САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО

Дисциплина: Бек-энд разработка

Отчет

Практическая работа 2: Знакомство с ORM Sequelize

Выполнил:

Косенко Филипп

Группа
К33392

Проверил:
Добряков Д. И.

Санкт-Петербург

2024 г.

Задачи

1) Продумать свою собственную модель пользователя
2) Реализовать набор из CRUD-методов для работы с пользователями средствами Express + Sequelize
3) Написать запрос для получения пользователя по id/email

Ход работы

Собственная модель пользователя:

id, name, age, course

Для создания модели используем код -

```
const { Sequelize } = require("sequelize");

//Создание новой таблице на базе бд SQLLite
const sequelize = new Sequelize({
    dialect: "sqlite",
    storage: "./data/users.sqlite"
});

// Описание модели пользователя
const user = sequelize.define("user", {
    id: {
        type: Sequelize.INTEGER,
        autoIncrement: true,
        primaryKey: true,
        allowNull: false
      },
      name: {
        type: Sequelize.STRING,
        allowNull: false
      },
      age: {
        type: Sequelize.INTEGER,
        allowNull: false
      },
      course:{
        type: Sequelize.INTEGER,
        allowNull: true
      }
});

module.exports = user;
```

Реализация CRUD-методов

Напишем маршруты, их я вписал в

основной файл приложения app.js

```
app.get("/", function(req, res){
    models.findAll({raw:true}).then(data =>{
        res.render("index.hbs", {
            users: data
        });
    }).catch(err => consolr.log(err));
});

app.get("/create", function(req, res){
    res.render("create.hbs");
});

app.post("/create", urlParser, function(req, res){
    if(!req) return res.statusCode(400);

    const username = req.body.name;
    const userage = req.body.age;
    const usercourse = req.body.course;

    models.create({ name: username, age: userage, course: usercourse}).then(()=>{
        res.redirect("/");
    }).catch(err => console.log(err));
});
```

```
//delete

app.post("/delete/:id", function(req,res){
    const userId = req.params.id;
    models.destroy({where: {id: userId}}).then(()=>{
        res.redirect("/");
    }).catch(err => console.log(err));
});

//update

app.get("/edit/:id", function(req, res){
    const userId = req.params.id;
    models.findAll({where: {id: userId}, raw: true}).then(data => {
        res.render("edit.hbs", {
            users: data[0]
        });
    }).catch(err => console.log(err));
});


app.post("/edit", urlParser, function(req, res){
    if(!req) return res.statusCode(400);

    const username = req.body.name;
    const userage = req.body.age;
    const usercourse = req.body.course;
    const userId = req.body.id;

    models.update({ name: username, age: userage, course: usercourse}, {where: {id: userId}}).then(()=>{
        res.redirect("/");
    }).catch(err => console.log(err));
});

//Ручка для тестов добавления нового пользока
//
// app.use("/testPost", function(req, res){
//     models.create({ name: "Fil", age: "20", course: "3"}).then(()=>{
//         res.redirect("/");
//     }).catch(err => console.log(err));
// });
```

ë

Вывод

В домашнем задании номер два познакомились с ORM Sequelize, удалось создать собственную модель пользователя и реализовать набор из CRUD-методов для работы с пользователями средствами Express + Sequelize.