Sequelize ORM

Benoit BRIATTE

2015 – 2016

Introduction

- Module node.js

- Permet de communiquer avec une base de données :

- *mysql*
- *mariadb*
- *sqlite*
- *postgres*
- *mssql*

- Installation :
- **npm install sequelize**
- **npm install DRIVER**

Création de la connexion

- Le constructeur de la classe prend 4 paramètres :
    - le nom de la base de données
    - Le nom de l'utilisateur pour se connecter
    - Mot de passe
    - Un objet de configuration

```
var sequelize = require("./sequelize");

module.exports = new Sequelize("dbname", "username", "password", {
    host: "localhost",
    port: 3306,
    pool: false
});
```

Déclaration des tables (1/3)

- Faire un fichier par table dans un dossier
- Inclure tous les fichiers dans l'export du ***index.js*** de ce même dossier

```
var sequelize = require("./sequelize");

module.exports = sequelize.import("user", function (sequelize, DataTypes) {
    return sequelize.define("User", {
        id: {
            type: DataTypes.BIGINT,
            primaryKey: true,
            autoIncrement: true
        },
        email: {
            type: DataTypes.STRING,
            unique: true
        }
    }, {
            paranoid: true,
            underscored: true,
            freezeTableName: true
        });
});
```

User.js

Déclaration des tables (2/3)

- *Paranoid* : Ne supprime pas la ligne lors d'une suppression d'éléments mais affecte uniquement un ***flag***
- *Underscored* : Utilise le format ***snake_case*** pour le nom des tables
- *FreezeTableName* : Permet à Sequelize de ne pas modifier le nom de votre table

Déclaration des tables (3/3)

- Il est possible d'ajouter des méthodes à une table (eq. classe)

```
var sequelize = require("./sequelize");

module.exports = sequelize.import("user", function (sequelize, DataTypes) {
    return sequelize.define("User", {
        id: {
            type: DataTypes.BIGINT,
            primaryKey: true,
            autoIncrement: true
        },
        email: {
            type: DataTypes.STRING,
            unique: true
        }
    }, {
            paranoid: true,
            underscored: true,
            freezeTableName: true,
            classMethods: {
                console : function() {
                    console.log("Hello : " + this);
                }
            }
        });
});
```

Synchronisation

- Lorsque toutes les tables sont définies dans ***Sequelize***, il faut le synchroniser pour qu'il puisse créer les tables dans la base de données

```
sequelize.sync();
```

Enregistrement d'une ligne

- Il faut utiliser la méthode ***create*** sur une table
- Il existe aussi ***findOrCreate***

```
User.create({
    "email": "benoit@briatte.com"
}).then(function(u) {
    if (u) {
        console.log("User created");
        User.console();
    }
}).catch(function(err) {
    throw err;
});
```

Récupération des éléments

- Utiliser la méthode *findAll*

```
User.findAll().then(function(users) {
    if (users) {
        console.log(users);
    }
}).catch(function(err) {
    throw err;
});
```

Récupération des éléments

- Utiliser la méthode ***find*** avec une clause ***where***

```
User.find({
    "where": {
        "id": 1
    }
}).then(function(user) {
    if (user) {
        console.log(user);
    } else {
        console.log("User not found");
    }
}).catch(function(err) {
    throw err;
});
```

Suppression d'un élément

- Utiliser la méthode ***destroy***

```
User.find({
    "where": {
        "id": 1
    }
}).then(function(user) {
    if (user) {
        user.destroy().then(function(user) {
            console.log("COMPLETED");
        }).catch(function(err) {
            throw err;
        });
    } else {
        console.log("User not found");
    }
}).catch(function(err) {
    throw err;
});
```

Modifier un élément

- Utiliser la méthode *updateAttributes*

```
User.findAll().then(function(users) {
    if (users) {
        for (var i = 0; i < users.length; i++) {
            users[i].updateAttributes({
                email: "esgi@yopmail.com"
            });
        }
    }
}).catch(function(err) {
    throw err;
});
```

Lien entre les tables

Il est possible de relier des tables entre elles :

- ***asOne*** - adds a foreign key to the target and singular association mixins to the source.
- ***belongsTo*** - add a foreign key and singular association mixins to the source.
- ***hasMany*** - adds a foreign key to target and plural association mixins to the source.
- ***belongsToMany*** - creates an N:M association with a join table and adds plural association mixins to the source. The junction table is created with sourceId and targetId.