UNIVERSIDAD DEL QUINDÍO
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Información general	
Fecha :	2015-02-26
Duración estimada en minutos:	60
Docente:	Christian Andrés Candela
Guía no.	7
Nombre de la guía:	Junit para DAOs

Información de la Guía

Objetivos: Aprender a usar el framewor Junit.

Conceptos Básicos: Manejo de Eclipse, Java, Bases de Datos, JDBC, DataSource, Entidades y Glassfish.

Contextualización Teórica: Dentro del desarrollo de software las pruebas de programador cumplen con una importante función a la hora de comprobar el correcto funcionamiento de la aplicación así como de encontrar y elminar los defectos que se puedan haber generado durante la etapa de codificación. Sin embargo entre más grande sea la aplicación, más modificaciones deban hacerce a funcionalidades ya codificadas y probadas, así como el trabajo en equipo sobre una misma funcionalidad hace necesario formas más rápidas y eficientes que permitan probar el correcto funcionamiento de la aplicación.

Java provee mecanismos que permiten realizar verificaciones dentro del código conocidos como assertions.

Assert: Es una sentencia Java usada para expresar y verificar una afirmación. El assert se puede usar con uno o dos parametros.

assert expresion_booleana;

```
assert 7 < 10;

int a = 7;

….

assert a < 10;
```

La expresión booleana representa la afirmación que se desea verificar, si el resultado de la expresión es verdadero, la ejecición del programa continuara de forma normal, en caso de que elresultado de la expresión sea falso, la ejecución se interrunpira para generar un AssertionError.

assert expresion_booleana : expresion_resultante;

```
assert 7 < 10 : “Afirmación invalida”;

int a = 7;
String mensaje = “Afirmación invalida”;
….

assert a < 10 : mensaje;
```

Esta forma funciona igual a la anterior, se diferencia por el uso de una expresión resultante, la cual debe ser un valor diferente de **void**. Bien puede ser una variable, expresión booleana, matemática o incluso el llamado a un método que retorne un valor. En caso de que la expresión booleana sea falfo, el valor resultante de evaluar la segunda expresión es convertido a String, y pasado por parámetro al AssertionError generado por el assert.

JUNIT

JUnit es un frameqork basado en los assert creado por or Erich Gamma y Kent Beck. JUnit es usado para la automatización de las pruebas unitarias. Junit permite la ejecución y comprobación de las funcionalidad de las clases de un proyecto de manera controlada. Todo ello con el fin de evaluar si los métodos de la clase funcionan correctamente. Para evaluar el funcionamiento de los métodos se brindan datos de entrada a cada uno de los métodos para los cuales se conoce la respuesta correcta, se compara el resultado de la ejecución con el resultado esperado, en caso de que los resultados coincidan la prueba es marcada como exitosa, en caso contrario se dice que la prueba fallo.

A partir de la versión 4 de JUnit se incluyen un conjunto de anotaciones basadas en Java 5 que facilitan mucho el desarrollo de las pruebas.

@Test: Permite marcar los métodos que serán usados como pruebas. Por medio de parametros es posible indicar a la prueba una duración maxíma, en cuyo caso si la prueba dura más del tiempo establecido la prueba fallará. Tambien es posible determinar un resultado específico por ejemplo una excepción, en cuyo caso si la prueba no arroja dicha excepción la prueba fallará.

```
@Test public void method() {
        ...
}

@Test(timeout=100) public void method2() {
        ...
}

@Test(expected = Exception.class) public void method3() {
        ...
}
```

@Before: Permite marcar los métodos que deberán ser ejecutados antes de cada uno de los métodos de prueba @Test. Generalmente los métodos @Before son usados para la inicialización de datos a ser usados en la prueba.

```
@Before public void method() {
```

```
        ...
}
```

@After: Permite marcar los métodos que deberán ser ejecutados despues de cada uno de los métodos de prueba @Test. Generalmente los métodos @After son usados para liberar los recursos usados en la prueba.

```
@After public void method() {
        ...
}
```

@BeforeClass: Permite marcar los métodos estaticos que deberán ser ejecutados antes de iniciar las pruebas de una clase. El método @BeforeClass es ejecutado una unica vez antes de todas las pruebas de una clase. Generalmente los métodos @BeforeClass son usados para la inicialización de datos comunes a ser usados por las prueba.

```
@BeforeClass public static void method() {
        ...
}
```

@AfterClass: Permite marcar los métodos estaticos que deberán ser ejecutados despues de ejecutar las pruebas de una clase. El método @AfterClass es ejecutado una unica vez despues de todas las pruebas de una clase. Generalmente los métodos @AfterClass son usados para liberar los recursos comunes usados por las prueba de una clase.

```
@AfterClass public static void method() {
        ...
}
```

@Ignore: Permite marcar los métodos de prueba @Test que por alguna razón no se desean ejecutar.

```
@Ignore @Test public static void method() {
        ...
}
```

Sentenciar assert en JUnit

JUnit proporciona un conjunto de sentencias assert encaminadas a la declaración de diferentes tipos de afirmaciones. Estas sentencias están disponibles como métodos de la clase Assert.

Fail: Provoca que la prueba falle. Recibe por parámetro el mensaje de error que se debe generar.

```
Assert.fail("Error al ejecutar la prueba");
```

assertTrue: Verifica que la expresión booleana propocionada sera verdadera, en caso de ser falsa la prueba falla. Además de la expresión booleana también es posible proporcionar un mensaje de error.

```
Assert.assertTrue(expresionBooleana);
```

Assert.assertTrue("Prueba fallida",expresionBooleana);

assertEquals: Verifica que dos objetos dados sean iguales, en caso de no serlos la prueba falla. Además de los objetos también es posible proporcionar un mensaje de error.

Assert.assertEquals(resultadoEsperado,resultadoReal);

Assert.assertEquals("El resultado real no coincide con el esperado",resultadoEsperado,resultadoReal);

assertNull: Verifica que el objeto dado sea null, en caso de no serlo la prueba falla. Además del objeto también es posible proporcionar un mensaje de error.

Assert.assertNull(objeto);

Assert.assertEquals("El objeto no es nulo",objeto);

Como los métodos anteriores, la clase Assert posee otros tantos metodos assert encaminados a verificar la validez de las afirmaciones propuestas. Puede encontrar un listado de estos método en la Api de JUnit (http://kentbeck.github.com/junit/javadoc/latest/).

Otra fuente bibliográfica que puede ser usada es: http://www.vogella.de/articles/JUnit/article.html

Arquilliam es un framework open source que facilita la elaboración de pruebas. Permite a los desarrolladores verificar el comportamiento del código brindando un ambiente de ejecución, dando la oportunidad al desarrollador de centrarse en las pruebas de comportamiento de la aplicación.

Un caso de prueba de arquilliam debe tener 3 elementos:

- La clase de prueba debe tener la anotación `@RunWith(Arquillian.class)`
- Un método estático anotado con `@Deployment` que retorne un elemento de tipo Archive (ShirnkWrap).
- La clase debe tener al menos un método marcado con la anotación `@Test`

Adicionalmente, los métodos que requieran el uso de transacciones deberán ser anotados con `@Transactional(value=TransactionMode.ROLLBACK)`, el atributo value puede tomar uno de los siguientes valores:

COMMIT
DEFAULT
DISABLED
ROLLBACK

Puede buscar más información en la página http://arquillian.org

Precauciones y Recomendaciones: Recuerde verificar que el servidor de aplicaciones soporte el motor de base de datos que usará, de igual forma debe verificar que eclipse este asiendo uso del JDK y no del JRE y recuerde adicionar al workspace el servidor de aplicaciones Glassfish antes de crear cualquier proyecto. También puede ser importante verificar que los puestos usados por Glassfish no estén ocupados (Para ello puede hacer uso del comando ***netstat -npl*** o ***netstat -a***)
Artefactos: Se requiere tener instalado el JDK y un IDE para el desarrollo de aplicaciones (Eclipse Luna), un servidor de aplicaciones que cumpla con las especificaciones de JEE, para esta práctica Glassfish y el motor de base de datos Mysql.
Evaluación o Resultado: Se espera que el alumno pueda hacer uso de Junit para la realización de las pruebas unitarias.

Procedimiento

1. Cree un proyecto de tipo maven project, use la versión 0.2.0

2. Adicione a su proyecto las entidades y Daos creadas en las guías anteriores, si lo prefiere puede hacer uso del proyecto creado anteriormente.

3. En el elemento Java Resource de su proyecto encontrará varias carpetas para su código fuente. Entre las carpetas verá la src/test/java, use dicha carpeta para crear las pruebas unitarias para sus daos.

4. Dentro del nuevo folder Cree un paquete para la realización de las pruebas.

5. Cree un nuevo Junit Test Case. Para ello acceda al menú new – other – Junit . Asignele al Junit el nombre de su preferencia y en el campo class under test especifique la clase que se desea probar (Uno de los DAOs específicos creado en clases anteriores). En la ventana siguiente seleccione los métodos que desea probar. Para nuestro ejemplo deberá seleccionar el método adicionar o insertar de su DAO.

6. Como resultado del paso anterior se debe haber creado una nueva clase con uno o varios métodos ***testMetodo***. Remueva de cada uno de dichos métodos la invocación del método fail.

7. Adicione a su clase TEST la anotación @RunWith(Arquillian.class)

```
@RunWith(Arquillian.class)
public class EntidadDaoTest extends TestCase {
…
}
```

8. Adicione a su clase TEST una instancia del Dao de su entidad. Si su Dao se llama PersonaDao cree una instancia, a través de la cual accederá a su Dao.

```
@Inject
private PersonaDao personaDao;
```

9. Cree un método estático marcado con la anotación @Deployment que retorne un elemento de tipo Archive<?>

```
@Deployment
```

```
public static Archive<?> createTestArchive() {
     return ShrinkWrap
               .create(WebArchive.class, "test.war")
               .addClasses(Pais.class, Dao.class, PaisDao.class,
                         Resources.class)
               .addAsResource("META-INF/persistence.xml",
                              "META-INF/persistence.xml")
               .addAsWebInfResource(EmptyAsset.INSTANCE, "beans.xml");

}
```

Como podrá ver el método addClasses le permite determinar que clases usará para la ejecución de las pruebas. En este punto remplace Pais.class y PaisDao por las clases que correspondan.

10. Adicione a su método TEST la anotación @Transactional(value=TransactionMode.ROLLBACK), esto permitirá que al finalizar la transacción los cambios sean reversados.

```
@Test
@Transactional(value=TransactionMode.ROLLBACK)
public void testInsert() {

}
```

11. En su método @Test cree una instancia de la entidad administrada por su DAO (Declare una variable de tipo Entidad y cree la instancia con **new**) y asigne a los atributos de la instancia los valores que desea almacenar en la base de datos (Para esto puede hacer uso de los métodos set de su entidad).

```
Persona nuevaPersona = new Persona( );
nuevaPersona.setCedula( “1952830813” );
nuevaPersona.setNombre( “Pedro Perez” );
```

12. Invoque el método insertar del Dao enviando como parámetro la instancia de la entidad creada previamente. (También dentro del método test)

```
nuevaPersona = personaDao.insert(nuevaPersona);
```

13. Para probar el funcionamiento del Junit de clic derecho sobre la clase, acceda al menú Run As y seleccione la opción Junit Test.

14. Verifique que se haya ejecutado correctamente. Para ello verifique que la entidad que creo se haya almacenado en la base de datos.

15. Adicione a su método de prueba las afirmaciones (***assert***) que le permitan verificar de forma automática que efectivamente se creo el registro en la base de datos.

```
Persona personaRegistrada = personaDao.findByKey(nuevaPersona.getCedula());
Assert.assertEquals( personaNueva , personaRegistrada );
```

NOTA: Para poder comparar dos entidades es necesario que la entidad Persona tenga reescritos los métodos

equals.

16. Cree los métodos de prueba que le permitan verificar el funcionamiento de los demás métodos del DAO.