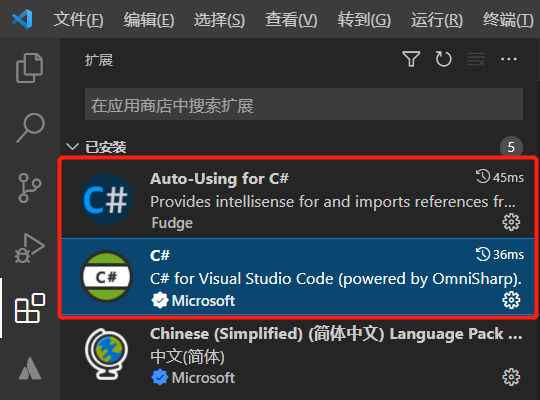
如下，本篇文章带你了解.NET的项目结构，为了更好的了解项目结构，本篇文章使用VSCode作为IDE

**vscode扩展**

使用vscode作为IDE，我们需要安装2个扩展



C#：必装

Auto-Using for C#：可选

**解决方案**

.NET的项目一般都是以解决方案开始的

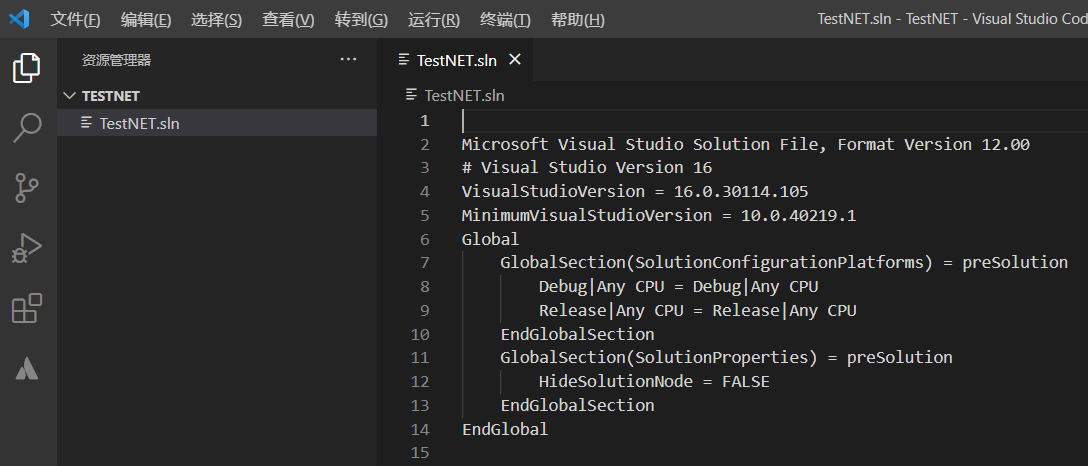
1. 创建解决方案

1）创建一个文件夹 TestNET

2）TestNET目录下执行：dotnet new sln，该命令创建一个解决方案

1. 解决方案

如下，我们可以看到解决方案其实就是一个文件



1. dotnet sln 命令

dotnet sln是解决方案的命令，如下列车笔记常见的选项

-h：常用，几乎所有命令都有该选项，命令帮助选项，示例 dotnet sln -h

add <PROJECT\_PATH>：添加项目到解决方案，示例 dotnet sln add Start

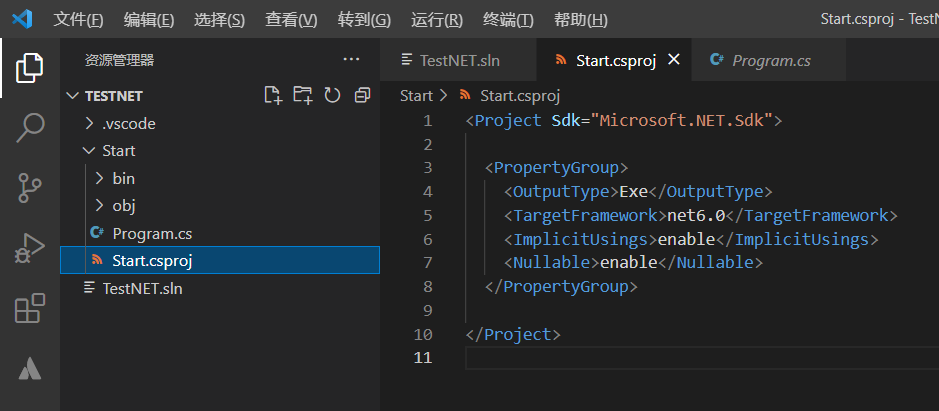
remove <PROJECT\_PATH>：从解决方案移除项目，示例 dotnet sln remove Start

**项目**

一个解决方案包含多个项目

1. 创建项目
2. 在解决方案下创建文件夹Start
3. 在Start目录下执行命令dotnet new console -f net6.0，该命令创建一个console（控制台）项目，目标框架为net6.0
4. 执行命令dotnet sln add Start，该命令项目添加到解决方案
5. 项目

如下，Start项目，Start.csproj就代表了Start项目，Program.cs是console类型项目所必须的，bin | obj是编译文件夹，不属于项目内容（可以删除，但每次运行它都会生成）



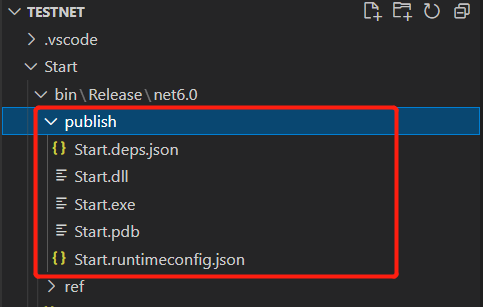
1. 运行项目

解决方案目录下执行 dotnet run --project Start，运行Start项目



**发布项目/解决方案**

1. 执行dotnet publish --configuration Release发布解决方案
2. 在\TestNET\Start\bin\Release\net6.0\publish目录可以看到发布的项目



1. 运行项目

发布目录下执行dotnet HelloWorld.dll运行程序

**类库**

1. 创建类库：dotnet new classlib -o TestLibrary -f net6.0

-o TestLibrary：类库的名称为TestLibrary，目录为TestLibrary

-f net6.0：目标框架为net6.0

1. 将项目添加到解决方案：dotnet sln add .\TestLibrary\
2. Start项目引用TestLibrary项目：dotnet add Start reference TestLibrary

至此，项目Start可以使用类库TestLibrary了

1. 移除项目引用：dotnet remove reference（不懂就 -h）

**单元测试**

1. 创建单元测试项目：dotnet new mstest -o TestLibraryTest -f net6.0
2. 将项目添加到解决方案：dotnet sln add TestLibraryTest
3. 引用要测试的项目：dotnet add TestLibraryTest reference TestLibrary
4. 运行单元测试：dotnet test TestLibraryTest



1. 发布版测试：dotnet test TestLibraryTest --configuration Release

默认情况下，我们运行的是debug版本的测试，要想测试发布版，需添加--configuration Release

**Nuget包使用**

如下，我们已TestLibrary项目为例对包进行讲解

1. 添加包引用：dotnet add TestLibrary package Newtonsoft.Json

TestLibrary：项目名

package Newtonsoft.Json：要安装的包Newtonsoft.Json

安装完成后我们就可以使用该包了

1. 移除包可以使用：dotnet remove package（不懂就 -h）
2. 还原包：dotnet restore

**\*\*\*.csproj项目文件**

如下说明项目文件的字段

<Project Sdk="Microsoft.NET.Sdk">

  <ItemGroup>

    <!-- 当前项目引用的项目 -->

    <ProjectReference Include="..\TestLibrary\TestLibrary.csproj" />

  </ItemGroup>

  <ItemGroup>

    <!-- 当前项目引用的包 -->

    <PackageReference Include="Newtonsoft.Json" Version="13.0.1" />

  </ItemGroup>

  <PropertyGroup>

    <!-- 项目的打包类型，这里为 exe 可执行文件 -->

    <OutputType>Exe</OutputType>

    <!-- 项目的目标框架，这里为 net6.0 -->

    <TargetFramework>net6.0</TargetFramework>

    <!-- 是否启用自动 Using，这是 net6.0 的特性，详情看 net6.0 说明 -->

    <ImplicitUsings>enable</ImplicitUsings>

    <!-- 这也是 net6.0 的特性，详情看 net6.0 说明 -->

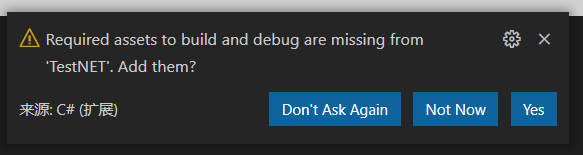
    <Nullable>enable</Nullable>

  </PropertyGroup>

</Project>

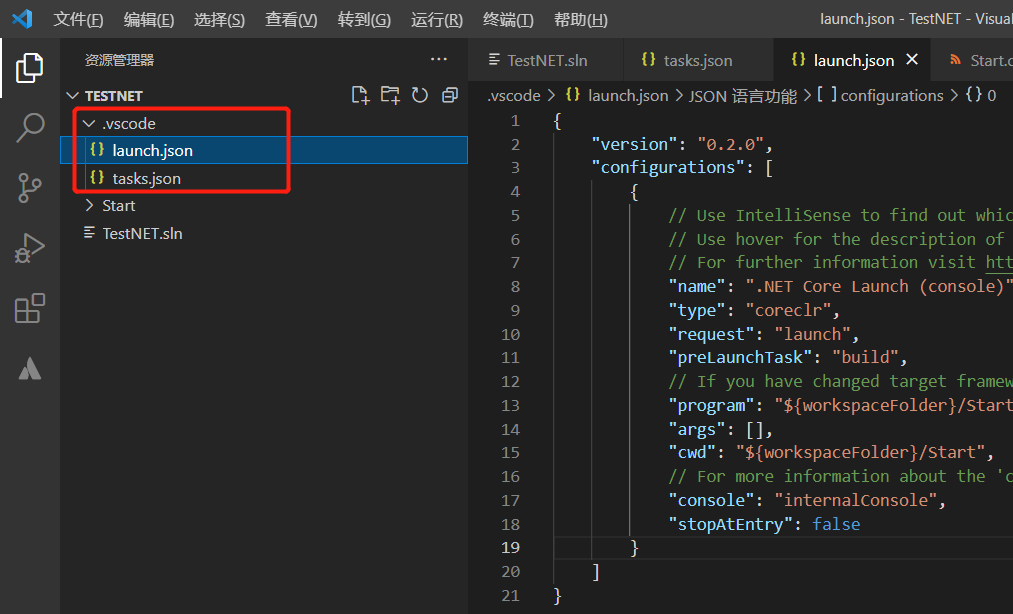
**vscode配置**

当你添加控制台项目时，vscode右下角会提示你添加构建和调试的配置（c#扩展插件的提示），点击Yes添加调试配置文件，如果没有提示那就重启vscode



1. 配置文件

如下是vscode的调试配置文件



launch.json是启动配置文件

tasks.json是任务配置文件

1. launch.json

如下，看注释

{

    "version": "0.2.0",

    "configurations": [

        {

            // 显示的名称

            "name": ".NET Core Launch (console)",

            // netcore CLR

            "type": "coreclr",

            "request": "launch",

            // 启动前执行的任务，如下执行名为 build 的任务

            "preLaunchTask": "build",

            // 启动的入口

            "program": "${workspaceFolder}/Start/bin/Debug/net6.0/Start.dll",

            "args": [],

            "cwd": "${workspaceFolder}/Start",

            // 控制台类型 internalConsole 为内部控制台

            "console": "internalConsole",

            "stopAtEntry": false

        }

    ]

}

1. tasks.json

tasks.json是任务配置文件，launch.json启动程序前可能会执行任务

{

    "version": "2.0.0",

    "tasks": [

        {

            // 任务的名称

            "label": "build",

            // 任务的命令

            "command": "dotnet",

            "type": "process",

            // 命令的参数

            "args": [

                "build",

                "${workspaceFolder}/Start/Start.csproj",

                "/property:GenerateFullPaths=true",

                "/consoleloggerparameters:NoSummary"

            ],

            "problemMatcher": "$msCompile"

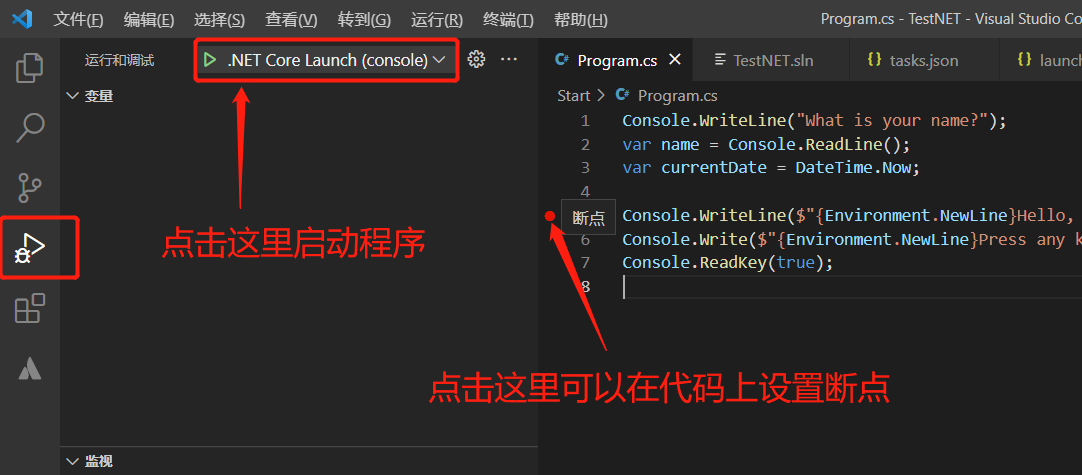
        }

    ]

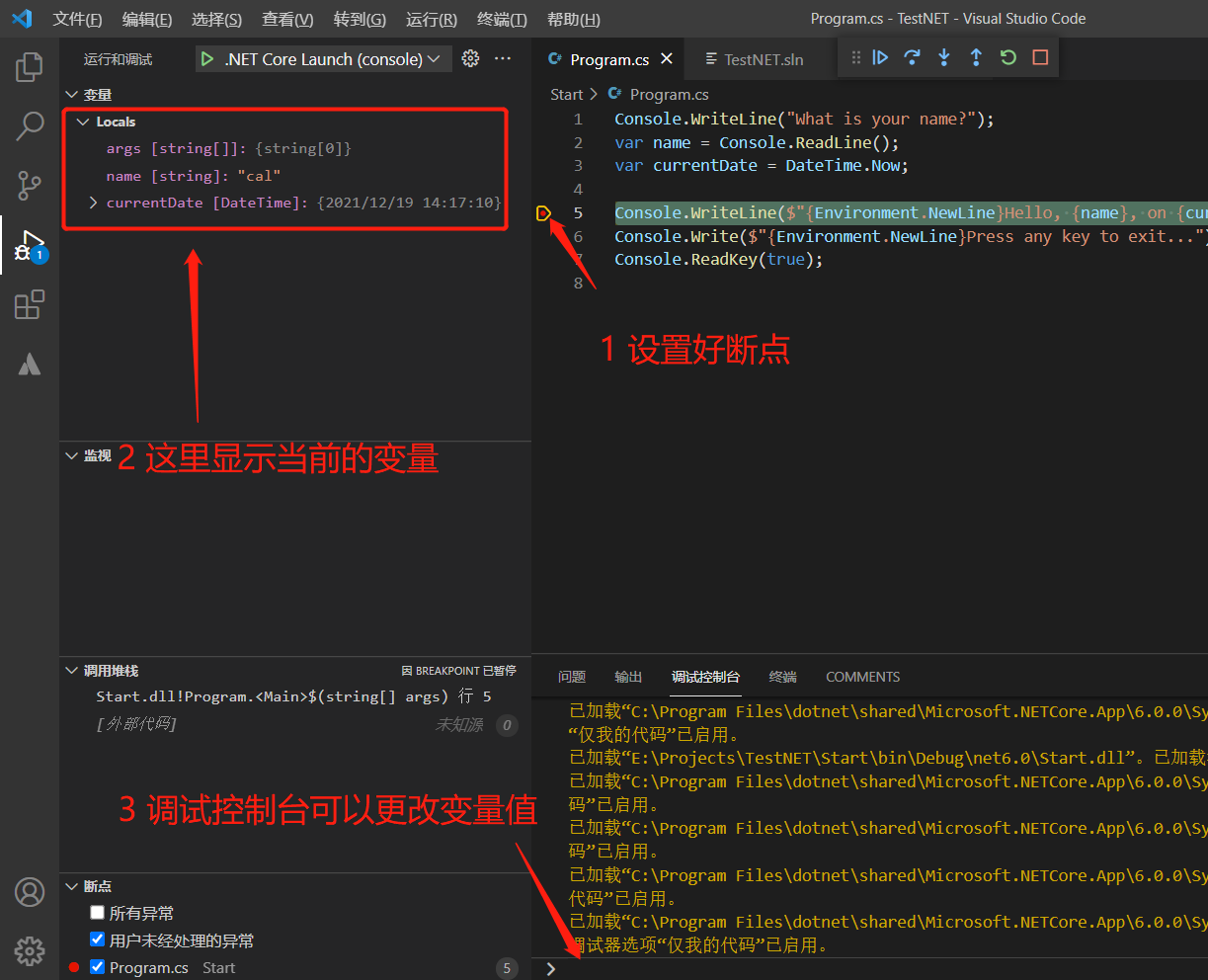
}

1. 启动和debug

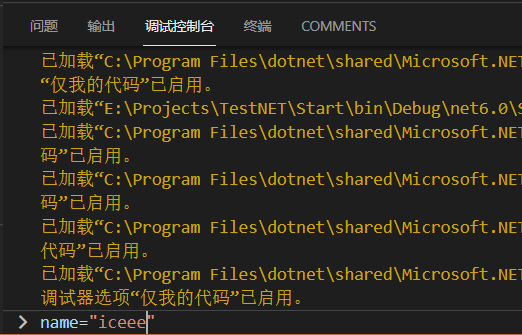
配置好vscode的配置文件后可以从vscode调试代码了，如下



1. 调试控制台讲解
2. 将launch.json的字段设为 "console": "externalTerminal"（内部控制台无法输入）
3. 调试控制台



1. 示例：更改name



1. 示例：更改args

输入args=new string[] { "AAA", "BBB" }

1. 手动添加vscode调试配置

就是按照上面的文件添加就行了