Creating Selenium based Automated UI Test

> **创建基于 Selenium 的自动化 UI 测试**

Refer to the following link as a fall back.

> 如有需要，请参考以下链接。

https://www.browserstack.com/guide/selenium-with-c-sharp-for-automated-test

Right Click OrgMgmt solution (not the project) ->Add New Project [select NUnit Test Project C#] as another project in the OrgMgmt solution. You can name it whatever you like.

> 在 OrgMgmt 解决方案上右键单击（不是项目）-> 添加 新项目 [选择 NUnit Test Project C#]，作为 OrgMgmt 解决方案中的另一个项目。你可以随意命名。

Add Selenium.WebDriver as well as Selenium.Support NuGet packages to the test project (both packages say “by Selenium” to be sure).

> 向测试项目中添加 Selenium.WebDriver 以及 Selenium.Support NuGet 包（确认两个包都标注为 “by Selenium”）。

Type chrome::/version on the browser to check the chrome driver version. For browsers such as FireFox or Safari refer to the above link or other online tutorials which will go through similar steps.

> 在浏览器中输入 chrome::/version 检查 Chrome 驱动版本。对于 FireFox 或 Safari 等浏览器，请参考上面链接或其他在线教程，它们会介绍类似的步骤。

Go to the following links to download the corresponding Chrome Driver version

> 前往以下链接下载相应版本的 Chrome 驱动.

https://googlechromelabs.github.io/chrome-for-testing/

If your chrome is even older then can try finding it here (you can consider upgrading your browser to the latest version but at your own risk 😊)

> 如果你的 Chrome 版本更旧，也可以尝试在此处查找（你也可以考虑将浏览器升级到最新版本，但请自行承担风险 😊）

:

https://developer.chrome.com/docs/chromedriver/downloads

create a “drivers” folder in the test project and add the chromedriver.exe (extracted from the downloaded zip file) into that folder.

> 在测试项目中创建一个 “drivers” 文件夹，并将 chromedriver.exe（从下载的压缩文件中解压）添加到该文件夹中。

UnitTest1.cs can look like as follows (url variable in the code should point to the url that shows up on the browser when you run OrgMgmt in VisualStudio):

> UnitTest1.cs 可以如下所示（代码中的 url 变量应指向在 Visual Studio 中运行 OrgMgmt 时浏览器中显示的 URL）：

```
using OpenQA.Selenium;
using OpenQA.Selenium.Chrome;
namespace TestProject1
{
public class Tests
    public class 测试
{
[SetUp]
public void Setup()
    public void 初始化()
{
}
[Test]
public void Test1()
{
string path = Directory.GetParent(Environment.CurrentDirectory).Parent.Parent.FullName;
ChromeDriver driver = new ChromeDriver(path + @"\drivers\");
string url = "http://localhost:5120";
//ChromeDriver driver = new ChromeDriver();
driver.Manage().Window.Maximize();
driver.Navigate().GoToUrl(url);
```

```
driver.Manage().Timeouts().ImplicitWait =
TimeSpan.FromSeconds(5);

driver.FindElement(By.LinkText("Clients")).Click();

driver.Manage().Timeouts().ImplicitWait =
TimeSpan.FromSeconds(5);

driver.FindElement(By.LinkText("Create New")).Click();

driver.Manage().Timeouts().ImplicitWait =
TimeSpan.FromSeconds(5);

driver.FindElement(By.Id("Name")).SendKeys("Marge Simpson");

driver.FindElement(By.Id("Address")).SendKeys("Springfield");

driver.FindElement(By.XPath("//Input[@type='submit']")).Click();

Assert.Pass();

}

[TearDownAttribute]

public void TearDown()

{

}

}

}
```

Click on the “Debug” tab on the top navigation bar of Visual Studio (it is between Build and Test tabs) and press Start without Debugging (make sure that there is no other browser window open with welcome page of your MVCSampleApp). Right click on the UnitTest1.cs and press run. You will see the above code automatically launching the Web App and creating a new client.

> 在 Visual Studio 顶部导航栏点击“调试”选项卡（它位于“生成”和“测试”选项卡之间），然后按“开始而不调试”（确保没有其他浏览器窗口打开着 MVCSampleApp 的欢迎页）。在 UnitTest1.cs 上右键并选择运行。你将看到上述代码自动启动 Web 应用并创建一个新客户端。