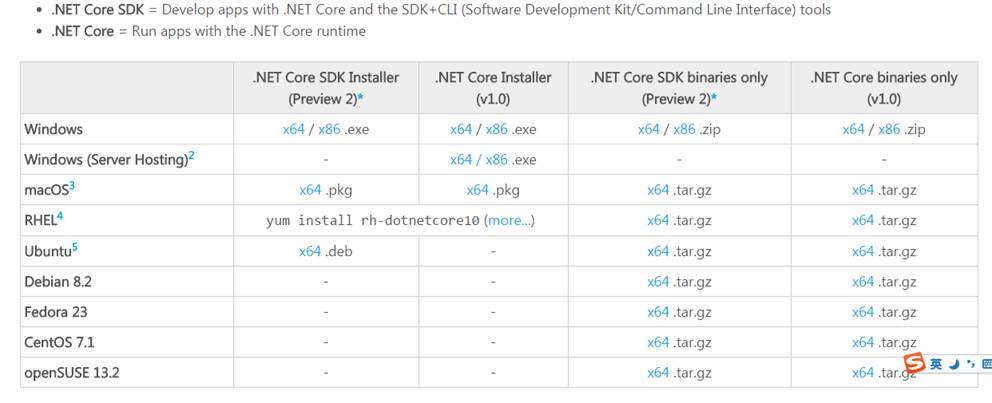
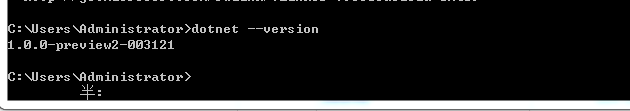
**[asp.net core开发环境准备](http://www.cnblogs.com/xuanhun/p/5629589.html)**

**1.1  安装sdk和运行时**

浏览器打开网址https://www.microsoft.com/net/download， 到.Net Core下载页面。



根据操作系统，下载对应的SDK进行安装。安装之后可以从命令行运行dotnet命令，查看是否安装成功。



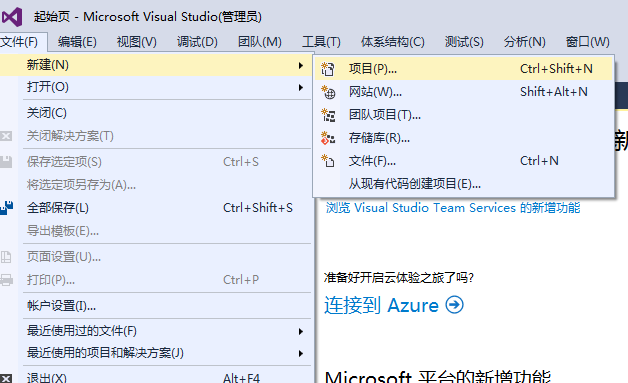
**1.2  使用VS2015**

首先确保你的电脑上安装Visual Studio 2015 Update3，当然我们也可以安装 Visual Studio Community 2015（<https://www.visualstudio.com/products/visual-studio-community-vs>），来替代收费版本的Visual Studio 2015.

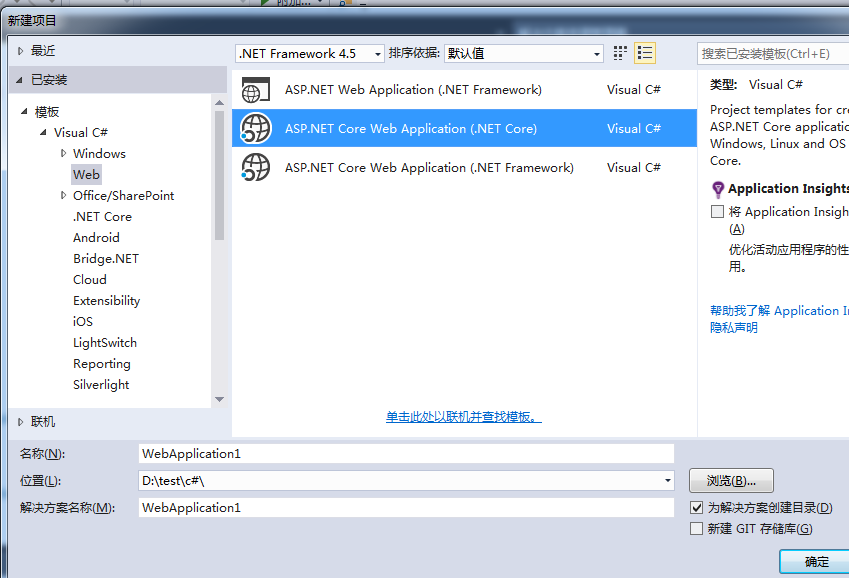
VS2015 下载地址：<https://www.visualstudio.com/zh-cn/downloads/download-visual-studio-vs.aspx>

然后到<https://go.microsoft.com/fwlink/?LinkId=817245>  下载“NET Core Tooling Preview 2 for Visual Studio 2015”

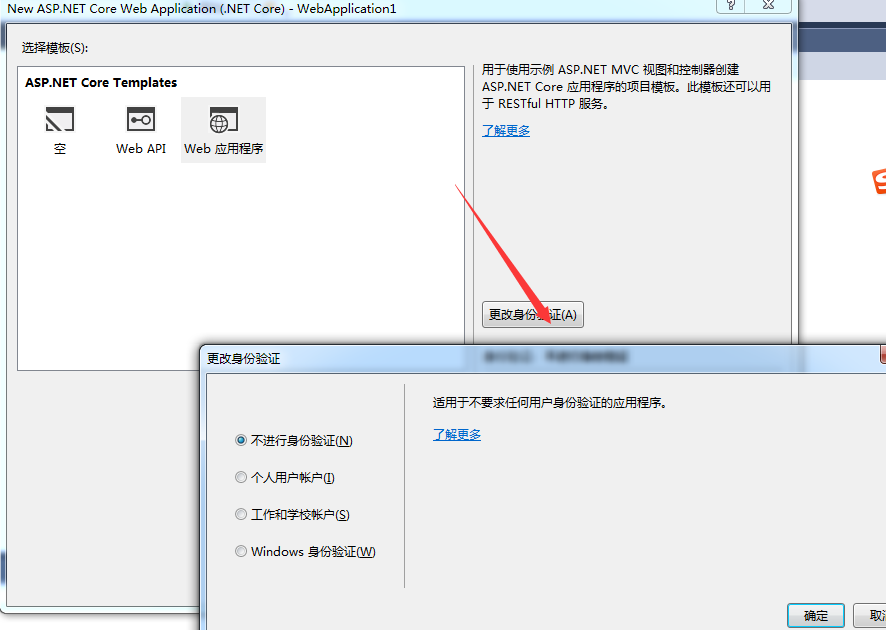
准备妥当之后，打开VS2015.



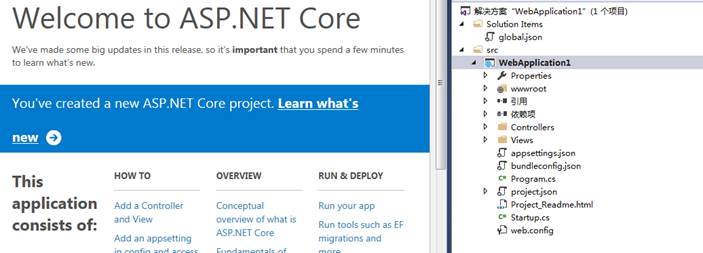
新建项目，选择Asp.net Core Application 项目。



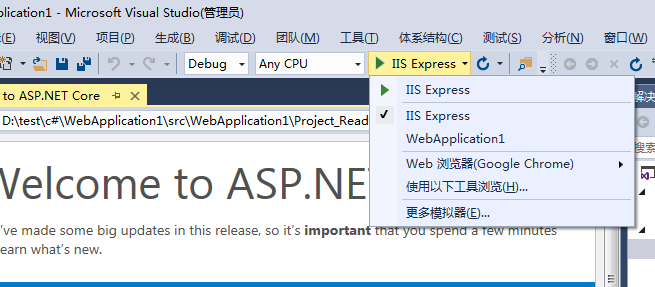
配置好项目文件的位置之后，点击确定，选择模板，身份验证方式。



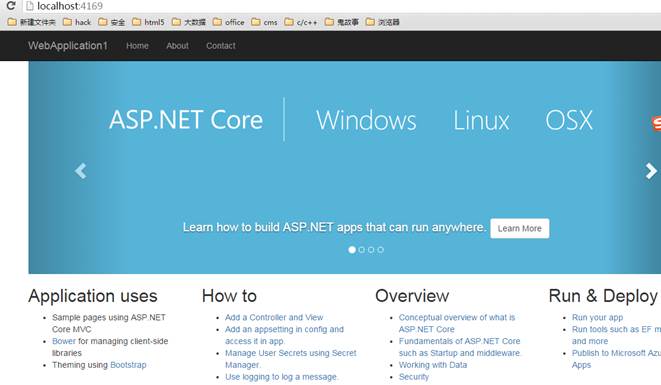
点击确定后，项目就生成了。



启动调试的时候有两种选择：



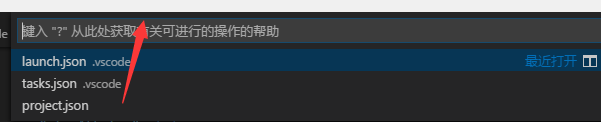
使用IIS Express做宿主或者选择 Application自宿主方式。效果如下：



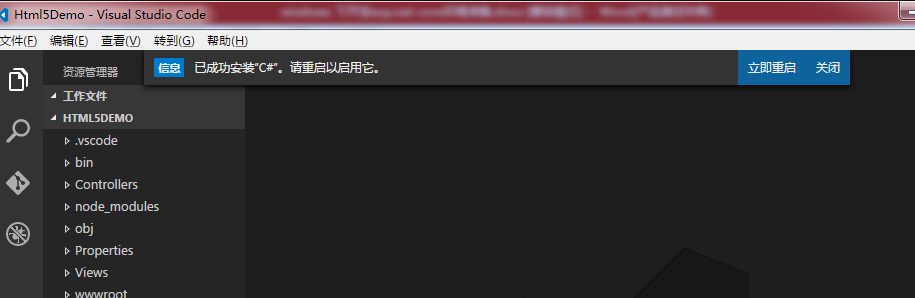
**1.3  使用VSCode(跨平台方案)**

通用的跨平台解决方案是使用Visual Studio Code，我们首先到<https://code.visualstudio.com/> 下载VS Code，进行安装。

安装之后，打开VS Code，Ctrl+P 打开Quick Open，输入如下命令，安装C# for Visual Studio Code。

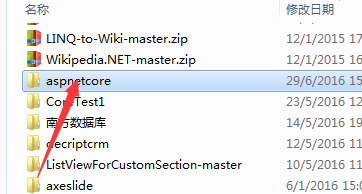


ext install csharp



安装成功之后，重启VS Code。

下面我们新建一个文件夹，用来测试Asp.net Core。

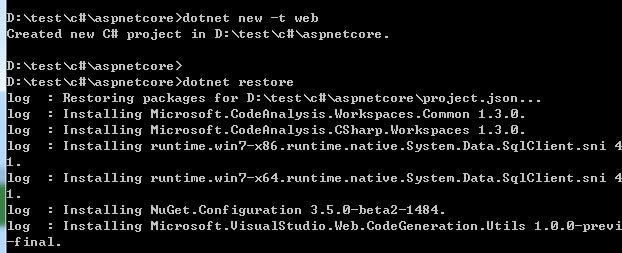


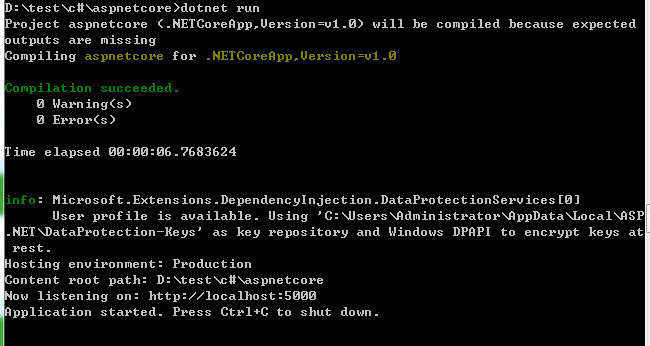
进入文件夹，在命令行分别执行如下命令：

dotnet new -t web

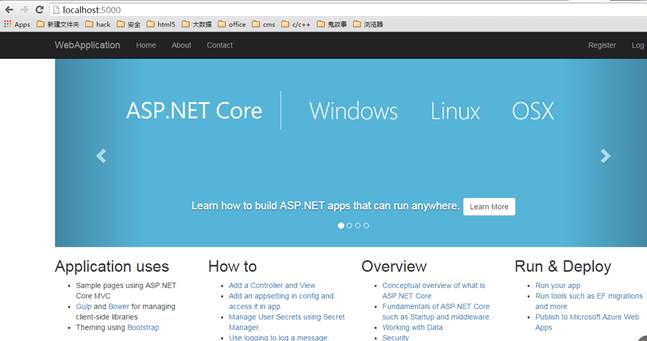
dotnet restore

dotnet run

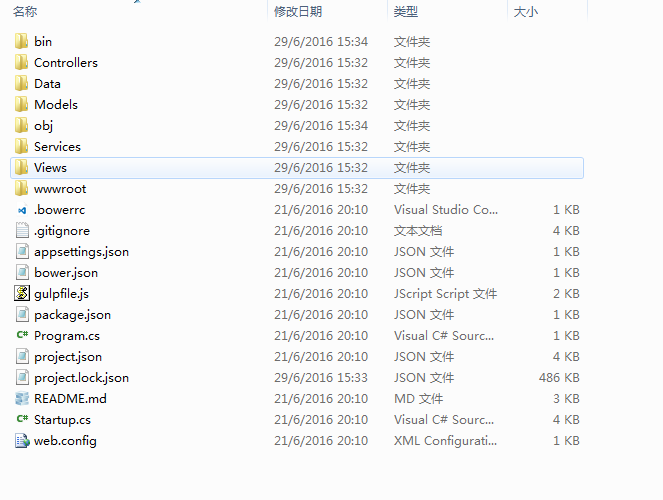




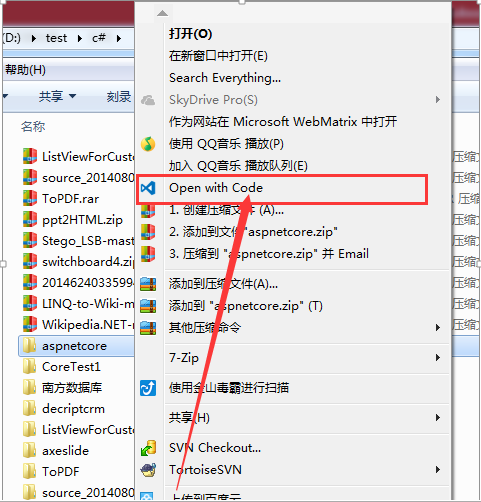
现在在浏览器打开[http://localhost:5000](http://localhost:5000/),可以看到默认生成的站点。



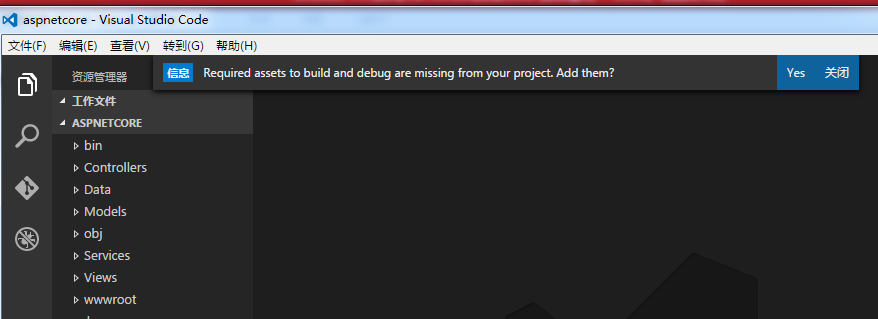
在命令行“CTRL+C”，结束测试，下面我们用VS Code打开项目文件夹。当前生成的内容如下：



右键用VS Code打开文件夹。



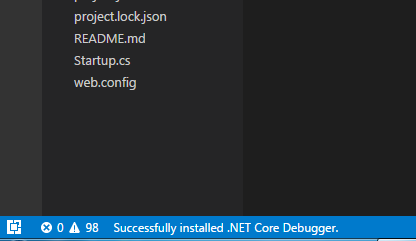
这个时候会提醒，下载调试和编译工具，点击“Yes”。



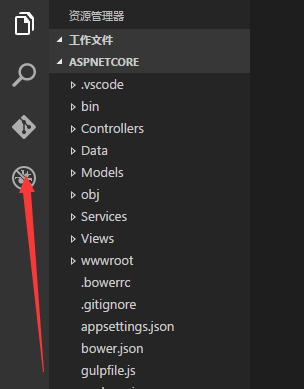
注意下面的状态栏信息，不过要注意的是，如果下载失败的话，可能需要考虑代理或者其他方式科学上网了。

http://images2015.cnblogs.com/blog/42533/201606/42533-20160630122216296-1700448349.png

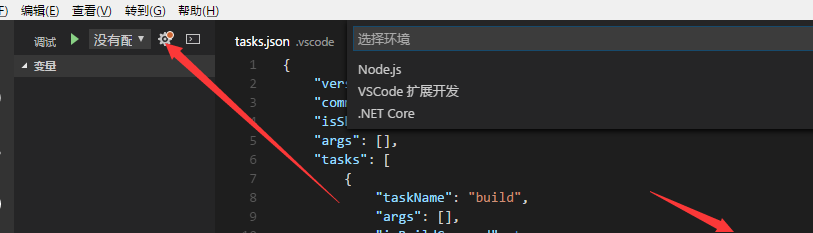
比较幸运，下载和安装成功。



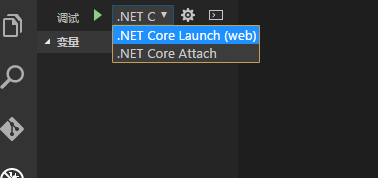
展开调试功能。



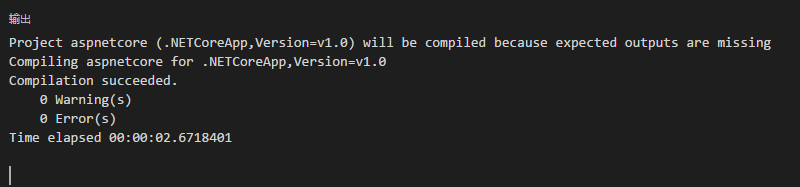
如果调试选项里显示没有匹配，在配置下拉中选择.NET Core。



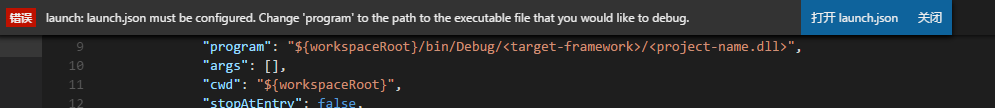
在调试器选择web项目。



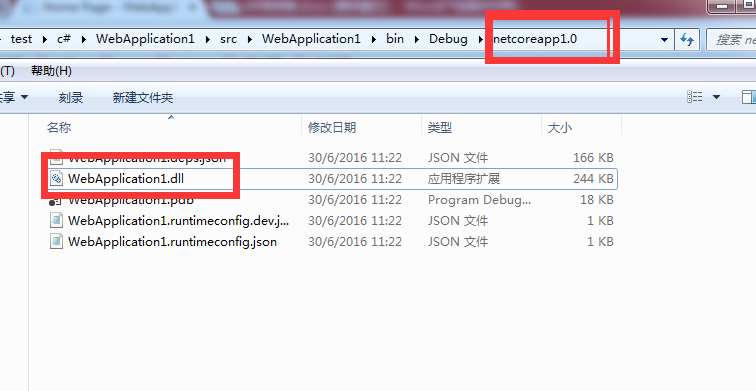
开始调试。



如果打开的是 VS2015创建的项目，可能会出现如下错误：

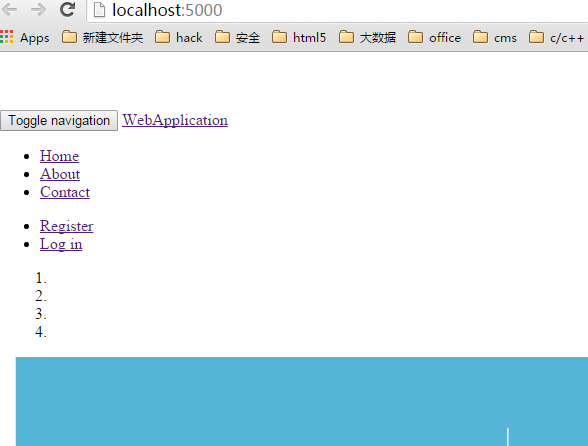


打开launch.json,修改program的路径，具体路径可以查看编译后的bin/debug文件夹里面的内容：

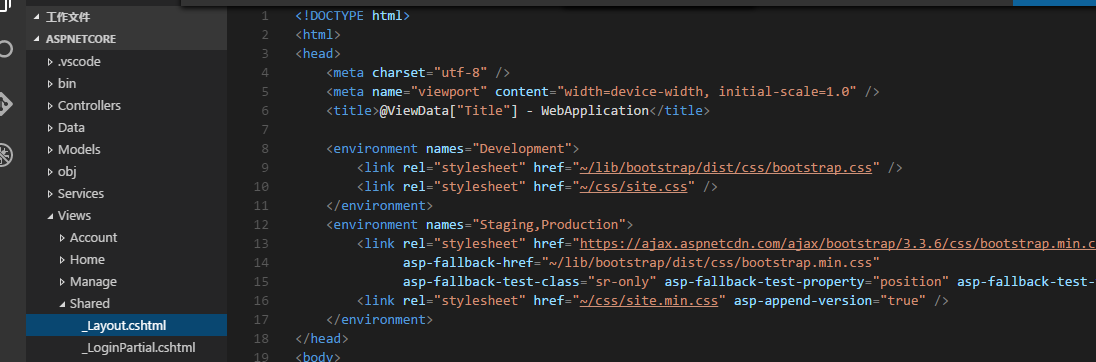




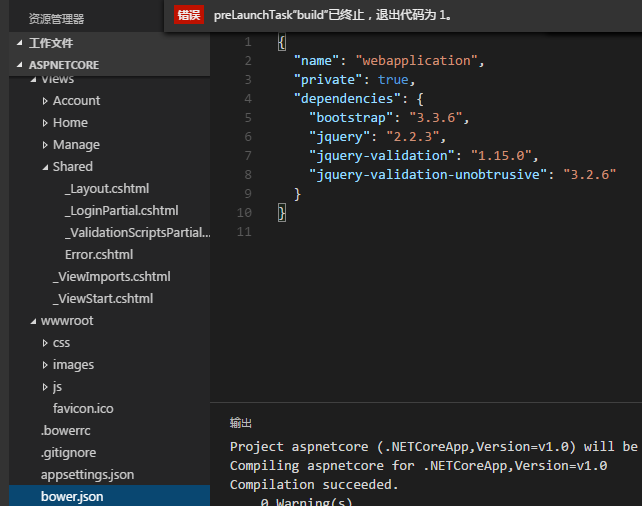
启动调试，编译成功，会自动启动浏览器。



但是我们看到的样式和从命令行看到的不同，原因在于调试模式和Product模式，使用的文件不一样，在\_Layout文件中我们可以看到原因。



开发模式下，引用的bootstrap文件在“wwwroot/lib/bootstrap”目录下，但是我们当前找不到这个目录。项目的JQuery和Bootstrap依赖是通过bower来安装的，这点我们可以从bower.json看到。



看来是需要先安装bower了，通过如下命令进行安装：

npm install -g bower

如果连npm都没有，那就先安装node了，这里我就不再啰嗦下去了。

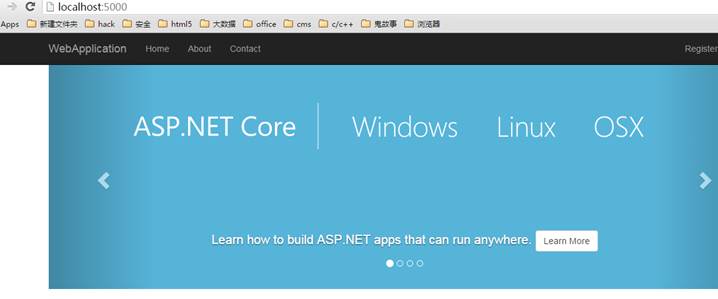
bower 安装成功之后，执行

bower install

安装依赖项。

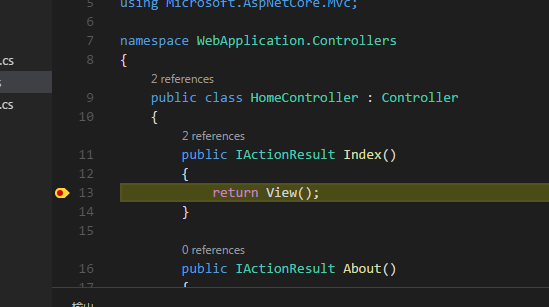


再次启动调试。



现在一切正常了。

需要调试代码的地方，直接单击代码左侧下断点就可以了。



**1.4 Yeoman**

在使用VS Code进行Asp.net Core 开发过程中，我们通常会使用Yeoman和VS Code打配合。Yeoman是一套工具的集合，通过模板的方式，生成Asp.net Core的项目文件，单独生成Contoller、View等文件。

首先我们通过NPM安装一整套依赖项。

npm install -g yo bower grunt-cli gulp

然后再执行下面的命令安装generator-aspnet。

npm install -g generator-aspnet

下面重新开始构建一个Asp.Net Core 项目。

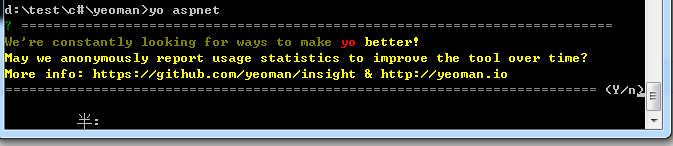
先创建一个测试项目的文件夹。



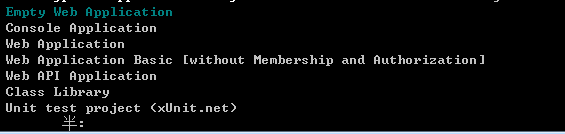
执行

yo aspnet

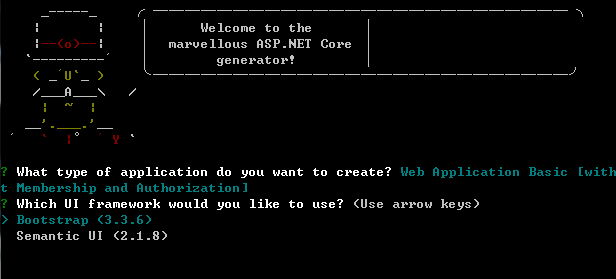
先蹦出来的是是否允许上传使用统计信息。



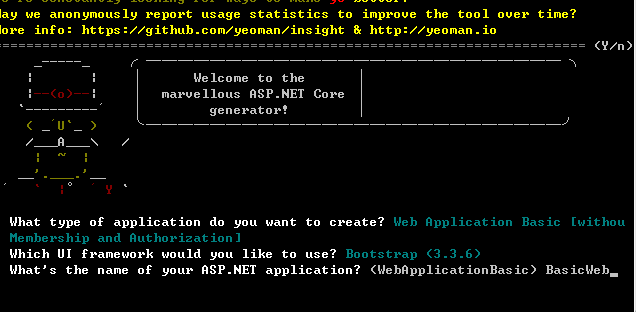
接下来是创建项目的选项：



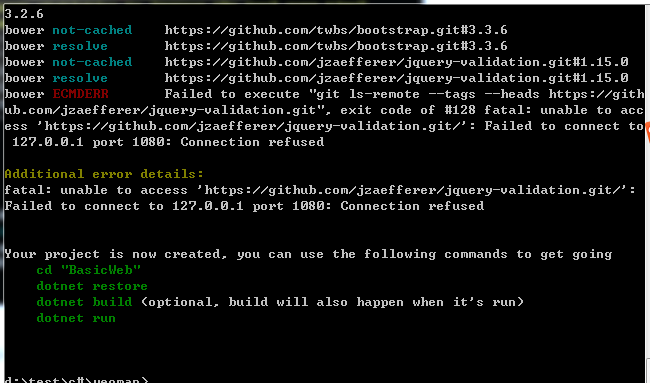
使用键盘上下键进行选择，然后回车键确认，这里我新建一个Basic类型的web项目。



接下来选择使用UI框架，我还是选择Bootstrap。

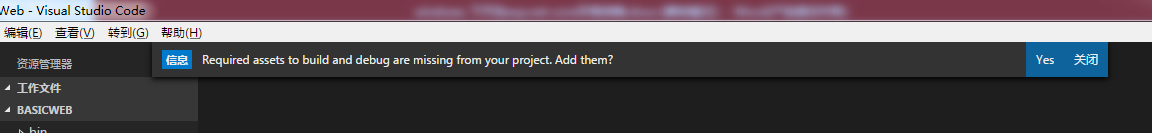


输入要创建项目的名称（BasicWeb）。

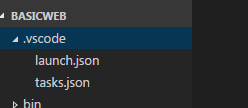


现在提示创建成功了，使用dotnet 的restore、build、run命令分别用来加载依赖，编译和运行。

现在使用VS Code打开该文件夹，可以进行编辑和调试了。

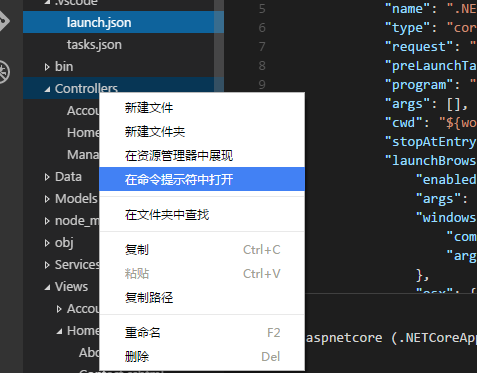


首次打开时仍然会有这样的提示，点击yes，会生成.vscode文件夹，下面会生成编译和启动相关的配置文件。



**添加Controller**

在VS Code中从命令行打开Controller文件夹。



执行命令：

**yo aspnet:MvcController DemoController**



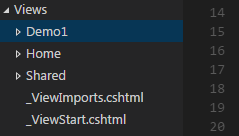
回到vscode中，可以看到生成的文件。



下面再来生成对应的view文件。

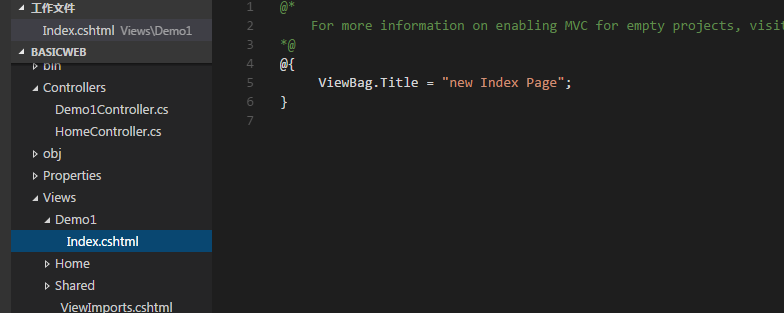
**添加 View**

现在Views文件夹下创建一个Demo1的文件夹用来和Demo1Controller对应。



从命令行打开Demo1文件夹，执行下面的命令创建Index 视图文件。

**yo aspnet:MvcView Index**



**1.5  小结**

乘着 .NET Core 1.0发布的东风，写了这么一篇入门的文章（我发现我一直在写入门的文章，水平有限）。在不可逆转的开源浪潮的驱动下，有理由相信.NET Core的发展会越来越好。我也会将更多更好的这一方面的原创博文分享给各位。让我们共同努力。

 最后要提醒一句，不论项目是何种方式创建的，VSCODE打开之前，都要运行下“dotnet restore”。当然在跨平台开发环境中，一定要熟悉dotnet cli的各种命令，离开了VS2015这样的巨无霸，命令行在linux和Mac下才是最实用的。