如何加载程序集：

使用System.Reflection.Assembly类的静态Load方法

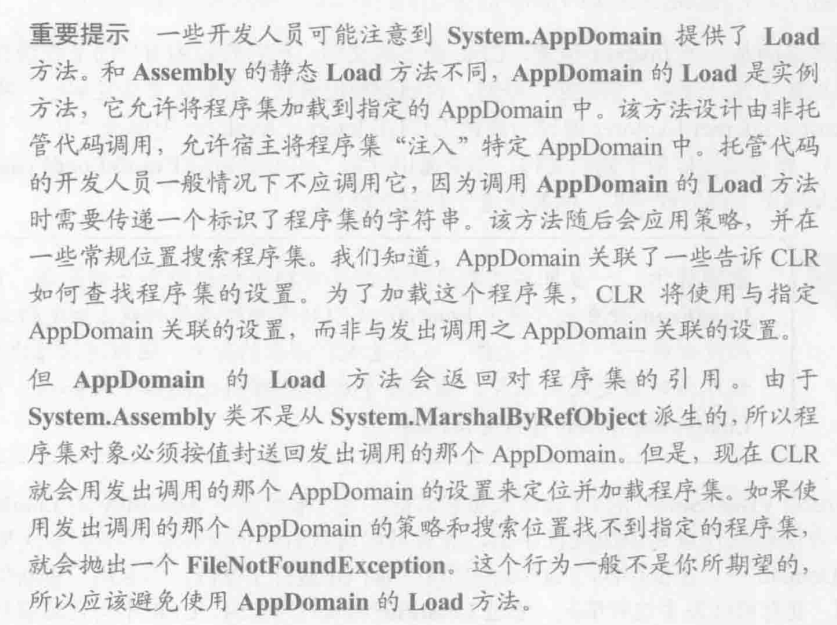
Load方法的全部重载：

社交网站的手机截图

描述已自动生成

Ps：String的 参数是程序集的标志

当使用Load加载一个强命名程序集时，CLR向程序集应用一个版本重定向策略，并在GAC中查找程序集，如果没找到就去应用程序的基目录、私有路径子目录和codebase位置查找。如果Load找到指定的程序集则返回代表已加载的那个程序集的一个Assembly对象的引用；如果Load没找到指定的程序集则抛出一个System.IO.FileNotFoundException异常。如果Load加载一个弱命名程序集，则不会应用版本重定向策略，也不会去GAC中查找程序集。

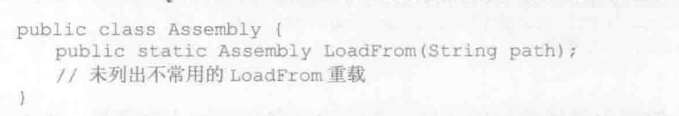


Load方法是加载程序集的首选方法，但它要求掌握构成程序集标识的各个部分。

Ps：程序集标识的构成：构成程序集标识的各个部分包括程序集的名称、版本、区域性和强名称。强名称程序集有一个完全限定的名称，由程序集的名称、区域性、公钥、版本号以及（可选）处理器体系结构组成。这些特性构成程序集的完整名称，在代码中引用程序集时必需使用。（见第三章）

当要从不在应用程序目录或 GAC 中的位置加载程序集时、当需要从远程位置（如网络共享或 URL）加载程序集时。此时无法直接使用Load方法，而应该使用LoadFrom方法。

LoadFrom方法：



作用是加载指定路径名的程序集

调用LoadFrom时

报纸上的文字

描述已自动生成

相当于使用了GetAssemblyName返回一个表示这个文件的AssemblyName对象，然后再调用Load方法使用传入AssemblyName的那个重载方法

但使用LoadFrom方法存在隐患：

一些文字和图片的手机截图

描述已自动生成

Ps：向LoadFrom方法传入一个URL作为实参时，CLR会下载文件然后把它安装到用户的下载缓存中，再从中加载文件。如果之前下载过这个文件，并且此时为脱机工作，就会使用之前下载的文件。

LoadFile方法：

和LoadFrom的加载方式相似，都是通过指定路径加载程序集。

LoadFile和LoadFrom方法的异同和各自的优点：

LoadFile：此方法从一个指定文件中加载程序集。当使用 LoadFile 方法从指定文件中加载程序集时，它不会自动加载目标程序集所引用和依赖的其他程序集。这意味着，如果目标程序集依赖于其他程序集，那么需要开发人员自己控制并显示加载所有依赖的程序集。此方法不会将文件加载到 load-from 上下文中，并且不会像 LoadFrom 方法一样使用加载路径解析依赖项。因此，当你需要从指定文件中加载程序集，并且可以自己控制并显示加载所有依赖的程序集时，你可以选择使用 LoadFile 方法。

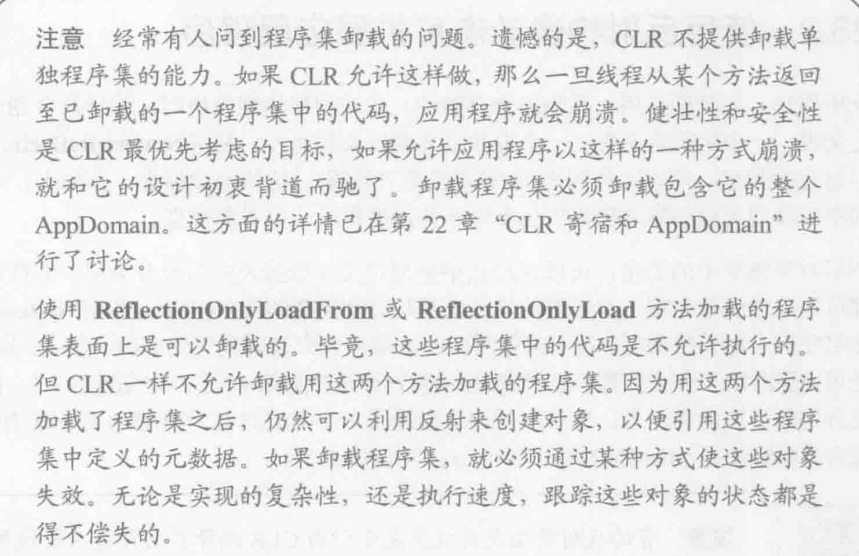
LoadFrom：此方法可以从指定文件中加载程序集，通过查找程序集的AssemblyRef元数据表，得知所有引用和需要的程序集，然后在内部调用 Load 方法进行加载。此方法首先会打开程序集文件，通过 GetAssemblyName 方法得到程序集名称，然后关闭文件，最后将得到的 AssemblyName 对象传入 Load 方法中。因此，与 Load 相比，LoadFrom 加载一个程序集时，会多次打开文件，可能会造成效率低下的现象。但是，当需要从指定文件中加载程序集，并且你希望自动处理所有依赖项时，可以选择使用 LoadFrom 方法。

ReflectionOnlyLoadFrom方法和ReflectionOnlyLoad方法（已过时）：

通过反射分析程序集的元数据并且确保程序集中的任何代码都不会执行，使用这两个方法加载的程序集都代码都被CLR禁止运行



CLR允许单独卸载使用这两个方法加载的程序集，因为这些程序集里的代码是无法执行的。对于其他的程序集，CLR不允许单独卸载，在第22章中就指出必须卸载该程序集所在的应用程序域，需要这么做的原因是为了保证安全性和健壮性。



书上提到了将DLL文件嵌入到EXE文件中的技术：

这种技术可以动态加载DLL文件，仅在需要用到这个DLL文件中的程序集时加载这个DLL文件。在将 DLL 文件嵌入到 EXE 文件之前，EXE 文件中并不包含 DLL 文件。嵌入 DLL 文件到 EXE 文件的过程是通过将 DLL 文件添加到项目中并将其设置为嵌入资源来实现的。动态加载DLL文件的好处不仅仅是减小存储空间。动态加载DLL文件可以提供更多的灵活性和控制。（应该在第二章中提到过动态加载）

动态加载DLL文件时可能出的问题：

文本

描述已自动生成

使用反射构建动态扩展应用程序：

反射做的事情是“解析元数据表”，使用System.Reflection命名空间中的类型为程序集或模块中包含的元数据提供一个对象模型。利用对象模型可以枚举类型定义元数据表中的所有类型（因此反射消耗性能效率低），还可以解析对应的元数据表查询类型的字段、属性、方法、事件、定制特性甚至里面类型引用的程序集等。

反射的性能：

反射的缺点：

1.编译时无法保证类型的安全性。因为反射极度依赖字符串，需要字符串输入正确才能获取正确的结果。

书上举的例子：

文本

描述已自动生成

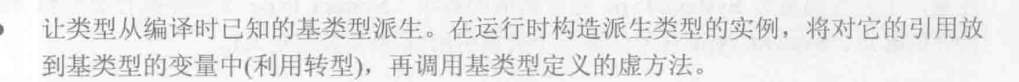
2.反射速度慢。使用反射时，类型以及其成员的名称在编译时未知，要用字符串名称标识每个类型及其成员，然后在运行时发现它们。

文本

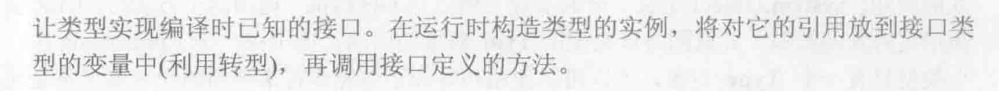
描述已自动生成

因为反射性能低，应该利用以下两种技术之一开发应用程序动态发现和构造类型实例：

1.让类型从编译时已知的基类型派生。



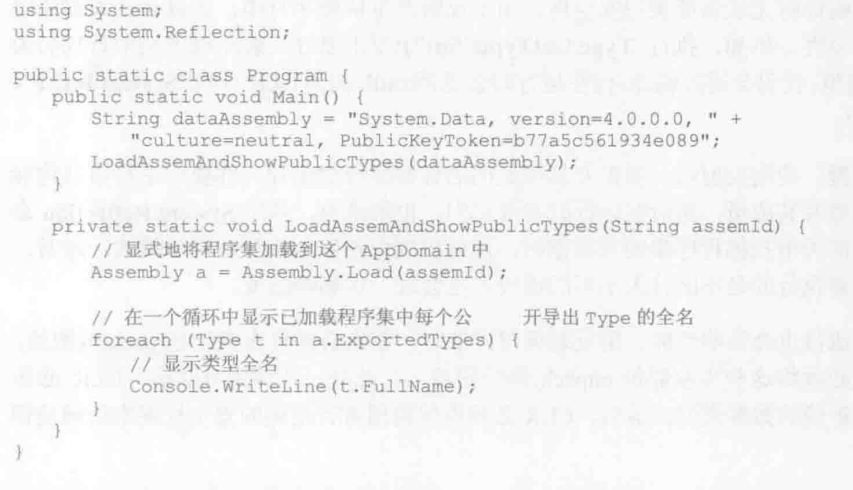
2.让类型实现编译时已知的接口。



Ps：基类技术一般用于版本控制，其他情况使用接口技术。

使用反射发现程序集中定义的类型：

Assembly.ExportedTypes属性，配合迭代器使用可以获得一个程序集中的所有public类型



类型对象的准确含义：

类型对象通常指代System.Type类型的实例

获取Type对象的方法：

1.Object类的实例GetType方法

2.System.Type类提供的静态GetType方法

3.System.Type类提供的静态ReflectionOnlyGetType方法(仅反射不能执行)

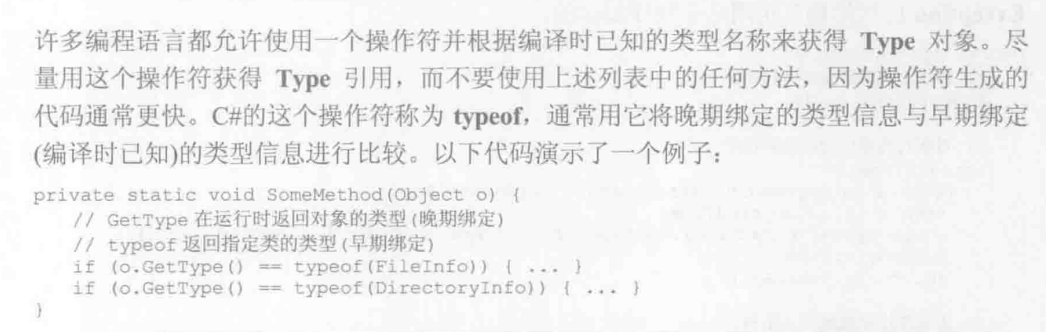
4.System.TypeInfo的实例成员DeclareNestedTypes和GetDeclareNestedTypes

5.Assembly类提供的实例GetType，DefinedTypes和ExportedTypes

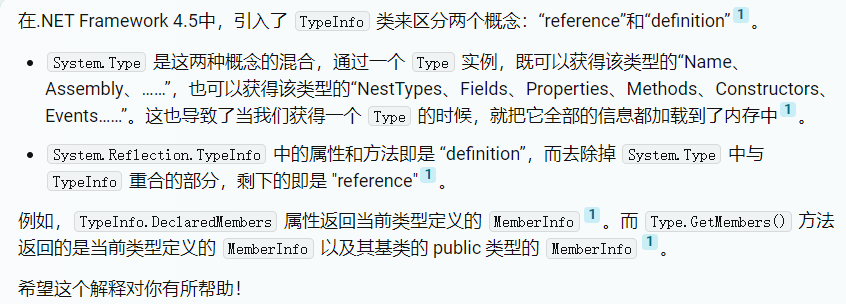
Ps：上面五个都是晚期绑定

6.typeof操作符

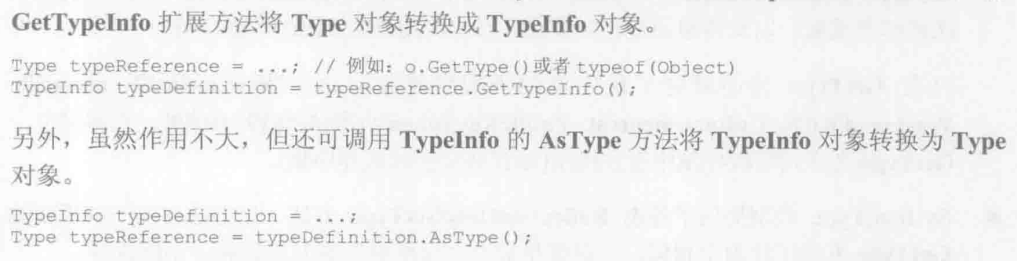
Ps：typeof操作符是早期绑定



Type和TypeInfo的差异：

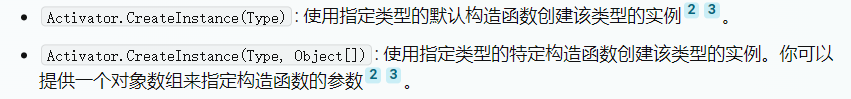


Type和TypeInfo的相互转换：



动态构造类型实例：

1. Activator.CreateInstance

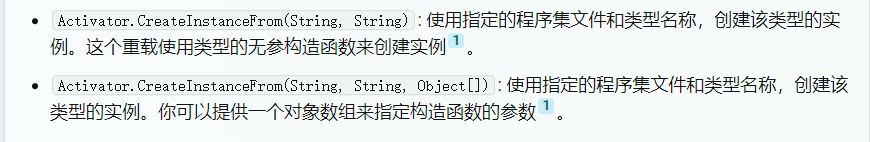


图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

2. Activator.CreateInstanceFrom

Activator.CreateInstanceFrom 是一个静态方法，它用于从指定的程序集文件中创建类型的实例。



图形用户界面, 文本, 应用程序, 电子邮件

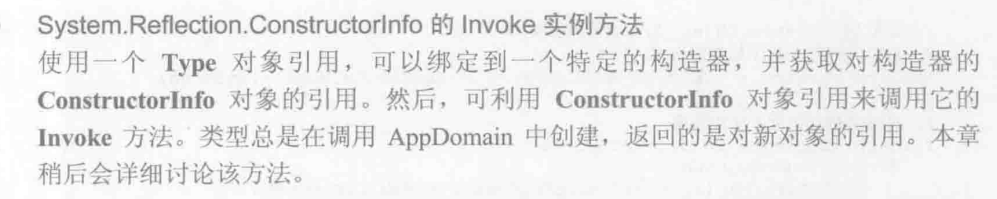
描述已自动生成

3.AppDomain类提供的几个实例方法，行为和Activator的方法相似，但是因为是实例方法可以指定在哪个应用程序域中构造对象。

文本

描述已自动生成

4.调用函数返回一个类型的对象？？？



例子：

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

这样也算吗？？？？

动态创建数组：

使用Array.Createinstance方法

动态创建委托：

使用MethodInfo.CreateDelegate方法

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

动态创建泛型类型实例，下面是书上的例子：

文本

描述已自动生成

主要是MakeGenericType实例方法的使用，MakeGenericType 是 System.Type 类的一个实例方法，它用于构造一个泛型类型的特定类型。这个方法接受一个 Type 类型的数组作为参数，这个数组中的每个元素都表示泛型类型定义中的一个类型参数。

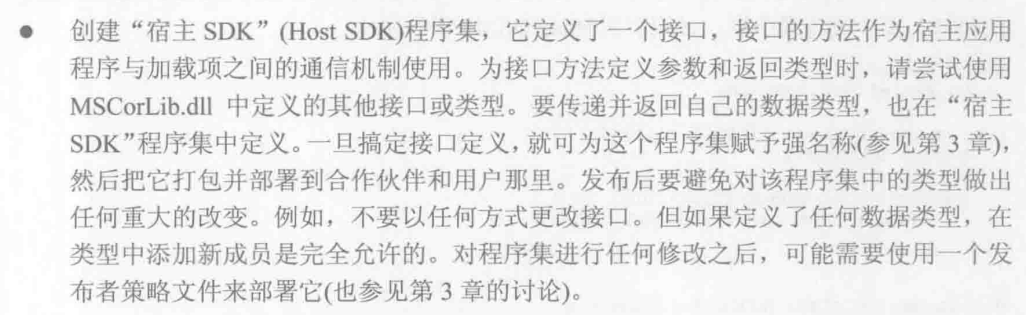
设计支持加载项的应用程序：（不是很懂）

什么是加载项？

加载项是一种可以添加到某些软件中以提供额外功能的组件。比如Edge浏览器的扩展插件。

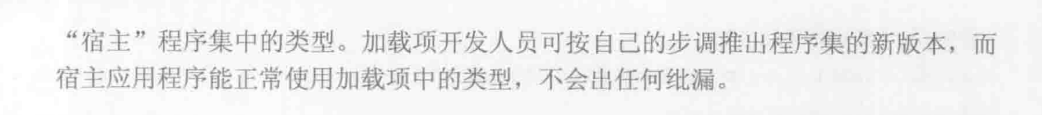
步骤：

1



2.





3.

文本

描述已自动生成

如何使用反射发现类型的成员：

System.Reflection.MemberInfo封装了所有类型成员都通用的一组属性。

下面是它的继承关系：

图示

描述已自动生成

使用的例子：

文本

描述已自动生成

DeclaredMembers 是 TypeInfo 类的一个实例属性，用于获取当前类型定义的所有成员。

表格

描述已自动生成

调用类型的成员：

表格

中度可信度描述已自动生成

（没太看懂书上的示例代码）

使用绑定句柄减少进程的内存消耗：



我理解的是把一个存在数组里的Type对象变成一个地址和这个数组的大小（偏移量）

文本

中度可信度描述已自动生成