UI 自动化 day02 随堂笔记

今日内容

- 剩余元素定位方法
  - 定位一组元素的方法
  - XPath 方法
  - CSS 方法

定位一组元素方法

```
# 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')

# 通过 name 属性值定位用户名和密码完成操作
driver.find_element_by_name('AAA').send_keys('admin')
# driver.find_element_by_name('AAA').send_keys('123456')
elements = driver.find_elements_by_name('AAA')
print(elements)
# 注意：元素定位方法如果带有 s，则执行结果返回的是列表类型数据，里面的
数据是多个元素对象
print(type(elements))
```

```
# 说明: 可以通过列表的下标(索引)获取对应的目标元素对象，再执行操作
elements[1].send_keys('123456')

# 其他定位方法也可以实行定义一组元素:
# 使用标签名方法定位电话和邮箱并完成操作
new_els = driver.find_elements_by_tag_name('input')
new_els[2].send_keys('13800001111')
new_els[3].send_keys('123@qq.com')

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

XPath 定位方法

```
说明:
1. XPath 即为 XML Path 的简称，它是一门在 XML 文档中查找元素信息的语言。
2. HTML 可以看做是 XML 的一种实现，所以 Selenium 用户可以使用这种强大的语言在 Web 应用中定位元素
注意: 无论使用哪一种 XPath 策略(方法)，定位方法都是同一个，不同策略只决定方法的参数的写法
方法: find_element_by_xpath('XPath策略')
```

路径策略

说明：

绝对路径：从最外层元素到指定元素之间所有经过元素层级的路径

1)．绝对路径以/html根节点开始，使用/来分隔元素层级；

如：/html/body/div/fieldset/p[1]/input

2)．绝对路径对页面结构要求比较严格，不建议使用

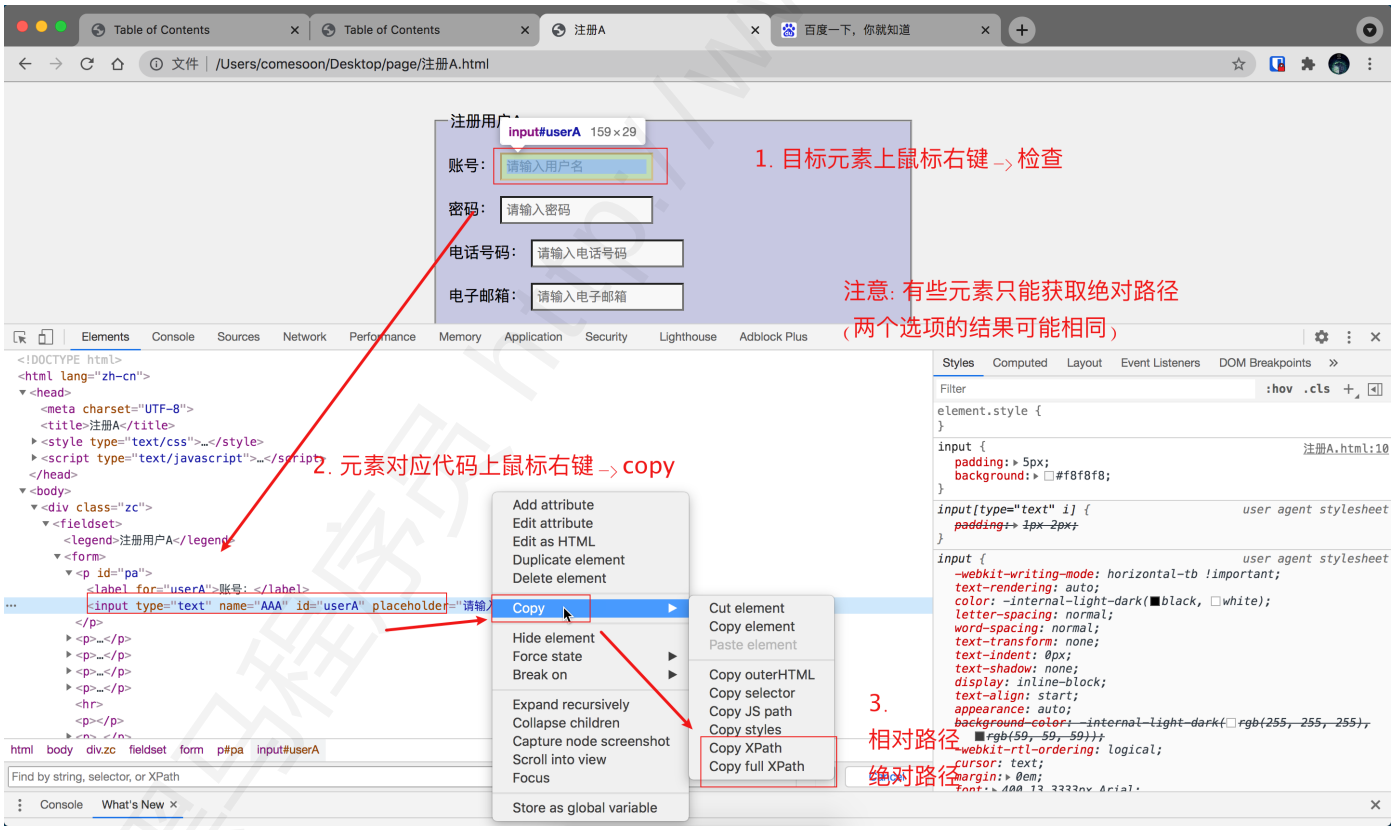
相对路径：匹配任意层级的元素，不限制元素的位置

1)．相对路径以//开始

2)．例如：//input 或者 //*

路径值获取方法

> 以谷歌浏览器为例



代码实现

```
"""
XPath:路径策略
"""
# 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).使用绝对路径定位用户名输入框，并输入：admin
driver.find_element_by_xpath('/html/body/div/fieldset/form/p[1]/input').send_keys('admin')
# 2).暂停2秒
sleep(2)
# 3).使用相对路径定位密码输入框，并输入：123
# 注意：使用相对路径时，需要注意方法参数的内外引号嵌套问题
driver.find_element_by_xpath('//*[@id="passwordA"]').send_keys('123')

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

利用元素属性策略

```
"""
XPath:利用元素属性策略
"""
# 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')

# 利用元素属性策略: 该方法可以使用目标元素的任意一个属性和属性值(需要保证唯一性)
# 语法1: //标签名[@属性名='属性值']
# 语法2: //*[@属性名='属性值']

# 需求: 打开注册A.html页面, 完成以下操作
# 1).利用元素的属性信息精确定位用户名输入框, 并输入: admin
# 注意: 使用 XPath 策略, 需要在浏览器工具中根据策略语法, 组装策略值, 验证后再放入代码中使用
# driver.find_element_by_xpath('//input[@placeholder="请输入用户名"]').send_keys('admin')
driver.find_element_by_xpath('//*[@placeholder="请输入用户名"]').send_keys('admin')

# 4. 展示效果
sleep(3)
# 5. 退出浏览器
```

```
driver.quit()
```

利用元素属性策略的注意事项

```
"""
XPath:利用元素属性策略注意事项
"""
# 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')



# //*[@属性名='属性值']
# 说明: // 任意层级 * 任意标签名

# 注意: 目标元素的有些属性和属性值, 可能存在多个相同特征的元素, 需要注
意唯一性.
driver.find_element_by_xpath('//*
[@type="text"]').send_keys('admin')

# 注意: 与 class_name 方法不同的是, 如果使用具有多个值的 class 属性,
则需要传入全部的属性值!
driver.find_element_by_xpath('//*[@class="emailA
dzyxA"]').send_keys('123@qq.com')
```

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

属性和逻辑结合

```
"""
XPath: 属性和逻辑结合
"""
# 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')

# 属性和逻辑集合: 解决目标元素单个属性和属性值无法定位为一个元素的问题时使用
# 语法: //*[@属性1="属性值1" and @属性2="属性值2"]
# 注意: 多个属性值由 and 连接, 每一个属性都要由@开头, 可以根据需求使用更多属性值

# 需求: 打开注册A.html页面, 完成以下操作
# 1).使用属性与逻辑结合定位策略, 在test1对应的输入框里输入: admin
```

```
driver.find_element_by_xpath('//*[@name="user" and
@class="login"]').send_keys('admin')

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

层级和属性结合策略

```
"""
XPath 策略:层级与属性结合
"""
# 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')

# 层级与属性结合：目标元素无法直接定位，可以考虑先定位其父层级或祖辈层
级，再获取目标元素
# 语法：父层级定位策略/目标元素定位策略

# 需求：打开注册A.html页面，完成以下操作
# 1).使用层级与属性结合定位策略，在test1对应的输入框里输入：admin
```

```
driver.find_element_by_xpath('//*
[@id="p1"]/input').send_keys('admin')

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

XPath 延伸方法

```
"""
XPath 策略: 延伸方法
"""
# 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')


# //*[contains(@属性名,'属性值的部分内容')] : 通过给定属性值的任意部
分内容进行元素定位
driver.find_element_by_xpath('//*
[contains(@id,"ss")]').send_keys('123456')
```

```
# //*[starts-with(@属性名,'属性值的开头部分内容')] : 通过给定属性值的开头部分内容进行元素定位
driver.find_element_by_xpath('//*[starts-with(@id,"te")]').send_keys('13800001111')

# //*[text()="文本信息"] : 通过文本信息定位目标元素(要求全部文本内容)
driver.find_element_by_xpath('//*[text()="访问 新浪 网站"]').click()

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

CSS 定位方法

```
说明: 通过 CSS 的选择器语法定位元素
注意:
1. Selenium 框架官方推荐使用 CSS, 因为其定位效率远高于 XPath
2. CSS 选择器策略也有很多, 但是无论哪一个策略使用的都是同一个定位方法
方法: find_element_by_css_selector('CSS策略')
```

CSS 策略: id选择器/class选择器/元素选择器/属性选择器

```
"""
CSS 策略: 前四种
"""
# 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).使用CSS定位方式中id选择器定位用户名输入框，并输入：admin
# 语法：#id属性值
driver.find_element_by_css_selector('#userA').send_keys('admin')

# 2).使用CSS定位方式中属性选择器定位密码输入框，并输入：123456
# 语法 1：[属性名="属性值"]
# 语法 2：标签名[属性名="属性值"]
# driver.find_element_by_css_selector('input[placeholder="请输入密码"]').send_keys('123456')
driver.find_element_by_css_selector('[placeholder="请输入密码"]').send_keys('123456')

# 3).使用CSS定位方式中class选择器定位电话号码输入框，并输入：18600000000
# 语法：.class属性值
driver.find_element_by_css_selector('.telA').send_keys('18600000000')
sleep(2)

# 4).使用CSS定位方式中元素选择器定位注册按钮，并点击
# 说明：元素选择器又名标签选择器
# 语法：标签名
driver.find_element_by_css_selector('button').click()
```

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

属性选择器使用注意事项

```
"""
CSS 策略: 前四种注意事项
"""
# 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')

# 注意: 与 class_name 方法不同的是, 如果使用具有多个值的 class 属性, 则需要传入全部的属性值!
driver.find_element_by_css_selector('[class="emailAdzyxA"]').send_keys('123@qq.com')

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

层级选择器

```
"""
CSS 策略:层级选择器
"""
# 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).使用CSS定位方式中的层级选择器定位用户名输入框，并输入：admin

# 层级选择器
# 父子层级关系：父层级策略>子层级策略
driver.find_element_by_css_selector('#pa>input').send_keys('admin')
# 祖辈后代层级关系：祖辈策略 后代策略
driver.find_element_by_css_selector('form [placeholder="请输入用户名"]').send_keys('admin')

# 注意：父子层级关系中也可以使用空格连接上下层级策略

# 扩展：XPath 祖辈和后代关系：只需要使用//连接祖辈和后代元素即可
```

```
driver.find_element_by_xpath('//form//*
[@id="userA"]').send_keys('admin')

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

CSS 延伸方法

```
"""
CSS 策略:延伸方法
"""
# 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')

# 标签名[属性名^="属性值开头部分内容"] : 根据给出的属性值开头部分内容
定位元素
driver.find_element_by_css_selector('[id^="pas"]').send_keys('12
3')

# 标签名[属性名$="属性值结尾部分内容"] : 根据给出的属性值结尾部分内容
定位元素
```

```
driver.find_element_by_css_selector('[id$="rdA"]').send_keys('123')

# 标签名[属性名*="属性值任意部分内容"] : 根据给出的属性值任意部分内容定位元素
driver.find_element_by_css_selector('[id*="ss"]').send_keys('123')

# 注意: 标签名可以省略!

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

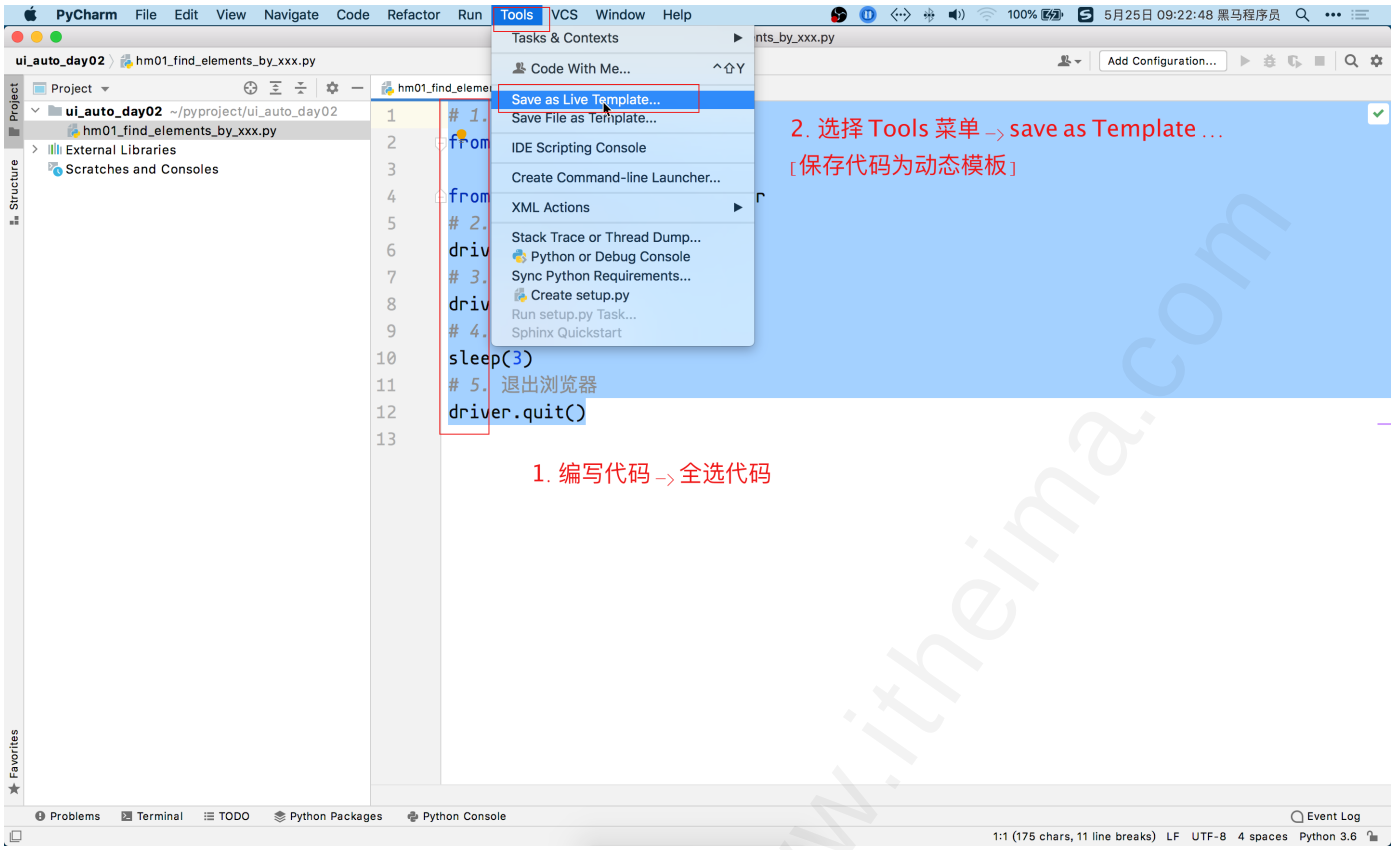
元素定位方法的另一种写法

```
# 1. 导入模块
from time import sleep

from selenium import webdriver
from selenium.webdriver.common.by import By

# 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')

# 说明: 在 Selenium 框架中, 元素定位方法的另一种写法
driver.find_element(By.ID, 'userA').send_keys('admin')
```

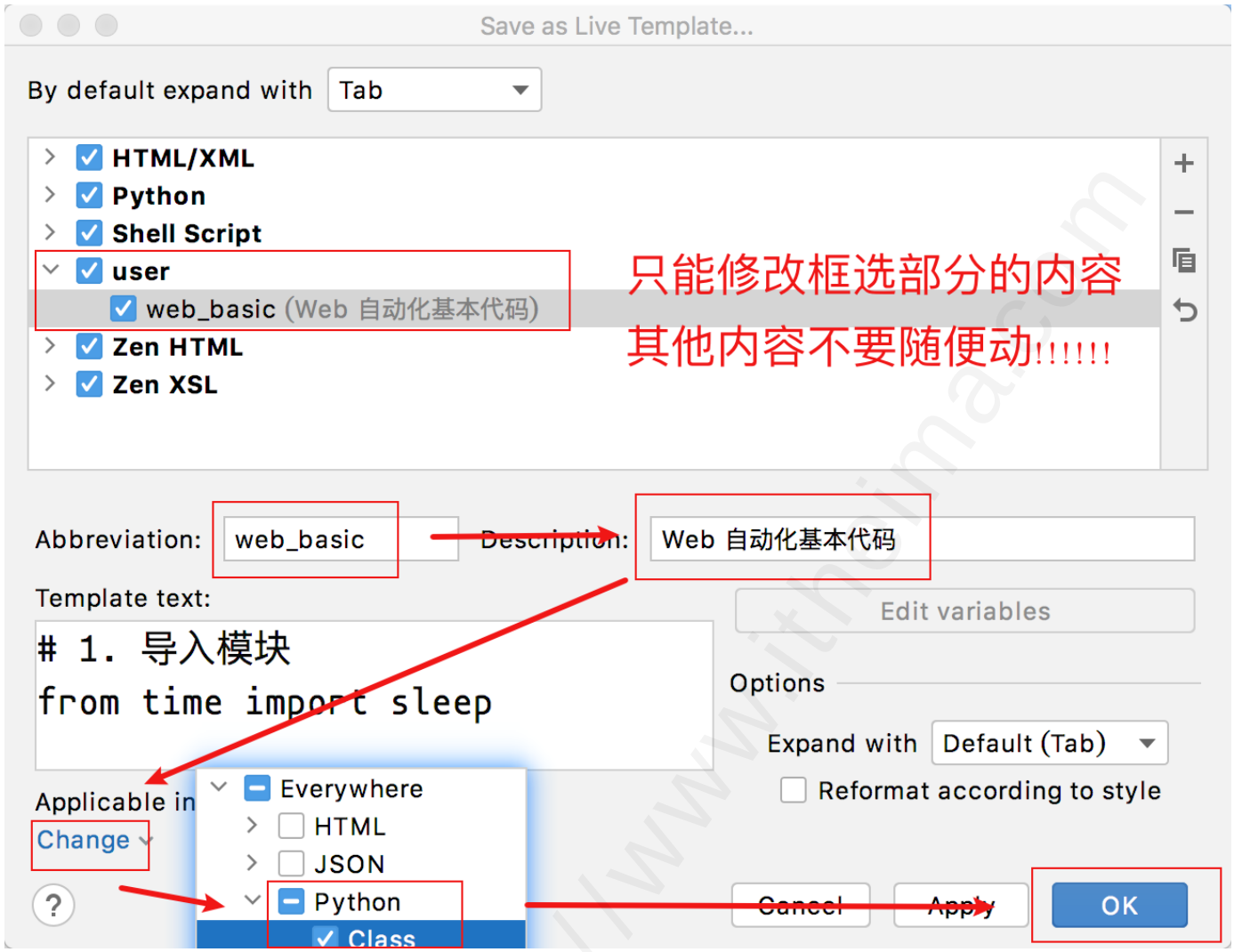
```
# 注意: By 和 by 方法可以同时使用!
driver.find_element_by_id('passwordA').send_keys('123456')

# 注释: 说法 1: by 方法是 By 方法的封装; 说法 2: By 方法是 by 方法的底层实现(原理)
# 查看原始代码方法: 按住 Ctrl 使用鼠标左键点击 by 方法即可

# 应用场景: 使用 PO 设计模式封装代码结构时需要使用 By 方法

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

扩展: PyCharm 中动态代码模板的使用步骤说明

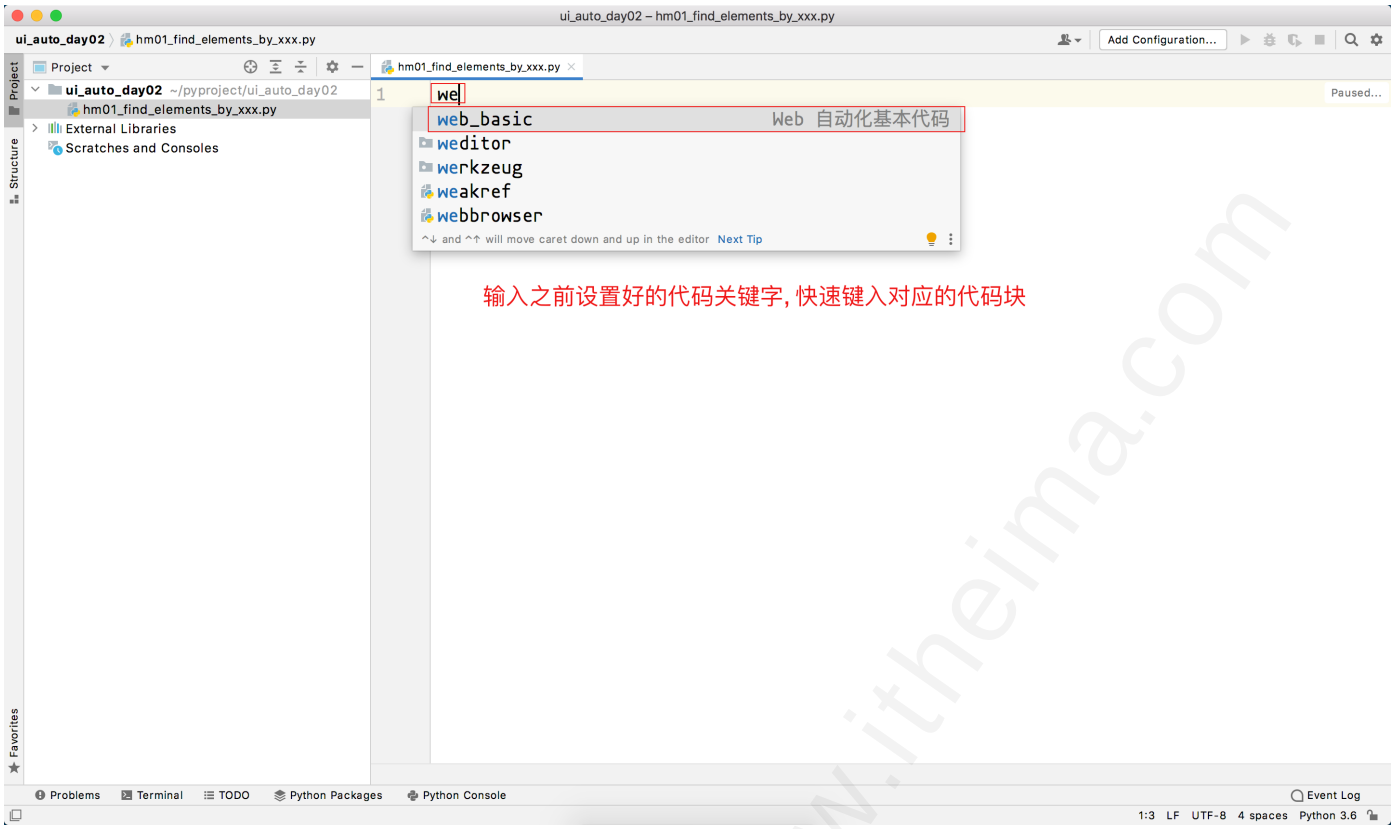
> 编写代码 -> 选中代码 -> 如图菜单选择 -> 创建动态模板



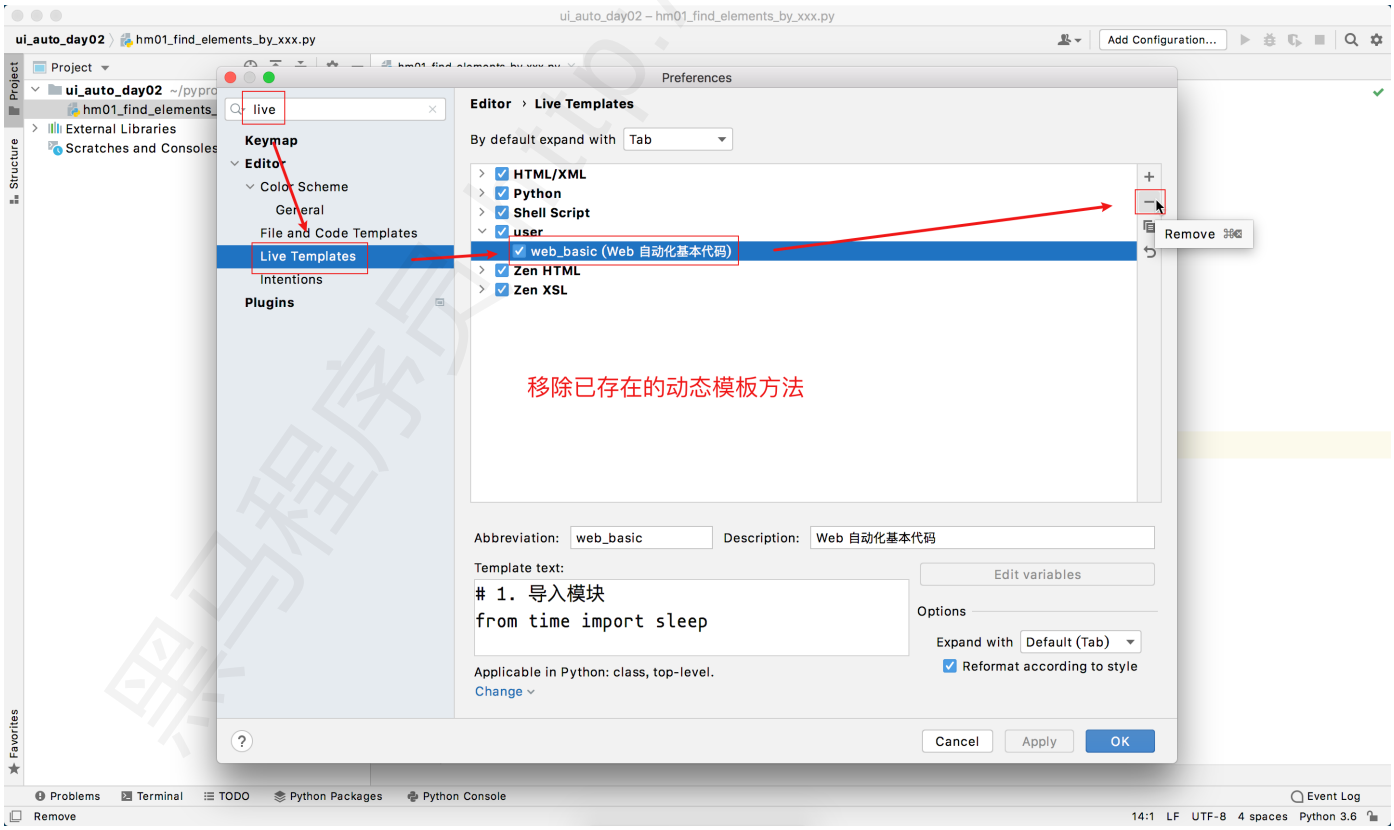
如图依次设置: 代码输入关键字 -> 描述信息 -> 语法匹配 -> 点击 OK 保存



使用步骤



删除已存在的动态模板方法(File -> Settings)



今日任务

- 总结所有元素定位方法 -> 截图提交
- 完成课后练习 -> 代码截图提交 -> 注意代码编写规范
- 预习任务 -> 截止到元素等待