



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

Información general	
Actualizada por:	Carlos Andrés Flórez Villarraga
Duración estimada en minutos:	90
Docente:	Christian Andrés Candela y Einer Zapata
Guía no.	11
Nombre de la guía:	EJB de Sesión sin Estado

Información de la Guía

OBJETIVO

Comprender que son los EJB de sesión y aprender a crearlos y configurarlos para el desarrollo de la lógica de negocio (capa de negocio de la aplicación).

CONCEPTOS BÁSICOS

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

CONTEXTUALIZACIÓN TEÓRICA

Los EJB de sesión sin estado pertenecen al API de Java “Enterprise Java Beans” (EJB) la cual forma parte del conjunto de APIs usadas para la construcción de aplicaciones empresariales. Los EJB constituyen un modelo de componentes de carácter distribuido del lado del servidor.

Características de los EJB:

- Los EJB por lo general contienen lógica de negocio que opera sobre los datos de la aplicación.
- Una instancia de un EJB es manejada en tiempo de ejecución por un contenedor (Los servidores de aplicación como JBoss y GlassFish hacen las veces de contenedores).
- Los EJB pueden distribuirse entre varias máquinas y se puede acceder a ellos de forma remota como si estuvieran en una máquina local.
- Los EJB son manejados de forma centralizada por un contenedor.
- Varios aspectos de la configuración de los EJB, tales como los atributos de las transacciones y la seguridad, se puede especificar junto con la lógica de negocio de la clase por medio de anotaciones, o por separado, en un descriptor de despliegue XML.
- El acceso de los clientes al EJB es mediado por el contenedor en el que el EJB es desplegado.
- Si un EJB es definido usando solo los aspectos de carácter estándar definidos por la especificación EJB podrá ser desplegado en cualquier contenedor que cumpla con dicho estándar. De forma contraria, si al momento de definir el EJB se usan características adicionales proporcionadas por un contenedor específico el EJB solo podrá ser desplegado en dicho contenedor o en contenedores que provean dicha característica.

Dentro de los tipos de EJB podemos encontrar los Session Object (EJB de sesión): Normalmente un Session Object tiene las siguientes características:

- Son transaccionales

- Pueden actualizar los datos en la base de datos.
- No representan directamente la base de datos, a pesar de que puede acceder y actualizar los datos.
- Tienen una vida relativamente corta.
- El EJB es removido en caso de que el contenedor falle. El cliente debe volver a establecer un nuevo objeto de sesión para continuar con la ejecución.

Los Session Object se subdividen en stateful, stateless y singleton. Pueden ser accedidos por medio de sus interfaces remotas (desde otra maquina virtual), locales (desde la misma maquina local) o directamente sin usar interfaz. Todos los Session Object soportan ejecución asíncrona en todas sus interfaces.

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 puertos usados por Glassfish no estén ocupados (Para ello puede hacer uso del comando **netstat -npl** o **netstat -a**)

ARTEFACTOS

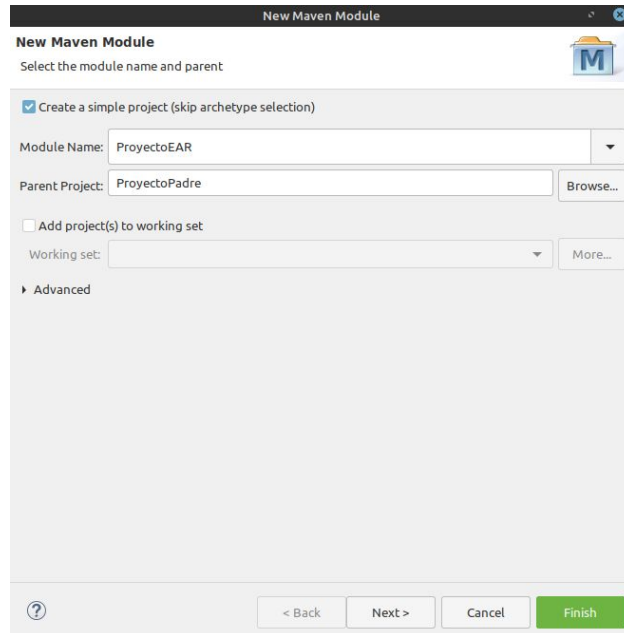
Se requiere tener instalado el JDK y un IDE para el desarrollo de aplicaciones (Eclipse JEE en su última versión), un servidor de aplicaciones que cumpla con las especificaciones de JEE, para esta práctica Glassfish y el motor de base de datos Mysql.

EVALUACIÓN O RESULTADO

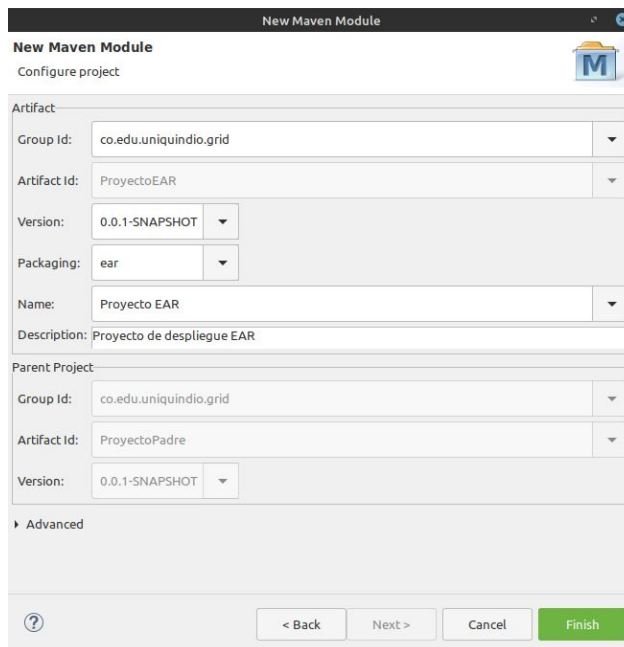
Se espera que el alumno pueda crear y usar exitosamente un EJB de sesión para la creación de la lógica de negocio de las aplicaciones.

Procedimiento

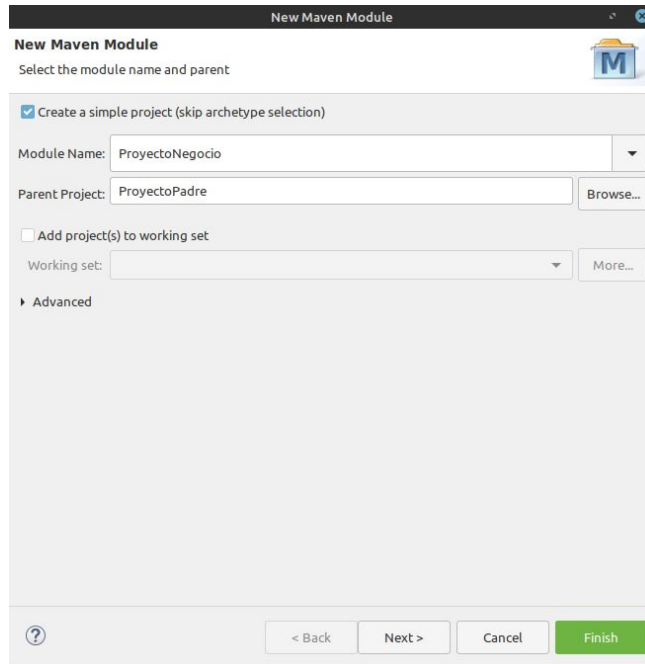
1. Para el desarrollo de esta guía necesitara una base de datos en MySQL, un proyecto de tipo Maven – POM, el cual contenga un proyecto maven con soporte para el uso de JPA y otro proyecto maven configurado para la realización de pruebas. Y una conexión a dicha base de datos para ser usada en la generación de las tablas.
2. Asegúrese de tener al menos una entidad en su proyecto JPA, de no tenerla deberá crearla.
3. Adicione un nuevo módulo a su proyecto padre. Un módulo en un proyecto padre no es otra cosa que otro proyecto maven, el cual hará parte del proyecto padre y heredará de él algunas de sus características. Adicione un nuevo módulo en el proyecto padre llamado ProyectoEAR. El proyecto EAR será el encargado de desplegar el proyecto en el servidor.



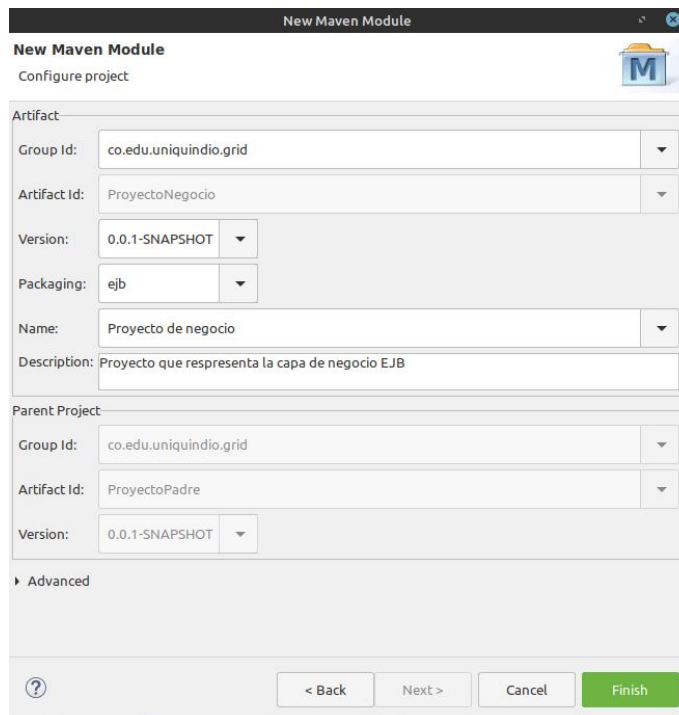
Next, Deberá seleccionar el tipo de empaquetamiento (ear) y proporcionar un nombre y una descripción.



4. Adicione un nuevo módulo a su Proyecto padre. El nuevo módulo maven será el proyecto EJB o Negocio. Para ello debe dar clic derecho sobre el proyecto padre y en la opción maven seleccionar new Maven Module Project. Para este módulo se creará la configuración por defecto. Así:



En la siguiente ventana deberá seleccionar como tipo de empaquetado EJB.

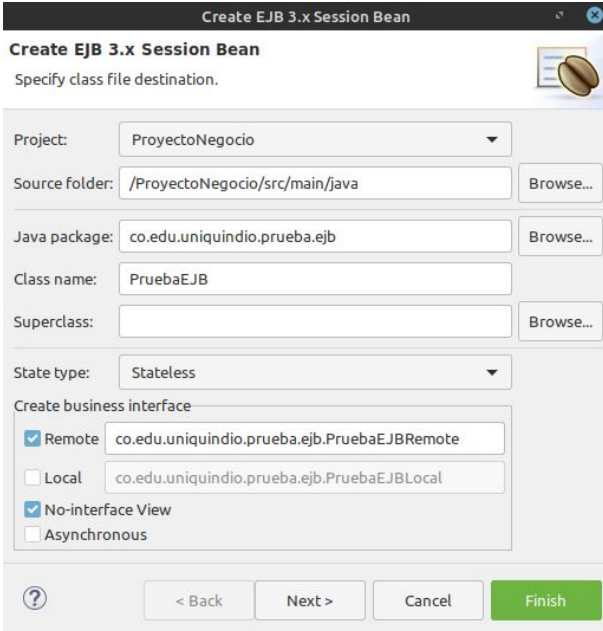


5. Adicione la siguiente dependencia a su archivo pom:

```
<dependencies>
```

```
<dependency>
  <groupId>javax</groupId>
  <artifactId>javaee-api</artifactId>
  <version>8.0</version>
  <scope>provided</scope>
</dependency>
</dependencies>
```

6. Como dependencia del Proyecto de Negocio EJB adicione el Proyecto de Persistencia.
7. En su Proyecto de negocio EJB, cree un EJB para la gestión de su capa de negocio con el nombre de su preferencia, para ello acceda al menu File – New – Other – Session Bean (EJB 3.x). Cuando se le pregunte por el paquete del EJB puede crear un paquete de su preferencia. Seleccione las opciones No-Interface View y Remote.



Como resultado se debió haber creado una clase y una interface.

8. Adicione a su EJB un atributo de tipo EntityManager, con la anotación @PersistenceContext

```
@PersistenceContext
private EntityManager entityManager;
```

9. Adicione como dependencia del Proyecto EAR el módulo del Proyecto de Negocio EJB. Después, debe dar clic derecho sobre el Proyecto padre y en la opción Maven seleccionar Update Project.

```
<dependencies>
  <dependency>
    <groupId>co.edu.uniquindio.grid</groupId>
```

```
<artifactId>ProyectoNegocio</artifactId>
<version>0.0.1-SNAPSHOT</version>
<type>ejb</type>
</dependency>
</dependencies>
```

10. Para probar la correcta configuración de los módulos nuevos, despliegue el proyecto en el servidor de aplicaciones. Para ello de clic derecho sobre el proyecto EAR y acceda a la opción Run As – Run on Server.
11. Si todo ha salido bien, podrá dar clic derecho sobre el servidor de aplicaciones y en el menú Glassfish seleccione la opción view log file, el cual también lo puede encontrar en la ruta glassfish5/glassfish/domains/domain1/logs/server.log. Dentro del log deberá encontrar una línea como la siguiente:

INFO: Portable JNDI names for EJB **NombreDelEJB**

12. En la base de datos verifique que se hayan creado las tablas correspondientes a las entidades de su proyecto.
13. Identifique al menos 5 operaciones de negocio y desarróllelas en métodos individuales. Por ejemplo, Si uno de los requerimientos de negocio es poder registrar nuevos usuarios en la aplicación, debería no solo guardar el usuario en la base de datos, sino también verificar si previamente dicho usuario ya ha sido registrado, o si el login (nombre de usuario, cédula, email) no está siendo usado por otro usuario. **Cree sus propias excepciones** de ser necesario.

```
public void registrarUsuario(Usuario usuario) throws Exception{

    if( entityManager.find(Usuario.class, usuario.getCedula()) != null ) {
        throw new Exception("El usuario ya se encuentra registrado");
    }

    if( buscarEmail(usuario.getEmail()) != null ) {
        throw new Exception("El email ya se encuentra en uso");
    }

    entityManager.persist(usuario);
}
```

14. Después de crear el método, debe registrarlo en la interface del EJB. Recuerde hacer esto por cada método que represente una operación de negocio de la aplicación.

NOTA: Es importante conservar este proyecto, ya que su desarrollo continuará en la siguiente guía.

15. Cree las pruebas unitarias que sean necesarias para verificar el funcionamiento de los EJB. Cree una nueva clase en el proyecto de pruebas que se llame `NegocioTest`, tenga en cuenta que dicha clase deberá ejecutarse con Arquillian, además, tenga en cuenta que en las pruebas unitarias las instancias de los EJB deben declararse así:

```
@EJB
private PruebaEJB pruebaEJB;
```

Y el método para el despliegue sería así:

```
@Deployment
public static Archive<?> createDeploymentPackage() {
    return ShrinkWrap.create(JavaArchive.class).addClass(NOMBRE_EJB.class)
        .addPackage(ENTIDAD.class.getPackage())
        .addAsResource("persistenceForTest.xml", "META-INF/persistence.xml")
        .addAsManifestResource(EmptyAsset.INSTANCE, "beans.xml");
}
```

Reemplace NOMBRE_EJB y ENTIDAD por los valores correctos. Recuerde añadir el Proyecto de negocio EJB como dependencia al proyecto de Pruebas.

Para la próxima clase

Consultar qué tipos de EJB existen y cuál es el objetivo de cada uno.