Java Persistence Api – Mapeo Directo

Objetivo

- Presentar la estrategia de mapeo directo (Clases a tablas) con JPA.
- Ejecutar el mapeo directo con una aplicación Maven, usando EclipseLink y PostgreSQL

Aprendizaje

Luego de la lectura de esta guía el alumno:

- Conocerá las anotaciones JPA para:
  - Identificar clases persistentes
  - Establecer clave primaria y estrategia de generación de valores
  - Establecer asociaciones (one-to-one, many-to-many, many-to-one, one-to-many) entre clases del modelo
- Aprenderá las tareas de configuración del IDE Eclipse para la gestión de la persistencia de datos (persistence.xml, orm.xml)
- Podrá codificar un programa básico que realice el mapeo de clases hacia una base de datos PostgreSQL vacía.

Contenido

Mapeo Directo

El mapeo directo o forward mapping consiste en generar un esquema sobre una base de datos relacional, a partir de Clases de un modelo orientado a objetos. Esto permite mantener el pensamiento orientado a objetos dentro de la etapa de implementación de un proyecto software. En el caso de mapeo inverso, es necesario que el modelo de datos relacional exista previamente.

Creación de proyecto Maven + JPA

Para realizar el mapeo directo, esta guía usará un proyecto Maven con el agregado de JPA y Eclipselink como su implementación. Los pasos para crear y configurar este proyecto ha sido descripto en una guía anterior. Se seguirán los mismos pasos indicados en ella, con las siguientes salvedades:

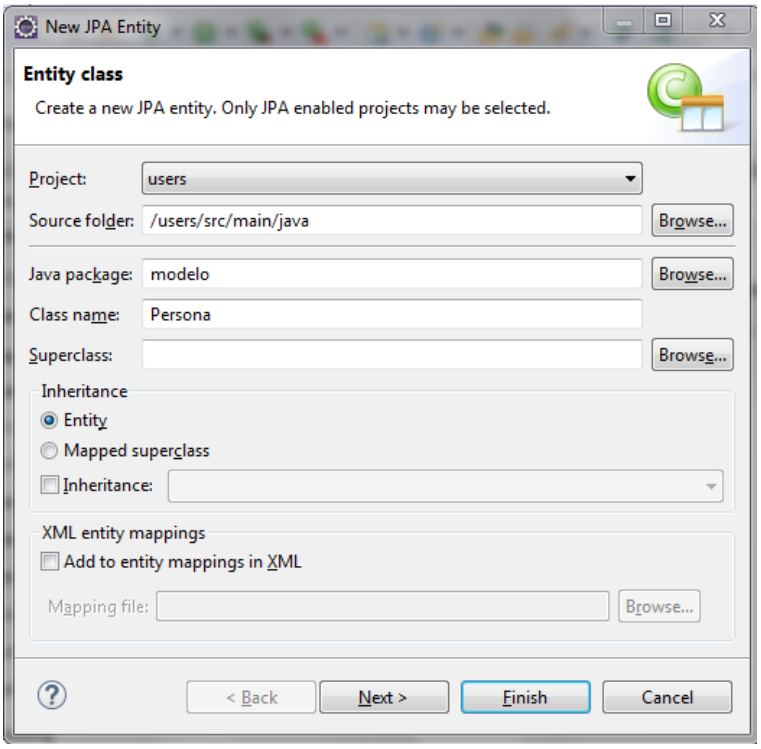
- Crear una base de datos vacía (sin tablas, vistas, secuencias, etc) y establecer la conexión con esa base de datos en Eclipse.
- En el archivo persistence.xml configurar, en la pestaña Schema Generation, la opción Database action en modo “Create”. Esto permitirá que se puedan generar las tablas desde las clases de Entidad

Creación de clases de Entidad (modelo de datos)

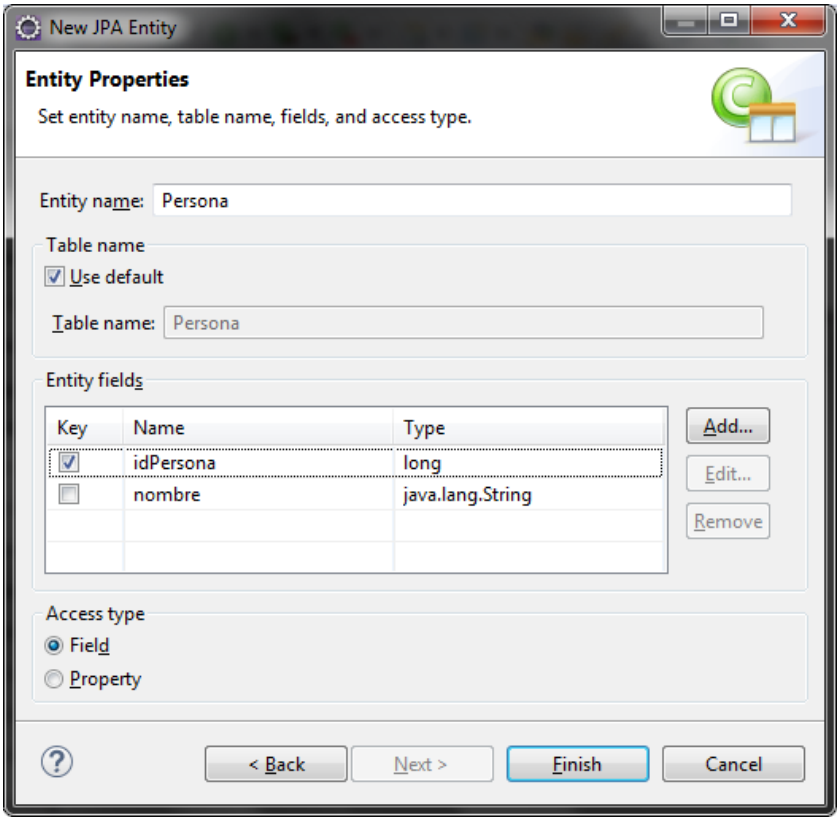
Para crear las clases de Entidad del proyecto, Eclipse cuenta con un asistente. Esto permitirá visualmente crear las clases, y agregar las anotaciones correspondientes. Es posible crear una clase básica Java y agregarle todas las anotaciones e importaciones, pero para esta guía se usará el asistente.

Los pasos son los siguientes:

1- Clic derecho en el proyecto -> New -> JPA Entity
   Se debe proporcionar los datos de:
   - Java Package, para organizar las clases de entidad en un package específico. Estas clases representan el modelo de datos del proyecto.
   - Class name, representa el nombre de la clase. Recordar que según convención, los nombres de clase van con mayúscula inicial, y en singular.
   - En modelos de datos con Herencia, se debe indicar la clase superior, y la estrategia de herencia utilizada para el mapeo (SINGLE_TABLE, TABLE_PER_CLASS, JOINED)

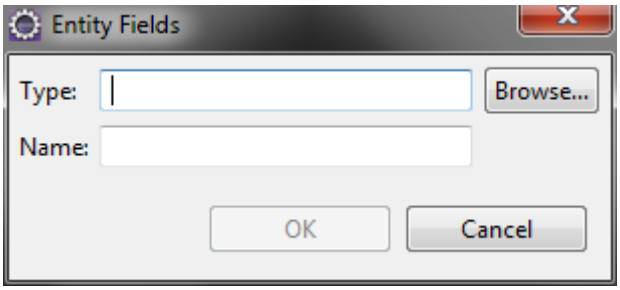


2- Establecer los atributos de la clase, indicando su tipo, alias y opcionalmente indicar cuál de ellos representa el atributo identificador.



Al agregar cada atributo, se muestra una ventana donde se especifica el tipo, y el alias correspondiente. Al escribir parcialmente el nombre del tipo, Eclipse sugiere en una lista desplegable el tipo a elegir.

Nota: Al momento de crear atributos de clases definidas por el usuario, se debe indicar el paquete donde se encuentra. Por ejemplo, para agregar un atributo de tipo Domicilio en la clase Persona (ambas clases en el paquete "modelo") se debe indicar en el campo Tipo el valor "modelo.Domicilio".



Clave Primaria

Cada clase de entidad en JPA requiere al menos tres características:

- Anotación @Entity
- Constructor sin argumentos
- Al menos un atributo con anotación @Id

Los motores de base de datos usan distintas estrategias de acuerdo a sus características. Podemos mencionar:

- Estrategia AUTO: es la estrategia por defecto, y deja a la implementación de JPA la elección del mecanismo de generación de clave primaria.
- Estrategia IDENTITY: utiliza una columna autoincremental para generar el valor de la clave primaria. Las bases de datos MySQL y SQL Server usan esta estrategia
- Estrategia SEQUENCE: obtiene el valor de la clave primaria a partir de una secuencia. Esto es común en bases de datos PostgreSQL y Oracle.
- Estrategia TABLE: usa una tabla específica para la obtención de clave primaria, similar a una secuencia. No es la mejor estrategia, y es preferible en este caso usar secuencias, si la base de datos lo permite.

Para esta guía se usará la estrategia SEQUENCE debido a que se conectará a una base de datos PostgreSQL.

La configuración de cada clase de entidad para generar clave primaria desde una secuencia se compone de dos anotaciones:

- @GeneratedValue: inmediatamente después de la anotación @Id se agrega esta anotación, la cual tiene como argumentos:

- strategy=GenerationType.SEQUENCE
- generator="*nombre-del-sequence-generator*"

```
@GeneratedValue(strategy=GenerationType.SEQUENCE, generator="class_seq")
```

- @SequenceGenerator: especifica los datos de la secuencia en la base de datos. Se coloca luego de la anotación @Entity. Tiene como argumentos:
  - name: nombre del SequenceGenerator (debe coincidir con el atributo *generator* establecido en *GeneratedValue*).
  - sequenceName: nombre del elemento secuencia en la base de datos. Puede ser el mismo valor de *name*.
  - initialValue: valor inicial desde el cual comenzará la secuencia.
  - allocationSize: cantidad de valores almacenados en memoria para la generación de registros. Esto permite agilizar la generación de datos, pero en sistemas concurrentes origina saltos en los valores que provee la secuencia. Con valor 1 originará que en cada inserción se deba recurrir a la base de datos para obtener el siguiente valor, pero mantiene una distribución uniforme al generar claves primaria.

```
@SequenceGenerator(name="class_seq", sequenceName="class_seq",initialValue=0, allocationSize=1)
```

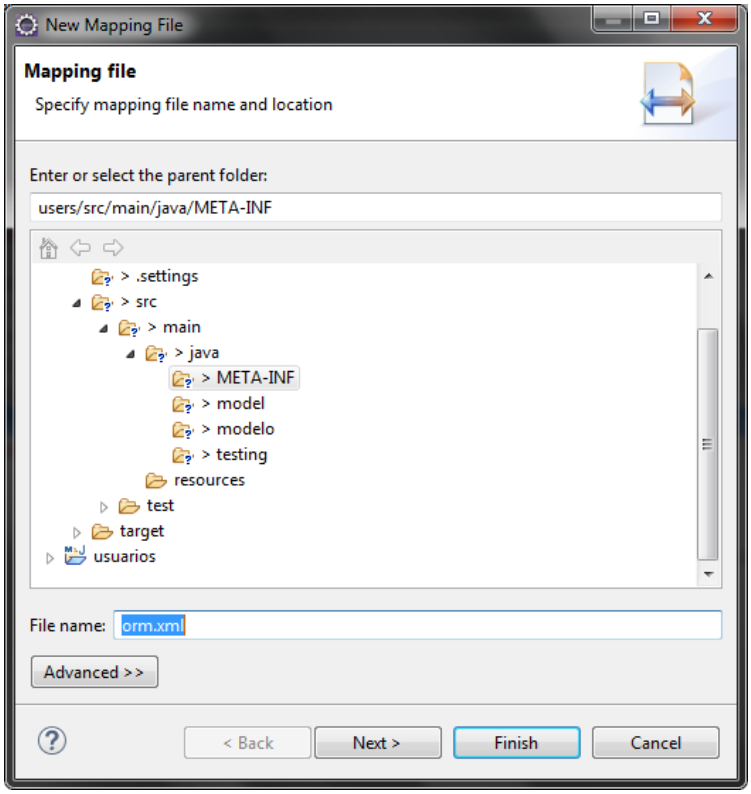
Establecer asociaciones entre clases

Se debe reflejar las asociaciones del modelo de datos usando las anotaciones correspondientes, presentadas en el documento JPA-Minibook.

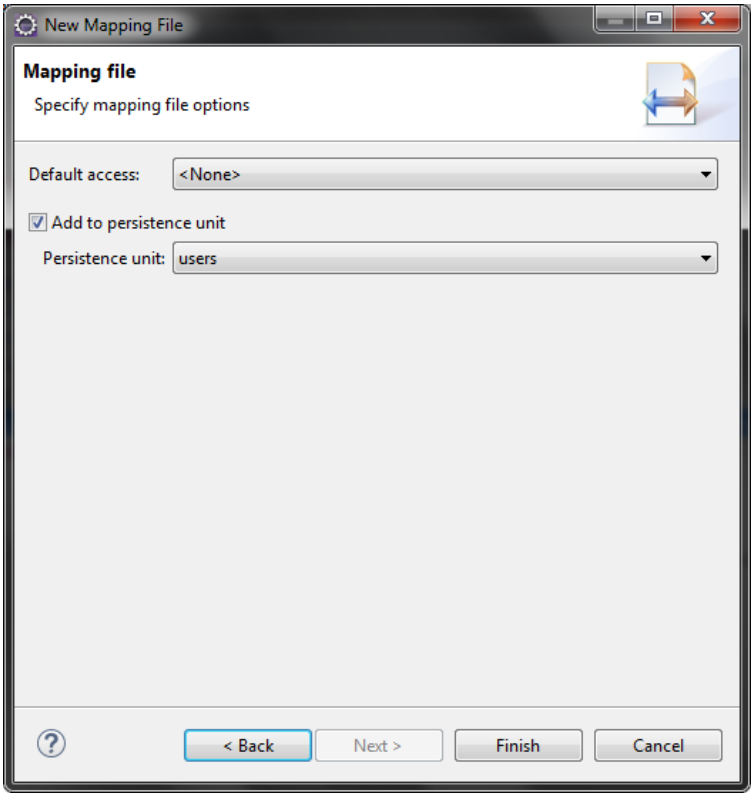
Archivo de configuración de mapeo orm.xml

Al crear un proyecto JPA es posible crear al mismo tiempo el archivo que contiene metadatos relacionados al mapeo. Este archivo se denomina orm.xml (**O**bject **R**elational **M**apping). Si no se creó al comienzo, es posible agregarlo al proyecto de manera simple, a través de un asistente de Eclipse.

- Clic derecho en el proyecto -> New -> JPA ORM Mapping File.
  Por defecto, el archivo será creado en la carpeta META-INF del proyecto.



Una de las opciones del archivo solicita la unidad de persistencia a la cual asociarse. Se debe seleccionar la correspondiente al proyecto



Este archivo orm.xml permitirá configurar de manera global las opciones de CASCADE para inserciones, actualizaciones o borrado de datos. Por ejemplo, para el caso de configuración global de inserciones, se debe agregar lo siguiente

```
<persistence-unit-metadata>
	<persistence-unit-defaults>
		<cascade-persist/>
	</persistence-unit-defaults>
</persistence-unit-metadata>
```

Actualización del archivo persistence.xml

Uno de los errores que puede acusar Eclipse se relaciona a que las clases de entidad creadas aún no están contempladas en el archivo de configuración de JPA (persistence.xml). Para agregarlas, simplemente se debe hacer clic derecho en el archivo persistence.xml y seleccionando la opción Synchronize Class List (puede variar dependiendo las versiones de Eclipse).

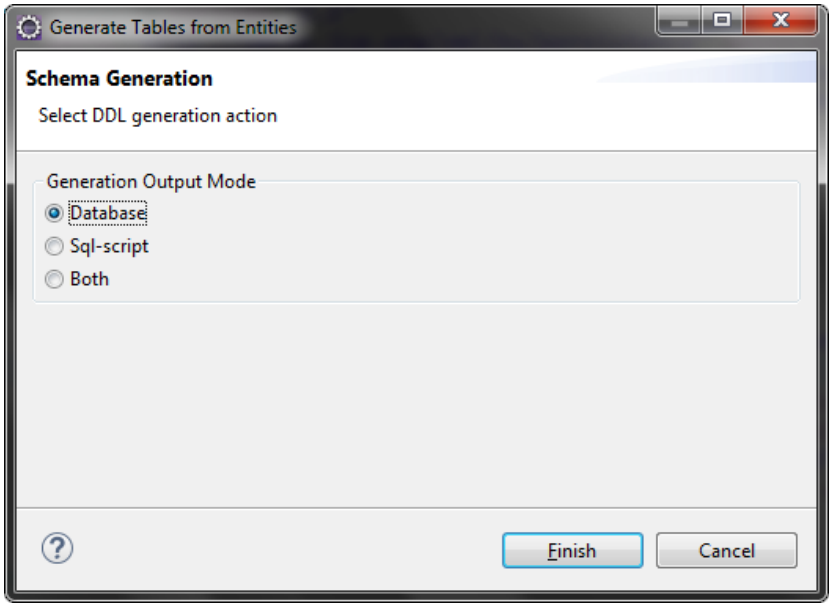
Mapeo directo de las clases

Finalmente, una vez creadas las clases, agregadas las anotaciones de identidad y asociaciones, y actualizados los archivos de configuración, es posible llevar adelante el mapeo hacia la base de datos.

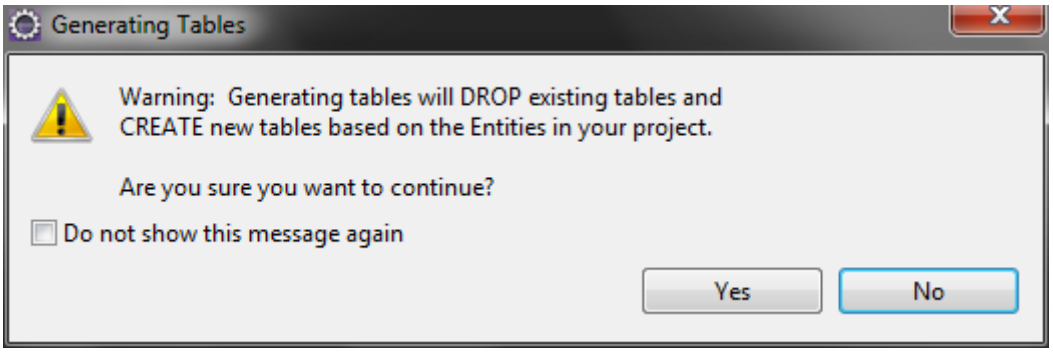
JPA automáticamente creará clases, claves primarias, tablas intermedias, secuencias y todo lo que sea necesario para reflejar el modelo de datos orientado a objetos, en una base de datos relacional.

Para realizar esta tarea se deben seguir estos pasos:

- Clic derecho en el proyecto -> JPA Tools -> Generate Tables from Entities
- Elegir la opción de mapeo
    - Database: impactará los cambios directamente en la base de datos a partir de la conexión previamente establecida
    - Sql-script: creará un script en la ubicación especificada en Schema Generation de persistence.xml, para luego ejecutarlo sobre la base de datos
    - Both: realizará ambos procesos. Impactará los cambios en la base, y guardará el script correspondiente



Eclipse alertará de que usar directamente la base de datos borrará y posteriormente creará las tablas a partir de las clases de entidad.



Una vez finalizado este proceso, se verifica en la base de datos si se crearon correctamente las tablas, secuencias y/o demás elementos correspondientes.