Patricia García Ferrández

# DAO EXPLICACIÓN

Este archivo Dao no está hecho por mí, he recibido ayuda de mis compañeros pero no soy capaz de realizarlo por mi cuenta. Sin embargo, sí que entiendo el código:

Para ejecutar el DAO se parte de una configuración que conecte la base de datos donde queremos realizar el CRUD y Java. Se crea para eso una interfaz **Configuration** con los métodos para recoger los datos necesarios para esta conexión: usuario, contraseña y url de la BD. Esta interfaz se implementa en la clase **ConfigurationImp**, donde se definen los métodos y se crea otro método para instanciar una nueva configuración sólo en el caso de que esta sea nula. Este método será el que se ejecute cuando se quiera abrir una conexión para ejecutar consultas.

Se tiene una clase abstracta **Entity** de la que extenderán las entidades, como Pizza, que tiene el atributo id; varios métodos para obtenerlo; un método para validar id, que consiste en hacer dos casteos y comprobar que sigue devolviendo un objeto que se corresponde con un UUID; y el override de los métodos equals y hashCode, para que no identifique como distintos dos objetos que a nuestro entender son iguales (mismo id).

Se crea otra interfaz **Runables** que contiene dos métodos: uno para obtener la sentencia SQL que se desea ejecutar y otro para ejecutar la sentencia PreparedStatement que se le pase.

Se tiene, además, otra interfaz, **Statement<T>**, con un único método, que recibe una sentencia ResultStatement y el objeto al que se le quiere aplicar, y no devuelve nada. Esta interfaz se implementará más adelante en forma de función lambda, en la que se darán las instrucciones para preparar la consulta.

La clase **RunablesImp** implementa la interfaz Runables. Para ejecutar una consulta se debe llamar a este constructor, al que se le mete un String con la consulta SQL, el objeto sobre el que realizarlo y el Statement<T> a aplicar, que será distinto para cada objeto. El método run es el encargado de ejecutarlo.

Los métodos “INSERT” y “SELECT” correspondientes a SQL se definen en la interfaz **EntityManager**, junto a los métodos addStatement, que recibe el objeto al que se le quiere pasar el statement, la sentencia SQL y el statement a realizar, y addRangeStatement, que hace lo mismo pero ejecuta varias statements. Al método select() se le pasa un argumento **Resultset<T>**, que es una interfaz que reúne los datos que se obtienen de una consulta SQL (ResultSet) y la entidad a la que se quiere llamar (T). Es una interfaz similar a Statement<T> y creada con la misma lógica.

Estos métodos se implementan y definen en la clase **EntityManagerImp**, además de dos métodos más que sirven para obtener la configuración fijada anteriormente. Se crea una lista de Runables para almacenar los statement que se escriban y se llama a Configuration para establecerla como null. Los métodos addStatement y addRangeStatement se definen de forma que se crean Runables con los argumentos que se les pasa y se devuelven en forma de entidad,

Los métodos save() y select(…) se definen de forma que siguen los pasos:

1. Establecer una conexión
2. Crear una sentencia (PreparedStatement)
3. Ejecutarla
4. Procesar el objeto resultante (ResultSet)
5. Cerrar la conexión

Método **save()**:

En orden de ejecución, se comienza estableciendo la conexión y se "apaga" el AutoCommit para que no de resultado a menos que se ejecuten todas las sentencias que escribamos. A continuación, se introducen los runables que se hayan creado para crear un PreparedStatement con la sentencia SQL que incluyan y se ejecutan. Tras terminar la iteración, se hace explícitamente un commit. Si se produjera una excepción de SQL se ejecuta un rollback. Tras este try-catch, se hace un clear de la lista runables, para que no ejecute querys de otros métodos, y se hace otro try-catch para cerrar la conexión.

Método **select(…):**

Este método recibe la clase del objeto que se quiere instanciar y el ResultSet, que será una función lambda que asigne los datos que reciba procedentes de la base de datos a los atributos de la entidad que se esté creando.

Esta clase puede recibir varios valores, con lo que necesita de un bucle while que vaya creando entidades para cada consulta.

Lo que se pretende con este código es crear interfaces fluidas que, aunque hacen más complejo y abstracto definir las clases y los métodos, son más sencillas de trabajar a la hora de ejecutarlos, ya que se les va llamando de seguido y facilita mucho ejecutar una sentencia. Es lo que se observa en la clase **App**, donde hacer un save() o un select(…) de una pizza es tan sencillo como in llamando a los métodos "punto tras punto" e indicarle los statements.