Selenium with lettuce and python

¿Qué es Selenium?

Selenium es un entorno de pruebas de software para aplicaciones web. Es posible escribir estas pruebas en numerosos lenguajes de programación tales como java, python, php, ruby, etc. En este caso, nos centraremos en cómo realizar las pruebas en Phyton, pero la realidad es que de un lenguaje a otro las diferencias son mínimas. Estas pruebas pueden ejecutarse utilizando la mayoría de los navegadores web y en diferentes sistemas operativos sin ningún problema.

Selenium está formado por numerosos componentes, entre los que nos encontramos: Selenium IDE, Selenium client API, Selenium remote control, Selenium WebDriver y Selenium Grid. Nosotros realizaremos el siguiente tutorial con Selenium WebDriver.

¿Para qué se utiliza Selenium WebDriver?

Selenium WebDriver se utiliza para testear el correcto funcionamiento de una página web. Es decir que la página haga lo que tenga que hacer. Esto lo realiza enviando comandos a una página web, y obteniendo resultados. Una de las cualidades principales, es que el Selenium WebDriver no necesita un servidor especial para ejecutar los tests, si no que inicia directamente el navegador indicado al comenzar la ejecución del test.

Ventajas de Selenium WebDriver

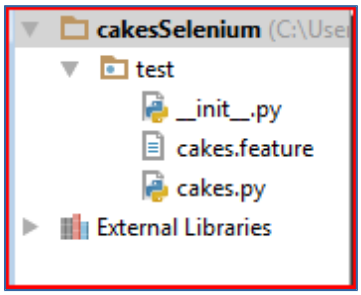
- Testing multibrowser: soportando ejecuciones directas sobre IE, Mozilla y Chrome.
- Control de varios frames, popups, alerts, etc.
- Navegación entre páginas.
- Es muy fácil añadir nuevas funcionalidades a los tests.
- No necesita un servidor propio para ejecutar los test (como ocurre en otros componentes de Selenium).
- Permite realizar pruebas en páginas que se ejecutan en dispositivos móviles gracias al AndroidDriver y IphoneDriver.
- Es muy rápido ya que realiza llamadas directas, sin ninguna intervención externa.

A continuación realizaremos paso a paso algunos ejemplos con Selenium WebDriver y lettuce sobre una página web.

Ejemplo:

Supongamos la web http://www.cheesecake.com/. A continuación realizaremos un testeo de alguna de las funciones posibles que podemos implementar con esta herramienta.

Lo primero que tenemos que hacer es crear un proyecto y dentro un archivo .feature y otro .py como muestra la siguiente imagen:



Una vez hecho esto, procederemos a crear el feature especificando las acciones (de manera más clara posible, con lo que queramos que haga nuestro test). En cuyo caso, el archivo puede ser de la siguiente manera:

```
Feature: found a cake to buy

Scenario: Buy a cake
    Given I click somewhere
    When I fill the text field
    and I select element from the box
    Then I press the bottom
```

Una vez hecho esto, procederemos a completar el archivo .py que contendrá tantos steps como líneas tenga el scenario.

El código de los steps podría ser de la siguiente manera:

```
__author__ = 'kwiznia'
from lettuce import *
from selenium import webdriver

driver = webdriver.Firefox()
driver.get("http://www.cheesecake.com/Cakes-9-Inch.asp")

@step('I click somewhere')
def click_somewhere(step):
    element = driver.find_element_by_xpath("/html/body/form/div[5]/div/div/div[3]/div[3]/d
    #assert "No results found." not in driver.page_source

@step('I fill the text field')
def fill_the_text_field(step):
    quantityText = driver.find_element_by_xpath("/html/body/form/div[5]/div/div/div[4]/div
    quantityText.clear()
    quantityText.send_keys(3)
    zipCodeText = driver.find_element_by_xpath("/html/body/form/div[5]/div/div/div[4]/div/
    zipCodeText.clear()
    zipCodeText.send_keys(10023)

@step('I select element from the box')
def select_element_from_the_box(step):
    select = driver.find_element_by_xpath("/html/body/form/div[5]/div/div/div[4]/div/div[2

@step('I press the bottom')
def press_the_bottom(step):
    addToCartButton = driver.find_element_by_xpath("/html/body/form/div[5]/div/div/div[4]/
```

Lo primero que tenemos que hacer es importar todas las librerías de lettuce y de Selenium, el WebDriver.

A continuación, pondremos como variable global el driver, que es la pagina a la cual le voy a realizar el testeo. Esta variable la pongo como global, ya que la utilizaré en todos los steps. Si no la pusiese como variable global, tendría que inicializarla en cada uno de los steps y se abriría una ventana nueva cada vez que accedo a un step.

A continuación, y como hemos dicho antes, tengo tantos steps como líneas tengo dentro del scenario.

Esta línea, me permite seleccionar un elemento. Es importante destacar que la línea completa es la siguiente:

```
element =
driver.find_element_by_xpath("/html/body/form/div[5]/div/di
v/div[3]/div[3]/div/div[2]/div/table/tbody/tr/td/div/div/a/
img").click()
```

A pesar de que no se aprecie en la imagen, termina con un ".click()". Esta línea me sirve en cualquier caso para seleccionar cualquier elemento, lo único que tengo que hacer es cambiar el xpath para cada uno de los elementos.

Es importante destacar, que además de la localización por xpath que estamos realizando en este caso, también se puede realizar la localización por nombre, id, tag, nombre de la clase o CSS selector con los siguientes métodos driver.find_element_by_name, driver.find_element_by_id, driver.find_element_by_tag_name, driver.find_element_by_class_name o driver.find_element_by_css_selector respectivamente.

Esta línea, permite borrar la información que hay en un cuadro de texto, si es que tiene información por defecto.

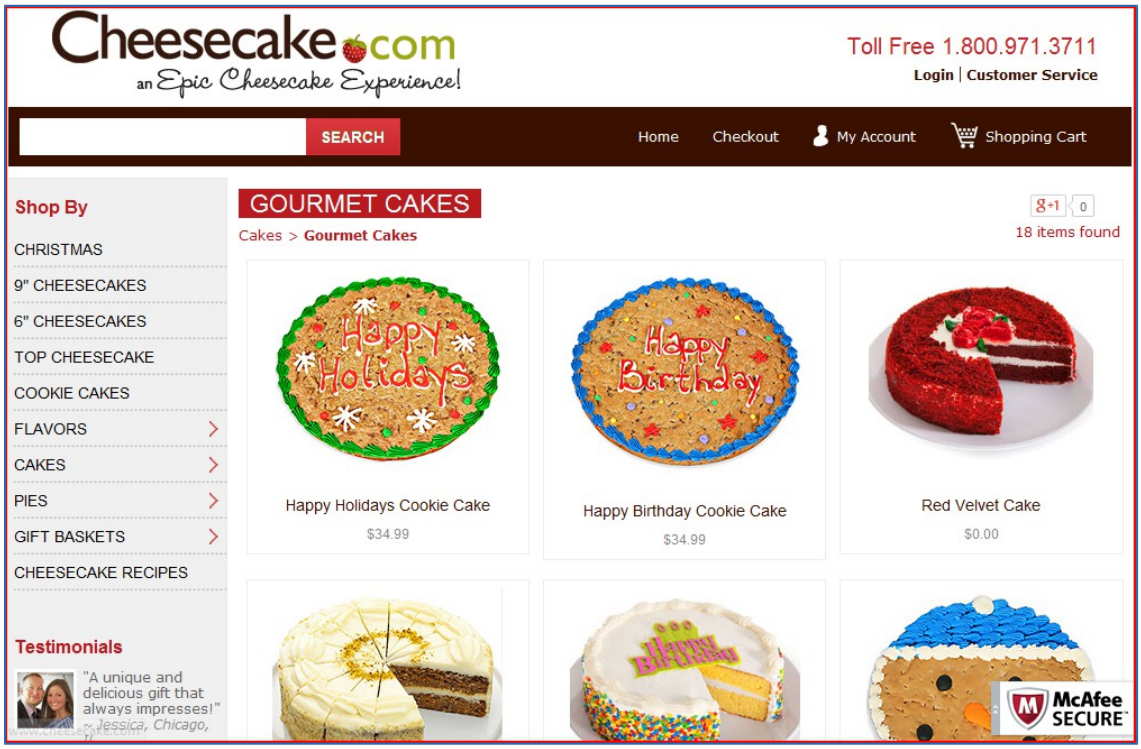
La siguiente línea nos permite escribir información en el cuadro de texto, en el que anteriormente borre su información por defecto. El valor que va entre () es aquel que escribiré dentro del box.

El select, nos permite seleccionar un elemento de un desplegable. El procedimiento es exactamente el mismo que antes, simplemente que el xpath cambia según el correspondiente. En este caso, igual que antes, la línea completa es la siguiente:

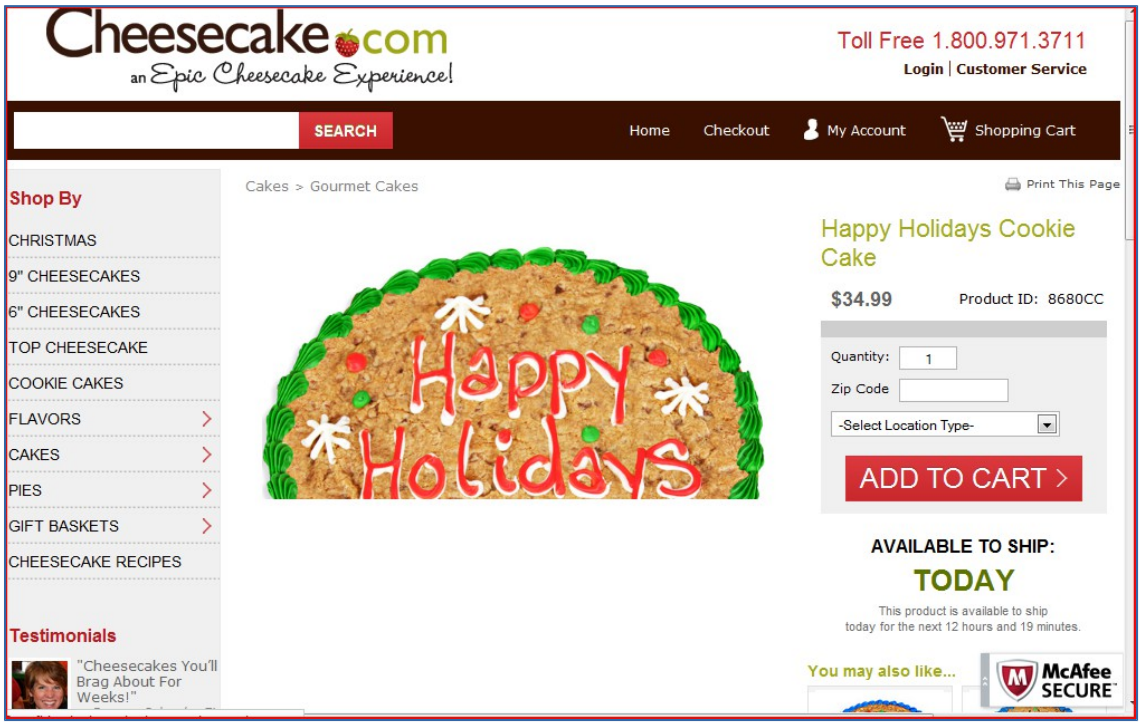
```
select =
driver.find_element_by_xpath("/html/body/form/div[5]/div/di
v/div[4]/div/div[2]/table/tbody/tr[5]/td/select/option[4]")
.click()
```

Resultados obtenidos:

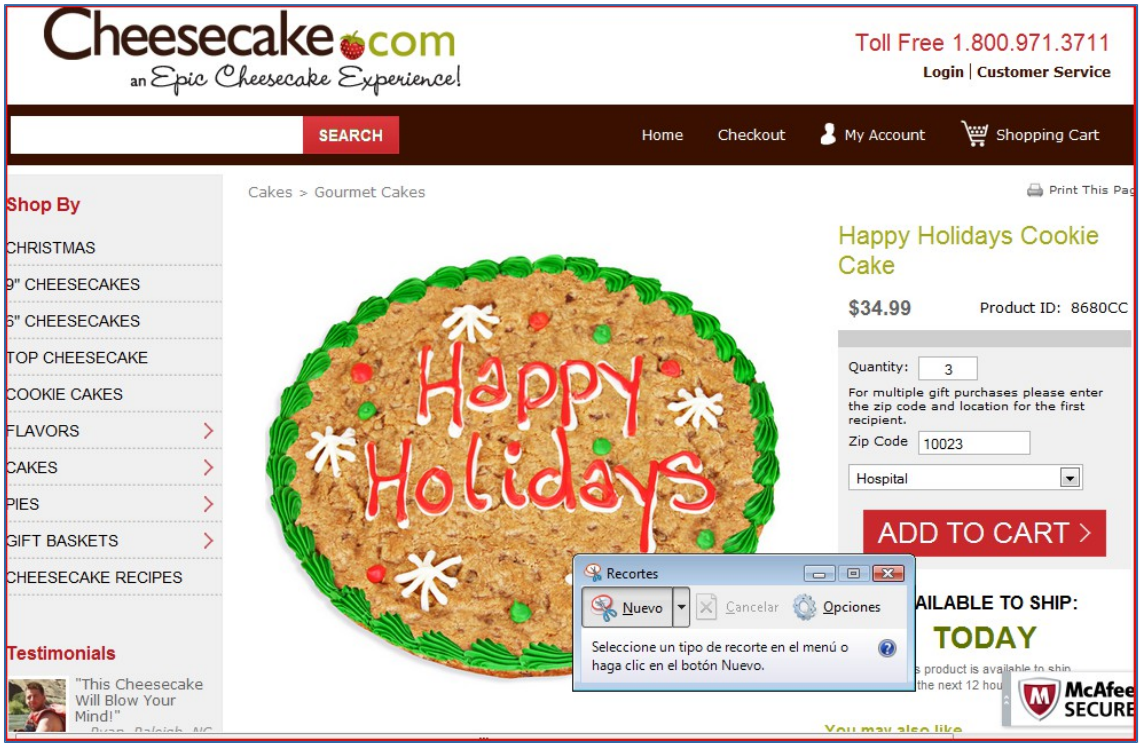
Después del primer step:



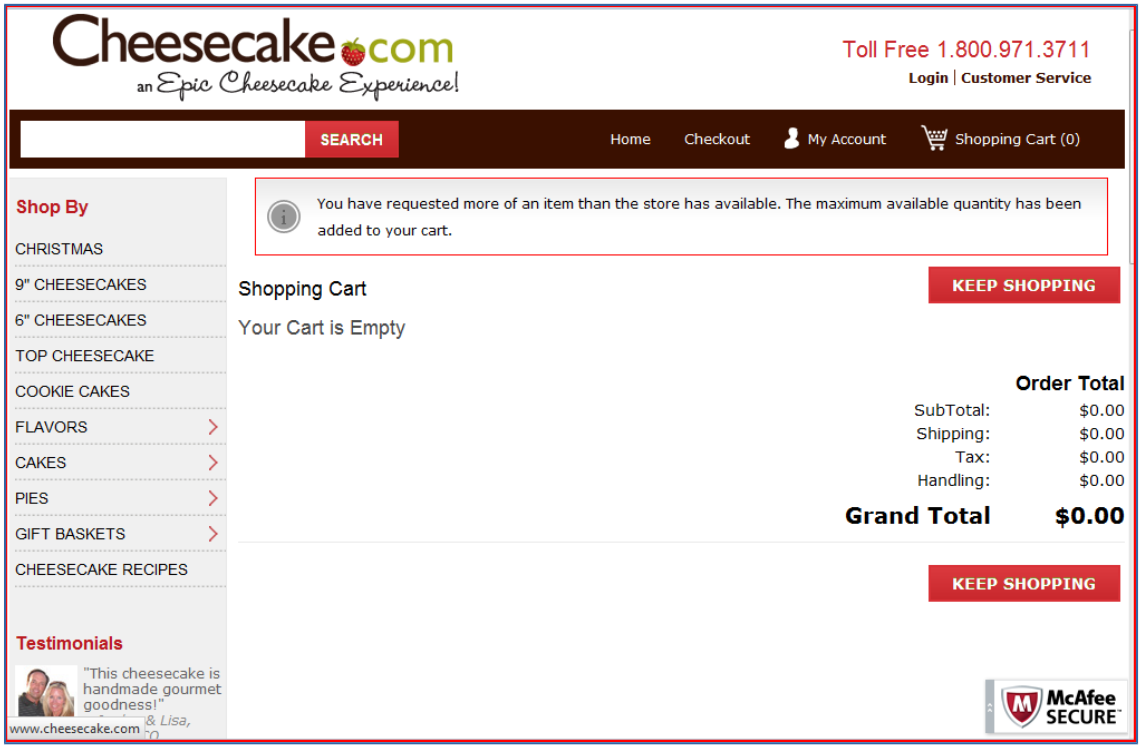
Después del segundo step:



Después del tercer step:



Después del último step:



Otras funcionalidades:

Además de todo esto, que es lo básico, nos interesa saber que otras funcionalidades tiene Selenium. Entre ellas nos encontramos:

- Drag and drop

  Para explicar esto, lo haremos mediante el ejemplo de www.amazon.co.uk.
  Al igual que antes creamos los .feature correspondientes al igual que los .py con los steps necesarios.
  Para realizar el drag and drop se hace de la siguiente manera:

```
element = driver.find_element_by_xpath("/html/body/div[3]/div/div[4]/div[2]/table/tboc
target = driver.find_element_by_xpath("/html/body/header/div/div[2]/div[2]/div[3]/form
ActionChains(driver).drag_and_drop(element, target).perform()
```

  La primera línea selecciona el objeto que desea mover.
  En la segunda línea, selecciona el xpath del lugar donde va a dejarse ese objeto
  En la tercera línea realizo el drag and drop de esos dos objetos.

- Foward:

  Con la línea `driver.forward()` avanzamos a la página siguiente

- Back:

  Con la línea `driver.back()` vamos a la página anterior.

- Explicit wait

  Se utiliza cuando queremos indicarle al test que espere hasta que se cumpla cierta condición.

- Implicit wait:

  Se utiliza para indicar que el máximo de tiempo que puede esperar es el indicado dentro de los paréntesis.

```
driver.implicitly_wait(10)
```

- Close:

  Se utiliza para cerrar la ventana una vez hayamos finalizado el test.

```
driver.close()
```

Excepciones:

Para poder utilizar alguna excepción es necesario importarla con la siguiente línea de código:

```
from selenium.common.exceptions import [nombreException]
```

Dentro de los [] hay que poner el nombre de la excepción. Si queremos importar más de una excepción hemos de hacerlo mediante comas. Las excepciones que podemos manejar son las siguientes:

- ElementNotSelectableException: has intentado seleccionar un elemento que no puede seleccionarse.
- ErrorInResponseException: lanza la excepción cuando hay un problema del lado del servidor.
- InvalidSwitchToTargetException: Cuando la ventana a la que quiero cambiar no existe.
- MoveTargetOutOfBoundsException: Cuando el elemento que quiero hacer drop es invalido.

Estos son algunas de muchas excepciones que podemos utilizar. Para más información sobre las excepciones visitar la página: https://selenium-python.readthedocs.org/api.html

lettuce_webdriver

lettuce_webdriver provee una serie de steps definidos para la utilización de lettuce con Python, utilizando a su vez el paquete de Selenium 2.8 o superior.

Requisitos para la utilización

Es necesario tener instaladas las librerías correspondientes tanto de lettuce como de Selenium 2.8 o superior.

Utilidad

Cuando creamos un scenario, se hace como en cualquier archivo de lettuce, pero esta librería te permite escribir escenarios, sin necesidad de implementar cada uno de los steps ya que estos, ya vienen implementados. Por ejemplo, en el caso de que quiera rellenar un formulario de una página podría tener el siguiente archivo .feature:

```
Feature: Google has my blog where I want it to be

Scenario: Filling out the signup form
  Given I go to "http://foo.com/signup"
   When I fill in "Name" with "Foo Bar"
    And I fill in "Email" with "nospam@gmail.com"
    And I fill in "City" with "San Jose"
    And I fill in "State" with "CA"
    And I uncheck "Send me spam!"
    And I select "Male" from "Gender"
    And I press "Sign up"
   Then I should see "Thank you for signing up!"
```

Y el archivo de steps sería el siguiente:

```
__author__ = 'kwiznia'
from lettuce import before, world
from selenium import webdriver
import lettuce_webdriver.webdriver

@before.all
def setup_browser():
    world.browser = webdriver.Firefox()
```

Esto quiere decir, que dicha librería tiene una serie de steps ya implementados. Esto hace que nuestro código sea mucho más compacto y evitamos la repetición de código entre un test y otro. Dicho esto, cabe destacar que dicha librería tiene una serie de steps ya implementados (además de los mencionados en el ejemplo anterior).

Steps incluidos en la librería

Los siguientes son algunos de los archivos que coinciden con los steps ya implementados (y que no hay necesidad de que conozcamos cual es su implementación):

urls

I visit "http://google.com/"
I go to "http://google.com/"

links

I click "Next page"
I should see a link with the url "http://foobar.com/"
I should see a link to "Google" with the url "http://google.com/"
I should see a link that contains the text "Foobar" and the url "http://foobar.com/"

general

I should see "Page Content"
I see "Page Content"
I should see "Page Content" within 4 seconds
I should not see "Foobar"
I should be at "http://foobar.com/"
I should see an element with id of "http://bar.com/"
I should see an element with id of "http://bar.com/" within 2 seconds
I should not see an element with id of "http://bar.com/"
The element with id of "cs_PageModeContainer" contains "Read"
The element with id of "cs_BigDiv" does not contain "Write"

browser

The browser's URL should be "http://bar.com/"
The browser's URL should contain "foo.com"
The browser's URL should not contain "bar.com"

forms

I should see a form that goes to "http://bar.com/submit.html"
I press "Submit"

checkboxes

I check "I have a car"
I uncheck "I have a bus"
The "I have a car" checkbox should be checked
The "I have a bus" checkbox should not be checked

select

I select "Volvo" from "Car Choices"
I select the following from "Car Choices":
  """
  Volvo
  Saab
  """
The "Volvo" option from "Car Choices" should be selected
The following options from "Car Choices" should be selected:
  """
  Volvo
  Saab
  """

radio buttons

I choose "Foobar"
The "Foobar" option should be chosen
The "Bar" option should not be chosen

text entry fields (text, textarea, password)
I fill in "Username" with "Smith"

Cabe destacar que estos deben estar escritos en ingles, y no en otro idioma si no, no encuentra la coincidencia.

Ventajas de lettuce_webdriver

Las Ventajas que puede aportar la librería de lettuce_webdriver son similares a las aportadas por Selenium, pero esta tiene una ventaja añadida que hace que los tests sean mucho mas compactos y rápidos.

La ventaja principal de utilizar esta librería, es que hay una serie de steps ya implementados. Esto nos permite que nuestro código sea mucho más compacto. Además nos da la posibilidad de no repetir código de un step a otro.

Diferencias entre Selenium y lettuce_webdriver

No existen grandes diferencias entre utilizar uno u otro, pero sí que hay una diferencia especialmente importante a la hora de utilizar uno u otro. La siguiente tabla explicará las diferencias entre uno y otro y sus beneficios.

	Selenium	Lettuce_webdriver
Testing multibrowser	✓	✓
Control de frames, popups y alertas	✓	✓
Navegación entre paginas	✓	✓
No necesita un servidor propio para ejecutar los test	✓	✓
Realiza llamadas directas	✓	✓
Código más Compacto		✓
Steps ya implementados		✓
Repetición de código en los distintos programas	✓	