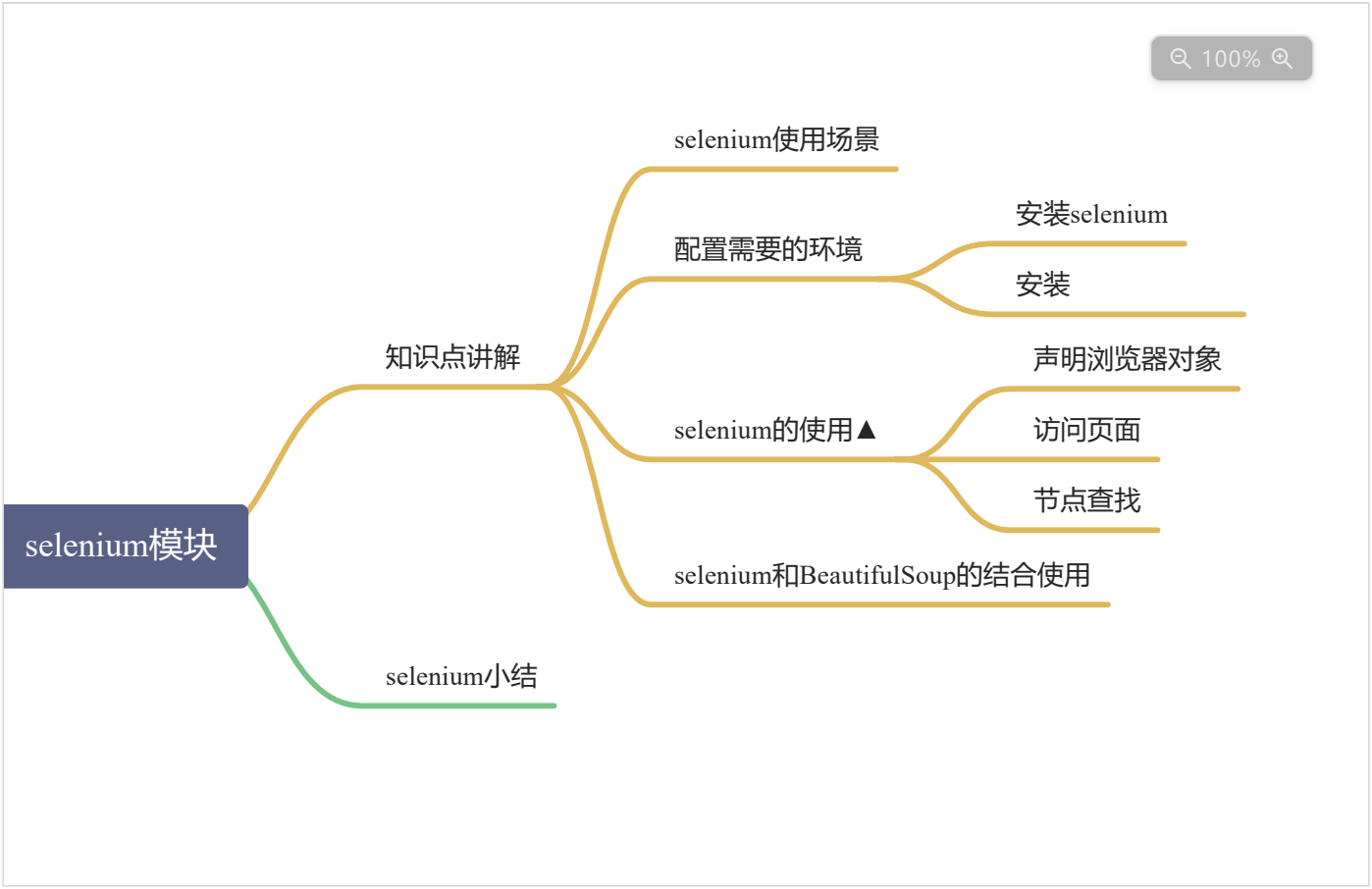
第9课 selenium模块

一、课程结构导图



注：▲为重点知识点。

二、知识点讲解

2.1 selenium使用场景

当我们需要爬取一些动态渲染的网页（比如QQ音乐和今日头条新闻等），使用requests库和BeautifulSoup库的方法会非常困难。另外有一些页面的图形是经过javascript计算之后生成的，无法直接爬取。

为了解决这些问题，我们可以使用模拟浏览器的方式来进行爬取，使用selenium模拟浏览器运行，可以做到网页检查模式中element里的源代码有什么，我们就能爬到什么。

2.2 配置需要的环境

2.2.1 安装selenium

windows用户在cmd终端输入pip install selenium，mac os用户在terminal终端输入pip3 install selenium安装selenium库。

2.2.2 安装ChromeDriver

首先请你在网页上搜索最新的的Chrome浏览器并安装（注意有一些浏览器图标和Chrome很像，请区分）。

- **mac os系统**

方法一：
打开Mac终端terminal ，输入命令:
curl -s https://localprod.pandateacher.com/python-manuscript/crawler-html/chromedriver/chromedriver-for-Macos.sh <https://localprod.pandateacher.com/python-manuscript/crawler-html/chromedriver/chromedriver-for-Macos.sh> | bash
等待程序安装完成就好了。

方法二：（请在方法一失败的情况下使用）
在http://npm.taobao.org/mirrors/chromedriver/ <http://npm.taobao.org/mirrors/chromedriver/2.44/> 链接里下载对应的mac64版本并解压。然后将解压后的文件配置到环境变量或将文件移动到属于环境变量的目录里。

例如：打开terminal终端，把解压后的文件拖拽到终端里，会看到一个路径，假如这个路径是/Users/xxx/Downloads/chromedriver ，复制这个路径，将其移动到/usr/bin，命令如下：

```
sudo mv /Users/xxx/Downloads/chromedriver /usr/bin
```

- **windows系统**

在http://npm.taobao.org/mirrors/chromedriver/ <http://npm.taobao.org/mirrors/chromedriver/2.44/> 链接里下载win32版本并解压，解压之后将文件移动到Python的根目录下即可，Python的根目录可在cmd终端里输入where python查看，在/python.exe之前的部分即为Python的根目录。

2.3 selenium的使用

2.3.1 声明浏览器对象

- **可视模式**

以Chrome浏览器为例，我们要模拟手动操作浏览器，就要去创建一个可操纵浏览器的对象。

```
from selenium import webdriver

brow = webdriver.Chrome()
```

通过上面的代码，我们就创建了一个可以操纵浏览器的对象brow，并且之后的操作多依赖此对象。

- **静默模式**

当我们不想直接打开浏览器但依旧想操纵浏览器的时候，可以只使用浏览器的内核，这种方式叫做静默模式。

```
# 本地Chrome浏览器的静默模式设置：
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
```

2.3.2 访问页面

- **get()方法**

用法：当我们声明了浏览器的对象后，可以使用浏览器对象.get()方法来请求到网页的完整源代码，当请求到网页源代码之后，可以使用page_source方法来打印获取到的网页源代码。

示例：

```
from selenium import webdriver
import time

brow = webdriver.Chrome()
brow.get('https://www.baidu.com/')
time.sleep(2)
print(brow.page_source)

brow.close()
```

2.3.3 节点查找

selenium的节点查找（标签定位）机制与BeautifulSoup的机制十分类似，可以类比学习。

- **单个节点**

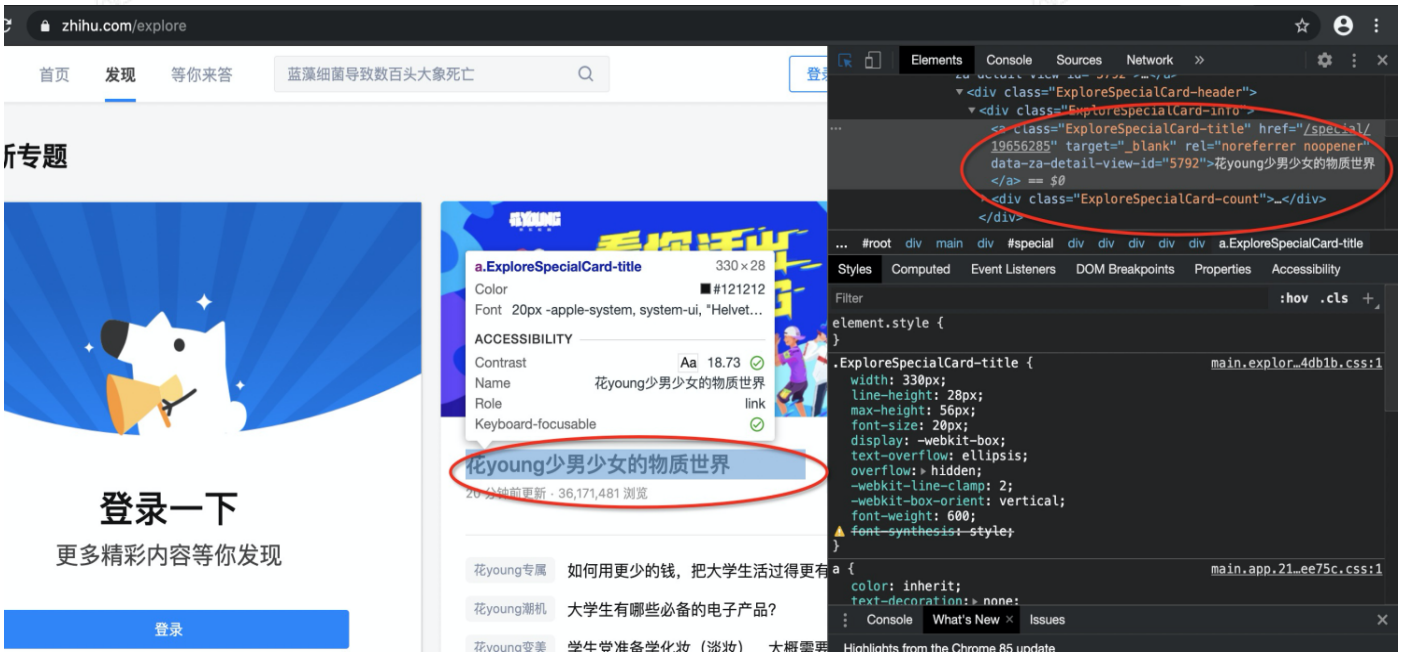
用法：

Selenium提取数据的方法

方法	作用
find_element_by_tag_name	通过元素的标签名称选择
find_element_by_class_name	通过元素的class属性选择
find_element_by_id	通过元素的id选择
find_element_by_name	通过元素的name属性选择
find_element_by_link_text	通过链接文本获取超链接
find_element_by_partial_link_text	通过链接的部分文本获取超链接

示例：

我们要爬取下面图片中圈出来的文字和它的链接(href标签)并打印出来。



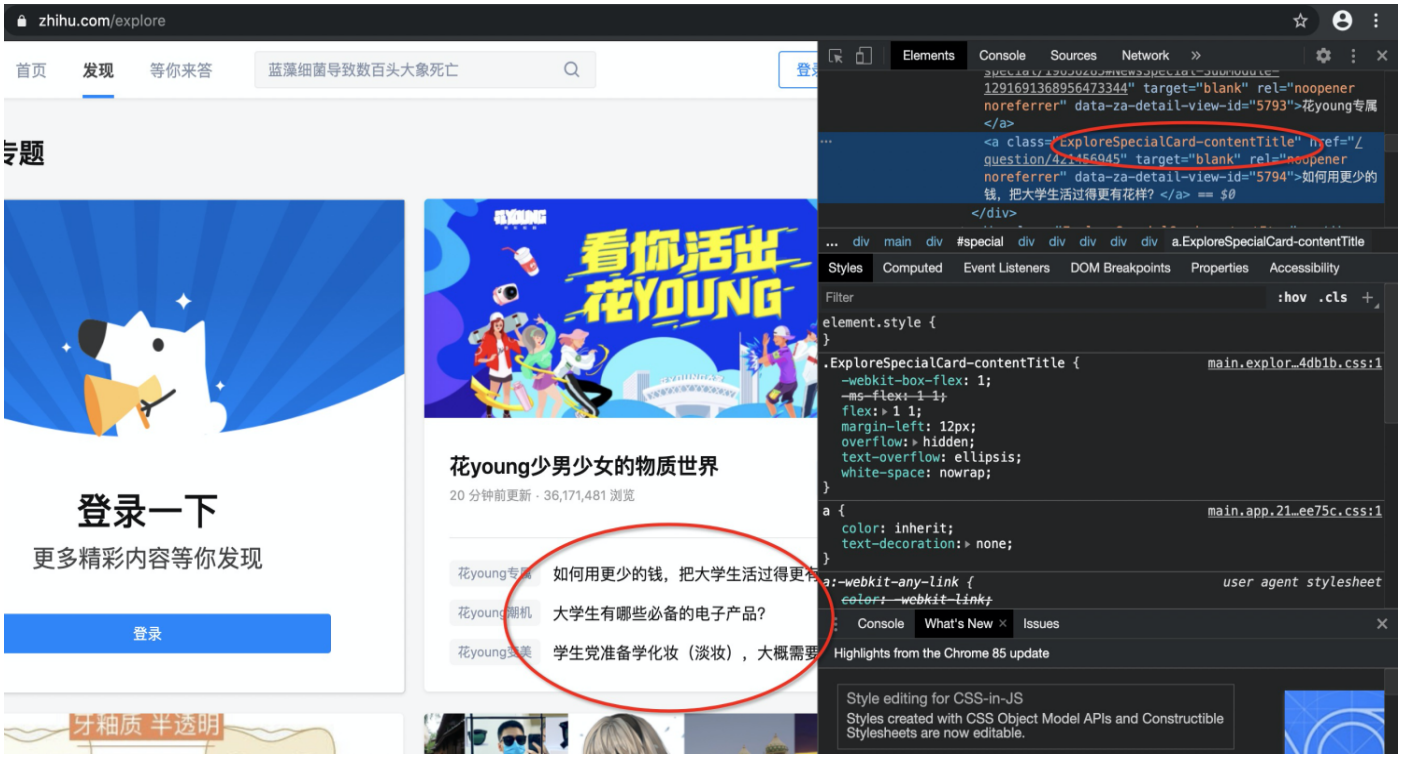
```python
from selenium import webdriver
import time

brow = webdriver.Chrome()
brow.get('https://www.zhihu.com/explore')
time.sleep(2)

target = brow.find_element_by_class_name('ExploreSpecialCard-title')
link = target.get_attribute('href')
print(target.text)
print(link)

brow.close()
```

- **多个节点**

用法：与查找单个节点的区别是find_element_by_xxx改为find_elements_by_xxx。

示例：

爬取下图中圈起来的三个内容，其中三个标签都有class='ExploreSpecialCard-contentTitle'属性。



示例：

```
from selenium import webdriver
import time

brow = webdriver.Chrome()
brow.get('https://www.zhihu.com/explore')
time.sleep(2)

target = brow.find_elements_by_class_name('ExploreSpecialCard-contentTitle')

for i in target:
    print(i.text)
    print(i.get_attribute('href'))

brow.close()
```

- **节点交互**

用法：我们可以在找到节点的基础上让浏览器模拟一些交互操作，比如在输入框里输入文字，点击按钮等。输入文字使用send_keys()方法，点击按钮使用click()方法。

示例：

模拟打开百度搜索引擎，并输入“苹果新品发布会”，然后点击搜索。其中输入框的标签拥有id='kw'属性，搜索按钮的标签拥有id='su'属性。


```python
from selenium import webdriver
import time

brow = webdriver.Chrome()
brow.get('https://www.baidu.com/')
time.sleep(2)

input = brow.find_element_by_id('kw')
input.send_keys('苹果新品发布会')

button = brow.find_element_by_id('su')
button.click()
time.sleep(1)

brow.close()
```

2.4 selenium和BeautifulSoup的综合使用

思路：回到selenium的使用场景，selenium适合爬取动态网页，可以通过操纵浏览器来获取到网页完整的经过渲染的源代码，我们拿到这个源代码之后就可以使用熟悉的BeautifulSoup库来查找目标标签。

代码示例及解析：

```
from selenium import webdriver
from bs4 import BeautifulSoup
import time

driver = webdriver.Chrome()
driver.get('https://y.qq.com/n/yqq/song/000xdZuV2LcQ19.html') # 访问页面
time.sleep(2)

button = driver.find_element_by_class_name('js_get_more_hot') # 根据类名找到【点击加载更多】
button.click() # 点击
time.sleep(2) # 等待两秒

pageSource = driver.page_source # 获取Elements中渲染完成的网页源代码
soup = BeautifulSoup(pageSource,'html.parser')  # 使用bs解析网页
comments = soup.find('ul',class_='js_hot_list').find_all('li',class_='js_cmt_li') # 使用bs提取
print(len(comments)) # 打印comments的数量

for comment in comments: # 循环
    sweet = comment.find('p') # 提取评论
    print ('评论：%s\n ---\n'%sweet.text) # 打印评论
driver.close() # 关闭浏览器 # 关闭浏览器
```

第13行我们通过浏览器对象.page_source来获取经过渲染的网页源代码，然后14行将源代码作为参数进行BeautifulSoup类的实例化。

三、selenium小结

- **浏览器对象**

属性/方法	功能	参数	返回值
page_source	获取网页源代码	无	无
get()	打开网页	一个URL	无
close()	关闭网页	无	无
find_element_by_class_name()	定位标签	标签中class属性对应的值	WebElement对象

find_element_by_id()		标签中id属性对应的值	
find_element_by_tag_name()		标签名称	
find_element_by_link_text()		标签中href属性对应的链接	
find_elements_by_class_name()	定位相同名称或属性的一组标签	标签中class属性对应的值	WebElement对象组成的列表
find_elements_by_id()		标签中id属性对应的值	
find_elements_by_tag_name()		标签名称	
find_elements_by_link_text()		标签中href属性对应的链接	

- **WebElement对象**

属性/方法	功能	参数	返回值
text	显示其文本内容	无	无
send_keys()	在输入框输入文字	需要输入的字符串	无
click()	点击按钮	无	无
get_attribute()	获取查找到标签的内部信息	标签的属性名的字符串，如'href'等	属性名对应属性值的字符串形式