# 路由

根据不同的请求方法，响应请求路由设置具体内容

//express提供了一个方式，专门用来包装路由

var express =require('express');

//1.创建一个路由容器

var router = express.Router();

//2.把路由挂载到router路由容器中

router.get('/students',function(req,res){

}

router.get('/students/new',function(req,res){

}

router.post('/students/new',function(req,res){

}

router.get('/students/update',function(req,res){

}

router.post('/students/update',function(req,res){

}

router.get('/students/delete',function(req,res){

}

//3.导出成员

module.exports = router;

//在app.js中将路由挂载到app上

app.use(router);

# 增删改查

## 处理模板

## 配置静态资源

## 配置模板引擎

## 设计路由

##路由设计

| 请求方法 |      请求路径     |get参数|          post参数            |        作用           |

|---------|------------------|------|-------------------------------|----------------------|

|get      | /students         |     |                               |  渲染首页            |

|get      | /students/new    |      |                               |  渲染添加页面         |

|post     | /students/new    |      |name,sex,age,address,hubbies   |  处理添加的信息提交    |

|get      | /students/update |id    |                               |  渲染修改页面         |

|post     | /students/update |      |id,name,sex,age,address,hubbies|  处理修改页面提交信息  |

|get      | /students/delete |id    |                               |  处理删除             |

## 提取路由模块

//express提供了一个方式，专门用来包装路由

var express =require('express');

//1.创建一个路由容器

var router = express.Router();

//2.把路由挂载到router路由容器中

//首页

router.get('/students',function(req,res){

}

//跳转添加页面

router.get('/students/new',function(req,res){

}

//做添加操作

router.post('/students/new',function(req,res){

}

//跳转添加页面

router.get('/students/update',function(req,res){

}

//做添加操作

router.post('/students/update',function(req,res){

}

//删除

router.get('/students/delete',function(req,res){

}

//导出router

module.exports = router;

## 封装一个处理文件数据的students.js ,可以先写好结构

查询所有数据All

查询ID

添加Add

更新update

删除delete

\*students.js封装数据操作模块

\*职责：操作文件中的数据，只处理数据，不关心业务

var fs = require('fs');

var paths = './db.json';

/\*\*

 \* 获取所有学生列表

 \* 回调函数   获取异步操作的结果

 \* //相当于

 \* callback=function(err,data){}

 \* callback（a,b）有两个返回值

 \* 第一个参数(a)是错误值：当错误时返回err错误对象,正确返回null

 \* 第二个参数(b)是结果：当错误时返回undefined,正确返回数组

 \*

 \*/

exports.All = function(callback){

    //读文件

    fs.readFile(paths,'utf8',function(err,data){

        if(err){

            return callback(err,undefined);

        }

        callback(null,JSON.parse(data).students);

    })

}

/\*\*

 \*

 \* 根据id获取参数

 \*/

exports.byid = function(id,callback){

    fs.readFile(paths,'utf8',function(err,data){

        if(err){

            return callback(err)

        }

        //将JSON对象转换为字符串

        var students  = JSON.parse(data).students

        //ES6中的一个方法find，需要接收一个函数作为参数

        //当某个遍历项(students)符合item.id = student.id条件时，find会终止遍历，同时返回遍历项

        //数据中的id等于用户传入的id,返回这条id数据的所有信息

        var stu =students.find(function(item){

            return item.id ===parseInt(id)

        })

        callback(null,stu)

    })

}

 /\*\*

  \* 添加保存学生信息

  \* student为对象输入的数据

  \* add({

  \*     name:'ss',

  \*     age:12

  \* },function(err){})

  \*/

 exports.add = function(student,callback){

    fs.readFile(paths,'utf8',function(err,data){

        //错误

        if(err){

            return callback(err)

        }

        //成功，将数据保存，并转换

        //json数据 json对象转化为字符串

        var studentsd = JSON.parse(data).students;

        //id  用户输入的id等于数据中的最大数+1

        student.id = studentsd[studentsd.length -1].id + 1;

        //插入数据 ，student为对象 用户输入的数据 例如： { id: 3, name: '小米3',sex: 1}

        studentsd.push(student);

        //将字符串转换为JSON对象

        var  retstring = JSON.stringify({ students : studentsd });

        //写入文件

        fs.writeFile(paths,retstring,function(err){

            //错误

            if(err){

                return callback(err);

            }

            //成功

            callback(null);

        })

    })

}

/\*\*

  \* 更新保存学生信息

  \*/

 exports.update = function(student,callback){

    fs.readFile(paths,'utf8',function(err,data){

        //失败

        if(err){

            return callback(err)

        }

        //成功

        var students = JSON.parse(data).students;

        //将id转为int类型

        student.id = parseInt(student.id)

        //当某个遍历项符合item.id===student.id时会终止遍历循环，同时返回结果

        var stu = students.find(function(item){

            return item.id === student.id

        })

         //循环修改id相对的数据 点击要修改项 等于用户重新输入的项

        for(var key in student){

            stu[key] = student[key]

        }

        //将json转为字符串

        var stustring = JSON.stringify({ students :students });

        //写入文件

         fs.writeFile(paths,stustring,function(err){

            if(err){

                return callback(err)

            }

            callback(null);

         })

    })

}

/\*\*

  \* 删除保存学生信息

  \*/

 exports.delete = function(id,callback){

    fs.readFile(paths,'utf8',function(err,data){

        //失败

        if(err){

            return callback(err)

        }

        //成功

        var students = JSON.parse(data).students;

        //根据条件查找下标

        var stu = students.findIndex(function(item){

            return item.id === parseInt(id)

        })

        //根据下标从数组中删除

        //splice(开始位，删除的个数，添加的数据)

        students.splice(stu,1)

         //将json转为字符串

         var stustring = JSON.stringify({ students :students });

         //写入文件

          fs.writeFile(paths,stustring,function(err){

             if(err){

                 return callback(err)

             }

             callback(null);

          })

    })

}

find,findIndex为ES6中的方法

find:查找目标元素，需要接收一个函数作为参数

当某个遍历项(students)符合item.id = student.id条件时，find会终止遍历，同时返回遍历项

找到就放回元素，找不到就返回undefind

findIndex:查找目标元素，找到就返回元素的位置（下标），找不到就放回-1

## 实现具体功能

* 通过路由收到请求
* 接收数据（ get , post）

Get :req.query

Post:req.body

* 调用students.js中的封装的对应方法
* 根据操作给客户端发送响应（有的是跳转页面，有的是重定向redirect）

跳转页面时回调函数function(err,students) {} 要带返回的结果参数

重定向时回调函数function(err){},值需要返回错误信息即可

## 业务功能顺序

* 列表
* 添加
* 修改
* 删除

# 总结

一个js做对应的事情，模块职责要单一

* db.json：存储数据
* App.js配置相关的资源

//app.js入口文件的内容职责

//1.创建服务

//2.做一些相关的配置文件

 //模板引擎

 //body-parser解析post请求体

 //提供静态资源服务

//3.挂载路由

//4.监听端口

* Router.js : 路由
* Students.js：操作文件中的数据，不关心业务