UI 自动化 day04 随堂笔记

今日内容

- 下拉框/弹窗/滚动条
- frame 框架/多窗口
- 截图操作/验证码的处理

下拉框

传统处理方案

```
"""
下拉框处理：传统方案
"""
# 1. 导入模块
from time import sleep

from selenium import webdriver

# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('file:///Users/comesoon/Desktop/page/%E6%B3%A8%E5%86%8CA.html')

# 需求：使用‘注册A.html’页面，完成对城市的下拉框的操作
# 1).选择‘广州’
driver.find_element_by_css_selector('[value="gz"]').click()
sleep(2)
```

```
# 2).暂停2秒，选择‘上海’
driver.find_element_by_css_selector('[value="sh"]').click()
sleep(2)
# 3).暂停2秒，选择‘北京’
driver.find_element_by_css_selector('[value="bj"]').click()

# 结论：可以通过直接定位下拉框中的内容对应的元素，完成对下拉框元素的处理!

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
driver.quit()
```

使用 Select 类处理

```
"""
下拉框操作：Select 类的使用
"""
# 1. 导入模块
from time import sleep

from selenium import webdriver
from selenium.webdriver.support.select import Select

# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('file:///Users/comesoon/Desktop/page/%E6%B3%A8%E5%86%8CA.html')
```

```python
# 需求：使用‘注册A.html’页面，完成对城市的下拉框的操作
# 0> 定位下拉框元素
sel = driver.find_element_by_id('selectA')
# 1> 实例化下拉框选择对象
se = Select(sel)
# 1).选择‘广州’
# 2> 通过索引选择目标元素(索引从 0 开始)
se.select_by_index(2)
sleep(2)
# 2).暂停2秒，选择‘上海’
# 2> 通过 value 属性值选择目标元素
se.select_by_value('sh')
sleep(2)
# 3).暂停2秒，选择‘北京’
# 2> 通过可见文本信息选择目标元素
se.select_by_visible_text('A北京')

# 注意:
# 1. 如果页面内需要操作的下拉框元素有多个，需要根据目标下拉框，依次实例化下拉框选择对象
# 2. 根据具体需求，三种下拉框内容元素选择方法，任选其一使用即可!!!!!!

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
driver.quit()
```

弹出框

应用场景：页面操作过程中，一旦出现弹窗，如果不进行处理，则后续操作无法执行!
弹窗分类：1> 系统弹窗(JS 实现) 2> 自定义弹窗(前端代码封装)

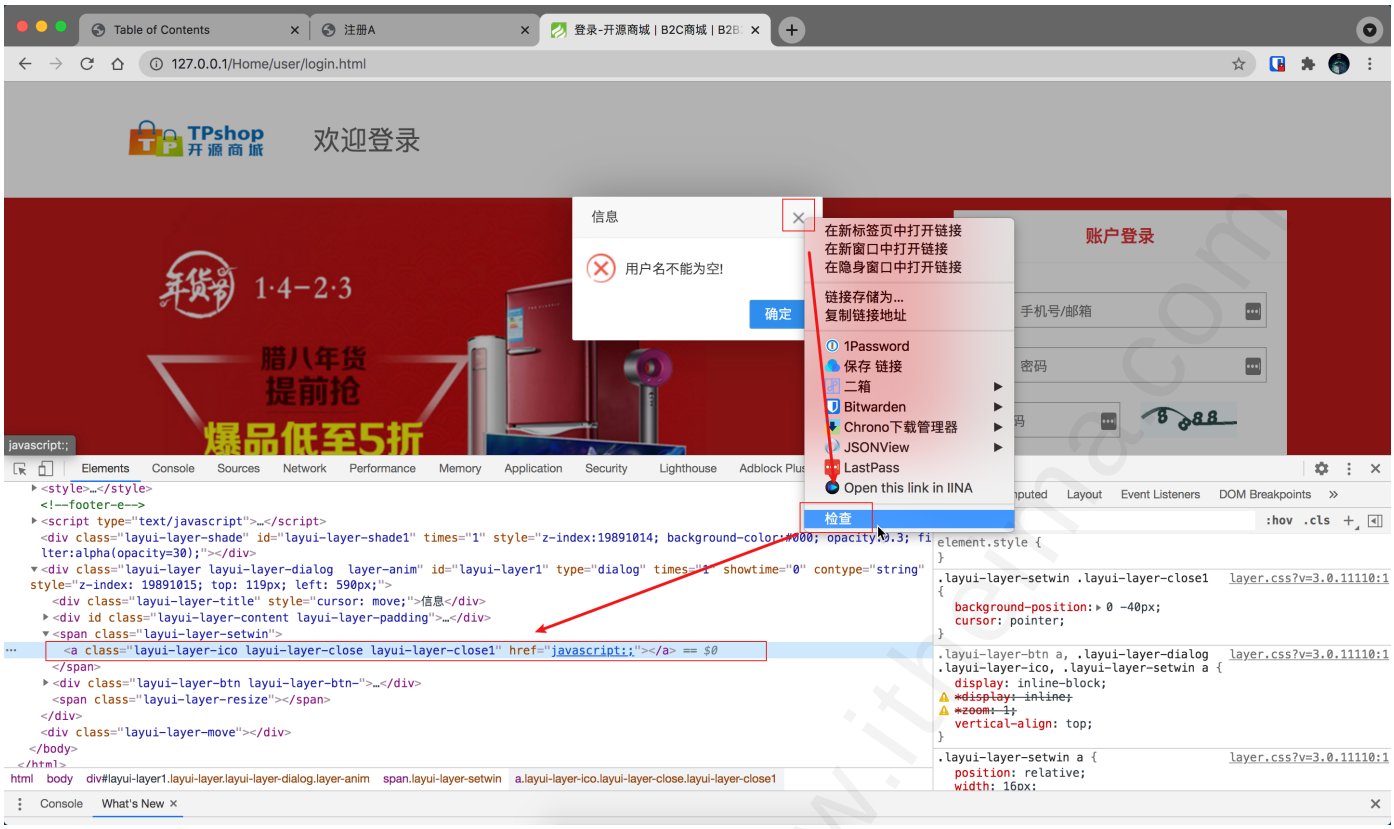
系统弹窗处理方法

```
"""
弹出框处理：系统弹窗处理
"""
# 1. 导入模块
from time import sleep

from selenium import webdriver

# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('file:///Users/comesoon/Desktop/page/%E6%B3%A8%E5%86%8CA.html')

# 需求：打开注册A.html页面，完成以下操作：
# 1).点击 alert 按钮
# 注意：凡是通过 JS 实现的系统弹窗，无法通过鼠标右键检查选项获取元素信息!!!!!!
# driver.find_element_by_id('alerta').click()  # 警告框
# driver.find_element_by_id('confirma').click()  # 确认框
driver.find_element_by_id('prompta').click()  # 提示框

# 2).关闭警告框
```

```
# 1> 切换到弹窗：只要是系统弹窗，无论哪一种，处理方法都是以下的步骤!!!!!!
alert = driver.switch_to.alert
# 2> 获取弹窗信息(可选)：获取弹窗信息必须在处理弹窗操作之前!
print('弹窗信息是:', alert.text)
sleep(2)
# 3> 去除弹窗(同意/移除)
alert.accept()  # 同意
# alert.dismiss()  # 移除

# 3).输入用户名：admin
driver.find_element_by_id('userA').send_keys('admin')

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
driver.quit()
```

自定义弹窗处理方法

说明：由于自定义弹窗可以通过鼠标右键的检查选项获取元素信息，因此出现自定义弹窗时，直接定义目标元素并操作，移除弹窗即可



滚动条

应用场景：

1．在HTML页面中，由于前端技术框架的原因，页面元素为动态显示，元素根据滚动条的下拉而被加载

2．页面注册同意条款，需要滚动条到最底层，才能点击同意

处理案例演示

```
"""
滚动条处理
"""
# 1. 导入模块
from time import sleep

from selenium import webdriver
```

```
# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('file:///Users/comesoon/Desktop/page/%E6%B3%A8%E5%86%8CA.html')

# 需求：打开注册页面A，暂停2秒后，滚动条拉到最底层，暂停 2 秒后，恢复原位置
# 注意：Selenium 框架中没有专门处理滚动条的方法，需要通过调用 JS 代码实现操作
sleep(2)
# 1> 准备 JS 代码：1000 只是一个尽可能大的值，不是准确值
js_down = "window.scrollTo(0,1000)"
# 2> 执行 JS 代码
driver.execute_script(js_down)
sleep(2)
# 向上：反向只需要将坐标归零即可
js_up = "window.scrollTo(0,0)"
driver.execute_script(js_up)

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
driver.quit()
```

---

frame 处理

应用场景：处于 frame 中的元素，虽然可以获取元素信息，但是代码执行时无法定位元素，因此需要先执行 frame 切换，才能进行元素定位操作
常见标签：1> frameset(已经不常用) 2> iframe(常见)

切换 frame 操作

```
"""
frame 操作：切换 frame
"""
# 1. 导入模块
from time import sleep

from selenium import webdriver

# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('file:///Users/comesoon/Desktop/page/%E6%B3%A8%E5%86%8C%E5%AE%9E%E4%BE%8B.html')

# 案例：打开‘注册实例.html’页面，完成以下操作
# 1). 填写主页面的注册信息
driver.find_element_by_id('user').send_keys('admin')
driver.find_element_by_id('password').send_keys('123456')

# 2). 填写注册页面A中的注册信息
# 说明：如果目标元素存在于 frame 中，就需要先执行切换 frame 操作，再定位元素
# 切换 frame：传入的是能够代表 frame 元素唯一性的特征值
driver.switch_to.frame('idframe1')
driver.find_element_by_id('userA').send_keys('admin1')
driver.find_element_by_id('passwordA').send_keys('123456')
```

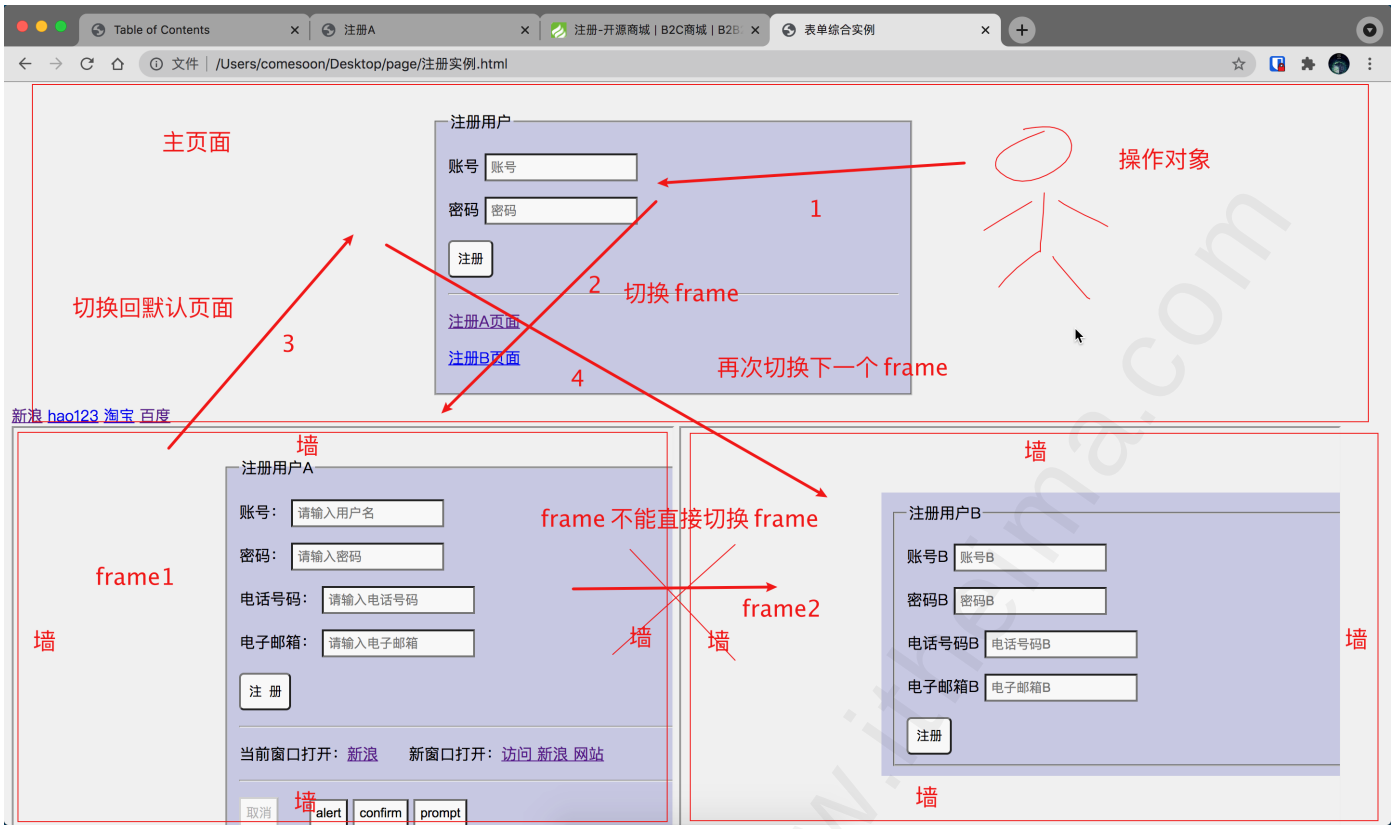
```
# 3). 填写注册页面B中的注册信息

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
driver.quit()
```

连续切换 frame

```
"""
frame 操作: 从 frame 切换到 frame
"""
# 1. 导入模块
from time import sleep

from selenium import webdriver

# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('file:///Users/comesoon/Desktop/page/%E6%B3%A8%E5%86%
8C%E5%AE%9E%E4%BE%8B.html')

# 案例: 打开‘注册实例.html’页面, 完成以下操作
# 1). 填写主页面的注册信息
driver.find_element_by_id('user').send_keys('admin')
driver.find_element_by_id('password').send_keys('123456')

# 2). 填写注册页面A中的注册信息
```

```
# 说明：如果目标元素存在于 frame 中，就需要先执行切换 frame 操作，再定位元素
# 切换 frame：传入的是能够代表 frame 元素唯一性的特征值
driver.switch_to.frame('idframe1')
driver.find_element_by_id('userA').send_keys('admin1')
driver.find_element_by_id('passwordA').send_keys('123456')

# 切换回默认页面：如果连续切换多个 frame，必须先回到默认页面，才能实现切换下一个 frame
driver.switch_to.default_content()

# 切换 frame
driver.switch_to.frame('myframe2')

# 3). 填写注册页面B中的注册信息
driver.find_element_by_id('userB').send_keys('admin2')
driver.find_element_by_id('passwordB').send_keys('123456')

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
driver.quit()
```

说明: 多个 frame 切换的执行原理



多窗口处理

应用场景：想要操作新开页面内的元素时，必须先要完成窗口切换，才能获取目标元素

注意：切换窗口操作需要依赖窗口的句柄值才能实现

案例实现

```
"""
多窗口操作
"""
# 1. 导入模块
from time import sleep

from selenium import webdriver
```

```
# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('file:///Users/comesoon/Desktop/page/%E6%B3%A8%E5%86%
8C%E5%AE%9E%E4%BE%8B.html')

# 需求：打开‘注册实例.html’页面，完成以下操作
# 1). 点击‘注册A页面’链接
driver.find_element_by_id('ZCA').click()

# 查看当前窗口的句柄值
# 说明：在浏览器的一个生命周期内(开启到关闭)，
# 任意一个窗口都有唯一的一个句柄值，可以通过句柄值完成窗口切换操作
print('当前句柄值为:', driver.current_window_handle)

# 切换窗口操作
# 1> 获取所含有窗口句柄(包含新窗口)
handles = driver.window_handles
print(handles)
print(type(handles))

# 2> 切换窗口：列表的 -1 索引对应的值，始终是最新窗口的句柄值
driver.switch_to.window(handles[-1])

# 2). 在打开的页面中，填写注册信息
driver.find_element_by_id('userA').send_keys('admin')
driver.find_element_by_id('passwordA').send_keys('123456')

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
driver.quit()
```

注意事项

```
"""
多窗口操作
"""
# 1. 导入模块
from time import sleep

from selenium import webdriver

# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('file:///Users/comesoon/Desktop/page/%E6%B3%A8%E5%86%
8C%E5%AE%9E%E4%BE%8B.html')

# 需求：打开‘注册实例.html’页面，完成以下操作
# 1). 点击‘注册A页面’链接
driver.find_element_by_id('ZCA').click()

# 查看当前窗口的句柄值
# 说明：在浏览器的一个生命周期内(开启到关闭)，
# 任意一个窗口都有唯一的一个句柄值，可以通过句柄值完成窗口切换操作
print('当前句柄值为:', driver.current_window_handle)

# 切换窗口操作
# 1> 获取所含有窗口句柄(包含新窗口)
handles = driver.window_handles
# print(handles)
# print(type(handles))

# 2> 切换窗口：列表的 -1 索引对应的值，始终是最新窗口的句柄值
```

```
driver.switch_to.window(handles[-1])

# 2). 在打开的页面中，填写注册信息
driver.find_element_by_id('userA').send_keys('admin1')
driver.find_element_by_id('passwordA').send_keys('123456')

sleep(2)
# 确认 close() 方法的作用：关闭当前页面
driver.close()

# 注意：如果还想重新操作原始页面，务必要先再完成窗口切换
driver.switch_to.window(handles[0])

# 填写原始页面信息
driver.find_element_by_id('user').send_keys('admin')
driver.find_element_by_id('password').send_keys('123456')

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
driver.quit()
```

截图操作

应用场景：在自动化测试中，出现错误时，可以通过截图，辅助判定错误原因

案例实现

```
"""
截图操作
```

```
"""
# 1. 导入模块
from time import sleep

from selenium import webdriver

# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('file:///Users/comesoon/Desktop/page/%E6%B3%A8%E5%86%
8CA.html')

# 需求：打开‘注册A.html’页面，完成以下操作
# 1). 填写注册信息
driver.find_element_by_id('userA').send_keys('admin')
driver.find_element_by_id('passwordA').send_keys('123456')

# 2). 截图保存：默认推荐使用 .png 格式
# driver.get_screenshot_as_file('./info.png')
# 说明：.jpg 虽然可以使用，但是会有使用警告
# driver.get_screenshot_as_file('./info.jpg')
# 扩展：指定图片文件存放路径：需要先手动创建文件夹!!!!!!
driver.get_screenshot_as_file('./image/info.png')

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
driver.quit()
```

扩展: 利用时间戳防止图片重名被覆盖

```
"""
截图操作
"""
# 1. 导入模块
from time import sleep, strftime

from selenium import webdriver

# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('file:///Users/comesoon/Desktop/page/%E6%B3%A8%E5%86%
8CA.html')

# 需求：打开‘注册A.html’页面，完成以下操作
# 1). 填写注册信息
driver.find_element_by_id('userA').send_keys('admin')
driver.find_element_by_id('passwordA').send_keys('123456')

# 扩展：使用时间戳防止文件重名被覆盖
# 说明：Windows 系统文件名不支持特殊符号，尽量只使用下划线，否则可能
无法生存截图文件
now_time = strftime('%Y%m%d_%H%M%S')  # %Y年%m月%d日%H时%M分%S秒
driver.get_screenshot_as_file('./info_{}.png'.format(now_time))

# 扩展：给元素截图
btn = driver.find_element_by_tag_name('button')
btn.screenshot('./btn.png')

# 4. 展示效果
sleep(3)
```

```
# 5. 退出浏览器
driver.quit()
```

验证码

说明：一种随机生成的信息（数字、字母、汉字、图片、算术题）等为了防止恶意的请求行为，增加应用的安全性。
注意：在自动化测试过程中，必须处理验证码，否则无法继续执行后续测试

面试题: 常见的验证码处理策略

1). 去掉验证码[常用]：测试环境下-采用
2). 设置万能验证码[常用]：生产环境和测试环境下-采用，必须注意保密不要外泄万能验证码
3). 验证码识别技术：通过 Python-tesseract 来识别图片类型验证码；识别率很难达到 100%
4). 记录 cookie：通过记录 cookie 进行跳过登录，不能对登陆本身功能进行测试时使用

---

cookie 绕过原理

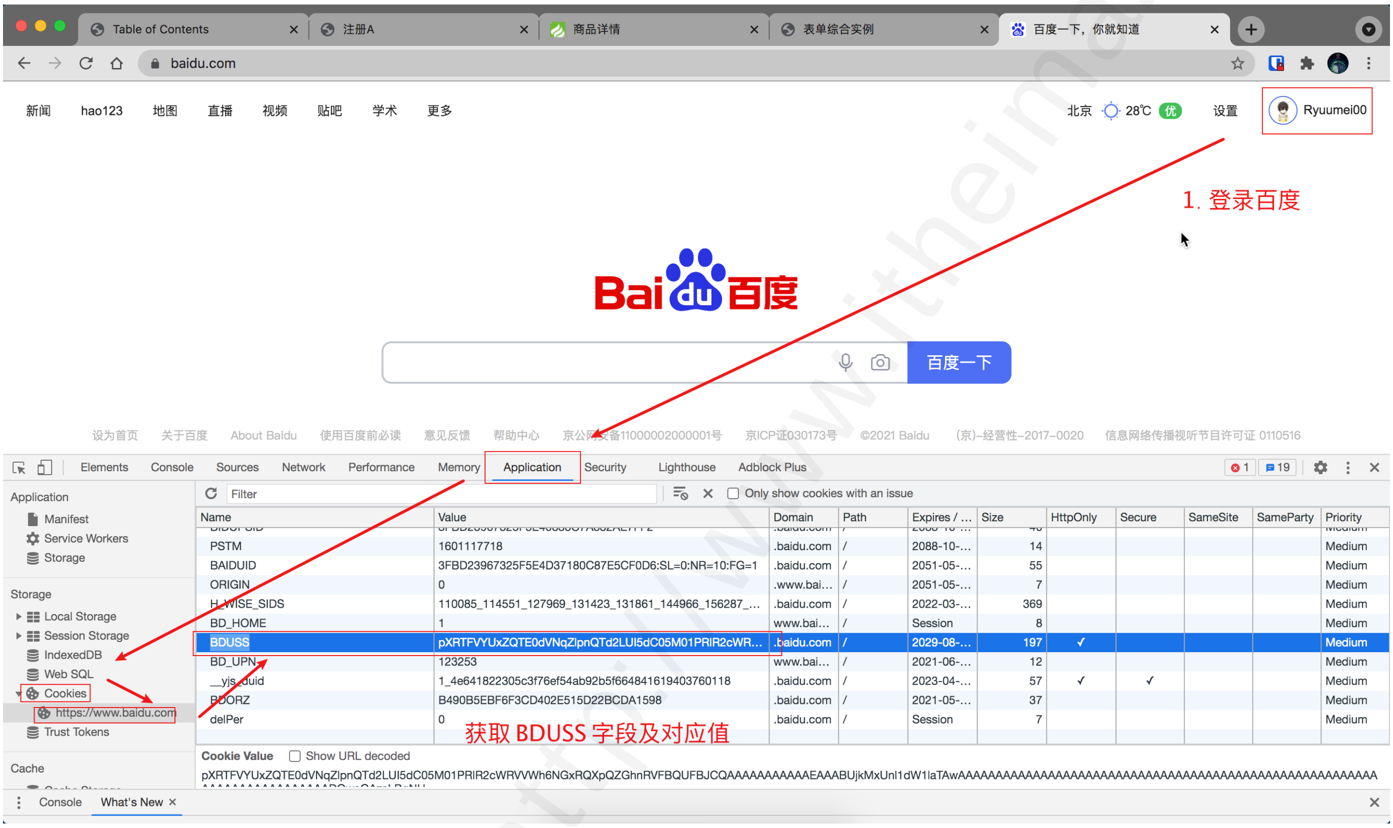
说明：客户端登录账号后，将登录状态相关的 cookie 信息发送给服务器保存，在发送请求，携带的 cookie 信息如果和服务器保留的一致，则服务器任务客户端为登录状态
应用场景：

1. 实现会话跟踪，记录用户登录状态
2. 实现记住密码和自动登录的功能
3. 用户未登录的状态下，记录购物车中的商品

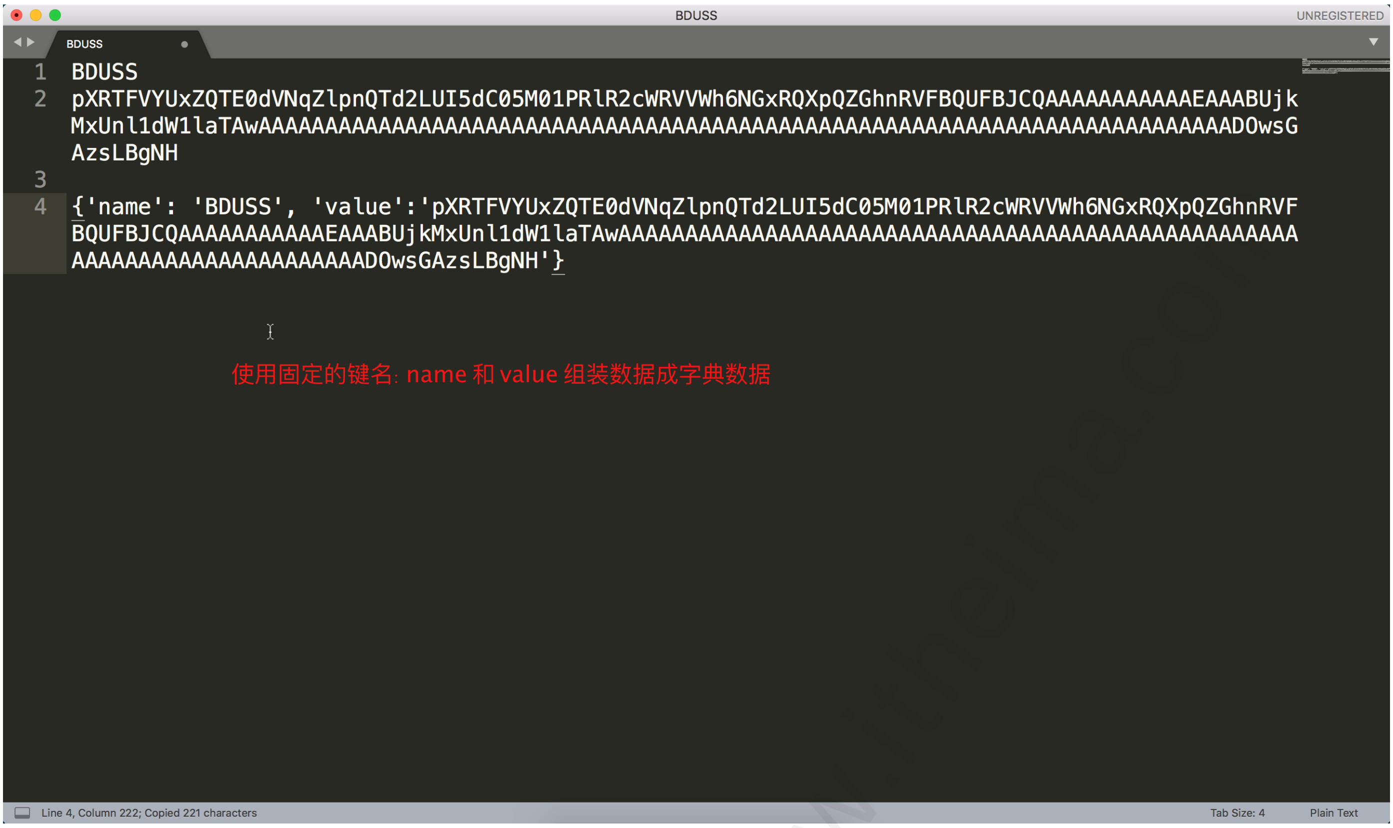
案例实现

准备工作

> 登录账号获取关键 cookie 字段信息



> 整理数据为字典数据



代码实现

```
"""
cookie: 绕过登录操作
"""
# 1. 导入模块
from time import sleep

from selenium import webdriver

# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('http://www.baidu.com')
driver.maximize_window()  # 窗口最大化

# cookie 绕过登录
# 1> 整理关键 cookie 信息为字典数据
```

```
# 注意: 字典数据的 key 必须是 name 和 value!!!!!!
cookie_value = {'name': 'BDUSS',
                'value':
'pXRTFVYUxZQTE0dVNqZlpnQTd2LUI5dC05M01PRlR2cWRVVWh6NGxRQXpQZGhnR
VFBQUFBJCQAAAAAAAAAAAEAAABUjkMxUnl1dW1laTAwAAAAAAAAAAAAAAAAAAAA
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAADOws'}

# 2> 调用方法添加 cookie 信息
driver.add_cookie(cookie_value)

# 3> 刷新页面 -> 发送 cookie 信息给服务器进行验证!!!!!!
driver.refresh()

# 注意:
# 1. 本地浏览器中登录的账号, 不能退出, 否则 cookie 信息过期, 需要重新
获取
# 2. 不同项目的能够进行登录功能绕过的 cookie 字段信息都不相同, 具体需
要询问开发
# 3. 利用 cookie 绕过登录, 则不能对登录功能本身进行测试

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
driver.quit()
```

其他方法

```
"""
cookie: 其他方法
"""
# 1. 导入模块
```

```python
from time import sleep

from selenium import webdriver

# 2. 实例化浏览器对象
driver = webdriver.Chrome()
# 3. 打开页面
driver.get('http://www.baidu.com')
driver.maximize_window()  # 窗口最大化

# 获取所有 cookie 信息
print(driver.get_cookies())

# 获取特定 cookie 信息: 需要传入 cookie 名
print(driver.get_cookie('BDUSS'))

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
driver.quit()
```

今天任务

- 整理今日内容
- 完成今日练习 -> 代码截图提交 -> 登录 TPShop 商城, 添加收货地址
- 扩展题目: 登录 TPShop 商城, 修改用户头像[选做]
- 预习内容: PyTest 框架&PO 模式一部分
- 安装命令: `pip install -i https://pypi.tuna.tsinghua.edu.cn/simple pytest`