Egg 中使用 Sequelize ORM 框架操作 Mysql-进行关联查询

主讲教师：（大地）
合作网站：www.itying.com （IT 营）
我的专栏：https://www.itying.com/category-79-b0.html

一、Sequelize 简介

前面的章节中，我们介绍了如何在框架中通过 egg-mysql 插件来访问数据库。而在一些较为复杂的应用中，我们可能会需要一个 ORM 框架来帮助我们管理数据层的代码。而在 Node.js 社区中，sequelize 是一个广泛使用的 ORM 框架，它支持 MySQL、SQLite 和 MSSQL 、PostgreSQL 等多个数据源。下面我们主要给大家讲讲 sequelize 结合 MySQL 的使用。

相关文档：

https://eggjs.org/zh-cn/tutorials/sequelize.html

https://sequelize.org/

二、Sequelize 操作 Mysql 数据库

1、安装 egg-sequelize 以及 mysql2

```
npm install --save egg-sequelize mysql2
```

2、在 config/plugin.js 中引入 egg-sequelize 插件

```
exports.sequelize = {
  enable: true,
  package: 'egg-sequelize',
};
```

3、 在 config/config.default.js 中编写 sequelize 配置

```
config.sequelize = {
      dialect: 'mysql',
      host: '127.0.0.1',
      port: 3306,
      database: 'test',
      username:"root",
      password:"123456"
};
```

三、Sequelize 操作 Mysql 实现增删改查

1、在 app/model/ 目录下编写数据库 Model,以用户表 user 为例

```
'use strict';

module.exports = app => {
  const { STRING, INTEGER, DATE } = app.Sequelize;

  const User = app.model.define('user', {
```

```
        id: { type: INTEGER, primaryKey: true, autoIncrement: true },
        name: STRING(30),
        age: INTEGER,
        created_at: DATE,
        updated_at: DATE,
    });

    return User;
};
```

或者

```
'use strict';

module.exports = app => {
    const { STRING, INTEGER, DATE } = app.Sequelize;

    const User = app.model.define('user', {
        id: { type: INTEGER, primaryKey: true, autoIncrement: true },
        name: STRING(30),
        age: INTEGER,
        created_at: DATE,
        updated_at: DATE,
    },{
            timestamps: false,    //自动增加创建时间
            tableName: 'user_info'     //设置表名称
    });
    return User;
};
```

2、定义 controller 实现数据库的增删改查

```
async index() {
        const ctx = this.ctx;
        ctx.body = await ctx.model.User.findAll({limit: 10, offset: 0,order:[["id","desc"]]});

        //指定返回的字段
        //ctx.body = await ctx.model.User.findAll({attributes: ['id', 'name'],limit: 10,order:[["id","desc"]]});

}
```

```
  async findOne() {
```

```
        const ctx = this.ctx;
        var result = await ctx.model.User.findByPk(106);
        ctx.body=result || '';
    }
```

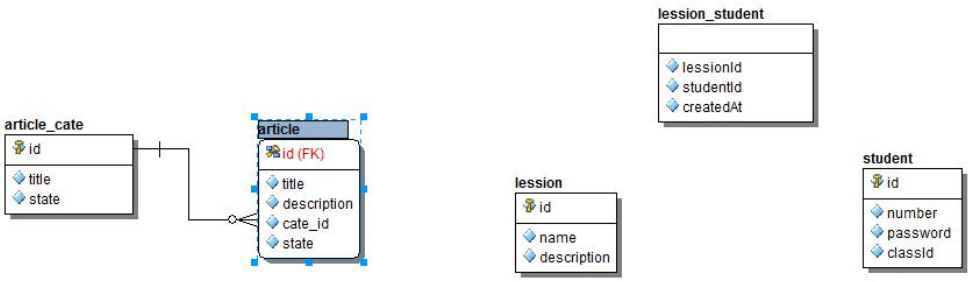
```
async create() {
        const ctx = this.ctx;
        const user = await ctx.model.User.create({ name:"张三", age:20 });
        ctx.status = 201;
        ctx.body = user;
}
```

```
async update() {
        const ctx = this.ctx;
        const id = 106;
        const user = await ctx.model.User.findByPk(id);
        if (!user) {
            ctx.status = 404;
            return;
        }
        await user.update({ name:"李四", age:43 });
        ctx.body = user;
    }
```

```
async destroy() {
        const ctx = this.ctx;
        const id = 213;
        const user = await ctx.model.User.findByPk(id);
        if (!user) {
            ctx.status = 404;
            return;
        }

        await user.destroy();
        ctx.status = 200;
        ctx.body="删除成功";
}
```

四、Sequelize 操作多表关联查询



一、1 对 1　hasOne 或者 belongsTo

```
  ArticleCate.associate = function (){
    // 1 对 1
    app.model.ArticleCate.hasOne(app.model.Article, {foreignKey: 'cateId'});
  }
```

查询语句

```
    const { ctx } = this;
    let result = await ctx.model.ArticleCate.findAll({
        include: {
          model: ctx.model.Article
        }
    });
```

```
  Article.associate = function (){
    // 1 对 1
    app.model.Article.belongsTo(app.model.ArticleCate, {foreignKey: 'cateId'});
  }
```

查询语句

```
    const { ctx } = this;
    let result = await ctx.model.Article.findAll({
        include: {
          model: ctx.model.ArticleCate
        }
    });
```

二、1 对多 hasMany

```
  ArticleCate.associate = function (){
    // 1 对多
    app.model.ArticleCate.hasMany(app.model.Article, {foreignKey: 'cateId'});
  }
```

查询语句

```
    const { ctx } = this;
    let result = await ctx.model.ArticleCate.findAll({
        include: {
          model: ctx.model.Article
        }
    });
```

三、多对多 belongsToMany

```
 Lesson.associate = function(){
        //一个课程可以被多个学生选修
        app.model.Lesson.belongsToMany(app.model.Student, {
            through: app.model.LessonStudent,
            foreignKey: 'lessonId',
            otherKey: 'studentId'
        });
}
```

```
Student.associate = function (){
        //一个学生可以选修多门课程
        app.model.Student.belongsToMany(app.model.Lesson, {
            through: app.model.LessonStudent,
            foreignKey: 'studentId',     //注意写法
            otherKey: 'lessonId'
        });
}
```

查询语句：

```
    const { ctx } = this;
    let result = await ctx.model.Lesson.findAll({
      include: {
        model:    ctx.model.Student
      }
    });


或者


  const { ctx } = this;
    let result = await ctx.model.Student.findAll({
      include: {
        model:    ctx.model.Lesson
      }
    });
```

五、Sequelize 常见的数据类型

```
Sequelize.STRING                      // VARCHAR(255)
Sequelize.STRING(1234)                // VARCHAR(1234)
Sequelize.STRING.BINARY               // VARCHAR BINARY
Sequelize.TEXT                        // TEXT
Sequelize.TEXT('tiny')                // TINYTEXT

Sequelize.INTEGER                     // INTEGER
Sequelize.BIGINT                      // BIGINT
Sequelize.BIGINT(11)                  // BIGINT(11)

Sequelize.FLOAT                       // FLOAT
Sequelize.FLOAT(11)                   // FLOAT(11)
```

```
Sequelize.FLOAT(11, 12)                 // FLOAT(11,12)

Sequelize.DOUBLE                        // DOUBLE
Sequelize.DOUBLE(11)                    // DOUBLE(11)
Sequelize.DOUBLE(11, 12)                // DOUBLE(11,12)

Sequelize.DECIMAL                       // DECIMAL
Sequelize.DECIMAL(10, 2)                // DECIMAL(10,2)

Sequelize.DATE                          // DATETIME 针对 mysql / sqlite, TIMESTAMP
WITH TIME ZONE 针对 postgres
Sequelize.DATE(6)                       // DATETIME(6) 针对 mysql 5.6.4+. 小数秒支持多
达 6 位精度
Sequelize.DATEONLY                      // DATE 不带时间.
Sequelize.BOOLEAN                       // TINYINT(1)
```