selenium —— 基于Web的UI自动化测试工具

一、selenium简介

1. 只能做Web应用（浏览器打开的）系统的UI（界面）的自动化测试

2. 主要做功能测试，兼容主流的浏览器（IE、Firefox、Chrome，手机上的浏览器）

3. 支持很多语言：python、java

4. 针对python而言，selenium是一个第3方的库（包），需要额外的安装

二、安装selenium的环境

1. 安装selenium这个第3方的python库

- 推荐使用pip程序来安装selenium
  - （一）在cmd中，输入：`pip install selenium`
    - 使用国内清华的镜像，提高下载速度

      `pip install -i https://pypi.tuna.tsinghua.edu.cn/simple selenium`
  - （二）在PyCharm中，进入到系统设置菜单（settings），project解释器去进一步搜索安装。

2. 下载和配置浏览器驱动程序

- selenium是依靠浏览器厂商提供的"浏览器驱动程序"去操作浏览器的，因此要下载对应浏览器及版本的驱动程序文件，将这个文件放置在环境变量path指定的目录下
- 下载地址

```
（1）Chrome（推荐）
   http://npm.taobao.org/mirrors/chromedriver
（2）Firefox
   https://github.com/mozilla/geckodriver/releases/
（3）IE
   http://selenium-release.storage.googleapis.com/index.html
```

- 将解压出来的驱动程序文件，放置在环境变量PATH能找得到的目录下

  PS：如果配置了环境变量，记得重启PyCharm

三、WebDriver —— 浏览器驱动类

1. 创建WebDriver类的实例对象 —— 启动浏览器（以Chrome浏览器为例）

```
from selenium.webdriver.chrome.webdriver import WebDriver

驱动对象 = WebDriver()  # 启动浏览器
```

2. 方法

- `驱动对象.get('url')` —— 打开指定URL的页面
- `驱动对象.close()` —— 关闭当前的页面（如果当前只有一个页面，也就等于退出浏览器了）
- `驱动对象.quit()` —— 退出浏览器
- **在页面上定位元素（八种策略、2套方法）**
  - **定位单个元素【find_element】 —— 返回的是Web元素类型的对象**
    - 方法
      - `驱动对象.find_element_by_id('标签上的id属性')` —— 根据元素的id属性
      - `驱动对象.find_element_by_name('标签上的name属性')` —— 根据元素的name属性
      - `驱动对象.find_element_by_class_name('标签上的class类名')` —— 根据元素的class属性
        - 如果标签上的class属性中，用了多个类样式名，只能传入一个类名
      - `驱动对象.find_element_by_tag_name('标签名')` —— 根据元素的标签名
      - `驱动对象.find_element_by_link_text('超链接的文本')` —— 根据超链接元素的文本
      - `驱动对象.find_element_by_partial_link_text('超链接的部分文本')` —— 根据超链接元素的部分文本
      - `驱动对象.find_element_by_css_selector('CSS选择器表达式')` —— 根据CSS选择器定位元素
      - `驱动对象.find_element_by_xpath('xpath表达式')` —— 根据XPATH表达式定位元素
      - `驱动对象.find_element(By.XXX, '条件')` —— 根据指定的策略和条件，定位元素
    - 注意点
      - 如果页面上有多个元素匹配，只返回找到的第1个元素
      - 如果页面上没有匹配的元素，会报错（引发异常：NoSuchElementException）
      - 尽量使用唯一、有含义的定位特征。
  - **定位多个元素【find_elements】 —— 返回的是list类型的对象**
    - 方法
      - `驱动对象.find_elements_by_id('标签上的id属性')` —— 根据元素的id属性
      - `驱动对象.find_elements_by_name('标签上的name属性')` —— 根据元素的name属性
      - `驱动对象.find_elements_by_class_name('标签上的class类名')` —— 根据元素的class属性
        - 如果标签上的class属性中，用了多个类样式名，只能传入一个类名
      - `驱动对象.find_elements_by_tag_name('标签名')` —— 根据元素的标签名

        - `驱动对象.find_elements_by_link_text('超链接的文本')` —— 根据超链接元素的文本
        - `驱动对象.find_elements_by_partial_link_text('超链接的部分文本')` —— 根据超链接元素的部分文本
        - `驱动对象.find_elements_by_css_selector('CSS选择器表达式')` —— 根据CSS选择器定位元素
        - `驱动对象.find_elements_by_xpath('xpath表达式')` —— 根据XPATH表达式定位元素
        - `驱动对象.find_elements(By.XXX, '条件')` —— 根据指定的策略和条件，定位元素
    - 注意点：
        - 永远返回的是list（列表类型），列表中装的是Web元素
        - 如果没有找到匹配的元素，并不会报错，只是返回的列表长度为0而已。

3. 属性

- `驱动对象.current_url` —— 获取当前页面的URL地址
- `驱动对象.title` —— 获取当前页面的标题

四、Web元素

- 本质：网页上的标签
- **方法**
    - `元素.send_keys('文本内容')` —— 向元素中输入内容
    - `元素.clear()` —— 清空元素中的文本
    - `元素.click()` —— 单击元素
    - `元素.get_attribute('属性名')` —— 获取元素标签上的指定的属性
- **属性**
    - `元素.text` —— 获取标签内部的文本
    - `元素.tag_name` —— 获取元素的标签名

五、CSS选择器

- 示例代码

```
from selenium.webdriver.chrome.webdriver import WebDriver
dr = WebDriver()
dr.get("file:///E:/PycharmProjects/SeleniumDay1/demo.html")
for elem in dr.find_elements_by_css_selector("h3"):
    print(elem.text)
for elem in dr.find_elements_by_css_selector("h3, p"):
    print(elem.text)
for elem in dr.find_elements_by_css_selector(".poem > h3, .poem > p"):
    print(elem.text)
for elem in dr.find_elements_by_css_selector(".poem  h3, .poem  p"):
    print(elem.text)
for elem in dr.find_elements_by_css_selector("#hot"):
```

```
    print(elem.text)
for elem in dr.find_elements_by_css_selector('[align="right"]'):
    print(elem.text)
for elem in dr.find_elements_by_css_selector("p[align=right]"):
    print(elem.text)
dr.quit()
```

六、XPATH表达式

- 理解：是为了在XML（可扩展的标记语言：html文件也算是xml文件）文件中搜索标签元素的表达式
- 路径描述符
  - `/` —— 从网页文件的根开始计算，每一个`/`代表进入下一级
  - `//` —— 任意一级
- `节点[限定条件]`
  - `节点[@属性名='值']` **—— 匹配指定的节点中，含有指定属性值的元素（使用最多）**
  - `节点[text()='值']` **—— 匹配内部文本等于指定值的元素**
  - `节点[N]` —— 匹配第N个子元素
  - `节点[last()]` —— 匹配最后一个子元素
  - `节点[contains(@属性名, '值')]` —— 匹配指定属性包含某个值的元素
  - `节点[contains(text(), "值")]` —— 匹配内部文本包含某个值的元素
  - `节点[starts-with(@属性名, '值')]` —— 匹配指定属性以某个值打头的元素
  - `节点[starts-with(text(), "值")]` —— 匹配内部文本以某个值打头的元素

七、By类（策略类）

- 理解：定义了8个类属性，代表8中策略
- 导入：`from selenium.webdriver.common.by import By`
- `By.ID`、`By.NAME`、`By.CLASS_NAME`、……