Sequelize

O que é? O que faz? Como se cria?

Andressa Cruz (Dessa)

Consultora de Software @ Thoughtworks
Pyladies Poa <3
Mãe de Bella, a guaipeca mais linda deste mundo

Cláudia S Moura

Desenvolvedora @ DBC Company
Bacharela em História da Arte
Pyladies Porto Alegre \o/

Sequelize: o que é?

- ORM para Node.js
- Baseado em Promise
- Suporta PostgreSQL, MySQL, SQLite e MSSQL

O que faz?

- ORM: Object-Relational Mapper == Mapeamento Objeto Relacional
- Abstrai o banco de dados relacional
  - Não precisa mexer no SQL;
- Framework's super completos;
  - Geralmente com suporte a vários tipos de BDs

E como se cria?

```
$ npm install --save sequelize    ||   $ yarn add sequelize
```

```
// Definindo o banco:
$ npm install --save pg pg-hstore ||  $ yarn add pg pg-hstore
$ npm install --save mysql2       ||  $ yarn add mysql2
$ npm install --save sqlite3      ||  $ yarn add sqlite3
$ npm install --save tedious      ||  $ yarn add tedious// MSSQL
```

Adicionando no projeto

```
const Sequelize = require('sequelize');
const sequelize = new Sequelize('banco', 'usuário', 'senha', {
  host: 'localhost',
  dialect: 'mysql'
});
```

```
const sequelize = new
Sequelize('mysql://usuario:senha@exemplo.com:5432/nomebd');
```

Testando a conexão

```
sequelize
  .authenticate()
  .then(() => {
    console.log('Conexão estabelecida :D');
  })
  .catch(err => {
    console.error('Errou', err);
  });
```

Iniciando pelo cli

```
Npm install --save sequelize-cli

$ node_modules/.bin/sequelize init
```

Cria as pastas e arquivos:

- **config** - arquivo config, com instruções de como conectar no banco de dados
- **Models** - todas as models do projeto
- **Migrations** - arquivos das migrações
- **Seeders** - arquivos de seed

Configurando

```
{
  "development": {
    "username": "root",
    "password": null,
    "database": "database_development",
    "host": "127.0.0.1",
    "dialect": "mysql"
  },
  "test": {
    "username": "root",
    "password": null,
    "database": "database_test",
    "host": "127.0.0.1",
    "dialect": "mysql"
  },
  "production": {
    "username": "root",
    "password": null,
    "database": "database_test",
    "host": "127.0.0.1",
    "dialect": "mysql"
  }
}
```

Sequelize: Migration

```
../node_modules/.bin/sequelize model:generate --name Pokemon --attributes
number:integer,name:string //cria a model Pokemon

../node_modules/.bin/sequelize model:generate --name Type --attributes
name:string //cria a model Type

../node_modules/.bin/sequelize db:migrate //faz a migração

../node_modules/.bin/sequelize db:migrate:undo //desfaz a migração
```

Models

```
const Pokemon= sequelize.define('pokemon', {
  id: {
    type: Sequelize.INTEGER,
    primaryKey: true
  },
  name: Sequelize.STRING,
  })

const Type = sequelize.define('type', {
  id: {
    type: Sequelize.INTEGER,
    primaryKey: true
  },
  name: Sequelize.STRING
})
```

Exemplos de tipos de dados

```
Sequelize.STRING          //VARCHAR(255)
Sequelize.TEXT            //TEXT
Sequelize.INTEGER         //INTEGER
Sequelize.FLOAT           //FLOAT
Sequelize.DOUBLE          //DOUBLE
Sequelize.BOOLEAN         //TINYINT(1)
Sequelize.DATE            //   DATETIME for mysql/sqlite,
                               TIMESTAMP WITH TIME ZONE for postgres
Sequelize.DATEONLY        //DATE without time.
```

Criando relacionamento

```
Pokemon.belongsToMany(Type, {through: 'PokemonType'});
Type.belongsToMany(Pokemon, {through: 'PokemonType'});
```

*Para relações Belongs-to-many, o parâmetro through é obrigatório.

Tipos de relacionamento

One-To-One

- belongsTo
- hasOne

One-To-Many

- hasMany

ManyToMany

- belongsToMany

Criando o pokémon

```
sequelize.sync()
.then => Type.create({
  name: 'grass'
})
.then => Pokemon.create({
  name: 'Bulbasaur',
  type_id: 1
})
```

Type
ID: 1
Name: grass

Pokemon
ID: 1
Name: Bulbasaur
Type_ID: 1 //grass

Criando o pokémon (outro jeito)

```
const poison = Type.build({
  name: 'poison'
})

const ekans = Pokemon.build({
  name: 'Ekans',
  type_id: 2,
})

poison.save()
ekans.save()
```

Type
ID: 2
Name: poison

Pokemon
ID: 2
Name: Ekans
Type_ID: 2 //poison

Editando o pokémon

```
// jeito 1
poison.name = 'veneno'
poison.save()

// jeito 2
ekans.update({
  name: 'Cobrinha'
})
```

Type
ID: 2
Name: veneno

Pokemon
ID: 2
Name: Cobrinha
Type_ID: 2 //veneno

Buscando todos os pokémons

```
Pokemon.findAll()
.then(pokemons => {
  mostrar(pokemons)
})
```

Pokemon
ID: 1
Name: Bulbasaur
Type_ID: 1 //grass

Pokemon
ID: 2
Name: Ekans
Type_ID: 2 //poison

Excluindo o pokémon

```
Pokemon.destroy({
   where: {
     id: 1
   }
 })
```

Pokemon
ID: 1
Name saur
Typ 1

Obrigada ;P

@aff_andressa

@ClaudiaStrm