BANCO DE DADOS COM NODE JS

Douglas Nassif Roma Junior

/douglasjunior

/in/douglasjunior

nassifrroma@gmail.com

Slides: https://github.com/douglasjunior/Bootcamp-DB1-2024

AGENDA

- Banco de dados SQL
  - MySQL
- Sequelize JS
- Definição dos modelos
- Usando os modelos
- Consultas
- Associações
- Transações
- Referências

BANCO DE DADOS RELACIONAL

- Assim como em linguagens como Java e C#, no NodeJS é preciso instalar o módulo (driver) para conexão com o banco de dados desejado.

- Para consultar a disponibilidade do banco de dados desejado, consulte o nome do banco no repositório http://npmjs.com

- Por exemplo, para MySQL poemos utilizar o módulo `mysql2` e para SQLite o módulo `sqlite3`.

- Para nossos exemplos vamos utilizar:

```
$ npm install mysql2
```

ANTES DE INICIAR

- Vamos criar um banco de dados de exemplo:

```
CREATE DATABASE `exemplo`;
```

```
USE `exemplo`;
```

- Vamos criar uma tabela para armazenar usuários:

```
DROP TABLE `usuarios`;
```

```
CREATE TABLE `usuarios` (
  `id` int NOT NULL AUTO_INCREMENT,
  `nome` varchar(255) DEFAULT NULL,
  `idade` int DEFAULT NULL,
  `data_criacao` datetime NOT NULL DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`id`)
);
```

EXEMPLO MYSQL

```
const mysql = require('mysql2');

const connection = mysql.createConnection({
  host: 'localhost', user: 'root',
  password: '1234', database: 'exemplo'
});

connection.query(
  'SELECT * FROM `usuarios` WHERE `nome` = "Douglas" AND `idade` > 20',
  function (err, results, fields) {
    if (err) {
      console.error(err);
      return;
    }

    console.log(results); // resultado contendo as linhas da consulta
    console.log(fields); // campos contendo metadados sobre os resultados
  }
);
```

SEQUELIZE JS

Sequelize

- Sequelize é um framework para Mapeamento de Objetos Relacionais (ORM) para Node JS.
- Possui suporte à PostgreSQL, MySQL, SQLite e MSSQL.
- Possui funcionalidades sólidas como relacionamento entre entidades e transações.

INTRODUÇÃO AO SEQUELIZE JS

- Para iniciar com o Sequelize basta instalar o módulo:

```
$ npm install sequelize
```

- Em seguida, instalar o driver para o banco de dados desejado:

```
$ npm install mysql2
```


Documentação oficial: https://sequelize.org/docs/v6/getting-started/

INTRODUÇÃO AO SEQUELIZE JS

- Conectando ao banco de dados MySQL.

```
const database = 'exemplo';
const user = 'root';
const password = 'root';

const sequelize = new Sequelize(
  database,
  user,
  password,
  {
    dialect: 'mysql',
    host: '127.0.0.1',
    port: '3306',
    define: {
      // desabilita colunas createdAt e updatedAt
      timestamps: false
    }
  }
);
```

INTRODUÇÃO AO SEQUELIZE JS

- Testando a conexão.

```
sequelize.authenticate()
   .then(() => {
      console.log('Banco de dados conectado com sucesso.');
   }).catch((err) => {
      console.error("Não foi possível se conectar ao banco de dados.", err);
   });
```

DEFINIÇÃO DOS MODELOS

- Vamos definir a entidade que representará a tabela "usuario" no banco de dados.

```
const Usuario = sequelize.define('usuarios', {
  id: {
    type: DataTypes.INTEGER,
    primaryKey: true,
    allowNull: false,
    autoIncrement: true
  },
  nome: {
    type: DataTypes.STRING(200),
    allowNull: true,
    defaultValue: null,
  },
  idade: {
    type: DataTypes.INTEGER,
    allowNull: true,
    defaultValue: null,
  },
  data_criacao: {
    type: DataTypes.DATE,
    allowNull: false,
    defaultValue: DataTypes.NOW
  },
});
```

https://sequelize.org/docs/v6/core-concepts/model-basics/

USANDO OS MODELOS

- Inserindo um elemento no banco de dados.

```
Usuario.create({
    nome: 'Douglas Junior', email: 'nassifrroma@gmail.com'
}).then(usuario => {
    // você pode acessar agora o usuário criado
    // através da variável "usuario"
    console.log("Usuário inserido:", JSON.stringify(usuario));
})
```

https://sequelize.org/docs/v6/core-concepts/model-instances/

USANDO OS MODELOS

- Buscando por chave primária

```
Usuario.findByPk(1)
  .then(usuario => {
    // Retorna o usuário correspondente ao ID especificado,
    // ou Null caso não seja encontrado.

    if (usuario) {
      console.log('Usuário encontrado.');

    } else {
      console.log('Usuário não encontrado.');
    }
  }).catch(err => {
    console.error("Erro ao consultar usuário.", err);
  })
```

https://sequelize.org/docs/v6/core-concepts/model-querying-basics/

USANDO OS MODELOS

- Buscando por um elemento específico

```
Usuario.findOne({
    where: {
        nome: 'Douglas Junior'
    }
}).then(usuario => {
    // Retorna o primeiro usuário com a condição especificada,
    // ou Null caso não seja encontrado.
})
```

https://sequelize.org/docs/v6/core-concepts/model-querying-basics/

USANDO OS MODELOS

- Atualizando um elemento no banco de dados.

```
// Maneira 1
usuario.nome = 'Douglas Nassif'
usuario.save().then(() => { });

// Maneira 2
usuario.update({
    nome: 'Douglas Nassif'
}).then(() => { });
```

```
// Maneira 3
Usuario.update({
    nome: 'Douglas Nassif'
},{
    where: {
        id: 123
    }
}).then(() => { });
```

https://sequelize.org/docs/v6/core-concepts/model-querying-basics/

USANDO OS MODELOS

- Excluindo um elemento no banco de dados.

```
// Maneira 1
usuario.destroy()
    .then(() => { });

// Maneira 2
Usuario.destroy({
    where: {
        id: 1
    }
}).then(() => { });
```

https://sequelize.org/docs/v6/core-concepts/model-querying-basics/

CONSULTAS

- Definindo quais atributos devem ser retornados na consulta.

```
Usuario.findAll({
    // SELECT `nome`, `idade` FROM ....
    attributes: ['nome', 'email']
}).then(usuarios => {
    console.log('Usuários selecionados:', JSON.stringify(usuarios));
})
```

https://sequelize.org/docs/v6/core-concepts/model-querying-basics/

CONSULTAS

- Executando funções do banco de dados, como COUNT, MAX, MIN, etc.

```
Usuario.findAll({
    attributes: [
        [sequelize.fn('COUNT', sequelize.col('id')), 'qtd_usuarios']
    ]
}).then(resultado => {
    console.log('Quantidade de usuários:', JSON.stringify(resultado));
})
```

https://sequelize.org/docs/v6/core-concepts/model-querying-basics/

CONSULTAS

- Filtrando consultas com “where”.

```
Usuario.findAll({
    where: {
        nome: {
            [Op.like]: '%douglas%’
        }
    }
}).then(usuarios => {
    console.log('Usuários selecionados:', JSON.stringify(usuarios));
})
```

https://sequelize.org/docs/v6/core-concepts/model-querying-finders/

CONSULTAS

- Filtrando e contando o total de registros, útil para uso em paginação.

```
Usuario.findAndCountAll({
    where: { },
    limit: 10,
    offset: 0,
}).then(usuarios => {
    console.log('Quantidade de usuários:', JSON.stringify(usuarios.count));
    console.log('Usuários selecionados:', JSON.stringify(usuarios.rows));
})
```

https://sequelize.org/docs/v6/core-concepts/model-querying-finders/

ASSOCIAÇÕES

- Associar a entidade "usuario" à "tarefa", onde a tarefa recebe a chave estrangeira do usuário.

```
// O usuário tem muitas tarefas
Usuario.hasMany(Tarefa, {
    onDelete: 'NO ACTION',
    onUpdate: 'NO ACTION'
})

// A tarefa tem a chave estrangeira do usuário
Tarefa.belongsTo(Usuario, {
    onDelete: 'NO ACTION',
    onUpdate: 'NO ACTION'
});
```

https://sequelize.org/docs/v6/core-concepts/assocs/

ASSOCIAÇÕES

- Consultando com JOINS.

```
Usuario.findAll({
    where: { },
    include: [{
        model: Tarefa,
        required: true, // true para inner join, false para left join
    }],
}).then(usuarios => {
    console.log('Usuários com tarefas:', JSON.stringify(usuarios));
})
```

https://sequelize.org/docs/v6/core-concepts/assocs/

TRANSAÇÕES

- Transações são utilizadas para garantir a integração entre diversas ações no banco de dados.

```
sequelize.transaction((transaction) => {
     return Usuario.create({
          nome: 'Douglas Junior', email: 'nassifrroma@gmail.com’,
     }, { transaction }).then(usuario => {
          console.log('Usuário criado:', JSON.stringify(usuario));
          return Tarefa.create({
               titulo: 'Minha tarefa’,
               usuarioId: usuario.id
          }, { transaction })
     }).then(tarefa => {
          console.log('Tarefa criada:', JSON.stringify(tarefa));
     })
}).then(() => {
     console.log('transação comitada');
}).catch(ex => {
     console.error('transação revertida:', ex);
})
```

http://docs.sequelizejs.com/manual/tutorial/transactions.html

REFERÊNCIAS

- Node MySQL2 - https://github.com/sidorares/node-mysql2
- Sequelize JS - https://sequelize.org/docs/v6/
- Promises - https://developer.mozilla.org/pt-BR/docs/Web/JavaScript/Reference/Global_Objects/Promise

DÚVIDAS?

Douglas Nassif Roma Junior

/douglasjunior

/in/douglasjunior

nassifrroma@gmail.com

Slides: https://github.com/douglasjunior/Bootcamp-DB1-2024