Selenium Webdriver test case automation

Author:
Robert Jaszewski
Gdynia 25.10.2022

1. Test Case 1.
ID: 002.
Title: Filling the account registration form without phone number.
Environment: Chrome version 62.0.3202.94, Ubuntu 16.04 LTS.

Preconditions:
1. Browser open.
2. Main page open: "http://automationpractice.com/index.php".
3. User is not logged in.

Steps:
1. Click "Sign in"
2. Enter email in "Email address" input field
3. Click on "Create an account" button
4. Check radio button with "Mr." option
5. Enter first name
6. Enter company name
7. Enter last name
8. Enter password (Five characters minimum)
9. Choose date of birth
10. Enter address
11. Enter address (line 2)
12. Enter city
13. Choose state
14. Enter zip/postal code
15. Enter address alias
16. Click on "Register" button

Expected results:
The user receives the information: "There is 1 error 1. You must register at least one phone number."

Final conditions:
1. Account is not created

II. Selenium webdriver test case automation

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select

#TEST DATA

email = "automationtester223@gmail.com"
aliasEmail = "aliastester223@gmail.com"
lastName = "Smith"
firstName = "John"
companyName = "Koolos"
password = "Ilikecats123"
streetAddres = "3398 Pick Street"
streetAddres2 = "CitySky Building, Floor 2 "
city = "Denver"
postalCode = "80203"
mobilePhoneNumber = "720-334-1530"
```

```
class RegistrationOfNewUser(unittest.TestCase):
    def setUp(self):
        # PRECONDITIONS
        # 1. Browser open
        self.driver = webdriver.Chrome()
        self.driver.maximize_window()
        # 2. Main page open
        self.driver.get("http://automationpractice.com/index.php")
        # 3. User is not logged in
        self.driver.implicitly_wait(15)
    def testCaseMobilePhoneNumberFieldEmpty(self):
        # STEPS
        # 1. Click "Sign in"
        self.driver.find_element(By.LINK_TEXT, "Sign in").click()
        # 2. Enter email in "Email address" input field
        self.driver.find_element(By.ID, "email_create").send_keys(email)
        # 3. Click on "Create an account" button
        self.driver.find_element(By.ID, "SubmitCreate").click()
        # 4. Check radio button with "Mr." option
        self.driver.find_element(By.ID, "uniform-id_gender1").click()
        # 5. Enter first name
        self.driver.find_element(By.ID, "customer_firstname").send_keys(firstName)
        # 6. Enter company name
        self.driver.find_element(By.ID, "company").send_keys(companyName)
        # 7. Enter last name
        self.driver.find_element(By.ID, "customer_lastname").send_keys(lastName)
        # 8. Enter password (Five characters minimum)
        self.driver.find_element(By.ID, "passwd").send_keys(password)
        # 9. Choose date of birth
        # Day
        Select(self.driver.find_element(By.ID, "days")).select_by_value("29")
        # Month
        Select(self.driver.find_element(By.ID, "months")).select_by_value("12")
        # Year
        Select(self.driver.find_element(By.ID, "years")).select_by_value("1982")
        # 10. Enter address
        self.driver.find_element(By.ID, "address1").send_keys(streetAddres)
        # 11. Enter address (line 2)
        self.driver.find_element(By.ID, "address2").send_keys(streetAddres2)
        # 12. Enter city
        self.driver.find_element(By.ID, "city").send_keys(city)
        # 13. Choose state
        Select(self.driver.find_element(By.ID, "id_state")).select_by_visible_text('Colorado')
        # 14. Enter zip/postal code
        self.driver.find_element(By.ID, "postcode").send_keys(postalCode)
        # 15. Enter address alias
        addressAliasField = self.driver.find_element(By.ID, "alias")
        addressAliasField.clear()
        addressAliasField.send_keys(aliasEmail)
        # WARNING! Only for negative test case's
        # 16. Click on "Register" button
        self.driver.find_element(By.ID, "submitAccount").click()
        # EXPECTED RESULTS
        # The user receives the information: "There is 1 error 1. You must register at least one
phone number."
        # 1. Looking for alert "There is 1 error"
        alertThereIs1ErrorAlertText = self.driver.find_element(By.XPATH, '//p[text() = "There is 1
error"]')
        self.assertTrue(alertThereIs1ErrorAlertText.is_displayed())
        print("Assertion 1 correct")
        # 2. Looking for alert "1. You must register at least one phone number"
        alertPhoneNumber = self.driver.find_element(By.XPATH, '//li[text() = "You must register at
least one phone number."]')
        self.assertTrue(alertPhoneNumber.is_displayed())
        print("Assertion 2 correct")
    def tearDown(self):
        # End of test
        self.driver.quit()
        # FINAL CONDITIONS
        # 1. Account is not created
```

1. Test Case 2.
ID: 002.
Title: Filling the account registration form with a password that is too short.
Environment: Chrome version 62.0.3202.94, Ubuntu 16.04 LTS.

Preconditions:
1. Browser open.
2. Main page open: "http://automationpractice.com/index.php".
3. User is not logged in.

Steps:
1. Click "Sign in"
2. Enter email in "Email address" input field
3. Click on "Create an account" button
4. Check radio button with "Mr." option
5. Enter first name
6. Enter company name
7. Enter last name
8. Enter password that is too short
9. Choose date of birth
10. Enter address
11. Enter address (line 2)
12. Enter city
13. Choose state
14. Enter zip/postal code
15. Enter mobile phone
16. Enter address alias
17. Click on "Register" button

Expected results:
The user receives the information: "There is 1 error 1. passwd is invalid."

Final conditions:
1. Account is not created

II. Selenium webdriver test case automation
```
import unittest
from selenium import webdriver
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import Select
from time import sleep

#TEST DATA
```

```python
email = "automationtester223@gmail.com"
aliasEmail = "aliastester223@gmail.com"
lastName = "Smith"
firstName = "John"
companyName = "Koolos"
password = "kot"
streetAddres = "3398 Pick Street"
streetAddres2 = "CitySky Building, Floor 2 "
city = "Denver"
postalCode = "80203"
mobilePhoneNumber = "720-334-1530"

class RegistrationOfNewUser(unittest.TestCase):
    def setUp(self):
        # PRECONDITIONS
        # 1. Browser open
        self.driver = webdriver.Chrome()
        self.driver.maximize_window()
        # 2. Main page open
        self.driver.get("http://automationpractice.com/index.php")
        # 3. User is not logged in
        self.driver.implicitly_wait(15)
    def testCasePasswordTooShort(self):
        # STEPS
        # 1. Click "Sign in"
        self.driver.find_element(By.LINK_TEXT, "Sign in").click()
        # 2. Enter email in "Email address" input field
        self.driver.find_element(By.ID, "email_create").send_keys(email)
        # 3. Click on "Create an account" button
        self.driver.find_element(By.ID, "SubmitCreate").click()
        # 4. Check radio button with "Mr." option
        self.driver.find_element(By.ID, "uniform-id_gender1").click()
        # 5. Enter first name
        self.driver.find_element(By.ID, "customer_firstname").send_keys(firstName)
        # 6. Enter company name
        self.driver.find_element(By.ID, "company").send_keys(companyName)
        # 7. Enter last name
        self.driver.find_element(By.ID, "customer_lastname").send_keys(lastName)
        # 8. Enter password that is too short
        self.driver.find_element(By.ID, "passwd").send_keys(password)
        # 9. Choose date of birth
        # Day
        Select(self.driver.find_element(By.ID, "days")).select_by_value("29")
        # Month
        Select(self.driver.find_element(By.ID, "months")).select_by_value("12")
        # Year
        Select(self.driver.find_element(By.ID, "years")).select_by_value("1982")
        # 10. Enter address
        self.driver.find_element(By.ID, "address1").send_keys(streetAddres)
        # 11. Enter address (line 2)
        self.driver.find_element(By.ID, "address2").send_keys(streetAddres2)
        # 12. Enter city
        self.driver.find_element(By.ID, "city").send_keys(city)
        # 13. Choose state
        Select(self.driver.find_element(By.ID, "id_state")).select_by_visible_text('Colorado')
        # 14. Enter zip/postal code
        self.driver.find_element(By.ID, "postcode").send_keys(postalCode)
        # 15. Enter mobile phone
        self.driver.find_element(By.ID, "phone_mobile").send_keys(mobilePhoneNumber)
        # 16. Enter address alias
        addressAliasField = self.driver.find_element(By.ID, "alias")
        addressAliasField.clear()
        addressAliasField.send_keys(aliasEmail)
        # WARNING! Only for negative test case's
        # 17. Click on "Register" button
        self.driver.find_element(By.ID, "submitAccount").click()

        # EXPECTED RESULTS
        # The user receives the information: "There is 1 error 1. passwd is invalid."
        # 1. Looking for alert "There is 1 error"
        alertThereIs1ErrorAlertText = self.driver.find_element(By.XPATH, '//p[text() = "There is 1
error"]')
        self.assertTrue(alertThereIs1ErrorAlertText.is_displayed())
        print("Assertion 1 correct")
        # 2. Looking for alert "1. passwd is invalid."
        allertPasswd1 = self.driver.find_element(By.XPATH, '//b[text() = "passwd"]')
        self.assertTrue(allertPasswd1.is_displayed())
        print ("Assertion 2 correct")
        alertPasswd2 = self.driver.find_element(By.XPATH, '//li[text() = " is invalid."]')
        self.assertTrue(alertPasswd2.is_displayed())
        print("Assertion 3 correct")
    def tearDown(self):
        # End of test
        self.driver.quit()
        # FINAL CONDITIONS
        # 1. Account is not created
```

III. Uwagi końcowe
Test case automation (functional test) successful. The test may be sensitive to changing the structure of the website due to the need to use long path in locators XPATH. For this reason, it is suggested to periodically monitor the course of the test and update the script if it is necessary.