**Lecture 13 - Assembly**

Assemblies form the fundamental unit of deployment, version control, reuse, activation scoping, and security permissions for a .NET-based application. Assemblies take the form of an executable (.exe) file or dynamic link library (.dll) file, and are the building blocks of the .NET Framework. They provide the common language runtime with the information it needs to be aware of type implementations. You can think of an assembly as a collection of types and resources that form a logical unit of functionality and are built to work together.

Assemblies can contain one or more modules. For example, larger projects may be planned in such a way that several individual developers work on separate modules, all coming together to create a single assembly.

Assemblies have the following properties:

* Assemblies are implemented as .exe or .dll files.
* You can share an assembly between applications by putting it in the global assembly cache. Assemblies must be strong-named before they can be included in the global assembly cache. For more information, see[Strong-Named Assemblies](http://msdn.microsoft.com/en-us/library/wd40t7ad.aspx).
* Assemblies are only loaded into memory if they are required. If they are not used, they are not loaded. This means that assemblies can be an efficient way to manage resources in larger projects.
* You can programmatically obtain information about an assembly by using reflection. For more information, see [Reflection](http://msdn.microsoft.com/en-us/library/ms173183.aspx).
* If you want to load an assembly only to inspect it, use a method such as [ReflectionOnlyLoadFrom](http://msdn.microsoft.com/en-us/library/system.reflection.assembly.reflectiononlyloadfrom.aspx).

[Assembly Manifest](javascript:void(0))

Within every assembly is an *assembly manifest*. Similar to a table of contents, the assembly manifest contains the following:

* The assembly's identity (its name and version).
* A file table describing all the other files that make up the assembly, for example, any other assemblies you created that your .exe or .dll file relies on, or even bitmap or Readme files.
* An *assembly reference list*, which is a list of all external dependencies—.dlls or other files your application needs that may have been created by someone else. Assembly references contain references to both global and private objects. Global objects reside in the global assembly cache, an area available to other applications, somewhat like the System32 directory. The [Microsoft.VisualBasic](http://msdn.microsoft.com/en-us/library/microsoft.visualbasic.aspx) namespace is an example of an assembly in the global assembly cache. Private objects must be in a directory at either the same level as or below the directory in which your application is installed.

Because assemblies contain information about content, versioning, and dependencies, the applications you create with Visual Basic and C# do not rely on registry values to function properly. Assemblies reduce .dll conflicts and make your applications more reliable and easier to deploy. In many cases, you can install a .NET-based application simply by copying its files to the target compute

**Questions**

1. What is assembly, types metadata, MSIL, manifest?
2. What does manifest contain?
3. Open assemblies and partial assemblies difference?
4. How does .NET search for assembly?