Testování softwaru (KI/TSW)

Pavel Beránek

8 Dubna, 2025

8. Pokročilé testování webového UI

8.1. Selenium a příprava prostředí

Selenium je populární framework pro automatizaci webových prohlížečů, který umožňuje psát testovací skripty simulující chování uživatele – klikání, vyplňování formulářů, přihlašování apod. Lze pomocí něj testovat práci s různými prohlížeči jako je: Chrome, Firefox, Edge atd. Selenium existuje pro mnoho jazyků včetně C#, Ruby, Java a Python.

V minulé lekci jste si Selenium zkoušeli na tradičních úlohách z oblasti testování webového UI. Připravte si znovu prostředí a ověřte, že Vám Selenium funguje

Zadání:

1. Nainstalujte Selenium, spusťte Chromium (nebo jiné jádro webového prohlížeče) přes WebDriver a ověřte, že se stránka https://example.com načetla a má nadpis první úrovně s textem "Example Domain".

Řešení:

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

driver = webdriver.Chrome()
driver.get("https://example.com")
assert driver.find_element(By.TAG_NAME, "h1").text == "Example Domain"
print("Test prošel.")
driver.quit()
```

8.2. Testování drag & drop (přetahování prvků)

Mnoho moderních webů využívá interaktivní komponenty jako drag & drop (táhni a pusť) pro předávání souborů do webového systému (např. nahraj dokument přetáhnutím) nebo organizaci grafických prvků na plátně (např. Miro). Pojďme tuto složitější formu interakce otestovat pomocí tzv. akčního řetězu, kam se uvádí za sebou prováděne interakce s webovým systémem. Otestujeme, jak zachytávání chytatelných prvků z webového systému, tak pokládání takových prvků na jiné místo v systému.

```
from selenium.webdriver import ActionChains

ActionChains(driver).drag_and_drop(source, target).perform()
```

Zadání:

1. Otevřete stránku https://jqueryui.com/droppable/ a přetáhněte prvek "Drag me to my target" do cíle ("Drop here").
2. Napište test, který testujte, tuto funkcionalitu. Po stažení stránky je nutné zavolat následující kód: `driver.switch_to.frame(0)`. Důvod je ten, že prvky se nachází v tzv. iframu a je nutný se do něj dostat.
3. Ověřte, že se změnil text v cíli na "Dropped!" pomocí automatizovaného testu.

Řešení:

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

driver = webdriver.Chrome()
driver.get("https://jqueryui.com/droppable/")
driver.switch_to.frame(0)
source = driver.find_element(By.ID, "draggable")
target = driver.find_element(By.ID, "droppable")
ActionChains(driver).drag_and_drop(source, target).perform()
assert "Dropped!" in target.text
print("Test prošel.")
driver.quit()
```

8.3. Testování lazy-loading / infinite scroll (scrollování pro načtení obsahu)

Často Vás na webových stránkách zajímá jen prvotní část webu a nechcete scrollovat nízko pro ušetření dat. Webový stránky umožňují využívat techniku tzv. líného načítání (lazy-loading), při kterém se načítá obsah až při posunu stránky. Pokud takovou stránku píšete, je vhodné testovat, zda se nové prvky správně zobrazují při líném načítání. Tento mechanismus znáte zejména ze sociálncíh sítí pro načítání nekonečných feedů.

Následující kód provedet scroll svisle dolu na maximální výšku dokumentu (dno). Jedná se o prvotní načtenou výšku před lazy-loadiangem.

```
#                                       x, y
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
```

Zadání:

1. Na stránce https://infinite-scroll.com/demo/full-page/ scrollujte na konec a ověřte, že se načetly nové články.
2. Rada: spočítejte si počet načítaných prvků před lazy-loadingem a po. Také musíte nechat webovému systému nějaký čas na provedení lazy-loadingu.

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

driver = webdriver.Chrome()
driver.get("https://infinite-scroll.com/demo/full-page/")
initial_articles = len(driver.find_elements(By.TAG_NAME, "article"))
driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
time.sleep(3)
new_articles = len(driver.find_elements(By.TAG_NAME, "article"))
assert new_articles > initial_articles
print(f"Test prošel. Původní počet: {initial_articles}, počet po scrollování: {new_articles}")
driver.quit()
```

8.4. Testování stahování souboru a kontrola v systému

Webové portály často poskytují soubory ke stažení. Testování stahování ověřuje, že soubor je dostupný ke stažení a uloží se správně.

```
download_dir = os.path.abspath("./downloads") # relativní cestu ignoruje, musíte vytvořit absolutní

options = webdriver.ChromeOptions()
options.add_experimental_option("prefs", {
    "download.default_directory": download_dir,
    "download.prompt_for_download": False,
    "plugins.always_open_pdf_externally": True # jinak nestáhne soubor, ale otevře ho v prohlížeči
})
```

Zadání:

1. Na stránce https://file-examples.com/index.php/sample-documents-download/sample-pdf-download/ stáhněte PDF soubor a ověřte, že existuje.
2. Rada: budete muset odkliknout Cookie consent. Také budete muset počkat než se soubor stáhne.

Řešení:

```
import os
import time
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

download_dir = os.path.abspath("./downloads")
os.makedirs(download_dir, exist_ok=True)

options = webdriver.ChromeOptions()
options.add_experimental_option("prefs", {
    "download.default_directory": download_dir,
    "download.prompt_for_download": False,
    "plugins.always_open_pdf_externally": True
})
driver = webdriver.Chrome(options=options)
driver.get("https://file-examples.com/index.php/sample-documents-download/sample-pdf-download/")

try:
    consent_button = WebDriverWait(driver, 5).until(
        EC.element_to_be_clickable((By.CLASS_NAME, "fc-cta-consent"))
    )
    consent_button.click()
    print("Cookie banner zavřen.")
except:
    print("Cookie banner se nezobrazil.")

download_link = driver.find_element(By.CLASS_NAME, "download-button")
download_link.click()
time.sleep(10)
assert any(fname.endswith(".pdf") for fname in os.listdir(download_dir))
print(f"Test prošel.")
driver.quit()
```

8.5. Testování formuláře s nahráváním souboru

Formuláře s nahráváním souborů jsou velmi časté na webových portálech (např. při posílání příloh). Měli bychom testovat, že se nahrané soubory skutečně nahrály do webového portálu.

Zadání:

1. Otevři stránku https://the-internet.herokuapp.com/upload, nahraj libovolný textový soubor a ověř, že se zobrazí zpráva "File Uploaded!".

```
upload = driver.find_element(By.ID, "file-upload")
upload.send_keys("/absolutni/cesta/k/souboru.txt")
```

Řešení:

```
from selenium import webdriver
from selenium.webdriver.common.by import By
import os
import time

driver = webdriver.Chrome()
driver.get("https://the-internet.herokuapp.com/upload")

test_file_path = os.path.abspath("./downloads/file-sample_150kB.pdf") # zadejte si svůj soubor

upload = driver.find_element(By.ID, "file-upload")
upload.send_keys(test_file_path)

driver.find_element(By.ID, "file-submit").click()
time.sleep(2)

message = driver.find_element(By.TAG_NAME, "h3").text
assert "File Uploaded!" in message
print(f"Test prošel.")
driver.quit()
```

8.6. Testování sessionStorage a localStorage

SessionStorage a localStorage jsou způsoby jak lze uchovávat data v prohlížeči na klientské straně. SessionStorage je neperzistentní a data mizí, zatímco localStorage je perzistentní. Tyto data často potřebujeme pro pokročilé chování a skladování behaviorálních dat nebo urychlení ergonomie práce (přihlášování, poslední interakce).

Zadání:

1. Projděte si následující článek: https://scrapingant.com/blog/selenium-local-storage
2. Na libovolné stránce nastav hodnotu sessionStorage[“test_key”] = “42” a ověř, že je hodnota opravdu uložená. Poté totéž proveď s localStorage.

8.7. Testování cookies a simulace přihlášení bez formuláře

Cookies je základní mechanismus persistence dat pro serverovou stranu. Slouží pro získání dat serverem z Vašeho počítače.

Zadání:

1. Pročtěte si následující článek na práci s Cookies: https://www.selenium.dev/documentation/webdriver/interactions/cookies/
2. Vyberte si libovolný website, nastavte mu cookie.
3. Obnovte stránku a ověřte, zda je cookie nastavená.

8.8. Paralelní testování pomocí Selenium Grid

Selenium Grid umožňuje spouštět testy paralelně na různých prohlížečích a OS. To je klíčové pro testování kompatibility a škálování v CI/CD. Jedná se o pokročilé téma, kde musíte rozumět i Dockeru.

Zadání:

1. Projděte si následující tutoriál: https://www.browserstack.com/guide/run-selenium-tests-in-docker

Domácí cvičení - Selenium a BDD

Naposledy jste si zkoušeli přepsat selenium testy do strukturované podoby s využitím Pytest frameworku. Zkuste si napsat alespoň 2 testy z dnešní lekce pomocí BDD.

Ukázka feature souboru:

```
Feature: Login functionality

  Scenario: Successful login with valid credentials
    Given I open the login page
    When I enter valid username and password
    And I click the login button
    Then I should be redirected to the dashboard
```

Ukázka bdd souboru s využitím frameworku behave:

```
from behave import given, when, then
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.common.keys import Keys
import time

@given('I open the login page')
def step_impl(context):
    context.driver = webdriver.Chrome()
    context.driver.get("https://example.com/login")

@when('I enter valid username and password')
def step_impl(context):
    context.driver.find_element(By.NAME, "username").send_keys("testuser")
    context.driver.find_element(By.NAME, "password").send_keys("password123")

@when('I click the login button')
def step_impl(context):
    context.driver.find_element(By.NAME, "password").send_keys(Keys.RETURN)
    time.sleep(2)

@then('I should be redirected to the dashboard')
def step_impl(context):
    assert "dashboard" in context.driver.current_url
    context.driver.quit()
```