Automatische GUI Tests mit Frank

März 2012

Julian Dax

Warum automatisches GUI Testing?

- Man muss auf jeden Fall manuell testen
- Manuell testen ist langweilig
- Manuell testen dauert lange
- Manuell testen ist unsicher
- Manuell testen heisst hoffen, dass alles noch funktioniert
- Beim manuellen Testen folgt man oft eingetretenen Pfaden

Warum Frank?

Für Frank[1] sprechen:

- High level
- Aktive Community
- Doku
- Automatisierbarkeit (CI)
- Feature/Behavior Driven Development möglich
- Sehr lesbare Tests

Es gibt noch vieles mehr:

- iCuke[2] - Ähnlich wie Frank, aber letzter commit 2 Jahre her
- UIAutomation - Sehr low level, wenig doku
- Tuneup - Javascript-Framework für UIAutomation
- KIF[3] alleine

Architektur Frank

Frank besteht aus folgenden Komponenten:

- Cucumber[4] - Test Driver

[1] http://www.testingwithfrank.com
[2] https://github.com/unboxed/icuke
[3] https://github.com/square/KIF
[4] http://cukes.info/

Architecture.png

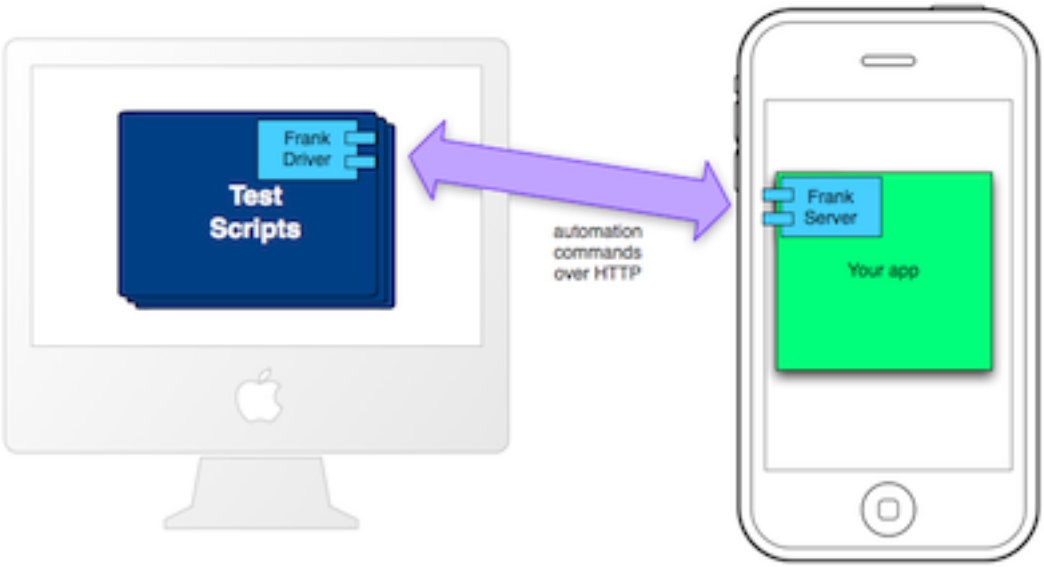

Abbildung 0.1: Frank Architektur

- KIF[5] - UI Interaction
- CocoaHTTPServer[6] - HTTP
- SBJSON[7] - JSON Framework
- Frankly - REST API

Installation

Die ausführliche Anleitung[8] ist auf dem neusten Stand und problemlos nachvollziehbar. Der Punkt “Configuration” ist leider veraltet und muss ignoriert werden. Das Tutorial leider auch.

Erste Schritte nach der Installation:

- Alles aus ~/Library/Developer/Xcode/DerivedData/ löschen
- Die Frankified App einmal im Simulator starten
- ‘cucumber’ im ‘Frank’-Verzeichnis starten
- App Bundle aus der Fehlermeldung kopieren und in env.rb einfügen

[5] https://github.com/square/KIF
[6] https://github.com/robbiehanson/CocoaHTTPServer
[7] https://github.com/stig/json-framework
[8] http://www.testingwithfrank.com/installing.html

- ‘cucumber’ noch mal laufen lassen und freuen

Cucumber

Cucumber[9] ist ein tool zum Ausführen von tests. Cucumber tests schreibt man in Gherkin[ˆgherkin]. Das ist eine DSL für use cases. [ˆgherkin]:https://github.com/cucumber/cucumber/wiki/Gherkin Beispiel:

```
Feature:
  As a user
  I want search for different materials

Background:
    Given I launch the app
    When I touch the tab marked "Material"
    Then I should see a window titled "Material"

Scenario:
    Search for glod
When I search for "Gold"
Then I should see a "Abbrechen" button
And I should see the following:
    |Gelbgold 333/1000 (Au Ag Cu)|
    |Gelbgold 375/1000 (Au Ag Cu)|
    |Gelbgold 416/1000 (Au Ag Cu)|

Scenario:
    Search for wasser
When I search for "Wasser"
Then I should see a "Abbrechen" button
And I should roughly see the following:
    |Wasser (bei 3.98|
    |Wasserstoff|
```

Eigene Steps definieren

Die “Given”/“Then”/“When” statements heissen steps werden in Ruby definiert. Es gibt schon einige per default[10] an denen kann man sich für seine eigenen orientieren. Ausserdem gibt es noch User Contributed Steps[11] auf der Frank Homepage, von denen man auf jeden fall einige

[9] http://cukes.info/

[10] https://github.com/moredip/Frank/blob/master/gem/lib/frank-cucumber/core_frank_steps.rb

[11] http://www.testingwithfrank.com/user_steps.html

gebrauchen kann.

Beispiele:

```
When /^I touch the tab marked "([^\"]*)"$/ do |mark|
  touch( "tabBarButton marked:'#{mark}'" )
end

Then /^I should see a window titled "([^\"]*)"$/ do |expected_mark|
    check_element_exists("navigationBar label marked:'#{expected_mark}'")
end

When /^I search for "([^\"]*)"$/ do |search_text|
  frankly_map( "searchBar textField", 'becomeFirstResponder' )
  text_fields_modified = frankly_map( "searchBar textField", "setText:", search_text )
  frankly_map( "searchBar textField", 'endEditing:', true )
  raise "could not find search field" if text_fields_modified.empty?
  #TODO raise warning if text_fields_modified.count > 1
end
```

UIQuery

Um eigene steps zu definieren muss man UIQuery können. Die UIQuery Doku[12] hilft dabei, außerdem rumprobieren in symbiote.

[12] http://code.google.com/p/uispec/wiki/Documentation#UIQuery