**.Net Framework Part 5**

**34. How will you decompile your assembly?**

Any assembly can be disassembled using ILDASM(Intermediate Language Disassembler), it is ships with the .Net framework SDK. Using third party tools like Reflector or Anakrino can also be easily decompile the assemblies.

**36. If we have two different version of same assembly in GAC how do we make a choice?**

Let us consider the scenario where one of the applications uses the dll which is available in GAC. Now we are creating the second version of the same dll and placed inside the GAC. So GAC contains both version of the assembly, since application referring the dll from GAC, definitely it will take latest version of the dll. But we need old version of the assembly to be executed. How to achieve this requirement?

**Answer:** using < bindingRedirect > tag in App.config file

**Example:**

**Step 1:** Create sample library class with MyVersion() method. This method will return current version of the assembly.

namespace AssemblyVersionExample

{

public class Class1

{

public string MyVersion()

{

return "The old version: 1.0.0.10";

}

}

}

**Step 2:**Modify the "AssemblyVersion" attribute with the old version say ‘"1.0.0.10"

// You can specify all the values or you can default the Build and Revision Numbers

// by using the '\*' as shown below:

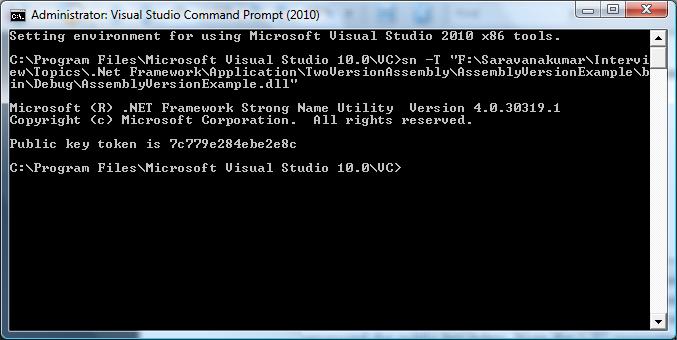
// [assembly: AssemblyVersion("1.0.\*")]

[assembly: AssemblyVersion("1.0.0.10")]

[assembly: AssemblyFileVersion("1.0.0.10")]

**Step 3:**Compile the dll and register to assembly using **"gacutil"**

**Step 4:**Create a public token key using following command**[ sn -T "filepath"]**. Now the public key for the assembly is created.



**Step 5:**Repeat the step 2,3 with different version for same assembly

public string MyVersion()

{

return "The new version: 1.0.0.20";

}

// by using the '\*' as shown below:

// [assembly: AssemblyVersion("1.0.\*")]

[assembly: AssemblyVersion("1.0.0.20")]

**Step6:** Now let's start creating the application, which refer the AssemblyVersionExample.dll Create a instance of the class and invoke the method. Output of the assembly will be new version.

static void Main(string[] args)

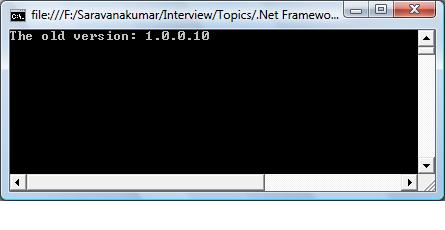
{

AssemblyVersionExample.Class1 objClass = new AssemblyVersionExample.Class1();

Console.WriteLine(objClass.MyVersion());

Console.ReadLine();

}



**Step 7:** Since we need to use the old version of the assembly from GAC, we should make use of "bindingRedirect" tag in the application. In the below sample, you can find that new attribute is set with old version value*(newVersion="1.0.0.10")* and attribute is set with new version value *(oldVersion="1.0.0.20")*. When we execute the application, resultant output will be from old version of the dll.

<configuration>

  <runtime>

    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">

      <dependentAssembly>

        <assemblyIdentity name="AssemblyVersionExample"

                          publicKeyToken="7c779e284ebe2e8c"

                          culture="neutral" />

        <bindingRedirect oldVersion="1.0.0.20"

                         newVersion="1.0.0.10"/>

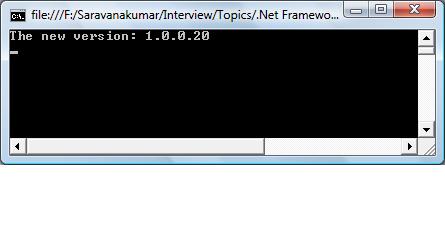
       </dependentAssembly>

    </assemblyBinding>

  </runtime>

</configuration>

**Output:**



**37. What is mean by Dll Hell?**

DLL hell means deploying the same DLL in your application multiple times. In windows application dlls are shared across multiple application. Suppose when App1 is using MyAssembly.dll and it is working fine. Suppose I am installing new application App2 which also having assembly MyAssembly.dll, while installing App2 it will override the old assembly with new MyAssembly.dll. Now only App2 will function properly where as App1 which depends on MyAssembly.dll will fail. This is called as Dll hell. It can be solved by assembly versioning.

**38. How's the DLL Hell problem solved in .NET?**

Assembly versioning allows the application to specify not only the library it needs to run (which was available under Win32), but also the version of the assembly.

**39. What's the difference between the System.Array.CopyTo() and System.Array.Clone()?**

* System.Array.CopyTo() - Performs a deep copy of the array
* System.Array.Clone()- Performs a shallow copy of the array

**40. What is difference between application running in Debug and Release mode?**

In a debug build mode the complete symbolic debug information is added to complile assembly to help while debugging applications and also the code optimization is not taken into account. While in release build the symbolic debug infrmation is not added to the compiled assembly and the code execution is optimized. Since debuging information is not added in a release build, the size of the final executable is lesser than a debug executable.

**41. What is the difference between traditional development and .NET development?**

In traditional programming languages, the source code of a program is compiled to a specific platform's assembly language and then machine language code. Later the library code required by the program is linked to it. Finally the operating system executes the program when desired by the user

In the presence of dot net framework, a program is not compiled to the native machine executable code; rather it gets compiled to an intermediate language code called Microsoft Intermediate Language (MSIL) or Common Intermediate Language (CIL). The Dot Net Common Language Runtime (CLR) then converts this intermediate code at runtime to the machine executable code. The optimization is carried out at runtime

**41. How true it is that .NET and Java programs are quite in-efficient when compared to C++?**

In .Net and Java programming, initial execution of the program will be little bit slower than the C++ programming. Because .Net and Java involves the hosting of CLR into managed applcaiotn process in .Net and starting the JVM in a new process in case of Java. Since, the CLR and JVM optimizes the code more efficiently than the static C++ compilers, the execution speed of the program may actually be faster after sometime of the program startup when most of the code is translated. Hence, in the longer run, the .Net and Java based programs should not be in-efficient when compared to C++.

**42. How Finaliz() method will work in .net?**

.Net framework provides ahte Object.Finalize() method to clean up objects unmanaged resources. In general garbage collector keeps track of objects that have Finalize methods, using an internal structure called the finalization queue. Each time your application creates an object that has a Finalize method, the garbage collector places an entry in the finalization queue that points to that object. The finalization queue contains entries for all the objects in the managed heap that need to have their finalization code called before the garbage collector can free their memory.

**Finalize** methods requires at least two garbage collections to free the resources. When the garbage collector performs a collection, it reclaims the memory for inaccessible objects without finalizers. At this time, it cannot collect the inaccessible objects that do have finalizers. Instead, it removes the entries for these objects from the finalization queue and places them in a list of objects marked as ready for finalization. Entries in this list point to the objects in the managed heap that are ready to have their finalization code called. The garbage collector calls the Finalize methods for the objects in this list and then removes the entries from the list. A future garbage collection will determine that the finalized objects are truly garbage because they are no longer pointed to by entries in the list of objects marked as ready for finalization. In this future garbage collection, the objects' memory is actually reclaimed.