET 2.0 both code will execute properly and there is no difference between them internally. Both catches will handle all kind of exceptions.

After 2.0 a catch which does have any code written in it gives a warning as shown below. So many developers mark this as a difference.

```
}
catch (Exception ex)
{
    The variable 'ex' is declared but never used
}
```

**Figure 2.32: - Exception**

But you can always overcome this issue by not putting a variable as shown in the below code.

```
catch (Exception)
{
```

```
}
```

## What are attributes and why do we need it?

*“Attribute is nothing but a piece of information”.*

This information can be attached to your method, class, namespace, assembly etc. Attributes are part of your code this makes developers life easier as he can see the information right upfront in the code while he is calling the method or accessing the class and take actions accordingly.

For instance below is a simple class where “Method1” is decorated by the “Obsolete” attribute. Attributes are defined by using the “[” symbol. So when developers starting coding in this class they are alerted that “Method1” is obsolete and code should be now written in “NewMethod1”.

```
public class Class1
{
    [Obsolete]
    public void Method1()
    {
    }
    public void NewMethod1()
    {
    }
}
```

In the same way if somebody is trying to create object of “Class1” he gets an alert in the tool tip as shown in the below code snippet that “Method1” is obsolete and he should use “NewMethod1”.

```
static void Main(string[] args)
{
    Class1 o = new Class1();

    o.Method1();
}
```

[deprecated] void Class1.Method1()

Warning:  
'ConsoleApplication48.Class1.Method1()' is obsolete

**Figure 2.33: -New Method1**

So in short Attributes are nothing small piece of information which is embedded declaratively in the code itself which developers can see upfront.

In case you want to show some message to the developers you can pass the message in the “Obsolete” attribute as shown in the below code snippet.

```
[Obsolete("Please use NewMethod1")]  
public void Method1()  
{  
  
}
```

---

If you want to be bit strict and do not developers to use that method, you can pass ‘true’ to the “Obsolete” attribute as shown in the below code.

```
[Obsolete("Please use NewMethod1",true)]  
public void Method1()  
{  
  
}
```

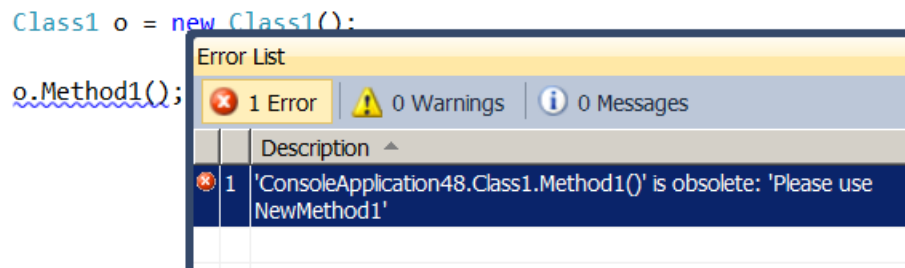
---

If you want to be bit strict and do not developers to use that method, you can pass ‘true’ to the “Obsolete” attribute as shown in the below code.

```
[Obsolete("Please use NewMethod1",true)]  
public void Method1()  
{  
  
}
```

---

Now in case developers try to make a call to “Method1” they will get error and not just a simple warning.



**Figure 2.34: - Error List**

## How can we create custom Attributes?

The “Obsolete” attribute which we discussed at the top is a readymade attribute. To create a custom attribute, you need to inherit from the attribute class. Below is a simple “HelpAttribute” which has a “HelpText” property.

```
class HelpAttribute : Attribute
{
    public string HelpText { get; set; }
}
```

“HelpAttribute” is applied to the “Customer” as shown in the code below. Now developers who see this class, see the information right in the front of their eyes.

```
[Help(HelpText="This is a class")]
class Customer
{
    private string _CustomerCode;

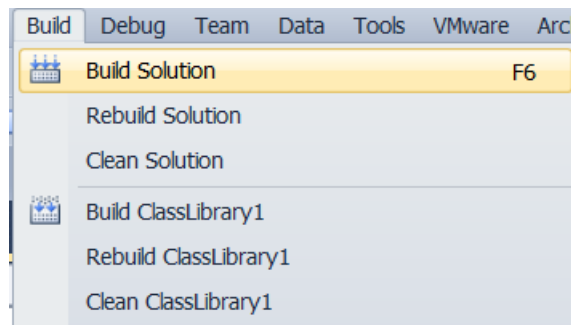
    [Help(HelpText = "This is a property")]
    public string CustomerCode
    {
        get { return _CustomerCode; }
        set { _CustomerCode = value; }
    }
}
```

```
[Help(HelpText = "This is a method")]  
public void Add()  
{  
}  
}
```

## How can we mark a method as deprecated?

Refer the previous answer.

## What is the difference between Build Vs Rebuild Vs Clean solution menu ?



**Figure 2.35: - Build VS Rebuild**

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

**Rebuild solution menu:** - This will delete all current compiled files (i.e. exe and dll's) and will build everything from scratch, irrespective of whether there is a code change in the file or not.

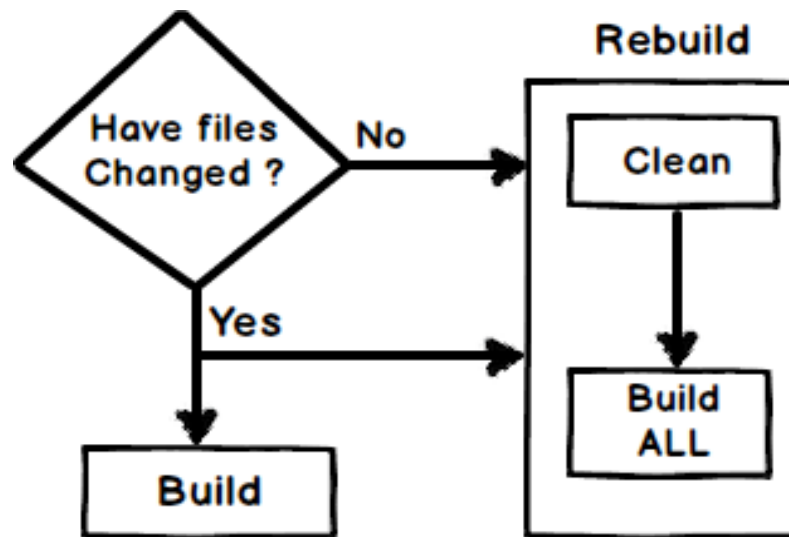


Figure 2.36: - Rebuild Solution

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

Now if you read the above 3 points I have discussed you can conclude that:-

**Rebuild = Clean + Build**

So the next question would be If you do a "Rebuild" and if you do "Clean" + "Build", what is the difference ?.

The difference is the way the build and clean sequence happens for every project. Let's say if your solution has two projects "proj1" and "proj2". If you do a rebuild it will take "proj1", clean ( delete) the compiled files for "proj1" and build it. After that it will take the second project "proj2", clean compiled files for "proj2" and compile "proj2".

But if you do a "clean" and build". It will first delete all compiled files for "proj1" and "proj2" and then it will build "proj1" first followed by "proj2".

Below image explains the same in a more visual format.

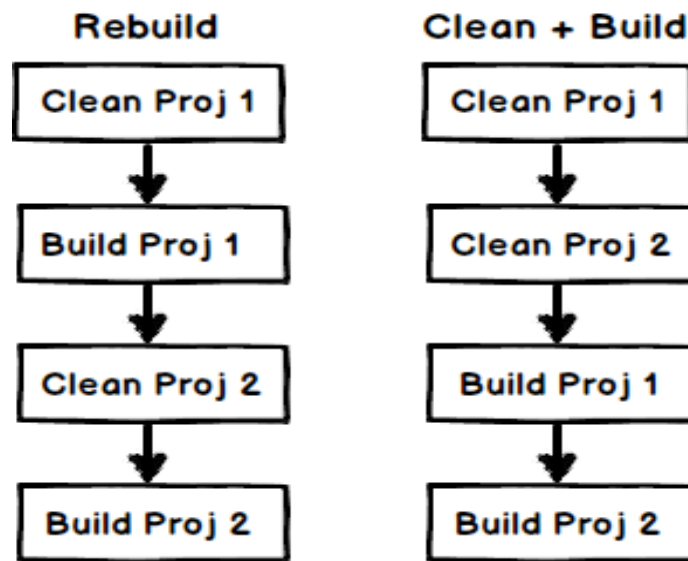


Figure 2.37: - Clean+Rebuild

**What is the difference between i++ vs ++i ?**

i++ :- In this scenario first the value is assigned and then increment happens. In the below code snippet first the value of “i” is assigned to “j” and then “i” is incremented.

```

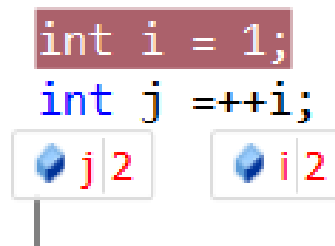
int i = 1;
int j = i++;

```

j	1	i	2
---	---	---	---

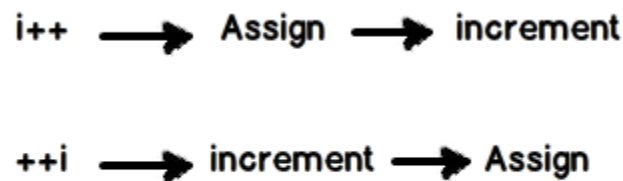
Figure 2.38: - i++ VS ++i

++i :- In this scenario first the increment is done and then value is assigned. In the below code snippet value of “j” is 2 and value of “i” is also 2.



**Figure 2.39: - ++i**

If you visualize in a pictorial manner below is how it looks like. So `i++` is a postfix, it first assigns and then increments. While `++i` is a prefix, it first increments and then assigns the value.



**Figure 2.40: - Assign and Increment**

Below are some sample code where postfix and prefix fits in.

```

while (i < 5) //evaluates conditional statement
{
    //some logic
    ++i;      //increments i
}
  
```

```

while (i++ < 5) //evaluates conditional statement with i value before increment
{
    //some logic
}
  
```

```

int i = 0;
int[] MyArray = new int[2];
MyArray[i++] = 1234; //sets array at index 0 to '1234' and i is incremented
  
```



```
MyArray[i] = 5678; //sets array at index 1 to '5678'
```

```
int temp = MyArray[--i]; //temp is 1234
```

---

## When should we use “??”(NULL Coalescing operator)?

“??” is a NULL coalescing operator. If you see the English meaning of coalescing it says “consolidate together”. Coalescing operator returns the first NON-NULL value from a chain. For example below is a simple coalescing code which chains four strings.

So if “str1” is null it will try “str2” , if “str2” is null it will try “str3” and so on until it finds a string with a non-null value.

```
string final =str1 ?? str2 ?? str3 ?? str4;
```

## Explain need of NULLABLE types ?

It is difficult to assign NULL directly for value types like int , bool , double etc. By using NULLABLE types you can set value types as NULL. To create a NULLABLE type we need to put “?” before the data type as shown in the below code.

```
int? num1 = null;
```

## In what scenario’s we will use NULLABLE types ?

The biggest user of NULLABLE types is when you are reading values from database. Database is one place where there is high possibility of column having NULL’s. So when we want to read those values in to value types NULLABLE types makes it easy as shown in the below code.

```
while (oreader.Read())  
{  
    int? salary = oreader["Salary"] as int? ;  
}
```

## What is the benefit of coalescing?

You do not need to write long if condition as shown below.

```
if (str1 != null)
{
    final = str1;
}
else
{
    if (str2 != null)
    {
        final = str2;
    }
    else
    {
        if (str3 != null)
        {
            final = str3;
        }
        else
        {
            if (str4 != null)
            {
                final = str4;
            }
        }
    }
}
```

## Chapter 3: OOP

Note :- We have come out with a exclusive book on ‘OOP’s with real project scenarios’. In this book we have tried to understand OOP’s fundamentals and also tried to apply them on projects by simulating project scenarios.

## **What is Object Oriented Programming?**

OOP is software designing technique where we think in terms of real world objects.

## **What is a Class and object?**

Class is a blue print / template. Objects are instances of classes, in other words they bring life in class. To use a class we need to create an object.

## **What are different properties provided by Object-oriented systems?**

Following are characteristics of Object Oriented System's:-

### **Abstraction**

Abstraction means show only what is necessary. Example color is abstracted to RGB. By just making the combination of these three colors we can achieve any color in world. So rather than remembering each and every color we just use RGB.

### **Encapsulation**

It is a process of hiding all the complex processing from the outside world and make's your objects simple.

### **Inheritance**

This concept helps to define parent child relationship between classes.

### **Polymorphism**

It's a property of object to act differently under different conditions. For instance a simple user object depending on conditions can act like a admin or like data entry object.

**Remember the word APIE (Abstraction, Polymorphism, Inheritance and Encapsulation).**

## **How can we implement encapsulation in .NET?**

Encapsulation can be achieved by using the below 5 access modifiers.

- Private: Only members of class have access to the variables.
- Protected: -All members in current class and in derived classes can access the variables.
- Friend (internal in C#):-Only members in current project have access to the elements.



- Protected friend (protected internal in C#):- All members in current project and all members in derived class can access the variables.
- Public: -All members have access in all classes and projects.

Note :- This question can also be tweaked by asking what are the different access modifiers in .NET .

## What's the difference between abstraction and encapsulation?

Note :- This question is a bit confusing question. Abstraction says show only what is necessary and encapsulation says hide complexity. Does it look like talking the same thing's with different faces ?.

Abstraction and encapsulation complement each other. Encapsulation implements abstraction. Abstraction is design process while encapsulation happens during coding phase which is achieved by using access modifiers. Abstraction is done in design phase while encapsulation is implemented in execution phase using access modifiers.

## How is inheritance implemented in .NET?

Inheritance is implemented by using the “:” symbol.

Below is a simple code snippet where we have “Customer” class which is the parent class. We have then created a child class called as “CustomerDiscount” which inherits all the properties and adds a “Discount” property.

```
class Customer
{
    public string customerName;
    public string customerCode;
}

class CustomerDiscount : Customer
{
    public double Discount;
}
```



## What are the two different types of polymorphism?

There are 2 kinds of polymorphism static and dynamic. Many people also call them as runtime or compile time polymorphism.

## How can we implement static polymorphism?

Static polymorphism is implemented by using method overloading. In compile time itself we come to know if there are mismatches. Below code snippet shows how method overloading is implemented. The add method can take either 2 inputs or 3 inputs.

Depending on the number of inputs the addition logic is executed.

```
// add method with 2 inputs.  
objmaths.add(1,2);  
  
// add method with 3 inputs.  
objmaths.add(1,2,3);
```

## How can we implement dynamic polymorphism?

Dynamic polymorphism is implemented by using overriding and virtual keyword.

Below is a simple code snippet which has three classes, Customer class is the parent class. CustomerDiscount10Percent and CustomerDiscount20Percent are child classes.

Customer parent class has a discount function which returns zero discounts. This function is defined as virtual and then overridden by both the child classes with 10 and 20% discount.

```
class Customer  
{  
    public string customerName;  
    public string customerCode;  
    public virtual int Discount()  
    {  
        return 0;  
    }  
}
```

```
class CustomerDiscount10Percent : Customer
{
public override int Discount()
{
return 10;
}

}
```

```
class CustomerDiscount20Percent : Customer
{
public override int Discount()
{
return 20;
}

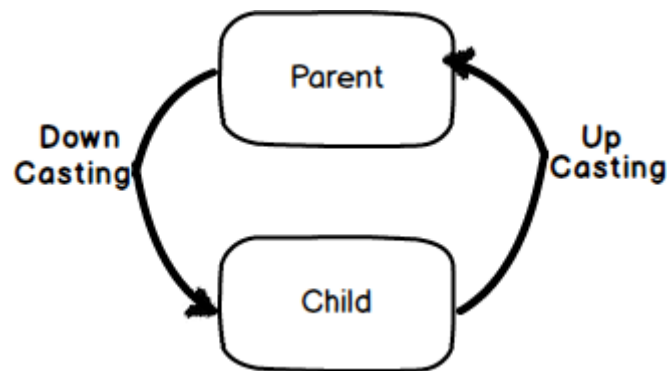
}
```

Now on the client side on the fly your parent object can point to any child classes and invoke the child implementation accordingly. This is called as dynamic polymorphism; the parent object can point to any of the child objects and invoke the child function dynamically.

```
Customer obj;
obj = new CustomerDiscount10Percent();
obj = new CustomerDiscount20Percent();
```

## What is downcasting and upcasting ?

“Upcasting” means moving subclass object to the parent class object. “DownCasting” is opposite to “Upcasting” moving the parent object to the child object.



**Figure 3.1: - Downcasting and Upcasting**

“Upcasting” is perfectly valid but “Downcasting” is not allowed in .NET. For instance below is a simple “Customer” parent class which is further inherited by a child class “GoldCustomer”.

```
class Customer
{
}
```

```
class GoldCustomer : Customer
{
}
```

Below is an “upcasting” code where the child parent class gold customer is pushed to the customer class.

```
Customer obj = new GoldCustomer();
```

Below is a sample of “downcasting” code where parent class object is tried to move to a child class object, this is not allowed in .NET.

```
GoldCustomer obj = new Customer(); // not allowed illegal
```

## What is the difference overriding and overloading?

Note: - I am not sure why this question is asked frequently. It’s like comparing apples with mangoes. Probably it’s just the common word “over” which confuses developers.



Overloading is a concept where we can have same method names with different input signature.

In overriding we have a parent class with virtual functions which are overridden in the child classes.

## What is operator overloading?

Operator overloading is a concept of polymorphism where you can redefine operators like +, -, \* etc with additional functionalities.

For instance we can redefine the + functionalities to add objects like obj1 + obj2. Below is simple code snippet which redefines + operator.

```
class SomeClass
{

private int someValue;

public SomeClass(int val)
{
    someValue = val;
}

public static SomeClass operator +(SomeClass arg1, SomeClass arg2)
{
    return new SomeClass(arg1.someValue + arg2.someValue);
}

}
```

You can now use the + operator to add objects of type someclass as shown in the below code snippet.

```
Obj = obj1 + obj2;
```



## What are abstract classes?

Abstract class is a half defined parent class. The full implementation of abstract class is defined by the child classes.

For example below code snippet shows a simple abstract class / half defined class called “DatabaseCommon” and later the concrete classes i.e. “SQLServer” and “Oracle” inherit and define a complete implementation for the same.

To define an abstract class we need to use the abstract keyword.

```
public abstract class DatabaseCommon
{
}

public class SQLServer : DatabaseCommon
{
}

public class Oracle : DatabaseCommon
{
}
```

---

## What are abstract methods?

Abstract classes can have abstract methods. Abstract methods when defined in a parent class have to be implemented in the child classes. If abstract methods are not implemented it will throw a error.

## What is an Interface?

Interface is a contract that defines the signature of the functionality. It looks like a class but has no implementation. It has only empty definition of methods, functions, events, and indexer. Interfaces provide forced implementation. For instance in the below code snippet we have created a simple interface called as “IDbCompulsory”. The below classes who implement interface “IDbCompulsory” has to provide implementation for “ExecSql”.

```
interface IDbCompulsory
{
    void ExecSql();
}

public class SQLServer : IDbCompulsory
{
}
```

```
public void ExecSql()
{
    // Here code for firing SQL Server SQL statements
    // are written
}
}

public class Oracle : IDbCompulsory
{
    public void ExecSql()
    {
        // Here code for firing Oracle SQL statements
        // are written
    }
}
```

### Do interface have accessibility modifier ?.

All elements in Interface should be public. So no accessibility modifier is required.

### Can we create an object of abstract class or an interface?

No.

### What is difference between abstract classes and interfaces?

	<b>Abstract class</b>	<b>Interface</b>
Implementation	Some methods in abstract classes can have implementation.	All methods, function, properties in interfaces have to empty compulsorily.
Scenario	Abstract classes are used when we want to share common functionality in parent child relationship.	Interfaces are used to define contract, enforce standardization, decoupling and dynamic polymorphism.
Variable declaration	We can declare variables	In interface we cannot.
Inheritance VS	Abstract classes are inherited.	Interfaces are implemented.

## Implementation

### **An abstract with only abstract method, how is it different from interfaces?**

If you define all methods, function, properties as abstract in abstract class it inhibits the same behavior as an interface.

### **If we want to update interface with new methods, what is the best practice?**

The biggest use of interface is to ensure that strict CONTRACT is followed between clients and components. So that when changes happen on the components clients do not have to change too much. In real world CONTRACTS between two parties do not change which implies that interfaces should also not change.

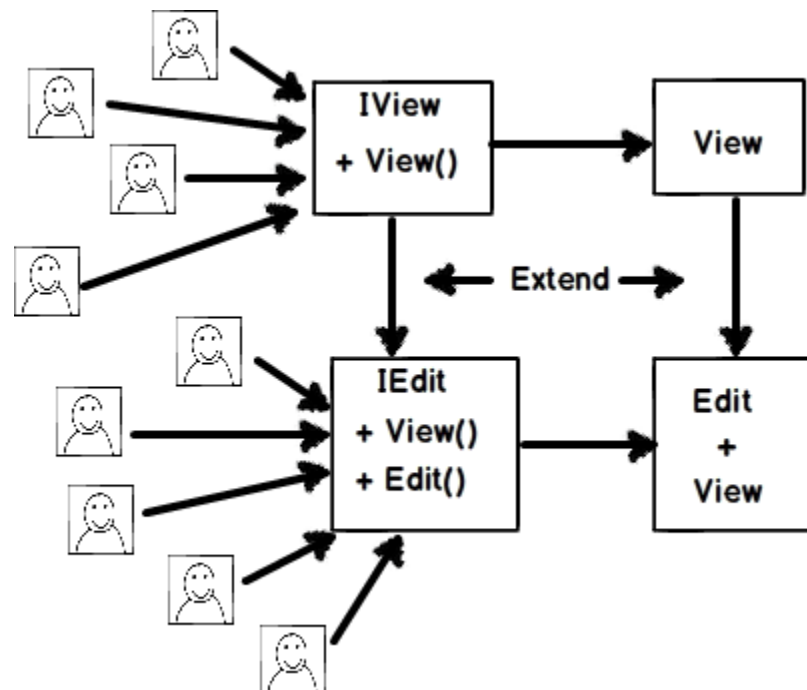
So if you want to add new methods or change methods in interfaces the best practice is to create new interfaces by inheriting. With this approach your older client who are using the interface stay happy and the new clients who want those new or changed methods get the benefits of the new functionality.

Let's consider you have a simple interface called as "IView" which helps you to view data , below is the code for the same. Let's consider that this interface is consumed by many clients.

```
interface Iview
{
    public void View();
}
```

---

Over a period of time some users demand "Edit" functionality as well but the rest of the users are happy with the 'View' functionality and they do not want them to get affected by these changes. Now if you go and change this interface ( "IView") a.k.a you will be disturbing everyone.



**Figure 3.2: - IView and IEdit**

So the best option would be to add a new interface i.e. “IEdit” which inherits from “IView”. So the “IEdit” will have the functionality for “Edit” as well as “View” because it’s also inheriting from “IView”. And the clients who are consuming “IView” do not need to update as “IView” is not changed at all.

```

interface IEdit : Iview
{
    public void Edit();
}
  
```

So putting this whole story in one single sentence for the interviewer.

Interface once fixed should not be changed. If we ever have to add new functions, new interfaces should be created so that we do not break compatibility with old clients.

## What is a delegate?

Delegate is an abstract pointer to a function or method. In other words you can create a pointer which points to a method or function and then pass that pointer wherever you wish and invoke the function / method.

## How can we create a delegate?

Creating a delegate is four step process:-

- Declare a delegate.
- Create an object reference.
- Point the reference to the method.
- Invoke the method via the delegate.

Below is the code snippet for the same.

```
// Declare a delegate
public delegate int PointToAdd(int i, int y);

// Create a reference pointer
PointToAdd objpointer = null;

// Point to the method
objpointer = Add;

// Invoke the function/method
objpointer.Invoke(10,20);
```

## What is a multicast delegate?

Normally when you create a delegate, your delegate points to only one function or method. In case you want to point multiple functions and invoke them sequentially, you need to use the multicast delegate.

To point to multiple function using delegate pointer we need to use “+=” sign as shown in the below code snippet.

```
ptrcall += Method1;
ptrcall += Method2;
```

## What are Events?

Events are higher level of encapsulation over delegates. Events use delegates internally.

Delegates are naked and when passed to any other code, the client code can invoke the delegate.

Event provides a publisher / subscriber mechanism model.



So subscribers subscribe to the event and publisher then push messages to all the subscribers. Below is a simple code snippet for the same :-

Create a delegate and declare the event for the same.

```
public delegate void CallEveryone();  
public event CallEveryone MyEvent;
```

---

Raise the event.

```
MyEvent();
```

---

Attached client methods to the event are fired / notified.

```
obj.MyEvent += Function1;
```

---

## What is the difference between delegate and events?

They can not be compared because one derives from the other.

- Actually, events use delegates in bottom. But they add an extra layer of security on the delegates, thus forming the publisher and subscriber model.
- As delegates are function to pointers, they can move across any clients. So any of the clients can add or remove events, which can be confusing. But events give the extra protection / encapsulation by adding the layer and making it a publisher and subscriber model.

Just imagine one of your clients doing this

```
c.XyzCallback = null
```

---

This will reset all your delegates to nothing and you have to keep searching where the error is.

## Do events have return type?

No, events do not have return type.

## Can events have access modifiers?

Yes.

## Can we have shared events?

Yes, you can have shared events, do note only shared methods can raise shared events.

## Explain Action , Func , Anonymous methods and Lambda expressions?

Left to the readers.

## What is shadowing?

Shadowing replaces the complete element of the parent class. For instance you can see in the below sample code where the clsParent has a variable int “i”, which is replaced by the child class clsChild by a method “i”.

In other words when you refer the parent object “i” it is a variable and when you refer the child object “i” it is a method.

```
class clsParent
{
    public int i=0;
}
class clsChild : clsParent
{
    public new void i()
    {
        Console.WriteLine("Hey i became a method");
    }
}
```

## What is the difference between Shadowing and Overriding?

Overriding redefines only the implementation while shadowing redefines the whole element.

## If we inherit a class do the private variables also get inherited?

No, the variables are donot get inherited.

## How can we stop the class from further inheriting?

We can stop the class from further inheriting by using the "Sealed" keyword. For instance below is a simple sample code where we have a class called as "Human" which is further inherited to create a "Male" or "Female" class.

Now the below code is great but we do not allow anyone to further inherit from "Male" or "Female" class. In simple words "Male" and "Female" are the last legs in this inheritance hierarchy. This can be done by using the "Sealed" keyword.

```
public class Human
{
    public sealed class Male : Human
    {
    }
    public sealed class Female : Human
    {
    }
```

---

If anyone tries to inherit the sealed classes he will end with the below error "cannot derive from sealed type".

## What is the use of “Must inherit” keyword in VB.NET?

If you want to create an abstract class in VB.NET it's done by using “Must Inherit” keyword. You cannot create an object of a class, which is marked as “Must Inherit”. When you define “Must Inherit” keyword for class, you can only use the class by inheriting.

## What are similarities between Class and structure?

Following are the similarities between classes and structures:-

- Both can have constructors, methods, properties, fields, constants, enumerations, events, and event handlers.
- Structures and classes can implement interface.
- Both of them can have constructors with and without parameter.
- Both can have delegates and events.

## What is the difference between Class and structure's?

Following are the key differences between them:-

- Structures are value types and classes are reference types. So structures use stack and classes use heap.



- Structures members cannot be declared as protected, but class members can be. You cannot do inheritance in structures.
- Structures do not require constructors while classes require.
- Objects created from classes are terminated using Garbage collector. Structures are not destroyed using GC.

## When to use Structures and When to use classes ?

You will use structures when:-

Point 1:- If you want to represent a custom value type. This custom value type is derived from primitive data types (int, double). Some of the example's of custom types are co-ordinates (which have X, Y), complex numbers (which have real and imaginary components). You can also term these things as value objects or technical objects. Technical objects do not represent real world objects like customer, supplier, invoice etc.

Point 2:- If you want to have low memory foot print. For instance let's say you want to plot a graph. To plot a graph you need to have 100's of objects created so that you can represent co-ordinates. Now if you create a class and generate those 100's of objects you end up putting lot of load on your garbage collector. If you create a "struct", it's a value type. So they get created and destroyed immediately. Thus putting less load on memory.

For all other scenarios use a class.

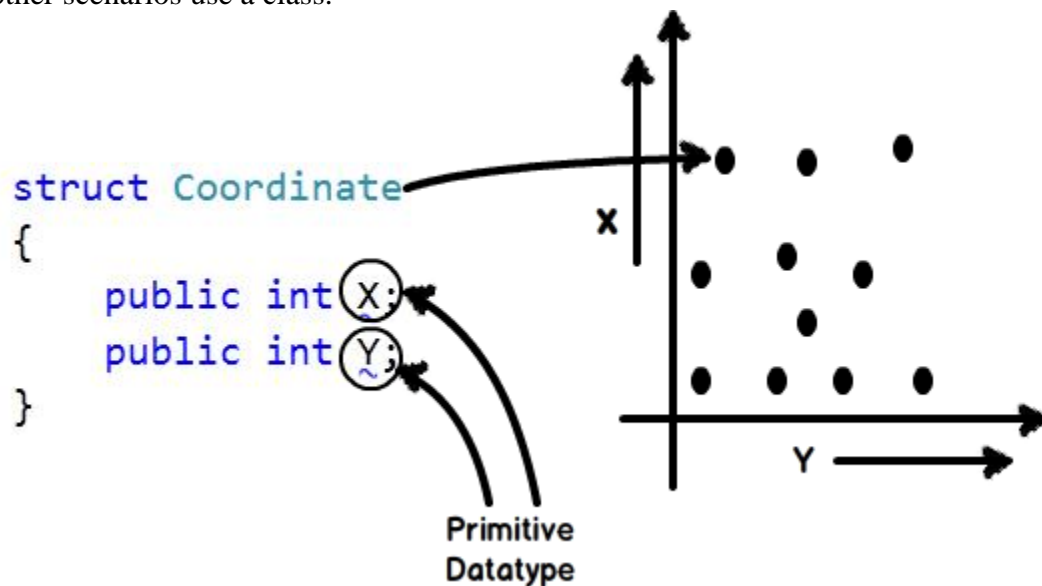


Figure 3.3: - Primitive Datatype

## What does virtual keyword mean?

They signify that method and property can be overridden by the child classes.

## What are shared (VB.NET)/Static(C#) variables?

When you define a variable as static or shared only one instance of the object or variable is created.

## What is ENUM and what are the benefits of using it?

ENUM helps to define, manage and assign constants in effective way. Now the below sample code is good but the level values are not readable.

```
if (level == 0) { Console.WriteLine("Below quality"); }  
else if (level == 1) { Console.WriteLine("Moderate quality"); }  
else if (level == 2) { Console.WriteLine("High quality"); }
```

---

Now by declaring a simple enum called as Quality as shown below.

```
enum Quality  
{  
    Low = 0,  
    Moderate = 1,  
    High = 2  
};
```

---

Our code would look more readable as shown below. The other big benefit is if we change the numeric values of quality we do not have to change throughout the project. So we can go a change the Low quality to 1 and no change in code is required.

```
if (level == Quality.Low) { Console.WriteLine("Below quality"); }  
else if (level == Quality.Moderate) { Console.WriteLine("Moderate quality"); }  
else if (level == Quality.High) { Console.WriteLine("High quality"); }
```

---

So summarizing ENUM has two big benefits:-

- Code becomes more readable.
- Easy to change constants without affecting throughout the project. Easy maintenance.

## What is the use of Flags in ENUM?

“Flags” is an ENUM attribute. If you want to set multiple values to an ENUM we need to use Flags.

```
[Flags]  
enum MyColors  
{
```

```
Green = 0,  
Red = 1,  
Blue = 2  
};
```

In the below code we are setting “MyColors” ENUM to “Blue” and “Green” value.

```
MyColors color = MyColors.Blue | MyColors.Green;  
if ((color & MyColors.Blue) == MyColors.Blue)  
{  
    Console.WriteLine(" Blue");  
}  
if ((color & MyColors.Green) == MyColors.Green)  
{  
    Console.WriteLine(" Green");  
}
```

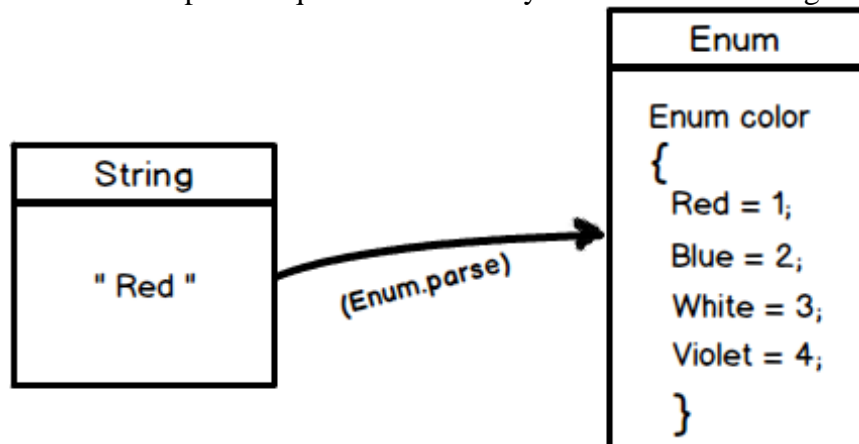
## How can we loop through ENUM values?

We can use “GetValues()” method of the “ENUM” class which returns an array of enum values.

```
var x = Enum.GetValues(typeof(MyColors));
```

## What is the use of “ENUM.Parse” method?

FYI :- Interviewers can also put this question differently how to convert string to a ENUM value.



**Figure 3.4: - ENUM.Parse**

Many times we would like to convert string in to ENUM objects. For example let's say you have a COLOR Enum and you would like to pass string "Red" and get ENUM object "color.Red". "ENUM.Parse" method parses string to an ENUM value. Like in the below code we pass "Red" string value and it gets type casted to enum which has green enum as a constant.

```
MyColors EnumColors = (MyColors)Enum.Parse(typeof(MyColors), "Red");
```

---

## What is nested Classes?

Nested classes are classes within classes.

### If you create the child class object which constructor will fire first?

```
public class class1
{
    public class1(){}
}

public class class2 : class1
{
    public class2(){}
}
```

---

Parent class constructor will fire first.

### In what instances you will declare a constructor to be private?

When we create a private constructor, we cannot create object of the class. Private constructors are used when we want only a single instance of the class to be created and externally no one can use the 'new' keyword to create the object.

### Can we have different access modifiers on get/set methods of a property?

Yes, we can have different access modifiers. The below code will compile perfectly well.

```
public string CustName
{
    get
```

---

```
{  
    return _Custname;  
}  
  
protected set  
{  
    Custname = value;  
}  
}
```

### **How can you define a property read only for external world and writable in the same assembly?**

This question is a variation to the previous question. So its possible that interviewer can ask the previous question in this format.

Let's us first try to understand this question. Let's say if you have a class called as "Customer" with a property "CustomerCode". Now you want that anyone can read from the "CustomerCode" property but this property can only be set from within the assembly.

In other words any one can run the below code.

```
Customer obj = new Customer();  
string x = obj.CustomerCode;
```

But setting of the value can be only done from within assembly. The below code will run only if it's within assembly and it will throw error if it's external to the assembly.

```
Customer obj = new Customer();  
obj.CustomerCode = "c001";
```

This can be achieved by have different access modifiers for "SET" and "GET" properties on the "CustomerCode" property.

So for the "GET" we will have public access modifiers and for "SET" we will apply internal access modifiers. Now because "GET" is public this property can be read anywhere and because the "SET" is internal it can only be accessed from within assembly. Below goes the code for the same.

```
class Customer  
{  
    private string _CustomerCode = "";
```

```
public string CustomerCode
{
    get { return _CustomerCode; }
    internal set { _CustomerCode = value; }
}
```

### Will the finally run in this code?

In the below code in the try statement we have a return statement, will the finally block still run.

```
public static void Main()
{
    try
    {
        // Some code
        return;
    }
    finally
    {
        //Will this run
    }
}
```

Yes, the finally code will run even though there is a return statement in the try. Finally block will run irrespective what happens in try block. That's the reason why finally block is a good place to put clean up code.

### What is an Indexer?

Many time classes have contained (aggregated or composed) collections. For example in the below code you can see we have a customer class which has an address collection.

```
public class Customer
{
    private List<Address> Addresses = new List<Address>();
}
```

Now let's say we would like to like fetch the addresses collection by "Pincode" and "PhoneNumber". So the logical step would be that you would go and create two overloaded functions one which fetches by using "PhoneNumber" and the other by "PinCode". You can see in the below code we have two functions defined.

```
public Address getAddress(int PinCode)
{
    foreach (Address o in Addresses)
    {
        if (o.Pincode == PinCode)
        {
            return o;
        }
    }
    return null;
}

public Address getAddress(string PhoneNumber)
{
    foreach (Address o in Addresses)
    {
        if (o.MobileNumber == PhoneNumber)
        {
            return o;
        }
    }
    return null;
}
```

Now on the client side your code would look something as shown below.

```
Customer Customers = new Customer();
Customers.getAddress(1001);
Customers.getAddress("9090");
```

Now the above code is great but how about simplifying things further. How about something more simple as shown in the below code. In other words we want to INDEXED the class itself.

```
Customer Customers = new Customer();  
Address o = Customers[10001];  
o = Customers["4320948"];
```

You can see the below image where we can have two overloaded indexers “PinCode” and “PhoneNumber”.

```
Address o = Customers[10001];  
o = Customer ▲ 1 of 2 ▼ Address Customer[int PinCode]
```

**Figure 3.5: - Address Customers**

This can be achieved by using an indexer. There are two things we need to remember about indexer:-

- “Indexer” is defined by using “this” keyword.
- “Indexer” is a property so we need to define set and get for the same.

Below is the code implementation for indexer. You can see we have used “this” keyword with two overloaded function one which will fetch us address by using the “phonenummer” and the other by using “pincode”.

```
public class Customer  
{  
  
private List<Address> Addresses = new List<Address>();  
public Address this[int PinCode]  
{  
    get  
{  
        foreach (Address o in Addresses)  
{  
            if (o.Pincode == PinCode)  
{  
                return o;  
            }  
        }  
    }  
}
```



```

    }
  }
  return null;
}
}

public Address this[string PhoneNumber]
{
  get
  {
    foreach (Address o in Addresses)
    {
      if (o.MobileNumber == PhoneNumber)
      {
        return o;
      }
    }
    return null;
  }
}
}

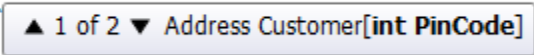
```

Once you put the above “indexer” code you should be able to access the address collection from the customer object using an indexer “[]” as shown in the below figure.

```

Address o = Customers[10001];
o = Customer

```



**Figure 3.6: - Address Customers**

Summarizing in one sentence: - Indexers helps to access contained collection with in a class using a simplified interface. It’s a syntactic sugar.

**SVG**

**Canvas**

Here's it's like draw and remember. In other words any shape drawn by using SVG can be remembered and manipulated and browser can render it again.

SVG is good for creating graphics like CAD software's where once something is drawn the user wants to manipulate it.

This is slow as it needs to remember the co-ordinates for later manipulations.

We can have event handler associated with the drawing object.

Resolution independent.

Canvas is like draw and forget. Once something is drawn you cannot access that pixel and manipulate it.

Canvas is good for draw and forget scenarios like animation and games.

This is faster as there is no intention of remembering things later.

Here we cannot associate event handlers with drawing objects as we do not have reference of them.

Resolution dependent.

## ***How to draw rectangle using Canvas and SVG using HTML 5 ?***

HTML 5 code Rectangle code using SVG.

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
<rect width="300" height="100"
  style="fill:rgb(0,0,255);stroke-width:1;stroke:rgb(0,0,0)" />
</svg>
```

HTML 5 Rectangle code using canvas.

```
var c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.rect(20,20,150,100);
ctx.stroke();
```

```
<svg
  <circle
    xmlns="http://www.w3.org/2000/svg"
    cx="100" cy="50" r="40"
    stroke="black"
    stroke-width="2"
    fill="red" />
</svg>
```

```
var canvas = document.getElementById('myCanvas');
```

```
var context = canvas.getContext('2d');  
var centerX = canvas.width / 2;  
var centerY = canvas.height / 2;  
var radius = 70;  
  
context.beginPath();  
context.arc(centerX, centerY, radius, 0, 2 * Math.PI, false);  
context.fillStyle = 'green';  
context.fill();  
context.lineWidth = 5;  
context.strokeStyle = '#003300';  
context.stroke();
```

---

```
<!DOCTYPE html>  
<html>  
<body onload="DrawMe();">  
<svg width="500" height="100">  
<circle id="circle1" cx="20" cy="20" r="10"  
      style="stroke: none; fill: #ff0000;"/>  
</svg>  
</body>  
<script>  
  
var timerFunction = setInterval(DrawMe, 20);  
alert("ddd");  
  
function DrawMe()  
{  
var circle = document.getElementById("circle1");  
var x = circle.getAttribute("cx");
```

```
var newX = 2 + parseInt(x);
if(newX > 500)
{
    newX = 20;
}
circle.setAttribute("cx", newX);
}
</script>
</html>
```

## Using JQuery

```
var timerFunction = setInterval(DrawMe, 20);

function DrawMe()
{

var circle = $("#circle1");
    alert("ddd");
var x = circle.attr("cx");
    alert("ddd1");
var newX = 2 + parseInt(x);
if (newX > 500) {
    newX = 20;
}
    circle.attr("cx",newX);

}
</script>
```

## Move a circle

```
<svgid="svg1"xmlns="http://www.w3.org/2000/svg">
<circleid="circle1"cx="20"cy="20"r="10"
style="stroke: none; fill: #ff0000;"/>
</svg>
```

```
<script>
```

```
$("#target").mousemove(function (event)
{
```

```
var circle = $("#circle1");

    circle.attr("cx", event.clientX);
    circle.attr("cy", event.clientY);

});

</script>
</body>
```

### SVG grouped with shapes

```
<svg x="100"
<g transform="rotate(45 50 50)">
<line x1="10" y1="10" x2="85" y2="10"
    style="stroke: #006600;"/>

<rect x="10" y="20" height="50" width="75"
    style="stroke: #006600; fill: #006600"/>

<text x="10" y="90" style="stroke: #660000; fill: #660000">
    Text grouped with shapes</text>

</g>
</svg>
```

### Rectangle with a rotate

```
<svg xmlns="http://www.w3.org/2000/svg"
    xmlns:xlink="http://www.w3.org/1999/xlink">

<rect x="50" y="50" height="110" width="110"
    style="stroke:#ff0000; fill: #ccccff"
    transform="translate(30) rotate(45 50 50)"
>
</rect>
<text x="70" y="100"
    transform="translate(30) rotate(45 50 50)"
>Hello World</text>
</svg>
```

### Transform and translate

```
<rect x="20" y="20" width="50" height="50"
      style="fill: #cc3333"/>

<rect x="20" y="20" width="50" height="50"
      style="fill: #3333cc"
      transform="translate(75,25)" />
```

## Rotate

```
<rect x="20" y="20" width="40" height="40"
      style="stroke: #3333cc; fill:none;"
      />

<rect x="20" y="20" width="40" height="40"
      style="fill: #3333cc"
      transform="rotate(15)"
      />
```

## Scale ups and downs a size

```
<rect x="10" y="10" width="20" height="30"
      style="stroke: #3333cc; fill:none;" />

<rect x="10" y="10" width="20" height="30"
```

```
style="stroke: #000000; fill:none;"

transform="scale(2)" />
```

## SVG path element

The <path> element is used to define a path.  
The following commands are available for path data:

- M = moveto
- L = lineto
- H = horizontal lineto
- V = vertical lineto
- C = curveto
- S = smooth curveto
- Q = quadratic Bézier curve
- T = smooth quadratic Bézier curveto
- A = elliptical Arc
- Z = closepath

Define a path that starts at position 150,0 with a line to position 75,200 then from there, a line to 225,200 and finally closing the path back to 150,0:

```
<svg xmlns="http://www.w3.org/2000/svg" version="1.1">
  <path d="M150 0 L75 200 L225 200 Z" />
</svg>
```

## What are selectors in CSS?

Selectors help to select an element to which you want to apply a style. For example below is a simple style called as ‘intro’ which applies red color to background of a HTML element.

```
<style>
.intro
{
background-color:red;
}
</style>
```

To apply the above “intro” style to div we can use the “class” selector as shown in the below figure.

```
<div class="intro">
```

```
<p>My name is Shivprasad koirala.</p>  
<p>I write interview questions.</p>  
</div>
```

---

### How can you apply CSS style using ID value?

So let's say you have a HTML paragraph tag with id "mytext" as shown in the below snippet.

```
<p id="mytext">This is HTML interview questions.</p>
```

---

You can create a style using "#" selector with the "id" name and apply the CSS value to the paragraph tag. So to apply style to "mytext" element we can use "#mytext" as shown in the below CSS code.

```
<style>  
#mytext  
{  
background-color:yellow;  
}  
</style>
```

---

### Quick revision of some important selectors.

Set all paragraph tags back ground color to yellow.

```
P,h1  
{  
background-color:yellow;  
}
```

---

Sets all paragraph tags inside div tag to yellow background.

```
div  
{  
background-color:yellow;  
}
```

---

P

Sets all paragraph tags following div tags to yellow background.



```
div+p
{
background-color:yellow;
}
```

---

Sets all attribute with “target” to yellow background.

```
a[target]
{
background-color:yellow;
}
```

```
<a href="http://www.questpond.com">ASP.NET interview questions</a>
```

```
<a href="http://www.questpond.com" target="_blank">c# interview questions</a>
```

```
<a href="http://www.questpond.org" target="_top">.NET interview questions with answers</a>
```

---

Set all elements to yellow background when control gets focus.

```
input:focus
{
background-color:yellow;
}
```

---

Set hyperlinks according to action on links.

```
a:link {color:green;}
a:visited {color:green;}
a:hover {color:red;}
a:active {color:yellow;}
```

---

## ***What is the use of column layout in CSS?***

CSS column layout helps you to divide your text in to columns. For example consider the below magazine news which is one big text but we need to divide the same in to 3 columns with a border in between. That’s where HTML 5 column layout comes to help.

Make no mistake about this — Shah Rukh Khan's hospitality is fantastic. When the superstar invites you home for a dinner party, you will be spoiled silly because he insists on personally taking care of his guests. If you land up unannounced and the gates of Mannat still open up for you, the food service may

not be that prompt, says SRK. According to him, the food order goes down to the kitchen based on the ground floor of his six-storey annex soon enough. How-ever, it takes a long time for the food trolley to come up to the floor on which he is entertaining. "Many a time the guests have already left," he laughs,

adding, "And then I hear the familiar sound of the trolley being wheeled in with all the food neatly laid out." Moral of the story: when you go to Mannat without prior notice, please be patient. The host doesn't wish to send you home hungry. It is just that the wooden trolley has a lot of ground to cover.

**Figure 9.14: - Layout in CSS**

To implement column layout we need to specify the following:-

- How many columns we want to divide the text in to?

To specify number of columns we need to use column-count. "webkit" and "moz-column" are needed for chrome and firefox respectively.

```
-moz-column-count:3; /* Firefox */
-webkit-column-count:3; /* Safari and Chrome */
column-count:3;
```

- How much gap we want to give between those columns ?

```
-moz-column-gap:40px; /* Firefox */
-webkit-column-gap:40px; /* Safari and Chrome */
column-gap:20px;
```

- Do you want to draw a line between those columns , if yes how much thick ?

```
-moz-column-rule:4px outset #ff00ff; /* Firefox */
-webkit-column-rule:4px outset #ff00ff; /* Safari and Chrome */
```

---

```
column-rule:6px outset #ff00ff;
```

---

Below is the complete code for the same.

```
<style>
.magazine
{
-moz-column-count:3; /* Firefox */
-webkit-column-count:3; /* Safari and Chrome */
column-count:3;

-moz-column-gap:40px; /* Firefox */
-webkit-column-gap:40px; /* Safari and Chrome */
column-gap:20px;

-moz-column-rule:4px outset #ff00ff; /* Firefox */
-webkit-column-rule:4px outset #ff00ff; /* Safari and Chrome */
column-rule:6px outset #ff00ff;
}
</style>
```

---

You can then apply the style to the text by using the class attribute.

```
<div class="magazine">
```

Your text goes here which you want to divide in to 3 columns.

```
</div>
```

---

## Can you explain CSS box model?

CSS box model is a rectangular space around a HTML element which defines border, padding and margin.

**Border:** - This defines the maximum area in which the element will be contained. We can make the border visible, invisible, define height and width etc.

**Padding:** - This defines the spacing between border and element.

**Margin:** - This defines the spacing between border and any neighboring elements.

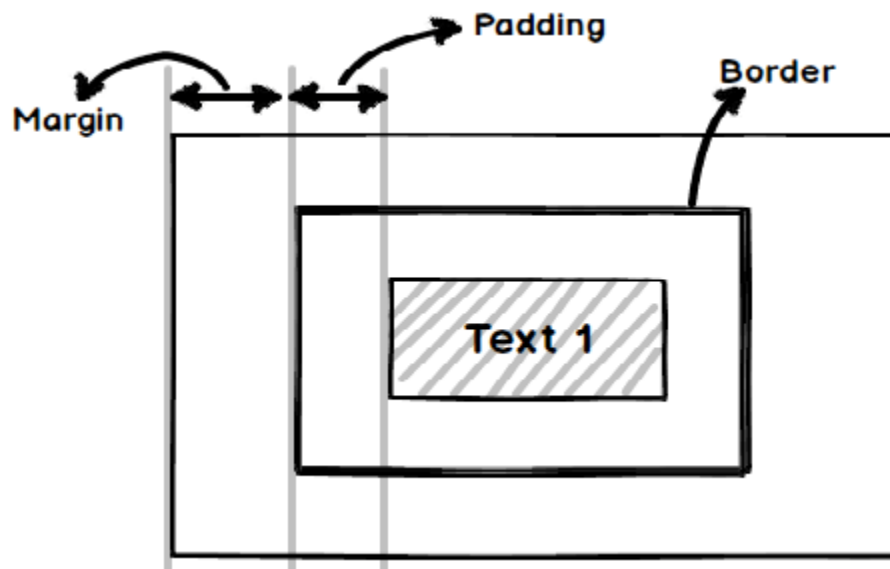


Figure 9.15: - CSS Box Model

For instance below is a simple CSS code which defines a box with border , padding and margin values.

```
.box {  
  width: 200px;  
  border: 10px solid #99c;  
  padding: 20px;  
  margin: 50px;  
}
```

}

Now if we apply the above CSS to a DIV tag as shown in the below code , your output would be as shown in the figure below. I have created two test “Some text” and “Some other text” so that we can see how margin property functions.

```
<div align="middle" class="box">
```

Some text

```
</div>
```

Some other text

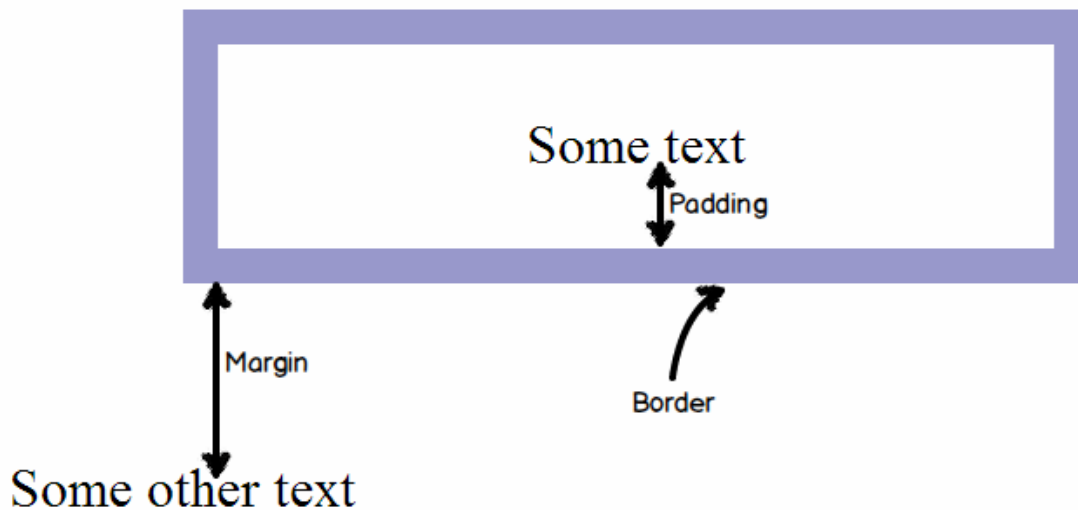


Figure 9.16: - Margin Property Function

### ***Can you explain some text effects in CSS 3?***

Here the interviewer is expecting you to answer one of two text effects by CSS. Below are two effects which are worth noting.

Shadow text effect

```
.specialtext
{
text-shadow:          5px          5px          5px          #FF0000;
}
```

Some text

Figure 9.17: - Shadow Text

Word wrap effect

```
<style>
.breakword
{word-wrap:break-word;}
</style>
```

This is a loooooooooooooooooong  
loooooooooooooooooong loooooooooooooooooong  
loooooooooooooooooong loooooong word..

Figure 9.18: - Word Wrap

### ***What are web workers and why do we need them ?***

Consider the below heavy for loop code which runs above million times.

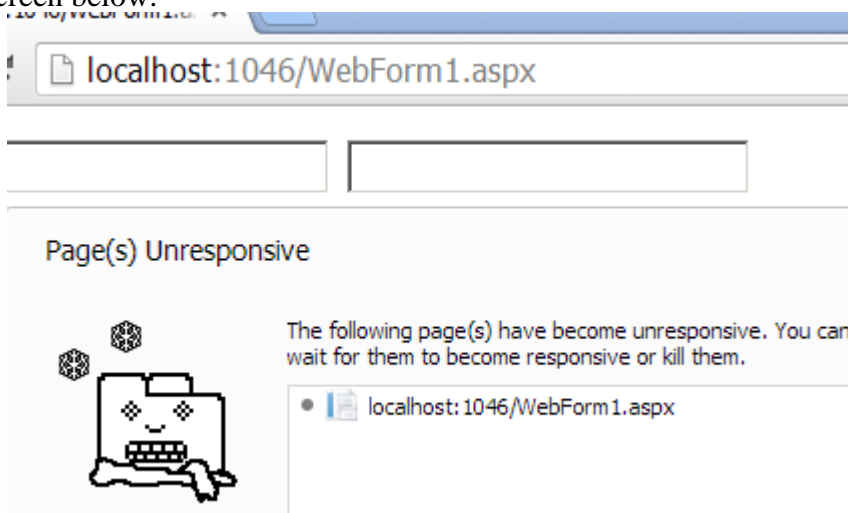
```
function SomeHeavyFunction()
{
for (i = 0; i < 100000000000000; i++)
{
x = i + x;
}
```

```
}  
}
```

Let's say the above for loop code is executed on a HTML button click. Now this method execution is synchronous. In other words the complete browser will wait until the for loop completes.

```
<input type="button" onclick="SomeHeavyFunction();" />
```

This can further lead to browser getting freezed and unresponsive with an error message as shown in the screen below.



**Figure 9.19: - Unresponsive Page**

So if we can move this heavy for loop in a JavaScript file and run it asynchronously that means the browser does not need to wait for the loop then we can have a more responsive browser. That's what web workers are for.

Web worker helps to execute JavaScript file asynchronously.

### ***What are the restrictions of Web Worker thread ?***

Web worker threads cannot modify HTML elements, global variables and some window properties like Window.Location. You are free to use javascript data types, XMLHttpRequest calls etc.

## So how do we create a worker thread in JavaScript?

To create a worker thread we need to pass the JavaScript file name and create the worker object.

```
var worker = new Worker("MyHeavyProcess.js");
```

To send message to the worker object we need to use “PostMessage”, below is the code for the same.

```
worker.postMessage("test");
```

When the worker thread sends data we get it in the “OnMessage” event on the callers end.

```
worker.onmessage = function (e)
{
    document.getElementById("txt1").value = e.data;
};
```



Figure 9.20: - On Message Event

The heavy loop is in the “MyHeavyProcess.js” javascript file , below is the code for the same. When the JavaScript file wants to send message he uses “postmessage” and any message sent from the caller is received in the “onmessage” event.

```
var x =0
self.onmessage = function (e) {
    for (i = 0; i < 1000000000; i++)
    {

        x = i + x;

    }
    self.postMessage(x);
```



```
};
```

## How to terminate a web worker

```
w.terminate();
```

## Why do we need HTML 5 server-sent events?

One of the common requirements in web world is getting updates from the server. Take example of a stock ticker application where the browser has to take regular updates from the server for the recent stock value.

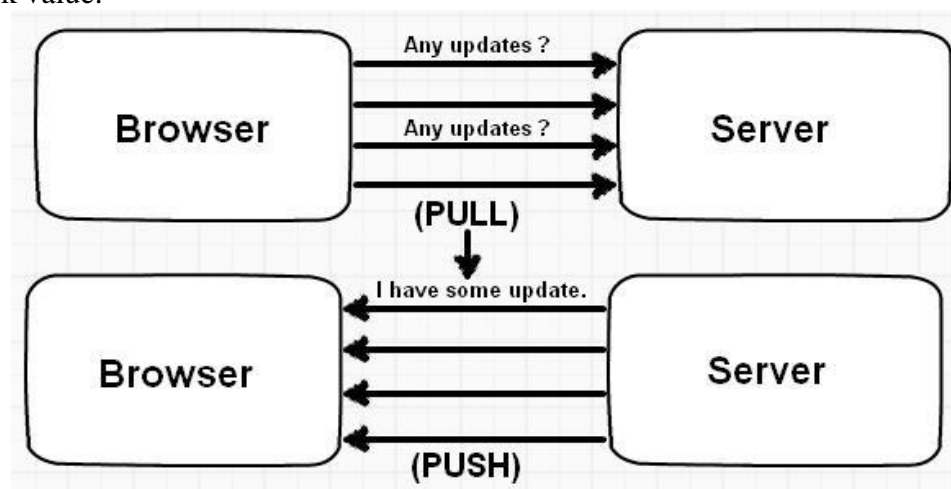


Figure 9.21: - Server Sent Events

Now to implement this kind of requirement developers normally write some kind of PULL code which goes to the server and fetches data in certain interval. Now PULL solution is good but it makes the network chatty with lot of calls and also it adds load on the server.

So rather than PULL it would be great if we can have some kind of PUSH solution. In simple words when the server has updates it will send updates to the browser client. That can be achieved by using “SERVER SENT EVENTS”.

So the first thing the browser needs to do is connect to the server source which will send updates. Let's say we have page “stock.aspx” which sends stock updates. So to connect to the page we need to use attach to the event source object as shown in the below code.

```
var source = new EventSource("stock.aspx");
```

We also need to attach the function where we will receive messages when server sends update. For than we need to attach function to the “onmessage” event as shown in the below code.

```
source.onmessage = function (event) {  
    document.getElementById("result").innerHTML += event.data + "<br>";  
};
```

Now from the server side we need to send events. Below are some lists of important events with command that needs to be sent from the server side.

Event	Command
Send data to the client.	data : hello
Tell client to retry in 10 seconds	retry : 10000
Raise a specific event with data	event : success data : You are logged in.

So for example if we want to send data below is the ASP.NET code for the same. Please note the content type is set to text/event.

```
Response.ContentType="text/event-stream";  
Response.Expires=-1;  
Response.Write("data:" + DateTime.Now.ToString());  
Response.Flush();
```

To retry after 10 second below is the command.

```
Response.Write("retry:10000");
```

If you want to attach an event we need to use the “addEventListener” event as shown in the below code.

```
source.addEventListener('message', function(e) {  
    console.log(e.data);  
}, false);
```

From the server side the below message will trigger the “message” function of javascript.

```
event: message
```

```
data : hello
```

---

## ***What is local storage concept in HTML 5?***

Many times we would like to store information about the user locally in the computer. For example let's say user has half-filled a long form and suddenly the internet connection breaks off. So the user would like you to store this information locally and when the internet comes back. He would like to get that information and send it to the server for storage.

Modern browsers have storage called as "Local storage" in which you can store this information.

## ***How can we add and remove data from local storage?***

Data is added to local storage using "key" and "value". Below sample code shows country data "India" added with key value "Key001".

```
localStorage.setItem("Key001","India");
```

---

To retrieve data from local storage we need to use "getItem" providing the key name.

```
var country = localStorage.getItem("Key001");
```

---

You can also store JavaScript object's in the local storage using the below code.

```
var country = {};  
country.name = "India";  
country.code = "I001";  
localStorage.setItem("I001", country);  
var country1 = localStorage.getItem("I001");
```

---

If you want to store in JSON format you can use "JSON.stringify" function as shown in the below code.

```
localStorage.setItem("I001",JSON.stringify(country));
```

---

### ***What is the lifetime of local storage?***

Local storage does not have a life time it will stay until either the user clear it from the browser or you remove it using JavaScript code.

### ***What is the difference between local storage and cookies?***

	<b>Cookies</b>	<b>Local storage</b>
<b>Client side / Server side.</b>	Data accessible both at client side and server side. Cookie data is sent to the server side with every request.	Data is accessible only at the local browser side. Server cannot access local storage until deliberately sent to the server via POST or GET.
<b>Size</b>	4095 bytes per cookie.	5 MB per domain.
<b>Expiration</b>	Cookies have expiration attached to it. So after that expiration the cookie and the cookie data get's deleted.	There is no expiration data. Either the end user needs to delete it from the browser or programmatically using JavaScript we need to remove the same.

### ***What is session storage and how can you create one?***

Session storage is same like local storage but the data is valid for a session. In simple words the data is deleted as soon as you close the browser.

To create a session storage you need to use “sessionStorage.variablename” . In the below code we have a created a variable called as “clickcount”.

If you refresh the browser the count increases. But if you close the browser and start again the “clickcount” variable starts from zero.

```
if(sessionStorage.clickcount)
{
sessionStorage.clickcount=Number(sessionStorage.clickcount)+1;
}
else
{
sessionStorage.clickcount = 0;
```

}

## ***What is difference between session storage and local storage?***

Local storage data persists forever but session storage is valid until the browser is open, as soon as the browser closes the session variable resets.

## ***What is WebSQL?***

WebSQL is a structured relational database at the client browser side. It's a local RDBMS inside the browser on which you can fire SQL queries.

## ***Is WebSQL a part of HTML 5 specification?***

No, many people label it as HTML 5 but it's not part of HTML 5 specification. The specification is based around SQLite.

## ***So how can we use WebSQL ?***

The first step we need to do is open the database by using "OpenDatabase" function as shown below. The first argument is the name of the database, the next is the version, then a simple textual title and finally the size of the database.

```
var db=openDatabase('dbCustomer','1.0','Customer app', 2 * 1024 * 1024);
```

---

To execute SQL we then need to use "transaction" function and call "executeSql" function to fire SQL.

```
db.transaction(function (tx)
{
tx.executeSql('CREATE TABLE IF NOT EXISTS tblCust(id unique, customername)');
tx.executeSql('INSERT INTO tblcust (id, customername) VALUES(1, "shiv")');
tx.executeSql('INSERT INTO tblcust (id, customername) VALUES (2, "raju")');
})
```

---

In case you are firing “select” query you will get data is “results” collection which we can loop and display in the HTML UI.

```
db.transaction(function (tx)
{
    tx.executeSql('SELECT * FROM tblcust', [], function (tx, results) {
        for (i = 0; i < results.rows.length; i++)
        {
            msg = "<p><b>" + results.rows.item(i).customerName + "</b></p>";
            document.querySelector('#customer').innerHTML += msg;
        }
    }, null);
});
```

## ***Is Web SQL database still used?***

WebSQL is deprecated and hence forth Indexed DB should be used.

## ***Explain Indexed DB?***

IndexedDB is a key-value pair database provided inside the browser. It is an alternative to Web SQL database because Web SQL Database has been deprecated. So hence forth Indexed DB should be used.

Some of the things to remember about indexed DB are as follows :-

- It stores data in key value pair.
- It's not a relational database and do not support SQL. It has API's via which you can access data.

```
//prefixes of implementation that we want to test
window.indexedDB = window.indexedDB || window.mozIndexedDB || window.webkitIndexedDB
|| window.msIndexedDB;

//prefixes of window.IDB objects
window.IDBTransaction = window.IDBTransaction || window.webkitIDBTransaction ||
window.msIDBTransaction;
window.IDBKeyRange = window.IDBKeyRange || window.webkitIDBKeyRange ||
window.msIDBKeyRange

if (!window.indexedDB) {
    window.alert("Your browser doesn't support a stable version of IndexedDB.")
}
```

```
var db;
var request = window.indexedDB.open("newDatabase", 1);
request.onsuccess = function (event) {
    db = request.result;
};

request.onupgradeneeded = function (event) {

    var db = event.target.result;
    var objectStore = db.createObjectStore("Persons", { keyPath: "id" });

}

var request = db.transaction(["Persons"], "readwrite")
    .objectStore("Persons")
    .add($scope.person);

request.onsuccess = function (event) {
    $scope.person = {};
    $scope.ReadAll();
};

request.onerror = function (event) {
    alert("Unable to");
}

function Person($scope) {
    var db;
    var request = window.indexedDB.open("newDatabase", 1);
    request.onsuccess = function (event) {
        debugger;
        db = request.result;
        $scope.ReadAll();
    };

    request.onupgradeneeded = function (event) {

        var db = event.target.result;
        var objectStore = db.createObjectStore("Persons", { keyPath: "id" });

    }
    debugger;
    $scope.person = {};
    $scope.person.id = 1;
    $scope.person.Name = "Shiv";
    $scope.person.Address = "Mum";
    $scope.persons = [];
    $scope.ReadAll = function(){

    var objectStore = db.transaction("Persons").objectStore("Persons");
    $scope.persons = [];
    objectStore.openCursor().onsuccess = function(event) {
```

```

var cursor = event.target.result;
if (cursor) {
  debugger;

  $scope.persons.push({Name : cursor.value.Name});
  cursor.continue();
}
else {
  $scope.$apply();
  alert("No more entries!");
}
}
}

$scope.Add = function () {
  debugger;
  var request = db.transaction(["Persons"], "readwrite")
    .objectStore("Persons")
    .add($scope.person, $scope.person.id);

  request.onsuccess = function (event) {
    $scope.person = {};
    $scope.ReadAll();
  };

  request.onerror = function (event) {
    alert("Unable to");
  }
}
}

```

## ***What is application cache in HTML5?***

One of the most demanded things by end user is offline browsing. In other words if internet connection is not available page should come from browser cache i.e. offline and application cache helps you to achieve the same.

Application cache helps you to specify which files should be cached and not cached.

## ***So how do we implement application cache in HTML 5 ?***

The first thing in we need to specify is the “manifest” file. “manifest” file helps you to define how your caching should work. Below is the structure of the manifest file :-

CACHE		MANIFEST
#	version	1.0



CACHE  
Login.aspx

---

- All manifest file starts with CACHE MANIFEST statement.
- #( hash tag) helps to provide the version of the cache file.
- CACHE command specifies which files needs to be cached.
- The content type of the manifest file should be “text/cache-manifest”.

Below is how cache manifest has been provided using ASP.NET C#.

```
Response.ContentType = "text/cache-manifest";
Response.Write("CACHE MANIFEST \n");
Response.Write("# 2012-02-21 v1.0.0 \n");
Response.Write("CACHE : \n");
Response.Write("Login.aspx \n");
Response.Flush();
Response.End();
```

---

One the cache manifest file is created the next thing is to provide the link of the manifest file in the HTML page as shown below.

```
<html manifest="cache.aspx">
```

---

When the above file runs first time it gets added in the browser application cache and in case server goes down the page is served from the application cache.

### ***So how do we refresh the application cache of the browser?***

Application cache is removed by changing version number to a new version number as specified in the “#” tag in the below code.

CACHE		MANIFEST
#	version	2.0(new)
CACHE		:
Login.aspx		
Aboutus.aspx		
NETWORK :		
Pages.aspx		

---



## ***What is fallback in Application cache?***

Fallback in application cache helps you to specify the file which will displayed if the server is not reachable. For instance in the below manifest file we are saying if someone hits “/home” and if the server is not reachable then “homeoffline.html” file should be served.

### **FALLBACK:**

/home/ /homeoffline.html

---

## ***What is network in application cache ?***

Network command says files which should not be cached. For example in the below code we are saying that “home.aspx” should never be cached and or available offline.

### **NETWORK:**

home.aspx

---