

Automatización de pruebas web con frameworks open-source

Boni García

boni.garcia@upm.es

1. Introducción
2. Automatización de pruebas
3. JUnit
4. Selenium
5. JMeter
6. Práctica

1. Introducción

- Las pruebas en el software (***testing***) evalúan y mejoran la calidad del software identificando defectos (***bugs***)
- Las pruebas manuales es un proceso muy **costoso** (20% o más)
- La **automatización** de pruebas mediante **frameworks** ayuda a reducir dichos esfuerzos

2. Automatización de pruebas (I)

- La automatización en las pruebas software (AST, *Automated Software Testing*) es:

“La aplicación e implementación de tecnología software durante todo el ciclo de pruebas para mejorar la eficacia y eficiencia del mismo”

2. Automatización de pruebas (II)

- La automatización de pruebas es más efectiva cuando está implementado por un **framework**
- Un framework es una estructura conceptual que sirve de soporte de desarrollo para proyectos software.
- Framework de pruebas:
 - Open-souce: **JUnit**, **Selenium**, **JMeter**, JSystem, ...
 - Comerciales: HP QuickTest Professional, IBM Rational Functional Tester, ...





3. JUnit

- Framework de pruebas **unitarias** para **Java**
- JUnit ha sido portado a otros lenguajes (familia xUnit): .NET (NUnit), Python (PyUnit)
- Licencia CPL (*Common Public License*)

JUnit 3.x	JUnit 4.x
<pre>import junit.framework.Assert; import junit.framework.TestCase; public class JUnit3 extends TestCase { public void testMultiplicacion() { Assert.assertEquals(6, 3 * 2); } }</pre>	<pre>import org.junit.Assert; import org.junit.Test; public class JUnit4 { @Test public void testMultiplicacion() { Assert.assertEquals(6, 3 * 2); } }</pre>

4. Selenium (I)





- Framework de pruebas de **sistema** para **web**
- Basado en grabación y reproducción de scripts
- Selenium permite la ejecución de pruebas funcionales directamente en un navegador.
- Licencia Apache 2.0. Componentes:

Proyecto		Descripción
Selenium IDE		Plugin Firefox que permite grabación y reproducción
Selenium RC		Control programático de ejecución web (cliente/servidor)
Selenium WebDriver		Control programático de ejecución web (nativo)
Selenium Grid		Permite ejecutar Selenium RC en diferentes máquinas










“Selenium is a key mineral which protects the body from Mercury toxicity”

4. Selenium (II)

- Compatibilidad navegadores:

	IDE 	RC 	WebDriver 
Explorer 	✗	✓	✓
Firefox 	✓	✓	✓
Safari 	✗	✓	✓
Opera 	✗	✓	✓
Chrome 	✗	✓	✓

- Compatibilidad lenguajes:

	IDE 	RC 	WebDriver 
C# 	✓	✓	✓
Java 	✓	✓	✓
Perl 	✓	✓	✗
PHP 	✓	✓	✗
Python 	✓	✓	✓
Ruby 	✓	✓	✓

4.1. Comandos (I)

- Comandos: operaciones Selenium. 3 tipos:
 - Acciones (*actions*): Manipulación páginas: “hacer click”, “seleccionar opción”, ...
 - Accesores (*accessors*): Examinar estado de la aplicación: “acceder al título”, ...
 - Aserciones (*assertions*): Comprobaciones: “verificar que checkbox está seleccionado”, ...

- Sintaxis comandos:

Command	[Target]	[Value]
---------	----------	---------

- Secuencia de comandos = *test script*

- **open** (`url`)
 - Abre una URL determinada (absoluta, relativa)
`open /mypage`
`open http://localhost/`
- **click/clickAndWait** (`elementLocator`)
 - Hace click en un elemento HTML
`click link=buscar`
`clickAndWait link=buscar`

4.1. Comandos (III)

- **type** (inputLocator, value)
 - Escribe un valor en un elemento de entrada (input)
`type identifier=username PEPITO`
- **waitForPageToLoad** (timeout)
 - Espera un número de ms a que cargue una página
`waitForPageToLoad 500`
- **waitForElementPresent** (locator)
 - Espera a que un elemento esté presente (útil en AJAX)
`waitForElementPresent identifier=username`

4.1. Comandos (IV)

- **verifyTitle** (pattern)
 - Verifica el título de una determinada página
`verifyTitle Home`
- **verifyElementPresent** (locator)
 - Comprueba que un elemento está presente
`verifyElementPresent link=buscar`
- **verifyTextPresent** (pattern)
 - Comprueba que un determinado texto está presente
`verifyTextPresent exact:string`



Descripción de todos los comandos de la última versión de Selenium Core:
<http://release.seleniumhq.org/selenium-core/1.0.1/reference.html>

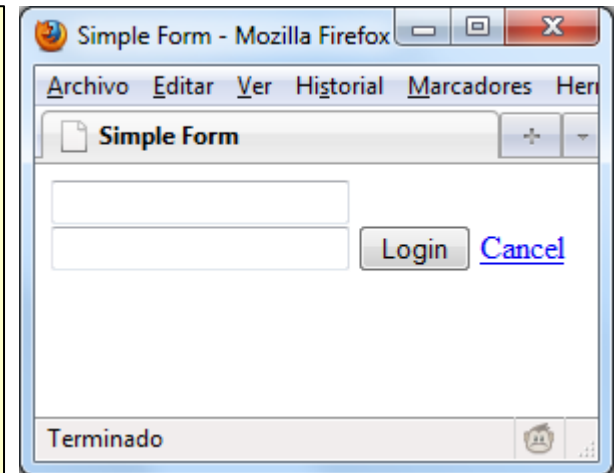
4.2. Localizadores (I)

- Sirven para buscar elementos HTML (8 tipos):

1.- Por identificador/nombre: Primer elemento con `id`. En caso de no existir, busca por `name`:

```
identifier=username
```

```
<html>
<head>
<title>Simple Form</title>
</head>
<body>
  <form id="loginForm">
    <input name="username" type="text" />
    <input name="password" type="password" />
    <input name="continue" type="submit" value="Login" />
    <a href="cancel.html">Cancel</a>
  </form>
</body>
</html>
```



4.2. Localizadores(II)

2.- Por identificador: Primer elemento con `id`.

```
id=loginForm
```

3.- Por nombre: Primer elemento con `name`.

```
name=username
```

4.- Por XPath (*XML Path Language*), que es un lenguaje que permite localizar elementos de un documento XML. Ejemplos:

```
xpath=/html/body/form[1]  
xpath=//form[@id='loginForm']  
xpath=//input[@name='username']  
xpath=//form[@id='loginForm']/input[1]
```

5.- Por el texto de un link:

```
link=Cancel
```

4.2. Localizadores (III)

6.- Por DOM (*Document Object Model*), que es una representación de documentos HTML/XML que puede ser accedida por JavaScript:

```
dom=document.getElementById('loginForm')
dom=document.forms['loginForm']
dom=document.forms[0]
```

7.- Por CSS (*Cascading Style Sheets*) que es el lenguaje de descripción de estilos de elementos HTML:

```
css=form#loginForm
css=input[name="username"]
```

8.- Por UI (*User Interface*), que es una notación específica de Selenium que mapea elementos semánticamente:

```
ui=loginPages::loginButton()
ui=settingsPages::toggle(label=Hide Email)
```

4.3. Patrones

- Sirven para buscar cadenas de texto (4 tipos):

1.- Global: comodines * y ? (opción por defecto):

```
glob:*Success!*
```

2.- Expresiones regulares (JavaScript):

```
regexp:/Chapter (\d+)\.\d*/
```

3.- Expresiones regulares no sensitivas:

```
regexpi:/Chapter (\d+)\.\d*/
```

4.- Palabras exactas:

```
exact:successful
```



Más información sobre expresiones regulares JavaScript:

<http://www.regular-expressions.info/javascript.html>

- Sirven para refinar la lista de candidatos determinados por un localizador. Hay 2 tipos

1.- Por valor:

```
value=pattern
```

2.- por índice:

```
index=indexNumber
```

4.5. Selenium IDE

google-selenium.html - Selenium IDE 1.9.0

Archivo (E) Editar Actions Options Ayuda

Base URL `http://www.google.es/`

Fast Slow

Test Case: **google-selenium**

Command	Target	Value
open	/	
type	id=gbqfq	selenium
click	id=gbqfb	
waitForElementPresent	link=Selenium - Web Browser Automation	
clickAndWait	link=Selenium - Web Browser Automation	

Command: open
Target: /
Value:

Runs: 1
Failures: 0

Log Reference UI-Element Rollup

open(url)
Arguments:

- url - the URL to open; may be relative or absolute

 Opens an URL in the test frame. This accepts both relative and absolute URLs. The "open" command waits for the page to load before proceeding, ie. the "AndWait" suffix is implicit. *Note:* The URL must be on the same domain as the runner HTML due to security restrictions in the browser (Same Origin Policy). If you need to open an URL on another domain, use the Selenium Server to start a new browser session on that domain.

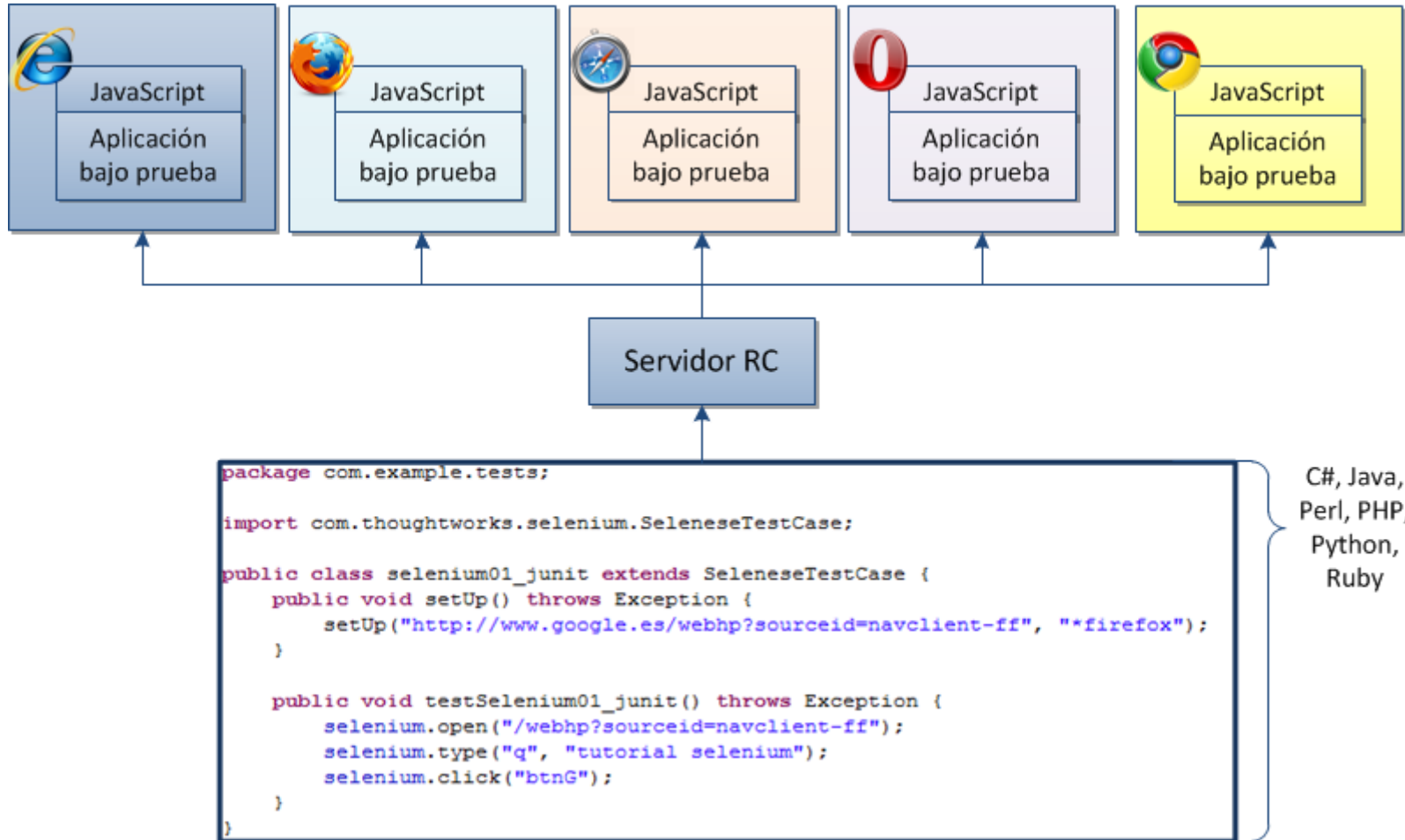
```
<table cellpadding="1" cellspacing="1" border="1">
<thead>
<tr><td rowspan="1" colspan="3">google-selenium</td></tr>
</thead><tbody>
<tr>
<td>open</td>
<td>/</td>
<td></td>
</tr>
<tr>
<td>type</td>
<td>id=gbqfq</td>
<td>selenium</td>
</tr>
<tr>
<td>click</td>
<td>id=gbqfb</td>
<td></td>
</tr>
<tr>
<td>waitForElementPresent</td>
<td>link=Selenium - Web Browser Automation</td>
<td></td>
</tr>
<tr>
<td>clickAndWait</td>
<td>link=Selenium - Web Browser Automation</td>
<td></td>
</tr>
</tbody></table>
```

Firefox

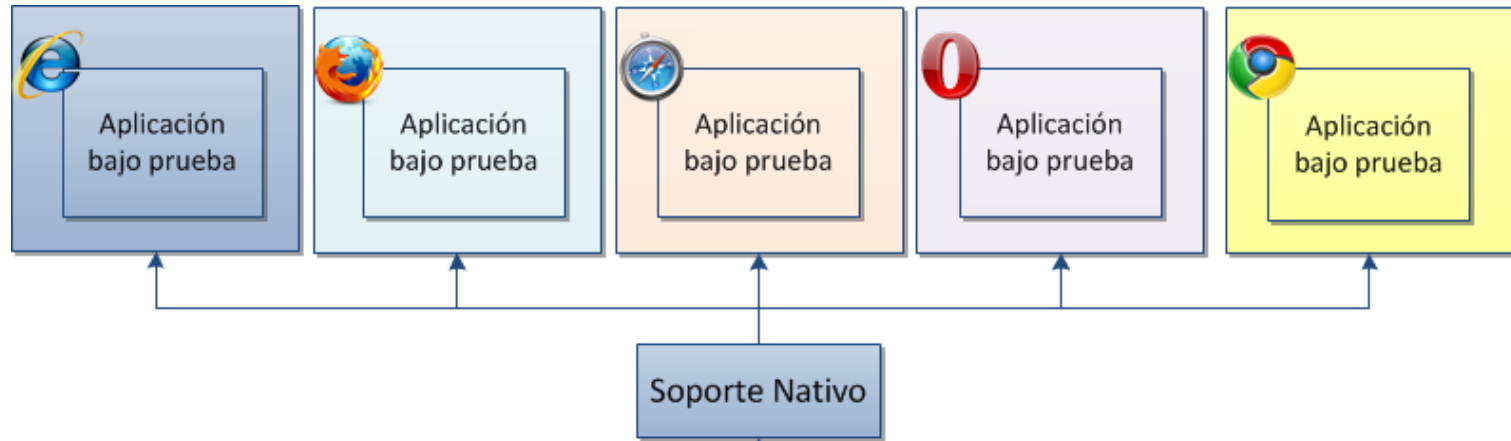
google-selenium

google-selenium		
open	/	
type	id=gbqfq	selenium
click	id=gbqfb	
waitForElementPresent	link=Selenium - Web Browser Automation	
clickAndWait	link=Selenium - Web Browser Automation	

4.6. Selenium RC



4.7. Selenium WebDriver



```

public class GoogleJUnit4WebDriverChrome {
    private WebDriver driver;
    private String baseUrl;
    private StringBuffer verificationErrors = new StringBuffer();

    @Before
    public void setUp() throws Exception {
        driver = new ChromeDriver();
        baseUrl = "https://www.google.es/";
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
    }

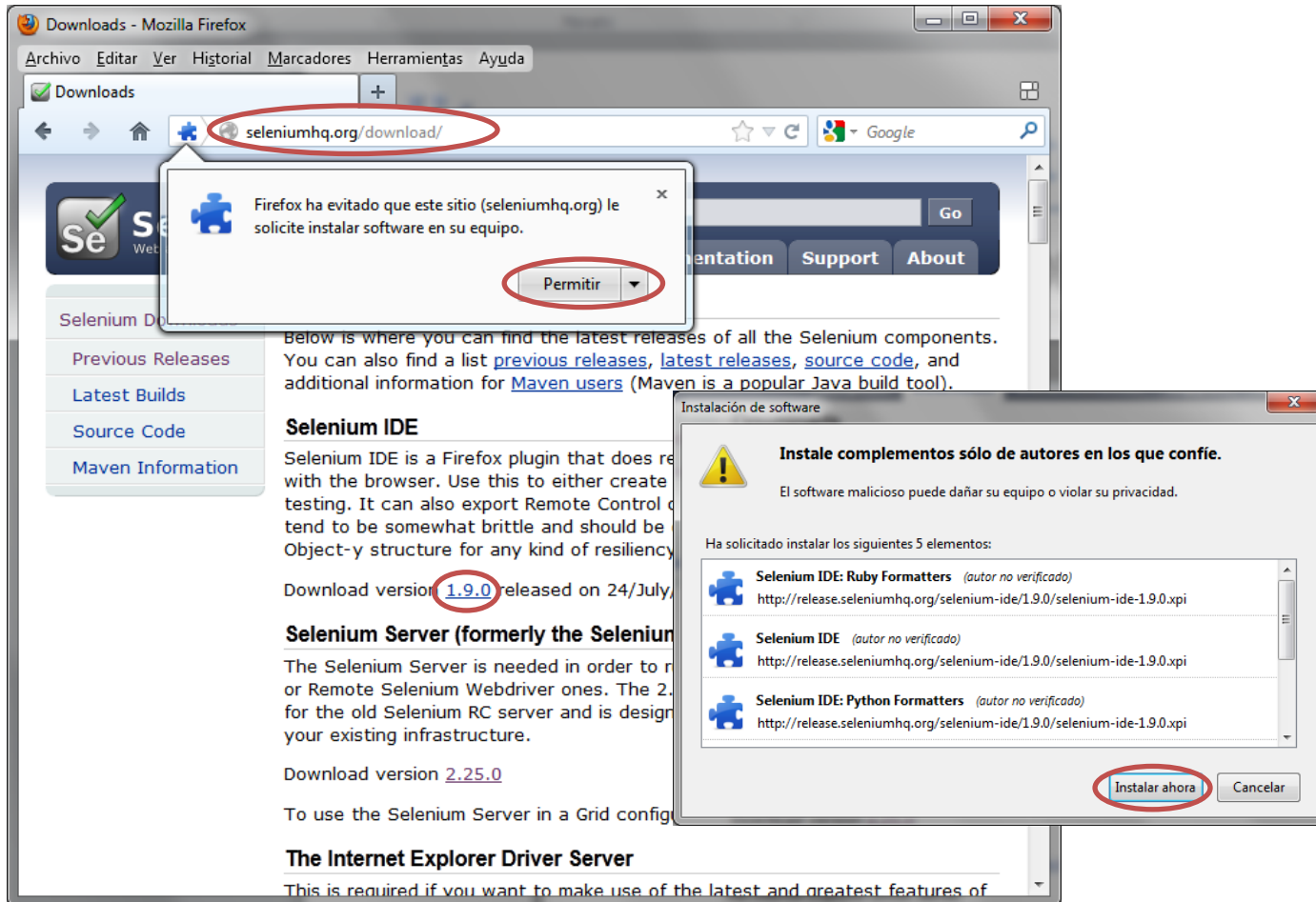
    @Test
    public void testGoogleJUnit4WebDriver() throws Exception {
        driver.get(baseUrl + "/");
        driver.findElement(By.id("gbqfq")).clear();
        driver.findElement(By.id("gbqfq")).sendKeys("testing");
        driver.findElement(By.id("gbqfb")).click();
    }
}

```

C#, Java,
Perl, PHP,
Python,
Ruby

4.8. Demostración (I)

1. Instalación de Selenium IDE:



2. Instalación herramienta adicional: FireBug

The image consists of two side-by-side screenshots from a Mozilla Firefox browser. The left screenshot shows the 'Firebug :: Complementos para Firefox' page on addons.mozilla.org. The URL 'https://addons.mozilla.org/es/firefox/addon/firebug/' is circled in red. Below the search bar, the 'Firebug 1.10.3' add-on is listed with a green 'Añadir a Firefox' button also circled in red. The right screenshot shows the Wikipedia main page. The Firebug developer tools are open at the bottom, displaying the HTML source code for the 'Today's featured article' section, which is circled in red. A blue arrow points from the 'Añadir a Firefox' button in the left screenshot to the Wikipedia page in the right screenshot.

3. Grabación y reproducción de un script:

The screenshot displays the Selenium IDE 1.9.0 interface. On the left, a browser window shows the Wikipedia page with the URL `en.wikipedia.org/wiki/Selenium_(software)` and the text "Selenium WebDriver" highlighted. The IDE interface on the right shows a table of commands:

Command	Target	Value
open	/wiki/Main_Page	
type	id=searchInput	selenium
clickAndWait	id=searchButton	
clickAndWait	link=Selenium (software)	
assertText	id=Selenium_WebDriver	Selenium WebDriver

Below the table, the details for the selected "open" command are shown:

```

Command: open
Target: /wiki/Main_Page
Value:
  
```

The IDE interface also shows a "Test Case" section with the name "wikipedia" and a "Runs" counter set to 1. The bottom of the IDE shows a "Log" section with the following text:

```

open(url)
Arguments:
  • url - the URL to open; may be relative or absolute
Opens a URL in the test frame. This accepts both relative and absolute URLs. The "open" command waits for the page to load before proceeding, i.e. the "AndWait" suffix is implicit. Note: The URL must be on the same domain as the runner HTML due to security restrictions in the browser (Same Origin Policy). If you need to open a URL on another domain, use the Selenium Server to start a new browser session on that domain.
  
```


5. Ejecución de caso prueba JUnit:

The screenshot shows the Wikipedia page for Selenium. The article title is "Selenium" and it includes a "WebDriver" label at the bottom right. The article text describes Selenium as a chemical element with symbol Se and atomic number 34. It is a nonmetal with properties that are intermediate between those of its periodic table column-adjacent chalcogen elements sulfur and tellurium. It rarely occurs in its elemental state in nature, or as pure ore compounds. Selenium (Greek *σεληνη* *selene* meaning "Moon") was discovered in 1817 by Jöns Jakob Berzelius, who noted the similarity of the new element to the previously-known tellurium (named for the Earth).

Selenium is found impurely in metal sulfide ores, where it partially replaces the sulfur. Commercially, selenium is produced as a byproduct in the refining of these ores, most often during copper production. Minerals that are pure selenide or selenate compounds are known, but are rare. The chief commercial uses for selenium today are in glassmaking and in pigments. Selenium is a semiconductor and is used in photocells. Uses in electronics, once important, have been mostly supplanted by silicon semiconductor devices. Selenium continues to be used in a few types of DC power surge protectors and one type of fluorescent quantum dot.

Selenium salts are toxic in large amounts, but trace amounts are necessary for cellular function in many organisms, including all animals. Selenium is a component of the antioxidant enzymes glutathione peroxidase and thioredoxin reductase (which indirectly reduce certain oxidized molecules in animals and some plants). It is also found in three deiodinase enzymes, which convert thyroid hormones to another. Selenium

```

package com.example.tests;

import static org.junit.Assert.assertEquals;

public class WikipediaJUnit4WebDriver {
    private WebDriver driver;
    private String baseUrl;
    private StringBuffer verificationErrors = new StringBuffer();

    @Before
    public void setUp() throws Exception {
        driver = new FirefoxDriver();
        baseUrl = "http://en.wikipedia.org/";
        driver.manage().timeouts().implicitlyWait(30, TimeUnit.SECONDS);
    }

    @Test
    public void testWikipediaJUnit4WebDriver() throws Exception {
        driver.get(baseUrl + "/wiki/Main_Page");
        driver.findElement(By.id("searchInput")).clear();
        driver.findElement(By.id("searchInput")).sendKeys("selenium");
        driver.findElement(By.id("searchButton")).click();
        driver.findElement(By.linkText("Selenium (software)")).click();
        assertEquals("Selenium WebDriver",
            driver.findElement(By.id("Selenium_WebDriver")).getText());
    }
}
    
```

The screenshot shows the Eclipse IDE with the Java test class `WikipediaJUnit4WebDriver.java`. The code uses Selenium WebDriver and JUnit 4 for testing. The console shows the test execution results, indicating that the test passed successfully.

5. JMeter (I)

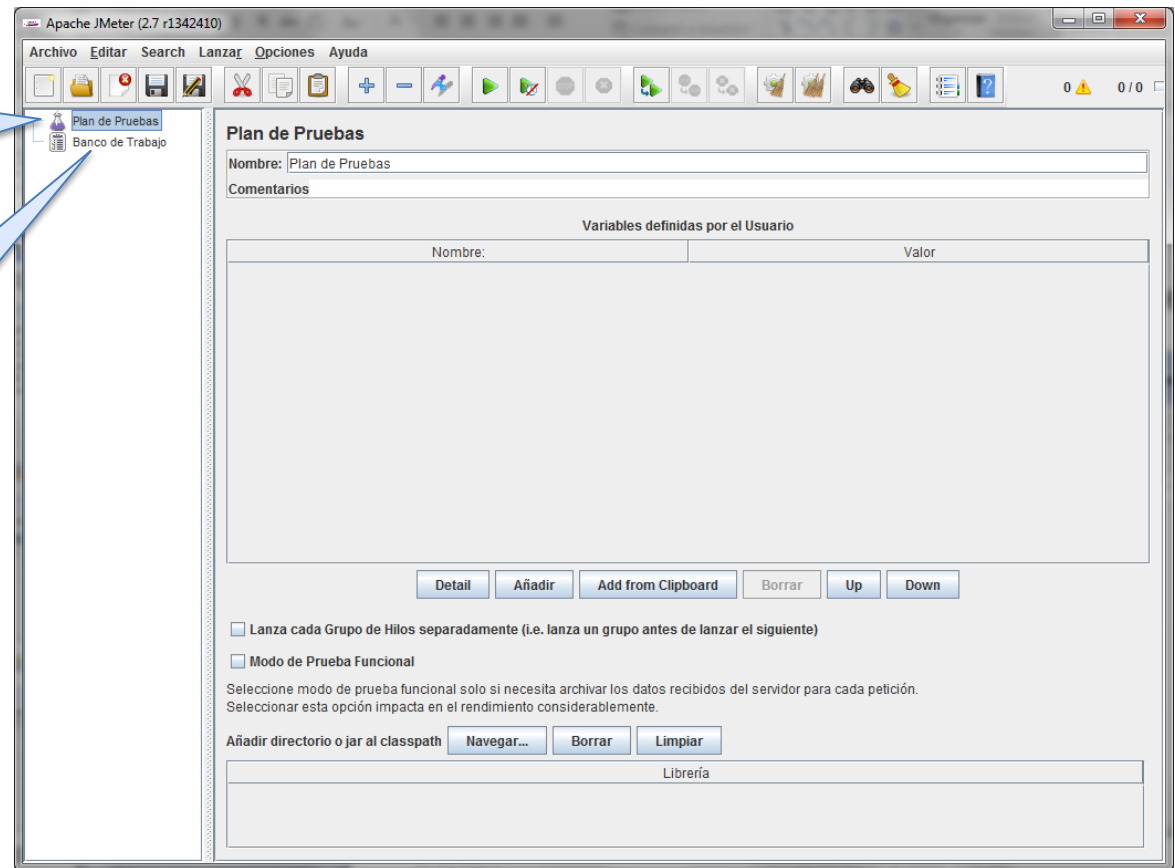
- Framework de pruebas de **rendimiento** para aplicaciones cliente/servidor: **web** (HTTP/HTTPS), FTP, JDBC, LDAP, Java, JUnit, ...
- Creado por la *Apache Software Foundation* (ASF).
- Licencia Apache 2.0
- Aplicación de escritorio desarrollada en Java (multiplataforma)

5. JMeter (II)

- Interfaz JMeter (ejecutando `jmeter.bat`):

Plan de pruebas
(*test plan*): Define
los pasos a ejecutar

Banco de pruebas
(*workbench*):
Elementos de
configuración



5.1. Plan de pruebas

- **Grupo de hilos** (*thread group*). Número de usuarios concurrentes que simulará JMeter.
- **Muestreadores** (*samplers*). Le dicen a JMeter como enviar peticiones (*request*) contra un servidor, por ejemplo peticiones HTTP, FTP, LDAP, SOAP
- **Receptores** (*listeners*). Acceso a la información recogida por JMeter. Por ejemplo, “Gráfico de resultados”.
- **Aserciones** (*assertions*). Verificación de las respuestas obtenidas del sistema bajo prueba (*System Under Test, SUT*).
- Controladores (*controllers*). Operadores de control, tales como condicionales (if) o bucles (ForEach, Loop, ...)
- Elementos de configuración (*configuration elements*). Configuración de los muestreadores.
- Temporizadores (*timers*). Configuración de los retrasos entre peticiones.
- Pre-Procesadores (*pre-processors*). Ejecutan alguna acción antes que las peticiones de los muestreadores.
- Post-Procesadores (*post-processors*). Ejecutan alguna acción después que las peticiones de los muestreadores.



Descripción detallada de todos los componentes JMeter:

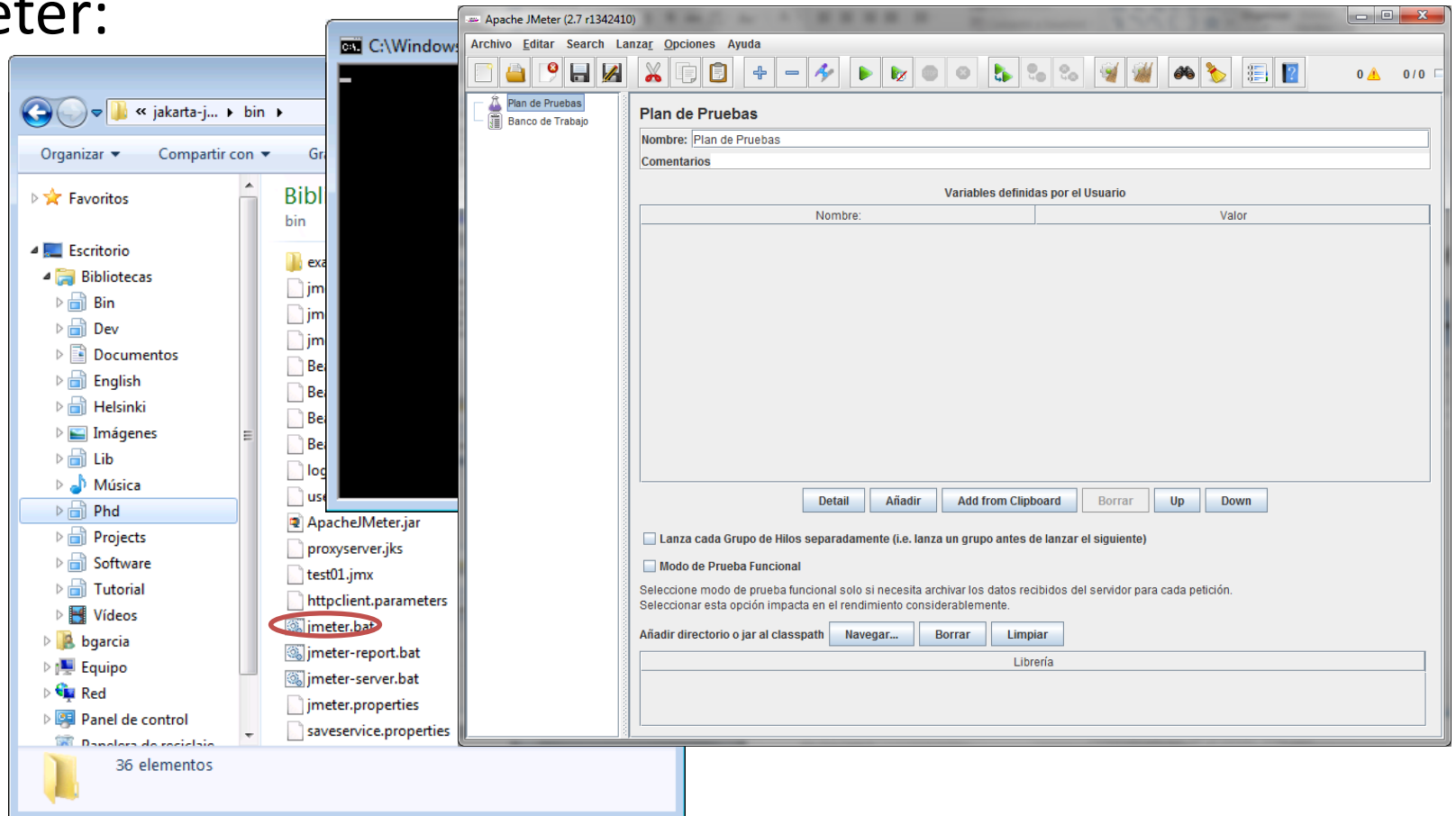
http://jakarta.apache.org/jmeter/usermanual/component_reference.html

5.2. Banco de pruebas

- Elementos de configuración (“*nodeprueba*”):
 - ***HTTP Proxy Server***: Proxy que permite a JMeter grabar interacciones con una aplicación web.
 - ***HTTP Mirror Server***: Servidor web de pruebas. Responde la petición tal y como le llega.
 - ***Property Display***: Configuración general de JMeter (`jmeter.properties`).

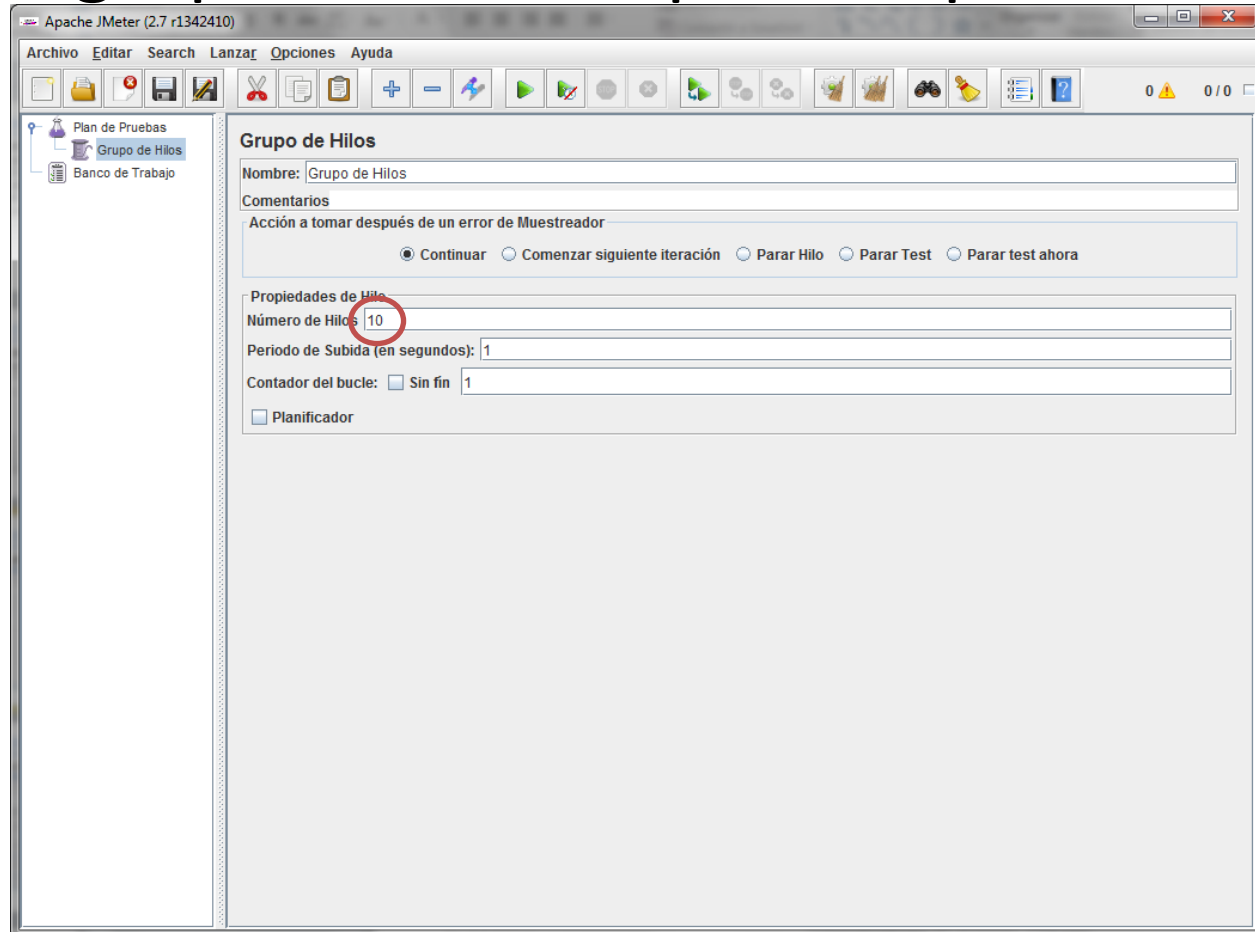
5.3. Demostración (I)

1. Descargar (<http://jmeter.apache.org/>) y ejecutar JMeter:



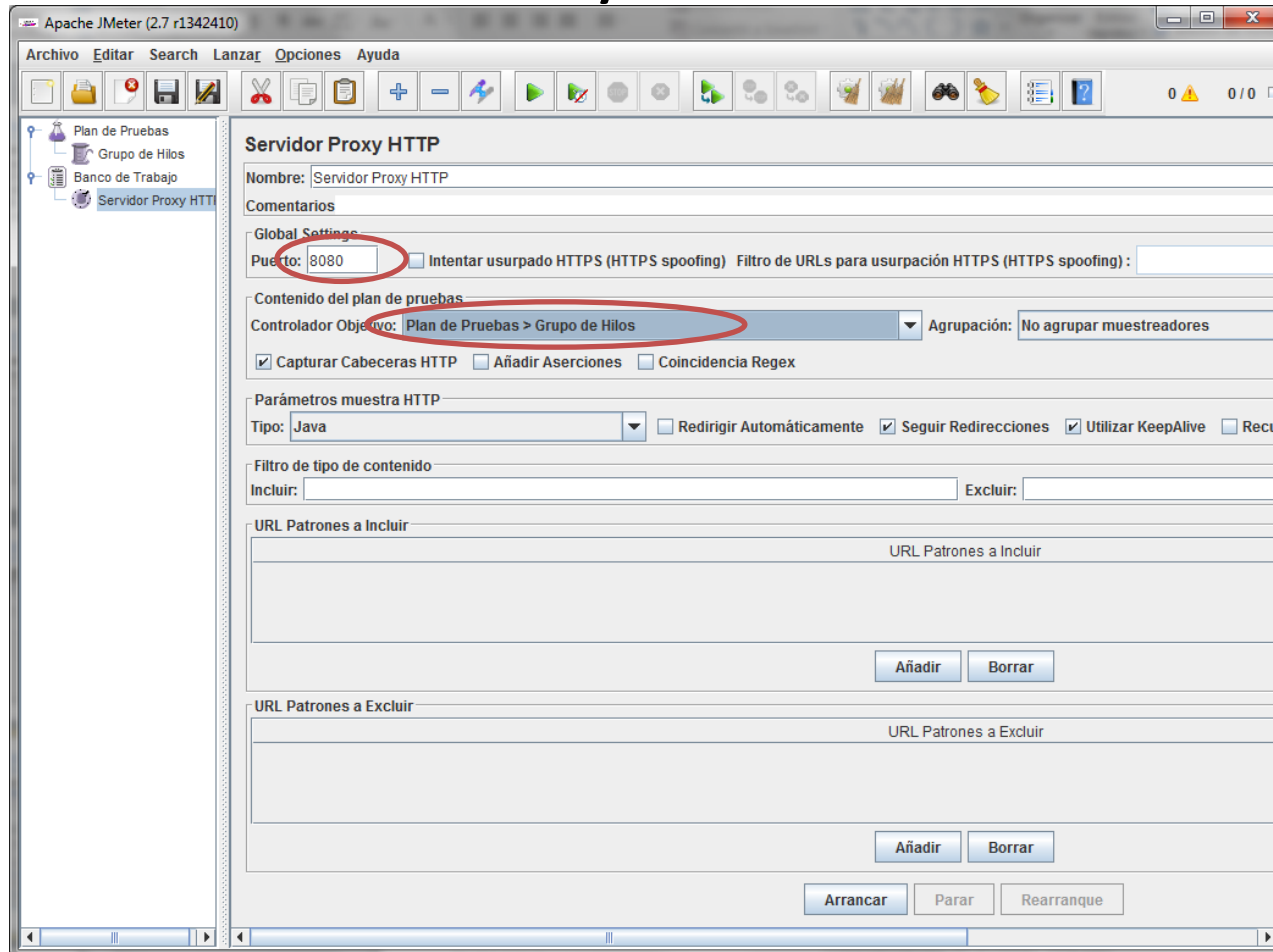
5.3. Demostración (II)

2. Añadir grupo de hilos al plan de pruebas:



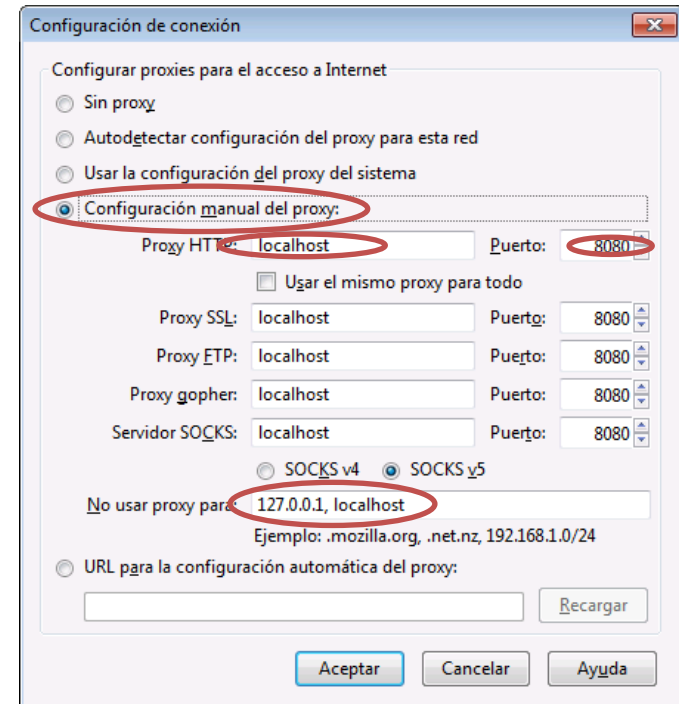
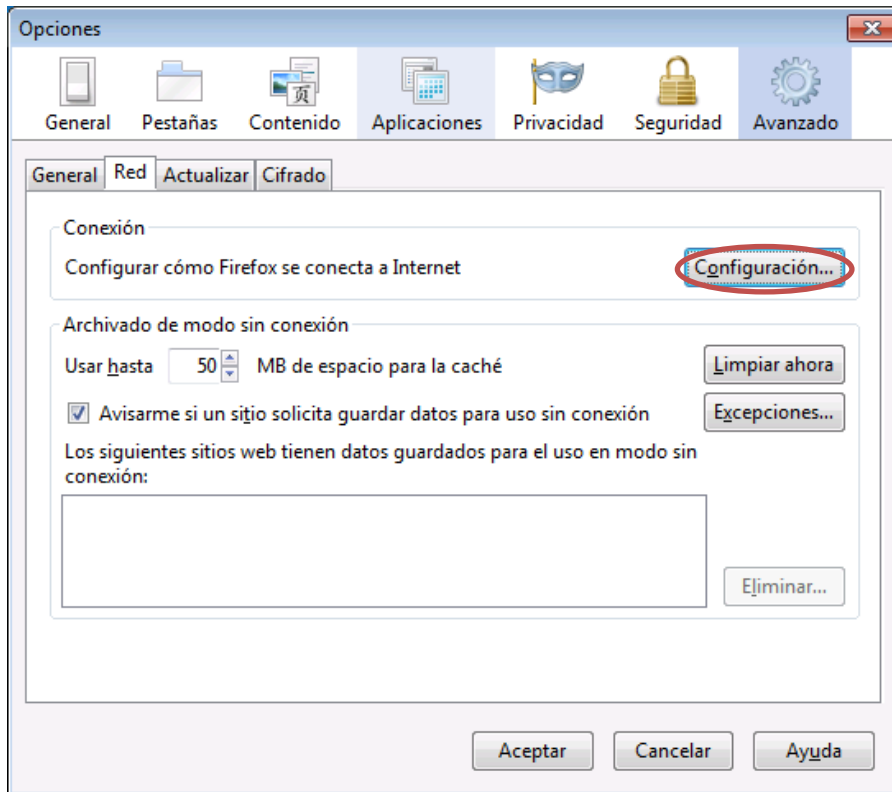
5.3. Demostración (III)

3. Añadir Servidor Proxy HTTP al banco de trabajo:

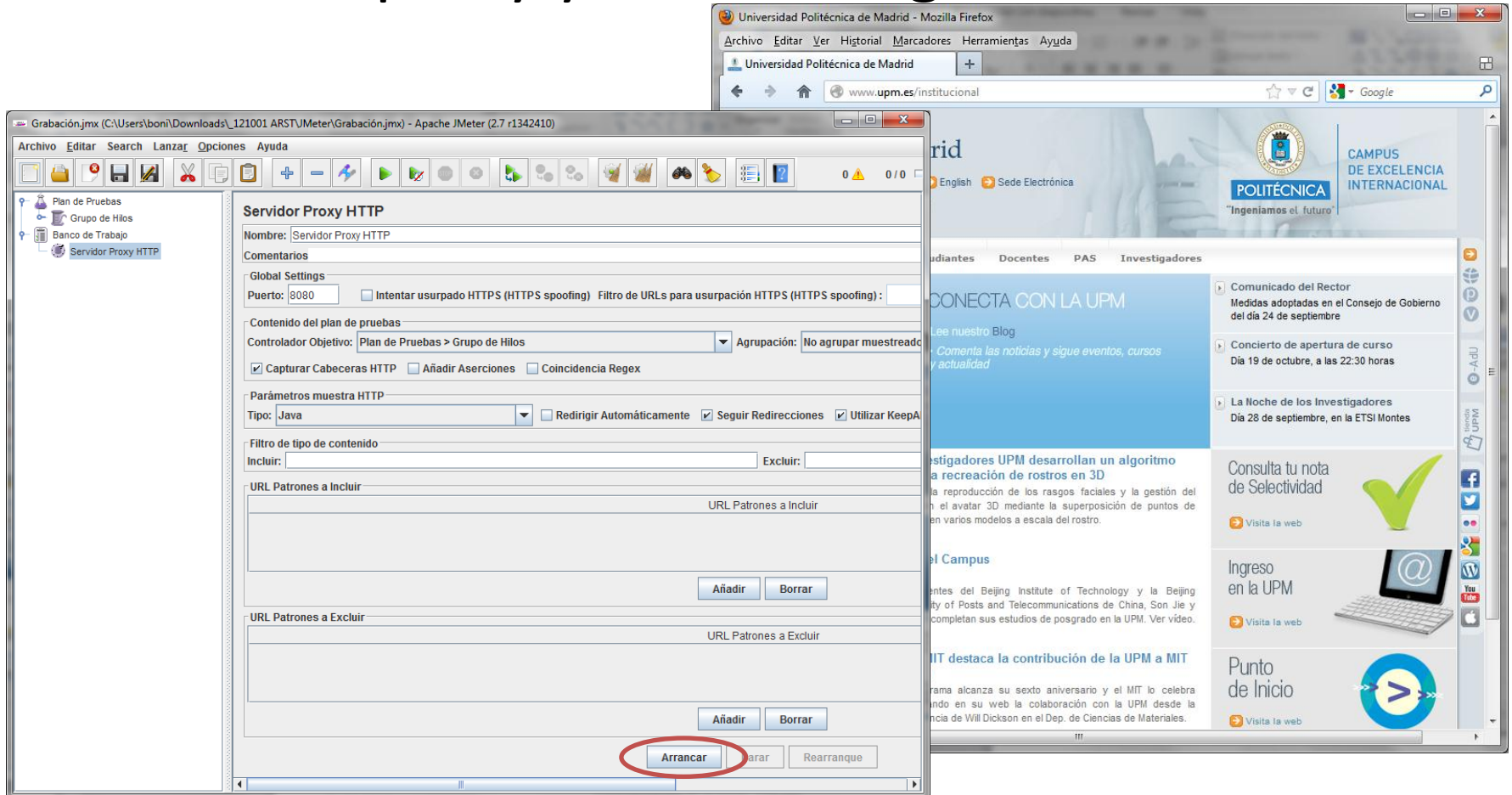


5.3. Demostración (IV)

4. Configurar proxy en el navegador web:



5. Arrancar proxy y comenzar grabación:



The image shows two overlapping windows. The background window is Mozilla Firefox displaying the website 'www.upm.es/institucional'. The foreground window is Apache JMeter (2.7 r1342410) with the 'Servidor Proxy HTTP' configuration panel open. The 'Arrancar' button at the bottom of the JMeter window is circled in red.

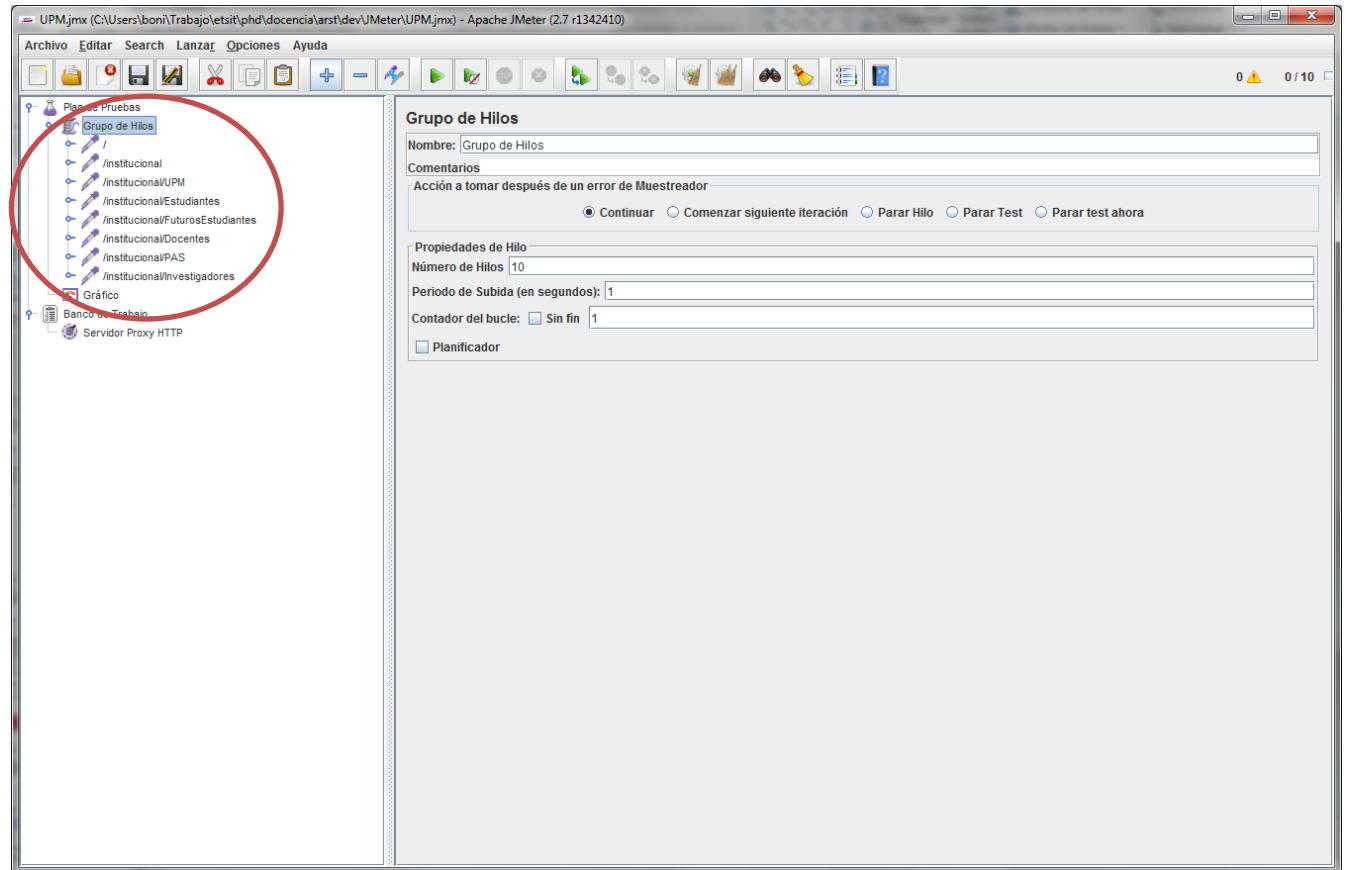
Apache JMeter Configuration:

- Nombre: Servidor Proxy HTTP
- Comentarios: (empty)
- Global Settings:
 - Puerto: 8080
 - Intentar usurpado HTTPS (HTTPS spoofing)
 - Filtro de URLs para usurpación HTTPS (HTTPS spoofing): (empty)
- Contenido del plan de pruebas:
 - Controlador Objetivo: Plan de Pruebas > Grupo de Hilos
 - Agrupación: No agrupar muestreador
 - Capturar Cabeceras HTTP
 - Añadir Aserciones
 - Coincidencia Regex
- Parámetros muestra HTTP:
 - Tipo: Java
 - Redirigir Automáticamente
 - Seguir Redirecciones
 - Utilizar KeepAlive
- Filtro de tipo de contenido:
 - Incluir: (empty)
 - Excluir: (empty)
- URL Patrones a Incluir: (empty)
- URL Patrones a Excluir: (empty)

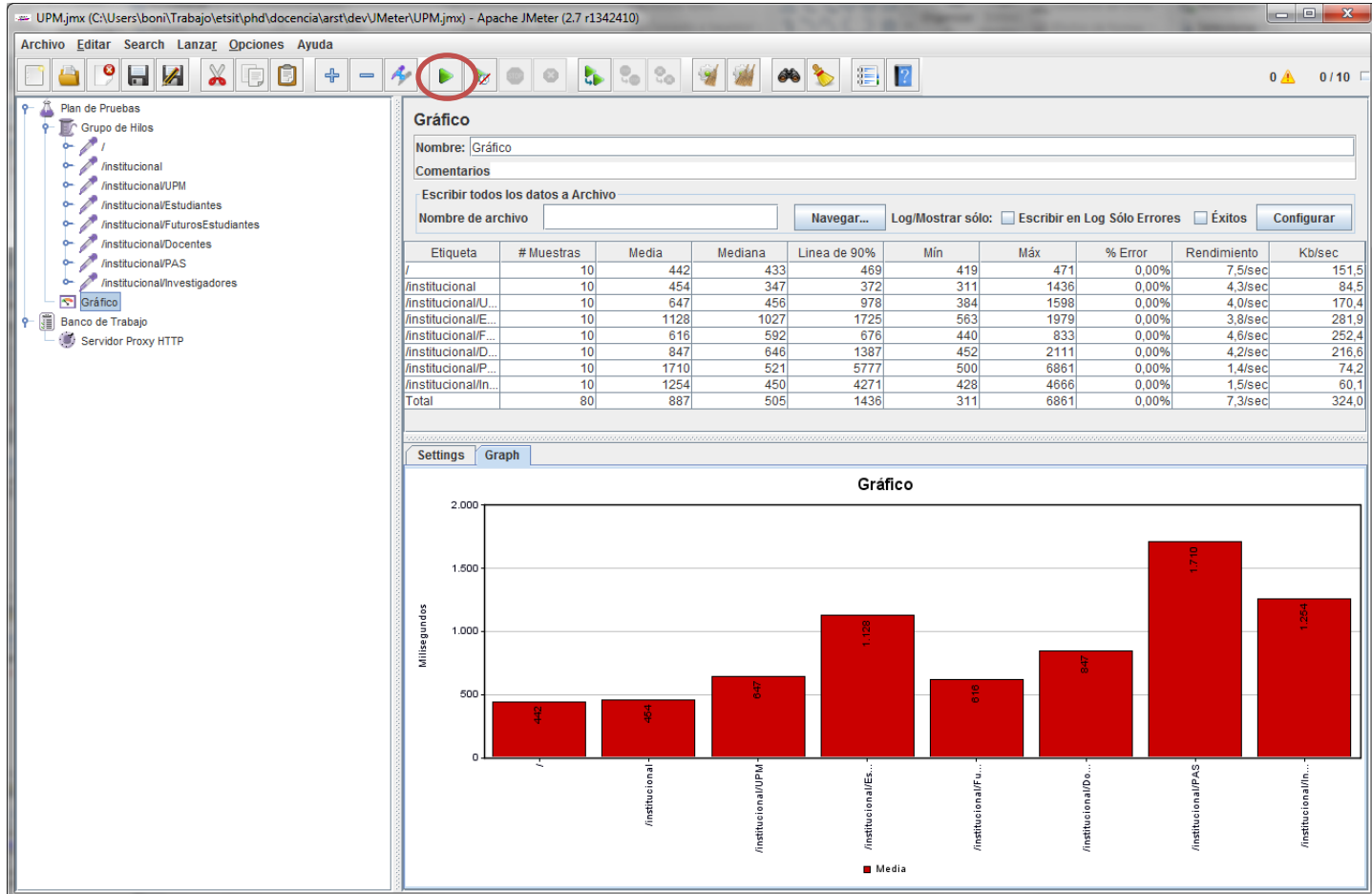
Buttons: Añadir, Borrar, Arrancar (circled), Parar, Rearranque.

5.3. Demostración (VI)

6. Parar grabación y añadir visor gráfico de resultados:



7. Reproducir plan grabado:



6. Práctica (I)

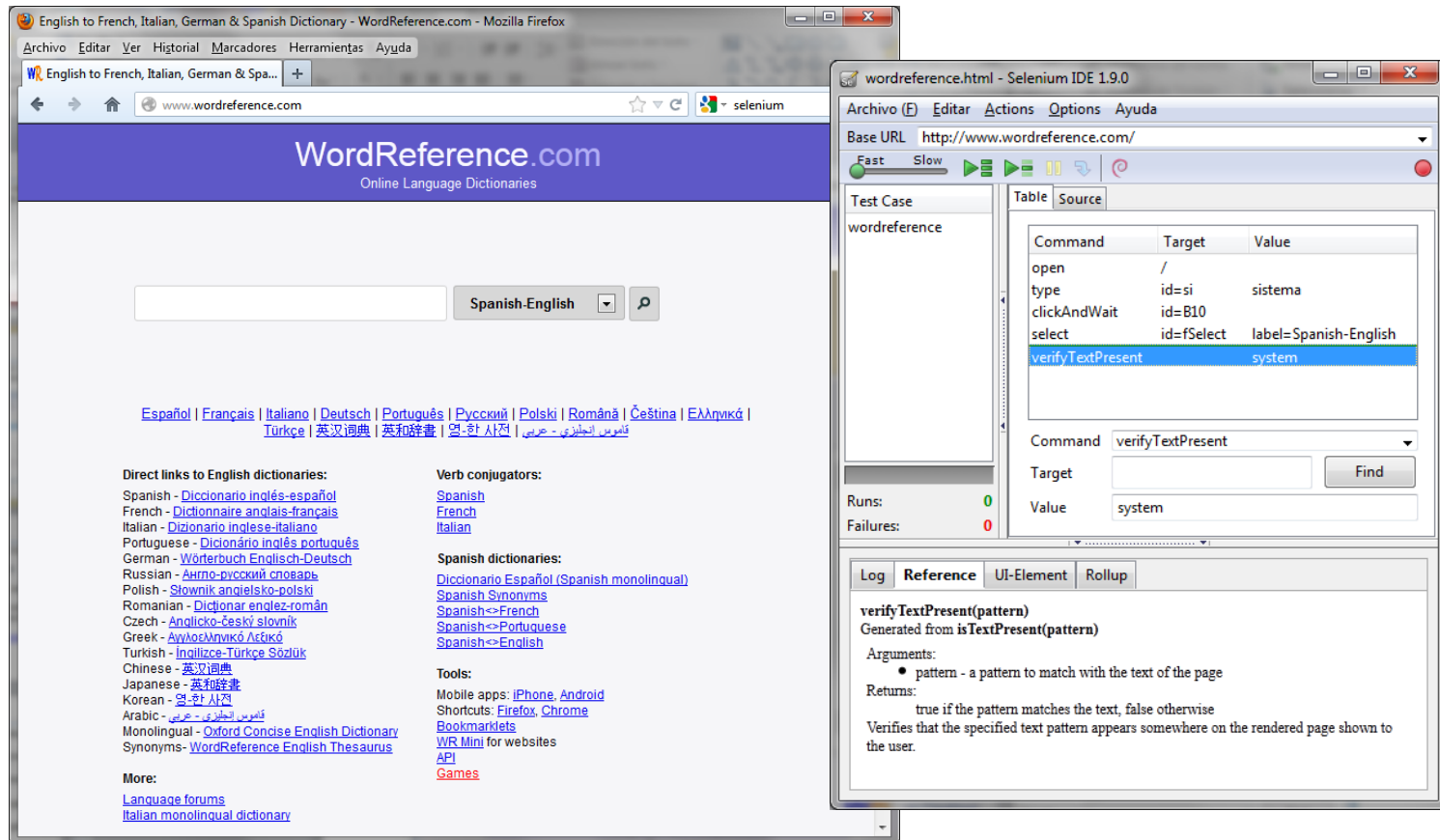
- Realizar un prueba contra la aplicación web www.wordreference.com que compruebe las siguientes traducciones español-inglés:
 - sistema → *system*
 - diccionario → *dictionary*
 - aplicación → *application*
 - librería → *library**
- Visualizar gráficos de rendimiento para una traducción español-inglés simulando 100 usuarios concurrentes



* Este último caso de pruebas debe fallar, ya que la traducción de librería debería ser *bookstore* o *bookshop*, y no *library*.

6. Práctica (II)

1. Instalar Selenium IDE y FireBug en Firefox (ver transparencia 21 y 22)
2. Grabar un script que compruebe una palabra contra la web bajo pruebas:



6. Práctica (III)

3. Exportar test script de Selenium IDE a JUnit v4 para WebDriver (ver transparencia 24)
4. Descargar Selenium Client Driver para Java de <http://seleniumhq.org/download/>
5. Importar en Eclipse el caso de prueba JUnit y las dependencias necesarias
6. Extender caso de prueba grabado según enunciado
7. Ejecutar casos de prueba JUnit (ver transparencia 25)
9. Descargar JMeter de http://jakarta.apache.org/site/downloads/downloads_jmeter.cgi
10. Ejecutar JMeter (ver transparencia 30)

6. Práctica (V)

11. Añadir grupo de 100 hilos en el plan de pruebas JMeter (transparencia 31)
11. Añadir un proxy en el banco de trabajo JMeter (transparencia 32)
12. Configurar navegador (Firefox) para usar el proxy (transparencia 33)
13. Arrancar grabación y realizar la interacción con la web según enunciado (transparencia 34)
14. Parar grabación y añadir *listener* de gráfico (transparencia 35)
15. Ejecutar grabación y generación de gráficas de resultado (transparencia 36)

- *Software Engineering, 9th Edition*. Ian Sommerville. Addison-Wesley. 2010.
- *Implementing Automated Software Testing*. Elfriede Dustin, Thom Garrett, Bernie Gauf. Addison-Wesley Professional. 2009.
- *xUnit Test Patterns. Refactoring Test Code*. Gerard Meszaros. Addison-Wesley. 2008.
- *Apache JMeter*. Emily H. Halili. Packt Publishing. 2008
- JUnit: <http://www.junit.org/>
- Selenium: <http://seleniumhq.org/>
- JMeter: <http://jakarta.apache.org/jmeter/>
- Eclipse: <http://www.eclipse.org/>