C#开发奇技淫巧：把dll放在不同的目录让你的程序更整洁

时间 2014-11-22 12:53:00  [码到功成](http://www.tuicool.com/sites/y2MZbq2)

原文  [http://www.cnblogs.com/marvin/p/PutDllToSpecificFolder.html](http://www.cnblogs.com/marvin/p/PutDllToSpecificFolder.html?utm_source=tuicool&utm_medium=referral)

主题 [动态链接库](http://www.tuicool.com/topics/11080040)[C#](http://www.tuicool.com/topics/11050001)

**系列文章**

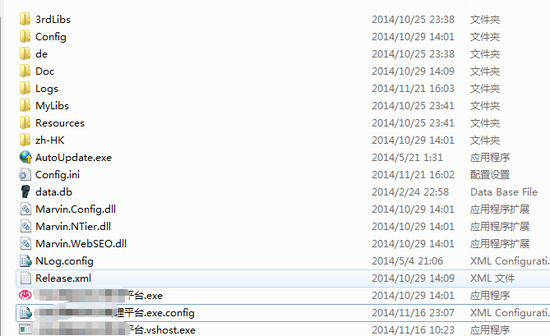
* [***C#开发奇技淫巧一：调试windows系统服务***](http://www.cnblogs.com/marvin/archive/2010/08/30/1812335.html)
* [***C#开发奇技淫巧二：根据dll文件加载C++或者Delphi插件***](http://www.cnblogs.com/marvin/archive/2011/07/26/loadunmanageddll.html)
* C#开发奇技淫巧三：把dll放在不同的目录让你的程序更整洁

**程序目录的整理**

想必C#的开发者都遇到过这个问题，引用的dll都放在根目录下，随着项目的日益增大，根目录下充满了各种各样的dll，非常的不美观。

如果能够把dll按照想要的目录来存放，那么系统就美观多了，以下是我常用的程序各文件的分布：

* 【3rdLibs】
  + NLog.dll
  + Newtonsoft.Json.dll
  + ……
* 【MyLibs】
* 【Resources】
* 【Images】
* Excecutable.exe
* Excecuteble.exe.config



网上有很多的文章述说这个，比如使用Assembly.Load，但是没有说明在程序中怎么使用，也没有给出具体的代码。这里我结合自己多年的实践经验，再把整个流程和方法详细叙述一遍，以便各位看官有个具体的体会。

**系统搜索dll的目录以及顺序**

CLR解析一个程序集会在一个根目录内进行搜索，整个探索过程又称Probing，这个根目录很显然就是当前包含当前程序集的目录。

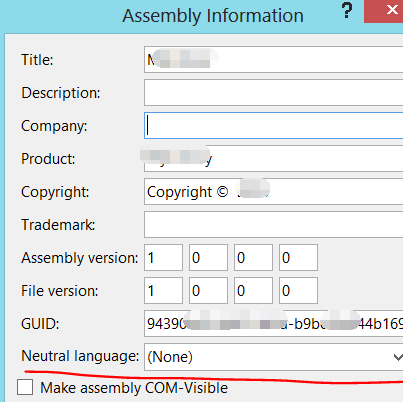
AppDomainSetup这个类存储着探索目录的信息，其成员包括： **ApplicationBase** 、 **PrivateBinPath** 。

程序搜索dll的顺序如下(区分强名称签名的和没有强名称签名的程序集)：

**没有做强名称签名的程序集：**

* 程序的根目录
* 根目录下面，与被引用程序集同名的子目录
* **根目录下面被明确定义为私有目录的子目录**
* 在目录中查找的时候，如果dll查找不到，则会尝试查找同名的exe
* 如果程序集带有区域性，而不是语言中立的，则还会尝试查找以语言区域命名的子目录

**具有强名称签名的程序集：**

* 全局程序集缓存
* 如果有定义codebase，则以codebase定义为准，如果 **codebase指定的路径找不到，则直接报告错误**
* 程序的根目录
* 根目录下面，与被引用程序集同名的子目录
* **根目录下面被明确定义为私有目录的子目录**
* 在目录中查找的时候，如果dll查找不到，则会尝试查找同名的exe
* 如果程序集带有区域性，而不是语言中立的，则还会尝试查找以语言区域命名的子目录   
  

**如何让程序识别不同目录下的dll?**

我们看到，上面的顺序无论是否有强名称签名看，都提到了一个名词 **“私有目录”**

方法一：配置App.config文件的 **privatePath** ——【推荐】

这是最简单的方法，当然也有一定的局限性，就是没法对dll做控制，另外，无法解决第三方 **DllImprt** 中引入的程序集不在根目录下的问题，不过无论怎么说，这个都基本解决了问题。

配置如下,多个目录用;分隔

**<runtime>**

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

**<probing privatePath="3rdLib;MyLibs;SubFolder\Sub.dll"/>**

**</assemblyBinding>**

**</runtime>**

**如下已验证：**

<!-- 规范dll，先检查应用程序目录有无dll，然后再检查privatePath中指定的目录 -->

<runtime>

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

<probing privatePath="DevExpress;DLLImport;bin\DLLImport"/>

</assemblyBinding>

</runtime>

放在<configuration>放在这里</configurat ion>中。

方法二：订阅程序集解析事件 **AssemblyResolve** 在代码中解析

应用程序集域中支持在程序集解析时的处理： **AppDomain.CurrentDomain.AssemblyResolve += CurrentDomain\_AssemblyResolve;** 。通过这个事件，我们可以在程序集解析时，根据不同的程序集做不用的处理，比如加载x86的程序集还是64位的程序集，当然也就可以指定程序集目录了

这也正是 **Assembly.Load** 和 **Assembly.LoadFrom** 等方法的用武之地。

**Assembly CurrentDomain\_AssemblyResolve(object sender, ResolveEventArgs args)**

**{**

**AssemblyName assemblyName = new AssemblyName(args.Name);**

**return Assembly.LoadFrom(Path.Combine(baseDirectory, "3rdLibs"));**

**}**

**方法三：在加载使用到dll的代码之前设置重置当前环境的目录**

这个方法就是通过 **Environment.CurrentDirectory=customPath** ，这样，在调用dll方法时，因为目录已经切换到了

这是一个 **取巧** 的方法，不是很实用，要来回切换程序集目录，但是在某些情况下非常好用

如何处理 **[dllImport]** 中的程序集的加载

自己写 **dllImport**

如果是自己写，那么久好控制了，可以直接指定相对的目录 **DllImport(3rdLibs\NLog.dll)** 。不过这种方法不一定可靠，在某些系统硬是加载不了，如果使用了dllImport还是，推荐下面的另外一种方法。

引用的C#的插件又使用了 **dllImport**

**这是很多文章都没有提及的：**

因为无法更改路径，那么只能够使用上述特殊的方法，更改当前程序的路径

当然，还有更省事一点的做法，就是在系统环境中，增加一条记录，指向要加载的dll的所在目录。因为C++的代码中，Windows目录和Windows\System32目录以及环境变量设定的目录都是搜索路径之一。

这里提供怎么从C#中修改系统环境变量的代码：

**static void AddEnvironmentPaths(IEnumerable<string> paths)**

**{**

**var path = new[] { Environment.GetEnvironmentVariable("PATH") ?? string.Empty };**

**string newPath = string.Join(Path.PathSeparator.ToString(), path.Concat(paths));**

**Environment.SetEnvironmentVariable("PATH", newPath);**

**}**