Práctica 3: Black Box testing

Verificación y Desarrollo de Software

Enrique Sánchez-Bayuela / 2016-2017

Selenium

- Selenium es un entorno de pruebas de software para aplicaciones basadas en la web

- *Selenium automates browser*

Selenium

- Provee una librería para comunicarnos con el navegador e interactuar con las páginas webs
- Funciona en Windows, Mac OS, Unix.
- Puede ser usado en Java, C#, Python, Ruby, PHP…

Selenium - Un poco de historia

- 2004 - Jason Huggins lo desarrolla internamente para ThoughtWorks. Posteriormente desarrollan Selenium Remote Control
- 2005 - Dan Fabulich and Nelson Sproul añaden parches que solucionan problemas con la interacción de ciertos elementos en la web
- 2007 - Huggins es fichado por Google y continúa con el proyecto. Simon Steward toma el control en ThoughtWorks y desarrolla WebDriver
- 2008 - Hanrigou desarrolla Selenium Grid que permite paralelizar los tests
- 2009 - Después de una reunión en Google Test Automation Conference se decide unificar Selenium y WebDriver

Componentes

- **Selenium IDE**
  - Es un Add-On de Firefox que permite grabar el comportamiento sobre una página web para después reproducirlo
  - Los scripts son grabados en Selenese y después se ejecutan sobre Firefox
- **Selenium API**
  - Es una alternativa a Selenese, es una API para escribir los tests en cualquier lenguaje de programación
- **Selenium RC**
  - Es un servidor escrito en Java que acepta comandos para el browser via HTTP
- **Selenium Webdriver**
- **Selenium Grid**

Componentes

- **Selenium Webdriver**
  - Es el sucesor de Selenium
  - Acepta comandos que se envían al browser
  - Estándar W3C
- **Selenium Grid**
  - Servidor que permite paralelizar la ejecución de los tests

Selenium

- Al turrón:
  - Instalemos Selenium para Python
    - `sudo pip install -U selenium`
  - Comprueba que tienes **Firefox** instalado

Ejecuta este código

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Firefox()
driver.get("http://www.google.es")
elem = driver.find_element_by_name("q")
elem.send_keys("u-tad")
elem.send_keys(Keys.RETURN)
driver.close()
```

Cómo buscar elementos en un DOM

- Un DOM no es más que un XML
- No necesitamos herramientas complicadas ni instalar nada, todos los navegadores modernos permiten analizar las páginas webs.

Cómo buscar elementos en un DOM

- Vamos a usar las **DevTools** de Chrome que viene instaladas por defecto.
- Abre Chrome y ves a www.google.es
- Botón derecho en el ratón y pulsa “**Inspect element**”

Cómo buscar elementos en un DOM

- Ahí tienes el HTML completo (y el JavaScript)
- Si quieres saber a cuál corresponde cada uno sólo tienes que pulsar la lupa

- Ahora puedes seleccionar con el ratón cualquier elemento

Cómo buscar elementos en un DOM

- Hay dos formas de seleccionar elementos: XPath y CSS
  - **XPath:** http://www.data2type.de/xml-xslt-xslfo/xpath/?L=3
  - **CSS**: http://www.w3schools.com/cssref/css_selectors.asp

Cómo buscar elementos en un DOM

- Selenium tiene unas cuantas funciones por defecto para hacer estas búsquedas:

- http://selenium-python.readthedocs.org/en/latest/locating-elements.html

Espera por elementos

- No todo es tan sencillo… para saber si se ha cargado una página debes de comprobar que los elementos se hayan cargado correctamente:

- http://selenium-python.readthedocs.org/en/latest/waits.html

Ejercicio - Optativo

- Vamos a hacer un login en Twitter (que ya lo hemos estudiado en esta asignatura este año)

- Cread un usuario ficticio con un correo cualquiera (no se os ocurra coger el vuestro)

Ejercicio - Optativo

- El ejercicio es muy sencillo:
  - Haz un programa que sea capaz de loguearte en Twitter usando ese usuario y contraseña y publique un tweet diciendo: “I’m using Selenium!!”
- Requisitos:
  - El ejercicio debe de estar hecho en Python y debe de usar Selenium
  - Crea un repo en Github y pásame el link por Blackboard

Práctica

- Esta práctica consistirá en añadir una web sencilla que nos permita introducir por pantalla un texto y nos devuelva una lista con las palabras y su número de apariciones.
- Para asegurarnos de que nuestra práctica es correcta añadiremos criterios de aceptación usando BDD
- Para asegurarnos de que estos criterios de aceptación se cumplen los automatizaremos usando Selenium/Webdriver

Práctica

- Crea una **web sencilla** usando Django (https://www.djangoproject.com/) o Flask (http://flask.pocoo.org/)

- Esta web deberá de tener un formulario muy sencillo con:
  - Un **textfield** en el que podrás escribir un texto libre (máximo 100 caracteres)
  - Dos botones:
    - **Reset**: borra todo lo que haya en el textfield
    - **Execute**: Arranca el proceso que habéis hecho de cálculo de palabras
  - Un **textfield/lista/lo que queráis** que muestre las palabras y el número de apariciones de las mismas

- Forms en Django: https://docs.djangoproject.com/en/1.10/topics/forms/
- Forms en Flask: https://blog.miguelgrinberg.com/post/the-flask-mega-tutorial-part-iii-web-forms

Práctica

- El funcionamiento de la web es sencillo:
  - Un usuario (no hace falta login/registro/etc) puede introducir **cualquier texto en el textfield** (con un máximo de 100 caracteres)
  - Si el usuario pulsa el **botón Reset todo el texto que haya en textfield deberá de desaparecer**. En caso de que no hubiera texto escrito el botón Reset no deberá de hacer nada.
  - Si el usuario pulsa el botón **Execute** y hay texto, la web deberá de **mostrar por pantalla un listado con las palabras y el número** de apariciones ordenadas de **mayor a menor** y, de igual forma, deberá de **borrarse el texto que** aparece en el textfield. En caso de que no hubiera ningún texto el botón no tendrá ningún efecto.

Práctica

- Antes de empezar a picar define en BDD las features y los escenarios que quieres desarrollar
  - Utiliza Lettuce que es la librería BDD para Python: http://lettuce.it/tutorial/simple.html
  - Crea un directorio features dentro de tu directorio tests y añade ahí los features que crees: http://lettuce.it/tutorial/simple.html#lettuce-project-structure

Práctica

- Define unos ids para tus elementos HTML de manera que sepas de antemano cómo referirte a ellos y cómo utilizarlos (https://www.w3schools.com/tags/att_global_id.asp)

- Añade a tus tests BDD automatización con Selenium/Webdriver de modo que tengas tests automáticos que pruebe la interacción con la página web (https://es.testingbot.com/support/getting-started/lettuce.html)

Práctica - Entrega

- Como la práctica es “complicada” tenéis dos semanas para entregarla. La fecha tope de entrega será el 4 de Mayo antes de las 23:59.
- La entrega será como siempre: enlace a vuestro repo de Github mediante Blackboard.
- Se valorará la calidad de los escenarios en BDD y la automatización de los mismos