Becoming a UI Testing Rock Star

Introduction

Selenium is a cross-platform/browser software testing framework for web applications that provides tools for writing test cases without learning how unit test frameworks work.

We have used JUnit/HtmlUnit for the integration testing for our projects. Although it works, it takes a lot of time for the developers to write tests against the user interface. Besides, it is harder to write the tests since you cannot see what you test until you run it. Using Selenium allows developers and even functional users to write test scripts through a Firefox plugin. The plugin records the actions you have done and turn them into test scripts. You can export the test scripts or test suites to different format, e.g. TestNG, JUnit, etc., and integrate those tests with your existing test setup.

The purpose of using Selenium is to leverage the burden of writing integration / web tests.

Tools

- Java SDK
- Selenium IDE
- Selenium Server (previously Selenium RC)
- selenium-java-client-driver.jar
- Firefox 3.x (Selenium IDE doesn't support Firefox 4 when this guide was created)

Technical Details

Installing Selenium IDE

Selenium IDE is a Firefox plugin that allow users to record and automate the UI test. It provides the functionalities to create single tests, test suites, and export the test scripts to different formats based on the programming languages picked by the user , e.g. html, java (JUnit / TestNG), ruby, php, perl, c#, python, etc.

Download information could be found at:
http://seleniumhq.org/projects/ide/

Downloading Selenium Server

Selenium server is a Jetty integrated web server that acts as a HTTP proxy for web requests. It has the ability to launch and kill browsers. By default, it listens to the port 4444 for incoming requests when it is started.

Download information could be found at:
http://seleniumhq.org/projects/remote-control/

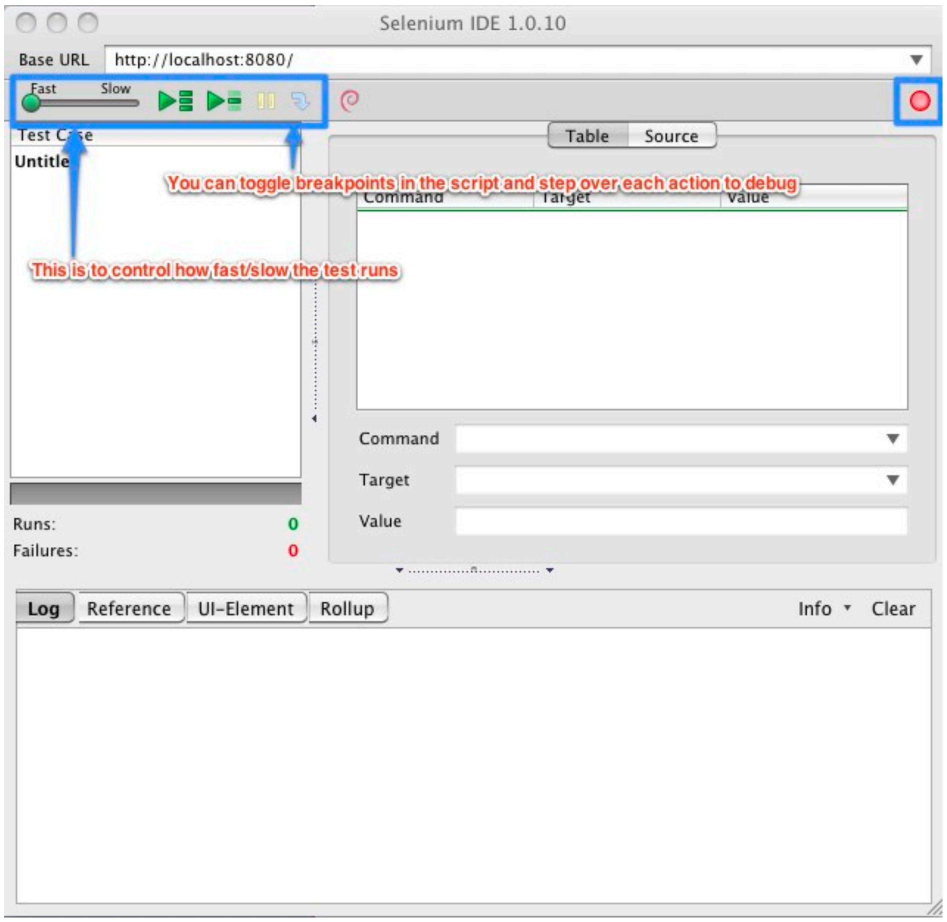
Downloading the Selenium Java Client Driver

This is used when you want to run Selenium tests as JUnit tests.

Download information could be found at:
http://selenium.googlecode.com/files/selenium-java-2.0b3.zip

How to create a Selenium test script using Selenium IDE

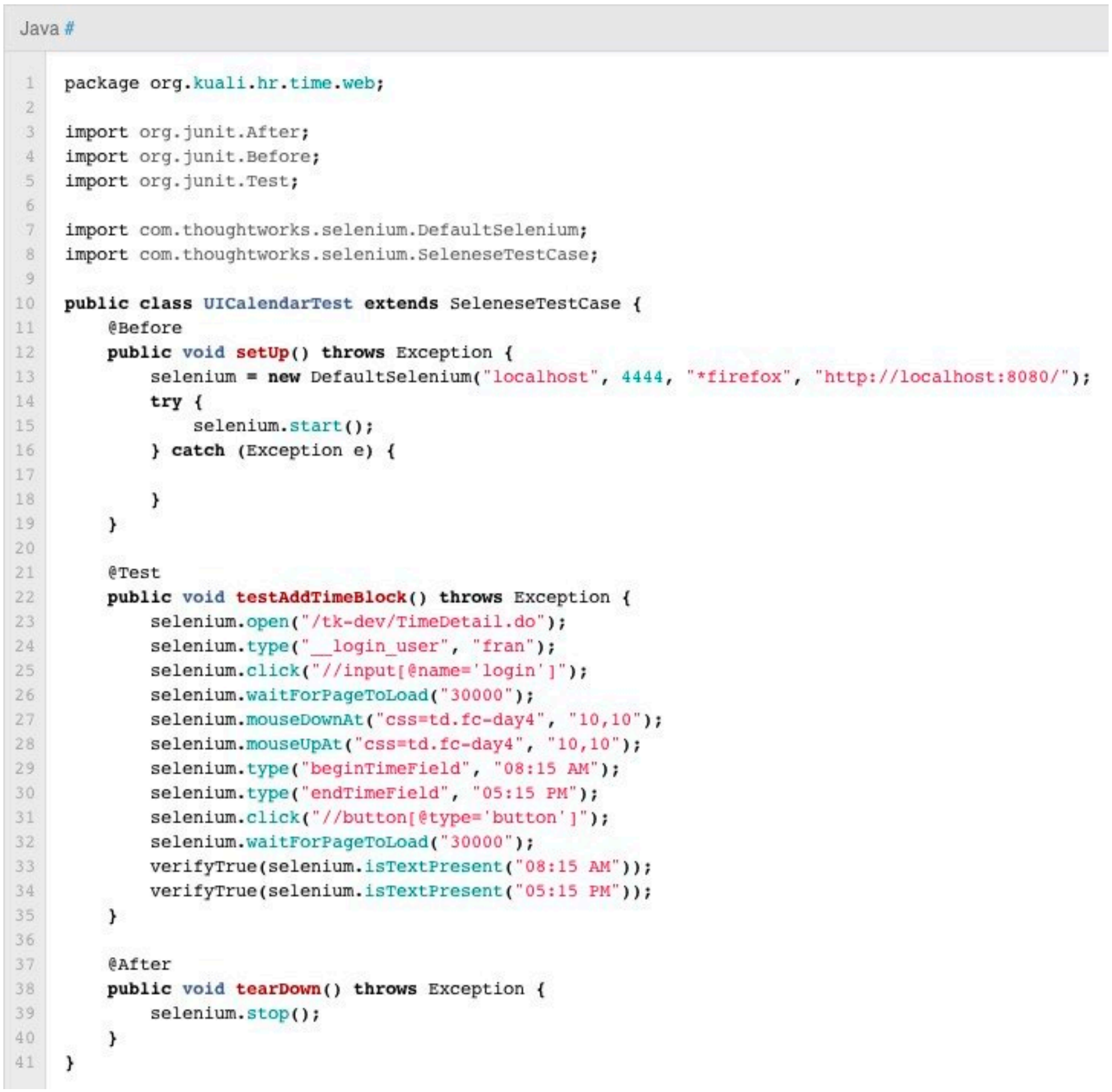
After installing the Selenium IDE, launch it in Firefox under Tools and you will see the screen like this:



Click the Record button to begin recording the actions. After you finish recording the actions, click the record button again to stop recording. See the Selenium commands section for more information about the available command and the options to locate the elements.

How to run a Selenium test as a JUnit test

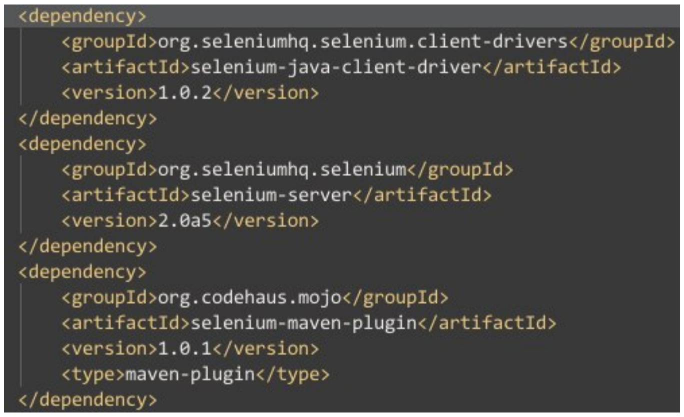
1. Create a test script using Selenium IDE.
2. Export the test script to the JUnit 4 format and it will create the Java code similar to below:



Java #

```java
package org.kuali.hr.time.web;

import org.junit.After;
import org.junit.Before;
import org.junit.Test;

import com.thoughtworks.selenium.DefaultSelenium;
import com.thoughtworks.selenium.SeleneseTestCase;

public class UICalendarTest extends SeleneseTestCase {
    @Before
    public void setUp() throws Exception {
        selenium = new DefaultSelenium("localhost", 4444, "*firefox", "http://localhost:8080/");
        try {
            selenium.start();
        } catch (Exception e) {

        }
    }

    @Test
    public void testAddTimeBlock() throws Exception {
        selenium.open("/tk-dev/TimeDetail.do");
        selenium.type("__login_user", "fran");
        selenium.click("//input[@name='login']");
        selenium.waitForPageToLoad("30000");
        selenium.mouseDownAt("css=td.fc-day4", "10,10");
        selenium.mouseUpAt("css=td.fc-day4", "10,10");
        selenium.type("beginTimeField", "08:15 AM");
        selenium.type("endTimeField", "05:15 PM");
        selenium.click("//button[@type='button']");
        selenium.waitForPageToLoad("30000");
        verifyTrue(selenium.isTextPresent("08:15 AM"));
        verifyTrue(selenium.isTextPresent("05:15 PM"));
    }

    @After
    public void tearDown() throws Exception {
        selenium.stop();
    }
}
```

3. Include the selenium-server-standalone-xxx.jar and selenium-java-client-driver.jar to the java classpath or in maven (pom.xml):



```xml
<dependency>
    <groupId>org.seleniumhq.selenium.client-drivers</groupId>
    <artifactId>selenium-java-client-driver</artifactId>
    <version>1.0.2</version>
</dependency>
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-server</artifactId>
    <version>2.0a5</version>
</dependency>
<dependency>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>selenium-maven-plugin</artifactId>
    <version>1.0.1</version>
    <type>maven-plugin</type>
</dependency>
```

4. Start the Selenium Server from the command line:
   ```
   java -jar selenium-server-standalone-2.0b3.jar –interactive
   ```
   The interactive mode allows you to test the Selenium command directly from the console.
5. Try the command below in the console to make sure you can successfully launch the browser from the selenium server:
   ```
   cmd=getNewBrowserSession&1=*firefox&2=http://www.google.com
   ```
   If you want to use a different browser than Firefox to run the test scripts. See the Troubleshooting section for more information.
6. Run the Java file as a JUnit test and it will launch the browser and execute the script. You can also set the break points in Java and step through each action.

How to setup Selenium on Jenkins

The goal of this task is to integrate the Selenium tests with Maven so they can be automatically run on Jenkins in the headless mode[1]. More specifically, we want to automate the process of :

1. Starting the selenium sever.
2. Running the selenium tests through JUnit / Maven in the headless mode.
3. Connecting to the selenium server and starting Firefox in the background.

The following steps assume Jenkins has been installed on Ubuntu.

1 Headless here means a computer system or device that has been configured to operate without a monitor (the missing "head"), keyboard, and mouse (from Wikipedia).

Setup Selenium Server

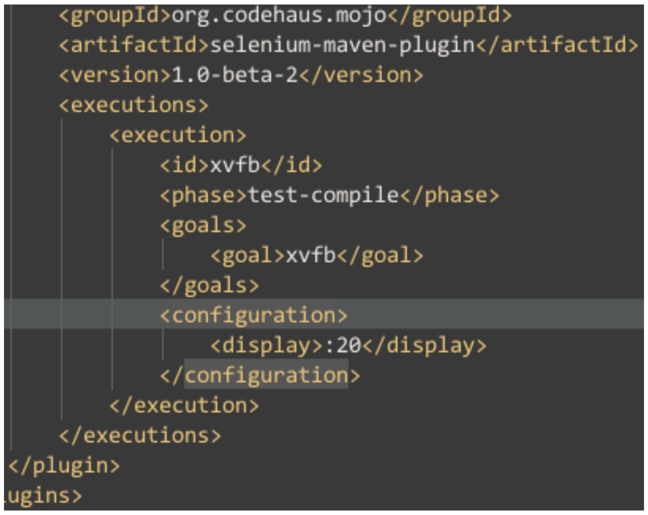
1. Download Selenium standalone server and create required folders.
```
sudo su
mkdir /usr/lib/selenium/
cd /usr/lib/selenium/
wget url-you-copied-above
mkdir -p /var/log/selenium/
chmod a+w /var/log/selenium/
```
2. Take the selenium.sh script under our code base : /src/test/docs/selenium.sh and save it as /etc/init.d/selenium
3. Make the file executable:
```
chmod 755 /etc/init.d/selenium
```
4. Start the selenium server and make sure it runs:
```
/etc/init.d/selenium start
ps -ef | grep selenium
```

5. Make the selenium server starts automatically when the server boots up
```
update-rc.d selenium defaults
```
6. Export the display port setting to .bashrc if you use bash as the default shell.
```
export DISPLAY=:20
```

Setup xvfb

Xvfb[2] allows maven to start Firefox in the background.

1. Install xvfb.
```
sudo apt-get install xvfb
```
2. Add xvfb to the pom.xml to start it before the selenium test runs. Note that the default display port for the selenium sever is 20, so the port setting of xvfb has to match it.



```
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>selenium-maven-plugin</artifactId>
    <version>1.0-beta-2</version>
    <executions>
        <execution>
            <id>xvfb</id>
            <phase>test-compile</phase>
            <goals>
                <goal>xvfb</goal>
            </goals>
            <configuration>
                <display>:20</display>
            </configuration>
        </execution>
    </executions>
</plugin>
ugins>
```

2 In the X Window System, Xvfb or X virtual framebuffer is an X11 server that performs all graphical operations in memory, not showing any screen output (from Wikipedia).

Troubleshooting

How to run Selenium server on different platforms

- Windows
  - Add the path of the browser folder to the PATH variable under environment variables. When you specify *firefox in the command, the system will then know where to find the executable file. If you want to run the tests using a different browser, you will have to do the same thing. This link shows you how to add environment variables:
    http://goo.gl/pjoqA
  - Here is a list of the browsers that Selenium supports:
    http://seleniumhq.org/about/platforms.html
  - Note that the command *chrome doesn't mean to use the Google Chrome as the browser. *googlechrome is the right one.
- Ubuntu
  - Since you can directly launch browsers from the command line, like:
    ```
    ~ ❯ firefox
    ```
    there isn't any extra setting necessary for the linux environments.
- Mac OS
  - Add Firefox to the system path and append the so that you can launch Firefox from the command line and you should be good to go.

    ```
    export PATH=/Applications/Firefox3.app/Contents/MacOS:$PATH
    ```

How to resolve the selenium server display port conflict with Jenkins

For some reason, Jenkins uses 20 as the default display port. To get around this, append the following setting to jenkins/bin/setenv.sh

```
-Djava.awt.headless=true
```

Resources

- Selenium Documentation
  http://seleniumhq.org/docs/
- Selenium commands
  - http://seleniumhq.org/docs/02_selenium_ide.html#commonly-used-selenium-commands
  - http://seleniumhq.org/docs/02_selenium_ide.html#locating-elements