sequelize / **express-example**

A proposal for the usage of Sequelize within an Express.JS application.

60 commits | **3** branches | **0** releases | **17** contributors

Branch: **master** | **New pull request** | **Create new file** | **Upload files** | **Find file** | **Clone or download**

sushantdhiman committed on Nov 25, 2017 Update (#79) ··· Latest commit `6328f7e` on Nov 25, 2017

bin	Fix Express Standard bin/www (#45)	2 years ago
config	Update (#79)	6 months ago
migrations	Update (#79)	6 months ago
models	Update (#79)	6 months ago
public/stylesheets	Update (#79)	6 months ago
routes	docs cleanup	2 years ago
test	sync before tests	2 years ago
views	Update (#79)	6 months ago
.gitignore	Ignore sqlite databases and the npm debug log	4 years ago
.sequelizerc	Update (#79)	6 months ago
README.md	Update (#79)	6 months ago
app.js	package: use pug (#77)	6 months ago
app.json	Heroku (#46)	2 years ago
package.json	Update (#79)	6 months ago

README.md

Express Example

This repository demonstrates the usage of Sequelize within an Express application. The implemented logic is a simple task tracking tool.

Deploy to Heroku

Starting App

Without Migrations

```
npm install
npm start
```

With Migrations

```
npm install
node_modules/.bin/sequelize db:migrate
npm start
```

This will start the application and create an sqlite database in your app dir. Just open http://localhost:3000.

Running Tests

We have added some Mocha based test. You can run them by `npm test`

Setup in Details

In order to understand how this application has been built, you can find the executed steps in the following snippet. You should be able to adjust those steps according to your needs. Please note that the view and the routes aren't described. You can find those files in the repo.

Express Setup

First we will create a bare Express App using `express-generator` Express Generator

```
# install express generator globally
npm install -g express-generator

# create the sample app
mkdir express-example
cd express-example
express -f

# install all node modules
npm install
```

Sequelize Setup

Now we will install all sequelize related modules.

```
# install ORM , CLI and SQLite dialect
npm install --save sequelize sequelize-cli sqlite3

# generate models
node_modules/.bin/sequelize init
node_modules/.bin/sequelize model:create --name User --attributes username:string
node_modules/.bin/sequelize model:create --name Task --attributes title:string
```

We are using `.sequelizerc` setup change config path for migrations. You can read more about this in migration docs

```
// .sequelizerc
const path = require('path');

module.exports = {
  'config': path.resolve('config', 'config.js')
}
```

You will now have a basic express application with some additional directories (config, models, migrations). Also you will find two migrations and models. One for the `User` and one for the `Task` .

In order to associate the models with each other, you need to change the models like this:

```
// task.js
// ...
  Task.associate = function(models) {
    // Using additional options like CASCADE etc for demonstration
    // Can also simply do Task.belongsTo(models.User);
    Task.belongsTo(models.User, {
      onDelete: "CASCADE",
      foreignKey: {
        allowNull: false
      }
    });
  }
// ...
```

```
// user.js
// ...
  User.associate = function(models) {
    User.hasMany(models.Task);
  }
// ...
```

This association will create an attribute `UserId` in `Task` model. We have to amend our `create-task` migration and add this column.

```
// xxxxxxx-create-task.js
// ...
  UserId: {
    type: Sequelize.INTEGER,
    onDelete: "CASCADE",
    allowNull: false,
    references: {
      model: 'Users',
      key: 'id'
    }
  }
// ...
```

If you want to use the automatic table creation that sequelize provides, you have to adjust the `bin/www` file to this:

```
#!/usr/bin/env node

var app = require('../app');
var debug = require('debug')('init:server');
var http = require('http');
var models = require("../models");

var port = normalizePort(process.env.PORT || '3000');
app.set('port', port);

var server = http.createServer(app);

// sync() will create all table if they doesn't exist in database
models.sequelize.sync().then(function () {
  server.listen(port);
  server.on('error', onError);
  server.on('listening', onListening);
});

function normalizePort(val) { /* ... */ }
function onError(error) { /* ... */ }
function onListening() { /* ... */ }
```

And finally you have to adjust the `config/config.js` to fit your environment. Once thats done, your database configuration is ready!