How to define relationship when models are defined in separate files? #4578

New issue

Closed **hezjing** opened this issue on Sep 26, 2015 · 19 comments

hezjing commented on Sep 26, 2015

Hi

I'm wondering what is the recommended way to define the relationship when the models are defined in separate files.

Assuming that I have a company model,

```
// models/Company.js
module.exports = function(sequelize, DataTypes) {
  return sequelize.define('Company', {
  }
};
```

and a project model,

```
// models/Project.js
module.exports = function(sequelize, DataTypes) {
  return sequelize.define('Project', {
  }
};
```

Then, I have an `index.js` that will read and import the above models,

```
// models/index.js
files.forEach(function(file) {
  sequelize.import(file);
});

// sequelize.models now contains Company and Project
// then define the relationship: a Company has many Projects
sequelize.models.Company.hasMany(sequelize.models.Project);
```

Is this the right way to define the relationships? I can imagine that there will be a lot of relationships defined in the same file when there are a lot of models.

mickhansen commented on Sep 27, 2015 Owner

I personally use a `relations.js` file, some people add an `associate` method to their models and call that after everything is imported.

mickhansen closed this on Sep 27, 2015

mjangir commented on May 31, 2016

Can you please show me the code of your relation.js file?

Assignees

No one assigned

Labels

None yet

Projects

None yet

Milestone

No milestone

Notifications

8 participants

mickhansen commented on May 31, 2016 Owner

```
module.exports = function(models) {
  // setup relations
};
```

mjangir commented on May 31, 2016

Hi **@mickhansen** thank you so much for this quick post. Actually I have really got stuck in the same situation. I have my two models in separate files:

company.model.js

```
module.exports = function(sequelize, DataTypes) {
  var Company = sequelize.define('Company', {
    id: {
      type: DataTypes.INTEGER(3),
      allowNull: false,
      primaryKey: true,
      autoIncrement: true,
      comment: "Primary and auto incremented key of the table"
    },
    companyName: {
      field: "company_name",
      type: DataTypes.STRING(255),
      allowNull: false,
      comment: "Company Name"
    }
},
{
    underscored: true,
    freezeTableName:true,
    tableName:'company',
    classMethods:{
      associate:function(models){
        Company.hasMany(models.Location);
      }
    },
  });
  return Company;
};
```

location.model.js

```
['use strict';

module.exports = function(sequelize, DataTypes) {
  var Location = sequelize.define('Location', {
    id: {
      type: DataTypes.INTEGER(5),
      allowNull: false,
      primaryKey: true,
      autoIncrement: true,
      comment: "Primary and auto incremented key of the table"
    },
    locationType: {
      field: "location_type",
      type: DataTypes.ENUM('HEAD_OFFICE','REGIONAL_OFFICE','FRANCHISE'),
      allowNull: false,
      comment: "Type of location"
    }
  },
  {
    timestamps: true,
    underscored: true,
    freezeTableName:true,
    tableName:'location',
    classMethods:{
      associate:function(models){
        Location.belongsTo(models.Company, { foreignKey:'company_id',
foreignKeyConstraint:true} );
      }
    }
  });

  return Location;
};
```

and I have the this final js file **index.js**

```
'use strict';

import path from 'path';
import config from '../config/environment';
import Sequelize from 'sequelize';

var sequelizeConnection = new Sequelize(config.sequelize.database, null, null,
config.sequelize.options);

var db = {
  Sequelize: Sequelize,
  sequelize: sequelizeConnection
};

// Insert models below
db.Company = db.sequelize.import('../api/company/company.model');
db.Location = db.sequelize.import('../api/location/location.model');
db.User = db.sequelize.import('../api/user/user.model');
db.Thing = db.sequelize.import('../api/thing/thing.model');

module.exports = db;
```

When I run the following code

```
db.sequelize.sync({force:true})
  .then(startServer)
  .catch(function(err) {
    console.log('Server failed to start due to error: %s', err);
  });
```

The tables are created absolutely fine but there is not foreign key collum on location table and nothing regarding foreign key constrain.

Can you please help me to solve it. I will really appreciate your solution. Thanks a ton.

mickhansen commented on May 31, 2016 Owner

I don't see you calling the .associate method anywhere. Keep in mind it's not a magic built in Sequelize method, it's just a pattern used by some examples.

mjangir commented on May 31, 2016

So what should I do to work it correctly. Do you have any proposed solution. I'm really new to sequelizeJS.

mickhansen commented on May 31, 2016 Owner

Either call the method or setup associations in a file like i suggested. Try SOMETHING, it doesn't sound like you've tried anything. The code that setups associations needs to be actually called, verify that happens :)

mjangir commented on May 31, 2016

I tried to call it using the following:

```
var db = {
  Sequelize: Sequelize,
  sequelize: sequelizeConnection
};

// Insert models below
db.Company = db.sequelize.import('../api/company/company.model');
db.Location = db.sequelize.import('../api/location/location.model');
db.User = db.sequelize.import('../api/user/user.model');
db.Thing = db.sequelize.import('../api/thing/thing.model');

Object.keys(db).forEach(function(modelName) {
  if (db[modelName].options.hasOwnProperty('associate')) {
    db[modelName].options.associate(db)
  }
})

module.exports = db;
```

janmeier commented on May 31, 2016 Owner

You are putting `associate` in classMethods, but looking for it in `options` - so the function is not called

mjangir commented on May 31, 2016

I also tried it `db[modelName].classMethods.associate(db)` man but nothing is working :(

mickhansen commented on May 31, 2016 Owner

classMethods are adding methods to the class, have you tried inspecting the object? Or just calling db[modelName].associate?

mickhansen commented on May 31, 2016 Owner

@mjangir Please do basic debugging on your own before taking up other peoples time, it's the polite thing to do :)

Basic debugging includes checking if and why a method is or is not called, this is basic javascript, nothing Sequelize specific.

👍 1 👎 13

mjangir commented on May 31, 2016

Wow, simply two lines worked like a charm for me.

```
db.Company.hasMany(db.Location);
db.Location.belongsTo(db.Company);
```

But I think its not a good practice. Please suggest me to manage all them in separate files whenever you get time.

By the way many many thanks to give your important time.

mickhansen commented on May 31, 2016 Owner

@mjangir I personally use a single `relations.js` file as mentioned.

👎 6

joinsunil commented on Aug 23

@mjangir Can you please post a code snippet , how you are managing all relationship in `relationship.js` file.

👍 5

Infer-On commented on Oct 15 • edited

@joinsunil I am doing something like this

index.js:

```
const CLASSMETHODS = 'classMethods';
const ASSOCIATE = 'associate';
var sequelize = new Sequelize(cfg.db.database, cfg.db.user, cfg.db.password, cfg.db.options

fs.readdirSync(__dirname).filter(function (file) {
    return (file.indexOf('.') !== 0) && (file !== 'index.js');
}).forEach(function (file) {
    var model = sequelize['import'](path.join(__dirname, file));
    db[model.name] = model;});
Object.keys(db).forEach(function (modelName) {
    if (CLASSMETHODS in db[modelName].options) {
     if (ASSOCIATE in db[modelName].options[CLASSMETHODS]) {
      db[modelName].options.classMethods.associate(db);
    }}});
```

Character.js:

```
module.exports = (sequelize, DataTypes) => {
    const Character = sequelize.define('Character', {
      Id: { type: DataTypes.INTEGER, primaryKey: true, autoincrement: true },
      FirstName: DataTypes.STRING,
      LastName: DataTypes.STRING,
      DoB: DataTypes.DATE,
      createdAt: DataTypes.DATE,
      updatedAt: DataTypes.DATE
    }, {
      classMethods: {
      associate: (models) => {
        Character.belongsTo(models.CharacterDetail, {
          foreignKey: 'fk_detail_id',
          as: 'Detail'})}}});
      return Character;
}
```

Have you discovered a better approach?

👍 1

joinsunil commented on Oct 18

@Infer-On You are correct..

delebash commented on Oct 21

This worked for me

model animal.js

```
/* jshint indent: 2 */
module.exports = (sequelize, DataTypes) => {
  const animal = sequelize.define('animal', {
      id: {
        type: DataTypes.INTEGER(11),
        allowNull: false,
        primaryKey: true,
        autoIncrement: true
      },
      non_shelter: {
        type: DataTypes.INTEGER(4),
        allowNull: true
      }
    }
  );

  animal.associate = function (models) {
    animal.hasMany(models.lookup_animal_type);
  };

  return animal
};
```

index.js

```
"use strict";

var fs        = require("fs");
var path      = require("path");
var Sequelize = require("sequelize");
var env       = process.env.NODE_ENV || "development";
const Op = Sequelize.Op;
var config    = require(path.join(__dirname, '..', 'config', 'config.json'))[env];
if (process.env.DATABASE_URL) {
  var sequelize = new Sequelize(process.env.DATABASE_URL,config);
} else {
  var sequelize = new Sequelize(config.database, config.username, config.password,
config);
}
var db = {};

fs
  .readdirSync(__dirname)
  .filter(function(file) {
    return (file.indexOf(".") !== 0) && (file !== "index.js");
  })
  .forEach(function(file) {
    var model = sequelize.import(path.join(__dirname, file));
    db[model.name] = model;
  });

Object.keys(db).forEach(function(modelName) {
  if ("associate" in db[modelName]) {
    db[modelName].associate(db);
  }
});

db.sequelize = sequelize;
db.Sequelize = Sequelize;
db.sequelize.sync();
modul
e.exports = db;
```

👍 1

shtse8 commented on Oct 22

importing single model file in sequelize is hell. Thanks for the above approaches.