Procesos de la Ingeniería Software
Tema 4

Soporte Java para construcción de aplicaciones empresariales

6. Pruebas en aplicaciones Java EE

Pruebas en aplicaciones Java EE

- La prueba de aplicaciones Java EE resulta compleja por la dependencia de los servicios del contenedor

- Pruebas Unitarias o **StandAlone**
  - Usando frameworks estilo **JUnit** gracias a la naturaleza POJO de los componentes
    - EJBs, Managed Beans, etc.
  - Utilización de objetos **Mock** para las dependencias
    - Tanto de otros EJBs como de recursos del entorno (Entity Manager, etc.)

- Pruebas de Integración
  - Deben ser realizadas en un contenedor para verificar que funcionan sus servicios (e.g. DI)
  - Pruebas de integración por capas o de componentes individuales (EJBs o Managed Beans)
    - Prueba en **contenedor embebido**
      - Java EE proporciona capacidad para el lanzamiento de EJBs en un contenedor embebido creado y lanzado desde la propia clase Junit
    - Prueba en **contenedor real**
      - Frameworks estilo **Arquillian**, que permiten lanzar el contenedor real desde el propio test
  - Pruebas de integración basadas en funcionalidad (end-to-end)
    - Frameworks estilo **Selenium** (para aplicaciones web) o FEST (para aplicaciones escritorio)

EJBContainer: Contenedor embebido

- **EJBContainer** es el contenedor embebido que proporciona Java EE

- Un contenedor embebido permite el despliegue de EJBs
  - y aunque se lanza desde un entorno Java SE (misma JVM que el cliente),
  - proporciona los mismos servicios que el contenedor real (inyección de dependencias, transacciones, seguridad, etc.)

- El contenedor embebido facilita labores de testing
  - Permite lanzar el contenedor desde las propias clases de test JUnit
  - Permite la depuración (en el contenedor real no es posible desde una clase JUnit)

- Funcionamiento básico:
  - Cuando se crea un EJBContainer, se despliegan en él por defecto todos los EJBs que estén en el "classpath"
    - Todos los EJB del proyecto o proyectos referenciados por dependencia si lo lanzamos desde Maven
    - Todos los EJB de un proyecto (o de proyectos referenciados a través del Build Path) si lo lanzamos desde eclipse
  - Funcionamiento en Maven => Añadir la dependencia a `glassfish-embedded-all`
  - Funcionamiento en Eclipse => Añadir la librería glassfish-embedded-static-shell.jar (disponible en la distribución de glassfish) al BuildPath del proyecto

Ejemplo de uso del EJBContainer en un JUnit Test Case

```
public class CalculadoraTest {

     private static EJBContainer ec;
     private static CalculatorEJBRemote calcu;

    @BeforeClass
    public static void initContainer() throws Exception {
        //Creamos el contenedor embebido, por defecto, inicializa todos los EJB que estén en el classpath del cliente
        ec = EJBContainer.createEJBContainer();
        // Buscamos el EJB a través de JNDI
        // El scope varía según se lance desde Eclipse o desde Maven.
        // El que se muestra es el de Maven (java:global/ejb-app/<NombreArchivoJAR>/<NombreEJB>
        calcu = (CalculatorEJBRemote) ec.getContext().lookup(
                        "java:global/ejb-app/calculadora2-ejb-0_0_1-SNAPSHOT/CalculatorEJB");
    }

    @AfterClass
    public static void closeContainer() throws Exception {
        //Cerramos el contenedor (Importante)
        if (ec != null) {
            ec.close();
        }
    }

    @Test
    public void testAdd() {
         assertTrue(calcu.add(2, 3) == 5);
    }
}
```

Ejemplo de uso de EJBContainer para prueba de componentes DAO

```
public class PeliculasDAOBeanTest {

    private static EJBContainer ec;
    private static PeliculasDAORemote f;
    private static Pelicula nuevaPeli;

    @BeforeClass
    public static void initContainer() throws Exception {

        Map properties = new HashMap();
        properties.put(EJBContainer.MODULES, new File[]{new File("classes")});

        properties.put("org.glassfish.ejb.embedded.glassfish.installation.root",
                       "C:/glassfish4/glassfish");

        ec = EJBContainer.createEJBContainer(properties);
        f = (PeliculasDAORemote) ec.getContext().lookup("java:global/ejb-app/PeliculasDAO/PeliculasDAOBean");

        nuevaPeli = new Pelicula("UnaPelicula","2015", "Terror");
    }

   @AfterClass
    public static void closeContainer() throws Exception {
        if (ec != null) {
            ec.close();
        }
    }

    @Test
    public void testCreatePelicula() {
        Long id = f.createPelicula(nuevaPeli);
        assertTrue(f.pelicula(id).equals(nuevaPeli));
    }
}
```

Instalación de glassfish, para que sepa dónde consultar los datos de la BBDD