Code Nation

Sequelize with MySQL

Mapping Objects to Databases

{codenation}®

Introduction

What is Sequelize

- Sequelize is a software library known as an Object Relational Mapper.
- Its job is to act as a converter between JavaScript objects and rows in a SQL based database.
- It allows database interactions to be easier to understand and visualise.

What is Sequelize

- It allows a programmer to define an object (known as a model) that represents a row in a table.

- Models provide crud operations (create, read, update, & delete) making the code easier to work with.

Installation

Installing Sequelize

- Sequelize is a library that uses other libraries and we must install more than just seqelize to work with mysql.

- mysql2 must be installed too.

- npm install sequelize mysql2

Configuration

Configuring Sequelize

- As with any library that interacts with a networked application, the mysql server software needs to be running.

- Please check you have either a:
  - Docker container running
  - Mysql process running
  - Connection string to an online mysql system

```
const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('test', 'root', 'password', {
  host: 'localhost',
  dialect: 'mysql'
});

const main = async() => {
    try {
      await sequelize.authenticate();
      console.log('Connection has been established successfully.');
    } catch (error) {
      console.error('Unable to connect to the database:', error);
    }
    await sequelize.close();
    process.exit();
}

main();
```

Configuring Sequelize

- SQL databases require slightly more code to set up and manage than mongo based databases, this isn't a bad things, just different.

- Something important to notice is that it is very important that connections are closed when done with.

Models

Models

- SQL databases require slightly more code to set up and manage than mongo based databases, this isn't a bad things, just different.

- Something important to notice is that it is very important that connections are closed when done with.

Models

```
const { Sequelize, DataTypes } = require("sequelize");
const connection = new Sequelize("database", "username", "password", {
  host: "localhost",
  dialect: "mysql"
});

const Cat = connection.define("Cat", {
  name: {
    type: DataTypes.STRING,
    allowNull: false
  }
}, {
  // This ensures unique fields will remain unique
  indexed: [{unique: true, fields: ["name"]}]
});
```

Models

- Using the Sequelize define method a “cat” model can be created that has a name property that will be of a string data type.

- You may have as many properties as you like, some may be required, some may not, some may be strings, some may not, it all comes down to the design of you application and what you need.

Models

- Once the models have been created they must be synchronised with the database

```
1 const main = async() => {
2     await Cat.sync({alter: true});
3 }
```

Create

Models

- To create an instance of a schema, it is familiar enough to creating instances of a class and follows many familiar conventions.

- Assuming the definition of Cat earlier, the way to create a cat would be...

Models

```
1 const cat = Cat.build({ name: “Fluffy” });
2 await cat.save();
```

Models

- The important thing to note is that simply creating an object (as in the previous slide) only creates it in memory and does not immediately save it to the database.

- Saving it to the database is another step.

Read

Models

- Retrieving data using sequelize follows SQL conventions, where by default SELECTING will return multiple results.

```
1 const cats = await Cat.findAll();
2 console.log("All cats:", JSON.stringify(cats, null, 2));
```

Models

- It is possible to filter results.

```
1 Cat.findAll({
2    where: {
3       name: “Fluffy”
4    }
5 });
```

Models

- It is possible to filter results using more comprehensive means.

- This would use the Op module from the sequelize package. It includes the and, or, =, > etc making the queries.

Update

Models

- Updating models in a database follows a familiar syntax to other methods in sequelize.

- A update object is given, followed by a filter, so objects will be changed as per the filter.

- Rows will update based on the update object where the filter matches records.

Models

```
1 await Cat.update({ name: "Doe" }, {
2    where: {
3       name: “Fluffy”
4    }
5 });
```

Delete

Models

- Deleting models in a database follows a familiar syntax to other methods in sequelize.

- When deleting the delete method is called with a constraint to ensure only that which needs to be deleted, is.

Models

```
1 await Cat.destroy({
2     where: {
3         name: “Fluffy”
4     }
5 });
```

Questions