Sequeliz :

1 – معرفی و نصب

Npm I @sequelize/core

بعد درایور دیتا بیس که میخوایم ازش استفاده کنیمو نصب میکنیم .

*# One of the following:*  
$ npm install --save pg pg-hstore *# Postgres*  
$ npm install --save mysql2  
$ npm install --save mariadb  
$ npm install --save sqlite3  
$ npm install --save tedious *# Microsoft SQL Server*  
$ npm install --save oracledb *# Oracle Database*

روش های اتصال به دیتابیس با sequelize :

1 –

import { Sequelize } from "@sequelize/core";

const sequelize = new Sequelize("task", "root", "hd6730mrm", {

    dialect: "mysql",

    host: "localhost",

});

2 –

const sequelize = new Sequelize("mysql://root:hd6730mrm@localhost/task");

3-

const sequelize = new Sequelize({

    dialect: "mysql",

    host: "localhosst",

    database: "task",

    username: "root",

    password: "hd6730mrm",

    logging: false

});

دیفالتش logging: console.log عه ینی هرچی قراره لاگ گرفته بشه بفرست به تابع console.log که میتونیم به جاش یه تابع دیگه بدیم .

به روش زیر میتونیم چک کنیم ببینیم که به دیتا بیس وصل شدیم یا نه :

try {

    await sequelize.authenticate();

    console.log("Connected");

} catch (error) {

    console.log(error.message);

}

db.js :

import { Sequelize } from "@sequelize/core";

const sequelize = new Sequelize({

    dialect: "mysql",

    host: "localhost",

    database: "task",

    username: "root",

    password: "hd6730mrm",

    logging: false,

});

try {

    await sequelize.authenticate();

    console.log("Connected Successfullt");

} catch (error) {

    console.log(error.message);

}

export default sequelize;

روش های ایجاد مدل :

1 –

import { DataTypes, Model } from "@sequelize/core";

import sequelize from "./db.js";

class User extends Model {}

User.init(

    {

        firstname: DataTypes.STRING,

        lastname: DataTypes.STRING(100),

        age: DataTypes.INTEGER

    },

    {

        sequelize: sequelize

    }

);

await User.sync()

تو این حالت خودش میاد برامون یه id پرایمری کی و یک createdAt و updatedAt برامون میسازه .

await User.sync() تو این حالت اگه دیتا بیس وجود داشته باشه هیچ کاریش نداره حتی تغییرش هم نمیده. فقط اگه وجود نداشته باشه میسازدش .

روش 2 :

const Task = sequelize.define(“Task”, {

    title: DataTypes.STRING(100),

    completed: DataTypes.BOOLEAN,

})

await Task.sync()

2-3- تعریف مدل ها :

const Task = sequelize.define(

    “Task”,

    {

        title: DataTypes.STRING,

        completed: DataTypes.INTEGER,

    },

    {

        timestamps: false,

        noPrimaryKey: false,

    }

);

الان دیگه اون id و createdAt و updatedAt رو که دیفالت میساخت دیگه نمیسازه .

نکته :

// drop table if exists, then create a new one

await User.sync({ force: true });

await Task.sync({ force: true });

نکته :

// alter table if any difference exists, but data will not deleted

await User.sync({ alter: true });

await Task.sync({ alter: true });

نکته :

// sync all tables

// await sequelize.sync();

// await sequelize.sync({alter: true});

// await sequelize.sync({force: true});

نکته :

// drop a table

await User.drop()

// drop a table

await sequelize.drop()

ایجاد مدل با تنظیمات ویژگی های ستون ها :

const Book = sequelize.define(

    "book",

    {

        pages: DataTypes.SMALLINT,

        title: {

            type: DataTypes.STRING(100),

            defaultValue: "",

        },

        description: {

            type: DataTypes.TEXT,

            defaultValue: "No Description",

        },

        published: {

            type: DataTypes.DATEONLY,

            defaultValue: DataTypes.NOW,

            allowNull: false,

        },

        category: {

            type: DataTypes.STRING,

            allowNull: false,

        },

    },

    {

        timestamps: false,

    }

);

await Book.sync({ alter: true });

ساخت یک رکورد :

await Book.create({ category: "Programming" });