SELENIUM WEBDRIVER 快速参考

浏览器自动化、元素交互、等待与断言

安装

安装

```
pip install selenium webdriver-manager
# webdriver-manager auto-downloads browser drivers
```

基础驱动设置

```
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from webdriver_manager.chrome import ChromeDriverManager
driver = webdriver.Chrome(
    service=Service(ChromeDriverManager().install()))
```

无头模式

```
options = webdriver.ChromeOptions()
options.add_argument("--headless=new")
options.add_argument("--no-sandbox")
driver = webdriver.Chrome(options=options)
```

支持的浏览器

`webdriver.Chrome()`	Google Chrome / Chromium
`webdriver.Firefox()`	Mozilla Firefox（GeckoDriver）
`webdriver.Edge()`	Microsoft Edge（Chromium）
`webdriver.Safari()`	Apple Safari（仅 macOS）

浏览器与导航

导航

```
driver.get("https://example.com")
driver.back()                # browser back
driver.forward()             # browser forward
driver.refresh()             # reload page
```

浏览器属性

`driver.title`	当前页面标题
`driver.current_url`	当前页面 URL
`driver.page_source`	完整页面 HTML 源码
`driver.get_cookies()`	列出所有 Cookie

窗口管理

```
driver.set_window_size(1920, 1080)
driver.maximize_window()
driver.minimize_window()
driver.quit()                # close all windows, end session
```

查找元素

定位策略

```
from selenium.webdriver.common.by import By
driver.find_element(By.ID, "login-btn")
driver.find_element(By.CLASS_NAME, "nav-item")
driver.find_element(By.CSS_SELECTOR, "div.card > h2")
driver.find_element(By.XPATH, "//input[@name='q']")
```

By 策略

`By.ID`	匹配元素 id 属性
`By.NAME`	匹配元素 name 属性
`By.CLASS_NAME`	匹配 CSS class（单个类名）
`By.TAG_NAME`	匹配 HTML 标签名
`By.CSS_SELECTOR`	CSS 选择器（最灵活）
`By.XPATH`	XPath 表达式
`By.LINK_TEXT`	精确匹配链接文本
`By.PARTIAL_LINK_TEXT`	部分匹配链接文本

查找多个元素

```
items = driver.find_elements(By.CSS_SELECTOR, "li.item")
for item in items:
    print(item.text)
# Returns empty list if none found (no exception)
```

交互

点击与输入

```
elem = driver.find_element(By.ID, "search")
elem.clear()                 # clear existing text
elem.send_keys("selenium python")
elem.submit()                # submit parent form
```

下拉框

```
from selenium.webdriver.support.ui import Select
select = Select(driver.find_element(By.ID, "country"))
select.select_by_visible_text("Canada")
select.select_by_value("ca")
select.select_by_index(2)
```

元素属性

`.text`	可见文本内容
`.get_attribute('href')`	HTML 属性值
`.is_displayed()`	元素是否可见
`.is_enabled()`	元素是否可交互
`.is_selected()`	复选框/单选框是否被选中
`.tag_name`	HTML 标签（如 'input'、'div'）
`.value_of_css_property('color')`	计算后的 CSS 属性值

等待

显式等待

```
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
elem = WebDriverWait(driver, 10).until(
    EC.presence_of_element_located((By.ID, "result")))
```

预期条件

`presence_of_element_located`	元素存在于 DOM 中
`visibility_of_element_located`	元素在页面上可见
`element_to_be_clickable`	元素可见且可点击
`text_to_be_present_in_element`	元素包含预期文本
`alert_is_present`	JavaScript alert 弹出
`staleness_of`	元素不再在 DOM 中
`title_contains`	页面标题包含文本

隐式等待

```
driver.implicitly_wait(10)  # seconds, applies globally
# Explicit waits are preferred — more precise control
```

Frame 与窗口

Frame

```
driver.switch_to.frame("frame-name")    # by name/id
driver.switch_to.frame(0)               # by index
driver.switch_to.frame(elem)            # by element
driver.switch_to.default_content()      # back to main
```

窗口与标签页

```
original = driver.current_window_handle
driver.switch_to.new_window("tab")      # open new tab
driver.switch_to.window(original)       # switch back
driver.close()                          # close current tab
```

弹窗

```
alert = driver.switch_to.alert
print(alert.text)
alert.accept()     # click OK
alert.dismiss()    # click Cancel
alert.send_keys("input text")
```

截图

截取截图

```
driver.save_screenshot("page.png")      # full page
elem = driver.find_element(By.ID, "chart")
elem.screenshot("chart.png")            # single element
```

截图为 Base64

```
b64 = driver.get_screenshot_as_base64()
png = driver.get_screenshot_as_png()    # bytes
```

动作

Action 链

```
from selenium.webdriver.common.action_chains import ActionChains
actions = ActionChains(driver)
actions.move_to_element(menu).click().perform()
```

键盘动作

```
from selenium.webdriver.common.keys import Keys
elem.send_keys(Keys.ENTER)
elem.send_keys(Keys.CONTROL, "a")       # select all
actions.key_down(Keys.SHIFT).click(elem).perform()
```

鼠标动作

`.click(elem)`	点击元素
`.double_click(elem)`	双击元素
`.context_click(elem)`	右键点击元素
`.move_to_element(elem)`	悬停在元素上
`.drag_and_drop(src, dst)`	拖拽源元素到目标
`.click_and_hold(elem)`	按住鼠标按钮
`.release()`	释放鼠标按钮

断言

常用断言（pytest）

```
assert "Dashboard" in driver.title
assert driver.find_element(By.ID, "msg").text == "Done"
assert driver.current_url.endswith("/home")
assert len(driver.find_elements(By.CSS_SELECTOR, "tr")) > 0
```

基于等待的断言

```
WebDriverWait(driver, 5).until(             # appears
    EC.visibility_of_element_located((By.ID, "success")))
WebDriverWait(driver, 5).until(             # disappears
    EC.invisibility_of_element_located((By.ID, "spinner")))
```

JavaScript 执行

```
result = driver.execute_script("return document.title")
driver.execute_script(
    "arguments[0].scrollIntoView(true);", elem)
```

常用模式

Page Object 模式

```
class LoginPage:
    URL = "/login"
    user_loc = (By.ID, "username")
    def login(self, drv, user, pwd):
        drv.find_element(*self.user_loc).send_keys(user)
```

上下文管理器

```
from selenium import webdriver
with webdriver.Chrome() as driver:
    driver.get("https://example.com")
    print(driver.title)
# driver.quit() called automatically
```

重试与清理

```
try:
    driver.get("https://example.com")
    WebDriverWait(driver, 10).until(
        EC.element_to_be_clickable((By.ID, "btn")))
finally: driver.quit()
```