BANCO DE DADOS COM NODE JS

Douglas Nassif Roma Junior

/douglasjunior

/in/douglasjunior

douglasjunior.me

nassifrroma@gmail.com

Slides: https://git.io/vdn2c

AGENDA

- Introdução ao Sequelize JS
- Definição dos modelos
- Usando os modelos
- Consultas
- Associações
- Transações
- Referências

INTRODUÇÃO AO 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 --save sequelize
```

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

```
$ npm install --save pg pg-hstore
$ npm install --save mysql2
$ npm install --save sqlite3
$ npm install --save tedious // MSSQL
```

INTRODUÇÃO AO SEQUELIZE JS

- Conectando ao banco de dados SQLite.

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

const sequelize = new Sequelize(null, null, null, {
   dialect: 'sqlite',
   storage: './database.sqlite',
});
```

- Conectando ao banco de dados MySQL.

```
const sequelize = new Sequelize('database', 'user', 'password', {
   dialect: 'mysql',
   host: '172.0.0.1',
   port: '3306'
});
```

INTRODUÇÃO AO SEQUELIZE JS

- Testando a conexão.

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

DEFINIÇÃO DOS MODELOS

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

```
const Usuario = sequelize.define('usuario', {
    id: {
        primaryKey: true,
        type: Sequelize.BIGINT,
    },
    nome: {
        type: Sequelize.STRING(200),
        allowNull: false,
    },
    nascimento: Sequelize.DATEONLY,
    email: Sequelize.STRING(150),
});
```

http://docs.sequelizejs.com/manual/tutorial/models-definition.html

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));
})
```

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

USANDO OS MODELOS

- Buscando por um elemento específico

```
Usuario.findById(123).then(usuario => {
    // Retorna o usuário correspondente ao ID especificado,
    // ou Null caso não seja encontrado.
})

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

http://docs.sequelizejs.com/manual/tutorial/models-usage.html

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(() => { });
```

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

USANDO OS MODELOS

- Excluindo um elemento no banco de dados.

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

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

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

CONSULTAS

- Definindo quais atributos devem ser retornados na consulta.

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

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

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));
})
```

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

CONSULTAS

- Filtrando consultas com “where”.

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

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

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));
})
```

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

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'
});
```

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

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));
})
```

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

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

- Sequelize JS - http://docs.sequelizejs.com
- Conteúdo do treinamento - https://git.io/vdn2c

DÚVIDAS?

Douglas Nassif Roma Junior

/douglasjunior

/in/douglasjunior

douglasjunior.me

nassifrroma@gmail.com

Slides: https://git.io/vdn2c