



*Guía de laboratorio*  
*Área de Programación y Algoritmia*



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

Información general	
ACTUALIZADO POR:	Carlos Andrés Flórez Villarraga
DURACIÓN ESTIMADA EN MINUTOS:	120
DOCENTE:	Christian Andrés Candela y Einer Zapata
GUÍA NO.	03
NOMBRE DE LA GUÍA.	Proyecto Maven

<b>Información de la Guía</b>
-------------------------------

## OBJETIVO

Estudiar el uso de las entidades y su aplicación en el modelamiento de datos.

## CONCEPTOS BÁSICOS

Manejo de Eclipse, Java, Bases de Datos, JDBC, XML, Glassfish.

## CONTEXTUALIZACIÓN TEÓRICA

Maven es un software cuya función principal es la gestión de dependencias de proyectos en Java. Maven es uno de los proyectos de Apache Software Foundation que permite gestionar y crear proyectos Java. Para la configuración y gestión de los proyectos usa archivos de configuración POM (Project Object Model), en los cuales se describe el proyecto y se definen los diferentes elementos a usar, como lo pueden ser plugins, módulos y dependencias externas. Maven se puede usar a través de línea de comandos (cmd o terminal), sin embargo, los principales IDEs de desarrollo ya poseen herramientas que permiten la fácil integración con Maven.

La estructura mínima de un archivo POM es:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>co.edu.uniquindio.grid</groupId>
  <artifactId>proyectomaven</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Proyecto Maven</name>
  <description>Proyecto Maven de Prueba</description>
</project>
```

Donde se tiene como principales elementos el `groupId` y el `artifactId`, el primero corresponde al identificador del grupo de artefactos (proyectos) al cual pertenece nuestro proyecto, el segundo por su parte es el identificador único de nuestro proyecto dentro del grupo de artefactos. Ambos, `groupId` y `artifactId`, conforman un identificador único para nuestro proyecto.

En caso de que el proyecto así lo requiera se pueden adicionar dependencias hacia otros proyectos o librerías.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>co.edu.uniquindio.grid</groupId>
  <artifactId>proyectomaven</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Proyecto Maven</name>
  <description>Proyecto Maven de Prueba</description>
  <dependencies>
    <dependency>
      <groupId>javax</groupId>
      <artifactId>javaee-api</artifactId>
      <version>8.0</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>
</project>
```

Como puede verse en este caso se adiciono una dependencia hacia el proyecto `javaee-api` en su versión 8. El atributo `scope` indica en que momento de la vida de nuestro proyecto se necesita la dependencia sus valores pueden ser: `compile`, `provided`, `runtime`, `test` y `system`.

- **Compile:** es la que tenemos por defecto sino especificamos `scope`. Indica que la dependencia es necesaria para compilar. La dependencia además se propaga en los proyectos dependientes.
- **Provided:** Similar a `Compile`, pero indica que se espera que el JDK (o servidor de aplicaciones) donde sea ejecutado el proyecto sea quien provea la dependencia. Significa que necesita el JAR para la compilación, pero en el tiempo de ejecución ya hay un JAR proporcionado por el entorno, por lo que no lo necesita empaquetar con la aplicación.
- **Runtime:** Indica que la dependencia no es requerida para compilar el proyecto, pero si para su ejecución
- **Test:** Indica que la dependencia no es requerida en la ejecución normal de la aplicación, pero si lo es para la ejecución de sus pruebas.
- **System:** Similar a `provided`, pero indica que el jar no se encuentra en ningún repositorio sino local y el mismo deberá ser indicado de forma específica.

Para indicar o realizar tareas específicas en un proyecto se puede hacer uso de plugins, el más usado es el que permite indicar la versión de java con que se debe compilar el proyecto.

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>co.edu.uniquindio.grid</groupId>
  <artifactId>proyectomaven</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Proyecto Maven</name>
  <description>Proyecto Maven de Prueba</description>
  <dependencies>
    <dependency>
```

```
<groupId>javax</groupId>
<artifactId>javaee-api</artifactId>
<version>8.0</version>
<scope>provided</scope>
</dependency>
</dependencies>
<build>
  <plugins>
    <!-- Establecer la version de java a ser usada por los proyectos -->
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version>
      <configuration>
        <source>1.8</source>
        <target>1.8</target>
      </configuration>
    </plugin>
  </plugins>
</build>
</project>
```

Para más información consulte: <https://maven.apache.org/guides/>

## 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 haciendo 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 el 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.

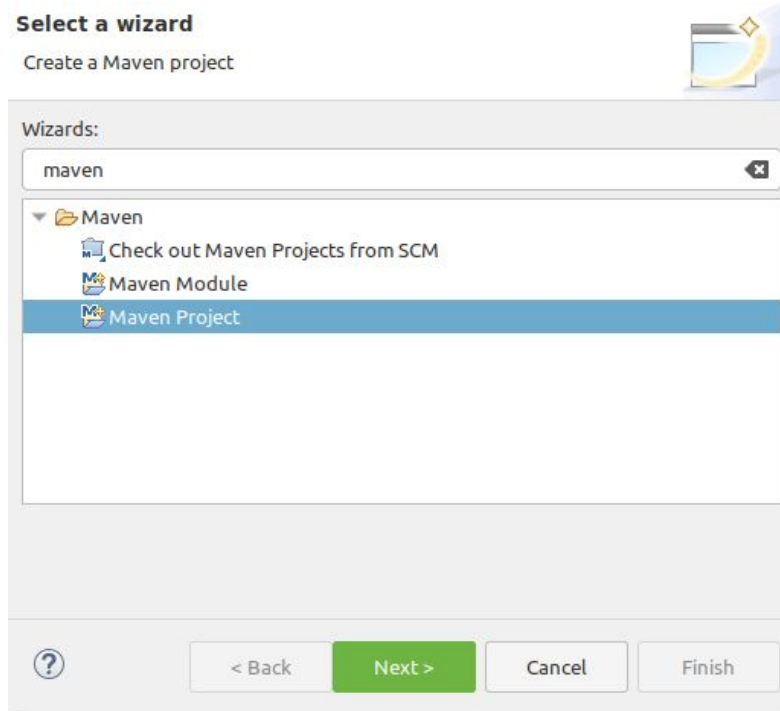
## EVALUACIÓN O RESULTADO

Se espera que el alumno logre crear y realizar configuraciones básicas de proyectos usando maven.

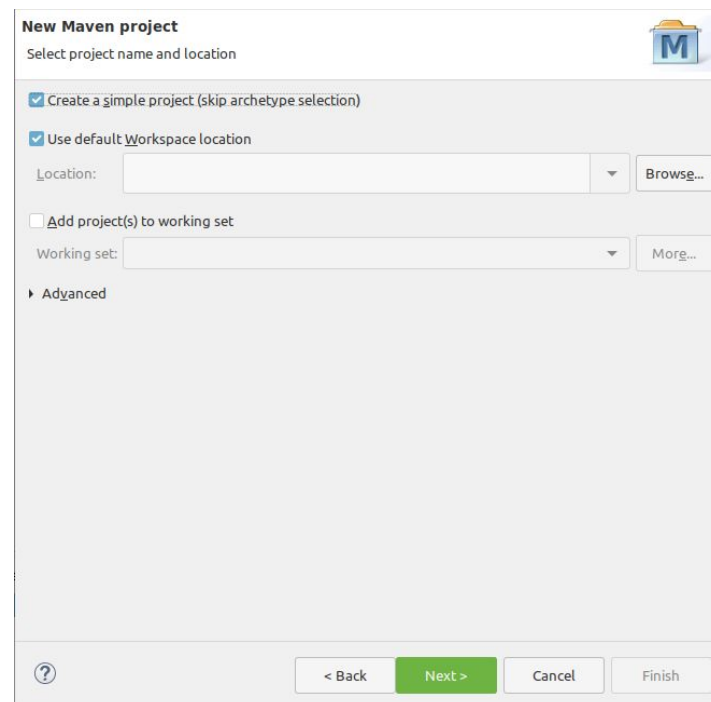
### Procedimiento

1. Se iniciará creando un proyecto maven que permita contener otros proyectos, este tipo de proyecto es llamado proyecto padre. Para ello deberá:

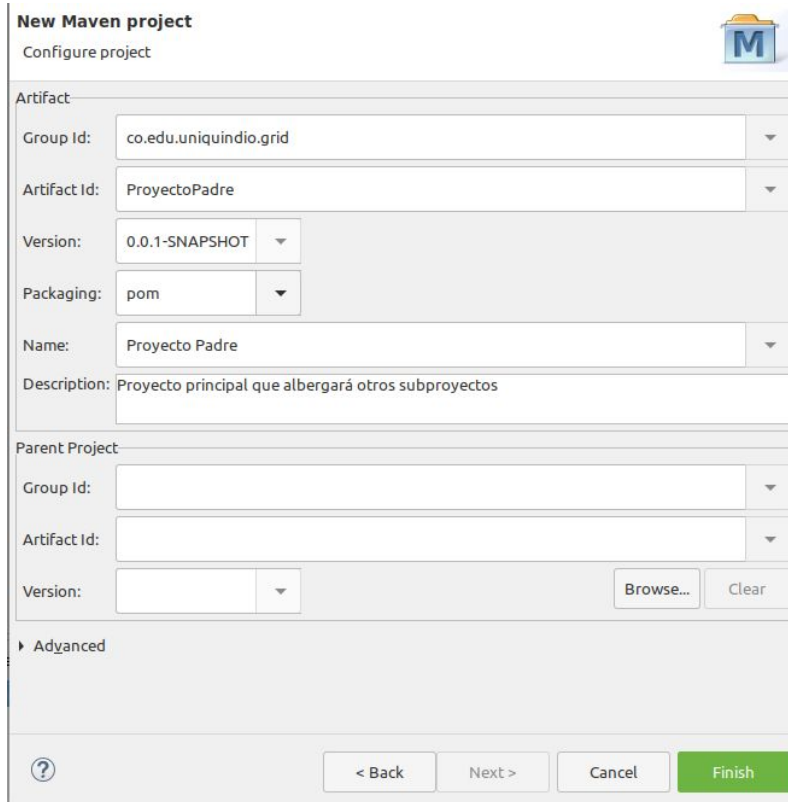
En el menú file – new – Maven Project, Cree un Maven Project con configuración por defecto



Next,



Next,



En este punto debe indicar al menos el groupid, el artefactid la versión y el tipo de empaquetado de su proyecto, en este caso el empaquetado es **POM**.

Finish.

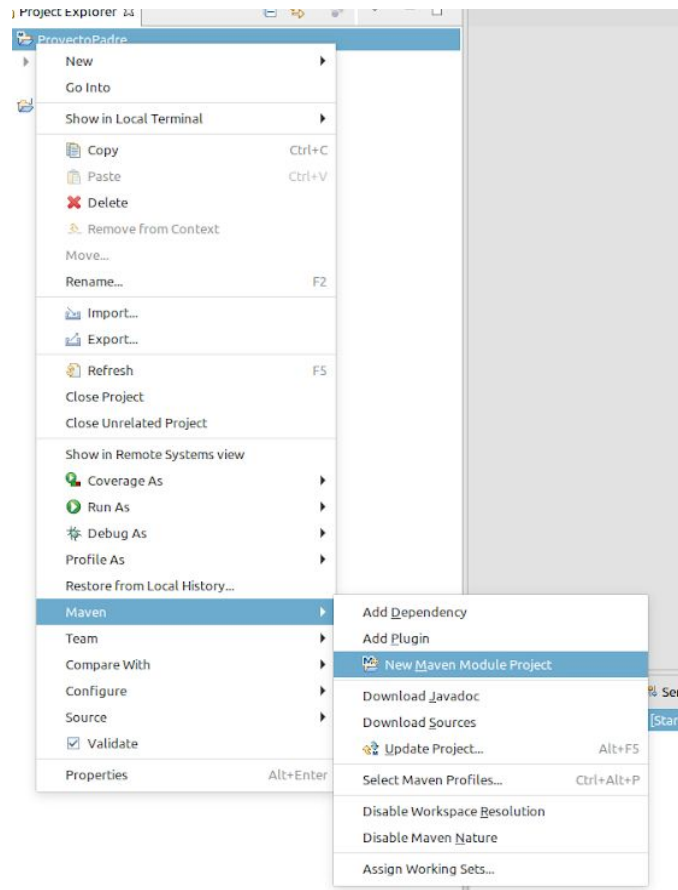
2. Modifique el archivo pom para establecer las configuraciones base de los plugins a ser usados en los proyectos hijos (agregue lo siguiente al final del pom).

```
<build>
  <pluginManagement>
    <plugins>
      <!-- Establecer la version de java a ser usada por los proyectos -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
      <!-- Establecer la version de EAR a ser usada -->
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-ear-plugin</artifactId>
        <version>2.10.1</version>
```

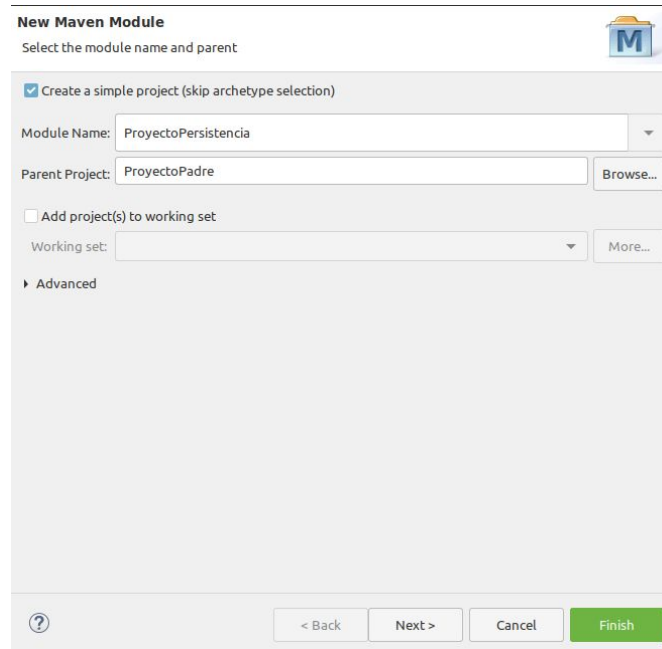
```
<configuration>
<version>7</version>
<defaultLibBundleDir>lib</defaultLibBundleDir>
<defaultJavaBundleDir>lib</defaultJavaBundleDir>
<skinnyWars>true</skinnyWars>
<generateModuleId>true</generateModuleId>
<fileNameMapping>no-version</fileNameMapping>
<archive>
  <manifest>
    <addClasspath>true</addClasspath>
  </manifest>
</archive>
</configuration>
</plugin>
<!-- Establecer la version de EJB -->
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-ejb-plugin</artifactId>
  <version>3.0.1</version>
  <configuration>
    <ejbVersion>3.2</ejbVersion>
  </configuration>
</plugin>
<!-- Establecer la version de WAR -->
<plugin>
  <groupId>org.apache.maven.plugins</groupId>
  <artifactId>maven-war-plugin</artifactId>
  <version>3.2.3</version>
  <configuration>
    <failOnMissingWebXml>false</failOnMissingWebXml>
  </configuration>
</plugin>
</plugins>
</pluginManagement>
</build>
```

### 3. Crear proyecto para persistencia como hijo del proyecto padre.

Debe dar clic derecho sobre el proyecto padre y en la opción maven seleccionar new maven module Project, al igual que en el módulo ear use la configuración por defecto



De un nombre a su módulo de persistencia, haga uso de la configuración por defecto.



**New Maven Module**  
Select the module name and parent

☒ Create a simple project (skip archetype selection)

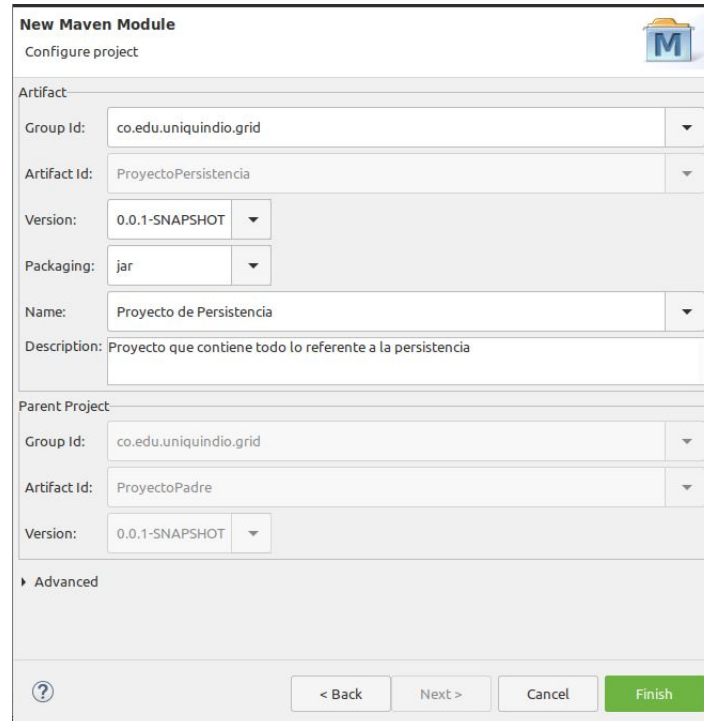
Module Name:

Parent Project:

☐ Add project(s) to working set  
Working set:

► Advanced

Next, en la siguiente ventana como tipo de empaquetador use el jar



**New Maven Module**  
Configure project

Artifact:

Group Id:

Artifact Id:

Version:

Packaging:

Name:

Description:

Parent Project:

Group Id:

Artifact Id:

Version:

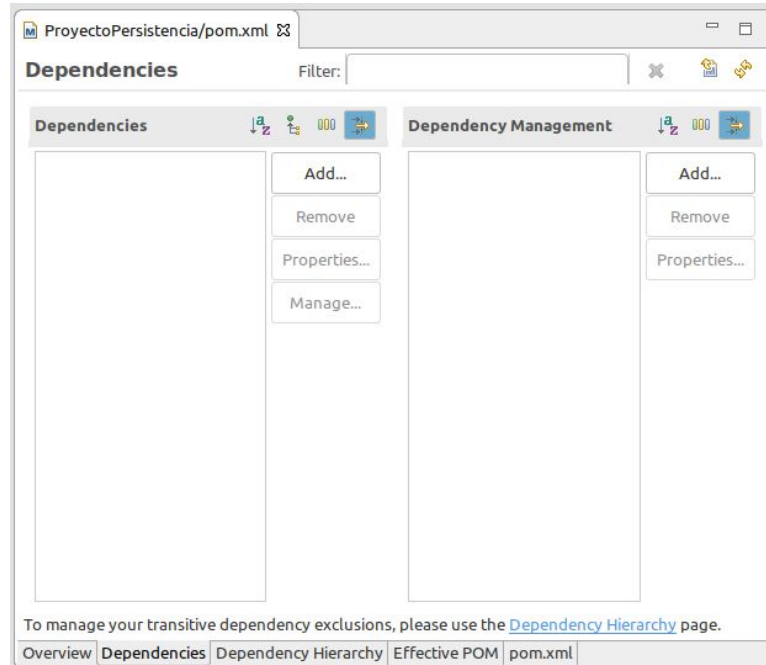
► Advanced

Finish.

4. Adicione a su proyecto de persistencia la dependencia `hibernate-jpa-2.1-api`:



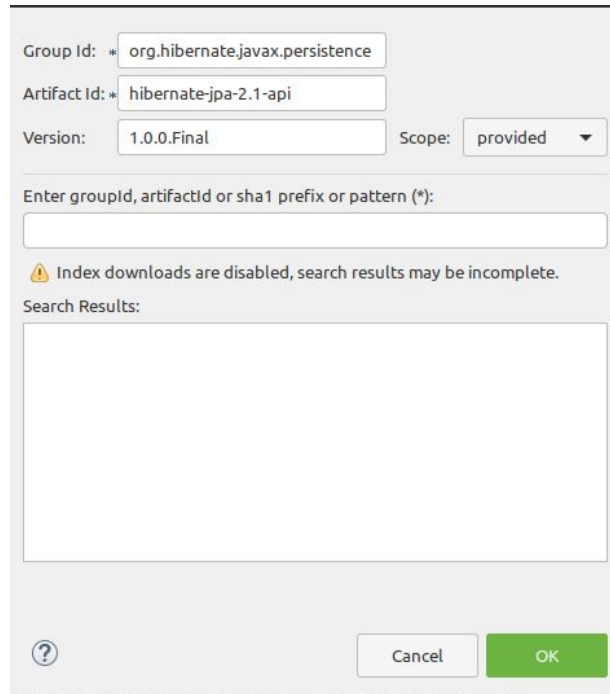
Para adicionar una nueva dependencia abra el archivo pom, y de clic en la pestaña de dependencias.



De clic en el botón Add... En la ventana emergente proporcione los siguientes datos

```
groupId: org.hibernate.javax.persistence  
artifactId: hibernate-jpa-2.1-api  
versión: 1.0.2.Final  
scope: provided
```

Esta dependencia permite adicionar el soporte de persistencia al proyecto.



De clic en Ok. Este proceso adiciona a su archivo pom las siguientes líneas.

```
<dependency>
  <groupId>org.hibernate.javax.persistence</groupId>
  <artifactId>hibernate-jpa-2.1-api</artifactId>
  <version>1.0.2.Final</version>
  <scope>provided</scope>
</dependency>
```

O esta otra dependiendo de si el proveedor de persistencia a usar es hibertante o eclipse

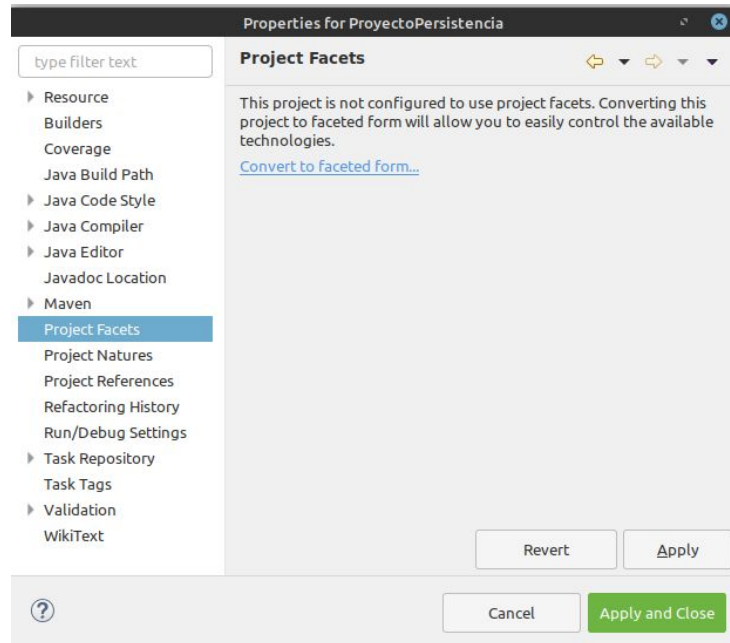
```
<dependency>
  <groupId>org.eclipse.persistence</groupId>
  <artifactId>javax.persistence</artifactId>
  <version>2.2.1</version>
  <scope>provided</scope>
</dependency>
```

5. Adicione a su proyecto de persistencia la siguiente dependencia.

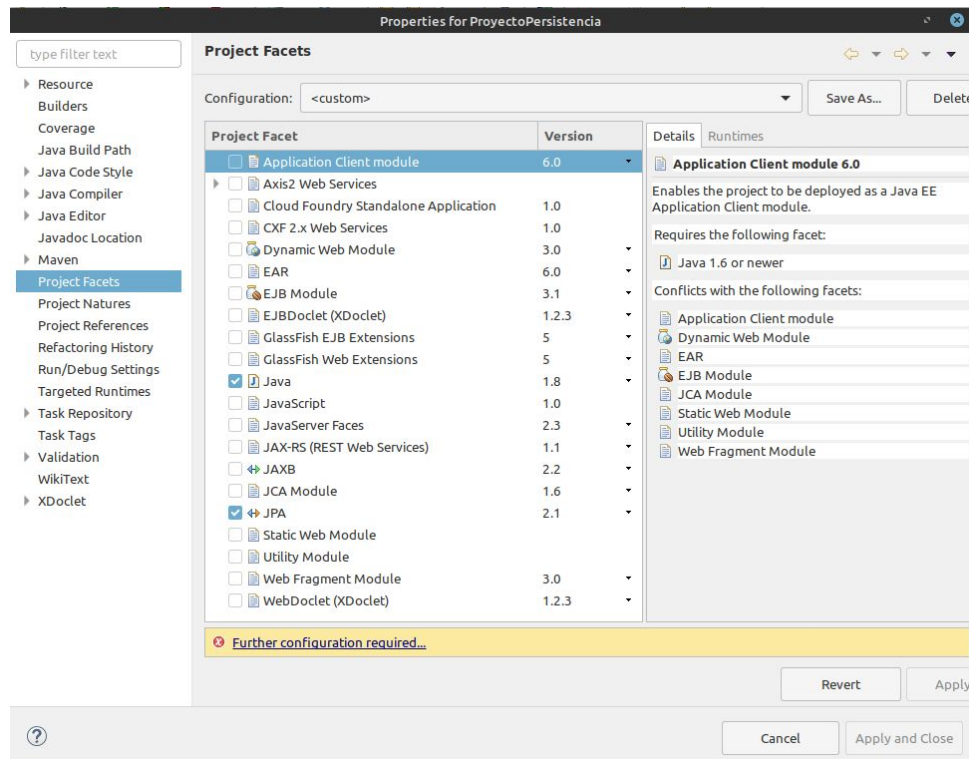
```
<dependency>
  <groupId>javax</groupId>
  <artifactId>javaee-api</artifactId>
  <version>8.0</version>
  <scope>provided</scope>
</dependency>
```

6. Ahora debe indicar a eclipse que su proyecto de persistencia hace uso de JPA, para ello, de clic derecho

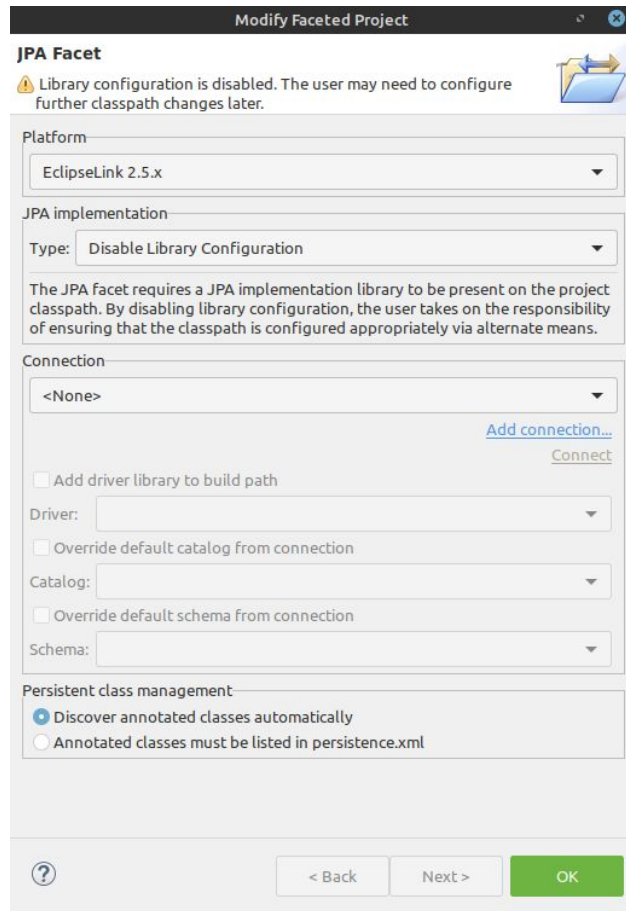
sobre el proyecto de persistencia y en el menú maven de clic en actualizar el proyecto. Nuevamente clic sobre el proyecto, acceda a las propiedades y a la opción Project Facets, de clic sobre convertir el proyecto para poder usar facet.



Allí seleccione la opción JPA en su versión 2.1. En esta venta verifique que la versión de Java es la 1.8.



De clic en el enlace de la parte de abajo que sale en rojo donde le solicitan establecer la configuración (Further configuration required...)



Seleccione la librería EclipseLink en su mayor versión, deshabilite la configuración y seleccione la opción de descubrir automáticamente las clases por medio de su anotación.

Dar click en OK y luego en Aplicar los cambios y cerrar.

7. Es necesario disponer de un Datasource en su servidor glassfish con acceso a una base de datos en mysql, si así lo desea puede hacer uso del creado en la Guia 02 o crear uno nuevo. Otra opción para aquellos que trabajan en los equipos de la Universidad es hacer uso del datasource `jdbc/test`.
8. Configure el archivo `persistence.xml` para que haga uso de su datasource y descubra las entidades de forma automática. Para abrir el archivo, acceda a la sección Source Code de su proyecto, en ella al `src/main/java/` y al `META-INF`.

De clic derecho sobre el archivo y seleccione abrir con Persistence Editor. En la pestaña General deselectione la opción *Exclude unlisted classes*. En la pestaña de Connection en la opción JTA Data Source ingrese el nombre de su data source. Luego ingrese a la pestaña *Schema Generation* y en la opción *Database* seleccione el valor *Create* y de igual forma en la opción *DDL Generation Type*.

\*persistence.xml

### General

**General**

Name:

Persistence provider:

Description:

**XML Mapping Files**

Specify the XML mapping files for this persistence unit.

☒ Exclude default EclipseLink XML mapping file (False)

**JAR Files**

**Managed Classes**

Specify the list of classes to be managed in this persistence unit.

☐ Exclude unlisted classes

General | Connection | Customization | Caching | Options | Schema Generation | Properties | Source

\*persistence.xml

### Connection

**Persistence Unit Connection**

Configure the data source or JDBC connection properties.

Transaction type:

Batch writing:

☒ Statement caching:

☒ Native SQL (False)

Database

JTA data source:

Non-JTA data source:

EclipseLink connection pool

[Populate from connection...](#)

Driver:

URL:

User:

Password:

☒ Bind parameters (True)

**Read Connection**

☒ Shared (False)

Minimum:

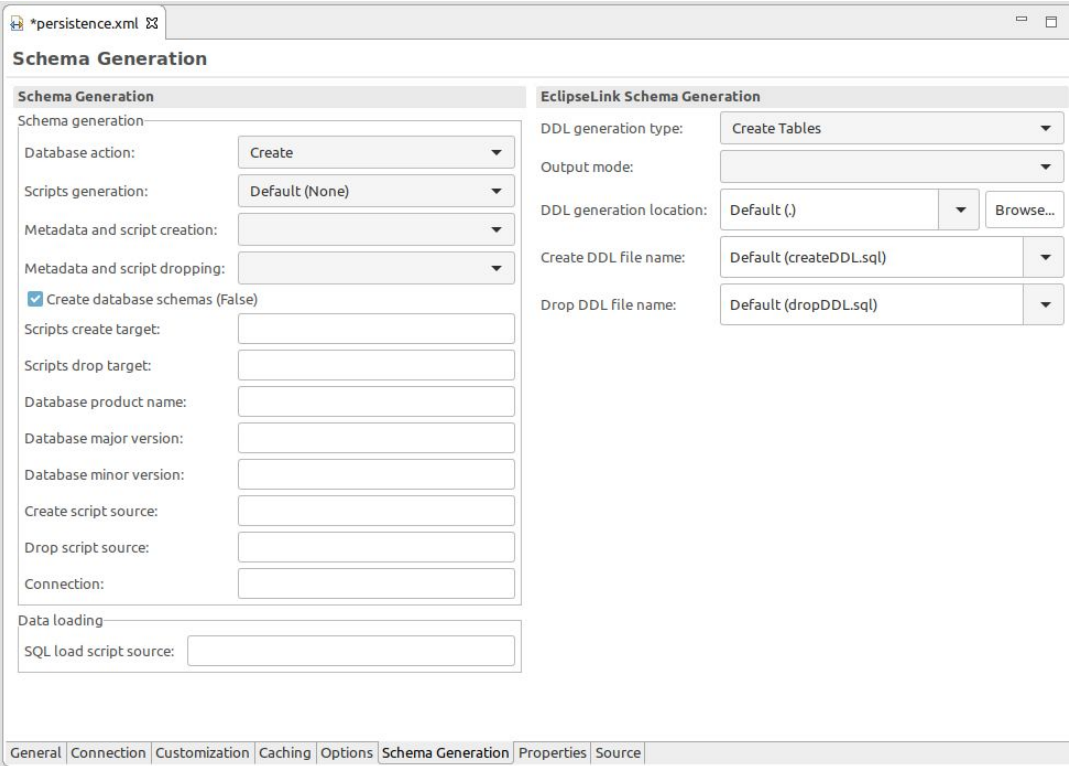
Maximum:

**Write Connection**

Minimum:

Maximum:

General | Connection | Customization | Caching | Options | Schema Generation | Properties | Source



El archivo de persistencia (persistence.xml) debe quedar así:

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.1" xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">
  <persistence-unit name="ProyectoPersistencia">
    <jta-data-source>jdbc/prueba</jta-data-source>
    <exclude-unlisted-classes>>false</exclude-unlisted-classes>
    <properties>
      <property name="javax.persistence.schema-generation.database.action"
value="create"/>
      <property name="eclipselink.ddl-generation" value="create-tables"/>
    </properties>
  </persistence-unit>
</persistence>
```

**IMPORTANTE:** Por último, debe mover la carpeta META-INF de su ubicación actual a la carpeta de código src/main/resources



9. De clic derecho en el proyecto padre y acceda al menú **Maven – Update project**
10. Adicione a su proyecto padre un nuevo **Maven Module** con configuración por defecto de tipo jar, se llamará ProyectoPruebas. Dicho modulo será usado para el desarrollo de las pruebas de su proyecto.
11. En el archivo pom del nuevo proyecto adicione los siguientes bloques:

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>org.jboss.arquillian</groupId>
      <artifactId>arquillian-bom</artifactId>
      <version>1.4.1.Final</version>
      <scope>import</scope>
      <type>pom</type>
    </dependency>
  </dependencies>
</dependencyManagement>

<!-- Para el uso de objectdb, en caso de no usar objectdb, el repositorio puede ser borrado -->
<repositories>
  <repository>
    <id>objectdb</id>
    <name>ObjectDB Repository</name>
    <url>http://m2.objectdb.com</url>
  </repository>
</repositories>
```

12. Como dependencias de su proyecto de pruebas adicione su Proyecto de persistencia y las siguientes dependencias.

```
<dependencies>
  <dependency>
    <groupId>javax.enterprise</groupId>
    <artifactId>cdi-api</artifactId>
    <version>2.0</version>
  </dependency>
  <dependency>
    <groupId>org.hibernate.javax.persistence</groupId>
    <artifactId>hibernate-jpa-2.1-api</artifactId>
    <version>1.0.2.Final</version>
  </dependency>
  <dependency>
    <groupId>javax</groupId>
    <artifactId>javaee-api</artifactId>
```

```
<version>8.0</version>
<scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.glassfish.main.extras</groupId>
  <artifactId>glassfish-embedded-all</artifactId>
  <version>5.1.0</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.primefaces</groupId>
  <artifactId>primefaces</artifactId>
  <version>6.2</version>
</dependency>

<!-- Dependencias para las pruebas -->
<dependency>
  <groupId>junit</groupId>
  <artifactId>junit</artifactId>
  <version>4.12</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.jboss.arquillian.junit</groupId>
  <artifactId>arquillian-junit-container</artifactId>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.jboss.arquillian.container</groupId>
  <artifactId>arquillian-glassfish-embedded-3.1</artifactId>
  <version>1.0.2</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>mysql</groupId>
  <artifactId>mysql-connector-java</artifactId>
  <version>5.1.45</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.jboss.arquillian.protocol</groupId>
  <artifactId>arquillian-protocol-servlet</artifactId>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.jboss.arquillian.extension</groupId>
  <artifactId>arquillian-persistence-dbunit</artifactId>
  <version>1.0.0.Alpha7</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>org.jboss.arquillian.extension</groupId>
  <artifactId>arquillian-transaction-jta</artifactId>
  <version>1.0.5</version>
  <scope>provided</scope>
</dependency>
<dependency>
  <groupId>co.edu.uniquindio.grid</groupId>
```



```
<artifactId>ProyectoPersistencia</artifactId>
<version>0.0.1-SNAPSHOT</version>
</dependency>
</dependencies>
```

13. Cree en su proyecto de prueba el archivo **sun-resources.xml**, el cual deberá estar ubicado en **src/test/resources**. Este archivo será el encargado de crear de forma automática el pool de conexiones y el data source que se usaran en las pruebas.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE resources PUBLIC "-//Sun Microsystems, Inc.//DTD Application Server 9.0 Resource
Definitions //EN" "http://www.sun.com/software/appserver/dtds/sun-resources_1_3.dtd">
<resources>
  <jdbc-connection-pool name="mysql_prueba_pool"
    datasource-classname="com.mysql.jdbc.jdbc2.optional.MysqlDataSource"
    res-type="javax.sql.DataSource">
    <property name="user" value="root" />
    <property name="password" value="12345" />
    <property name="url"
      value="jdbc:mysql://localhost:3306/prueba" />
  </jdbc-connection-pool>
  <jdbc-resource enabled="true" jndi-name="jdbc/prueba"
    object-type="user" pool-name="mysql_prueba_pool" />
</resources>
```

Note que el atributo *user* debe corresponder el usuario empleado para conexión a su base de datos, así mismo el atributo *password* deberá contener la clave de acceso de dicho usuario.

Por otra parte, el atributo *url* tiene la siguiente estructura:

***jdbc:mysql://SERVIDOR\_DONDE\_ESTA\_LA\_BD:PUERTO\_DEL\_MOTOR\_BD/NOMBRE\_BD***

en el caso del ejemplo se ha establecido **localhost** como la dirección del equipo donde está la base de datos. Para el puerto se ha dejado el **puerto** por defecto 3306, finalmente se está haciendo uso de la base de datos *prueba* de *mysql*.

14. En la misma carpeta donde creo el archivo **sun-resources** cree un archivo con nombre **arquillian.xml** este archivo contiene la configuración básica de arquillian.

```
<?xml version="1.0" encoding="UTF-8"?>
<arquillian xmlns="http://jboss.org/schema/arquillian"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://jboss.org/schema/arquillian
    http://jboss.org/schema/arquillian/arquillian_1_0.xsd">
  <container qualifier="glassfish" default="true">
    <configuration>
      <property name="resourcesXml">src/test/resources/sun-resources.xml</property>
    </configuration>
  </container>
  <extension qualifier="persistence-script">
    <!-- Estos son para desactivar la verificacion de llaves foraneas en mysql -->
```

```
<property name="scriptsToExecuteBeforeTest">SET foreign_key_checks =  
0;</property>  
<property name="scriptsToExecuteAfterTest">SET foreign_key_checks =  
1;</property>  
</extension>  
</arquillian>
```

15. En la misma carpeta donde creo el archivo sun-resources y arquillian.xml cree un archivo con nombre **persistenceForTest.xml** este archivo contiene la configuración básica de la persistencia usada para las pruebas.

```
<?xml version="1.0" encoding="UTF-8"?>  
<persistence version="2.1"  
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence  
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">  
  <persistence-unit name="ProyectoPersistencia">  
    <jta-data-source>jdbc/prueba</jta-data-source>  
    <exclude-unlisted-classes>>false</exclude-unlisted-classes>  
    <properties>  
      <property  
        name="javax.persistence.schema-generation.database.action"  
        value="create" />  
      <property name="eclipselink.ddl-generation"  
        value="create-tables" />  
    </properties>  
  </persistence-unit>  
</persistence>
```

16. En la misma carpeta donde creo el archivo sun-resources.xml y arquillian.xml cree un archivo con nombre **persistenceForObjectDB.xml** este archivo contiene la configuración básica de la persistencia usada para las pruebas (**Este archivo solo es necesario si se va a usar ObjectDB**).

```
<?xml version="1.0" encoding="UTF-8"?>  
<persistence version="2.1"  
  xmlns="http://xmlns.jcp.org/xml/ns/persistence"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence  
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_1.xsd">  
  <persistence-unit name="jdbc/prueba"  
    transaction-type="JTA">  
    <provider>com.objectdb.jpa.Provider</provider>  
    <exclude-unlisted-classes>>false</exclude-unlisted-classes>  
    <properties>  
      <property name="javax.persistence.jdbc.url"  
        value="$objectdb/db/objectdbTest.odb" />  
      <property name="javax.persistence.jdbc.user" value="admin" />  
      <property name="javax.persistence.jdbc.password"  
        value="admin" />  
      <property  
        name="javax.persistence.schema-generation.database.action"  
        value="create" />  
      <property name="activation" value="A52M-8Y5T-4R1Y-RQWQ-6YSY" />  
    </properties>  
  </persistence-unit>  
</persistence>
```

```
</properties>  
  </persistence-unit>  
</persistence>
```

### Para la próxima clase

Leer y entender el concepto de entidad en Java  
Leer y entender el concepto de base de datos relacional.  
Leer y entender el concepto de JavaBeans.  
Leer y entender el concepto de Enterprise JavaBeans.  
Leer y entender el concepto de Java Persistence Api.