**Value lab interview questions**

1. **What is the named parameters in c#?**

Introduction

C# 4.0 introduced two new concepts, Named parameters and Optional parameters. In this article and code examples, let's see how we can use and take advantages of named parameters adn optional parameters in C# and .NET.

Named Parameter

Named Parameters allow developers to pass a method arguments with parameter names. Prior to these this feature, the method parameters were passed using a sequence only. Now, using named parameters in C#, we can put any parameter in any sequence as long as the name is there. The right parameber value based on their names will be mapped to the right variable. The parameters name must match with the method definition parameter names.

The syntax of parameter names is (parametername: value).

In the following code example, the method AddNumber(int firstNumber, int secondNumber) takes two parameters and returns a sum of the two numbers. Here, firstNumber and secondNumber are two parameters that are passed in AddNumber() method using named parameters. If it doesn’t match the method definition parameter name and Named Parameter Name then it throws compilation error “*The best overload for 'AddNumber' does not have a parameter named 'secondNumber1*'”.

The following method definition I will also use in my following examples.

1. **public** **static** **int** AddNumber(**int** firstNumber, **int** secondNumber)
2. {
3. **return** firstNumber+ secondNumber;
4. }

To call the above method, we use the following method call. As you can see from this below code snippet, the name of the parameter is used with a ":" followed by the value.

1. AddNumber(firstNumber: 123, secondNumber: 64);

Here are some cases that show how to pass parameters in a method.

**Case 1**: Using Normal Way

1. AddNumber(12, 13);

Thi is the way we usually call methods. In this case, the value of the parameters are passed to the arguments in same order. In this case, the order of the parameters in the call must be exactly same as in the method definition. It means in above example, firstNumber value will be 12 and secondNumber value will be 13.

**Case 2**: Using Named Parameters

1. AddNumber(firstNumber: 12, secondNumber: 13);

It is another way to pass parameters to a method. It is not mandatory to keep the parameter sequence because we are using named parameters. Regardless of the order of the parameters, the value of the agument will be matched based on their names. Here we are passing firstNumber as 12 and secondNumber as 13. We can write same code like:

1. AddNumber(firstNumber: 12, secondNumber: 13);

A parameter can be positional parameter as shown below:

1. AddNumber(12, secondNumber: 13);

Here we pass 12 as first parameter and 13 as second parameter.

But positional parameters doesn’t follow Named Parameters. It throws compilation error “*Named argument specifications must appear after all fixed arguments have been specified*”.

1. AddNumber(firstNumber: 12, 13); // It will throw compilation error.

Here is the complete C# code shows how to use named parameters.

1. **static** **void** Main(**string**[] args)
2. {
3. // The method can be called in the normal way, by using positional arguments.
4. Console.WriteLine(AddNumber(12, 13));
6. // Named arguments can be supplied for the parameters in either order.
7. Console.WriteLine(AddNumber(firstNumber: 12, secondNumber: 13));
8. Console.WriteLine(AddNumber(secondNumber: 13, firstNumber: 12));
10. // Positional arguments cannot follow named arguments. It throws compilation error.
11. Console.WriteLine(AddNumber(firstNumber: 12, 13);
13. // Named arguments can follow positional arguments.
14. Console.WriteLine(AddNumber(12, secondNumber: 13));
15. }

**2.what is threading in c# and concepts?**

# Threads and threading

Multithreading allows you to increase the responsiveness of your application and, if your application runs on a multiprocessor or multi-core system, increase its throughput.

## Processes and threads

A process is an executing program. An operating system uses processes to separate the applications that are being executed. A thread is the basic unit to which an operating system allocates processor time. Each thread has a [scheduling priority](https://docs.microsoft.com/en-us/dotnet/standard/threading/scheduling-threads) and maintains a set of structures the system uses to save the thread context when the thread's execution is paused. The thread context includes all the information the thread needs to seamlessly resume execution, including the thread's set of CPU registers and stack. Multiple threads can run in the context of a process. All threads of a process share its virtual address space. A thread can execute any part of the program code, including parts currently being executed by another thread.

1. **Foreground Thread**
2. **Background Thread**

#### Foreground Thread

A thread which keeps on running to complete its work even if the Main thread leaves its process, this type of thread is known as foreground thread. Foreground thread does not care whether the main thread is alive or not, it completes only when it finishes its assigned work. Or in other words, the life of the foreground thread does not depend upon the main thread.

**Example:**

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // C# program to illustrate the  // concept of foreground thread  using System;  using System.Threading;    class GFG {        // Main method      static void Main(string[] args)      {            // Creating and initializing thread          Thread thr = new Thread(mythread);          thr.Start();          Console.WriteLine("Main Thread Ends!!");      }        // Static method      static void mythread()      {          for (int c = 0; c <= 3; c++) {                Console.WriteLine("mythread is in progress!!");              Thread.Sleep(1000);          }          Console.WriteLine("mythread ends!!");      }  } |

**Output:**

Main Thread Ends!!

mythread is in progress!!

mythread is in progress!!

mythread is in progress!!

mythread is in progress!!

mythread ends!!

**Explanation:** In the above example, thr thread runs after main thread ended. So, the life of thr tread doesn’t depend upon the life of the main thread. The thr thread only ends its process when it completes its assigned task.

#### Background Thread

A thread which leaves its process when the Main method leaves its process, these types of the thread are known as the background threads. Or in other words, the life of the background thread depends upon the life of the main thread. If the main thread finishes its process, then background thread also ends its process.

**Note:** If you want to use a background thread in your program, then set the value of IsBackground property of the thread to true.

**Example:**

filter\_none

edit

play\_arrow

brightness\_4

|  |
| --- |
| // C# program to illustrate the  // concept of Background thread  using System;  using System.Threading;    class GFG {        // Main method      static void Main(string[] args)      {          // Creating and initializing thread          Thread thr = new Thread(mythread);            // Name of the thread is Mythread          thr.Name = "Mythread";          thr.Start();            // IsBackground is the property of Thread          // which allows thread to run in the background          thr.IsBackground = true;            Console.WriteLine("Main Thread Ends!!");      }        // Static method      static void mythread()      {            // Display the name of the          // current working thread          Console.WriteLine("In progress thread is: {0}",                              Thread.CurrentThread.Name);            Thread.Sleep(2000);            Console.WriteLine("Completed thread is: {0}",                            Thread.CurrentThread.Name);      }  } |

**Output:**

In progress thread is: Mythread

Main Thread Ends!!

**Explanation:** In the above example, IsBackground property of Thread class is used to set the thr thread as a background thread by making the value of IsBackground true. If you set the value of IsBackground false, then the given thread behaves as a foreground Thread. Now, the process of the thr thread ends when the process of the main thread ends.

**What is Lock?**

Lock is another synchronization mechanism in C# and one of the famous multi-threading interview questions in .NET. It restricts the critical region so that only one thread can enter a critical region at a time.

Locks needs an object to continue its operation. It apply a lock on a target object and only one thread can lock that target object at a time. Below is the example of lock object.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | static object obj = new object();    public static void DoWork()  {      lock(obj)      {          Console.WriteLine("Multi threading Interview questions in C#");      }  } |

**What is the Difference between Lock and Monitor?**

Lock is just a shortcut for Monitor statement. Compiler internally convert lock statement to Monitor.Enter and Exit statements.

Monitor class provides some useful method which are not in lock statement. These methods are very useful in advanced scenarios.

1. Monitor provides TryEnter method. This method is useful when we need to provide timeout value.
2. TryEnter is also useful when we have to check whether lock is taken or not. We can pass a boolean parameter which returns true if lock is taken else returns false.
3. Pulse method notifies a waiting thread of a change in the locked object's state.
4. Wait method release the current acquired lock and block the current thread until it reacquires the lock.

**3.diference b/w the Post and put?**

# REST – PUT vs POST

It has been observed that many people struggle to choose between **HTTP PUT vs POST** methods when designing a system. Though, [RFC 2616](https://www.ietf.org/rfc/rfc2616.txt) has been very clear in differentiating between the two – yet complex wordings are a source of confusion for many of us. Let’s try to solve the puzzle **when to use PUT or POST**.

Let’s compare them for better understanding.

|  |  |
| --- | --- |
| **PUT** | **POST** |
| RFC-2616 clearly mention that PUT method requests for the enclosed entity be stored under the supplied [Request-URI](https://restfulapi.net/resource-naming/). If the Request-URI refers to an already existing resource – an update operation will happen, otherwise create operation should happen if Request-URI is a valid resource URI (assuming client is allowed to determine resource identifier).  PUT /questions/{question-id} | The POST method is used to request that the origin server accept the entity enclosed in the request as a new subordinate of the resource identified by the Request-URI in the Request-Line. It essentially means that POSTrequest-URI should be of a collection URI.  POST /questions |
| PUT method is [idempotent](https://restfulapi.net/idempotent-rest-apis/). So if you send retry a request multiple times, that should be equivalent to single request modification. | POST is NOT idempotent. So if you retry the request N times, you will end up having N resources with N different URIs created on server. |
| Use PUT when you want to modify a singular resource which is already a part of resources collection. PUT replaces the resource in its entirety. Use PATCH if request updates part of the resource. | Use POST when you want to add a child resource under resources collection. |
| Though PUT is idempotent, we shall not cache it’s response. | Responses to this method are not [cacheable](https://restfulapi.net/caching/), unless the response includes appropriate Cache-Control or Expires header fields. However, the 303 (See Other) response can be used to direct the user agent to retrieve a cacheable resource. |
| Generally, in practice, always use PUT for UPDATE operations. | Always use POST for CREATE operations. |

## PUT vs POST : An Example

Let’s say we are designing a network application. Let’s list down few URIs and their purpose to get better understanding when to use POST and when to use PUT operations.

GET /device-management/devices : Get all devices

**POST** /device-management/devices : Create a new device

GET /device-management/devices/{id} : Get the device information identified by "id"

**PUT** /device-management/devices/{id} : Update the device information identified by "id"

DELETE /device-management/devices/{id} : Delete device by "id"

**4.difference b/w angular and angularjs?**

First of all, **Angular** is based on TypeScript while **AngularJS** is based on JavaScript. TypeScript is a superset of ES6 and it's backward compatible **with** ES5. ...**AngularJS** uses terms of scope and controller. To scope a variable you can add many variables that will be visible in View as well as in Controller.

How Are They Different?

Below is a comparison of AngularJS to Angular, because Angular includes both version 2 and version 4. We compare architecture, language, expression syntax, mobile support, and routing.

1. Architecture

AngularJS

The architecture of AngularJS is based on model-view-controller (MVC) design. The model is the central component that expresses the application's behavior and manages its data, logic, and rules. The *view* generates an output based on the information in the *model*. The *controller* accepts input, converts it into commands and sends the commands to the *model* and the *view*.

Angular

In Angular 2, controllers and $scope were replaced by components and directives. Components are directives with a template. They deal with a view of the application and logic on the page. There are two kinds of directives in Angular 2. These are structural directives that alter the layout of the DOM by removing and replacing its elements, and attributive directives that change the behavior or appearance of a DOM element.

In Angular 4, the structural derivatives ngIf and ngFor have been improved, and you can use if/else design syntax in your templates.

2. Language

AngularJS

AngularJS is written in JavaScript.

Angular

Angular uses Microsoft’s TypeScript language, which is a superset of ECMAScript 6 (ES6). This has the combined advantages of the TypeScript features, like type declarations, and the benefits of ES6, like iterators and lambdas.

Angular 4 is compatible with the most recent versions of TypeScript that have powerful type checking and object-oriented features.

3. Expression Syntax

AngularJS

To bind an image/property or an event with AngularJS, you have to remember the right ng directive.

Angular

Angular focuses on “( )” for event binding and “[ ]” for property binding.

4. Mobile Support

AngularJS was not built with mobile support in mind, but Angular 2 and 4 both feature mobile support.

5. Routing

AngularJS uses $routeprovider.when() to configure routing while Angular uses @RouteConfig{(…)}.

Performance

AngularJS was originally developed for designers, not developers. Although there were a few evolutionary improvements in its design, they were not enough to fulfill developer requirements. The later versions, Angular 2 and Angular 4, have been upgraded to provide an overall improvement in performance, especially in speed and dependency injection.

1. Speed

By providing features like 2-way binding, AngularJS reduced the development effort and time. However, by creating more processing on the client side, page load was taking considerable time. Angular2 provides a better structure to more easily create and maintain big applications and a better change detection mechanism. Angular 4 is the fastest version yet.

2. Dependency injection

Angular implements unidirectional tree-based change detection and uses Hierarchical Dependency Injection system. This significantly boosts performance for the framework.

Advantages and Disadvantages

Because they are Google products, all Angular versions are trustworthy and enjoy great support from Google engineers and the large community of Angular users and developers. However, each version has its own advantages and disadvantages.

1. AngularJS

Advantages

* It is unit testing ready.
* It has great MVC data binding makes app development fast.
* Using HTML as a declarative language makes it very intuitive.
* It is a comprehensive solution for rapid front-end development since it does not need any other frameworks or plugins.
* AngularJS apps can run on every significant program and advanced cells including iOS and Android-based phones and tablets.

Disadvantages

* It is big and complicated due to the multiple ways of doing the same thing.
* Implementations scale poorly.
* If a user of an AngularJS application disables JavaScript, nothing but the basic page is visible.
* There’s a lagging UI if there are more than 200 watchers.

2. Angular 2

Advantages

* TypeScript allows code optimization using the OOPS concept.
* It is mobile-oriented.
* It has improved dependency injection and modularity.
* It provides more choice for languages such as Dart, TypeScript, ES5, and ES6 for writing codes.
* It offers simpler routing.

Disadvantages

* It is more complicated to set up compared to AngularJS.
* It’s inefficient if you only need to create simple, small web apps.

3. Angular 4

Advantages

* It enables a fast development process.
* It’s ideal for single-page web applications with an extended interface.
* Full TypeScript support helps in building bulky applications.
* Tests are easy to write.
* An improved View Engine generates less code in AOT mode.
* It has a modularized animation package.

Disadvantages

* It’s slow when displaying enormous amounts of data.

To know more about Angular 4 read this [article](https://www.simplilearn.com/what-is-angular-4-and-how-to-master-article).

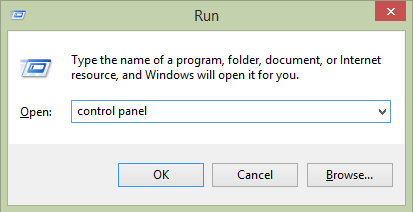
**5.what is msmq in wcf**?

MSMQ is Microsoft Message Queuing developed by Microsoft and deployed in windows operating system. MSMQ Queue ensures that reliable messaging between a client and a [Windows Communication Foundation (WCF)](http://www.c-sharpcorner.com/technologies/wcf) service. So first we need to enable this feature in our operating system.

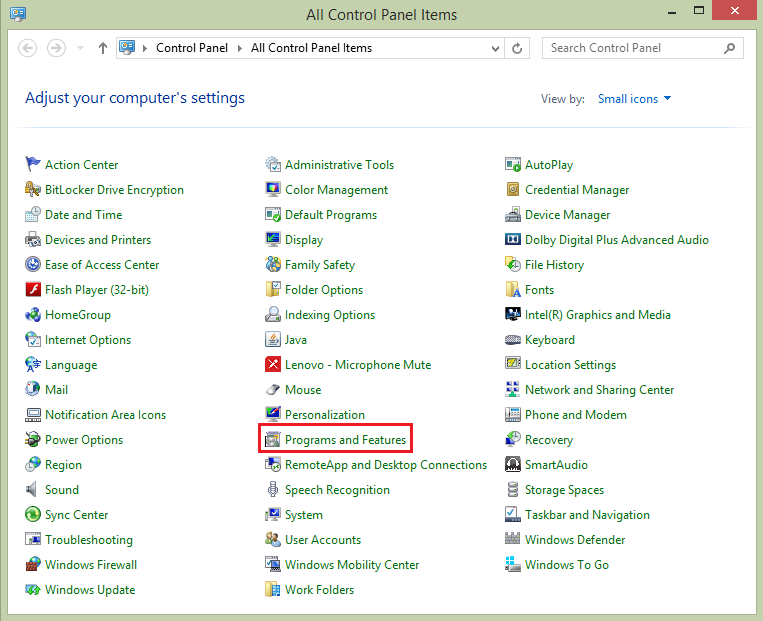
**How to enable MSMQ**

Press Window key + R, it will open Run window.

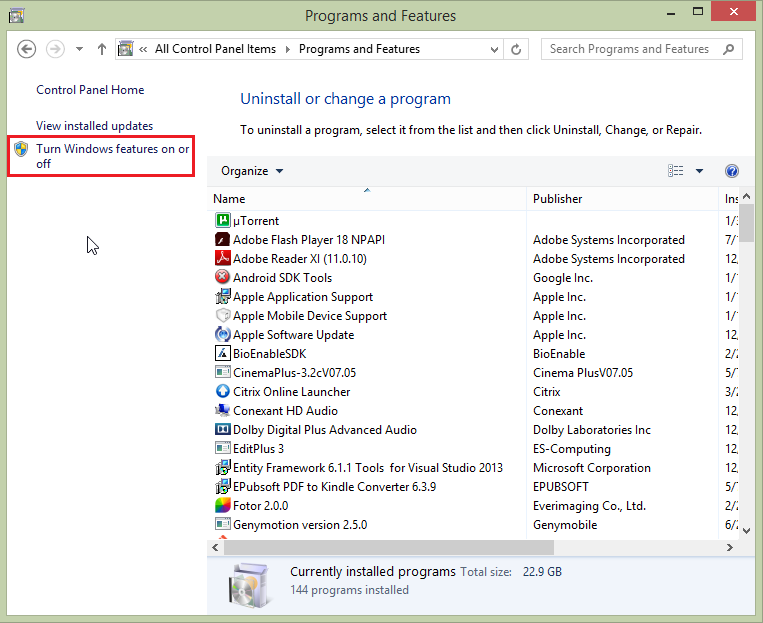
Enter ‘Control Panel’ and press ‘OK’ button, it will open control panel,



Click on ‘Programs and Features’,

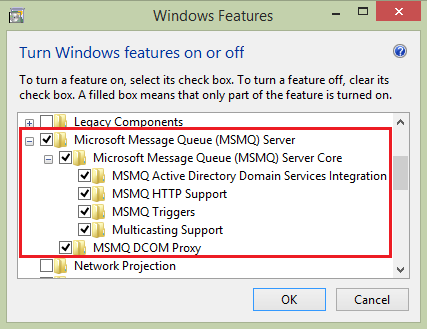


From the left panel click on ‘Turn Windows features on or off’,



Select ‘Microsoft Message Queue (MSMQ) Server’ and all its child options.

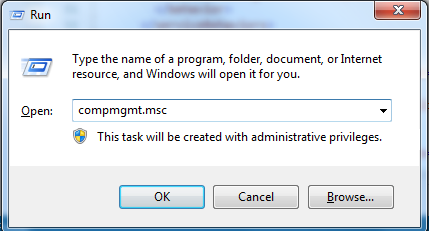
Press ‘OK’ button, it will enable MSMQ for your operating system,



**How to create a queue?**

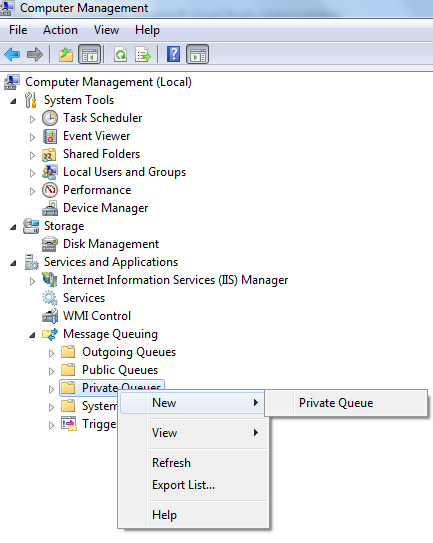
Press Window key + R, it will open Run window.

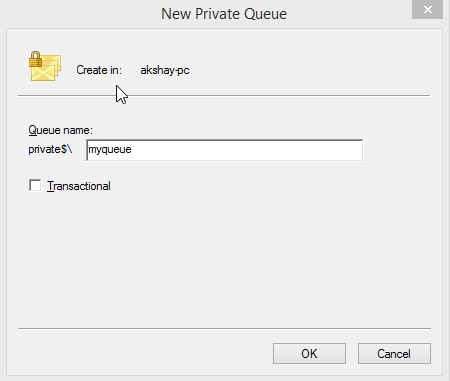
Enter ‘compmgmt.msc’ and press ‘OK’ button, it will open Computer Management window.



Under ‘Services and Applications’ expand ‘Message Queuing’.

Right click on ‘Private Queueus’, select ‘New’ and click on ‘Private Queue’.

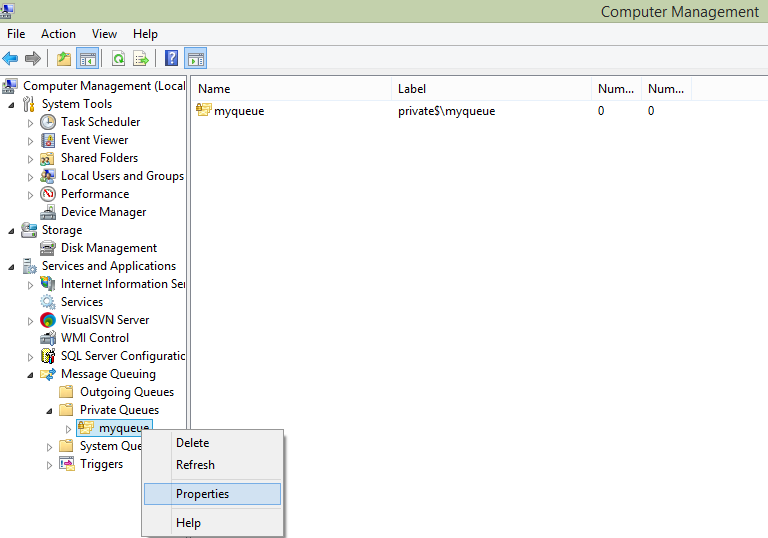
  
Enter queue name, here I have entered ‘myqueue’, which we will use in our application later.



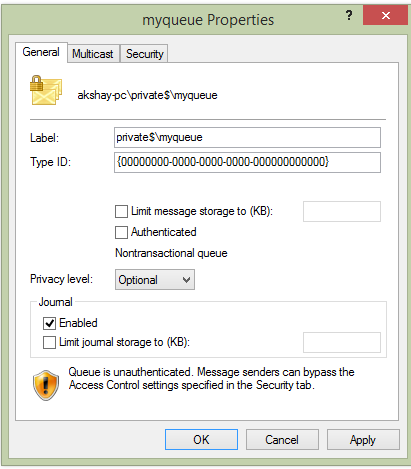
You can find preceding queue under Private Queues.

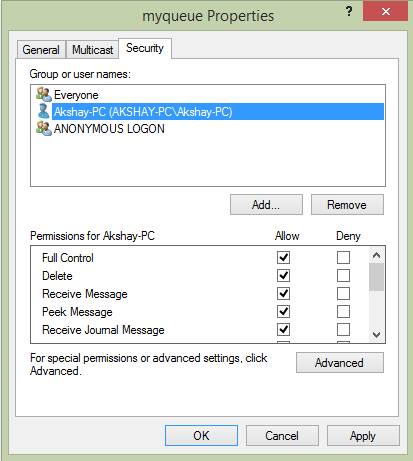
Select ‘Private Queues’ and you will have list of all queues with details like Name, Label, Number of Messages and Number of Journal Messages.

Right click on ‘myqueue’ and select ‘Properties’,



Check ‘Journal Enabled’ checkbox,

  
  
Select the user and allow full permission for this queue,

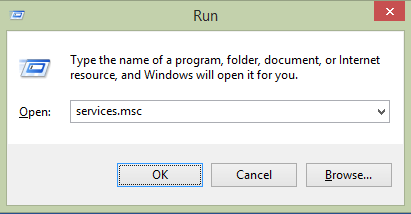


**Verify Queue Service is started**

After creating and setting permissions for the queue, we need to verify if queue service is running or not.

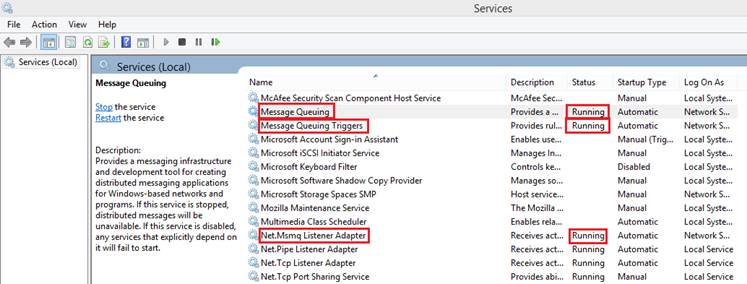
Press Window key + R, it will open Run window,

Enter ‘services.msc’ and press ‘OK’ button, it will open Services window.



Change below services status to ‘Running’,

* Message Queuing
* Message Queuing Triggers
* Net.Msmq Listener Adapter



Now let us create Service Application

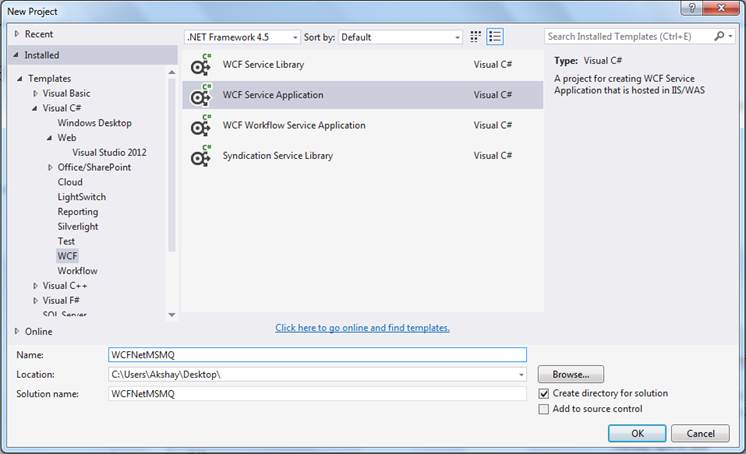
**Create WCF Service Application**

Open Visual Studio Editor,

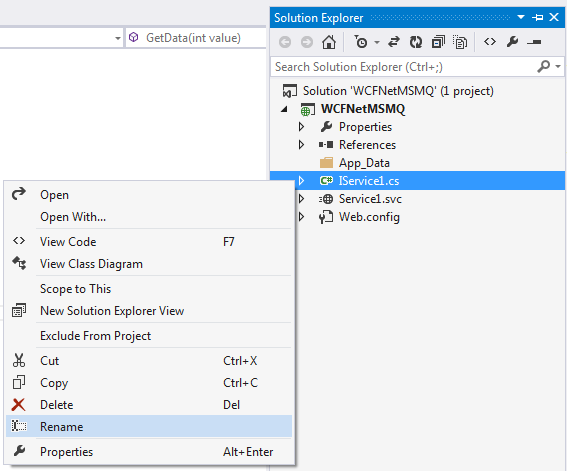
Select ‘File’ menu, expand ‘New’ and click on ‘Projects’,

Select ‘WCF’ from left panel, select ‘WCF Service Application’,

Provide appropriate name, select the location and press ‘OK’ button,



Right click on interfact IService1 and rename it to IQueueService,



Right click on Service1.svc and rename it to ‘QueueService.svc’,

Right click on ‘QueueService.svc’ and click on ‘View Markup’,

Change the name of service to ‘WCFNetMSMQ.QueueService’,

1. <%@ ServiceHost Language="C#" Debug="true" Service="WCFNetMSMQ.QueueService" CodeBehind="QueueService.svc.cs" %>

Configure service and binding in the web.config under Service Model section. Under service we need to set endpoint with address, binding, contract and binding configuration,

* Address – set queue address with net.msmq, use queuename which we have created previously in the last.
* Binding – binding name which we will configure under bindings section.
* Contract – Provide the interface name

1. <services>
2. <service name="WCFNetMSMQ.QueueService">
3. <endpoint address="net.msmq://localhost/private/myqueue"
4. binding="netMsmqBinding"
5. bindingConfiguration="MsmqBindingConfig"
6. contract="WCFNetMSMQ.IQueueService" />
7. </service>
8. </services>

Configure binding under netMsmqBinding section. Set security as none.

1. <bindings>
2. <netMsmqBinding>
3. <binding name="MsmqBindingConfig" exactlyOnce="false">
4. <security mode="None" />
5. </binding>
6. </netMsmqBinding>
7. </bindings>

In the interface, we have operation contract namely ‘GetData’, Set ‘IsOneWay’ attribute as true and make return type as void.

1. **namespace** WCFNetMSMQ
2. {
3. [ServiceContract]
4. **public** **interface** IQueueService
5. {
6. [OperationContract(IsOneWay=**true**)]
7. **void** GetData(**int** value);
8. }
9. }

At the implementation part, trace the value which you are going to pass as parameter in the operation contract.

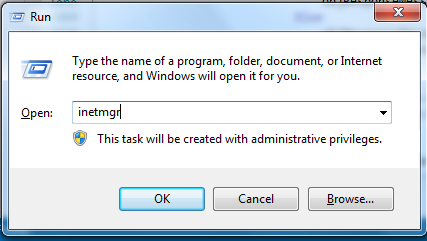
1. **namespace** WCFNetMSMQ
2. {
3. **public** **class** QueueService : IQueueService
4. {
5. **public** **void** GetData(**int** value)
6. {
7. System.Diagnostics.Trace.Write(**string**.Format("You entered: {0}", value));
8. }
9. }
10. }

Now we are done with the service application having operation contract which accepts integer number and log it. In the next step we need to host our application in IIS.

**Host WCF Service Application in IIS**

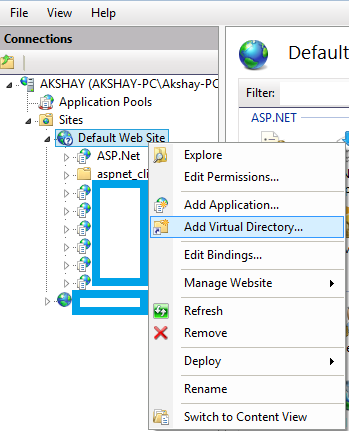
Press Window key + R, it will open Run window.

Enter ‘inetmgr’ and press ‘OK’ button, it will open IIS window.



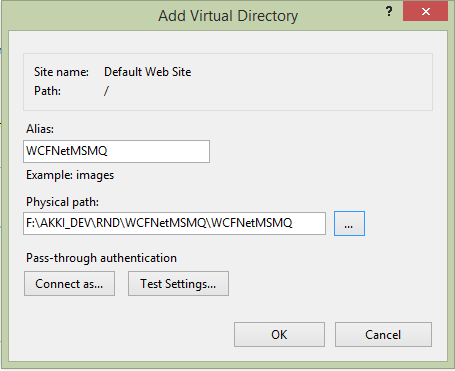
Expand the ‘Sites’ and you will find ‘Default Web Site’.

Right click on ‘Default Web Site’ and click on ‘Add Virtual Directory…’ in order to host the application,

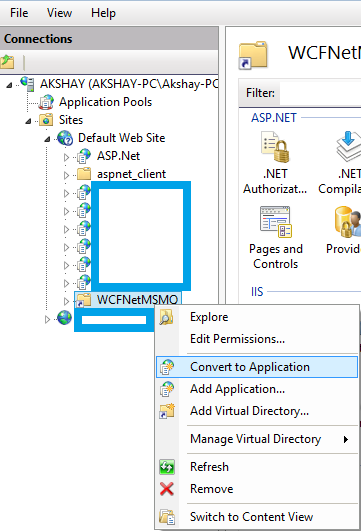


Provide alias name and select our application physical path.

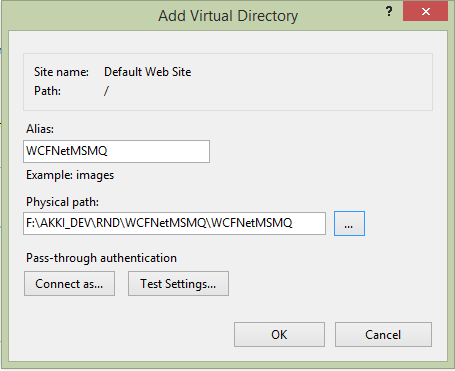
Press ‘OK’ button,



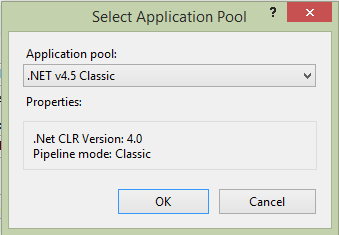
Right click on ‘WCFNetMSMQ’ and click on ‘Convert to Application’.



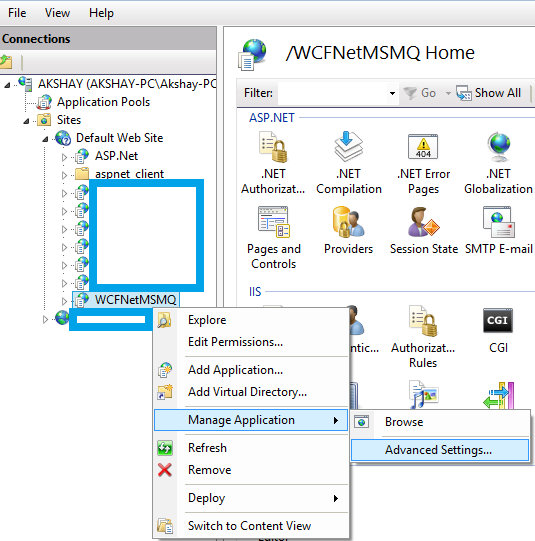
Click on ‘Select…’ button and select the application pool.



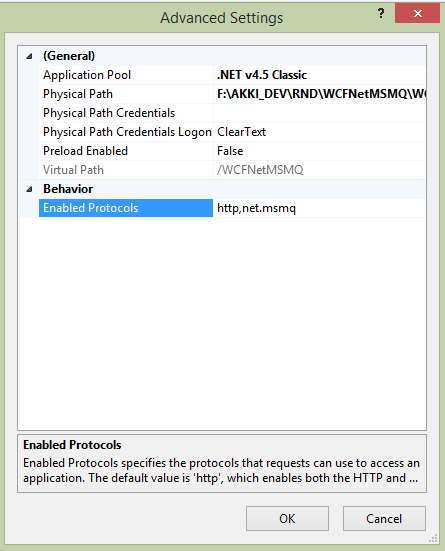
Select ‘.NET v4.5 Classic’ application pool and press ‘OK’ button.



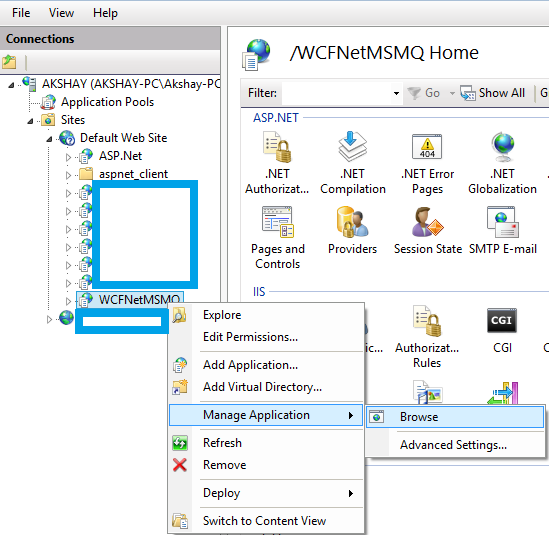
In order to enable net.msmq protocol for this application, Right click on ‘WCFNetMSMQ’, expand ‘Manage Application’ and click on ‘Advanced Settings…’



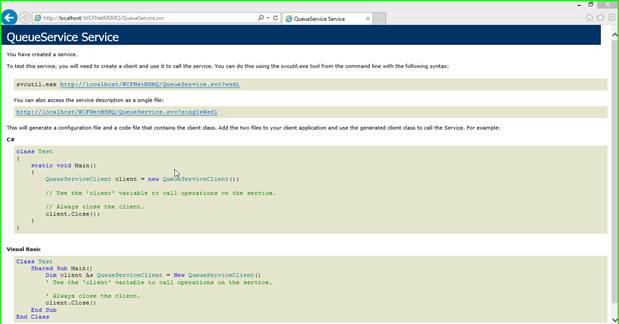
In Enabled Protocols add net.msmq along with http, It will enable net.msmq protocol for our application.



Now check whether configured site is working or not, Right click on ‘WCFNetMSMQ’, expand ‘Manage Application’ and click on ‘Browse’.

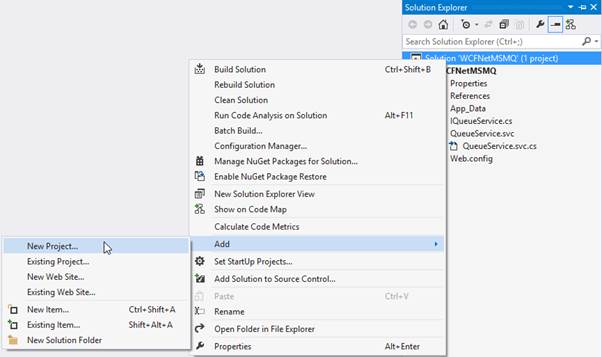


In the browser we can see that service is running fine.



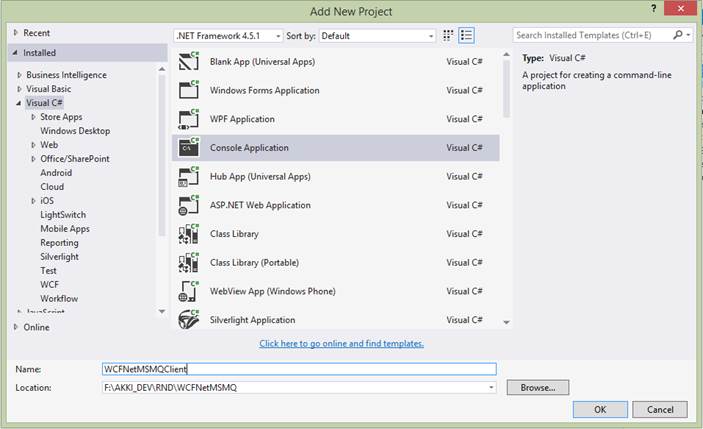
Now it’s time to create client application to consume service.

**Create Client Application**  
Right click on Solution, expand ‘Add’ and click on ‘New Project…’.

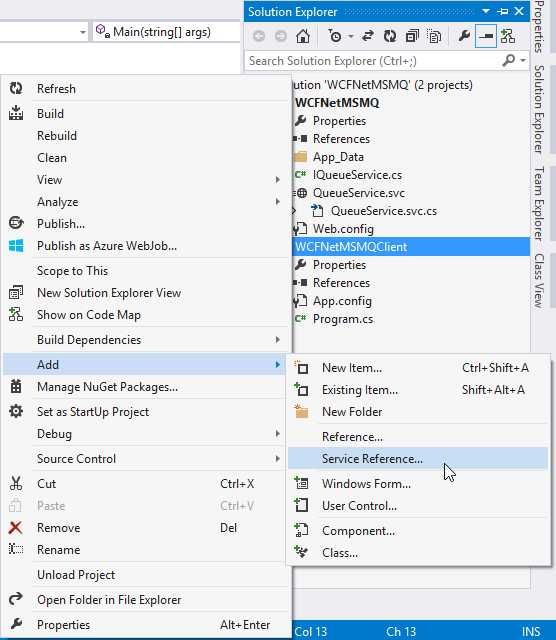


Select ‘Visual C#’ from the left panel and select ‘Console Application’.

Provide appropriate name, select location and click on ‘OK’ button.



Add service reference in our client application, Right click on the project, expand ‘Add’ and click on ‘Service Reference…’,

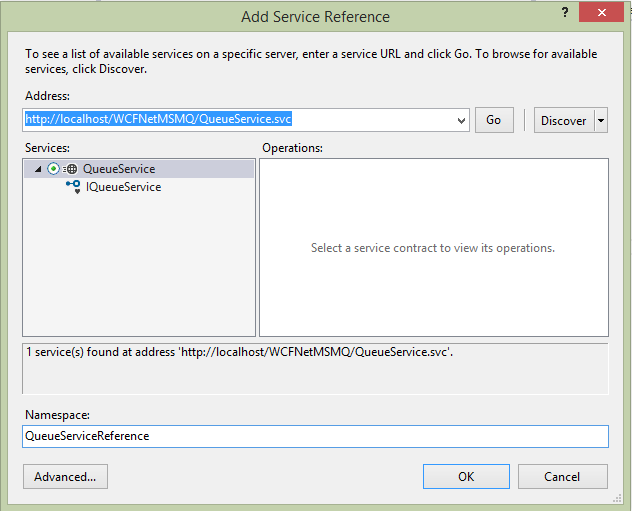


Browse application from IIs and copy address from the address bar of the browser.

Paste same address in the ‘Add Service Reference’ window and press ‘Go’ Button.

Select ‘Queue Service’ from Services list.

Give the namespace and press ‘OK’ button.

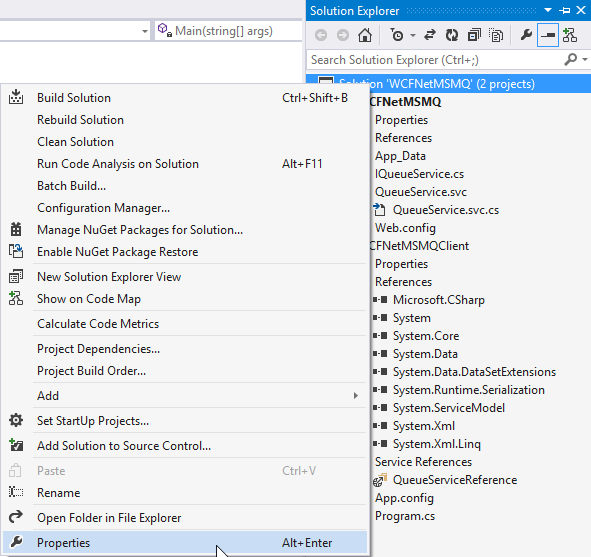


Create an instance of QueueService client.

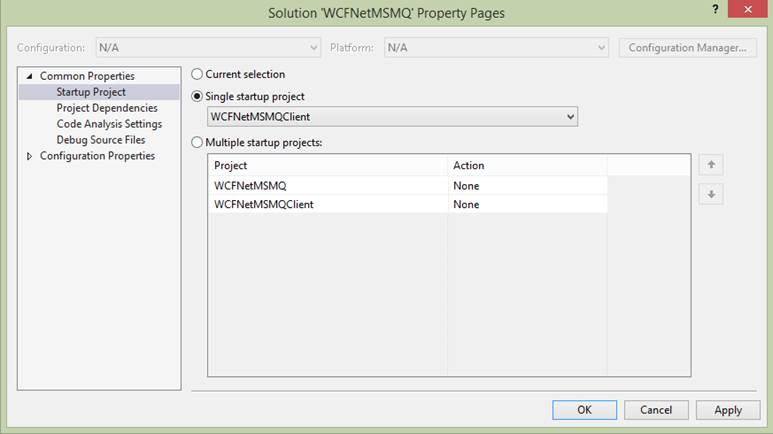
Call GetData operation contract using this instance,

1. **namespace** WCFNetMSMQClient
2. {
3. **class** Program
4. {
5. **static** **void** Main(**string**[] args)
6. {
7. QueueServiceReference.QueueServiceClient serviceClient = **new** QueueServiceReference.QueueServiceClient();
8. serviceClient.GetData(5677);
9. }
10. }
11. }

In order to set client application as startup project, right click on solution and click on ‘Properties’,

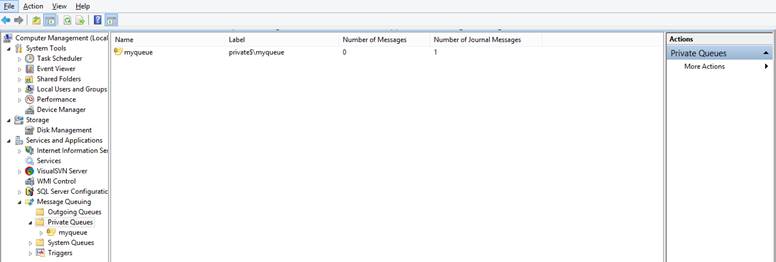


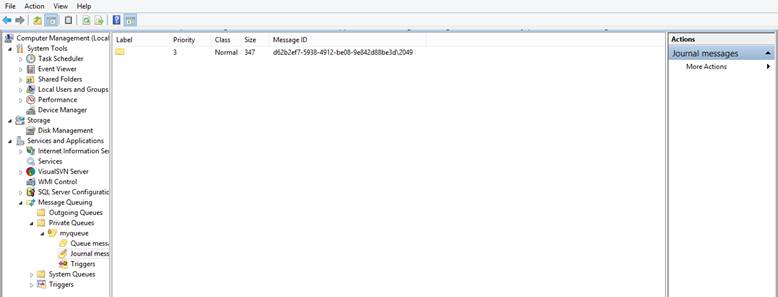
Select client application under ‘Single Startup Project’,



Now run the project and check ‘myqueue’ queue.

You can see that one message is added in Journal Messages.



Select Journal Messages under myqueue and you can see the message details like label, priority, class, size and message id.  
  


The main benefit of using queue mechanism is if the destination server is offline when the client sends a message, the message will be queued on the client until the server comes back online.

Read more articles on **WCF**:

* [Data Contract And Known Type In WCF](http://www.c-sharpcorner.com/UploadFile/c1de77/data-contract-and-known-type-in-wcf/)
* [Create Secure WCF REST API With Custom Basic Authentication](http://www.c-sharpcorner.com/UploadFile/vendettamit/create-secure-wcf-rest-api-with-custom-basic-authentication/)

**6.Why linq?**

Advantages of **LINQ**

Standardized way of querying multiple data sources: The same **LINQ** syntax can be used to query multiple data sources. Compile time safety of queries: It provides type checking of objects at compile time. IntelliSense Support: **LINQ** provides IntelliSense for generic collections.

**7.what are the advantages of entity framework over ado.net?**

          The ADO.NET Entity Framework enables developers to create data access applications by programming against a conceptual application model instead of programming directly against a relational storage schema. The goal is to decrease the amount of code and maintenance required for data-oriented applications. Entity Framework applications provide the following benefits:

1. Applications can work in terms of a more application-centric conceptual model, including types with inheritance, complex members, and relationships.
2. Applications are freed from hard-coded dependencies on a particular data engine or storage schema.
3. Mappings between the conceptual model and the storage-specific schema can change without changing the application code.
4. Developers can work with a consistent application object model that can be mapped to various storage schemas, possibly implemented in different database management systems.
5. Multiple conceptual models can be mapped to a single storage schema.
6. Language-integrated query (LINQ) support provides compile-time syntax validation for queries against a conceptual model.

Regards

**8.Performance tuning in entity framework ?**

**9.What is grep in jquery?**

**jQuery** $.**grep**() Method

The **jQuery** method of $.**grep**() is used to filter the contents of an array. It removes items from an array such that all remaning items can pass a provided test.The test is a function that is passed out as an array item and the index of the item within the array as parameters.

**Categories**

# 25 tips to Improve SQL Query Performance

September 7, 2015

33856

2

Every Customer/ User always wants a fast response on their data retrieval process. So we need to design a good database that provides best performance during data manipulation which results into the best performance of an application. However, there is no straightforward way to define the best performance but we can choose multiple ways to improving SQL query performance, which falls under various categories like creation of Indexes, usage of joins, and rewrite a subquery to use JOIN, etc.

As a developer, we know any SQL query can be written in multiple ways but we should follow best practices/ techniques to achieve better query performance. I’m highlighting some of them below:

**10.Performance tuning in sql?**

1. Use EXISTS instead of IN to check existence of data.  
2. Avoid \* in SELECT statement. Give the name of columns which you require.  
3. Choose appropriate Data Type. E.g. To store strings use varchar in place of text data type. Use text data type, whenever you need to store large data (more than 8000 characters).  
4. Avoid nchar and nvarchar if possible since both the data types takes just double memory as char and varchar.  
5. Avoid NULL in fixed-length field. In case of requirement of NULL, use variable-length (varchar) field that takes less space for NULL.  
6. Avoid Having Clause. Having clause is required if you further wish to filter the result of an aggregations.  
7. Create Clustered and Non-Clustered Indexes.  
8. Keep clustered index small since the fields used in clustered index may also used in non-clustered index.  
9. Most selective columns should be placed leftmost in the key of a non-clustered index.  
10. Drop unused Indexes.  
11. Better to create indexes on columns that have integer values instead of characters. Integer values use less overhead than character values.  
12. Use joins instead of sub-queries.  
13. Use WHERE expressions to limit the size of result tables that are created with joins.  
14. Use TABLOCKX while inserting into a table and TABLOCK while merging.  
15. Use WITH (NOLOCK) while querying the data from any table.  
16. Use SET NOCOUNT ON and use TRY- CATCH to avoid deadlock condition.  
17. Avoid Cursors since cursor are very slow in performance.  
18. Use Table variable in place of Temp table. Use of Temp tables required interaction with TempDb database which is a time taking task.  
19. Use UNION ALL in place of UNION if possible.  
20. Use Schema name before SQL objects name.  
21. Use Stored Procedure for frequently used data and more complex queries.  
22. Keep transaction as small as possible since transaction lock the processing tables data and may results into deadlocks.  
23. Avoid prefix “sp\_” with user defined stored procedure name because SQL server first search the user defined procedure in the master database and after that in the current session database.  
24. Avoid use of Non-correlated Scalar Sub Query. Use this query as a separate query instead of part of the main query and store the output in a variable, which can be referred to in the main query or later part of the batch.  
25. Avoid Multi-statement Table Valued Functions (TVFs). Multi-statement TVFs are more costly than inline TVFs.

<https://www.toptal.com/sql-server/sql-database-tuning-for-developers>

**Angular 4 features?**

**What Extension methods?**

**Angular view state?**

**Angular view encapsulation?**

**How can u write one component dependent on other?**

**IMI Mobile Interview questions**

**1.what are the solid principles?  
2.WRITE Array sort program?**

class Program

{

static void Main(string[] args)

{

// declaring and initializing the array

int[] arr = new int[] { 1, 9, 6, 7, 5, 9 };

int temp;

// traverse 0 to array length

for (int i = 0; i < arr.Length - 1; i++)

// traverse i+1 to array length

for (int j = i + 1; j < arr.Length; j++)

// compare array element with

// all next element

if (arr[i] > arr[j])

{

temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

// print all element of array

foreach (int value in arr)

{

Console.Write(value + " ");

}

Console.Read();

}

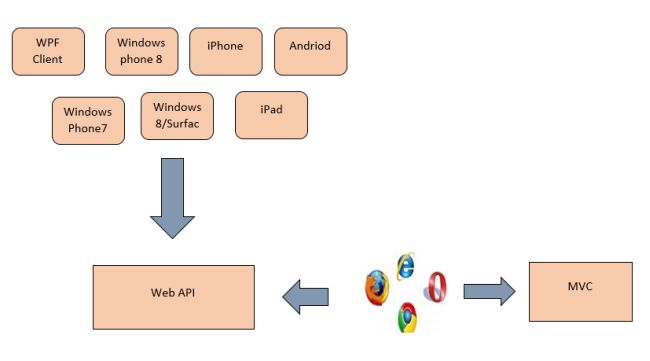
}  
**3.difference b/w mvc and web api?**

There are many differences between MVC and Web API, including:

* We can use the MVC for developing  the Web application that replies as both data and views but the Web API is used for generating the HTTP services that replies only as data.
* In the Web API the request performs tracing with the actions depending on the HTTP services but the MVC request performs tracing with the action name.
* The Web API returns the data in various formats, such as JSON, XML and other format based on the accept header of the request. But the MVC returns the data in the JSON format by using JSONResult.
* The Web API supports content negotiation, self hosting. All these are not supported by the MVC.
* The Web API includes the various features of the MVC, such as routing, model binding but these features are different and are defined in the "System.Web.Http" assembly. And the MVC features are defined in the " System.Web.Mvc" assembly.
* The Web API helps the creation of RESTful services over the .Net Framework but the MVC does not support.

**When we combined the MVC with Web API:**

* When we do the self hosting on the application, in it we combine both the MVC controller and the API in a single project and it helps for managing the AJAX requests and returns the response in XML, JSON and other Formats.
* We combined the MVC and Web API for enabling the authorization for an application. In it we create two filters, one for the Web API and another for MVC.



This figure describes the combination of ASP.NET MVC and ASP.NET Web API.

**4.how to provide security in mvc?**

[Introduction](https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications#Introduction)

[1) Security Misconfiguration (Error Handling Must Setup Custom Error Page)](https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications#1)_Security_Misconfiguration_(Error_Handling_Must_Setup_Custom_Error_Page))

[2) Cross-Site Request Forgery (CSRF)](https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications#2)_Cross-Site_Request_Forgery_(CSRF))

[3) Cross-Site Scripting (XSS) attacks](https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications#3)_Cross-Site_Scripting_(XSS)_attacks)

[4) Malicious File Upload.](https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications#4)_Malicious_File_Upload.)

[5) Version Discloser](https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications#5)_Version_Discloser)

[6) SQL Injection Attack.](https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications#6)_SQL_Injection_Attack.)

[7) Sensitive Data Exposure](https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications#7)_Sensitive_Data_Exposure)

[8) Audit trail](https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications#8)_Audit_trail)

[9) Broken authentication and session management](https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications#9)_Broken_authentication_and_session_management)

[10) Unvalidated Redirects and Forwards](https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications#10)_Unvalidated_Redirects_and_Forwards)

<https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications>

**what are the filters in mvc**

## Introduction

In this article we will try to see how we can use custom filters and attributes in an ASP.NET MVC application. Custom filters and attributes are an excellent way of injecting extra processing logic into the MVC request response pipeline. We will try to understand all about these and will see them in action using a simple sample application.

## Background

In an ASP.NET MVC application the request from the user first lands at the UrlRoutingModule. This module parses the requested URL and then invokes the corresponding controller and action. The controller will then render the appropriate view and the response will be sent to the user.

Now what if we want to inject some extra processing logic in this request-response life cycle. Some extra logic that is written once and can be reused across multiple controllers and/or actions.

ASP.NET MVC provides a way for us to do that by writing custom filters that can be used to inject extra processing logic in the request-response life cycle.

### What are attributes and filters

MVC provides a very clean way of injecting the pre-processing and post-processing logic for actions and controllers. They way we can put the pre-processing and post-processing logic is by decorating the actions with attributes which will invoke an attribute class implementing the filter's logic.

For example, If we need some action to be executed when the user has been authenticated then we can adorn the action with the [Authorize] attribute. This will take care of calling the attribute class which implements the authorization filter to check whether the user has is authorized or not.

Hide   Copy Code

[Authorize]

public ActionResult Index()

{

return View();

}

So the way to implement custom filters would be to implement the interface that is needed for implementing the required filter. Now we can decorate the actions with this attribute so that our filter logic will be executed when this action is called. If we want all the actions of a controller to use this filter we can decorate the controller itself with this attribute.

## Using the code

Let us now try to look at the type of filters we can implement to inject our custom processing logic.

### Type of filters

Now taking this discussion further, Let us first discuss the various types of filters that can be implemented to inject custom processing logic.

* Authorization filter
* Action filter
* Result filter
* Exception filter

### Implementing Custom Filters

Now let us try to look at implement these filters. We will simply implement the custom filters and put a simple message in the ViewBag collection. We will then use these filters with an action of controller and try to see the custom messages we inserted in the ViewBag collection on our view page.

#### Authorization filter

This filter provides authentication and authorization logic. It will be executed before the action gets executed. To implement this action the interface IAuthorizationFilter should be implemented by the custom attribute class.

Hide   Copy Code

public class CustomAuthorizationAttribute : FilterAttribute, IAuthorizationFilter

{

void IAuthorizationFilter.OnAuthorization(AuthorizationContext filterContext)

{

filterContext.Controller.ViewBag.OnAuthorization = "IAuthorizationFilter.OnAuthorization filter called";

}

}

Now when we decorate the action method with this attribute the OnAuthorize filter method will be called and our custom logic will get executed.

**Note:** In the above code we have created an attribute which will only run when the authorization is being done by the application. In our own filter method we are not doing anything related to authorization. If we were to do custom authentication and authorization then we will have to derive this attribute from AuthorizeAttributeclass and implement custom authorization logic. Perhaps we will discuss that separately. For now this filter will run run when the authorization is being done and before calling the action method so that we can inject our custom logic in it.

#### Action filter

This filter will be called before and after the action starts executing and after the action has executed. We can put our custom pre-processing and post-processing logic in this filter.

Now to implement this filter we need to create a custom filter attribute class and implement the

Hide   Copy Code

IActionFilter

filter interface. This interface provides us two methods OnActionExecuting and OnActionExecuted which will be called before and after the action gets executed respectively.

Hide   Copy Code

public class CustomActionAttribute : FilterAttribute, IActionFilter

{

void IActionFilter.OnActionExecuted(ActionExecutedContext filterContext)

{

filterContext.Controller.ViewBag.OnActionExecuted = "IActionFilter.OnActionExecuted filter called";

}

void IActionFilter.OnActionExecuting(ActionExecutingContext filterContext)

{

filterContext.Controller.ViewBag.OnActionExecuting = "IActionFilter.OnActionExecuting filter called";

}

}

#### Result filter

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

To implement the result filters we need to create a custom filter attribute class and implement the

Hide   Copy Code

IResultFilter

interface. this interface provides two methods OnResultExecuting and OnResultExecuted which will be called before and after the action result respectively.

Hide   Copy Code

public class CustomResultAttribute : FilterAttribute, IResultFilter

{

void IResultFilter.OnResultExecuted(ResultExecutedContext filterContext)

{

filterContext.Controller.ViewBag.OnResultExecuted = "IResultFilter.OnResultExecuted filter called";

}

void IResultFilter.OnResultExecuting(ResultExecutingContext filterContext)

{

filterContext.Controller.ViewBag.OnResultExecuting = "IResultFilter.OnResultExecuting filter called";

}

}

#### 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 custom error logging module.

To implement this filter we need to create a custom filter attribute class which implements IExceptionFilter. This interface gives us a methods called OnException which is a perfect place to call the exception logging module and to redirect to some error page.

Hide   Copy Code

public class CustomExceptionAttribute : FilterAttribute, IExceptionFilter

{

void IExceptionFilter.OnException(ExceptionContext filterContext)

{

filterContext.Controller.ViewBag.OnException = "IExceptionFilter.OnException filter called";

}

}

### Order of Execution

Now with all the above filters we have the following filter methods.

* IAuthorizationFilter.OnAuthorization
* IActionFilter.OnActionExecuted
* IActionFilter.OnActionExecuting
* IResultFilter.OnResultExecuted
* IResultFilter.OnResultExecuting
* IExceptionFilter.OnException

Now assuming that we have all the filters attached to a single action method what will be the order of execution of these filers. These filters will execute in following order under normal(non-exception) scenario.

1. IAuthorizationFilter.OnAuthorization
2. IActionFilter.OnActionExecuting
3. IActionFilter.OnActionExecuted
4. IResultFilter.OnResultExecuting
5. IResultFilter.OnResultExecuted

In case there is an exception, OnException will will be called as instead of the result filters.

#### Using the Custom Filters

Now from our application we just need to decorate the actions on which we need the custom filter functionality. Lets try to do this on a single action method as:

Hide   Copy Code

public class HomeController : Controller

{

[CustomAuthorization]

[CustomAction]

[CustomResultAttribute]

[CustomExceptionAttribute]

public ActionResult Index()

{

*//throw new Exception("Dummy Exception");*

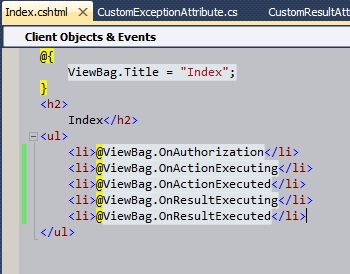
ViewBag.Message = "Index Action of Home controller is being called.";

return View();

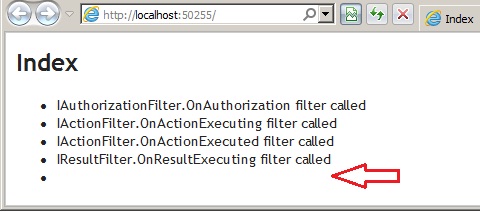
}

}

And the code to see these on the view page:



And when we try to run the application:



The important thing to note in the running application is that ViewBag.OnResultExecuted is empty. the reason for this is that the function IResultFilter.OnResultExecuted gets called when the view has been rendered i.e. the action result has been completed.

**Note:** It is advisable to put breakpoints on all filter methods and then run the application to understand the sequence of these filter methods. Also, un-commenting the line in controller which throws a dummy exception will invoke the IExceptionFilter.OnException filter method too.

### Built-in Attributes

ASP.NET MVC comes with a some of built in attribute classes that provides a some boilerplate functionality. We can create custom classes that derives from these built in classes and further provide specialized behavior as per our needs. Let us try to see some of these built in attributes.

* AuthorizeAttribute: MVC framework provides AuthorizeAttribute which is helpful in specifying our custom authorization policies.
* ActionFilterAttribute: This is the built in implementation of IActionFilter and IResultFilter. This attribute can be used as base class to implement the custom behavior for action and result filters.
* HandleErrorAttribute: This is the built in implementation of IExceptionFilter which makes it easier to implement the exception handling strategy.

ASP.NET MVC provides the following action filters −

* **Output Cache** − This action filter caches the output of a controller action for a specified amount of time.
* **Handle Error** − This action filter handles errors raised when a controller action executes.
* **Authorize** − This action filter enables you to restrict access to a particular user or role.

**Which routiong is better in mvc?**

# Attribute Routing vs Convention Routing - ASP.NET MVC Demystified

 4 YEARS AGO

[MVC-DEMYSTIFIED](https://exceptionnotfound.net/tag/mvc-demystified/) · [ASP.NET-MVC](https://exceptionnotfound.net/tag/asp-net-mvc/) · [TUTORIALS](https://exceptionnotfound.net/tag/tutorials/)

For most of ASP.NET MVC's lifetime, routing has been accomplished via **Convention Routing**, which allows developers to specify a format or group of formats which can be used to parse incoming URLs and determine the appropriate actions, controllers, and data to use for that request.

In MVC 5, though, Microsoft introduced another scheme called **Attribute Routing**. Attribute Routing allows us to define routes in close proximity to their actions, giving us greater flexibility.

Let's dive into Routing as a whole, and then show how to implement Convention Routing and Attribute Routing and why they actually work together nicely.



## What is Routing?

ASP.NET Routing is **the ability to have URLs represent abstract actions rather than concrete, physical files**.

In "traditional" websites, every URL represents a physical file, whether it is an HTML or ASPX page, or a script file, or some other content. If I see a URL of [www.example.com/articles/post.aspx?id=65](http://www.example.com/articles/post.aspx?id=65), I'm going to assume that that URL represents a folder called articles at the root of the site, and within that folder a page called post.aspx.

In MVC, no such physical folders and pages exist, and so the MVC architecture allows us to map routes to controllers and actions which may represent [many kinds of results](http://www.exceptionnotfound.net/asp-net-mvc-demystified-actionresults/). **Routing is a layer of abstraction on top of regular URLs that allows programmers greater control over what those URLs mean and how they are formatted**.

#### "Hackable" URLs

One of the things Routing allows us to do is to create "hackable" URLs; that is, URLs whose meaning is easily read, understood, and extended upon by human beings. We can use Routing to turn this messy URL:

[www.example.com/article.aspx?id=69&title=my-favorite-podcasts](http://www.example.com/article.aspx?id=69&title=my-favorite-podcasts)

into a much cleaner one:

[www.example.com/articles/69/my-favorite-podcasts](http://www.example.com/articles/69/my-favorite-podcasts)

The concept of "hackable" URLs goes a bit further, too. If I was a user looking at the clean URL, I might append "/comments" on the end of it:

[www.example.com/articles/69/my-favorite-podcasts/comments](http://www.example.com/articles/69/my-favorite-podcasts/comments)

"Hackable" URLs implies that this should display the comments for that article. If it does, then I (the user) have just discovered how I can view the comments for any given article, without needing to scroll through the page and hunt down the appropriate link.

So how do we actually implement Routing in MVC? It all starts with something called the Route Table.

## The Route Table

  
The Route Table is a collection of all possible routes that MVC can use to match submitted URLs. Items in the Route Table specify their format and where certain values are in the route structure. These values can then be mapped to controllers, actions, areas, etc. depending on their placement within the route.

Any URL that is submitted to the application will be compared to the routes in the Route Table, and the system will redirect to the first matching route found in that table. In versions of MVC up to version 5, we added routes to this table at a specific place, usually in RouteConfig. With the introduction of Attribute Routing, this method of adding routes has been retroactively termed Convention Routing.

## Convention Routing

Convention Routing approaches the routing problem general-case-first; by default, you are given a route that will probably match most if not all of your routes, and are asked to define if there are any more specific routes you would also like to handle.

A call to set up convention-based routes might look like this:

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapRoute(

name: "Special",

url: "Special/{id}",

defaults: new { controller = "Home", action = "Special", id = UrlParameter.Optional }

); //Route: /Special/12

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

); //Route: /Home/Index/12

}

Let's break down some of the pieces here:

* **routes** is the Route Table of type [RouteCollection](https://msdn.microsoft.com/en-us/library/system.web.routing.routecollection%28v=vs.110%29.aspx), and stores all of the possible routes we can match for a given URL.
* A call to **IgnoreRoute** allows us to tell ASP.NET to ignore any URL that matches that tokenized structure and process it normally.
* A call to **MapRoute** allows us to add a route to the route table.

MapRoute includes a few parameters: a name of the route that must be unique to each route, a tokenized URL structure, and default values that will be applied if alternate values are not supplied by the route. The tokenized URL structure is then used to match supplied values in the URL.

#### Matching by Convention

Say we have this route:

routes.MapRoute(

name: "PersonDefault",

url: "{controller}/{person}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

Now imagine we have this controller:

public class HomeController : Controller

{

public ActionResult Index(string person) { ... }

public ActionResult Documents(string person, int id) { ... }

}

If we submit a URL of **/Home/Dude-Person/Documents/17**, we will be directed to the Home controller's Documents action, and the person and id parameters will have values of "Dude-Person" and 17 respectively.

If we submit a URL of **/Home/Dude-Person/Documents?id=17**, we will again be directed to Home controller and Documents action with the same values as before, because MVC will look at query string values if no route values exist that match the expected parameters.

If we submit a URL of **/Home/Dude-Person**, we will be directed to the Index action (because that's what was specified in the defaults) with parameter person having the value "Dude-Person".

If we submit a URL of **/Home** we will be redirected to the Index action and person will be an empty string. If no matching value is found for a given parameter, the default value for that parameter's type is used.

One thing to keep in mind when designing your routes is that **the order in which the routes are added to the table matters**. The routing engine will take the first route that matches the supplied URL and attempt to use the route values in that route. Therefore, less common or more specialized routes should be added to the table first, while more general routes should be added later on.

For example, for the routes above, if we submit a URL of **/Home/Documents** we will be redirected to the Index action with parameter person having the value "Documents", which is probably not the desired behavior.

In short, Convention Routing approaches Routing from the general case; you generally add some routes that will match all or most of your URLs, then add more specific routes for more specialized cases. The other way to approach this problem is via Attribute Routing.

### Attribute Routing

Attribute Routing (introduced in MVC 5) is the ability to add routes to the Route Table via attributes so that the route definitions are in close proximity to their corresponding actions. We will still populate the Route Table, but we will do it in a different manner.



Before we can start using Attribute Routing, though, we must first enable it.

#### Enable Attribute Routing

If you want to use Attribute Routing, you have to enable it by calling MapMvcAttributeRoutes on the RouteCollection for your app (usually this is done in RouteConfig):

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

routes.IgnoreRoute("{resource}.axd/{\*pathInfo}");

routes.MapMvcAttributeRoutes(); //Enables Attribute Routing

}

}

#### Simple Example

A simple example of Attribute Routing might look like this:

public class HomeController : Controller

{

[Route("Users/Index")] //Route: /Users/Index

public ActionResult Index() { ... }

}

What that [Route] attribute does is specify a route to be added to the Route Table which maps to this action. The parameters to [Route]'s constructor are where the real functionality happens.

#### Parameters

For example, what if we need a way to specify that we want to include parameter data in the routes? We can do so with the {} syntax:

[Route("Users/{id}")] //Route: /Users/12

public ActionResult Details(int id) { ... }

Notice that the name in the curly braces matches the name of one of the inputs to the action. By doing this, that value of that parameter will appear in the route rather than in the query string.

We can also specify if a parameter is optional by using ?:

[Route("Users/{id}/{name?}")] //Route: /Users/12/Matthew-Jones or /Users/12

public ActionResult Details(int id, string name) { ... }

If we need a given parameter to be of a certain type, we can specify a constraint:

[Route("Users/{id:int}")] //Route: /Users/12

public ActionResult Details(int id) { ... }

[Route("{id:alpha}/Documents")] //Route: /product/Documents

public ActionResult Documents(string id) { ... }

There are quite a few different constraints we can use; Attribute Routing even includes support for regular expressions and string lengths. Check [this article from MSDN](https://blogs.msdn.com/b/webdev/archive/2013/10/17/attribute-routing-in-asp-net-mvc-5.aspx) for full details.

#### Route Prefixes

We can specify a RoutePrefix that applies to every action in a controller:

[RoutePrefix("Users")]

public class HomeController : Controller

{

[Route("{id}")] //Route: /Users/12

public ActionResult Details(int id) { ... }

}

If we need to have an action that overrides the Route Prefix, we can do so using the absolute-path prefix ~/:

[RoutePrefix("Users")]

public class HomeController : Controller

{

[HttpGet]

[Route("~/special")] //Route: /special

public ActionResult Special() { ... }

}

#### Default Routes

Specifying the default route for the application also uses the absolute-path prefix ~/. We can also specify a default route for a given route prefix by passing an empty string:

[RoutePrefix("Users")]

public class HomeController : Controller

{

[Route("~/")] //Specifies that this is the default action for the entire application. Route: /

[Route("")] //Specifies that this is the default action for this route prefix. Route: /Users

public ActionResult Index() { ... }

}

We can also specify default routes another way: by capturing them as inputs.

[RoutePrefix("Users")]

[Route("{action=index}")] //Specifies the Index action as default for this route prefix

public class HomeController : Controller

{

public ActionResult Index()

{

return View();

}

}

#### Names and Order

[Route] will also accept names and order values for the routes:

[RoutePrefix("Users")]

public class HomeController : Controller

{

[Route("Index", Name = "UsersIndex", Order = 2)]

public ActionResult Index() { ... }

[Route("{id}", Name = "UserDetails", Order = 1)]

public ActionResult Details(int id) { ... }

}

**Order is still very important!** In the above example, if we give a route of /Users/Index, we will get an exception because that matches the UserDetails route, which has a higher order. If no order is specified, the routes are inserted into the Route Table in the order they are listed.

We can solve the above conflict by either adding a constraint on UserDetails that ID must be an integer or reordering the routes and placing UsersIndex at a higher order.

Because each of the defined routes is close to their respective action, it is my recommendation that each route be as specific as possible. In other words, **when using Attribute Routing, define the routes to be specific to the decorated action**and not any other action.

#### Using Convention and Attribute Routing Together

You can also implement both Attribute and Convention routing at the same time:

public class RouteConfig

{

public static void RegisterRoutes(RouteCollection routes)

{

...

routes.MapMvcAttributeRoutes(); //Attribute routing

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

}

}

Notice that the setup above gives all the Order weight to the Attribute routes, since they were added to the Route Table first.

## Which is Better?

In this developer's opinion, **Attribute Routing is a more flexible solution than Convention Routing**, if only because it allows you quite a bit more flexibility and places the routes next to the actions that will actually use them. However there are certainly benefits to using both in tandem, particularly in situations when you know how some routes will look but aren't sure about others.

But don't take my opinion as gospel; try them both for yourself!

Chapter 9 of [Professional ASP.NET MVC 5](https://www.amazon.com/gp/product/1118794753/ref=as_li_tl?ie=UTF8&camp=1789&creative=390957&creativeASIN=1118794753&linkCode=as2&tag=excnotfou-20&linkId=FKNM646QJCJR7J6E) has some more detail on Attribute Routing vs. Convention Routing, including some good guidelines on when to choose one or the other and how to deal with routing in Areas. Go take a look; having another reference book never hurts.

Happy Coding!

**What is delegate and why it can be use?**

A delegate is a type safe a function pointer that can reference a method that has the same signature as that of the delegate. You can take advantage of delegatese in C# to implement events and call back methods. A multicast delegate is one that can point to one or more methods that have identical signatures.

*Delegates have the following properties:*

* Delegates are similar to C++ function pointers, but are type safe.
* Delegates allow methods to be passed as parameters.
* Delegates can be used to define callback methods.
* Delegates can be chained together; for example, multiple methods can be called on a single event.
* Methods don't need to match the dele gate signature exactly. For more information, see Covariance and Contra variance.
* C# version 2.0 introduces the concept of Anonymous Methods, which permit code blocks to be passed as parameters in place of a separately defined method.

**Difference between promise and obseravble?**

203

Both **Promises** and **Observables** will help us work with the **asynchronous functionalities** in JavaScript. They are very similar in many cases, however, there are still some differences between the two as well, promises are values that will resolve in asynchronous ways like **http** calls. On the other hand, observables deal with a sequence of **asynchronous events**. The main differences between them are listed below:

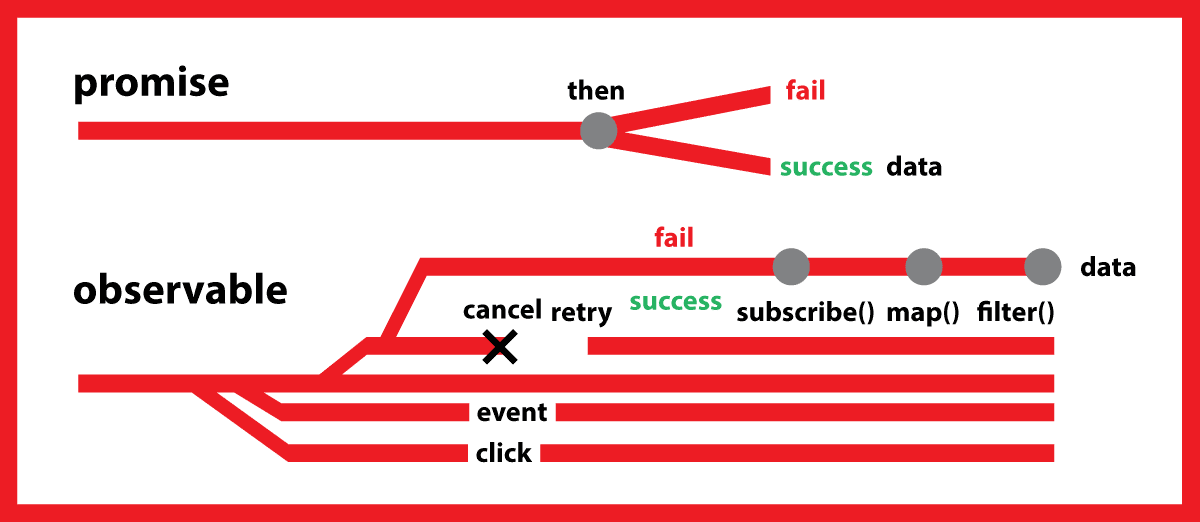
**promise:**

* having one pipeline
* usually only use with async data return
* not easy to cancel

**observable:**

* are cancellable
* are re-triable by nature such as retry and retryWhen
* stream data in multiple pipelines
* having array-like operations like map, filter etc
* can be created from other sources like events
* they are functions, which could be subscribed later on

Also, I've created the graphical image for you below to show the differences visually:

[](https://i.stack.imgur.com/Ewn3b.png)

**How can provide the security in angular?**

<https://angular.io/guide/security>

# ANGULAR SECURITY BEST PRACTICES

*Software security is a hot topic nowadays. We, web developers, need to be up-to-date with all latest security issues that we could encounter when developing a web application. In this blog we’ll check what kind of best practices we should have in mind when building an Angular app so we limit the amount of security issues we could have.*

## UP-TO-DATE ANGULAR LIBRARIES

The angular team is doing releases at regular intervals for feature enhancements, bug fixes and security patches as appropriate. So, it is recommended to update the Angular libraries at regular intervals. Not doing so may allow attackers to attack the app using known security vulnerabilities present within older releases.

## 1. PREVENTING CROSS-SITE SCRIPTING (XSS)

XSS enables attackers to inject client-side scripts into web pages viewed by other users. Such code can then, for example, steal user data or perform actions to impersonate the user. This is one of the **most common attacks** on the web.

### 1.1. SANITIZATION AND SECURITY CONTEXTS

To systematically block XSS bugs, Angular treats all values as **untrusted by default**. When a value is inserted into the DOM from a template, via property, attribute, style, class binding, or interpolation, Angular sanitizes and escapes untrusted values.

This is the declaration of the sanitization providers in the BrowserModule:

export const BROWSER\_SANITIZATION\_PROVIDERS: Array<any> = [

{provide: Sanitizer, useExisting: DomSanitizer},

{provide: DomSanitizer, useClass: DomSanitizerImpl},

];

@NgModule({

providers: [

BROWSER\_SANITIZATION\_PROVIDERS

...

],

exports: [CommonModule, ApplicationModule]

})

export class BrowserModule {}

### THE DOM SANITIZATION SERVICE

The goal of the DomSanitizer is to clean untrusted parts of values.

The skeleton of the class looks like this:

export enum SecurityContext { NONE, HTML, STYLE, SCRIPT, URL, RESOURCE\_URL }

export abstract class DomSanitizer implements Sanitizer {

abstract sanitize(context: SecurityContext, value: SafeValue|string|null): string|null;

abstract bypassSecurityTrustHtml(value: string): SafeHtml;

abstract bypassSecurityTrustStyle(value: string): SafeStyle;

abstract bypassSecurityTrustScript(value: string): SafeScript;

abstract bypassSecurityTrustUrl(value: string): SafeUrl;

abstract bypassSecurityTrustResourceUrl(value: string): SafeResourceUrl;

}

As you can see, there are two kinds of method patterns. The first one is the sanitize method, which gets the contextand an untrusted value and returns a trusted value. The other ones are the bypassSecurityTrustX methods which are getting the untrusted value according to the value usage and are returning a trusted object.

#### THE SANITIZE METHOD

If a value is trusted for the context, this sanitize method will (in case of a SafeValue) unwrap the contained safe value and use it directly. Otherwise, the value will be sanitized to be safe according to the security context.

There are three main helper functions for sanitizing the values. The sanitizeHtml function sanitizes the untrusted HTML value by parsing the value and checks its tokens. The sanitizeStyle and sanitizeUrl functions sanitize the untrusted style or URL value by regular expressions.

#### HOW CAN WE DISABLE THE SANITIZATION LOGIC?

In specific situations, it might be necessary to disable sanitization. Users can bypass security by constructing a value with one of the bypassSecurityTrustX methods, and then binding to that value from the template.

An example:

import {BrowserModule, DomSanitizer} from '@angular/platform-browser'

@Component({

selector: 'my-app',

template: `

<div [innerHtml]="html"></div>

`,

})

export class App {

constructor(private sanitizer: DomSanitizer) {

this.html = sanitizer.bypassSecurityTrustHtml('<h1>DomSanitizer</h1><script>ourSuperSafeCode()</script>') ;

}

}

*Be careful: If you trust a value that might be malicious, you are introducing a security vulnerability into your application!*

### 1.2. CONTENT SECURITY POLICY (CSP)

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross Site Scripting (XSS) and data injection attacks. These attacks are used for everything from data theft to site defacement or distribution of malware.

To enable CSP, configure your web server to return an appropriate Content-Security-Policy HTTP header. You can find a very detailed manual how to enable CSP on the [MDN website](https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP). To check if your CSP is valid you can use the [CSP evaluator from google](https://csp-evaluator.withgoogle.com/).

### 1.3. USE THE OFFLINE TEMPLATE COMPILER (AKA AOT-COMPILER)

Angular templates are the same as executable code: HTML, attributes, and binding expressions (but not the values bound) in templates are trusted to be safe. This means that if an attacker can control a value that is being parsed by the template we have a security leak. Never generate template source code by concatenating user input and templates. To prevent these vulnerabilities, use the **offline template compiler**, also known as template injection.

If you use the Angular CLI, it’s easy to enable AOT:

ng build --aot

ng serve --aot

More info can be found on the [Angular Guide website](https://angular.io/guide/aot-compiler).

### 1.4. AVOID DIRECT USE OF THE DOM APIS

The built-in browser DOM APIs don’t automatically protect you from security vulnerabilities. For example, document, the node available through ElementRef, and many third-party APIs contain unsafe methods. Avoid interacting with the DOM directly and instead use **Angular templates** where possible.

### 1.5. SERVER-SIDE XSS PROTECTION

Injecting template code into an Angular application is the same as injecting executable code into the application. So, validate all data on server-side code and escape appropriately to prevent XSS vulnerabilities on the server. Also, Angular recommends not to generate Angular templates on the server side using a templating language.

## 2. HTTP-LEVEL VULNERABILITIES

Angular has built-in support to help prevent two common HTTP vulnerabilities, cross-site request forgery (CSRF or XSRF) and cross-site script inclusion (XSSI). Both of these must be mitigated primarily on the server side, but Angular provides helpers to make integration on the client side easier.

### 2.1. CROSS-SITE REQUEST FORGERY (XSRF)

Cross-site request forgery (also known as one-click attack or session riding) is abbreviated as CSRF or XSRF. It is a type of malicious exploit of a website where unauthorized commands are transmitted from a user that the web application trusts.

In a common anti-XSRF technique, the application server sends a randomly generated authentication token in a cookie. The client code reads the cookie and adds a custom request header with the token in all subsequent requests. The server compares the received cookie value to the request header value and rejects the request if the values are missing or don’t match.

This technique is effective because all browsers implement the same origin policy. Only code from the website on which cookies are set can read the cookies from that site and set custom headers on requests to that site. That means only your application can read this cookie token and set the custom header.

Angular HttpClient provides built-in support for doing checks on the client side. Read further details on [Angular XSRF Support](https://angular.io/guide/http#security-xsrf-protection).

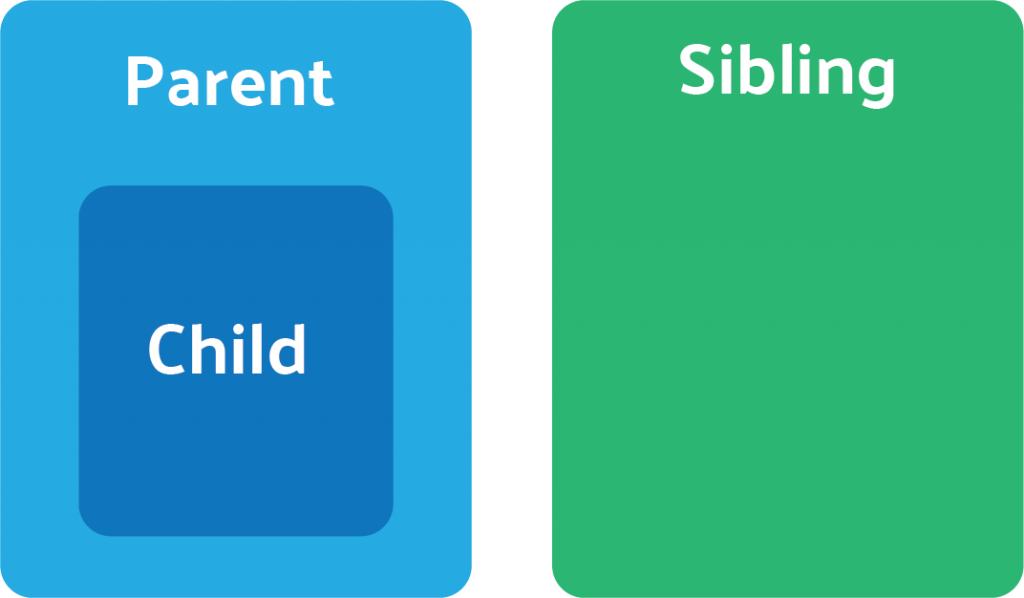
### 2.2. CROSS-SITE SCRIPT INCLUSION (XSSI)

Cross-site script inclusion (also known as **JSON vulnerability**) can allow an attacker’s website to read data from a JSON API. The attack works on older browsers by overriding native JavaScript object constructors, and then including an API URL using a <script> tag. This attack is only successful if the returned JSON is executable as JavaScript.

Servers can prevent an attack by prefixing all JSON responses to make them non-executable, by convention, using the well-known string ")]}',\n". Angular’s HttpClient library recognizes this convention and automatically strips the string ")]}',\n" from all responses before further parsing.

**How many ways to pass the data from one component to another in angular?**

Data sharing is an essential concept to understand before diving into your first Angular project. In this lesson, I provide four different methods for sharing data between Angular components.

The Parent-Child-Sibling structure of our Angular app.

## Parent to Child: Sharing Data via Input

This is probably the most common and straightforward method of sharing data. It works by using the [**@Input() decorator**](https://angular.io/docs/ts/latest/api/core/index/Input-interface.html) to allow data to be passed via the template.

### parent.component.ts

|  |
| --- |
| import { Component } from '@angular/core';  **@Component**({  selector: 'app-parent',  template: `  <app-child [childMessage]="parentMessage"></app-child>  `,  styleUrls: ['./parent.component.css'] }) export class ParentComponent{  parentMessage = "message from parent"  constructor() { } } |

### child.component.ts

|  |
| --- |
| import { Component, Input } from '@angular/core';  **@Component**({  selector: 'app-child',  template: `  Say {{ message }}  `,  styleUrls: ['./child.component.css'] }) export class ChildComponent {   **@Input**() childMessage: string;   constructor() { }  } |

## Child to Parent: Sharing Data via ViewChild

[**ViewChild**](https://angular.io/api/core/ViewChild) allows a one component to be injected into another, giving the parent access to its attributes and functions. One caveat, however, is that child won’t be available until after the view has been initialized. This means we need to implement the AfterViewInit lifecycle hook to receive the data from the child.

### parent.component.ts

|  |
| --- |
| import { Component, ViewChild, AfterViewInit } from '@angular/core'; import { ChildComponent } from "../child/child.component";  **@Component**({  selector: 'app-parent',  template: `  Message: {{ message }}  <app-child></app-child>  `,  styleUrls: ['./parent.component.css'] }) export class ParentComponent implements AfterViewInit {   **@ViewChild**(ChildComponent) child;   constructor() { }   message:string;   ngAfterViewInit() {  this.message = this.child.message  } } |

### child.component.ts

|  |
| --- |
| import { Component} from '@angular/core';  **@Component**({  selector: 'app-child',  template: `  `,  styleUrls: ['./child.component.css'] }) export class ChildComponent {   message = 'Hola Mundo!';   constructor() { }  } |

## Child to Parent: Sharing Data via Output() and EventEmitter

Another way to share data is to emit data from the child, which can be listed to by the parent. This approach is ideal when you want to share data changes that occur on things like button clicks, form entires, and other user events.

In the parent, we create a function to receive the message and set it equal to the message variable.

In the child, we declare a messageEvent variable with the Output decorator and set it equal to a new event emitter. Then we create a function named sendMessage that calls emit on this event with the message we want to send. Lastly, we create a button to trigger this function.

The parent can now subscribe to this messageEvent that’s outputted by the child component, then run the receive message function whenever this event occurs.

### parent.component.ts

|  |
| --- |
| import { Component } from '@angular/core';  **@Component**({  selector: 'app-parent',  template: `  Message: {{message}}  <app-child (messageEvent)="receiveMessage($event)"></app-child>  `,  styleUrls: ['./parent.component.css'] }) export class ParentComponent {   constructor() { }   message:string;   receiveMessage($event) {  this.message = $event  } } |

### child.component.ts

|  |
| --- |
| import { Component, Output, EventEmitter } from '@angular/core';  **@Component**({  selector: 'app-child',  template: `  <button (click)="sendMessage()">Send Message</button>  `,  styleUrls: ['./child.component.css'] }) export class ChildComponent {   message: string = "Hola Mundo!"   **@Output**() messageEvent = new EventEmitter<string>();   constructor() { }   sendMessage() {  this.messageEvent.emit(this.message)  } } |

## Unrelated Components: Sharing Data with a Service

When passing data between components that lack a direct connection, such as siblings, grandchildren, etc, you should you a shared service. When you have data that should aways been in sync, I find the [**RxJS BehaviorSubject**](https://xgrommx.github.io/rx-book/content/subjects/behavior_subject/index.html) very useful in this situation.

You can also use a regular RxJS Subject for sharing data via the service, but here’s why I prefer a BehaviorSubject.

* It will always return the current value on subscription - there is no need to call onnext
* It has a getValue() function to extract the last value as raw data.
* It ensures that the component always receives the most recent data.

In the service, we create a private BehaviorSubject that will hold the current value of the message. We define a currentMessage variable handle this data stream as an observable that will be used by the components. Lastly, we create function that calls next on the BehaviorSubject to change its value.

The parent, child, and sibling components all receive the same treatment. We inject the DataService in the constructor, then subscribe to the currentMessage observable and set its value equal to the message variable.

Now if we create a function in any one of these components that changes the value of the message. when this function is executed the new data it’s automatically broadcast to all other components.

### data.service.ts

|  |
| --- |
| import { Injectable } from '@angular/core'; import { BehaviorSubject } from 'rxjs';  **@Injectable**() export class DataService {   private messageSource = new BehaviorSubject('default message');  currentMessage = this.messageSource.asObservable();   constructor() { }   changeMessage(message: string) {  this.messageSource.next(message)  }  } |

### parent.component.ts

|  |
| --- |
| import { Component, OnInit } from '@angular/core'; import { DataService } from "../data.service";  **@Component**({  selector: 'app-parent',  template: `  {{message}}  `,  styleUrls: ['./sibling.component.css'] }) export class ParentComponent implements OnInit {   message:string;   constructor(private data: DataService) { }   ngOnInit() {  this.data.currentMessage.subscribe(message => this.message = message)  }  } |

### sibling.component.ts

|  |
| --- |
| import { Component, OnInit } from '@angular/core'; import { DataService } from "../data.service";  **@Component**({  selector: 'app-sibling',  template: `  {{message}}  <button (click)="newMessage()">New Message</button>  `,  styleUrls: ['./sibling.component.css'] }) export class SiblingComponent implements OnInit {   message:string;   constructor(private data: DataService) { }   ngOnInit() {  this.data.currentMessage.subscribe(message => this.message = message)  }   newMessage() {  this.data.changeMessage("Hello from Sibling")  }  } |

**MicroSoft**

**----------------------------------------**

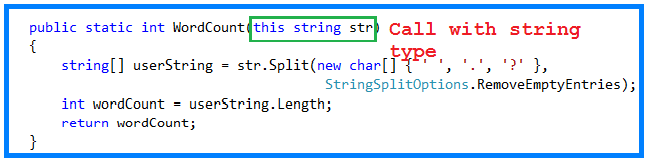
**1. what are the extension methods?**

Extension Methods are a new feature in C# 3.0. An Extension Method enables us to add methods to existing types without creating a new derived type, recompiling, or modify the original types. We can say that it extends the functionality of an existing type in .NET. An Extension Method is a static method to the existing static class. We call an Extension Method in the same general way; there is no difference in calling.

An Extension Method is:

1. It is a static method.
2. It must be located in a static class.
3. It uses the "this" keyword as the first parameter with a type in .Net and this method will called by a given type instance on the client side.
4. It also shown by VS intellisense. When we press the dot (.) after a type instance then it comes in VS intellisense.
5. An Extension Method should be in the same namespace as it is used or you need to import the namespace of the class by a using statement.
6. You can give any name of for the class that has an Extension Method but the class should be static.
7. If you want to add new methods to a type and you don't have the source code for it then the solution is to use and implement Extension Methods of that type.
8. If you create Extension Methods that have the same signature methods as the type you are extending then the Extension Methods will never be called.

We create an Extension Method for a string type so string will be specified as a parameter for this Extension Method and that method will be called by a string instance using the dot operator.



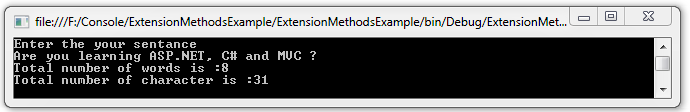
In the above method WordCount(), we are passing a string type with this so it will be called by the string type variable, in other words a string instance.

Now we create a static class and two static methods, one for the total word count in a string and another for the total number of characters in a string without a space.

1. **using** System;
2. **namespace** ExtensionMethodsExample
3. {
4. **public** **static** **class** Extension
5. {
6. **public** **static** **int** WordCount(**this** **string** str)
7. {
8. **string**[] userString = str.Split(**new** **char**[] { ' ', '.', '?' },
9. StringSplitOptions.RemoveEmptyEntries);
10. **int** wordCount = userString.Length;
11. **return** wordCount;
12. }
13. **public** **static** **int** TotalCharWithoutSpace(**this** **string** str)
14. {
15. **int** totalCharWithoutSpace = 0;
16. **string**[] userString = str.Split(' ');
17. **foreach** (**string** stringValue **in** userString)
18. {
19. totalCharWithoutSpace += stringValue.Length;
20. }
21. **return** totalCharWithoutSpace;
22. }
23. }
24. }

Now we create an executable program that has a string as an input and uses an Extension Method to count the total words in that string and the total number of characters in that string then show the result in a console screen.

1. **using** System;
2. **namespace** ExtensionMethodsExample
3. {
4. **class** Program
5. {
6. **static** **void** Main(**string**[] args)
7. {
8. **string** userSentance = **string**.Empty;
9. **int** totalWords = 0;
10. **int** totalCharWithoutSpace = 0;
11. Console.WriteLine("Enter the your sentance");
12. userSentance = Console.ReadLine();
13. //calling Extension Method WordCount
14. totalWords = userSentance.WordCount();
15. Console.WriteLine("Total number of words is :"+ totalWords);
16. //calling Extension Method to count character
17. totalCharWithoutSpace = userSentance.TotalCharWithoutSpace();
18. Console.WriteLine("Total number of character is :"+totalCharWithoutSpace);
19. Console.ReadKey();
20. }
21. }
22. }



**2. what is the delegates when will be use?**

**3. boxing and un boxing ?**

Boxing and unboxing enable a unified view of the type system wherein a value of any type can ultimately be treated as an object. Converting a value type into reference type is called Boxing. Unboxing is an explicit operation.

C# provides a "unified type system". All types including value types derive from the type object. It is possible to call the object methods on any value, even values of "primitive" types, such as int. The example is shown below.

1. **using** System;
2. **class** Test
3. {
4. **static** **void** Main()
5. {
6. Console.WriteLine(3.ToString());
7. }
8. }

It calls the object-defined ToString method on an integer literal. The example -

1. **class** Test
2. {
3. **static** **void** Main()
4. {
5. **int** i = 1;
6. **object** o = i; // boxing
7. **int** j = (**int**)o; // unboxing
8. }
9. }

An int value can be converted into object and back again into int.

**4. difference between post and put?**

**5. Difference Between dispose() and finalize() in C#?**

**6. mvc archtecture?**

**7. how to send data to view to controller?**

View-to-Controller

Let us first discuss how to pass data from a ASP.NET MVC View to Controller. There are four ways to pass the data from View to Controller which are explained below:

1. Traditional Approach: In this approach, we can use the request object of the HttpRequestBase class. This object contains the input field name and values as name-value pairs in case of the form submit. So we can easily get the values of the controls by their names using as indexer from the request object in the controller.

For example: Let's say you are having an input in the form with name 'txtName', then its values can be retrieved in controller from request object like below:

Hide   Copy Code

string strName = Request["txtName"].ToString();

1. Through FormCollection: We can also get post requested data by the FormCollection object. This object also has requested data as the name/value collection as the Request object.

For example:

Hide   Copy Code

[HttpPost]

public ActionResult Calculate(FormCollection form)

{

string strName = form["txtName"].ToString();

. . . . . . . . . . . . . . . . . . . .

}

1. Through Parameters: We can also pass the input field names as parameters to the post action method by keeping the names same as the input field names. These parameters will have the values for those fields and the parameter types should be string. Also, there is no need to define the parameters in any specific sequence.

For example:

Hide   Copy Code

[HttpPost]

public ActionResult Calculate(string txtName)

{

string strName = Convert.ToString(txtName);

. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .

}

In all of the above approaches, we need to even convert the non-string type to string type due to which if any parsing fails, then the entire action may fail here. Here we have to convert each value to avoid any exceptions but, in the below 4th approach of passing data from view to controller, it reduces the amount of code.

1. Strongly typed model binding to view: Here, we need to create a strongly typed view which will bind directly the model data to the various fields of the page.

For example:

* 1. Create a model with the required member variables.

Let's say we have a model named 'Person' with member variable named as 'Name'

* 1. Now pass the empty model to the view as parameter in the controller action.

For example:

Hide   Copy Code

public ActionResult GetName()

{

Person person = new Person();

return View(person);

}

* 1. Prepare the strongly typed view to display the model property values through html elements as below:

For example:

Hide   Copy Code

<div><%= Html.Encode(person.Name) %></div>

* 1. Create the action method that handles the POST request & processes the data.

For example:

Hide   Copy Code

[HttpPost]

public ActionResult GetPersonName(Person person)

{

return Content(person.Name.ToString());

}

**8. difference between string and stringbuilder?**

### String vs. StringBuilder

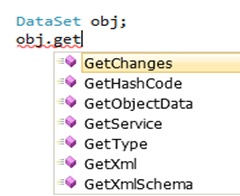
* String
  + Under System namespace
  + Immutable (readonly) instance
  + Performance degrades when continuous change of value occurs
  + Thread-safe
* StringBuilder (mutable string)
  + Under System.Text namespace
  + Mutable instance
  + Shows better performance since new changes are made to an existing instance

**9. what is dynamic in c#?**

**10.difference between dynamic and var key words?**

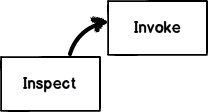
What is Reflection and why we need it?

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 the members of the object. This is possible because of Reflection.



Reflection also goes one step further; it can also invoke a member which is inspected. For instance if Reflection detects that there is a method called GetChanges in an object, we can get a reference to that method instance and invoke it on runtime.

In simple words Reflection passes through two steps: “Inspect” and “Invoke” (optional). The "Invoke" process is optional.



How do we implement Reflection?

Implementing reflection in c# is a two step process ,  1st 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 *ClassLibrary1.dll* which has a class called Class1. We can use the Assembly (belongs to the 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 a reference to the type of the object.

Hide   Copy Code

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 a reference of the type of the object we can then call GetMembers or GetProperties to browse through the methods and properties of the class.

Hide   Copy Code

*// 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:

Hide   Copy Code

parameterType.InvokeMember("Display",BindingFlags.Public |

BindingFlags.NonPublic | BindingFlags.InvokeMethod |

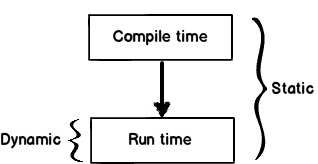
BindingFlags.Instance,null, objMyClass, null);

What are the practical uses of Reflection?

* If you are creating an application like a Visual Studio editor where you want to show the internals of an object by using intellisense.
* If you are creating a unit testing framework. In unit testing frameworks we need to invoke methods and properties dynamically for testing purpose.
* Sometimes we would like to dump properties, methods, and assembly references to a file or show it on screen.

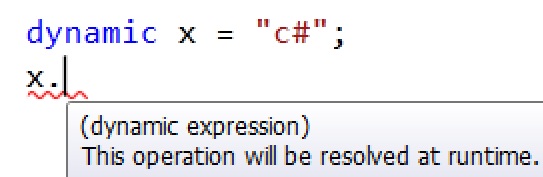
What is the use of the dynamic keyword?

Programming languages can be divided into two categories: strongly typed and dynamically typed. Strongly typed languages are those where checks happen during compile time while dynamic languages are those where type checks are bypassed during compile time. In a dynamic language object types are known only during runtime and type checks are activated only at runtime.



We would like to take advantage of both worlds. Because many times we do not know the object type until the code is executed. In other words we are looking at something like a dynamically and statically typed kind of environment. That’s what the 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”.



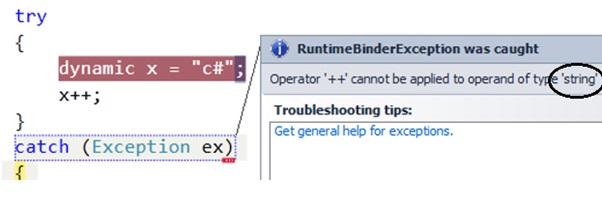
Now try the below code out. In the code I have created a dynamic variable which is initialized with string data. And in the second line I am trying to have fun by trying to execute a numeric incremental operation. So what will happen now? Think....

Hide   Copy Code

dynamic x = "c#";

x++;

Now this code will compile fine without any complaints. 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 the general data type to a specific data type (e.g.: string for the below code).



What are the practical uses of the dynamic keyword?

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

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

Hide   Copy Code

*// 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.

Hide   Copy Code

*// 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];

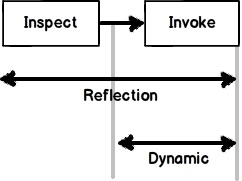
What is the difference between Reflection and dynamic?

* Both Reflection and dynamic are used when we want to operate on an object during runtime.
* Reflection is used to inspect the meta-data of an object. It also has the ability to invoke members of an object at 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 the compiler is least bothered if those methods / properties 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.
* dynamic is instance specific: you don't have access to static members; you have to use Reflection in those scenarios.

Below is the detailed comparison table which shows in which scenario they are suited:

|  |  |  |
| --- | --- | --- |
|  | **Reflection** | **Dynamic** |
| **Inspect (meta-data)** | Yes | No |
| **Invoke public members** | Yes | Yes |
| **Invoke private members** | Yes | No |
| **Caching** | No | Yes |
| **Static class** | Yes | No |

Below is a simple diagram which summarizes visually what Reflection can do and what the dynamic keyword can do.

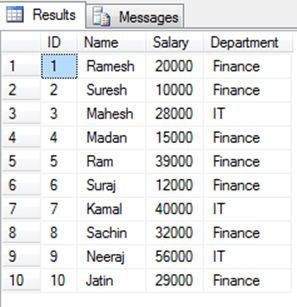


**Morgan**

**--------------------------------------**

**1. what is the use iQuerable ?**

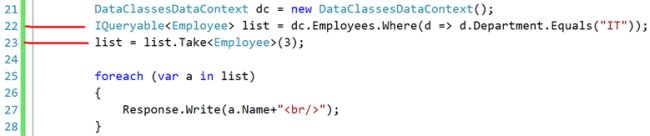
IEnumerable and IQueryable are used for data manipulation in LINQ from the database and collections.  
  
For getting the data from the database, I have created a table named "Employee" that has some data and looks like:



Then creating the Data Context class (.dbml class) in your project that converts the database table named "Employee" as a class.

Now I will tell you the functionality and some basic differences between IEnumerable and IQueryable using the object of the Data Context class..

**IEnumerable Code**

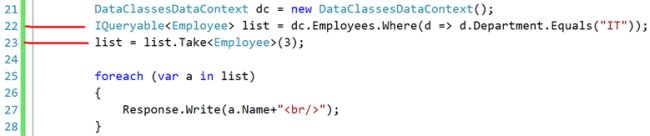


**SQL statement after execution of above query**

After the execution of line number 18, the SQL statement will look like the following until the end:

https://www.c-sharpcorner.com/UploadFile/a20beb/ienumerable-vs-iqueryable-in-linq/Images/IEnumerable%20vs%20IQueryable%20in%20LINQ3.jpg

**IQueryable Code**



**SQL statement after execution of the preceding query**

After the execution of line number 22, the SQL statement will look like:

https://www.c-sharpcorner.com/UploadFile/a20beb/ienumerable-vs-iqueryable-in-linq/Images/IEnumerable%20vs%20IQueryable%20in%20LINQ5.jpg

But after the execution of line number 23, SQL statement will add the Top for the filtering.

https://www.c-sharpcorner.com/UploadFile/a20beb/ienumerable-vs-iqueryable-in-linq/Images/IEnumerable%20vs%20IQueryable%20in%20LINQ6.jpg

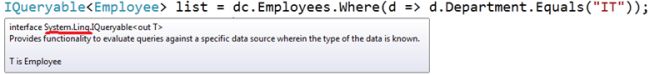
In both syntaxes I am accessing the data from the Employee table and then taking only 3 rows from that data.

Differences

**IEnumerable**

1. IEnumerable exists in the System.Collections namespace.  
     
   
2. IEnumerable is suitable for querying data from in-memory collections like List, Array and so on.
3. While querying data from the database, IEnumerable executes "select query" on the server-side, loads data in-memory on the client-side and then filters the data.  
     
   
4. IEnumerable is beneficial for LINQ to Object and LINQ to XML queries.

**IQueryable**

1. IQueryable exists in the System.Linq Namespace.  
     
   
2. IQueryable is suitable for querying data from out-memory (like remote database, service) collections.
3. While querying data from a database, IQueryable executes a "select query" on server-side with all filters.  
     
   https://www.c-sharpcorner.com/UploadFile/a20beb/ienumerable-vs-iqueryable-in-linq/Images/IEnumerable%20vs%20IQueryable%20in%20LINQ10.jpg
4. IQueryable is beneficial for LINQ to SQL queries.

**2. what is Tuples?**

Often, we want to return more than one value from a class method. Prior to the introduction of tuples in .NET, there were three common ways to do so.

* Out parameters
* Class or struct types
* Anonymous types returned through a dynamic return type

Tuples solve this problem. Tuples aren’t new to C# or .NET. Tuples were first introduced as a part of .NET Framework 4.0.

A C# tuple is a data structure that provides an easy way to represent a single set of data. The System.Tuple class provides static methods to create tuple objects.

Tuples allow us to,

* Create, access, and manipulate a data set
* Return a data set from a method without using out parameter
* Pass multiple values to a method through a single parameter

## Create and Access Tuples

We can create a Tuple<> using its constructor or the "Create" method. The code snippet in Listing 1 creates a 3-tuple using a constructor. The tuple is a set of 3 data types including two strings and one int that represents an author's name, book title, and year of publication.

1. // Create a 3-tuple
2. var author = **new** Tuple<**string**, **string**, **int**>("Mahesh Chand", "ADO.NET Programming", 2003);
4. // Display author info
5. System.Console.WriteLine("Author {0} wrote his first book titled {1} in {2}.", author.Item1, author.Item2, author.Item3);

Listing 1.

The code snippet in Listing 2 creates a 5-tuple using the static "Create" method. The tuple is a set of 5 data types including three strings, one int, and one double data type.

1. // Create a 5-tuple
2. var pubAuthor = Tuple.Create("Mahesh Chand", "Graphics Programming with GDI+", "Addison Wesley", 2004, 49.95);
4. System.Console.WriteLine("Author {0} wrote his fourth book titled {1} for {2} in {3}. Price: {4}", pubAuthor.Item1, pubAuthor.Item2, pubAuthor.Item3, pubAuthor.Item4, pubAuthor.Item5);

Listing 2.

## Nested Tuples

.NET framework supports tuples with up to seven elements. To have a tuple with more than seven elements, you can use the 8th element, TRest, to created nesting tuple objects. The code snippet in Listing 3 creates a tuple with a nested tuple inside it.

1. var even8 = **new** Tuple<**int**, **int**, **int**, **int**, **int**, **int**, **int**, Tuple<**double**, **double**, **double**>> (2, 4, 6, 8, 10, 12, 14, Tuple.Create(1.1,1.2,1.3));
2. Console.WriteLine("{0},{1},{2}", even8.Rest.Item1, even8.Rest.Item2, even8.Rest.Item3);

Listing 3.

## Tuples in methods

A tuple is useful when you need to pass a data set as a single parameter of a method without using ref and out parameters. The code snippet in Listing 4 passes a tuple as a parameter of the method.

1. ***public*** ***void*** SetTupleMethod(Tuple<***string***, ***string***, ***int***> tupleAuthor)
2. {
3. var author2 = tupleAuthor;
4. Console.WriteLine(*"Author:{0}, Title:{1}, Year:{2}."*,
5. author2.Item1, author2.Item2, author2.Item3);
6. }

Listing 4.

The following code snippet in Listing 5 calls the method.

1. ts.SetTupleMethod(**new** Tuple<**string**, **string**, **int**>(
2. "Mike Gold", "Code UML", 2005));

Listing 5.

## Return Tuples

A tuple can be used to return a data set as a single variable of a method. The code snippet in Listing 6 returns a tuple with 3 values.

1. **public** **static** Tuple<**string**, **string**, **int**> GetTupleMethod()
2. {
3. // Create a 3-tuple and return it
4. var author = **new** Tuple<**string**, **string**, **int**>(
5. "Mahesh Chand", "Programming C#", 2002);
6. **return** author;
7. }

Listing 6.

The code snippet in Listing 7 calls the method, gets a tuple, and reads its values.

1. var author2 = TupleSamples.GetTupleMethod();
2. Console.WriteLine("Author:{0}, Title:{1}, Year:{2}.", author2.Item1, author2.Item2, author2.Item3);

Listing 7.

## C# 7.0 and Tuples

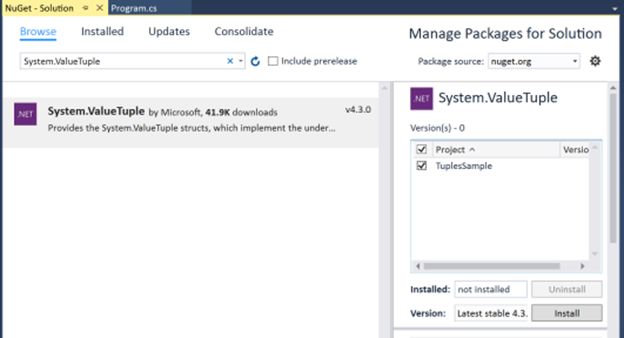
**Note**C# 7.0 features may not be available as a part of Visual Studio 2017 RC or other previous versions. To add the Tuples feature, you may want to download and install the NuGet package.

**Step 1**

Right click on the project name in Solution Explorer and select “Manage NuGet Package”.

**Step 2**

Click on the "Browse" tab and type System.ValueTuple in the TextBox. You will see the package name, as shown below.



**Step 3**

Select “TuplesSample” and click on the "Install" button.

Follow the instructions.

You’re now ready to use C# 7.0 tuples.

C# 7.0 extends the tuples' functionality by adding tuple types and tuple literals.

We can replace the above tuples code sample with the code listed in Listing 8, where the TupleReturnLiteral method returns a tuple type of three values.

1. // tuple return type
2. **public** (**string**, **string**, **long**) TupleReturnLiteral(**long** id)
3. {
5. **string** name = **string**.Empty;
6. **string** title = **string**.Empty;
7. **long** year = 0;
9. **if** (id == 1000)
10. {
11. name = "Mahesh Chand";
12. title = "ADO.NET Programming";
13. year = 2003;
14. }
16. // tuple literal
17. **return** (name, title, year);
19. }

Listing 8.

The code listed in Listing 9 calls the above TupleReturnLiternal method and returns a tuple variable. The code reads the tuple values using Item1, Item2, and Item3 of tuple and displays the values on the console.

1. TupleSamples ts = **new** TupleSamples();
2. var author = ts.TupleReturnLiteral(1000);
3. Console.WriteLine($"Author {author.Item1} {author.Item2} {author.Item3} ");

Listing 9.

To make the above code more readable, we can name the tuple return type values. The code snippet in Listing 10 changes the method signature by giving the tuple type values names.

1. // tuple return type
2. **public** (**string** name, **string** title, **long** year) TupleReturnLiteral(**long** id)
3. {
5. **string** name = **string**.Empty;
6. **string** title = **string**.Empty;
7. **long** year = 0;
9. **if** (id == 1000)
10. {
12. name = "Mahesh Chand";
13. title = "ADO.NET Programming";
14. year = 2003;
15. }
17. // tutle literal
18. **return** (name, title, year);
20. }

Listing 10.

The code snippet in Listing 11 calls the method listed in Listing 10. As you can see from Listing 11, the tuple values can be accessed using the tuple variable names.

1. TupleSamples ts = **new** TupleSamples();
2. var author = ts.TupleReturnLiteral(1000);
3. Console.WriteLine($"Author {author.name} {author.title} {author.year} ");

Listing 11.

## Summary

In this article, we learned about tuples in C# and how to apply the new changes introduced in C# 7.0.

**Next C# 7.0 Feature >>**[Out Variable In C# 7.0](http://www.c-sharpcorner.com/article/out-variables-in-c-sharp-7-0/)

**References**

References used to write this article,

https://blogs.msdn.microsoft.com/dotnet/2016/08/24/whats-new-in-csharp-7-0/

**3. what is fragmentation caching?**

Fragment caching allows to cache specific portions of the page rather than the whole page. It is done by implementing the page in different parts by creating everything in form of user controls and caching each user control individually.

**4. what is lock and monitor?**

**5. what is satilite assembly?**

A satellite assembly is a .NET Framework assembly containing resources specific to a given language. Using satellite assemblies, you can place resources for different languages in different assemblies, and the correct assembly is loaded into memory only if the user selects to view the application in that language.

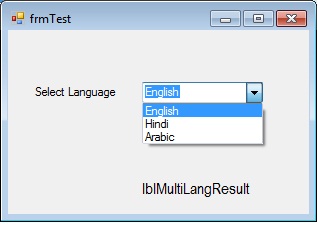
(Source: <http://vb.net-informations.com/framework/satellite-assembly.htm>)

## Using the code

Here is the step by step procedure to create an application with a satellite assembly:

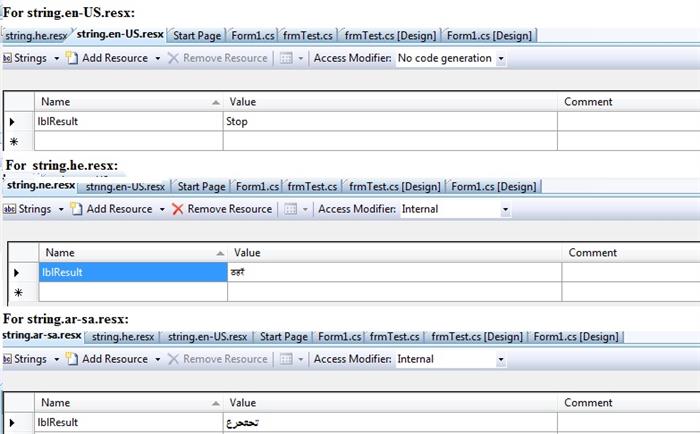
1.Create a new Windows project (say: TestApp).

2.Add a ComboBox (name it cmbLanguage) and two Labels (one label for “Select Language” and the other for Displaying the Result (name it lblMultiLangResult)) as shown below:



1. Add three Resx files (string.en-US.resx, string.he.resx, and string.ar-sa.resx for English, Hindi, and Arabic, respectively). Note: For other language codes, visit: <http://msdn.microsoft.com/en-us/library/ms533052(v=vs.85).aspx>

2. In the resx files, enter the values shown below:



5. Next, open the code file (.cs file) and create the object for the resource manager at class level:

Hide   Copy Code

System.Resources.ResourceManager rm = new System.Resources.ResourceManager("TestApp.string", Assembly.GetExecutingAssembly());

Here in the first parameter: TestApp is the name of your Windows Application and string is the name of the Resource file part before the language code. Second parameter is the Main Assembly for the resources (for this you have to add the namespace: using System.Reflection;).

6. Write the following function for the culture:

Hide   Copy Code

private void ChangeCulture(string sLangCode)

{

Thread.CurrentThread.CurrentUICulture = new CultureInfo(sLangCode);

Thread.CurrentThread.CurrentCulture = CultureInfo.CreateSpecificCulture(sLangCode);

lblMultiLangResult.Text = rm.GetString("lblResult");

}

For the above function to run, please add the following namespaces:

Hide   Copy Code

using System.Threading; *//For Threading*

using System.Globalization; *//For CultureInfo*

7. On the ComboBox item change event, add the following code:

Hide   Copy Code

private void cmbLanguage\_SelectedIndexChanged(object sender, EventArgs e)

{

string sLangCode;

if (cmbLanguage.SelectedIndex == 0)

{

sLangCode = "en-US";

ChangeCulture(sLangCode);

}

else if (cmbLanguage.SelectedIndex == 1)

{

sLangCode = "he";

ChangeCulture(sLangCode);

}

else if (cmbLanguage.SelectedIndex == 2)

{

sLangCode = "ar-sa";

ChangeCulture(sLangCode);

}

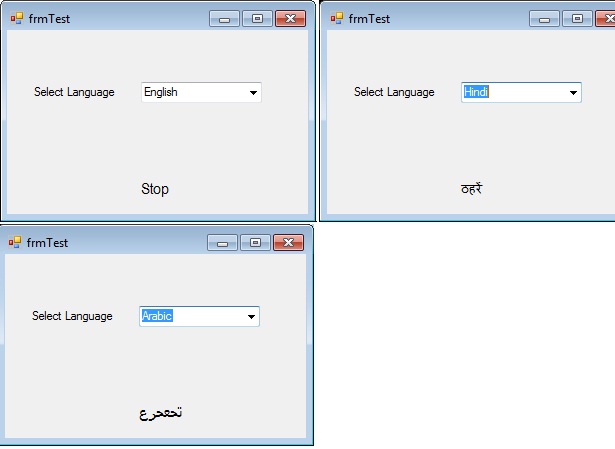
}

8. On page load, add the following line of code:

Hide   Copy Code

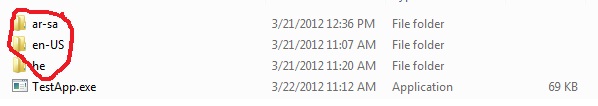
ChangeCulture("en-US");

9. Run the application and see the output as below:



## Points of Interest

When you look at ApplicationFolder/bin/release (check bin/debug if you are running in debug mode), there are three folders containing the same name of the DLL but each for different culture. The correct one will load when the user selects one of them.



**6. what is difference between private and shared assembly?**

**7. what is code access security?**

**8. what is early and late binding in c#?**

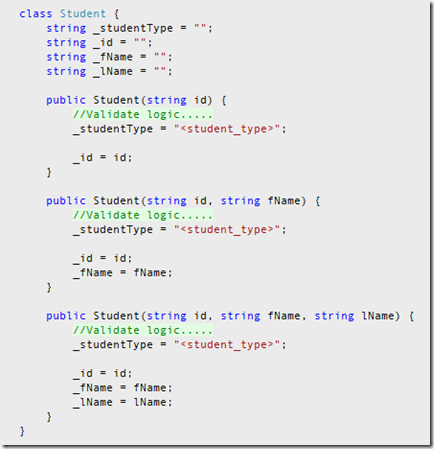
**9. what is this keyword in c#? where will be use other than the object?**

**10. what is constructor chaining in c#?**

### What is Constructor Chaining?

Constructor Chaining is an approach where a constructor calls another constructor in the same or base class.

This is very handy when we have a class that defines multiple constructors. Assume we are developing a class Student. And this class has three constructors. On each constructer, we have to validate the student's ID and categorize him/her. So if we do not use the constructor chaining approach, it would be something similar to what is shown below:

[](http://lh5.ggpht.com/_zprglM1x5x4/TQB0_UpDl7I/AAAAAAAAANU/f-qvk3CVT_Y/s1600-h/screen_01%5b4%5d.png)

Even though the above approach solves our problem, it duplicates code. (We are assigning a value to ‘\_id’ in all our constructors). This is where constructor chaining is very useful. It will eliminate this problem. This time, we only assign values in one constructor which consists of the most number of parameters. And we call that constructor when the other two constructers are called.

Hide   Copy Code

class Student {

string \_studentType = "";

string \_id = "";

string \_fName = "";

string \_lName = "";

public Student(string id)

: this(id, "", "") {

}

public Student(string id, string fName)

: this(id, fName, "") {

}

public Student(string id, string fName, string lName) {

*//Validate logic.....*

\_studentType = "<student\_type>";

\_id = id;

\_fName = fName;

\_lName = lName;

}

}

**\*\*Please note**: If you do not specify anything [in this example, we used ‘this’), it will be considered that we are calling the constructor on the base class. And it’s similar to using ‘: base(…)’].

In the following example, the Person[class](https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/class) defines a copy constructor that takes, as its argument, an instance of Person. The values of the properties of the argument are assigned to the properties of the new instance of Person. The code contains an alternative copy constructor that sends the Name and Age properties of the instance that you want to copy to the instance constructor of the class.

C#Copy

class Person

{

// Copy constructor.

public Person(Person previousPerson)

{

Name = previousPerson.Name;

Age = previousPerson.Age;

}

//// Alternate copy constructor calls the instance constructor.

//public Person(Person previousPerson)

// : this(previousPerson.Name, previousPerson.Age)

//{

//}

// Instance constructor.

public Person(string name, int age)

{

Name = name;

Age = age;

}

public int Age { get; set; }

public string Name { get; set; }

public string Details()

{

return Name + " is " + Age.ToString();

}

}

class TestPerson

{

static void Main()

{

// Create a Person object by using the instance constructor.

Person person1 = new Person("George", 40);

// Create another Person object, copying person1.

Person person2 = new Person(person1);

// Change each person's age.

person1.Age = 39;

person2.Age = 41;

// Change person2's name.

person2.Name = "Charles";

// Show details to verify that the name and age fields are distinct.

Console.WriteLine(person1.Details());

Console.WriteLine(person2.Details());

// Keep the console window open in debug mode.

Console.WriteLine("Press any key to exit.");

Console.ReadKey();

}

}

// Output:

// George is 39

// Charles is 41

**11. when will be use delegates in c#?**

**12. what is eventbubbling in javascript?**

**Event bubbling** is a term you might have come across on your **JavaScript** travels. It relates to the order in which **event** handlers are called when one element is nested inside a second element, and both elements have registered a listener for the same **event**

**13. what is javascript:void(0)?**

**14. what is call and apply javascript?**

**15. what is diff b/w var and let keyword in JQuery?**

**17. what is difference between var and dynamic in c# ?**

**18. what is difference between constant and readonly ?**

**19. what is type nullable ?**

**Thomson Router**

**----------------------------------------------------**

**1. how to inject two classes into one interface on demand in dependency injection?**

**2. where we load xml ?**

**XmlDocument xmlDoc = new XmlDocument();**

**xmlDoc.Load("d:\\product.xml");**

**XmlNodeList nodeList = xmlDoc.DocumentElement.SelectNodes("/Table/Product");**

**string proID = "", proName = "", price="";**

**foreach (XmlNode node in nodeList)**

**{**

**proID = node.SelectSingleNode("Product\_id").InnerText;**

**proName = node.SelectSingleNode("Product\_name").InnerText;**

**price = node.SelectSingleNode("Product\_price").InnerText;**

**MessageBox.Show(proID + " " + proName + " " + price);**

**}**

**3. how to get default values from Generics?**

**You are looking for this:**

**default(T);**

**so:**

**public T Foo<T>(T Bar)**

**{**

**return default(T);**

**}**

**4. open closed principle?**

**5. what is reflection ?**

**var assemblies = Assembly.LoadFile(@args[0]).GetReferencedAssemblies();**

**if (assemblies.GetLength(0) > 0)**

**{**

**foreach (var assembly in assemblies)**

**{**

**Console.WriteLine(assembly);**

**}**

**return 0;**

**}**

1. **how to make synchronous call in angular?**
2. **Tell me about your last Project?**
3. **Tell me about route.config? How does it work and what is the life cycle?**
4. **why JQery is better than Java Script**
5. **Try-Catch in base and derived Concept??**
6. **Where you used oops in your project & Explain?**
7. **How to handle exceptions raised in sql server & show then in UI**
8. **There are many controls in a MVC Page.How to check whether they are valid or not?**
9. **Model State is Client Side or Servier Side?**
10. f an asp.net is taking time to load. How will you check where is the issue?
11. We gave an intranet application. Which binding will you use?
12. Why we use TCP instead of HTTP?
13. How to enable metadata in WCF?where will u write this?
14. How to create Proxy?
15. Difference between Custom Controls & User Controls?
16. Partial View in MVC?
17. Validations in ASP.Net
18. **What are the challenges did you faced in your Project?**
19. Difference between readonly & Constant?
20. Var Keyword in C#?
21. How you implemented Inheritance in your Project? Syntax
22. In ASP.Net, how you achieved interface in your Project?
23. Schema in Sql. where the Schema will be stored?
24. What is dbo?
25. Catching in ASP.Net?
26. IEnumerable in Linq?
27. **How to find Index value in Linq?**
28. **Error handling in Stroed Precedure?**
29. **Stored Procedure is taking too much time.How to solve the Problem?**
30. Difference between table variable & Temp table? performance wise which one is better and which scenario you will go table variable?
31. What is generics?
32. Difference between Is and As?
33. ASP.Net Vs MVC.
34. Where session is saved?What is the Mechanism in ASP.Net?How do you implement session MVC?
35. Action Filters in MVC?
36. Model binder in MVC? When do you implement and what are the advantages?
37. **Validations in MVC? without ASP.Net validations how will you implement in MVC?**
38. Full form of LINQ? Advantages of LINQ
39. **Advantages of IEnumerable?**
40. How do you implement JQery in ASP.Net?
41. What is clustered index?
42. Tuning in sql?
43. Why you go for Stored Procedure?
44. **Two columns are unique then can you created clustered index?**
45. Dependency injection in MVC?
46. Areas in MVC?
47. How to access Master Page control in content page?
48. Asp.net Page life cycle?
49. Unit Testing
50. Razor in MVC?
51. ASP.Page is slow then how will you identify that this issue is from only asp.net? How will you improve performance of it?
52. Difference between IEnumerable and IQueriable? why do you use IEnumerable?
53. **What types of conversion related issue have you faced in LINQ?**
54. What is WCF? What are the various ways to create the Proxy?
55. You have a gridview and inside the gridview you have a textbox then how will you get the e from textbox and how will you assign the value to textbox?
56. when do you use which indexing?expalain with scenario?
57. **Table is not available in database then can we write select \* from tablename?**
58. How do you check using the Store procedure that table is available in database or not?
59. what is table hint? expalin with example?

1. Difference between IQueryable, IEnumerable and List?

2. How to get a master page control value into the aspx page, give code?

3. How to set different timeout for different roles?

4. What are different joins in SQL Server? Explain all

5. What is Restful service? What is model binding in web api? What is content negotiation in web api?

**Prokarma**

**1.Which is best option to convention routing or attribute routing?**

Both options are valid. Following are some suggestions on when to use each one. Consider choosing traditional routes when:

* You want centralized configuration of all your routes.
* You use custom constraint objects.
* You have an existing working application you don't want to change

Consider choosing attribute routes when:

* You want to keep your routes together with your action's code
* You are creating a new application or making significant changes to an existing.
* You want to match route parameter names with an actual parameter of the method(action), this will make the route parameter more descriptive and omit the weird error where the route ID would not match and that usually happens because we didn't configure routing correctly and we were using asp.net default route.

But the reasons why many developers recommend the **Attribute Routes** is because it allows you quite a bit more flexibility and places the routes next to the actions that will actually use them. You can switch from option to another at any time is not difficult

**2.can we override action method in mvc?**

As far as i m understanding your question these are the answers :

**First Answer :**

it's not possible to have two controller actions with the same name but with a different result also:

For example:

ActionResult YourAction() { ... }

FileContentResult YourAction() { ... }

In MVC you can also do this :

[HttpGet]

[ActionName("AnyAction")]

ActionResult YourAction(firstModel model1) { ... }

[HttpPost]

[ActionName("AnyAction")]

FileContentResult YourAction(secondModel model1) { ... }

The main idea here is that you can use the ActionNameAttribute to name several action methods with the same name.

----------------------------------------------------------------OR--------------------------------------------------------------

**Second Answer :**

[NonAction]

public override ActionResult YourAction(FormCollection form)

{

// do nothing or throw exception

}

[HttpPost]

public ActionResult YourAction(FormCollection form)

{

// your implementation

}

**3.Difference between the mvc and web APi?**

Similarities

1) both inherits from ihhtphandler for the asyncrequest so basically apicontroller or mvc controller both are the wrapper around the web.http

Differences: 1) mvc controller is very heavy if you could go through its definition you can see how many interfaces and the base code it has used, web api is lighter controller and distinguish request by its passed parameters ( yes we can change it too!)

2) MVC controller has too many features like it return views, action result , javascript result etc but in web api has either JSON or XML

3) API is for implementing Restful(get, post,put, delete, options) services which can independently can hosted any where without the depending upon views, MVC controller cant support that as it tightly integrated with the views.

1. [Asp.Net MVC](https://en.wikipedia.org/wiki/ASP.NET_MVC) is used to create web applications that return both views and data but Asp.Net Web API is used to create full-blown HTTP services with an easy and simple way that returns only data, not view.
2. Web API helps to build REST-ful services over the .NET Framework and it also supports content-negotiation(it's about deciding the best response format data that could be acceptable by the client. it could be JSON, XML, ATOM or other formatted data), self-hosting which are not in MVC.
3. Web API also takes care of returning data in particular format like JSON, XML or any other based upon the Accept header in the request and you don't worry about that. MVC only return data in JSON format using JsonResult.
4. In Web API the request is mapped to the actions based on HTTP verbs but in MVC it is mapped to actions name.
5. Asp.Net Web API is a new framework and part of the core ASP.NET framework. The model binding, filters, routing, and other MVC features exist in Web API are different from MVC and exists in the new System.Web.Http assembly. In MVC, these features exist within.System.Web.Mvc Hence Web API can also be used with Asp.Net and as a stand-alone service layer.
6. You can mix Web API and MVC controller in a single project to handle advanced AJAX requests which may return data in JSON, XML or any others format and building a full-blown HTTP service. Typically, this will be called Web API self-hosting.
7. When you have mixed MVC and Web API controller and you want to implement the authorization then you have to create two filters one for MVC and another for Web API since both are different.
8. Moreover, Web API is lightweight architecture and except the web application, it can also be used with smartphone apps.

**Difference between web api routes and mvc routing?**

For Web API it is:

config.Routes.MapHttpRoute(

name: "DefaultApi",

routeTemplate: "api/{controller}/{id}",

defaults: new { id = RouteParameter.Optional }

);

For MVC, it is:

routes.MapRoute(

name: "Default",

url: "{controller}/{action}/{id}",

defaults: new { controller = "Home", action = "Index", id = UrlParameter.Optional }

);

If you are familiar with ASP.NET MVC, Web API routing is very similar to MVC routing. The main difference is that Web API uses the HTTP verb, not the URI path, to select the action. You can also use MVC-style routing in Web API. This article does not assume any knowledge of ASP.NET MVC.

**Can we overload mvc action methods?**

[HttpPost]

[ActionName("ProcessForm1")]

public ActionResult ProcessForm(Customer obj)

{

return View();

}

[HttpPost]

[ActionName("ProcessForm2")]

public ActionResult ProcessForm(Employee emp)

{

return View();

}

public ActionResult Index()

{

return View();

}

[HttpPost]

public ActionResult Index(Customer obj)

{

//do something with obj

return View();

}

[Route("Home/DoWork/{status:int}")]

public ActionResult DoWork(int status)

{

return View();

}

[Route("Home/DoWork/{flag:bool}")]

public ActionResult DoWork(bool flag)

{

return View();

}

[HttpPost]

public ActionResult Calculate()

{

int i = int.Parse(Request.Form["num1"]);

int j = int.Parse(Request.Form["num2"]);

return Calculate(i , j);

}

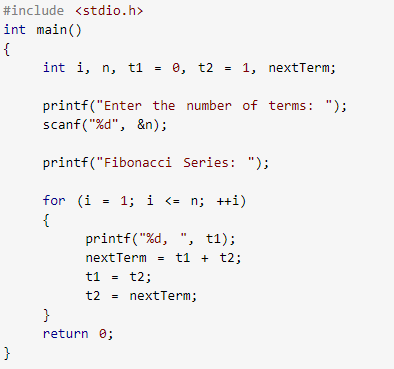
[NonAction]

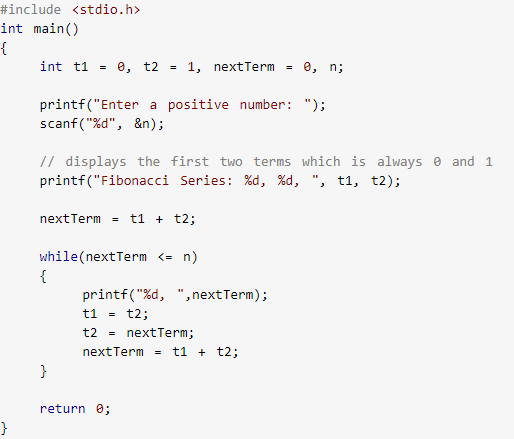
public ActionResult Calculate(int i, int j)

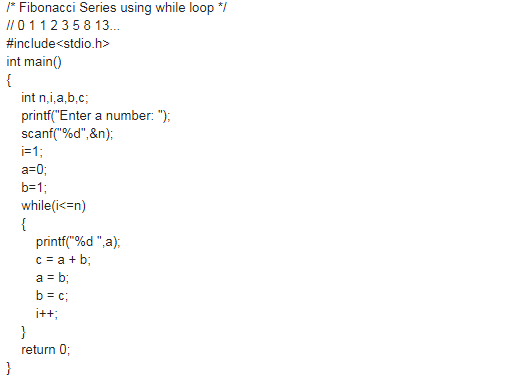
{

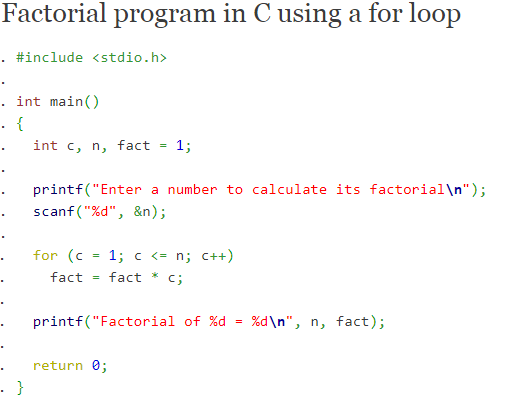
return View( i + j );

}

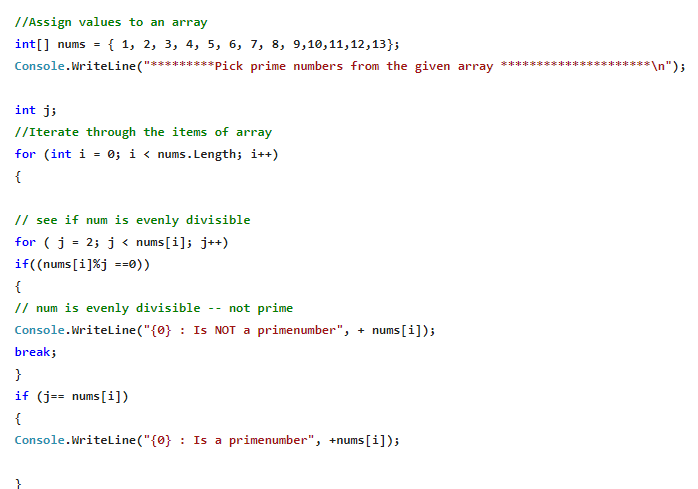
**Write program fibonacci series?**

****

****

****

**Extract prime numbers from given integer arrays**

****

**Delta Technology:**

## What is an angular interceptors?

# HttpInterceptor

INTERFACE

Intercepts and handles an [HttpRequest](https://angular.io/api/common/http/HttpRequest) or [HttpResponse](https://angular.io/api/common/http/HttpResponse).

[See more...](https://angular.io/api/common/http/HttpInterceptor#description)

interface [HttpInterceptor](https://angular.io/api/common/http/HttpInterceptor) {

[intercept(req: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>>](https://angular.io/api/common/http/HttpInterceptor#intercept)

}

Most interceptors transform the outgoing request before passing it to the next interceptor in the chain, by calling next.handle(transformedReq). An interceptor may transform the response event stream as well, by applying additional RxJS operators on the stream returned by next.handle().

More rarely, an interceptor may handle the request entirely, and compose a new event stream instead of invoking next.handle(). This is an acceptable behavior, but keep in mind that further interceptors will be skipped entirely.

It is also rare but valid for an interceptor to return multiple responses on the event stream for a single request.

|  |
| --- |
| Identifies and handles a given HTTP request. |
| intercept(req: [HttpRequest](https://angular.io/api/common/http/HttpRequest)<any>, next: [HttpHandler](https://angular.io/api/common/http/HttpHandler)): Observable<[HttpEvent](https://angular.io/api/common/http/HttpEvent)<any>> **Parameters**  |  |  |  | | --- | --- | --- | | **req** | [HttpRequest](https://angular.io/api/common/http/HttpRequest) | The outgoing request object to handle. | | **next** | [HttpHandler](https://angular.io/api/common/http/HttpHandler) | The next interceptor in the chain, or the backend if no interceptors remain in the chain. |  **Returns** Observable<[HttpEvent](https://angular.io/api/common/http/HttpEvent)<any>>: An observable of the event stream. |

## Usage notes

To use the same instance of HttpInterceptors for the entire app, import the [HttpClientModule](https://angular.io/api/common/http/HttpClientModule) only in your AppModule, and add the interceptors to the root application injector . If you import [HttpClientModule](https://angular.io/api/common/http/HttpClientModule) multiple times across different modules (for example, in lazy loading modules), each import creates a new copy of the [HttpClientModule](https://angular.io/api/common/http/HttpClientModule), which overwrites the interceptors provided in the root module.

<https://alligator.io/angular/httpclient-interceptors/>

## what is difference between Environment.ts and prod.ts in angular?

"configurations": {

"production": { ... },

"staging": {

"fileReplacements": [

{

"replace": "src/environments/environment.ts",

"with": "src/environments/environment.staging.ts"

}

]

}

}

**environment.ts** If you refer environment object properties in your Angular project, during development mode i.e. ng serve or ng build all values shall be read from this file.

**environment.prod.ts** When you build your application for production mode using ng build --prod in that case, all values of environment.ts file shall get overridden by environment.prod.ts files.

The above variables are related to your Angular application. Whereas .env and .env.example are for the Laravel application.

**.env** as it should be is out of version controlled and ignored when you push your project to any repository. This is for your own safety.

**.env.example** which contains very generic information is copied as .env on fresh install and a few changes made. ex APP\_KE

## Exception filters in mvc what is usage?

## Exception handling in ASP.NET MVC (6 methods explained)

## Introduction

In this [Asp.net MVC step by step](https://www.learnmvc.in/) article we have discuss 6 ways of handling exceptions in ASP.NET MVC.In this article we also talk about best practices of MVC exception handling.

## Method 1:- Simple way

The simplestwayis to use the traditional .NET exception handling style i.e. try and catch block. Now when exception happens catch block gets executed and it redirects to the error view.

But if we use this method then we will not be utilizing MVC exception mechanismproperly and completely. In the further sections we will discuss five important ways by which we can utilize MVC provided features for exception handling.

Hide   Copy Code

public ActionResult SomeError()

{

try

{}

catch(Exception ex)

{return View("Error");}

}

## Method 2:- Override “OnException” method

In this method we can override the “OnException” event of the controller and set the “Result” to the view name. This view gets invoked when error occurs in this controller. In the below code you can see we have set the “Result” to a view named as “Error”.

We have also set the exception so that it can be displayed inside the view.

Hide   Copy Code

public class HomeController : Controller

{

protected override void OnException(ExceptionContext filterContext)

{

Exception ex = filterContext.Exception;

filterContext.ExceptionHandled = true;

var model = new HandleErrorInfo(filterContext.Exception, "Controller","Action");

filterContext.Result = new ViewResult()

{

ViewName = "Error",

ViewData = new ViewDataDictionary(model)

};

}

}

To display the above error in view we can use the below code:-

Hide   Copy Code

@Model.Exception;

The problem with this approach is we cannot reuse the error handling logic across multiple controllers.

## Method 3:- Using “HandleError” Attribute

The other way of handling error is my using “HandleError” attribute. Implementing “HandleError” attribute is a two-step process:-

Step 1 :- We need to first decorate the action method with “HandleError” attribute as shown in the below code.

Hide   Copy Code

public class HomeController : Controller

{

[HandleError()]

public ActionResult SomeError()

{

throw new Exception("test");

}

}

Step 2:- In the “Web.config” file you need to add the “customErrors” tag and point to the “Error” view as shown in the below “Web.config” code snippet.

Hide   Copy Code

<system.web>

<customErrors defaultRedirect="Error.cshtml" mode="On">

</customErrors>

</system.web>

In case you want different error views for different exception types you can decorate action method with multiple “HandleError” attribute point to multiple views as per exception types.

Hide   Copy Code

public class HomeController : Controller

{

[HandleError(ExceptionType=typeof(ArithmeticException),View="Arthimetic")]

[HandleError(ExceptionType = typeof(NotImplementedException),View ="Error1")]

public ActionResult SomeError()

{

}

}

## Method 4:- Inheriting from “HandleErrorAttribute”

One of the biggest drawbacks of all the previous method was reusability. Error handling logic cannot be reused across other controllers.

In order to reuse error handling logic across controller we can inherit from “HandleErrorAttribute”class anddecorate this class as attribute across controller.

Hide   Copy Code

public class Err : HandleErrorAttribute

{

public override void OnException(ExceptionContext filterContext)

{

Exception ex = filterContext.Exception;

filterContext.ExceptionHandled = true;

var model = new HandleErrorInfo(filterContext.Exception, "Controller", "Action");

filterContext.Result = new ViewResult()

{

ViewName = "Error1",

ViewData = new ViewDataDictionary(model)

};

}

}

## Method 5:- Handling HTTP errors

All MVC exception handling techniques discussed till now do not handle HTTP errors like file not found, HTTP 500 error’s etc. For that we need to make an entry of the error action and the error status code as shown in the below config file.

Hide   Copy Code

<system.web>

<customErrors

mode="On" defaultRedirect="Error1">

<error statusCode="404" redirect="~/Testing/NoPageFound"/>

</customErrors>

</system.web>

## Method 6:- Global Error handling in MVC

If you wish to do global error handling across your application you can override the “Application\_Error” event and do a response.redirect from the global error event. So if the error handling is not done at the controller level it will get propagated to “Global.asax” file.

Hide   Copy Code

public class MvcApplication : System.Web.HttpApplication

{

protected void Application\_Error(object sender, EventArgs e)

{

Exception exception = Server.GetLastError();

Server.ClearError();

Response.Redirect("/Home/Error");

}

}

## What’s the best practice ?

The best is combination of “Method 4” and “Method 6”. Create error handling classes which inherit from “HandleErrorAttribute” class and decorate them respectively on controllers and action methods. So this takes care of errors happening on controllers and actions.

As a safety enable Global error handling as a fallback for any unexpected and unhandled errors byusing “Application\_Error” event as described in “Method 6”.

## What is the filters in mvc and types of filters?

## Dependency injection what are the steps you need to follow?

## Method overloading and its benefits?

## The advantage of method overloading is that it increases the readability of the program because you don't need to use different names for same action. You can perform method overloading in C# by two ways: By changing number of arguments.

## No need to consider the returntype in method overloading if it is consider same signature different returntype it is showing compilation error.

## Palidrom belongs to which data structure?

## ViewData store at clientside or serverside?

## Difference b/w authorization and authentication?

## How to validate entity models?

You can create a partial class, separate from the EF generated class, to store the metadata in.

//Contact.cs - The original auto-generated file

[System.ComponentModel.DataAnnotations.MetadataType(typeof(ContactMetadata))]

public partial class Contact

{

public int ContactID { get; set; }

public string ContactName { get; set; }

public string ContactCell { get; set; }

}

//ContactMetadata.cs - New, seperate class

using System.ComponentModel;

using System.ComponentModel.DataAnnotations;

internal sealed class ContactMetadata

{

[Required(ErrorMessage = "Name is required.")]

[StringLength(5)]

public string ContactName;

}

## How to check entity frame values at runtime?

## AmpleLogic

## What is schema and its usage in sql server?

* A database schema is a way to logically group objects such as tables, views, stored procedures etc. Think of a schema as a container of objects.
* You can assign a user login permissions to a single schema so that the user can only access the objects they are authorized to access.
* Schemas can be created and altered in a database, and users can be granted access to a schema. A schema can be owned by any user, and schema ownership is transferable.

Advantages of using Schema

* Act as object protection tool: A schema can be a very effective object projection tool combined with the appropriate level of user permissions. A DBA can maintain control access to an object that would be very crucial.
* Managing a logical group of database objects within a database: Schemas allow database objects to be organized into a logical group. This would be advantagous when multiple teams are working on the same database application and the design team wants to maintain integrity of the database tables.
* Easy to maintain the database: A schema allows a logical grouping of the database objects, so the schema can help us in situations where the database object name is the same but falls in a different logical group.

Other Advantages

* A single schema can be shared among multiple databases and database users.
* A database user can be dropped without dropping database objects.
* Manipulation of and access to the object is now very complex and more secure. The schema acts as an additional layer of security.
* Database objects can be moved among schemas.
* The ownership of schemas is transferable.

Conclusion

A schema is a very useful database concept and helps us to separate database users from the database object owners and also helps to create a logical grouping of database objects.

## What is synonyms in sql server?

A synonym is a database object that serves the following purposes:

* Provides an alternative name for another database object, referred to as the base object, that can exist on a local or remote server.
* Provides a layer of abstraction that protects a client application from changes made to the name or location of the base object.

## Design Pattens are used in your project?

## Dependency injection usages and how to create?

UnityConfig- Start

using Domos.Data.Infrastructure;

using Domos.Web.Controllers;

using Microsoft.Practices.Unity;

using System.Linq;

using System.Web.Mvc;

using Unity.Mvc5;

namespace Domos.Web

{

public static class IocExtensions

{

/// <summary>

/// Register the type with Hierachical life time

/// </summary>

/// <typeparam name="T1">Type</typeparam>

/// <typeparam name="T2">Map To</typeparam>

/// <param name="container">container object</param>

public static void BindInRequestScope<T1, T2>(this IUnityContainer container) where T2 : T1

{

container.RegisterType<T1, T2>(new HierarchicalLifetimeManager());

}

/// <summary>

/// Register the type with Container controlled life time

/// </summary>

/// <typeparam name="T1">Type</typeparam>

/// <typeparam name="T2">Map To</typeparam>

/// <param name="container">container object</param>

public static void BindInSingletonScope<T1, T2>(this IUnityContainer container) where T2 : T1

{

container.RegisterType<T1, T2>(new ContainerControlledLifetimeManager());

}

}

public static class UnityConfig

{

private const string RepositoryTypeEndsWith = "Repository";

private const string ServiceTypeEndsWith = "Service";

public static void RegisterComponents()

{

BuildUnityContainer();

}

/// <summary>

/// Build unity container, register types and resolve

/// </summary>

private static void BuildUnityContainer()

{

var container = new UnityContainer();

//Register the Unit of work type

container.BindInRequestScope<IUnitOfWork, UnitOfWork>();

//Register the DbFactory type

container.BindInRequestScope<IDbFactory, DbFactory>();

//TODO: Will be removed below lins of code after testing done

container.RegisterType<AccountController>(new InjectionConstructor());

// Register the Repositories types

container.RegisterTypes(

AllClasses.FromLoadedAssemblies().Where(t => t.Name.EndsWith(RepositoryTypeEndsWith)),

WithMappings.FromMatchingInterface,

WithName.Default,

WithLifetime.Hierarchical);

//Register the Services types

container.RegisterTypes(AllClasses.FromLoadedAssemblies().Where(t => t.Name.EndsWith(ServiceTypeEndsWith)),

WithMappings.FromMatchingInterface,

WithName.Default,

WithLifetime.Hierarchical);

//Resolve the configured depencies

DependencyResolver.SetResolver(new UnityDependencyResolver(container));

}

}}UnityConfig- End

## Solid principles?

## Output caching in mvc?

* Hence you can take advantage of the output cache to avoid executing a database query every time a user invokes the same controller action. In this case, the view will be retrieved from the cache instead of being regenerated from the controller action.
* Caching enables you to avoid performing redundant work on the server.
* Let’s take a look at a simple example of caching in our project.
* [OutputCache(Duration = 60)]
* public ActionResult Index(){
* var employees = from e in db.Employees
* orderby e.ID
* select e;
* return View(employees);
* }

Cache Profile

You can create a cache profile in the web.config file. It is an alternative to configuring output cache properties by modifying properties of the [OutputCache] attribute. It offers a couple of important advantages which are as follows.

* Controls how controller actions cache content in one central location.
* Creates one cache profile and apply the profile to several controllers or controller actions.
* Modifies the web configuration file without recompiling your application.
* Disables caching for an application that has already been deployed to production.

Let’s take a look at a simple example of cache profile by creating the cache profile in web.config file. The <caching> section must appear within the <system.web> section.

<caching>

<outputCacheSettings>

<outputCacheProfiles>

<add name = "Cache10Min" duration = "600" varyByParam = "none"/>

</outputCacheProfiles>

</outputCacheSettings>

</caching>

You can apply the Cache10Min profile to a controller action with the [OutputCache] attribute which is as follows.

[OutputCache(CacheProfile = "Cache10Min")]

public ActionResult Index(){

var employees = from e in db.Employees

orderby e.ID

select e;

return View(employees);

}

## how to resolve performance issues in your project?

1. Implement Gzip.
2. Use asynchronous rendering for partial views.
3. Minimize database hits.
4. Use a compiled query.
5. Run a profiler and find out unnecessary hits. Optimize all stored procedures which are taking more than 1 second to return a response.
6. Use caching.
7. Use [bundling minification](http://www.asp.net/mvc/overview/performance/bundling-and-minification) optimization.
8. Use HTML 5 utilities like session cache and local storage for readonly contents.

+50

A compiled list of possible sources of improvement are below:

**General**

* Make use of a profiler to discover memory leaks and performance problems in your application. personally I suggest [dotTrace](http://www.jetbrains.com/profiler/)
* Run your site in Release mode, not Debug mode, when in production, and also during performance profiling. Release mode is much faster. Debug mode can hide performance problems in your own code.

**Caching**

* Use CompiledQuery.Compile() recursively avoiding recompilation of your query expressions
* Cache not-prone-to-change content using OutputCacheAttribute to save unnecessary and action executions
* Use cookies for frequently accessed non sensitive information
* Utilize [ETags](http://www.google.co.uk/search?q=asp.net+mvc+etag) and expiration - Write your custom ActionResult methods if necessary
* Consider using the RouteName to organize your routes and then use it to generate your links, and try not to use the expression tree based ActionLink method.
* Consider implementing a route resolution caching strategy
* Put repetitive code inside your PartialViews, avoid render it xxxx times: if you end up calling the same partial 300 times in the same view, probably there is something wrong with that. [Explanation And Benchmarks](http://codeclimber.net.nz/archive/2009/04/17/how-to-improve-the-performances-of-asp.net-mvc-web-applications.aspx)

**Routing**

* Use Url.RouteUrl("User", new { username = "joeuser" }) to specify routes. [ASP.NET MVC Perfomance by Rudi Benkovic](http://www.slideshare.net/rudib/aspnet-mvc-performance)
* Cache route resolving using this helper UrlHelperCached [ASP.NET MVC Perfomance by Rudi Benkovic](http://www.slideshare.net/rudib/aspnet-mvc-performance)

**Security**

* Use Forms Authentication, Keep your frequently accessed sensitive data in the authentication ticket

**DAL**

* When accessing data via LINQ [rely on IQueryable](https://stackoverflow.com/questions/1106802/why-use-asqueryable-instead-of-list)
* [Leverage the Repository pattern](https://stackoverflow.com/questions/1223194/loading-subrecords-in-the-repository-pattern)
* Profile your queries i.e. [Uber Profiler](http://hibernatingrhinos.com/products/UberProf)
* Consider second level cache for your queries and add them an scope and a timeout i.e. [NHibernate Second Cache](http://nhibernate.info/blog/2009/02/09/quickly-setting-up-and-using-nhibernate-s-second-level-cache.html)

**Load balancing**

* Utilize reverse proxies, to spread the client load across your app instance. (Stack Overflow uses [HAProxy](http://haproxy.1wt.eu/) ([MSDN](http://www.microsoft.com/casestudies/Case_Study_Detail.aspx?casestudyid=4000006676)).
* Use [Asynchronous Controllers](http://msdn.microsoft.com/en-in/library/ee728598(v=vs.98).aspx) to implement actions that depend on external resources processing.

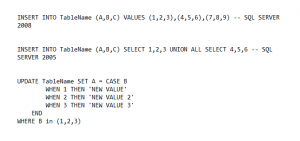
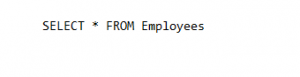
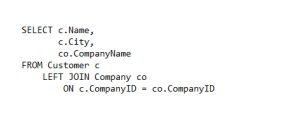
**Client side**

* Optimize your client side, use a tool like [YSlow](http://developer.yahoo.com/yslow/) for suggestions to improve performance
* Use AJAX to update components of your UI, avoid a whole page update when possible.
* Consider implement a pub-sub architecture -i.e. Comet- for content delivery against reload based in timeouts.
* Move charting and graph generation logic to the client side if possible. Graph generation is a expensive activity. Deferring to the client side your server from an unnecessary burden, and allows you to work with graphs locally without make a new request (i.e. Flex charting, [jqbargraph](http://www.workshop.rs/jqbargraph/), [MoreJqueryCharts](http://www.reynoldsftw.com/2009/02/6-jquery-chart-plugins-reviewed/)).
* Use CDN's for scripts and media content to improve loading on the client side (i.e. [Google CDN](http://code.google.com/apis/ajaxlibs/documentation/))
* Minify -[Compile](http://code.google.com/closure/compiler/)- your JavaScript in order to improve your script size
* Keep cookie size small, since cookies are sent to the server on every request.
* Consider using [DNS and Link Prefetching](http://en.wikipedia.org/wiki/Link_prefetching) when possible.

**Global configuration**

* If you use Razor, add the following code in your global.asax.cs, by default, Asp.Net MVC renders with an aspx engine and a razor engine. This only uses the RazorViewEngine.

ViewEngines.Engines.Clear(); ViewEngines.Engines.Add(new RazorViewEngine());

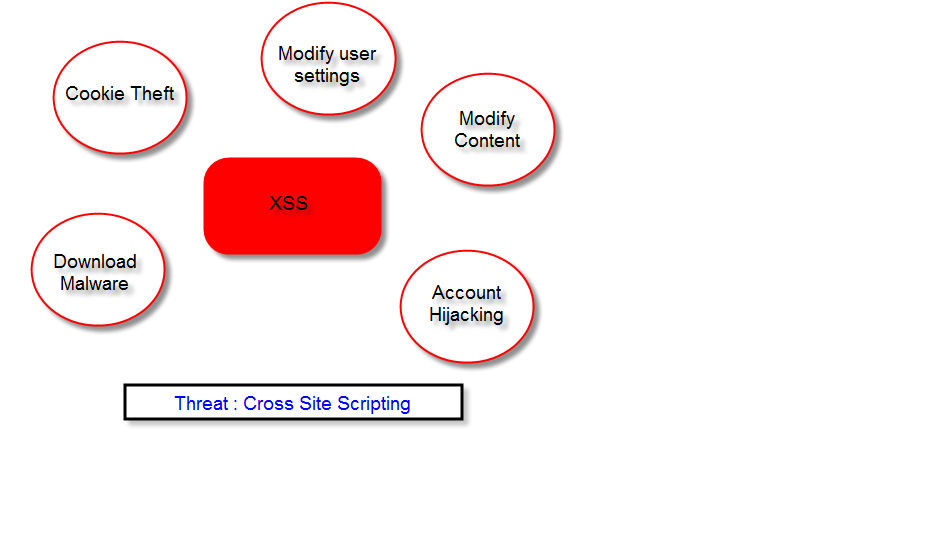
* Add gzip (HTTP compression) and static cache (images, css, ...) in your web.config <system.webServer> <urlCompression doDynamicCompression="true" doStaticCompression="true" dynamicCompressionBeforeCache="true"/> </system.webServer>
* Remove unused HTTP Modules
* Flush your HTML as soon as it is generated (in your web.config) and disable viewstate if you are not using it <pages buffer="true" enableViewState="false">
* Database tuning is both an art and a science. There is no right answer when it comes to tuning queries in an SQL Server. The issues you have solved on one system may not be a problem for another, and vice versa.
* Therefore, performance tuning in SQL Server can become an incredibly daunting task. Sometimes, even the minutest change can have a dramatic impact on the database performance. However, with a few tweaks in your approach, you can make slow SQL Servers go faster.
* Performance tuning aids in fixing queries that have been poorly written and indexes that are inefficient. The following techniques will give you a heads up and help you improve your query performance in a measurable way:
* **Improve Your Indexes**
* Indexes are data structures that increase the speed of data retrieval operations on a database table. Creating useful indexes is one of the most important steps for performance tuning in SQL Server. Useful indexes provide rapid random lookups and help you locate data that has fewer disk I/O operations and less system resource usage.
* It is important to understand the type of queries and the frequency with which they are run before you create efficient indexes. Try to index the major searching and ordering columns. However, indexing may hinder the performance of your database if the tables are constantly loaded with INSERT, UPDATE, and DELETE.
* **Separate Data and Log Files**
* You should keep the data and log files separated into different physical drives arrays when using DAS or SAN. Although this practice is often ignored, there is no good reason to overlook it.
* The main purpose of this practice is to separate the random access of data from the sequential access, which occurs while writing the transaction logs. This technique will amaze you with the difference it can make when the volume of your transactions increases.
* **Reduce the Use of Temp Tables**
* Temporary tables tend to increase the complexity of a query. If your application makes use of tempdb too often or creates too many temp tables, it may lead you to run into some contention related to internal structures that are associated with the tempdb files.
* When you absolutely need to use temp tables, create an index within the temp tables to improve performance. More importantly, do not wait for the temp tables to be deleted automatically rather delete it immediately after you are done with the table to clear tempdb resources.
* **Locate I/O Bottlenecks**
* I/O bottlenecks are one of the key factors that hinder performance in SQL Server. Detect the I/O issues by using one of the following techniques:
* Check to find high **page\_IO\_latch waits** or **log\_write waits** in your wait statistics.
* Use the **DMF sys.dm\_io\_virtual\_file\_stats()** to detect any areas where there is excessive physical I/O or excessive stalls on that I/O.
* Use the **trusty PerfMon**counters or use the **Avg. Disk sec/Read** and **Avg. Disk sec/Write** counters to check the latency of the reads and writes.
* Once you find the I/O bottlenecks, detect the queries that are contributing to the physical I/O, and try to tune them before adding more hardware.
* **Avoid Coding Loops for Performance Tuning in SQL Server**
* Imagine the following scenario in which about 1000 queries are directed to your database in sequence:
* 
* To tune the performance of your SQL Server, you should avoid such loops in your code. Instead, you can modify the code by using a unique INSERT or UPDATE statement with multiple rows and values:
* 
* Do keep it in mind to ensure that the **WHERE** clause does not update the stored value if it matches the existing value. This technique can dramatically enhance the performance of SQL Server.
* **Avoid the Use of SELECT**
* One important tip to optimize the performance tuning in SQL Server is to avoid using **SELECT \***. Instead, try to individually include the specific columns that you need. A query runs faster when there is fewer data to be retrieved. For instance, use the code –
* 
* in place of –
* http://www.springpeople.com/blog/wp-content/uploads/2017/04/code1_4-300x42.png
* A**void Correlated Subqueries**
* A correlated subquery uses values from the parent query. It usually runs row-by-row, once for each row returned by the outer query, thereby decreasing SQL query performance. The following is an example of a correlated subquery:
* 
* In the above example, the inner query (SELECT CompanyName…) is run for each row returned by the outer query (SELECT c.Name…). Why go over the same row again and again and slow down the SQL Server?
* Instead, refactor the correlated subquery as a join:
* 
* This lets the query go over the Company table just once, at the onset, and JOIN it with the Customer table.
* **Do Not Shrink Data Files**
* Shrinking data files can negatively impact the performance of an SQL Server. First of all, the process of shrinking can get very painful at times. It also causes a lot of fragmentation, which results in the poor performance of the subsequent queries. If Instant File Initialization is not turned on, the resultant growth later may potentially cause timeouts and can also affect the performance.
* At times shrinking a file might be absolutely necessary. However, do not forget to analyze the impact before you try it.
* We hope these do(s) and don’t(s) will help you when writing queries or processes and will aide you for performance tuning in SQL Server. However, remember to evaluate each situation individually to comprehend which method works best for your database.

## How to provide security in your application?

<https://www.codeproject.com/Articles/1116318/Points-to-Secure-Your-ASP-NET-MVC-Applications>

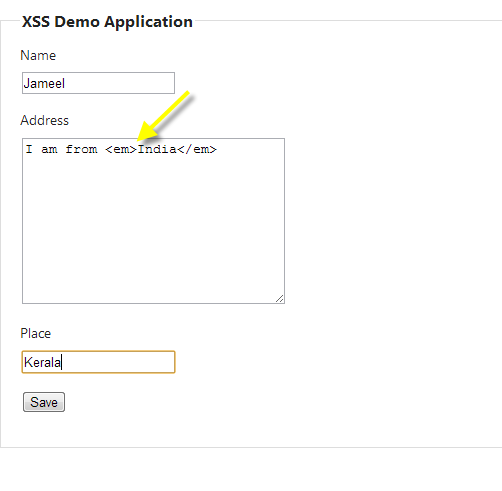
## XSS

There are some specific threats we will face. One popular attack of this phase is Cross Site scripting attack or XSS. In Cross scripting attack, the malicious user will try to have your website load a malicious script into the user’s browser. It could be a malicious script, Active-X control, and even some malicious HTML. The malicious scripts can theft the cookie, Modify user settings, Download Malware, or Modify content. One of the worst cross site script attack is Account Hijacking; the malicious users can access the user’s credentials and personal information.

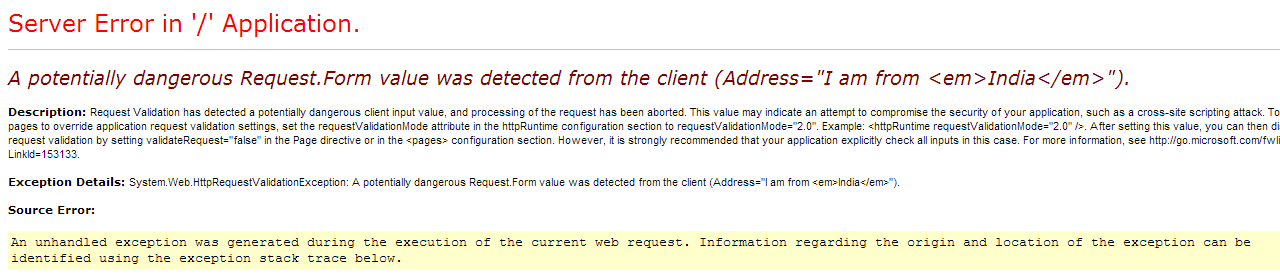


Once this happens, your users become vulnerable to any number of problems.

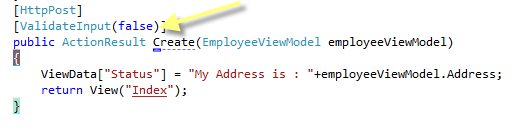
#### Demo



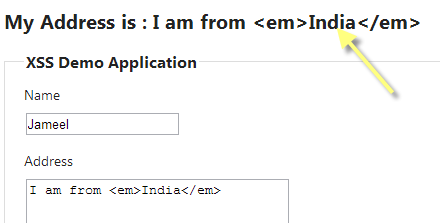
This is a simple application for saving employee information. I am putting some HTML tags like that I am from <em>India</em> and then when I try to save this, ASP.NET automatically rejects this request to prevent Cross site scripting attack because the ASP.NET is going to look for anything that resembles the HTML and just rejects the request. Actually, there is nothing wrong with the emphasis tag but ASP.NET is not trying to make a distinction here anything that looks like HTML is going to be rejected.



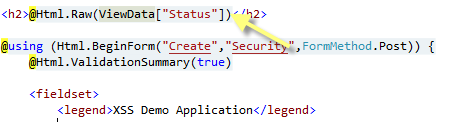
Sometimes, users need to upload some HTML to the server then there are always circumvents to this request validation. You have to be extremely careful. One option is put ValidationInput attribute to the destiny here in Create action.



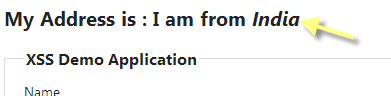
So you can successfully process this request.



Now we can have a problem that HTML encoded here. This is because razor is going to encode everything by default, which is good. There is another defense against the cross site scripting and we can fix that easily. However, the validate input false is completely disabling the check for cross site scripting, malicious HTML, and really we only need HTML inside of one specific property. So you can allow HTML to one property using the AllowHtml attribute. Also, some changes need to be made, remove ValidateInput attribute from the Create action and also make sure that we should pass the EmployeeViewModel class as an action parameter, that means model binding will take place and will move the HTML to that property. Also, one change in the view to show the HTML without encoding by putting ViewData in Html.Raw helper.

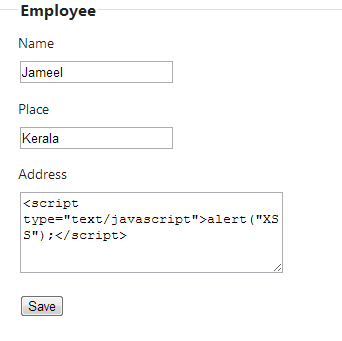


And then again, going to save one more and display the ViewData in the same view containing HTML tag.



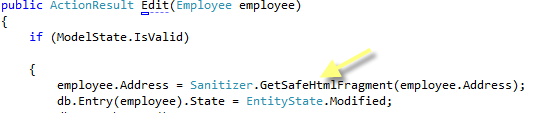
## Anti XSS Library

Someone comes to a form and enters some script like below:



It’s also more malicious. Fortunately, Microsoft provides a library for prevent this. You can download it via nugget or Library Package Manager Console (**Visual Studio>>Tools>>Library Package Manager>>Package Manager Console** and type Install-Package AntiXSS and press enter).

What I am going to do is I am putting a line of code in the below Edit action post method.



And this code will remove all the malicious things.



## Cross Site Request Forgery

Cross Site Request Forgery is a dangerous and an extremely major attack. Imagine a user comes into the site, trying to update some information that requires authentication before they are allowed to perform the update. Once the user logs in the Form Authentication, your site will send the users browser an authentication cookie and every subsequent request of the site, the users browser will send that cookie along and ASP.NET will see the user is already being authenticated. There is nothing wrong with the browser to sending the cookie along this is how the browser and cookie works - that means the user doesn’t need to enter the username and password in every single request they make. They authenticate themselves once and the cookie will allow them to remain authenticated at least for the duration of the session.

### Then What Is the Problem?

If the user visits some other site or is strict in picking up some HTML from a malicious source which had bad intentions, then this malicious source can provide a form, just like a form that our application would give to the user and then if the user submits the form, the call again will be authenticated because the authentication cookie be given to the users' browser. Always travel along every request and it will save the information into the database like we always do once we have authenticated the request. Only the information in the request probably is in something the user wants to submit. Someone tricked the user into transferring money or editing their account. The problem here is not to simply say we need the user to be authenticated when submitting some information. We also have to be checking the information that the user is submitting coming from a form that our application presented to the user. We want to be preventing them when submitting the form from a malicious source.

#### Demo

To demonstrate a CSRF, I am applying the authorize attribute to my two Edit action methods of my application.

Hide   Shrink https://www.codeproject.com/images/arrow-up-16.png   Copy Code

*//*

*// GET: /CSRF/Edit/5*

[Authorize]

public ActionResult Edit(int id = 0)

{

Employee employee = db.Employees.Find(id);

if (employee == null)

{

return HttpNotFound();

}

return View(employee);

}

*//*

*// POST: /CSRF/Edit/5*

[HttpPost]

[Authorize]

public ActionResult Edit(Employee employee)

{

if (ModelState.IsValid)

{

db.Entry(employee).State = EntityState.Modified;

db.SaveChanges();

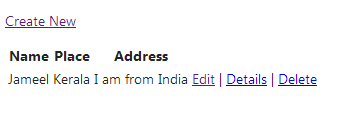
return RedirectToAction("Index");

}

return View(employee);

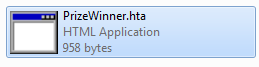
}

I can save, edit the records because I had already authenticated. Below is a sample record that I had saved into the database successfully.

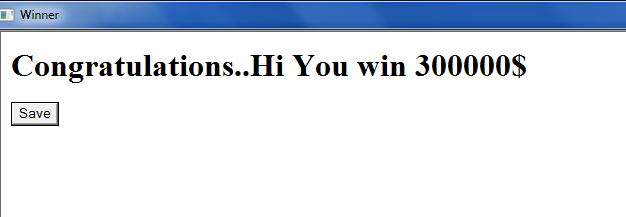


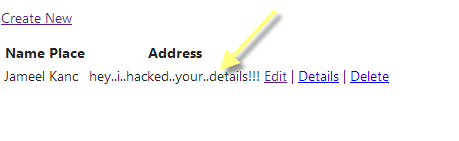
In the developer point of view, we are confident that I had authorized an attribute in place for preventing malicious user from editing an Employee details.

Watch would happen that I logged in as a user. I came across an interesting link in my system.



Maybe this link will you get from an email or from another website or some other areas of internet. Now I am going to click the link and see a page will come up.



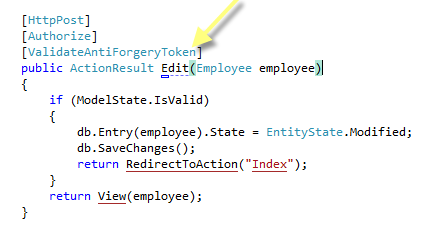


Look at the action that the form points to which has the same URL where the employee is posted. The form contains all of the input needed to complete the request and also at the bottom some line of JavaScript for automatically submitting the form when the page loads.

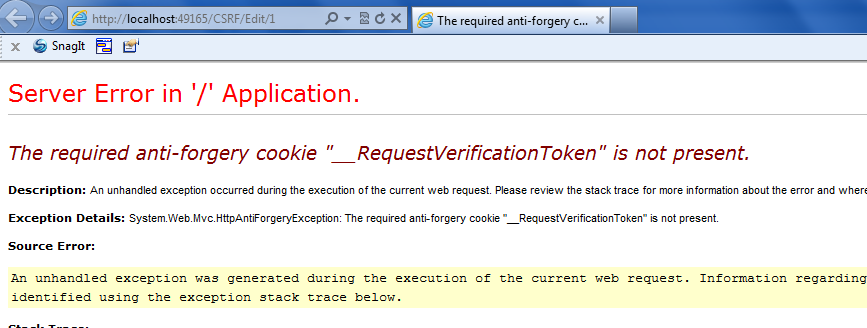
## How Can We Prevent This?

Use @Html.AntiForgeryToken() inside the form tag. This token will add a hidden input value that is unique to the browsing session. Also send a matching value in a cookie to the users browser so the user accepts this cookie and that something malicious website would not be able to do.

Also you should put an attribute ValidateAntiForgeryToken for matching the form value and cookie value:



I am again going to edit my record that the malicious user had done. Now I am going to click that link again and the ASP.NET MVC throws an exception that AntiForgeryToken is not supplied or invalid.



## Normalization in sql server?

**What is Normalization?**  
When we want to store the some piece of information, we store it in database. Now storing a data in database mean, it should be stored properly so that while retrieving it, it should be easy. So we say store it in Normalized way. In short, Normalization can be defined as the process of organizing the data in database efficiently. The result of normalization is a logical database design and is called as Normal Form.  
  
**Why Normalization?**  
Goals of Normalization process are:

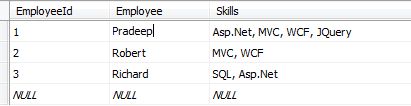
1. It helps you to eliminate the redundant data from same table.
2. It ensures the data dependencies between the tables are proper.
3. A Normalized database design makes it easy to change in modification is required.

**Advantages of Normalization**

1. Data redundancy is removed
2. Faster update as redundant columns from a tables are removed.
3. Easy understanding of structure
4. Improvement in Index as be achieved
5. Long term maintainability of database get easier

**Disadvantages of Normalization**

1. Query to some extent get complicated
2. Performance may degrade due to multiple joins

Suppose I have some data with me, say  
  
**UnNormalized Table  
  
**  
As you can see above, table is not properly managed  
  
Normalization process is mainly divided in to stages which we call as Normal Form. Let’s talk about each Normal Form one by one with an example. Basically a database can be normalized into various normal forms such as-

First Normal Form (1NF)

Second Normal Form (2NF)

Third Normal Form (3NF)

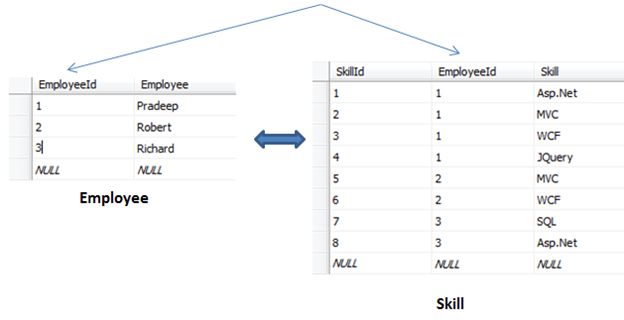
Boyce Codd Normal Form (BCNF)

and Fourth Normal Form (4NF) and so on.

But today I would like to talk about only up to Boyce Codd Normal Form because Fourth Normal Form and others is rarely used in the database design.

**First Normal Form (1NF)**It says,

* Eliminate all repeating groups in individual table
* One cell should contain only one data
* Create a separate table for each set of related data
* Identify each set of related data with a primary key



As you can see above,

* Our UnNormalized table does not have repeating group. So this is not applicable over here.
* We have eliminated the comma separated values and shifted to another table Skill. Each employee in Employee table is related to Skill table through EmployeeId column (Note: It is not a foreign key relationship. We are just replacing the comma separated values in each cell in to rows and creating separate table for it)
* Each cell is containing single value
* Skills of table is identified with primary key

**Second Normal Form (2NF)**It says,

* Tables should be in First Normal Form (1NF)
* Eliminate partial primary key dependencies
* Non key column must depend on the entire composite primary key and create separate tables for sets of values that apply to multiple records
* Relate these tables with a foreign key

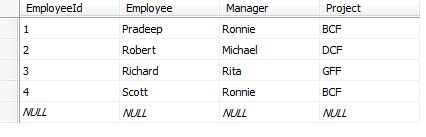


As you can see above,

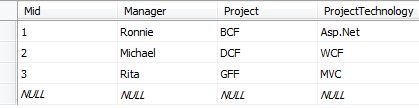
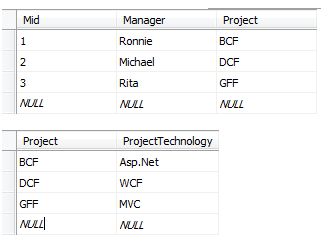
* We have eliminated the partial primary key dependency of EmployeeId and created a new table which will contain the relation of EmployeeId and SkillId. This both columns will be the respective foreign key for Employee and Skill table.

**Third Normal Form (3NF)**

* Table should be in Second Normal Form
* Eliminate fields that are not dependent on key i.e. - Eliminate Transitive Dependencies. Create a separate table for it.

Let’s suppose I want to add new columns in Employee Table Manager and Project  
  
  
  
Now, adding this 2 column violates the third Normal Form because the non key column Project is dependent on another not key column i.e. - Manager.  
So we will normalize the table by separating nondependent column to another table. This way we can achieve third Normal Form.  
  
  
  
**Boyce Codd Normal Form (BCNF)**It says,

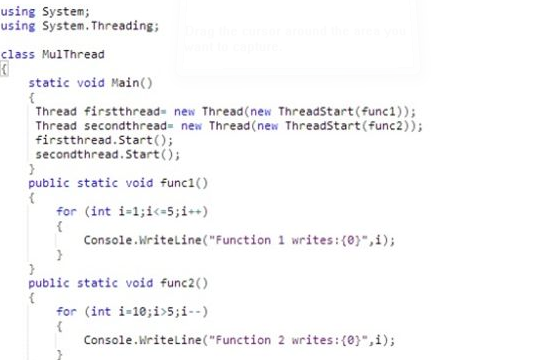
* BCNF is based on the concept of a determinant.
* A determinant is any attribute (simple or composite) on which some other attribute is fully functionally dependent.
* A relation is in BCNF is, and only if, every determinant is a candidate key.
* The same as 3NF except in 3NF we only worry about non-key attributes

**Note:**If there is only one candidate key then 3NF and BCNF are the same.  
  
Consider another column in below table say ProjectTechnology  
  
  
  
As shown above, each manager will be handling unique project, so we can say particular project is determined by particular manger where Manager and Project depicts a candidate key.  
If we delete the entry of Manager Ronnie from the table we lose not only information of Project BCF but also the fact that project was developed in Asp.Net technology. We cannot make the entry of the fact that BCF project was developed using Asp.Net.  
So let’s break this into separate tables  
  
 **Fourth Normal Form (4NF)**It says,

* Database design should follow the 1NF, 2NF, 3NF and BCNF if possible
* There must not be more than one multivalued dependencies other than a candidate key

**Virtusa**

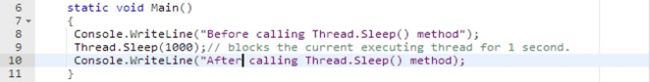
* **How do you implement multithreading?**

****

**Most Commonly used Static Member of System.Threading.Thread class**

The following are the most commonly used static members of the System.Threading.Thread class:

* **CurrentThread()**: gives a reference to the currently executing thread
* **Thread.Sleep()**: cause the currently executing thread to pause temporarily for the specified amount of time



When you execute the preceding program, you will notice a delay of 1 second between the printing of the two lines.

**Most Common Instance Member of the System.Threading.Thread class**

The following are the most common instance members of the System.Threading.Thread class:

* **Name**   
    
  A property of string type used to get/set the friendly name of the thread instance.
* **Priority**   
    
  A property of type System.Threading.ThreadPriority to schedule the priority of threads.
* **IsAlive**   
    
  A Boolean property indicating whether the thread is alive or terminated.
* **ThreadState**   
    
  A property of type System.Threading.ThreadState, used to get the value containing the state of the thread.
* **Start()**Starts the execution of the thread.
* **Abort()**   
    
  Allows the current thread to stop the execution of the thread permanently.
* **Suspend()**   
    
  Pauses the execution of the thread temporarily.
* **Resume()**Resumes the execution of a suspended thread.
* **Join()**  
    
  Make the current thread wait for another thread to finish.
* **Diff b/w viewbag and viewdata?**

**Difference between ViewBag & ViewData:**

ViewData is a dictionary of objects that is derived from ViewDataDictionary class and accessible using strings as keys. ViewBag is a dynamic property that takes advantage of the new dynamic features in C# 4.0. ViewData requires typecasting for complex data type and check for null values to avoid error. ViewBag doesn’t require typecasting for complex data type.

**ViewBag & ViewData Example:**

public ActionResult Index()

{

ViewBag.Name = "Arun Prakash";

return View();

}

public ActionResult Index()

{

ViewData["Name"] = "Arun Prakash";

return View();

}

Calling in View

@ViewBag.Name

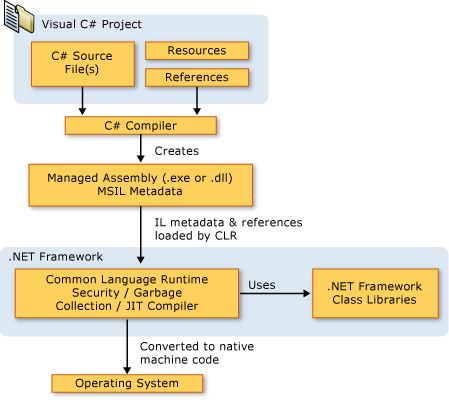
@ViewData["Name"]

* **Mvc filters?**
* **Design patterns?**
* **Diff b/w wcf and webapi?**
* **What is CLR?**

Key attributes of .NET CLR

* As part of the Microsoft  .NET Framework, the Common Language Runtime (CLR) is the programming (Virtual Machine component) that manages the execution of programs written in any language that uses the .NET Framework, for example C#, VB.Net, F# and so on.
* Programmers write code in any language, including VB.Net, C# and F# when they compile their programs into an intermediate form of code called CLI in a portable execution file (PE) that can be managed and used by the CLR and then the CLR converts  it into machine code to be will executed by the processor.
* The information about the environment, programming language, its version and what class libraries will be used for this code are stored in the form of metadata with the compiler that tells the CLR how to handle this code.
* The CLR allows an instance of a class written in one language to call a method of the class written in another language.

Got it? Let's try to look at the process of a C# application in this below diagram.



As you can see from the above diagram, the CLR provides several services. 

Functions of .NET CLR

* Convert code into CLI
* Exception handling
* Type safety
* Memory management (using the Garbage Collector)
* Security
* Improved performance
* Language independency
* Platform independency
* Architecture independency

Components of .NET CLR

The key components of CLR includes the following:

* Class Loader - Used to load all classes at run time.
* MSIL to Native code - The Just In Time (JTI) compiler will convert MSIL code into native code.
* Code Manager - It manages the code at run time.
* Garbage Collector - It manages the memory. Collect all unused objects and deallocate them to reduce memory.
* Thread Support - It supports multithreading of our application.
* Exception Handler - It handles exceptions at run time.

Benefits of .NET CLR

The runtime provides the following benefits:

* Performance improvements.
* The ability to easily use components developed in other languages.
* Extensible types provided by a class library.
* Language features such as inheritance, interfaces, and overloading for object-oriented programming.
* Support for explicit free threading that allows creation of multithreaded, scalable applications.
* Support for structured exception handling.
* Support for custom attributes.
* Garbage collection.
* Use of delegates instead of function pointers for increased type safety and security.
* **What is CTS?**

CTS and CLS are parts of .NET CLR and are responsible for type safety within the code. Both allow cross-language communication and type safety. In this article, I would like to expose the relationship between these two.

### CLS

CLS stands for Common Language Specification and it is a subset of CTS. It defines a set of rules and restrictions that every language must follow which runs under the .NET framework. The languages which follow these set of rules are said to be CLS Compliant. In simple words, CLS enables cross-language integration or Interoperability.

**For Example**

1. if we talk about C# and VB.NET then, in C# every statement must have to end with a semicolon. it is also called a statement Terminator, but in VB.NET each statement should not end with a semicolon(;).

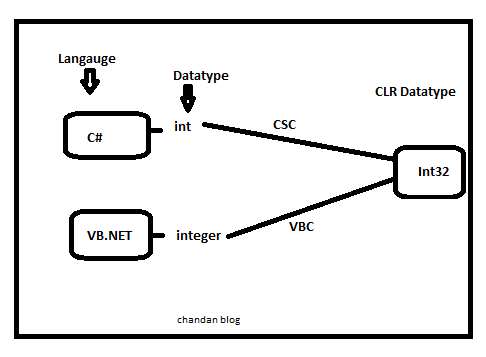
**Explanation of the above Example**

So these syntax rules which you have to follow from language to language differ but CLR can understand all the language Syntax because in .NET each language is converted into MSIL code after compilation and the MSIL code is language specification of CLR.

**CTS**

Common Type System (CTS) describes the datatypes that can be used by managed code. CTS defines how these types are declared, used and managed in the runtime. It facilitates cross-language integration, type safety, and high-performance code execution. The rules defined in CTS can be used to define your own classes and values.

OR we can also understand like,  
  
CTS deals with the data type. So here we have several languages and each and every language has its own data type and one language data type cannot be understandable by other languages but .NET Framework language can understand all the data types.

C# has an **int**data type and VB.NET has **Integer**data type. Hence a variable declared as an int in C# and Integer in VB.NET, finally after compilation, uses the same structure Int32 from CTS.  
**Note**All the structures and classes available in CTS are common for all .NET Languages and the purpose of these is to support language independence in .NET. Hence it is called CTS

* **What are valueType and Reference Types?**

**Value Types**  
  
Value type is either **struct** type or**enumeration** type. All value types implicitly**inherit** from **object** class. Value types are implicitly **sealed** thus it is not possible for any type to derive from value type:

* A variable of a value type always contains a value of that type. Unlike reference types, it is not possible for a value of a value type to be **null**, or to reference an object of a more derived type.
* Assignment to a variable of a value type creates a copy of value being assigned. This differs from assignment to a variable of a reference type, which copies the reference but not the object identified by the reference.
* All value types implicitly inherit from **System.ValueType** class, which, in turn, inherits from **System.Objectclass**.
* Note that **System.ValueType** is not itself **value-type**. Rather, it is class-type from which all value-types are derived.

**Reference Type**  
A reference type is a **class** type, an **interface** type, an array type, or a **delegate** type.

* A reference type is **reference** to instance of type, the latter known as an object.
* Special value **null** is compatible with all reference types and indicates **absence**of an instance.
* Reference types allocated on **managed heap** its reference address is maintained in stack.
* The special value null indicate that stack memory do not have any reference to **heap** memory.
* **Garbage collection?**

# Garbage Collection in C# | .NET Framework

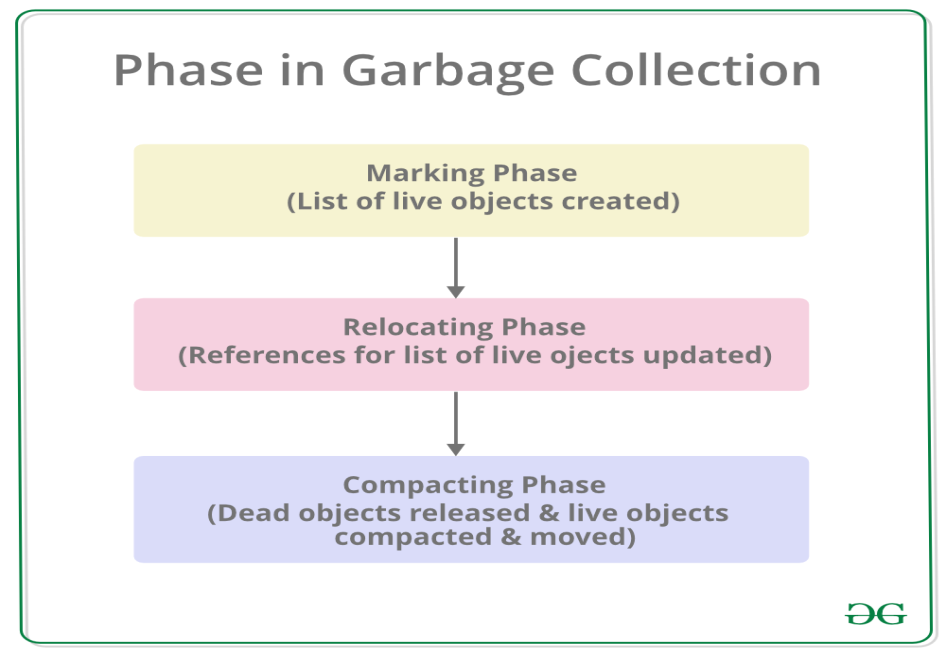
Automatic memory management is made possible by **Garbage Collection in .NET Framework**. When a class object is created at runtime, certain memory space is allocated to it in the heap memory. However, after all the actions related to the object are completed in the program, the memory space allocated to it is a waste as it cannot be used. In this case, garbage collection is very useful as it automatically releases the memory space after it is no longer required.

Garbage collection will always work on **Managed Heap** and internally it has an Engine which is known as the **Optimization Engine**.

Garbage Collection occurs if at least one of multiple conditions is satisfied. These conditions are given as follows:

* If the system has low physical memory, then garbage collection is necessary.
* If the memory allocated to various objects in the heap memory exceeds a pre-set threshold, then garbage collection occurs.
* If the GC.Collect method is called, then garbage collection occurs. However, this method is only called under unusual situations as normally garbage collector runs automatically.Phases in Garbage Collection

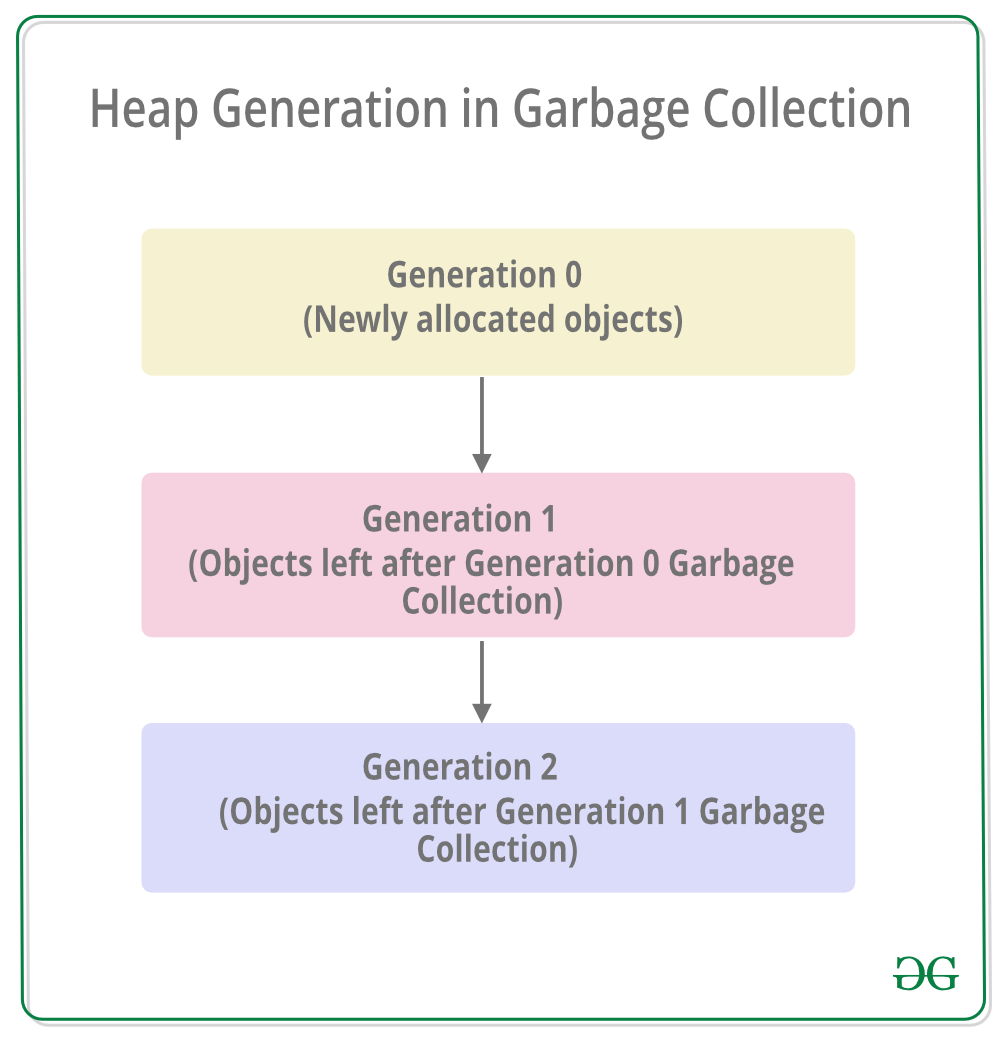
There are mainly **3** phases in garbage collection. Details about these are given as follows:

[](https://media.geeksforgeeks.org/wp-content/uploads/20190417112537/PhaseInGarbageCollection.png)

1. **Marking Phase:** A list of all the live objects is created during the marking phase. This is done by following the references from all the root objects. All of the objects that are not on the list of live objects are potentially deleted from the heap memory.
2. **Relocating Phase:** The references of all the objects that were on the list of all the live objects are updated in the relocating phase so that they point to the new location where the objects will be relocated to in the compacting phase.
3. **Compacting Phase:** The heap gets compacted in the compacting phase as the space occupied by the dead objects is released and the live objects remaining are moved. All the live objects that remain after the garbage collection are moved towards the older end of the heap memory in their original order.

#### Heap Generations in Garbage Collection

The heap memory is organized into 3 generations so that various objects with different lifetimes can be handled appropriately during garbage collection. The memory to each Generation will be given by the [**Common Language Runtime(CLR)**](https://www.geeksforgeeks.org/common-language-runtime-clr-in-c-sharp/) depending on the project size. Internally, Optimization Engine will call the Collection Means Method to select which objects will go into Gneration 1 or Generation 2.

[](https://media.geeksforgeeks.org/wp-content/uploads/20190417112732/HeapGenerationInGarbageCollection.png)

* **Generation 0 :**All the short-lived objects such as temporary variables are contained in the generation 0 of the heap memory. All the newly allocated objects are also generation 0 objects implicitly unless they are large objects. In general, the frequency of garbage collection is the highest in generation 0.
* **Generation 1 :**If space occupied by some generation 0 objects that are not released in a garbage collection run, then these objects get moved to generation 1. The objects in this generation are a sort of buffer between the short-lived objects in generation 0 and the long-lived objects in generation 2.
* **Generation 2 :**If space occupied by some generation 1 objects that are not released in the next garbage collection run, then these objects get moved to generation 2. The objects in generation 2 are long lived such as static objects as they remain in the heap memory for the whole process duration.

**Note:**Garbage collection of a generation implies the garbage collection of all its younger generations. This means that all the objects in that particular generation and its younger generations are released. Because of this reason, the garbage collection of generation 2 is called a full garbage collection as all the objects in the heap memory are.released. Also, the memory allocated to the Generation 2 will be greater than Generation 1’s memory and similarly the memory of Generation 1 will be greater than Generation 0’s memory(**Generation 2 > Generation 1 > Generation 0**).

A program that demonstrates the number of heap generations in garbage collection using the GC.MaxGeneration property of the GC class is given as follows:

filter\_none

brightness\_4

|  |
| --- |
| using System;    public class Demo {        // Main Method      public static void Main(string[] args)      {          Console.WriteLine("The number of generations are: " +                                             GC.MaxGeneration);      }  } |

**Output:**

The number of generations are: 2

In the above program, the GC.MaxGeneration property is used to find the maximum number of generations that are **supported by the system** i.e. 2. If you will run this program on online compilers then you may get different outputs as it depends on the system.

#### Methods in GC Class

The GC class controls the garbage collector of the system. Some of the methods in the GC class are given as follows:

**GC.GetGeneration() Method :**This method returns the generation number of the target object. It requires a single parameter i.e. the target object for which the generation number is required.

A program that demonstrates the GC.GetGeneration() method is given as follows:

filter\_none

brightness\_4

|  |
| --- |
| using System;    public class Demo {        public static void Main(string[] args)      {          Demo obj = new Demo();          Console.WriteLine("The generation number of object obj is: "                                            + GC.GetGeneration(obj));      }  } |

**Output:**

The generation number of object obj is: 0

**GC.GetTotalMemory() Method :** This method returns the number of bytes that are allocated in the system. It requires a single boolean parameter where true means that the method waits for the occurrence of garbage collection before returning and false means the opposite.

A program that demonstrates the GC.GetTotalMemory() method is given as follows:

filter\_none

brightness\_4

|  |
| --- |
| using System;    public class Demo {        public static void Main(string[] args)      {          Console.WriteLine("Total Memory:" + GC.GetTotalMemory(false));            Demo obj = new Demo();            Console.WriteLine("The generation number of object obj is: "                                             + GC.GetGeneration(obj));            Console.WriteLine("Total Memory:" + GC.GetTotalMemory(false));      }  } |

**Output:**

Total Memory:4197120

The generation number of object obj is: 0

Total Memory:4204024

**Note:** The output may vary as it depends on the system.

**GC.Collect() Method :**Garbage collection can be forced in the system using the GC.Collect() method. This method requires a single parameter i.e. number of the oldest generation for which garbage collection occurs.

A program that demonstrates the GC.Collect() Method is given as follows:

filter\_none

brightness\_4

|  |
| --- |
| using System;    public class Demo {        public static void Main(string[] args)      {          GC.Collect(0);          Console.WriteLine("Garbage Collection in Generation 0 is: "                                            + GC.CollectionCount(0));      }  } |

**Output:**

Garbage Collection in Generation 0 is: 1

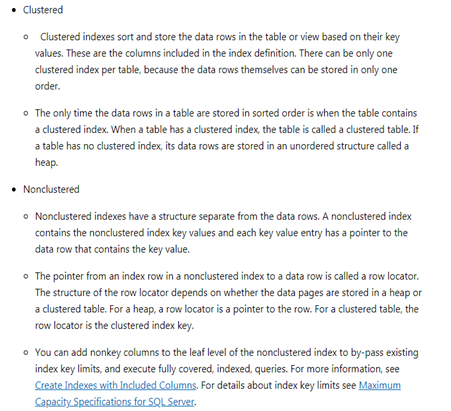
**Benefits of Garbage Collection**

* Garbage Collection succeeds in allocating objects efficiently on the heap memory using the generations of garbage collection.
* Manual freeing of memory is not needed as garbage collection automatically releases the memory space after it is no longer required.
* Garbage collection handles memory allocation safely so that no objects use the contents of another object mistakenly.
* The constructors of newly created objects do not have to initialize all the data fields as garbage collection clears the memory of objects that were previously released.
* **Dispose vs Finalize?**

### Methods dispose() and finalize() are the methods of C# which are invoked to free the unmanaged resources held by an object. The dispose() method is defined inside the interface IDisposable whereas, the method finalize() is defined inside the class object. The main difference between dispose() and finalize() is that the method **dispose**() has to be explicitly invoked by the user whereas, the method **finalize()** is invoked by the garbage collector, just before the object is destroyed.

| **BASIS FOR COMPARISON** | **DISPOSE( )** | **FINALIZE( )** |
| --- | --- | --- |
| Defined | The method dispose( ) is defined in the interface IDisposable interface. | The method finalize( ) id defined in java.lang.object class. |
| Syntax | public void Dispose( ){ // Dispose code here } | protected void finalize( ){ // finalization code here } |
| Invoked | The method dispose( ) is invoked by the user. | The method finalize( ) is invoked by the garbage collector. |
| Purpose | Method dispose( ) is used to free unmanaged resources whenever it is invoked. | Method finalize( ) is used to free unmanaged resources before the object is destroyed. |
| Implementation | The method dispose( ) is to be implemented whenever there is a close( ) method. | The method finalize( ) is to be implemented for unmanaged resources. |
| Access specifier | The method dispose( ) is declared as public. | The method finalize( ) is declared as private. |
| Action | The method dispose( ) is faster and instantly disposes an object. | The method finalize is slower as compared to dispose |
| Performance | The method disposes( ) performs the instantaneous action hence, does not effect the performance of websites. | The method finalize( ) being slower affects the performance of the websites. |

* **Performance tuning in sql server?**
* **Diff b/w clustered and non clustered indexes?**

****

**WINWIRE**

**1. how to write nth salary of employee using c# linq?**

emplist.OrderByDescending(x => x.Salary).Select(x=>x.Salary).Distinct().Take(3).Skip(2).First();

**2.generics and collection in c#?**

**Generics means it will be separate the logic from the datatypes**

**3.what is view in sql server how it is useful?**

**4.difference between abstract class and interface when it will be useful?**

**5.extenstions method sudo code?**

**6.any design pattern used in your projects?**

**7.difference between return false and e.prevendefault?**

 AHeadrace

**1.how provide the security in web api?**

**2.web api owin and how to work token based authentication?**

ASP.NET Web API can be accessed over Http by any client using the Http protocol. Typically, in a Line of Business (LOB) application, using Web API is a standard practice now-a-days. This framework enables data communication in JSON format (by default) and hence helps in lightweight communication.

  In token-based authentication, you pass your credentials [user name and password], which go to authentication server. Server verifies your credentials and if it is a valid user then it will return a signed token to client system, which has expiration time. Client can store this token to locally using any mechanism like local storage, session storage etc and if client makes any other call to server for data then it does not need to pass its credentials every time. Client can directly pass token to server, which will be validated by server and if token is valid then you will able to access your data.

**How token based authentication actually works?**

In the Token based approach, the client application first sends a request to Authentication server endpoint with an appropriate credential. Now If the username and password are found correct then the Authentication server send a token to the client as a response. This token contains enough data to identify a particular user and an expiry time.The client application then uses the token to access the restricted resources in next requests till the token is valid.  
 *In this article, I have used Visual Studio 2015*  
  
**1:- Create New Project.**  
Go to the file menu > create > projet > select "asp.net web application" under web > enter application name > select your project location > and then click on add button  
  
    It will bring up a new dialog window for select template > here I will select empty template > and then checked MVC & Web API checkbox from Add folder and core references for > and then click on Ok button.  
  
We are using **OWIN [Open Web Interface for .Net]** that is an interface between your web server and web application. So, it works as a middle ware in applications, which process your incoming request and validate it. Here we are using ***AuthServerProvider***, which is nothing but a class which validate user based on their credentials. You can find this class below.  
  
**2: Add required references from NuGet packages into our application.**  
To Implement token based authentication in WEB API, we need to install followings references from NuGet packages

* Microsoft.Owin.Host.SystemWeb
* Microsoft.Owin.Security.OAuth
* Microsoft.Owin.Cors

for adding following resources from NuGet, Go to Solution Explorer >  Right Click on References > Click on Manage NuGet packages > Search for the Microsoft.Owin.Host.SystemWeb, Microsoft.Owin.Security.OAuth & Microsoft.Owin.Cors and install.  
 **3: Add a class for validating user credentials asking for tokens.**  
add a class in our application for validate the credentials for users and generate token.  
  
In this class we will inherit "OAuthAuthorizationServerProvider" class for  overriding 2 methods "ValidateClientAuthentication" and "GrantResourceOwnerCredentials".  
  
"ValidateClientAuthentication" method is used for validating client app (for the sake of simplicity, we will  deep dive on "ValidateClientAuthentication" method later) and in the "GrantResourceOwnerCredentials"  method we will validate the credentials of users and if we found valid credential, we will generate the signed token, using which user can access authorized resources of server.  
  
**AuthServerProvider.cs**

**using** **Microsoft.Owin.Security.OAuth**;

**using** **System**;

**using** **System.Collections.Generic**;

**using** **System.Linq**;

**using** **System.Security.Claims**;

**using** **System.Threading.Tasks**;

**using** **System.Web**;

**namespace** **WebApiTokenBasedAuth**

{

**public** **class** **AuthServerProvider**: OAuthAuthorizationServerProvider

{

**public** **override** **async** Task **ValidateClientAuthentication**(OAuthValidateClientAuthenticationContext context)

{

context.Validated();

}

**public** **override** **async** Task **GrantResourceOwnerCredentials**(OAuthGrantResourceOwnerCredentialsContext context)

{

**var** identity = **new** ClaimsIdentity(context.Options.AuthenticationType);

**if** (context.UserName == "admin" && context.Password == "admin")

{

identity.AddClaim(**new** Claim(ClaimTypes.Role, "admin"));

identity.AddClaim(**new** Claim("username", "admin"));

identity.AddClaim(**new** Claim(ClaimTypes.Name, "Suraj Mad. Admin"));

context.Validated(identity);

}

**else** **if** (context.UserName == "user" && context.Password == "user")

{

identity.AddClaim(**new** Claim(ClaimTypes.Role, "user"));

identity.AddClaim(**new** Claim("username", "user"));

identity.AddClaim(**new** Claim(ClaimTypes.Name, "Suraj Mad."));

context.Validated(identity);

}

**else**

{

context.SetError("invalid\_grant", "Provided username and password is incorrect");

**return**;

}

}

}

}

**4: Add Owin Start Up class.**  
Now we will add OWIN Startup class. This is the startup class used to configure application startup.  
It accepts IAppBuilder interface to manage middleware for the application. In this case it is OWIN. This function uses the **OAuthAuthorizationServerOptions**class which provides the information needed to control Authorization server middleware behavior. The code sets some properties for this class.  
Go to Solution Explorer > Right Click on Project Name form Solution Explorer > Add > New Item > Select OWIN Startup class > Enter class name > Add.  
  
**Startup.cs**

**using** **System**;

**using** **Microsoft.Owin**;

**using** **Owin**;

**using** **Microsoft.Owin.Security.OAuth**;

**using** **System.Web.Http**;

[assembly: OwinStartup(typeof(WebApiTokenBasedAuth.Startup))]

**namespace** **WebApiTokenBasedAuth**

{

**public** **class** **Startup**

{

**public** **void** **Configuration**(IAppBuilder app)

{

//enable cors origin requests

app.UseCors(Microsoft.Owin.Cors.CorsOptions.AllowAll);

**var** myProvider = **new** WebApiTokenBasedAuth.AuthServerProvider();

OAuthAuthorizationServerOptions options = **new** OAuthAuthorizationServerOptions

{

AllowInsecureHttp = **true**,

TokenEndpointPath = **new** PathString("/token"),

AccessTokenExpireTimeSpan = TimeSpan.FromDays(**1**),

Provider = myProvider

};

app.UseOAuthAuthorizationServer(options);

app.UseOAuthBearerAuthentication(**new** OAuthBearerAuthenticationOptions());

HttpConfiguration config = **new** HttpConfiguration();

WebApiConfig.Register(config);

}

}

}

In the above class we need the following important properties  
  
· **TokenEndPointPath**- The client application communicates as part of the OAuth protocol. To complete the login with the token, /Token is used. The grant\_type must be shared by the client to complete the login using access token generated by the server.  
  
· **AccessTokenExpirationTimeSpan**- This is the time span for the life of the authorization token after being issued.  
  
· **Provider**- The value for the property is set by the OAuthorizationServerProvider object. This class is responsible for providing behavior to requests. This is used to validate client authentication, claims etc.  
  
**AllowInsecureHttp**- This is a Boolean property used to allow authorize and token requests to arrive on http URI address.  
  
**5: Add an another Class for override authorize attribute.**  
When building an HTTP REST API, we should use appropriate HTTP response codes to indicate the status of a response. I always use 401 and 403 status code for getting authentication/authorization status. 401 (Unauthorized) - indicates that the request has not been applied because it lacks valid authentication credentials for the target resource. and 403 (Forbidden) - when the user is authenticated but isn’t authorized to perform the requested operation on the given resource.  
  
Unfortunately, the ASP.NET MVC/Web API [Authorize] attribute doesn’t behave that way – it always emits 401. So, here in our Web API application, I am going to add a class for override this behavior. Here we will return 403 when the user is authenticated but not authorized to perform the requested operation.  
  
**AuthorizeAttribute.cs**

**using** **System**;

**using** **System.Collections.Generic**;

**using** **System.Linq**;

**using** **System.Web**;

**namespace** **WebApiTokenBasedAuth**

{

**public** **class** **AuthorizeAttribute**: System.Web.Http.AuthorizeAttribute

{

**protected** **override** **void** **HandleUnauthorizedRequest**(System.Web.Http.Controllers.HttpActionContext actionContext)

{

**if** (!HttpContext.Current.User.Identity.IsAuthenticated)

{

**base**.HandleUnauthorizedRequest(actionContext);

}

**else**

{

actionContext.Response = **new** System.Net.Http.HttpResponseMessage(System.Net.HttpStatusCode.Forbidden);

}

}

}

}

**6: Add WEB API Controller.**  
Add a WEB API Controller , Where we will add some action So we can check the token authentication is working fine or not.

[](https://1.bp.blogspot.com/-uloFJHcd9K4/Wp7dXOzhkhI/AAAAAAAAMjU/kZyLsiQTAz82chOqtOStj3dHvsfOE39egCLcBGAs/s1600/WebApiController.PNG)

**7: Add an action for getting data from the server for all anonymous user.**  
I have added this action for all anonymous users. All type of request, whether it is authenticated or not can access this action.

[AllowAnonymous]

[HttpGet]

[Route("api/MyWebApi/GetAll")]

**public** IHttpActionResult **Get**()

{

**return** **Ok**("Now server time is: " + DateTime.Now.ToString());

}

**8: Add an another action for getting data from the server for all authenticated user.**  
I have added this action for all type of authenticated users, whether it is Admin user or normal user.

[Authorize]

[HttpGet]

[Route("api/MyWebApi/authenticate")]

**public** IHttpActionResult **GetForAuthenticate**()

{

**var** identity = (ClaimsIdentity)User.Identity;

**return** **Ok**("Hello " + identity.Name);

}

**9: Add an another action for getting data from the server only for Admin user.**  
I have added this action only for Admin role type users.

[Authorize(Roles = "admin")]

[HttpGet]

[Route("api/MyWebApi/authorize")]

**public** IHttpActionResult **GetForAdmin**()

{

**var** identity = (ClaimsIdentity)User.Identity;

**var** roles = identity.Claims

.Where(c => c.Type == ClaimTypes.Role)

.Select(c => c.Value);

**return** **Ok**("Hello " + identity.Name + " Role: " + **string**.Join(",", roles.ToList()));

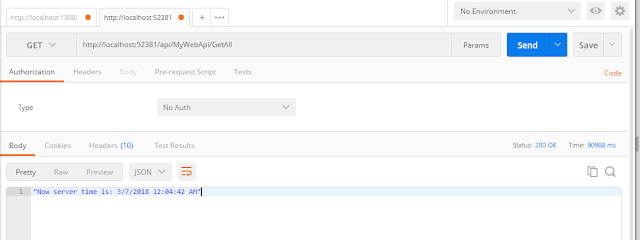
}

**10: Run Application.**  
Our WEB API Configuration is ready, Now we will test our application with POSTMAN  
  
Postman is an extension of Chrome, which is used as a client application to test the request and response between web service and client.

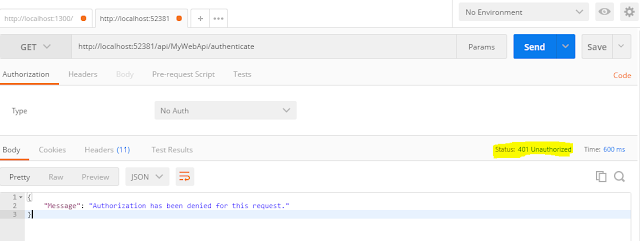
**Test 1:**

Select GET (see below picture section 1),  Enter this url http://localhost:/api/MyWebApi/GetAll  and then click on send button.

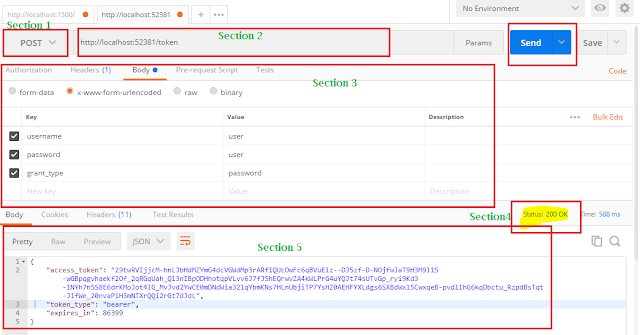
We will get 200 OK status code (see section 3 in the below picture) and the result in the section 4 in that picture. That means our first action working fine when the request is anonymous.

[](https://3.bp.blogspot.com/-N6_xnjnAJSk/Wp7frcu-OfI/AAAAAAAAMjg/Asthq6bc8x0on2CCXb9TXGkmBT9SN5NJgCLcBGAs/s1600/test1.PNG)

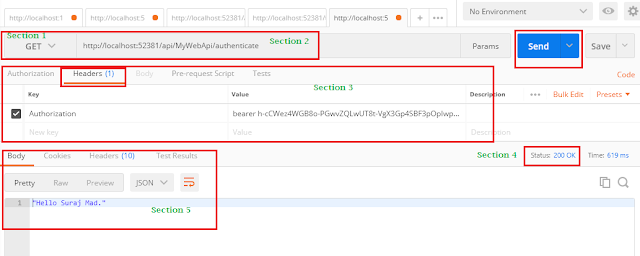
But now if we try to access our 2nd action (GetForAuthenticate with url : http://localhost:52381/api/MyWebApi/authenticate) then we will get 401 Unauthenticated because the request is not authenticated yet. It will same for the 3rd action also.

[](https://4.bp.blogspot.com/--U9A_qqYvq0/Wp7hUaSEWqI/AAAAAAAAMjs/vsZMTTBpYNYPMIAxMeVEWesf-Hkw0ezIwCLcBGAs/s1600/test2.1.PNG)

So, what we need to access the 2nd and 3rd action? We need access token from server first and then we can access the 2nd and 3rd action with that access token.  
  
**Test 2 : Getting access token.**  
Select POST (in section 1),  Enter this URL http://localhost:/token (in section 2) and then click on body (in section 3) and select select x-www-form-urlencoded and then enter 3 parameter, 1. username (value : user) 2. password (value: user) and 3. grant\_type (value: password) and then click on  send button. After click on send button we will get 200 OK (see section 4) and access token (see section 5)

[](https://3.bp.blogspot.com/-Gg0R1CvuG78/Wp7jd1sz9_I/AAAAAAAAMkE/XVtlzBWMS8IqMfKtDrvIt7pBxTNzgblcACLcBGAs/s1600/test2.PNG)

Now we can access <http://localhost:52381/api/MyWebApi/authenticate> with that access token.  
  
**Test 3: Access restricted resource with access token.**  
Select GET(in section 1),  Enter this URL <http://localhost:52381/api/MyWebApi/authenticate> (in section 2) and then click on Headers(in section 3) and enter 1 parameter, Authorization (value : Bearer) and then click on  send button. After click on send button we will get 200 OK (see section 4) and the result (see section 5).

[](https://2.bp.blogspot.com/-dX1LfjFZv4M/Wp7lKWMV7cI/AAAAAAAAMkQ/_cdwOALPSqE_CrvX2Xo1DmpWOzRSF7CKQCLcBGAs/s1600/test3.PNG)

In the same way, we can access our 3rd action but we have to get token logged in with username : admin and password: admin because the 3rd action accessible only for Admin role user.

**3.how to implement the caching in web api?**

## Problem in Output Caching

If ASP.NET "OutputCache" added to rest method, "cache-control" in "Http Response Headers" always shows "no-cache".

#### Example

Hide   Copy Code

public class ProductController : ApiController

{

[OutputCache(Duration = 120)]

public string Get(int id)

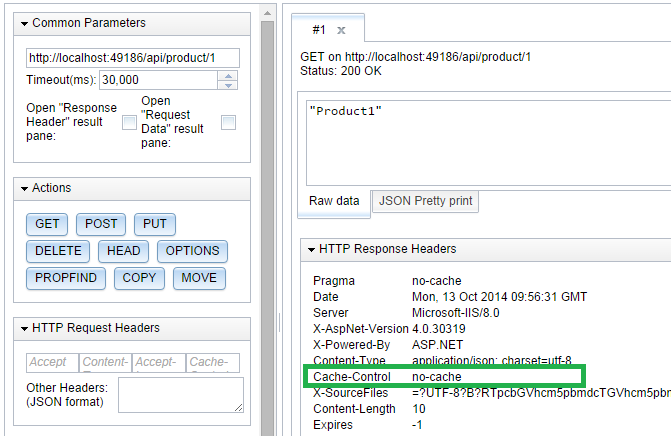
{

return "Product"+ 1;

}

}

Response in Chrome extension - XHR-Poster



## Solution

It's easy to implement basic caching using "ActionFilterAttribute". We need to add cache control in OnActionExecuted methods as follows:

Hide   Copy Code

public class CacheClientAttribute : ActionFilterAttribute

{

public int Duration { get; set; }

public override void OnActionExecuted(HttpActionExecutedContext actionExecutedContext)

{

actionExecutedContext.Response.Headers.CacheControl = new CacheControlHeaderValue

{

MaxAge = TimeSpan.FromSeconds(Duration),

MustRevalidate = true,

Public = true

}

}

}

Now, we can use this attribute in API method.

Hide   Copy Code

public class ProductController : ApiController

{

[CacheClient(Duration = 120)]

public string Get(int id)

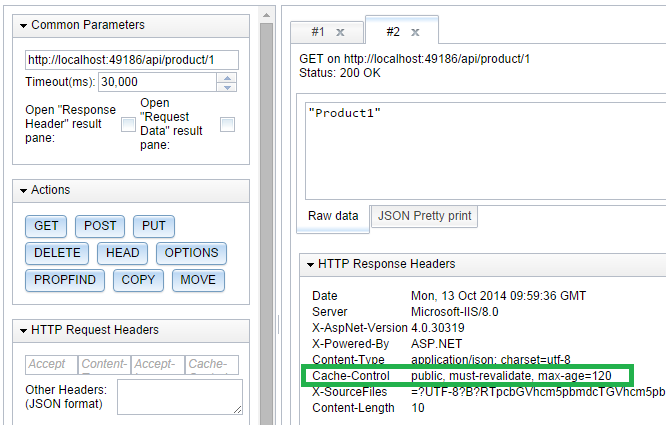
{

return "Product"+ 1;

}

}

Response in Chrome extension - XHR-Poster



# Caching without third party library in a very simple way

I had a need for this. I implemented some basic authentication for the ASP.NET Web API with Tokens and wanted to cache Tokens temporarily in memory to avoid going into the database for every HTTP request.

The solution is very simple.  
You just need to use Microsoft class for this called MemoryCache which resides in System.Runtime.Caching dll.

I created a simple helper class for caching with few lines of code:

public class MemoryCacher

{

public object GetValue(string key)

{

MemoryCache memoryCache = MemoryCache.Default;

return memoryCache.Get(key);

}

public bool Add(string key, object value, DateTimeOffset absExpiration)

{

MemoryCache memoryCache = MemoryCache.Default;

return memoryCache.Add(key, value, absExpiration);

}

public void Delete(string key)

{

MemoryCache memoryCache = MemoryCache.Default;

if (memoryCache.Contains(key))

{

memoryCache.Remove(key);

}

}

}

When you want to cache something:

memCacher.Add(token, userId, DateTimeOffset.UtcNow.AddHours(1));

And to get something from the cache:

var result = memCacher.GetValue(token);

   
NOTE: You should be aware that MemoryCache will be disposed every time IIS do app pool recycle.

If your Web API :

* Does not receive any request for more that 20 minutes
* Or hit default IIS pool interval 1740 minutes
* Or you copy a new version of your ASP.NET Web API project into an IIS folder (this will auto-trigger app pool recycle)

So you should have a workaround for this. If you don’t get value from the cache you should grab it for example from a database and then store it again in the cache:

var result = memCacher.GetValue(token);

if (result == null)

{

// for example get token from database and put grabbed token

// again in memCacher Cache

}

**4.how to handle sessions in web api?**

in Global.asax add

public override void Init()

{

this.PostAuthenticateRequest += MvcApplication\_PostAuthenticateRequest;

base.Init();

}

void MvcApplication\_PostAuthenticateRequest(object sender, EventArgs e)

{

System.Web.HttpContext.Current.SetSessionStateBehavior(

SessionStateBehavior.Required);

}

**5.how to handle the exceptions in web api?**

Please refer to this link [Exception Handling in ASP.NET Web API - A Guided Tour](https://www.exceptionnotfound.net/the-asp-net-web-api-exception-handling-pipeline-a-guided-tour/). There are 4 level exception handling pipeline:

* Level 1 - HttpResponseException
* Level 2 - Exception Filters
* Level 3 - Logging
* Level 4 - Exception Handlers

Web API provides us a great deal of flexibility in terms of exception handling. To recap:

* Use HttpResponseException or the shortcut methods to deal with unhandled exceptions at the action level.
* Use Exception Filters to deal with particular unhandled exceptions on multiple actions and controllers.
* Use ExceptionLogger to log any unhandled exception.
* Use Exception Handlers (one per application) to deal with any unhandled exception application-wide.

<https://www.exceptionnotfound.net/the-asp-net-web-api-exception-handling-pipeline-a-guided-tour/>

**6. what is static class where it can be useful in your project?**

**Static class** does not allow user to create instances of **the class** as well as it restrict **the** user to inherit any data members/functions to derived **class**. So **A static class can** make **your** implementation simpler ,safe and faster because you do not have to create **an** object in order to invoke its **methods**

**7.how to restrict the users to access the action menthods in mvc?**

My question that is there any better solution for checking whether user is logged-in or not and according to it redirect user to login or dashboard page so that user can't manipulate url and get access to functionality to which he is not authorized.

Yes, there's already a built-in method for doing this that does not rely on ASP.NET Sessions. It is called [Forms Authentication](http://msdn.microsoft.com/en-us/library/ff647070.aspx).

You don't need to be writing any custom Authorize attributes. Once you verified the credentials of the user simply set the FormsAuthentication cookie:

if (checking username and password exist in DB or not)

{

// Emitting forms authentication cookie

FormsAuthentication.SetAuthCookie(Info.Username, false);

//Redirect to dashboard

}

and then simply use the built-in Authorize attribute to decorate your protected controller actions:

public class AdminController : Controller

{

[Authorize(ValidRole = "Admin")]

public ActionResult Index()

{

// At this stage the user is authenticated and has the role Admin.

// You could get the current username using the User.Identity.Name property

return View();

}

}

Forms Authentication is stateless. It does not rely on any state on the server to track the currently authenticated user on the server. The information about the current user is contained in an encrypted forms authentication cookie that is sent along each request. This way you don't need to be thinking about handling complex scenarios when your application is hosted in a web farm in which case you would have needed to use distributed ASP.NET Sessions.

**8. what is polymorphism and what is method overloading and overriding where it can be useful?**

**9. what is use of stored procedures?**

**Aclara Company**

* **Difference between @viewchild and @input?**
* **What is lifecycle events in angular and its order?**
* **Properties get and set will be different types?**
* **Kendo ui grid code in angular?**
* **Write a linq query for missing column in list?**
* **Class A{**

**Public int test1(){**

**Try{**

**Int i =1/0;**

**}**

**Catch(exception ex)**

**{**

**Return 0;**

**}**

**Finally{**

**Return 1;**

**}**

**}**

**What is the output for given program?**

* **Pagination in angular?**
* **Wcf bindings?**
* **How do we change happened entity framework?**