

¿Qué ofrece Autentia Real Business Solutions S.L?

Somos su empresa de **Soporte a Desarrollo Informático**.
 Ese apoyo que siempre quiso tener...

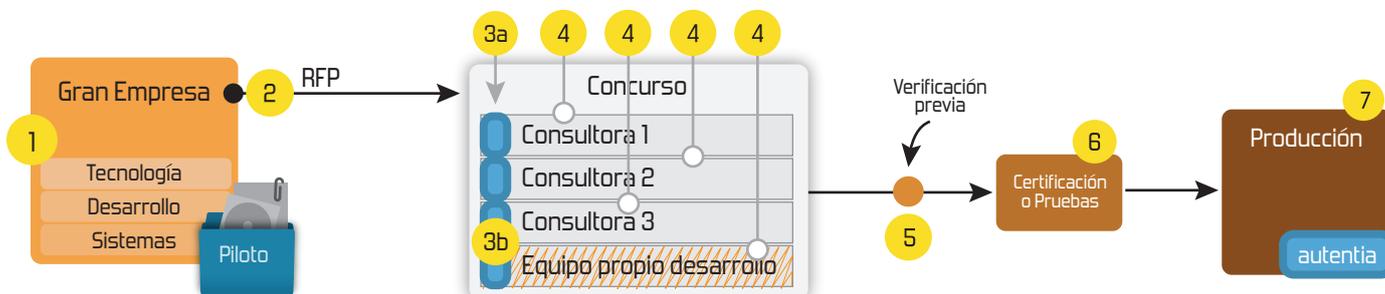
1. Desarrollo de componentes y proyectos a medida



2. Auditoría de código y recomendaciones de mejora

3. Arranque de proyectos basados en nuevas tecnologías

1. Definición de frameworks corporativos.
2. Transferencia de conocimiento de nuevas arquitecturas.
3. Soporte al arranque de proyectos.
4. Auditoría preventiva periódica de calidad.
5. Revisión previa a la certificación de proyectos.
6. Extensión de capacidad de equipos de calidad.
7. Identificación de problemas en producción.



4. Cursos de formación (impartidos por desarrolladores en activo)

Spring MVC, JSF-PrimeFaces /RichFaces,
 HTML5, CSS3, JavaScript-jQuery

Gestor portales (Liferay)
 Gestor de contenidos (Alfresco)
 Aplicaciones híbridas

Tareas programadas (Quartz)
 Gestor documental (Alfresco)
 Inversión de control (Spring)

Control de autenticación y
 acceso (Spring Security)
 UDDI
 Web Services
 Rest Services
 Social SSO
 SSO (Cas)

JPA-Hibernate, MyBatis
 Motor de búsqueda empresarial (Solr)
 ETL (Talend)

Dirección de Proyectos Informáticos.
 Metodologías ágiles
 Patrones de diseño
 TDD

BPM (jBPM o Bonita)
 Generación de informes (JasperReport)
 ESB (Open ESB)

NUEVO ¿Quieres saber cuánto ganas en relación al mercado? pincha aquí...

[Ver cursos que ofrece Autentia](#)

[Descargar comics en PDF y alta resolución](#)



[¡NUEVO!] 2008-12-01



2008-11-17



2008-09-01



2008-07-31

Estamos escribiendo un libro sobre la profesión informática y estas viñetas formarán parte de él. Puedes opinar en la sección [comic](#).

Tutorial desarrollado por



Víctor Javier Madrid

Consultor tecnológico de desarrollo de proyectos informáticos.

Ingeniero en Informática por la Universidad de Alcalá de Henares.

Puedes encontrarme en [Autentia](#)

Somos expertos en Java/J2EE

Catálogo de servicios de Autentia

[Descargar \(6,2 MB\)](#)

[Descargar en versión comic \(17 MB\)](#)

[AdictosAlTrabajo.com](#) es el Web de difusión de conocimiento de [Autentia](#).



[Catálogo de cursos](#)

Descargar este documento en formato PDF: [selenium_rc.pdf](#)

Fecha de creación del tutorial: 2009-02-24

Selenium Remote Control.

0. Índice de contenidos.

- 1. Introducción.
- 2. Entorno.
- 3. Requerimientos.
- 4. Instalación.
- 5. Configuración.
- 6. Utilización.
- 7. Ejemplo.
- 8. Conclusiones.

1. Introducción

Selenium Remote Control (RC) es una herramienta del paquete [SeleniumHQ](#) creada con el fin de ampliar y complementar las funcionalidades de las herramientas ya explicadas en anteriores tutoriales:

1. [Selenium IDE \(Tutorial\)](#)
2. [Selenium Core \(Tutorial\)](#)

Nota: Resulta interesante visualizar los tutoriales anteriores antes de meterse de lleno con esta herramienta.

Selenium Remote Control es una herramienta que permite automatizar las pruebas sobre aplicaciones web, para ello:

- Permite escribir los test en cualquier lenguaje que permita realizar peticiones vía HTTP (Por ejemplo : Java,C#,Perl,...).
- Estos test se aplicarán sobre un sitio web HTTP.
- Para ello utilizará un navegador que permita usar Javascript (Lo tiene que tener habilitado).

Ayuda: Para la generación de un test en el lenguaje elegido se aconseja utilizar la exportación disponible en Selenium IDE. De esta forma se proporciona la mayor parte del código necesaria para la ejecución de la prueba.

Lo dicho, podremos generar nuestros scripts en nuestro lenguaje favorito, pudiendo utilizar todo nuestro conocimiento para dotar de mucha mayor potencia y flexibilidad a los scripts. La única pega, es que tendremos que utilizar un navegador que permita el uso de Javascript, cualquier navegador que lo tenga habilitado para ello nos vale, pero aquellos que no están considerados en los requerimientos pueden presentar algún tipo de limitación debido a sus opciones de seguridad.

OJO: Mucho cuidado con las opciones de seguridad de los navegadores, a veces juegan malas pasadas. ;-)

La función principal de esta herramienta es la de ejecutar los test en **diferentes navegadores** y en **diferentes plataformas**. Esto es lo realmente interesante, porque al final lo ideal sería que pudiéramos probar nuestra aplicación en todos los entornos posibles, pero bastará con que lo probemos en aquellos entornos que utilizará el cliente (algo totalmente lógico).

Por ejemplo: En algunos sitios se impone como único entorno ("típico") de uso el formado por Windows e Internet Explorer. Por suerte esto es cada vez menor y cada vez es más fácil ver gente que utiliza otros navegadores u otros sistemas operativos. Esto lo que genera es mayor diversidad de usuarios que hay que tener en cuenta a la hora de implementar una aplicación.

Ejemplos de entornos que se pueden utilizar:

Catálogo de servicios Autentia (PDF 6,2MB)



[En formato comic...](#)



- Web
- [www.adictosaltrabajo.com](#)

Últimos tutoriales

2009-02-24
[Selenium Remote Control](#)

2009-02-22
[Integración de Groovy, JRuby y BeanShell con Spring 2](#)

2009-02-18
[Instalación de Pentaho BI Suite Community Edition 1.7.0](#)

2009-02-18
[Replicar Web PHP en máquina local](#)

2009-02-16
[Selenium Core : El motor de Selenium.](#)

2009-02-16
[Integración de JasperReports con PHP](#)

2009-02-09
[EJB 3.0 y pruebas unitarias con Maven, JUnit 4 y Embedded JBoss sobre Java 6](#)

2009-02-09
[Web Service Security](#)

2009-02-09
[Manual Avanzado de Firebug](#)

2009-01-29
[Ejemplo con Mockito](#)

Últimas ofertas de empleo

2009-02-21
[Otras - Estética/Peluquería - MADRID.](#)

2009-02-13
[T. Información - Otros no catalogados - MADRID.](#)

2009-02-13
[T. Información - Otros no catalogados - MADRID.](#)

2009-02-13

Sistema Operativo	Navegador
Windows	Internet Explorer
Windows	Firefox
Linux	Firefox
OS X	Safari
OS X	Firefox

T. Información - Otros no catalogados - MADRID.

2009-02-13
T. Información - Diseñador Gráfico - MADRID.

Anuncios Google

Recodad que esta herramienta se plantea como solución para poder resolver los problemas de seguridad que aparecen en el navegador (bloques de test , etc.) cuando queremos ejecutar test en servidores en los que no se encuentra instalado Selenium Core. (Para ver más información sobre este tema ver el tutorial de **Selenium Core (Tutorial)**)

Para poder controlar el navegador, Selenium Remote Control proporciona un servidor ("servidor Selenium") desde el cual se permite arrancar, controlar y detener "casi" cualquier navegador. Digo lo de "casi" por lo que he dicho antes, tiene que tener Javascript habilitado y una configuración de seguridad apropiada.

Comentar que en el fondo el servidor Selenium utiliza la herramienta Selenium Core para ejecutar los test.

2. Entorno.

El tutorial está escrito usando el siguiente entorno:

- Hardware: Portátil Asus G50Vseries (Core Duo P8600 2.4GHz, 4GB RAM, 320 GB HD).
- Sistema operativo: Windows Vista Ultimate.
- Internet Explorer 7.0.6001.1800
- Mozilla Firefox 3.0.6
- Opera 9.63
- Safari 3.2.1

3. Requerimientos.

En este punto se indicará la compatibilidad de este plugin con diferentes configuraciones (navegador / sistema operativo / lenguaje de programación), describiendo las acciones permitidas o bien los problemas encontrados al ejecutarlo con esa configuración.

Navegador:

Navegador	Funcionamiento
Firefox 3	Iniciar navegador y reproducir test
Firefox 2	Iniciar navegador y reproducir test
IE 8b1	?
IE 7	Iniciar navegador y reproducir test
Safari 3	Iniciar navegador y reproducir test
Safari 2	Iniciar navegador y reproducir test
Opera 9	Iniciar navegador y reproducir test
Opera 8	Iniciar navegador y reproducir test
Otros	Posible soporte parcial (*)

(*) Selenium Remote Control puede arrancar y ejecutar cualquier navegador, pero depende de la configuración del navegador el poderse utilizarlo con alguna limitación en su funcionalidad.

Sistema operativo:

Sistema operativo	Funcionamiento
Windows	Iniciar navegador y reproducir test
OS X	Iniciar navegador y reproducir test
Linux	Iniciar navegador y reproducir test
Solaris	Iniciar navegador y reproducir test
Otros	Iniciar navegador y reproducir test (*)

(*) El servidor Selenium Remote Control esta escrito en Java, por lo tanto debería de poderse ejecutarse en cualquier sistema que soporte Java y que tuviera un navegador disponible.

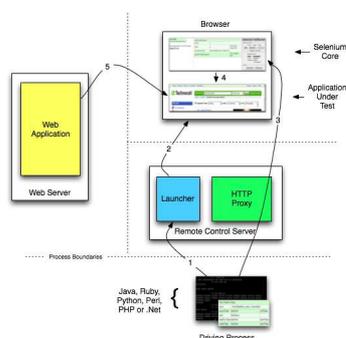
Lenguaje de Programación:

Lenguaje	Funcionamiento
C#	Soporte librería ("driver")
Java	Soporte librería ("driver")
Perl	Soporte librería ("driver")
PHP	Soporte librería ("driver")
Python	Soporte librería ("driver")
Ruby	Soporte librería ("driver")
Otros	Comandos vía petición HTTP (*)

(*) Cualquier lenguaje que permite peticiones HTTP debería de funcionar sin problemas.

4. Funcionamiento.

Gráfico de funcionamiento



Descripción del funcionamiento:

El cliente tendrá el test adaptado al lenguaje de programación con el que se realizará la prueba. Para ello, hará uso de las librerías proporcionadas para manejar los objetos necesarios durante la prueba (Ver apartado de configuración).

Estos objetos permiten manejar la aplicación.

Lo primero que hará el cliente es conectarse al servidor Selenium, el cual deberá de estar arrancado como un proceso independiente y activo durante toda la prueba. En caso de que por alguna necesidad se necesite interrumpir dicho servidor se realizará cualquier actividad de parada o arranque desde la línea de comandos (esta parte será explicada en el apartado de configuración).

Después el servidor Selenium ejecutará el navegador seleccionado para el test, para ello abrirá dos ventanas

- **Ventana de Selenium Core:** Visualiza los comandos que se están ejecutando.
- **Ventana de página web:** Visualiza las "respuestas visuales" realizadas por los comandos anteriores.

Dichas ventanas serán visibles desde el cliente que las ejecuto.

Es muy importante decir que la ejecución del navegador se realiza mediante el uso de un servidor proxy, el cual se establece entre el navegador y el sitio web. De esta forma, Selenium permite habilitar un navegador para poder ejecutarse en sitios arbitrarios.

Nota: El servidor Selenium no necesita correr en la misma máquina virtual (JVM) o en la misma máquina física.

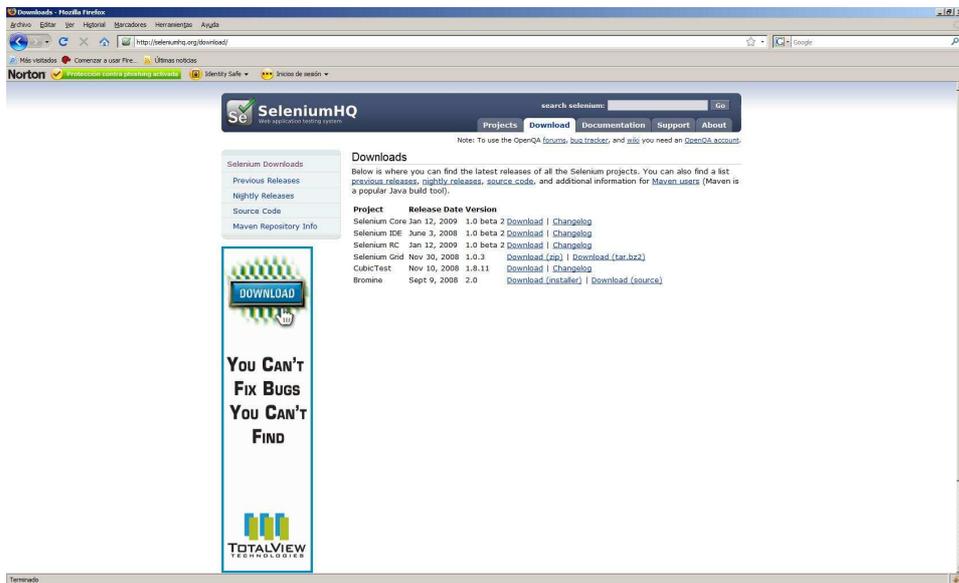
Una vez el servidor Selenium recibe la primera instrucción del cliente (vía HTTP proxy), realiza la acción recibida en la primera instrucción. Posteriormente el servidor web se "sincronizará" con la página, visualizando los cambios realizados (se podrán ver los resultados de las acciones realizadas en la ventana de página web).

Se aconseja usar los clientes integrados con algún tipo de framework de prueba, por ejemplo para Java JUnit o Testng.

5. Instalación.

Estos son los pasos que hay que realizar para instalar Selenium Remote Control:

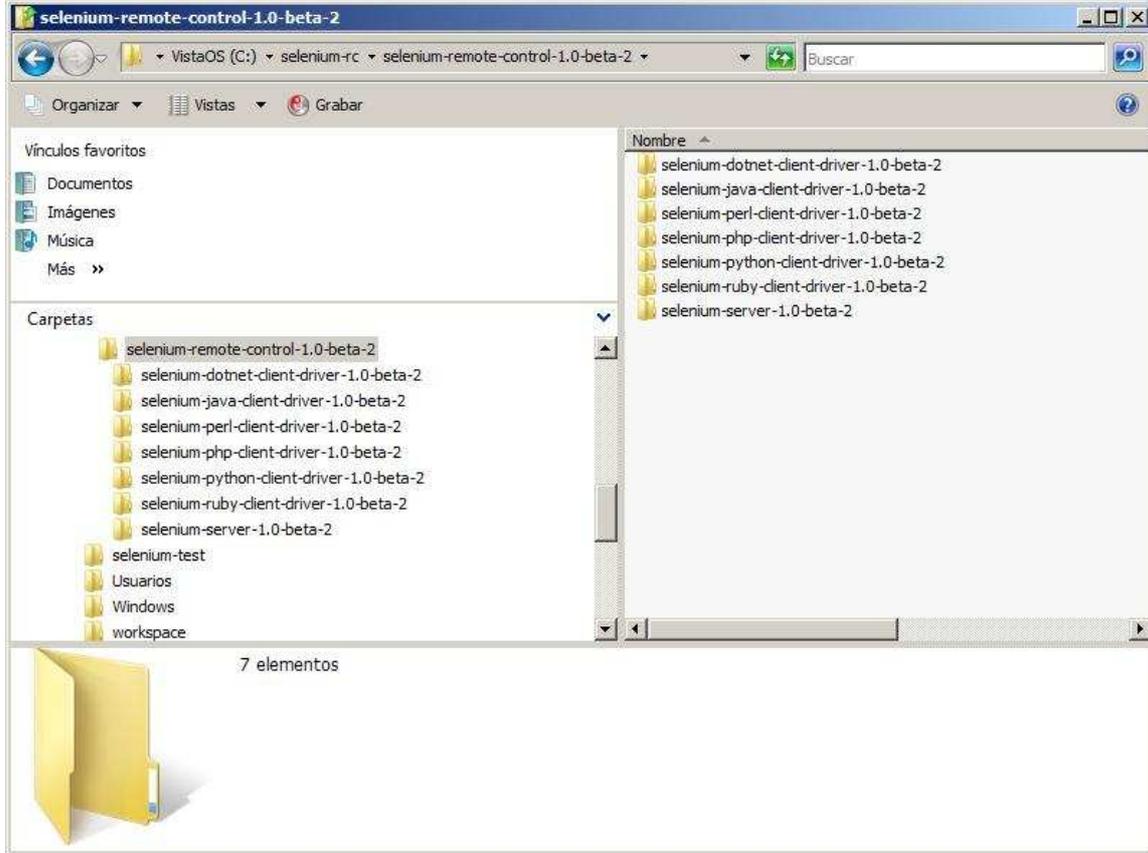
Paso 1: Descargar Selenium Remote control desde su página web <http://seleniumhq.org/download/>.



Paso 2: Pulsar sobre download.



Paso 3: Una vez se haya descargado, descomprimir el archivo en un directorio. Es aconsejable que el nombre del directorio sea Selenium Remote Control X.X (siendo el valor X.X la versión), de esta manera tendremos controlada la versión en todo momento.



Como se puede observar en la imagen, al descomprimir la carpeta se han generado los directorios correspondientes a las librerías de los lenguajes con los que se puede utilizar Selenium RC. Por otro lado, en esta misma ubicación se encuentra la carpeta correspondiente al servidor Selenium.

6. Configuración.

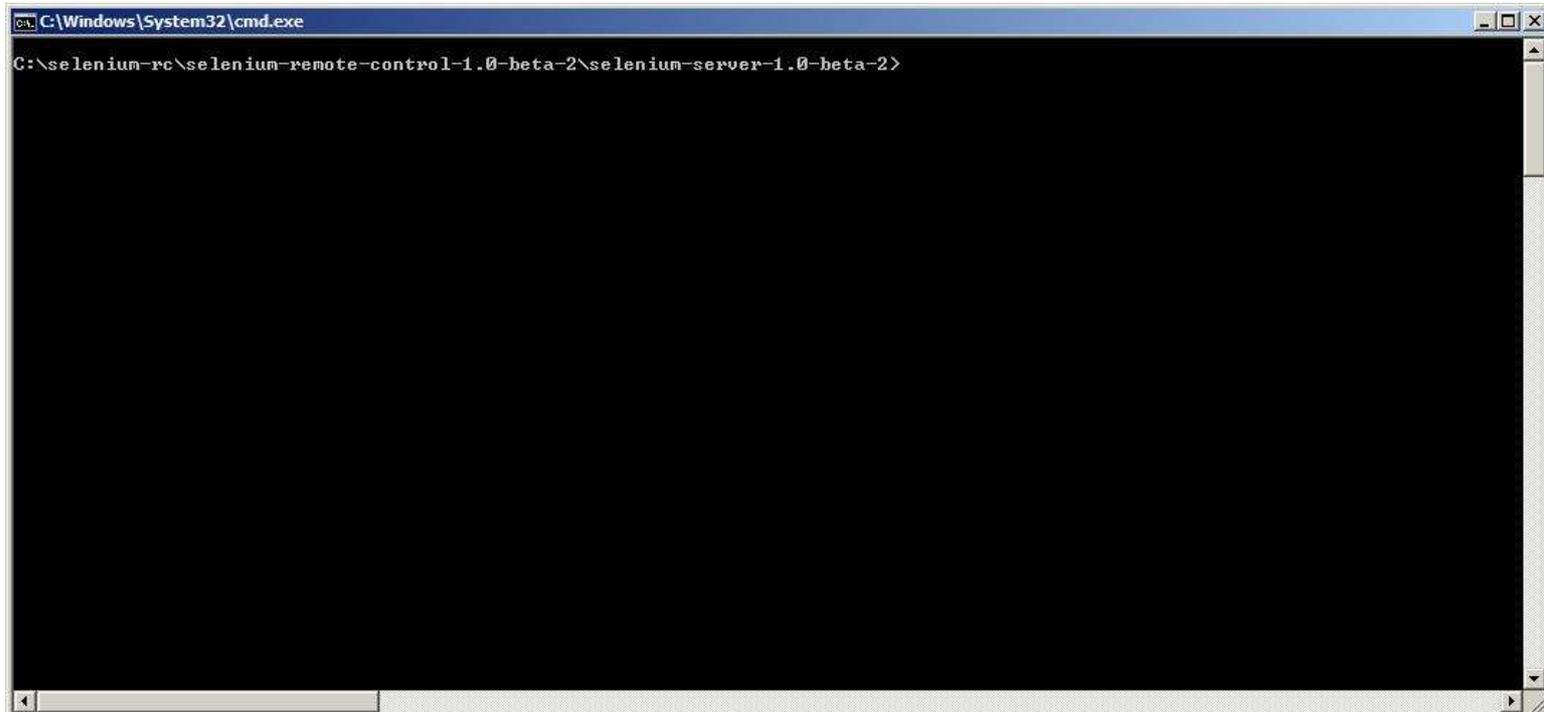
En este apartado se explicará en detalle las operaciones principales que se podrán utilizar con el servidor Selenium.

Primero de todo indicar que existen dos formas de arrancar el servidor Selenium :

- **Modo normal** : No permite modificar las condiciones de ejecución de forma directa. (Los cambios se introducen al arrancarlo)
- **Modo interactivo** : Permite modificar las condiciones de ejecución de forma directa.

Ejecutar el servidor Selenium en modo normal

Para poder arrancar el servidor Selenium con la configuración utilizada por defecto hay que abrir la consola y situarse en la carpeta donde se ha descomprimido el servidor Selenium.

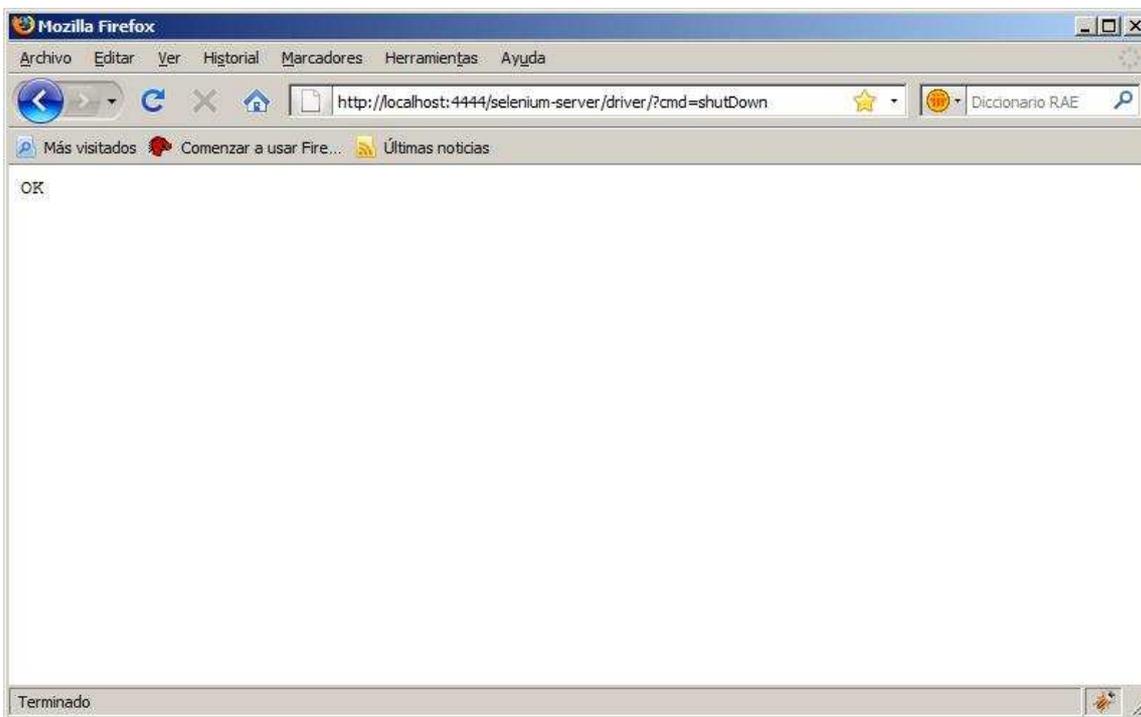


Escribir la siguiente instrucción : **java -jar selenium-server.jar**

```
C:\Windows\System32\cmd.exe - java -jar selenium-server.jar
12:43:22.780 INFO - Java: Sun Microsystems Inc. 1.5.0_15-b04
12:43:22.780 INFO - OS: Windows Vista 6.0 x86
12:43:22.782 INFO - v1.0-beta-2 [25711, with Core v1.0-beta-2 [23301]
12:43:22.865 INFO - Version Jetty/5.1.x
12:43:22.866 INFO - Started HttpContext[/,/]
12:43:22.867 INFO - Started HttpContext[/selenium-server,/selenium-server]
12:43:22.867 INFO - Started HttpContext[/selenium-server/driver,/selenium-server/driver]
12:43:22.894 INFO - Started SocketListener on 0.0.0.0:4444
12:43:22.894 INFO - Started org.mortbay.jetty.Server@2e7263
```

Para poder parar esta ejecución hay que escribir en la URL de cualquier navegador la siguiente dirección:
http://localhost:4444/selenium-server/driver/?cmd=shutDown

Con el siguiente mensaje de "OK" se indicará que se ha cerrado el servidor correctamente.



Como se ha podido ver en las indicaciones anteriores, hemos aprendido a arrancar el servidor Selenium con las opciones de configuración básicas. Decir, que estas opciones se pueden configurar desde la línea de comandos al arrancarlo para hacerlo de forma "avanzada" para ello se introducirán modificaciones como los siguientes:

Uso :java -jar selenium-server.jar [-interactive] [options]

Ejemplos de opciones:

- -port <num> : El puerto que Selenium debería de utilizar (Por defecto 4444)
- -timeout <num> : El número de segundos de debería de tener de timeout.

Para visualizar el resto de opciones de arranque visitar la siguiente página <http://seleniumhq.org/documentation/remote-control/options.html>

Ejecutar el servidor Selenium en modo interactivo

Para poder arrancar el servidor Selenium en modo interactivo hay que abrir la consola y situarse en la carpeta donde se descomprimos el servidor Selenium.

```
C:\Windows\System32\cmd.exe
C:\selenium-rc\selenium-remote-control-1.0-beta-2\selenium-server-1.0-beta-2>
```

Escribir la siguiente instrucción : **java -jar selenium-server.jar -interactive**

```
C:\Windows\System32\cmd.exe - java -jar selenium-server.jar -interactive
C:\selenium-rc\selenium-remote-control-1.0-beta-2\selenium-server-1.0-beta-2>java -jar selenium-server.jar -interactive
12:57:57.625 INFO - Java: Sun Microsystems Inc. 1.5.0_15-b04
12:57:57.626 INFO - OS: Windows Vista 6.0 x86
12:57:57.627 INFO - v1.0-beta-2 [2571], with Core v1.0-beta-2 [2330]
12:57:57.709 INFO - Version Jetty/5.1.x
12:57:57.710 INFO - Started HttpContext[/,/]
12:57:57.712 INFO - Started HttpContext[/selenium-server,/selenium-server]
12:57:57.712 INFO - Started HttpContext[/selenium-server/driver,/selenium-server/driver]
12:57:57.739 INFO - Started SocketListener on 0.0.0.0:4444
12:57:57.739 INFO - Started org.mortbay.jetty.Server@2e7263
Entering interactive mode... type Selenium commands here (e.g: cmd=open&l=http://www.yahoo.com)
-
```

También para poder parar esta ejecución hay que escribir en la URL del navegador la siguiente dirección:

<http://localhost:4444/selenium-server/driver/?cmd=shutDown>

O bien ejecutar un comando que realiza la misma operación, como se puede ver en la imagen al ejecutarse en modo interactivo (y permitir la introducción de comandos por consola) se puede conseguir el mismo efecto si se escribe la instrucción : **quit**

```

C:\Windows\System32\cmd.exe
C:\selenium-rc\selenium-remote-control-1.0-beta-2\selenium-server-1.0-beta-2>java -jar selenium-server.jar -interactive
12:57:57.625 INFO - Java: Sun Microsystems Inc. 1.5.0_15-b04
12:57:57.626 INFO - OS: Windows Vista 6.0 x86
12:57:57.627 INFO - v1.0-beta-2 [25711, with Core v1.0-beta-2 [2330]]
12:57:57.709 INFO - Version Jetty/5.1.x
12:57:57.710 INFO - Started HttpContext[/,/]
12:57:57.712 INFO - Started HttpContext[/selenium-server,/selenium-server]
12:57:57.712 INFO - Started HttpContext[/selenium-server/driver,/selenium-server/driver]
12:57:57.739 INFO - Started SocketListener on 0.0.0.0:4444
12:57:57.739 INFO - Started org.mortbay.jetty.Server@2e7263
Entering interactive mode... type Selenium commands here (e.g: cmd=open&1=http://www.yahoo.com)
quit
Stopping...
13:01:30.334 INFO - Stopping Acceptor ServerSocket[addr=0.0.0.0/0.0.0.0, port=0, localport=4444]
13:01:30.434 INFO - Stopped SocketListener on 0.0.0.0:4444
13:01:30.454 INFO - Stopped HttpContext[/,/]
13:01:30.472 INFO - Stopped HttpContext[/selenium-server,/selenium-server]
13:01:30.489 INFO - Stopped HttpContext[/selenium-server/driver,/selenium-server/driver]
13:01:30.489 INFO - Stopped org.mortbay.jetty.Server@2e7263
C:\selenium-rc\selenium-remote-control-1.0-beta-2\selenium-server-1.0-beta-2>_

```

Debido a que su ejecución es en modo interactivo podemos modificar las condiciones de ejecución para ello permite introducir comandos como:

-> cmd=open&1=http://www.google.com

-> cmd=getNewBrowserSession&1=*iexplore&2=http://www.google.com

De esta manera podremos realizar cambio en caliente de las configuraciones sobre las que realizaremos las pruebas.

Navegadores

Es importante tener claro bajo que tipo de navegador se va a utilizar la prueba, debido a que habrá que especificarlo en el comando que permitirá controlar al navegador. Para ellos hay que distinguir dos tipos de familias de navegadores:

Normales

Representan a los navegadores habituales que se suelen utilizar Selenium RC.

Navegador	Constante que lo representa
Internet Explorer	*iexplore
Firefox	*firefox
Safari	*safari
Custom	*custom /path/to/browser

Experimentales

Son aquellos que permiten probar las aplicaciones en cualquier sitio. Hay que distinguir 2 tipos:

- Subir los privilegios de seguridad:

Los navegadores que aquí se muestran no requieren el uso de servidores proxy.

Navegador	Constante que lo representa	Descripción
Internet Explorer en modo HTA	*iehta	Ejecuta Internet Explorer en modo HTA (Incrementa sus privilegios y permite escribir en disco)
Firefox	*chrome	Ejecuta Firefox con la URL del Chrome

- Inyección de proxy:

Los navegadores que aquí se muestran usan servidores proxy "agresivos" para poder ejecutar las acciones sobre cualquier sitio, incluso permiten "modificar" la aplicación.

Navegador	Constante que lo representa
Internet Explorer en modo HTA	*piexplore
Firefox	*piifirefox

Nota: Para poder utilizar estos dos navegadores hay que arrancar el servidor Selenium de un modo especial: **java -jar selenium-server.jar -proxyInjectionMode**

El uso de este tipo de navegadores es menos frecuente, por regla general no es necesario su uso a no ser que las condiciones de seguridad sean muy complejas.

Objeto DefaultSelenium

En las librerías de los lenguajes de programación descargadas en la carpeta de instalación se dispondrán de los objetos desde los cuales nos van a permitir manejar al navegador. En este apartado se enseñará a utilizar el objeto disponible para JAVA.

Recordar que este objeto será usado desde el cliente y que evidentemente diferirá en su uso en función del lenguaje de programación utilizado.

Constructor: Selenium selenium = new DefaultSelenium(String seleniumServerHost, int seleniumServerPort, String browserType, String baseURL);

- **String seleniumServerHost:** Es el lugar donde se va a ejecutar el servidor Selenium (Por defecto será localhost)
- **int seleniumServerPort:** Es el puerto por el cual el servidor Selenium estará a la escucha (Por defecto 4444)
- **String browserType:** Es el navegador utilizado en la prueba (Será la constante que representa al navegador)
- **String baseURL:** Es la URL base sobre la que se va a probar.

OJO: El objeto creado queda vinculado a una URL concreta y sólo se puede utilizar con esa URL. Así que no puede cambiar de URL base en mitad del script.

Por otro lado cuando se quiera arrancar el navegador se utilizará la siguiente instrucción:

```
selenium.open("http://www.site.com/page.html");
```

Y cuando se quiera finalizar el uso del navegador se utilizará la siguiente instrucción:

```
selenium.stop();
```

7. Integración con JUnit.

JUnit es un framework que permite realizar pruebas unitarias sobre aplicaciones Java.

Se basa principalmente en el concepto de **caso de prueba (test case)**.

Un **caso de prueba** es la forma de probar una parte del sistema en función de sus datos de entrada, condiciones de ejecución y del resultado esperado. Su representación en código fuente suele ser una clase que dispone de métodos para probar a su vez los métodos de una clase o módulo concreto.

Por regla general cada clase que quisiéramos probar debería de tener su clase de caso de prueba.

Para poder tener organizados los casos de prueba se suele utilizar el concepto de **suite de prueba (test suite)**

Una **suite de pruebas** es un conjunto de casos de prueba, los casos de prueba deberían de estar relacionados de alguna forma.

Funcionamiento: Se evalúa el resultado esperado respecto a unos datos de entrada concretos. En caso de verificarse que el resultado obtenido es igual que el esperado, JUnit indicará que la prueba se ha realizado con éxito, en caso contrario, mostrará el error en el método especificado.

Instalación

Paso 1: Descargar JUnit 4.5 de www.junit.org

Paso 2: Descomprimir en una carpeta del sistema. Por ejemplo : C:\JUnit4.5\

Paso 3: Comprobar que el JDK utilizado está en la variable de entorno PATH.

Paso 4: Incluir en la variable de entorno CLASSPATH las librerías: selenium-java-client-driver.jar / junit-4.4.jar;

Plantillas JUnit

Plantilla de Caso de prueba (Test Case)

Las condiciones para que se considere una clase de prueba test case son las siguientes:

- La clase creada tendrá un nombre con el siguiente formato <NombreClase>Test.
- La clase creada heredará de la clase junit.framework.TestCase.
- Dispondrá de un método setUp(): Reserva los recursos necesarios para realizar la prueba.
- Dispondrá de métodos test<NombreCasoPrueba>(): Realizará una prueba específica.
- Dispondrá de un método tearDown(): Libera los recursos necesarios tras realizar la prueba.

```
view plain print ?
01. import junit.framework.TestCase;
02. import com.thoughtworks.selenium.*;
03.
04. public class <NombreClase>Test extends TestCase {
05.
06.     private Selenium browser;
07.
08.     public void setUp() {
09.         browser = new DefaultSelenium(<SERVIDOR>,<PUERTO>,<NAVEGADOR>,<URL>);
10.         browser.start();
11.     }
12.
13.     public void test<NombreCasoPrueba>() throws InterruptedException {
14.
15.         <Código Java del caso de prueba exportado de Selenium IDE >
16.         //verifyTrue(browser.isTextPresent("Primeros pasos con Selenium IDE"));
17.     }
18.
19.     public void tearDown() {
20.         browser.stop();
21.     }
22. }
```

OJO: Los comandos **verify** grabados por Selenium IDE no son reconocidos por JUnit, se aconseja cambiar estos métodos por **assert**.

Plantilla de Suite de prueba (Test Suite)

Las condiciones para que se considere una clase test suite son las siguientes:

- La clase tendrá el nombre que se quiera, pero se aconseja incluir el sufijo "TestSuite".
- La clase heredará de la clase junit.framework.TestSuite.
- Incorporar método suite(): Devuelve una instancia de la clase TestSuite.
- Los casos de prueba se indicarán en el método suite().
- El método suite se llamará desde el método main(String[] args).

```

view plain print ?
01. import junit.framework.Test;
02. import junit.framework.TestSuite;
03. import junit.textui.TestRunner;
04.
05.
06. public class <NombreClase>TestSuite extends TestSuite {
07.
08.     public <NombreClase>TestSuite(String name) {
09.         super(name);
10.     }
11.
12.     public static void main(String[] args) {
13.         TestRunner.run(suite());
14.     }
15.
16.     public static Test suite() {
17.         TestSuite suite = new <NombreClase>("Ejemplo Test Suite Java");
18.
19.         //Establecemos los casos de prueba a ejecutar
20.         suite.addTestSuite(<CasoPrueba1>.class);
21.         suite.addTestSuite(<CasoPrueba2>.class);
22.         .
23.         .
24.         .
25.         suite.addTestSuite(<CasoPruebaN>.class);
26.
27.         return suite;
28.     }
29. }

```

8. Ejemplo.

Generar Script : Acceso a la página de "Adictos al trabajo" desde Google

Este ejemplo esta sacado de la generación de código en formato Java que se obtiene de la herramienta Selenium IDE. Ver [Tutorial Selenium IDE](#)

Objetivo: Acceder a la página web de "Adictos al trabajo" desde la página del buscador "Google".

Nombre del fichero: AccederAdictosDesdeGoogleTest.java

Generar un fichero con el nombre anterior y el contenido siguiente:

```

view plain print ?
01. import junit.framework.TestCase;
02. import com.thoughtworks.selenium.*;
03.
04. public class AccederAdictosDesdeGoogleTest extends TestCase {
05.
06.     private final String URL = "http://www.google.com";
07.     private final String BROWSER_TYPE = "*firefox";
08.
09.     private Selenium browser;
10.
11.     public void setUp() {
12.         browser = new DefaultSelenium("localhost",4444,BROWSER_TYPE,URL);
13.         browser.start();
14.     }
15.
16.     public void testAccederAdictosDesdeGoogle() throws InterruptedException {
17.
18.         browser.open("http://www.google.es/");
19.         assertEquals("Google", browser.getTitle());
20.         browser.type("q", "Adictos al trabajo");
21.         browser.click("btnG");
22.         browser.waitForPageToLoad("50000");
23.         assertEquals("Adictos al trabajo - Buscar con Google", browser.getTitle());
24.         browser.click("link=Adictos al Trabajo. Formación y desarrollo | JAVA, JEE, UML, XML ...");
25.         browser.waitForPageToLoad("50000");
26.         assertEquals("Adictos al Trabajo. Formación y desarrollo | JAVA, JEE, UML, XML |. Tutoriales sobre nuev:
27.     }
28.
29.     public void tearDown() {
30.         browser.stop();
31.     }
32. }

```

Generar Script : Acceso al tutorial de "Primeros pasos con Selenium IDE" desde la pagina de "Adictos al trabajo"

Este ejemplo esta sacado de la generación de código en formato Java que se obtiene de la herramienta Selenium IDE.

Objetivo: Acceder al tutorial "Primeros pasos con Selenium IDE" desde la página web de "Adictos al trabajo".

Nombre del fichero: AccederTutorialSeleniumDesdeAdictosTest.java

Generar un fichero con el nombre anterior y el contenido siguiente:

```

view plain print ?
01. import junit.framework.TestCase;
02. import com.thoughtworks.selenium.*;
03.
04. public class AccederTutorialSeleniumDesdeAdictosTest extends TestCase {
05.
06.     private final String URL = "http://www.adictosaltrabajo.com/";
07.     private final String BROWSER_TYPE = "**firefox";
08.
09.     private Selenium browser;
10.
11.     public void setUp() {
12.         browser = new DefaultSelenium("localhost",4444,BROWSER_TYPE,URL);
13.         browser.start();
14.     }
15.
16.     public void testAccederTutorialSeleniumDesdeAdictos() throws InterruptedException {
17.
18.         browser.open("http://www.adictosaltrabajo.com/");
19.         assertEquals("Adictos al Trabajo. Formación y desarrollo | JAVA, JEE, UML, XML |. Tutoriales sobre nuev:
20.         browser.click("link=Quienes somos");
21.         browser.waitForPageToLoad("50000");
22.         assertEquals("Adictos al Trabajo. Formación y desarrollo | JAVA, JEE, UML, XML |. Tutoriales sobre nuev:
23.         browser.click("/a[@href='tutoriales-autor.php?autor=44']");
24.         browser.waitForPageToLoad("50000");
25.         assertEquals("Adictos al Trabajo. Formación y desarrollo | JAVA, JEE, UML, XML |. Tutoriales sobre nuev:
26.         browser.click("link=Primeros pasos con Selenium IDE");
27.         browser.waitForPageToLoad("50000");
28.         assertEquals("Adictos al Trabajo. Formación y desarrollo | JAVA, JEE, UML, XML |. Tutoriales sobre nuev:
29.         assertTrue(browser.isTextPresent("Primeros pasos con Selenium IDE"));
30.     }
31.
32.     public void tearDown() {
33.         browser.stop();
34.     }
35. }

```

Generar Suite

Objetivo: Generar un Test Suite con los Casos de prueba anteriores.

Nombre del fichero: EjemploTestSuite.java

Generar un fichero con el nombre anterior y el contenido siguiente:

```

view plain print ?
01. import junit.framework.Test;
02. import junit.framework.TestSuite;
03. import junit.textui.TestRunner;
04.
05.
06. public class EjemploTestSuite extends TestSuite {
07.
08.     public EjemploTestSuite(String name) {
09.         super(name);
10.     }
11.
12.     public static void main(String[] args) {
13.         TestRunner.run(suite());
14.     }
15.
16.     public static Test suite() {
17.         TestSuite suite = new EjemploTestSuite("Ejemplo Test Suite Java");
18.
19.         //Establecemos los casos de prueba a ejecutar
20.         suite.addTestSuite(AccederAdictosDesdeGoogleTest.class);
21.         suite.addTestSuite(AccederTutorialSeleniumDesdeAdictosTest.class);
22.
23.         return suite;
24.     }
25.
26. }

```

Generar Compilador

Para compilar las clases se ha realizado un ejecutable .bat con el código siguiente:

OJO: Fijaros que las librerías utilizadas se le pasan como parámetro, y estas se encuentran en una subcarpeta "lib" en el lugar donde se encuentra el ejecutable.

Los códigos fuente de los casos de prueba y de la suite se encuentran en una carpeta "src" en el mismo nivel que el ejecutable encargado de compilar.

Generar Ejecutable

```

view plain print ?
01. @echo off
02. echo
03. echo "-----"
04. echo "Ejemplo Test Selenium-RC - build"
05. echo "-----"
06.
07. rmdir /S/Q build
08.
09. mkdir build
10. mkdir build\test-classes
11.
12. set CP="lib\junit-4.5.jar;lib\selenium-java-client-driver.jar"
13. set CP=%CP%;"build\test-classes"
14.
15. javac -classpath %CP% -sourcepath src -d build\test-classes src\EjemploTestSuite.java
16.
17. pause

```

Para ejecutar las pruebas bastará con generar un fichero ejecutable .bat con el código siguiente:

OJO: Fijaros que las librerías utilizadas se le pasan como parámetro, y estas se encuentran en una subcarpeta "lib" en el lugar donde se encuentra el ejecutable.

```
view plain print ?
01. @echo off
02.
03. echo "-----"
04. echo "Ejemplo Test Selenium-RC - test"
05. echo "-----"
06.
07. set CP="lib\junit-4.5.jar;lib\selenium-java-client-driver.jar"
08. set CP=%CP%;"build\test-classes"
09.
10. java -classpath %CP% EjemploTestSuite
11.
12. pause
```

Ejecución de ejemplo

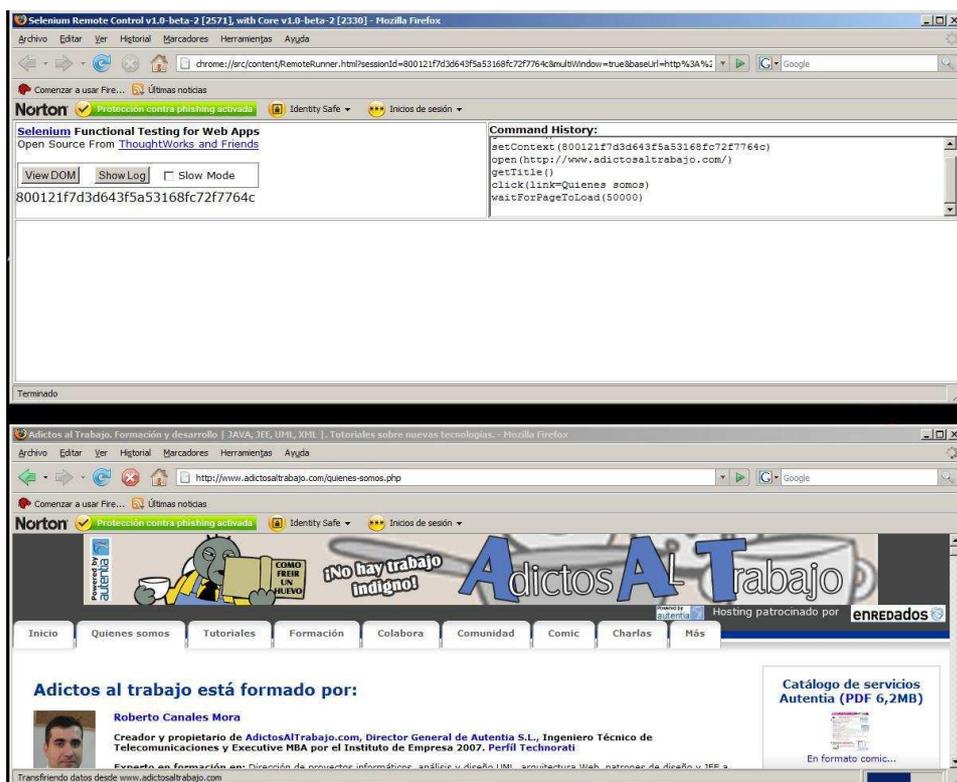
Paso 1 : Arrancar el servidor Selenium.

Paso 2 : Compilar las clases a probar.

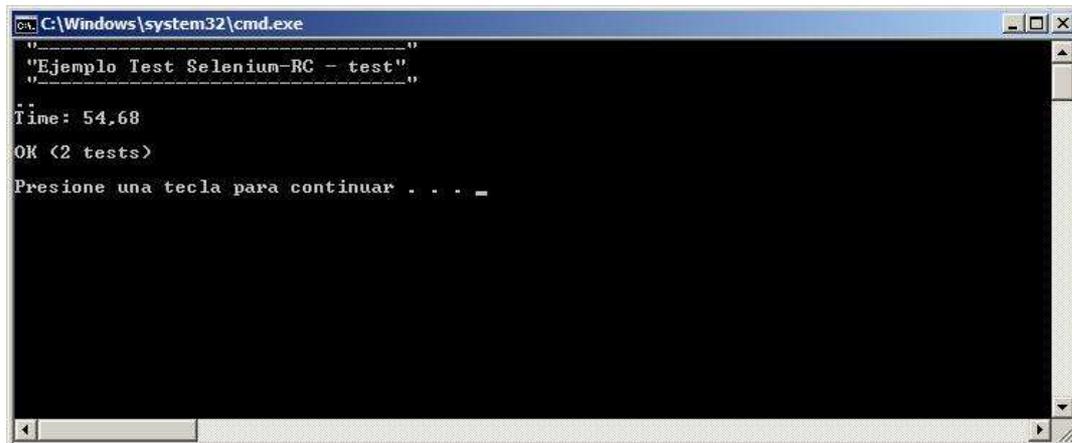
Paso 3 : Ejecutar las clases a probar.

Paso 4 : Parar el servidor Selenium.

A continuación se comenzarán a ejecutar las pruebas indicadas en el Test Suite para ello se cargarán las dos ventanas indicadas donde se mostrarán los comandos y resultados de los comandos en cada ejecución de los comandos de los scripts.



Al final se mostrará una ventana que resume la ejecución de las pruebas, en nuestro caso se muestra que las dos pruebas que tenía asignadas se han ejecutados de forma exitosa.



9. Conclusiones.

En este tutorial os he presentado la mejor forma de averiguar si se ha producido una regresión en nuestro SW, así que se acaba el repetir siempre las mismas pruebas cada vez que se libera una versión. Imaginaos todo el tiempo que dedicamos a esta parte y lo cómodo que nos va a resultar "despachar" este trabajo a la misma aplicación, es decir, que se pruebe ella....jejeje ;-)

Espero haberos podido ayudar a entender un poquito más esta "chulísima" aplicación. Y sobre todo haberos aclarado la parte de integración con JUnit y Maven. De su integración con Maven se hablará en un siguiente tutorial.

Un saludo.

Víctor

mailto:vmadrid@autentia.com

¿Qué te ha parecido el tutorial? Déjanos saber tu opinión y ¡vota!

Muy malo Malo Regular Bueno Muy bueno



Votar

- Puedes opinar sobre este tutorial [haciendo clic aquí](#).
- Puedes firmar en nuestro libro de visitas [haciendo clic aquí](#).
- Puedes asociarte al grupo AdictosAlTrabajo en XING [haciendo clic aquí](#).
- Añadir a favoritos Technorati.  ADD THIS BLOG TO MY **Technorati FAVORITES**



Esta obra está licenciada bajo [licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5](#)

Recuerda

Autentia te regala la mayoría del conocimiento aquí compartido ([Ver todos los tutoriales](#)). Somos expertos en: J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ... y muchas otras cosas.

¿Nos vas a tener en cuenta cuando necesites consultoría o formación en tu empresa?, ¿Vas a ser tan generoso con nosotros como lo tratamos de ser con vosotros?

Somos pocos, somos buenos, estamos motivados y nos gusta lo que hacemos ...

Autentia = Soporte a Desarrollo & Formación.

info@autentia.com

soluciones reales para su **negoci**

Servicio de notificaciones:

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales.

Formulario de suscripción a novedades:

E-mail

Tutoriales recomendados

Nombre	Resumen	Fecha	Visitas	Valoración	Votos	Pdf
Selenium Core : El motor de Selenium.	Selenium Core es un aplicación perteneciente al juego de herramientas SeleniumHQ que permite realizar juegos de pruebas sobre aplicaciones web.	2009-02-16	256	Muy bueno	4	
Pruebas de Rendimiento y Funcionales Web	Jose María Toribio, nos enseña en este tutorial como podemos utilizar la aplicación gratuita JMeter para realizar pruebas de rendimiento y funcionales (vitales para la regresión y reingeniería) sobre nuestras aplicaciones Web	2005-04-17	36569	Muy bueno	8	
Primeros pasos con Selenium IDE	Nuestro amigo y compañero Víctor nos enseña, para acabar bien el año, el uso de una herramienta muy interesante para la realización de pruebas sobre aplicaciones web	2008-12-31	1330	Muy bueno	18	
Pruebas unitarias Web para aplicaciones JSF	En este tutorial se puede encontrar una introducción y un análisis de los diferentes frameworks disponibles para realizar pruebas unitarias web de aplicaciones JSF	2006-11-13	6837	Bueno	1	
Servicio Web con NetBeans 6 y prueba con SoapUI	En este tutorial os enseñamos cómo crear y probar un servicio web de una manera sencilla utilizando netbeans 6	2007-08-02	8945	Bueno	4	
JUnit 4. Pruebas de Software Java	Tutorial que describe como utilizar la herramienta JUnit 4 para realizar pruebas de integridad y errores sobre Java.	2006-06-02	12724	Bueno	1	
Pruebas unitarias con jwebunit	En este tutorial nos vamos a aproximar al framework jWebUnit, que es un proyecto muy interesante para realizar rápidamente una buena batería de pruebas unitarias para nuestra aplicación web.	2006-11-17	4339	-	-	
EJB 3.0 y pruebas unitarias con Maven, JUnit y Embedded JBoss	En este tutorial Alejandro Pérez nos enseña como realizar test unitarios sobre EJB 3.0. Para ello se usará Maven, JUnit y Embedded JBoss	2007-08-09	5663	-	-	
Pruebas de integración con Maven	Este tutorial nos muestra un ejemplo para lanzar las pruebas de integración "engañando" a Maven para que no se lanzen en la fase de test teniendo únicamente un módulo para ambas	2007-02-08	4412	-	-	
Pruebas Web con JWebUnit	Os mostramos como automatizar las pruebas de caja negra (desde el punto de vista de usuario final) de vuestro Web con el Framework gratuito JWebUnit. Esta técnica es perfecta para crear test de regresión de aplicaciones Web complejas.	2004-06-30	9608	-	-	

Nota:

Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento. Los contenidos y comentarios de los tutoriales son responsabilidad de sus respectivos autores. En algún caso se puede hacer referencia a marcas o nombres cuya propiedad y derechos es de sus respectivos dueños. Si algún afectado desea que incorporemos alguna reseña específica, no tiene más que solicitarlo. Si alguien encuentra algún problema con la información publicada en este Web, rogamos que informe al administrador rcanales@adictosaltrabajo.com para su resolución.