10–用程序指挥浏览器

selenium简介

他是一个强大的python库，我们可以通过它控制浏览器，作出自动打开、输入、点击等操作

静态网页和动态网页

1.用html写出的网页，就是静态网页。我们使用BeautifulSoup爬取这类型网页，因为网页源代码中就包含着网页的所有信息，因此，网页地址栏的URL就是网页源代码的URL。

2.要爬取的数据不在HTML源代码中，而是在json中，不能直接使用网址栏的URL，而需要找到json数据的真实URL。这就是一种动态网页。

浏览器总是在向服务器发起各式各样的请求，当这些请求完成后，它们会一起组成开发者工具的Elements中所展示的，渲染完成的网页源代码。

selenium使用场景

selenium的优点：

遇到页面交互复杂或是URL加密逻辑复杂的情况时，我们可以使用selenium打开一个浏览器，等待所有数据都加载到Elements中之后，再把这个网页当做静态网页爬取

selenium的缺点：

由于要真实地运行本地浏览器，打开浏览器以及等待网渲染完成需要一些时间，selenium的工作不可避免地牺牲了速度和更多资源。

selenium的配置

selenium安装

```
pip install selenium # Windows电脑安装selenium

pip3 install selenium # Mac电脑安装selenium
```

浏览器配置

http://npm.taobao.org/mirrors/chromedriver/

本地Chrome浏览器设置方法：

```
#从selenium库中调用webdriver模块
from selenium import webdriver
# 设置引擎为Chrome，从本地打开一个Chrome浏览器
driver = webdriver.Chrome()
#以上就是浏览器的设置方式：把Chrome浏览器设置为引擎，然后赋值给变量driver。driver是实例化的浏览器，在后面的代码中我们都需要添加这些内容。从而使的我们能够控制浏览器。
```

selenium获取处理数据用法

获取数据

```
# 本地Chrome浏览器设置方法
from selenium import webdriver #从selenium库中调用webdriver模块
#导入时间模块
import time
driver = webdriver.Chrome() # 设置引擎为Chrome，真实地打开一个Chrome浏览器
driver.get('https://xiaoke.kaikeba.com/example/X-Man/') # 打开网页
time.sleep(3)
driver.close() # 关闭浏览器
```

get(URL)是webdriver的一个方法，它的使命是为你打开指定URL的网页。

dirver是我们实例化后的一个浏览器，我们就是通过dirver打开网页，网页中的数据加载到浏览器以后，数据我们就获取到了。

dirver.close()是关闭浏览器的方法，我们每次使用webdirver以后，都需要关闭它，不然他会一只占用我们的内存。

解析并提取数据

selenium库也具备解析数据、提取数据的能力，它和BeautifulSoup的底层原理是一致的，不同的是，selenium解析提取的是Elements中的数据，BeautifulSoup解析的是Nextwork中第0个请求的响应

```
from selenium import webdriver #从selenium库中调用webdriver模块
#导入time模块
import time
driver = webdriver.Chrome() # 设置引擎为Chrome，从本地打开一个Chrome浏览器
driver.get('https://xiaoke.kaikeba.com/example/X-Man/') # 访问页面
time.sleep(3) # 等待3秒
label = driver.find_element_by_tag_name('label') # 解析网页并提取第一个<label>标签
print(label.text) # 打印label的文本
driver.close() # 关闭浏览器
```

提取数据的方法：

```
#通过元素id提取一个或者所有数据
dirver.find_element_by_id()
dirver.find_elements_by_id()
```

```
#通过元素的class提取一个或者所有数据
dirver.find_element_by_class_name()
dirver.find_elements_by_class_name()
#通过元素的标签名tag来提取一个或者所有数据
dirver.find_element_by_tag_name()
dirver.find_elements_by_tag_name()
#通过元素的neme提取一个或者所有数据
dirver.find_element_by_name()
dirver.find_elements_by_name()
#通过链接的部分文本来获取一个或者所有超链接
dirver.find_element_by_partial_link_text()
dirver.find_elements_by_partial_link_text()
#通过链接的文本来获取一个或者所有超链接
dirver.find_element_by_link_text()
dirver.find_elements_by_link_text()
```

解析数据的方法：

```
#获取文字
webElement.text
#带参数(属性名)，可以获取属性值
webElement_get_attribute()
```

总结

1.selenium操作数据转换：

1.dirver.get()获取数据

2.dirver.find_element_by提取数据

3.webelement解析数据

2.selenium配合BeautifulSoup处理数据

```
#获取到渲染完整的网页源代码
HTML = driver.page_source
#page_source获取到的数据已经是str类型，不需要text转化
bs_movies = BeautifulSoup(HTML,'html.parser')
```

3.selenium常用操作

```
.send_keys() # 模拟按键输入，自动填写表单
.click() # 点击元素
.clear()#清楚元素所有内容
```

实操代码

```
# 本地Chrome浏览器设置方法
from selenium import  webdriver # 从selenium库中调用webdriver模块
import time # 调用time模块
driver = webdriver.Chrome() # 设置引擎为Chrome，真实地打开一个Chrome浏览器
driver.get('https://xiaoke.kaikeba.com/example/X-Man/') # 访问页面
```

```
time.sleep(2) # 暂停两秒，等待浏览器缓冲
teacher = driver.find_element_by_id('teacher') # 找到【请输入你喜欢的老师】下面的输入框位置
teacher.send_keys('开课吧') # 输入文字
assistant = driver.find_element_by_name('assist') # 找到【请输入你喜欢的助教】下面的输入框位置
assistant.send_keys('全都喜欢') # 输入文字
time.sleep(2)
button = driver.find_element_by_tag_name('button') # 找到【提交】按钮
time.sleep(1)
#点击提交按钮
button.click()
time.sleep(5)
#关闭浏览器
driver.close()
```

qq音乐获取歌曲《说好不哭》全部精彩评论

```
# 这段代码在本地执行
from selenium import webdriver #从selenium库中调用webdriver模块
import time
driver = webdriver.Chrome() # 设置引擎为Chrome，从本地打开一个Chrome浏览器
driver.get('https://y.qq.com/n/yqq/song/001qvvgF38HVc4.html') # 访问页面
time.sleep(2)
while True:
    try:
#找到记载更多的按钮
        clickformore = driver.find_element_by_class_name('js_get_more_hot')
        time.sleep(0.5)
#点击加载更多
        clickformore.click()
    except:
#如果出错，执行except
        print("不能再点击加载更多啦")
        break
comments = driver.find_element_by_class_name('js_hot_list').find_elements_by_class_name('js_cmt_li') # 使用class_name找到评论
print(len(comments))
for i in range(len(comments)): # 循环
    comment = comments[i].find_element_by_tag_name('p') # 找到评论
    print ('评论'+str(i)+':'+comment.text + '\n--------------------------------------------------') # 打印评论
driver.close() # 关闭浏览器
```

浏览器静默模式设定

```
# 本地Chrome浏览器的静默默模式设置：
from selenium import  webdriver #从selenium库中调用webdriver模块
```

```
from selenium.webdriver.chrome.options import Options # 从options模块中调用Options类
chrome_options = Options() # 实例化Option对象
chrome_options.add_argument('--headless') # 把Chrome浏览器设置为静默模式
driver = webdriver.Chrome(options = chrome_options) # 设置引擎为Chrome，在后台默默运行
```