Testeo web con cucumber

Linux-Malaga.org

imagen: http://freefoodphotos.com/imagelibrary/vegetables/slides/cucumber_slice.html (Creative Commons)

Linux-Málaga

Asociación de Usuarios de Software Libre y Linux de Málaga

http://linux-malaga.org

@linux_malaga

David Moreno

Desarrollador web en Zoconet

https://github.com/linuxonrails

@linuxonrails

¿Qué es Cucumber?

Es una herramienta para BDD

¿Y qué es BDD?

image: https://www.flickr.com/photos/mag3737/ (Creative Commons)

Behavior-Driven Development

(desarrollo guiado por comportamiento)

image: https://www.flickr.com/photos/flickrchimera/ (Creative Commons)

Su principal objetivo es comprobar **qué debería hacer** la aplicación **antes** y durante el proceso de desarrollo.

Esto se consigue mediante tests de comportamiento de la aplicación.

http://en.wikipedia.org/wiki/Behavior-driven_development

Ejemplo:

Queremos comprobar como se comporta una web durante el proceso de **autenticación** (login/logout).

Comportamiento de la autenticación en una web:

- Debemos ver un enlace de Acceder o Salir en la web
- El enlace Acceder debe mostrarnos un formulario de identificación
- **El formulario de identificación, si se rellena correctamente, debería identificarnos en la web.**
- Si nos equivocamos al cumplimentar los datos en el formulario deberíamos ver un mensaje de error y no estar identificados en la web
- Etc...

Ejemplo de test (Gherkin)

Feature: Login
In order to use the web
As a user
I want to login to my account

Scenario: The login form should work
Given I am on the login page
When I fill the login form correctly
And I press "Sign in"
Then I should see "¡Hola usuario!"
And I should not see "Error"

Feature

Una funcionalidad o página que estemos testeando,
por ejemplo: el login en la web.

Cada feature es un fichero único en el directorio features/

Scenario

Un test, todo lo sencillo y concreto que podamos,
por ejemplo: que exista un enlace para hacer login.

Una **Feature** contiene uno o más **Scenarios**

fichero **login.feature:**

```
Feature: Login
In order to use the web
As a user
I want to login to my account

# A comment here
  Scenario: The login page should work
    Given I am on the login page
     When I fill the login form correctly
     Then I should see "¡Bienvenido usuario!"
      And I should not see "Error"

@current
  Scenario: Don't show link to the login page when we are logged in
    Given I am logged in
     When I am on the home page
     Then I should see not see "Log in"
```

Los tests se escriben en **lenguaje natural**

¡El cliente, tu jefe o negocio pueden leerlos sin conocimientos técnicos!

(y escribir nuevos tests ;-)

Cucumber se ha traducido...

...a más de 40 idiomas

```
# language: es
Característica: adición
  Para evitar hacer errores tontos
  Como un matemático idiota
  Quiero saber la suma de los números

  Esquema del escenario: Sumar dos números
    Dado que he introducido <entrada_1> en la calculadora
    Y que he introducido <entrada_2> en la calculadora
    Cuando oprimo el <botón>
    Entonces el resultado debe ser <resultado> en la pantalla

  Ejemplos:
    | entrada_1 | entrada_2 | botón | resultado |
    | 20        | 30        | add   | 50        |
    | 2         | 5         | add   | 7         |
    | 0         | 40        | add   | 40        |
```

...incluyendo el castellano

```
# language: ca
Característica: Suma
  Per evitar fer errors tontos
  Com un matemàtic idiota
  Vull saber la suma dels números

  Esquema de l'escenari: Sumar dos números
    Donat que he introduït <entrada_1> a la calculadora
    I que he introduït <entrada_2> a la calculadora
    Quan premo el <botó>
    Aleshores el resultat ha de ser <resultat> a la pantalla

  Exemples:
    | entrada_1 | entrada_2 | botó | resultat |
    | 20        | 30        | add  | 50       |
    | 2         | 5         | add  | 7        |
    | 0         | 40        | add  | 40       |
```

...y el catalán

https://github.com/cucumber/cucumber/tree/master/examples/i18n

Se permiten
comentarios
con la almohadilla

Ejecutar un solo test:

```
$ cucumber -r features features/login.feature:14
    # ejecuta el scenario de la linea 14 del fichero login.feature
```

Puedes agrupar tests mediante @tags

Ejemplos:

@current, @todo, @my_branch, @work_in_progress, @broken, etc. para ejecutar algunos tests o evitar otros

Ejecutar grupos de scenarios mediante sus tags:

$ cucumber -t @critical -r features features/

$ cucumber -t ~@slow -r features features/

$ cucumber -t @todo,@forms,@breadcrumbs ...

Nosotros nos encargamos
de escribir la **equivalencia**
del test a código
Ruby/Java/otro

BDD parte de TDD

Recordatorio:

BDD es Behavior-driven development

=

Desarrollo guiado por comportamiento

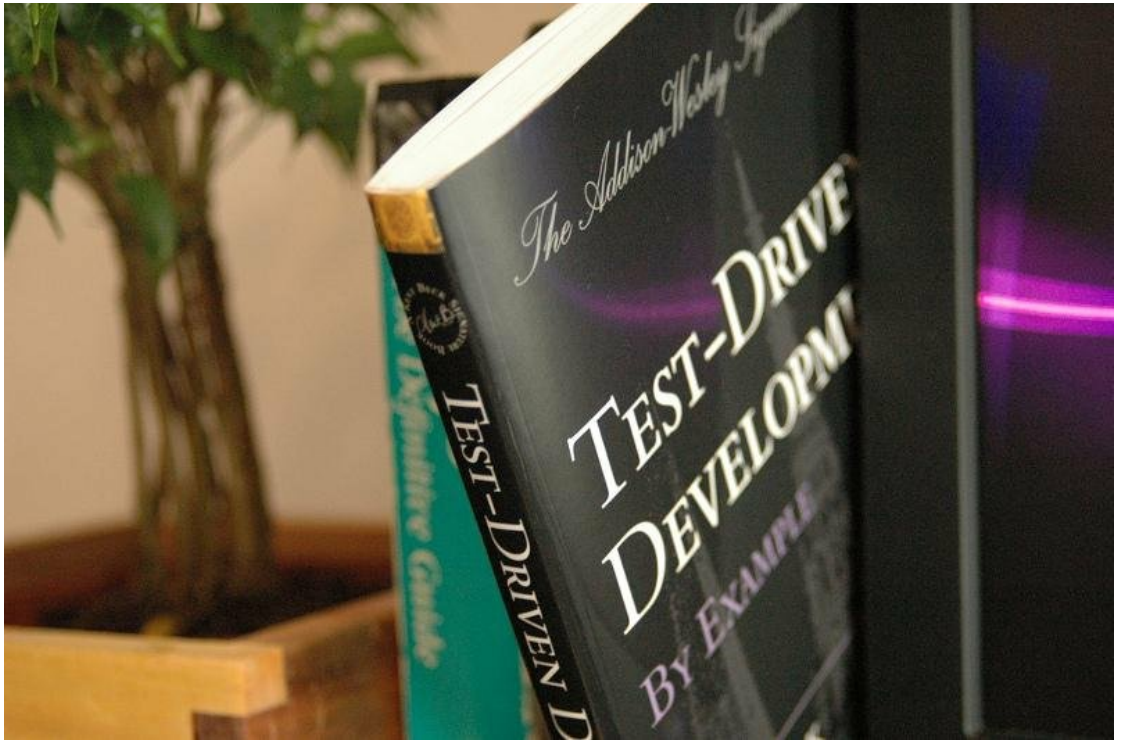
Más siglas y definiciones... m-(

image: https://www.flickr.com/photos/fettlaus/ (Creative Commons)

Image: https://www.flickr.com/photos/oberazzi/ (Creative Commons)

Test-Driven Development o **TDD**

(desarrollo guiado por pruebas)



El **desarrollo guiado por pruebas** (Test-Driven Development o **TDD**) es una práctica de programación que implica:

- Escribir las pruebas primero (Test First Development)
- Refactorizar (Refactoring)

La idea es que los requisitos sean traducidos a pruebas.

Si el código pasa las pruebas se garantiza que cumple con los requisitos que se han establecido.

http://es.wikipedia.org/wiki/Desarrollo_guiado_por_pruebas

El propósito del ***desarrollo guiado por pruebas*** es lograr un código **limpio** que funcione.

Se intenta escribir el **mínimo** código posible

En primer lugar, se escribe una prueba y se verifica que las pruebas fallan.

A continuación, se implementa el código que hace que la prueba pase satisfactoriamente.

Y finalmente se refactoriza el código escrito.

Un ciclo...

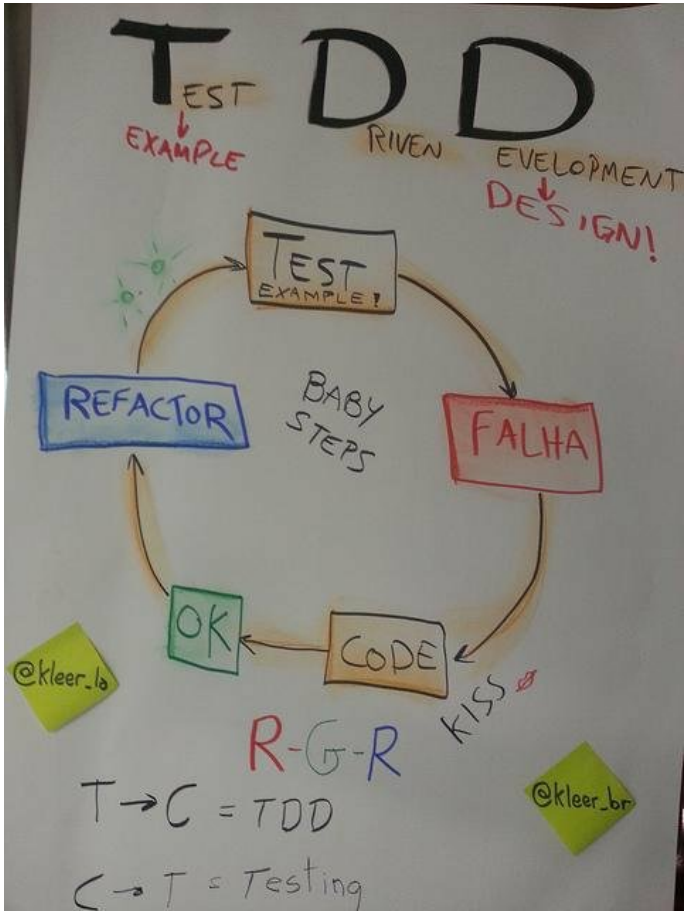


Image: https://www.flickr.com/photos/pablolfc/ (Creative Commons)

Fases del ciclo

1. Escribe un test, debe fallar
2. Implementa el código que debería pasar el test
3. **Ejecuta el test**
4. ¿El test no se pasa? **Refactoriza** y **vuelve al paso 3**

¿Porqué cucumber?

- Escribimos código más simple y limpio
- Buscamos requisitos claros
- Mejoramos el tiempo de desarrollo
- Facilitamos la comunicación con negocio/clientes

```
Scenario: login should work
  Given I am on the home page
    And I follow "Log in"
   When I fill the login form correctly
    And I press "Sign in"
   Then I should see "¡Hola usuario!"
    And I should not see "Log in"
```

Steps

Scenario: -> Descripción breve del test
Given -> Precondición
And -> Más precondiciones
When -> Acción
And -> (otra Acción adicional)
Then -> Resultado comprobable
And -> (otro Resultado comprobable)

No importa el orden de los ***steps*** pero se agradece un mínimo de coherencia semántica :)

```
When I press "Sign in"
```

Corresponde con una definición de step:

```
When /^I press "(.*)"$/ do |button_text|
  click_button(button_text)
end
```

...en el fichero **steps_definitions/web_steps.rb**

¡El casamiento se produce mediante **expresiones regulares**!

Puedes escribir las definiciones de steps con...

- Ruby
- Java (Cucumber JVM)
- .Net
- Flex
- PHP (behat)

Y testear aplicaciones web escritas en **cualquier** lenguaje

https://github.com/cucumber/cucumber/wiki#getting-started

Los steps pueden ser **muy** reutilizables

```
When I login as user "strickland"
```

Y contener a otros steps

```
When /^I login as user "(.*)"$/ do |username|
  step "I register with username #{username}"

  visit login_path # /login.php
  # load user form db
  user = User.find_by_username(username)
  fill_in "Username", with: user.username
  fill_in "Password", with: user.password
  click_button("Sign in")
end
```

Hay steps para depurar durante los tests

Then show me the page

(nos abre el navegador actual con la página visitada en el momento de la ejecución del step)

Capybara

https://github.com/jnicklas/capybara

Navegar

```
visit "/posts/23/comment"
```

Enlaces y botones

```
click_link('id-del-enlace')
click_link('haz click aquí')

click_button('Guardar')

click_on('Enlace o botón')
```

Interaccionando con formularios

```
fill_in "label", with: "nuevo valor"

choose('radio button')

check('a checkbox label')
uncheck('a checkbox label')

attach_file('Image', '/path/to/image.jpg')

select('option', from: 'select')
```

Selectores

```
page.has_selector?('ul li:nth-child(2)')

page.has_css?('form.login button')

page.has_xpath?('//table/tr')
```

Finding

```
find_field('Nombre').value

find_link('Clickeame').visible?

find_button('Enviar').click
```

Scoping

```
within('ul#users-list li') do
  # ...
  click_link('enviar email')
  # ...
end
```

Scripting (ejecutar javascript)

```
page.execute_script("$('a').hide()")

result = page.evaluate_script('4 + 4')
```

Peticiones a servidores remotos

```
Capybara.app_host = 'http://google.com'

visit('/')
```

Y más en la documentación:

trabajar con ventanas del navegador, javascript asíncrono, depuración, modales, etc...

https://github.com/jnicklas/capybara

Capybara-webkit

https://github.com/thoughtbot/capybara-webkit

Testearemos como si utilizáramos el motor de Chrome:

features/support/env.rb:

```
Capybara.javascript_driver = :webkit
Capybara.default_driver = :webkit
```

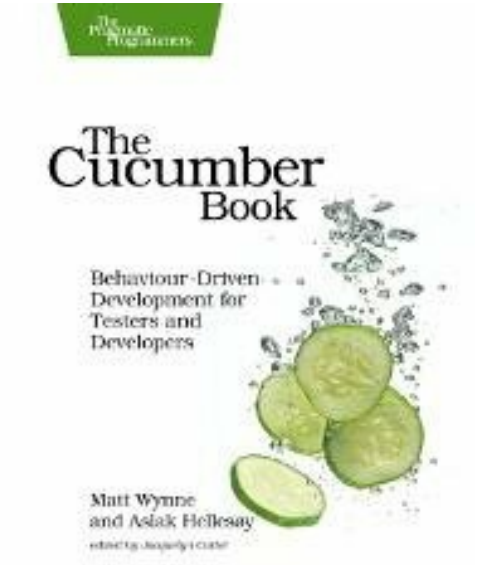
Ejecución muy rápida, con javascript, totalmente en background, etc.

Muy probado. Lo utilizamos asiduamente.

Hay otros: racktest, Selenium, Poltergeist (PhantomJS), etc.

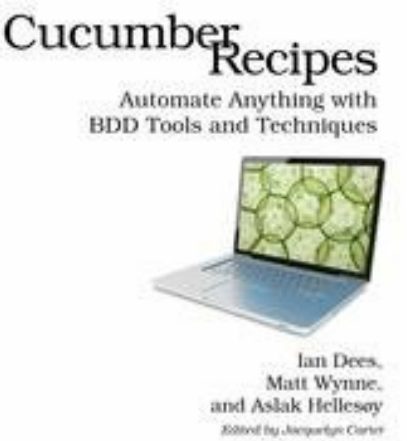
The Cucumber Book

Guía de Matt Wynn y Aslak Hellesøy, creador de Cucumber



The Cucumber Recipes

Guía de ***recetas*** prácticas para escritorio, web, móvil y aplicaciones de servidor



Gracias a todos

¿Preguntas?