El blog de Zarovich

videojuegos, desarrollo, ocio, opinión

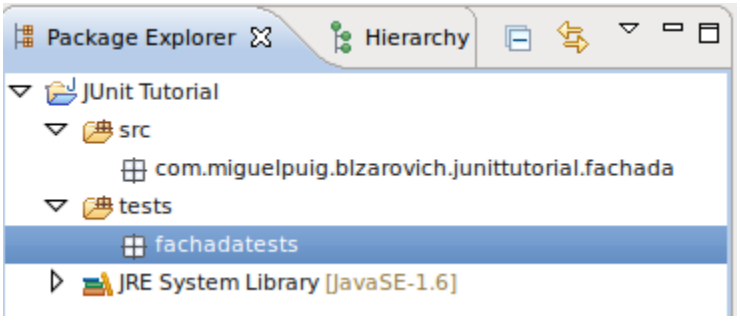
Tutorial JUnit en Eclipse

JUnit es un framework desarrollado para la realización de tests de unidad. Tiene como principal característica poder definir Tests de forma rápida y sencilla a través de la utilización de etiquetas. También es posible crear clases que sirvan para lanzar conjuntos de tests (TestSuite) y de esta manera automatizar más aún el proceso.

Veamos un ejemplo rápido que espero os ayude a comprender la creación de tests de unidad de una forma eficaz en Eclipse SDK Galileo.

Primero de todo vamos a crear un nuevo proyecto, para ello vamos a *File->New->Java Project* o pulsamos Shift+Alt+N y seleccionamos *Java Project*. Da igual el nombre que se le ponga, aseguraros de que teneis la última versión de Java seleccionada, en este caso yo estoy trabajando con JavaSE-1.6.

Una vez creado debereis de crear una carpeta de código por separado para los tests, es una práctica habitual aúnque no es necesario para su funcionamiento. Pero la idea es tener el código del proyecto separado de los tests de unidad.



Como podeis ver en la captura, la idea es crear una estructura similar a esto. En src tendreis el código del proyecto y en tests estarán los tests de unidad. Imaginemos que tenemos una serie de fachadas en el proyecto donde están algunos de los casos de uso que os interesa comprobar su funcionamiento.

Pués para ello creemos un paquete en los tests que tendrá los tests relacionados con este material. Sería interesante crear un paquete más que contenga otros casos de uso, yo crearé un paquete

com.miguelpuig.blzarovich.junittutorial.fachada2. Esto lo hago para mostraros más adelante como hacer una Test que lance los tests de las dos fachadas.

Ahora crearemos en fachada1 dos clases *Suma* y *Multiplicacion*. Estas dos clases tan simples e inútiles simplemente guardarán dos variables que se podrán sumar y restar. Su código sería tal que así.

view plain copy to clipboard print ?

```
01. package com.miguelpuig.blzarovich.junittutorial.fachada;
02.
03. public class Suma {
04.     private int a;
05.     private int b;
06.
07.     public Suma(int a, int b) {
08.         this.a = a;
09.         this.b = b;
10.     }
11.
12.     /**
13.      * Suma a + b
14.      * @return a+b
15.      */
16.     public int suma() {
17.         return a + b;
18.     }
19.
20.     /**
21.      * Resta a - b
22.      * @return a-b
23.      */
24.     public int resta() {
25.         return a - b;
26.     }
27. }
```

Suma tiene dos métodos, para poder hacer la prueba de meter dos tests en una misma clase de junit.

view plain copy to clipboard print ?

```
01. package com.miguelpuig.blzarovich.junittutorial.fachada;
02.
03. public class Multiplicacion {
04.     private int a;
05.     private int b;
06.
07.     public Multiplicacion(int a, int b) {
08.         this.a = a;
09.         this.b = b;
10.     }
11.
12.     public int multiplicar() {
```

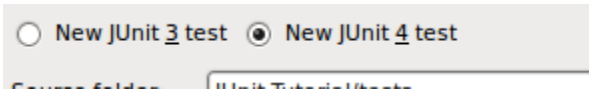
```
13.             return a*b;
14.         }
15.     }
```

Bien, ahora en la fachada2 vamos a hacer lo mismo para la división.

view plain copy to clipboard print ?

```
01. package com.miguelpuig.blzarovich.junittutorial.fachada2;
02.
03. public class Division {
04.     private int a;
05.     private int b;
06.
07.     public Division(int a, int b) {
08.         this.a = a;
09.         this.b = b;
10.     }
11.
12.     public int divide() {
13.         return a/b;
14.     }
15. }
```

Ahora ya tenemos material para poder probar junit. Vereis que es realmente sencillo. Primero vamos a realizar una clase para testear la suma. Para ello botón derecho en el paquete fachada de la carpeta de tests, vais a nuevo y seleccionais JUnit Test Case (en caso de que no aparezca tendreis que pulsar Other y seleccionarlo, la próxima vez aparecerá en el menú directamente). A la clase le podeis llamar SumaTest, aseguraros de seleccionar JUnit4.



Ahora ya podeis crear la clase con el siguiente código.

view plain copy to clipboard print ?

```
01. package fachadatests;
02.
03. import static org.junit.Assert.assertTrue;
04.
05. import org.junit.After;
06. import org.junit.Before;
07. import org.junit.Test;
08.
09. import com.miguelpuig.blzarovich.junittutorial.fachada.Suma;
10.
11. public class SumaTest {
12.     public Suma suma;
13.
14.     @Before
15.     public void antesDelTest() {
```

```
16.         /**
17.          * El metodo precedido por la etiqueta @Before
18.          * es para indicar a JUnit que debe ejecutarlo
19.          * antes de ejecutar los Tests que figuran en
20.          * esta clase.
21.          */
22.         this.suma = new Suma(3, 5);
23.     }
24.
25.     @After
26.     public void despuesDelTest() {
27.         /**
28.          * La etiqueta @After es la antítesis de @Before.
29.          * Simplemente este metodo se ejecutara despues de
30.          * ejecutar todos los tests de esta clase.
31.          */
32.         // en este caso no hago nada, solo esta de ejemplo
33.     }
34.
35.     @Test
36.     public void testSuma() {
37.         /**
38.          * Marcais el metodo con la etiqueta @Test y es
39.          * importante que el nombre del metodo comience
40.          * siempre por test.
41.          */
42.         int resultado = this.suma.suma();
43.         // con esto verificamos que el resultado es el esperado
44.         assertTrue(resultado == 8);
45.     }
46.
47.     @Test
48.     public void testResta() {
49.         int resultado = this.suma.resta();
50.         assertTrue(resultado == -2);
51.     }
52. }
```

Ahora con la multiplicación, sólo que sin before y after. Ya que es una tontería para una clase así, y en la clase anterior sólo se pusieron a modo de demostración. Before y After en realidad están geniales para
cuando realizas tests de unidad al nivel de acceso a datos de una aplicación. Ahí en el Before prepararías
la base de datos para realizar las pruebas y en After desharías los cambios simplemente.

view plain copy to clipboard print ?

```
01. package fachadatests;
02.
03. import static org.junit.Assert.assertTrue;
04.
05. import org.junit.Test;
06.
07. import com.miguelpuig.blzarovich.junittutorial.fachada.Multiplicacion;
08.
09. public class MultiplicacionTest {
```

```
10.     /**
11.      * Lo mismo de antes pero esta vez sin
12.      * before y after.
13.      */
14.
15.     @Test
16.     public void testMultiplicar() {
17.         Multiplicacion multiplicacion = new Multiplicacion(3, 5);
18.         int resultado = multiplicacion.multiplicar();
19.         assertTrue(resultado == 15);
20.     }
21. }
```

Ahora finalmente creamos una clase para la División, será exactamente igual que la anterior pero con la división. Además como la Division está en la fachada2, pués el Junit Test Case lo crearemos en fachada2 de la carpeta de tests.

view plain copy to clipboard print ?

```
01. package fachada2tests;
02.
03. import static org.junit.Assert.assertTrue;
04.
05. import org.junit.Test;
06.
07. import com.miguelpuig.blzarovich.junittutorial.fachada2.Division;
08.
09. public class DivisionTest {
10.     @Test
11.     public void testDivide() {
12.         Division division = new Division(4, 2);
13.         int resultado = division.divide();
14.         assertTrue(resultado == 2);
15.     }
16. }
```

Bien, ya tenemos todos los tests de unidad. Ahora pulsando botón derecho sobre las clases de los tests podreis lanzar los tests pulsando Run as JUnit Test. Yo en principio aún no lo voy a hacer, lo reservo para el final.

Si lo probasteis, podreis observar que igual puede ser un coñazo tener que ejecutar cada Test Case por separado, uno a uno. Esto se puede evitar. Para ello vamos a agrupar los Tests de fachada1 en uno sólo que los ejecutará automáticamente.

Para ello creareis la clase *FachadaTests*. Que simplemente necesitará de la declaración de la clase, que no tendrá nada dentro. Esta clase estará marcada por dos etiquetas *RunWith* que le indicará con que clase del framework debe cargar esta clase; y *SuiteClasses* que tendrá una lista de las clases que son Tests o SuiteClasses. Esto implica que puedes meter SuiteClasses dentro de SuiteClasses y así poder organizar jerárquicamente tus tests.

view plain copy to clipboard print ?

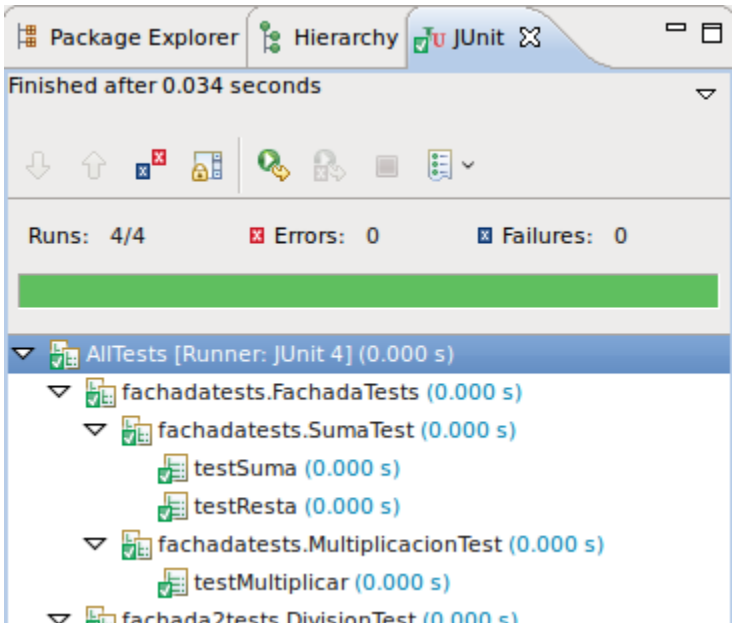
```
01. package fachadatests;
02.
03. import org.junit.runner.RunWith;
04. import org.junit.runners.Suite.SuiteClasses;
05. import org.junit.runners.Suite;
06.
07. @RunWith(Suite.class)
08. @SuiteClasses({SumaTest.class, MultiplicacionTest.class})
09. public class FachadaTests {}
```

Así de simple queda la clase. Ahora realizaremos una clase similar a esta en el raiz de la carpeta tests. Para ejecutar todos los tests conjuntamente, esta se llamará AllTests. Os la meterá en el default package, tranquilos.

view plain copy to clipboard print ?

```
01. import org.junit.runner.RunWith;
02. import org.junit.runners.Suite;
03. import org.junit.runners.Suite.SuiteClasses;
04.
05. import fachada2tests.DivisionTest;
06. import fachadatests.FachadaTests;
07.
08. @RunWith(Suite.class)
09. @SuiteClasses({FachadaTests.class, DivisionTest.class})
10. public class AllTests {}
```

Como veis en el SuiteClasses podeis mezclar tests simples y TestSuite :D. Ya teneis vuestros tests preparados para ejecutar por lote. Vamos ahora a pulsar botón derecho en *AllTests* y seleccionar Run as JUnit Test. Automáticamente el Eclipse lanzará todos los tests que teneis programados y os aparecerá un resultado parecido a esto.



Como veis todos los tests salieron con resultados satisfactorios. Ahora os recomiendo jugar y probar vosotros mismos con todo esto, meter en los assertTrue resultados no esperados, para que veais que pasa cuando falla, añadir más clases, hacer pruebas más complejas o probar toda la colección de asserts que podeis utilizar (assertEquals, assertNull, assertNotNull, etc).

Código fuente del proyecto utilizado como ejemplo

Links relacionados:

Página oficial de JUnit

Página oficial de Eclipse

This entry was posted in Desarrollo, Tutorial and tagged eclipse, java, junit on January 31, 2010 [http://blog.zarovich.org/index.php/2010/01/junit-en-eclipse/] .

3 thoughts on “Tutorial JUnit en Eclipse”

panchof
October 10, 2012 at 13:33

Buenisimo tutorial muchas gracias!!

yveca
June 19, 2013 at 00:40

Muchas gracias por el tutorial, fue de gran ayuda

Albert Ruiz
June 12, 2014 at 09:39

Muy práctico para iniciarme en JUNIT.

Muchas gracias.
