

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Warsztaty  Quality Excites Janina Adamiec |  | Jak testować aplikacje mobilne  Calabash |  | jadamiec@future-processing.com |

Jak testować aplikacje mobilne

Wprowadzenie do frameworku - Calabash

Wprowadzenie

Calabash jest to framework, dzięki któremu możemy tworzyć testy akceptacyjne UI. Zapewnia wsparcie dla platform Android oraz iOS.

|  |  |
| --- | --- |
|  | **RUBY**  Do prawidłowego działania frameworka konieczna jest instalacja środowiska Ruby, który jest dostępny na stronie <http://rubyinstaller.org/downloads/>  Konieczne jest pobranie instalatora Ruby 2.3.0 oraz Development Kit „for use with Ruby 2.0 and above”  By sprawdzić czy Ruby jest zainstalowany na komputerze wystarczy w konsoli wpisać: ruby -v . Jeśli Ruby jest zainstalowany na komputerze, zwrócona zostanie wersja oprogramowania np. ruby 2.3.0p0 (2015-12-25 revision 53290) [x64-mingw32] |

|  |  |
| --- | --- |
|  | **ŚRODOWISKO**  Ponadto potrzebne jest zainstalowane środowisko Java Development Kit (JDK) oraz dodane zmiennych środowiskowych tj. JAVA\_HOME (np. C:\Program Files\Java\jdk1.8.0 oraz ANDROID\_HOME kierującą do folderu zawierającego Android sdk. |

|  |  |
| --- | --- |
|  | **SUBLIME TEXT**  Jest to pomocny edytor, który po zainstalowaniu pluginów Ruby oraz Cucumber pozwalają w przejrzysty i wygodny sposób tworzyć nowe kroki testów. |

**Instalacja:**

Wystarczy w konsoli (cmd) wpisać gem install calabash-android. (Lub w razie konieczności sudo gem install calabash-android). Instalacja powinna zostać uruchomiona automatycznie.

Na warsztatach te kroki nie muszą być wykonywane. Środowisko jest gotowe do działania.

Tworzenie szkieletu

**ZADANIE 1**

W plikach dostarczonych na początku warsztatów znajduje się katalog Calabash.

1. Sprawdź czy w folderze znajduje się plik QE2016.apk
2. Skopiuj do schowka ścieżkę do folderu Calabash np. C:\QE2016\Calabash
3. Otwórz konsole CMD
4. Przejdź do katalogu Calabash w konsoli cd C:\QE2016\Calabash

Dzięki tym krokom znajdujemy się w miejscu gdzie możemy stworzyć szkielet testów.

1. Wracamy do konsoli, do miejsca gdzie przechodziliśmy do katalogu Calabash
2. Wpisujemy polecenie **calabash-android gen**
3. Potwierdzamy swój wybór i w ten sposób automatycznie wygenerowaliśmy szkielet testowy.
4. Możemy w eksploratorze plików podejrzeć rezultat: W folderze Calabash powinien znajdować się nowy folder **features** a w nim kolejne katalogi **step\_definitions** oraz **support** oraz najważniejszy plik feature – **my\_first.feature**

Praca z konsolą Calabasha

**ZADANIE 2 – Uruchomienie konsoli**

Konsola Calabasha pozwala nam uzyskać informacje o elementach aplikacji. Dzięki którym można sprawnie tworzyć testy.

1. Uruchom emulator urządzenia np. poprzez Android studio
2. Upewnij się, że w folderze Calabash znajduje się plik QE2016.apk
3. Wróć do konsoli, upewnij się, że nadal jesteś w katalogu Calabash i wpisz: **android-calabash console QE2016.apk**
4. Gdy serwer wystartuje, wpisz: **reinstall\_apps**, aplikacja powinna zostać zainstalowana na emulatorze
5. Wpisz w konsoli by uruchomić aplikację na urządzeniu: **start\_test\_server\_in\_background**

|  |  |
| --- | --- |
|  | Upewnij się, że aplikacja została uruchomiona na emulatorze. Jeśli tak można rozpocząć tworzenie zapytań. |

**ZADANIE 3 – Tworzenie zapytań oraz wykonywanie akcji**

Gdy mam aktywny marker *irb(main) >*możemy zacząć tworzyć zapytania w konsoli. Zawsze zwracana jest tablica wyników. Możemy wyszukiwać widoki, obiekty oraz tworzyć asercje.

* Podstawowe zapytanie, które wyświetli nam wszystkie elementy:
  + query("\*")
* Na podstawie zwróconych elementów możemy odfiltrować dane po klasie. Składnia:

**query(„nazwaKlasy”)** Wpisz poniższe query i obserwuj wyniki:

* + query("View")
  + query("TextView")
  + query("ActionBarContainer")
* Spróbuj sam stworzyć query które wyświetli Ci dane dotyczące buttona.
* **Dodaj manualnie na symulatorze owoc: Banan**, a następnie wylistuj elementy na konsoli. Zwróć uwagę, że zwrócono więcej elementów. Możemy filtrować wyniki na podstawie np. id wtedy składnia wygląda tak query("nazwaKlasy" id: ‘wartość\_Id’)
* Przetestuj działanie query:
  + query("AppCompatTextView")
  + query("AppCompatTextView" id:’text’)
  + query("AppCompatTextView text:’Banan’")
* Gdy nie znamy nazwy klasy a tylko widzimy jakiś element na ekranie możemy zastosować uproszczone query:
  + query("\* marked:’Banan’")
  + query("\* marked:’Category: fruits’")

|  |  |
| --- | --- |
|  | Zwróć uwagę że można filtrować dane po dowolnym parametrze tj. ID, text itp.. Wystarczy znać nazwę klasy i wartości parametrów. Ta umiejętność przydatna jest przy tworzeniu własnych kroków, podobnie jak asercje i wykonywanie akcji, o których mowa poniżej. |

* Calabash oferuje nam możliwość sprawdzenia z konsoli czy dany element istnieje, poczekania aż dany element się wyświetli. Przetestuj działanie takich zapytań:
  + *element\_does\_not\_exist(uniquery)* np. element\_does\_not\_exist(‘textView’). W rezultacie otrzymamy wynik False ponieważ taki element istnieje.
  + *element\_exists(uniquery)* np. element\_exists("textView"), jako wynik otrzymujemy wartość True ponieważ taki element został znaleziony.
  + Wykorzystując rozbudowany sposób tworzenia query*: element\_exists("\* marked:’Banan’ "),* lub *element\_exists("\* text:’Banan’").*
* Symulowanie tąpnięć na ekranie. Przetestuj działanie query (musisz równocześnie mieć widoczny symulator)
  + *touch("\* marked:’Banan’ ")* W efekcie, element się staje aktywny i można go edytować. Idąc dalej możemy zmienić tekst z Banan na Banany
  + *keyboard\_enter\_text("2")*
* Wyjść z konsoli calabasha można za pomocą wpisania **quit** w konsoli

Wykorzystywanie predefiniowanych kroków do tworzenia scenariuszy testowych

**ZADANIE 4 – Uruchomienie domyślnego scenariusza testowego**

1. Otwórz konsolę, przejdź do katalogu Calabash w którym znajduje się folder feature oraz plik APK.
2. Upewnij się że emulator jest uruchomiony
3. Uruchom komendę Calabash-android run QE2016.apk
4. Obserwuj okna emulatora i konsoli
5. Jaki jest wynik testu? Podejrzyj zawartość folderu Calabash – co się nowego pojawiło?

**ZADANIE 5 – Tworzenie prostego scenariusza testowego z wykorzystaniem predefiniowanych kroków**

|  |  |
| --- | --- |
|  | Pełna lista predefiniowanych kroków jest dostępna pod adresem: https://github.com/calabash/calabash-android/blob/master/ruby-gem/lib/calabash-android/canned\_steps.md |

1. Przejdź do katalogu Calabash w eksploatorze, a następnie do folderu features
2. Otwórz plik **my\_first.feature**
3. **Zawartość pliku:**

*Feature: Login feature*

*Scenario: As a valid user I can log into my app*

*When I press "Login"*

*Then I see "Welcome to coolest app ever"*

Domyślny test klika w element Login i weryfikuje pojawienie się określonego tekstu.

1. Spróbujmy zmodyfikować ten scenariusz tak by sprawdzał czy nagłówek jest wyświetlany oraz czy następny ekran się pojawia po kliknięciu w przycisk add.

|  |  |
| --- | --- |
|  | Użyjemy kilku predefiniowanych kroków |

*Feature: Add product*

*Scenario: As an application user I can see ‘add a new product to the list’ screen*

*Given I see the text "QE 2016"*

*When I press image button number 1*

*Then I wait to see "Category"*

*And I wait to see "Priority"*

Uruchamiamy test w konsoli cmd poprzez komendę : calabash-android run NAZWA.APK

**calabash-android run QE2016.apk**

1. Dodajemy kolejny scenariusz w my\_steps.feature pozwalający na dodawanie produktu do listy i uruchamiamy test.

*Scenario: As an application user I can add a new product*

*Given I see the text "QE 2016"*

*When I press image button number 1*

*And I wait to see "Category"*

*And I wait to see "Priority"*

*Then I enter "New product" into input field number 1*

*And I press "fruits"*

*And I press "rtv"*

*And I press "normal"*

*And I press "minor"*

*And I press image button number 1*

1. Dodajemy scenariusz pozwalający na usuwanie dodanego produktu i uruchamiamy test.

*Scenario: As a an application user I can remove a product from the list*

*Given I see the text "New product"*

*When I press "deleteButton"*

*Then I don't see "New product"*

|  |  |
| --- | --- |
|  | Możemy zobaczyć podsumowanie wyników w konsoli:  3 scenarios (3 passed)  17 steps (17 passed) |

Tworzenie własnych kroków testowych

1. Stwórzmy w folderze Calabash -> feature nowy plik **my\_own.fetaure**
2. Dodaj własne opisy, analogicznie dla pierwszego scenariusza ale umieszczając go w my\_own.feature

*Feature: Add product - my own steps*

*Scenario: As an application user I can see add a ‘new product to the list’*

*Given I see Quality Excites header*

*When I click on Add new product button*

*Then I will see "Category"*

*And I will see "Piority"*

1. Uruchom testy (calabash-android run QE2016.APK)
2. Obserwuj wynik:

You can implement steps definitions for undefined steps with these snippets

*Given(/^I see Quality Excites header$/) do*

*Pending # Write code here that turns the phrase above into concreto actions*

*end*

*When(/^I click on Add new product buton$/) do*

*Pending # Write code here that turns the phrase above into concreto actions*

*end*

*Then (/^I will see "([^"]\*)"$/) do |arg1|*

*Pending # Write code here that turns the phrase above into concreto actions*

*end*

1. Wystarczy skopiować to do pliku Calabas\Steps\_definition\Calabas\_steps.rb i uzupełnić   
   definicje kroków na podstawie umiejętności pracy z konsolą calabasha.
2. Uruchom konsolę calabasha i wyszukaj tekst QE2016, widzimy ze QE należy do klasy TextView zatem
3. W Calabash\_stebs.rb wklej:

*Given(/^I see Quality Excites header$/) do*

*wait\_for\_text(„QE2016”)*

*end*

i uruchom testy, zauważ że nie rozpoznaje już tylko 2 kroków a nie 3 jak poprzednio.

*When(/^I click on Add new product button$/) do*

*tap\_when\_element\_exists("\* id:'fabAdd'")*

*end*

*Then (/^I will see a "([^"]\*)"$/) do |arg1|*

*element\_exists("\*:'arg1'")*

*end*

1. Uruchom testy i zobacz jak wygląda feature zawierający napisane przez nas kroki

**ZADANIE 6 (OPCJONALNE) – spróbuj samodzielnie dodać swoje kroki i stworzyć odpowiednie definicje kroków dla scenariusza drugiego i trzeciego**