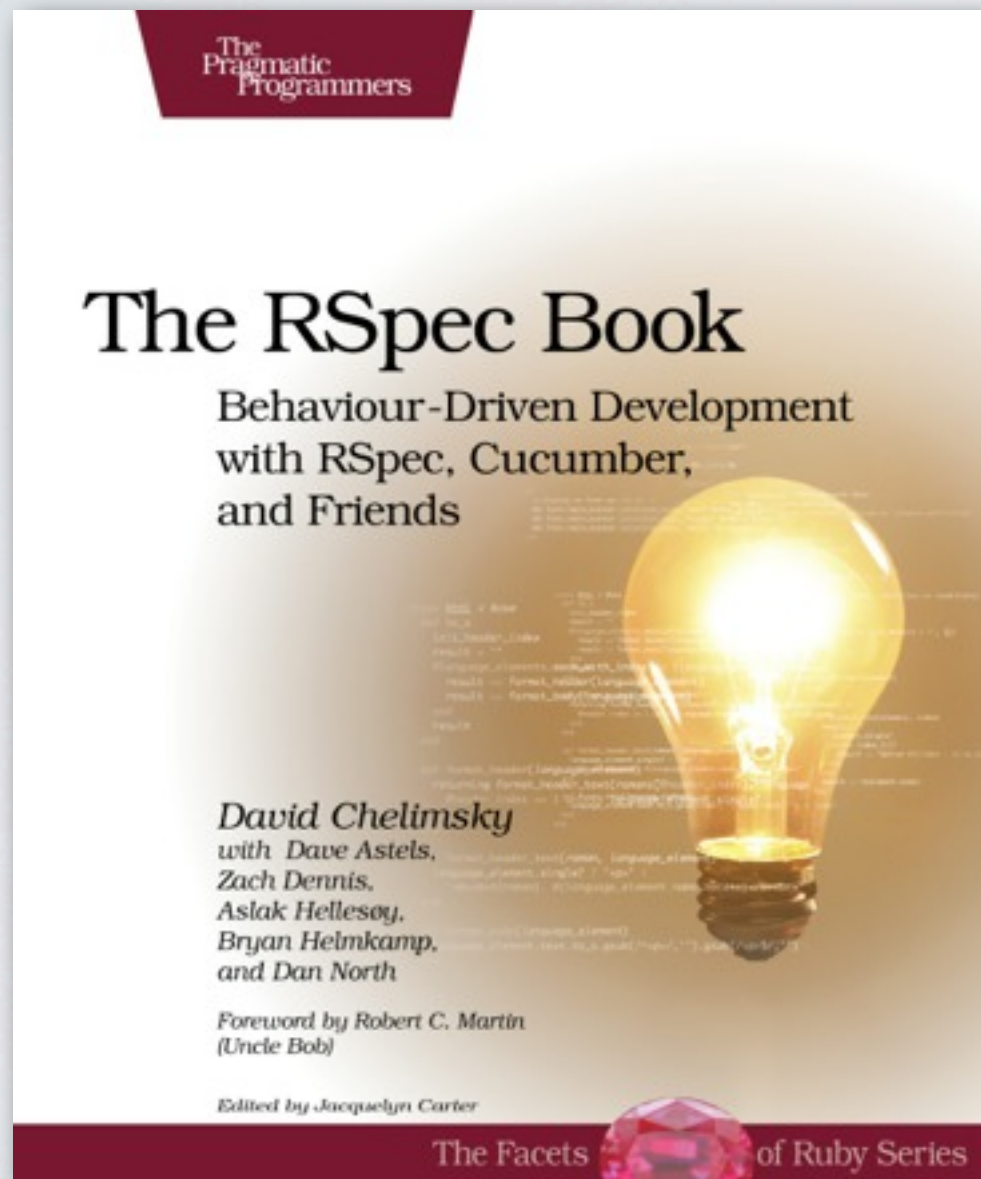


Acceptance Testing von auf Ruby on Rails basierenden Webapplikationen mit Cucumber

Christian Flothmann
Nils Haldenwang

Quellen/Literatur

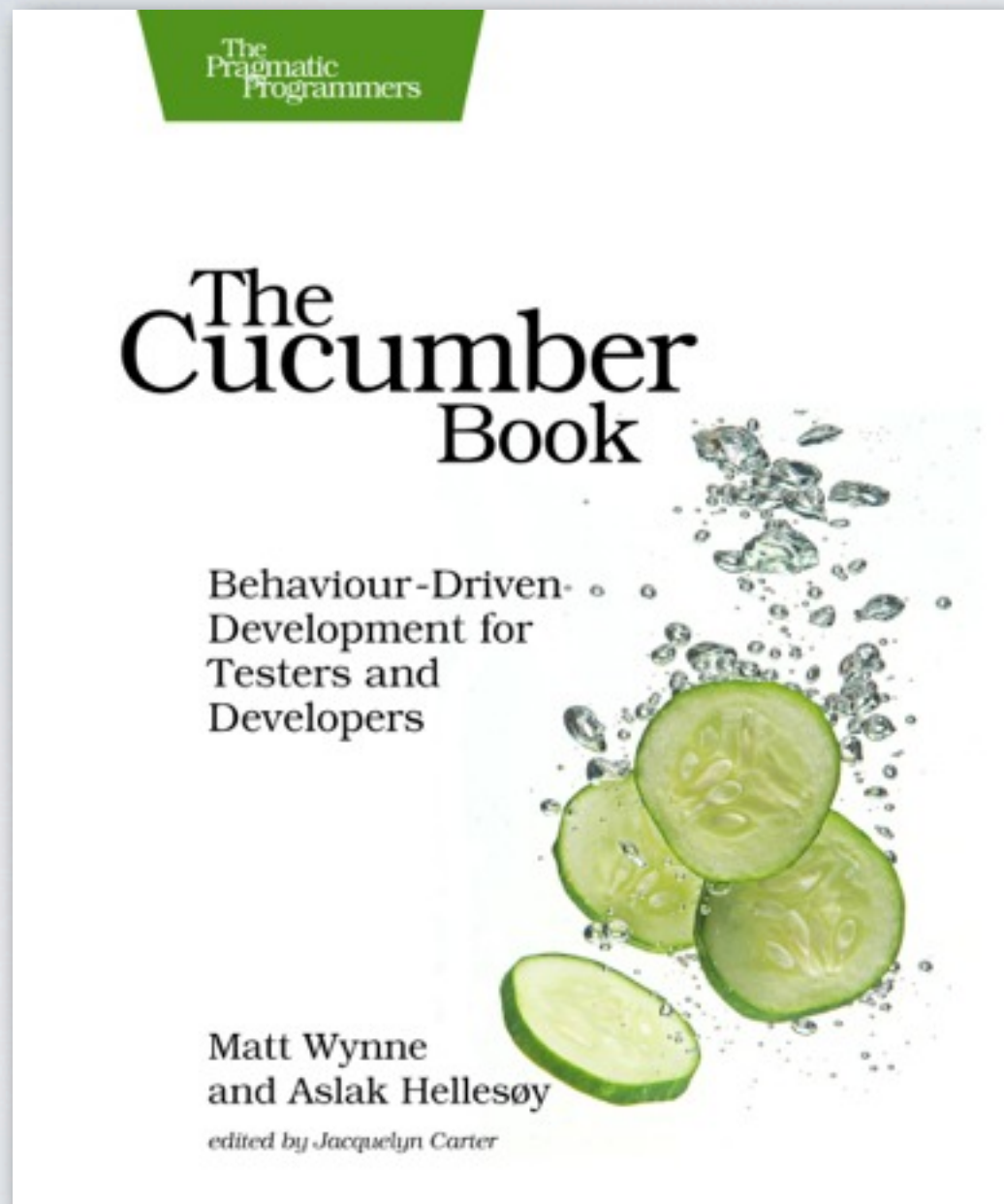


Chelimsky, D. und Astels, D.
und Dennis, Z. und Helleøy,
A. and Helmkamp, B.,
The RSpec Book,
Pragmatic Bookshelf, 2010



Ruby on Rails

Quellen/Literatur



Wynne, M., Hellesøy, A.,
The Cucumber Book,
Pragmatic Bookshelf, 2012



Ruby on Rails

Voraussetzungen

Ruby

<http://www.ruby-lang.org/en/>

Installation: <https://rvm.io>

Ruby on Rails

<http://rubyonrails.org>

Acceptance Testing

- automatisiertes Testen der Spezifikation
- Kriterien werden vom Kunden vorgegeben
- Black-Box testing
- keine hinreichende Qualitätssicherung
- schafft während der Entwicklung eine Basis für weitergehende Qualitätssicherung
- meistens Test-First innerhalb Agiler Vorgehensmodelle

Cucumber

<http://cukes.info/>

Lizenz: MIT

Lauffähig auf allen Plattformen

“Cucumber reads plain-text descriptions of application features with example scenarios and uses the scenario steps to automate interaction with the code being developed.”

Installation

Installation unabhängig von Rails

```
$ gem install cucumber
```

Installation in Rails

Gemfile

```
gem "cucumber"
```

```
$ bundle install  
$ rails generate cucumber:install
```


CucumberGreeter

lib/cucumber_greeter.rb

```
class CucumberGreeter
  def greet
    "Hello Cucumber!"
  end
end
```


Hello Cucumber

```
# features/greeter_says_hello.feature
```

Feature: Greeter says hello

In order to start learning RSpec and Cucumber
As a reader of The RSpec Book
I want a greeter to say Hello

Scenario: Greeter says hello

Given a greeter

When I send it the greet message

Then I should see "Hello Cucumber!"

Feature ausführen

```
$ rake cucumber
```

```
Feature: Greeter says hello
```

```
  In order to start learning RSpec and Cucumber
```

```
  As a reader of The RSpec Book
```

```
  I want a greeter to say Hello
```

```
  Scenario: Greeter says hello
```

```
# ..._hello.feature:9
```

```
    Given a greeter
```

```
# ..._hello.feature:10
```

```
    When I send it the greet message
```

```
# ..._hello.feature:11
```

```
    Then I should see "Hello Cucumber!"
```

```
# ..._says_hello.feature:12
```

```
1 scenario (1 undefined)
```

```
3 steps (3 undefined)
```

```
0m0.002s
```


Step Definitions

Regular Expression

```
require "cucumber_greeter"
```

```
Given /^a greeter$/ do
```

```
  @greeter = CucumberGreeter.new
end
```

Instanzvariable

```
When /^I send it the greet message$/ do
```

```
  @message = @greeter.greet
end
```

```
Then /^I should see "(.*?)"$/ do |greeting|
```

```
  @message.should == greeting
end
```

Ergebnis

```
$ rake cucumber
```

```
[...]
```

```
1 scenario (1 passed)
```

```
3 steps (3 passed)
```

```
0m0.002s
```


Simulation der Anwenderinteraktion mit einer Rails- Applikation

Direct Model Access

```
Given /^a movie$/ do
  @movie = Movie.create!
end
```

- ✓ Schnelle Ausführung
- ✗ Nicht aus Anwendersicht
- ✗ Umgeht Controller und View

Simulated Browser

Webrat:

<https://github.com/brynary/webrat/>

Capybara:

<https://github.com/jnicklas/capybara/>

Installation: Einfügen der entsprechenden Bibliothek ins Gemfile

Simulated Browser: Webrat

```
When /^I create 'Caddyshack' in the Comedy genre $/ do
  visit movies_path
  click_link "Add Movie"
  fill_in "Title", with: "Caddyshack"
  select "1980", from: "Release Year"
  check "Comedy"
  click_button "Save"
end
```

- ✓ Testet das Zusammenspiel aller Schichten aus Anwendersicht
- ✗ Größerer Aufwand führt zu längeren Laufzeiten

Automated Browser

Startet im Hintergrund einen Browser ohne Oberfläche und steuert diesen automatisiert

Selenium:

<http://seleniumhq.org/>

selenium-client:

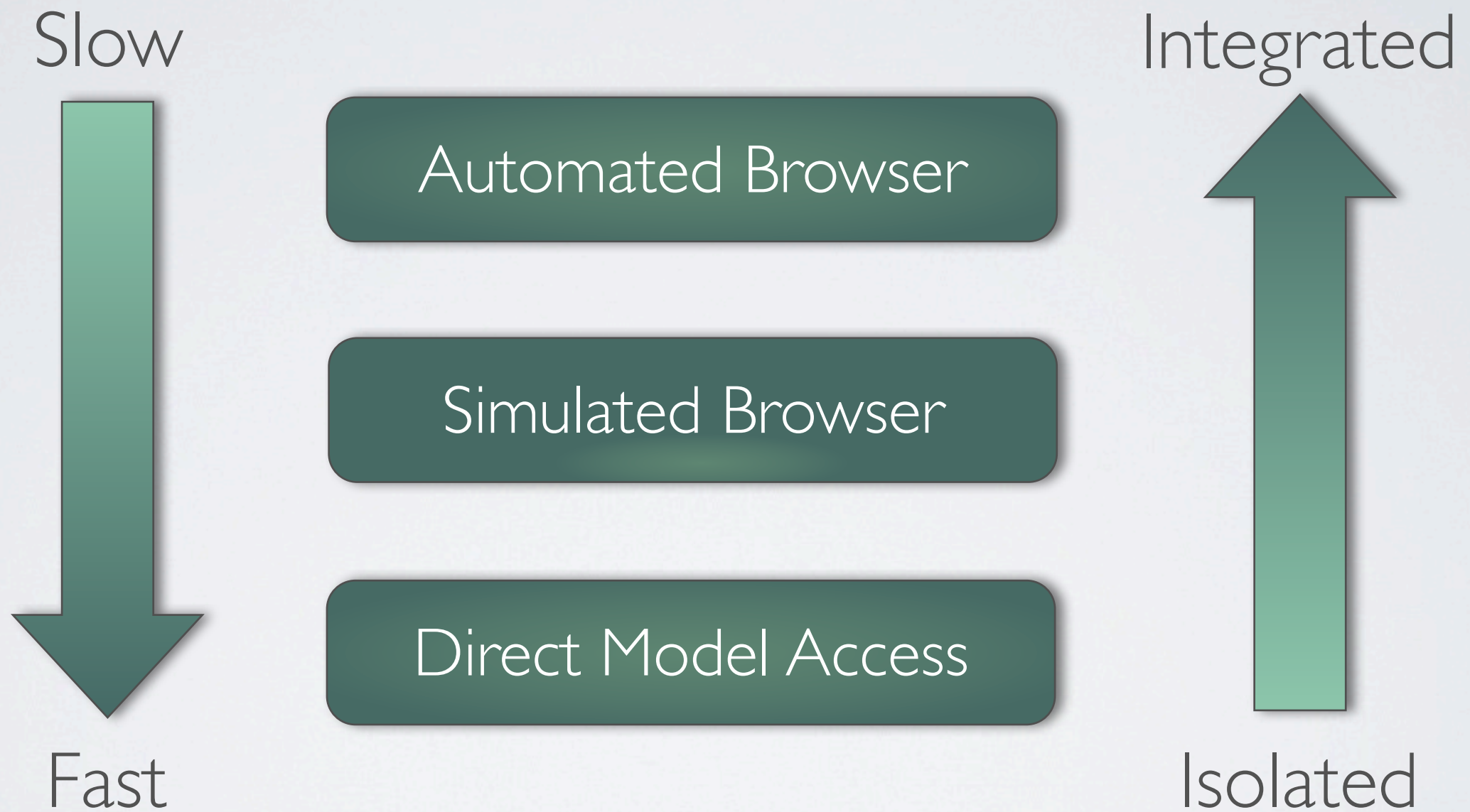
<https://github.com/ph7/selenium-client/>

Automated Browser

API ist kompatibel mit Webrat/Capybara

- ✓ Testet Zusammenspiel aller Schichten aus Anwendersicht
- ✓ Ermöglicht Testen von JavaScript
- ✗ Größerer Aufwand führt zu längeren Laufzeiten
- ✗ Aufsetzen und Debugging der Testumgebung ist sehr komplex

Übersicht



Richtlinien

- Simulated Browser für *When* und *Then*
- Direct Model Access für *Given* (sofern die eigentliche Funktionalität bereits getestet ist)
- Simulated Browser für *Given*, wenn Browserstatus hergestellt werden muss (z.B. Login-Session)
- Automated Browser für kritische JavaScript- und AJAX-Features (nur *happy path*)

Testabdeckung

Standardausgabe nur Binär

```
[...]
```

```
1 scenario (1 passed)
```

```
3 steps (3 passed)
```

```
0m0.002s
```


Anweisungsüberdeckung messen

Möglich mit weiteren Bibliotheken

SimpleCov

<https://github.com/colszowka/simplecov>

Installation: entsprechender Eintrag im Gemfile

Anweisungsüberdeckung

All Files (94.12%)

Controllers (96.24%)

Models (97.3%)

Mails (100.0%)

Helpers (93.96%)

Libraries (87.71%)

Plugins (100.0%)

Controllers (96.24% covered at 5.94 hits/line)

12 files in total. 346 relevant lines. 333 lines covered and 13 lines missed

Search:

File	% covered	Lines	Relevant Lines
app/controllers/data_imports_controller.rb	60.0 %	17	10
app/controllers/application_controller.rb	83.33 %	20	12
app/controllers/lectures_controller.rb	93.55 %	65	31
app/controllers/assignment_fork_sessions_controller.rb	94.12 %	30	17
app/controllers/assignment_collections_controller.rb	96.49 %	128	57
app/controllers/users_controller.rb	96.77 %	54	31
app/controllers/lecture_instances_controller.rb	97.37 %	80	38
app/controllers/assignment_attachments_controller.rb	100.0 %	51	25
app/controllers/assignment_forks_controller.rb	100.0 %	28	12
app/controllers/assignments_controller.rb	100.0 %	108	47
app/controllers/lecture_instance_participations_controller.rb	100.0 %	80	40
app/controllers/study_courses_controller.rb	100.0 %	57	26

Showing 1 to 12 of 12 entries

Anweisungsüberdeckung

app/controllers/data_imports_controller.rb

60.0 % covered

10 relevant lines. 6 lines covered and 4 lines missed.






```
1. class DataImportsController < ApplicationController 2
2.   before_filter :grab_lecture_instance_from_lecture_instance_id 2
3.
4.   def new 2
5.   end
6.
7.   def create 2
8.     OtterImporter.new(@lecture_instance, params[:collection_type], params[:data_file]).import
9.     flash[:notice] = "Daten wurden importiert."
10.    redirect_to @lecture_instance
11.  end
12.
13.  private 2
14.  def grab_lecture_instance_from_lecture_instance_id 2
15.    @lecture_instance = LectureInstance.find(params[:id])
16.  end
17. end
```

app/controllers/data_imports_controller.rb

Demo

Zusammenfassung

Cucumber ist ein im Rahmen Agiler Vorgehensmodelle eingesetztes Werkzeug zum Acceptance Testing.

-  Keine hinreichende Qualitätssicherung
-  Bietet Basis für tiefgreifende Qualitätssicherung
-  Stellt Qualität im Sinne von Erfüllung der Spezifikation sicher
-  Leicht zu installieren und einzubinden
-  Funktionen und Qualitätsmaße ergänzbar