**How You Will Do Code Review In Ur Project**

1. Make sure that there shouldn't be any project warnings.
2. All unused using’s need to be removed. Code cleanup for unnecessary code is always a good practice.
3. ‘N**ull'** check needs to be performed wherever applicable to avoid the Null Reference Exception at runtime.
4. Camel and Pascal casing to be followed where ever applicable.
5. Redundant and unused code to be removed.
6. Use **access specifiers** (private, public, protected, internal, protected internal) as per the scope need of a method, a class, or a variable.
7. Write **comments**on top of all methods to describe their usage and expected input types and return type information.
8. Avoid **type casting and type conversions** as much as possible; because it is a performance penalty.
9. Check whether any **unreachable code** exists and modify the code if it exists.

**How To Improve Page Performance**

1. Disable Viewstate when not needed.
2. Avoid Session and Application Variables
3. Use Caching
4. Avoid server side validation if possible by using JavaScript
5. Clear the Garbage Collection

**How To Improve Sp Performance**

## Choose Appropriate Data Type

Choose appropriate SQL Data Type to store your data since it also helps in to improve the query performance. Example: To store strings use varchar in place of text data type since varchar performs better than text. Use text data type, whenever you required storing of large text data (more than 8000 characters). Up to 8000 characters data you can store in varchar.

## Avoid nchar and nvarchar

Practice to avoid nchar and nvarchar data type since both the data types takes just double memory as char and varchar. Use nchar and nvarchar when you required to store Unicode (16-bit characters) data like as Hindi, Chinese characters etc.

## Avoid NULL in fixed-length field

Practice to avoid the insertion of NULL values in the fixed-length (char) field. Since, NULL takes the same space as desired input value for that field. In case of requirement of NULL, use variable-length (varchar) field that takes less space for NULL.

## Avoid \* in SELECT statement

Practice to avoid \* in Select statement since SQL Server converts the \* to columns name before query execution. One more thing, instead of querying all columns by using \* in select statement, give the name of columns which you required.

* 1. ***-- Avoid***
  2. **SELECT \* FROM tblName**
  3. ***--Best practice***
  4. **SELECT col1,col2,col3 FROM tblName**

## Use EXISTS instead of IN

Practice to use EXISTS to check existence instead of IN since EXISTS is faster than IN.

* 1. ***-- Avoid***
  2. **SELECT Name,Price FROM tblProduct**
  3. **where ProductID IN (Select distinct ProductID from tblOrder)**
  4. ***--Best practice***
  5. **SELECT Name,Price FROM tblProduct**
  6. **where ProductID EXISTS (Select distinct ProductID from tblOrder)**

## Avoid Having Clause

Practice to avoid Having Clause since it acts as filter over selected rows. Having clause is required if you further wish to filter the result of an aggregations. Don't use HAVING clause for any other purpose.

## Create Clustered and Non-Clustered Indexes

Practice to create clustered and non clustered index since indexes helps in to access data fastly. But be careful, more indexes on a tables will slow the INSERT,UPDATE,DELETE operations. Hence try to keep small no of indexes on a table.

## Keep clustered index small

Practice to keep clustered index as much as possible since the fields used in clustered index may also used in nonclustered index and data in the database is also stored in the order of clustered index. Hence a large clustered index on a table with a large number of rows increase the size significantly. Please refer the article[Effective Clustered Indexes](http://www.simple-talk.com/sql/learn-sql-server/effective-clustered-indexes/)

## Avoid Cursors

Practice to avoid cursor since cursor are very slow in performance. Always try to use SQL Server cursor alternative. Please refer the article [Cursor Alternative](http://www.dotnet-tricks.com/Tutorial/sqlserver/IT5G180512-SQL-Server-Cursor-Alternatives.html).

## Use Table variable inplace of Temp table

Practice to use Table varible in place of Temp table since Temp table resides in the TempDb database. Hence use of Temp tables required interaction with TempDb database that is a little bit time taking task.

## Use UNION ALL inplace of UNION

Practice to use UNION ALL in place of UNION since it is faster than UNION as it doesn't sort the result set for distinguished values.

## Use Schema name before SQL objects name

Practice to use schema name before SQL object name followed by "." since it helps the SQL Server for finding that object in a specific schema. As a result performance is best.

* 1. ***--Here dbo is schema name***
  2. **SELECT col1,col2 from dbo.tblName**
  3. ***-- Avoid***
  4. **SELECT col1,col2 from tblName**

## Keep Transaction small

Practice to keep transaction as small as possible since transaction lock the processing tables data during its life. Some times long transaction may results into deadlocks. Please refer the article [SQL Server Transactions Management](http://www.dotnet-tricks.com/Tutorial/sqlserver/c2XF120412-SQL-Server-Transactions-Management.html)

## SET NOCOUNT ON

Practice to set NOCOUNT ON since SQL Server returns number of rows effected by SELECT,INSERT,UPDATE and DELETE statement. We can stop this by setting NOCOUNT ON like as:

* 1. **CREATE PROCEDURE dbo.MyTestProc**
  2. **AS**
  3. **SET NOCOUNT ON**
  4. **BEGIN**
  5. **.**
  6. **.**
  7. **END**

## Use TRY-Catch

Practice to use TRY-CATCH for handling errors in T-SQL statements. Sometimes an error in a running transaction may cause deadlock if you have no handle error by using TRY-CATCH. Please refer the article[Exception Handling by TRY…CATCH](http://www.dotnet-tricks.com/Tutorial/sqlserver/O3P3120412-SQL-Server-Exception-Handling-by-TRY%E2%80%A6CATCH.html)

## Use Stored Procedure for frequently used data and more complex queries

Practice to create stored procedure for quaery that is required to access data frequently. We also created stored procedure for resolving more complex task.

## Avoid prefix "sp\_" with user defined stored procedure name

Practice to avoid prefix "sp\_" with user defined stored procedure name since system defined stored procedure name starts with prefix "sp\_". Hence SQL server first search the user defined procedure in the master database and after that in the current session database. This is time consuming and may give unexcepted result if system defined stored procedure have the same name as your defined procedure.

**Exception handling in mvc using exceptionfilter**

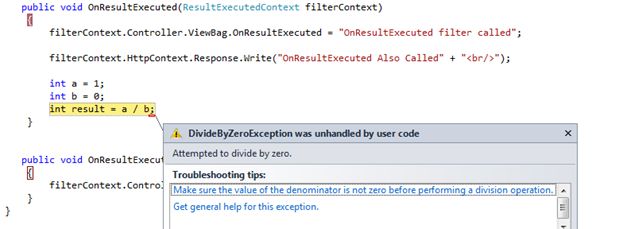
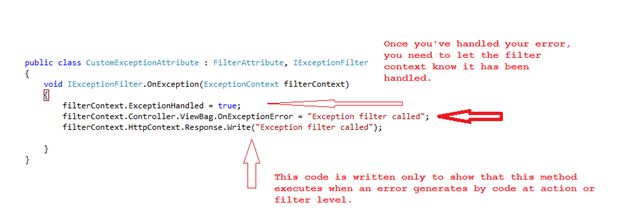
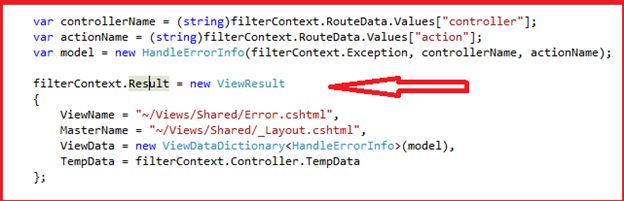
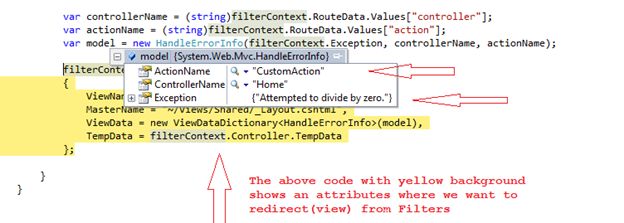
This article describes how to return a desired view from an IExceptionFilter in MVC with practices. It is a part of exception handling at the filter level. Custom filters and attributes are an excellent way of injecting extra processing logic into the MVC request response pipeline. This is another most frequently asked question.  
  
  
  
In MVC applications the user first lands on a routing module and matches the URL to best fit the route collection, this module parses the request and invokes the respected controller and action and then the controller renders the view in the browser.  
  
Now somewhere we would like to inject some pre-processing and post-processing logic for actions and controllers. In that case we use filters. IExceptionFilter is also a part of filters.  
  
You can also have a look at articles related to MVC here:

* [Smart Working With Custom Value Providers in ASP.NET MVC](http://www.c-sharpcorner.com/UploadFile/97fc7a/smart-working-with-custom-value-providers-in-Asp-Net-mvc/)
* [EXECUTION ORDER OF FILTERS IN MVC 4 WITH PRACTICES: IMPORTANT FAQ](http://www.dotnetpiper.com/2014/06/execution-order-of-filters-in-mvc-4.html)
* [Invoke Action With Model Binders in MVC](http://www.c-sharpcorner.com/UploadFile/97fc7a/invoke-action-with-model-binders-in-mvc/)
* [Extension Helpers Method in MVC](http://www.c-sharpcorner.com/UploadFile/97fc7a/extension-helpers-method-in-mvc/)
* [Custom Button With TagBuilder Using MVC Razor Engine](http://www.c-sharpcorner.com/UploadFile/97fc7a/custom-button-with-tagbuilder-technique-using-mvc-razor-engi/)
* [Precompiled Razor View Using RazorGenerator MVC and PreCompiledViewEngine in MVC 4](http://www.c-sharpcorner.com/UploadFile/97fc7a/article-precompiled-razor-view-using-razorgenerator-mvc-and/)

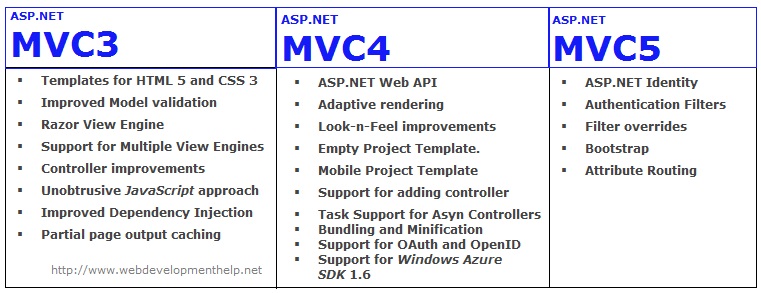
**Result filter**  
  
This filter executes before and after the result of the action method has executed. We can use this filter if we want some modification to be done in the action's result.

* OnResultExecuting occurs just before the result is executed (before the view is rendered)
* OnResultExecuted occurs after the result is executed (after the view is rendered)

To implement the result filters we need to create a custom filter attribute class and implement the IResultFilter interface. This interface provides the two methods **OnResultExecuting**and**OnResultExecuted**.  
  
When we execute the code we get an exception as shown in the image below:

  
  
  
We can handle an error using an Exception Filter.  
  
This filter will be invoked whenever a controller or action of the controller throws an exception. This is particularly useful when we need a custom error logging module.  
  
To implement this filter we need to create a custom filter attribute class that implements an IExceptionFilter. Here we have also set the ExceptionHandled property to let the filter know that an exception has been handled at the filter level. Kindly have a look at the image depicted below:  
  
  
Here I've also added a code segment to handle an error at the OnException method level.  
  
  
  
We can easily identify an error and related details after placing the code segment into an OnException filter. Kindly see the image shown below:  
  
  
  
Now press F5 and see the output to understand the execution order of filters. Please see the depicted image below.  
  
As shown in the above image we have been redirected to a Custom Error page from the action level. I hope it will help you to understand the exception handling in MVC.

**Different Mvc Versions And Features :**



## Asp.Net MVC1

1. Released on Mar 13, 2009
2. Runs on .Net 3.5 and with Visual Studio 2008 & Visual Studio 2008 SP1
3. MVC Pattern architecture with WebForm Engine
4. Html Helpers
5. Ajax helpers
6. Routing
7. Unit Testing

## Asp.Net MVC2

1. Released on Mar 10, 2010
2. Runs on .Net 3.5, 4.0 and with Visual Studio 2008 & 2010
3. Strongly typed HTML helpers means lambda expression based Html Helpers
4. Templated Helpers
5. Support for Data Annotations Attribute
6. Client-side validation
7. UI helpers with automatic scaffolding & customizable templates
8. Attribute-based model validation on both client and server
9. Overriding the HTTP Method Verb including GET, PUT, POST, and DELETE
10. Areas for partitioning a large applications into modules
11. Asynchronous controllers

## Asp.Net MVC3

1. Released on Jan 13, 2011
2. Runs on .Net 4.0 and with Visual Studio 2010
3. The Razor view engine
4. Improved Support for Data Annotations
5. Remote Validation
6. Compare Attribute
7. Sessionless Controller
8. Child Action Output Caching
9. Dependency Resolver
10. Entity Framework Code First support
11. Partial-page output caching
12. ViewBag dynamic property for passing data from controller to view
13. Global Action Filters
14. Better JavaScript support with unobtrusive JavaScript, jQuery Validation, and JSON binding
15. Use of NuGet to deliver software and manage dependencies throughout the platform
16. Good Intellisense support for Razor into Visual Studio

## Asp.Net MVC4

1. Released on Aug 15, 2012
2. Runs on .Net 4.0, 4.5 and with Visual Studio 2010SP1 & Visual Studio 2012
3. ASP.NET Web API
4. Enhancements to default project templates
5. Mobile project template using jQuery Mobile
6. Display Modes
7. Task support for Asynchronous Controllers
8. Bundling and minification
9. Support for the Windows Azure SDK

## Asp.Net MVC5

1. Released on 17 October 2013
2. Runs on .Net 4.5, 4.5.1 and with Visual Studio 2013
3. One Asp.Net
4. Asp.Net Identity
5. ASP.NET Scaffolding
6. Authentication filters - run prior to authorization filters in the ASP.NET MVC pipeline
7. Bootstrap in the MVC template
8. ASP.NET Web API2

**Different C# Version And Features :**

## .NET Framework versions

There have been seven significant releases of the .NET Framework, excluding service packs. The framework includes the compilers, runtime, and libraries. Additionally, there are other profiles such Silverlight which complicate matters.

* 1.0 - released in 2002
* 1.1 - released in 2003
* 2.0 - released in 2005, with a new CLR (to handle generics and nullable types) and compilers for C# 2 and VB 8.
* 3.0 - released in 2006, this is just 2.0 plus new libraries: Windows Presentation Foundation, Windows Communication Foundation, Workflow Foundation, and Cardspace
* 3.5 - released in 2007, this is 3.0 plus new libraries (primarily LINQ and some extra "base" libraries such as TimeZoneInfo) and new compilers (for C# 3 and VB 9)
* 4 - released in 2010, this includes a new CLR (v4), new libraries, and the DLR (Dynamic Language Runtime)
* 4.5 - released in 2012, this allows for WinRT development on Windows 8 as well as extra libraries - with much wider async APIs

## C# language versions

There are five significant language versions:

* C# 1
* C# 2, introducing generics, nullable types, anonymous methods, iterator blocks and some other more minor features
* C# 3, introducing implicit typing, object and collection initializers, anonymous types, automatic properties, lambda expressions, extension methods, query expressions and some other minor features
* C# 4, introducing dynamic typing, optional parameters, named arguments, and generic variance
* C# 5, introducing asynchronous functions, caller info attributes, and a tweak to foreach iteration variable capture

See the [specifications page](http://csharpindepth.com/Articles/Chapter1/Specifications.aspx) for downloads for each version, from Microsoft and ECMA.

**Different Sql Version And Features :**

### 7. SQL Server CE

As you'd expect, SQL Server 2000 Windows CE Edition uses a different code base than other SQL Server editions do. SQL Server CE has about a 1MB footprint and doesn't include Analysis Services or Data Transformation Services (DTS). SQL Server CE supports data exchange with other SQL Server editions through anonymous merge replication.

### 6. Enterprise Evaluation Edition

The Enterprise Evaluation Edition is essentially the same as the Enterprise Edition; however, the Evaluation Edition has a built-in 120-day time limit. You can upgrade from the Evaluation Edition to either the standard or enterprise edition.

### 5. Developer Edition

Supplied as a part of Visual Studio Enterprise Edition, the Developer Edition supports the same feature set as SQL Server Enterprise Edition does. However, licensing for the Developer Edition allows only development work; you can't use this edition as a production database.

### 4. Personal Edition

The Personal Edition comes with the SQL Server Standard Edition or Enterprise Edition—you can't purchase it separately. The Personal Edition runs on Windows 2000, Windows NT, or Windows 9x and supports one to two processors on Win2K Professional. However, this edition supports only five concurrent connections. Each Personal Edition requires a Client Access License (CAL).

### 3. MSDE

Microsoft SQL Server Desktop Engine (MSDE) is a standalone runtime engine for database applications and can be freely distributed. MSDE's core database engine is essentially the same as that of the Personal Edition. However, MSDE provides no management tools and doesn't require a CAL.

### 2. Standard Edition

The Standard Edition is the most popular version of SQL Server 2000. This edition requires a minimum of Win2K Server or NT Server 4.0 and supports one to four processors. The Standard Edition includes Analysis Services, DTS, XML integration, and English Query, as well as the Personal Edition and MSDE. Each client that connects to the Standard Edition needs a CAL.

### 1. Enterprise Edition

The Enterprise Edition includes all the Standard Edition features plus built-in Storage Area Network (SAN) support, federated databases, log shipping, and partitioned OLAP cubes. The Enterprise Edition runs on Win2K Server or NT Server 4.0 or later; when running on Win2K Advanced Server or NT Server, Enterprise Edition (NTS/E), this edition supports more than four processors and four-node failover clustering. Each client must have a CAL.

**BASIC C# QUESTIONS :**

**5. What is an object?**

An object is an instance of a class through which we access the methods of that class. “New” keyword is used to create an object. A class that creates an object in memory will contain the information about the methods, variables and behavior of that class.

**6. Define Constructors?**

A constructor is a member function in a class that has the same name as its class. The constructor is automatically invoked whenever an object class is created. It constructs the values of data members while initializing the class.

**7. What is Jagged Arrays?**

The array which has elements of type array is called jagged array. The elements can be of different dimensions and sizes. We can also call jagged array as Array of arrays.

**8. What is the difference between ref & out parameters?**

An argument passed as ref must be initialized before passing to the method whereas out parameter needs not to be initialized before passing to a method.

**9. What is the use of using statement in C#?**

The using block is used to obtain a resource and use it and then automatically dispose of when the execution of block completed.

**10. What is serialization?**

When we want to transport an object through network then we have to convert the object into a stream of bytes. The process of converting an object into a stream of bytes is called Serialization. For an object to be serializable, it should inherit ISerialize Interface.  
De-serialization is the reverse process of creating an object from a stream of bytes.

**11. Can “this” be used within a static method?**

We can’t use ‘This’ in a static method because we can only use static variables/methods in a static method.

**12. What is difference between constants and read-only?**

Constant variables are declared and initialized at compile time. The value can’t be changed after wards. Read-only variables will be initialized only from the Static constructor of the class. Read only is used only when we want to assign the value at run time.

**13. What is an interface class?**

Interface is an abstract class which has only public abstract methods and the methods only have the declaration and not the definition. These abstract methods must be implemented in the inherited classes.

**14. What are value types and reference types?**

Value types are stored in the Stack whereas reference types stored on heap.  
Value types:

int, enum , byte, decimal, double, float, long

Reference Types:

string , class, interface, object

**MVC QUESTIONS :**

**Breifly explain us what is ASP.Net MVC?**

ASP.Net MVC is a pattern which is used to split the application's implementation logic into three components i.e. models, views, and controllers.

**Tell us something about Model, view and Controllers in Asp.Net MVC?**

Model : It is basically a business entity which is used to represent the application data. Controller : The Request which is sent by the user always scatters through controller and it's responsibility is to redirect to the specific view using View () method. View : it's the presentation layer of ASP.Net MVC.

**Do you know about the new features in ASP.Net MVC 4 (ASP.Net MVC4)?**

Following are features added newly : Mobile templates Added ASP.NET Web API template for creating REST based services. Asynchronous controller task support. Bundling of the java scripts. Segregating the configs for ASP.Net MVC routing, Web API, Bundle etc.

**How does the 'page lifecycle' of ASP.Net MVC works?**

Below are the processed followed in the sequence -

* App initializWhat is Separation of Concerns in ASP.NET ASP.Net MVCation
* Routing
* Instantiate and execute controller
* Locate and invoke controller action
* Instantiate and render view.

**Explain the advantages of ASP.Net MVC over ASP.NET?**

* Provides a clean separation of concerns among UI (Presentation layer), model (Transfer objects/Domain Objects/Entities) and Business Logic (Controller).
* Easy to UNIT Test.
* Improved reusability of model and views. We can have multiple views which can point to the same model and vice versa.
* Improved structuring of the code.

**What is Separation of Concerns in ASP.NET ASP.Net MVC?**

It is the process of breaking the program into various distinct features which overlaps in functionality as little as possible. ASP.Net MVC pattern concerns on separating the content from presentation and data-processing from content.

**JQUERY QUESTIONS :**

### What is jQuery?

Ans: jQuery is **fast, lightweight and feature-rich**client side JavaScript Library/Framework which helps in to traverse HTML DOM, make animations, add Ajax interaction, manipulate the page content, change the style and provide cool UI effect. It is one of the most popular client side library and as per a survey it runs on every second website.

### Q2. Why do we use jQuery?

Ans: Due to following advantages.

* Easy to use and learn.
* Easily expandable.
* Cross-browser support (IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+)
* Easy to use for DOM manipulation and traversal.
* Large pool of built in methods.
* AJAX Capabilities.
* Methods for changing or applying CSS, creating animations.
* Event detection and handling.
* Tons of plug-ins for all kind of needs.

### Q3. How JavaScript and jQuery are different?

Ans: JavaScript is a language While jQuery is a library built in the JavaScript language that helps to use the JavaScript language.

### Q4. Is jQuery replacement of Java Script?

Ans: **No.** jQuery is not a replacement of JavaScript. jQuery is a different library which is written on top of JavaScript. jQuery is a lightweight JavaScript library that emphasizes interaction between JavaScript and HTML.

### Q5. Is jQuery a library for client scripting or server scripting?

Ans. Client side scripting.

### Q6. Is jQuery a W3C standard?

Ans: No. jQuery is not a W3C standard.

### Q7. What is the basic need to start with jQuery?

Ans: To start with jQuery, one need to make reference of it's library. The latest version of jQuery can be downloaded from [jQuery.com](http://jquery.com/).

### Q8. Which is the starting point of code execution in jQuery?

Ans: The starting point of jQuery code execution is $(document).ready() function which is executed when DOM is loaded.

### Q9. What does dollar sign ($) means in jQuery?

Ans: Dollar Sign is nothing but it's an alias for JQuery. Take a look at below jQuery code.

Hide   Copy Code

$(document).ready(function(){

});

Over here $ sign can be replaced with "jQuery" keyword.

Hide   Copy Code

jQuery(document).ready(function(){

});

### Q10. Can we have multiple document.ready() function on the same page?

Ans: **YES**. We can have any number of document.ready() function on the same page.

### Q11. Can we use our own specific character in the place of $ sign in jQuery?

Ans: **Yes.** It is possible using jQuery.noConflict().

### Q12. Is it possible to use other client side libraries like MooTools, Prototype along with jQuery?

Ans: **Yes.**

**OOPS Concepts :**

### ****Encapsulation****

Encapsulation is the process of wrapping up of data (properties) and behavior (methods) of an object into a single unit; and the unit here is a Class (or interface).

Encapsulate in plain English means to enclose or be enclosed in or as if in a capsule. In Java, everything is enclosed within a class or interface, unlike languages such as C and C++ where we can have global variables outside classes.

Encapsulation enables **data hiding,**hiding irrelevant information from the users of a class and exposing only the relevant details required by the user. We can expose our operations hiding the details of what is needed to perform that operation.

#### 

### ****Inheritance****

Inheritance describes the parent child relationship between two classes.

A class can get some of its characteristics from a parent class and then add more unique features of its own. For example, consider a Vehicle parent class and a child class Car. Vehicle class will have properties and functionalities common for all vehicles. Car will inherit those common properties from the Vehicle class and then add properties which are specific to a car.

In the above example, Vehicle parent class is known as base class or superclass. Car is known as derived class, Child class or subclass.

Java supports single-parent, multiple-children inheritance and multilevel inheritence (Grandparent-> Parent -> Child) for classes and interfces.

Java supports multiple inheritance (multiple parents, single child) only through interfaces. This is done to avoid some confusions and errors such as diamond problem of inheritance.

### ****Plymorphism****

The ability to change form is known as polymorphism. Java supports different kinds of polymorphism like oveloading and overriding.

**Overloading**

The same method name (method overloading) or operator symbol (operator overloading) can be used in different contexts.

Java doesn't allow operator overloading except that "+" is overloaded for class String. The "+" operator can be used for addition as well as string concatenation.

Overloading may be also called compile time polymorphism.

**Overriding (or subtype polymorphism)**

We can override an instance method of parent class in the child class.

When you refer to a child class object using a Parent reference (e.g.  Parent p = new Child()) and invoke a method, the overriden child class method will be invoked. Here, the actual method called will depend on the object at runtime, not the reference type.

Overriding is not applicable for static methods or variables (static and non-static). In case of variables (static and non-static) and static methods, when you invoke a method using a reference type variable, the method or variable that belong to the reference type is invoked.

Overriding may be also called runtime polymorphism.

### ****Abstraction****

In plain English, abstract is a concept or idea not associated with any specific instance and does not have a concrete existence.

Abstraction in Object Oriented Programming refers to the ability to make a class abstract.

Abstraction captures only those details about an object that are relevant to the current perspective, so that the programmer can focus on a few concepts at a time.

Java provides interfaces and abstract classes for describing abstract types.

* An **interface**is a contract or specification without any implementation. An interface can't have behavior or state.
* An **abstract class** is a class that cannot be instantiated, but has all the properties of a class including constructors. Abstract classes can have state and can be used to provide a skeletal implementation.

**SP and functions difference**

## Basic Difference

1. Function must return a value but in Stored Procedure it is optional( Procedure can return zero or n values).
2. Functions can have only input parameters for it whereas Procedures can have input/output parameters .
3. Functions can be called from Procedure whereas Procedures cannot be called from Function.

## Advance Difference

1. Procedure allows SELECT as well as DML(INSERT/UPDATE/DELETE) statement in it whereas Function allows only SELECT statement in it.
2. Procedures can not be utilized in a SELECT statement whereas Function can be embedded in a SELECT statement.
3. Stored Procedures cannot be used in the SQL statements anywhere in the WHERE/HAVING/SELECT section whereas Function can be.
4. The most important feature of stored procedures over function is to retention and reuse the execution plan while in case of function it will be compiled every time.
5. Functions that return tables can be treated as another rowset. This can be used in JOINs with other tables.
6. Inline Function can be though of as views that take parameters and can be used in JOINs and other Rowset operations.
7. Exception can be handled by try-catch block in a Procedure whereas try-catch block cannot be used in a Function.
8. We can go for Transaction Management in Procedure whereas we can't go in Function.

**TRIGGERS :**

## Types of Triggers

In Sql Server we can create four types of triggers Data Definition Language (DDL) triggers, Data Manipulation Language (DML) triggers, CLR triggers and Logon triggers.

## DDL Triggers

In SQL Server we can create triggers on DDL statements (like CREATE, ALTER, and DROP) and certain system defined stored procedures that perform DDL-like operations.

**Example :**If you are going to execute the CREATE LOGIN statement or the sp\_addlogin stored procedure to create login user, then both these can execute/fire a DDL trigger that you can create on CREATE\_LOGIN event of Sql Server.

**We can use only FOR/AFTER clause in DDL triggers not INSTEAD OF clause means we can make only After Trigger on DDL statements.**

DDL trigger can be used to observe and control actions performed on the server, and to audit these operations. DDL triggers can be used to manage administrator tasks such as auditing and regulating database operations.

## DML Triggers

In SQL Server we can create triggers on DML statements (like INSERT, UPDATE, and DELETE) and stored procedures that perform DML-like operations. DML Triggers are of two types

## After Trigger (using FOR/AFTER CLAUSE)

This type of trigger fires after SQL Server finish the execution of the action successfully that fired it.

**Example :**If you insert record/row in a table then the trigger related/associated with the insert event on this table will fire only after the row passes all the constraints, like as primary key constraint, and some rules. If the record/row insertion fails, SQL Server will not fire the After Trigger.

## Instead of Trigger (using INSTEAD OF CLAUSE)

This type of trigger fires before SQL Server starts the execution of the action that fired it. This is differ from the AFTER trigger, which fires after the action that caused it to fire. We can have an INSTEAD OF insert/update/delete trigger on a table that successfully executed but does not include the actual insert/update/delete to the table.

**Example :**If you insert record/row in a table then the trigger related/associated with the insert event on this table will fire before the row passes all the constraints, such as primary key constraint and some rules. If the record/row insertion fails, SQL Server will fire the Instead of Trigger.

## CLR Triggers

CLR triggers are special type of triggers that based on the CLR (Common Language Runtime) in .net framework. CLR integration of triggers has been introduced with SQL Server 2008 and allows for triggers to be coded in one of .NET languages like C#, Visual Basic and F#.

We coded the objects(like trigger) in the CLR that have heavy computations or need references to objects outside the SQL Server. We can write code for both DDL and DML triggers, using a supported CLR language like C#, Visual basic and F#. I will discuss CLR trigger later.

## Logon Triggers

Logon triggers are special type of trigger that fire when LOGON event of Sql Server is raised. This event is raised when a user session is being established with Sql Server that is made after the authentication phase finishes, but before the user session is actually established. Hence, all messages that we define in the trigger such as error messages, will be redirected to the SQL Server error log. Logon triggers do not fire if authentication fails. We can use these triggers to audit and control server sessions, such as to track login activity or limit the number of sessions for a specific login.

**Synatx for Logon Trigger**

* 1. **CREATE TRIGGER trigger\_name**
  2. **ON ALL SERVER**
  3. **[WITH ENCRYPTION]**
  4. **{FOR|AFTER} LOGON**
  5. **AS**
  6. **sql\_statement [1...n ]**

## Syntax for Trigger

1. **CREATE TRIGGER trigger\_name**
2. **ON {table|view}**
3. **[WITH ENCRYPTION|EXECUTE AS]**
4. **{FOR|AFTER|INSTEAD OF} {[CREATE|ALTER|DROP|INSERT|UPDATE|DELETE ]}**
5. **[NOT FOR REPLICATION]**
6. **AS**
7. **sql\_statement [1...n ]**

## trigger\_name

This is the name of the trigger. It should conform to the rules for identifiers in Sql Server.

## table|view

This is the table/view on which the trigger is to be created.

## ENCRYPTION

This option is optional. If this option is specified, original text of the CREATE TRIGGER statement will be encrypted.

## EXECUTE AS

This option is optional. This option specifies, the security context under which the trigger is executed.

## FOR/AFTER

FOR/AFTER specifies that the trigger is After Trigger. AFTER is the default, if FOR is the only keyword specified.**AFTER triggers cannot be defined on views**.

## INSTEAD OF

INSTEAD OF specifies that the trigger is Instead Of Trigger.

## CREATE|ALTER|DROP|INSERT|UPDATE|DELETE

These keywords specify on which action the trigger should be fired. One of these keywords or any combination of these keywords in any order can be used.

## NOT FOR REPLICATION

Indicates that the trigger should not be executed when a replication process modifies the table involved in the trigger.

## AS

After this we specifies the actions and condition that the trigger perform.

## sql\_statement

These are the trigger conditions and actions. The trigger actions specified in the T-SQL statements.

**GARBAGE COLLETION :**

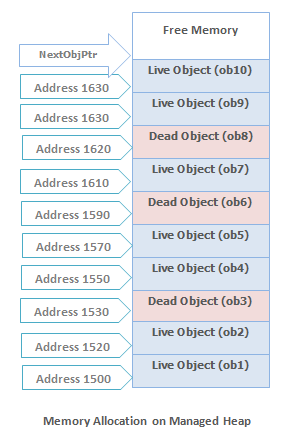
mory management is the main concern for any application whether application is window based or web based. In .Net, CLR has garbage collector that executes as a part of our program and responsible for reclaiming the memory of no longer used objects. Garbage collector free the memory for objects that are no longer referenced and keeps the memory for future allocations.

## Advantage of Garbage Collector

1. Allow us to develop an application without having worry to free memory.
2. Allocates memory for objects efficiently on the managed heap.
3. Reclaims the memory for no longer used objects and keeps the free memory for future allocations.
4. Provides memory safety by making sure that an object cannot use the content of another object.

## Memory Allocation in Managed Heap

The managed heap is a series of allocated memory segments (approx 16Mb in size each) to store and manage objects. The memory for newly created object is allocated at the next available location on the managed heap. If there is available free memory, the garbage collector doesn't search the dead objects for memory reclaim and memory allocations has been done very fast. If the memory is insufficient to create the object, the garbage collector search the dead objects for memory reclaim for the newly object.



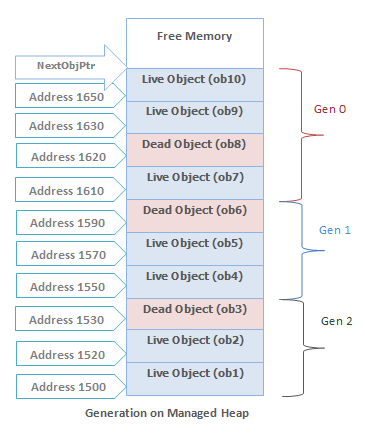
An object is created using the new operator. This operator first makes sure that the bytes required by the new object fit in the reserved region (committing storage if necessary). If the object fits, **NextObjPtr** points to the object in the heap and object's constructor is called and the new operator returns the address of the object.

## Key points about Garbage Collector

1. All objects in the heap are allocated from one contiguous range of memory address and heap is divided into generations so that it is easy to eliminate the garbage objects by looking at only a small fraction of the heap.
2. Gen 0 and Gen 1 occupy a single segment known as the ephemeral segment. Gen 2 is a set of further segments and the large object heap is yet another group of segments.
3. Almost, all objects with-in a generation are of the same age.
4. The newest objects are created at higher memory address while oldest memory objects are at lowest memory address with in the heap.
5. The allocation pointer for the new objects marks the boundary between the allocated and free memory.
6. Periodically the heap is compacted by removing the dead objects and sliding up the live objects towards the lower memory address end of the heap as shown in above fig.
7. The order of objects (after memory reclaims) in memory remains the same as they were created.
8. There are never any gaps among the objects in the heap.
9. Only some of the free memory is committed when required and more memory is acquired from the OS in the reserved address range.

## Generations in Managed Heap

The managed heap is organized into three generations so that it can handle short lived and long lived objects efficiently. Garbage collector first reclaim the short lived objects that occupy a small part of the heap.



## Generation 0

This is the youngest generation and contains the newly created objects. Generation 0 has short-lived objects and collected frequently. The objects that survive the Generation 0 are promoted to Generation 1.

**Example :** A temporary object.

## Generation 1

This generation contains the longer lived objects that are promoted from generation 0. The objects that survive the Generation 1 are promoted to Generation 2. Basically this generation serves as a buffer between short-lived objects and longest-lived objects.

## Generation 2

This generation contains the longest lived objects that are promoted from generation 1 and collected infrequently.

**Example :** An object at application level that contains static data which is available for the duration of the process.

## Garbage Collector Working Phase

## Marking Phase

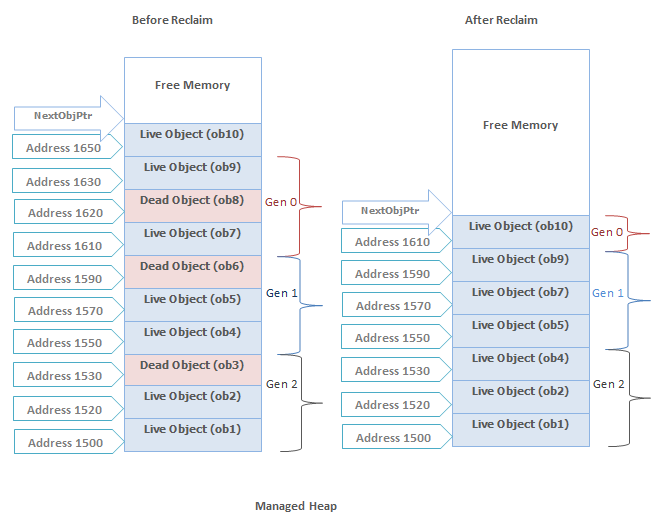
In this phase garbage collector finds and creates a list of all live objects.

## Relocating Phase

In this phase garbage collector updates the references to the objects that will be compacted.

## Compacting Phase

In this phase garbage collector reclaims the memory occupied by the dead objects and compacts the surviving objects. The compacting phase moves the surviving objects toward the older end of the memory segment.



#### Note

1. The large object heap is not compacted, because copying large objects imposes a performance penalty.

## Garbage Collection Algorithm

Garbage collector determine whether any object in the heap is dead or not being used by the application. If such objects exist then memory used by these objects can be reclaimed. But how garbage collector know about these objects?

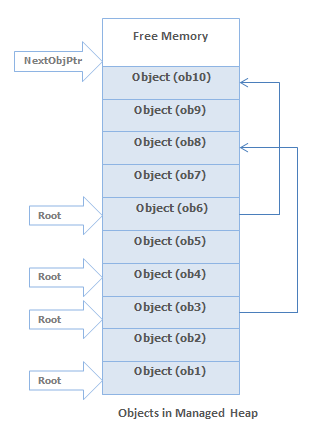
Each and every application has a set of roots and these identify the storage locations for the objects on the managed heap.

**Example :** All the global, static objects pointers and all the local variable/ parameter object pointers on the thread's stack in the application are considered part of the application's roots. More over any CPU registers containing pointers to objects in the managed heap are also considered a part of the application's roots.

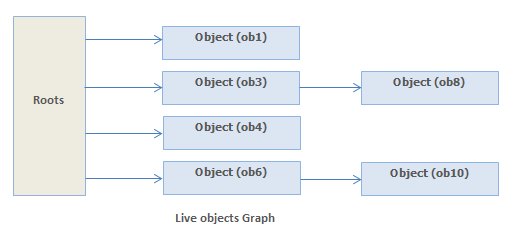
The list of active roots is maintained by the JIT compiler and CLR, and is made accessible to the garbage collector's algorithm.

### Memory Reclaim Process

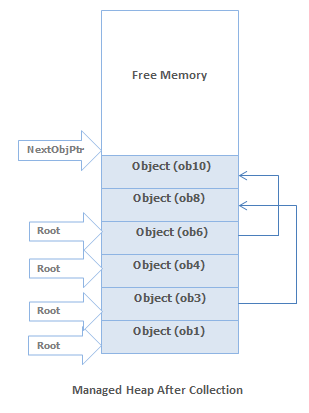
Now the garbage collector starts go through the roots and make a graph of all the objects reachable from the roots. The below fig. shows a heap with allocated objects. In this heap the application roots directly refer to the objects 1,3,4,6 and object 3 & 6 refers to the objects 8 & 10. Hence all these objects will become the part of the live objects graph.



The objects which are not reachable from application's roots, are considered as garbage since these are not accessible by the application. In above heap objects 2,5,7,9 will be considered as dead objects.



The garbage collector then remove the dead objects from the heap and live objects will move toward the older end of the memory segment as shown in below fig. Garbage collector also updates all the references(including root references) to the moving objects in the heap.



**Ref, Out Keywords In C# :**

Ref and out parameters are used to pass an argument within a method. In this article, you will learn the differences between these two parameters.

## Ref

The ref keyword is used to pass an argument as a reference. This means that when value of that parameter is changed in the method, it gets reflected in the calling method. An argument that is passed using a ref keyword must be initialized in the calling method before it is passed to the called method.

## Out

The out keyword is also used to pass an argument like ref keyword, but the argument can be passed without assigning any value to it. An argument that is passed using an out keyword must be initialized in the called method before it returns back to calling method.

### Program with ref and out keyword

1. **public class Example**
2. **{**
3. **public static void Main() *//calling method***
4. **{**
5. **int val1 = 0; *//must be initialized***
6. **int val2; *//optional***
8. **Example1(ref val1);**
9. **Console.WriteLine(val1); *// val1=1***
11. **Example2(out val2);**
12. **Console.WriteLine(val2); *// val2=2***
13. **}**
15. **static void Example1(ref int value) *//called method***
16. **{**
17. **value = 1;**
18. **}**
19. **static void Example2(out int value) *//called method***
20. **{**
21. **value = 2; *//must be initialized***
22. **}**
23. **}**
25. ***/\* Output***
26. ***1***
27. ***2***
28. ***\*/***

**ASSCEES SPECIFIERS :**

# C# Access Modifiers , C# Access Specifiers

**Access Modifiers (Access Specifiers)** describes as the scope of accessibility of an Object and its members. All C# types and type members have an accessibility level . We can control the scope of the member object of a class using access specifiers. We are using access modifiers for providing security of our applications. When we specify the accessibility of a type or member we have to declare it by using any of the access modifiers provided by [CSharp](http://csharp.net-informations.com/) language.

C# provide five access specifiers , they are as follows :

***public, private , protected , internal and protected internal*** .

**public :**

public is the most common access specifier in C# . It can be access from anywhere, that means there is no restriction on accessibility. The scope of the accessibility is inside class as well as outside. The type or member can be accessed by any other code in the same assembly or another assembly that references it.

**private :**

The scope of the accessibility is limited only inside the classes or struct in which they are declared. The private members cannot be accessed outside the class and it is the least permissive access level.

**protected :**

The scope of accessibility is limited within the class or struct and the class derived (Inherited )from this class.

**internal :**

The internal access modifiers can access within the program that contain its declarations and also access within the same assembly level but not from another assembly.

**protected internal :**

Protected internal is the same access levels of both protected and internal. It can access anywhere in the same assembly and in the same class also the classes inherited from the same class .

**MVC LIFE CYCLE :**

****

**LINQ Sample Queries :**

#### Get all the records from tbl\_Dept

Hide   Copy Code

Select \* from dbo.tbl\_Dept // T-Sql

var res= dev.tbl\_Dept.ToList(); //lambda

var res=from res in dev.tbl\_Dept select res; //query

#### Get all the records from tbl\_Dept with column aliasing

Hide   Copy Code

Select Did as &lsquo;Department Id&rsquo;, \_

DName as &lsquo;Department Name&rsquo; from dbo.tbl\_Dept // T-Sql

var res = dev.tbl\_Dept.Select(x => \_

new { DepartmentId = x.Did, DepartmentName = x.DName }); //lambda

var res=from re in dev.tbl\_Dept \_

select new{Department\_Id=re.Did,Department\_Name=re.DName}; //query

#### Get top two records from tbl\_Dept

Hide   Copy Code

Select top(2) \* from tbl\_Dept // T-Sql

var res = dev.tbl\_Dept.Take(2).ToList(); //lambda

var res = from re in dev.tbl\_Dept.Take(2) select re; // query

#### Get all the records from tbl\_Dept which are sorted by Did ascending

Hide   Copy Code

select \* from tbl\_Dept order by Did

var res = dev.tbl\_Dept.OrderBy(x => x.Did).ToList();//lambda

var res = from re in dev.tbl\_Dept orderby (re.Did) select re;//query

#### Get all the records from tbl\_Dept which are sorted by Did descending

Hide   Copy Code

Select \* from tbl\_Dept order by Did desc

var res = from re in dev.tbl\_Dept orderby (re.Did) descendingselect re; //query

var res = dev.tbl\_Dept.OrderByDescending(x => x.Did).ToList(); //lambda

#### Get the record from tbl\_Dept with highest Did

Hide   Copy Code

Select top(1) \* from tbl\_Dept order by Did desc //T-Sql

var res = dev.tbl\_Dept.OrderByDescending(x => x.Did).Take(1); //lambda

var res = (from re in dev.tbl\_Dept orderby \_

(re.Did) descending select re).Take(1).ToList(); //query

#### Get all the records from tbl\_Dept which are sorted by DName and then by Did ascending

Hide   Copy Code

Select \* from tbl\_Dept order by DName, Did //T-Sql

var res =dev.tbl\_Dept.OrderBy(X => X.DName).ThenBy(X => X.Did); //lambda

var res = from re in dev.tbl\_Dept orderby (re.DName) orderby (re.Did) select re; //query

#### Get all the records from tbl\_Dept whose Did is less than or equal to 4

Hide   Copy Code

Select \* from tbl\_Dept Where Did <= 4

var res = dev.tbl\_Dept.Where(x => x.Did <= 4); //lambda

var res = from re in dev.tbl\_Dept where (re.Did <= 4) select re; //query

#### Get all the records from tbl\_Dept whose Did is either 4 or 7

Hide   Copy Code

Select \* from tbl\_Dept Where Did = 4 OR Did = 7

var res = dev.tbl\_Dept.Where(x => x.Did == 4 || x.Did == 7).ToList(); //lambda

var res = from re in dev.tbl\_Dept where (re.Did == 4 || re.Did == 7) select re; //query

#### Get all the records from tbl\_Dept whose Did is among 1, 5 and 6

Hide   Copy Code

select \* from tbl\_Dept Where Did IN (1, 5, 6) //T-Sql

var res = from re in dev.tbl\_Dept where (re.Did == 1 || \_

re.Did == 5||re.Did==6) select re; //query

var res = dev.tbl\_Dept.Where(x => x.Did == 1 || x.Did == 5||x.Did==6).ToList(); //lambda

#### Get all the records from tbl\_Dept whose Did is neither 3 nor 4

Hide   Copy Code

select \* from tbl\_Dept Where Did <> 3 and Did <> 4

var res = dev.tbl\_Dept.Where(x => x.Did != 3 && x.Did != 4).ToList(); //lambda

var res = from re in dev.tbl\_Dept where (re.Did != 3 && re.Did != 4) select re; //query

#### Get all the records from tbl\_Dept whose Did is not among 1, 5 and 6

Hide   Copy Code

select \* from tbl\_Dept Where Did NOT IN (1, 5, 6)

var res = from re in dev.tbl\_Dept \_

where (re.Did != 1 && re.Did != 5&& re.Did!=6) select re; //query

var res = dev.tbl\_Dept.Where\_

(x => x.Did != 1 && x.Did != 5 && x.Did!=6).ToList(); //lambda

#### Get all the records from tbl\_Dept whose Did is greater than or equal to 2 and less than or equal to 4

Hide   Copy Code

select \* from tbl\_Dept Where Did >= 2 and Did <= 4

var res = dev.tbl\_Dept.Where(x => x.Did>=2 && x.Did<=4).ToList(); //lambda

var res = from re in dev.tbl\_Dept where (re.Did >= 2 && re.Did <=4) select re; //query

#### Get all the records from tbl\_Dept whose Did is between 2 and 4

Hide   Copy Code

select \* from tbl\_Dept Where Did between 2 and 4

var res = from re in dev.tbl\_Dept where (re.Did > 2 && re.Did <4) select re;//query

var res = dev.tbl\_Dept.Where(x => x.Did>2 && x.Did<4).ToList();