DTF

Manual de Usuario

Contenido

[I. Introducción 3](#_Toc393145995)

[Definición de términos 3](#_Toc393145996)

[II. Instalación y configuración 4](#_Toc393145997)

[III. Estructura de carpetas 6](#_Toc393145998)

[IV. Script de Pruebas 9](#_Toc393145999)

[Comandos 9](#_Toc393146000)

[Validaciones 12](#_Toc393146001)

[Funciones de validación 12](#_Toc393146002)

[Campos 13](#_Toc393146003)

[Combinación de validaciones 14](#_Toc393146004)

[Casos de prueba 15](#_Toc393146005)

[Componentes 16](#_Toc393146006)

[Suite de pruebas 16](#_Toc393146007)

[V. Ejecución de pruebas 17](#_Toc393146008)

[Configuraciones globales (settings.xml) 17](#_Toc393146009)

[Ejecución de las pruebas 18](#_Toc393146010)

[Fases de DTF 19](#_Toc393146011)

[Configuraciones avanzadas (advanced-settings.xml) 19](#_Toc393146012)

[Extensiones de usuario 20](#_Toc393146013)

[Restauración de datos 21](#_Toc393146014)

[VI. Reportes 22](#_Toc393146015)

[Generación de reportes 22](#_Toc393146016)

# Introducción

El presente manual está enfocado a usuarios con conocimientos técnicos en pruebas y programación, los tales pueden ser ingenieros de pruebas o programadores, ya que el framework DTF se enfoca en facilitar el trabajo que estos realizan. El contenido se enfoca en el uso de la versión 2 del framework disponible en <https://github.com/jesfre/dtf/tree/dtf2>.

Una característica del framework DTF se encuentra en permitir el amplio uso de los comandos de Selenium, por lo tanto, mientras más comandos conozca y sepa usar el ingeniero de pruebas o programador más capacidades podrá otorgar al framework.

## Definición de términos

*Proyecto de pruebas*: Se refiere a la carpeta descomprimida con las configuraciones, estructuras de directorios y scripts de pruebas que se van a ejecutar usando DTF.

*Repositorio local de Maven*: Es un cache de dependencias remotas descargadas a una copia en el equipo local. Por defecto el repositorio local se encuentra en ${user.home}/.m2/repository/. Para mayor información vaya a <http://maven.apache.org/guides/introduction/introduction-to-repositories.html>.

*Casos de prueba por defecto*: En un script de pruebas, es el conjunto de casos de prueba que serán generados a partir de las combinaciones de cada validación de un campo con el grupo de valores válidos del resto de los campos.

# Instalación y configuración

DTF ahora es más sencillo de instalar y configurar. Se presenta en forma de un archivo comprimido. Para poner a punto un proyecto de pruebas con DTF se deben seguir los siguientes pasos:

* Instalar Java

1. Descargar del sitio oficial <http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html>
2. Instalar el JDK. La guía para Windows se encuentra en <http://docs.oracle.com/javase/7/docs/webnotes/install/windows/jdk-installation-windows.html>.
3. Recuerde agregar el JAVA\_HOME al path. <http://jesfre.blogspot.mx/2010/03/comendando-con-java-3-la-variable.html>.

* Instalar Maven

1. Descargar del sitio oficial en <http://maven.apache.org/download.cgi>.
2. Seguir la guía de instalación disponible en el mismo sitio de descarga, en la sección Installation Instructions.

* Crear un proyecto de pruebas

1. Descargar la plantilla de proyectos DTF disponible en <https://github.com/jesfre/dtf/blob/dtf2/dtf-project-2.0.0.zip>.
2. Descomprimir la plantilla en una ubicación conocida. La carpeta descomprimida será el proyecto de pruebas.
3. Opcional. Desde una ventana de comandos navegue hasta el directorio donde descomprimió dtf-project-2.0.0.zip y ejecute lo siguiente para comprobar las configuraciones y descargar un conjunto inicial de dependencias al repositorio local de Maven.

>mvn clean dtf:clean

Con estos comandos, Maven se encargará de descargar las dependencias necesarias y limpiar el proyecto de archivos generados previamente. La Ilustración 1 muestra la salida en pantalla de la ejecución del comando anterior. Asegúrese que el resultado de la ejecución sea BUILD SUCCESS.

NOTA: En la sección [Fases de DTF](#_Fases_de_DTF2) se detallan los comandos más útiles del framework.

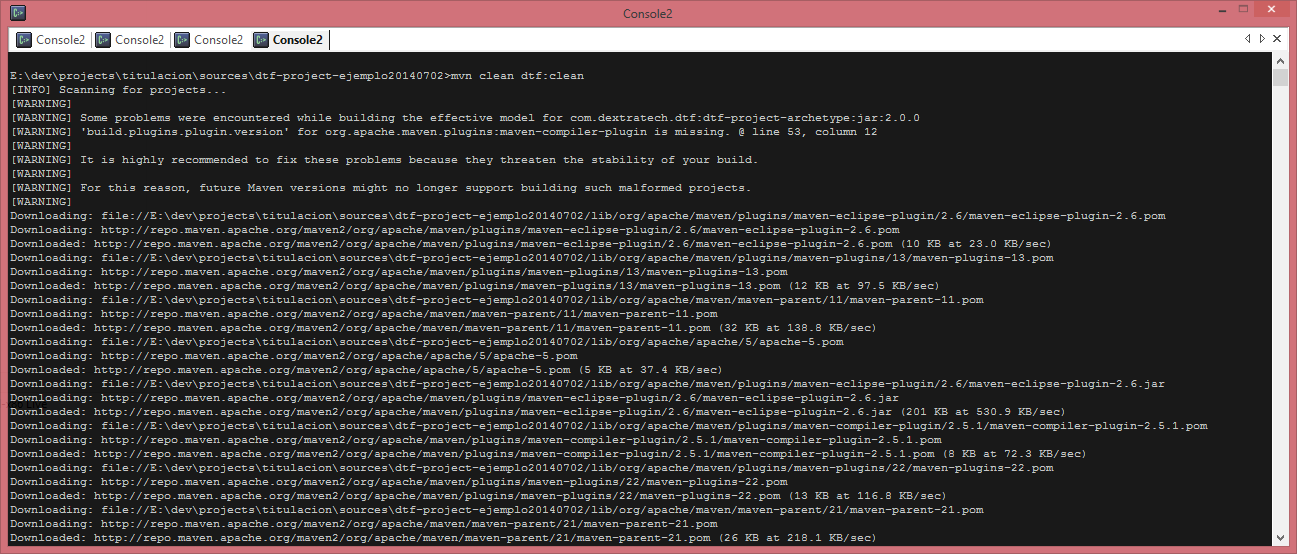
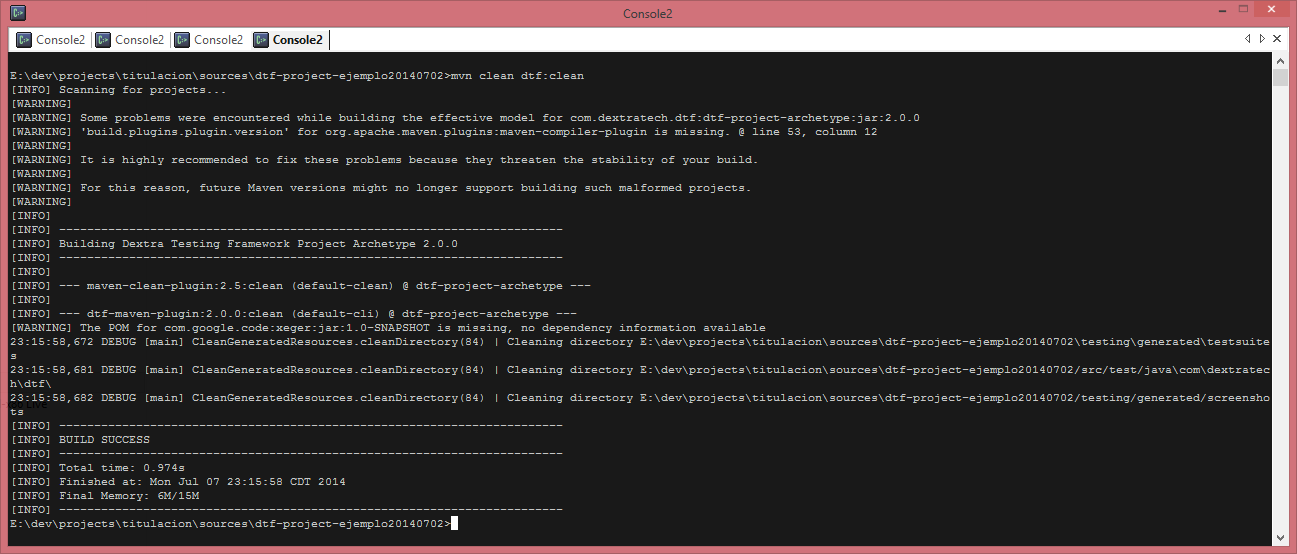


Ilustración Ejecución inicial de comando Maven

Con estos sencillos pasos tendremos la configuración necesaria para correr un proyecto de pruebas.

# Estructura de carpetas

El proyecto de pruebas está conformado por una estructura de carpetas bien definida y archivos de configuración. En la Ilustración 2 se muestran las carpetas principales de DTF.

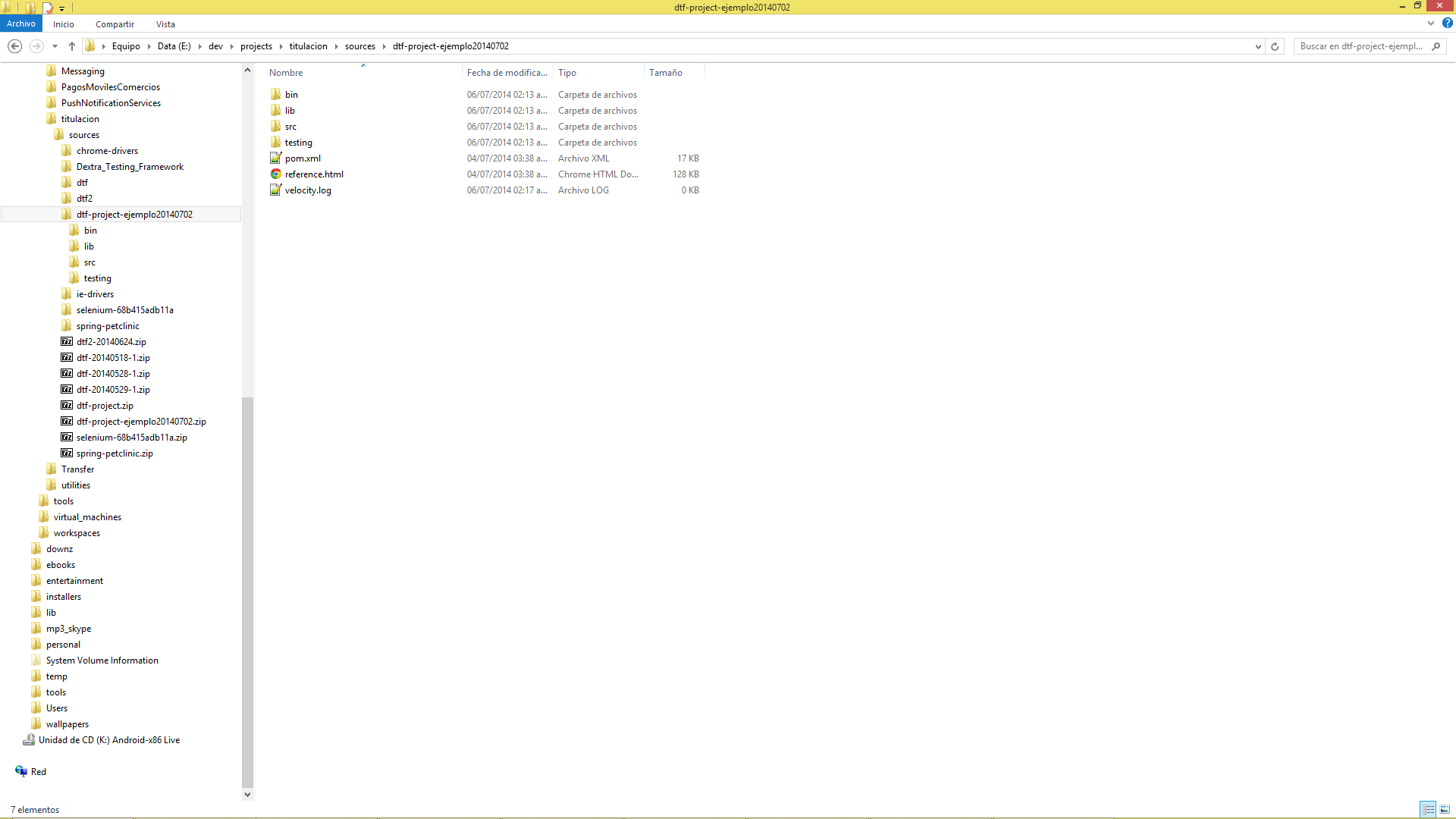


Ilustración Estructura de carpetas de DTF2

La jerarquía de los archivos y carpetas se describe como sigue:

* bin: En esta carpeta se encuentran los archivos ejecutables para correr las pruebas de manera simple. Sin embargo, DTF cuenta con un conjunto de comandos que pueden ser utilizados tanto de forma independiente, como en grupo que otorgan mayor capacidad y versatilidad al framework. Contenido de la carpeta:
* run-tests.bat
* setEnv.bat
* lib: Contiene un conjunto de librerías que serán instaladas en el repositorio local de Maven. Estas librerías no están disponibles en un repositorio en Internet.
* src: Es la carpeta correspondiente a la estructura estándar de Maven donde se alojan los archivos fuente de un proyecto de tipo Maven. En DTF también alojan archivos fuente.

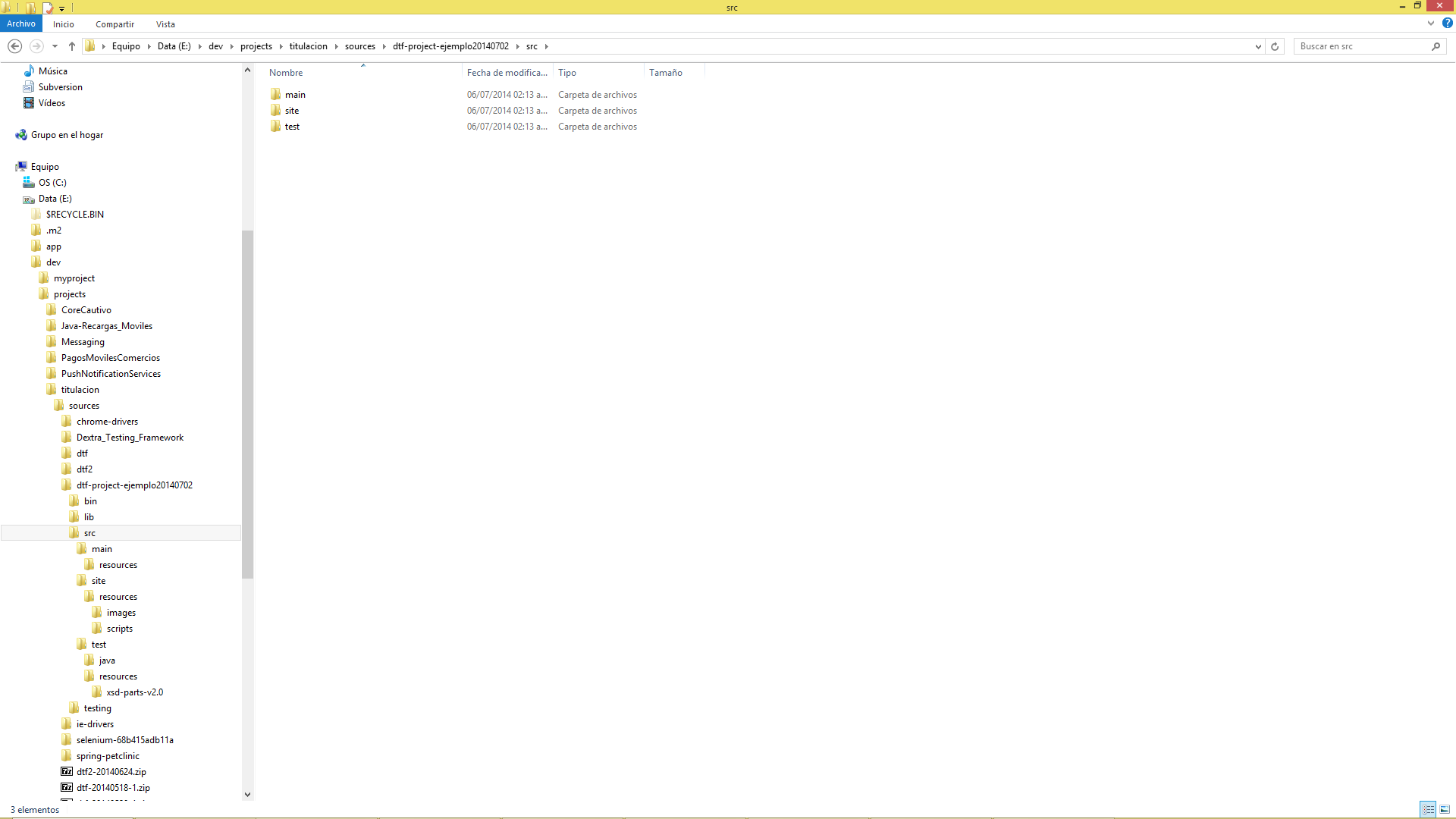


Ilustración Contenido de carpeta “src”

* + main/resources: Contiene archivos de configuración necesarios para el correcto funcionamiento del framework. Este directorio y su contenido no deben ser alterados.

- applicationContext.xml

- jdbc.properties

- log4j.xml

- script-onload-alerts.js

* + site/resources: En esta carpeta se encuentran recursos para la generación de un sitio de reportes de las pruebas realizadas. Estos recursos son imágenes y archivos Javascript. Su estructura y contenido no deben ser alterados.

- images

- scripts

* + src/test/java: En este directorio serán generados automáticamente un conjunto de archivos Java que representarán los archivos de prueba ejecutables usando JUnit/WebDriver. Este directorio y su contenido no deberán ser manipulados manualmente, ya que serán regenerados en cada ejecución de pruebas.
  + src/test/resources: Aquí se almacenan los esquemas para verificar el contenido de los scripts de pruebas.

- configuration.xsd: Esquema para escribir el archivo de configuración global *settings.xml*.

- testsuite-v2.0.xsd: Este archivo corresponde al esquema para validar los scripts de pruebas.

- testsuite.xsd: Es el esquema de validación de scripts de pruebas escritos en la versión 1.x del framework.

- xsd-parts-v2.0: Esta carpeta contiene secciones del esquema testsuite-v2.0.xsd.

* testing: Es la carpeta principal de un proyecto DTF. Aquí se almacenarán los scripts de pruebas y resultados de la ejecución de los mismos.
  + conf: Conjunta las configuraciones globales para la ejecución de las pruebas. Contiene el archivo de configuraciones avanzadas del framework y las configuraciones para la ejecución de las suites de pruebas; así como extensiones de usuario y selección de base de datos que será usado en las pruebas.
  + generated: Aquí se almacenan los archivos de pruebas generados automáticamente como resultado de la ejecución de los casos de prueba. Los archivos generados se organizan respectivamente en carpetas para respaldos de bases de datos, capturas de pantalla, casos de prueba HTML y reportes de resultados.

- dbsnapshots

- reports

- screenshots

- testsuites

* + logs: Almacén de las bitácoras de ejecución de las pruebas.

- plugin.log: Bitácora de ejecución del framework.

- testsuites.log: Bitácora de ejecución de los casos de prueba.

* + testsuites: La carpeta donde se deberán almacenar y organizar los scripts de casos de prueba.
* pom.xml: Archivo principal de Maven. Contiene las tareas y configuración de plugins de Maven necesarios para el framework. No será necesario manipular este archivo.
* reference.html: Una referencia de los comandos de Selenium que podrán ser utilizados en los script de pruebas.

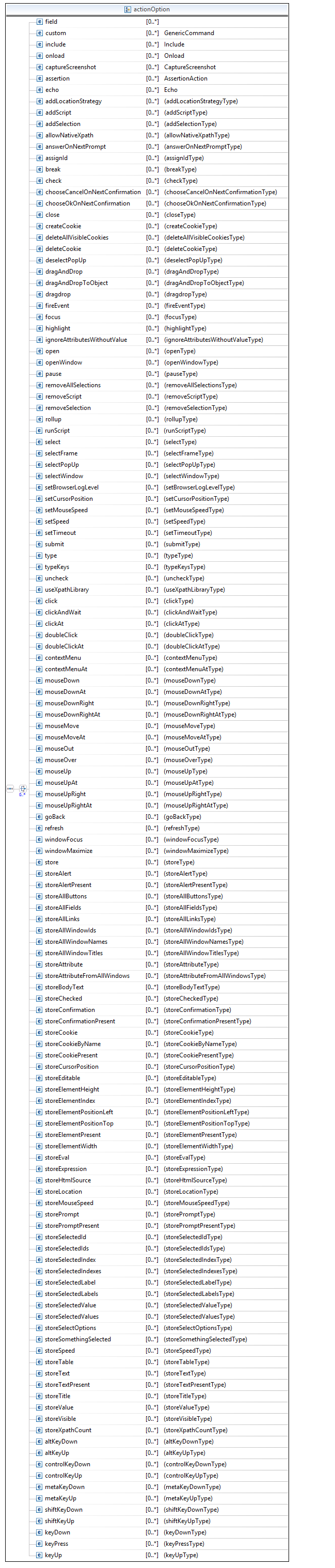
# Script de Pruebas

Las pruebas en DTF se componen por suites, casos de pruebas, funciones de validación y componentes. Cada uno de estos se deberá guardar en las estructuras de la carpeta *testing*. Se recomienda utilizar Eclipse o algún IDE que permita realizar validaciones de XML usando XSD, así como autocompletado para mayor facilidad.

A continuación se describirán los elementos que componen los scripts de pruebas y la forma de escribir y configurar cada uno de ellos. Para cada uno de estos elementos se presentará una tabla con los atributos y elementos principales; sin embargo, se puede consultar una descripción más detallada y profunda en la documentación disponible en línea <https://cdn.rawgit.com/jesfre/dtf/dtf2/dtf-doc/testsuite-v2.0.xsd.reference.html>.

## Comandos

Un caso de prueba puede contener uno o más comandos. Un comando es una acción que deberá ser llevada a cabo por el navegador o una acción específica a realizar durante una prueba. Estos comandos están descritos como *<comandos>* en la sección de [Casos de prueba](#_Testcases). Cada comando tendrá sus propios atributos requeridos y opcionales. La Ilustración 4 muestra la lista de comandos disponibles; pero una descripción más detallada de cada uno se puede encontrar en el archivo *<dtf\_project>/reference.html* o en la documentación en línea <https://cdn.rawgit.com/jesfre/dtf/dtf2/dtf-doc/reference.html>.



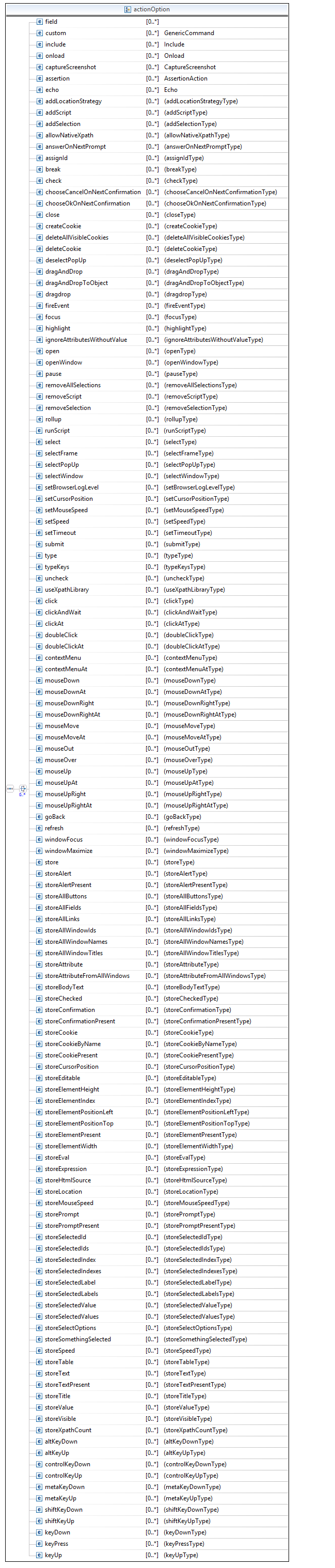


Ilustración Comandos disponibles

## Validaciones

Las validaciones son sub-acciones que resultan en un nuevo caso de prueba, en conjunción con los valores válidos de los demás campos. La finalidad de la validación es, precisamente, validar un campo. Las validaciones aplicables a un campo pueden ser definidas como valores constantes y funciones que pueden generar valores numéricos, alfanuméricos, caracteres especiales, fechas, etc.

Por ejemplo, para la siguiente definición de caso de prueba se generarán siete casos de prueba distintos; uno para el conjunto de valores válidos de los campos identificados por los elementos *validValue* y *validValueDef*, y uno por cada elemento *validate*, *validateFunction* y *validateFunctionRef*.

<testcase name=*"newOwner"* url=*"/owners/new"*>

<field name=*"firstname"* locator=*"id=firstname"*>

<validValueDef><literal>ok</literal></validValueDef>

<validate name=*"fn1"*>test 1</validate>

</field>

<field name=*"city"* locator=*"id=city"*>

<validValueDef><literal>cacahuatepec</literal></validValueDef>

<validateFunction type=*"string"* id=*"length8"*>

<length>8</length>

</validateFunction>

</field>

<field name=*"telephone"* locator=*"id=telephone"*>

<validValue>1234567890</validValue>

<validateFunction type=*"string"* id=*"length2"*>

<length>2</length>

</validateFunction>

<validate name=*"alpha"*>numero equivocado</validate>

<validateFunction type=*"integer"* id=*"min2"*>

<minValue>2</minValue>

<maxValue>9999999999</maxValue>

</validateFunction>

</field>

<field name=*"address"* locator=*"id=address"*>

<validValue>callecita</validValue>

<validate name=*"ultimateAddress"*>por alli</validate>

</field>

</testcase>

Ilustración Ejemplo de validaciones

## Funciones de validación

Una función de validación permite generar un valor aleatorio de una forma bien definida a través del elemento *validateFunction* en un campo. El elemento *validateFunction* es mostrado en la Ilustración 6.

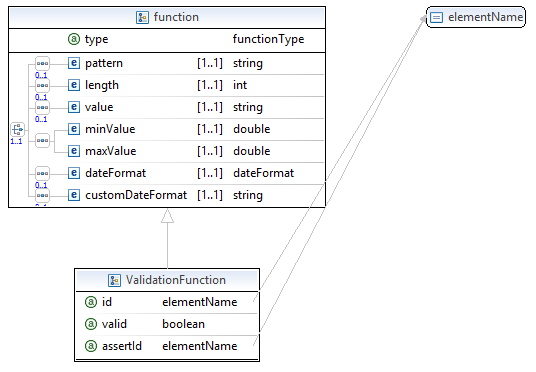


Ilustración Elementos de una función de validación

|  |  |  |
| --- | --- | --- |
| Atributo/Elemento | Req. | Descripción |
| type | Si | Determina el tipo de dato que generará la función. |
| id | No | Un identificador único para la función. Este servirá para poder referenciar a la función desde distintos campos. |
| valid | No | True para especificar que la función generará un valor válido. |
| assertId | No | El identificador de una aserción en caso de requerir validar el resultado del caso de prueba para el valor generado. |
| pattern | No | Especificará un patrón para la generación del valor. |
| length | No | Especificará la longitud de caracteres a generar. |
| value | No | Un valor constante. |
| minValue/maxValue | No | Especificarán el rango en el cual se generará el valor. |
| dateFormat | No | Especifica un conjunto de formatos de fecha pre-establecidos. Genera un valor de fecha. |
| customDateFormat | No | Especifica un formato de fecha personalizado. Genera un valor de fecha. |

## Campos

Un campo es una forma especializada del comando *<type />*. Es utilizado para definir y escribir valores a un campo, y permitir de esta forma realizar validaciones al mismo. En la Ilustración 7 se muestran los atributos y elementos que conforman un campo.

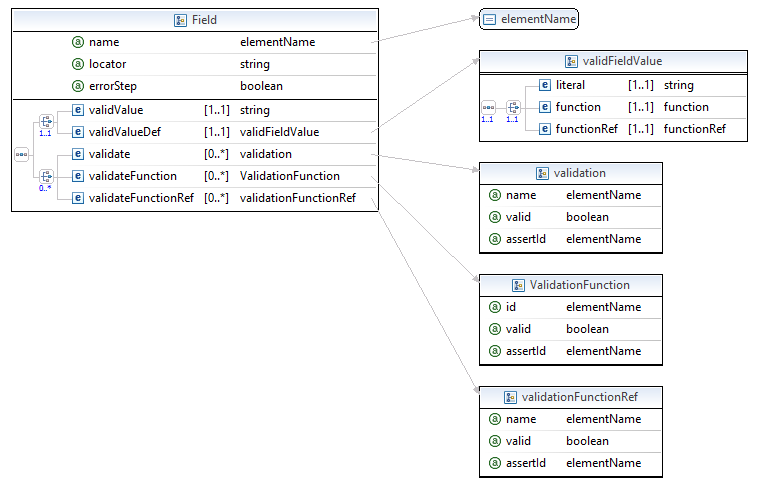


Ilustración Elementos de Field

|  |  |  |
| --- | --- | --- |
| Atributo/Elemento | Req. | Descripción |
| name | Si | Especifica el nombre del campo. |
| locator | Si | La forma en la que el campo será localizado en la aplicación Web. |
| errorStep | No | True para especificar que esta acción provocará un error. |
| validValue | Si | Una cadena de texto que representa un valor válido para el campo. |
| validValueDef | Si | Permite definir una literal, una función o referencia a una, para generar un valor válido para el campo. |
| validate | No | Permite definir una validación para el campo usando un valor constante. |
| validateFunction | No | Permite definir una validación para el campo a través de un valor generado por una función. |
| validateFunctionRef | No | Permite definir una validación para el campo a través de un valor generado por una función referenciada. |

## Combinación de validaciones

Este elemento permite agrupar un conjunto de validaciones de campos para conformar un nuevo caso de prueba con su propia aserción. Se pueden generar tantas combinaciones como se requieran. La Ilustración 8 presenta los elementos que conforman las combinaciones.

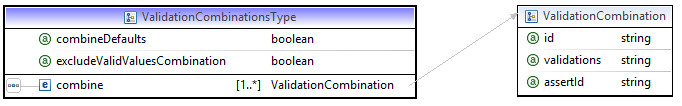


Ilustración Elemento de combinación de validaciones

|  |  |  |
| --- | --- | --- |
| Atributo/Elemento | Req. | Descripción |
| combineDefaults | No | False para evitar generar los casos de prueba por defecto. |
| excludeValidValuesCombination | No | True para evitar generar el caso de prueba con los valores válidos de los campos. |
| combine | Si | Uno o más elementos de combinación de validaciones. |
| id | Si | Identificador de la combinación. |
| validations | Si | Lista de identificadores de validaciones separados por comas. |
| assertId | No | Identificador de la aserción para la combinación. |

Tomando el caso de prueba del ejemplo en la sección [Validaciones](#_Validaciones) se podría tener un conjunto de validaciones como sigue:

<combineValidations combineDefaults=*"true"* excludeValidValuesCombination=*"false"*>

<combine id=*"combinationBBB"* validations=*"min2,fn1"*/>

<combine id=*"combinationAAA"* validations=*"fn1,alpha"* /> <combine validations=*"ultimateAddress,length2"* id=*"combinationCCC"*/>

</combineValidations>

Ilustración Ejemplo de combinación de validaciones

## Casos de prueba

Un caso de prueba representa el flujo de ejecución de una prueba. La Ilustración 10 muestra los elementos que componen un caso de prueba. Para una referencia completa de este elemento, refiérase a la documentación disponible en línea <https://cdn.rawgit.com/jesfre/dtf/dtf2/dtf-doc/testsuite-v2.0.xsd.reference.html>.

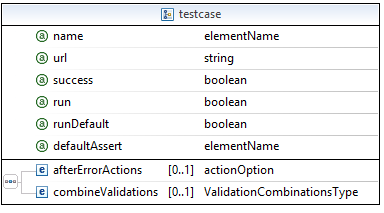


Ilustración Elementos de un caso de prueba

|  |  |  |
| --- | --- | --- |
| Atributo/Elemento | Req. | Descripción |
| name | Si | Especifica el nombre del caso de prueba. |
| url | Si | La URL desde donde comenzará el flujo de la prueba. |
| success | No | Especifica si la prueba deberá resultar satisfactoria. |
| run | No | Especifica si este caso de prueba deberá ser ejecutado. |
| runDefault | No | Determina si se debe ejecutar el caso de prueba por defecto. El correspondiente a los valores válidos de los campos. |
| defaultAssert | No | El identificador de la aserción que deberá validar el éxito o falla del caso de prueba por defecto. |
| afterErrorActions | No | Conjunto de pasos a ejecutar cuando suceda un error como resultado de un caso de prueba fallido. |
| combineValidations | No | Grupo de validaciones que conformarán nuevos casos de prueba. |
| <comandos> | No | Todos los comandos que deberán ser ejecutados en el caso de prueba. Para mayor información de todos los comandos disponibles para los casos de prueba refiérase a la documentación en línea. |

## Componentes

Un componente es una pieza de caso de prueba reutilizable. Agrupa un conjunto de acciones que pueden ser de utilidad a varios casos de prueba, evitando así repetir las mismas acciones en cada uno. Puede contener cualquiera de los comandos presentes en la sección [Comandos](#_Comandos). Pueden existir componentes en dos niveles distintos:

1. A nivel de suite de pruebas. Los componentes existen dentro de la suite de pruebas y son compartidos por los distintos casos de pruebas que componen la suite.
2. En el archivo *components.xml*. Los componentes que residen en este archivo son compartidos por cualquier caso de prueba en cualquier suite.

<component name=*"login"*>

<field name=*"usuario"* locatorType=*"id"* locatorValue=*"j\_username"* validValue=*"admin"* />

<action actionType=*"click"* locatorType=*"id"* locatorValue=*"entrar1"* />

<assert type=*"waitForTextPresent"* param1=*"Token"* />

<field name=*"pass"* locatorType=*"id"* locatorValue=*"j\_password"* validValue=*"admin"* />

<field name=*"token"* locatorType=*"id"* locatorValue=*"j\_token"* validValue=*"123456"* />

<action actionType=*"clickAndWait"* locatorType=*"id"* locatorValue=*"entrar2"* />

<assert type=*"assertTextPresent"* param1=*"BIENVENIDO:"* />

</component>

Ilustración Ejemplo de componente de login de usuario

## Suite de pruebas

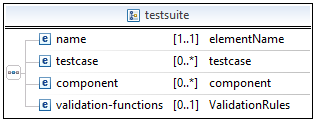


Ilustración Elementos de una suite de pruebas

|  |  |  |
| --- | --- | --- |
| Atributo/Elemento | Req. | Descripción |
| name | Si | Especifica el nombre de la suite. |
| testcase | No | Cero o más casos de prueba. |
| component | No | Cero o más componentes disponibles a los casos de prueba dentro de la suite. |
| validation-functions | No | Cero o más funciones de validación disponibles a los casos de prueba dentro de la suite. |

# Ejecución de pruebas

Habiendo realizado los pasos anteriores, ya se contará con la configuración de ambiente necesaria para ejecutar los casos de prueba. Pero la ejecución de los casos de prueba requiere que se realicen algunas configuraciones en el proyecto de pruebas.

## Configuraciones globales (settings.xml)

Cuando ya se han creado los scripts de pruebas será necesario agregarlos al archivo de configuración *settings.xml* para que puedan ser encontrados y referenciados por DTF. Este archivo de configuración será modificado con poca frecuencia. Las configuraciones contenidas en este archivo serán válidas para todas las pruebas que se ejecutarán en el proyecto de pruebas con DTF.

La Ilustración 13 muestra los atributos y elementos que componen el archivo de configuración *settings.xml*.

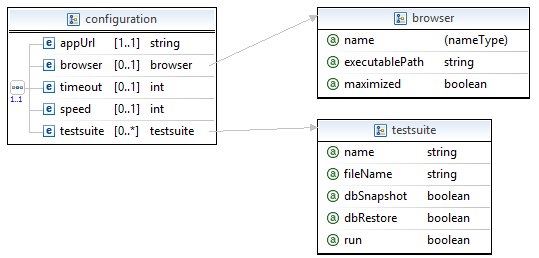


Ilustración Elementos de settings.xml

|  |  |  |
| --- | --- | --- |
| Atributo/Elemento | Req. | Descripción |
| appUrl | Si | La URL de la aplicación Web que será probada. |
| browser | No | Especifica el explorador de Internet que será usado para las pruebas. |
| name | Si | Especifica el nombre del navegador. El navegador deberá estar instalado en el equipo donde se ejecutarán las pruebas. |
| executablePath | No | Permite definir la ruta en el equipo local donde el framework podrá encontrar el archivo ejecutable del driver específico para el navegador. Esto es requerido para Google Chrome, por ejemplo, que necesita del driver disponible en https://code.google.com/p/selenium/wiki/ChromeDriver. |
| maximized | No | True para especificar que la ventana del navegador de Internet iniciará maximizada. |
| timeout | No | Especifica el tiempo de espera en milisegundos para que una acción sea llevada a cabo. |
| speed | No | Especifica el tiempo de ejecución entre dos acciones. |
| <testsuite> | No | Cero o más suites de pruebas que conformarán el total de las pruebas de la aplicación Web. |
| name | Si | Especifica el nombre de la suite. |
| fileName | No | Especifica el nombre y/o ubicación de archivo XML. El nombre debe ser el mismo que el nombre del archivo XML que contiene los scripts de pruebas. Es requerido cuando una suite no se encuentra directamente en la carpeta *testsuite* del proyecto de pruebas, sino en algunas sub-carpetas de la misma; o cuando el nombre de la suite no es igual al nombre del archivo XML que contiene los scripts de pruebas. |
| dbSnapshot | No | True para que el framework genere una copia de los datos existentes en la base de datos antes de ejecutar las pruebas. |
| dbRestore | No | True para que el framework restablezca los datos iniciales en la base de datos. Requiere la existencia del archivo testing\generated\dbsnapshots\export.xml. |
| run | No | False para especificar que el framework no deberá ejecutar los casos de prueba contenidos en esta suite. |

## Ejecución de las pruebas

Creados los scripts de pruebas y realizadas las configuraciones necesarias, procede la ejecución de los casos de prueba. Para realizarlo simplemente se debe ejecutar el archivo por lotes run-tests.bat localizado en la carpeta *bin* del proyecto de pruebas.

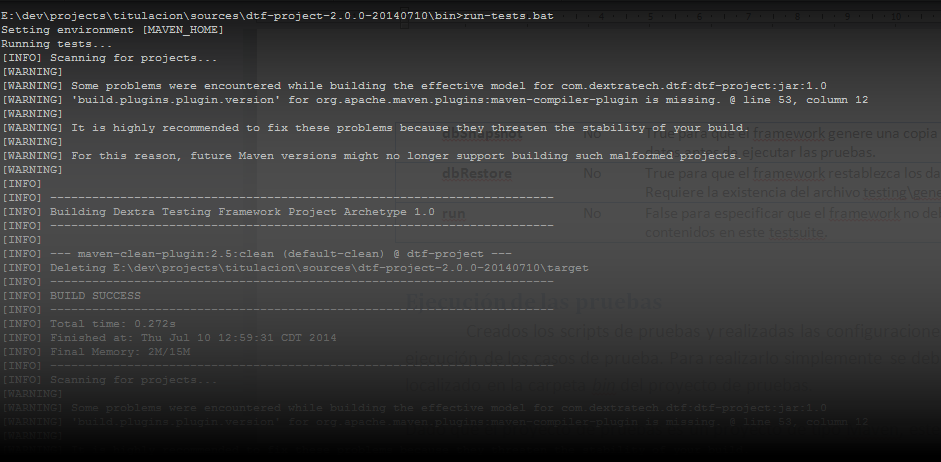


Ilustración Ejecución de run-tests.bat

La ejecución de cualquier comando disponible en DTF debe resultar en BUILD SUCCESS. Un BUILD FAILED o cualquier otro resultarían en un error acontecido durante la preparación del ambiente de ejecución de las pruebas o una excepción durante la ejecución de las mismas.

Dado que el proyecto de pruebas es un proyecto de tipo Maven, este contiene plugins para la fácil ejecución de las tareas. Se pueden llevar a cabo las mismas ejecuciones hechas con el archivo por lotes anterior usando los siguientes comandos de Maven:

> mvn clean dtf:clean integration-test

Los comandos anteriores se encargarán de limpiar eliminar del proyecto archivos autogenerados previamente y ejecutar la fase de pruebas de integración, en la cual se ejecutarán los scripts de pruebas. La ejecución, en sí misma, la realizan plugins configurados de facto en el proyecto de pruebas.

En la sección [Fases de DTF](#_Fases_de_DTF2) se listan los comandos disponibles en el framework que pueden ser de mucha utilidad.

## Fases de DTF

La ejecución de los casos de prueba realizada en la sección anterior, internamente lleva a cabo diversas tareas o fases de forma subsecuente. A continuación se listan estas fases y la descripción de las mismas.

|  |  |  |
| --- | --- | --- |
| Fase | Comandos | Descripción |
| Configuración de ambiente |  | En esta fase se ejecuta el archivo por lotes setEnv.bat para inicializar variables de entorno. |
| Limpieza | clean,  dtf:clean | Realiza la limpieza de directorios que contienen archivos generados automáticamente en cada ejecución. |
| Proceso de recursos | process-resources | Se realiza la copia de algunos archivos de recursos. Estos no necesitan manipulación externa. |
| Generación de clases de prueba | dtf:xml-testsuites | Esta fase es la más importante. En ella se llevarán a cabo sub-fases que corresponden a:   * Leer los scripts XML * Traducir los scripts XML en archivos HTML con comandos Selenese * Traducir los HTML generados a código Java que contendrán los scripts JUnit/Webdriver de las pruebas |
| Ejecución de los casos de prueba | test,  surefire:test | Se compilarán los archivos Java generados en la fase anterior y se ejecutarán las pruebas unitarias mediante Surefire. |
| Generación de reportes | site,  dtf:fix-surefire-reports | Una vez terminadas todas las pruebas, con los reportes XML y TXT generados por Surefire, se generará un reporte HTML y se parchará para agregar información de resultados adicionales generados por DTF2. El reporte HTML final será mostrado en un navegador configurado. Ver la sección Reportes. |

## Configuraciones avanzadas (advanced-settings.xml)

DTF permite realizar algunas configuraciones especiales al framework a través del archivo *advanced-settings.xml*. Este archivo es un archivo de configuración de Maven, usado en este caso para definir configuraciones globales avanzadas para el framework. El archivo presenta dos conjuntos de configuraciones; uno engloba las configuraciones que cambian el comportamiento de la tarea *clean* del plugin de DTF, las configuraciones de acceso a base de datos, y otro engloba a las configuraciones por defecto del framework, como rutas de archivos y carpetas, y versiones de librerías. Se recomienda no modificar este último conjunto de configuraciones. Algunas de las propiedades han sido descontinuadas en la nueva versión de DTF.

A continuación se describen las configuraciones disponibles en el archivo.

|  |  |
| --- | --- |
| Propiedad | Descripción |
| report.browser | La ruta al archivo ejecutable del navegador con el que se abrirá el reporte de resultados de las pruebas. |
| dtf.plugin.clean.testsuites | True para limpiar la carpeta de suites de pruebas generadas. Usada al ejecutar la tarea *clean*. |
| dtf.plugin.clean.java | True para limpiar la carpeta de clases Java generadas. Usada al ejecutar la tarea *clean*. |
| dtf.plugin.clean.screenshots | True para limpiar la carpeta de capturas de pantallas generadas. Usada al ejecutar la tarea *clean*. |
| db.host.port | Nombre y puerto del servidor de base de datos de la aplicación Web que será probada. |
| db.name | Nombre de la base de datos de la aplicación Web que será probada. |
| db.username | Usuario de la base de datos de la aplicación Web que será probada. |
| db.password | Contraseña de la base de datos de la aplicación Web que será probada. |
| dtf.plugin.verbose | Descontinuado. |
| dtf.plugin.seleniumHost | Descontinuado. |
| dtf.plugin.seleniumPort | Descontinuado. |
| dtf.plugin.browser | Descontinuado. |
| dtf.plugin.browser.maximized | Descontinuado. |
| dtf.plugin.baseUrl | Descontinuado. |
| dtf.plugin.timeout | Descontinuado. |
| dtf.plugin.speed | Descontinuado. |
| dtf.plugin.format | Descontinuado. |
| dtf-testsuites-dir | Ruta a la carpeta de scripts de pruebas. |
| settings-file | Ruta al archivo de configuración. En este caso, *settings.xml*. |
| generated-html-testsuites-dir | Ruta a la carpeta donde se guardarán los scripts Selenese/HTML que se generarán. |
| generated-screenshots-dir | Ruta a la carpeta donde se guardarán las capturas de pantalla que se generarán. |
| generated-reports-dir | Ruta a la carpeta donde se guardarán los reportes de resultados de las pruebas. |
| generated-java-testsuites-dir | Ruta a la carpeta donde se guardarán los archivos Java que se generarán. |
| generated-db-snapshot-file | Ruta al archivo que será generado al copiar los datos originales de la base de datos antes de ejecutar las pruebas a la aplicación Web. |
| userExtensions-dir | Ruta donde se almacenarán las extensiones de usuario. |
| project.reporting.outputEncoding | Codificación de caracteres del reporte generado. |
| default.base.package | Paquete o espacio de nombres por defecto de las clases Java generadas durante las pruebas. |
| spring.version | Versión de las librerías de Spring Framework. |
| hibernate.version | Versión de las librerías de Hibernate. |
| dtf.version | Versión de las librerías del framework DTF que utilizará el proyecto. |

## Extensiones de usuario

Las extensiones de usuario son archivos Javascript que contienen métodos que extienden las funcionalidades de Selenium agregando acciones, aserciones y estrategias de localización de elementos. Para que DTF cargue las extensiones de usuario estas deben almacenarse en la carpeta *testing/conf/user-extensions*. A continuación se muestra un ejemplo de acción personalizada.

Selenium.prototype.doTypeRepeated = **function**(locator, text) {

// All locator-strategies are automatically handled by "findElement"

**var** element = **this**.page().findElement(locator);

// Create the text to type

**var** valueToType = text + text;

// Replace the element text with the new text

**this**.page().replaceText(element, valueToType);

};

Ilustración Ejemplo de extensión de usuario

La guía para escribir extensiones de usuario está disponible en línea en <http://docs.seleniumhq.org/docs/08_user_extensions.jsp>.

## Restauración de datos

Generalmente las aplicaciones Web funcionan haciendo uso de una base de datos, las cuales son afectadas durante las pruebas automatizadas de la aplicación, y en muchos casos estas pruebas requieren de cierto estado inicial de los datos para poder ser realizadas. Pero esto puede ser problemático si no se cuenta con un respaldo de los datos previos a la ejecución de las pruebas.

Para afrontar estas situaciones DTF ofrece la posibilidad de realizar una copia de los datos disponibles en la base de datos de la aplicación Web que será probada, y restaurar tales datos copiados al finalizar las pruebas. Para que DTF realice estas tareas se deben considerar las siguientes opciones de configuración:

1. En el archivo *settings.xml* se deben configurar los atributos *dbSnapshot* y/o *dbRestore* de las suites de pruebas con las cuales se deben llevar a cabo dichas tareas.
2. Se debe crear un archivo nombrado como el motor de base de datos en la ubicación *testing/conf/using-rdbms*. Por ejemplo, si el motor de base de datos que usa la aplicación Web que se va a probar es Mysql, se debe crear un archivo llamado *mysql*. A continuación se listan los motores de base de datos soportados por DTF y el nombre respectivo del archivo que se deberá crear.

|  |  |
| --- | --- |
| RDBMS | Nombre de archivo |
| Mysql | mysql |
| Derby | derby |
| H2 | h2 |
| HSQLDB | hsqldb |
| Oracle | oracle |
| PostgreSQL | postgresql |
| SQLServer | sqlserver |

1. En el archivo *advanced-settings.xml* se deben especificar las propiedades para la base de datos que se usará.

|  |  |
| --- | --- |
| Propiedad | Descripción |
| db.host.port | Servidor y puerto de la base de datos. |
| db.name | Nombre de la base de datos o esquema. |
| db.username | Nombre de usuario. Deberá tener permisos para borrar y escribir datos. |
| db.password | Contraseña de usuario. |

Con estas configuraciones DTF podrá generar el archivo *testing/generated/dbsnapshots/export.xml* con los datos exportados y volver a restaurarlos posterior a la ejecución de las pruebas.

# Reportes

DTF ofrece la posibilidad de generar reportes HTML con los resultados de la ejecución de las pruebas automatizadas. Los insumos para los reportes son generados por Surefire y por DTF durante la ejecución de las pruebas; y estos son usados por un plugin de Maven para generar el reporte HTML.

## Generación de reportes

Para llevar a cabo la generación de reportes se hace uso de los comandos de Maven site y dtf:fix-surefire-reports, o simplemente post-site. El comando dtf:fix-surefire-reports se ejecuta para parchar el reporte HTML generado por site con la finalidad de agregar resultados adicionales generados por el framework DTF.

La Ilustración 16 muestra el ejemplo de un reporte HTML generado. El reporte muestra que la ejecución del tercer caso de prueba fue fallida y presenta los detalles del error. Los detalles del error incluyen el comando que estaba siendo ejecutado cuando ocurrió el error, el elemento y valor sobre el que ocurrió el error y la captura de pantalla tomada cuando éste ocurrió.

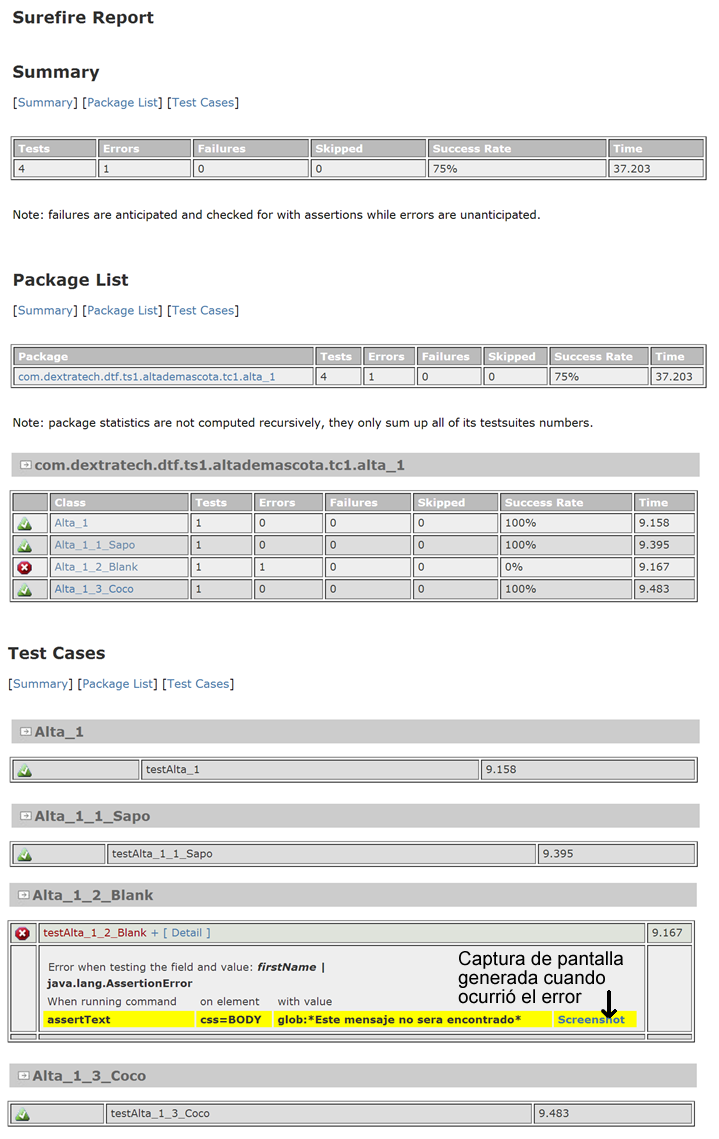


Ilustración Ejemplo de reporte HTML