Mapeo directo

Anteriormente vimos el mapeo inverso, donde a partir de una base de datos previamente creada obteníamos cada una de las clases y sus relaciones correspondientes. Ahora veremos el otro caso, donde no tenemos la base de datos creada.

Se debe crear una base de datos con solo el nombre, es decir, vacía.

Creamos en Eclipse una conexión nueva a la base de datos vacía creada.

Creamos un nuevo proyecto maven, en el repositorio git con las dependencias que corresponda, vamos a las propiedades/Facets y seleccionamos JPA.

* Group Id vendría ser el dominio de manera inversa, ej. ar.edu, org.edu

En Schema Generation del persistence.xml, este indica como se va comportar JPA para la generación de las tablas y demás. En Database action seleccionar “Create” para la creación de las secuencias, etc.

Ahora vamos a src/main/java creamos un nuevo paquete con nombre modelo que va contener las clases. Dentro del paquete modelo vamos a crear JPA Entity, introducimos el nombre de la clase en este caso Accion, se crean los atributos:

* Long id
* String nombre

Seleccionamos finish y podemos observar cómo se creó la clase muy similar a las creadas con el proceso “Mapeo inverso”.

Vamos a crear las siguientes clases:

* Permiso {id, Entidad, Acción}, se debe especificar de la forma “Paquete.Entidad” ej. Modelo.Entidad.
* Acción {id, nombre}
* Entidad {id, nombre},
* Rol {id, nombre, List<Permiso>} será una relación bidireccional.}
  + Permisos 🡪Type: List<model.Permiso>
* Nota: JPA necesita la anotación @Entity, @Id y un construcción sin argumentos para que una clase sea persistente.

Una vez creadas las tablas, ahora vamos a ver la estrategia de generación “Secuencia”de los valores para la clave primaria.

Lo hacemos mediante:

@Id

@GeneratedValue(strategy = GenerationType.SEQUENCE, generator = “entidad\_id\_seq”)

Luego después de la anotación @Entity:

@SequenceGenerator (name = “entidad\_id\_seq”, sequenceName = “entidad\_id\_seq”, initialValue = 0, allocationSize = 1)

Anotaciones para la asociación

Tener en cuenta que una clase es dominada y otra es dominante, por lo general ésta última es la que tiene el “muchos” en la asociación.

En este caso Permiso es la clase dominante

@ManyToOne()

Private Entidad entidad;

@ManyToMany()

@JoinTable(name”relación\_permiso\_rol”) // relación bidireccional

Private List<Permiso> permisos;

Permiso será la dominada

@ManyToMany(mappedBy= “permisos”)

Private List<Rol> roles;

Creación del archivo orm.xml

Archivo de configuración donde estableceremos como se va a manejar el mapeo relación, uno de ellos las estrategias en cascada.

Para crearlo hacemos click derecho en el proyecto, nuevo/JPA ORM Mapping File, dejamos por defecto y damos finish.

Ahora vamos a persistence.xml, hacemos click derecho y seleccionamos la opción Synchronize Class List

Mapeo de los objetos hacia la base de datos

Click derecho en el proyecto vamos a JPA Tools/Generate Tables from Entities. Saldra una ventana con 3 opciones:

* DataBase
* Sql-script
* Both (ambos)

Seleccionamos contra la base de datos, saldrá una alerta sobre el DROP de las tablas existentes, le damos “Yes”

Luego de la finalización del proceso de mapeo mediante consola, podemos ver las tablas creadas y sus secuencias en la base de datos correspondiente.

Ya podemos crear una nueva clase “test” con el main para poder probar la conexión y comunicación con la base de datos.