

¿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)



[Home](#) | [Quienes Somos](#) | [Empleo](#) | [Tutoriales](#) | [Contacte](#)

<p>Tutorial desarrollado por: Jose Carlos López (Autentia) es consultor tecnológico de desarrollo de proyectos informáticos. Contacta en jclopez@autentia.com</p>	<p>www.adictosaltrabajo.com es el Web de difusión de conocimiento de www.autentia.com</p>  <p>autentia real business solutions</p> <p>Catálogo de cursos</p>
---	---

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

[Firma en nuestro libro de Visitas](#)

Master Java Certificado

Nuevo Temario-UML-JSF-AJAX-
J2ME Trabajo Garantizado-Bolsa
de Empleo
www.exes.es

Edicom - Soluciones EDI

Software XML EDI y CRP
Integración con ERP's
www.edicom.es

Java Struts o J2EE

¿experiencia en Java Struts o
J2EE? Mejora tu carrera. Unete a
nosotros
www.soitsa-intesys.com/em...

Accesibilidad Web

Curso On Line Subvencionado
_Creación de Web Sites Accesibles
www.ich.es

[Anuncios Google](#)

[Anunciarse en este sitio](#)

Pruebas de integración con Maven

Creación: 25-01-2007

1. Introducción

Actualmente Maven, en su versión 2.0, no soporta la configuración de distintos directorios para las pruebas unitarias y las pruebas de integración, ya que únicamente tenemos un `testSourceDirectory` por módulo, por lo que deben convivir en el mismo directorio. Ésto nos causa un problema, porque en la fase de test de maven, se lanzarían las pruebas unitarias conjuntamente con las pruebas de integración, que posiblemente necesiten un contenedor iniciado para que se realicen correctamente.

En el libro "[Better Builds with Maven](#)" nos explican como configurar Maven para ejecutar pruebas de integración creando un módulo específico para esta tarea y solucionar el problema comentado anteriormente. En este tutorial vamos a ver como configurar éstas pruebas dentro del mismo módulo donde tenemos configuradas nuestras pruebas unitarias. Damos por hecho de que tenemos instalado Maven y el Tomcat en nuestro equipo. Realizaremos un proyecto web llamado "autentiaWeb", donde convivan ambos tipos de prueba y se ejecuten correctamente en su fase correspondiente.

Nota: Existen [rumores](#) de que la versión 2.1 de Maven soportará la configuración de un directorio distinto para las pruebas de integración; eso esperamos.

2. Entorno

El tutorial está escrito usando el siguiente entorno:

- Hardware: HP COMPAQ Presario V6000 (Centrino Duo 1.66GHz, 2048 MB RAM, 100 GB HD)
- Sistema Operativo: Windows XP Home Edition
- Máquina Virtual Java: JDK 1.6.0 de Sun Microsystems
- Maven 2.0.4
- Tomcat 5.5.20
- Cargo 0.8 (plugin para maven)

3. Creando el proyecto

Para empezar, vamos al crear el proyecto con Maven. Suponiendo que nuestros proyectos los tenemos en C:\home\workspace basta con hacer:

```
C:\>cd home/workspace
```

```
C:\home\workspace>mvn archetype:create -DgroupId=com.autentia.demoapp -
DartifactId=autentiaWeb -DarchetypeArtifactId=maven-archetype-webapp
```

Donde groupId es el identificador único de la organización o grupo que crea el proyecto, artifactId es el identificador único del artefacto principal de este proyecto y le indicamos mediante el atributo archetypeArtifactId que queremos usar la plantilla de aplicaciones web de Maven.

Ahora editamos el archivo index.jsp que nos ha creado Maven y ponemos:

```
<html>

<title>MyIntegrationTest</title>

<body>

<h2>Integration Test!!!!</h2>

</body>

</html>
```

Podemos probar nuestro proyecto publicandolo en nuestro contenedor favorito y vemos como funciona.

4. Creando los ficheros de prueba

Vamos a crear los ficheros para las pruebas, que meteremos en la carpeta `src/test/java` del proyecto, que es donde Maven configura por defecto el directorio *testSourceDirectory* para localizar los ficheros de pruebas.

Antes de nada vamos a estudiar un poco el comportamiento del plugin de Maven [maven-surefire-plugin](#), que es el plugin utilizado por Maven automaticamente en la fase de test.

Como podemos ver en la documentación, en la descripción del parámetro `includes`, si no especificamos lo contrario, el plugin ejecuta todos test que encuentre en el directorio por defecto de test que cumplan alguno de estos patrones:

```
**/Test*.java, **/*Test.java, **/*TestCase.java
```

Por lo tanto, y siguiendo la documentación, creamos la clase `MyTest.java` (termina en `Test.java` cumpliendo uno de los patrones para que se ejecute en la fase de test) con el siguiente código:

```
import junit.framework.TestCase;

public class MyTest extends TestCase {

    public void testHolaMundo() {

        assertEquals("HolaMundo", "HolaMundo");

    }

}
```

Ahora creamos otro fichero para el test de la fase de integración, pero como no queremos que se ejecute en la fase de test de Maven, lo nombramos incumpliendo todos los patrones anteriores, por ejemplo lo llamamos `MyTestIt.java`. De este modo, al lanzarse las pruebas unitarias en la fase de test, esta prueba no será lanzada (más adelante veremos como configurar mediante este mismo plugin las pruebas de integración y que esta vez si se lance).

```
import junit.framework.*;

import com.meterware.httpunit.*;

public class MyTestIt extends TestCase {

    public void testDisplayMainPage() throws Exception {

        WebConversation wc = new WebConversation();

        WebRequest request = new GetMethodWebRequest("http://localhost:8080/autentiaWeb/index.jsp");

        WebResponse response = wc.getResponse(request);

        assertEquals("MyIntegrationTest", response.getTitle());

    }

}
```

Esta prueba realiza una petición al servidor local del archivo, y tendrá éxito cuando me devuelva la página con el título correcto, una prueba sencilla. Como vemos en esta prueba se hace referencia a la librería `httpunit`, de la cual habrá que crear una dependencia en el pom del proyecto.

5. Configurando el pom.xml

Lo primero es declarar todas las dependencias, incluyendo a la librería `httpunit` comentada anteriormente y a las librerías del plugin de Cargo, que utilizaremos para arrancar y parar el contenedor Tomcat antes de lanzar las pruebas de integración (en las fases `pre-integration-test` y `post-integration-test` respectivamente).

```
<dependencies>

  <dependency>

    <groupId>junit</groupId>

    <artifactId>junit</artifactId>

    <version>3.8.1</version>

    <scope>test</scope>

  </dependency>

  <dependency>

    <groupId>org.codehaus.cargo</groupId>

    <artifactId>cargo-core-uberjar</artifactId>

    <version>0.8</version>

    <scope>test</scope>

  </dependency>

  <dependency>

    <groupId>org.codehaus.cargo</groupId>

    <artifactId>cargo-ant</artifactId>

    <version>0.8</version>

    <scope>test</scope>

  </dependency>

  <dependency>

    <groupId>httpunit</groupId>

    <artifactId>httpunit</artifactId>

    <version>1.6.1</version>

    <scope>test</scope>

  </dependency>

</dependencies>
```

Ahora añadimos la configuración de Cargo para arrancar y parar el contenedor Tomcat en las fases correspondientes:

```
<build>

  [...]

  <plugin>

    <groupId>org.codehaus.cargo</groupId>

    <artifactId>cargo-maven2-plugin</artifactId>

    <executions>

      <execution>

        <id>start-container</id>

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

        <goals>
```

```

                                <goal>start</goal>
                            </goals>
                        </execution>
                    <execution>
                        <id>stop-container</id>
                        <phase>post-integration-test</phase>
                        <goals>
                            <goal>stop</goal>
                        </goals>
                    </execution>
                </executions>
            <configuration>
                <!-- Configuración del contenedor -->
                <wait>false</wait>
                <container>
                    <containerId>tomcat5x</containerId>
                    <home>c:/servidores/tomcat/</home>
                </container>
            </configuration>
        </plugin>
    [...]
</build>

```

Tomcat lo tenemos instalado en el directorio C:/servidores/tomcat. El container lo arrancamos con start en la fase de pre-integration-test y lo paramos con stop en la fase de post-integration-test. El wait a false en la configuración del contenedor es debido a que al hacer start se quedaría esperando (arrancaría el servidor y estaría listo para hacer peticiones) y no seguiría avanzando hacia las siguientes fases, que es lo que queremos para lanzar seguidamente la prueba de integración.

Ahora configuramos la prueba de integración con el plugin de Maven:

```

<build>
    [...]
    <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <executions>
            <execution>
                <id>it-test</id>
                <phase>integration-test</phase>
                <goals>
                    <goal>test</goal>
                </goals>
            </execution>
        </executions>
    </plugin>
</build>

```

```

</includes>

<include>**/*It.class</include>

</includes>

</configuration>

</execution>

</executions>

</plugin>

[...]

</build>

```

Como vemos en la configuración, le decimos que tiene que ejecutar los ficheros que terminen en `*It.class`, así logramos ejecutar únicamente los ficheros de pruebas de integración.

6. Lanzamos el proyecto

Ahora ejecutamos con Maven para ver el resultado:

```
C:\home\workspace\autentiaWeb>mvn clean post-integration-test
```

```
[INFO] Scanning for projects...
```

```
[INFO] -----
```

```
[INFO] Building autentiaWeb Maven Webapp
```

```
[INFO] task-segment: [clean, post-integration-test]
```

```
[INFO] -----
```

```
[INFO] [clean:clean]
```

```
[INFO] Deleting directory C:\home\workspace\tutoriales\autentiaWeb\target
```

```
[INFO] Deleting directory C:\home\workspace\tutoriales\autentiaWeb\target\classes
```

```
[INFO] Deleting directory C:\home\workspace\tutoriales\autentiaWeb\target\test-classes
```

```
[INFO] [resources:resources]
```

```
[INFO] Using default encoding to copy filtered resources.
```

```
[INFO] [compiler:compile]
```

```
[INFO] No sources to compile
```

```
[INFO] [resources:testResources]
```

```
[INFO] Using default encoding to copy filtered resources.
```

```
[INFO] [compiler:testCompile]
```

```
Compiling 2 source files to C:\home\workspace\tutoriales\autentiaWeb\target\test-classes
```

```
[INFO] [surefire:test] SE LANZAN LAS PRUEBAS UNITARIAS
```

```
[INFO] Surefire report directory: C:\home\workspace\tutoriales\autentiaWeb\target\surefire-reports
```

```
-----
```

```
T E S T S
```

```
-----
```

```
Running MyTest
```

```
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.015 sec
```

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] [war:war]

[INFO] Exploding webapp...

[INFO] Assembling webapp autentiaWeb in C:\home\workspace\tutoriales\autentiaWeb\target\autentiaWeb

[INFO] Copy webapp webResources to C:\home\workspace\tutoriales\autentiaWeb\target\autentiaWeb

[INFO] Generating war C:\home\workspace\tutoriales\autentiaWeb\target\autentiaWeb.war

[INFO] Building war: C:\home\workspace\tutoriales\autentiaWeb\target\autentiaWeb.war

[INFO] [cargo:start {execution: start-container}] SE LANZA EL CONTENEDOR TOMCAT

[INFO] [talledLocalContainer] Tomcat 5.5.20 starting...

[INFO] [CopyingLocalDeployer] Deploying [C:\home\workspace\tutoriales\autentiaWeb\target\autentiaWeb.war] to [C:\home\workspace\tutoriales\autentiaWeb\target\tomcat5x\webapps]...

[INFO] [talledLocalContainer] Tomcat 5.5.20 started on port [8080]

[INFO] [surefire:test {execution: it-test}] SE LANZAN LAS PRUEBAS DE INTEGRACION

[INFO] Surefire report directory: C:\home\workspace\tutoriales\autentiaWeb\target\surefire-reports

T E S T S

Running MyTestIt

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 1.235 sec

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] [cargo:stop {execution: stop-container}] SE PARA EL CONTENEDOR

[INFO] [talledLocalContainer] Tomcat 5.5.20 is stopping...

[INFO] [talledLocalContainer] Tomcat 5.5.20 is stopped

[INFO] -----

[INFO] BUILD SUCCESSFUL

[INFO] -----

[INFO] Total time: 14 seconds

[INFO] Finished at: Thu Jan 25 17:39:00 CET 2007

[INFO] Final Memory: 6M/14M

[INFO] -----

7. Conclusiones

Como veis, hemos conseguido lanzar las pruebas de integración "engañando" a Maven para que no se lanzaran en la fase de test teniendo únicamente un módulo para ambas.

8. Sobre el autor

José Carlos López Díaz, Ingeniero en Informática

jclopez@autentia.com

[Autentia](#) Real Business Solutions S.L - "Soporte a Desarrollo"



This work is licensed under a [Creative Commons Attribution-NonCommercial-No Derivative Works 2.5 License](#).



[Puedes opinar sobre este tutorial aquí](#)

Recuerda

que el personal de [Autentia](#) te regala la mayoría del conocimiento aquí compartido ([Ver todos los tutoriales](#))

¿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?

info@autentia.com

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

Autentia = Soporte a Desarrollo & Formación

soluciones reales para su negocio

[Autentia S.L.](#) Somos expertos en:

J2EE, Struts, JSF, C++, OOP, UML, UP, Patrones de diseño ..
y muchas otras cosas

Nuevo servicio de notificaciones

Si deseas que te enviemos un correo electrónico cuando introduzcamos nuevos tutoriales, inserta tu dirección de correo en el siguiente formulario.

Subscribirse a Novedades	
e-mail	
	<input type="button" value="Enviar"/>

Otros Tutoriales Recomendados ([También ver todos](#))

Nombre Corto

Descripción

[Crear un repositorio remoto y como hacer una 'release' con varios proyectos en Maven y Eclipse](#)

En este tutorial vamos a explicar como podemos trabajar teniendo varios proyectos relacionados en Maven y en Eclipse

[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.

[Maven, nunca antes resultó tan fácil compilar, empaquetar, ...](#)

En este tutorial aprenderemos el uso de esta herramienta que nos permite compilar, empaquetar, generar documentación, pasar los test, preparar las builds de nuestros proyectos

[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

[JUnit 4. Pruebas de Software Java](#)

Tutorial que describe como utilizar la herramienta JUnit 4 para realizar pruebas de integridad y errores sobre Java.

[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

[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.

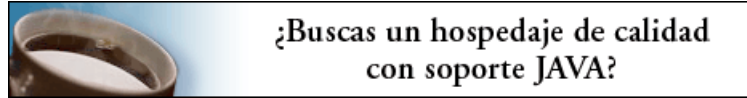
Nota: Los tutoriales mostrados en este Web tienen como objetivo la difusión del conocimiento.

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.

[Patrocinados por enredados.com Hosting en Castellano con soporte Java/J2EE](#)



www.AdictosAlTrabajo.com Optimizado 800X600