

Riferimento rapido Selenium WebDriver

Automazione del browser, interazione con elementi, attese e asserzioni

Configurazione

Installazione

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

Configurazione base del 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()))
```

Modalità headless

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

Browser supportati

<code>webdriver.Chrome()</code>	Google Chrome / Chromium
<code>webdriver.Firefox()</code>	Mozilla Firefox (GeckoDriver)
<code>webdriver.Edge()</code>	Microsoft Edge (Chromium)
<code>webdriver.Safari()</code>	Apple Safari (solo macOS)

Browser e navigazione

Navigazione

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

Proprietà del browser

<code>driver.title</code>	Titolo della pagina corrente
<code>driver.current_url</code>	URL della pagina corrente
<code>driver.page_source</code>	Sorgente HTML completa della pagina
<code>driver.get_cookies()</code>	Elenco di tutti i cookie

Gestione delle finestre

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

Ricerca degli elementi

Strategie di localizzazione

```
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']")
```

Strategie By

By.ID	Corrisponde all'attributo id dell'elemento
By.NAME	Corrisponde all'attributo name dell'elemento
By.CLASS_NAME	Corrisponde alla classe CSS (una sola classe)
By.TAG_NAME	Corrisponde al nome del tag HTML
By.CSS_SELECTOR	Selettore CSS (il più flessibile)
By.XPATH	Espressione XPath
By.LINK_TEXT	Testo esatto del link
By.PARTIAL_LINK_TEXT	Corrispondenza parziale del testo del link

Ricerca multipla

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

Interazione

Clic e digitazione

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

Menu a tendina

```
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)
```

Proprietà degli elementi

<code>.text</code>	Contenuto testuale visibile
<code>.get_attribute('href')</code>	Valore dell'attributo HTML
<code>.is_displayed()</code>	True se l'elemento è visibile
<code>.is_enabled()</code>	True se l'elemento è interattivo
<code>.is_selected()</code>	True se la casella/radio è selezionata
<code>.tag_name</code>	Tag HTML (es. 'input', 'div')
<code>.value_of_css_property('color')</code>	Valore calcolato della proprietà CSS

Attese

Attesa esplicita

```
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")))
```

Condizioni attese

<code>presence_of_element_located</code>	L'elemento esiste nel DOM
<code>visibility_of_element_located</code>	L'elemento è visibile nella pagina
<code>element_to_be_clickable</code>	L'elemento è visibile e abilitato
<code>text_to_be_present_in_element</code>	L'elemento contiene il testo atteso
<code>alert_is_present</code>	È visualizzato un alert JavaScript
<code>staleness_of</code>	L'elemento non è più nel DOM
<code>title_contains</code>	Il titolo della pagina contiene il testo

Attesa implicita

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

Frame e finestre

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
```

Finestre e schede

```
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
```

Alert

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

Screenshot

Cattura screenshot

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

Screenshot come Base64

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

Azioni

Catene di azioni

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

Azioni da tastiera

```
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()
```

Azioni del mouse

<code>.click(elem)</code>	Clic sull'elemento
<code>.double_click(elem)</code>	Doppio clic sull'elemento
<code>.context_click(elem)</code>	Clic destro sull'elemento
<code>.move_to_element(elem)</code>	Passa il mouse sull'elemento
<code>.drag_and_drop(src, dst)</code>	Trascina la sorgente sulla destinazione
<code>.click_and_hold(elem)</code>	Tieni premuto il pulsante del mouse
<code>.release()</code>	Rilascia il pulsante del mouse

Asserzioni

Asserzioni comuni (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
```

Asserzioni basate su attesa

```
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")))
```

Esecuzione JavaScript

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

Riferimento rapido Selenium WebDriver

Pattern comuni

Pattern 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)
```

Context manager

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

Ripetizione e pulizia

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