Ejercicios de repaso

S07-P07

Supongamos que tenemos que implementar tests para el DAO AlumnoDAO, concretamente para el método **void addAlumno(Alumno alu)**. Contesta a las siguientes preguntas, justificando claramente tus respuestas:

a) Justifica por qué podemos realizar pruebas de integración sobre dicho método.

Porque tiene una dependencia externa que no es un método de una clase java, es una Base de Datos. Si no creamos dobles de nuestras dependencias externas estaremos haciendo pruebas de integración. Ya que estaremos probando la interacción entre las distintas unidades que conforman nuestro proyecto.

b) Indica qué librerías, de las vistas en clase, necesitarías para implementar dichos tests y cómo configurarías el pom del proyecto con dichas librerías. Justifica claramente para qué necesitas cada una de las librerías.

Las librerías nombradas a continuación serán las que necesitaremos para implementar duchos tests y se situarían en el apartado <dependencies> del POM:

- DBUnit: La utilizaremos para poder usar el API de DBUnit en los tests.
- JUnit: La emplearemos para implementar y ejecutar los tests.
- Mysql: Sin esta dependencia no podremos hacer operaciones con la BD.

Tanto en DBUnit como en JUnit pondré el <scope> test porque esas dos dependencias sólo se usarán en src/test.

c) Muestra el código java para implementar un test de integración suponiendo que el objeto Alumno está formado por los atributos id (de tipo int), nombre, dirección, estos dos últimos de tipo String, y que la base de datos contiene la tabla alumno, con los campos alu_id, alu_nombre, alu_direccion. El método añade los datos del alumno en la citada tabla. Supón también que la tabla alumno inicialmente está vacía.

```
private IDatabaseTester databaseTester;

// Recuerda que los métodos apuntados con @Before se llaman antes de ejecutar cada método
// apuntado con @Test
@Before
public void setUp() throws Exception {
        // fijamos los datos para acceder a la BD
        databaseTester = new JdbcDatabaseTester("jdbc:mysql://localhost:3306/ejercicio",
        "root", "ppss");
        // inicializamos el dataset
        DataFileLoader loader = new FlatXmlDataFileLoader();
        IDataSet dataSet = loader.load("/alumno-init.xml"); // tabla de alumnos VACIA
        databaseTester.setDataSet(dataSet);
        // llamamos a la operación por defecto setUpOperation
        databaseTester.onSetup();
}

@Test
public void testInsert() throws Exception {
        AlumnoTO alumno = new AlumnoTO();
        alumno.setID(1);
        alumno.setNombre("Elena Aguirre Juarez");
        alumno.setDireccion(“Calle 1”);

        // método a probar
        new FactoriaDAO().getAlumnoDAO().addAlumno(alumno);

        // recuperamos la conexión con la BD
        IDatabaseConnection connection = databaseTester.getConnection();

        // obtenemos los valores reales de la tabla a partir de la conexión
        IDataSet databaseDataSet = connection.createDataSet();
        ITable actualTable = databaseDataSet.getTable("alumno");

        // determinamos los valores esperados desde alumno-expected.xml
        DataFileLoader loader = new FlatXmlDataFileLoader();
        IDataSet expectedDataSet = loader.load("/alumno-expected.xml");

        ITable expectedTable = expectedDataSet.getTable("alumno");

        // determinamos los valores esperados desde alumno-expected.xml
        Assertion.assertEquals(expectedTable, actualTable);
}
```

S08-P08

Contesta a las siguientes cuestiones:

a) Dados los tests implementados en prácticas con Selenium IDE, indica con cuál de los dos métodos de diseño de casos de prueba vistos en clase hemos obtenido dichos tests. Justifica claramente tu respuesta.

Hemos visto un modelo basado en escenarios, ya que nuestros test se han basado en un conjunto de requerimientos en un mismo escenario. Además describe un funcionamiento normal de la aplicación (crear usuario, comprar mascota...)

b) Podemos integrar la ejecución de los tests citados en el apartado a) con la construcción del proyecto? Justifica claramente tu respuesta, indicando, en cada caso, por qué no podemos hacerlo, si tu respuesta es negativa, y cómo podemos hacerlo si tu respuesta es afirmativa.

No se puede, ya que se trata de comandos Selenese, los cuales en nuestro entorno de prácticas los hemos implementado en un plugin de firefox que permite crear scripts de control (action, accessors y assertions) sobre páginas web.

c) Pon un ejemplo de caso de prueba diseñado con el método basado en escenarios para realizar una prueba de uso Bitbucket, de forma que comprobemos que la práctica P08, que está formada por 2 ejercicios se ha subido correctamente (el escenario debe contemplar las acciones previas a la consulta en Bitbucket). Implementa el driver usando comandos selenese.

Diseño: Nuestra *Base URL* será https://bitbucket.org/ una vez dentro pinchamos en "Log in", una vez estemos dentro de la página de logeo verificamos que efectivamente podemos ver en ésta el texto "Log in". Introducimos nuestros datos (correo y contraseña) y pinchamos en "Log in". A continuación se abrirá nuestro dashboard, verificaremos que nuestro repositorio existe: "ppss-Gx-ap1-ap2-20xx". Una vez lo hemos verificado, pinchamos en el link de nuestro repositorio. Una vez dentro, verificamos que aparece el texto "Source" en la nueva página que hemos abierto y pinchamos en el mismo. Una vez dentro de la siguiente página, verificaremos que el texto "P08" existe, a continuación pinchamos en "P08". Una vez dentro de la siguiente página verificamos que aparece el texto "ejercicio1.html" al igual que el texto "ejercicio2.html".

Implementación:

Comand	Target	Value
clickAndWait	link=Log in	
verifyText	//input[@value='Log in']	
type	id=js-email-field	test@test.com
type	id=js-password-field	password123
clickAndWait	//input[@value='Log in']	
verifyText	link=ppss-Gx-ap1-ap2-20xx	ppss-Gx-ap1-ap2-20xx
clickAndWait	link=ppss-Gx-ap1-ap2-20xx	
verifyText	link=Source	Source
clickAndWait	css=Source	
verifyText	link=P08	P08
clickAndWait	link=P08	
verifyText	link=ejercicio1.html	ejercicio1.html
verifyText	link=ejercicio2.html	ejercicio2.html

S09-P09

Contesta a las siguientes cuestiones:

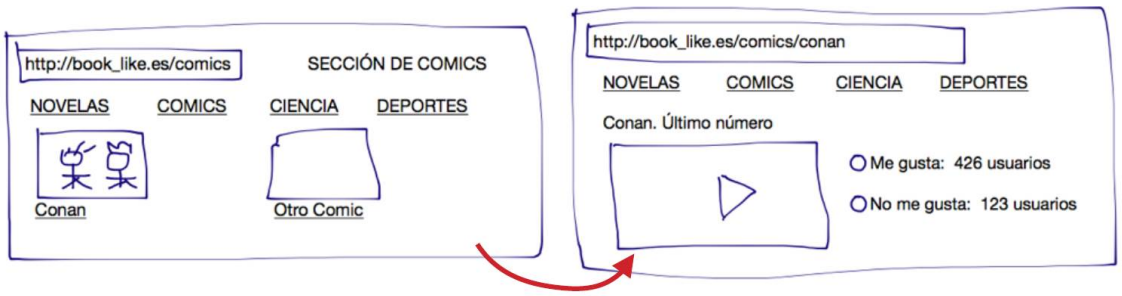
a) Dados los tests implementados en prácticas con Selenium IDE ¿Podríamos haber implementado esos mismos tests con Selenium WebDriver? Justifica claramente tu respuesta.

Si, ya que WebDriver nos da las herramientas necesarias para crear tests con la misma finalidad que Selenium IDE, además cuenta con la gran ventaja de tratarse de un lenguaje de programación, por lo que dispone de ventajas como la reusabilidad de código o el hecho de poder iterar en un bucle.

b) ¿Podemos integrar la ejecución de los tests anteriores (que usan Selenium WebDriver) con la construcción del proyecto? Justifica claramente tu respuesta, indicando por qué no podemos hacerlo, si tu respuesta es negativa, y cómo podemos hacerlo si tu respuesta es afirmativa.

Si, necesitaremos hacer uso de la librería JUnit al igual que de la librería WebDriver, para poder implementar tests (anotados con la etiqueta @Test). Será necesario que añadamos la dependencia de Selenium-java en nuestro pom.xml para poder usar WebDriver.

Supón que accedemos a un sitio web (http://books_like.es/comics), que nos muestra la sección de cómics, estamos interesados en leer el último número del comic de "Conan". En esa página podemos leer cómics on-line, y además podemos ver el número de opiniones "me gusta" y "no me gusta" realizados de forma anónima por los usuarios que han leído el cómic. Cada vez que un usuario "pincha" sobre el icono "me gusta" o "no me gusta" (solamente puede activarse uno cada vez) se incrementa en uno el contador correspondiente...



Implementa el test anterior utilizando Selenium WebDriver y el patrón page Object (no es necesario que implementes los métodos de las page Object, pero tienes que indicar para dichas clases, los atributos y las signaturas de los métodos que contienen).

PageFactory prototipado:

ComicsMain.java

```
WebDriver driver;
@FindBy(linkText="Conan") WebElement ConanPage;
/**
* Para getComicsMainTitle() no necesitamos otro WebElement ya que podemos
* hacer driver.getTitle();
*/
public ComicsMain(WebDriver driver);     // Establece la primera conexión
public String getComicsMainTitle();      // Usada para validación
public Conan gotoConan();                // Redirige a Conan y devuelve instancia
```

Conan.java

```
WebDriver driver;
@FindBy(name=”vote”) List<WebElement> voto;
/**
* Podemos emplear el WebElement voto para recoger el número de votos en
* getCurrentVotos()
*/
public Conan(Webdriver driver);          // this.driver = driver;
public String getConanMainTitle();       // Devuelve el titulo de la pagina
public int getCurrentMGVotos();          // Devuelve el número de votos positivos
public int getCurrentNMGVotos();         // Devuelve el número de votos negativos
public Conan setVoto(boolean bMeGusta); // Vota en el radial button list
```

Implementación del test:

```
public class TestComics {
    WebDriver driver;
    ComincsMain poComics;
    Conan poConan;

    @Before
    public void setup(){
        driver = new FirefoxDriver();
        poComics = PageFactory.initElements(driver, ComincsMain.class);
    }

    @Test
    public void test_VoteSystem(){
        String comicsPageTitle = poComics.getComicsMainTitle();
        Assert.assertTrue(comicsPageTitle.toLowerCase().contains("sección de comics"));
        poConan = poComics.gotoConan();

        Pausa.milisegundos(2000);
        String conanPageTitle = poConan.getConanMainTitle();
        Assert.assertTrue(comicsPageTitle.toLowerCase().contains("conan"));

        // Votamos positivamente
        int pastMGVotes = poConan.getCurrentMGVotos();
        int pastNMGVotes = poConan.getCurrentNMGVotos();
        poConan.setVoto(true);
        Pausa.milisegundos(1000);
        int currentMGVotes = poConan.getCurrentMGVotos();
        int currentNMGVotes = poConan.getCurrentNMGVotos();
        // Hemos aumentado en 1 la cantidad de votos positivos y en 0 los negativos
        Assert.assertEquals((pastMGVotes + 1), currentMGVotes);
        Assert.assertEquals(pastNMGVotes, currentNMGVotes);

        // Intentamos volver a votar positivamente
        pastMGVotes = poConan.getCurrentMGVotos();
        pastNMGVotes = poConan.getCurrentNMGVotos();
        poConan.setVoto(true);
        Pausa.milisegundos(1000);
        currentMGVotes = poConan.getCurrentMGVotos();
        currentNMGVotes = poConan.getCurrentNMGVotos();
        // No debe de subir porque ya hemos votado previamente
        Assert.assertEquals(pastMGVotes, currentMGVotes);
        Assert.assertEquals(pastNMGVotes, currentNMGVotes);
    }

}
```