基于Python3.X的Selenium自动测试环境搭建与简介

Selenium简介

- Selenium是一个用于Web应用程序的自动化测试工具，Selecninm是直接运行在浏览器上，Selenium功能非常强大，能够模拟基本所有的手动操作功能，能够达到真实用户操作一样。同时支持浏览器也非常全面，包括Chrome、FireFox、IE6\7\8、Phantomjs、移动端等主流浏览器，也支持多种编程语言，如Java、C#、Ruby、Python等。

Selenium API介绍

- 首先我们先了解下SeleniumAPI的基本功能，看看它的强大之处，环境搭建先不要慌。

定位元素(Locating Elements)

- 这个是非常简单的，如果用过JQuery或做过前端的朋友那就好了，简直零门槛。不了解的朋友也不要着急，同样也简单。
- 定位元素有很多中途径，包括：ID、CLASS、CSS、NAME和我最喜欢的XPATH等等。如通过ID方式获得元素：

```
    html:
        <input type="text" id="input"/>
    selenium:
        driver = webdriver.Chrome()
        driver.get("http://www.XXX.com")
        driver.find_element_by_id("input")
```

详情可查阅 官方文档-Locating Elements

等待(Waits)

> Waits非常重要，尤其现在页面异步请求(AJAX)非常多，不可能设置一个time.sleep，这样非常浪费时间不说，同时也非常的不方便和不专业。

> 等待分为显示(Explicit Waits)和隐式(Implicit Waits),隐式就不说了，跟time.sleep差不多，我们主要介绍显示等待。

- 显示等待主要是给定一个预期条件，比如等待某个元素消失、等待某个元素可以点击、存在等等，它将设置一个等待时间如10秒，在10秒的时间内每过500毫秒检测一下预期条件是否满足，如果满足就停止等待继续执行后面的代码，如果10秒后还未满足预期条件，将抛出一个超时的错误异常。
- 示例代码，以等待元素可点击为例

```
selenium:
```

```
element = WebDriverWait(driver, 10).until(    #10表示等待的时间(秒)
        #By.ID用于定位元素的类型
        EC.element_to_be_clickable((By.ID, "input"))
    )
```

> expected_conditions(EC)模块中提供多种预期等待条件,基本能够满足，同时也可自定义等待条件，详情可查阅官方文档-Waits

---

简单介绍就说到这里了，码字太累了，还有较多API你们自己去看了，如Alert弹框处理、鼠标移动元素效果Action Chains等等。

我的建议是将整个文档都看一遍，这样非常有帮助，可以全局的了解Selenium，毕竟这个文档比较短官方文档

环境搭建

> 环境搭建可大可小，要知道好多朋友都死在环境搭建上，弄了许久都没成功就直接放弃了，一个好好的人才就这样埋没了，我开始也费了好大功夫，网上找了许多资料也不全面，好多都是复制粘贴，估计连自己都没搞懂，再说好多都是Java+Selenium，Python+Selenium的资料比较片面。

- 这里我就不介绍Linux环境了，大同小异，我到觉得Linux相对简单，而且Linux自带Python不用单独安装，需要注意Python的版本。

环境搭建所需的包

- 以Windows+Python3.5为例，需要注意的是在下载安装Python的时候尽量保持和操作系统一样的位数，不然有可能造成不兼容，安装相关的依赖包会失败。

1.下载Python3.5安装包并安装，安装时注意勾选 Add Python 3.5 to PATH ，这样就不需要单独在配置环境变量。

- 查看是否安装成功，打开命令行运行(win+r输入cmd回车)

```
python --version
```

- 出现：Python 3.5.0 表示Python安装成功，如出现不是内部或外部命令...那就是环境变量出问题了，不用着急，咋们手动配置：
    - 首先找到Python的安装目录，如我Python安装在C:\Python35，复制此路径。
    - win+r 打开运行输入sysdm.cpl，在选中高级，点击下面的环境变量。
    - 在用户变量的Path中加入 ;C:\Python35;C:\Python35\Scripts，注意分号也要加上，表示分隔符，最后的可以不用加。
    - 在系统变量的PATHEXT加入 ;.PYM;.PY;.PYW，注意分号和点，同理最后的不加分号。

- 在次运行python --version应该就没问题了

2.升级pip

```
python -m pip install -U pip
```

3.安装Pillow

- Pillow是PIL的一个分支,用于图像处理，主要是配合Selenium的API获得验证码元素坐标截取保存图片，在通过tesseract识别验证码，达到自动登录的效果。

```
pip install Pillow
```

4.安装tesseract

- tesseract-ocr可对图片文字进行分析识别，我们这里主要用于识别截取的验证码图片， 下载地址，下载为一个exe文件可直接安装，安装完成后执行命令:

```
tesseract -v
```

- 出现：tesseract 3.02 ... 表示安装成功,如出现不是内部或外部命令...，按照上面的方式配置环境变量。
- 识别率高不高关键在于训练的字符库是否包含类似的图形，安装后有默认的字体库，可以识别数字、字母、正楷汉字，精度很高，识别较为复杂的图形需要自己训练，网上有很多文章介绍，有兴趣的朋友可以自己研究。
- 只要有相关的字体库，使用起来也非常方便，在命令行中运行:

```
    tesseract [原始图片路径] [识别后保存的路径] -l [字体库名称]
例：
原始图片路径：c:\test.png
识别后保存的路径：test
字体库名称：  安装后有个默认的字体库 eng
最后就是：
    tesseract c:\test.png test -l eng
成功后将在c盘生成一个test.txt的文本文件，里面就是识别的内容。
```

5.下载浏览器驱动程序

- 不同浏览器有不同的驱动程序，你要使用哪种浏览器测试就下载对用的驱动，这里需要注意IE浏览器需要区分32还是64位，如果64位用32位的浏览器驱动，是可以用，就是在测试的时候输入文本这些会非常的慢，相反同理。驱动下载地址1 驱动下载2，驱动下载后是一个exe文件，需要将下载的文件移动到Python的安装目录，就是前面配置的环境变量的目录(C:\Python35\)即可。

6.安装Selenium

```
pip install selenium
```

至此，环境搭建完成，上面6点除了1、2点其它安装不分先后，也可不用安装4、3点，可通过程序控制手动输入验证码，只是没那么完美。