1. 单表查询：

-- 查询不是计算机系的学生信息

-- select \* from xsb where szx <> '计算机系'

-- select \* from xsb where szx != '计算机系'

-- 查询除co1、c02课程的分数信息

-- select \* from cjb where kch <> 'c01' and kch<>'c02'

-- select \* from cjb where kch not in ('c01','c02')

-- 查询年龄大于18的学生信息，并按所在系降序排序，如果是同一个系则按年龄升序排序

-- select \* from xsb where age>18 order by szx desc,age desc

-- **null 与 ''**

-- 查询成绩为null的分数信息

-- select \* from cjb where cj is not null

-- select \* from cjb where kch=''

-- **limit m,n n表示每次要显示的条数 m表示游标位置**

-- 第一页内容：

-- select \* from xsb limit 0,2

-- 第二页内容：

-- select \* from xsb limit 2,2

-- 第三页内容：

-- **注意：实现分页时: m的值=(第几页-1)\*每页要显示的条数**

-- select \* from xsb limit 4,2

-- **随机查询3个学生的信息**

-- **order by rand() : 随机排序**

select \* from xsb order by rand() limit 3

-- 查询成绩最低的3个分数信息

-- select \* from cjb order by cj asc limit 3

-- 查询总的学生人数：

-- select count(\*) from xsb

-- **分组查询：group by**

-- **查询男生、女生的人数**：

-- select count(\*),sex from xsb group by sex

-- 查询每门课程的选修人数：

-- select count(\*) as num,kch from cjb group by kch

-- **查询每门课程的平均分数**：

-- select avg(cj),kch from cjb group by kch

-- **查询每门课程的最高分**：

-- select max(cj),kch from cjb group by kch

-- **查询分数不为null的每门课程选修人数，并按人数降序排序 ，只显示前二个**

select count(xh) as num,kch from cjb where cj is not null group by kch order by num desc limit 2

-- **having:对查询结果进行二次筛选**

-- **where:对表数据进行筛选**

-- 查询除表演系之外每个系的人数，只显示人数在3人及以上的系，并按人数降序排序

-- select count(szx) as num ,szx from xsb where szx<>'表演系' group by szx having num >= 3 order by num desc

-- 查询每门课程的分数在80分及以上的总分数，只显示总分数在200分以上的课程，并按总分数升序排序

-- select sum(cj) as score,kch from cjb where cj >= 80 group by kch having score > 200 order by score asc limit 3

-- **联表查询**：

**--**  **全连接： inner join 只有当满足on条件时才会出现在查询结果中**

**-- 左连接： left join left join 左边的表不管满足不满足on条件都会出现在查询结果中，如果左边的表不满足on条件则left join右表内容全部以null来显示，如果左边的表满足on条件则left join右表显示具体内容，left join 右表只有满足on条件时才会出现在查询结果中。**

-- 查询每个女生的分数信息：

-- select \* from xsb inner join cjb on xsb.xh = cjb.xh where sex='女' order by cj desc limit 3

-- select \* from xsb as x left join cjb as c on x.xh = c.xh where sex = '女'

-- 查询属每个学生的学生信息并显示他们的分数

-- select \* from xsb left join cjb on xsb.xh=cjb.xh

-- 查询每个学生的分数信息

-- select \* from cjb left join xsb on xsb.xh=cjb.xh

-- 查询每门课程的平均分、课程名 ，并按平均分降序排序

-- select avg(cj),c.kch from cjb as c inner join kcb as k on c.kch=k.kch group by c.kch

-- select avg(cj),c.kch from cjb as c left join kcb as k on c.kch = k.kch group by c.kch

-- **查询计算机系的每个学生所有的分数、课程名、姓名、年龄，并按成绩降序排序只显示前四个**

select xm,age,cj,kcm from xsb as x left join cjb as c on x.xh=c.xh left join kcb as k on c.kch=k.kch where szx='计算机系' order by cj desc limit 4

1. Express操作mysql:
2. 在项目根目录安装mysql包： npm i mysql
3. Express操作mysql:

1)创建连接：

const mysql = require('mysql');

//创建连接

let sqlObj = mysql.createConnection({

    host: '主机名',

    user: '用户名',

    password: '密码',

    port: 端口号,

    database: '数据库名'

});

//开始连接

sqlObj.connect();

2)执行sql语句：

//注意：query()方法是异步方法

sqlObj.query('sql语句'[,参数],(err,result)=>{});

1. Express操作mysql实例：

const express = require('express');

const mysql = require('mysql');

const app = express();

app.listen(8080, () => {

    console.log('server port at 8080');

});

// console.log(mysql);

//创建连接：

let sqlObj = mysql.createConnection({

    host: 'localhost',

    user: 'root',

    password: '123456',

    port: '3306',

    database: 'demos'

});

//开始连接

sqlObj.connect();

//执行sql语句：

//注意：query()方法是异步方法

// sqlObj.query('sql语句'[,参数],(err,result)=>{});

//添加数据：

app.get('/add', (req, res) => {

    //接参

    let { xh, xm, sex, age, szx } = req.query;

    //添加sql：

    let sql = `insert into xsb(xh,xm,sex,age,szx)values('${xh}','${xm}','${sex}','${age}','${szx}')`;

    //执行添加sql语句：

    sqlObj.query(sql, (err, result) => {

        console.log(err, result);

    })

    res.send('添加');

});

//查询数据：

app.get('/find', (req, res) => {

    //接参：

    let curusr = req.query.usr;

    let cond = '';

    if (curusr != '') {

        cond = `where xm='${curusr}'`;

    }

    //查询所有学生信息

    let sql = `select \*  from xsb ${cond}`;

    sqlObj.query(sql, (err, result) => {

        console.log(err, result);

        let str = '';

        result.forEach(item => {

            str += `<li>${item.xm} | ${item.sex}  | ${item.age}</li>`;

        });

        res.send(`<ul>${str}</ul>`);

    });

    //  console.log('testing....', 999);

});

1. 部署小U商城Api项目：
2. 使用express生成器创建项目：

d:\>express --view=ejs xyouShopApi

1. 切换目录至xyouShopApi并安装依赖包：

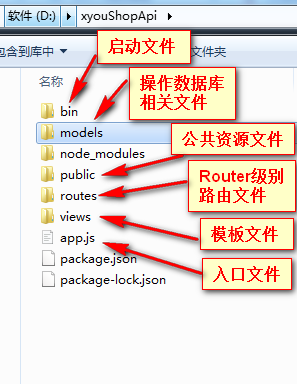
d:\>cd xyouShopApi

d:\xyouShopApi>npm install

1. 使用nodemon启动项目：

D:\xyouShopApi> nodemon ./bin/www

1. 在项目目录xyouShopApi中创建”models”目录以存放操作mysql数据库类的模块文件， 目录结构如下：



1. 封装express操作mysql类：
2. 在xyouShopApi\models目录中新建config.js文件，新增mysql数据库配置内容信息，如下：

exports.configure = {

    "host": 'localhost',

    "user": 'root',

    "password": '123456',

    "port": '3306',

    "database": 'xiaou'

}

1. 在xyouShopApi\models目录中新建dbMysql.js文件，存放express操作mysql的类，如下：

const mysql = require('mysql');

const { configure } = require('./config.js');

// console.log(conf);

//操作mysql数据库

class dbMysql {

    constructor() {

        //创建连接：

        this.sqlObj = mysql.createConnection(configure);

        //开始连接

        this.sqlObj.connect();

    }

    //执行sql语句

    exec(sql) {

        return new Promise((resolve, reject) => {

            this.sqlObj.query(sql, (err, result) => {

                // console.log(err, result);

                if (err) { //失败

                    resolve([err, '']);

                } else { //成功

                    resolve(['', result]);

                }

            });

        });

    }

    async querys(sqls) {

        let data = await this.exec(sqls);

        //  console.log(data, 999);

        return data;

    }

}

// let obj = new dbMysql();

// //obj.exec('select \* from xsb');

// //注意：调用这个obj.querys()方法本身是同步的，该方法会返回一个Promise对象

// // let d = obj.querys('select \* from xsb');

// // console.log(d, 888);

// async function test() {

//     let d = await obj.querys('select \*  from xsb');

//     console.log(d, 666);

// }

// test();

module.exports = new dbMysql();

**总结**：

**(1)、因为mysql.query(‘sql语句’,(err,result)=>{})这个方法为异步方法，想把该方法执行的sql语句结果传递出来则要使用async...await与promise技术；**

**(2)、直接使用对象调用上面类中封装的querys()方法得到的是Promise对象而不是sql语句结果，在调用该querys()方法时则需要使用async...await技术。**

1. 设计小U商城前台用户注册接口：
2. xyouShopApi\routes\users.js文件中新增下面的路由：

var express = require('express');

const mysqlObj = require('../models/dbMysql.js');

var router = express.Router();

//用户注册：

router.post('/register', async(req, res) => {

    //接参(用户提交过来帐号、密码)

    let { uname = '', upwd = '' } = req.body;

    if (uname == '' || upwd == '') {

        res.send({ "code": "500", "msg": "帐号、密码不能为空" });

        return;

    }

    //执行sql语句：

    let sql = `select count(\*) as n from member where username='${uname}'`;

    let [err, data] = await mysqlObj.querys(sql);

    //判断帐号是否注册过

    if (data[0].n !== 0) { //帐号存在

        res.send({ "code": "500", "msg": "帐号已注册过" });

        return;

    }

    //将新注册的用户写放数据库

    sql = `insert into member(username,password,uid,createdate)values('${uname}','${upwd}','7788','1587027991200')`;

    [err, data] = await mysqlObj.querys(sql);

    if (err == '') { //添加成功

        res.send({ "code": "200", "msg": "添加成功" });

    } else { //添加失败

        res.send({ "code": "500", "msg": "添加失败" });

    }

});

module.exports = router;

**注意**：

要将router.post('/register', async(req, res) => { })中的回调方法加async，否则无法使用await获取封装的操作mysql类中的querys()方法执行sql语句结果。