Referência Rápida do Selenium WebDriver

Automação de navegador, interação com elementos, esperas e asserções

Configuração

Instalação

```
pip install selenium webdriver-manager
# webdriver-manager baixa drivers de navegador automaticamente
```

Configuração Básica do Driver

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

Modo Headless

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

Navegadores Suportados

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

Navegador e Navegação

Navegação

```
driver.get("https://example.com")
driver.back()                # voltar no navegador
driver.forward()             # avançar no navegador
driver.refresh()             # recarregar página
```

Propriedades do Navegador

`driver.title`	Título da página atual
`driver.current_url`	URL da página atual
`driver.page_source`	Código-fonte HTML completo da página
`driver.get_cookies()`	Lista todos os cookies

Gerenciamento de Janela

```
driver.set_window_size(1920, 1080)
driver.maximize_window()
driver.minimize_window()
driver.quit()                # fecha todas as janelas, encerra
sessão
```

Localizando Elementos

Estratégias de Localização

```
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']")
```

Estratégias By

`By.ID`	Corresponde ao atributo id do elemento
`By.NAME`	Corresponde ao atributo name do elemento
`By.CLASS_NAME`	Corresponde à classe CSS (classe única)
`By.TAG_NAME`	Corresponde ao nome da tag HTML
`By.CSS_SELECTOR`	Seletor CSS (mais flexível)
`By.XPATH`	Expressão XPath
`By.LINK_TEXT`	Texto exato da âncora
`By.PARTIAL_LINK_TEXT`	Correspondência parcial do texto da âncora

Localizar Múltiplos

```
items = driver.find_elements(By.CSS_SELECTOR, "li.item")
for item in items:
    print(item.text)
# Retorna lista vazia se nenhum encontrado (sem exceção)
```

Interação

Clicar e Digitar

```
elem = driver.find_element(By.ID, "search")
elem.clear()                 # limpa texto existente
elem.send_keys("selenium python")
elem.submit()                # envia o formulário pai
```

Listas Suspensas

```
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)
```

Propriedades do Elemento

`.text`	Conteúdo de texto visível
`.get_attribute('href')`	Valor do atributo HTML
`.is_displayed()`	True se o elemento está visível
`.is_enabled()`	True se o elemento é interativo
`.is_selected()`	True se caixa de seleção/rádio está marcada
`.tag_name`	Tag HTML (ex.: 'input', 'div')
`.value_of_css_property('color')`	Valor da propriedade CSS computada

Esperas

Espera Explícita

```
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")))
```

Condições Esperadas

`presence_of_element_located`	Elemento existe no DOM
`visibility_of_element_located`	Elemento está visível na página
`element_to_be_clickable`	Elemento está visível e habilitado
`text_to_be_present_in_element`	Elemento contém o texto esperado
`alert_is_present`	Alerta JavaScript está exibido
`staleness_of`	Elemento não está mais no DOM
`title_contains`	Título da página contém o texto

Espera Implícita

```
driver.implicitly_wait(10)  # segundos, aplica globalmente
# Esperas explícitas são preferíveis — controle mais preciso
```

Frames e Janelas

Frames

```
driver.switch_to.frame("frame-name")    # por nome/id
driver.switch_to.frame(0)                # por índice
driver.switch_to.frame(elem)             # por elemento
driver.switch_to.default_content()       # volta ao principal
```

Janelas e Abas

```
original = driver.current_window_handle
driver.switch_to.new_window("tab")       # abre nova aba
driver.switch_to.window(original)        # volta para a original
driver.close()                           # fecha aba atual
```

Alertas

```
alert = driver.switch_to.alert
print(alert.text)
alert.accept()     # clica OK
alert.dismiss()    # clica Cancelar
alert.send_keys("texto de entrada")
```

Capturas de Tela

Capturar Capturas de Tela

```
driver.save_screenshot("page.png")       # página completa
elem = driver.find_element(By.ID, "chart")
elem.screenshot("chart.png")             # elemento único
```

Captura de Tela como Base64

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

Ações

Cadeias de Ação

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

Ações de Teclado

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

Ações do Mouse

`.click(elem)`	Clica no elemento
`.double_click(elem)`	Duplo clique no elemento
`.context_click(elem)`	Clique direito no elemento
`.move_to_element(elem)`	Passa o cursor sobre o elemento
`.drag_and_drop(src, dst)`	Arrasta a origem para o destino
`.click_and_hold(elem)`	Pressiona e segura o botão do mouse
`.release()`	Solta o botão do mouse

Asserções

Asserções Comuns (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
```

Asserções Baseadas em Espera

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

Execução de JavaScript

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

Referência Rápida do Selenium WebDriver

Padrões Comuns

Padrão Page Object

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

Gerenciador de Contexto

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

Retry e Limpeza

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