

¿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)



Estás en:

[Inicio](#) [Tutoriales](#) Como ejecutar los test de JUnit de todos los proyectos en Eclipse, gracias ...



DESARROLLADO POR:
Alejandro Pérez García

Alejandro es socio fundador de Autentia y nuestro experto en J2EE, Linux y optimización de aplicaciones empresariales.

Ingeniero en Informática y Certified ScrumMaster

Si te gusta lo que ves, puedes contratarle para darte ayuda con soporte experto, impartir **cursos presenciales** en tu empresa o para que **realicemos tus proyectos como factoría** (Madrid). Puedes encontrarme en Autentia: Ofrecemos servicios de soporte a desarrollo, factoría y formación

Catálogo de servicios
Autentia



SUBEESCALERAS | PLATAFORMAS ELEVADORAS
Llámenos gratuitamente

Fecha de publicación del tutorial: 2009-02-26



Share |

[Regístrate para votar](#)

Como ejecutar los test de JUnit de todos los proyectos en Eclipse, gracias a ClasspathSuite

Creación: 27-06-2011

Índice de contenidos

- [1. Introducción](#)
- [2. Entorno](#)
- [3. Como usar ClasspathSuite](#)
 - [3.1. Descargar el jar de ClasspathSuite](#)
 - [3.2. Crear un nuevo proyecto y añadir el jar de ClasspathSuite](#)
 - [3.3. Añadir los proyectos de los que queremos lanzar todos los test](#)
 - [3.4. Como ejecutar todos los test de todos los proyectos](#)
- [4. Conclusiones](#)
- [5. Sobre el autor](#)

1. Introducción

Todos los que estamos acostumbrados a trabajar en Eclipse, seguro que estamos hartos de posicionarnos sobre un proyecto y hacer **Botón derecho --> Run As --> JUnit Test** (o su equivalente **CMD+ALT+X T**), para ejecutar todos los test de JUnit que se encuentran dentro de ese proyecto.

Pero es bastante común y buena práctica tener más de un proyecto, ya que una misma aplicación la separamos en varios módulos. Si además de esto trabajamos con TDD tendremos un número considerable de test, y las buenas prácticas nos dicen que debemos ejecutar todos los test cada vez.

Últimas Noticias

- [Alfresco Day 2011](#)
- [XVII Charla Autentia - Grails - Vídeos y Material](#)
- [iii 15 millones de descargas de tutoriales !!!](#)
- [XVII Charla Autentia - Grails](#)
- [Charla en WhyFLOSS en el IE: la ppt](#)

[Histórico de NOTICIAS](#)

Últimos Tutoriales

- [Spring cache abstraction.](#)
- [Organizar eventos con Doodle](#)
- [Mejorar nuestro TDD gracias a Happyprog](#)
- [EpubLib, una librería Java para leer Epub](#)
- [Eclipse Indigo, la versión 3.7 de Eclipse](#)

Donde *todos* significa *todos*, es decir, los test de todos los proyectos y no sólo los test del proyecto donde hemos tocado código (es cierto que si los test son unitarios no deberían afectar a otros módulos, pero más vale prevenir que curar, y por eso lo llaman *buenas prácticas* ;)

Pero ¿en Eclipse es posible ejecutar de golpe todos los test de todos los proyectos? Lamentablemente la respuesta es NO, ni siquiera en la última versión Eclipse Indigo.

En este tutorial vamos a ver precisamente eso, como gracias a [ClasspathSuite](#) podemos ejecutar todos los test de JUnit de todos los proyectos de nuestro workspace.

2. Entorno

El tutorial está escrito usando el siguiente entorno:

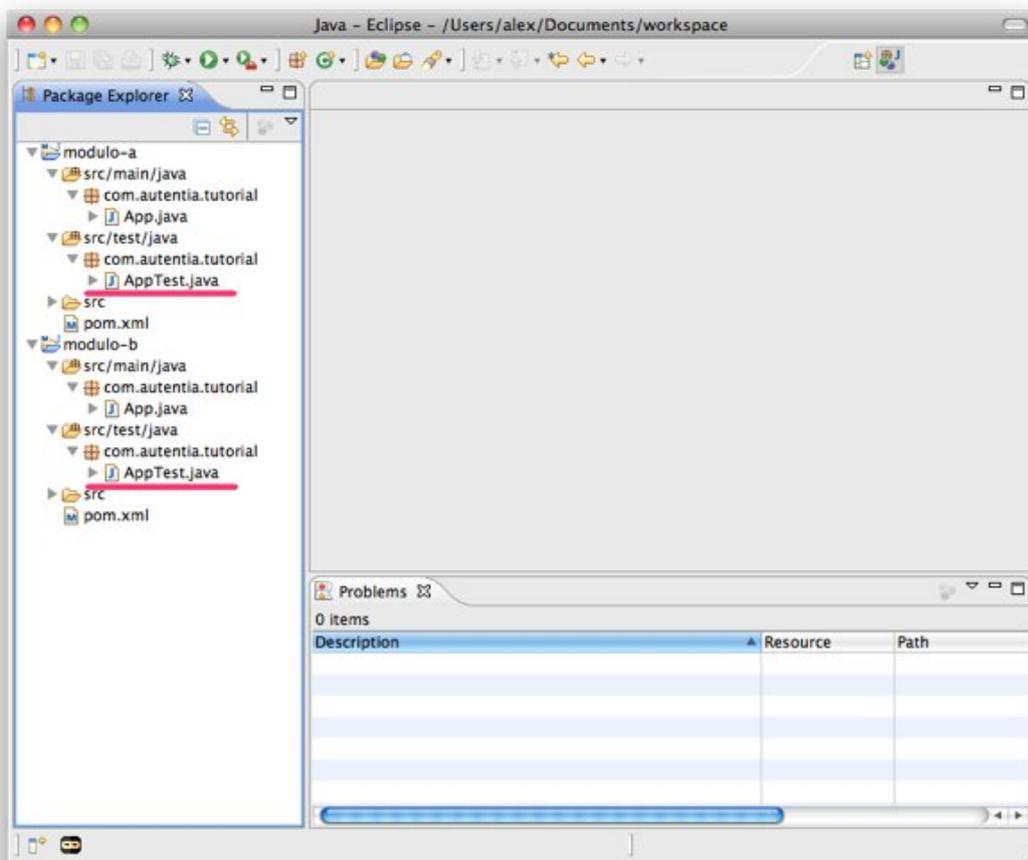
- Hardware: Portátil MacBook Pro 17' (2.8 GHz Intel i7, 8GB DDR3 SDRAM, 256GB Solid State Drive).
- NVIDIA GeForce GT 330M with 512MB
- Sistema Operativo: Mac OS X Snow Leopard 10.6.8
- Eclipse 3.7, codename Indigo
- JUnit 4.8.2
- ClasspathSuite 1.2.5

3. Como usar ClasspathSuite

ClasspathSuite es un simple jar con un *Runner* de JUnit. Este Runner se encarga de buscar todos los test JUnit por el classpath y ejecutarlos.

Con esta idea, el mecanismo para ejecutar los test de todos los proyectos es sencillo, simplemente tenemos que crear un nuevo proyecto Java donde añadiremos como dependencia el resto de proyectos (todos aquellos proyectos de los que queremos lanzar los test de JUnit).

Así, supongamos que tenemos dos proyectos: *modulo-a* y *modulo-b*. Donde cada uno de ellos tiene sus respectivos test.



3.1. Descargar el jar de ClasspathSuite

Lo primero que vamos a hacer es descargarnos el jar de ClasspathSuite. Para ello vamos a la

Últimos Tutoriales del Autor

Mejorar nuestro TDD gracias a Happyprog

Eclipse Indigo, la versión 3.7 de Eclipse

Trabajando con GIT, introducción al uso de los branch y git-completion.bash

RVM y como actualizar Ruby a la versión 1.9.2 en Snow Leopard 10.6.7

REST y como hacer con jQuery un PUT hacia Spring MVC

Síguenos a través de:



Últimas ofertas de empleo

011-06-20
 Comercial - Ventas - SEVILLA.

011-05-24
 Contabilidad - Especialista
:ontable - BARCELONA.

011-05-14
 Comercial - Ventas - TARRAGONA.

011-04-13
 Comercial - Ventas - VALENCIA.

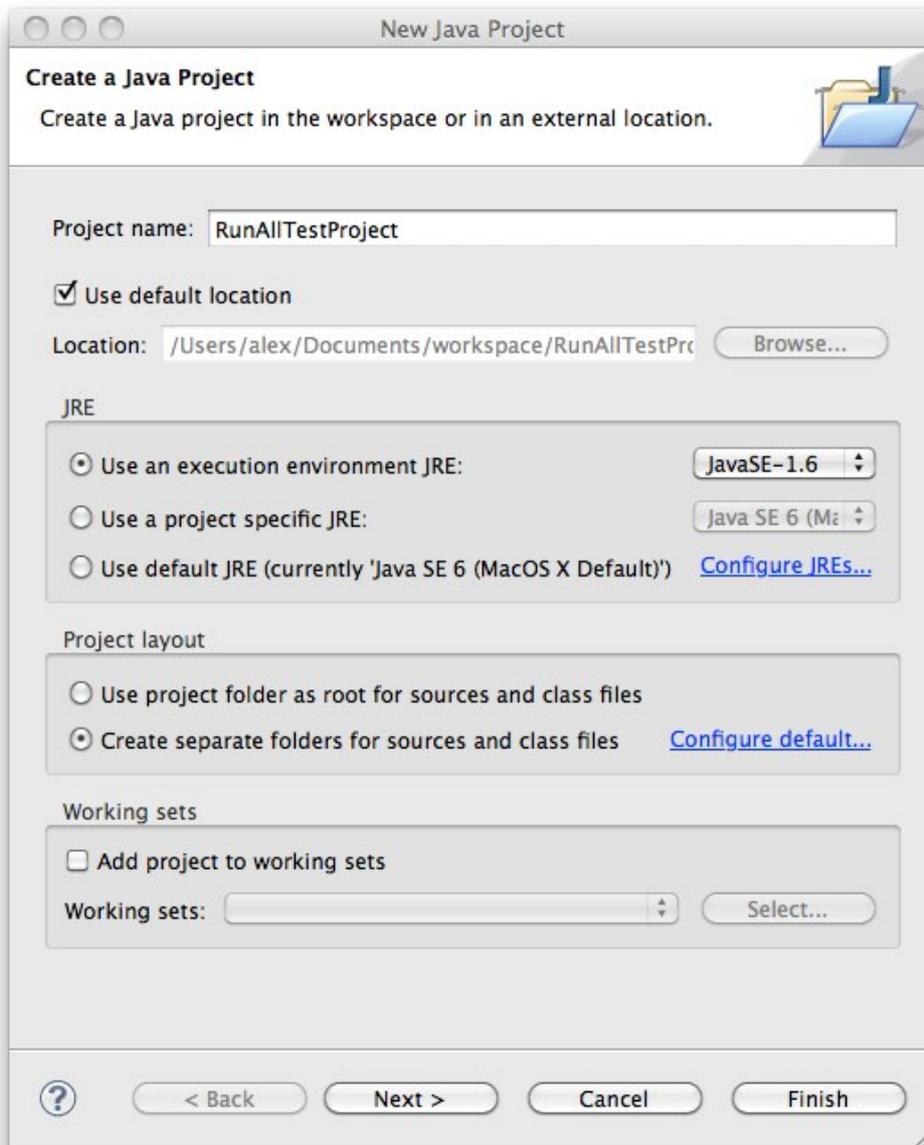
011-04-04
 Comercial - Compras - :ANTABRIA.

Alejandro Pérez
alejandroprogar

página ClasspathSuite y elegimos la versión que nos interese (en mi caso **cpsuite-1.2.5-jar**, que es para *JUnit 4.5*).

3.2. Crear un nuevo proyecto y añadir el jar de ClasspathSuite

Ahora vamos a crear un nuevo proyecto Java, para ello hacemos File --> New --> Java Project



Pulsamos el botón *Finish* para terminar la creación del proyecto.

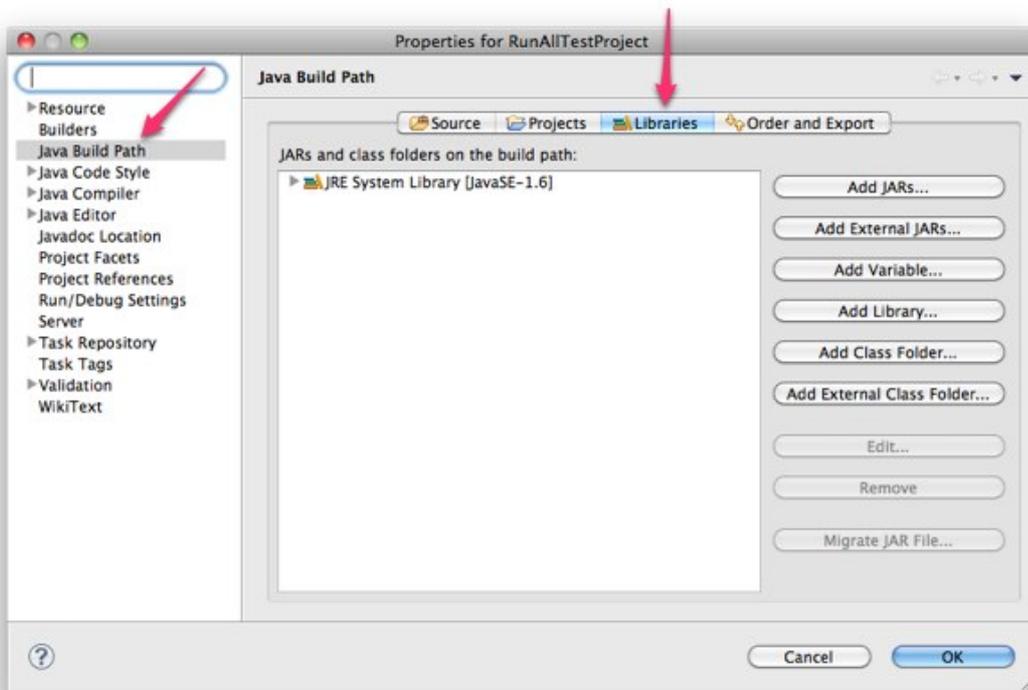
Ahora vamos a añadir el jar de ClasspathSuite al proyecto que acabamos de añadir. Para ello hacemos botón derecho sobre el proyecto recién creado y nos vamos a *Properties* (también lo podemos hacer con CMD+I sobre el proyecto). Nos posicionamos en la izquierda sobre la opción *Java Build Path*, y luego en la pestaña *Libraries*.

@unclebobmartin my
180 RSpecs = 64 sec.
167 (838 steps)
Cukes (including UI
smoke tests) = 261
sec. (just completed
pre-release tests :-)
3 days ago · reply · retweet ·
favorite

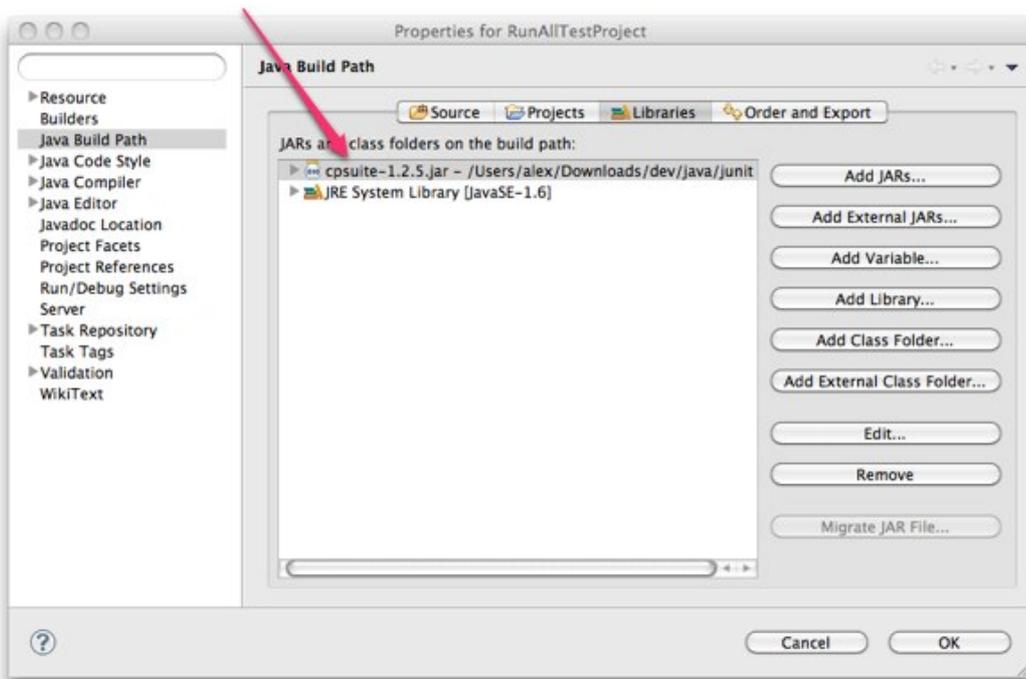
Why do we still
tolerate tests with
long start-up times.
Come on people, slow
tests are death. You
should be able to run
a test in seconds
3 days ago · reply · retweet ·
favorite

@ecomba
@rcanalesmora

Join the conversation



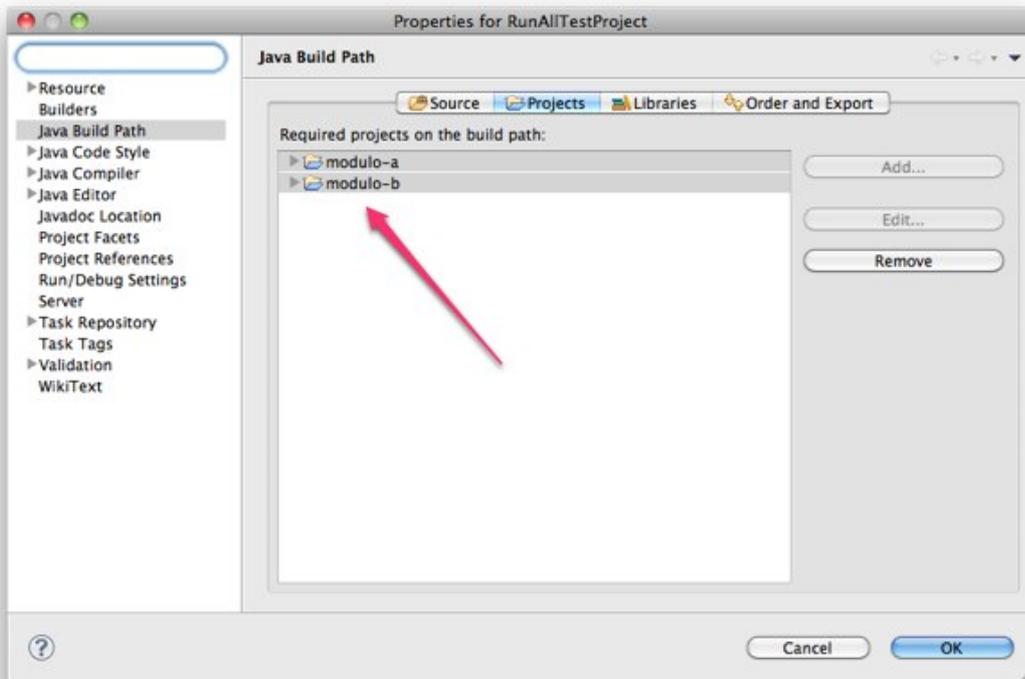
Pulsamos el botón de *Add External JARs...* e indicamos la ruta donde nos hemos descargado el jar. Ahora deberíamos ver el jar en nuestra lista de librerías.



Pulsamos el botón *OK* para aceptar los cambios.

3.3. Añadir los proyectos de los que queremos lanzar todos los test

Ya tenemos creado el proyecto y añadida la librería. Ahora tenemos que marcar como dependencias todos los proyectos de los que queremos lanzar todos los test. Igual que antes hacemos *Botón Derecho sobre el proyecto --> Propiedades --> Java Build Path --> Pestaña Projects*, y añadimos (botón *Add...*) todos los proyectos (en nuestro ejemplo: *modulo-a* y *modulo-b*).



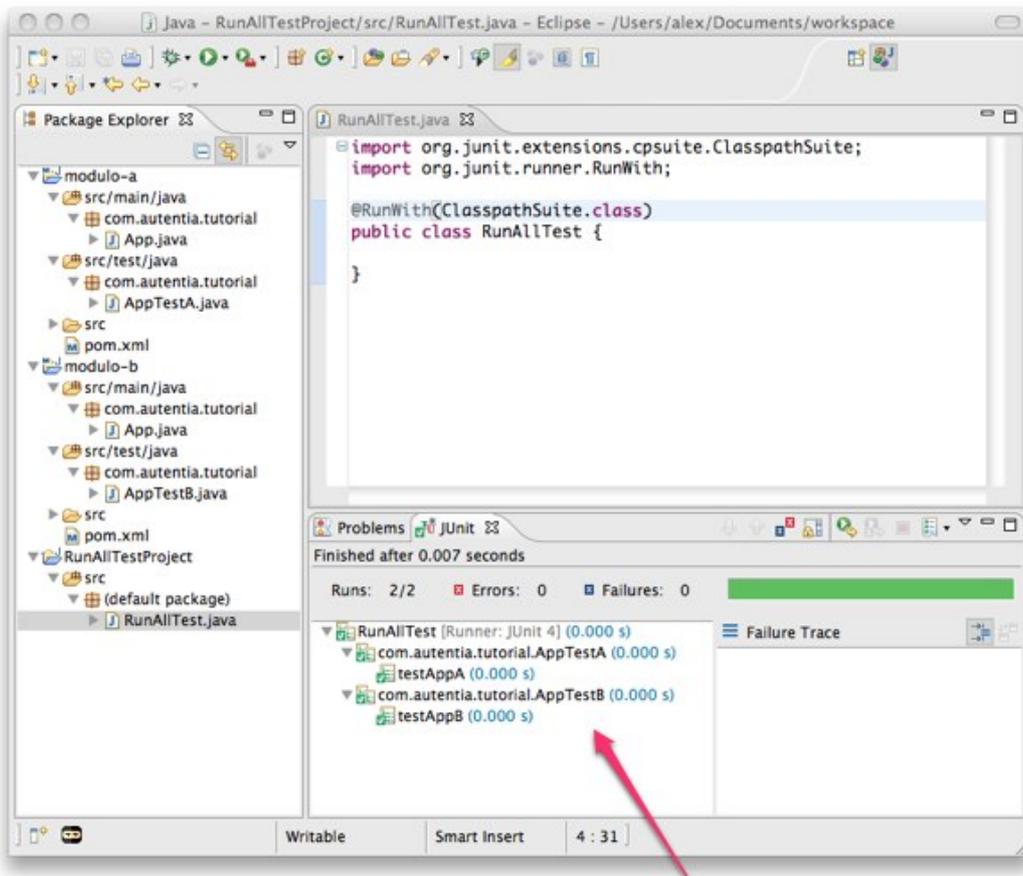
3.4. Como ejecutar todos los test de todos los proyectos

Ahora en nuestro nuevo proyecto basta con crear un test de la siguiente forma:

```
1 import org.junit.extensions.cpsuite.ClasspathSuite;
2 import org.junit.runner.RunWith;
3 @RunWith(ClasspathSuite.class)
4 public class RunAllTest {
5 }
```

Vemos como la clase está vacía, no define ningún test. Lo interesante está en la línea 4, donde le decimos que use el runner `ClasspathSuite.class`. Este es el que se encarga de buscar y ejecutar todos los test en los proyectos de los que depende.

Si ejecutamos esta clase de test (Sobre la clase `ALT+CMD+X T`, o botón derecho `Run As --> JUnit Test`) obtendremos el siguiente resultado:



Se ve claramente como se han ejecutado los test de ambos proyectos (he renombrado los proyectos y los test respecto a la primera imagen para que se vea más mejor como se han ejecutado todos los test).

3.5. Ejecutando los test unitarios y los test de integración por separado

Dentro del desarrollo de una aplicación es normal que tengamos muchos test unitarios y unos pocos de integración. El problema que tienen estos últimos es que requieren configurar el entorno (levantar el contexto de Spring, preparar una base de datos, ...) por lo que suelen tardar más en ejecutarse; mientras que los test unitarios suele ser cuestión de unos pocos milisegundos.

Con lo que hemos visto en el punto anterior, constantemente estamos lanzando todos los test, por lo que se puede hacer una tarea aburrida ya que tenemos que esperar a que terminen los test de integración. Y esta situación suele desembocar en que dejamos de lanzar los test o por lo menos no los lanzaremos todo lo que deberíamos (cada vez que hacemos cualquier cambio, incluso si cambiamos un comentario ;)

Pero no preocuparse porque ClasspathSuite viene preparado para todo y nos va a permitir realizar **filtros**. De esta forma si queremos ejecutar sólo los test unitarios podríamos hacernos una clase del estilo:

```

01 package com.autentia.asefa;
02
03 import org.junit.extensions.cpsuite.ClasspathSuite;
04 import org.junit.extensions.cpsuite.ClasspathSuite.ClassnameFilters;
05 import org.junit.runner.RunWith;
06
07 @RunWith(ClasspathSuite.class)
08 @ClassnameFilters({ "!.*IntegrationTest" })
09 public class AllUnitTestRunner {
10 }

```

Mientras que si queremos lanzar sólo los test de integración podemos hacernos una clase del estilo:

```

01 package com.autentia.asefa;
02
03 import org.junit.extensions.cpsuite.ClasspathSuite;
04 import org.junit.extensions.cpsuite.ClasspathSuite.ClassnameFilters;
05 import org.junit.runner.RunWith;
06
07 @RunWith(ClasspathSuite.class)
08 @ClassnameFilters({ ".*IntegrationTest" })
09 public class AllIntegrationTestRunner {
10 }

```

La gracia de los dos ejemplos está en la **línea 08**, donde se ve como con la anotación **@ClassnameFilters** podemos definir una serie de filtros para indicar a ClasspathSuite las clases de test que tiene que tener en cuenta.

En nuestro ejemplo estamos tomando la convención de llamar a las clases de test de integración con el sufijo *IntegrationTest*, de esta forma con la primera clase y la admiración "!" estamos negado que

se ejecuten estas clases, mientras que en la segunda clase sólo estamos ejecutando estas clases.

Para más información sobre qué cosas se pueden hacer con ClasspathSuite os recomiendo una visitar a su página donde explica algunos trucos más.

4. Conclusiones

Después de este tutorial ya no hay excusa para no estar ejecutando constantemente nuestro test, sobre todos los unitarios que se ejecutan en cuestión de milisegundos.

Además para facilitar esto os recomiendo ir a las preferencias del Eclipse: En el menú, *Eclipse --> Preferences...* --> *Run/Debug --> Launching --> Launch Operation*, y marcar activa la opción de *Always launch the previously launched application*.

De esta forma una vez ejecutéis la clase que lanza todos los test unitarios de todos los proyectos, podéis hacer *CMD+SHIFT+F11* y, con independencia del fichero sobre el que os encontréis, estaréis lanzando de nuevo todos los test. Así nuestro ciclo de trabajo será:

1. hacer un cambio,
2. *CMD+S* (Guardar),
3. *CMD+SHIFT+F11* (Lanzar todos los test unitarios),
4. si todo va bien, volver al primer punto

5. Sobre el autor

Alejandro Pérez García, Ingeniero en Informática (especialidad de Ingeniería del Software) y Certified ScrumMaster

Socio fundador de Autentia (Desarrollo de software, Consultoría, Formación)

<mailto:alejandropg@autentia.com>

Autentia Real Business Solutions S.L. - "Soporte a Desarrollo"

<http://www.autentia.com>

Anímate y coméntanos lo que pienses sobre este **TUTORIAL**:

Puedes opinar o comentar cualquier sugerencia que quieras comunicarnos sobre este tutorial; con tu ayuda, podemos ofrecerte un mejor servicio.

Enviar comentario

(Sólo para usuarios registrados)

» **Regístrate** y accede a esta y otras ventajas «

COMENTARIOS



Esta obra está licenciada bajo licencia Creative Commons de Reconocimiento-No comercial-Sin obras derivadas 2.5