

Gdańsk, 02.22.2020

QA Has Power Cypress

BASIC

STXNEXT
python powerhouse

Agenda warsztatów

1. Setup
2. Powitanie
3. Trochę teorii
4. Cypress vs Selenium
5. Drzewo DOM, HTML, CSS
6. Cypress:
 - a. Podstawowe metody
 - b. Struktura
7. Automatyzacja formularza
8. Automatyzacja:
 - a. Fixtury
 - b. Tworzenie własnych funkcji
 - c. Zmienne środowiskowe
 - d. Zapis do pliku
9. Automatyzacja: dynamicznie generowane klasy



About STX Next



We are Europe's largest Python Powerhouse, with over **14 years of experience** and with over **300 passionate individuals** in total, ready to supercharge your project with extraordinary code.

At the core of our company are our **Python** and **JavaScript** and **React Native** developers, experts and enthusiasts of their craft. They are supported by a network of Scrum Masters, Product Owners and Service Delivery Managers to direct and streamline the programming work. As well as Product Design Team to consult and create the most outstanding digital products.

To ensure full efficiency and availability, we use a system as simple as it is effective:

One room. One team. One client. Full focus.

200+

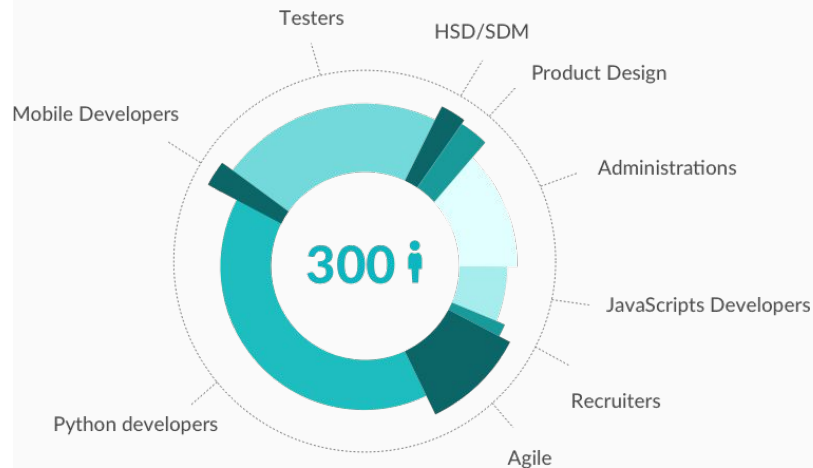
Successful projects

100+

Companies served

14+

Years of the Market



50

Technology Fast 50
2016 CANADA
Deloitte.



Poznajmy się





Po co automatyzować? (I co)

Po co automatyzować?

Automatyzacja w projekcie powinna być przemyślana i zaplanowana.

- 1 Automatyzowanie powtarzających się czynności** tj:
konfiguracja środowiska, generowanie dużej liczby danych testowych, testy regresji, testy wydajnościowe, testy API.
- 2 Szybka informacja zwrotna na temat testowanego produktu**
testy automatyczne wraz z narzędziami do ciągłej integracji pozwalają szybko uzyskać informację o statusie produktu.
- 3 Niektórych testów nie da się zrealizować w sposób manualny**
testy wydajnościowe symulowania działania wielu użytkowników na raz.
- 4 Szybsze wydawanie wersji klientowi** zautomatyzowane testy regresji, wdrożone CI/CD (ciągłe integracja/dostarczanie)





Selenium WebDriver vs Cypress

Selenium WebDriver vs Cypress



Selenium WebDriver jest frameworkiem do testowania aplikacji internetowych, który umożliwia komunikację z przeglądarką i sterowanie nią za pomocą WebDriver API.



Cypress to narzędzie do testowania aplikacji internetowych, które umożliwia testowanie aplikacji w przeglądarce i jest łatwiejsze w użyciu niż Selenium.

Selenium WebDriver vs Cypress

Selektory



<code>browser.find_element_by_css_selector('p.content')</code>	<code>cy.get('p.content')</code>
<code>browser.find_element_by_xpath('//form[input/@name='username'])</code>	<code>cy.xpath('//form[input/@name='username'])</code>
<code>browser.find_element_by_tag_name('h1')</code>	<code>cy.get('h1')</code>
<code>browser.find_element_by_class_name(content)</code>	<code>cy.get('.content')</code>

Selenium WebDriver vs Cypress



Installation



Video



Documentation



Headless



Switch tabs



Several browsers



Community



Jak zacząć?

Selenium do uruchomienia zwykle potrzebuje wcześniejszego zainstalowania powiązań języka i konfiguracji sterowników i innych dodatkowych bibliotek.

Cypress po wskazaniu docelowej lokalizacji może zostać zainstalowany poprzez menedżera pakietów npm.

W przypadku Windows'a wystarczy uruchomić plik .exe, a wszystkie sterowniki zostaną automatycznie zainstalowane.

Pozwala to na rozpoczęcie pracy w kilka minut.

Getting started ✓

Introduction

Selenium installation ✓

Installing Selenium libraries ✓

Installing WebDriver binaries ✓

Installing Standalone server ✓

Getting started with WebDriver ✓

First you need to install t

Java

Installation of Selenium I

```
<dependency>
<groupId>org.seleniumhq.selenium
<artifactId>selenium-java
```

Installing #

```
>_ npm install
```

Install Cypress via npm:

```
$ cd /your/project/path
```

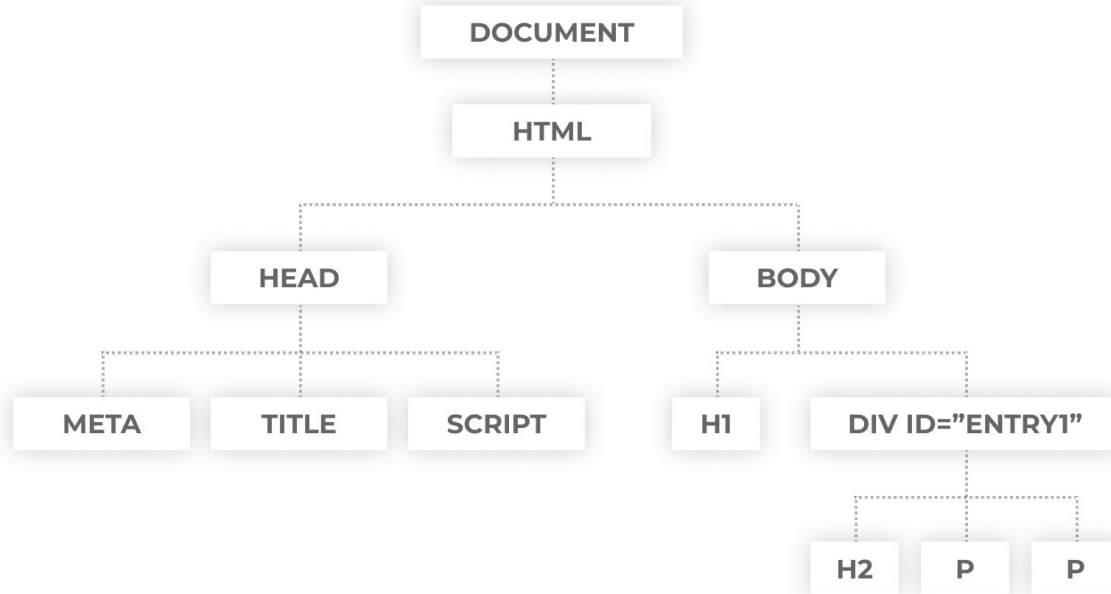
```
$ npm install cypress --save-dev
```

This will install Cypress locally as a dev dependency for your project.



Drzewo DOM, HTML, CSS

DOM - Document Object Model





Podstawowe metody



`cy.visit()`

`cy.get()`

`cy.contains()`

`.should()`

`.click()`

`.type()`

Znajdźmy element

```
<button id="main" class="btn btn-large" name="submission"  
  role="button" data-cy="submit">Submit</button>
```

```
cy.get('button').click()
```



```
cy.get('.btn.btn-large').click()
```



```
cy.get('#main').click()
```



```
cy.get('[name=submission]').click()
```



```
cy.contains('Submit').click()
```



```
cy.get('[data-cy=submit]').click()
```



Asercje przy pomocy metody - .should()



```
// retry until this textarea has the correct value  
cy.get('textarea').should('have.value', 'foo bar baz')|
```

Value

Class

```
// retry until this input does not have class disabled  
cy.get('form').find('input').should('not.have.class', 'disabled')
```

Text

```
// retry until this span does not contain 'click me'  
cy.get('a').parent('span.help').should('not.contain', 'click me')
```

Struktura plików

Po dodaniu nowego projektu Cypress automatycznie wyodrębni sugerowaną strukturę folderów

/cypress

 /fixtures

 - example.json

 /integration

 /examples

 - actions.spec.js

 - aliasing.spec.js

 - assertions.spec.js

 ...

 - viewport.spec.js

 - waiting.spec.js

 - window.spec.js

 /plugins

 - index.js

 /support

 - commands.js

 - index.js

 /screenshots

 /videos

Fixtures - zawiera pliki z danymi które mogą zostać użyte w testach.

Integration - zawiera pliki z Twoimi testami. Testy mogą być zapisane w formatach:

 .js

 .jsx

 .coffee

 .cjsx

Domyślnie znajduje się tu również folder z przykładami.

Aby rozpocząć pisanie testów aplikacji, utwórz nowy plik `app_spec.js`, w folderze `cypress / integration`. Odśwież listę testów w Cypress Test Runner, a nowy plik powinien pojawić się na liście.

Plugins - możliwość rozszerzenia Cypress o własny kod.

Support - `index.js` jest uruchamiany przed każdym plikiem `spec.js`.

Screenshots , Videos - cypress automatycznie tworzy te foldery i zapisuje w nich zrzuty i nagrania z przebiegu i wyniku testów.



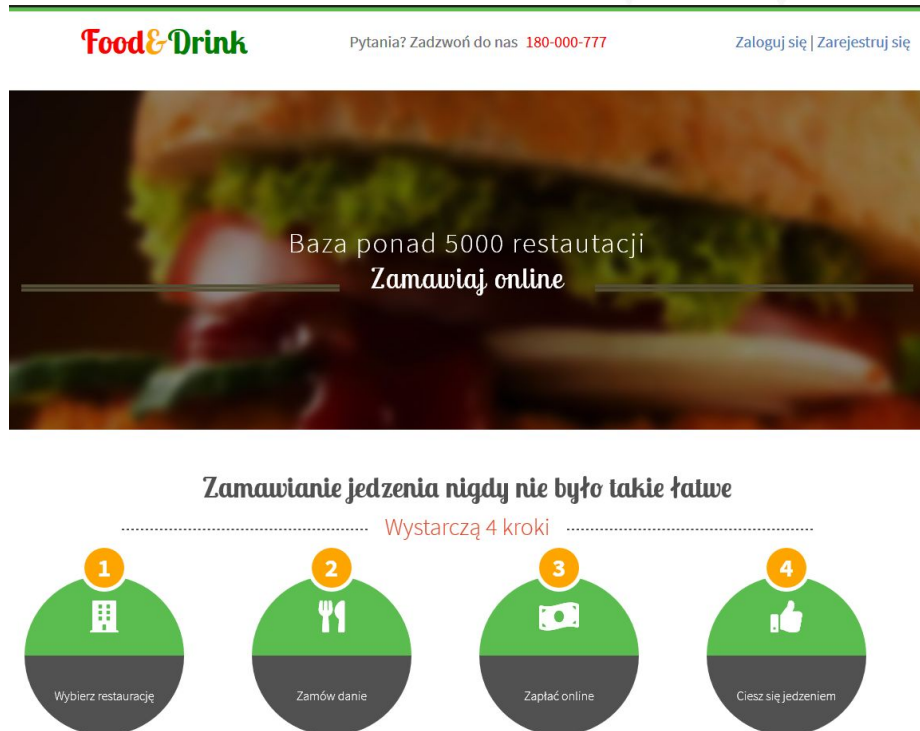


**TESTUJEMY FORMULARZ
LOGOWANIA!**

Plan działania:

Czyli co dokładnie chcemy zrobić i w jakiej kolejności.

1. Wejście na wybraną stronę www.
2. Przejście do strony logowania.
3. Wpisanie loginu.
4. Wpisanie hasła.
5. Kliknięcie przycisku “Zaloguj”.
6. Weryfikacja, czy na pewno znajdujemy się na stronie zalogowanego użytkownika.





1. Wejście na wybraną stronę www.

Przy pomocy `cy.visit` przechodzimy na wybrany pod wybrany adres URL. Jednocześnie sprawdzamy, czy na stronie widoczny jest element dla niej odpowiedni.

```
before('navigate to home page', function() {  
  cy.visit('http://qa-has-power.herokuapp.com/');  
})  
  
it("login happy path", function() {  
  cy.contains('Food&Drink')  
})
```

Ciemność widzę, ciemność :(

Niestety loader utrudnia nam debugowanie, na nagraniu testu zakrywa wszystkie kroki, które dzieją się “pod spodem”. Rozwiązanie:

```
1 before('navigate to home page', function() {  
2   cy.visit('http://qa-has-power.herokuapp.com/')  
3   cy.get('.se-pre-con').should('not.be.visible')  
4 })
```

2. Przejście do strony logowania.

Namierzamy przycisk przekierowujący do strony logowanie, naciskamy go i sprawdzamy, czy przeszliśmy na odpowiednią stronę - weryfikacja URL.

```
1 before('navigate to home page', function() {  
2   cy.visit('http://qa-has-power.herokuapp.com/')  
3   cy.get('.se-pre-con').should('not.be.visible')  
4 })  
5  
6 it('login happy path', function() {  
7   cy.contains('Food&Drink')  
8   cy.get('[href="/login"]').click()  
9   cy.url().should('include', '.com/login')  
10  })
```

3. Wpisanie loginu.

Znajdujemy pole na adres email / login, wpisujemy go i sprawdzamy, czy pole przyjęło wpisane dane.

```
6 it('login happy path', function() {  
7   cy.contains('Food&Drink')  
8   cy.get('[href="/login"]').click()  
9   cy.url().should('include', '.com/login')  
10  cy.get('#email')  
11    .type('cypress123@automation.com')  
12    .should('have.value', 'cypress123@automation.com')  
13 })  
14
```


4. Wpisanie hasła.

Znajdujemy pole na hasło, wpisujemy je i sprawdzamy, czy pole przyjęło wpisane dane.

```
6 it('login happy path', function() {  
7   cy.contains('Food&Drink')  
8   cy.get('[href="/login"]').click()  
9   cy.url().should('include', '.com/login')  
10  cy.get('#email')  
11    .type('cypress123@automation.com')  
12    .should('have.value', 'cypress123@automation.com')  
13  cy.get('#password')  
14    .type('AdamKowalski123!')  
15    .should('have.value', 'AdamKowalski123!')  
16 })  
17
```

5. Klikamy 'Zaloguj'.

Znajdujemy przycisk 'Zaloguj', klikamy go.

```
13 cy.get('.btn')  
14     .should('contain', 'Zaloguj').click()  
15 })  
16
```

6. Sprawdzamy, czy jesteśmy na stronie widocznej dla zalogowanego użytkownika.

Weryfikujemy poprawność URL na jaki zostaliśmy przekierowani i wybrany element widoczny na stronie.

```
13 cy.get('.se-pre-con').should('not.be.visible')
14 cy.url().should('include', '.com/dashboard')
15 cy.get('[href="/edit_form"]').should('be.visible')
16 })
```



Fixtury i funkcje

1. Dodajemy test na logowanie z użyciem fixtur

Przykładem użycia `cy.fixture` jest wypełnianie formularza za pomocą danych zapisanych w pliku.

Fixtury zapisujemy w osobnym katalogu i mogą mieć różne rozszerzenia, nie tylko json

```
{  
  "email": "cypress123@automation.com",  
  "password": "AdamKowalski123!"  
}
```



Metoda **cy.fixture()** pozwala nam załadować dowolny plik i wykorzystać jego zawartość w scenariuszach testowych



```
cy.fixture('defaultUser').then((user) => {  
  cy.get('#email').type(user.email)  
  cy.get('#password').type(user.password + '{enter}')  
})
```

2. Logujemy się z użyciem własnej funkcji

Funkcje definiujemy katalogu **support/commands.js** i możemy używać ich wielokrotnie w różnych scenariuszach testowych (po zaimportowaniu!)

```
Cypress.Commands.add("test", () => {  
  cy.log('Function works well')  
})
```

2. Logujemy się z użyciem własnej funkcji

Własne funkcje przydają się, gdy wielokrotnie wykonujemy daną czynność i gdy ta jest złożona, a bazuje na niej wiele przypadków testowych

```
Cypress.Commands.add("login", (user, message) => {  
  return cy.request({  
    method: "POST",  
    url: "/login",  
    form: true,  
    body: user  
  })  
})
```


3. Co jeśli funkcja zawiedzie?

Nasza strona zawsze zwraca status 200, gdy próbujemy się zalogować.

Po zalogowaniu za pomocą `cy.request` nie zostajemy przekierowani

```
Cypress.Commands.add("login", (user, message) => {  
  return cy.request({  
    method: "POST",  
    url: "/login",  
    form: true,  
    body: user  
  })  
})
```

3. Sprawdźmy odpowiedź!



```
Cypress.Commands.add("login", (user, message) => {  
  return cy.request({  
    method: "POST",  
    url: "/login",  
    form: true,  
    body: user  
  }).then((response) => {  
    expect(response.body).to.contain(message)  
  })  
});
```

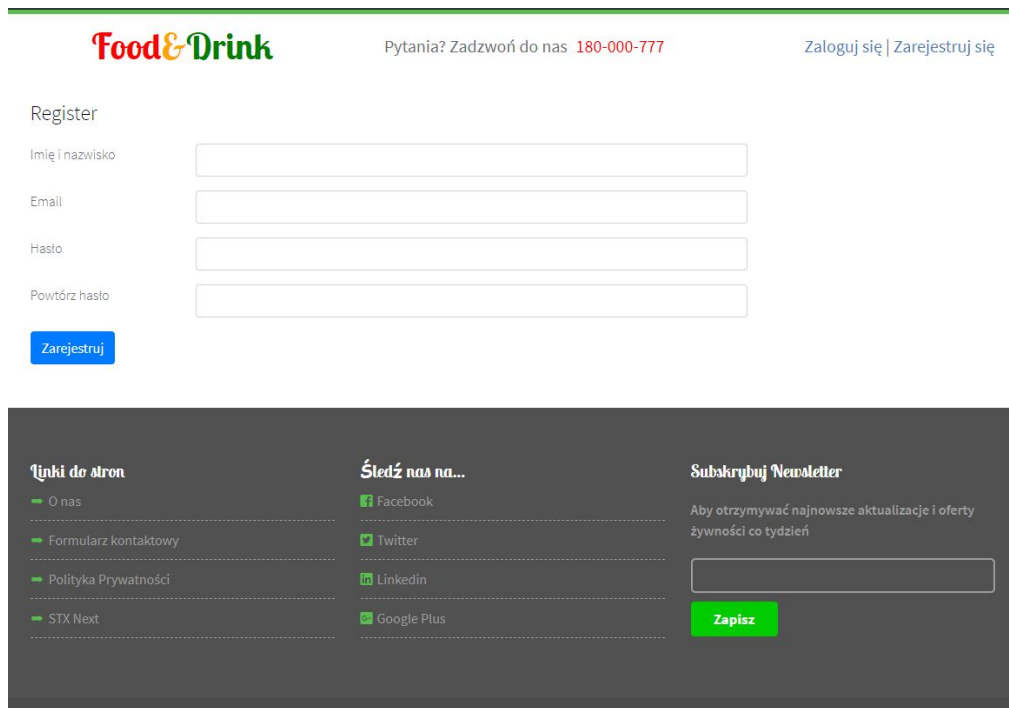


Czas na samodzielne zadanie

Plan działania:

Wykorzystując dotychczas zdobytą wiedzę

1. Zarejestruj nowego użytkownika
2. Wyślij pusty formularz rejestracji i sprawdź wiadomości walidacji
3. Stwórz funkcję wykorzystującą `cy.request`



The screenshot displays the registration interface for 'Food&Drink'. At the top right, there is a large, faint 'X' logo composed of geometric shapes. The header features the 'Food&Drink' logo on the left, contact information 'Pytania? Zadzwoń do nas 180-000-777' in the center, and links 'Zaloguj się | Zarejestruj się' on the right. The main content area is titled 'Register' and contains four input fields: 'Imię i nazwisko', 'Email', 'Hasło', and 'Powtórz hasło'. A blue 'Zarejestruj' button is positioned below the fields. The footer is divided into three sections: 'Linki do stron' with links to 'O nas', 'Formularz kontaktowy', 'Polityka Prywatności', and 'STX Next'; 'Śledź nas na...' with social media icons for Facebook, Twitter, LinkedIn, and Google Plus; and 'Subskrybuj Newsletter' with a text input field and a green 'Zapisz' button.

Food&Drink

Pytania? Zadzwoń do nas 180-000-777

Zaloguj się | Zarejestruj się

Register

Imię i nazwisko

Email

Hasło

Powtórz hasło

Zarejestruj

Linki do stron

- O nas
- Formularz kontaktowy
- Polityka Prywatności
- STX Next

Śledź nas na...

- Facebook
- Twitter
- LinkedIn
- Google Plus

Subskrybuj Newsletter

Aby otrzymywać najnowsze aktualizacje i oferty żywności co tydzień

Zapisz

Przydatne linki :)

Dokumentacja Cypress:

<https://docs.cypress.io/guides/getting-started/installing-cypress.html>

Repo:

<https://github.com/piropetka/qa-has-power-cypress>

Linkedin:

<https://www.linkedin.com/company/stx-next-python-experts/>

STX Next careers:

<https://stxnext.com/wp/working-at-stx-next/>

