Testarea sistemelor software

Testarea automată a aplicațiilor web - Tema 10

Dăncău Sebastian - 333

Selenium - tool de testare automată a aplicațiilor web **2**

Generarea testelor pe exemple proprii **2**

Aplicația testată 2

Datele pentru testare 3

Scenarii de testare 3

Testarea aplicației 4

Referințe **5**

Cod **5**

Demo video **5**

Graf **5**

Selenium - tool de testare automată a aplicațiilor web

Selenium Webdriver este un tool care emulează o instanță a unui browser, în fața căruia există o serie de drivere, care ne permit sa controlam browser-ul in mod automat. Astfel, reușim să emulăm interacțiunea unui om cu un browser prin intermediul unui set de instrucțiuni.

Selenium ne oferă posibilitatea de a naviga pe pagini web, interacționa cu butoane, link-uri, cauta elemente in pagina si a focusa pe ele, completat formulare și verificat date din pagina.

Deși este un tool de testare de frontend, având la baza driver-ul unui browser, poate intercepta network-ul, putând sa valideze și să testeze request-uri.

De asemenea, selenium poate fi rulat remote, pentru a interacționa cu un browser care rulează pe un server remote.

Webdriver-ul selenium are mai multe configurări, prin care putem să specificăm browser-ul pe care il folosim (Chrome, Firefox, etc.), dimensiunea lui, versiunea, etc.

Selenium funcționează ca o librarie, disponibilă pentru mai multe limbaje de programare: Java, Python, CSharp, Ruby, Javascript, Kotlin.

Generarea testelor pe exemple proprii

Aplicația testată

Aplicația web testată constă într-o platformă de live streaming. Sunt testate cateva scenarii uzuale: un utilizator se inregistreaza în platforma, se loghează, verifică detaliile contului, se deloghează.

Datele pentru testare

Pentru a putea genera teste automate, este nevoie sa definim cum funcționează aplicația. Datele sunt salvate în data frame-uri din libraria pandas, deoarece selectarea rândurilor și elementelor este mai rapidă.

Am definit 5 data frame-uri, unde putem adauga sau sterge elemente;

- df_pages - informații despre o pagina web, id, url relativ, url absolut, un nume friendly
- df_input_elements - informații despre elemente din pagină, id, nume friendly, un selector prin care putem identifica elementul, tipul (input, buton, link, etc.), un atribut, in cazul in care elementul are o valoare atribuită
- df_attributes - informații despre atribute, id, nume friendly, tipul (string, int), subtipul (nume, email, parolă)
- df_selectors - informații despre selectori, id, nume friendly și selectorul XPATH sau CSS, prin care identificam elementele html
- df_components - informații despre componente, id. nume friendly, grupuri de elemente, tipul componentei (formular, buton, link)
-

```
df_pages = pd.DataFrame({
   'id': ['page_1', 'page_2', 'page_3', 'page_4'],
   'relativeUrl': ['/', '/login', '/register', '/account'],
   'friendly': ['homepage', 'login', 'register', 'account']
})
df_pages['absoluteUrl'] = 'http://localhost:3000'
```

crearea datelor pentru pagini

Scenarii de testare

Pentru a crea scenarii de test, creăm un graf, al carui noduri sunt reprezentate de pagini și componente. Dupa care, creăm legăturile dintre noduri.

```
# adaugarea unei pagini în graf
self.add_vertex_to_graph('homepage', 'page_1', 'page')
# adaugarea unei componente în graf
self.add_vertex_to_graph('register_button', 'component_4', 'component')
# legarea a două noduri din graf
self.link_vertexes('homepage', 'register_button')
```

După ce graful este construit, programul găsește toate drumurile de la start pana la finalul grafului, deoarece, într-o aplicație web există mai multe moduri prin care putem să ajungem la același rezultat.

Testarea aplicației

Programul ia fiecare rută din graf și testează fiecare nod din rută. Dacă un node este de tipul pagină, driver-ul va încerca să navigheze către pagina respectivă. Dacă un nod este o componentă, parcurgem fiecare element al componentei.

Inițial, mutăm cursorul pe element.

```
   def get_html_element(self, element):
            selector = self.data.df_selectors[self.data.df_selectors['id'] ==
element['selectorId']].iloc[0]['selector']
        try:
            html_element = WebDriverWait(self.driver, timeout=10).until(lambda
d: d.find_element(By.XPATH, selector))
        except:
            html_element = WebDriverWait(self.driver, timeout=10).until(lambda
d: d.find_element(By.CSS_SELECTOR, selector))

ActionChains(driver=self.driver).move_to_element(html_element).perform()
        return html_element
```

Dacă elementul este un buton, îl apăsăm.

```
def click_element(self, element):
        html_element = self.get_html_element(element)

ActionChains(driver=self.driver).move_to_element(html_element).perform()
        start_time = time.perf_counter()
        while time.perf_counter()-start_time <= 10:
            try:
                ActionChains(driver=self.driver).click(html_element).perform()
                return
            except Exception as e:
                time.sleep(0.5)
```

Dacă elementul este un input, generăm un atribut random conform datelor mapate si populăm input-ul.

```
def handle_input(self, element):
        html_element = self.get_html_element(element)
          attribute = self.data.df_attributes[self.data.df_attributes['id'] ==
element['attributeId']].iloc[0]
        if attribute['value'] is None:
            attribute['value'] = self.generate_value(attribute)
                  self.data.df_attributes.loc[self.data.df_attributes['id'] ==
element['attributeId'], 'value'] = attribute['value']

ActionChains(driver=self.driver).move_to_element(html_element).perform()
        html_element.clear()
        html_element.send_keys(attribute['value'])
```

Dacă este un element a cărui valoare trebuie să o validăm, luăm text-ul elementului html și o comparăm cu valoarea pe care o avem deja salvată. (ex.: validăm că email-ul cu care ne-am logat este prezent și în pagina cu detaliile contului).

Referințe

- https://www.selenium.dev/documentation/webdriver/drivers/remote_webdriver/ [documentație oficială Selenium]

Cod

- https://github.com/Ssebi1/livestream-web-tests

Demo video

- https://github.com/Ssebi1/livestream-web-tests/blob/live/demo.mov

Graf

- https://drive.google.com/file/d/1r6z-3_nu8rXJgIpjx5XnmCiaK9yqgaIp/view?usp=sharing