# EXPLORING ASSEMBLY

1. **What is .NET assembly?**

All the .NET assemblies contain the definition of types, versioning information for the type, meta-data, and manifest. It is a compiled unit of code, that can be executed by .NET runtime.

1. **How many ways assemblies can be classified?**

.NET assemblies can be broadly classified into 2 types:

Weak Named Assemblies

Strong named assemblies

1. **How to generate key pair?**

To generate the key pair, use strong naming tool

sn.exe -k C:\KeyFile.snk

To strong name an assembly

[assembly: AssemblyKeyFile(“KeyFile.snk”)]

1. **What should a strong name assembly should have ?**

A strong named assembly should have all of the following:

* The textual assembly name.
* The assembly version number
* The assembly should have been signed with private/public key pair.

1. **What does assemblies consists of ?**

A assembly name consists of 4 parts

* Simple textual name
* Version number
* Culture information (otherwise the assembly is language neutral)
* Public Key Token

We use AssemblyVersion attributes to specify the assembly version. The default is 1.0.0.0. The version number of an assembly consists of the following 4 parts.

* Major version
* Minor version
* Build number
* Revision number

1. What is versioning managed in assembly?

The version information for an assembly typically consists of four values: a major version number, a minor version number, and optionally, a build number and a revision number. These version numbers play a crucial role in identifying and managing different versions of the assembly, enabling developers to track and control changes made to the assembly over time.

Major Version

This is the internal version of the product and is assigned by the application team. It should not change during the development cycle of a product release.

Minor Version

This should only change when there is a small changes to existing features. It is assigned by the application team, and it should not be changed during the development cycle of a product release.

Build Number

Typically incremented automatically as part of every build performed on the Build Server. Using the build number in conjunction with the source number allows you to identify what was built and how. This allows each build to be tracked and tested.

Revision

This is the number taken from source control to identify what was actually built. This is set to zero for the initial release of any major or minor version of the solution.

1. **What is the steps to add an assembly to GAC?**

* Make a class library project to create a dll or assembly
* Make a .NET console project and use or add reference the assembly into this project.
* Generate a key pair using the command: **sn.exe -k c:\MyStrongKeys.snk**
* Go to Properties of .NET console project and add key pair for strong named assembly: **[assembly:AssemblyKeyFile("C:\\MyStrongKeys.snk")]**
* Go to VS Command Prompt and add the assembly to GAC : **gacutil.exe -I MathFunctionDLL.dll**
* You can find the assembly added to GAC : C:\Windows\Microsoft.NET\assembly\GAC\_MSIL\ {Your assembly name}

1. **How to uninstall assembly ?**

* **gacutil -u MathFunctionDLL** (no mention the extension)

1. **Features of Assembly**

* They are self-describing. They consist of metadata which tells what are the methods, properties etc present in the assembly.
* Assembly can be loaded side-by side thus achieving side by side execution.
* Installation of an assembly is easier.
* Assemblies solve the DLL HELL problem.

1. **Types of Assembly**

They are implemented as .exe or .dll.

1. **Difference between DLL and EXE ?**

|  |  |
| --- | --- |
| DLL | EXE |
| Dynamic Lin k Library with .dll extension. | Executable with .exe extension. |
| Can have many entry point. | Has only one entry point |
| Does not contain UI | Contains UI |
| It is an in-process file, that means it runs in someone else memory. | It is out process file. It can run independently. Its standalone. |

1. **What the physical location of assembly**

2 Paths:

1. C:\Windows\Assembly – For .NET 2.0 – 3.5 assemblies
2. C:\Windows\Microsoft.NET\assembly - For .NET 4.0 assemblies.
3. **Different types of Assembly**

Private Assembly

An assembly used by a single application is called private assembly. Its present in the bin folder.

Shared Assembly

An assembly used by more than one application. Its present in GAC.

Satellite Assembly

These are resource files which are compiled to assemblies.

1. **What is DLL Hell ?**

DLL HELL is a problem which arises when a new version of application with a new set of DLL overrides the older version.The application which are using older version of DLL crashes because those have been replaced by new DLL which are not compatible with the old applicaitons.

1. **How is DLL Hell solved in .Net?**

It is solved with the help of Assemblies.Assemblies allow different versions of DLL to co-exist among themselves.This feature is called Versioning.

1. **What are the contents of an assembly?**

Manifest

IL Code

Resource Files

Metadata

1. **What is manifest?**

Manifest of an assembly contains the following-

Identity

List of file present in assembly

A list of referenced assemblies

A set of permissions request.

1. **What is metadata?**

Contains all the types and member information present in the code.

1. **What is ILDASM?**

Its a tool used to view contents of an assembly.

1. **What is GAC ?**

GAC is a central repository in a system in which assemblies are registered to share between application.

GACUtil.exe is used to view and change the content of GAC in system

GAC can contain multiple versions on .net assemblies

The gautil.exe/I<assembly\_name> is used to install assembly in GAC

1. **How does the GAC work ? What is the prupose in .NET Framework? Pros and Cons?**

Its purpose is to reduce duplication of code, promote reusability, and simplify versioning.

When an application requests an assembly, the CLR checks the GAC before probing other locations. If found, it loads the appropriate version based on the application’s configuration or the latest available version.

Pros of using the GAC include:

1. Reduced disk space usage by sharing assemblies.

2. Simplified version management with side-by-side execution.

3. Enhanced security through strong-named assemblies.

Cons of using the GAC are:

1. Increased complexity in deployment due to additional steps for registering assemblies.

2. Potential conflicts when multiple versions coexist.

3. Dependency on strong-named assemblies may hinder development flexibility.

1. **How do install and uninstall assembly in GAC ?**

Two ways-

Simply darg and drop

Use GacUtil.exe (GAC Utility Tool)

install- gacutil -i <assembly name>

uninstall-gacutil -u <assembly name>

1. **How do you add/remove assembly from GAC?**

you can add assembly by using below syntax:

gacutil /i [assemblyName | assemblyPath]

you can remove assembly by using below syntax:

gacutil /u [assemblyName | assemblyPath]

1. **What is Satellite Assembly?**

Satellite assemblies are assemblies that is used to deploy culture and language for an application. A separate product id is assigned to each language and a satellite assembly is installed in language specific sub directory.