安装mysql模块：npm i mysql -S

其本操作应用：

连接数据库必要信息:

引入：const mysql = require('mysql')

连接的必备信息：const conn = mysql.createConnection({ host: 'localhost', user: 'kaikeba_admin',

password: 'admin', database: 'kkb' })

连接：conn.connect();

查询操作：conn.query(sql, value, callback)

参数传递：

conn.query('select * from user where id=?, 1, cb)//单个参数

conn.query('select * from user where name=? and age>?, ['tom',20], cb)//多个参数

项目应用：

1、将数据库的连接提取出来在需要的地方调用

在项目根目录创建一个新的目录并创建一个新的JS文件，作为连接数据的配置文件

```
// 连接数据库
const mysql = require('mysql')
const cfg = {
    host: 'localhost',
    user: 'kaikeba_admin',
    password: 'admin',
    database: 'kkb'
}
module.exports = {
    query: function (sql, value) {
        return new Promise((resolve, reject) => {
            // resolve函数在异步操作成功时执行
            // reject函数在异步操作失败时执行
            const conn = mysql.createConnection(cfg);
            conn.connect();//此步骤可省略
            conn.query(sql, value, (err, results) => {
                if (err) reject(err);
                else resolve(results);
            })
            conn.end();
        })
    }
}
```

2、在需要使用的地方引入：

const {query} = require('../models/db');

基本用法：采用**Promise**对象模式解决回调问题，要使用的关键字**async，await**

```
router.get('/', async (req, res, next) => {
    try {
        const sql = 'SELECT * FROM open_course';// LIMIT offset,size
        const results = await query(sql);
        res.render('open-courses', {
            title: '公开课',
            openCourses: results
        });
    } catch (err) {
        next(err);
    }
})
```

在路由中的其本用法

```
module.exports.initLocals = async function (req, res, next) {
        const sql = 'SELECT * FROM kkb.vip_course';
        try {
            const courses = await query(sql);//query必须返回Promise才能使用await
            res.locals.courses = courses;
            next();// 进入后续中间件
        } catch (err) {
            next(err)
        }
    }
}
```

在中间件里的使用

数据库的连接池：

1、引入**MySQL**：**const mysql = require('mysql')**

2、创建连接数据库必要信息

const cfg = {

host: 'localhost',

user: 'kaikeba_admin',

password: 'admin',

database: 'kkb'

}

3、创建连接池：**const pool = mysql.createPool(cfg);**

4、导出数据库连接模块

```
//导出查询模块
module.exports = {
    query: function (sql, value) {
      return new Promise((resolve, reject) => {
          //第一种查询方式
          pool.getConnection((err, conn) => {
              conn.query(sql, value, (err, results) => {
              // resolve函数在异步操作成功时执行    reject函数在异步操作失败时执行
                  if (err) reject(err);
                  else resolve(results);
              });
              conn.release();
          })
      })
    }
}
```

```
//导出查询模块                    第二种方式为实际应用方式
module.exports = {
    query: function (sql, value) {
      return new Promise((resolve, reject) => {
          //第二种查询方式
          pool.query(sql, value, (err, results) => {
              if (err) reject(err);
              else resolve(results);
          })
      })
    }
}
```

5、使用：同上面使用方法一样

数据库整合之Sequelize

1、概述：基于**Promise**的**ORM**（**Object Relation Mapping**）**,** 支持多种数据库，事务，关联

2、安装：**npm i sequelize mysql2 -S**

3、在项目根目录下创建一个**models**目录，然后在**models**目录下创建一个**mode.js**文件 该文件用来集成**Sequelize**

4、**mode.js**集成**Sequelize**的方法：

1、导入文件操作模块和路径模块

```
const fs = require('fs');//模型导入需要
```

```
const path = require('path'); //路径
```

2、创建sequelize对象

```
const Sequelize = require('sequelize');
```

3、创建sequelize的连接

```
const sequelize = new Sequelize('sshcrm01', 'root', '123456', {
    host: 'localhost',      //       数据库名      用户名      密码
    dialect: 'mysql', // 方言设置
    pool: {max: 5, acquire: 30000, idle: 10000}, // 连接池
    define: {    //禁止seq自动添加createdAt，updatedAt
        timestamps: false,
        freezeTableName: true
    }
});
```

4、动态地将外面定义好的模型导入（操作数据库的模型单独在**models**目录下创建）

```
const db = {Sequelize, sequelize};
// 动态导入模型
// 读取当前目录中所有文件名
fs.readdirSync(__dirname)
    //从当前目录下，过滤出不是模型的文件
    .filter(file => (file !== 'sequelizess.js' && file !== 'mysql_ljc.js'))
    .forEach(file => { // 从文件中导入模型
        const model = sequelize.import(path.join(__dirname, file));
        db[model.name] = model;
    });
module.exports = db;
```

5、创建模型（在**models**目录下创建）

前题条件模型在创建前数据库中一定是有一个要操作的表是定义好的

依据这个已经定义好的表来创建模型

models目录下创建一个新的**xxx.js**文件为操作数据的模型。模型的定义：

```
module.exports = (sequelize, Types) => {
  const OpenCourse = sequelize.define('OpenCourse', {
      name: Types.STRING(50),
      description: Types.STRING(100),
      time: Types.DATE,
      count: Types.INTEGER,
  }, {
      tableName: 'open_course', // 指定要操作的表名（数据库中已有的表）自己命名表名
      timestamps: false, // 禁止seq自动添加createdAt，updatedAt
  })
  OpenCourse.sync();
  return OpenCourse;
}
```

具体说明：

const OpenCourse 其中**OpenCourse**是一个自定义的对象名，该名称将来会在操作数据库的地方使用

define('OpenCourse'{} 其中**OpenCourse**是自定义的表名映射别名

name: Types.STRING(50)..... 是对应已有表中已有的字段名称及数据类型

OpenCourse.sync()； 这个方法可以有以下参数及其它方法,默认可以没有方法及参数

参数：**OpenCourse.sync({force: true})** **{force: true}**：同步数据库,**FORCE**如果为**true**则会删除已存在同名表并重建表

方法：**sync({force: true}).then(() => {**

```
// 插入若干数据
return OpenCourse.create({
  name: 'xiaoming',
  description: '一个人'
  .....................
})
})
```

在插入数据后还可以查询插入的数据

完整写法：

```
OpenCourse.sync({force: true}).then(() => {
    // 插入若干测试数据
    return OpenCourse.create({
      firstName: 'Tom',
      lastName: 'Cruise'
    })
}).then(() => {
```

```
// 查询前面插入数据
OpenCourse.findAll().then( opencourse=> {
 console.log(opencourse);
 })
})
```

6、使用集成好**Sequelize**的**mode.js**来操作数据库

1、在路由的**js**文件中引用**mode.js**

```
const {OpenCourse} = require('../models/mode');
```

2、使用**Sequelize**中定义好的对象方法

```
const results = await OpenCourse.findAll();
```

完整操作：

```
router.get('/aaa', async (req, res, next)=> {
    //测试数据库
    try {
        const results = await OpenCourse.findAll();
            console.log(results);
            res.render('index', {
                title: '你好,Express',
                showVideo: false,
                resultes:results
            });
    }catch (err) {
        next(err)
    }
});
```

返回结果集合

查询所有数据

显示到hbs渲染页面