SQL ORM USING MYSQL, NODE, EXPRESS, AND SEQUELIZE

FULL STACK SKILLS BOOTCAMP

SQL ORM USING MYSQL, NODE, EXPRESS, AND SEQUELIZE

- **Lesson Overview:**
- In this lesson, we will be introduced to:

1. What is an ORM
2. What is Sequelize
3. Data Models
4. CRUD operations
5. Data Seeding

WHAT IS AN ORM?

- ORM stands for Object-Relational Mapping.
- It allows developers to interact with a database using an object-oriented paradigm.
- Abstracts SQL queries into methods and models.
- Simplifies database interactions by using objects and relationships.

WHY USE AN ORM?

- Reduces boilerplate SQL code.
- Enhances code maintainability and readability.
- Provides database-agnostic capabilities.
- Helps manage relationships between data more efficiently.

WHAT IS SEQUELIZE?

- Sequelize is a promise-based Node.js ORM for MySQL, PostgreSQL, MariaDB, SQLite, and Microsoft SQL Server.
- Provides built-in support for CRUD operations.
- Allows defining models and relationships easily.
- Includes support for database migrations and seeding.

INSTALLING SEQUELIZE & ADDING TO AN EXPRESS SERVER

- **Installation:**

```
npm install sequelize mysql2
```

- **Setting up in Express:**

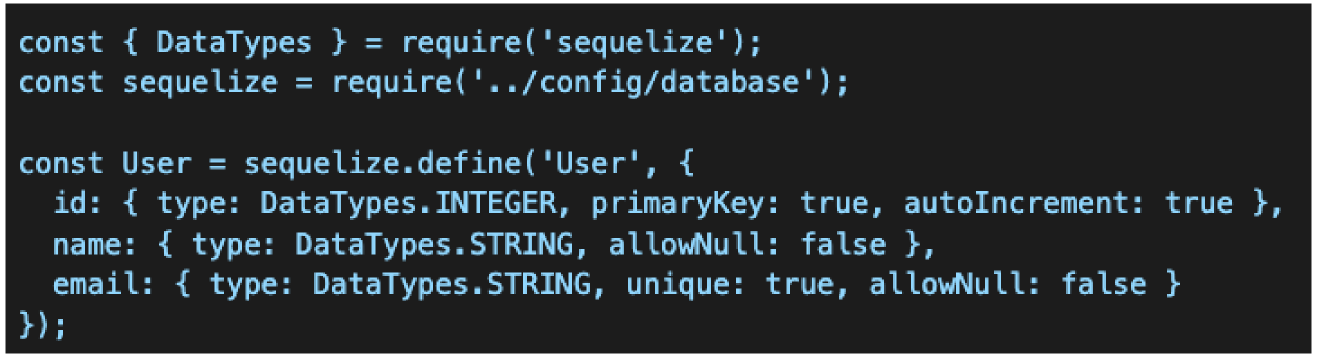
  demo…

```
const { Sequelize } = require('sequelize');
const sequelize = new Sequelize('database_name', 'username', 'password', {
  host: 'localhost',
  dialect: 'mysql'
});
```

WHAT ARE MODELS?

- Models define the structure of a table in the database.
- Represent database tables as JavaScript classes.
- Example Model Definition….

  Demo…



```
const { DataTypes } = require('sequelize');
const sequelize = require('../config/database');

const User = sequelize.define('User', {
  id: { type: DataTypes.INTEGER, primaryKey: true, autoIncrement: true },
  name: { type: DataTypes.STRING, allowNull: false },
  email: { type: DataTypes.STRING, unique: true, allowNull: false }
});
```

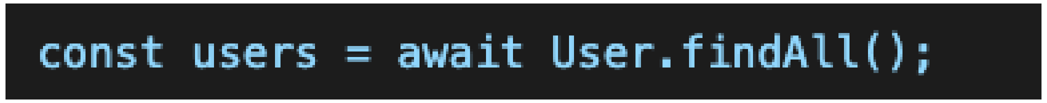
MODEL METHODS FOR CRUD

- **Creating a Record:**

```
await User.create({ name: 'John Doe', email: 'john@example.com' });
```

MODEL METHODS FOR CRUD

- **Reading Records:**



```
const users = await User.findAll();
```

MODEL METHODS FOR CRUD

- **Update a Record:**

```
await User.update({ name: 'Jane Doe' }, { where: { id: 1 } });
```

MODEL METHODS FOR CRUD

- **Deleting a Record:**

```
await User.destroy({ where: { id: 1 } });
```

MODEL RELATIONSHIPS

- Sequelize supports various relationships:
- One-to-One (hasOne)
- One-to-Many (hasMany)
- Many-to-Many (belongsToMany)

Example:

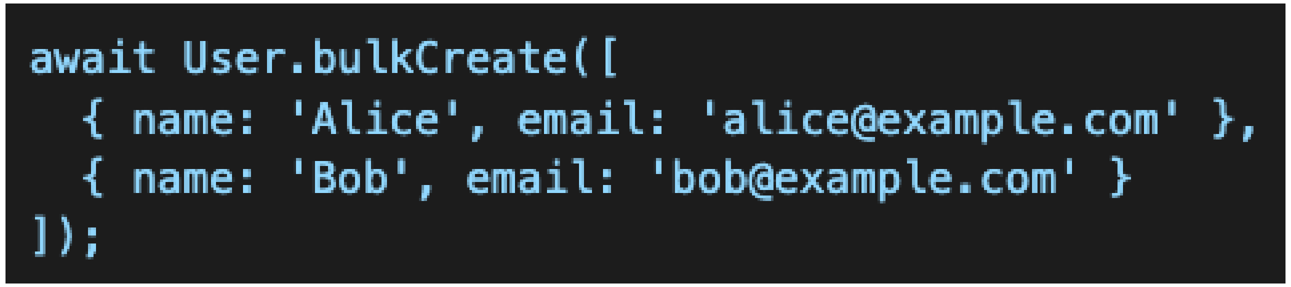
```
User.hasMany(Post);
Post.belongsTo(User);
```

WHAT IS DATABASE SEEDING?

- Seeding refers to populating a database with initial data.
- Useful for testing and development.
- Allows automated insertion of sample records.

SEEDING A DATABASE USING BULKCREATE

- **Example:**



```
await User.bulkCreate([
  { name: 'Alice', email: 'alice@example.com' },
  { name: 'Bob', email: 'bob@example.com' }
]);
```

USEFUL ONLINE RESOURCES

- Sequelize Official Docs
- Node.js Documentation
- MySQL Official Docs
- MDN Web Docs

CONCLUSION

- ORM simplifies database interaction.
- Sequelize is a powerful tool for Node.js applications.
- Practice by building small projects with Sequelize.

QUESTIONS?