ui自动化测试day03(web自动化测试)

每日反馈：

1、方法比较多记不住　　就是多练习多写代码

2、上课给的练习时间少，　尽量多给一些练习时间，但是也不能给太长。

3、环境和测试流程在app阶段会有相关的课程知识点讲解。

3、获取元素信息（接day02）

- 为什么要学习获取元素信息的方法

  主要为了获取相关的信息进行断言，判断自动化用例最终的执行结果。

- 获取元素常用的方法：
  - size　获取元素的大小　返回的是一个字典，里面包含 元素高度和宽度的值
  - text　获取元素的文本内容
  - get_attribute("attribute") 获取元素对应属性名称的属性值，　attribute表示的是属性名

```
# 导包
import time

from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver.common.by import By
driver = webdriver.Chrome()
```

```
driver.maximize_window()
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%E5%90%88/pagetest/%E6%B3%A8%E5%86%8CA.html")
# 1).获取用户名输入框的大小
print(driver.find_element(By.ID, "userA").size)
# 2).获取页面上第一个超链接的文本内容
print(driver.find_element(By.LINK_TEXT, "新浪").text)
# 3).获取页面上第一个超链接的地址
print(driver.find_element(By.LINK_TEXT, "新浪").get_attribute("href"))
# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

- is_displayed()　　判断元素是否可见　返回值为true或者false
- is_enabled()　　判断元素是否可用，返回值为true或者false
- is_selected()　　判断复选框或者单选框是否被选中，返回值为true或者false

```
# 导包
import time
from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver.common.by import By
driver = webdriver.Chrome()
driver.maximize_window()
# 打开测试网站
```

```
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%E5%90%88/pagetest/%E6%B3%A8%E5%86%8CA.html")
# 4).判断页面中的span标签是否可见
print("判断元素是否可见，默认应该是False：",
driver.find_element(By.NAME,
"sp1").is_displayed())
# 5).判断页面中取消按钮是否可用
print("判断取消按钮是否可用，默认应该是False：",
driver.find_element(By.ID,
"cancelA").is_enabled())
# 6).判断页面中'旅游'对应的复选框是否为选中的状态
print("判断旅游复选框是否选中，默认应该是True:",
driver.find_element(By.ID, "lyA").is_selected())
# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

一、鼠标和键盘操作

1、鼠标操作

1.1 鼠标操作实现方式

- selenium提供鼠标操作的方法及步骤

  需要导入ActionChains类

  - 通过ActionChains实例化鼠标对象　action = ActionChains(driver)　# driver表示的是浏览器驱动对象
  - 调用鼠标的事件方法
  - 调用鼠标的执行方法　action.perform()

1.2 鼠标右击操作

- 右击操作的实现步骤:

  针对由html自定义的右键菜单。可以使用右击的方式来进行操作。

  - 创建鼠标对象　action = ActionChains(driver)
  - 调用右击事件方法 action.context_click(element) # element表示的是一个元素对象
  - 调用鼠标执行方法 action.perform()

```
# 导包
import time
from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver import ActionChains
from selenium.webdriver.common.by import By
driver = webdriver.Chrome()
driver.maximize_window()
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%E5%90%88/pagetest/%E6%B3%A8%E5%86%8CA.html")

# 在用户名文本框上点击鼠标右键
# 创建鼠标对象
action = ActionChains(driver)
# 调用鼠标右击的方法
action.context_click(driver.find_element(By.ID, "userA"))
# 调用鼠标执行的方法
action.perform()
# 等待3S
time.sleep(3)
```

```
# 退出浏览器驱动
driver.quit()
```

1.3 鼠标双击操作

- 鼠标双击的实现步骤
  - 创建鼠标对象 action=ActionChains(driver)
  - 调用鼠标双击事件方法 action.double_click(element) # element表示是元素对象
  - 调用鼠标的执行方法 action.perform()

```
# 导包
import time
from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver import ActionChains
from selenium.webdriver.common.by import By
driver = webdriver.Chrome()
driver.maximize_window()
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%E5%90%88/pagetest/%E6%B3%A8%E5%86%8CA.html")

# 输入用户名admin，暂停3秒钟后，双击鼠标左键，选中admin
element = driver.find_element(By.ID, "userA")
element.send_keys("admin")
time.sleep(3)
# 创建鼠标对象
action = ActionChains(driver)
# 调用鼠标双击事件方法
action.double_click(element)
```

```
# 调用鼠标执行方法
action.perform()

# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

1.4 鼠标拖动操作

- 鼠标拖动的实现步骤：
  - 创建鼠标对象 action = ActionChains(driver)
  - 调用鼠标拖动事件方法 action.drag_and_drop(source, target) # source表示的是源元素，被拖动的元素， target表示是目标源，也就是要拖动到哪个元素上。
  - 调用鼠标执行方法 action.perform()

```
# 导包
import time
from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver import ActionChains
from selenium.webdriver.common.by import By
driver = webdriver.Chrome()
driver.maximize_window()
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%E5%90%88/pagetest/drag.html")

# 把红色方框拖拽到蓝色方框上
source = driver.find_element(By.ID, "div1")
target = driver.find_element(By.ID, "div2")
```

```
# 实例化鼠标对象
action = ActionChains(driver)
# 调用鼠标拖动事件方法
action.drag_and_drop(source, target)
# 调用鼠标执行方法
action.perform()

# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

1.5 鼠标悬停操作

- 鼠标悬停实现步骤：
  - 实例化鼠标对象 action = ActionChains(driver)
  - 调用鼠标悬停事件方法 action.move_to_element(element) # element表示的是元素对象
  - 调用鼠标执行方法 action.perform()

```
# 导包
import time
from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver import ActionChains
from selenium.webdriver.common.by import By
driver = webdriver.Chrome()
driver.maximize_window()
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%E5%90%88/pagetest/%E6%B3%A8%E5%86%8CA.html")
```

```
# 模拟鼠标悬停在‘注册’按钮上
element = driver.find_element(By.CSS_SELECTOR,
"button")

# 创建鼠标对象
action = ActionChains(driver)
# 调用鼠标悬停事件方法
action.move_to_element(element)
# 调用鼠标执行方法
action.perform()

# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

1.6 鼠标单元素拖动操作

- 鼠标单元素拖动实现步骤
  - 创建鼠标对象 action=ActionChains(driver)
  - 调用鼠标单元素拖动事件方法 action.drag_and_drop_by_offset(element, x, y) x, y 表示的元素拖动时横向和纵向移动的距离，单位为像素，element表示的是元素对象 移动的像素最终要比在web页面中看到的移动像素值要大，最好大于5个像素或者10像素
  - 调用鼠标执行方法 action.perform()

```
# 导包
import time

from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver import ActionChains
from selenium.webdriver.common.by import By
```

```
driver = webdriver.Chrome()
driver.maximize_window()
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%E5%90%88/pagetest/%E9%AA%8C%E8%AF%81%E7%A0%81/index.html")

# 模拟鼠标实现滑块验证码的操作
element = driver.find_element(By.CSS_SELECTOR, '.handler_bg')
# 创建鼠标对象
action = ActionChains(driver)
# 调用鼠标单元素拖动事件方法
action.drag_and_drop_by_offset(element, 265, 0)
# 调用鼠标执行方法
action.perform()

# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

2、键盘操作

- 模拟键盘上面的快捷键的操作
- 调用键盘操作的快捷键的方法 element.send_keys(快捷键的键值)

  需要导入Keys类, 第一个字母是大写

  单键值： 直接传入对应的键值

组合键： 键值之间由逗号分隔

send_keys(Keys.CONTROL，Keys.SHIFT，'i')

常用的快捷键

1. send_keys(Keys.BACK_SPACE) 删除键(BackSpace)
2. send_keys(Keys.SPACE) 空格键(Space)
3. send_keys(Keys.TAB) 制表键(Tab)
4. send_keys(Keys.ESCAPE) 回退键(Esc)
5. send_keys(Keys.ENTER) 回车键(Enter)
6. send_keys(Keys.CONTROL,'a') 全选(Ctrl+A)
7. send_keys(Keys.CONTROL,'c') 复制(Ctrl+C)
8. send_keys(Keys.CONTROL, 'v') 粘贴

```
# 导包
import time

from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver import ActionChains
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys

driver = webdriver.Chrome()
driver.maximize_window()
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%E5%90%88/pagetest/%E6%B3%A8%E5%86%8CA.html")
# 1). 输入用户名：admin1，暂停2秒，删除1
element = driver.find_element(By.ID, 'userA')
element.send_keys("admin1")
time.sleep(2)
```

```
element.send_keys(Keys.BACK_SPACE)  # 删除最后一个字符串 clear()
# 2). 全选用户名：admin，暂停2秒
element.send_keys(Keys.CONTROL, 'a')
time.sleep(2)
# 3). 复制用户名：admin，暂停2秒
element.send_keys(Keys.CONTROL, 'c')
time.sleep(2)
# 4). 粘贴到密码框
driver.find_element(By.ID, 'passwordA').send_keys(Keys.CONTROL, 'V')

# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

二、元素等待

HTML加载需要时间，影响HTML加载的因素：

- 服务器性能
- 网络速度
- 本身电脑的配置

1、隐式等待

概念：首先要等待整个页面加载完成，再去进行元素定位，如果在定位过程中找到了元素，直接返回该元素，继续后面的操作，如果在指定的时间内没有找到该元素，那么每隔0.5秒再去找，如果超过了指定时间，就会抛出NoSuchElementException的异常错误。

隐式等待实现方式:

driver.implicitlty_wait(timeout) # timeout表示的是最长的等待时间 单位为S

隐式等待只需要设置一次，对所有的元素定位的方法都是有效的。

```
# 导包
import time

from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver.common.by import By
driver = webdriver.Chrome()
driver.maximize_window()
driver.implicitly_wait(5)  # 隐式等待的时间，设置为5S
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%E5%90%88/pagetest/%E6%B3%A8%E5%86%8CA.html")
# 针对第一个延时框输入admin
print("开始时间：", time.strftime("%H:%M:%S"))
driver.find_element(By.XPATH,
"//div[@id='wait']/input[1]").send_keys("admin")
print("找到第一个元素的时间:",
time.strftime("%H:%M:%S"))
# 针对第二个延时框输入admin2
driver.find_element(By.XPATH,
"//div[@id='wait']/input[2]").send_keys("admin2")
print("找到第二个元素的时间：",
time.strftime("%H:%M:%S"))

# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

2、显示等待

概念：定位指定元素时，如果能找到该元素，那么就直接返回该 元素，如果找不到，那么每隔指定的时间再去定位该 元素，如果超出最长等待时间，那么就抛出TimeOutException。

显示等待的实现：

WebDriverWait(driver, timeout, poll_frequency=0.5).until(lambda x:x.find_element(By.ID, "userA"))

WebDriverWait等待类需要导入

driver指的是浏览器驱动对象

timeout表示的是最长等待时间

poll_frequency表示的是检测的间隔时间，默认是0.5和

后面跟上until方法，在until方法跟上匿名函数来实现元素定位。

显示等待与隐式等待的区别:

1、抛出的异常不一样 ， 隐式等待超时，报的NoSuchElementException ，显示等待超时，报的是TimeOutException

2、作用域不一样， 隐式等待对所有元素定位的方法都有效，只需要定义一次， 显示等待只针对单个元素

3、显示等待不需要等待整个页面的HTML的DOM树加载完成，显式等待的效率更高，工作当中一般使用显示等待。而隐式等待需整个HTML DOM树加载完成。

```
# 导包
import time
```

```
from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import
WebDriverWait

driver = webdriver.Chrome()
driver.maximize_window()
# driver.implicitly_wait(5)  # 隐式等待的时间，设置为5S
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8
%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8
%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%
E5%90%88/pagetest/%E6%B3%A8%E5%86%8CA.html")

# 通过显示等待的方式定位延时输入框输入admin
element = WebDriverWait(driver, 9, 1).until(lambda
x: x.find_element(By.XPATH, "//*
[@id='wait']/input[1]"))
element.send_keys("admin")
# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

3、强制等待

- 强制等待就是让代码休眠，不做任何的操作

  time.sleep(time)　单位为time
- 常用的场景：

- 当要获取元素的文本内容时，而元素的文本内容是需要通过后台接口请求并渲染的，此时，如果使用隐式等待或显示等待是没有办法获取到文本内容，所以需要使用强制等待
- 当要操作的元素已经存在，但是有其他的元素需要等待且与该操作的元素有业务关联，如果使用隐式等待或显示等待对该元素进行操作的话，是没有办法进行的，也需要使用强制等待。

```
# 导包
import time

from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import
WebDriverWait

driver = webdriver.Chrome()
driver.maximize_window()
driver.implicitly_wait(5)
driver.get("http://tpshop-test.itheima.net/")

# 先登录
driver.find_element(By.CSS_SELECTOR, '.red').click()
# 输入用户名密码等信息
driver.find_element(By.CSS_SELECTOR,
"#username").send_keys("13012345678")
driver.find_element(By.CSS_SELECTOR,
"#password").send_keys("12345678")
driver.find_element(By.CSS_SELECTOR,
"#verify_code").send_keys("8888")
driver.find_element(By.CSS_SELECTOR, ".J-login-
submit").click()

# 获取购物车中的商品数量
time.sleep(2)
```

```
print("购物车商品数量:",
driver.find_element(By.CSS_SELECTOR,
"#cart_quantity").text)

# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

三、下拉选择框、弹出框、滚动条操作

1、下拉选择框操作

- 下拉选 择实现步骤：

  1、导入Select类

  2、实例化select对象　　select=Select(element) # element对象表示的是select元素对象

  3、通过select的相关方法选择option选项

  - select.select_by_index(index)　参数index表示的option索引
  - select.select_by_value(value)　参数value表示的是option元属中value的属性值
  - select.select_by_visible_text(visible_text) 参数visible_text表示的是option的文本内容。

```
# 导包
import time
```

```
from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver.common.by import By
from selenium.webdriver.support.select import
Select

driver = webdriver.Chrome()
driver.maximize_window()
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A
%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/
web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%
E9%9B%86%E5%90%88/pagetest/%E6%B3%A8%E5%86%8CA.ht
ml")
element = driver.find_element(By.CSS_SELECTOR,
"#selectA")
select = Select(element)
# 通过select对象的index来选择广州
time.sleep(2)
select.select_by_index(2)
# 通过select对象的value来选择上海
time.sleep(2)
select.select_by_value("sh")
# 通过select对象的visible来选择深圳
time.sleep(2)
select.select_by_visible_text("深圳")

# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

2、弹出框操作

- 弹出框处理步骤：
  - driver.switch_to.alert 获取弹出框对象
  - 处理弹出框
    - alert.text 获取弹出框提示信息
    - alert.accept() 确定弹出框
    - alert.dismiss() 取消弹出框

```
# 导包
import time

from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
driver.maximize_window()
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%E5%90%88/pagetest/%E6%B3%A8%E5%86%8CA.html")
# 点击alert按钮
driver.find_element(By.ID, "alerta").click()
time.sleep(3)
# 处理弹出框
# 获取弹出框
alert = driver.switch_to.alert
# 打印信息，然后取消
print(alert.text)
alert.dismiss()
# 在用户名输入框中输入admin
driver.find_element(By.ID,
'userA').send_keys("admin")

# 等待3S
```

```
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

3、滚动条操作

- 滚动实现步骤：

控制滚动条到最下方

  - 1、定义js

    js = "window.scrollTo(0, 2000)" # 如果想要移动到最下方，y值给最大值就可以了。

  - 2、执行JS

    driver.execute_script(js)

```
# 导包
import time

from selenium import  webdriver
# 实例化浏览器驱动
from selenium.webdriver.common.by import By

driver = webdriver.Chrome()
driver.maximize_window()
# 打开测试网站
driver.get("file:///D:/software/UI%E8%87%AA%E5%8A%A8%E5%8C%96%E6%B5%8B%E8%AF%95%E5%B7%A5%E5%85%B7/web%E8%87%AA%E5%8A%A8%E5%8C%96%E5%B7%A5%E5%85%B7%E9%9B%86%E5%90%88/pagetest/%E6%B3%A8%E5%86%8CA.html")
time.sleep(3)
```

```
# 控制滚动条到最下方
# 1、定义js
js = "window.scrollTo(0, 2000)"
# 2、执行JS
driver.execute_script(js)

# 等待3S
time.sleep(3)
# 退出浏览器驱动
driver.quit()
```

四、frame切换、多窗口切换

1、frame切换

2、多窗口切换