1) Pre-JIT - Complies complete source code into native code at the time of deployment.

2) Econo-JIT - Complies methods that are called at runtime.

3) Normal-JIT - Complies methods that are called at runtime and get stored in cache. Next time when the same method is called, it will be taken from cache.

To install assembly in Cache, use Gacutil. To run Gacutil, goto "Visual Studio Command Prompt" and type "gacutil -i <assembly\_name>", where (assembly\_name) is the DLL name of the project. To uninstall assembly, type gacutil –u <assembly name> in Visual Studio Command Prompt.

A satellite assembly are used when multilingual (UI) application are created. Satellite assembly is a compiled library that contains localized resources which provides us with the capability of designing and deploying solutions to multiple cultures, rather than hard coding texts, bitmaps etc

We can prevent .NET DLL to be decompiled upto an extent by Obfuscate Source code, asymmetric encryption and encrypted w32 wrapper application.

Code Document Object Model are code generators which are used to minimize repetitive coding tasks, and to minimize the number of human-generated source code lines.

Ngen.exe creates compiled processor-specific machine code called native images which are files and installs them into the native image cache on the local computer. The runtime will use native images from the cache rather than using the JIT compiler to compile the original assembly.

CLS - VB case sensitive and C# is not

CTS- int in vb and c# have different syntax but get compiled to common data type

Expando is a dynamic type to which members can be added (or removed) at runtime. dynamicis designed to allow .NET to interoperate with types when interfacing with dynamic typing languages such as Python and JavaScript.

So, if you need to handle a dynamic type: use dynamic and if you need to handle dynamic data such as XML or JSON: use ExpandoObject

Application domain is the boundary within which an application runs. A process can contain multiple application domains. Application domains provide an isolated environment to applications that is similar to the isolation provided by processes. An application running inside one application domain cannot directly access the code running inside another application domain. To access the code running in another application domain, an application needs to use a proxy.

AppDomains are usually created by hosts. Examples of hosts are the Windows Shell, ASP.NET and IE. When you run a .NET application from the command-line, the host is the Shell. The Shell creates a new AppDomain for every application. AppDomains can also be explicitly created by .NET applications.

The AppDomain class is used to create and terminate Application Domains, load and unload assemblies and types and enumerates assemblies and threads in a domain.

CreateDomain(),CreateInstance(),Load()

You need to assign a strong name to an assembly to place it in the GAC and make it globally accessible. A strong name consists of a name that consists of an assembly's identity (text name, version number, and culture information), a public key and a digital signature generated over the assembly. The .NET Framework provides a tool called the Strong Name Tool (Sn.exe), which allows verification and key pair and signature generation.

ResGen.exe is a tool that is used to convert resource files in the form of .txt or .resx files to common language runtime binary .resources files that can be compiled into satellite assemblies.

CAS - Defines permissions and permission sets that represent the right to access various system resources.

The caspol tool grants and modifies permissions to code groups at the user policy, machine policy, and enterprise policy levels.

The CLR maintains a table called the intern pool that contains the literal strings in a program. This ensures that repeated use of the same constant strings in your code will utilize the same string reference. The System.String class provides an Intern method that ensures a string is in the intern pool and returns the reference to it.

DLL Hell describes the difficulty in managing DLLs on a system; this includes multiple copies of a DLL.

DLL Hell problem is solved in .NET with the introduction of Assembly versioning.

- It allows the application to specify not only the library it needs to run, but also the version of the assembly.

- Versioning is the technique to provide the .dll file to prevent them from replacement.

Web.config

It is used to store the application level configuration. Sometimes it inherits the setting from the machine.config. It parses at runtime, means if you make any changes then web application will load all the settings in the config file.

App.config

It is also a special type of configuration file which is basically used with Windows Services, Windows application, Console Apps or it can be WPF application or any others.

When you run the application which contains the app.config, at the time of compilation a copy of app.config with different name moves into build folder for running the application, So that's why we need to restart the program if any changes made in app.config.

Machine.config

It is a special type of configuration file which creates into the OS when you install visual studio. This stores machine level configuration setting.

int i = 7;

Console.WriteLine("Actual");

Console.WriteLine(i); //7

Console.WriteLine("Before");

Console.WriteLine(i++); //7

Console.WriteLine("After");

Console.WriteLine(i); //8

int i = 7;

Console.WriteLine("Actual");

Console.WriteLine(i); //7

Console.WriteLine("Before");

Console.WriteLine(++i); //8

Console.WriteLine("After");

Console.WriteLine(i); //8

 Application pool which contains the multiple worker process called**“Web Garden”.**

when there is huge numbers of  incoming traffic for your web sites, one standalone server is not sufficient to process the request.  You may need to use multiple server to host the application and divide the traffic among them.  This is called **“Web Farm”**

[Stopwatch](http://msdn.microsoft.com/en-us/library/system.diagnostics.stopwatch.aspx) is designed for this purpose and is one of the best way to measure time execution in .NET.

var watch = System.Diagnostics.Stopwatch.StartNew();

// the code that you want to measure comes here

watch.Stop();

var elapsedMs = watch.ElapsedMilliseconds;

URL rewriting is often used for mapping old sets of URLs to a new set of URLs. Contrast that to Routing, which is focused on mapping a URL to a resource.

Response.Redirect, a new request is generated from client-side for redirected page. It's a kind of additional round trip. As new request is generated from client, so the new URL is visible to user in browser after redirection. Browser make the transfer happen.

Server.Transfer, a request is transferred from one page to another without making a round trip from client. For the end user, URL remains the same in browser even after transferring to another page. Server make the transfer happen.

<http://www.c-sharpcorner.com/UploadFile/akkiraju/url-rewriting-in-Asp-Net/>

A class can have multiple main methods but we need to specify startup object for the same

An application can contain assemblies compiled from different languages in dotnet

|  | **Server.Transfer** | **Response.Redirect** |
| --- | --- | --- |
| Redirection | Redirection is done by the server. | Redirection is done by the browser client. |
| Browser URL | Does not change. | Changes to the redirected target page. |
| When to use | Redirect between pages of the same server. | Redirect between pages on different server and domain. |

|  |  |  |
| --- | --- | --- |
|  | **Name** | **Description** |
| System_CAPS_pubmethod | [Equals(Object)](https://msdn.microsoft.com/en-us/library/bsc2ak47(v=vs.110).aspx) | Determines whether the specified object is equal to the current object. |
| System_CAPS_pubmethodSystem_CAPS_static | [Equals(Object, Object)](https://msdn.microsoft.com/en-us/library/w4hkze5k(v=vs.110).aspx) | Determines whether the specified object instances are considered equal. |
| System_CAPS_protmethod | [Finalize()](https://msdn.microsoft.com/en-us/library/system.object.finalize(v=vs.110).aspx) | Allows an object to try to free resources and perform other cleanup operations before it is reclaimed by garbage collection. |
| System_CAPS_pubmethod | [GetHashCode()](https://msdn.microsoft.com/en-us/library/system.object.gethashcode(v=vs.110).aspx) | Serves as the default hash function. |
| System_CAPS_pubmethod | [GetType()](https://msdn.microsoft.com/en-us/library/system.object.gettype(v=vs.110).aspx) | Gets the [Type](https://msdn.microsoft.com/en-us/library/system.type(v=vs.110).aspx) of the current instance. |
| System_CAPS_protmethod | [MemberwiseClone()](https://msdn.microsoft.com/en-us/library/system.object.memberwiseclone(v=vs.110).aspx) | Creates a shallow copy of the current Object. |
| System_CAPS_pubmethodSystem_CAPS_static | [ReferenceEquals(Object, Object)](https://msdn.microsoft.com/en-us/library/system.object.referenceequals(v=vs.110).aspx) | Determines whether the specified Object instances are the same instance. |
| System_CAPS_pubmethod | [ToString()](https://msdn.microsoft.com/en-us/library/system.object.tostring(v=vs.110).aspx) | Returns a string that represents the current object. |

IEnumerable is the base interface for all non-generic collections that can be enumerated.

IEnumerable contains a single method, GetEnumerator, which returns an IEnumerator.

IEnumerator provides the ability to iterate through the collection by exposing a Current property and MoveNext and Reset methods.

IEquatable interface is implemented by types whose values can be equated (for example, the numeric and string classes).

A value type or class implements the Equals method to create a type-specific method suitable for determining equality of instances.

The IEquatable<T> interface is used by generic collection objects such as Dictionary<TKey, TValue>, List<T>, and LinkedList<T> when testing for

equality in such methods as Contains, IndexOf, LastIndexOf, and Remove. It should be implemented for any object that might be stored in a generic collection.

IComparable interface is implemented by types whose values can be ordered or sorted. It requires that implementing types define a single method, CompareTo(Object), that indicates whether the position of the current instance in the sort order is before, after, or the same as a second object of the same type. The instance's IComparable implementation is called automatically by methods such as Array.Sort and ArrayList.Sort.

The implementation of the CompareTo(Object) method must return an Int32 that has one of three values, as shown in the following table.

ICloneable interface enables you to provide a customized implementation that creates a copy of an existing object.

The ICloneable interface contains one member, the Clone method, which is intended to provide cloning support beyond that supplied by Object.MemberwiseClone.

The ICloneable interface simply requires that your implementation of the Clone method return a copy of the current object instance.

It does not specify whether the cloning operation performs a deep copy, a shallow copy, or something in between.

Nor does it require all property values of the original instance to be copied to the new instance.

IConvertible interface provides methods to convert the value of an instance of an implementing type to a common language runtime type that has an equivalent value.

The common language runtime also uses the IConvertible interface internally, in explicit interface implementations,

to simplify the code used to support conversions in the Convert class and basic common language runtime types.