

[IF977] Engenharia de Software

Prof. Vinicius Cardoso Garcia
vcg@cin.ufpe.br :: @vinicius3w :: assertlab.com

AssertLab
Advanced Software and Systems
Engineering Research Technologies



Licença do material

Este Trabalho foi licenciado com uma Licença

**Creative Commons - Atribuição-NãoComercial-
CompartilhaIgual 3.0 Não Adaptada.**

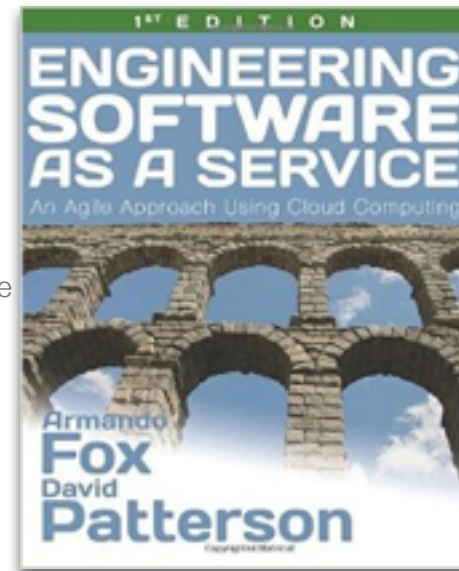
Mais informações visite

<http://creativecommons.org/licenses/by-nc-sa/3.0/deed.pt>



Referências

- A biblioteca do Desenvolvedor de Software dos dias de hoje
 - <http://bit.ly/TDOA5L>
- SWEBOK
 - Guide to the Software Engineering Body of Knowledge (SWEBOK): <http://www.computer.org/web/swebok>
- Engineering Software as a Service: An Agile Approach Using Cloud Computing (Beta Edition)
 - <http://www.saasbook.info/>



Outline

- (ESaaS 7.5) Introducing Cucumber & Capybara
- (ESaaS 7.6) Running Cucumber & Capybara
- (ESaaS 7.8a) Enhancing Rotten Potatoes Again
- (ESaaS 7.8b) Running Rotten Potatoes Again

Usando recursos públicos

- [Introduction to Outside-in Development with Cucumber screencast](#)
 - Um passo a passo na criação de um teste cucumber.
- Artigo [Cucumber Backgrounder](#) no [Cucumber Wiki](#) trata de Cucumber com Rails
- A [página do projeto Capybara](#) inclui um [Getting Started](#) e a [Documentação da API do Capybara](#)
- Assim como acontece com a maioria das linguagens, [a documentação](#) é sua melhor amiga.
- Se você precisar de mais recursos sobre as outras tecnologias que utilizaremos nesta disciplina, certifique-se de conferir a página de apoio [no nosso site](#).
- Além disso, você pode assistir aos Screencasts SaaSTV nessa lista de [reprodução do YouTube](#)



Introdução ao Cucumber & Capybara

Histórias do usuário => Testes de Aceitação

- Não seria ótimo mapear automaticamente histórias de usuários em cartões 3x5 em testes para que o usuário decida se aceita o app?
- Como você combinaria o texto em Inglês para código de teste?
- Como você poderia executar os testes sem um ser humano no circuito para executar as ações?

Cucumber: Uma GRANDE ideia

- Testes a partir de histórias de usuário amigáveis para os clientes
 - Aceitação: garantir um cliente satisfeito
 - Integração: assegurar interfaces consistentes entre módulos, comunicar e interoperar corretamente.
- Cucumber encontra o meio do caminho entre o cliente e o desenvolvedor
 - Histórias de usuários e não código, de modo claro para o cliente e que podem ser usadas para chegar a um acordo
 - Também não são completamente livres, por isso que podem se conectar a testes reais

Exemplo de História do Usuário

Feature: User can manually add movie

1 Feature

Scenario: Add a movie

>= 1 Cenários / Feature

Given I am on the RottenPotatoes home page
When I follow "Add new movie"
Then I should be on the Create New Movie page
When I fill in "Title" with "Men In Black"
And I select "PG-13" from "Rating"
And I press "Save Changes"
Then I should be on the RottenPotatoes home page
And I should see "Men In Black"

3 a 8 passos / Cenário

Cucumber :: História do Usuário, Feature e Passos

- **História do Usuário**: refere-se a uma **feature** única
- **Feature**: ≥ 1 **cenários** que mostram maneiras diferentes que um recurso é usado
 - Palavras-chave Feature e Scenario Identificam os respectivos componentes
 - Mantidos em arquivos **.feature**
- **Scenario**: 3-8 **steps** que descrevem cenário
- Definições de **Steps**: código Ruby para testar os passos
 - Mantidos em arquivos **x_controller.rb**



Palavras-chave dos 5 Steps



1. **Given steps** representam o estado antes do evento:
pré-condições
2. **When steps** representa o evento
 - e.g., simular o usuário apertando um botão
3. **Then steps** representa as pós-condições esperadas;
verificar se é verdadeira
4. / 5. **And** & **But** estendem o passo anterior

Steps => Definição via Regular Expressions

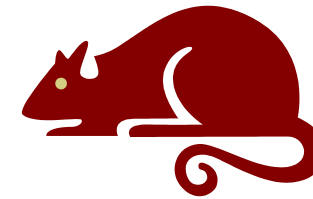
- **Regexes combinam frases em inglês em steps de cenários para definições de steps!**
- `Given /^(:|I)am on (.+)$/`
- `"I am on the Rotten Potatoes home page"`
- Definições de step (código Ruby) como se usar uma string capturada
- `"Rotten Potatoes home page"`

More on "Cuke"

- Precisa instalar a Gem Cucumber
 - Apenas para o ambiente de desenvolvimento e teste, não para o de produção
- Quando instalar o Cucumber, criar as definições de step mais comumente utilizadas
- Necessita de um banco de dados de teste para rodar a app
- Atualize o arquivo **.features** para adicionar features

Usuário "fake" para tentar os cenários?

- Ferramenta que "finge" ser o usuário seguindo os cenários de histórias do usuário
- Capybara simula o navegador
 - Pode interagir com app para receber as páginas
 - Analisa (parse) o HTML
 - Enviar formulários como um usuário qualquer faria



Pergunta

O que é **FALSO** sobre o Cucumber e Capybara?

- A. Definições dos passos (steps) estão em Ruby, e são semelhantes a chamadas de método, enquanto passos (steps) são em Inglês e são semelhantes a definições de métodos
- B. Uma feature tem um ou mais cenários, que são compostos normalmente de 3-8 Passos
- C. Passos usam **Given** para o estado atual, **When** para ações, e **Then** para consequências das ações
- D. Cucumber casa definições de passos para passos de cenários usando expressões regulares, e Capybara "finge" ser um usuário que interage com a app SaaS

15


Enunciate – 3 minutes

1st is FALSE; step definitions are like method definitions, and steps are like method calls

Mix of colors

Pergunta

O que é **FALSO** sobre o Cucumber e Capybara?

- 
- A. Definições dos passos (steps) estão em Ruby, e são semelhantes a chamadas de método, enquanto passos (steps) são em Inglês e são semelhantes a definições de métodos
 - B. Uma feature tem um ou mais cenários, que são compostos normalmente de 3-8 Passos
 - C. Passos usam **Given** para o estado atual, **When** para ações, e **Then** para consequências das ações
 - D. Cucumber cria definições de passos para passos de cenários usando expressões regulares, e Capybara "finge" ser um usuário que interage com a app SaaS

16

Enunciate – 3 minutes

1st is FALSE; step definitions are like method definitions, and steps are like method calls

Mix of colors



Rodando o Cucumber e o Capybara

(Engineering Software as a Service §7.6)

Análise Red-Yellow-Green

- Cor dos passos do Cucumber
- **Green** para passou
- **Yellow** para ainda não implementado
- **Red** para falhando (em seguida, próximos passos serão **Azul**)
- **Goal**: Obtenha todos os passos Green
 - (Daí um vegetal verde para o nome da ferramenta)

Demo

- Adicione uma feature para cobrir as funcionalidades existentes
- Nota: Este exemplo começa errado, ele deve escrever os testes primeiro
- Apenas por razões pedagógicas
- Assista ao screencast: <http://vimeo.com/34754747>

19

Time for demo in Fall 2012: to 54:00 to 1:04:30, so 10 minutes, 30 secs

(need to be smoother – turn on mirroring – practice window selection, copy and paste (or memorize short cuts. Like a navigator)

Type in password saasbook. Start with clear window and center window

Mention preconditions, actions, postconditions

Create .feature file (Addmovie.feature)

Tells you want to do

Copies the error message name

Find route via rake route

Have terminal windows already open for

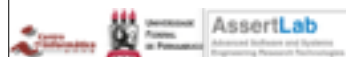
1) In rottenpotatoes

2) Addmovie.feature in editor in features directory

3) paths.rb in editor in features/support directory



Melhorando o Rotten Potatoes



20

(Engineering Software as a Service §7.8)

Adicionando uma nova feature

- E se acrescentarmos algo mais?
 - e.g., incluir um formulário para ser preenchido
 - e.g., precisar de uma interface de usuário
 - e.g., precisa adicionar uma rota para conectar a uma interface para o controlador
 - e.g., incluir tanto um caminho feliz quanto um caminho triste

Integração com o The Movie Database (TMDb)

- Nova Feature: Popular a partir do TMDb versus entrar com os dados manualmente
- Necessidade de adicionar a capacidade de buscar no TMDb a partir da página do Rotten Potatoes
- Precisamos de LoFi UI e Storyboard

Storyboard TMDb

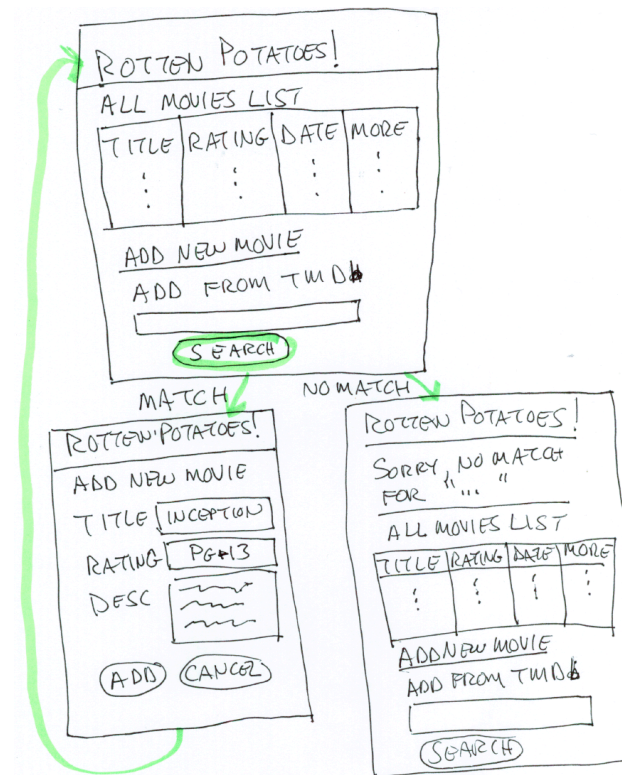


Figure 7.6 of Engineering Software as a Service

23

The home page of Rotten Potatoes, which lists all movies, will be augmented with a search box where we can type some title keywords of a movie and a “Search” button that will search TMDb for a movie whose title contains those keywords.

If the search does match—the so-called “happy path” of execution—the first movie that matches will be used to “pre-populate” the fields in the Add New Movie page that we already developed

If the search doesn’t match any movies—the “sad path”—we should be returned to the home page with a message informing us of this fact.

Easy Story Board

História do Usuário Search TMDb (Figure 7.7 in Engineering SaaS)

Feature: User can add movie by searching in The Movie Database (TMDb)

As a movie fan

So that I can add new movies without manual tedium

I want to add movies by looking up their details in TMDb

Scenario: Try to add nonexistent movie (sad path)

Given I am on the RottenPotatoes home page

Then I should see "Search TMDb for a movie"

When I fill in "Search Terms" with "Movie That Does Not Exist"

And I press "Search TMDb"

Then I should be on the RottenPotatoes home page

And I should see "'Movie That Does Not Exist' was not found in TMDb."

24

Usually start with happy path; but we'd have to implement method for searching and matching TMDb, which we'll do in next chapter.
(Have to explain both BDD and TDD, so we'll do BDD first – so don't do sad path)

Instead, do a sad path where can't find it, and put in dummy method that always fails to find, which makes sad path work.

BDD next will go further in TDD to build the method you need. Not a good idea, but works for this chapter. In future, will go back and forth between BDD design and TDD implementation

HamI para página Search TMDb (Figure 7.8 in Engineering SaaS)

```
-# add to end of app/views/movies/index.html.haml:
```

```
%h1 Search TMDb for a movie
```

```
= form_tag :action => 'search_tmdb' do
```

```
  %label{:for => 'search_terms'} Search Terms
```

```
    = text_field_tag 'search_terms'
```

```
    = submit_tag 'Search TMDb'
```

<http://pastebin.com/18yYBVbC>

25

We will need a new view to go with new UI, and then include a form to fill in

HamI – more concise version of Markup Language than HTML,

1st line is search movie. Need to make a name for the action, we'll call it "search_tmdb"

Label is form where fill in the information for movie when it does match

Line 3: %h1 Search TMDb for a movie

is the text that allows Then I should see "Search TMDb for a movie" to pass.

As with any user interaction in a view, we need a controller action that will handle that Web interaction. In this case the interaction is submitting the form with search keywords.

Line 5 = form_tag :action => 'search_tmdb' do

says that when the form is submitted, the controller action search_tmdb will receive the form submission.

Haml :: últimas 2 linhas

- Este Haml:

```
= text_field_tag 'search_terms'
```

```
= submit_tag 'Search TMDb'
```

- Se transforma neste HTML:

```
<label for='search_terms'>Search Terms</label>
```

```
<input id="search_terms" name="search_terms" type="text" />
```

- para o atributo de tag **label** corresponde o atributo **id** da tag de **input**, a partir do **text_field_tag** helper (acima)
- for attribute of label tag matches id attribute of input tag, from text_field_tag helper (above)

2) the use of the HTML label tag. Figure 2.14 in Chapter 2 tells us that lines 7 and 8 (=...) will expand to the following more verbose HTML markup (<...

The key is that the for attribute of the label tag matches the id attribute of the input tag, which was determined by the first argument to the text_field_tag helper called in line 8 of Figure 7.8. This correspondence allows Cucumber to determine what form field is being referenced by the name “Search Terms” in line 11 of Figure 7.7: When I fill in “Search Terms”. . . .

Vamos tentar o Cucumber?

- Se tentar o Cucumber, ele vai falhar
- Falta a rota
- Além disso **MoviesController#search_tmdb** é a ação do controlador que deve receber o form, ainda não existente em **movies_controller.rb**
- Deve usar Test Driven Development (próxima aula) para implementar o método **search_tmdb**
- Em vez disso, para terminar triste caminho, adicione o método de controle fake que sempre falha

Disparar o Fake Controller quando o form é enviado (POSTed) (Figure 7.9)

```
# add to routes.rb, just before or just  
after 'resources :movies' :
```

```
# Route that posts 'Search TMDb' form  
  
post '/movies/search_tmdb'
```

<http://pastebin.com/FrfkF6pd>

28

we have to make sure there is a route to this new controller action.

Above is the line you must add to config/ routes.rb to add a form submission (POST) route to that action

Método Fake do Controlador: Vai falhar buscando um Filme (Figure 7.9)

```
# add to movies_controller.rb, anywhere inside

# 'class MoviesController < ApplicationController':

def search_tmdb

  # hardwired to simulate failure

  flash[:warning] = "'#{params[:search_terms]}' was not
found in TMDb."

  redirect_to movies_path

end
```

<http://pastebin.com/smw xv70i>

For now, use fix It to fail, which let's us complete the sad path. This “fake” controller method always behaves as if no matches were found. It retrieves the keywords typed by the user from the params hash (as we saw in Chapter 3), stores a message in the flash[], and redirects the user back to the list of movies. Recall from Chapter 3 that we added code to app/views/layouts/application.html.haml to display the contents of the flash on every view.

Pergunta

Qual expressão é **VERDADEIRA**?

- A. Normalmente você completa a fase Behavior Driven Design com Cucumber antes de iniciar a fase de Test Driven Development com RSpec
- B. Normalmente você codificar os caminhos tristes primeiro
- C. Um caminho triste pode passar sem ter escrito o código necessário para fazer um caminho feliz passar
- D. Nenhuma das opções acima é verdadeira

30

VertLeftWhiteCheck1

No, its usually iterative, going back and forth between BDD & TDD


Usually the happy path first

3, Correct. That is the one we did; we put in a dummy method to let us move ahead.

17:30

Pergunta

Qual expressão é **VERDADEIRA**?

- A. Normalmente você completa a fase Behavior Driven Design com Cucumber antes de iniciar a fase de Test Driven Development com RSpec
- B. Normalmente você codificar os caminhos tristes primeiro
-  C. Um caminho triste pode passar sem ter escrito o código necessário para fazer um caminho feliz passar
- D. Nenhuma das opções acima é verdadeira

31

VertLeftWhiteCheck1

No, its usually iterative, going back and forth between BDD & TDD

Usually the happy path first

3, Correct. That is the one we did; we put in a dummy method to let us move ahead.

17:30



Rodando o Rotten
Potatoes novamente

(Engineering Software as a Service §7.8)

Demo

- Adicione a feature para buscar filmes no TMDb
 - Nota: Este será um caminho triste, no qual você não encontrará nada
 - Iremos utilizar um método “fake” (até começarmos a utilizar TDD)
- Assista ao screencast: <http://vimeo.com/34754766>

33

When add a feature, Usually need to write new UI, new step definitions, and new controller method.

Be sure to save edits. Use tabs from pastebin

First step OK, since on RP page

{Then I should see “Search TMDb for a movie”} should fail (red), because we haven't yet added this text to the home page

Try to run cuke on new feature: cucumber features/search_tmdb.feature

Says 1 passed, rest failed

2. In book says to fix that view, need to copy code from Figure 7.7, which is in PasteBin.

It modifies app/views/movies/index.html.haml.

If use vi, be sure to put in paste mode - :set paste (:set nopaste changes mode)

Try to run cuke

Now complains no route

3. In book says to fix route, need to copy code from Figure 7.9 (top). It modifies config/routes.rb

4. We will also set dummy controller method now from Figure 7.9 (bottom). It modifies app/controllers/movies_controller.rb

Try to run cuke, all steps green

Caminho feliz para o TMDb

- Encontre um filme existente. Depois deve retornar para a página do Rotten Potatoes
- Mas alguns passos são repetidos nos caminhos felizes e nos tristes
- Como tornar isso DRY?
- **Background** possibilita ter passos executados antes de cada cenário

TMDb com 2 cenários: Background (Fig 7.10)

Feature: User can add movie by searching for it in The Movie Database (TMDb)

As a movie fan

So that I can add new movies without manual tedium

I want to add movies by looking up their details in TMDb

Background: Start from the Search form on the home page
Given I am on the RottenPotatoes home page
Then I should see "Search TMDb for a movie"

Scenario: Try to add nonexistent movie (sad path)

When I fill in "Search Terms" with "Movie That Does Not Exist"

And I press "Search TMDb"

Then I should be on the RottenPotatoes home page

And I should see "'Movie That Does Not Exist' was not found in TMDb."

Scenario: Try to add existing movie (happy path)

When I fill in "Search Terms" with "Inception"

And I press "Search TMDb"

Then I should be on the RottenPotatoes home page

And I should see "Inception"

Cucumber :: Sumário

- Nova feature => UI para feature, escreva novas definições de passo, ou mesmo escrever novos métodos antes do Cucumber obter os passos verdes
- Normalmente faça os caminhos felizes primeiro
- Background permite que tenhamos cenários DRY para a mesmas features
- BDD/Cucumber testam comportamento; TDD/RSpec na próxima aula é sobre como escrever métodos para fazer todos os cenários passarem

E, concluindo

- Cucumber - "magicamente" mapeia as histórias de usuários no formato cartão de 3x5 para testes de aceitação e testes de integração para a app SaaS



Atividade

- Resolução da Lista de Exercícios 3

<http://bit.ly/if977-hw3>