


工作在浏览器上人-YangBobin

知识不在广泛，在于精通。知识不在积累，在于消化。学习不在激情，在于坚持。书不在多，一两本真正看懂就行。书读百遍，其义自现。

随笔 - 892, 文章 - 1, 评论 - 68, 阅读 - 123万

目录
导航

导航

博客园
首页
订阅 
管理

搜索

我的标签

设计模式(24)
C#语言新特性(10)
DevExpress(9)
并行编程(9)
报表(5)
git(4)
Socket(4)
WCF(3)
ADO.NET(3)
VS Code(3)
更多

随笔分类 (938)

00 常用知识(5)
01 HTML(17)
02 CSS(24)
03 JavaScript(47)
04 TypeScript(24)
05 Vue(7)
06 JQuery(38)
08 Echarts(12)
09 BootStrap(33)
12 C#语言基础(130)
13 Visual Studio(40)
15 WinForm(38)
16 ASP.NET(26)
17 ASP.NET Core(19)
18 EF Core(10)
19 .NET设计模式(29)
20 三方控件(65)
21 Oracle(32)
22 SqlServer(55)
23 MongoDB(11)

C#爬虫 (03) : 使用Selenium

目录

- 一、介绍：
 - 1、安装Selenium：
- 二、等待
 - 1、隐式等待：ImplicitlyWait
 - 2、显示等待：WebDriverWait()
- 三、查找（定位对象）
- 四、获取页面元素和元素内容
 - 1.Title：标题
 - 2.Url：链接
 - 3.Text：元素的文本值
 - 4.Selected勾选情况、TagName标记名标、Enabled编辑状态、Displayed显示状态
 - 5.GetAttribute () 获取标签的属性
 - 6.弹出对话框的处理
- 五、操作元素对象WebElement
 - 1、模拟鼠标点击元素
 - 2、下拉列表框Select的操作
 - 2、执行JS
 - 3、页面导航
 - 4、拖拽操作(可以实现滑动验证码的验证)
 - 5、模拟鼠标晃动
 - 6、截图功能
 - 7、selenium操作滚轮滑动到底部
- 六、获得窗口标识WindowHandles
 - 1、关闭多个子Browser窗口
 - 2、对iframe中元素的定位

30 ABP(54)
31 Python(91)
40 GitHub项目(1)
50 Office(19)
51 SoftWare分享(43)
53 Linux系统(24)
55 非技术(22)
88 微信开发(12)
99 English Learning(10)

随笔档案 (891)

2022年5月(22)
2022年4月(2)
2022年2月(5)
2021年11月(4)
2021年9月(1)
2021年8月(2)
2021年7月(9)
2021年6月(3)
2021年5月(2)
2021年4月(1)
2021年3月(2)
2021年1月(1)
2020年12月(6)
2020年11月(8)
2020年10月(61)
2020年9月(22)
2020年8月(6)
2020年7月(39)
2020年6月(80)
2020年5月(54)
2020年4月(9)
2020年3月(40)
2020年2月(27)
2020年1月(55)
2019年12月(32)
2019年11月(33)
2019年10月(30)
2019年9月(17)
2019年8月(27)
2019年7月(8)
2019年6月(5)
2019年3月(25)
2019年2月(18)
2019年1月(24)
2018年12月(14)
2018年11月(5)
2018年10月(23)
2018年9月(9)
2018年8月(118)

- 七、Cookies
- 八、Window窗口控制
- 九、事件
- 十、关闭浏览器

一、介绍:

Selenium 是一个用于Web应用程序测试的工具。Selenium测试直接运行在浏览器中,就像真正的用户在操作一样。

1、Selenium WebDriver (也就是Selenium2, Selenium3) 和Selenium RC (Selenium 1) 一样提供了web自动化的各种语言调用接口库。相比Selenium RC, Selenium WebDriver的编程接口更加直观易懂,也更加简练。

但是和Selenium RC不同的是, Selenium WebDriver是通过各种浏览器的驱动 (web driver) 来驱动浏览器的,而不是通过注入JavaScript的方式。

我们的代码运行起来是一个进程,里面调用Selenium WebDriver的库和各个浏览器的驱动进程 进行交互,传递Selenium命令 给它们,并且获取命令执行的结果,返回给我们的代码进行处理。

2、Selenium WebDriver目前包括两个版本Selenium 2和Selenium 3。这两个版本从开发代码调用接口上来看,几乎没什么区别。区别在于库的实现和web driver的实现。

Selenium2是Selenium组织帮各种浏览器写web driver的,而Selenium 3里面的web driver是由各个浏览器厂商 (Apple,Google,Microsoft,Mozilla) 自己提供的。所以Selenium 3的自动化效率更高,成功率也更高。

3、Selenium WebDriver 支持浏览器众多:

- Google Chrome
- Microsoft Internet Explorer 7, 8, 9, 10, 11在 Windows Vista, Windows 7, Windows 8, Windows 8.1.
- Microsoft Edge
- Firefox
- Safari
- Opera

利用它可以驱动浏览器执行特定的动作,如点击、下拉等操作,同时还可以获取浏览器当前呈现的页面的源代码,做到可见即可爬。

所以Selenium现在被广泛用于Python爬虫。查了下资料,发现这个工具确实强大,最重要的是,C#也是可以调用的。

官方支持Java, C#, Python, Ruby, PHP, Perl, Javascript等语言

官方文档 (有C#示例) : <https://www.selenium.dev/documentation/en/>

1、安装Selenium:

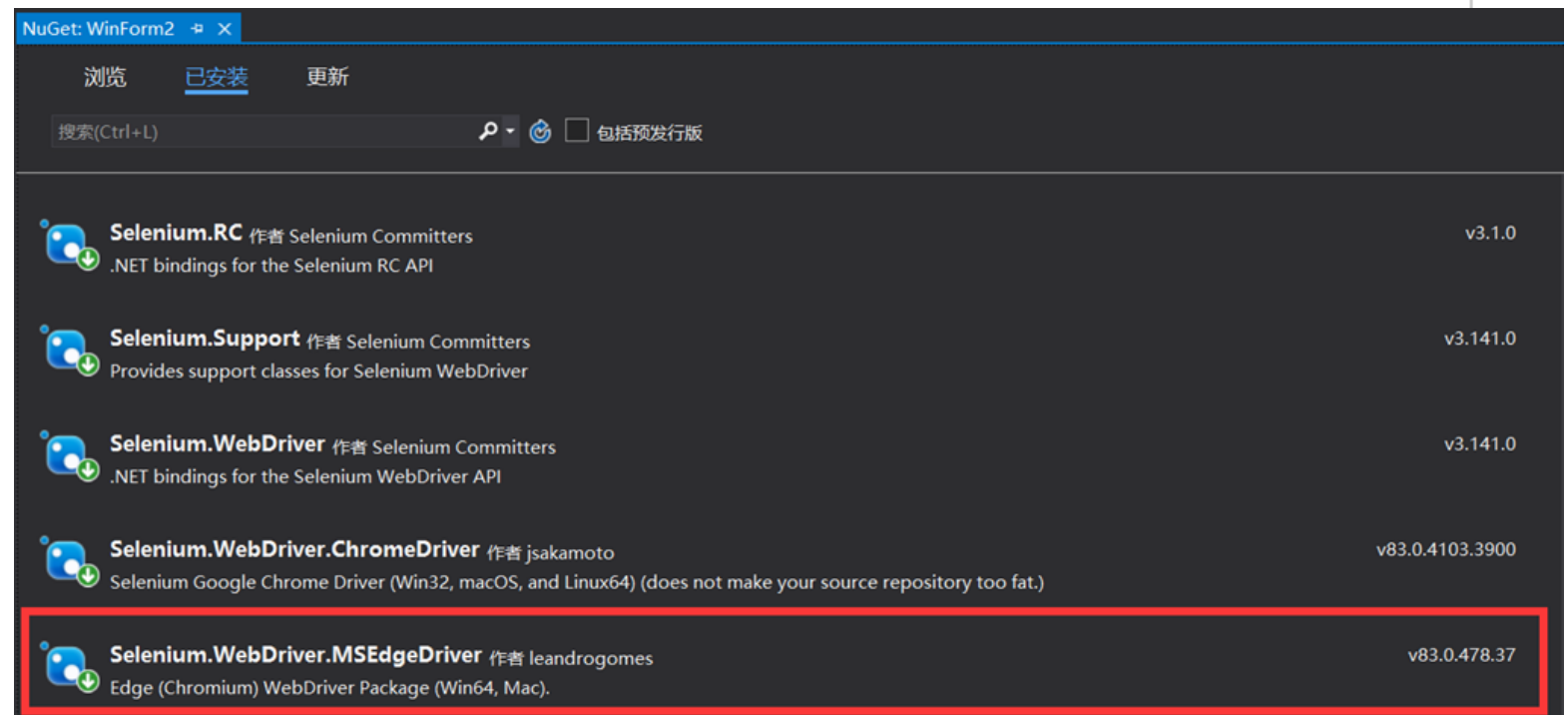
1、我们新建一个C#控制台程序

2、使用Nuget搜索以下依赖库

需要引用的核心库是Selenium.RC, Selenium.Support, Selenium.WebDriver

然后再需要引用浏览器驱动库, 这里我以新版Edge浏览器为例, 新版Edge使用方式跟Chrome是一样的, 程序包名称为

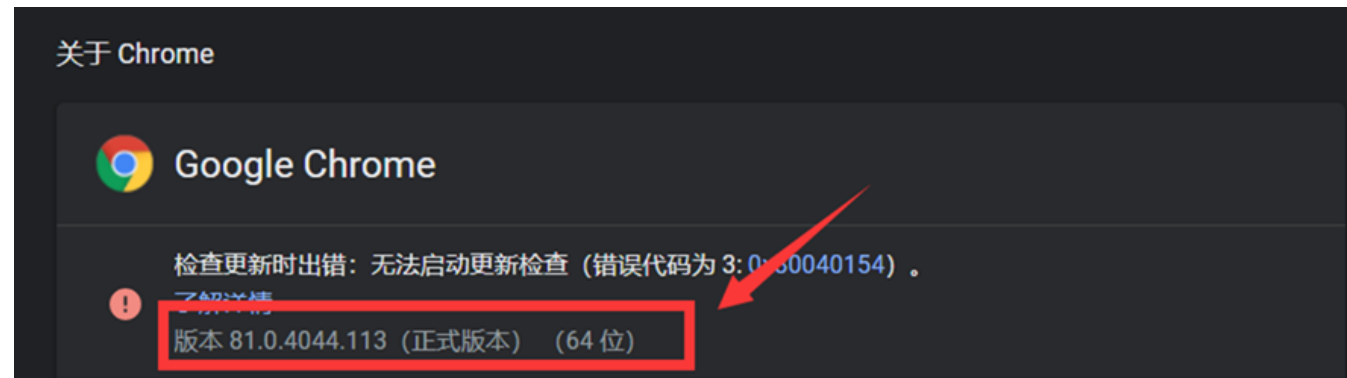
Selenium.WebDriver.MSEdgeDriver。



备注: 也可以在微软WebDriver官网下载Edge (Chromium)的webdriver, 需要和当前浏览器版本一致。然后下载放置到项目可执行文件的目录。

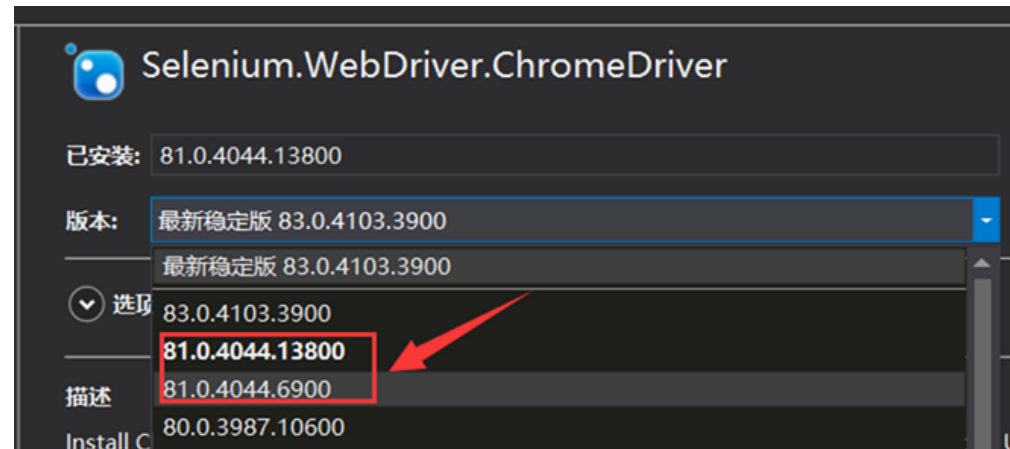
如果使用Chrome:

先查下本机Chrome的版本



然后去Nuget搜索Selenium.WebDriver.ChromeDriver进行下载安装。

注意: webdriver版本只需要和当前浏览器主版本一致即可。



3、在Main函数中输入以下代码

```
using OpenQA.Selenium;
using OpenQA.Selenium.Edge;
using System;
using System.Windows.Forms;

namespace WinForm2
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
    }
}
```

```
private void Form1_Load(object sender, EventArgs e)
{
    var service = EdgeDriverService.CreateDefaultService(@".", "msedgedriver.exe");
    using (IWebDriver driver = new OpenQA.Selenium.Edge.EdgeDriver(service))
    {
        driver.Navigate().GoToUrl("http://www.baidu.com"); //driver.Url = "http://www.baidu.com"是一样的
        var source = driver.PageSource;
        this.textBox1.Text = source;
    }
}
```

如果是Chrome浏览器，可以这样：

```
IWebDriver driver = new OpenQA.Selenium.Chrome.ChromeDriver();
driver.Navigate().GoToUrl("http://www.baidu.com");
```

运行，会弹出IE浏览器，网页加载完成后，浏览器会自动关闭。控制台输入结果如下



这样我们就可以轻松的获取动态渲染页面的源码。

二、等待

常用的等待分为**显示等待**`WebDriverWait()`、**隐式等待**`ImplicitlyWait()`、**强制等待**`sleep()`三种，下面我们就分别介绍一下这三种等待的区别

1. `Sleep()`: 强制等待，设置固定休眠时间。
2. `ImplicitlyWait()`: 隐式等待，也叫智能等待，是 `webdriver` 提供的一个超时等待。隐的等待一个元素被发现，或一个命令完成。如果超出了设置时间的则抛出异常。
3. `WebDriverWait()`: 显示等待，同样也是 `webdriver` 提供的方法。在设置时间内，默认每隔一段时间检测一次当前页面元素是否存在，如果超过设置时间检测不到则抛出异常。默认检测频率为0.5s，默认抛出异常为：`NoSuchElementException`

1、隐式等待: `ImplicitlyWait`

用到`Timeouts`对象。这个对象是用来对设置器进行一些设置的。

- `ImplicitlyWait`: 设置脚步在查找（定位）元素时最大的超时时间。如`FindElement()`方法在一些超大网页中进行定位时的超时时间。
- `PageLoad`: 设置页面操作超时时间（不是页面加载时间）。是在页面进行跳转操作或刷新操作时的等待时间。如`Navigation`对象的各种操作方法，以及在页面上进行某种操作后的等待时间。
- `AsynchronousJavaScript`: 设置脚步异步执行的超时时间。

代码如下:

```
driver.Navigate().GoToUrl("http://www.baidu.com");
ITimeouts timeouts = driver.Manage().Timeouts();

//设置查找元素最大超时时间为30秒
timeouts.ImplicitWait = new TimeSpan(0, 0, 30);
//设置页面操作最大超时时间为30秒
timeouts.PageLoad = new TimeSpan(0, 0, 30);
//设置脚本异步最大超时时间为30秒
timeouts.AsynchronousJavaScript = new TimeSpan(0, 0, 30);
```

2、显示等待: `WebDriverWait()`

```
//等待页面元素加载完成
//默认等待100秒
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(100));
//等待页面上ID属性值为submitButton的元素加载完成
IWebElement myElement = wait.Until((d) =>
{
    return d.FindElement(By.Id("submitButton"));
});
```

三、查找 (定位对象)

通过FindElement()这个方法来找的。然后把参数传递过去。

```
driver.FindElement(By.Id ("kw")).SendKeys("搜索关键字");  
driver.FindElement(By.Id( "su")).Click();
```

其中By.id("su")就是定位参数,传递一个对象过去。有8种定位方式。传递方式如下图:

```
... public static By ClassName( string classNameToFind );  
... public static By CssSelector( string cssSelectorToFind );  
... public static By Id( string idToFind );  
... public static By LinkText( string linkTextToFind );  
... public static By Name( string nameToFind );  
... public static By PartialLinkText( string partialLinkTextToFind );  
... public static By TagName( string tagNameToFind );  
... public static By XPath( string xpathToFind );
```

注意: 其中PartialLinkText是模糊查找。比如百度网页中的关于 参数写“关”就可以了,不用写*这种符号。

```
//通过ID获取元素  
var byID = driver.FindElement(By.Id("cards"));  
  
//通过类名获取元素by class name  
var byClassName = driver.FindElements(By.ClassName("menu"));  
  
// 通过标签名获取元素by tag name  
var byTagName = driver.FindElement(By.TagName("iframe"));  
  
// 通过名字获取元素  
var byName = driver.FindElement(By.Name("__VIEWSTATE"));  
  
// 通过链接文本获取元素by linked text http://www.google.com>linkedtext<  
var byLinkText = driver.FindElement(By.LinkText("linkedtext"));  
  
// 通过部分链接文本获取元素by partial link text :http://www.google.com>linkedtext<  
var byPartialLinkText = driver.FindElement(By.PartialLinkText("text"));  
  
//通过CSS选择器获取元素by css  
var byCss = driver.FindElement(By.CssSelector("#header .content .logo"));  
  
// 通过XPath来获取元素(by xpath  
var byXPath = driver.FindElements(By.XPath("//div"));
```

各方法使用优先原则:

优先使用id,name,classname,link; 次之使用CssSelector(); 最后使用Xpath();

因为Xpath()方法的性能和效率最低下。

四、获取页面元素和元素内容

1.Title: 标题

```
Console.WriteLine(driver.Title);//输出标题名
```

2.Url: 链接

```
Console.WriteLine(driver.Url);//输出链接
```

3.Text: 元素的文本值

```
Console.WriteLine(web.Text);//输出元素标记中文本的信息
```

4.Selected勾选情况、TagName标记名标、Enabled编辑状态、Displayed显示状态

5.GetAttribute () 获取标签的属性

```
var byIDAttributeText = byID.GetAttribute("id");
```

6.弹出对话框的处理

首先, 要先了解三种对话框: Alert、Confirmation以及Prompt。测试网页test.html:

```
<html>
  <head>
    <title>这是标题</title>
  </head>

  <body>
    <input type="button" onclick="alert('这是Alert');" value="Alert" /><br/>
    <input type="button" onclick="confirm('这是confirm');" value="confirm" /><br/>
    <input type="button" onclick="prompt('这是Prompt');" value="prompt" /><br/>
  </body>

</html>
```

下面进行测试:

```
var service = EdgeDriverService.CreateDefaultService(@".", "msedgedriver.exe");
IWebDriver driver = new OpenQA.Selenium.Edge.EdgeDriver(service);

driver.Navigate().GoToUrl("file:///C:/Users/bobin.yang/Source/Repos/WinForm2/bin/Debug/HTMLPage1.html");
```



```
IWebElement web = driver.FindElement(By.XPath("//input[1]"));
web.Click();

WebDriverWait wait = new WebDriverWait(driver, new TimeSpan(0,0,2));
//Wait for the alert to be displayed
wait.Until(ExpectedConditions.AlertIsPresent());

Console.WriteLine(driver.SwitchTo().Alert().Text); //在接收消息前输出
System.Threading.Thread.Sleep(1000);
driver.SwitchTo().Alert().Accept();

IWebElement web2 = driver.FindElement(By.XPath("//input[2]"));
web2.Click();
WebDriverWait wait2 = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
wait2.Until(ExpectedConditions.AlertIsPresent());
Console.WriteLine(driver.SwitchTo().Alert().Text); //在接收消息前输出
System.Threading.Thread.Sleep(1000);
driver.SwitchTo().Alert().Accept();

IWebElement web3 = driver.FindElement(By.XPath("@html/body/input[3]"));
web3.Click();
System.Threading.Thread.Sleep(1000);
Console.WriteLine(driver.SwitchTo().Alert().Text); //在接收消息前输出
driver.SwitchTo().Alert().SendKeys("这是输入的内容");
driver.SwitchTo().Alert().Accept();
```

五、操作元素对象WebElement

主要是进行Click和SendKeys操作，如图。其它的自己查看定义就知道了。

```
... string text { get; }
... void Clear();
... void Click();
... string GetAttribute( string attributeName );
... string GetCssValue( string propertyName );
... void SendKeys( string text );
... void Submit();
```

1、模拟鼠标点击元素

```
driver.FindElement(By.Id("copyright")).Click();
```

1. SendKeys就是在定位到输入框后，把参数text赋值进去

2. Click就是进行鼠标点击操作, 比如点击按钮等。和IDE上一样的。单选、复选都是通过这个方法点击的。
3. Clear方法: 是用于清空输入框的值, 和SendKeys正好的作用正好相反。自己测试时, 建议先给输入框赋值, 再用Thread.Sleep (3000) 来暂停一下, 再用Clear方法, 不然你还没看到效果时, 程序已经完成了。
4. Submit: 特殊之处在于, 当定位的是Form表单中任何一个元素, 当操作完之后, 直接调用那个Submit方法就能对整个Form表单完成提交。不用再返回重新查找表单元素。

Selenium中在指定的文本框中输入指定的字符串

```
//在文本框中输入指定的字符串sendKeys()  
Driver.FindElement(By.Id("tranAmtText")).SendKeys("123456");
```

2、下拉列表框Select的操作

```
driver.Navigate().GoToUrl("http://tieba.baidu.com/f/search/adv");  
IList listOption = driver.FindElement(By.Name("sm")).FindElements(By.TagName("option"));  
string targetStr = "按相关性排序";  
  
foreach (var option in listOption)  
{  
    if (option.Text == targetStr) // if (option.GetAttribute("value").Equals(targetStr))  
        option.Click();  
}
```

2、执行JS

```
var jsReturnValue = (IWebElement)((IJavaScriptExecutor)driver).ExecuteScript("jsfunname");
```

3、页面导航

```
driver.Navigate().Forward();  
driver.Navigate().Back();
```

Selenium中移动光标到指定的元素上

```
//移动光标到指定的元素上perform  
Actions action=new Actions(driver);  
action.MoveToElement(Find(By.XPath("//input[@id='submit' and @value='确定']"))).Perform();
```

4、拖拽操作(可以实现滑动验证码的验证)

```
var element = driver.FindElement(By.Name("source"));  
IWebElement target = driver.FindElement(By.Name("target"));  
(new Actions(driver)).DragAndDrop(element, target).Perform();
```

5、模拟鼠标晃动

```
//模拟光标晃动movebyoffset()  
Actions action = new Actions(driver);  
action.MoveByOffset(2, 4);
```

6、截图功能

```
//WebDriver中自带截图功能  
Screenshot screenShotFile = ((ITakesScreenshot)driver).GetScreenshot();  
screenShotFile.SaveAsFile("test", ImageFormat.Jpeg);
```

7、selenium操作滚轮滑动到底部

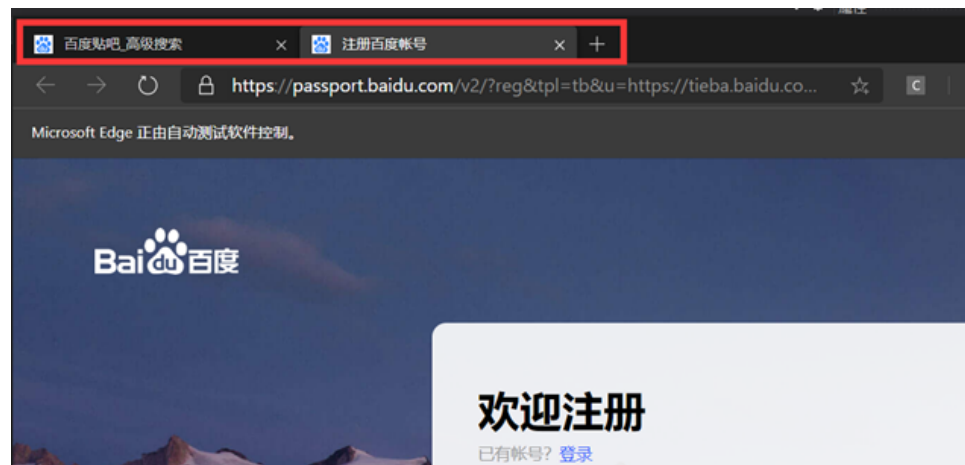
```
driver.execute_script("window.scrollTo(0,document.body.scrollHeight);")
```

六、获得窗口标识WindowHandles

要在不同的浏览器窗口之间切换，必须获得不同的窗口（标签）的标识符。用一个集合来保存这些数据。当需要到新窗口里进行操作时，需要将测试器前往新的窗口。开始创建的测试器是对窗口来的，所以当我们需切换窗口时。需要调用 测试器.SwitchTo().window(获得标识) 这个方法返回一个新的测试器对象。新的对象是代表的是切换的窗口。代码如下：

```
var service = EdgeDriverService.CreateDefaultService(".", "msedgedriver.exe");  
IWebDriver driver = new OpenQA.Selenium.Edge.EdgeDriver(service);  
  
driver.Navigate().GoToUrl("http://tieba.baidu.com/f/search/adv");  
//找到注册元素  
IWebElement register = driver.FindElement(By.XPath(@"//*[@id='com_userbar']/ul/li[5]/div/a"));  
register.Click();  
  
//显示所有标识  
IList<string> listHand = driver.WindowHandles;//拿到所有标识  
foreach (string item in listHand)  
{  
    Console.WriteLine(item);  
}  
  
/*这里一会插入代码*/  
  
Console.ReadKey();  
driver.Quit();
```

效果如下：



下面切换到新打开的窗口后，输入一个12345来表示我们成功了
在上面的代码基础下 添加下面代码

```
//切换到注册窗口再输入12345  
driver.SwitchTo().Window(listHand[1]);  
driver.FindElement(By.Name("userName")).SendKeys("12345");
```

结果如图：



1、关闭多个子Browser窗口

```
//获取所有的WindowHandle, 关闭所有子窗口
string oldwin = driver.CurrentWindowHandle;
ReadOnlyCollection<string> windows = driver.WindowHandles;
foreach (var win in windows)
{
    if (win != oldwin)
    {
        driver.SwitchTo().Window(win).Close();
    }
}
driver.SwitchTo().Window(oldwin);
```

2、对iframe中元素的定位

1、切换焦点到id为固定值的iframe上

进入页面后, 光标默认焦点在DefaultContent中, 若想要定位到iframe 需要转换焦点

```
driver.SwitchTo().DefaultContent();
//切换焦点到mainFrame
driver.SwitchTo().Frame("mainFrame");
```

需要注意的是: 切换焦点之后若想切换焦点到其他iframe上 需要先返回到defaultcontent, 再切换焦点到指定的iframe上。

2、切换焦点到id值为动态值的iframe上

有时候 页面上浮出层的id为动态值, 此时需要先获取所有符合记录的iframe放置在数组中, 然后遍历数组切换焦点到目标iframe上。

如下方法:

```
protected string bizFrameId = string.Empty;
protected string bizId = string.Empty;
//获取动态iframe的id值
protected void SetIframeId()
{
    ReadOnlyCollection els = driver.FindElements(By.TagName("iframe"));
    foreach (var e in driver.FindElements(By.TagName("iframe")))
    {
        string s1 = e.GetAttribute("id");
        if (s1.IndexOf("window") >= 0 && s1.IndexOf("content") >= 0)
        {
            bizFrameId = e.GetAttribute("id");
            string[] ss = s1.Split(new char[] { '_' });
            bizId = ss[1];
        }
    }
}
```

七、Cookies

在C#中, 通过Cookies属性来获取当前的Cookie集合, 然后进行增删改查操作。

Cookie由5个部分组成: 名称、值、所在域、路径和过期时间。

下面我们进入百度首页, 然后获取cookie, 并让它变动一下来看到效果。代码如下:

```
var service = EdgeDriverService.CreateDefaultService(@".", "msedgedriver.exe");
IWebDriver driver = new OpenQA.Selenium.Edge.EdgeDriver(service);

driver.Navigate().GoToUrl("http://www.baidu.com");

//获取Cookie
ICookieJar listCookie = driver.Manage().Cookies;
// IList listCookie = driver.Manage().Cookies.AllCookies; //只是显示 可以用Ilist对象
//显示初始Cookie的内容
Console.WriteLine("-----");
Console.WriteLine($"当前Cookie集合的数量: \t{listCookie.AllCookies.Count}");
for (int i = 0; i < listCookie.AllCookies.Count; i++)
{
    Console.WriteLine($"Cookie的名称:{listCookie.AllCookies[i].Name}");
    Console.WriteLine($"Cookie的值:{listCookie.AllCookies[i].Value}");
    Console.WriteLine($"Cookie的所在域:{listCookie.AllCookies[i].Domain}");
    Console.WriteLine($"Cookie的路径:{listCookie.AllCookies[i].Path}");
    Console.WriteLine($"Cookie的过期时间:{listCookie.AllCookies[i].Expiry}");
    Console.WriteLine("-----");
}

//添加一个新的Cookie
Cookie newCookie = new Cookie("新Cookie", "新值", "", DateTime.Now.AddDays(1));

listCookie.AddCookie(newCookie);
Console.WriteLine("-----");
Console.WriteLine($"当前Cookie集合的数量: \t{listCookie.AllCookies.Count}");
for (int i = 0; i < listCookie.AllCookies.Count; i++)
{
    Console.WriteLine($"Cookie的名称:{listCookie.AllCookies[i].Name}");
    Console.WriteLine($"Cookie的值:{listCookie.AllCookies[i].Value}");
    Console.WriteLine($"Cookie的所在域:{listCookie.AllCookies[i].Domain}");
    Console.WriteLine($"Cookie的路径:{listCookie.AllCookies[i].Path}");
    Console.WriteLine($"Cookie的过期时间:{listCookie.AllCookies[i].Expiry}");
    Console.WriteLine("-----");
}

//删除这个Cookie并再次显示总数
listCookie.DeleteCookieNamed(newCookie.Name);

Console.WriteLine($"当前Cookie集合的数量: \t{listCookie.AllCookies.Count}");
```

```
Console.ReadLine();  
driver.Quit();
```

运行效果如下:

```
-----  
当前Cookie集合的数量: 7  
Cookie的名称:新Cookie  
Cookie的值:新值  
Cookie的所在域:www.baidu.com  
Cookie的路径:/  
Cookie的过期时间:2020/6/12 10:44:15  
-----  
Cookie的名称:H_PS_PSSID  
Cookie的值:  
Cookie的所在域:.baidu.com  
Cookie的路径:/  
Cookie的过期时间:  
-----
```

八、Window窗口控制

这个属性是可以对当前的窗口进行简单的控制。如获取坐标和大小,还可以将其最大化。下面我们用过示例代码来试试效果。
下面的代码是先打开网页,打印坐标和大小,再控制它最大化,再次打印坐标和大小。

```
var service = EdgeDriverService.CreateDefaultService(@".", "msedgedriver.exe");  
IWebDriver driver = new OpenQA.Selenium.Edge.EdgeDriver(service);  
  
driver.Navigate().GoToUrl("http://www.baidu.com");  
  
//打印现在的坐标和大小  
IWindow window = driver.Manage().Window;  
Console.WriteLine("第一次打印");  
Console.WriteLine($"坐标X为{window.Position.X}\tY为{window.Position.Y}");  
Console.WriteLine($"大小长为{window.Size.Width}\t宽为{window.Size.Height}");  
Console.WriteLine("-----");  
  
//控制最大化  
window.Maximize();  
  
//再次打印数据  
Console.WriteLine("第二次打印");  
Console.WriteLine($"坐标X为{window.Position.X}\tY为{window.Position.Y}");  
Console.WriteLine($"大小长为{window.Size.Width}\t宽为{window.Size.Height}");  
Console.WriteLine("-----");  
  
Console.ReadLine();  
driver.Quit();
```


效果如下：

```
第一次打印
坐标X为9   Y为9
大小长为1011   宽为1093
-----
第二次打印
坐标X为-8   Y为-8
大小长为2064   宽为1128
-----
```

这里有两个奇怪的地方：

- 1.我的屏幕是1080P的，输出后的数据中，长只有1936.如果减去16的话到正好是1920.但是宽应该是1080，如果任务栏的宽度是24的话，到也能说得过去。只是数据和我们有的有点偏差，这里需要注意一下。
- 2.window属性居然只有位置、大小和最大化方法。居然没有最小化或还原（退出最大化状态）方法。

九、事件

首先是准备好要添加的事件，然后再挂接。这里可以使用C#的语法糖。在+=后面直接按两次tab键，然后再移动到事件区编辑。

```
eventDriver.Navigating += EventDriver_Navigating; //导航前
eventDriver.Navigated += EventDriver_Navigated; //导航后
eventDriver.FindingElement += EventDriver_FindingElement; //查找元素前
eventDriver.FindElementCompleted += EventDriver_FindElementCompleted; //查找元素后
eventDriver.ElementClicking += EventDriver_ElementClicking; //元素单击前
eventDriver.ElementClicked += EventDriver_ElementClicked; //元素单击后
eventDriver.ElementValueChanging += EventDriver_ElementValueChanging; //元素值改变前
eventDriver.ElementValueChanged += EventDriver_ElementValueChanged; //元素值改变后
eventDriver.ExceptionThrown += EventDriver_ExceptionThrown; //异常发生后事件
```

挂接事件：

```
#region 事件区

///
/// 导航前发生的事件
///
///
private void EventDriver_Navigating(object sender, WebDriverNavigationEventArgs e)
{
    this.listMessage.Add("-----");
    this.listMessage.Add($"即将要跳转到的URL为: {e.Driver.Url}");
}

///
/// 导航后发生的事件
///
///
'''
```

```
///
///
private void EventDriver_Navigated(object sender, WebDriverNavigationEventArgs e)
{
    this.listMessage.Add("-----");
    this.listMessage.Add($"跳转到的URL为: {e.Driver.Url}");
}

///
/// 查找元素前发生
///
///
///
private void EventDriver_FindingElement(object sender, FindElementEventArgs e)
{
    this.listMessage.Add("-----");

    this.listMessage.Add($"即将查找的元素为: {e.FindMethod.ToString()}");
}

///
/// 查找元素后发生
///
///
///
private void EventDriver_FindElementCompleted(object sender, FindElementEventArgs e)
{
    this.listMessage.Add("-----");
    this.listMessage.Add($"找到元素, 条件为: {e.FindMethod.ToString()}");
}

///
/// 单击元素前发生
///
///
///
private void EventDriver_ElementClicking(object sender, WebElementEventArgs e)
{
    this.listMessage.Add("-----");
    this.listMessage.Add($"要单击的元素的value属性为: {e.Element.GetAttribute("value")}");
}

///
/// 单击元素后发生
///
///
///
private void EventDriver_ElementClicked(object sender, WebElementEventArgs e)
{
    System.Threading.Thread.Sleep(3 * 1000); // 暂停3秒
    this.listMessage.Add("-----");
    this.listMessage.Add($"单击元素后, 现在的URL为: {e.Driver.Url}");
}
```

```
///
/// 单击元素前发生
///
///
///
private void EventDriver_ElementValueChanging(object sender, WebElementEventArgs e)
{
    this.listMeassage.Add("-----");
    this.listMeassage.Add($"元素更改前的值为: {e.Element.GetAttribute("value")}");
}

///
/// 单击元素后发生
///
///
///
private void EventDriver_ElementValueChanged(object sender, WebElementEventArgs e)
{
    this.listMeassage.Add("-----");
    this.listMeassage.Add($"元素更改后的值为: {e.Element.GetAttribute("value")}");
}

///
/// 异常(保存截图到本地)
///
///
///
private void EventDriver_ExceptionThrown(object sender, WebDriverExceptionEventArgs e)
{
    //地址
    string strPath = $"D:\\Desktop\\{DateTime.Now.ToString("yyyy-MM-dd HH-mm-ss")}.png";

    //保存截图
    Screenshot screen = (sender as EventFiringWebDriver).GetScreenshot();
    screen.SaveAsFile(strPath, System.Drawing.Imaging.ImageFormat.Png);

    //输出保存信息
    this.listMeassage.Add("-----");
    this.listMeassage.Add($"发生异常, 截图已保存到: {strPath}");
}
```

十、关闭浏览器

有下面两种:

1. Close():关闭WedDriver对象所在的窗口;

第一个是关闭一个窗口, 一个wedDriver对象是可以有多个窗口的 (之前的窗口切换也是提到过), 需要关闭时要保证当前激活

的窗口。比如一个webDriver对象里有好多窗口，你要关闭第2个窗口，就要用SwitchTo().Window()方法切换到第2个窗口才能关闭，不能直接关闭第几个窗口的。

2. Quit():关闭所有相关窗口;

第二个关闭和这个webDriver对象所有相关的窗口。当然，一个脚本是可以有多个webDriver对象

下面代码的代码展示这两个方法的用法和用途。

- 1.打开百度首页，单击“注册”超级链接。
- 2.在弹出的窗口（百度账户注册）中，调用Close () 方法，关闭新弹出的页面
- 3.再一次点击“注册”超级链接，调用Quit () 方法来结束测试。

```
var service = EdgeDriverService.CreateDefaultService(@".", "msedgedriver.exe");
IWebDriver driver = new OpenQA.Selenium.Edge.EdgeDriver(service);

//导航到百度首页
driver.Navigate().GoToUrl("http://www.baidu.com");

//进行点击
Console.WriteLine("-----");
Console.WriteLine("进行点击");
driver.FindElement(By.LinkText("登录")).Click();
System.Threading.Thread.Sleep(3 * 1000);
driver.FindElement(By.LinkText("立即注册")).Click();

//获取窗口句柄
IList<string> listHand = driver.WindowHandles;

//切换到注册窗口并关闭
Console.WriteLine("-----");
Console.WriteLine("切换到注册窗口");
driver.SwitchTo().Window(listHand[1]);
System.Threading.Thread.Sleep(3 * 1000);
Console.WriteLine("-----");
Console.WriteLine("关闭注册窗口");
driver.Close();
System.Threading.Thread.Sleep(3 * 1000);

//切换到主窗口并结束测试
Console.WriteLine("-----");
Console.WriteLine("切换到主窗口并结束测试");
driver.SwitchTo().Window(listHand[0]);
driver.FindElement(By.LinkText("立即注册")).Click();
System.Threading.Thread.Sleep(3 * 1000);
driver.Quit();

Console.ReadLine();
```

分类: [15 WinForm](#), [20 三方控件](#)

好文要顶

关注我

收藏该文

springsnow

粉丝 - 96 关注 - 5

20

0

+加关注

« 上一篇: [MongoDB \(08\) : 索引](#)

» 下一篇: [MongoDB \(10\) : 在C#中使用MongoDB](#)

posted on 2020-06-10 19:44

springsnow

阅读(4392)

评论(0)

编辑

收藏

举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论，立即 [登录](#) 或者 [逛逛](#) 博客园首页

编辑推荐:

- 斗鱼 H5 直播原理解析，它是如何省了 80% 的 CDN 流量?
- 超强的纯 CSS 鼠标点击拖拽效果
- 新零售SaaS架构：中央库存系统架构设计
- 不安装运行时运行 .NET 程序 - NativeAOT
- 从 C# 崩溃异常 中研究页堆布局

最新新闻:

- 微软秋季发布会：5G版Surface Pro亮相 加深与苹果生态融合
- 扎克伯格谈新款万元VR头显：成本价，我们不像苹果那样定高价
- 比亚迪×奔驰的火爆新车，让我开到半夜不回家
- 抖音集团上线新 Logo
- 腾讯视频否认将接入 88VIP
- » 更多新闻...

历史上的今天:

2011-06-10 C# (99) : 自定义集合类

目录
导航

Powered by:

博客园

Copyright © 2022 springsnow

Powered by .NET 6 on Kubernetes

目录
导航