Testowanie aplikacji internetowej

Wykorzystanie narzędzia Protractor

Dominika Fusek, Edyta Godula, Justyna Hyla

Agenda

- Co to jest Protractor
- Jak działa
- Instalacja frameworka
- Budowa spec oraz config plików
- Lokatory identyfikowania elementów
- Rodzaje wykonywanych akcji
- Operacje wykonywane na listach elementów
- Oczekiwanie na element – wait
- Rodaje asercji
- Co to jest promise
- Page Object Pattern (co to jest?, po co to jest?, jak się tego używa?)

Ontrack® KLDiscovery™

Protractor - wstęp

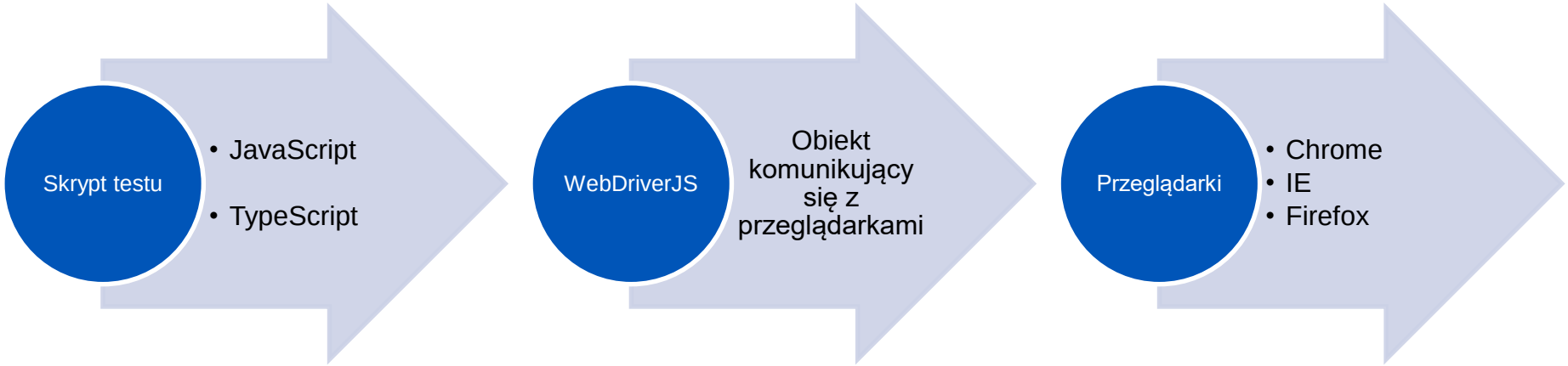
Co to jest?

- Narzędzie pozwalające automatyzować testowanie aplikacji internetowych poprzez wywoływanie
- Dedykowany pod plaftormy Angular and AngularJS - obsługuje specyficzne strategie lokalizacyjne, które umożliwiają testowanie elementów charakterystycznych dla Angular'a bez jakiegokolwiek wysiłku związanego z konfiguracją
- Wywołuje testy aplikacji uruchomionej w prawdziwej przeglądarce, wchodząc z nią w interakcję z użytkownikiem.
- Wykorzystuje bibliotekę Selenium WebDriver

Ontrack® KLDiscovery™

Protractor – jak to działa

Narzędzie bardzo wygodne dla programisty API pozwalające na interakcję z przeglądarką



Ontrack® KLDiscovery™

Protractor - Setup

- Globalna instalacja za pomocą menadżera plików Npm

  npm install -g protractor

- Powyższe polecenie instaluje dwie konsolowe aplikacje:
  - protractor
  - webdriver-manager
- webdriver- manager jest narzędziem pomocniczym, które pozwala łatwo uzyskać instancję działającego serwera Selenium.
- Należy go użyć, aby pobrać niezbędne pliki binarne za pomocą:

  webdriver-manager update

- Aby uruchomić serwer należy wykonać:

  webdriver-manager start

Protractor – spec i config pliki

Plik spec.ts

- Plik spec.ts składa się z:
  - **import** - importowane paczek
  - **describe** – cześć skryptu zawierająca testy dotyczące np. jednej funkcji
  - **it** – część skryptu zawierająca instrukcje wykonujące jeden test z całej grupy describe

```
import { browser, element, by } from 'protractor';

describe('e2e tests', () => {
  it('should open Udemy page', () => {
    expect(browser.getPageSource()).toContain('Top Courses in');
  });
});
```

Protractor – spec i config pliki

Plik spec.ts – zmienne globalne

- Plik Protractor eksportuje zmienne globalne do pliku spec.ts:
  - **browser** – umożliwia wykonywanie operacji na przeglądarce np. browser.getPageSource
  - **element** – wyszukuje wybrany element na stronie, wchodzi w interakcję z DOM(ang. Document Object Model)
  - **by** – zbiór lokatorów za pomocą których wyszukiwany jest element na stronie

Protractor – spec i config pliki

Plik protractor.conf.js

- **allScriptsTimeout** – domyślny czas oczekiwania w milisekundach aż angularowa w pełni zostanie załadowana przed wykonaniem kojenego polecenia,
- **spec** – zbiór plików spec, które mają się wykonać podczas uruchamiania testów
- **capabilities** –początkowe kryteria, ustawienia przeglądarki jakie musi ona posiadać przed uruchomieniem testów

```
allScriptsTimeout: 11000,
specs: [
    './e2e/**/*.e2e-spec.ts'
],
capabilities: {
    'browserName': 'chrome',
    'chromeOptions': {
        'args': [
            '--start-maximized'
        ],
    }
},
```

Protractor – spec i config pliki

Plik protractor.conf.js

- **directConnect** – pozwala kontrolować czy używany będzie serwer Selenium
- **baseURL** – adres testowanej aplikacji
- **framework**– framework testowy, który będzie wykorzystywany do pisania testów, np. Jasmine
- **jasmineNodeOpts** – początkowe ustawienia dla wybranego frameworka, np. pokazywanie koloru w terminalu, domyślny czas oczekiwania zanim test zakończy się niepowodzeniem

```
directConnect: true,
baseUrl: 'https://www.udemy.com/',
framework: 'jasmine',
jasmineNodeOpts: {
    showColors: true,
    defaultTimeoutInterval: 30000,
    print: function() {}
},
onPrepare() {
    require('ts-node').register({
        project: 'e2e/tsconfig.e2e.json'
    });
    jasmine.getEnv().addReporter(new SpecReporter({
        spec: { displayStacktrace: true }
    }));
}
```

- **onPrepare -** określa, które pliki zawierające kod, mają zostać uruchomione

by.css('.myclass')

Protractor - lokatory

- by.css
- by.id
- by.className
- by.model
- by.buttonText
- by.repeater
- by.linkText
- by.name
- by.tagName
- by.xpath

```
element(by.tagName('div'));
```


Ontrack® KLDiscovery™

Protractor – akcje

const elem = element(by.tagName('div'));

- elem.click();
- elem.sendKeys('my text');
- elem.clear();
- elem.getAttribute(value');
- elem.getText();
- elem. isEnabled();
- browser.actions().sendKeys(protractor.Key.TAB).perform();

```
const elem = element(by.tagName('div'))
  elem.clear().then(() => {
    elem.sendKeys('my Text');
})
```

http://www.protractortest.org/#/api

Ontrack® KLDiscovery™

Protractor – lista elementów

const elem = element.**all**(by.tagName('div'));

- elem.first();
- elem.last();
- elem.filter();
- elem.count();
- elem.get(index);
- elem.map();

```
element.all(by.tagName('div')).count();
```

Protractor – wait

- waitForAngularEnabled() – jeśli false, protractor nie będzie czekał na zakończenie zadań angularowych:

```
browser.waitForAngularEnabled(false);
```

- waitForAngular() – oczekiwanie aż wszystkie angularowe

```
browser.waitForAngular();
```

- wait() – oczekujemy na konkretną zadeklarowaną wykonaną czynność

```
return browser.wait(() => fs.existsSync(filePath), 30000).then(() => {
  return fs.pathExists(filePath).then((exists: boolean) => {
    return exists;
  });
```

Protractor – asercja

Sprawdzenie czy wynik wykonanych operacji jest prawdziwy z założeniem:

expect(wartość_pobrana).toBe(wartość_oczekiwana, ,'ewentualna_informacja_zwrotna'*)*

- toBe()
- .not.toBe()
- toEqual()
- Not.toEqual()
- toContain()

```
expect('apple').toBe('apple');
```

```
expect(browser.getPageSource()).toContain('Top Courses in');
```


Ontrack® KLDiscovery™

Protractor – promise

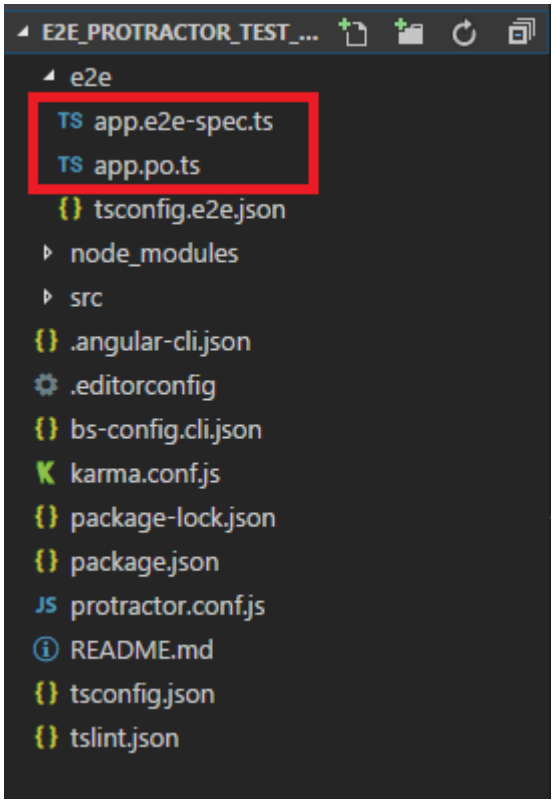
- Klasa Promise reprezentuje ewentualny, spodziewany rezultat, wartość dla zakończonej operacji.
- Każdy promise może mieć trzy stany:
  - Pending
  - Resolved
  - rejected

```
const elem = element(by.tagName('div'))
  elem.clear().then(() => {
    elem.sendKeys('my Text');
})
```

Protractor - Page Object Pattern

definicja

jest to wzorzec projektowy skupiony głównie na obsłudze interfejsu. Polega on na przypisaniu to każdej strony w aplikacji osobnego obiektu, który będzie odpowiedzialny za obsługę wszystkich funkcji danej strony. Z tych obiektów korzysta się przy pisaniu testów automatycznych.



po co?

Używa się głównie przy pisaniu testów, żeby zwiększyć ich czytelność oraz zmniejszyć ilość powtórzeń tego samego kodu. Co więcej jeżeli zmieni się sposób uzyskania efektu końcowego funkcji na danej stronie, wystarczy zaktualizować obiekt odpowiedzialny za jej obsługę, a nie wszystkie testy dotyczące tej strony.

Protractor - Page Object Pattern

jak?

Dla każdej strony tworzy się osobną klasę. Klasa ta zawiera metody odpowiedzialne za uruchamianie wszystkich możliwych funkcji na obsługiwanej stronie. Dodatkowo klasa ta może posiadać również metody sprawdzające stan obsługiwanej strony, dzięki temu assercje w testach można zapisać jedynie jako: ***expect(obiektStrony.metoda).toBe(true);***

Dana klasa zajmuje się tylko obsługą możliwości danej strony, nie powinna natomiast zajmować się obsługą części wewnętrznej (np. WebDrivera dla aplikacji Webowych).

Dana klasa nie musi odzwierciedlać całej strony, może być również odpowiedzialna za sekcję która powtarza się na wielu stronach w aplikacji.

Protractor - Page Object Pattern

podsumowanie

- Publiczne metody obiektu przedstawiają możliwości danej strony
- Klasa nie musi zawsze odzwierciedlać całej strony
- Metody mogą zwracać inne obiekty
- Różne rezultaty dla tej samej akcji są obsłużone przez różne metody tego samego obiektu

Przydatne linki

Protractor

http://www.protractortest.org/#/

Page Object Pattern

http://martinfowler.com/bliki/PageObject.html

http://www.assertselenium.com/automation-design-practices/page-object-pattern/

http://docs.seleniumhq.org/docs/06_test_design_considerations.jsp#page-object-design-pattern

KLDiscovery™

Ontrack®

Dominika Fusek
Test Engineer
dominika.fusek@kldiscovery.com

Edyta Godula
Test Engineer
edyta.godula@kldiscovery.com

Justyna Hyla
QA Engineer
justyna.hyla@kldiscovery.com