



# • .NET组件化开发基础

---

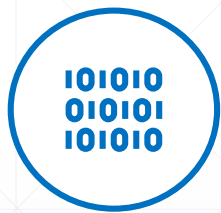
北京理工大学计算机学院  
金旭亮

# 代码重用技术发展的历史



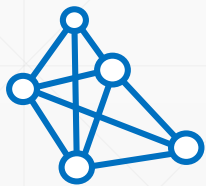
## 源代码级别的重用

- 同一种编程语言
- 所有代码均编译到单个可执行程序文件中
- 基于网络的源代码重用：npm



## 二进制级别的重用

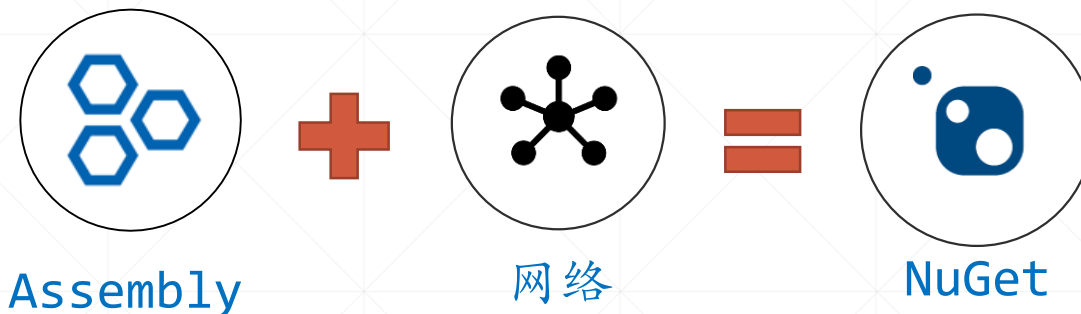
- 可以实现跨语言的重用
- 可复用代码通常以dll, jar包等形式单独存在
- 二进制组件的跨网络共享（如maven和nuget）



## 跨越机器边界的重用

- 服务聚合（RESTful Service, SOAP, 微服务）
- 远程方法调用（RPC）

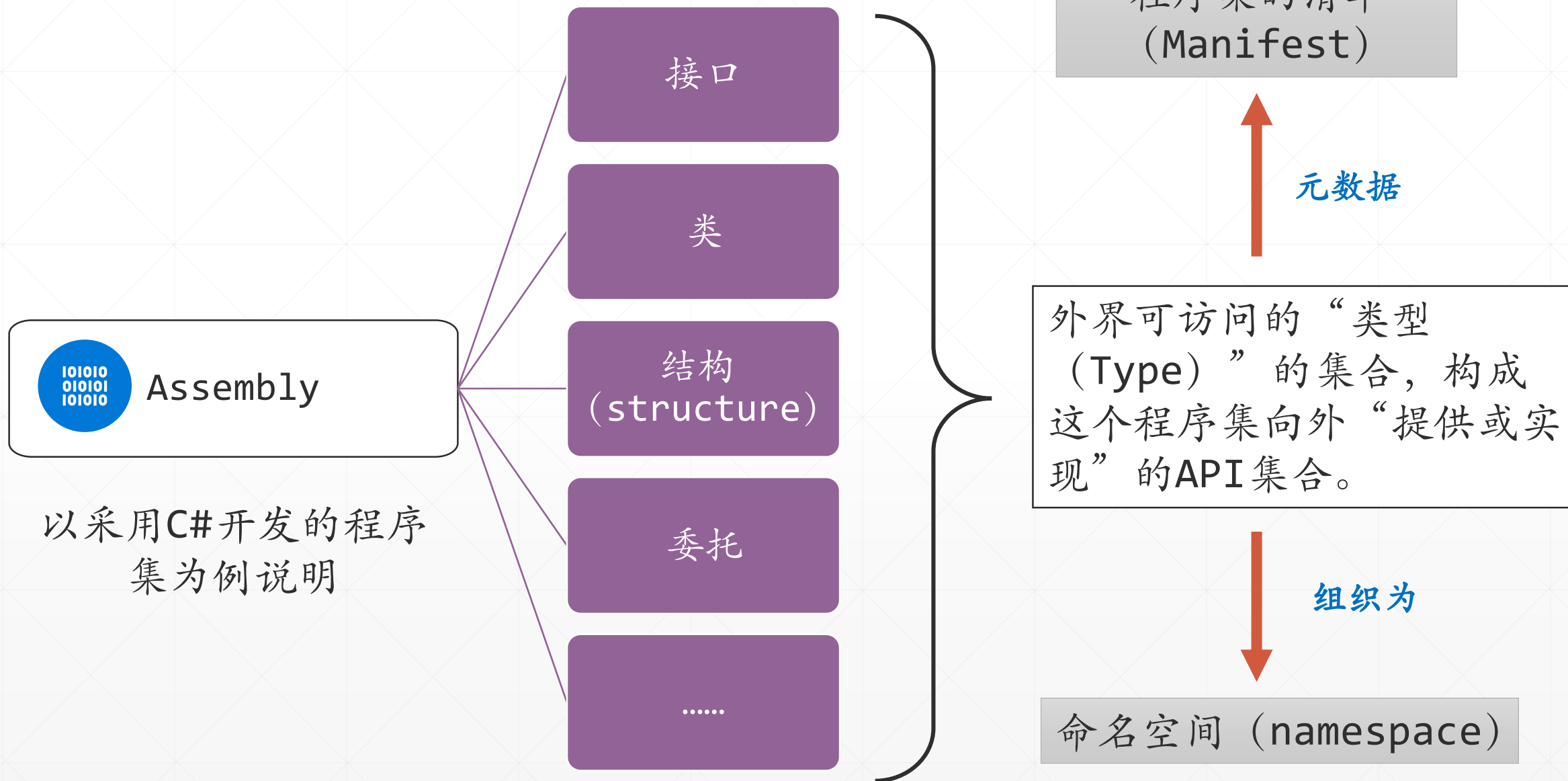
# .NET平台的组化开发技术



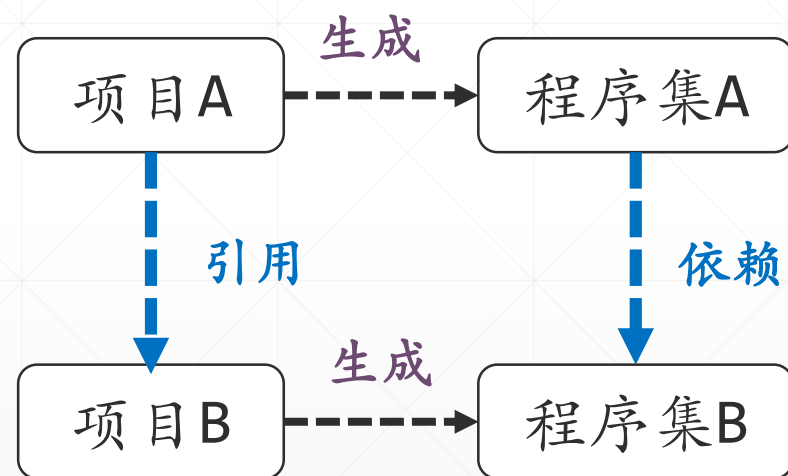
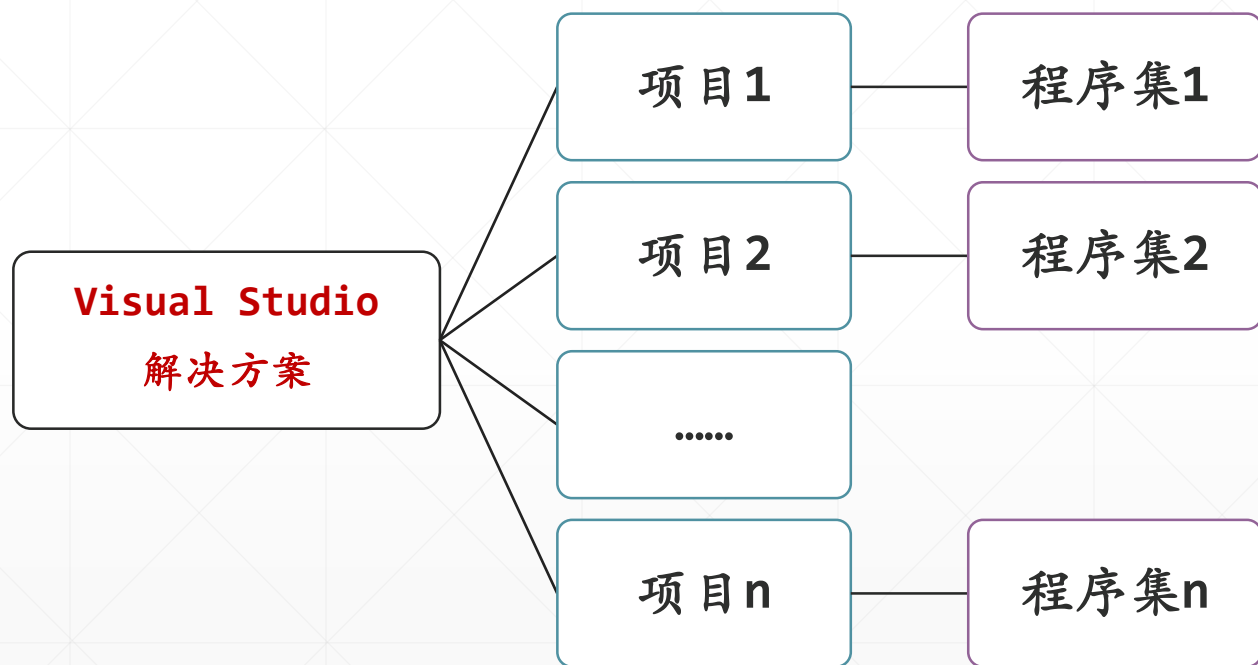
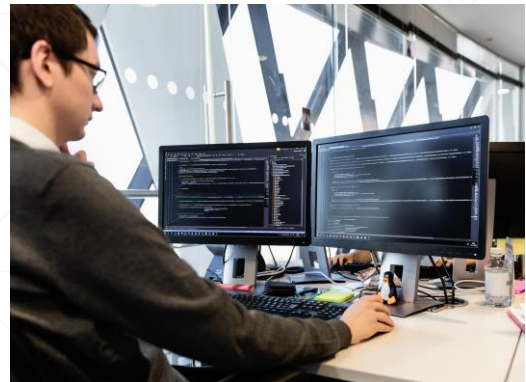
构建.NET应用的主要构件称为“**程序集 (Assembly)**”，.NET项目之间通过“**添加引用 (Add Reference)**”来重用程序集。

新一代的.NET Core应用则普遍使用NuGet包的形式基于互联网实现二进制级别的代码重用。

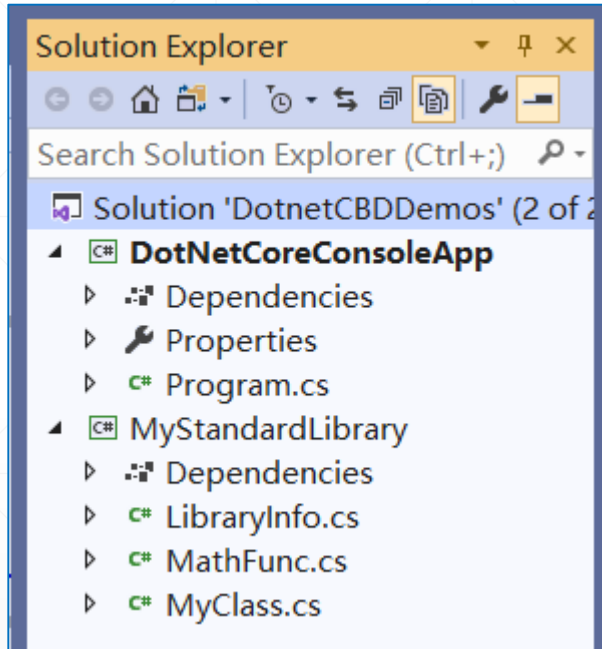
# 程序集中有什么？



# 基于组件构建.NET应用是一种什么体验?



# .NET组件化开发示例项目：DotnetCBDDemos

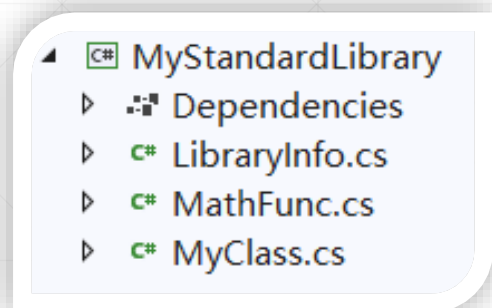


示例“解决方案 (Solution)”中包含两个“项目 (Project)”。

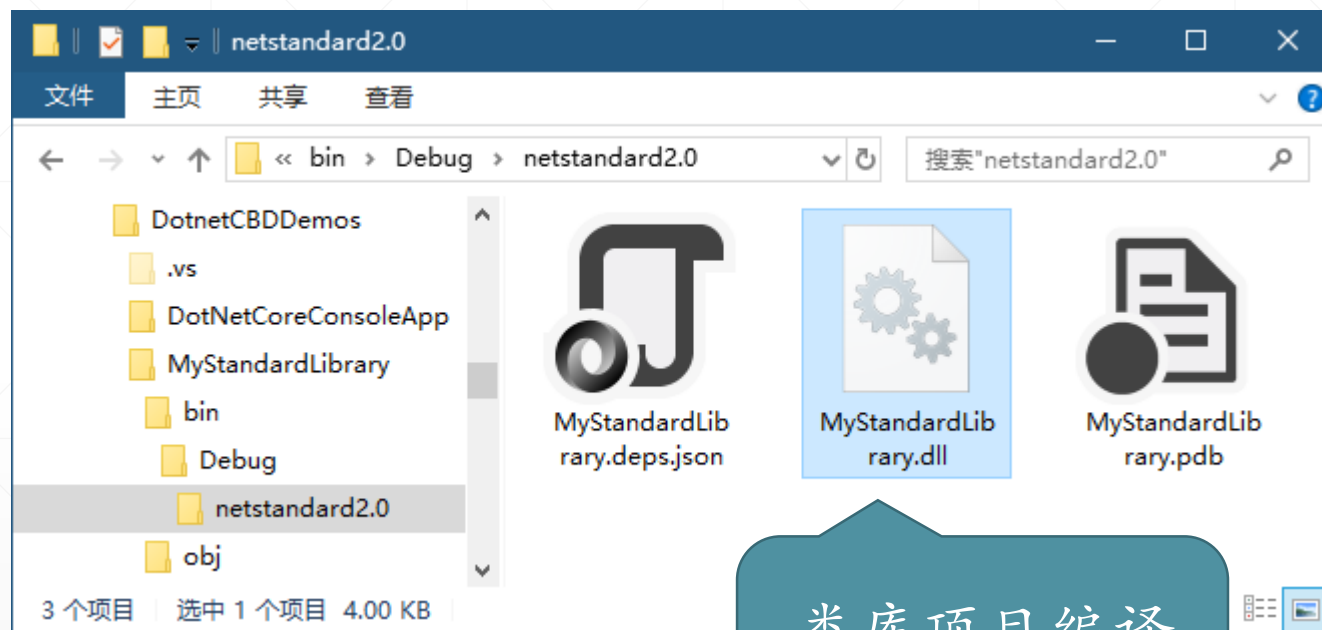
MyStandardLibrary是一个以.NET Standard 2.0为目标框架的类库项目，其中包含若干个类。

DotNetCoreConsoleApp则是一个以.NET Core 3.0为目标框架的控制台项目，它需要使用MyStandardLibrary项目中的类。

# 编译类库项目，得到程序集

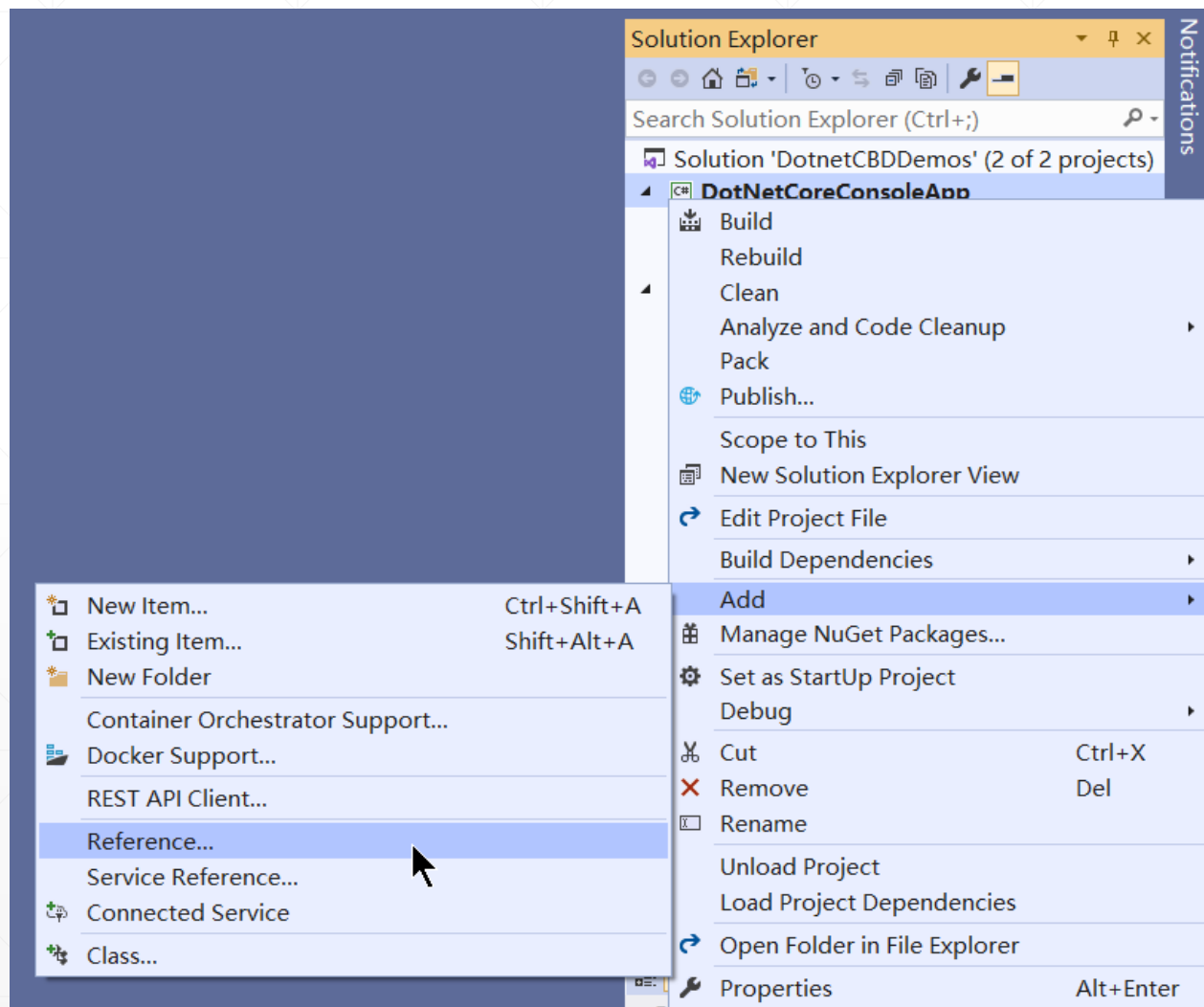


编译



类库项目编译得到的程序集

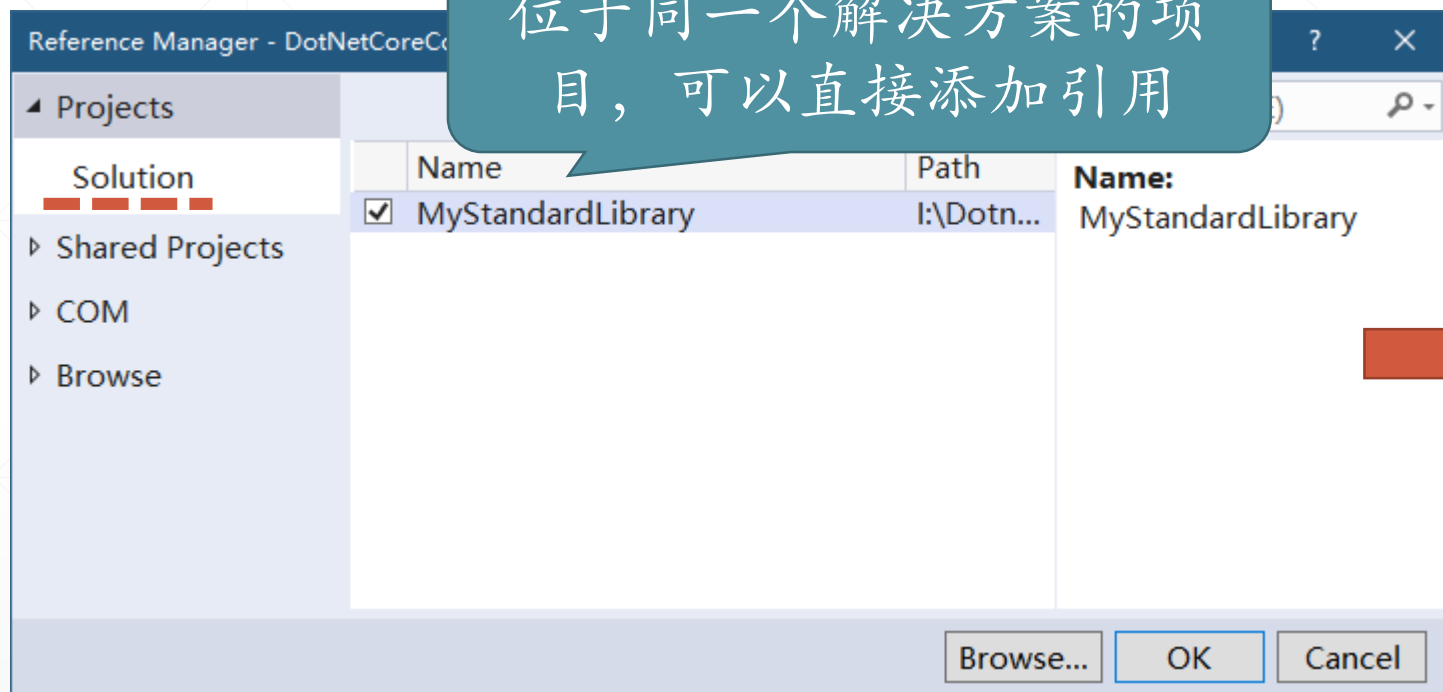
# 在项目之间建立引用关系-1



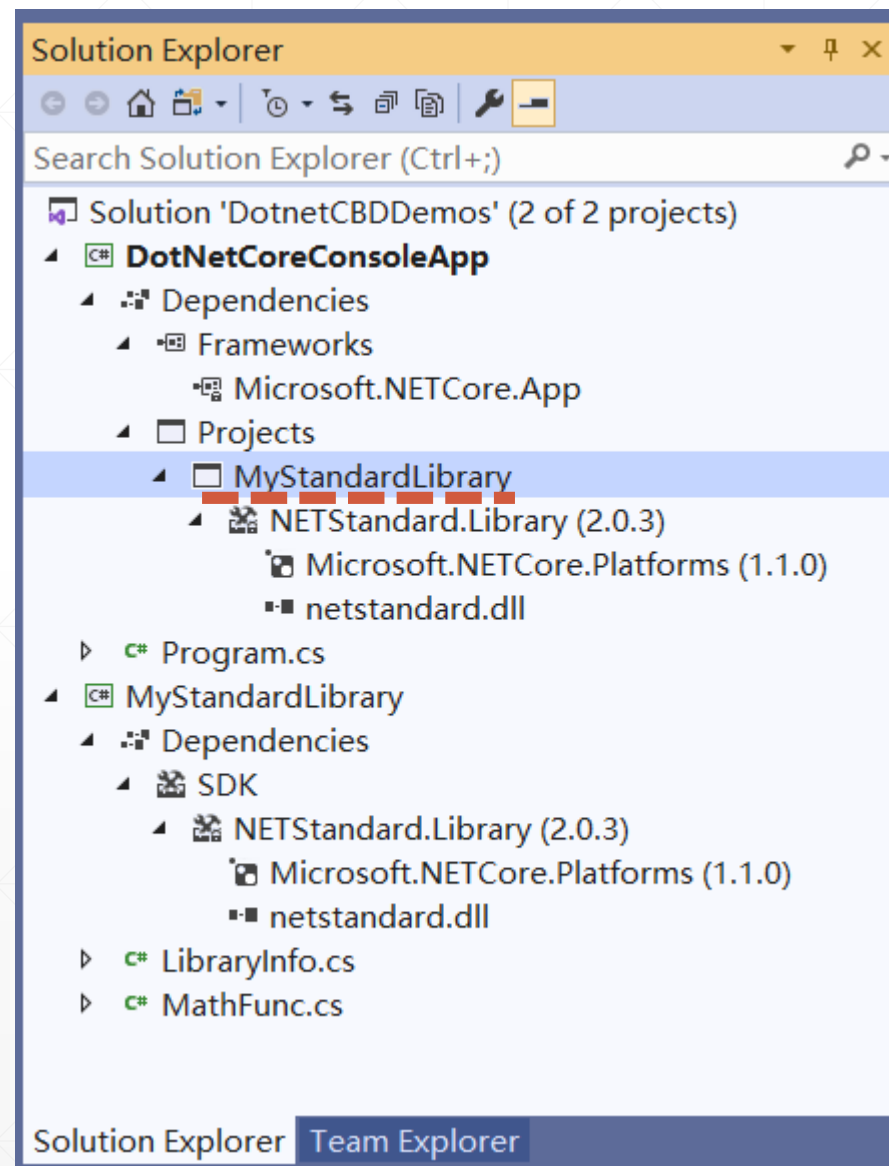
同一个解决方案中可以有多个项目，在项目节点上右击，可以给此项目添加“**引用 (Reference)**”，其实就是它希望使用放在其他项目（或外部程序集）中的代码。



## 在项目之间建立引用关系-2

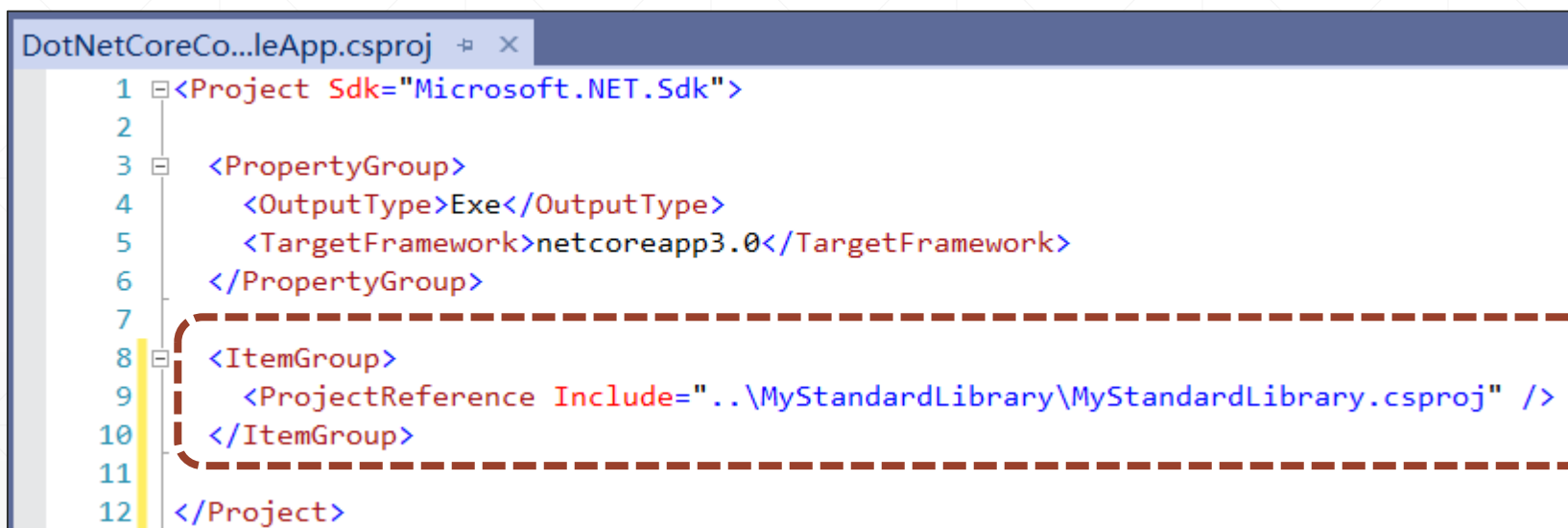


对于已经编译好的程序集，点击此按钮找到dll（或exe）文件后，添加引用。



# 同一解决方案项目之间的引用关系

在项目之间建立了引用关系之后，可以在项目文件（.csproj）中看到添加了相应的**<ProjectReference>**元素，其值指明了本项目依赖于解决方案中的哪个项目。



```
DotNetCoreCo...leApp.csproj x
1 <Project Sdk="Microsoft.NET.Sdk">
2
3   <PropertyGroup>
4     <OutputType>Exe</OutputType>
5     <TargetFramework>netcoreapp3.0</TargetFramework>
6   </PropertyGroup>
7
8   <ItemGroup>
9     <ProjectReference Include="..\MyStandardLibrary\MyStandardLibrary.csproj" />
10  </ItemGroup>
11
12 </Project>
```

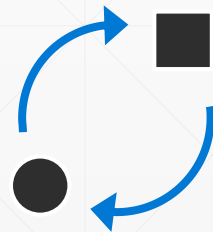
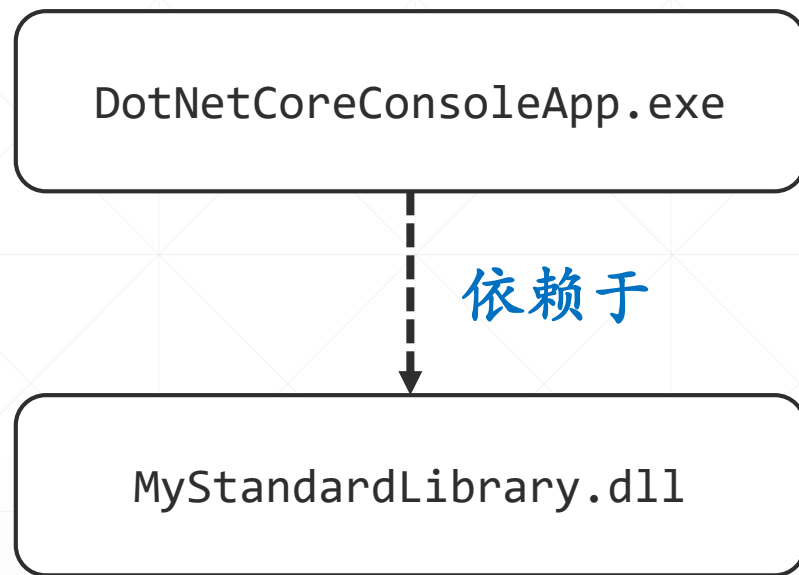
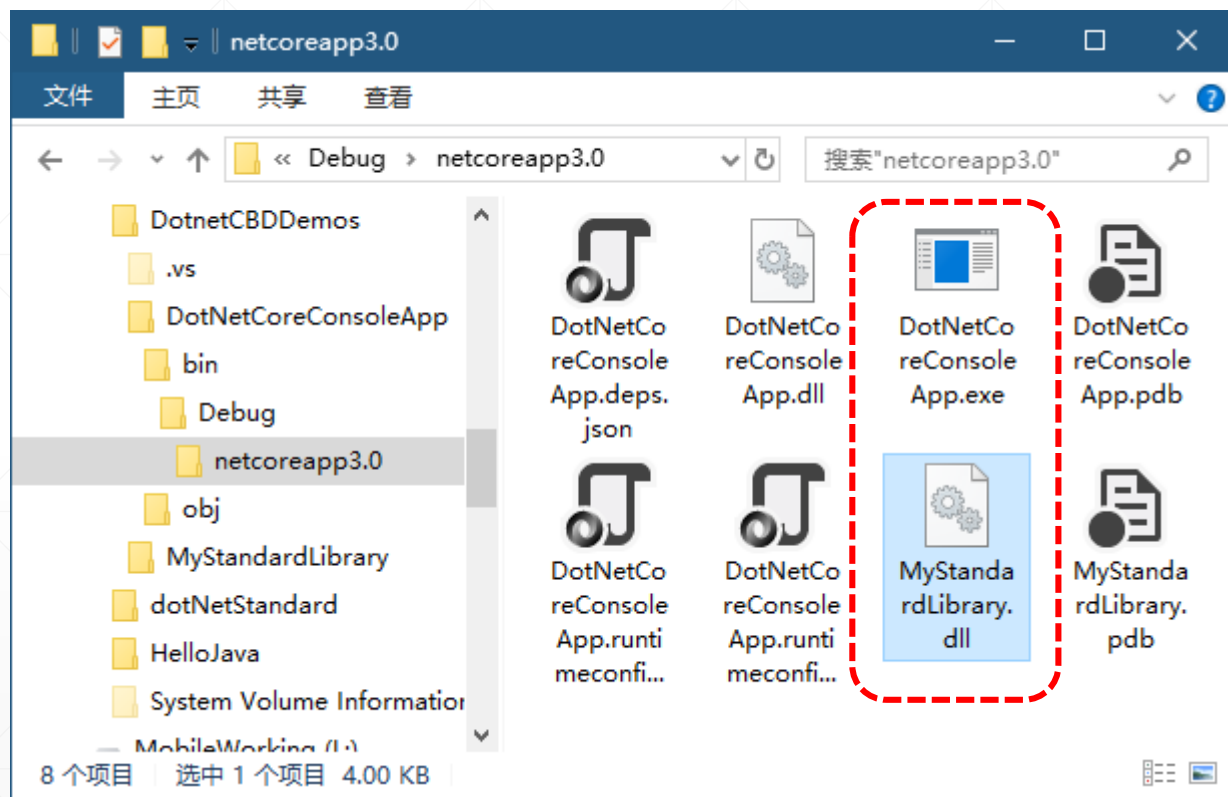
The screenshot shows a code editor window titled "DotNetCoreCo...leApp.csproj". The code is XML for a .NET project. Line 8 contains an `<ItemGroup>` element, and line 9 contains a `<ProjectReference Include="..\MyStandardLibrary\MyStandardLibrary.csproj" />` element. A dashed red box highlights the `<ItemGroup>` and its contents, indicating the project reference.

```
Program.cs*  X
C# DotNetCoreConsoleApp  DotNetCoreConsoleApp.Program  Main(string[] args)
1  using System;
2  //添加对于引用项目命名空间的引用
3  using MyStandardLibrary;
4
5  namespace DotNetCoreConsoleApp
6  {
7      0 references
7      class Program
8      {
9          0 references
9          static void Main(string[] args)
10         {
11             //调用MyStandardLibrary中类的方法
12             Console.WriteLine(LibraryInfo.About());
13             Console.WriteLine($"100+200={MathFunc.Add(100, 200)}");
14         }
15     }
16 }
```

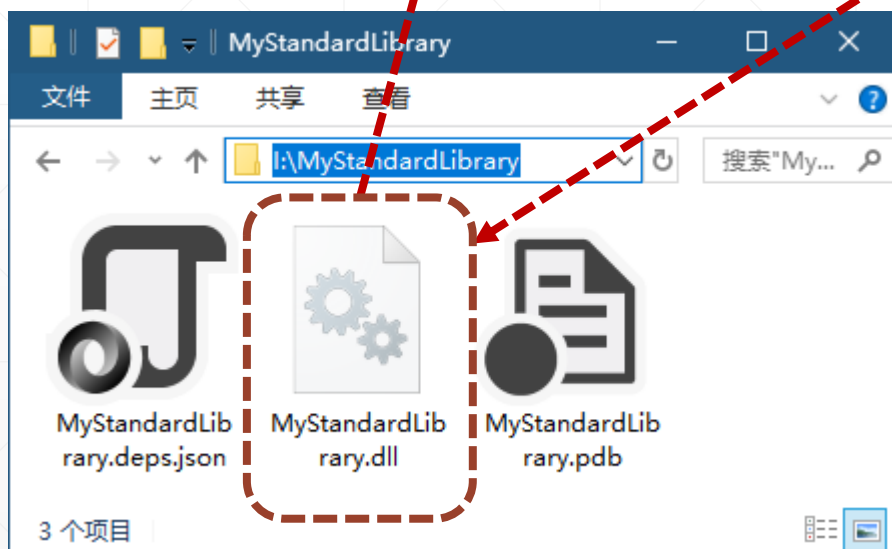
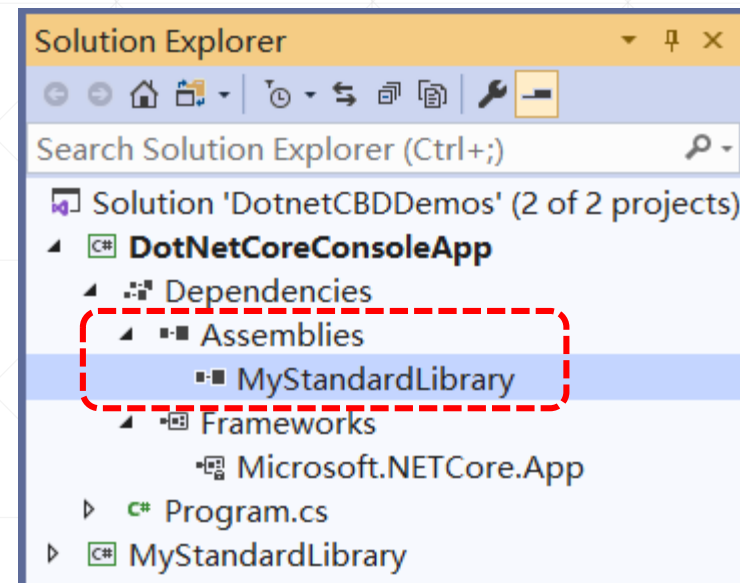
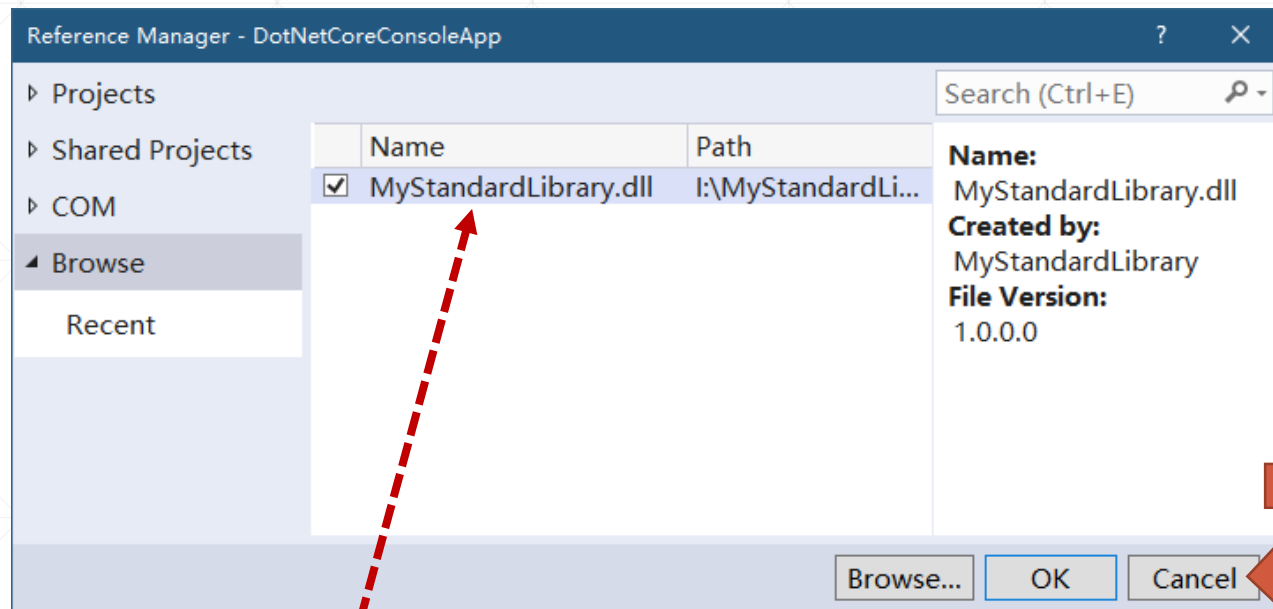
```
Microsoft Visual Studio Debug Console
一个以.NET Standard 2.0为目标框架的示例类库项目
100+200=300

I:\DotnetCBDDemos\DotNetCoreConsoleApp\bin\Debug\netcoreapp3.0
\DotNetCoreConsoleApp.exe (process 1204) exited with code 0.
Press any key to close this window . . .
```

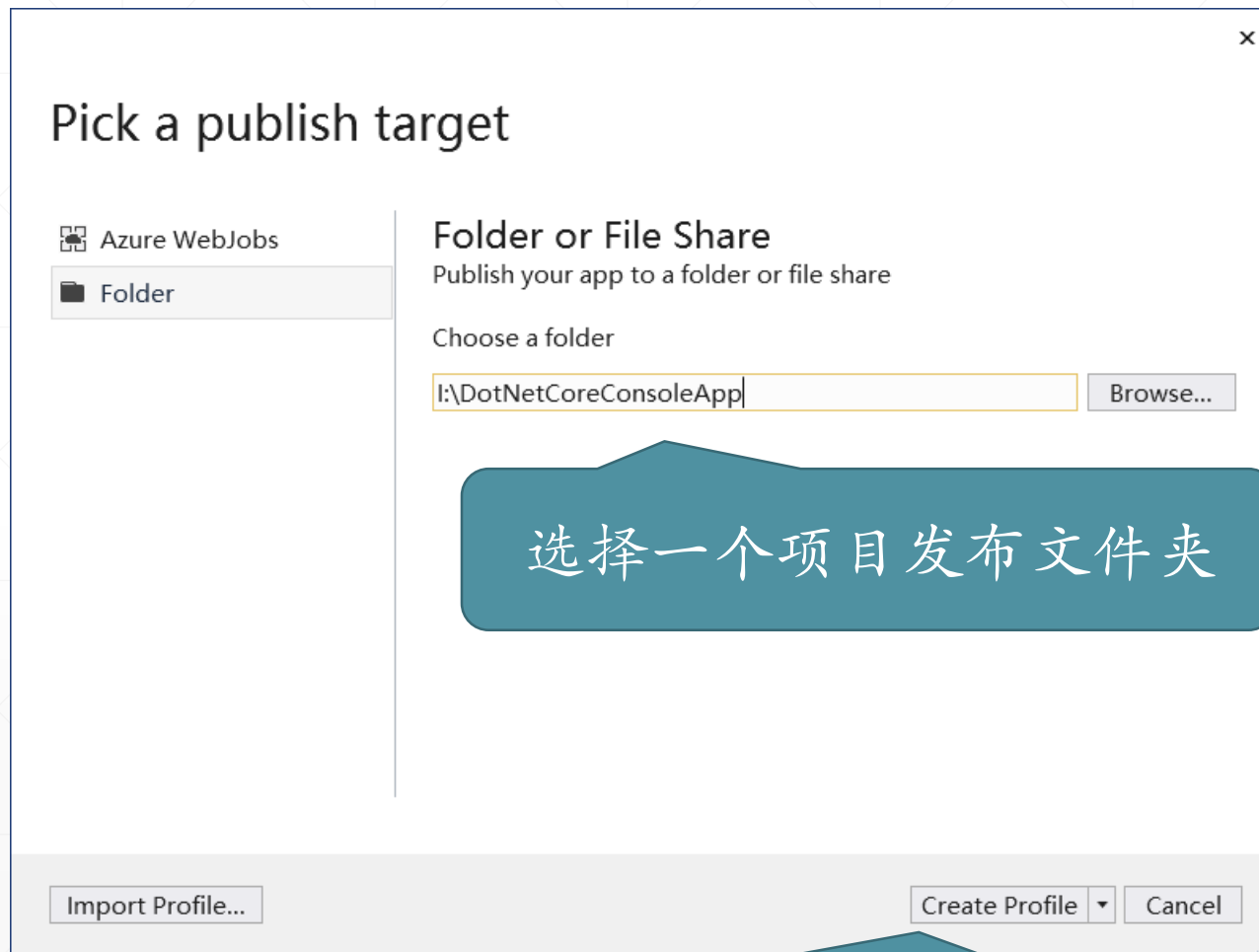
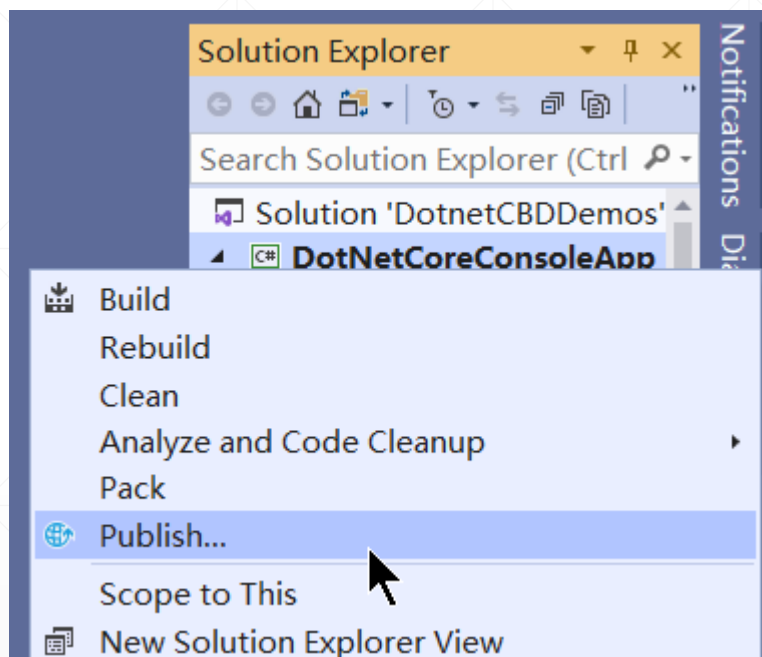
# 程序集之间的依赖关系



程序集间的循环依赖不允许!

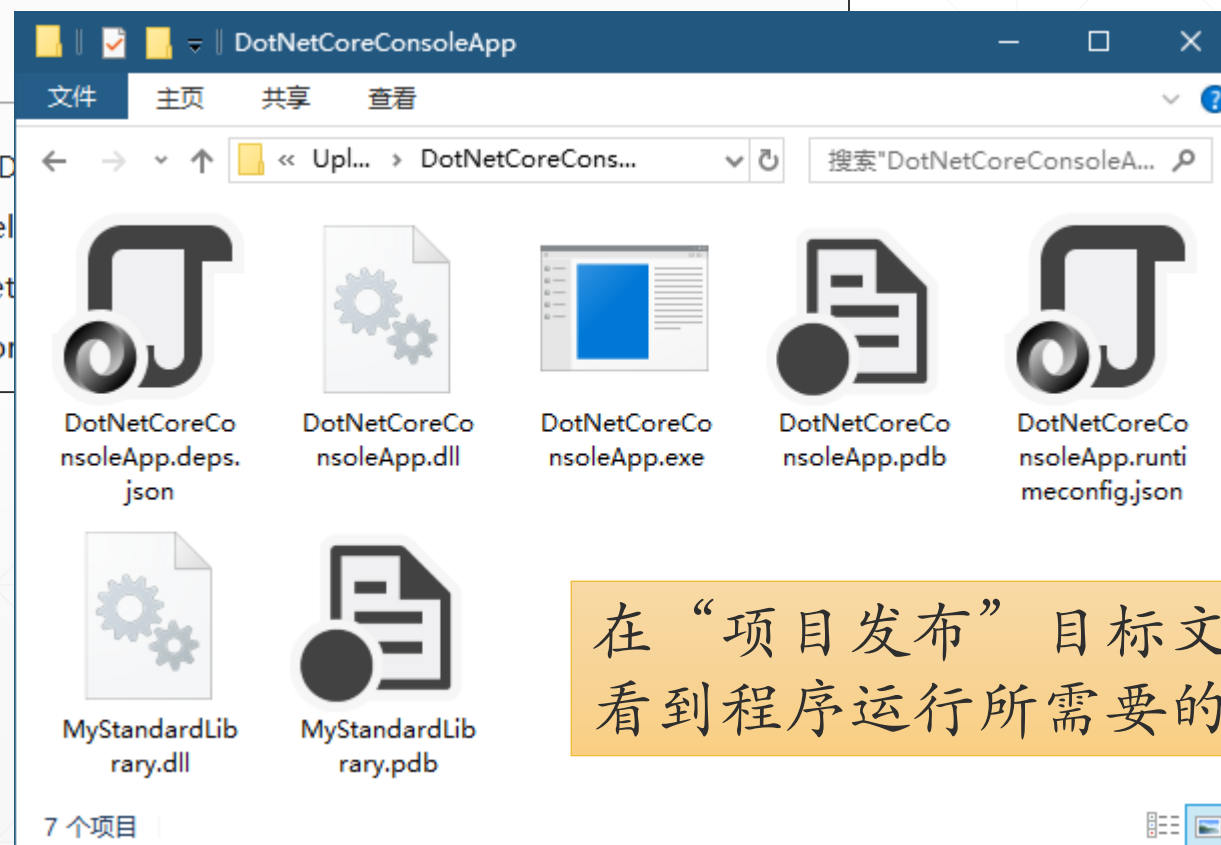
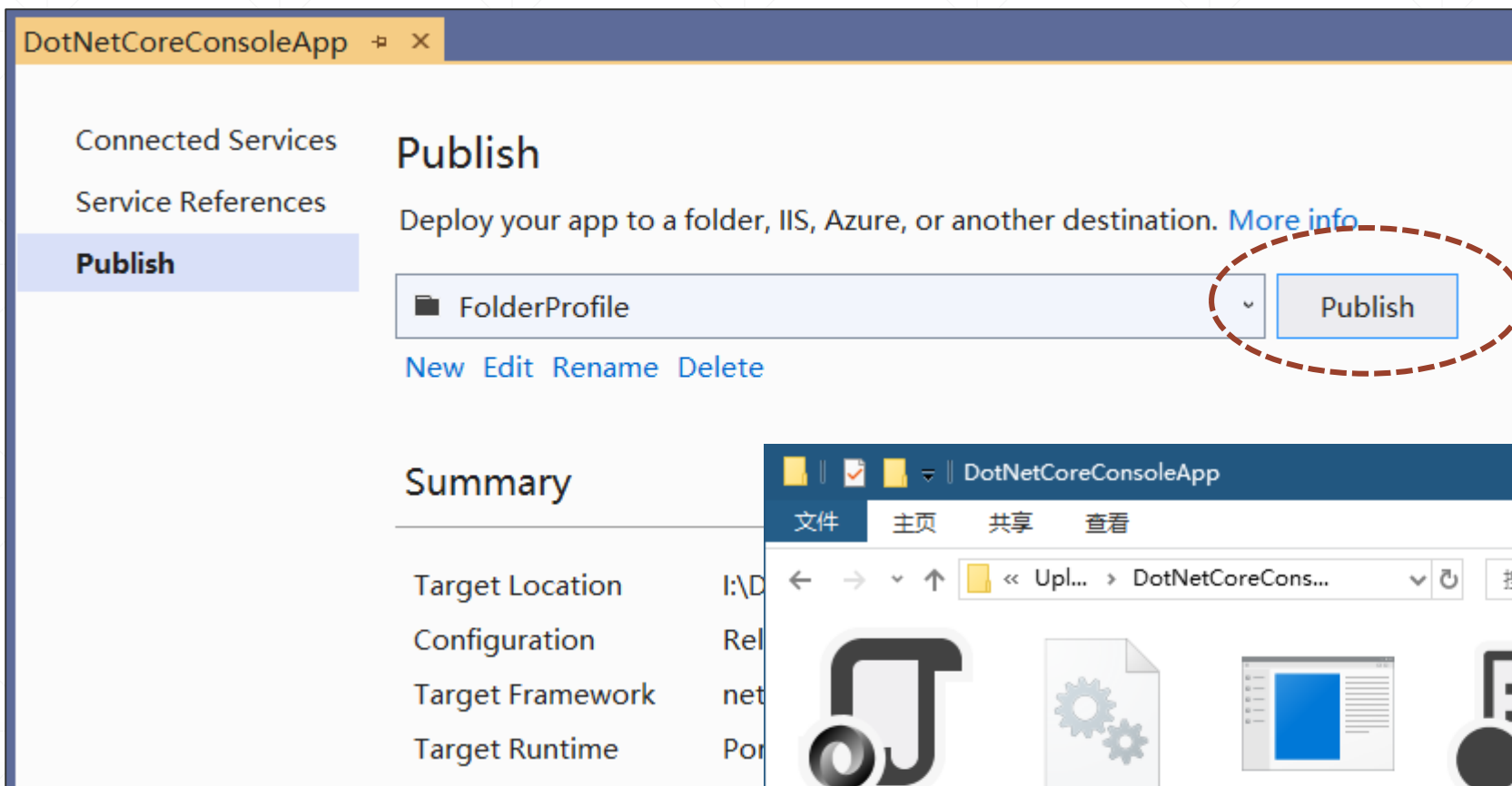


# 组件化的.NET项目的发布



选择一个项目发布文件夹

创建“项目发布”脚本



在“项目发布”目标文件夹中可以看到程序运行所需要的各种文件。

# 运行

```
命令提示符

I:\DotNetCoreConsoleApp>dir
驱动器 I 中的卷是 Upload
卷的序列号是 A084-4F54

I:\DotNetCoreConsoleApp 的目录

2019/07/27  20:40    <DIR>          .
2019/07/27  20:40    <DIR>          ..
2019/07/27  20:40                766 DotNetCoreConsoleApp.deps.json
2019/07/27  20:40            4,608 DotNetCoreConsoleApp.dll
2019/07/27  20:40        160,256 DotNetCoreConsoleApp.exe
2019/07/27  20:40            460 DotNetCoreConsoleApp.pdb
2019/07/27  20:40            172 DotNetCoreConsoleApp.runtimeconfig.json
2019/07/27  20:40            4,096 MyStandardLibrary.dll
2019/07/27  20:40            564 MyStandardLibrary.pdb
                7 个文件            170,922 字节
                2 个目录 209,545,330,688 可用字节

I:\DotNetCoreConsoleApp>DotNetCoreConsoleApp.exe
一个以 .NET Standard 2.0 为目标框架的示例类库项目
100+200=300

I:\DotNetCoreConsoleApp>dotnet DotNetCoreConsoleApp.dll
一个以 .NET Standard 2.0 为目标框架的示例类库项目
100+200=300

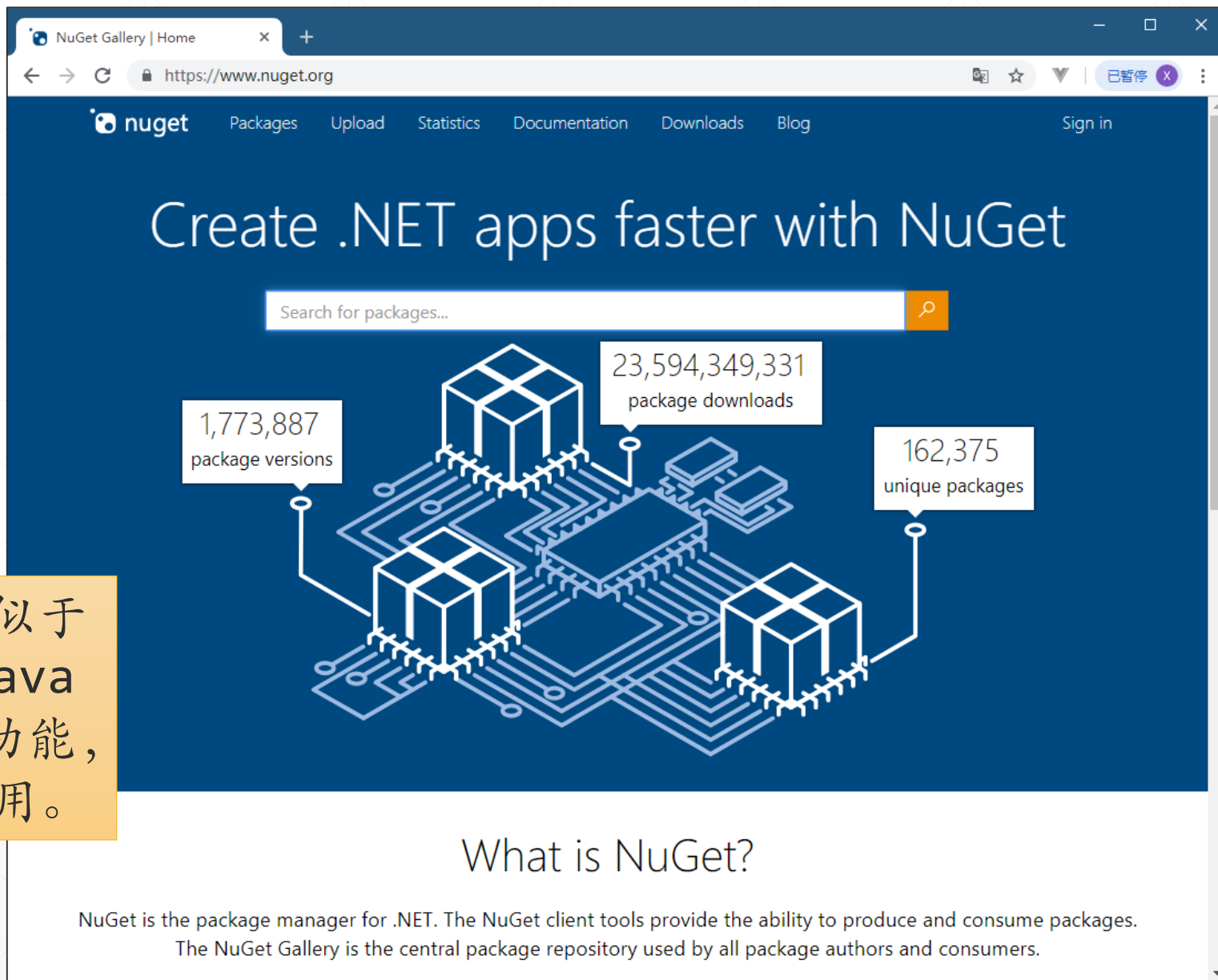
I:\DotNetCoreConsoleApp>_
```



# 重用社区他人代码

<https://www.nuget.org>

.NET平台的NuGet，其作用类似于Node.js中的npm，也类似于Java中的Maven，都具备包管理的功能，可以用于跨互联网实现代码重用。



The screenshot shows the NuGet Gallery homepage with a dark blue header and a white search bar. The main content area features a large illustration of a circuit board with three 3D cubes representing packages. Callout boxes provide statistics: 1,773,887 package versions, 23,594,349,331 package downloads, and 162,375 unique packages. Below the illustration, the text 'What is NuGet?' is followed by a definition of NuGet and the NuGet Gallery.

NuGet Gallery | Home

https://www.nuget.org

nuget Packages Upload Statistics Documentation Downloads Blog Sign in

## Create .NET apps faster with NuGet

Search for packages...

1,773,887 package versions

23,594,349,331 package downloads

162,375 unique packages

### What is NuGet?

NuGet is the package manager for .NET. The NuGet client tools provide the ability to produce and consume packages. The NuGet Gallery is the central package repository used by all package authors and consumers.

# 使用NuGet安装基于互联网共享的.NET组件

The screenshot displays the Visual Studio interface with the NuGet Package Manager and Solution Explorer windows open.

**NuGet Package Manager: DotNetCoreConsoleApp**

Package source: nuget.org

Search (Ctrl+L) [ ] Include prerelease

**Installed Packages:**

- NUnit** by Charlie Poole, Rob Prouse, 47.2M downloads, v3.12.0. Description: NUnit is a unit-testing framework for all .NET languages with a strong TDD...
- Newtonsoft.Json** by James Newton-King, 254M downloads, v12.0.2. Description: Json.NET is a popular high-performance JSON framework for .NET
- EntityFramework** by Microsoft, 67.3M downloads, v6.2.0. Description: Entity Framework is Microsoft's recommended data access technology for...
- MySQL.Data** by Oracle, 8.11M downloads, v8.0.17. Description: MySQL.Data.MySqlClient .Net Core Class Library

Each package is licensed to you by its owner. NuGet is not responsible for, nor does it grant any licenses to, third-party packages.

☐ Do not show this again

**Newtonsoft.Json Details:**

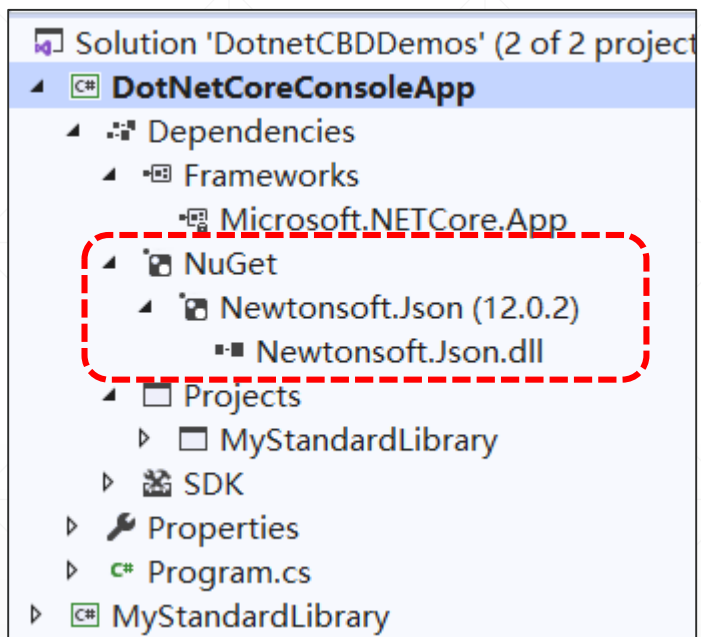
- Version: Latest stable 12.0.2
- Install button
- Options: [ ]
- Description: Json.NET is a popular high-performance framework for .NET
- Version: 12.0.2
- Author(s): James Ne...
- License: MIT
- Date published: Monday, (4/22/20...
- Project URL: https://w...

**Solution Explorer:**

Solution 'DotnetCBDDemos' (2 of 2 projects)

- DotNetCoreConsoleApp**
  - Dependencies
    - Add Reference...
    - Add Connected Service
    - Manage NuGet Packages... (highlighted)
    - Scope to This
    - New Solution Explorer View
- MyStandardLibrary
  - Dependencies
    - SDK
    - LibraryInfo.cs
    - MathFunc.cs
    - MyClass.cs

# 使用NuGet包



DotNetCoreConsoleApp.csproj

```
<ItemGroup>  
  <PackageReference Include="Newtonsoft.Json" Version="12.0.2" />  
</ItemGroup>
```

Program.cs

```
//使用通过NuGet安装的Newtonsoft.json包中的类  
1 reference  
private static void UseClassesInNewtonSoftJson()  
{  
    var obj = new MyClass()  
    {  
        Value = 100,  
        info = "Hello,Newtonsoft.json"  
    };  
    //输出: {"Value":100,"info":"Hello,Newtonsoft.json"}  
    Console.WriteLine($"{JsonConvert.SerializeObject(obj)}");  
}
```

# 小结



基于.NET平台开发，通常会将整个系统划分为若干个组件，然后按照特定的顺序依次开发这些组件，最后把组件装配为完整的应用。



对于位于系统中间层次的组件，通常使用“类库（Class Library）”项目来开发它们，编译之后得到dll程序集。



.NET Core应用通过“项目引用”或“直接引用外部程序集”的方式重用本地组件，使用NuGet方式重用第三方组件。