Código DAO

Irene Guillén Zapata

Interfaz Statement

Mediante esta interfaz, se envían las órdenes SQL individuales a la base de datos a través del controlador JDBC y se recogen los resultados de las mismas.

Es una interfaz que nos protege de las inyecciones SQL, es decir, evita que otra persona introduzca sentencias SQL no deseadas en el código.

Clase Pizza

En la clase Pizza definimos los atributos id, name y url.

Interfaz IRunables

Esta interfaz tiene dos métodos, uno de ellos (public String getSQL()), sirve para llamar a la transacción SQL. El otro, es un método que no devuelve nada, pero la característica principal de un objeto PreparedStatement es que, recibe una declaración SQL cuando se crea. La ventaja de esto es que, en la mayoría de los casos, esta declaración SQL se envía al DBMS de inmediato, donde se compila.

Esto significa que cuando se ejecuta PreparedStatement, el DBMS puede simplemente ejecutar la declaración SQL PreparedStatement sin tener que compilarla primero.

Clase Runables

Esta clase es genérica, es decir, puede tomar el valor de cualquier cosa que le indiques e implementa la interfaz IRunable descrita anteriormente. Se definen los atributos sql, entity y statement, el constructor y dos métodos.

El primer método te devuelve un String del atributo sql. El segundo no te devuelve nada, implementa un PreparedStatement de los atributos statement y entity.

Interfaz Resultset

En esta interfaz obtenemos por parámetros un ResultSet y una entity genérica. Su función es encargarse de insertar los datos obtenidos del ResultSet a la entity genérica usando Lambda.

Interfaz IEntityManager

En esta interfaz definimos 4 métodos sin cuerpo. Un método save() en el que se guardarán las transacciones, los otros 3 son métodos genéricos, 2 de ellos de la interfaz misma, en los que con el addStatement añades un statement (sentencia) a una lista que se va a ejecutar para la base de datos y con el addRangeStatement, le pasas varios statement en vez de uno.

El ultimo es el método optional Resultset, el cual usamos para definir el select.

Clase Abstracta Entity

En esta clase definimos la creación del atributo UUID, los getters y setters del mismo y la validación a la hora de pasarlo en el Select para que se pueda leer a través de la BBDD. También definimos los métodos equals y hashcode para devolvernos el ID y realizar la comparación correspondiente y comprobar si se cumple la igualdad.

Clase Converter

Esta clase sirve para convertir un UUID a ByteArray para poder usarlo posteriormente. También definimos el caso contrario.

Interfaz IConfiguration

Es una interfaz que define los métodos getUser(), getPassword() y getUrl(), se encargarán de obtener los datos de usuario, la contraseña y la url de la base de datos.

Clase Configuration

Contiene un atributo de la interfaz IConfiguration que se inicializa a null. Por otro lado, tiene un método de esa misma interfaz en la que instancia un objeto de la presente clase si el atributo configuration es null y, si no lo es, retornará ese atributo.

A continuación, implementa y define los 3 métodos de la interfaz IConfiguration, cada una te devuelve la información correspondiente de las variables de entorno de la base de datos para poder realizar la conexión.

Implementación EntityManager

En esta clase, por un lado, tenemos un ArrayList de la interfaz Runables, el constructor y un atributo inicializado a null de la interfaz Configuration. Se distinguen varios métodos:

- **Método save():** inicializa la conexión a null y, en un try catch se establece la conexión con la base de datos mediante el DriverManager, llamando a los métodos de la interfaz Configuration: getUser, getPassword y getUrl. En un bucle se recorre el ArrayList de la interfaz Runables y rescata con el getSQL la consulta realizada y la ejecuta. Por último, en los catch se implementan las excepciones. En resumen, este método es de escritura, es decir, en él se ejecutarán las consultas de update, insert o delete.

- **Método <T> T Select:** básicamente tiene la misma estructura que el anterior método, sin embargo, te devuelve la información de una consulta SQL de un SELECT.

- **Método addStatement:** pertenece a la interfaz IEntityManager en la cual, se guardará en el ArrayList mencionado anteriormente la sentencia que se va a ejecutar en la base de datos y la retorna.

- **Método addRangeStatement:** aquí realiza la misma acción que el método anterior, con la diferencia de que, en éste se almacenará en el ArrayList varias sentencias a la vez que se ejecutarán en la base de datos y los devolverá.

- **Método buildConnection:** es un método estático de la interfaz IEntityManager y le pasa como parámetro una variable de la interfaz IConfiguration y retorna una nueva instancia de la clase de la variable pasada como parámetro, es decir, construye una nueva conexión.

App

Este es el main, donde se ejecutará el programa. Se instancia un objeto de la clase Pizza, proporcionándole los valores de los atributos de dicha clase. Una vez hecho, llamamos al método builConnection para construir la configuración, es decir, la conexión a la base de datos. Seguidamente, con el método addStatement añadimos a la base de datos la sentencia SQL de INSERT del objeto de la clase Pizza.

En el caso del save, vamos a guardar en la base de datos una nueva pizza, donde, con los statements correspondientes, introducimos el orden en el cual guardamos los valores de id, name y url. Dicho id lo transformamos con un método de UUID a ByteArray para poderse leer ya que, es del tipo Binary. Por último, ejecutamos el método Save para guardarla.

Seguidamente se observan las sentencias de setInt y dos setString, en los que se les pasa el número de la columna y el nombre, por ejemplo, en setInt tendríamos el número 1 y la entidad id correspondientes a cómo está distribuida esa tabla en la base de datos. En el caso del Select el código sería similar, solo que usamos la interfaz ResultSet definida anteriormente para sacar a través de la clase Entity los atributos id, name y url.