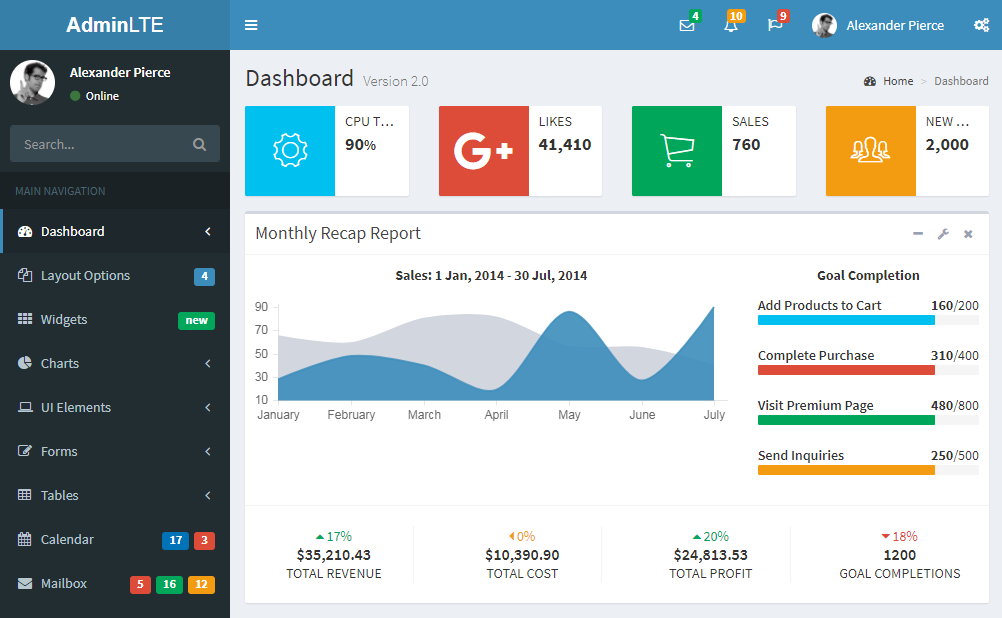
一、ASP.NET Core Web App应用第三方Bootstrap模板

**引言**

作为后端开发来说，前端表示玩不转，我们一般会选择套用一些开源的Bootstrap 模板主题来进行前端设计。那如何套用呢？今天就简单创建一个ASP.NET Core Web MVC 模板项目为例，来应用第三方Bootstrap Template——[Admin LTE](https://adminlte.io/)。



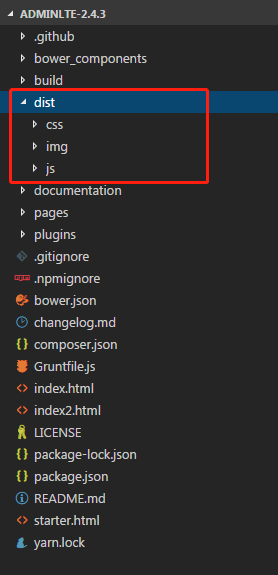
**1、创建项目**

命令行执行dotnet new mvc -n z\_AdminLTE，即可创建预置的MVC模板项目。

从项目结构来看，我们可以看到wwwroot目录下包含了css、images、js、lib目录，其中lib目录默认引用了bootstrap、jquery相关包。因为是简单的模板项目，所以UI就很将就。

**2、下载AdminLTE**

目前AdminLte在计划发布AdminLTE 3.0版本，不过现在还处于Alpha版本。我们[下载AdminLTE-V2.4.3](https://github.com/almasaeed2010/AdminLTE/releases/tag/v2.4.3)来使用。 下载后解压得到的项目结构如下：



**3、替换模板**

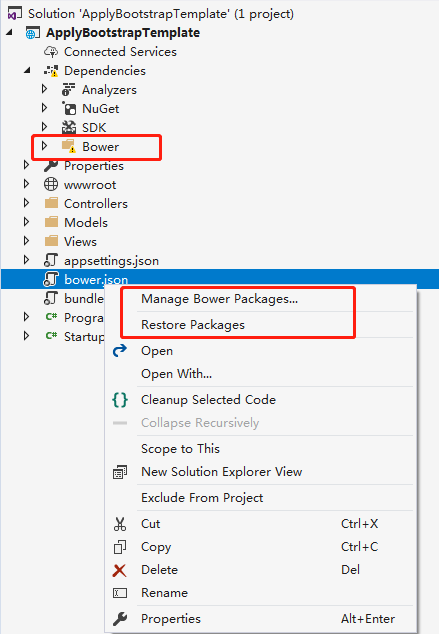
基于AdminLTE进行开发，仅需要复制dist目录，及其依赖的bower包就可以了。

**第一步：**我们清空wwwroot下的全部目录（我这边暂时保留了images文件夹，后面会用到）。

**第二步：**然后复制dist目录到wwwroot下。其依赖的bower包是安装在bower\_components目录下的。我们无需直接复制整个bower\_components文件夹，我们复制bower.json包定义文件即可。

**第三步**：复制AdminLTE下的bower.json到ASP.NET Core Mvc根目录下。

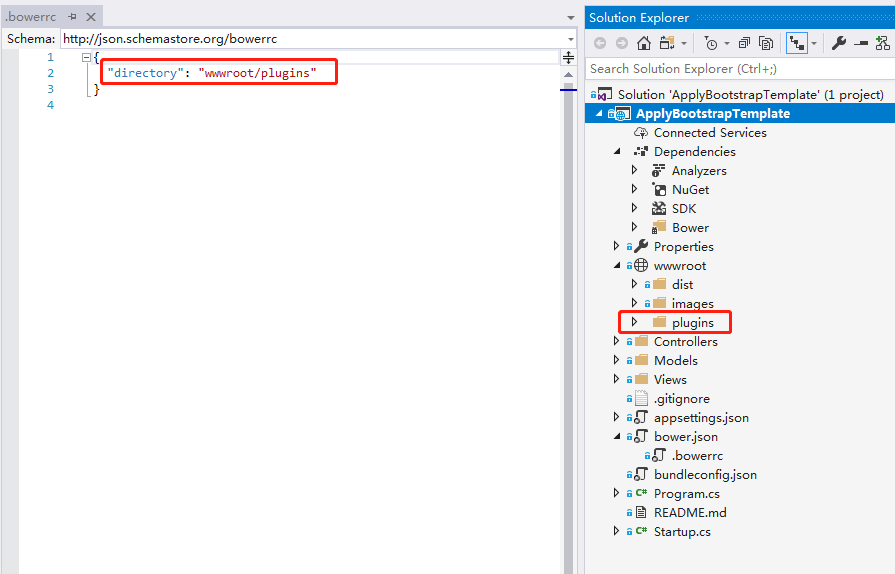
**第四步：**使用VS2017打开项目后，我们可以看到VS2017已经可以识别到未安装的Bower包。



右键就可以还原bower包。不过先慢着，我们现在还原就会直接还原bower包到根目录下了，并没有还原bower包到wwwroot文件夹下。

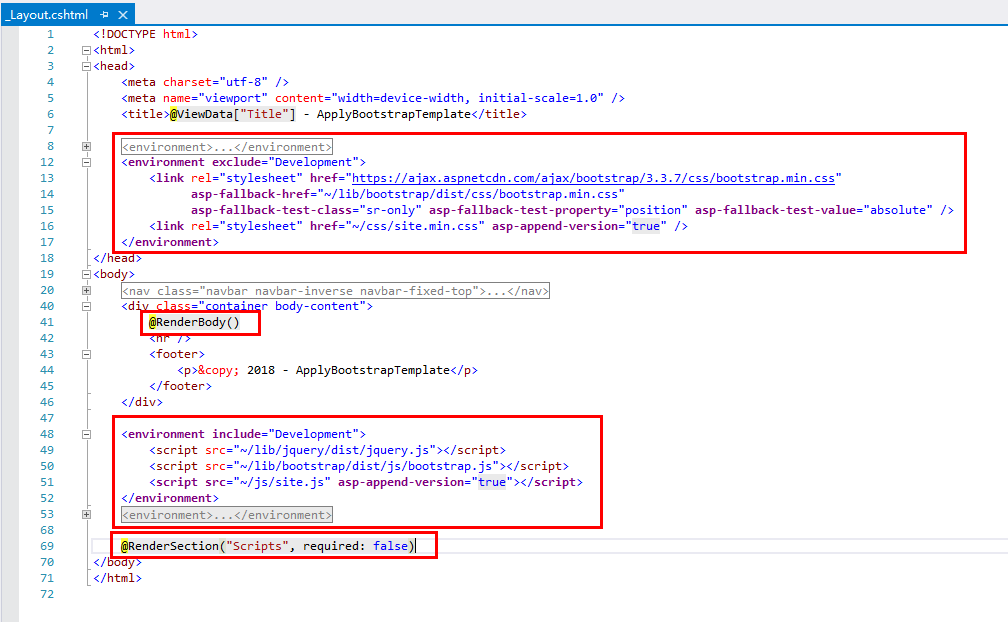
**第五步：**新增.bowerrc文件，配置包安装路径即可。这里我们指定为了wwwroot\plugins。（这里没有指定为wwwroot\bower\_components，与原始AdminLTE的目录结构保持一致，是因为如果指定为wwwroot\bower\_components，还原包后bower\_components默认不会包含在项目中。）

**第六步：**Restore Package，还原成功后，我们会发现plugins文件夹已包含显示在wwwroot目录下了。



**4、修改\_Layout.cshtml**

接下来我们将AdminLTE的预置起始页面starter.html移植进我们的布局页面\_Layout.cshtml。 我们先来观察一下我们默认的布局页。



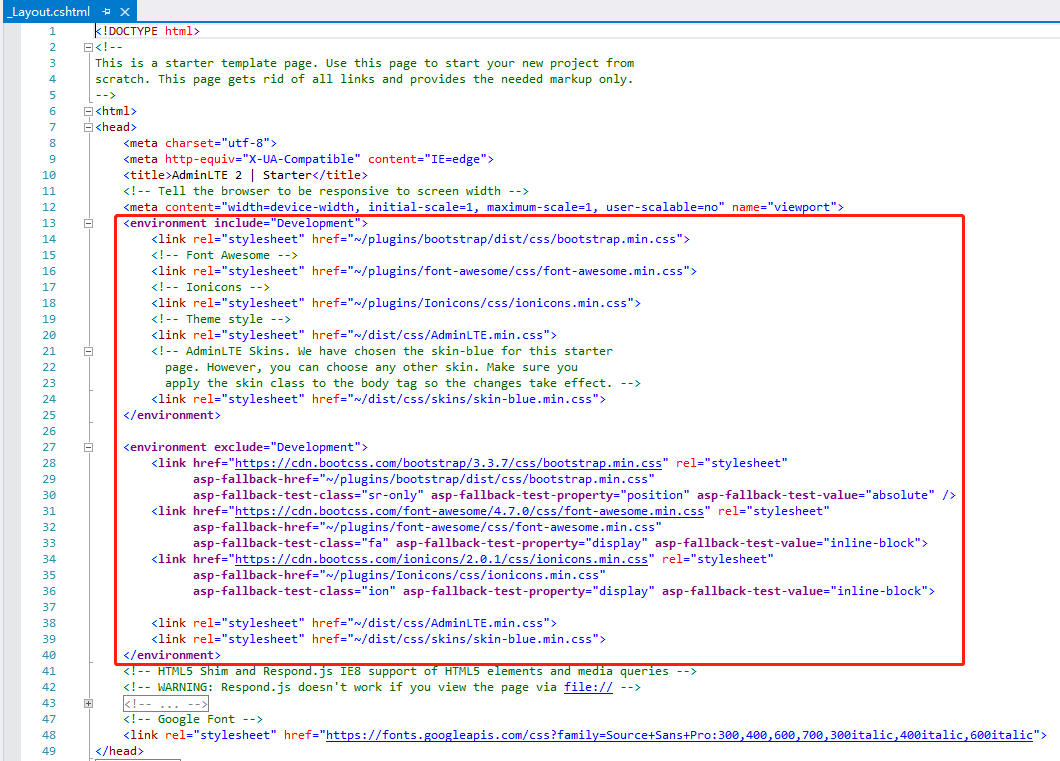
  主要有以上几个地方需要注意。

根据环境配置css和js的加载

@RenderBody()

@RenderSection("Scripts", required: false)

我们直接暴力复制starter.html的内容复制粘贴到\_Layout.cshtml，然后再将以上三个点进行修改即可。然后修改引用的css、js路径即可。修改后的截图如下：





二、给ASP.NET Core Web发布包做减法

**引言**

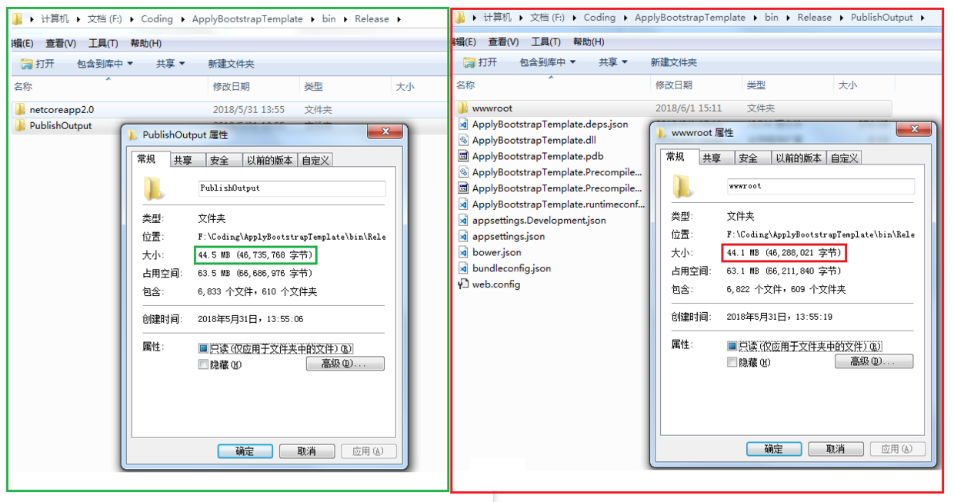
这一节我们来讲讲如何优化ASP.NET Core Web发布包繁重的问题。

在ASP.NET Core Web App中我们可以通过Bower或NPM来安装一些JS、CSS插件，来方便我们组织前端组件。但是这也给我带来了一个问题，那就是发布时需要把安装的Bower包或NPM包都要打包上传到服务器。如果现在发布ASP.NET Core Web App，wwwroot下已包含到项目中的文件都会被发布。虽然我们可以使用捆绑和微小的技术对js、css进行压缩来减少网页大小来提升加载速度。但是，我们发布包的大小却不能减少。

如果我们项目中引用了较少的前端包文件，也无可厚非。但当我们引用了较多的包文件时。那我们的发布包将会占用很大一部分空间。尤其是当我们进行CI/CD时，将会耗费大量的时间来进行包还原和包文件上传。

**2. 思路**

我们就以集成AdminLte的ASP.NET Core Mvc项目为例，看看发布的包大小究竟有多大。



从上图我们看到发布后wwwroot/plugins文件夹就占了很大一部分空间。而wwwroot/plugins中就是安装的Bower包。那这些Bower包中的文件我们都有用到吗？显然没有。我们就顶多引用了个js和css文件而已。到这里，减负的思路我们就清晰了。剔除ASP.NET Core Web中未引用的Bower包文件，把没有引用到的文件删除不就得了？！

但是你随便打开一个Bower包文件夹，你就不想这么做了，一个一个删要删到什么时候。而且如果直接去删除Bower包中无用的文件，可能会影响bower包的管理，比如bower包的升级降级。

**思路如下：**

新建一个文件夹，将引用的文件复制到另外的目录。（保持原bower包中的目录层级）

修改项目中的引用到新的文件夹拷贝路径下。

将原来的wwwroot\plugins 排除到项目外（Exclude From Project）

你可能会说，这么复杂啊，还不如我一个一个删除啊。别怕，我们让这一切自动化。而这个自动化工具就是[Gulp.js](https://www.gulpjs.com.cn/)。

**3. 行动**

全局安装 gulp： $ npm install --global gulp

作为项目的开发依赖（devDependencies）安装：

$ npm install --save-dev gulp

$ npm install --save-dev path

$ npm install --save-dev del

安装成功后会在项目根目录创建package-lock.json文件和node\_components文件夹。 3. 在项目根目录下创建一个名为 gulpfile.js 的文件。将以下代码粘贴复制进去。

const gulp = require('gulp');//1. 引用gulp

var path = require('path');//2. 引用path

var del = require('del');//3.引用del

//定义路径

const paths = {

src: 'wwwroot/plugins/',

dest: 'wwwroot/lib/'

};

//定义需要完整复制的Bower文件夹

const copyFolders = [

"bootstrap",

"font-awesome"

];

//定义项目中需要引用的bower包中的js、css文件

const copyFiles = [

"Ionicons/css/ionicons.css",

"jquery/dist/jquery.min.js",

"bootstrap/dist/js/bootstrap.min.js"

];

//在复制之前先清空生成目录

gulp.task('clean:all', function (cb) {

del([paths.dest], cb);

});

//复制文件

gulp.task('copy:file', () => {

//循环遍历文件列表

var tasks = copyFiles.map(function (file) {

//拼接文件完整路径

var scrFullPath = path.join(`${paths.src}`, file);

//拼接完整目标路径

var index = file.lastIndexOf('/');

var destPath = file.substring(0, index);

var destFullPath=path.join(`${paths.dest}`, destPath);

return gulp.src(scrFullPath).pipe(gulp.dest(destFullPath));

});

});

//复制文件夹

gulp.task('copy:folder', () => {

var tasks = copyFolders.map(function (folder) {

//拼接完整目标路径

var destFullPath = path.join(`${paths.dest}`, folder);

return gulp.src(path.join(`${paths.src}`, folder + '/\*\*/\*'))

.pipe(gulp.dest(destFullPath));

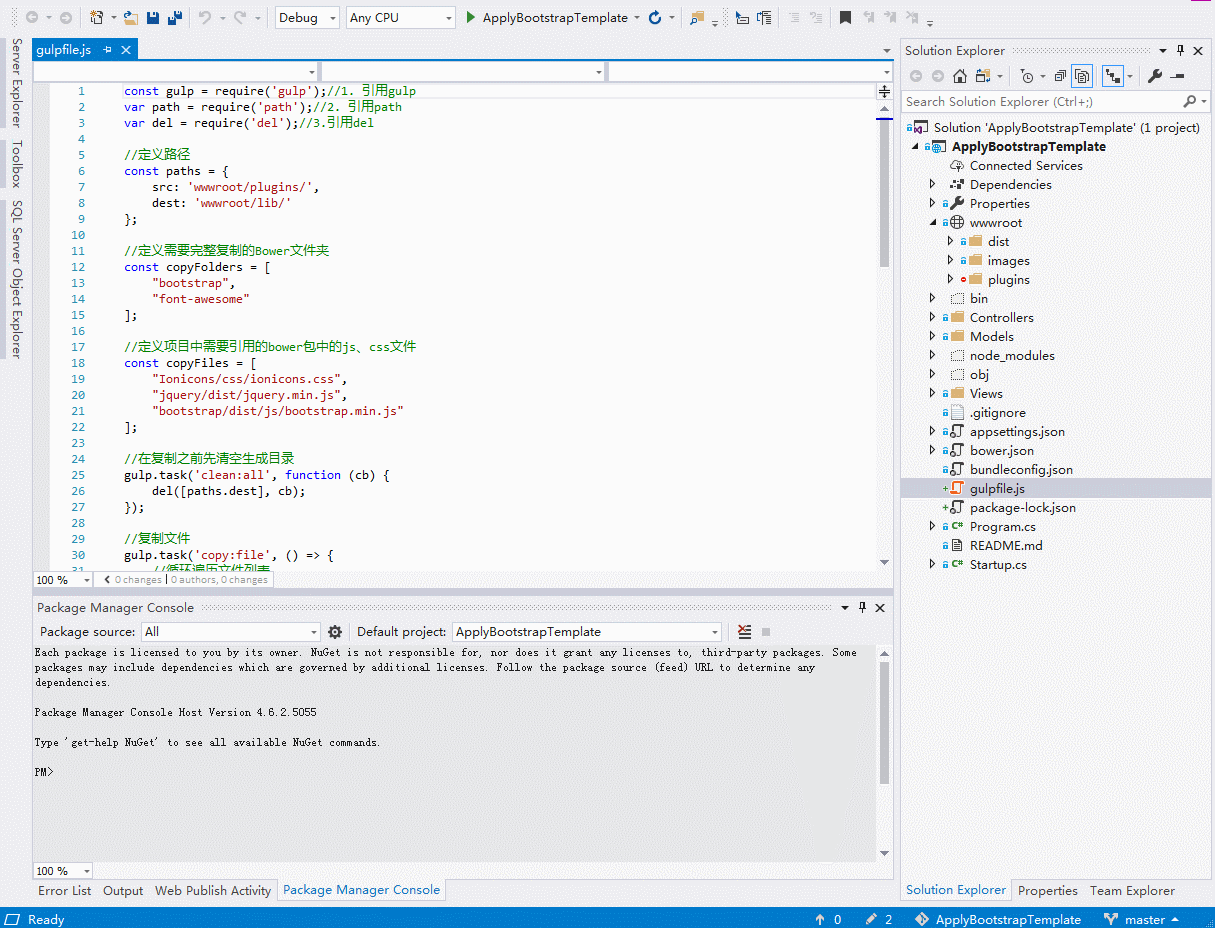
});

});

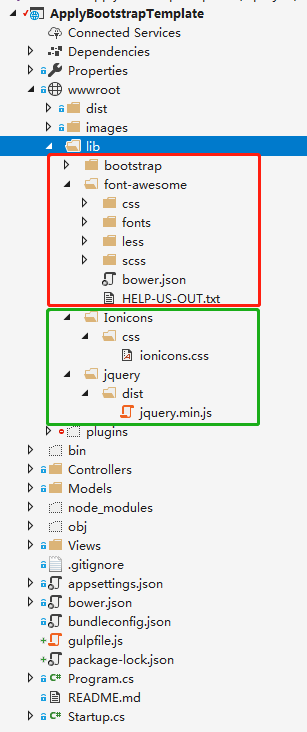
//将三个任务组装在一起

gulp.task('default', ['clean:all', 'copy:file', 'copy:folder']);

代码注释的很详细，就不过多赘述了。有一点需要解释下，为什么需要完整拷贝bootstrap和font-awesome呢？因为引用的font-awesome.min.css会引用包文件的一些字体文件等，为了省事，就把包全部拷贝了一遍。而一般绝大多数包都是简单拷贝css和js文件就ok了的。而至于什么时候拷贝文件，什么时候文件夹。很简单，默认先拷贝文件，运行项目，然后浏览器F12，如果发现有无法加载的error，那就是了。 4. 运行gulp 右键gulpfile.js-->Task Runner Exploerer-->双击Gulpfile.js-Tasks-default，即可运行。操作动图如下：



运行后，需要复制的Bower包文件和文件夹就会复制到wwwroot\lib文件夹下。如图：



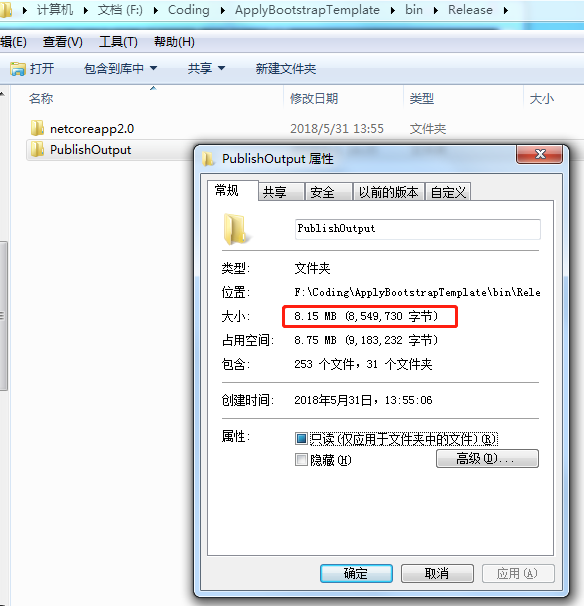
将bower包安装文件夹排除到项目外。

更新项目中现有文件的引用到lib目录下。

That's all, thank you.

**4. 效果**

重新发布，我们可以发现发布的包大小已有40M减小到8M。



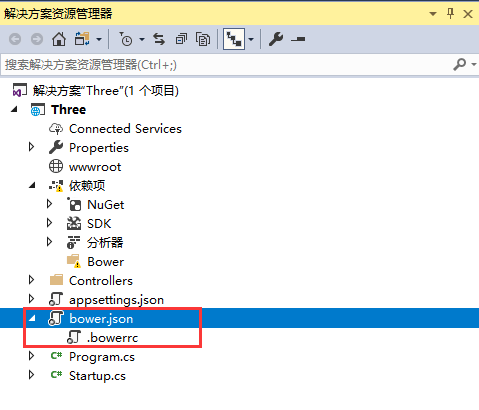
三、**Bower包引用**

1、添加bower两个配置文件，内容见下方。（这里有个问题，我vs更新到最新版本后，无法通过新建项添加，以前是可以的。大家想办法加就是了）

2、添加配置文件后，项目右键 -> 管理Bower程序包，进入管理界面。

3、选择要安装的组件，选择版本，安装。

4、安装后，查看安装结果。



**1. bower.json 文件内容**

{

"name": "asp.net",

"private": true,

"dependencies": {

}

}

**2. .bowerrc 文件内容**

{

"directory": "wwwroot/lib"

}

