Integración de pruebas Web con Selenium usando el Framework de Pruebas Dextra (DTF)

Contents

[Pasos para ejecutar pruebas con DTF 2](#_Toc356205421)

[Supuestos 2](#_Toc356205422)

[Requisitos 2](#_Toc356205423)

[Pasos 2](#_Toc356205424)

[Creación del proyecto de pruebas 2](#_Toc356205425)

[Descripción del proyecto de pruebas 2](#_Toc356205426)

[Creación de pruebas y ejecución 4](#_Toc356205427)

[Generación de reportes 5](#_Toc356205428)

[Correr pruebas de Selenium usando archivos HTML 5](#_Toc356205429)

[Errores comunes 6](#_Toc356205430)

[1. El servidor Selenium está corriendo 6](#_Toc356205431)

[2. No puede arrancar una sesión del navegador 6](#_Toc356205432)

[3. Error de compilación 1 7](#_Toc356205433)

# Pasos para ejecutar pruebas con DTF

## Supuestos

* La aplicación a probar se encuentra en ejecución en un servidor de aplicaciones
* La aplicación es accesible desde otras IP, si no se probará la aplicación en e localhost.

## Requisitos

* Java 1.5 o superior, preferentemente el 6
* Maven 2.2.x o superior, preferentemente el 3.x.x
* Un cliente de SVN, preferentemente si es un plugin para el IDE, como Subclipse para Eclipse.
* Un IDE o editor que soporte XML. Se recomienda alguna versión de Eclipse, ya que incluye herramientas de edición de XML.
* La versión actual del DTF es 1.1.0-RELEASE.

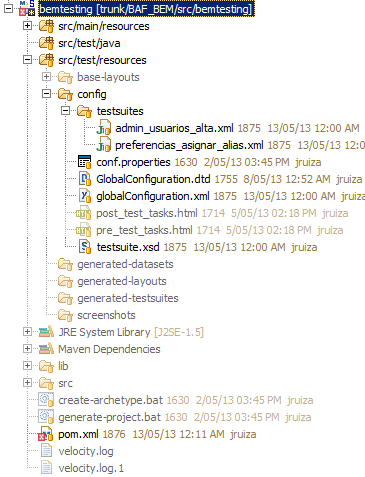
## Pasos

### Creación del proyecto de pruebas

1. Descargar el arquetipo Maven del repositorio :pserver:[user]@192.168.10.150:/opt/cvshome/TESTING/dtf-project-archetype/target/generated-sources/archetype/target/dtf-project-archetype-<versión>.jar o de alguna de las revisiones liberadas.
2. Instalar el arquetipo en el repositorio local de Maven. Esto se ejecuta fácilmente corriendo el archivo por lotes **create-archetype.bat**.
3. Copiar el archivo por lotes **generate-project.bat** a una nueva ubicación, preferentemente donde se quiera generar el nuevo proyecto de pruebas.
4. Editar el archivo **generate-project.bat** cambiando el parámetro artifactId por el nombre deseado para el proyecto de ptuebas.
5. Correr **generate-project.bat**. Esto generará un proyecto de pruebas. Este contiene carpetas donde serán generados casos y datos de pruebas, además de recursos para generar reportes.
6. Importe el nuevo proyecto a su IDE favorito como un proyecto Maven.

### Descripción del proyecto de pruebas

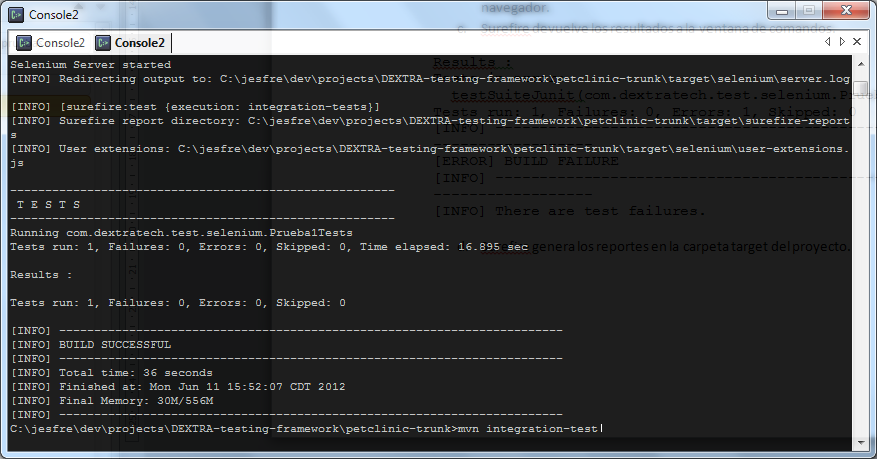
El proyecto de pruebas tiene una estructura estándar de Maven.



* **src/test/resources:** aContiene todos los recursos de configuración que podrán ser manipulados por el ingeniero de pruebas. Para la versión actual del DTF, solo será necesario manipular los archivos dentro de la carpeta **config**.
  + **config/conf.properties:** Incluye archivos de configuración para las ejecuciones automáticas. Estos archivos serán usados por el DTF solamente. No es necesaria su manipulación.
  + **config/GlobalConfiguration.dtd:** Contiene la definición de la estructura del documento de configuración global.
  + **config/globalConfiguration.xml:** Define configuraciones globales del proyecto de pruebas. Incluye ejemplos de configuración y un listado de navegadores soportados para las ejecución de pruebas.
  + **config/testsuite.xsd:** Es el esquema para representar casos de pruebas automatizadas.
  + **config/testsuites/\*.xml:** Aquí se almacenarán los archivos XML que definen las pruebas automatizadas. Se incluyen archivos de ejemplo de configuración de casos de prueba.
* **src/main/resources**: Contiene archivos de configuración de Spring que serán utilizados por el DTF.
* **src/test/java**: Carpeta de código fuente donde serán generadas las pruebas finales usando el API de Selenium para Junit4 y Remote Control.
* **pom.xml:** El archivo de configuración de Maven para el proyecto. Contiene una sección de propiedades que pueden ser modificadas por el ingeniero de pruebas para personalización de ciertas configuraciones.

### Creación de pruebas y ejecución

1. Dentro del nuevo proyecto de pruebas.
2. Generamos un nuevo archivo de casos de prueba dentro de la carpeta <PROYECTO\_PRUEBAS>/src/test/resources/config/testsuites. Se recomienda seguir los ejemplo incluidos.
3. Modificar el archivo de configuración en <PROYECTO\_PRUEBAS>/ /src/test/resources/config/globalConfiguration.xml con los datos necesarios.
4. Para correr las pruebas, ejecutar el comando **>mvn clean integration-test**. Este correrá cada prueba de acuerdo a las configuraciones establecidas en globalConfiguration.xml.
   1. De forma automática, ejecutará los comandos dtf:clean y dtf:xml-testsuites para generar los casos de prueba (HTML y Java) en base a a las configuraciones establecidas en los archivos XML de los pasos anteriores.
   2. Levanta el servidor Selenium.
   3. Abre una ventana del navegador (Google Chrome) y ejecuta las pruebas de Selenium en el navegador.
   4. Surefire devuelve los resultados a la ventana de comandos.



### Generación de reportes

1. Para generar los reportes, correr **>mvn post-site.**
2. Este comando abrirá el reporte en el navegador especificado por la propiedad de Maven: report.browser.
3. El reporte se encontrará en <PROYECTO\_PRUEBAS>/target/site/surefire-report.html.

# Correr pruebas de Selenium usando archivos HTML

Dadas las configuraciones del primer documento. Se siguen los siguientes pasos:

1. Agregar la siguiente ejecución al pluginselenium-maven-plugin

<properties>

…

<selenium.browser>\*firefox</selenium.browser>

<selenium.suite>src/test/resources/generated-testsuites/testSuite1.html</selenium.suite>

<selenium.startURL>http://google.com.mx</selenium.startURL>

</properties>

<plugin>

<groupId>org.codehaus.mojo</groupId>

<artifactId>selenium-maven-plugin</artifactId>

<version>2.3</version>

**<configuration>**

**<browser>${selenium.browser}</browser>**

**<suite>${selenium.suite}</suite>**

**<startURL>${selenium.startURL}</startURL>**

**</configuration>**

<executions>

<execution>

<id>start</id>

<phase>pre-integration-test</phase>

<goals>

<goal>start-server</goal>

</goals>

<configuration>

**<skip>true</skip>**

<background>true</background>

<logOutput>true</logOutput>

<multiWindow>true</multiWindow>

</configuration>

</execution>

<execution>

<id>stop</id>

<phase>post-integration-test</phase>

<goals>

<goal>stop-server</goal>

</goals>

<configuration>

<skip>true</skip>

</configuration>

</execution>

**<execution>**

**<id>test-html</id>**

**<phase>integration-test</phase>**

**<goals>**

**<goal>selenese</goal>**

**</goals>**

**<configuration>**

**<skip>false</skip>**

**</configuration>**

**</execution>**

</executions>

</plugin>

**Esta nueva ejecución espera encontrar el archivo de suite de pruebas definido en la propiedad <selenium.suite>.**

1. Correr**>mvn integration-test** o **>mvn selenium:selenese**. El primero causa el problema mencionado en la sección de Errores comunes.

# Errores comunes

## 1. El servidor Selenium está corriendo

**Causa**

Este error ocurrió porque se estaba ejecutando la tarea definida en la ejecución start-server porque se está ejecutando directamente la fase **integration-test**. Para evitar que la tarea inicie el servidor Selenium, basta con agregar la configuración **<skip>true</skip>**a la ejecución.

**Solución**

Entrar a la dirección: <http://localhost:4444/selenium-server/driver/?cmd=shutDownSeleniumServer>

La respuesta debe ser **OKOK**, en caso de que el servidor realmente no esté levantado entonces marcará que no se puede conectar.

## 2. No puede arrancar una sesión del navegador

Cuando intenta ejecutar las pruebas aparece el siguiente error:

Could not start Selenium session: Failed to start new browser session: java.lang.RuntimeException: Firefox 3 could not be found in the path!

Please add the directory containing ''firefox.exe'' to your PATH environment

variable, or explicitly specify a path to Firefox 3 like this:

\*firefox3c:\blah\firefox.exe

**Causa**

Esto es debido a que no el driver de Selenium no conoce la ruta del .exe del navegador en la maquina local. Por lo tanto no puede ejecutar el navegador.

**Solución**

En la configuración del navegador a usar (globalConfiguration.xml), en blowser se deberá poner la ruta del navegador. Por ejemplo, para Chrome en Windows:

\*googlechrome C:\\Program Files (x86)\\Google\\Chrome\\Application\\chrome.exe

## 3. Error de compilación 1

Un error de compilación al momento de intentar correr el comando **integration-test** puede ser el siguiente:

Compilation failure: Compilation failure:

[ERROR]ruta\a\Scenario1.java:[47,29] unclosed string literal

**Causa**

Puede ser ocasionado por que alguna frase puede tener saltos de línea en el HTML, por ejemplo en un texto esperado en un alert, provocando que el archivo Java generado tenga ese salto de línea y no compile correctamente.

**Solución**

Asegurarse al generar la plantilla HTML para la prueba que no existen saltos de línea.