

# .NET组件化开发技术基础

---

北京理工大学计算机学院  
金旭亮

# .NET平台的组件即“程序集”

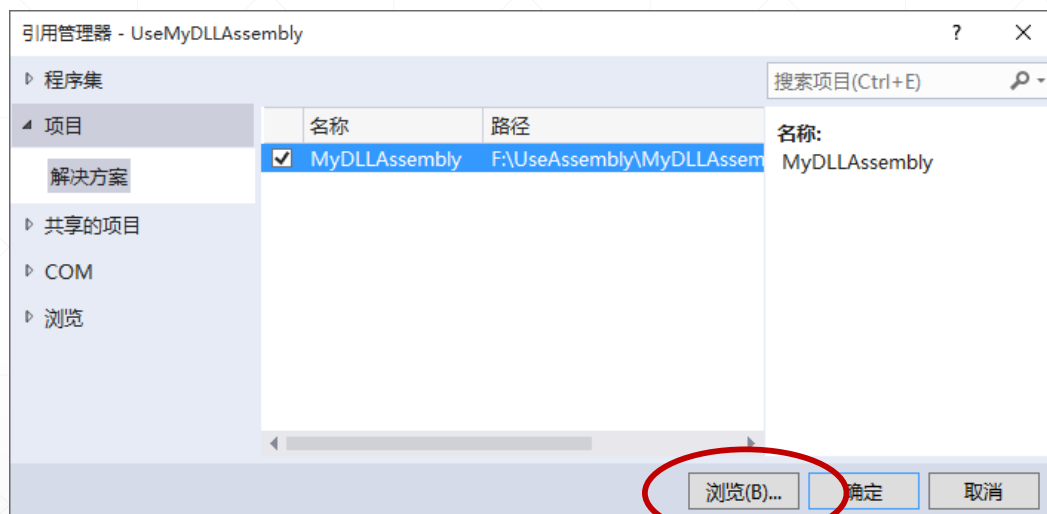
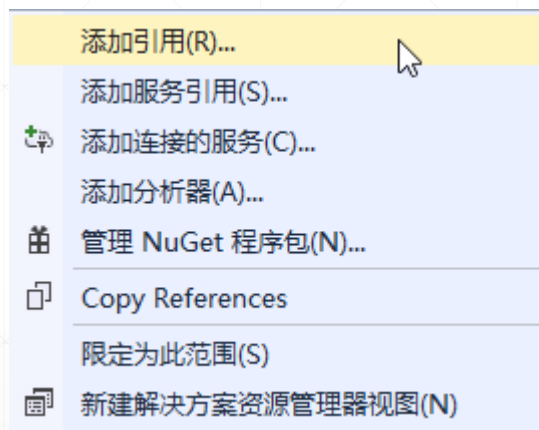
“**程序集 (Assembly)**”是.NET Framework中基本的软件模块，它可以包容数目不限的“**类型 (Type)**”，其常见的载体为一个或多个DLL文件，也可以是一个可独立执行的EXE文件。



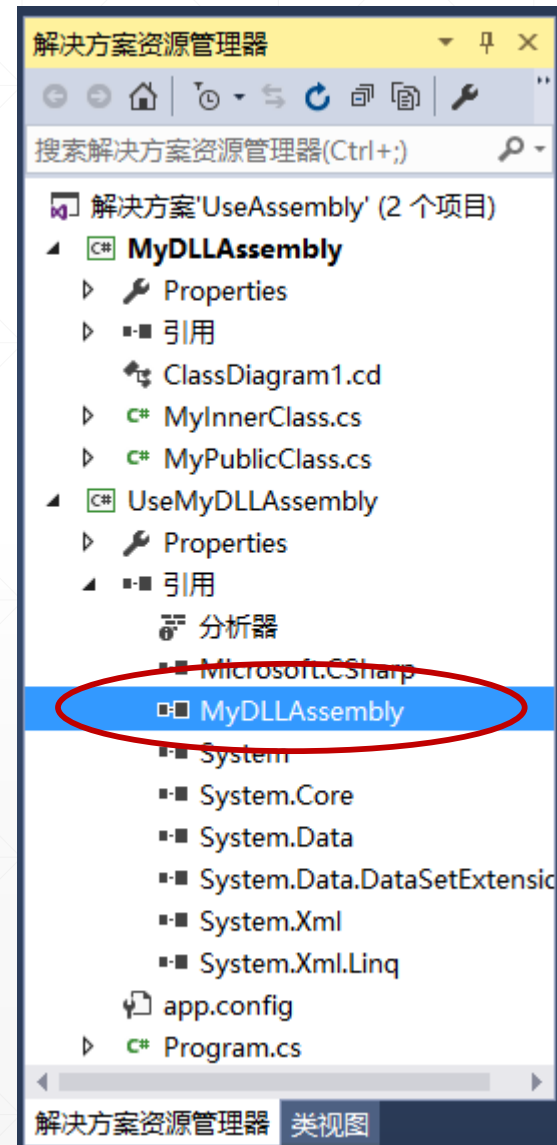
一个项目如果需要使用特定程序集中的类型，需要添加对此程序集的“引用 (Reference)”。

在项目节点上右击，从弹出菜单中选择“添加引用”

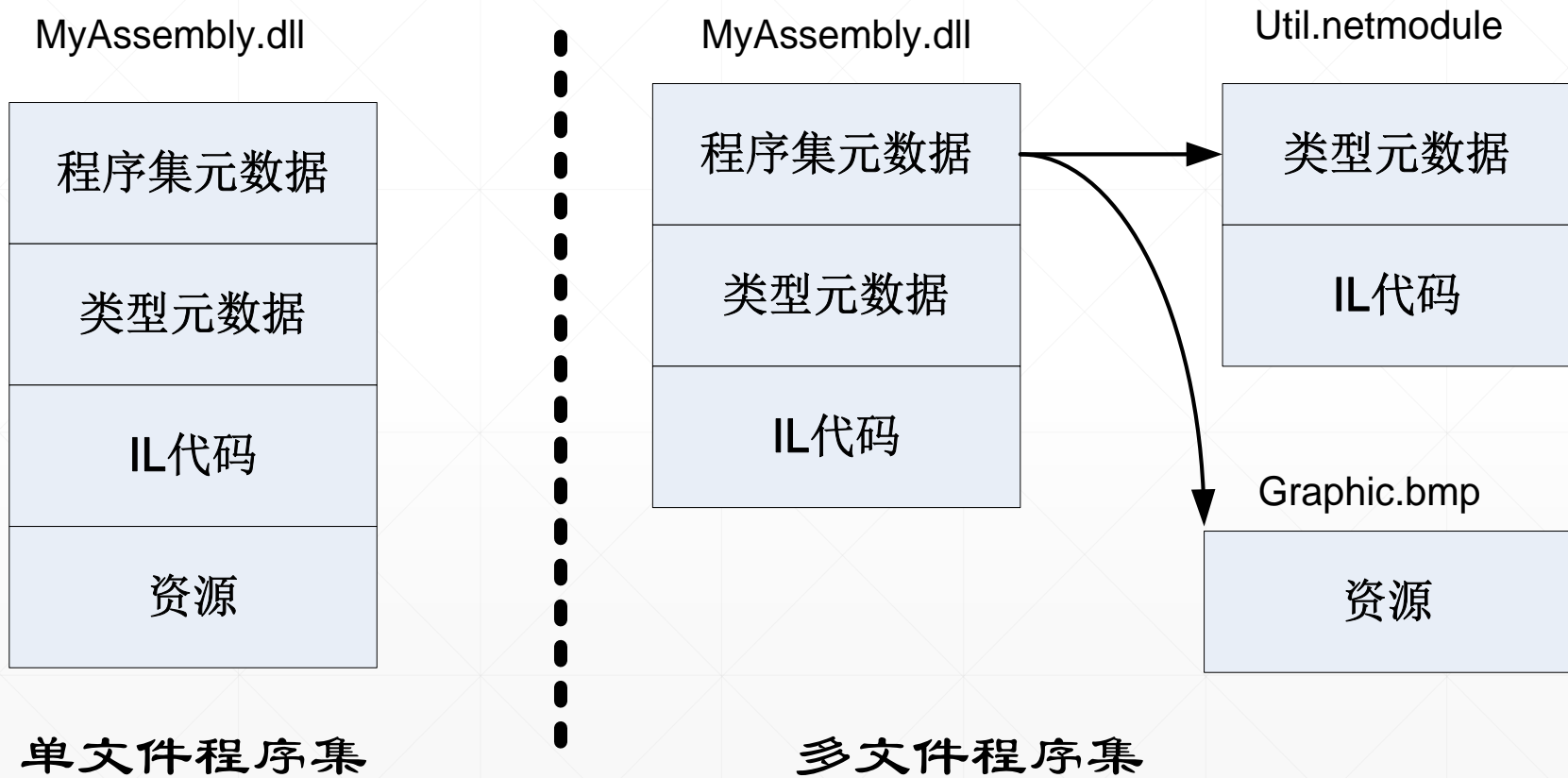
当被添加引用的项目位于同一解决方案时，推荐直接引用项目



只有程序集文件（dll或exe文件）时，通过“Browse（浏览）”按钮添加引用。



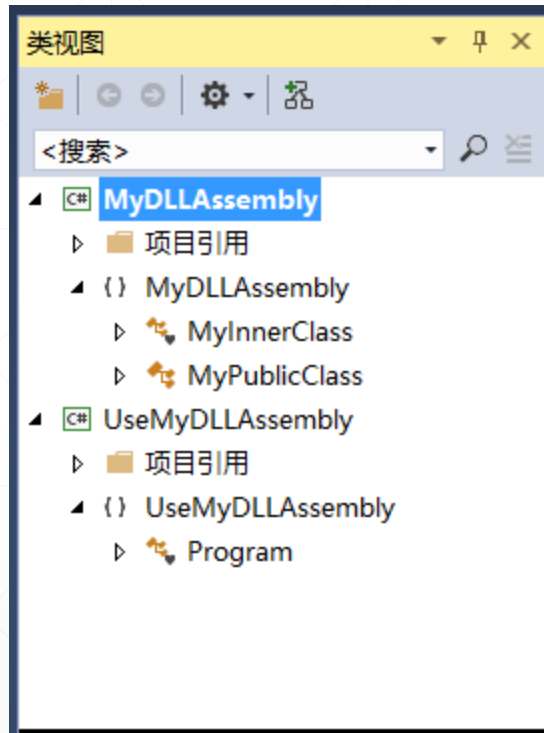
# 程序集的内部结构



.NET编译器将程序员编写的代码转换为**IL指令**

放在程序集文件中的**元数据**，表明这个程序集的版本号，由谁开发，其中定义了多少个类型，每个类型中又定义了多少成员……

# 组件设计时的“对外接口最小化”原则



一个程序集中，可以放置**任意多**个类。

仅有需要为外界所使用的类，才设置为“**公有 (public)**”的。

设置为公有的供外部使用的类，其**公有成员**“**越少越好**”，这样的类易于使用和维护。

# 组件之间的“依赖性”

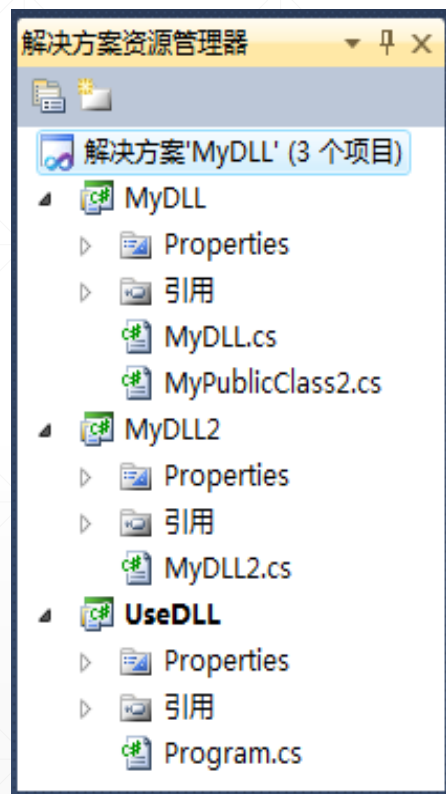
包容于某组件内部的类，如果需要调用另一个组件中的类实现的功能，则称为“**组件依赖**”。

我们使用“**耦合性**”这个术语来描述“组件依赖”。

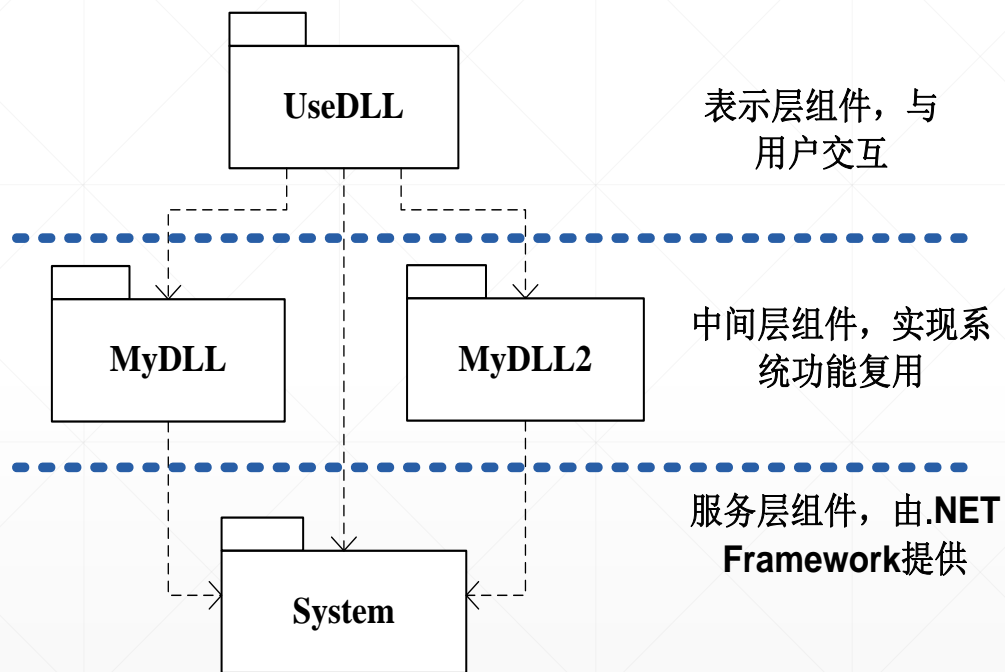
如果某个组件内部代码频繁地调用另一个组件中的代码，我们说这两个组件之间具有“**强耦合性**”。

设计各组件时，尽量减少组件之间过强的耦合性。

# 组件间的依赖关系



示例解决方案MyDLL



实际开发中，通常设计“**单向**”的组件依赖关系。

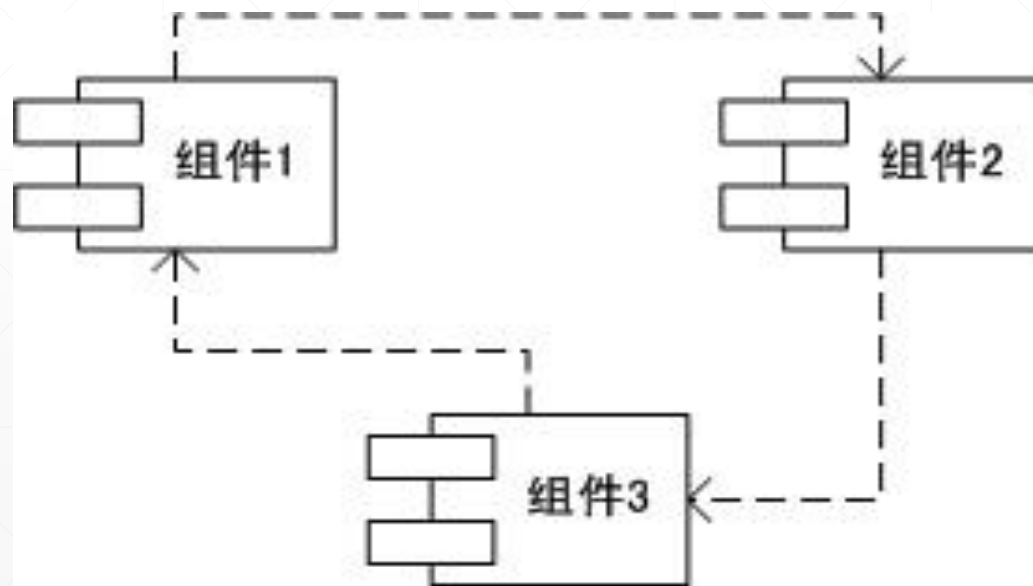
具有相同的抽象级别的多个组件组成一个“**层**”，层之间的依赖关系，也是单向的。

**组件化多层架构**，是当前软件系统最常见的架构。

# 避免出现组件循环依赖的情况

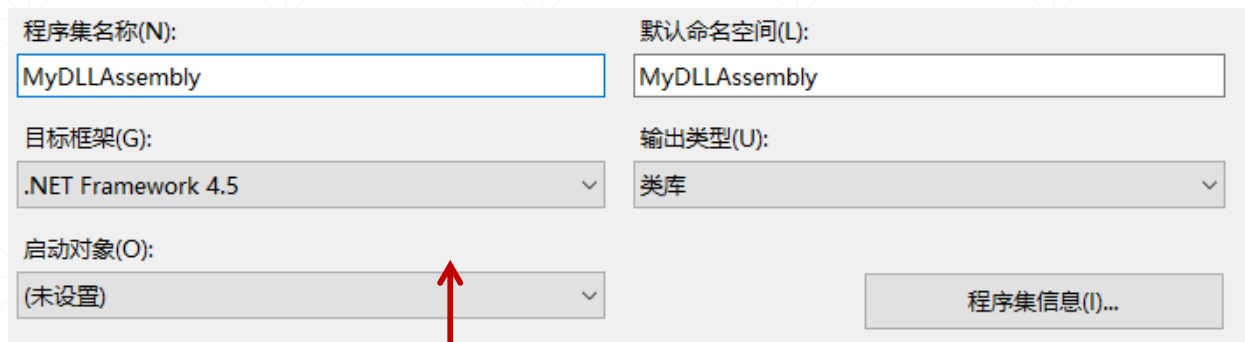


可以使用**在组件间移动类**的方式，将相互依赖的类移到同一个组件中，从而断开组件之间的循环依赖。





# 组件的版本



程序集名称(N): MyDLLAssembly

默认命名空间(L): MyDLLAssembly

目标框架(G): .NET Framework 4.5

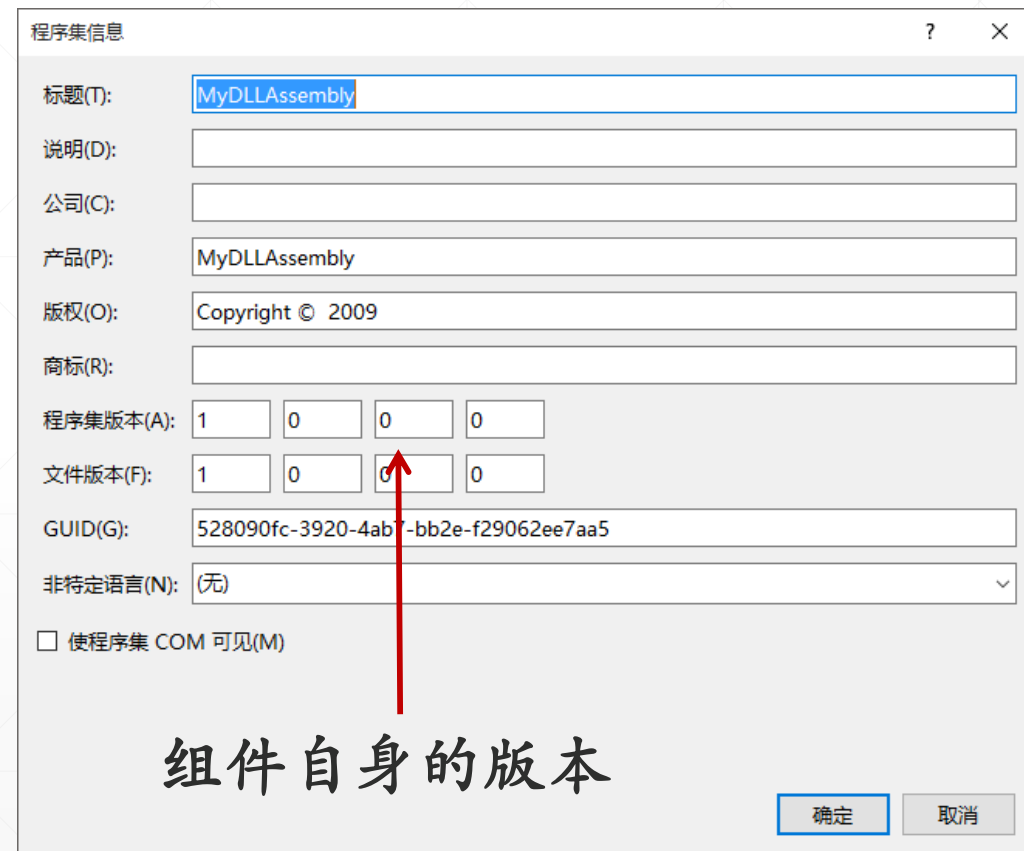
输出类型(U): 类库

启动对象(O): (未设置)

程序集信息(I)...

组件运行时要求的平台版本

组件化开发时，应该高度注意组件的版本一致性问题，否则，系统可能会出现很难排查的BUG。



程序集信息

标题(T): MyDLLAssembly

说明(D):

公司(C):

产品(P): MyDLLAssembly

版权(O): Copyright © 2009

商标(R):

程序集版本(A): 1 0 0 0

文件版本(F): 1 0 0 0

GUID(G): 528090fc-3920-4ab7-bb2e-f29062ee7aa5

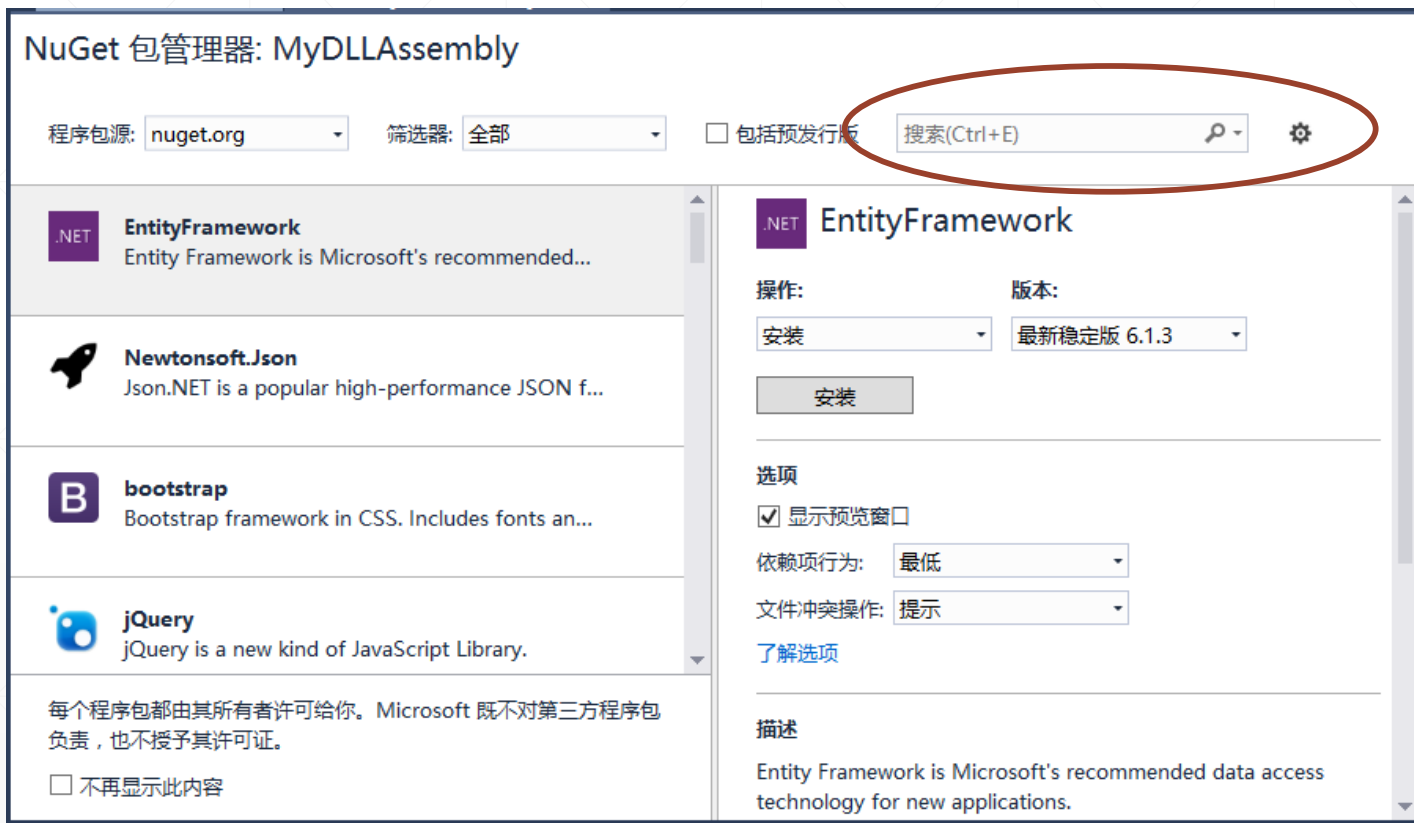
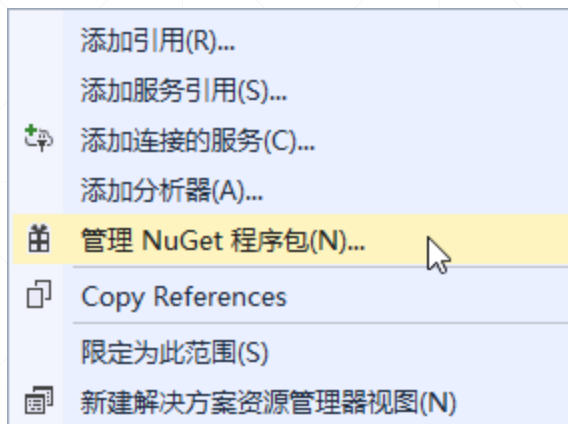
非特定语言(N): (无)

☐ 使程序集 COM 可见(M)

确定 取消

组件自身的版本

Visual Studio使用**NuGet**从网络上检索、安装和配置组件，已成主流的组件管理方式。



# 组件化开发实践

第一次大作业，你完成了一个“小小计算器”，现在，请你将计算功能独立出来，移到一个程序集中，尝试和体会一下“组件化开发”这种方式。