Metody Programowania

Laboratorium : „Wprowadzenie do Selenium”

Zadania do samodzielnego wykonania w kolejności chronologicznej:

Zadanie 1: Przygotowanie projektu do pracy (10 pkt)

1. Pobierz odpowiednią wersję chromedriver do przeglądarki na komputerze
    a. W celu sprawdzenia wersji przeglądarki wybierz Pomoc->Informacja o przeglądarce Google Chrome
    b. Z strony https://chromedriver.chromium.org/ wybierz odpowiednią wersję zgodną z wersją przeglądarki
    c. Pobierz archiwum zip zgodne z używanym systemem operacyjnym
2. Pobierz projekt z wykładu ShopPomSelenium
3. Otwórz projekt przez plik pom.xml -> otwórz jako projekt
4. Rozpakuj archiwum zip z chromedriver
5. Przenieść zawartość archiwum do ShopPomSelenium/src/main/resources
6. Modyfikacja klasy LoginTest
    a. Otwórz klasę LoginTest z src/test/java/
    b. Zmodyfikuj linię 16 tak aby dopasować ją do swojej wersji chromedriver:
        i. Przy linux/macOS – brak konieczności modyfikacji
        ii. Przy windows – dodaj rozszerzenie pliku chromedriver ‘.exe’
7. Uruchom klasę LoginTest
8. **Zrób zrzut ekranu z wynikiem testu potwierdzający poprawne wykonanie testu**

Zadanie 2: Utworzenie bazowej klasy testowej (10 pkt)

Aby zapobiec duplikacji kodu, dobrą praktyką jest ustanowienie tzw. Bazowej Klasy Testowej, która zawiera w sobie odpowiednie adnotacje @Before i @After, które mogą posłużyć do przygotowania danych testowych i/lub posprzątania po wykonanym teście.

W celu utworzenia klasy bazowej wykonaj następujące polecenia:

1. Utwórz nowego użytkownika w sklepie online :
    a. Przejdź na stronę http://automationpractice.com/index.php?controller=authentication&back=my-account
    b. Utwórz nowe konto (panel po lewej stronie) podając unikatowy, zmyślony adres email (skopiuj jego wartość do dowolnego pliku aby użyć w kolejnych punktach)
        i. Wypełnij formularz zmyślonymi danymi (nie podawaj swoich danych osobowych)
        ii. Kod pocztowy to 5 dowolnych cyfr np. 22222
        iii. Zapisz podane przez siebie hasło – będzie potrzebne w 2 podpunkcie
2. Utwórz nową klasę testową w pakiecie src/java/test/…
    a. Jako nazwę klasy wprowadź „BaseTest” lub „AuthorizedUserBaseTest”

b. Dodaj WebDriver jako pole klasy – upewnij się, aby miało odpowiedni modyfikator dostępu (tak aby pole było dostępne tylko dla klas dziedziczących i jednocześnie nie było dostępne dla wszystkich możliwych klas)
c. Utwórz nową metodę z adnotacją @Before np. 'loginToShop()':
    i. Metoda powinna uruchamiać nową instancję przeglądarki google chrome
    ii. Wchodzić na stronę http://automationpractice.com/index.php
    iii. Logować się danymi z punktu 1.
d. Dodaj metodę z adnotacją @After np. 'logoutFromShop()':
    i. Rozszerz model NavigationBar o metodę logout():LoginPage
    ii. W tym celu będziesz musiał zlokalizować element który klika na przycisk logout
    iii. Zwróć nową instancję klasy LoginPage(driver) aby upewnić się, że test zaczeka na pojawienie się widoku do logowania

3. **Zamieść kod z klasy bazowej testów w sprawozdaniu a także zaktualizowany kod klasy NavigationBar**

Zadanie 3: Utworzenie pierwszego testu dla koszyka (10 pkt)

1. Utwórz nową klasę testową w tym samym pakiecie co LoginTest
    a. O nazwie np. 'CartTest'
    b. Klasa powinna rozszerzać klasę bazową testów – co zagwarantuje poprawne zalogowanie i wylogowanie się na potrzeby testów
2. Dodaj pierwszą metodę z adnotacją @Test, która zweryfikuje, że domyślnie koszyk jest pusty
    a. Rozszerz NavigationBar o metodę pobierającą status koszyka w postaci tekstowej np. 'getCartStatus():String'
    b. W metodzie testowej pobierz status koszyka i postaw asercje, że status zawiera słowo 'empty'
3. **Zamieść kod z klasy testowej w sprawozdaniu**
4. **Zamieść kod z zaktualizowanej klasy NavigationBar w sprawozdaniu**

Zadanie Dodatkowe: Utworzenie drugiego testu dla koszyka (6 pkt)

1. Rozszerz klasę testową z poprzedniego zadania (np. CartTest)
2. Dodaj metodę testową (@Test), która zweryfikuje, że po dodaniu przedmiotu do koszyka, liczba przedmiotów jest poprawna (1)
    a. Rozszerz NavigationBar o metodę wyszukującą towar np. search(String)
    b. Metoda powinna zwracać obiekt reprezentujący widok Rezultatów poszukiwań np. SearchResultsPage
    c. Metoda najpierw szuka zadanej frazy w polu Search a następnie zwróca nowy obiekt typu PageObject (jak w pkt b)
3. Utwórz Page Object dla widoku rezultatów poszukiwań np. SearchResultsPage
    a. W konstruktorze zaczekaj na poprawne załadowanie strony (patrz przykład z Loginpage)
    b. Następnie zadeklaruj inicjalizację PageFactory
4. Utwórz metodę w klasie z pkt 3 która doda pierwszy znaleziony element do koszyka

    a. Możesz ułatwić to zadanie znajdując lokator który wskaże wszystkie towary gotowe do zakupu a następnie z tej listy wybierzesz 1 element (ten o indeksie 0)
    b. Po kliknięciu przycisku ‘Add to Cart’ niech metoda domyślnie potwierdza akcję ‘Continue shopping’ – możesz w tym celu utworzyć kolejny Page Object
5. W metodzie testowej przeprowadź scenariusz w którym skorzystasz z wszystkich wcześniej utworzonych elementów tj :
    a. Po zalogowaniu się wyszukaj dowolną frazę np. ‘Blouse’
    b. Dodaj pierwszy dostępny przedmiot do koszyka
    c. Pobierz status koszyka
    d. Sprawdź czy status koszyka zawiera dokładnie 1 przedmiot
6. **Zamieść Kod z wszystkich utworzonych klas oraz wynik wykonania testu**

<u>Rozwiązanie zadań powinno mieć następującą formę:</u>

- dokument .pdf
- nazwa [IMIE]_[NAZWISKO]_MPR_LAB_[NUMER_LABORATORIUM]
- Zadanie 1:
    - Zrzut ekranu potwierdzający poprawne wykonanie testów
- Zadanie 2:
    - Kod z klasy bazowej testów
    - Zaktualizowany kod z klasy NavigationBar
- Zadanie 3:
    - Kod z utworzonej klasy testowej
    - Zaktualizowany kod z klasy NavigationBar
    - Zrzut ekranu potwierdzający poprawne wykonanie testu
- Zadanie dodatkowe:
    - Zaktualizowany kod wszystkich modyfikowanych klas
    - Zrzut ekranu potwierdzający poprawne wykonanie testu