CONEXIÓN A BASES DE DATOS NO RELACIONALES (MONGODB)

Acceso a Datos

3 DE NOVIEMBRE DE 2024

ERIK AMO TOQUERO

Miniintroducción

Quizá este es el acceso a bases de datos más complejos que se puede ver, porque al ser un modelo de datos no relacional tiene más libertad de uso y no es fácil automatizar todo.

Para este caso, vamos a usar MongoDB, un modelo de bases de datos no relacionales que trabaja con Documentos de pares Clave-Valor.

```
<!--Dependencia de MongoDB Manager-->
<dependency>
  <groupId>org.mongodb</groupId>
  <artifactId>mongodb-driver</artifactId>
  <version>3.12.11</version>
</dependency>
```

Guardar cerca también Gson, un plugin oficial de Google que permite parsear fácilmente de JSON a objeto de Java

```
<!--Dependencia de Gson/Parser de JSON a Objeto de Java-->
<dependency>
  <groupId>com.google.code.gson</groupId>
  <artifactId>gson</artifactId>
  <version>2.11.0</version>
</dependency>
```

Acceso a la Base de Datos

Para acceder a la base de datos hay que crear varias cosas:

1. **Conexión (MongoClient):** Objeto que se conecta con el gestor
2. **Base de datos (MongoDatabase):** Objeto que se conecta a la base de datos en concreto
3. **Colección (MongoCollection<Document>):** Objeto que se conecta a la colección de la base de datos (La colección es la tabla por así decirlo)

```
public MongoCollection<Document> conectar(){
    try {
        MongoClient cliente = new MongoClient(new
MongoClientURI("mongodb://"+user+":"+pass+"@"+host+":"+port+"/?authSou
rce="+source)); //Se conecta a la base de datos en función de la url
        MongoDatabase database = cliente.getDatabase("CochesMultas");
        MongoCollection<Document> coleccion =
database.getCollection(collection);
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

- **User:** El nombre de usuario que puede acceder a la BBDD
- **pass:** Su contraseña
- **Host:** Dirección a la base de datos
- **Port:** Puerto
- **Source:** Desde dónde quieres acceder

```
user=admin
password=1234
host=localhost
port=27017
source=admin
```

Métodos CRUD

Al igual que en las bases de datos relacionales, CRUD significa Create, Read, Update y Delete. Aquí los métodos son bastante distintos entre ellos y tienen distintas formas de hacerse, dependiendo si usan o no la librería GSON:

Los datos se guardan en bloques de objetos llamados Document, que no dejan de ser conjuntos clave valor como si de un mapa se tratara. GSON lo que hace es tratar estos documentos automáticamente y guardar la información en un objeto

De todos modos, aquí voy a tratar de enseñar las dos formas de hacerlo, tanto con GSON como sin él. Voy a trabajar con el siguiente objeto:

```
public class Coche {
    private int id;
    private String matricula;
    private String marca;
    private String modelo;
    private String tipo;

}
```

Inserción

Guarda una serie de datos como documento en una colección:

Sin GSON: Añadir los datos uno a uno con .append(clave, valor)

```
public void insertarCoche(Coche coche) {
    try {
        Document newCar = new Document("matricula",
coche.getMatricula())
                .append("marca", coche.getMarca())
                .append("modelo", coche.getModelo())
                .append("tipo", coche.getTipo()); //Crea el documento
con los datos del coche
        collection.insertOne(newCar); //Inserta el documento
    }catch (Exception e){
    }
}
```

Con GSON: Pasar el objeto a JSON, parsearlo e insertarlo:

```
public void insertarCoche(Coche coche) {
    try {
        Document newCar = Document.parse(gson.toJson(coche)); //Crea
el documento con los datos del coche
        collection.insertOne(newCar); //Inserta el documento
    }catch (Exception e){
    }
}
```

Edición

Edita el objeto que le pase usando $set.

Sin GSON: Crea una colección con el dato de referencia y otra con $set y otro documento dentro:

```
public void actualizarCoche(Coche coche) {
collection.updateOne(new Document("matricula", coche.getMatricula()),
                            new Document("$set",
                                    new Document("marca", coche.getMarca())
                                            .append("modelo", coche.getModelo())
                                            .append("tipo", coche.getTipo())));
}
```

Eliminación

Borra el objeto que elija el usuario. Este método utiliza Filters, un objeto con métodos estáticos de comparación (eq “equals”, gte “greater-than-equals”, lte “less-than-equals”...)

```
    try{
        collection.deleteOne(Filters.eq("matricula", coche));
    } catch (Exception e){
    }
}
```

Búsqueda

Busca una colección y la parsea. Se puede hacer tanto con Gson como sin él.

Con GSON: Coger el documento y parsearlo con fromJson:

```
public Coche getCoche(String matricula){
    Coche coche = null;
    try {
        Document document = collection.find(Filters.eq("matricula", matricula)).first();
        if (document != null) {
            coche = gson.fromJson(document.toJson(), Coche.class);
        }
    }catch (Exception e){
        System.out.println("Error de base de datos");
    }
    return coche; //Devuelve el coche, tanto si es nulo como si no
}
```

Sin GSON: Guardar a mano cada atributo:

```
public Coche getCoche(String matricula){
    Coche coche = null;
    try {
        Document document = collection.find(Filters.eq("matricula", matricula)).first();
        if (document != null) {
            coche = new Coche(
                    document.getString("matricula"),
                    document.getString("marca"),
                    document.getString("modelo"),
                    document.getString("tipo")
            );
        }
    }catch (Exception e){
        System.out.println("Error de base de datos");
    }
    return coche; //Devuelve el coche, tanto si es nulo como si no
}
```

Listado

Usa el mismo método que la búsqueda, pero guarda todos los resultados que encuentre en una lista:

Con GSON:

```
public ArrayList<Coche> listarCoches(){ //Función que busca en la base de datos
    ArrayList<Coche> coches = new ArrayList<>();
    MongoCursor<Document> cursor = collection.find().iterator();
//Crea un iterador que recorre toda la colección de la base de datos
    try {
        while (cursor.hasNext()){ //Mientras haya datos a recorrer en el cursor
            String json = cursor.next().toJson(); //Convierte el dato del cursor en json
            Coche coche = gson.fromJson(json, Coche.class);
//Convierte el json a clase con la librería Gson
            coches.add(coche); //Añade el coche
        }
    }catch (Exception e){
        System.out.println("Error de programa");
    } finally {
        cursor.close(); //Cierra el cursor
    }
    return coches; //Devuelve la lista
}
```

Sin GSON:

```
public ArrayList<Coche> listarCoches(){ //Función que busca en la base de datos
    ArrayList<Coche> coches = new ArrayList<>();
    MongoCursor<Document> cursor = collection.find().iterator();
//Crea un iterador que recorre toda la colección de la base de datos
    try {
        while (cursor.hasNext()){ //Mientras haya datos a recorrer en el cursor
            Document document = cursor.next(); //Coge el dato del cursor
            Coche coche = new Coche(
                    document.getString("matricula"),
                    document.getString("marca"),
                    document.getString("modelo"),
                    document.getString("tipo")
            ); //Convierte el json a clase con la librería Gson
            coches.add(coche); //Añade el coche
        }
    }catch (Exception e){
        System.out.println("Error de programa");
    } finally {
        cursor.close(); //Cierra el cursor
    }
```