微软 DevOps
动手实验手册六
单元测试和 Selenium 自动化界面测试

Contents

1. 概述

在这个实验中，你将学习到如何使用 Visual Studio 2015 创建默认 MVC 网站项目与单元测试。并且手动创建基于 Selenium 框架的自动化界面测试

1.1. 必要条件

- **开发工具 IDE**
  Visual Studio 2012 或者更高版本。

1.2. 虚拟场景

本次培训中的系列动手实验，都是围绕一个虚拟的研发场景展开的，即某企业随着业务的拓展，需要开发一个包含企业产品展示，订单提交、资讯管理等功能的企业门户。

根据这一业务场景，我们组建研发团队，模拟研发过程的不同阶段，包括项目计划、需求管理，代码开发、构建管理、发布管理、软件监控等，并且模拟开发人员在不同阶段使用的不同的开发工具完成特定的工作。

本实验是为了让大家了解在 VS 中添加，运行单元测试，并使用 Selenium 创建基于单元测试的自动化界面测试脚本。我们可以使用这些自动化测试验证网站功能，并且在发布过程中运行自动化功能测试。

1.3. 实验内容

本次实验包含下面的几个练习：

1. 使用 Visual Studio 创建默认 MVC 网站项目与单元测试项目，并运行单元测试。
2. 创建 Selenium 自动化测试项目，编写脚本，运行测试。

1.4. 实验时间

预计完成本次实验需要耗时 30 分钟。

2. 实验一：使用 Visual Studio 创建默认 MVC 网站项目与单元测试项目，并运行单元测试

在这个练习中，你将学会如何使用 Visual Studio 创建默认 MVC 项目并同时创建默认单元测试项目。并在已经创建好的 WebApplication1 解决方案中运行单元测试

1. 打开 **Visual Studio**
2. 创建 **项目**

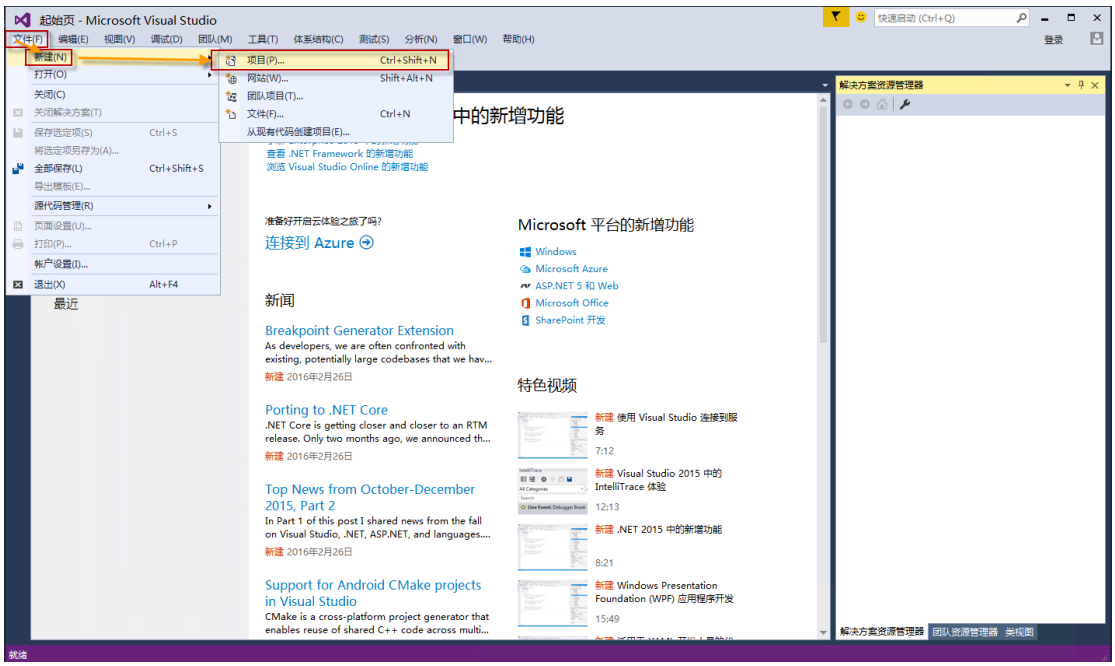

Figure 1 – VS **创建项目**

3. 选择 Web 项目模板中的 **ASP.Net Web 应用程序**。创建项目的文件地址为：[团队项目工作区映射根目录]\DemoProject\Dev\[姓名拼音]。

注： 由于网络原因，在创建项目时不添加 Application Insights。

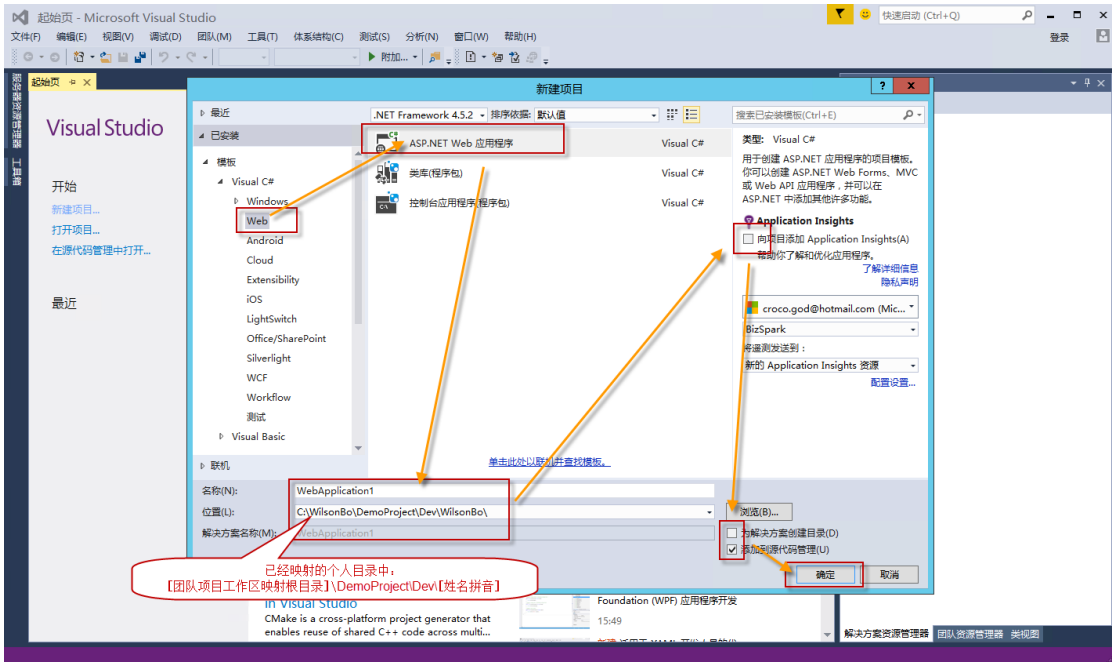

Figure 2 – VS **项目模板**

4. 选择 MVC 网站模板并 **添加单元测试**

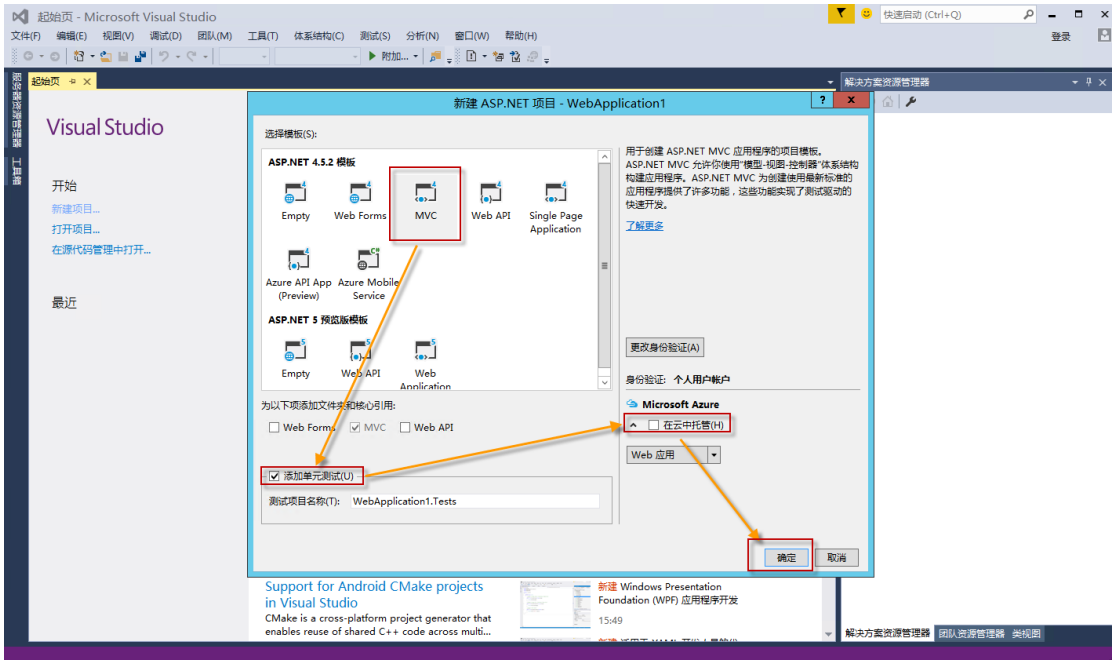

Figure 3 – VS 网站模板

5. 右键点击 **解决方案 WebApplication1 | 生成解决方案。**

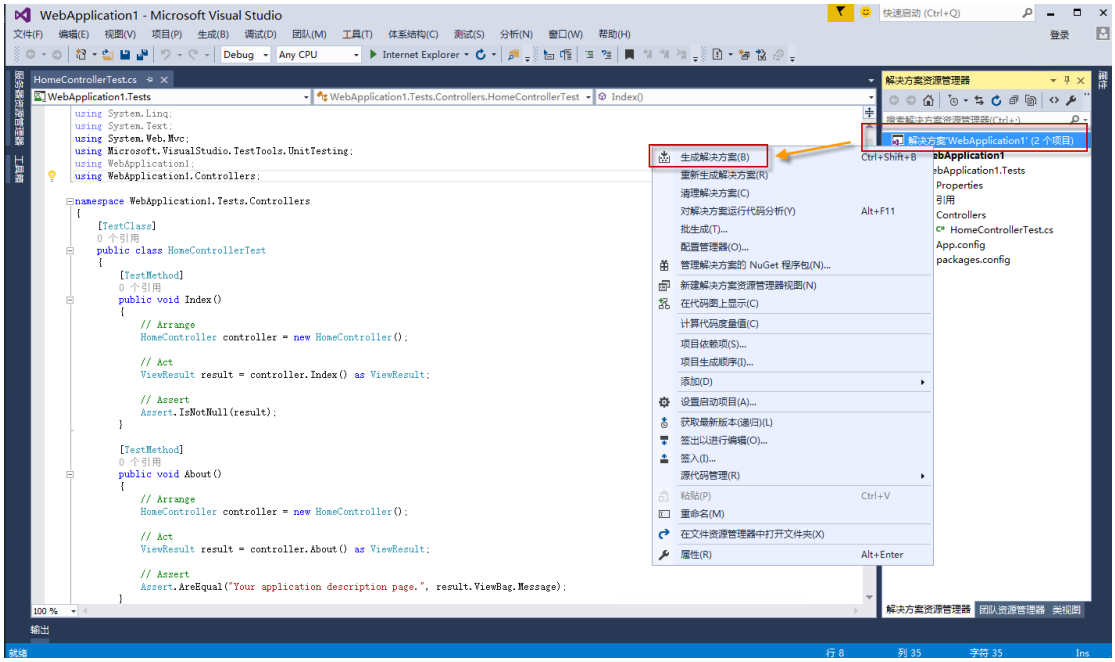

Figure 4 – **生成解决方案**

6. 打开 **测试资源管理器。**

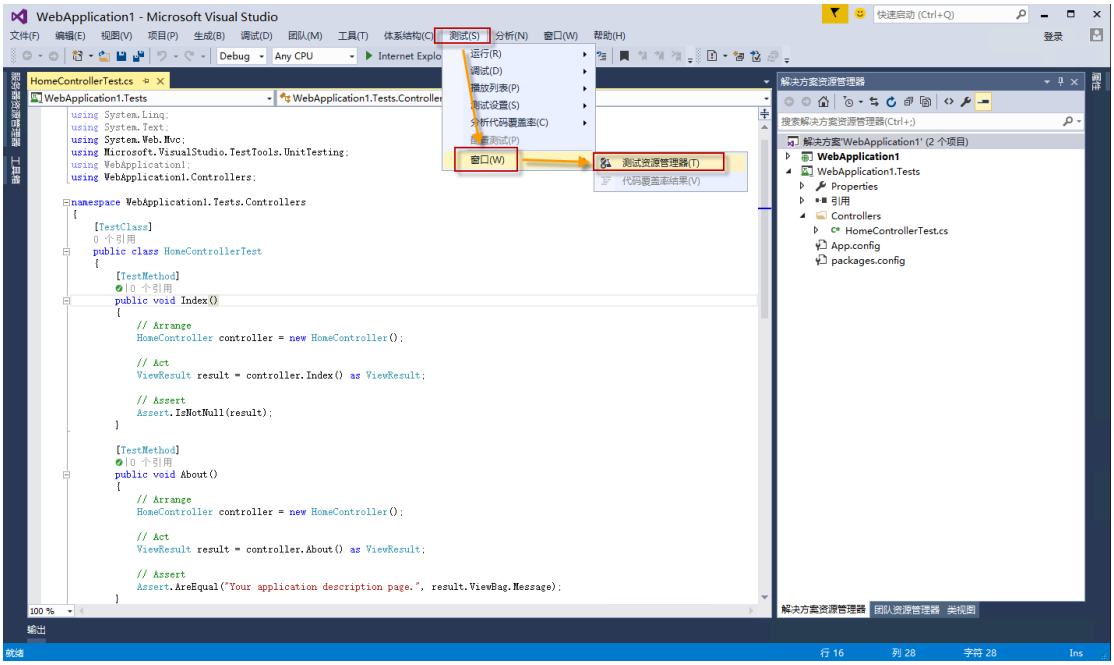

Figure 5 - **打开测试资源管理器**

7. 运行全部单元测试

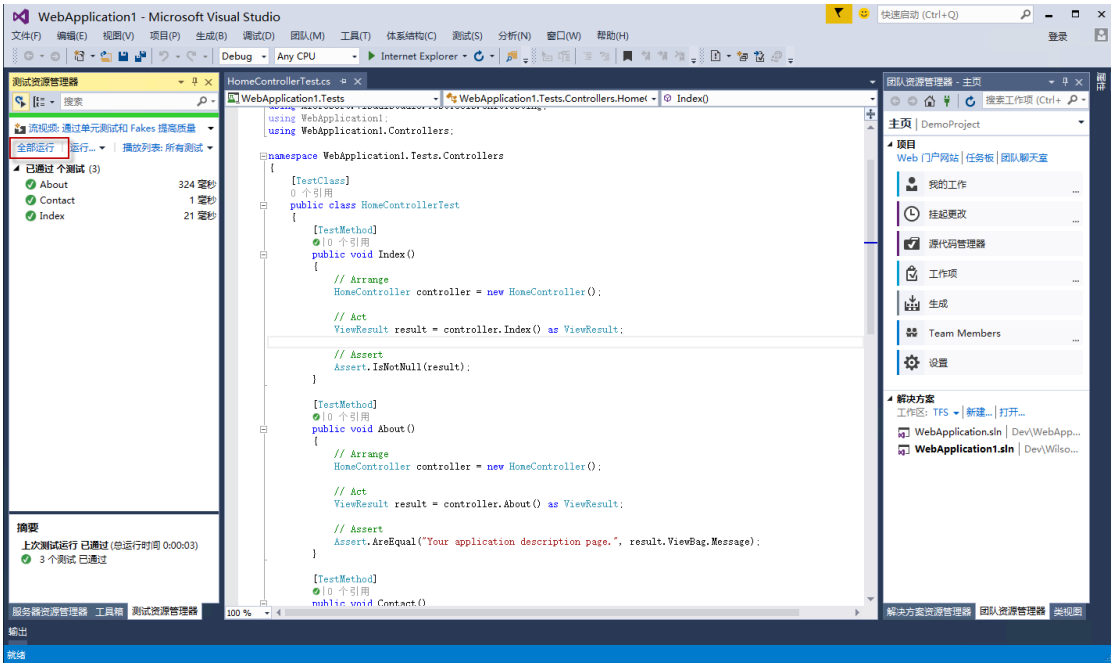

Figure 6- **运行单元测试**

3. 实验二 ：创建 Selenium 自动化测试项目，编写脚本，运行测试

在这个练习中，你将学会如何使用 Visual Studio 在已经创建好的 WebApplication1 中创建单元测试项目，并添加一个基于 Selenium 框架的自动化界面测试。基于默认网站添加单元测试方法，并执行单元测试。

1. 在当前解决方案中添加新的单元测试项目，命名为 **Selenium**。

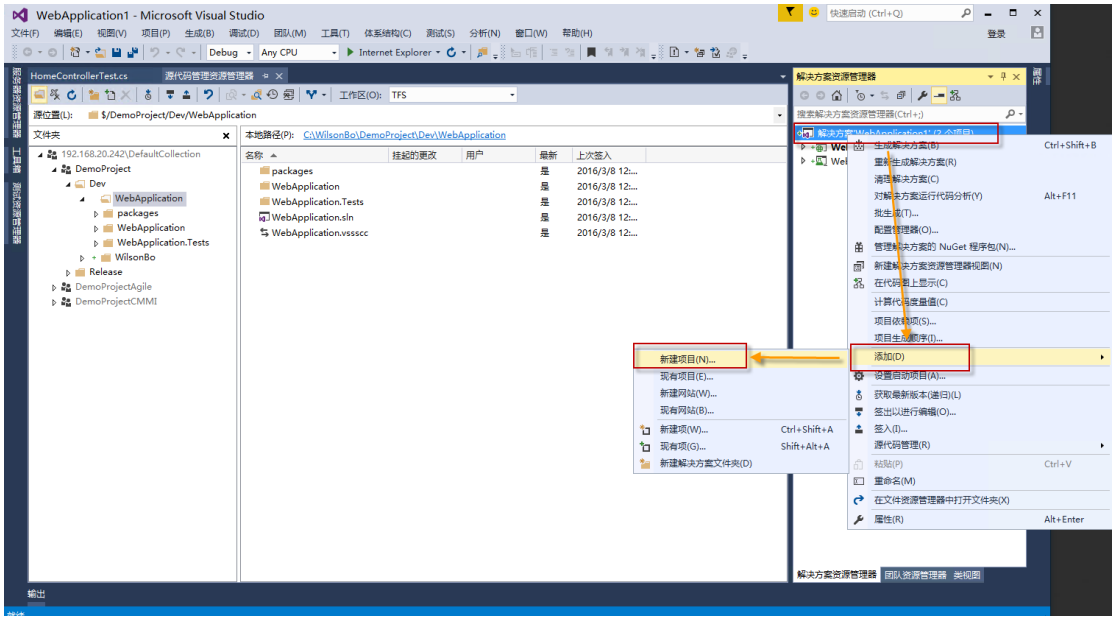

Figure 7 – 添加 VS 项目

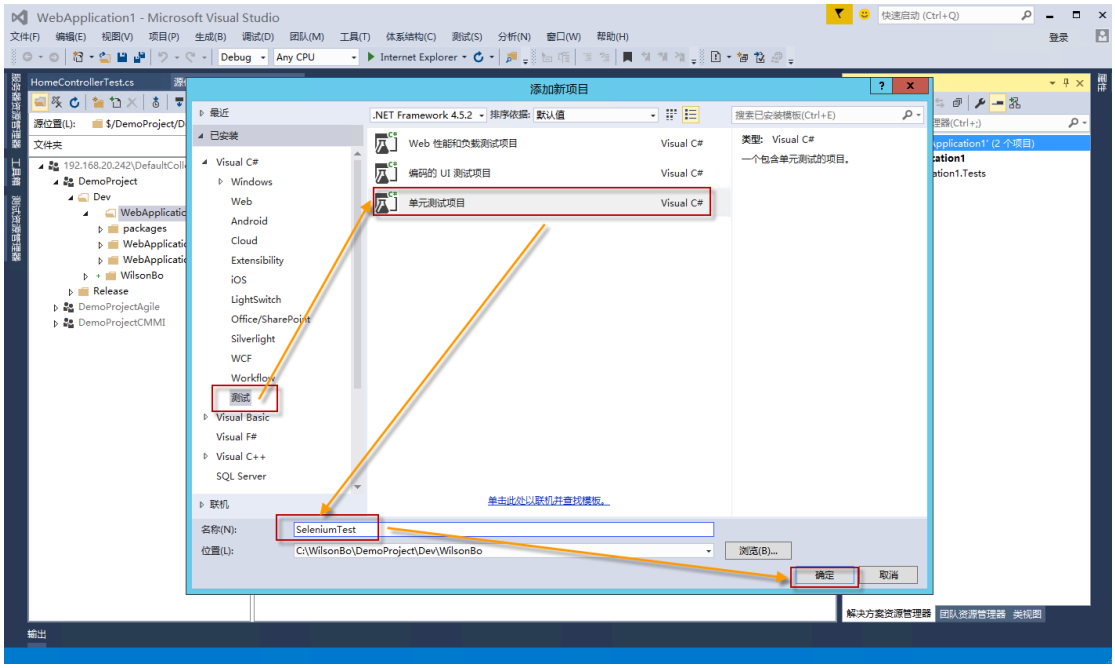

Figure 8 – 创建单元测试项目

2. 重命名单元测试文件 **UnitTest1.cs --> ContactTest.cs**

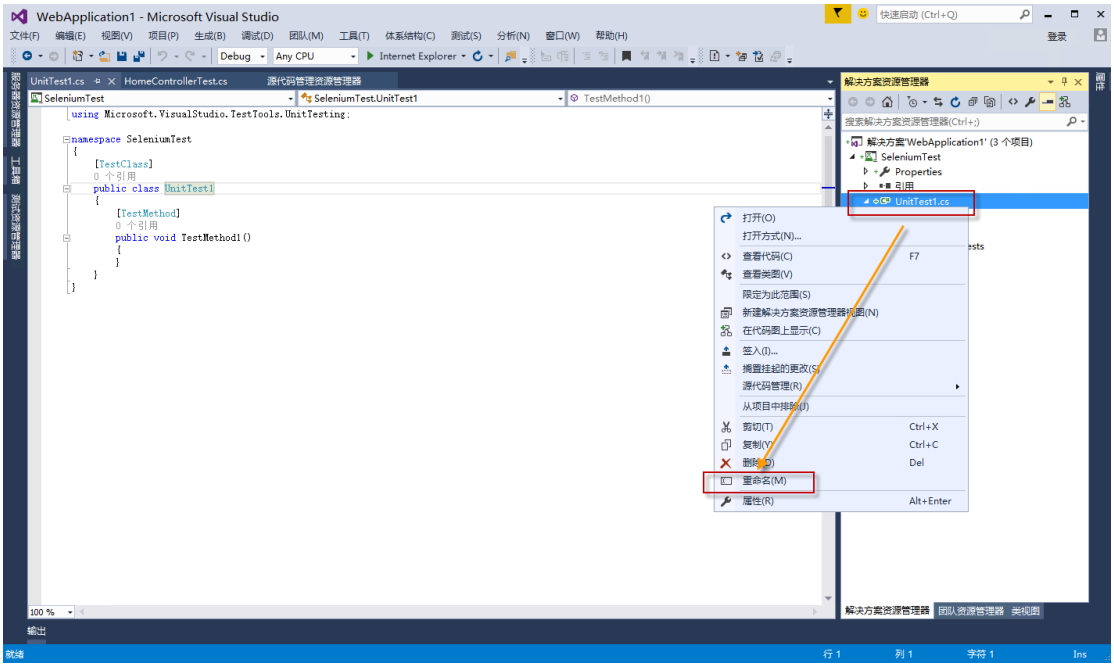

Figure 9 – **重命名文件**

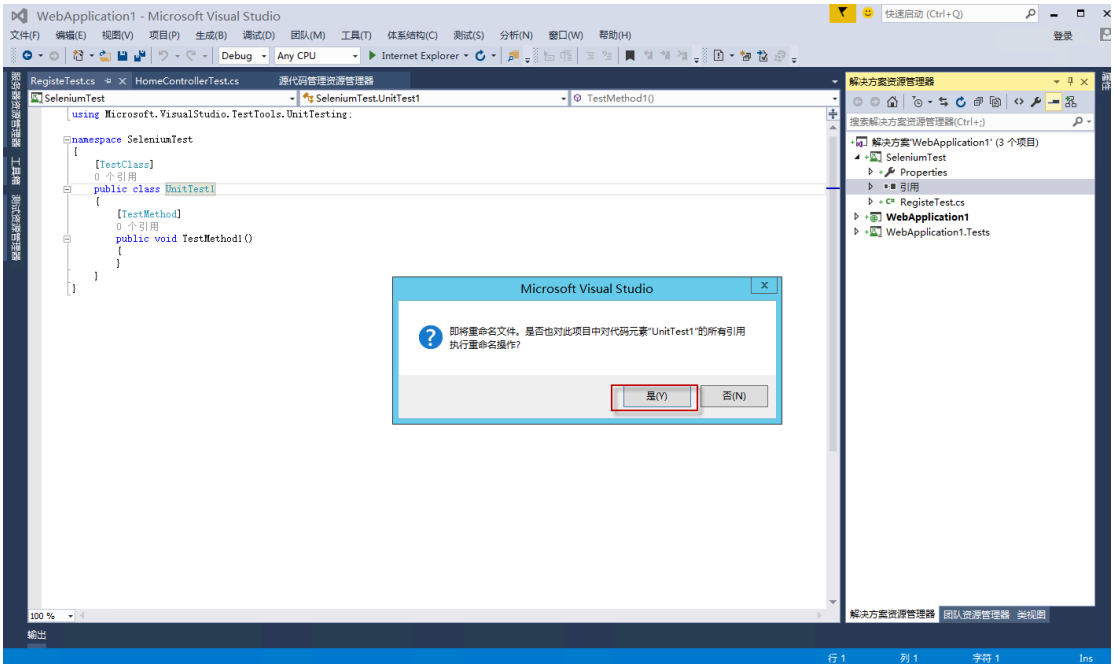

Figure 10 – **重命名单元测试类**

4. 从 NuGet 中安装 Selenium。(如果不能上网，请跳到操作 **5**)。

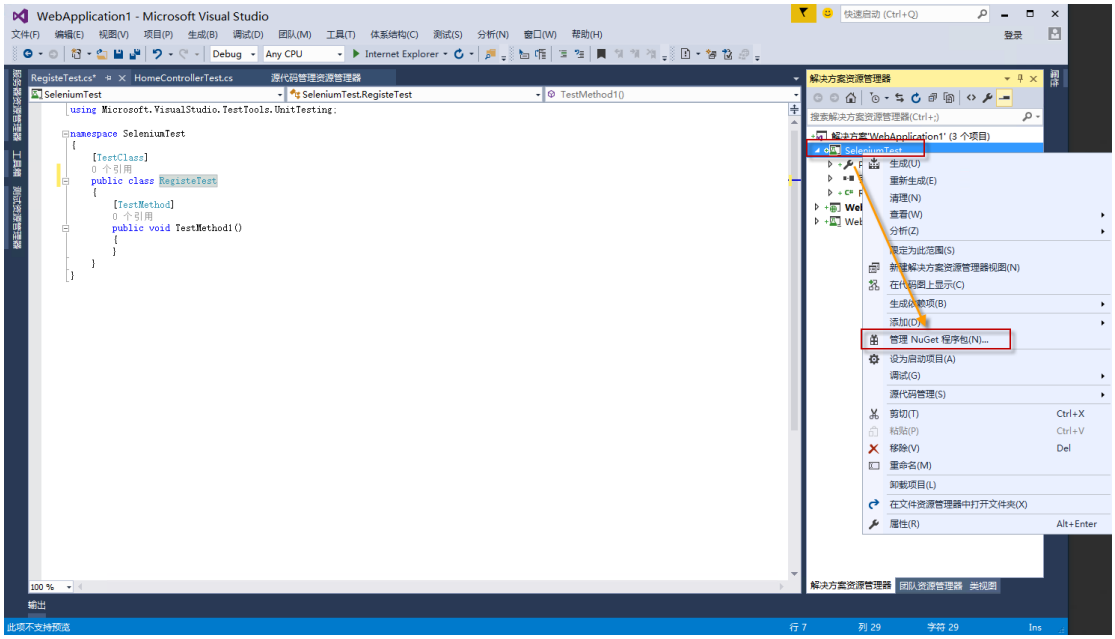

Figure 11 – **打开** NuGet **管理器**

安装程序包：

- Selenium.WebDriver
- Selenium.Support
- Selenium.WebDriver.ChromeDriver (可选，本地计算机安装 Chrome 可安装)
- Selenium.WebDriver.IEDriver (可选，本地计算机安装 32 位 IE 可安装)
- Selenium.WebDriver.IEDriver64 (可选，本地计算机安装 64 位 IE 可安装)

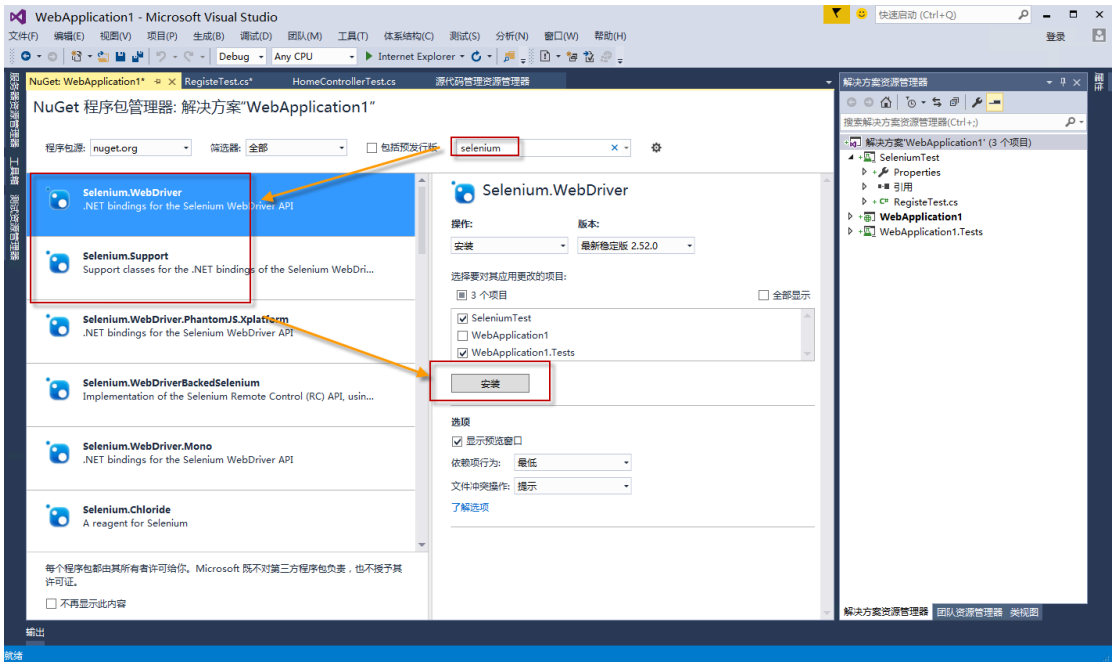

Figure 12 – **安装** NuGet **包**

5. 引用 Selenium 应用程序集 （如果完成操作 **4**， 请跳过本操作）。

将文件夹**$/DemoProject/Dev/Reference/Selenium** 及文件拷贝到 Dev 分支下的个人文件夹中。文件夹中包含 **Selenium 的应用程序集**及**浏览器驱动器**。(如果已经存在对应文件，请忽略这一步)

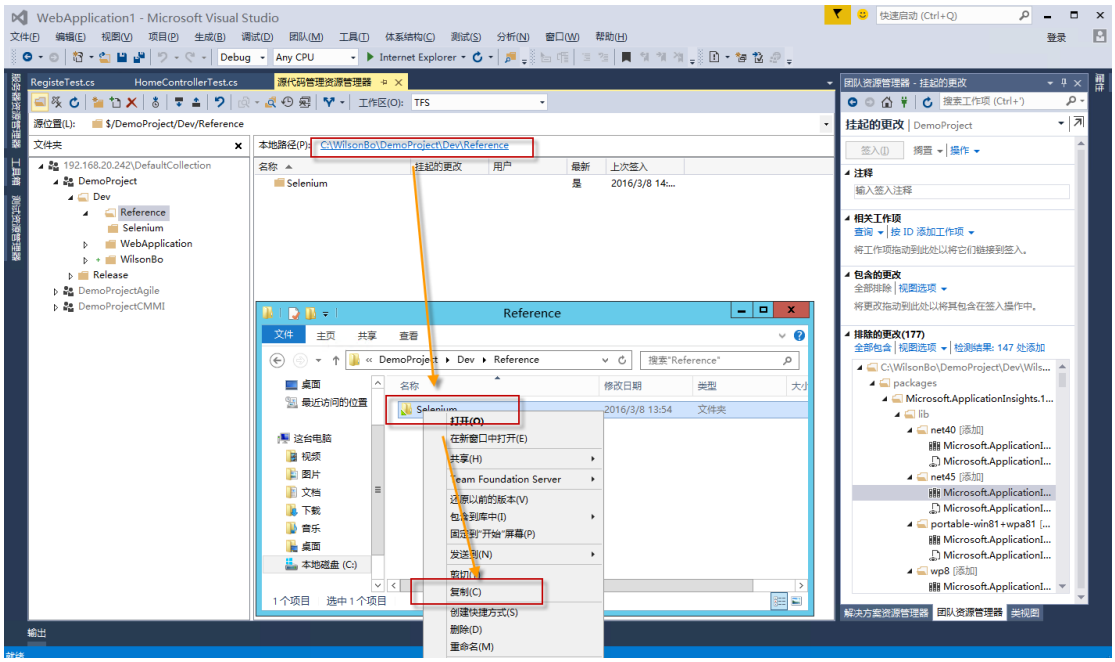

Figure 13 – **拷贝文件**

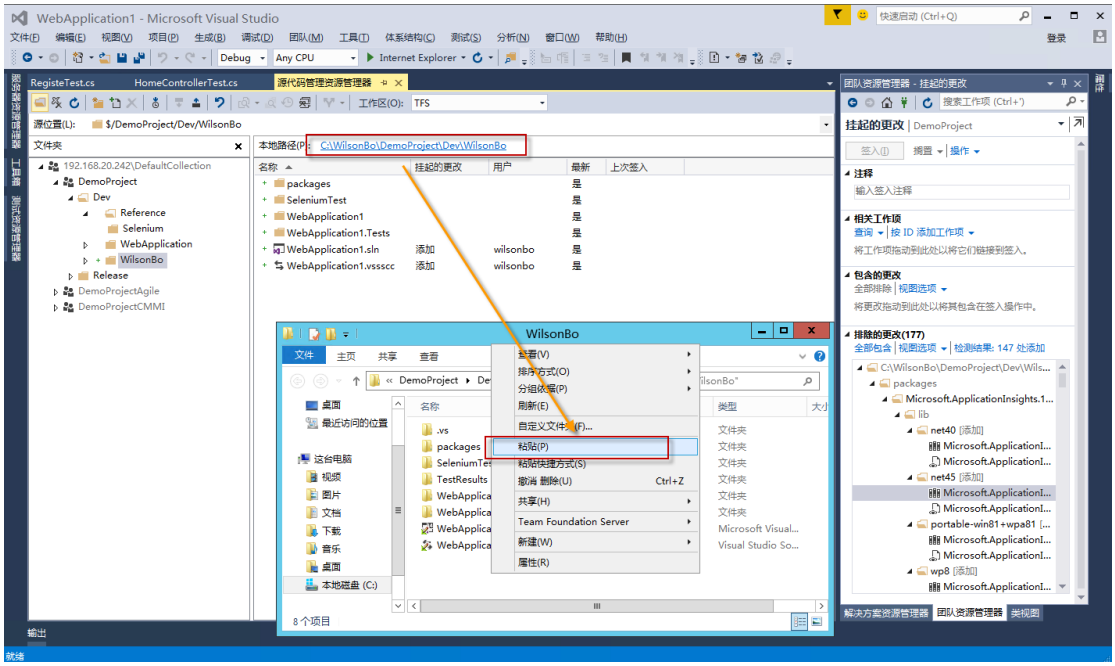

Figure 14 – **粘贴文件**

在 **Selenium** 项目中添加 **Selenium 应用程序集**的引用。

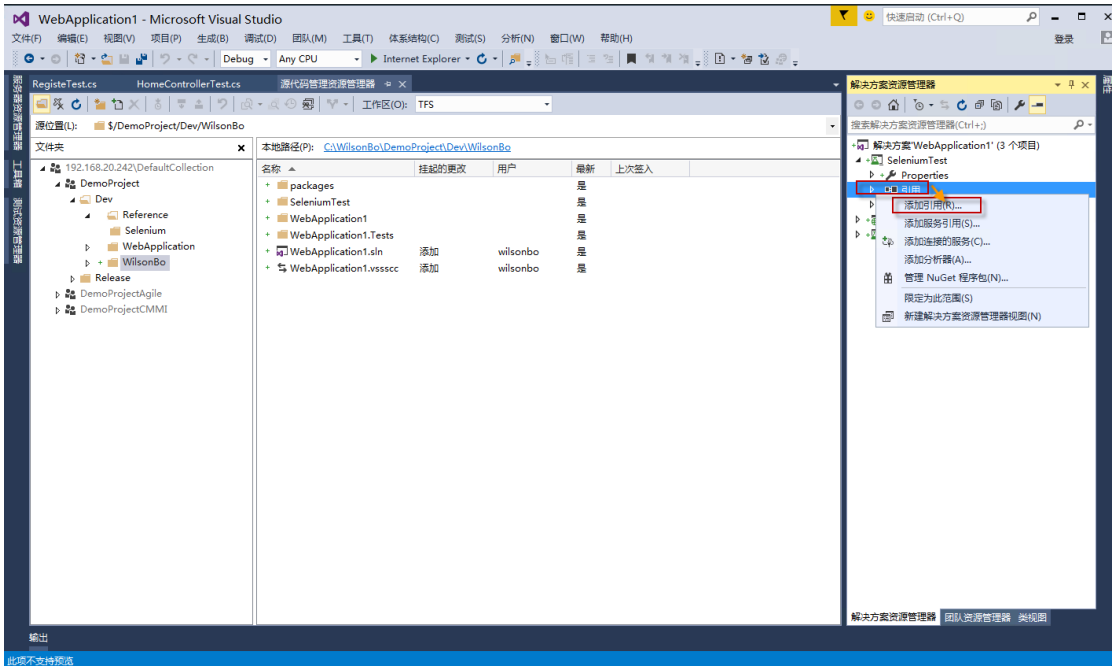

Figure 15 – **添加项目引用**

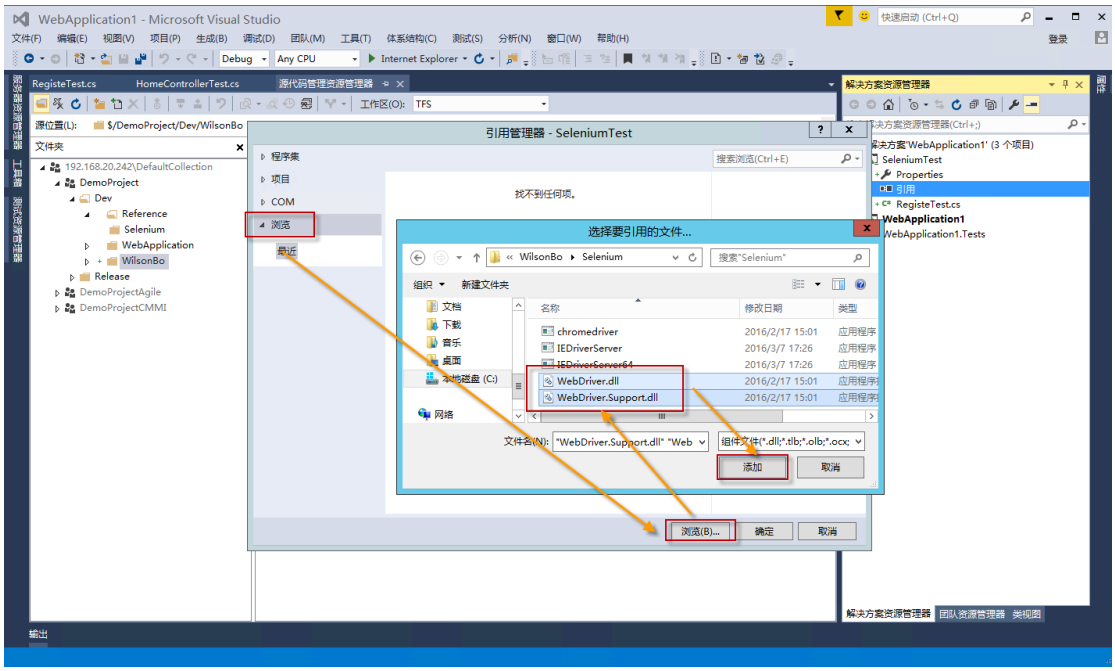

Figure 16 – **添加引用**

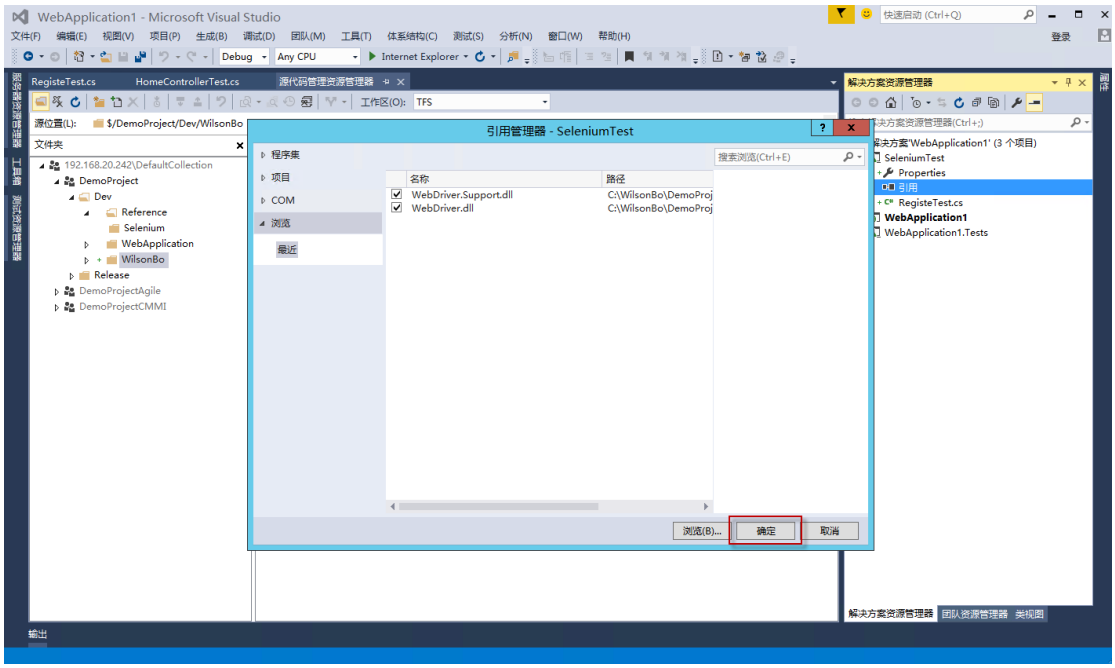

Figure 17 – **添加引用**

将**浏览器驱动器**添加到单元测试项目中

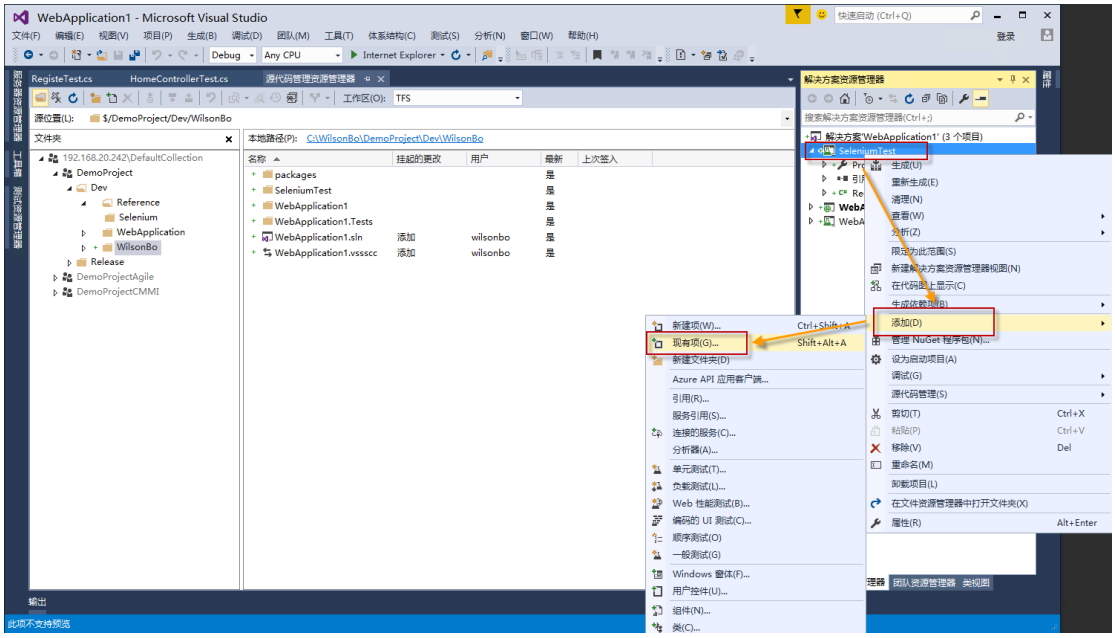

Figure 18 – **添加现有项**

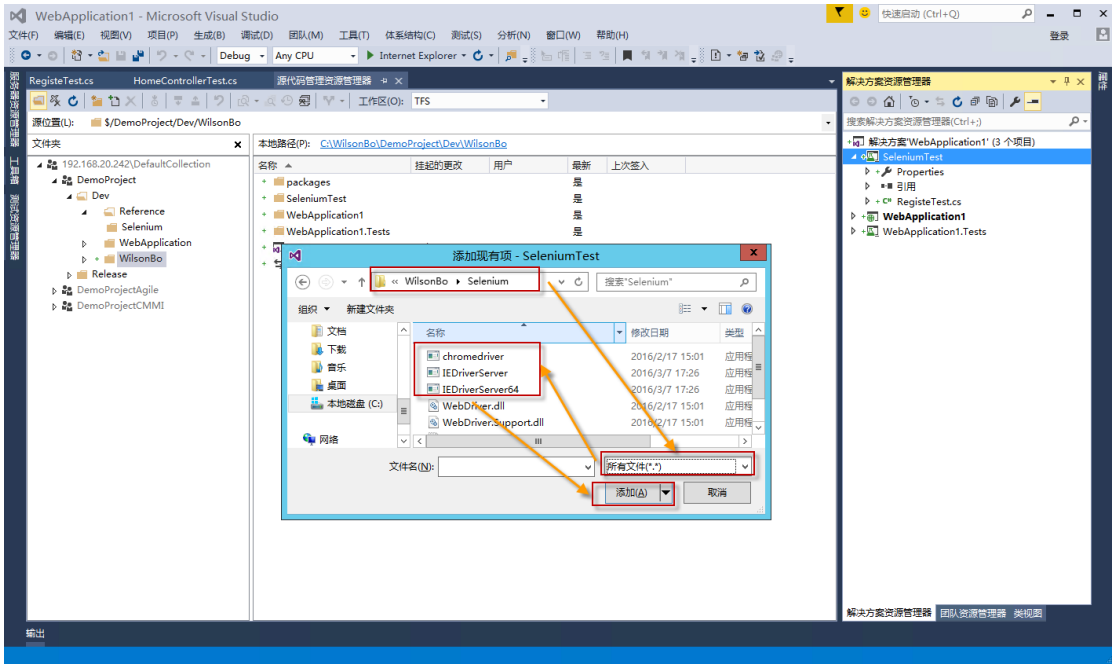

Figure 19 – **添加现有项**

修改文件属性

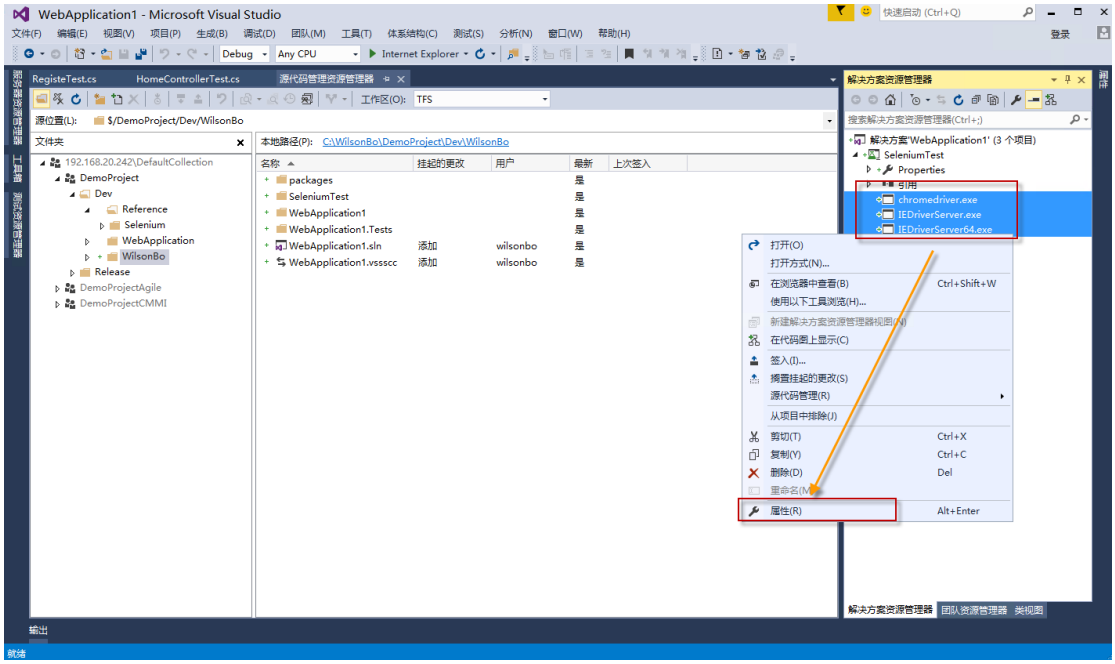

Figure 20 – **打开属性菜单**

设置文件属性 **复制到输出目录** 为 **始终复制**

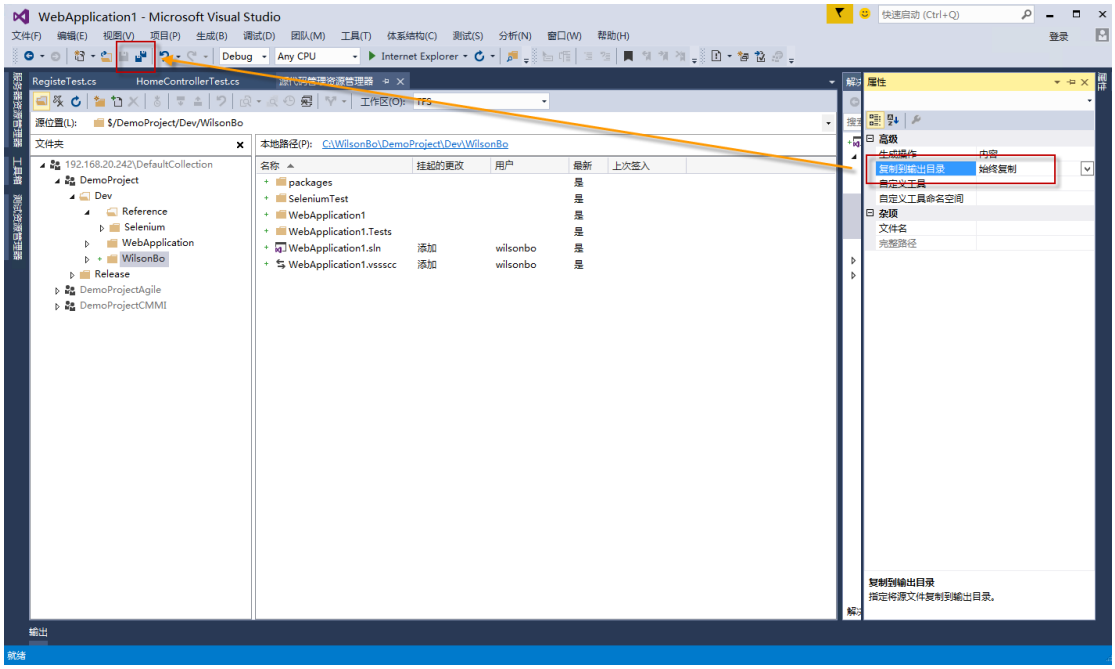

Figure 21 – **设置属性**

6. 修改单元测试文件 **ContactTest.cs**

添加 **引用**

```
using OpenQA.Selenium;
using System.Text;
using System.Threading;
using OpenQA.Selenium.Chrome;
using OpenQA.Selenium.IE;
using OpenQA.Selenium.Firefox;
```

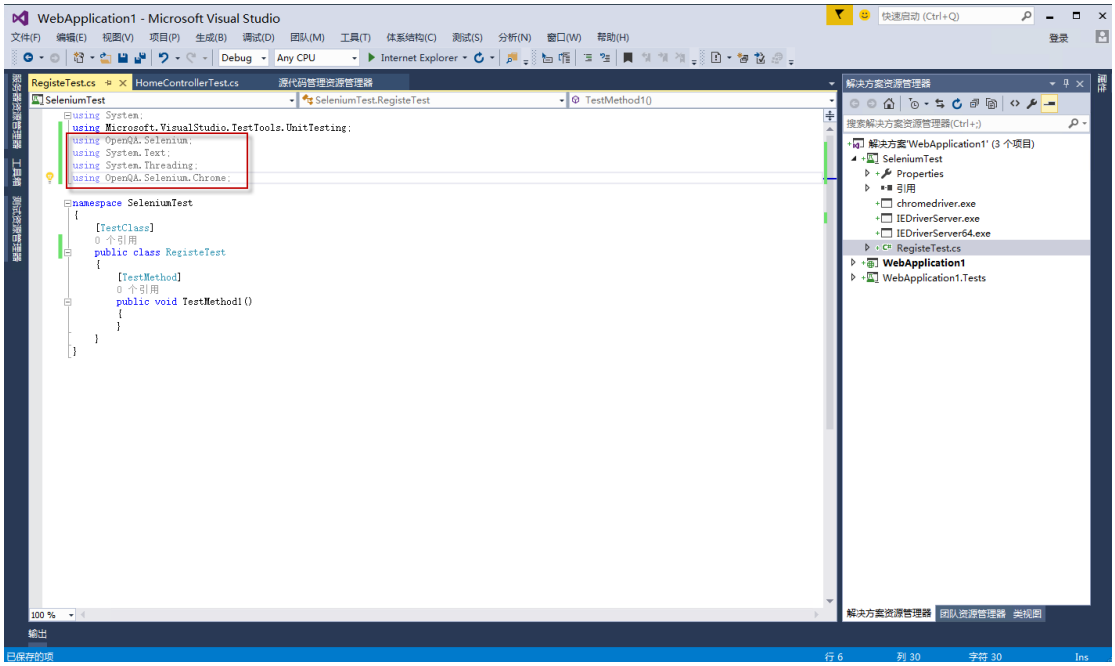

Figure 22 – 编辑源代码文件

添加变量

```
private IWebDriver driver;
private StringBuilder verificationErrors;
private string baseURL;
```

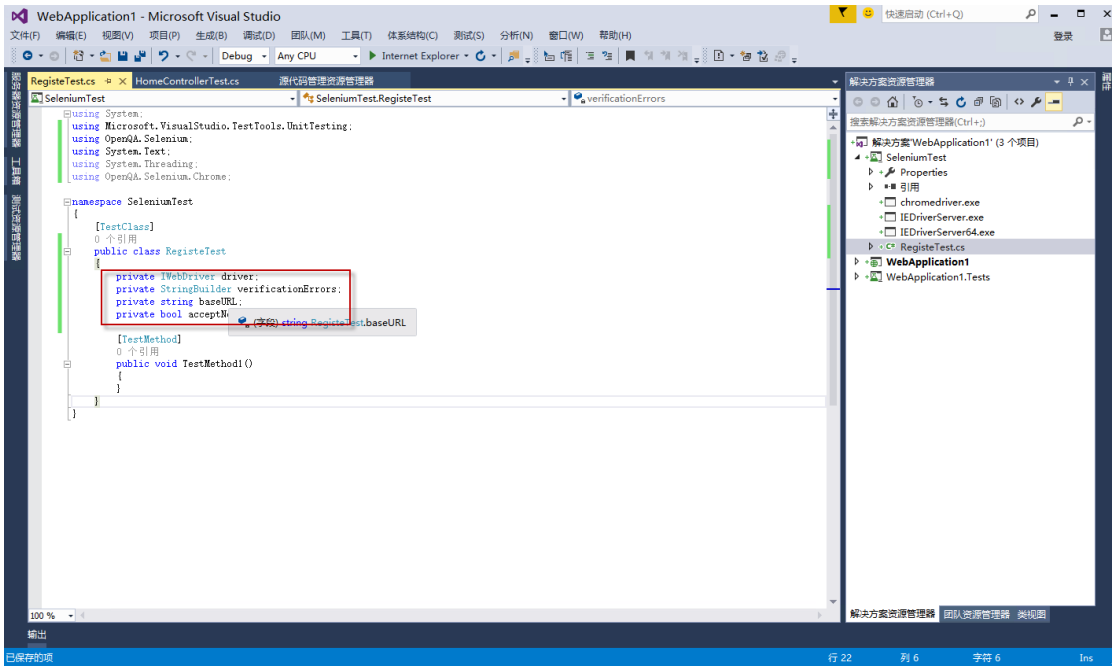

Figure 23 – 编辑源代码文件

添加测试方法

```
[TestMethod]
        public void Contact_Test()
        {
```

```
            driver.Manage().Window.Maximize();
              driver.Navigate().GoToUrl(baseURL);
              for (int second = 0; ; second++)
              {
                  if (second >= 60) Assert.Fail("timeout");
                  try
                  {
                      if (IsElementPresent(By.Id("contact"))) break;
                  }
                  catch (Exception)
                  { }
                  Thread.Sleep(1000);
              }
              driver.FindElement(By.Id("contact")).Click();
              try
              {
                  //assert
                  Assert.IsNotNull(driver.Title.ToLower().StartsWith("messa
ge"));
              }
              catch (Exception e)
              {
                  verificationErrors.Append(e.Message);
              }
      }

      [TestInitialize()]
      public void MyTestInitialize()
      {
          driver = new ChromeDriver();
          baseURL = “http://192.168.20.242:8000”;
          verificationErrors = new StringBuilder();
      }

      [TestCleanup()]
      public void MyTestCleanup()
      {
          try
          {
              driver.Quit();
          }
          catch (Exception)
          {
              // Ignore errors if unable to close the browser
```

```
        }
        Assert.AreEqual("", verificationErrors.ToString());
    }

    private bool IsElementPresent(By by)
    {
        try
        {
            driver.FindElement(by);
            return true;
        }
        catch (NoSuchElementException)
        {
            return false;
        }
    }
```

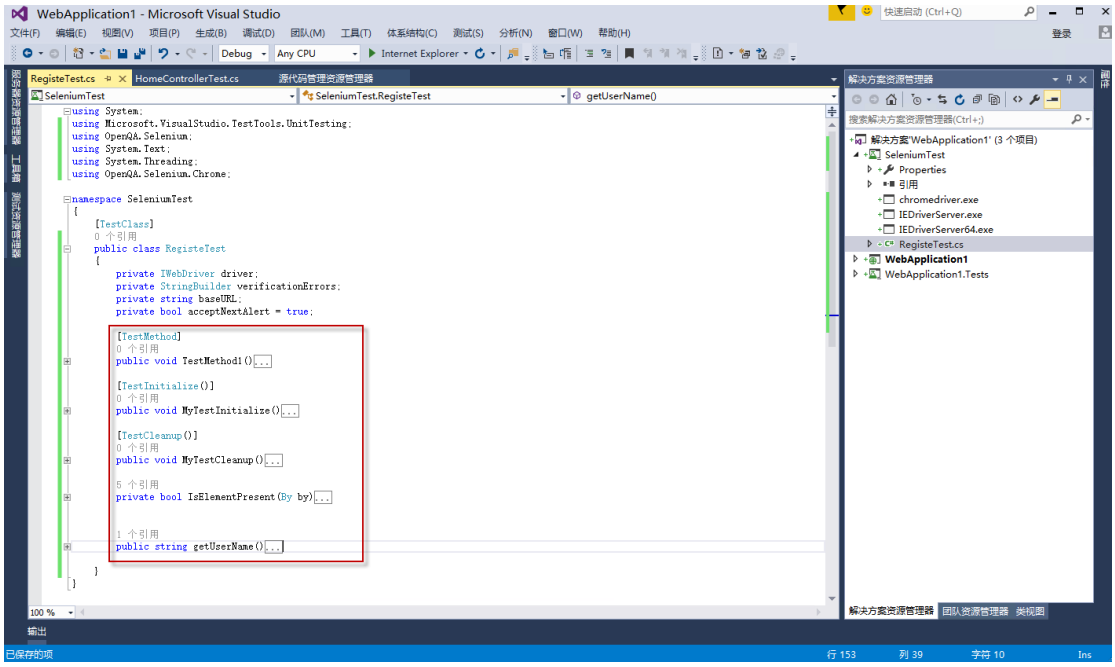

Figure 24 – **编辑源代码文件**

在 **MyTestInitialize** 方法中，修改代码：

`driver = new ChromeDriver();`	根据本地计算机安装的浏览器类型进行修改：
	`driver = new ChromeDriver();`
	`driver = new FirefoxDriver();`
	`driver = new InternetExplorerDriver();`

baseURL = “http://192.168.20.242:8000”	你的网站端口

7. 运行 Selenium 自动化界面测试

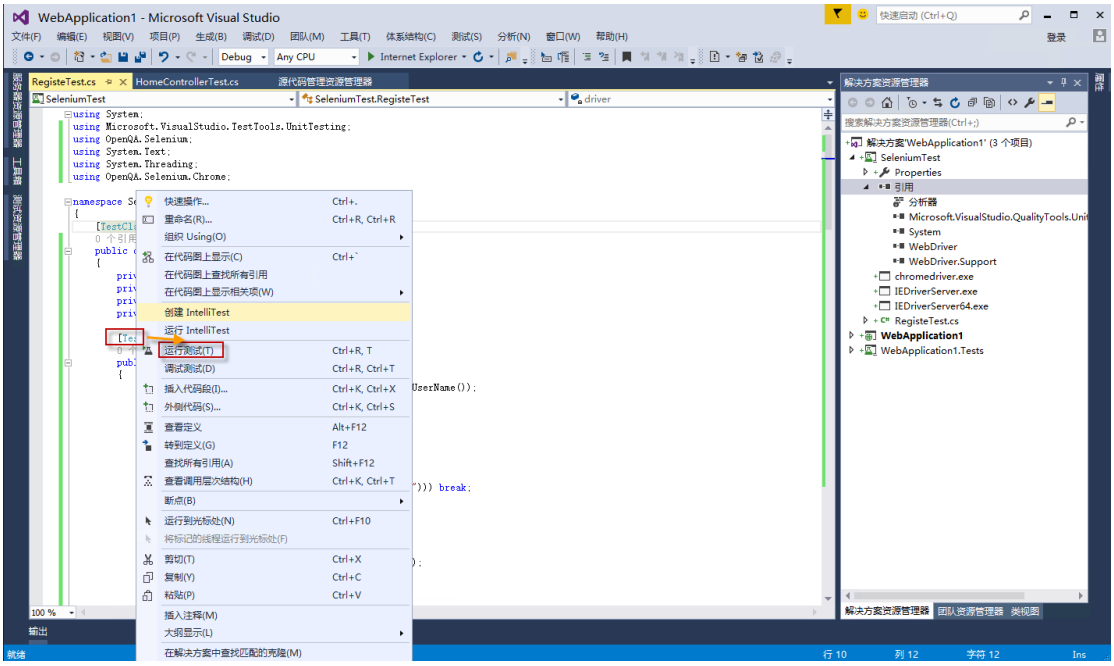

Figure 25 – **运行单元测试**