Zad. 1 Poprawienie funkcji testującej otwieranie przeglądarki w taki sposób, aby użyć funkcji setUP (otwarcie), tearDown (zamknięcie) plus funkcja testująca, w której znajdzie się pozyskanie url (get)

**Testowanie Interakcji Użytkownika**

Zad. 2 Wrócmy do poprzedniego testu z zadania 1 i rozszerzmy o następujące elementy

1. Zaimporotwanie Keys

**from** selenium.webdriver.common.keys **import** Keys

1. Zaimportowanie time

**import** time

1. Pod testem sprawdzającym nazwę w tytule przeglądarki; tworzymy zmienną header\_text do której przypisujemy szukanie po tagu. Do tego służy polecenie

find\_element\_by\_tag\_name('h1').text

1. Wykonanie sprawdzanie za pomocą assertIn

Zaimplementowanie następującego kodu

*#Dodanie zadania*

inputbox = self.browser.find\_element\_by\_id('id\_new\_item')

self.assertEqual(

inputbox.get\_attribute('placeholder'),

'Enter a to-do item'

)

inputbox.send\_keys('Gotowanie')

*# Po wciśnięciu przycisku ENTER następuje aktualizacja strony i listy*

inputbox.send\_keys(Keys.ENTER)

time.sleep(1)

table = self.browser.find\_element\_by\_id('id\_list\_table')

rows = table.find\_elements\_by\_tag\_name('tr')

self.assertTrue(

any(row.text == '1: Gotowanie' **for** row **in** rows)

)

Używamy kilku metod, które Selenium zapewnia do badania stron internetowych: find\_element\_by\_tag\_name, find\_element\_by\_id i find⁠\_ele⁠ment⁠s⁠\_by⁠\_ tag\_name (litera s, oznacza, że zwróci kilka elementów, a nie tylko jeden).

Używamy również send\_keys, które są sposobem Selenium na wpisywanie elementów wejściowych.

Klasa Keys (nie zapomnij jej zaimportować) pozwala nam wysyłać specjalne klucze, takie jak Enter.

Kiedy naciśniemy Enter, strona odświeży się. Funkcja time.sleep jest po to, aby upewnić się, że przeglądarka zakończyła ładowanie, zanim zrobimy jakiekolwiek zapewnienia dotyczące nowej strony.

1. Stworzenie pliku home.html w folderze templates

<html>

<title>Lista do zrobienia</title>

</html>

1. Odpowiednie poprawienie pliku views.py
2. Poprawienie pliku tests usunięcie spacji

**Klient testowy Django**

Jednym ze sposobów sprawdzenia tego jest ręczne wyrenderowanie szablonu w teście, a następnie porównanie tego z tym, co zwraca widok.

Django ma funkcję o nazwie render\_to\_string, która pozwoli nam to zrobić:

1. W pliku tests.py należy edytować funkcję do testowania strony głównej po przez dodanie zmiennej

expected\_html = render\_to\_string('home.html')

1. I odpowiednia wywołanie assertEqual

II sposób (ten sam plik i ta sama funkcja)

1. Zamiast ręcznie tworzyć obiekt HttpRequest i bezpośrednio wywoływać funkcję widoku, wywołujemy self.client.get, przekazując mu adres URL, który chcemy przetestować.
2. Na razie zostawimy tam stare testy, aby upewnić się, że wszystko działa tak, jak myślimy.
3. .assertTemplateUsed to metoda testowa, którą zapewnia nam klasa Django TestCase. Pozwala nam sprawdzić, jaki szablon został użyty do wyrenderowania odpowiedzi (uwaga - będzie działać tylko dla odpowiedzi pobranych przez klienta testowego).

Polecenie

self.assertTemplateUsed(response, 'home.html')

**Edycja frontendu**

Edycja pliku html tak, aby po wykonaniu testów z pliku z zadania 1 dały wynik ok