记账本项目使用restful API版本

# 1.复制项目

## 把accounts-mongoose复制一份改名：accounts-mongoose-restful-api，然后先打开终端定位到这个目录，运行npm start，发现能正常工作

|  |
| --- |
|  |
|  |

# 2.注册api路由规则

## 1）把routes文件夹下面的users.js删除，可以看到现在已经没有这个文件了

|  |
| --- |
|  |

### 为什么要删除这个文件？因为我们的项目用不到

## 2).需要把users相关的路又也处理一下，打开app.js,把user相关的代码删除

|  |
| --- |
| var createError = require('http-errors');  var express = require('express');  var path = require('path');  var cookieParser = require('cookie-parser');  var logger = require('morgan');  var indexRouter = require('./routes/index');  // var usersRouter = require('./routes/users');  var app = express();  // view engine setup  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs');  app.use(logger('dev'));  app.use(express.json());  app.use(express.urlencoded({ extended: false }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, 'public')));  app.use('/', indexRouter);  // app.use('/users', usersRouter);//这个路由规则本项目用不到先注释掉  // catch 404 and forward to error handler  app.use(function(req, res, next) {    next(createError(404));  });  // error handler  app.use(function(err, req, res, next) {    // set locals, only providing error in development    res.locals.message = err.message;    res.locals.error = req.app.get('env') === 'development' ? err : {};    // render the error page    res.status(err.status || 500);    res.render('error');  });  module.exports = app; |
| var createError = require('http-errors');  var express = require('express');  var path = require('path');  var cookieParser = require('cookie-parser');  var logger = require('morgan');  var indexRouter = require('./routes/index');  var app = express();  // view engine setup  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs');  app.use(logger('dev'));  app.use(express.json());  app.use(express.urlencoded({ extended: false }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, 'public')));  app.use('/', indexRouter);  // catch 404 and forward to error handler  app.use(function(req, res, next) {    next(createError(404));  });  // error handler  app.use(function(err, req, res, next) {    // set locals, only providing error in development    res.locals.message = err.message;    res.locals.error = req.app.get('env') === 'development' ? err : {};    // render the error page    res.status(err.status || 500);    res.render('error');  });  module.exports = app; |

## 3)在routes文件夹下面新建一个web文件夹，然后把index.js移动到web文件夹里面，然后vscode会问你是否更新index.js的倒入

|  |
| --- |
|  |

### 点击始终，然后服务器会重启，一切正常

|  |
| --- |
|  |

## 4)在routes文件夹下面新建一个api文件夹，然后在这里新建一个account.js

|  |
| --- |
|  |

## 5)把index.js的代码复制一份，然后粘贴到account.js中

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入模型  let accountModel = require('../../model/accountModel')  //导入moment库  let moment = require('moment')  // console.log(moment('2023-02-23').toDate());//ok  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由，从数据库读取所有文档数据渲染到页面  router.get('/account', function(req, res, next) {    //获取所有账单信息,按时间倒序排列    accountModel.find().sort({time:-1}).then(data=>{      res.render('list',{data:data,moment:moment});    }).catch(err=>{      res.status(500).send("读取数据失败")      return    })    });  //添加记录的路由  router.get('/account/create', function(req, res, next) {    res.render('create');  });  //接收表单数据的路由  router.post('/account', function(req, res) {    //express的脚手架帮我们添加了解析表单数据的中间件功能，所以这里我们可以直接使用req.body    // console.log(req.body);    //用mongodb保存数据    accountModel.create({        ...req.body,        time:moment(req.body.time).toDate()      })     .then(data=>{       console.log(data);       let msg = "数据添加成功"       let url = '/account'       res.render('success',{msg,url})     }).catch(err=>{       log(err)       res.status(500).send("插入数据失败")       return     })    });  //删除记录的路由  router.get('/account/:id', function(req, res, next) {    let id = req.params.id    // console.log(id)    //删除记录    accountModel.deleteOne({\_id:id}).then(data=>{        console.log(data);        let msg = "数据删除成功"        let url = '/account'        res.render('success',{msg,url}) //复用上面的success页面    }).catch(err=>{      log(err)       res.status(500).send("插入数据失败")       return    })    });  module.exports = router; |

## 6）在app.js中导入account.js,然后注册一个api路由规则

|  |
| --- |
| var createError = require('http-errors');  var express = require('express');  var path = require('path');  var cookieParser = require('cookie-parser');  var logger = require('morgan');  var indexRouter = require('./routes/web/index');  //导入account.js接口路由文件  let accountRouter = require('./routes/api/account')  var app = express();  // view engine setup  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs');  app.use(logger('dev'));  app.use(express.json());  app.use(express.urlencoded({ extended: false }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, 'public')));  app.use('/', indexRouter);  //注册api路由规则  app.use('/api', accountRouter);  // catch 404 and forward to error handler  app.use(function(req, res, next) {    next(createError(404));  });  // error handler  app.use(function(err, req, res, next) {    // set locals, only providing error in development    res.locals.message = err.message;    res.locals.error = req.app.get('env') === 'development' ? err : {};    // render the error page    res.status(err.status || 500);    res.render('error');  });  module.exports = app; |

### 测试一下，发现也是好用的

|  |
| --- |
|  |

# 3.编写获取账单接口

## 1）打开account.js文件，把get(“/account”,...)路由规则的成功处理改为res.json({

## //响应码，通常是0000或者20000，建议使用0000，注意：这里不是http状态码

## //响应信息

## //响应数据 })

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入模型  let accountModel = require('../../model/accountModel')  //导入moment库  let moment = require('moment')  // console.log(moment('2023-02-23').toDate());//ok  //显示账本列表的路由，从数据库读取所有文档数据渲染到页面  router.get('/account', function(req, res, next) {    //获取所有账单信息,按时间倒序排列    accountModel.find().sort({time:-1}).then(data=>{      // res.render('list',{data:data,moment:moment});      //api接口是返回json数据给客户端的，json数据有以下属性      //1.code，通常是0000或者20000，建议使用0000注意这里不是http状态码      //2.msg也就是信息      //3.data也就是响应数据      res.json({        code:'0000',        msg:'读取成功',        data:data      })    }).catch(err=>{      res.status(500).send("读取数据失败")      return    })    });  //添加记录的路由  router.get('/account/create', function(req, res, next) {    res.render('create');  });  //接收表单数据的路由  router.post('/account', function(req, res) {      //用mongodb保存数据    accountModel.create({        ...req.body,        time:moment(req.body.time).toDate()      })     .then(data=>{       console.log(data);       let msg = "数据添加成功"       let url = '/account'       res.render('success',{msg,url})     }).catch(err=>{       log(err)       res.status(500).send("插入数据失败")       return     })    });  //删除记录的路由  router.get('/account/:id', function(req, res, next) {    let id = req.params.id    // console.log(id)    //删除记录    accountModel.deleteOne({\_id:id}).then(data=>{        console.log(data);        let msg = "数据删除成功"        let url = '/account'        res.render('success',{msg,url}) //复用上面的success页面    }).catch(err=>{      log(err)       res.status(500).send("插入数据失败")       return    })    });  module.exports = router; |

然后我们在浏览器中输入：<http://localhost:3000/api/account/> ，修改如下

|  |
| --- |
|  |

## 2）把响应失败的代码也处理一下

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入模型  let accountModel = require('../../model/accountModel')  //导入moment库  let moment = require('moment')  // console.log(moment('2023-02-23').toDate());//ok  //显示账本列表的路由，从数据库读取所有文档数据渲染到页面  router.get('/account', function(req, res, next) {    //获取所有账单信息,按时间倒序排列    accountModel.find().sort({time:-1}).then(data=>{      // res.render('list',{data:data,moment:moment});      //api接口是返回json数据给客户端的，json数据有以下属性      //1.code，通常是0000或者20000，建议使用0000注意这里不是http状态码      //2.msg也就是信息      //3.data也就是响应数据      res.json({        code:'0000',        msg:'读取成功',        data:data      })    }).catch(err=>{      res.json({          code:'1001',//json响应失败的状态码是1xxxx，如1001          msg:'读取失败',          data:null      })      return    })    });  //添加记录的路由  router.get('/account/create', function(req, res, next) {    res.render('create');  });  //接收表单数据的路由  router.post('/account', function(req, res) {      //用mongodb保存数据    accountModel.create({        ...req.body,        time:moment(req.body.time).toDate()      })     .then(data=>{       console.log(data);       let msg = "数据添加成功"       let url = '/account'       res.render('success',{msg,url})     }).catch(err=>{       log(err)       res.status(500).send("插入数据失败")       return     })    });  //删除记录的路由  router.get('/account/:id', function(req, res, next) {    let id = req.params.id    // console.log(id)    //删除记录    accountModel.deleteOne({\_id:id}).then(data=>{        console.log(data);        let msg = "数据删除成功"        let url = '/account'        res.render('success',{msg,url}) //复用上面的success页面    }).catch(err=>{      log(err)       res.status(500).send("插入数据失败")       return    })    });  module.exports = router; |

### 测试响应失败的方法是把mongodb数据库服务停止，然后它就获取不到数据，非常困难这里就不测试了

# 4.编写添加账单接口

## 1）打开api/account.js文件，把添加账单的路由删除，因为api接口不会返回html只会返回json数据，然后我们来修改post路由规则

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入模型  let accountModel = require('../../model/accountModel')  //导入moment库  let moment = require('moment')  // console.log(moment('2023-02-23').toDate());//ok  //显示账本列表的api返回json数据  router.get('/account', function(req, res, next) {    //获取所有账单信息,按时间倒序排列    accountModel.find().sort({time:-1}).then(data=>{      // res.render('list',{data:data,moment:moment});      //api接口是返回json数据给客户端的，json数据有以下属性      //1.code，通常是0000或者20000，建议使用0000注意这里不是http状态码      //2.msg也就是信息      //3.data也就是响应数据      res.json({        code:'0000',        msg:'读取成功',        data:data      })    }).catch(err=>{      res.json({          code:'1001',//json响应失败的状态码是1xxxx，如1001          msg:'读取失败',          data:null      })      return    })    });  //接收表单数据的api  router.post('/account', function(req, res) {      //用mongodb保存数据    accountModel.create({        ...req.body,        time:moment(req.body.time).toDate()      })     .then(data=>{      res.json({          code:'0000',          msg:'插入数据成功',          data:data        })     }).catch(err=>{      res.json({          code:'1002',//json响应失败的状态码是1xxxx，如1002          msg:'插入数据失败',          data:null      })      return     })    });  //删除记录的路由  router.get('/account/:id', function(req, res, next) {    let id = req.params.id    // console.log(id)    //删除记录    accountModel.deleteOne({\_id:id}).then(data=>{        console.log(data);        let msg = "数据删除成功"        let url = '/account'        res.render('success',{msg,url}) //复用上面的success页面    }).catch(err=>{      console.log(err)       res.status(500).send("插入数据失败")       return    })    });  module.exports = router; |

### 用apipost工具测试一下，效果如下

|  |
| --- |
|  |

### 然后我们故意把title删除，再发送请求，因为我们在模型里面设置title为必须的，就会发现添加失败，返回我们设置的结果

|  |
| --- |
|  |

## 2.)其实我们可以为api添加表单验证功能，如验证是否是所有的必填项都有数据，如果不是就可以返回失败响应给用户,根据不同的错误类型返回不一样的状态码

# 5.编写删除账单的接口

## 打开api/account.js,把删除路由代码修改如下，把get方法改为delete方法

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入模型  let accountModel = require('../../model/accountModel')  //导入moment库  let moment = require('moment')  // console.log(moment('2023-02-23').toDate());//ok  //显示账本列表的路由，从数据库读取所有文档数据渲染到页面  router.get('/account', function(req, res, next) {    //获取所有账单信息,按时间倒序排列    accountModel.find().sort({time:-1}).then(data=>{      // res.render('list',{data:data,moment:moment});      //api接口是返回json数据给客户端的，json数据有以下属性      //1.code，通常是0000或者20000，建议使用0000注意这里不是http状态码      //2.msg也就是信息      //3.data也就是响应数据      res.json({        code:'0000',        msg:'读取成功',        data:data      })    }).catch(err=>{      res.json({          code:'1001',//json响应失败的状态码是1xxxx，如1001          msg:'读取失败',          data:null      })      return    })    });  //接收表单数据的路由  router.post('/account', function(req, res) {      //用mongodb保存数据    accountModel.create({        ...req.body,        time:moment(req.body.time).toDate()      })     .then(data=>{      res.json({          code:'0000',          msg:'插入数据成功',          data:data        })     }).catch(err=>{      res.json({          code:'1002',//json响应失败的状态码是1xxxx，如1001          msg:'插入数据失败',          data:null      })      return     })    });  //删除记录的api  router.delete('/account/:id', function(req, res, next) {    let id = req.params.id    // console.log(id)    //删除记录    accountModel.deleteOne({\_id:id}).then(data=>{      res.json({          code:'0000',          msg:'删除数据成功',  data:{}          })    }).catch(err=>{      res.json({          code:'1003',//json响应失败的状态码是1xxxx，如1001          msg:'删除数据失败',   data:null      })      return    })    });  module.exports = router; |

### 用apipost工具测试一下，成功

|  |
| --- |
|  |

# 6、编写获取一条账单的接口

## 打开api/account.js,添加一条路由规则get(“/account/:id”,(req,res,next)=>{})

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入模型  let accountModel = require('../../model/accountModel')  //导入moment库  let moment = require('moment')  // console.log(moment('2023-02-23').toDate());//ok  //显示账本列表的路由，从数据库读取所有文档数据渲染到页面  router.get('/account', function(req, res, next) {    //获取所有账单信息,按时间倒序排列    accountModel.find().sort({time:-1}).then(data=>{      // res.render('list',{data:data,moment:moment});      //api接口是返回json数据给客户端的，json数据有以下属性      //1.code，通常是0000或者20000，建议使用0000注意这里不是http状态码      //2.msg也就是信息      //3.data也就是响应数据      res.json({        code:'0000',        msg:'读取成功',        data:data      })    }).catch(err=>{      res.json({          code:'1001',//json响应失败的状态码是1xxxx，如1001          msg:'读取失败',          data:null      })      return    })    });  //新增账单的api  router.post('/account', function(req, res) {      //用mongodb保存数据    accountModel.create({        ...req.body,        time:moment(req.body.time).toDate()      })     .then(data=>{      res.json({          code:'0000',          msg:'插入数据成功',          data:data        })     }).catch(err=>{      res.json({          code:'1002',//json响应失败的状态码是1xxxx，如1001          msg:'插入数据失败',          data:null      })      return     })    });  //修改账单的api  router.patch('/account/:id',function(req,res,next){        let {id} = req.params        accountModel.updateOne({\_id:id},req.body)        .then(data=>{          res.json({            code:'0000',            msg:'更新数据成功',            data:data          })        }).catch(err=>{          res.json({            code:'1005',            msg:'更新数据失败',            data:null          })        })  })  //删除记录的api  router.delete('/account/:id', function(req, res, next) {    let id = req.params.id    // console.log(id)    //删除记录    accountModel.deleteOne({\_id:id}).then(data=>{      res.json({          code:'0000',          msg:'删除数据成功',          data:{}        })    }).catch(err=>{      res.json({          code:'1003',//json响应失败的状态码是1xxxx，如1001          msg:'删除数据失败',          data:null      })      return    })    });  //获取一条账单的api接口  router.get('/account/:id',function(req,res,next) {    let id = req.params.id    accountModel.findById({\_id:id}).then(data=>{              res.json({                code:'0000',                msg:'查询数据成功',                data:data            })         }).catch(err=>{          res.json({            code:'1004',            msg:'查询数据失败',            data:null        })      })  })  module.exports = router; |
|  |

### 测试一下：成功

|  |
| --- |
|  |

# 7、编写修改账单的api

## 打开api/account.js,然后添加修改账单的api路由，需要使用patch方法

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入模型  let accountModel = require('../../model/accountModel')  //导入moment库  let moment = require('moment')  // console.log(moment('2023-02-23').toDate());//ok  //显示账本列表的路由，从数据库读取所有文档数据渲染到页面  router.get('/account', function(req, res, next) {    //获取所有账单信息,按时间倒序排列    accountModel.find().sort({time:-1}).then(data=>{      // res.render('list',{data:data,moment:moment});      //api接口是返回json数据给客户端的，json数据有以下属性      //1.code，通常是0000或者20000，建议使用0000注意这里不是http状态码      //2.msg也就是信息      //3.data也就是响应数据      res.json({        code:'0000',        msg:'读取成功',        data:data      })    }).catch(err=>{      res.json({          code:'1001',//json响应失败的状态码是1xxxx，如1001          msg:'读取失败',          data:null      })      return    })    });  //新增账单的api  router.post('/account', function(req, res) {      //用mongodb保存数据    accountModel.create({        ...req.body,        time:moment(req.body.time).toDate()      })     .then(data=>{      res.json({          code:'0000',          msg:'插入数据成功',          data:data        })     }).catch(err=>{      res.json({          code:'1002',//json响应失败的状态码是1xxxx，如1001          msg:'插入数据失败',          data:null      })      return     })    });  //修改账单的api  router.patch('/account/:id',function(req,res,next){        let {id} = req.params        accountModel.updateOne({\_id:id},req.body)        .then(data=>{          res.json({            code:'0000',            msg:'更新数据成功',            data:data          })        }).catch(err=>{          res.json({            code:'1005',            msg:'更新数据失败',            data:null          })        })  })  //删除记录的api  router.delete('/account/:id', function(req, res, next) {    let id = req.params.id    // console.log(id)    //删除记录    accountModel.deleteOne({\_id:id}).then(data=>{      res.json({          code:'0000',          msg:'删除数据成功',          data:{}        })    }).catch(err=>{      res.json({          code:'1003',//json响应失败的状态码是1xxxx，如1001          msg:'删除数据失败',          data:null      })      return    })    });  //获取一条账单的api接口  router.get('/account/:id',function(req,res,next) {    let id = req.params.id    accountModel.findById({\_id:id}).then(data=>{              res.json({                code:'0000',                msg:'查询数据成功',                data:data            })         }).catch(err=>{          res.json({            code:'1004',            msg:'查询数据失败',            data:null        })      })  })  module.exports = router; |

## 我们先获取一下列表，然后我们想修改一下第二条数据，

|  |
| --- |
|  |

## 我们用apipost工具发送一条patch请求，如下，返回成功信息

|  |
| --- |
|  |

## 然后我们再看看列表，发现第二条数据已经修改了

|  |
| --- |
|  |

## 但是有一点不够好，就是返回值不太对劲，我们需要返回更新后的数据，而不是统计信息，我们来修改一下代码，解决办法就是需要调用模型对象的更新方法后，再调用模型对象的查询方法，然后把查询到的结果返回

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入模型  let accountModel = require('../../model/accountModel')  //导入moment库  let moment = require('moment')  // console.log(moment('2023-02-23').toDate());//ok  //显示账本列表的路由，从数据库读取所有文档数据渲染到页面  router.get('/account', function(req, res, next) {    //获取所有账单信息,按时间倒序排列    accountModel.find().sort({time:-1}).then(data=>{      // res.render('list',{data:data,moment:moment});      //api接口是返回json数据给客户端的，json数据有以下属性      //1.code，通常是0000或者20000，建议使用0000注意这里不是http状态码      //2.msg也就是信息      //3.data也就是响应数据      res.json({        code:'0000',        msg:'读取成功',        data:data      })    }).catch(err=>{      res.json({          code:'1001',//json响应失败的状态码是1xxxx，如1001          msg:'读取失败',          data:null      })      return    })    });  //新增账单的api  router.post('/account', function(req, res) {      //用mongodb保存数据    accountModel.create({        ...req.body,        time:moment(req.body.time).toDate()      })     .then(data=>{      res.json({          code:'0000',          msg:'插入数据成功',          data:data        })     }).catch(err=>{      res.json({          code:'1002',//json响应失败的状态码是1xxxx，如1001          msg:'插入数据失败',          data:null      })      return     })    });  //修改账单的api  router.patch('/account/:id',function(req,res,next){        let {id} = req.params        accountModel.updateOne({\_id:id},req.body)        .then(data=>{          //这里是默认的处理方式，他会返回一个统计对象，如果你不想要，就需要在下面做处理，获取你需要的数据          // res.json({          //   code:'0000',          //   msg:'更新数据成功',          //   data:data          // })         //由于我们需要更新后的数据，所以这里需要根据id查询数据,然后把结果返回           accountModel.findById(id).then(data=>{              res.json({                code:'0000',                msg:'更新数据成功',                data:data              })           })        }).catch(err=>{          res.json({            code:'1005',            msg:'更新数据失败',            data:null          })        })  })  //删除记录的api  router.delete('/account/:id', function(req, res, next) {    let id = req.params.id    // console.log(id)    //删除记录    accountModel.deleteOne({\_id:id}).then(data=>{      res.json({          code:'0000',          msg:'删除数据成功',          data:{}        })    }).catch(err=>{      res.json({          code:'1003',//json响应失败的状态码是1xxxx，如1001          msg:'删除数据失败',          data:null      })      return    })    });  //获取一条账单的api接口  router.get('/account/:id',function(req,res,next) {    let id = req.params.id    accountModel.findById({\_id:id}).then(data=>{              res.json({                code:'0000',                msg:'查询数据成功',                data:data            })         }).catch(err=>{          res.json({            code:'1004',            msg:'查询数据失败',            data:null        })      })  })  module.exports = router; |

### 测试一下，效果如图

|  |
| --- |
|  |

## 说明成功了