**log4net logging mechanisam**

With a logging library (such as log4net), you can do all the things you are asking about easily. You can set up your logging in a config file to go to event log, file, console, etc (or any combination) and set the logging levels independently.

That is, you can say the file will have informationals, warnings, errors, and fatals, but the event log will log only the errors and fatals.

In general, nearly all of the logging libraries have these features. The examples below are just in terms of log4net but the same concepts apply to most logging libraries...

To get the logger in any class in your solution, you just need to ask for a logger in your class:

\_log.Debug("This is a debug message.");

\_log.Info("This is an informational message.");

\_log.Warn("This is a warning message.");

\_log.Error("This is an error message.");

\_log.Fatal("This is a fatal message.");

**Which protocol does sql server use**

**TCP - Transmission Control Protocol**

TCP IS KNOWN AS TRANSMISSION CONTROL PROTOCOL.. IT IS A PROTOCAL WHICH SENDS THE DATA FROM SOURCE TO DESTINATION SAFELY, AS IT IDENTIFIES THE ERRORS OF DATA PACKETS…..TCP TRANSFORMS DATA PACKETS WHICH ARE OUT OF ORDER TO ORDER FORM USING HEADERS......... IT IS CONNECTION ORIENTED SUCH AS ROUTER, ANTENNA ETC

**UDP - User Datagram Protocol**

UDP IS CONNECTION LESS ORIENTED, THIS IS JUST AN ALTERNATIVE SOLUTION TO TCP……AND FASTER TO SEND THE DATA FROM HOST TO DESTINATION AS IT DOESNT DETECT ERRORS LIKE TCP......

**The following constraints are commonly used in SQL:**

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

UNIQUE - Ensures that all values in a column are different

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

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

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

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

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

**Who is responsible to create a controller object in MVC.**

IControllerFactory interface is responsible for creating controller object. DefaultControllerFactory is its default framework provided implementation class. If you add a parameter less constructor to the HomeController class and set break point in that and run application with debug mode, you will find that the code execution process is hold on to that breakpoint.

you can see that IControllerFactory type variable named factory contains DefaultControllerFactory object. DefaultControllerFactory has various methods like Create, GetControllerInstance, CreateController those are playing vital role for creating HomeController object. You know that ASP.NET MVC framework is open source project, so if you want to know more details about those methods and their behavior then you can download its source code and see the implementation. ASP.NET MVC implements abstract factory design pattern for creating controller class. If you debug code and see the value with quick watch then you can see that DefaultControllerFactory is automatically created as CurrentControllerFactory (IControllerFactory).<o:p>

**Generics, why use them?**

Allows you to write code/use library methods which are type-safe, i.e. a List<string> is guaranteed to be a list of strings.

As a result of generics being used the compiler can perform compile-time checks on code for type safety, i.e. are you trying to put an int into that list of strings? Using an ArrayList would cause that to be a less transparent runtime error.

Faster than using objects as it either avoids boxing/unboxing

**ASP.NET has 3 kinds of caching:**

Output Caching,

Fragment Caching,

Data Caching

We can use **Page Output Caching** for those pages whose content is relatively static. So rather than generate a page on each user request, we can cache the page using page output caching so that it can be accessed from the cache itself. Pages can be generated once and then cached for subsequent fetches. Page output caching allows the entire content of a given page to be stored in the cache.

**Page Fragment Caching**: ASP.NET provides a mechanism for caching portions of pages, called page fragment caching. To cache a portion of a page, you must first encapsulate the portion of the page you want to cache into a user control. In the user control source file, add an OutputCache directive specifying the Duration and VaryByParam attributes. When that user control is loaded into a page at runtime, it is cached, and all subsequent pages that reference that same user control will retrieve it from the cache

**Data Caching**: Caching data can dramatically improve the performance of an application by reducing database contention and round-trips. Simply, data caching stores the required data in cache so that the web server will not send requests to the DB server every time for each and every request, which increases web site performance. I'd also add that you can also store user data in this cache provided you are aware of the limitations (the length of time the data is available for, for example) as well as data from many other kinds of data store.

## Suggested Uses of Caching Types

| **Situation** | **Suggested Caching Type** |
| --- | --- |
| The generated page generally stays the same, but there are several tables shown within the output that changes regularly. | Use fragment caching. |
| The generated page constantly changes, but there are a few objects that don’t change very often. | Use data caching for the objects. |
| The generated page changes every few hours as information is loaded into a database through an automated processes. | Use output caching and set the duration to match the frequency of the data changes. |

**What is difference between Partial View and layout?**

Layout is just like a master page, eg. navigation menu.

Partial view is a simple view without any layout associated, eg. table with data in view.

**Instance**: instance means just creating a reference(copy).

**object**: means when memory location is associated with the object (is a run-time entity of the class) by using the new operator.

In simple words, Instance refers to the copy of the object at a particular time whereas object refers to the memory address of the class.

**LINQ offers a common syntax for querying any type of data sources**

Secondly, it binds the gap between relational and object-oriented approachs

LINQ expedites development time by catching errors at compile time and includes IntelliSense & Debugging support

LINQ expressions are Strongly Typed.

**What we mean by Strongly Typed?**

Strongly typed expressions ensure access to values as the correct type at compile time & prevents type mismatch errors being caught when the code is compiled rather at run-time.

Try something like this

ALTER TABLE MainTable

ADD CONSTRAINT fk\_xyz

FOREIGN KEY (xyz)

REFERENCES ChildTable (xyz) ON DELETE CASCADE

I think u want some thing like this.

ON DELETE CASCADE Specifies that if an attempt is made to delete a row with a key referenced by foreign keys in existing rows in other tables, all rows that contain those foreign keys are also deleted.

**Force Table index to be used in sql**

**Syntax**

with (index(Visitor\_ID\_IX\_TK), nolock)

select \*

from #Conversion\_staging\_updated stage

join Visitors(nolock) v on v.Cookie\_ID=cast(stage.AddlInfo1 as varchar(50))

join TK TK **with (index(visitor\_id\_ix\_tk), nolock)** on tk.Visitor\_ID=v.Visitor\_ID

**View -DMl Statements used?**

Yes, View Is Virtual table. It has two types 1.simple view 2.complex view . In simple view We create view on single base table That's why we can perform all DML operations.it also called as Updatable view. But In case of Complex view We create view on multiple base tables that's why we cannot perform DML operations It is ReadOnly View (Only Select Operation).

**UDF-DMl Statements used?**

UDF's (User-defined functions) cannot be used to perform actions that modify the database state. We have to use only the code which will not change the state of the db table.

All the UDF's must be deterministic, that is the we cannot use any DML operations in UDF's.

**What is cross apply in sql**

Let’s say you have two tables, Customer and Order. Customers have many Orders.

I want to create a view that gives me details about customers, and the most recent order they've made. With just JOINS, this would require some self-joins and aggregation which isn't pretty. But with Cross Apply, its super easy:

SELECT \*

FROM Customer

CROSS APPLY (

SELECT TOP 1 \*

FROM Order

WHERE Order.CustomerId = Customer.CustomerId

ORDER BY OrderDate DESC

) T

**Unique key instead of primary key?**

+-----------------------------------------+-----------------------------------------------+

| Primary Key | Unique Key |

+-----------------------------------------+-----------------------------------------------+

| Primary Key can't accept null values. | Unique key can accept only one null value. |

+-----------------------------------------+-----------------------------------------------+

| By default, Primary key is clustered | By default, Unique key is a unique |

| index and data in the database table is | non-clustered index. |

| physically organized in the sequence of | |

| clustered index. | |

+-----------------------------------------+-----------------------------------------------+

| We can have only one Primary key in a | We can have more than one unique key in a |

| table. | table. |

+-----------------------------------------+-----------------------------------------------+

| Primary key can be made foreign key | In SQL Server, Unique key can be made foreign |

| into another table. | key into another table. |

+-----------------------------------------+-----------------------------------------------+

This can be done by creating a simple primary foreign key relationship and setting the foreign key column to unique in the following manner:

CREATE TABLE [Employee] (

[ID] INT PRIMARY KEY

, [Name] VARCHAR(50)

);

CREATE TABLE [Salary] (

[EmployeeID] INT UNIQUE NOT NULL

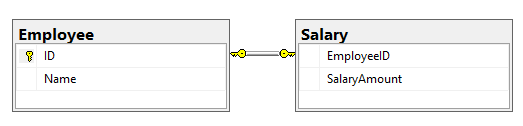
, [SalaryAmount] INT

);

ALTER TABLE [Salary]

ADD CONSTRAINT FK\_Salary\_Employee FOREIGN KEY([EmployeeID])

REFERENCES [Employee]([ID]);

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

INSERT INTO [Employee

# Differences Between SCOPE IDENTITY, IDENT CURRENT, and IDENTITY

<https://www.c-sharpcorner.com/UploadFile/8911c4/different-between-scope-identity-ident-current-and-identit/>

As for the difference between SCOPE\_IDENTITY and @@IDENTITY, suppose that you have a stored procedure P1 with three statements:  
- An INSERT that generates a new identity value  
- A call to a stored procedure P2 that also has an INSERT statement that generates a new identity value  
- A statement that queries the functions SCOPE\_IDENTITY and @@IDENTITY The SCOPE\_IDENTITY function will return the value generated by P1 (same session and scope). The @@IDENTITY function will return the value generated by P2 (same session irrespective of scope).

**difference between nvarchar varchar in sql**

**NVARCHAR** can store Unicode characters and takes 2 bytes per character.

<https://dev.to/twostepdevelopers/what-is-difference-between-varchar-nvarchar-4bki>

**About the Varchar:**

It is a variable that has a length data type.

It is use to store non-Unicode characters.

It Occupy 1 byte of space for each character.

**About the NVARCHAR data:**

It is a variable that has length data type.

It is use to store Unicode characters.

It Occupy 2 bytes of space for each character.

# [Correct use of transactions in SQL Server](https://stackoverflow.com/questions/10153648/correct-use-of-transactions-in-sql-server)

Add a try/catch block, if the transaction succeeds it will commit the changes, if the transaction fails the transaction is rolled back:

BEGIN TRANSACTION [Tran1]

BEGIN TRY

INSERT INTO [Test].[dbo].[T1] ([Title], [AVG])

VALUES ('Tidd130', 130), ('Tidd230', 230)

UPDATE [Test].[dbo].[T1]

SET [Title] = N'az2' ,[AVG] = 1

WHERE [dbo].[T1].[Title] = N'az'

COMMIT TRANSACTION [Tran1]

END TRY

BEGIN CATCH

ROLLBACK TRANSACTION [Tran1]

END CATCH

At the beginning of stored procedure one should put [SET XACT\_ABORT ON](https://msdn.microsoft.com/en-us/library/ms188792.aspx) to instruct Sql Server to automatically rollback transaction in case of error. If ommited or set to OFF one needs to test [@@ERROR](http://msdn.microsoft.com/en-us/library/aa933181%28v=sql.80%29.aspx) after each statement or use [TRY ... CATCH rollback](http://msdn.microsoft.com/en-us/library/ms175976.aspx) block.

SET XACT\_ABORT ON -- Turns on rollback if T-SQL statement raises a run-time error.

**Disadvantage of temp table**

Temporary tables require extra resources to create and maintain the data within them. These extra resources come from one of the hardest hit areas on most servers, tempdb. Also temporary tables and the statistics maintained in them can lead to recompiles. Recompiles cause quite a lot of blocking when they occur which can lead to other problems.

**Index not making execution faster, and in some cases is slowing down the query. Why is it so?**

**What is poor indexing?**

Any SQL Server table configuration where performance suffers due to excessive, improper, or missing indexes is considered to be poor indexing.

If indexes are not properly created, SQL Server has to go through more records in order to retrieve the data requested by a query. Therefore, it uses more hardware resources (processor, memory, disk, and network) and obtaining the data lasts longer.

A wrong index can be an index created on a column that doesn’t provide easier data manipulation or an index created on multiple columns which instead of speeding up queries, slows them down.

A table without a clustered index can also be considered as a poor indexing practice. Execution of a SELECT statement, inserting, updating, and deleting records is in most cases slower on a heap table than on a clustered one.

**Warning vs Error**

A warning tends to be a notice (either from the compiler or a library) that does not imply a fatal error, but instead something that could go wrong (eg memory leakage, unused variables) or something that might not work in the future (eg deprecation). An error means that something went wrong, either recoverably or unrecoverably. A recoverable error is one after which the program can (probably) still continue whereas an unrecoverable error is one which causes the program to 'hang' or 'crash'.

**Runtime and compile time error**

A **run time error** will only occur when the code is actually running. These are the most difficult - and lead to program crashes and bugs in your code which can be hard to track down.

Run time errors are errors of logic primarily. Due to something the programmer has overlooked, the program crashes e.g. division by 0, accessing a variable without initializing it first etc.

**An example** might be trying to convert a string: "hello" into an integer:

string helloWorld = "hello";

int willThrowRuntimeError = Convert.ToInt32(helloWorld);

The compiler may not see this as a problem but when run an error will be thrown.

**Compile time errors** are errors of syntax and semantics.

Compiler errors are due to inaccuracies in code, where the compiler throws an error to alert you to something which will not compile, and therefore cannot be run.

An example of a compiler error would be:

int = "this is not an int";

**How to set DateTime to null**

DateTime is a non-nullable value type

DateTime? newdate = null;

**Why to use External StyleSheet?**

It's much easier to reuse your CSS code if you have it in a separate file. Instead of typing the same CSS code on every web page you have, simply have many ===pages refer to a single CSS file with the "link" tag. You can make drastic changes to your web pages with just a few changes in a single CSS file

**Stylsheet Attribute Override?**

<div style="background: red;">

The inline styles for this div should make it red.

</div>

We can fight that with this:

div[style] {

background: yellow !important;

}

**CustomExceptionFilter in MVC**

Public class CustomExceptionFilter : ExceptionFilterAttribute

Exception filters are filters that can be used to handle unhandled exceptions that are generated in your Web API controller methods. In other words, you can use exception filters to catch unhandled exceptions in Web API that can originate from your controller methods. Note that a global error filter is a good approach to handle exceptions in your Web API if unhandled exceptions are thrown and not handled in your controller methods.

You can register your exception filters in one of the following three ways:

At the action level

At the controller level

Globally

GlobalConfiguration.Configuration.Filters.Add(new DBFilterAttribute());

public class NotImplExceptionFilterAttribute : ExceptionFilterAttribute

{

public override void OnException(HttpActionExecutedContext context)

{

if (context.Exception is NotImplementedException)

{

context.Response = new HttpResponseMessage(HttpStatusCode.NotImplemented);

}

}

**State of object**

State - Instance Variable, the variable you use in the class.

Behavior - Method/function, the method/function you use in the class.

state tells us about the type or the value of that object where as behaviour tells us about the **operations or things that the object can perform.**

For example let’s say we have an Object called car

so car object will have colour, engine type, wheels etc as it’s state,

this car object can run at 180kmph, it can turn right and left, it can go back and forth, it can carry 4 people etc. These are it’s behaviours.

**JQUERY AJAX POST EXAMPLE**

var  formData = "name=ravi&age=31";  //Name value Pair

    or

var formData = {name:"ravi",age:"31"}; //Array

$.ajax({

    url : "AJAX\_POST\_URL",

    type: "POST",

    data : formData,

    success: function(data, textStatus, jqXHR)

    {

        //data - response from server

    },

    error: function (jqXHR, textStatus, errorThrown)

    {

    }

});

**JQUERY AJAX GET EXAMPLE**

$.ajax({

url: "https://app.asana.com/-/api/0.1/workspaces/",

type: 'GET',

dataType: 'json', // added data type

success: function(res) {

console.log(res);

alert(res);

}

});

**Difference in text and val in jquery**

.val() works on input elements (or any element with a value attribute?) and .text() will not work on input elements. .val() gets the value of the input element -- regardless of type. .text() gets the innerText (not HTML) of all the matched elements:

**.text()**

The result is a string that contains the combined text contents of all matched elements. This method works on both HTML and XML documents. Cannot be used on input elements. For input field text use the val attribute.

**.val()**

Get the content of the value attribute of the first matched element

.val() function returns value from input element and .text() function returns value from other than input elements. We can also pass string argument to these functions to **set the value of the calling element**. Below code shows how to set value to DOM elements using .val() and .text() functions:

HTML Part:

<form id="form1"><input id="first" type="text" /><input type="submit" /></form>

<div id="second">Click the "Submit Query" to see it work</div>

Jquery Part:

$(document).on("submit", "form", function (e) {

$("#first").val("This input is set by .val() function");

$("#second").text("A new text is set using .text() function!");

return false;

})

# [Set timeout for ajax (jQuery)](https://stackoverflow.com/questions/5225597/set-timeout-for-ajax-jquery)

You could use the timeout setting in the ajax options like this:

$.ajax({

url: "test.html",

timeout: 3000,

error: function(){

//do something

},

success: function(){

//do something

}

});

**Ajax Events**

ajaxStart (Global Event)

This event is triggered if an Ajax request is started and no other Ajax requests are currently running.

beforeSend (Local Event)

This event, which is triggered before an Ajax request is started, allows you to modify the XMLHttpRequest object (setting additional headers, if need be.)

ajaxSend (Global Event)

This global event is also triggered before the request is run.

success (Local Event)

This event is only called if the request was successful (no errors from the server, no errors with the data).

ajaxSuccess (Global Event)

This event is also only called if the request was successful.

error (Local Event)

This event is only called if an error occurred with the request (you can never have both an error and a success callback with a request).

ajaxError (Global Event)

This global event behaves the same as the local error event.

complete (Local Event)

This event is called regardless of if the request was successful, or not. You will always receive a complete callback, even for synchronous requests.

ajaxComplete (Global Event)

This event behaves the same as the complete event and will be triggered every time an Ajax request finishes.

ajaxStop (Global Event)

This global event is triggered if there are no more Ajax requests being processed.

**Jquery Size() vs Length**

Size and length both returns the number of element in an object. But length is faster than the size because length is a property and size is a method.

jQuery .size() method returns number of element in the object. But it is not preferred to use the size()method as jQuery provide .length property and which does the same thing. But the .length property is preferred because it does not have the overhead of a function call

jquery.size() and jquery.length

**===Crest**

**How to find count of word in paragraph in linq**

myDbText.Split(' ').Where(token => token.Equals(word)).Count();

**find replace word using linq in c#**

TestList.ForEach(x => x.TestType.Replace("", "DataNotAvailable"));

or the one below

foreach (TestModel item in TestList.Where(x => x.ID == ""))

{

item.ID = "NoDataAvailable";

}

**convert list to dictionary c#**

Try this:

var res = list.ToDictionary(x => x, x => x);

The first lambda lets you pick the key, the second one picks the value.

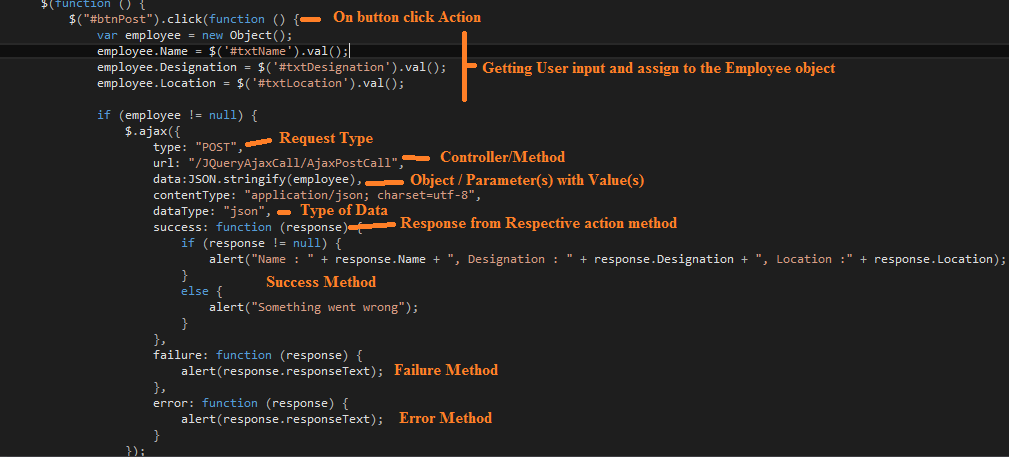
You can play with it and make values differ from the keys, like this:

var res = list.ToDictionary(x => x, x => string.Format("Val: {0}", x));

If your list contains duplicates, add Distinct() like this:

var res = list.Distinct().ToDictionary(x => x, x => x);

**How ajax request mapped MVC control Action**



The JSON.stringify() method converts a JavaScript value to a JSON string. It's optional when we are just passing parameter(s) with Values(s)

Use [FromBody] if you are dev in .net core 2.0 this help to avoid having null values when passing from ajax to controller method ;)

**Find third max salary in SQL\LINQ**

var ThirdHighestSal = from vr in Employee.getAllEmployee()

group vr by vr.Salary into gr

orderby gr.Key descending

select( new {salary=gr.OrderBy(x=>x.Salary).Skip(2).First()});

In SQL Server 2012+, OFFSET...FETCH would be an efficient way to achieve this:

DECLARE @N AS INT;

SET @N = 3;

SELECT

EmpSalary

FROM

dbo.Salary

ORDER BY

EmpSalary DESC

OFFSET (@N-1) ROWS

FETCH NEXT 1 ROWS ONLY

SELECT TOP 1 salary

FROM employee

WHERE salary IN (SELECT DISTINCT TOP 3 salary

FROM employee

ORDER BY salary DESC)

ORDER BY salary ASC

**Best Way**

SELECT \* FROM tableName ORDER BY columnName DESC LIMIT 2, 1

## Types of Cursors in SQL

There are the following two types of cursors in SQL:

Implicit Cursor

Explicit Cursor

**Implicit Cursor**

These types of cursors are generated and used by the system during the manipulation of a DML query (INSERT, UPDATE and DELETE). An implicit cursor is also generated by the system when a single row is selected by a SELECT command.

**Explicit Cursor**

 This type of cursor is generated by the user using a SELECT command. An explicit cursor contains more than one row, but only one row can be processed at a time. An explicit cursor moves one by one over the records. An explicit cursor uses a pointer that holds the record of a row. After fetching a row, the cursor pointer moves to the next row.

**What is distributed transaction**

A distributed transaction is a transaction that works across several computers. Say you start a transaction in some method in a program on computer A. You then make some changes to data in the method on computer A, and afterwords the method calls a web service on computer B. The web service method on computer B fails and rolls the transaction back. Since the transaction is distributed, this means that any changes made on computer A also need to be rolled back. The combination of the distributed transaction coordinator on windows and the .net framework facilitate this functionality.

**===TSYS**

**Inbulid Interfaces you used .net framework used**

public sealed class String : IComparable, ICloneable, IConvertible, IComparable<string>, IEnumerable<char>, IEnumerable, IEquatable<string>

public abstract class Stream : MarshalByRefObject, IDisposable

public class Hashtable : IDictionary, ICollection, IEnumerable, ISerializable, IDeserializationCallback, ICloneable

IList, ICollection, IEnumerable, ICloneable

**IDisposable interface**This interface is very powerful if your class works with unmanaged resources (files, streams, database connections .etc.). This interface contains one method.  
void Dispose();

**.pdb file use**

PDB files are generated when you build your project. They contain information relating to the built binaries which Visual Studio can interpret.

When a program crashes and it generates a crash report, Visual Studio is able to take that report and link it back to the source code via the PDB file for the application. PDB files must be built from the same binary that generated the crash report!

**Program database** (**PDB**) is a [proprietary](https://en.wikipedia.org/wiki/Proprietary_software) file format (developed by [Microsoft](https://en.wikipedia.org/wiki/Microsoft)) for storing debugging information about a program (or, commonly, program modules such as a [DLL](https://en.wikipedia.org/wiki/Dynamic-link_library) or [EXE](https://en.wikipedia.org/wiki/EXE)). PDB files commonly have a .pdb [extension](https://en.wikipedia.org/wiki/Filename_extension). A PDB file is typically created from source files during compilation. It stores a list of all [symbols](https://en.wikipedia.org/wiki/Symbol_(computing)) in a module with their addresses and possibly the name of the file and the line on which the symbol was declared. This symbol information is not stored in the module itself, because it takes up a lot of space.

I think a fair argument is also that **not** leaving the PDBs on the live servers is a risk. In the case where production is crashing and the problems can't be reproduced on dev or UAT, it's much more time consuming (and perhaps impossible) to diagnose where the error is occurring.

At the very least, the **PDBs that match the deployed DLLs** should be in a ZIP file on the production server somewhere. They should be easily located by people other than yourself in case you aren't around to assist.

**Managed code and unmanaged code in .NET**

**Unmanaged Code :-**

1.The code, which is developed outside .NET, Framework is known as unmanaged code.

2.Applications that do not run under the control of the CLR are said to be unmanaged, and certain languages such as C++ can be used to write such applications, which, for example, access low - level functions of the operating system. Background compatibility with code of VB, ASP and COM are examples of unmanaged code.

3.Unmanaged code is executed with help of wrapper classes.

4.Wrapper classes are of two types: CCW (COM Callable Wrapper) and RCW (Runtime Callable Wrapper).

5.Wrapper is used to cover difference with the help of CCW and RCW. Managed Code The resource, which is with in your application domain is, managed code. The resources that are within domain are faster.

**Managed Code** 1.The code, which is developed in .NET framework, is known as managed code. This code is directly executed by CLR with help of managed code execution. Any language that is written in .NET Framework is managed code.

2.Managed code uses CLR which in turns looks after your applications by managing memory, handling security, allowing cross - language debugging, and so on.

**Why singleton and not static**

static classes are not for anything that needs state. It is useful for putting a bunch of functions together i.e Math (or Utils in projects). So the class name just gives us a clue where we can find the functions and nothing more.

Singleton is my favorite pattern and I use it to manage something at a single point. It's more flexible than static classes and can maintain it's state. It can implement interfaces, inherit from other classes and allow inheritance.

My rule for choosing between static and singleton:

If there is a bunch of functions that should be kept together, then static is the choice. Anything else which needs single access to some resources, could be implemented as a singleton.

Singletons are easier to work with when unit testing a class. Wherever you pass singletons as a parameter (constructors, setters or methods) you can instead substitute a mocked or stubbed version of the singleton.

Lazy Loading

Support of interfaces, so that separate implementation can be provided

Ability to return derived type (as a combination of lazyloading and interface implementation)

Singleton object stores in Heap but, static object stores in stack.

We can clone the object of Singleton but, we can not clone the static class object.

Singleton can use the Object Oriented feature of polymorphism but static class cannot.

**===FIS Global**

**What is Reflection in C#?**

Reflection provides objects (of type Type) that describe assemblies, modules and types. We can use reflection to dynamically create an instance of a type, bind the type to an existing object, or get the type from an existing object and invoke its methods or access its fields and properties. If we use attributes in our code, reflection enables us to access them.

Here's a simple example of reflection using the static method GetType - inherited by all types from the Object base class - to obtain the type of a variable,

// Using GetType to obtain type information:

int i = 42;

System.Type type = i.GetType();

System.Console.WriteLine(type);

The output is,

System.Int32

The following example uses reflection to obtain the full name of the loaded assembly.

// Using Reflection to get information from an Assembly:

System.Reflection.Assembly info = typeof(System.Int32).Assembly;

System.Console.WriteLine(info);

The output is,

mscorlib, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089

|  |  |
| --- | --- |
| **Assembly.LoadFrom()** | **Assembly.LoadFile()** |
|          LoadFrom() can use either the name or the path of the assembly, It uses a probing algorithm to find the assembly if, you have two assemblies that have the same identity but different locations, you can get some unexpected behavior. |          LoadFile() will expect the path of the assembly by using LoadFile(), you can load the desired assembly as needed. |
|          LoadFrom() goes through Fusion and can be redirected to another assembly at a different path but with that same identity if one is already loaded in the LoadFrom context. |          LoadFile() doesn't bind through Fusion at all - the loader just goes ahead and loads exactly what the caller requested. It doesn't use either the Load or the LoadFrom context. |

**typeof** keyword takes the Type itself as an argument and returns the underline Type of the argument whereas GetType() can only be invoked on the instance of the type.  
  
**typeof**  
*System.Type t1= typeof(Employee); // Employee is a Type*  
  
**GetType()**  
*Employee employee= new Employee();  
System.Type t2= employee.GetType(); // employee is an instance of the type Employee.*  
  
Both of examples create an instance of a type class for the Employee type. After you have a reference to the type, you can extract extra information from them as shown below.

1. Console.WriteLine("AssmeblyQualifiedName: {0}", t1.AssemblyQualifiedName);
2. Console.WriteLine("FullName: {0}", t1.FullName);
3. Console.WriteLine("IsValueType: {0}", t1.IsValueType);
4. Console.WriteLine("Name: {0}", t1.Name);
5. Console.WriteLine("Namespace: {0}", t1.Namespace);

**Closures in JavaScript with Example**

A closure is an inner function that has access to the variables in the outer (enclosing) function’s scope chain. The closure has access to variables in three scopes; specifically: (1) variable in its own scope, (2) variables in the enclosing function’s scope, and (3) global variables.

Here is an example:

var globalVar = "xyz";

(function outerFunc(outerArg) {

var outerVar = 'a';

(function innerFunc(innerArg) {

var innerVar = 'b';

console.log(

"outerArg = " + outerArg + "\n" +

"innerArg = " + innerArg + "\n" +

"outerVar = " + outerVar + "\n" +

"innerVar = " + innerVar + "\n" +

"globalVar = " + globalVar);

})(456);

})(123);

In the above example, variables from innerFunc, outerFunc, and the global namespace are **all** in scope in the innerFunc. The above code will therefore produce the following output:

outerArg = 123

innerArg = 456

outerVar = a

innerVar = b

globalVar = xyz

**Does JavaScript is a loosely typed** ?

JavaScript is a **loosely typed** language, meaning you don’t have to specify what type of information will be stored in a variable in advance. JavaScript automatically types a variable based on what kind of information you assign to it (e.g., that '' or " " to indicate string values). Many other languages, like Java, require you to declare a variable’s type, such as int, float, boolean, or String.

Unlike Java or C, which are strongly typed languages, Javascript is smart and hence is also called a weakly typed language. You don’t have to specify the data-type of your variable while declaring it. Javascript is smart enough to interpret what variable type you want to use.

For eg:

var x = 5;

var y = “5”

this will store x as a 64-Bit Double as that is the standard format in which numbers are stored in JS variables. However y will store a string “5” because of the Quotations “”.

in Java, being a strongly typed language you have to specify the datatype of your variable

For eg:

int x = 5;

String y = “5”;

this is Strongly typed as you can clearly see.

Note:

Be weary of the fact when comparing such variables. You should use strict equality operator “===” rather than just the regular “==” ones because they will give you an incorrect boolean in return. Also be careful when you want to change your variable from one type to the other in your program(int to String etc etc). It is generally a bad practice to keep changing your variable datatype

**Attribute in C# can we inherit them**

When Inherited = true (which is the default) it means that the attribute you are creating can be inherited by sub-classes of the class decorated by the attribute.

So - if you create MyUberAttribute with [AttributeUsage (Inherited = true)]

[AttributeUsage (Inherited = True)]

MyUberAttribute : Attribute

{

string \_SpecialName;

public string SpecialName

{

get { return \_SpecialName; }

set { \_SpecialName = value; }

}

}

Then use the Attribute by decorating a super-class...

[MyUberAttribute(SpecialName = "Bob")]

class MySuperClass

{

public void DoInterestingStuf () { ... }

}

If we create an sub-class of MySuperClass it will have this attribute...

class MySubClass : MySuperClass

{

...

}

Then instantiate an instance of MySubClass...

MySubClass MySubClassInstance = new MySubClass();

Then test to see if it has the attribute...

MySubClassInstance <--- now has the MyUberAttribute with "Bob" as the SpecialName value.

**Attribute inheritance is enabled by default.**

You can change this behavior by:

[AttributeUsage (Inherited = False)]

**Global variable in JavaScript**

Yes, as the others have said, you can use var at global scope (outside of all functions) to declare a global variable:

<script>

var yourGlobalVariable;

function foo() {

// ...

}

</script>

Alternately, you can assign to a property on window:

<script>

function foo() {

window.yourGlobalVariable = ...;

}

</script>

...because in browsers, all global variables global variables declared with var are properties of the window object.

**Cross Side Scripting**

[**https://portswigger.net/web-security/cross-site-scripting**](https://portswigger.net/web-security/cross-site-scripting)

**JsonRequestBehavior.AllowGet**

By default, the ASP.NET MVC framework does not allow you to respond to an HTTP GET request with a JSON payload. If you need to send JSON in response to a GET, you'll need to explicitly allow the behavior by using JsonRequestBehavior.AllowGet as the second parameter to the Json method. However, there is a chance a malicious user can gain access to the JSON payload through a process known as JSON Hijacking. You do not want to return sensitive information using JSON in a GET request.  
  
Suppose if we have method like below

[HttpPost]

public JsonResult amc(){}

By default it "Deny Get".  
In the below method

public JsonResult amc(){}

When you need to allowget or use get ,we have to use JsonRequestBehavior.AllowGet.

public JsonResult amc()

{

return Json(new Modle.JsonResponseData { Status = flag, Message = msg, Html = html }, JsonRequestBehavior.AllowGet);

}

**Is the Stored procedure is pre-compiled ?**

Yes a StoredProcedure is Pre-Compiled. When you run a normal Query in your SQL management Studio e.g

select \* from My table

SQL needs to find a best possible way of retrieving your data and it needs to also check if you have indexes that can help bring back the data. This is called Execution plan. Now in a StoredProcedure these steps are not performed because they are determined , the stored procedure stores the execution plan and it just use the plan to run the stored procedure.

Stored procedure are not pre-compiled, but compiled only during the first run. For every subsequent runs, it is for sure pre-compiled.  
When we create a Stored procedure, first time and we will find that there are not caching entry for the executions of the Stored procedure.  
After running the stored procedure for the first time, the entry for the cache is made in the system.  
That's why Stored Procedure takes long time to run for first time.

if you want to recompile that SP then You can Use  
WITH RECOMPILE  
Keyword while creating SP

**===TechM**

# [How to keep a default page with Forms Authentication?](https://stackoverflow.com/questions/7136961/how-to-keep-a-default-page-with-forms-authentication)

# Your issue is that you've got Default.aspxspecified as the "loginUrl" property within your "Forms" authentication tag. Your second issue is probably that you are not allowing anonymous users access to Default.aspx.

# Modify your web.config to look like this:

<authentication mode="Forms">

<forms defaultUrl="Default.aspx" loginUrl="Login.aspx" timeout="20" protection="None" cookieless="UseCookies"/>

</authentication>

# [Multiple ActionFilterAttribute order of execution assured?](https://stackoverflow.com/questions/24828072/multiple-actionfilterattribute-order-of-execution-assured)

In **MVC5** you inherit from **ActionFilter** and indicate the order (**using Order** property of ActionFilter) in the custom attribute like this:

[Common.PortalSecurity.Login(Order=1)]

[Common.PortalSecurity.UserRole(Order=2)]

public HttpResponseMessage GetAll(string sessionToken)

{

return new HttpResponseMessage();

}

**AntiForgeryToken how it works**

<http://blog.at-dot.net/archive/2014/05/13/mvc-what-is-html-dot-antiforgerytoken-and-how-does-it-actually-work/>

So how does it actually work?

Under the covers, the call to Html.AntiForgeryToken() does two things. First, it causes a hidden input to be created in the form, like this:

**<input name="\_\_RequestVerificationToken" type="hidden" value="MRRR4ga6eu8EiCeHKdJAiZgF1EO4OC52U3L6UUe\_nqIfe7I7jONPmsJ2Vc\_chDi7gA7drUYy4hyehi\_UDkso2wrbmLgaKXGypM0rAfHec7g1" />**

Second, it causes a cookie also called \_\_RequestVerificationToken to be created with the same value.

When the page gets posted to the server, the ValidateAntiForgeryToken attribute causes the MVC framework to marry the hidden form parameter with the cookie value. If they are not the same, or either one is missing, the server knows that the post did not come from our own server and that we shouldn’t trust the input, causing an exception to be thrown.

The magic that occurs here is that the browser will only send the cookie if it’s posting to the domain that created the cookie. Thus, if a user inputs data into an untrusted site, which then posts back to our site, the anti-forgery token on the page will not match the one previously created and stored in the cookie. When MVC smells something fishy, it throws an exception, protecting our user from an attack.

**Include antiforgerytoken in ajax post ASP.NET MVC**

**@using (Html.BeginForm(null, null, FormMethod.Post, new { id = "\_\_AjaxAntiForgeryForm" }))**

**{**

**@Html.AntiForgeryToken()**

**}**

<div id="myDiv" data-url="@Url.Action("Index", "Home")">

Click me to send an AJAX request to a controller action

decorated with the [ValidateAntiForgeryToken] attribute

</div>

<script type="text/javascript">

$('#myDiv').submit(function () {

**var form = $('#\_\_AjaxAntiForgeryForm');**

var token = $('input[name="\_\_RequestVerificationToken"]', form).val();

$.ajax({

url: $(this).data('url'),

type: 'POST',

data: {

\_\_RequestVerificationToken: token,

someValue: 'some value'

},

success: function (result) {

alert(result.someValue);

}

});

return false;

});

</script>

**In TextARea how to check malicious script and avoid posting in MVC**

In .NET 4.5+ or by adding System.Web.Security.AntiXss to the older version of .NET, there is a good way to address this issue. We can use [AllowHtml] and a custom annotation attribute together. The approach should whitelist the HTML tags inside the string and validate the request.

[AllowHtml]

[RemoveScript(regexPattern: @"\<((?=(?!\b(a|b|i|p|u|br|hr)\b))(?=(?!\/\b(a|b|i|p|u)\b))).\*?\>")]

public string Body { get; set; }

**JSON.stringify()**

The JSON.stringify() method converts a JavaScript object or value to a JSON string, optionally replacing values if a replacer function is specified or optionally including only the specified properties if a replacer array is specified.

**In which language javascript built**

Javascript is an implementation of the [ECMAScript](http://en.wikipedia.org/wiki/ECMAScript) standard, but there is no singular canonical interpreter like you see with PHP.

Most of the major implementations (standalone or as parts of web browsers) out there tend to be largely written in C or C++ for performance reasons, but that's not necessarily always the case. Rhino, an engine maintained by Mozilla, is written in Java.

**Why generics when we can achieve in non-generics**

One thing I think you need to consider is that a generic collection is not always a drop in replacement for a non-generic collection. For example, Dictionary<object,object> can not simply be plugged in for an instance of Hashtable. They have very different behavior in a number of scenarios that can and will break programs. Switching between these two collections forces a good programmer to examine the use cases to ensure the differences do not bite them.

**Why is HTTPS required?**

To verify whether the website is authenticated/certified or not (uncertified websites can do evil things). An authenticated website has a unique personal certificate purchased from one of the **CA**’s.

**Who are CA’s (Certificate Authorities)?**

CA’s are globally trusted companies like GoDaddy, GeoTrust, VeriSign etc who provide digital certificates to the websites.

**1. To validate the certificate of a website:**

https://i.stack.imgur.com/wuSdJ.png

1) When you enter the URL www.Google.com, Google’s server gives its public key and certificate (which was signed by GeoTrust) to the Browser.

2) Now browser has to verify the authenticity of the certificate i.e. it’s actually signed from GeoTrust or not. As browsers come with a pre-installed list of public keys from all the major CA’s, it picks the public key of the GeoTrust and tries to decrypt the digital signature of the certificate which was encrypted by the private key of GeoTrust.

3) If it’s able to decrypt the signature (which means it’s a trustworthy website) then it proceeds to the next step else it stops and shows a red cross before the URL.

**2. To create a secure connection (encrypts outgoing and incoming data) so that no one else can read it:**

[enter image description here](https://i.stack.imgur.com/PtiF0.png)

1) As I mentioned, Google sends its public key when you enter www.Google.com . Any data encrypted with this public key can only be decrypted by Google’s private key which Google doesn’t share with anyone.

2) After validating the certificate, browser creates a new key let’s call it **Session Key** and make 2 copies of it. These keys *can encrypt as well as decrypt the data*.

3) The browser then encrypts *(1 copy of session key + other request data)* with the Google's public key . Then it sends it back to the Google server.

4) Google’s server decrypts the encrypted data using its private key and gets the session key , and other request data.