



Pruebas sobre aplicaciones web con SeleniumHQ

Realizada por : Víctor Madrid Triviño
Consultor Tecnológico

Índice General



www.autentia.com

- Introducción
- Terminología
- Paquete SeleniumHQ
 - Selenium IDE
 - Selenium Core
 - Selenium RC
 - Integración con JUnit.
 - Integración Maven 2 / Cargo / Tomcat.
 - Selenium Grid
- Conclusiones
- Ruegos y preguntas



Introducción

Introducción



www.autentia.com

- Empresa:
 - Incremento en las exigencias de calidad.
 - Exigencias del “Mercado”.
 - Mercado exige: Cosas mejores.
 - Productos software:
 - Incremento en la complejidad.
 - Incremento del tamaño.
 - Mantenible.
 - Principales necesidades en cuanto al SW:
 - Reducción en los costes.
 - Reducción en el tiempo.
 - Ver su “total” funcionamiento.

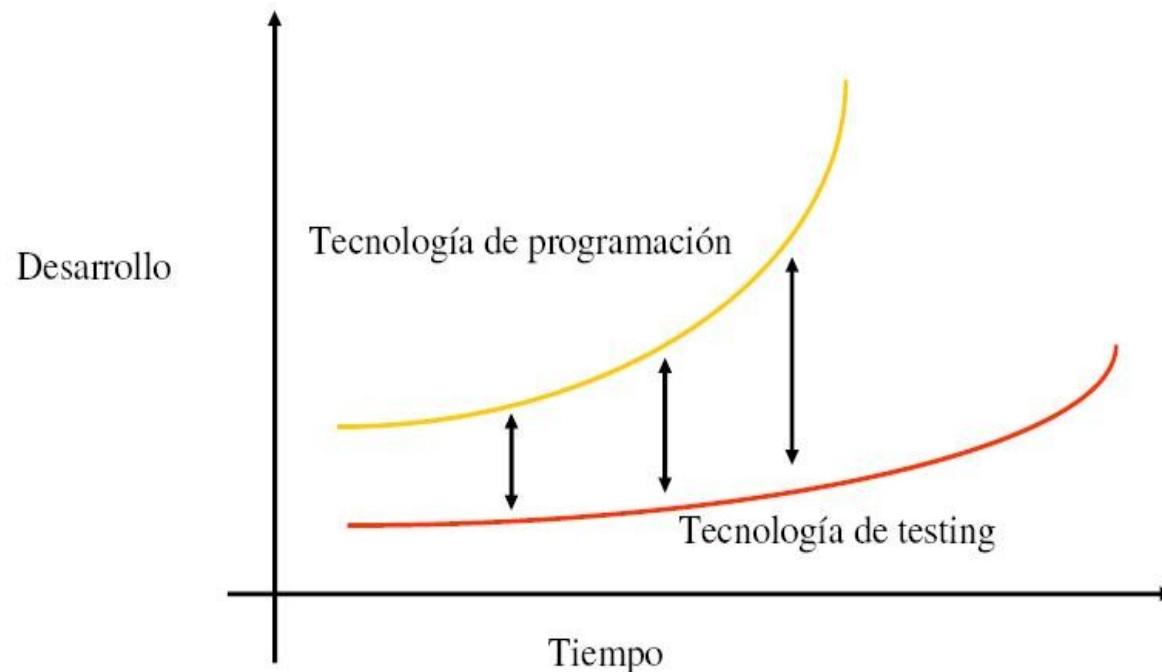


Introducción



www.autentia.com

- Gráfica Tecnología / Desarrollo / Tiempo





Terminología

Terminología



www.autentia.com

- **Caso de prueba (Test Case)**

Def: *Conjunto de condiciones o variables que verifican un requerimiento específico.*

- Hay al menos un caso de prueba por cada requisito.
- Algunas metodologías → (Test Case :Positivo / Negativo)
 - Ejemplo : Probar la autenticación con un sistema.
- Datos Entrada + Cond.Ejecución = Resultado Esperado (y Verificación).
- Clasificación de casos de prueba “escritos” :
 - Script de prueba.
 - Suite de prueba.

- **Procedimiento de prueba**

Def: *Método a seguir para realizar la prueba.*



- *Instrucciones detalladas para : Config. , Ejecución y Evaluación.*
- *Aplicable a 1 o + casos de prueba.*



- **Prueba de software**

Def: *Ejecución de un programa con la intención de descubrir un error.*

- Técnica experimental para la búsqueda de errores en los programas (Arte / > 50% Esfuerzo / Experiencia).
- Verifica la calidad de un producto SW.
- Objetivos:
 - Probar si el SW no hace lo que debe hacer.
 - Probar si el SW hace lo que no debe hacer → Daños Colaterales

Prueba de un sistema

Se define como el proceso de ejercitar o evaluar el sistema, por medios manuales o automáticos, para verificar que satisface los requerimientos o, para identificar diferencias entre los resultados esperados y los que produce el sistema (IEEE)

Terminología



www.autentia.com

– Principios de las pruebas:

- 1) Una prueba es un proceso que trata de **localizar errores**.
- 2) Es **imposible** probar completamente un sistema.
- 3) Una prueba implica ser **creativo**.
- 4) Una prueba permite **prevenir** posibles errores.
- 5) Las pruebas deberían de ser realizadas por **personas diferentes** a aquellas que desarrollaron el sistema.

– Técnicas para realizar pruebas:

- Utilizar datos de entrada bien definidos. (Ejemplo: $3 + 2 = 5$)
- Utilizar datos de prueba simples → pruebas complejas.
- Cuando se detecte un problema y se realicen las modificaciones oportunas , hay que realizar un sólo cambio en las pruebas cada vez que se utilice.
- Verificar la detección de datos de entrada incorrectos.



Cuanto antes se detecte un error, menor es el coste para solucionarlo

– Etapas de las pruebas

- Seleccionar qué es lo que se va a probar.
- Determinar cómo se va a probar.
- Desarrollar los casos de prueba.
- Determinar cuales deberían de ser los resultados esperados.
- Ejecutar los casos de prueba.
- Comparar los resultados de la prueba con los resultados esperados.





- **Automatización**

Def: *Proceso de conversión de tareas realizadas manualmente a tareas realizadas de manera automática.*

- Surge como complemento al proceso de pruebas.
- Implica el uso de SW para:
 - Controlar la ejecución de las pruebas.
 - Comparar los resultados.
 - Creación de un entorno de pruebas.
 - Otras funciones de control y análisis de las pruebas.

Hay que recordar que se trata de automatizar la prueba manual.

Terminología



www.autentia.com

– Diferencias entre prueba manual y prueba automatizada:

• Prueba manual:

- Adaptabilidad a cambios.
- Hace frente a la complejidad.
- Detección de errores de un vistazo.
- Introduce variantes útiles para descubrir bug.
- Sirve de exploración previa para definir los casos a automatizar



• Prueba automatizada:

- Se limita a la evaluación o verificación que se define explícitamente el script → Reproducción.
- Complementa al proceso de prueba manual.
- No sustituye al proceso de prueba.



Terminología



www.autentia.com

- El proceso de automatización de pruebas debe verse como un proyecto independiente.
 - Deben de asignarse:
 - Tiempo
 - Recursos
 - Personal (Especializado)
 - Requiere :
 - Planificación
 - Diseño
 - Desarrollo
 - Prueba
 - Control de versiones:
 - Scripts de prueba
 - Documentación
 - Aplicación



• Pruebas de regresión

Def: *Aquella prueba que trata de verificar que no ocurrió una regresión de la calidad del producto después del cambio realizado.*

- Ejecutar alguna o todas las pruebas realizadas anteriormente.
- Asegura que la incorporación de nuevo código no ha anulado los efectos anteriores.
- Se suelen realizar :
 - Durante el desarrollo del SW
 - Tareas de mantenimiento (correcciones / mejoras / adaptaciones)

Tipo de Regresión
Problemas antiguos
Problemas resueltos
Efectos colaterales



- **Herramientas de grabación**

Def:*Aquellas herramientas que permiten generar scripts a partir de la grabación de las acciones que realiza el usuario.*

- *Permiten la parametrización.*
 - *Utilización de datos dinámicos*
- *Generación del esqueleto del script.*
- *Asistencia en la identificación de objetos (Muy importante)*
 - *Firebug.*
 - *XPath.*



Terminología



www.autentia.com

• Desarrollo de un script

- Script bien estructurado
- Mantenible y flexible.
- Partes diferenciadas:
 - Datos de prueba
 - Datos del script
 - Flujo de acciones
- “Buenas prácticas” de programación.
- Definir y utilizar código comprensible.
- Cambios en la aplicación
 - Posición
 - Solución: Usar ID
 - Identificador (G. Automática)
 - Solución:
 - Parametrizar.
 - Otro atributos :
Por ejemplo: alt=""



Paquete SeleniumHQ



Conjunto de herramientas para automatizar las pruebas sobre aplicaciones web a través de diferentes plataformas.

- Funciona en “diferentes navegadores” y con “diferentes sistemas operativos”.
- Puede ser controlado por muchos lenguajes de programación y frameworks de prueba.
- Permite crear pruebas de regresión.
- Proyecto Open Source (Comunidad OpenQA)





- Esta compuesto por
 - **Selenium Core**
 - Ejecución de pruebas automatizadas
 - **Selenium IDE**
 - Creación y mantenimiento de pruebas automatizadas.
 - **Selenium Remote Control (RC)**
 - Creación de pruebas escritas en lenguajes de programación como Java o C#.
 - **Selenium GRID**
 - Ejecución de pruebas escritas en los anteriores lenguajes de forma paralela.
 - **Selenium on Rails**
 - Realización de pruebas sobre aplicaciones Rails con Selenium Core.
 - **Selenium on Ruby**
 - Proporciona el hub para relacionar Ruby con los proyectos Selenium
 - **CubicTest**
 - Plugin gráfico para Eclipse que permite escribir los test de Selenium



Selenium IDE

Selenium IDE



www.autentia.com

- **Selenium IDE**
 - Es un plugin de Firefox.
 - Pertenece al juego de herramientas SeleniumHQ.
 - Primera herramienta que hay que aprender a utilizar.
 - Funcionalidad:
 - Creación y mantenimiento de pruebas web automatizadas.
 - Reproducción del script generado en el navegador Firefox.
 - Tiene integrado **Selenium Core**.
 - Es una completa herramienta de desarrollo de pruebas web.
 - » Grabación de las acciones realizadas por un usuario.

No es solamente una herramienta de grabación.

Selenium IDE



www.autentia.com

– Características:

- Facilidad de grabación y ejecución de los test.
- Referencias a objetos DOM en base al ID, nombre o a través de XPath.
- Autocompletado de comandos.
- Herramienta de depuración y puntos de ruptura (breakpoints)
- Las acciones pueden ser ejecutadas paso a paso.
- Los test pueden ser almacenados como HTML u otros formatos.
- Soporte para Selenium: user-extension.js

– Lenguajes de programación:

Java / C# / Perl / Php /Python /Ruby

Selenium IDE



– Requerimientos : Navegador

Navegador	Funcionamiento
Firefox 2 / 3	Graba y reproduce test
IE 7/8b1 Safari 2/3 Opera 2/3 Otros	No esta soportado

– Requerimientos : Sistema operativo

Sistema Operativo	Funcionamiento
Windows OS X Linux Solaris	Trabaja con Firefox 2 o +
Otros	Debería de trabajar con Firefox 2 o +

– Requerimientos : Lenguaje de programación

Lenguaje de programación	Funcionamiento
C# Java Perl Php Python Ruby	Genera código
Otros	Genera código personalizado

Selenium IDE



www.autentia.com

– Instalación:

- 1) Conectarse a la página de descargas de SeleniumHQ
→ <http://www.seleniumhq.org/download/>
- 2) “Descargarse” Selenium IDE.
- 3) Instalar Selenium IDE.
- 4) Reiniciar el Firefox.

– Ejecución:

- Opción Ver → Panel lateral → Selenium IDE
- Opción Herramientas → Selenium IDE (Ventana nueva)

Selenium IDE

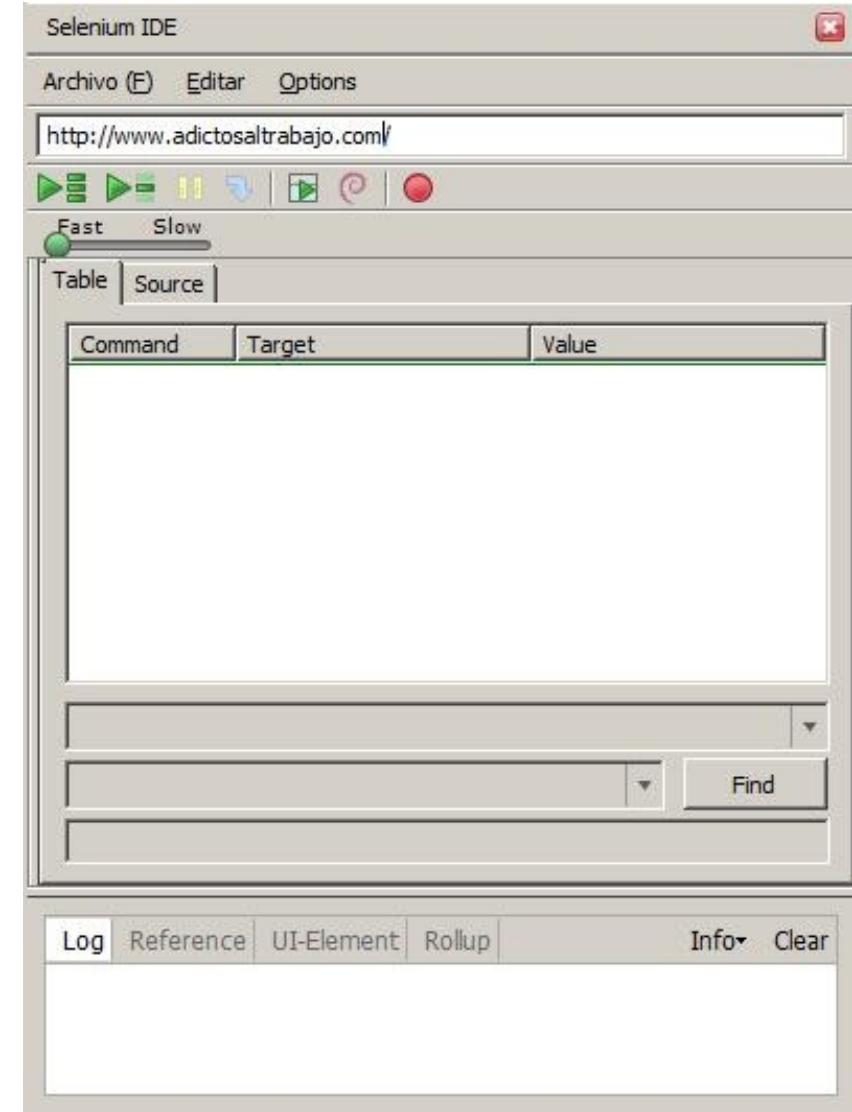


www.autentia.com



– Panel de control:

- Grabación (HTML):
 - Test Case
 - Test Suite
- Opciones configuración:
 - Codificación.
 - Timeout.
 - Formato.
 - Grabar assertTitle automáticamente.
 - Grabar URL absoluta.
- Opciones reproducción:
 - Reproducir todos.
 - Reproducir uno.
 - Test Runner.
 - Velocidad.
 - Grabar / Parar.



Selenium IDE



www.autentia.com



- Tabla
 - Comando.
 - Objetivo.
 - Valor.
- Código
 - Por defecto HTML.
- Panel Edición
 - Find : localiza elemento.
- Panel de información:
 - Log.
 - Referencia de los comandos.

The screenshot shows the Selenium IDE interface. At the top, there's a menu bar with "Archivo (F)", "Editar", and "Options". Below the menu is a toolbar with various icons for navigating and executing scripts. A speed slider is present between the toolbar and the main content area. The main area is divided into several panels:

- Table:** A table showing a list of recorded commands. The table has columns for "Command", "Target", and "Value". One row is selected, showing "assertTitle" as the command, "Google" as the target, and "Adictos al trabajo" as the value.
- Source:** A text input field containing the recorded script, which includes the "assertTitle" command.
- assertTitle:** A search panel with a dropdown menu set to "assertTitle", a text input field containing "Google", and a "Find" button.
- Log:** A panel showing the log output for the recorded session.
- Reference:** A panel displaying the documentation for the "assertTitle" command. It includes the command name, a note that it's generated from "getTitle()", the return type ("the title of the current page"), and a description ("Gets the title of the current page").
- UI-Element:** A panel showing a preview of the UI element being targeted.
- Rollup:** A panel showing a summary or rollup of the recorded session.

Selenium IDE



www.autentia.com

- Comandos

- *Def: Un comando es lo que le dice a Selenium que hacer.*
 - Hay 3 tipos:

- **Acciones (Actions)**

Modifican el estado de la aplicación.

Por ejemplo : Pulsar un botón / enlace.

Debido a su forma de finalizar pueden ser:

- **Incorrectas**: Si la acción tiene un error o falla en algún momento de su ejecución, entonces el test que se está ejecutando para.
 - **Correctas**: Si la acción no presenta errores continua con la ejecución.

- **Accesos (Accessors)**

Comprueban el estado de la aplicación y almacenan el resultado en variables.

Por ejemplo : “storeTitle” → Devuelve el título de la página actual.



- **Afirmaciones (Assertions)**

Son como los anteriores, con la diferencia de que estos comprueban el estado de la aplicación con lo que se esperaba.

Por ejemplo: Comprobar que el título de una página es XXXX

Las afirmaciones se pueden clasificar en 3 tipos:

- **Assert** : Cuando hay un fallo se aborta la prueba.
- **Verify** : Cuando hay un fallo continua con la ejecución → log.
- **WaitFor**: Espera a una condición.

Consejo : Con un único assert se comprueba que nos encontramos en la página correcta, es resto de comprobaciones deberían de ser del tipo verify.

Selenium IDE



- Localizadores (“Identificadores”):

- Def: Clave alfanumérica de búsqueda e identificación de un elemento.
- Los elementos que indican a la aplicación sobre qué elemento HTML se refiere un comando específico.
- Formato : *locatorType=argumento*
- Estrategias de localización:

Estrategia	Descripción
identifier=id	Selecciona el elemento con el atributo @id (No tiene → @name)
<u>id=id</u>	Selecciona el elemento con el atributo @id
name=name	Selecciona el primer elemento con el atributo @name
dom=javascriptExpression	Selecciona el elemento resultado de evaluar la expresión JS
<u>xpath=xpathExpression</u>	Selecciona el elemento resultado de evaluar la expresión XPath
link=textPattern	Selecciona el elemento indicado en el patrón
css=cssSelectorSyntax	Selecciona el elemento usando selectores CSS

Selenium IDE



www.autentia.com

- Ejemplos localizadores:

```
dom=document.forms['myForm'].myDropdown  
xpath=/input[@name='name2' and @value='yes']  
css=a[href="#id3"]
```

- user-extensions.js

- Características:

- Carga el código Javascript.
 - Nombre del fichero (Por defecto).
 - Permite incorporar características.
 - Extensión
 - La distribución no lo trae disponible. (Depende del Usuario)

Selenium IDE



www.autentia.com

– Plantilla Script de prueba (Código HTML)

```
01.  <?xml version="1.0" encoding="UTF-8"?>
02.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
03.  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
04.  <head profile="http://selenium-ide.openqa.org/profiles/test-case">
05.  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
06.  <link rel="selenium.base" href="" />
07.  <title>TITULO</title>
08.  </head>
09.  <body>
10.  <table cellpadding="1" cellspacing="1" border="1">
11.  <thead>
12.  <tr><td rowspan="1" colspan="3">TITULO</td></tr>
13.  </thead><tbody>
14.  <tr>
15.    <td>COMANDO</td>
16.    <td>OBJETIVO</td>
17.    <td>VALOR</td>
18.  </tr>
19.  .
20.  .
21.  .
22.  <tr>
23.    <td>COMANDO</td>
24.    <td>OBJETIVO</td>
25.    <td>VALOR</td>
26.  </tr>
27. </tbody></table>
28. </body>
29. </html>
```

Selenium IDE



- Plantilla Suite de prueba (Código HTML)
 - Orden secuencial.
 - Establecer localización Test Case

```
01.  <?xml version="1.0" encoding="UTF-8"?>
02.  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
03.  <html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
04.    <head>
05.      <meta content="text/html; charset=UTF-8" http-equiv="content-type" />
06.      <title>TITULO </title>
07.    </head>
08.    <body>
09.      <table id="suiteTable" cellpadding="1" cellspacing="1" border="1" class="selenium">
10.        <tbody>
11.          <tr><td><b>TITULO</b></td></tr>
12.          <tr><td><a href="LOCALIZACION">TITULO CASO DE PRUEBA</a></td></tr>
13.          .
14.          .
15.          .
16.          <tr><td><a href="LOCALIZACION">TITULO CASO DE PRUEBA</a></td></tr>
17.        </tbody>
18.      </table>
19.    </body>
20.  </html>
```

Selenium IDE



- Ejemplos Selenium IDE:
 - **Ejemplo 1 :** Grabación :Acceder a la web de “Adictos al trabajo” desde la página de Google.
Explicación breve Firebug.
Explicación breve XPath.
 - **Ejemplo 2 :**Ejecutar Test Suite → Acceder a tutorial de Selenium IDE.
 - Acceso desde a la web de “Adictos al trabajo”
 - Buscar el tutorial “Primeros pasos con Selenium IDE”.
 - **Ejemplo 3 :** Generalizar el ejemplo 1 para acceder a la primera página del buscador de Google.

Selenium IDE



www.autentia.com

- Comparativa de comandos Ejemplo 1 y Ejemplo 3

EJEMPLO 1

Table		
Command	Target	Value
open	http://www.google...	
assertTitle	Google	
type	q	Adictos al trabajo
clickAndWait	btnG	
assertTitle	Adictos al trabajo -...	
clickAndWait	link=Adictos al Tra...	
assertTitle	Adictos al Trabajo. ...	

EJEMPLO 3

Table		
Command	Target	Value
open	http://www.google...	
assertTitle	Google	
type	q	Autentia
keyPress	q	\13
waitForPageT...	5000	
clickAndWait	//a[@class='l']	



Selenium Core

Selenium Core



www.autentia.com

- **Selenium Core**

- Pertenece al juego de herramientas SeleniumHQ.
- También llamado (TestRunner)
- Funcionalidad:
 - Reproducir los test en diferentes navegadores.
- No Graba / Ni Edita Test → Selenium IDE
- Problema de restricciones de seguridad.
- Componente de Selenium RC
- Ejecución:
 - Independiente
 - Integrado en una aplicación web → Llamada URL

Selenium Core



– Características:

- Fácil instalación.
- Interfaz intuitiva.
- Facilidad en la ejecución de los test.
- Ejecución en varios navegadores (Muy importante).
- Referencia a objetos DOM.
- Ejecución de scripts en modo “paso a paso”.

Selenium Core



– Requerimientos : Navegador

Navegador	Funcionamiento
Firefox 2 / 3 IE 7 Safari 2/3 Opera 2/3	Reproduce test
Otros	Reproduce test (*)
IE 8b1	?

– Requerimientos : Sistema operativo

Sistema Operativo	Funcionamiento
Windows OS X Linux Solaris	Reproduce test
Otros	Reproduce test (*)

(*) Selenium Core esta escrito en Javascript y por lo tanto debería de funcionar en aquellos buscadores que soporten Javascript.
Nota: Algunas operaciones solo se pueden realizar mediante Selenium RC debido a las opciones de seguridad de los navegadores.

Selenium Core

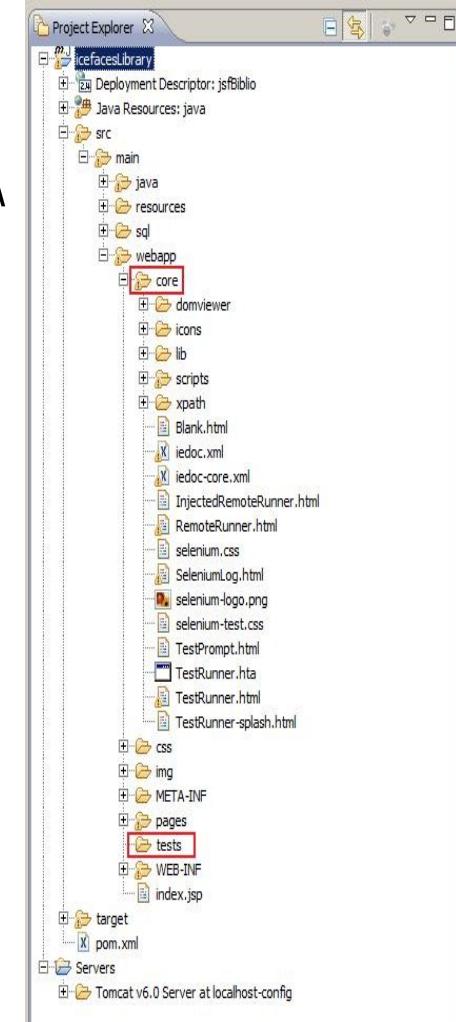


- **Instalación**

- Descargar <http://seleniumhq.org/download/>.
- Descomprimir en una carpeta → Ej: C:\selenium-coreX.X\
- Ejecutar archivo “index.html”.
 - Lanzador de Test Suite.
 - Verificar si el navegador esta soportado.
 - Pruebas unitarias / Test Case.

- **Integración en un proyecto**

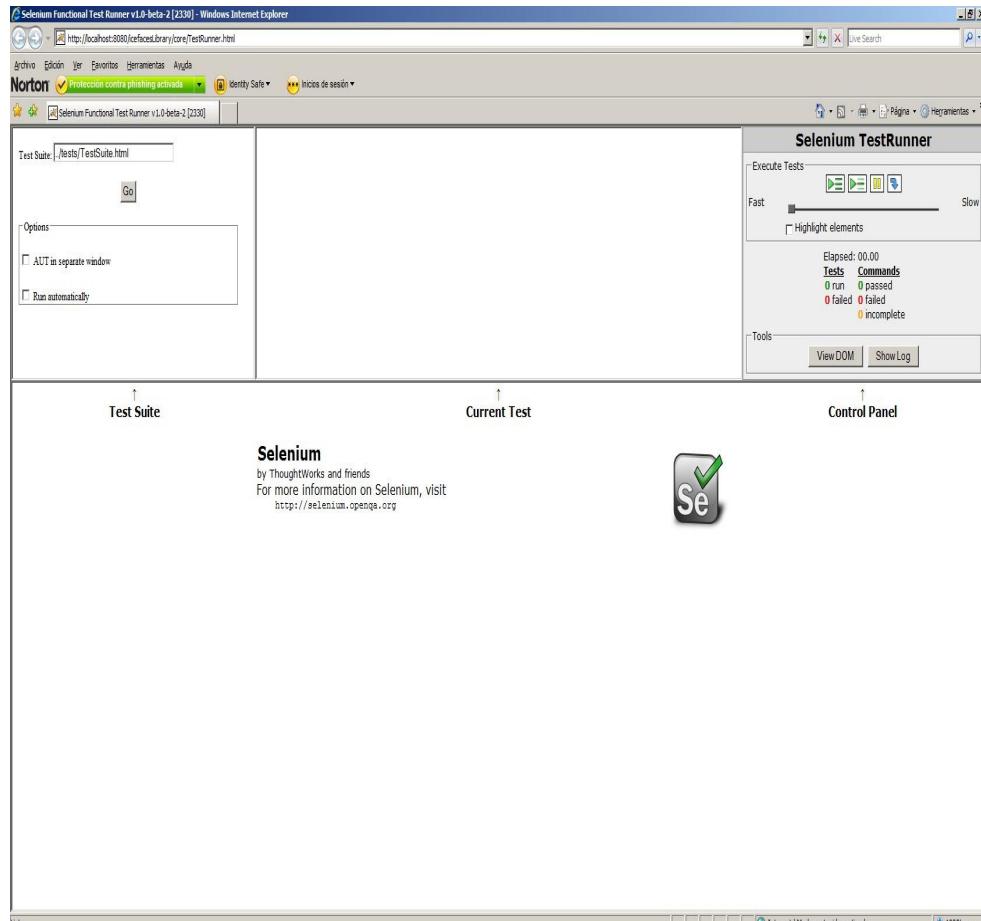
- Copiar la carpeta “core” (directorio web accesible)
 - core : Contiene el lanzador de scripts.
- Crear la carpeta “tests” al mismo nivel que core
 - tests : Contendrá los Test Suite y Test Case.
- Desplegar aplicación en servidor de aplicaciones
- <http://<hostname>:<port>/core/TestRunner.html>



Selenium Core



www.autentia.com



- Selenium TestRunner
 - Áreas específicas
 - Test Suite
 - Carga Test
 - Ventana independiente
 - » Problema Firefox
 - Ejecución automática
 - Current Test
 - Control Panel
 - Ver DOM
 - Mostrar Log

Selenium Core



- Ejemplos:

Para ello arrancar Apache Tomcat 6 y desplegar Ap. Interna

- **Ejemplo 1 :** Ejecutar Selenium Core en varios navegadores.
 - Comprobar que podemos ejecutar Selenium Core sobre el proyecto de uso interno en los navegadores: IE / Firefox y Opera
- **Ejemplo 2 :** Ejecutar Test Suite SeleniumIDE
 - Ejecutar : ../tests/TestSuite1.html
- **Ejemplo 3 :** Ejecutar Test Suit Proyecto interno
 - Ejecutar : ../tests/TestSuite2.html

Parar el Tomcat

Selenium Core



- Problema de seguridad de Javascript en los navegadores:
 - Muchas operaciones (comandos) se consideran ataques → Bloqueados o sin concesión permisos.
 - Soluciones:
 - Selenium IDE → Extensión Mozilla Firefox.
 - Selenium RC → Servidor proxy.
 - Selenium Core requiere estar instalado en el servidor donde se ejecuta la aplicación. (Ej : Google)
 - Funciona correctamente en servidores propios.



Selenium Core



- Integración
 - Recuperación parámetros GET / POST
 - Implementación propia o Selenium RC
 - Pueden ser :
 - Parámetros de configuración de las pruebas (Ej: auto,close ,save, etc.)
 - Resultados obtenidos: (nº test fallidos / pasados, etc.)
- Modo ejecución HTA
 - Fichero especial .hta (HTML) → No seguridad /Si escribir disco
 - Funciona : Windows y Internet Explorer
 - Carpeta “core”.
 - Modos
 - Ejecutable
 - Línea de comandos → Compatible con integración



Selenium Remote Control (Selenium RC)



- **Selenium Remote Control (RC)**

- Permite escribir test automatizados de aplicaciones web
 - Escritos en cualquier lenguaje → comunicación HTTP
 - Sobre cualquier sitio web (HTTP)
 - Usando un navegador con Javascript habilitado.
 - Servidor Selenium
 - Selenium Core
- Poder utilizar un “lenguaje de verdad” va a permitir:
 - Test dinámicos.
 - Gran potencia.
- Pertenece al juego de herramientas SeleniumHQ.
- Funcionalidad:
 - Permite ejecutar las pruebas desde **múltiples navegadores** y desde **múltiples plataformas**. → *Lo ideal*.
 - Script en lenguaje de programación seleccionado.

Selenium RC



– Requerimientos : Navegador

Navegador	Funcionamiento
Firefox 2/3 IE 7 Safari 2/3 Opera 2/3	Iniciar navegador y reproducir test
Otros	Possible soporte parcial(Seguridad)
IE 8b1	?

– Requerimientos : Sistema operativo

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

– Requerimientos : Lenguaje de programación

Lenguaje de programación	Funcionamiento
C# Java Perl Php Python Ruby	Soporte librería (“Driver”)
Otros	Comandos via petición HTTP(**)

(*) El servidor de Selenium Remote Control esta escrito en Java, por lo que puede ejecutarse en otros sistemas y, siempre que haya un navegador disponible.

(**)Cualquier lenguaje puede hacer una llamada HTTP pasando comandos de control remoto al servidor Selenium RC..

Selenium RC



www.autentia.com

- Se compone de 2 partes
 - Servidor
 - Contiene Selenium Core
 - Pone en marcha automáticamente, para y controla a los navegadores.
 - » Comunicación con el navegador : Uso AJAX (XmlHttpRequest)
 - Proxy HTTP para sus peticiones web.
 - No necesita correr en la misma máquina virtual (JVM) o en la misma máquina física.
 - Cliente
 - Conecta con el servidor
 - Librerías clientes para los lenguajes de programación (HTTP).
 - Se aconseja utilizar un framework de testeo : JUnit o Testng (JAVA).

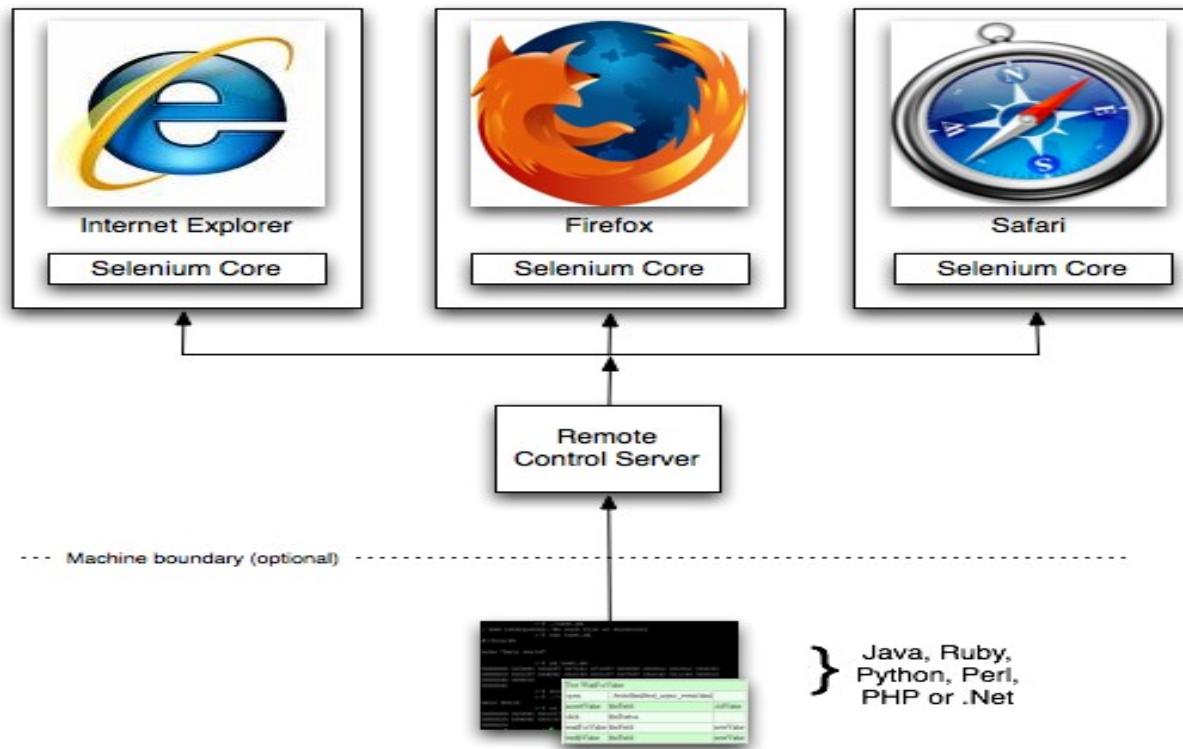
Selenium RC



www.autentia.com

Funcionamiento

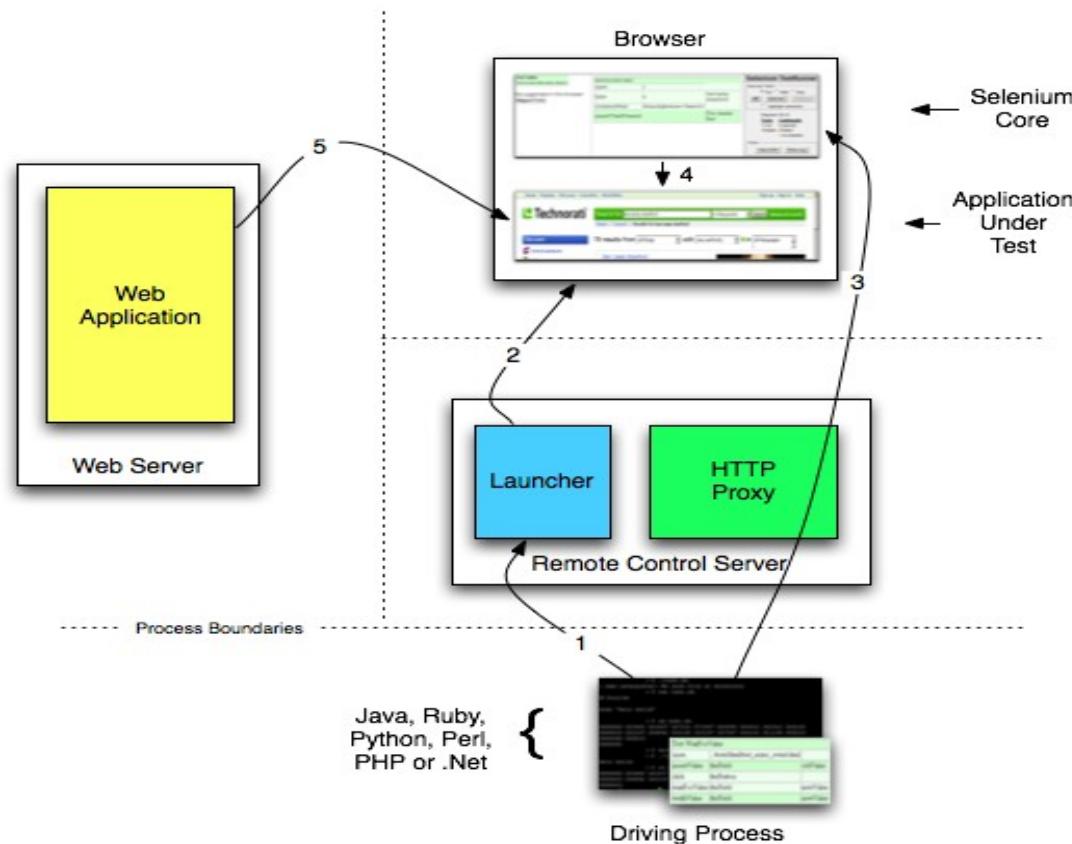
Windows, Linux, or Mac (as appropriate)...



Selenium RC



Funcionamiento Detallado

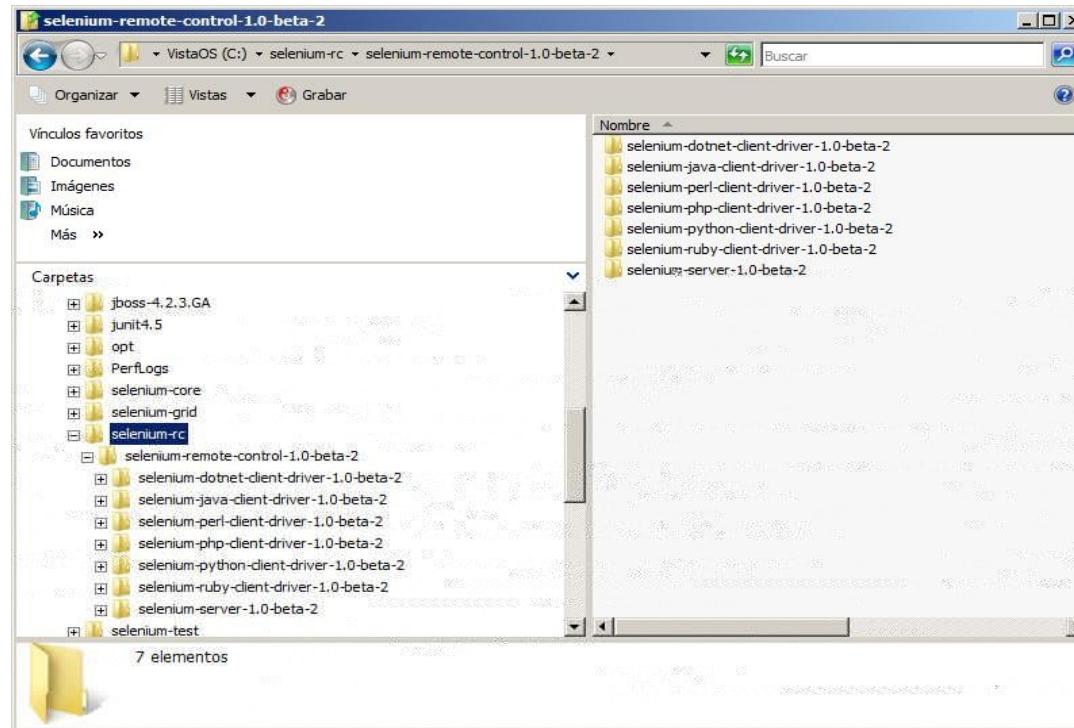


Selenium RC



www.autentia.com

- Instalación:
 - Descargar <http://seleniumhq.org/download/>.
 - Descomprimir en una carpeta → Ej: C:\selenium-rc\



Selenium RC



www.autentia.com

- Operaciones en el Servidor Selenium:

- Arrancar el servidor:

- Abrir la consola y situarse en la la carpeta que hace referencia al servidor:

C:\selenium-rc\selenium-remote-control-1.0-beta-2\selenium-server-1.0-beta-2

- Modo normal :Ejecutar la siguiente instrucción:

java -jar selenium-server.jar

- Modo interactivo :Ejecutar la siguiente instrucción:

java -jar selenium-server.jar -interactive

- Parar el servidor:

- Abrir el navegador y escribir la siguiente URL (Normal / Interac.):

http://localhost:4444/selenium-server/driver/?cmd=shutdown

- Si se ejecuta en modo interactivo:

quit

Selenium RC



www.autentia.com

- Incorporar parámetros al arranque normal:

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 nº de segundos de debería de tener de timeout.

- Incorporar parámetros al arranque interactivo:

- Se introducen en la línea de comandos

- Ejemplos :

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

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

Selenium RC



www.autentia.com

- Navegadores utilizados:
 - Hay que indicar el tipo de navegador.
 - Tipos:
 - **Normales** : *firefox, *iexplore, *safari y *custom /path/to/browser
 - **Experimentales** :
 - Permiten probar las aplicaciones en cualquier sitio.
 - Tipos:
 - » **Subir los privilegios de seguridad**: *iehta y *chrome
 - » **Modo inyección de proxy**: *piiexplore y *piifirefox

Arrancar el navegador: `java -jar selenium-server.jar -proxyInjectionMode`

Selenium RC



www.autentia.com

- Objeto DefaultSelenium en el Test Case (JAVA)
 - Permite configurar su ejecución (Modificar el constructor).

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

- **Importante:** El objeto creado esta vinculado a una URL concreta y sólo se puede utilizar con esa URL.
- Cuando se quiera comenzar:
`selenium.open("http://www.site.com/page.html");`
- Cuando se quiera finalizar:
`selenium.stop();`



- Integración con JUnit

- Descargar JUnit 4.5 de www.junit.org
- Descomprimir en una carpeta → Ej: C:\JUnit4.5\
- Comprobar que el JDK utilizado esta en la variable de entorno PATH.
- Incluir en el CLASSPATH las librerías:
 - selenium-java-client-driver.jar
 - junit-4.4.jar;
- Crear un directorio que contenga las pruebas.
Por ejemplo: <C:\selenium-test>.
- Crear los Test Cases
- Crear un Test Suite
- Generar un build de esas clases (compilación)
- Generar un ejecutable (para la clase Test Suite)

Selenium RC



www.autentia.com

- Crear los casos de prueba (Test Case)
 - Ejemplo de plantilla para la generación de un Test Case
 - Los test case de una misma test suite deberían de tener la misma URL Base
 - Los test case no de pueden ejecutar de forma individual.

```
view plain print ?
01. import junit.framework.TestCase;
02. import com.thoughtworks.selenium.*;
03.
04. public class <NOMBREPRUEBA>Test extends TestCase {
05.
06.     private Selenium browser;
07.
08.     public void setUp() {
09.         browser = new DefaultSelenium(<CONFIGURACION>);
10.         browser.start();
11.     }
12.
13.     public void test<NombrePrueba>() throws InterruptedException {
14.
15.         <CODIGO IMPORTADO JAVA DESDE SELENIUM IDE>
16.     }
17.
18.     public void tearDown() {
19.         browser.stop();
20.     }
21. }
```



- Crear el Test Suite
 - Ejemplo de plantilla para la generación de Test Suite

```
01. import junit.framework.Test;
02. import junit.framework.TestSuite;
03. import junit.textui.TestRunner;
04.
05.
06. public class <NOMBRE>TestSuite extends TestSuite {
07.
08.     public <NOMBRE>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 <NOMBRE>TestSuite("Ejemplo Test Suite Java");
18.
19.         //Establecemos los casos de prueba a ejecutar
20.         suite.addTestSuite(<NOMBRE_TEST_CASE>.class);
21.         .
22.         .
23.         .
24.         suite.addTestSuite(<NOMBRE_TEST_CASE>.class);
25.
26.         return suite;
27.     }
28.
29.
```

Selenium RC



www.autentia.com

- Ejemplo JUnit

- Abrir una consola.
- Situarse en : C:\selenium-rc\selenium-remote-control-1.0-beta-2\selenium-server-1.0-beta-2\
- Arrancar Servidor Selenium : **java -jar selenium-server.jar -interactive**
- Situarse en c:\selenium-test\Ejemplo RC JUnit\
- Mostrar el código
- Ejecutar build
- Ejecutar test.
- Parar el servidor interactivo

Selenium RC



www.autentia.com

- Ejecutar un Test Suite
 - Abrir la consola y desde la ubicación del servidor Selenium ejecutar la siguiente instrucción:

```
java -jar selenium-server.jar -htmlsuite <browser> <url> <path to testsuite> <where to store results>
```

Por ejemplo:

```
java -jar selenium-server.jar -multiwindow -htmlSuite "*iexplore" "http://www.autentia.com"  
"C:\selenium-test\Ejemplo RC Suite\Testsuite.html" "C:\selenium-test\Ejemplo RC Suite\results.html"
```

Selenium RC



www.autentia.com

- Integración Selenium / Maven 2 / Cargo / Tomcat
 - POM : Integrar Selenium con Maven 2
 - Definimos como propiedades comunes al proyecto

```
01. <project>
02. .
03. .
04. .
05. <!-- Propiedades Comunes -->
06.   <properties>
07.     <selenium-version>1.0-SNAPSHOT</selenium-version>
08.     <selenium.port>4444</selenium.port>
09.     <selenium.background>true</selenium.background>
10.   </properties>
11. .
12. .
13. .
14. </project>
```

Selenium RC



www.autentia.com

- Integramos el plugin : selenium-maven-plugin

```
01. <plugins>
02. .
03. .
04. .
05.     <plugin>
06.         <groupId>org.codehaus.mojo</groupId>
07.         <artifactId>selenium-maven-plugin</artifactId>
08.         <executions>
09.             <execution>
10.                 <phase>pre-integration-test</phase>
11.                 <goals>
12.                     <goal>start-server</goal>
13.                 </goals>
14.                 <configuration>
15.                     <background>${selenium.background}</background>
16.                 </configuration>
17.             </execution>
18.         </executions>
19.     </plugin>
20. .
21. .
22. .
23. </plugins>
```

- *Este plugin permite : Arrancar / Parar / Ejecutar el Servidor RC.*
- *En nuestro caso sólo vamos a definir que lo arranque.*



- Integramos el plugin : maven-surefire-plugin

```
01. <plugins>
02. .
03. .
04. .
05.     <!-- Colocar a continuacion del anterior -->
06.     <plugin>
07.         <groupId>org.apache.maven.plugins</groupId>
08.         <artifactId>maven-surefire-plugin</artifactId>
09.         <configuration>
10.             <!-- Permite saltar los test unitarios para ejecutar los de integracion -->
11.             <skip>false</skip>
12.         </configuration>
13.         <executions>
14.             <!-- Filtramos todas las pruebas que terminen por *It.class -->
15.             <execution>
16.                 <id>it-test</id>
17.                 <phase>integration-test</phase>
18.                 <goals>
19.                     <goal>test</goal>
20.                 </goals>
21.                 <configuration>
22.                     <skip>false</skip>
23.                     <includes>
24.                         <include>**/*It.class</include>
25.                     </includes>
26.                 </configuration>
27.             </execution>
28.         </executions>
29.     </plugin>
30. .
31. .
32. </plugins>
```

- Plugin que ejecuta los test unitarios durante la fase de test (Junit >=3.8)
 - » Genera 2 informes en \${basedir}/target/surefire-reports
- Problema: Filtramos los test para las pruebas de integración.
 - » Ocurre para Maven 2.0 → Posible solución en Maven 2.1

Selenium RC



www.autentia.com

- Integrar el repositorio de plugins para los plugins anteriores

```
01. <project>
02. .
03. .
04. .
05. <!-- Repositorio de plugin -->
06.   <pluginRepositories>
07.     <pluginRepository>
08.       <id>codehaus snapshot repository</id>
09.       <url>http://snapshots.repository.codehaus.org/</url>
10.       <releases>
11.         <enabled>true</enabled>
12.       </releases>
13.     </pluginRepository>
14.   </pluginRepositories>
15. .
16. .
17. .
18. </project>
```

- Permite bajar las librerías necesarias.

Selenium RC



- Incorporar las dependencias:
 - Selenium-server
 - Selenium-java-client-driver
 - JUnit

```
01. </dependencies>
02. .
03. .
04. .
05. <dependency>
06.   <groupId>junit</groupId>
07.   <artifactId>junit</artifactId>
08.   <version>4.5</version>
09.   <scope>test</scope>
10. </dependency>
11. <dependency>
12.   <groupId>org.openqa.selenium.client-drivers</groupId>
13.   <artifactId>selenium-java-client-driver</artifactId>
14.   <version>${selenium-version}</version>
15.   <scope>test</scope>
16. </dependency>
17. <dependency>
18.   <groupId>org.openqa.selenium.server</groupId>
19.   <artifactId>selenium-server</artifactId>
20.   <version>${selenium-version}</version>
21. </dependency>
22. .
23. .
24. .
25. </dependencies>
```

Selenium RC



www.autentia.com

- POM : Integrar Cargo con Maven 2
 - Integramos el plugin : cargo-maven2-plugin

```
01. <plugins>
02. .
03. .
04. .
05. <!-- Integramos Cargo a continuacion de Selenium -->
06. <plugin>
07.   <groupId>org.codehaus.cargo</groupId>
08.   <artifactId>cargo-maven2-plugin</artifactId>
09.   <version>1.0-beta-2</version>
10.   <executions>
11.     <execution>
12.       <id>start-container</id>
13.       <phase>pre-integration-test</phase>
14.       <goals>
15.         <goal>start</goal>
16.       </goals>
17.     </execution>
18.     <execution>
19.       <id>stop-container</id>
20.       <phase>post-integration-test</phase>
21.       <goals>
22.         <goal>stop</goal>
23.       </goals>
24.     </execution>
25.   </executions>
26.   <configuration>
27.     <background>${selenium.background}</background>
28.     <wait>false</wait> <!-- Evita que se quede esperando el contenedor -->
29. 
30.     <!-- Configuracion Container -->
31.     <container>
32.       <containerId>tomcat6x</containerId>    <!-- Especifica el contenedor -->
33.       <home>C:/apache-tomcat-6.0.18</home> <!-- Especifica la ruta del contenedor -->
34.     </container>
35. 
36.     <!-- Configuracion para usar Container -->
37.     <configuration>
38.       <type>standalone</type>
39.     </configuration>
40. 
41.   </configuration>
42. </plugin>
43. .
44. .
45. .
46. </plugins>
```



Ejecutar ejemplo de proyecto interno :

- Arrancar Eclipse : *workspaceCurso*
- Mostrar test
- Ejecutar en línea de comandos
 - mvn test
 - mvn integration-test
 - mvn install



Selenium Grid

Selenium Grid



www.autentia.com

- **Selenium Grid**

- Es una extensión de Selenium Remote Control para distribuir las pruebas en múltiples plataformas e incluso a la vez.
- Puede ejecutar varios Selenium RC (1 o +).
- Pertenece al juego de herramientas SeleniumHQ.
- Funcionalidad:
 - Permite ejecutar las pruebas desde **múltiples navegadores** y desde **múltiples plataformas**. → Lo ideal.
 - **Ejecución Paralela.**

Selenium Grid



- Instalación
 - Verificar instalación de JDK 1.5 (Ejecutar : java -version)
 - Verificar instalación de Ant 1.7 (Ejecutar: ant -version)
 - Descargar Ant 1.7 desde su página.
 - Descomprimir en un directorio del sistema (por ejemplo :C:\ant1.7\)
 - Modificar las variables de entorno.
 - Instalar Selenium Grid
 - Descargar Selenium Grid desde su página
<http://selenium-grid.seleniumhq.org/download.html>
 - Descomprimir en un directorio del sistema
(por ejemplo : C:\selenium-grid\)
 - Verificar instalación de Selenium Grid
 - Abrir la consola ,situarse en la carpeta de instalación y ejecutar :
ant sanity-check

Selenium Grid



- Funcionamiento:

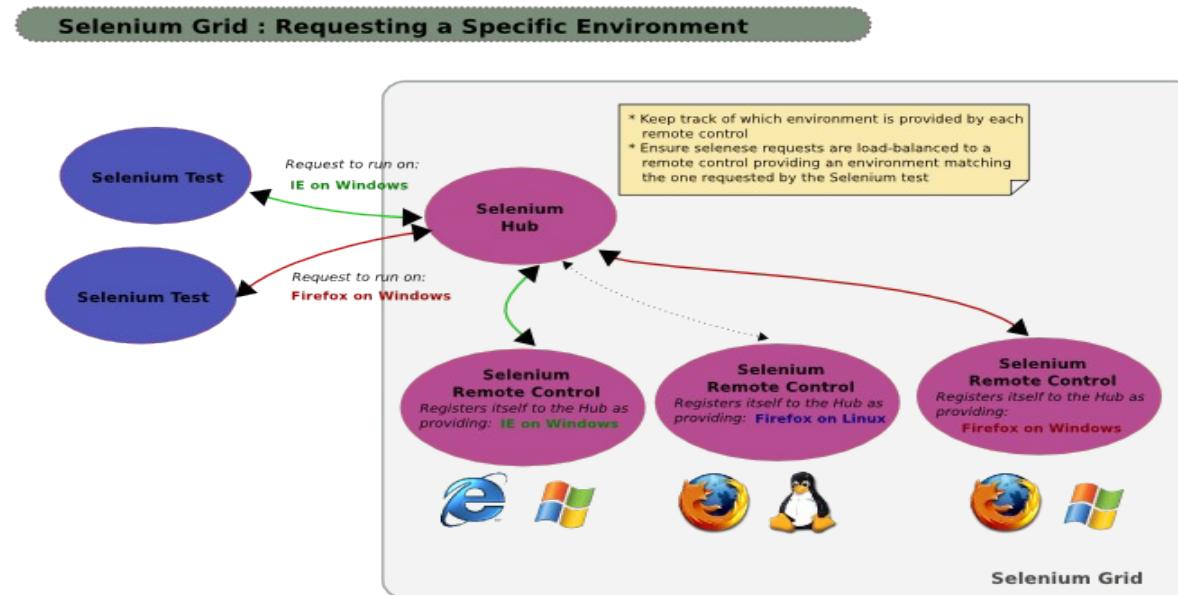
- [Recordar] : Selenium RC usa un lenguaje de programación para manejar al navegador.
 - Envía peticiones HTTP (protocolo específico Selenese).
- Funcionamiento de Selenium RC:
 - Bueno → Pocos casos de prueba
 - Malo → Muchos casos de prueba (Test Suite Compleja)
 - Inconvenientes:
 - » Selenium RC es muy lento manejando el navegador.
Puede ser el cuello de botella de tus pruebas
 - » La ejecución paralela de test en el mismo RC esta limitada
Nº máximo : 6 navegadores/RC.
Incremento de la inestabilidad : Internet Explorer
 - » Los test pueden utilizar múltiples RC pero teniendo en cuenta la limitación.
 - **Importante:** No usar Test Suite cuyos tiempos sean >30'

Selenium Grid



www.autentia.com

- Los test Selenium, la aplicación donde se ejecuta y la asociación navegador / control remoto no tienen que localizarse → HTTP.
 - Los test Selenium y su aplicación web → asociados a un proyecto concreto.
 - Ni el Control Remoto ni el navegador están asociados a una aplicación concreta.



Selenium Grid



www.autentia.com

- Selenium Hub.
 - Es un componente de Selenium Grid.
 - Interfaz externa → Mando a distancia tradicional.
 - No hay cambios en el código.
 - Protege a los test de la infraestructura (HUB o RC)
 - Facilita la vida del desarrollador.
 - En cada prueba se asignan Controladores Selenium.
 - Encamina las peticiones Selenesse desde la prueba al controlador.
 - Aprovechar ejecuciones paralelas: JUnit o Testng.
 - Ejecutar Hub
 - ant launch-hub**
 - Acceder a la consola Hub
 - http://localhost:4444/console**



Selenium Grid



www.autentia.com

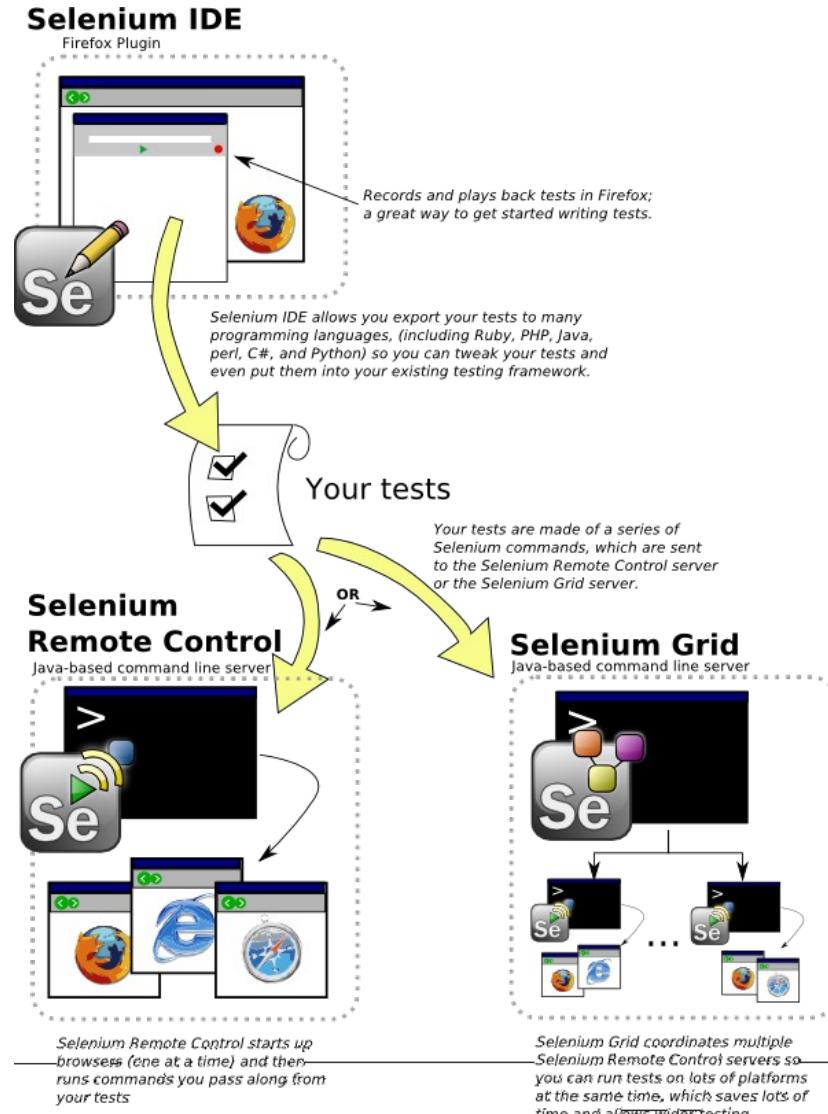
- Selenium Grid y las Pruebas de Carga
 - Selenium Grid no está diseñado para realizar pruebas de carga y rendimiento.
 - Está diseñado para realizar pruebas funcionales o de aceptación.
 - **Motivo:** Resulta muy costoso realizar la carga con un navegador real.
 - Esta carga se produciría a escala
 - Y la carga real es muy inconsistente.
 - Para realizar pruebas de carga se aconseja utilizar otras herramientas:
 - JMeter.
 - Grinder.
 - Browser Mob.





Conclusiones

Conclusiones



- SeleniumHQ es gratuito.
- Abarca todo tipo de necesidades de ejecución de pruebas.
- Permite usarse en diferentes navegadores y plataformas,
- Ejecución “ligeramente” paralela.
- Inclusión como parte de un proyecto.
- Integración con Maven.



Ruegos y preguntas

Nota



- Las imágenes referentes a las aplicaciones utilizadas han sido tomadas de la siguiente URL:
<http://seleniumhq.org/>
- El resto de imágenes son libres.