# 项目1是没有数据库的版本，[使用的是lowdb来模拟数据库](nodejs笔记9-node-express实战项目1.docx)

# 项目2.我们把项目1和mongodb结合起来，

## 1.我们把项目1复制一份，改名accounts-mongoose

|  |
| --- |
|  |

### 先测试一下，发现可以正常运行

|  |
| --- |
|  |

## 改造开始：

## 2安装mongoose库：npm i mongoose

|  |
| --- |
|  |

## 3.把我们上一节课编写的模块化数据库操作代码拷贝过来

|  |
| --- |
|  |

## 4.修改入口文件./bin/www,在这里导入我们写的db/db,js里面的db函数，然后把所有的代码都放入db函数里面的success回调函数里面，这里就是要保证先能够成功连接到数据库才开启web服务，否则数据库都没有连接上，开启服务没有意义，因为无法保存和读取数据又有什么用？

### ./bin/www,

|  |
| --- |
| #!/usr/bin/env node  //导入自己封装的数据库操作函数  let db = require('../db/db')  //对应db函数  db(()=>{    /\*\*     \* Module dependencies.     \*/    var app = require('../app');    var debug = require('debug')('accounts:server');    var http = require('http');    /\*\*     \* Get port from environment and store in Express.     \*/    var port = normalizePort(process.env.PORT || '3000');    app.set('port', port);    /\*\*     \* Create HTTP server.     \*/    var server = http.createServer(app);    /\*\*     \* Listen on provided port, on all network interfaces.     \*/    server.listen(port);    server.on('error', onError);    server.on('listening', onListening);    /\*\*     \* Normalize a port into a number, string, or false.     \*/    function normalizePort(val) {      var port = parseInt(val, 10);      if (isNaN(port)) {        // named pipe        return val;      }      if (port >= 0) {        // port number        return port;      }      return false;    }  /\*\*   \* Event listener for HTTP server "error" event.   \*/    function onError(error) {      if (error.syscall !== 'listen') {        throw error;      }      var bind = typeof port === 'string'        ? 'Pipe ' + port        : 'Port ' + port;      // handle specific listen errors with friendly messages      switch (error.code) {        case 'EACCES':          console.error(bind + ' requires elevated privileges');          process.exit(1);          break;        case 'EADDRINUSE':          console.error(bind + ' is already in use');          process.exit(1);          break;        default:          throw error;      }    }  /\*\*   \* Event listener for HTTP server "listening" event.   \*/    function onListening() {      var addr = server.address();      var bind = typeof addr === 'string'        ? 'pipe ' + addr        : 'port ' + addr.port;      debug('Listening on ' + bind);    }  }) |

### 测试一下，没有问题

|  |
| --- |
|  |

## 5.创建数据模型，把model文件夹里面的bookModel复制一份改名accountModel.js,然后修改代码

### model/accountModel.js

|  |
| --- |
| let mongoose = require('mongoose')  let accountSchema = new mongoose.Schema({      title:{          type:String,          required:true      },      time:Date,      type:{          type:number,          default:-1      },      account:{          type:Number,          required:true      },      remarks:String,    })  //创建模型对象,accountModel就是对应accounts集合，她可以完成数据的增删改查  let accountModel = mongoose.model("accounts",accountSchema)  //导出模型对象  module.exports = accountModel |

## 6.我们来实现插入数据功能，这里需要一个库叫做moment，先把它安装了：npm i moment

### moment的基本用法，可以传递一个日期格式的字符串，然后再调用toDate()方法

|  |
| --- |
|  |
|  |

## 7.修改config/config.js,把数据库名称改为“myaccount”，

|  |
| --- |
| //配置文件  module.exports = {      DBHOST:'127.0.0.1',      DBPORT:27017,      DBNAME:"myaccount"  } |

## 8.编写“/post”路由的代码

|  |
| --- |
| 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')  //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     })    });    module.exports = router; |

### 进入<http://127.0.0.1:3000/account/create>，填写好表单，点击添加，发现能够跳转到成功页面，我们用工具查看，发现数据添加成功

|  |
| --- |
|  |

## 9.完成读取数据库文件的功能，添加成功后转到success页面，不过这时候还不能点击链接，我们先来完成”/account”的get方法的编写

|  |
| --- |
| 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});    }).catch(err=>{      res.status(500).send("读取数据失败")      return      })    // console.log(data);    });  //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) {    // console.log(req.body); //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    //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     })    }); |

### 我们进入新增页面，添加一个数据点击添加，效果如下

|  |
| --- |
|  |

## 10还有一个问题，我们看到所有的类型都是支出，这是不对的，我们需要修改一下views/list.ejs把type===“1”改为type===1，提示我们实现一个功能，根据类型改变颜色

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>账本列表</title>      <script src="https://www.itxst.com/package/jquery-3.5.1/jquery.min.js"></script>      <script src="https://www.itxst.com/package/bootstrap-4.5.0/js/bootstrap.min.js"></script>      <link href="https://www.itxst.com/package/bootstrap-4.5.0/css/bootstrap.css" rel="stylesheet">      <script src="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/js/bootstrap-datepicker.min.js"></script>      <script src="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/locales/bootstrap-datepicker.zh-CN.min.js"></script>      <link href="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/css/bootstrap-datepicker.min.css" rel="stylesheet">      <style>          .red{              color:red;              border: 1px solid red;              background-color: #b1a2a2;              border-radius: 7px;              padding: 2px;          }          .blue{              color:blue;              border: 1px solid rgb(135, 235, 157);              border-radius: 7px;              background-color: lightblue;              padding: 2px;          }      </style>  </head>  <body>      <div class="container">              <h2>记账本</h2>              <hr>              <% data.forEach(item =>{%>                  <div>                      <ul class="list-group">                          <%  let actstatus = item.type===1?'list-group-item-success':'list-group-item-danger' %>                          <li class="list-group-item <%=actstatus%>"><%=item.time %></li>                          <!-- <li class="list-group-item active"><%=item.time %></li> -->                          <li class="list-group-item">事项：<%=item.title %></li>                          <%  let myclass = item.type===1?'blue':'red' %>                          <li class="list-group-item" >类型：<span class="<%=myclass%>"><%=item.type===1?"收入":"支出" %></span></li>                          <li class="list-group-item">金额：<strong><%=item.account %></strong></li>                          <li class="list-group-item">备注：<%=item.remarks %></li>                          <li class="list-group-item"><a class="btn btn-warning" href="/account/<%=item.id%>">删除</a></li>                      </ul>                    </div>                <%})%>      </div>    </body>  </html> |

### 刷新一下页面，效果出来了

|  |
| --- |
|  |

### 说明这个功能也做好了，

## 11.其实我们可以在这里增加一个添加按钮，跳转到/account/create,需要修改list.ejs文件

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>账本列表</title>      <script src="https://www.itxst.com/package/jquery-3.5.1/jquery.min.js"></script>      <script src="https://www.itxst.com/package/bootstrap-4.5.0/js/bootstrap.min.js"></script>      <link href="https://www.itxst.com/package/bootstrap-4.5.0/css/bootstrap.css" rel="stylesheet">      <script src="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/js/bootstrap-datepicker.min.js"></script>      <script src="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/locales/bootstrap-datepicker.zh-CN.min.js"></script>      <link href="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/css/bootstrap-datepicker.min.css" rel="stylesheet">      <style>          .red{              color:red;              border: 1px solid red;              background-color: #b1a2a2;              border-radius: 7px;              padding: 2px;          }          .blue{              color:blue;              border: 1px solid rgb(135, 235, 157);              border-radius: 7px;              background-color: lightblue;              padding: 2px;          }      </style>  </head>  <body>      <div class="container">              <h2>记账本</h2>              <hr>              <% data.forEach(item =>{%>                  <div>                      <ul class="list-group">                          <%  let actstatus = item.type===1?'list-group-item-success':'list-group-item-danger' %>                          <li class="list-group-item <%=actstatus%>"><%=item.time %></li>                          <!-- <li class="list-group-item active"><%=item.time %></li> -->                          <li class="list-group-item">事项：<%=item.title %></li>                          <%  let myclass = item.type===1?'blue':'red' %>                          <li class="list-group-item" >类型：<span class="<%=myclass%>"><%=item.type===1?"收入":"支出" %></span></li>                          <li class="list-group-item">金额：<strong><%=item.account %></strong></li>                          <li class="list-group-item">备注：<%=item.remarks %></li>                          <li class="list-group-item">                              <a class="btn btn-primary" href="/account/create">添加</a>                              <a class="btn btn-warning" href="/account/<%=item.id%>">删除</a>                          </li>                      </ul>                    </div>                <%})%>      </div>    </body>  </html> |

### 刷新页面，效果如下

|  |
| --- |
|  |

### 点击添加，就会跳转到新增页面，

|  |
| --- |
|  |

### 然后能够新增数据

|  |
| --- |
|  |

## 12.还有一个小问题，就是日期格式不太满意，我们可以调整一下，moment库也是可以做到的

|  |
| --- |
|  |

### 我们来修改一下index.js

|  |
| --- |
| 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     })    }); |

### 然后修改一下views/list.ejs

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>账本列表</title>      <script src="https://www.itxst.com/package/jquery-3.5.1/jquery.min.js"></script>      <script src="https://www.itxst.com/package/bootstrap-4.5.0/js/bootstrap.min.js"></script>      <link href="https://www.itxst.com/package/bootstrap-4.5.0/css/bootstrap.css" rel="stylesheet">      <script src="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/js/bootstrap-datepicker.min.js"></script>      <script src="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/locales/bootstrap-datepicker.zh-CN.min.js"></script>      <link href="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/css/bootstrap-datepicker.min.css" rel="stylesheet">      <style>          .red{              color:red;              border: 1px solid red;              background-color: #b1a2a2;              border-radius: 7px;              padding: 2px;          }          .blue{              color:blue;              border: 1px solid rgb(135, 235, 157);              border-radius: 7px;              background-color: lightblue;              padding: 2px;          }      </style>  </head>  <body>      <div class="container">              <h2>记账本</h2>              <hr>              <% data.forEach(item =>{%>                  <div>                      <ul class="list-group">                          <%  let actstatus = item.type===1?'list-group-item-success':'list-group-item-danger' %>                          <!-- <li class="list-group-item <%=actstatus%>"><%=item.time %></li> -->                          <li class="list-group-item <%=actstatus%>"><%=moment(item.time).format('YYYY-MM-DD') %></li>                          <li class="list-group-item">事项：<%=item.title %></li>                          <%  let myclass = item.type===1?'blue':'red' %>                          <li class="list-group-item" >类型：<span class="<%=myclass%>"><%=item.type===1?"收入":"支出" %></span></li>                          <li class="list-group-item">金额：<strong><%=item.account %></strong></li>                          <li class="list-group-item">备注：<%=item.remarks %></li>                          <li class="list-group-item">                              <a class="btn btn-primary" href="/account/create">添加</a>                              <a class="btn btn-warning" href="/account/<%=item.id%>">删除</a>                          </li>                      </ul>                    </div>                <%})%>      </div>    </body>  </html> |

### 刷新一下页面，效果出来了

|  |
| --- |
|  |

## 13.我们来实现删除数据功能，先实现基本删除功能

### a.修改list.ejs

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>账本列表</title>      <script src="https://www.itxst.com/package/jquery-3.5.1/jquery.min.js"></script>      <script src="https://www.itxst.com/package/bootstrap-4.5.0/js/bootstrap.min.js"></script>      <link href="https://www.itxst.com/package/bootstrap-4.5.0/css/bootstrap.css" rel="stylesheet">      <script src="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/js/bootstrap-datepicker.min.js"></script>      <script src="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/locales/bootstrap-datepicker.zh-CN.min.js"></script>      <link href="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/css/bootstrap-datepicker.min.css" rel="stylesheet">      <style>          .red{              color:red;              border: 1px solid red;              background-color: #b1a2a2;              border-radius: 7px;              padding: 2px;          }          .blue{              color:blue;              border: 1px solid rgb(135, 235, 157);              border-radius: 7px;              background-color: lightblue;              padding: 2px;          }      </style>  </head>  <body>      <div class="container">              <h2>记账本</h2>              <hr>              <% data.forEach(item =>{%>                  <div>                      <ul class="list-group">                          <%  let actstatus = item.type===1?'list-group-item-success':'list-group-item-danger' %>                          <!-- <li class="list-group-item <%=actstatus%>"><%=item.time %></li> -->                          <li class="list-group-item <%=actstatus%>"><%=moment(item.time).format('YYYY-MM-DD') %></li>                          <li class="list-group-item">事项：<%=item.title %></li>                          <%  let myclass = item.type===1?'blue':'red' %>                          <li class="list-group-item" >类型：<span class="<%=myclass%>"><%=item.type===1?"收入":"支出" %></span></li>                          <li class="list-group-item">金额：<strong><%=item.account %></strong></li>                          <li class="list-group-item">备注：<%=item.remarks %></li>                          <li class="list-group-item">                              <a class="btn btn-primary" href="/account/create">添加</a>                              <a class="btn btn-warning" href="/account/<%=item.\_id%>">删除</a>                          </li>                      </ul>                    </div>                <%})%>      </div>    </body>  </html> |

### b.修改index.js

|  |
| --- |
| 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    // db.get("accounts").remove({id:id}).write()    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; |

## 14.我们删除文档数据的时候要完善一下，尽量帮助用户避免误操作导致误删数据,我们需要在list.ejs里面添加对用户删除的确认处理，只有当用户点击确定才删除数据，否则取消操作

### views/list.ejs

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>账本列表</title>      <script src="https://www.itxst.com/package/jquery-3.5.1/jquery.min.js"></script>      <script src="https://www.itxst.com/package/bootstrap-4.5.0/js/bootstrap.min.js"></script>      <link href="https://www.itxst.com/package/bootstrap-4.5.0/css/bootstrap.css" rel="stylesheet">      <script src="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/js/bootstrap-datepicker.min.js"></script>      <script src="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/locales/bootstrap-datepicker.zh-CN.min.js"></script>      <link href="https://www.itxst.com/package/bootstrap-datepicker-1.9.0/css/bootstrap-datepicker.min.css" rel="stylesheet">      <style>          .red{              color:red;              border: 1px solid red;              background-color: #b1a2a2;              border-radius: 7px;              padding: 2px;          }          .blue{              color:blue;              border: 1px solid rgb(135, 235, 157);              border-radius: 7px;              background-color: lightblue;              padding: 2px;          }      </style>  </head>  <body>      <div class="container">              <h2>记账本</h2>              <hr>              <% data.forEach(item =>{%>                  <div>                      <ul class="list-group">                          <%  let actstatus = item.type===1?'list-group-item-success':'list-group-item-danger' %>                          <!-- <li class="list-group-item <%=actstatus%>"><%=item.time %></li> -->                          <li class="list-group-item <%=actstatus%>"><%=moment(item.time).format('YYYY-MM-DD') %></li>                          <li class="list-group-item">事项：<%=item.title %></li>                          <%  let myclass = item.type===1?'blue':'red' %>                          <li class="list-group-item" >类型：<span class="<%=myclass%>"><%=item.type===1?"收入":"支出" %></span></li>                          <li class="list-group-item">金额：<strong><%=item.account %></strong></li>                          <li class="list-group-item">备注：<%=item.remarks %></li>                          <li class="list-group-item">                              <a class="btn btn-primary" href="/account/create">添加</a>                              <a class="btn btn-warning delBtn" href="/account/<%=item.\_id%>">删除</a>                          </li>                      </ul>                    </div>                <%})%>      </div>   <script>      //获取所有item的删除链接      let delBtns = document.querySelectorAll('.delBtn')      //给他们绑定事件      delBtns.forEach(btn =>{         btn.addEventListener("click",function(e) {              if(confirm("删除数据?")){                  return true              } else {                 e.preventDefault()              }         })      })   </script>  </body>  </html> |

### 测试一下，我们点击一个事项，如：买菜，然后点击取消

|  |
| --- |
|  |

### 刷新后，发现数据还在

|  |
| --- |
|  |

### 然后我们再点击买菜事项的删除，然后在弹出的对话框中点击确定

|  |
| --- |
|  |

### 点击确定，删除成功

|  |
| --- |
|  |

### 点击跳转按钮，回到列表

|  |
| --- |
|  |

### 发现买菜没有了，的确删除成功

# 到此，第二个版本的记账本项目完成