Introduction to .NET

Part II

Assembly Attributes

- When you build a project, VS.NET always generates a .cs file called AssemblyInfo.cs
- You can set custom attributes for your assembly in this file such as Company, Copyright, Trademark, Title, and Version number

Assembly Attributes

- using System.Reflection
- // set name, copyright, trademark, version
- [assembly:AssemblyCompany("My Company")]
- [assembly:AssemblyCopyright("Copyright © 2009 RZ")]
- [assembly:AssemblyTrademark("Blah is a registered trademark")]
- [assembly:AssemblyVersion("3.1.0.0")]

- The version number has 4 parts:
- Major Number 3 – major and minor = public perception
- Minor Number 1 – i.e., this is version 3.1 of the software
- Build Number 0
- Revision Number 0

Private Deployment

- Easiest way to deploy you're your application is just to copy all files to the same directory.
- That's it – no registry settings and no active directory trouble
- Assemblies deployed to the same directory as the application are called **privately deployed assemblies** because the files aren't shared with any other application (unless also deployed to same directory)

Private Deployment

- This is a big plus because you just copy everything into one folder and the CLR will load and execute each one no problem.
- In addition, the referencing assembly scopes every type.
- **This means that an application always binds to the exact type that it was built and tested with (including version number) -- the CLR cannot load a different assembly that just happens to provide a type with the same name. No More "DLL Hell".**

Simple Administrative Control

- To allow administrative control over an application, a configuration file can be placed in the same directory as the application.
- The CLR interprets the content of this file to **alter its policies for locating and finding assemblies.**

Simple Administrative Control

- Example: Assume a directory named AppDir contains myapp.exe and an app.config file.
- Underneath this folder is a directory named AuxFilesDir and it contains a dll named Other.dll which is needed by myapp.exe.
- Since Other.dll library is not in the same directory as the main app, the CLR won't be able to locate and load it.
- Would get a System.IO.FileNotFoundException

Simple Administrative Control

- To fix this problem, you would create an XML configuration file and put it in the app's main directory.
- The name of this configurtation file is always app.config and it is placed in the same folder as the main app.
- When you compile, VS Net will generate a file named myapp.exe.config and it will be put in the bin/debug folder where myapp.exe resides.

Simple Administrative Control

- The app.config file in this example would look like:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<runtime>
  <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
    <probing privatePath="AuxFilesDir"/>
  </assemblyBinding>
</runtime>
</configuration>
```

Simple Administrative Control

- When CLR attempts to locate a file, it looks in main directory first, and if not found it will now look in the AuxFilesDir.
- You can specify multiple, semicolon delimited paths for the probing element's privatePath attribute.
- Each path must be relative to main app directory. It cannot be off in another folder unrelated to the main directory.
- System.Configuration namespace has classes that allow you to manipulate the configuration file at runtime.