nodejs项目实战1-记账本

# 0.效果展示

|  |
| --- |
|  |
|  |

# 1.基础框架搭建，

## 这个练习项目叫做accounts，我们使用express-generator来创建，前提是已经全局安装了express-generator，我们在nodejs-code目录下面新建一个projects文件夹，然后在这里打开终端，输入：express -e accounts，然后就会生成基本结构，如图

|  |
| --- |
|  |

# 2.切换到accounts目录下面，运行npm install安装依赖

|  |
| --- |
|  |

# 3.如果你全局安装了nodemon，可以把start脚本命令改为”nodemom .bin/www”

|  |
| --- |
|  |

# 4.测试一下，正常运行

|  |
| --- |
|  |

# 5.我们先设置路由规则

## 打开app.js,把/users规则注释掉，因为本练习用不到，然后进入到routes/index.js,我们在这里设置路由规则，我们先写一些最基本的代码

|  |
| --- |
| var express = require('express');  var router = express.Router();  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    // res.render('index', { title: 'Express' });    res.send("账本列表")  });  //添加记录的路由  router.get('/account/create', function(req, res, next) {    // res.render('index', { title: 'Express' });    res.send("添加记录")  });  module.exports = router; |

### 效果：

|  |
| --- |
|  |
|  |

# 6.添加静态网页响应功能

## 我们打开index.js文件，修改路由/account的处理函数

|  |
| --- |
| var express = require('express');  var router = express.Router();  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    res.render('list');  });  //添加记录的路由  router.get('/account/create', function(req, res, next) {    // res.render('index', { title: 'Express' });    res.send("添加记录")  });  module.exports = router; |

## 然后我们在views文件夹里面新建一个create.ejs文件和list.ejs

### ./views/create.ejs

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>添加记录</title>  </head>  <body>      <div>          <h2>添加记录</h2>          <hr>          <form action="/account" method="post">              <div>                  <label for="title">事项</label>  <br/>                  <input type="text" name="title">              </div>              <div>                  <label for="time">发生时间</label> <br/>                  <input type="text" name="time">              </div>              <div>                  <label for="type">类型</label>   <br/>                  <select name="type" id="type"><br/>                      <option value="-1">支出</option>                      <option value="1">收入</option>                  </select>              </div>              <div>              <label for="account">金额</label>  <br/>              <input type="text" name="account">              </div>              <label for="remarks">备注</label>   <br/>              <textarea name="remarks"></textarea>              </div>              <hr>              <div>                  <button>添加</button>              </div>            </form>        </div>  </body>  </html> |

### ./views/list.ejs暂时没有编写，我们后面会编写

### 几经周折，我找到了在线使用bootstrap-datepicker的方法，参考这个网站：[bootstrap-datepicker中文文档 - itxst.com](https://www.itxst.com/bootstrap-datepicker/tutorial.html)，

### 我们需要引入相应的依赖，然后在一个script标签中新建一个日期对象，引用的代码如下，也就是我们的creat2.ejs

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>添加记录</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">  </head>  <body>      <div>          <h2>添加记录</h2>          <hr>          <form action="/account" method="post">              <div>                  <label for="title">事项</label>  <br/>                  <input type="text" name="title">              </div>              <div>                  <label for="time">发生时间</label> <br/>                  <input id="inputDate" class="form-control" placeholder="请选择日期" name="time">              </div>              <div>                  <label for="type">类型</label>   <br/>                  <select name="type" id="type"><br/>                      <option value="-1">支出</option>                      <option value="1">收入</option>                  </select>              </div>              <div>              <label for="account">金额</label>  <br/>              <input type="text" name="account">              </div>              <label for="remarks">备注</label>   <br/>              <textarea name="remarks"></textarea>              </div>              <hr>              <div>                  <button>添加</button>              </div>            </form>        </div>      <script>          $("#inputDate").datepicker({              language: 'zh-CN', //语言              autoclose: true, //选择后自动关闭              clearBtn: true,//清除按钮              format: "yyyy-mm-dd"//日期格式          });      </script>  </body>  </html> |

### 然后我看到老师的class属性设置，我也成功的使用了在线bootstrap，修改后的create2.ejs内容如下

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>添加记录</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">  </head>  <body>      <div class="container">          <div class="row">              <div class=" col-xs-12 col-lg-8 col-lg-offset-2">                  <h2>添加记录</h2>          <hr>              <form action="/account" method="post">                  <div class="form-group">                      <label for="title">事项</label>  <br/>                      <input type="text" name="title" class="form-control">                  </div>                  <div class="form-group">                      <label for="time">发生时间</label> <br/>                      <input id="inputDate" class="form-control" placeholder="请选择日期" name="time">                  </div>                  <div class="form-group">                      <label for="type">类型</label>   <br/>                      <select name="type" id="type" class="form-control"><br/>                          <option value="-1">支出</option>                          <option value="1">收入</option>                      </select>                  </div>                  <div class="form-group">                      <label for="account">金额</label>  <br/>                      <input type="text" name="account" class="form-control">                  </div>                  <div class="form-group">                      <label for="remarks">备注</label>   <br/>                      <textarea name="remarks" class="form-control"></textarea>                  </div>                  <hr>                  <div class="form-group">                      <button type="submit" class="btn btn-primary btn-block">添加</button>                  </div>                  </form>                </div>          </div>      </div>            <script>          $("#inputDate").datepicker({              language: 'zh-CN', //语言              autoclose: true, //选择后自动关闭              clearBtn: true,//清除按钮              format: "yyyy-mm-dd"//日期格式          });      </script>  </body>  </html> |

#### 效果，已经和老师的差不多了

|  |
| --- |
|  |

### 然后我们可以把原来的create.ejs删除，把create2.ejs改名为create.ejs

# 7.获取表单数据

## 打开index.js,添加一条路由post规则，用来处理表单数据

|  |
| --- |
| var express = require('express');  var router = express.Router();  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    res.render('list');  });  //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    res.send('添加记录')  });  module.exports = router; |

### 测试一下，是能够拿到数据的

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

# 8.lowdb的使用

## 1, 安装，我们这里使用1.0.0的版本：npm i lowdb@1.0.0

## 2.使用

|  |
| --- |
|  |

### 1）创建数据文件并且初始化数据

### 我们在项目中新建一个test文件夹，在里面新建一个lowdb.js文件，输入如下代码

|  |
| --- |
|  |
| const low = require('lowdb')  const FileSync = require('lowdb/adapters/FileSync')    const adapter = new FileSync('db.json')  //创建数据库  const db = low(adapter)  //初始化数据  db.defaults({ posts: [], user: {} }).write() |

### 然后我们打开命令行，定位到test目录下面，运行命令：node lowdb.js,会生成一个db.json文件，里面有一个空数组和一个空对象、

|  |
| --- |
|  |
|  |

### 2）写入数据

|  |
| --- |
| const low = require('lowdb')  const FileSync = require('lowdb/adapters/FileSync')    const adapter = new FileSync('db.json')  //创建数据库  const db = low(adapter)  //初始化数据  // db.defaults({ posts: [], user: {} }).write()  //写入数据  db.get('posts').push({id:1,title:"今天天气非常热，让你受不了"}).write() |

### 再次运行文件，发现db.json里面的数据被修改了

|  |
| --- |
|  |
|  |

### 也可以在前面插入，使用数组的unshift方法

|  |
| --- |
| const low = require('lowdb')  const FileSync = require('lowdb/adapters/FileSync')    const adapter = new FileSync('db.json')  //创建数据库  const db = low(adapter)  //初始化数据  // db.defaults({ posts: [], user: {} }).write()  //写入数据  // db.get('posts').push({id:1,title:"今天天气非常热，让你受不了"}).write()  db.get('posts').unshift({id:2,title:"今天真走运，买到很好的菜"}).write() |

### 再次运行lowdb.js,效果如下

|  |
| --- |
|  |
|  |

### 3）获取数据

#### a.获取所有数据

|  |
| --- |
| const low = require('lowdb')  const FileSync = require('lowdb/adapters/FileSync')    const adapter = new FileSync('db.json')  //创建数据库  const db = low(adapter)  //初始化数据  // db.defaults({ posts: [], user: {} }).write()  //写入数据  // db.get('posts').push({id:1,title:"今天天气非常热，让你受不了"}).write()  // db.get('posts').unshift({id:2,title:"今天真走运，买到很好的菜"}).write()  //获取数据  //1.获取所有数据  console.log(db.get('posts').value()); |

#### 效果

|  |
| --- |
|  |

#### b.查找数据

|  |
| --- |
| const low = require('lowdb')  const FileSync = require('lowdb/adapters/FileSync')    const adapter = new FileSync('db.json')  //创建数据库  const db = low(adapter)  //初始化数据  // db.defaults({ posts: [], user: {} }).write()  //写入数据  // db.get('posts').push({id:1,title:"今天天气非常热，让你受不了"}).write()  // db.get('posts').unshift({id:2,title:"今天真走运，买到很好的菜"}).write()  //获取数据  //1.获取所有数据  // console.log(db.get('posts').value());  //2.查找数据  console.log(db.get('posts').find({ id: 1 }).value()); |

#### 运行文件效果如下

|  |
| --- |
|  |

### 4）更新数据，需要先找到这一条数据，然后调用assign方法传递新值然后再调用write方法

|  |
| --- |
| const low = require('lowdb')  const FileSync = require('lowdb/adapters/FileSync')    const adapter = new FileSync('db.json')  //创建数据库  const db = low(adapter)  //初始化数据  // db.defaults({ posts: [], user: {} }).write()  //写入数据  // db.get('posts').push({id:1,title:"今天天气非常热，让你受不了"}).write()  // db.get('posts').unshift({id:2,title:"今天真走运，买到很好的菜"}).write()  //获取数据  //1.获取所有数据  // console.log(db.get('posts').value());  //2.查找数据  // console.log(db.get('posts').find({ id: 1 }).value());  //3.修改一条数据  db.get('posts').find({'id': 1}).assign({title:"今天真冷啊。。。"}).write() |

#### 效果

|  |
| --- |
|  |

### 5）删除数据，只需要找到它的id

|  |
| --- |
| const low = require('lowdb')  const FileSync = require('lowdb/adapters/FileSync')    const adapter = new FileSync('db.json')  //创建数据库  const db = low(adapter)  //初始化数据  // db.defaults({ posts: [], user: {} }).write()  //写入数据  // db.get('posts').push({id:1,title:"今天天气非常热，让你受不了"}).write()  // db.get('posts').unshift({id:2,title:"今天真走运，买到很好的菜"}).write()  //获取数据  //1.获取所有数据  // console.log(db.get('posts').value());  //2.查找数据  // console.log(db.get('posts').find({ id: 1 }).value());  //3.修改一条数据  // db.get('posts').find({'id': 1}).assign({title:"今天真冷啊。。。"}).write()  //删除数据  db.get('posts').remove({id:1}) .write();//把id为1的那条数据删除 |

#### 效果

|  |
| --- |
|  |

#### 其实这个操作是有返回值的，返回被删除的对象

|  |
| --- |
|  |

### 操作db里面的user对象

#### 1>设置属性

|  |
| --- |
| const low = require('lowdb')  const FileSync = require('lowdb/adapters/FileSync')    const adapter = new FileSync('db.json')  //创建数据库  const db = low(adapter)  //初始化数据  // db.defaults({ posts: [], user: {} }).write()  //写入数据  // db.get('posts').push({id:1,title:"今天天气非常热，让你受不了"}).write()  // db.get('posts').unshift({id:2,title:"今天真走运，买到很好的菜"}).write()  //获取数据  // console.log(db.get('posts').find({ id: 1 }).value());  //编辑users对象  db.set('user.name',"Mary").write()  db.set('user.gender',"Female").write()  db.set('user.age',30).write() |

#### 效果

|  |
| --- |
|  |

#### 2.获取属性值

|  |
| --- |
| const low = require('lowdb')  const FileSync = require('lowdb/adapters/FileSync')    const adapter = new FileSync('db.json')  //创建数据库  const db = low(adapter)  //初始化数据  // db.defaults({ posts: [], user: {} }).write()  //写入数据  // db.get('posts').push({id:1,title:"今天天气非常热，让你受不了"}).write()  // db.get('posts').unshift({id:2,title:"今天真走运，买到很好的菜"}).write()  //获取数据  // console.log(db.get('posts').find({ id: 1 }).value());  //设置user对象的属性  // db.set('user.name',"Mary").write()  // db.set('user.gender',"Female").write()  // db.set('user.age',30).write()  //获取user的属性  console.log(db.get('user.name').value());  console.log(db.get('user.gender').value());  console.log(db.get('user.age').value()); |

#### 效果：

|  |
| --- |
|  |

# 9.保存账单记录

## 1.）我们新一个data文件夹，在里面新建一个db.json

|  |
| --- |
|  |

### 内容如下

|  |
| --- |
| {      "account":[]  } |

## 2）然后把我们的测试代码在的初始化数据的部分引入index.js中

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入lowdb库  const low = require('lowdb')  const FileSync = require('lowdb/adapters/FileSync')   //创建数据适配器  const adapter = new FileSync(\_\_dirname+'/../data/db.json')  //创建数据库  const db = low(adapter)  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    res.render('list');  });  //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    res.send('添加记录')  });  module.exports = router; |

## 3）在post路由里面添加保存数据的方法，其实还是满简单的：db.get("account").push(req.body).write()

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入lowdb库  const low = require('lowdb')  const FileSync = require('lowdb/adapters/FileSync')    //创建数据适配器  const adapter = new FileSync(\_\_dirname+'/../data/db.json')  //创建数据库  const db = low(adapter)  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    res.render('list');  });  //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lowdb保存数据    db.get("account").push(req.body).write()    res.send('添加记录')  });  module.exports = router; |

#### 效果

|  |
| --- |
|  |

## 4）代码需要改良，因为我们的数据没有id编号，使用起来不方便，这里需要用的一个工具包，叫做shortId，

### a、安装

|  |
| --- |
| npm i shortid |

### b.使用，就是在添加数据的时候用...扩展操作符把req.body拆分开来，放入一个对象，然后再给对象添加一个属性

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入lowdb库  const low = require('lowdb')  //引入shortid随机自动生成id  let shortid = require('shortid')  const FileSync = require('lowdb/adapters/FileSync')    //创建数据适配器  const adapter = new FileSync(\_\_dirname+'/../data/db.json')  //创建数据库  const db = low(adapter)  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    res.render('list');  });  //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lowdb保存数据    db.get("account").push({id:shortid.generate(),...req.body}).write()    res.send('添加记录')  });  module.exports = router; |

### 效果：现在我们的每一条数据都是有id的了

|  |
| --- |
|  |

## 5）还有一点需要改良，就是增加数据时不要使用push，用unshift方法比较好，目的就是让新数据在上面，方便查看

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入lowdb库  const low = require('lowdb')  //引入shortid随机自动生成id  let shortid = require('shortid')  const FileSync = require('lowdb/adapters/FileSync')    //创建数据适配器  const adapter = new FileSync(\_\_dirname+'/../data/db.json')  //创建数据库  const db = low(adapter)  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    res.render('list');  });  //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lowdb保存数据    db.get("account").unshift({id:shortid.generate(),...req.body}).write()    res.send('添加记录')  });  module.exports = router; |

### 效果

|  |
| --- |
|  |

## 6）完善成功提示，

### 我们在views里面新建一个success.ejs,内容如下

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>添加成功</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>        .h-50{          height: 50px;        }     </style>  </head>  <body>      <div class="container">          <div class="h-50"></div>          <div class="alert alert-success" role="alert">              <h1>:) 数据添加成功</h1>              <p>点击跳转</p>          </div>      </div>  </body>  </html> |

### 然后，我们修改运行index.js,在post路由里面把最后一句话改为res.render(‘success’)

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入lowdb库  const low = require('lowdb')  //引入shortid随机自动生成id  let shortid = require('shortid')  const FileSync = require('lowdb/adapters/FileSync')    //创建数据适配器  const adapter = new FileSync(\_\_dirname+'/../data/db.json')  //创建数据库  const db = low(adapter)  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    res.render('list');  });  //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lowdb保存数据    db.get("account").unshift({id:shortid.generate(),...req.body}).write()    res.render('success')  });  module.exports = router; |

### 效果

|  |
| --- |
|  |

## 7.）我们想把提示的内容变为动态的，使得添加数据成功后显示添加成功，删除数据后显示删除成功，需要使用模板语法

### a.我们修改一下index.js文件，给模板文件传递一个msg参数

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入lowdb库  const low = require('lowdb')  //引入shortid随机自动生成id  let shortid = require('shortid')  const FileSync = require('lowdb/adapters/FileSync')    //创建数据适配器  const adapter = new FileSync(\_\_dirname+'/../data/db.json')  //创建数据库  const db = low(adapter)  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    res.render('list');  });  //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lowdb保存数据    db.get("account").unshift({id:shortid.generate(),...req.body}).write()    let msg = "数据添加成功"    res.render('success',{msg})  });  module.exports = router; |

### b.然后我们在success.ejs模板文件里面输出index.js传递进来的信息

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>添加成功</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>        .h-50{          height: 50px;        }     </style>  </head>  <body>      <div class="container">          <div class="h-50"></div>          <div class="alert alert-success" role="alert">              <h1>:) <%=msg %></h1>              <p>点击跳转</p>          </div>      </div>  </body>  </html> |

#### 效果是一样的，不过这样子更灵活，同一个页面可以用在不同的地方显示不同的信息

|  |
| --- |
|  |

## 8）我们需要给p标签添加一个超链接，使得数据添加成功后我们点击一个链接能够跳转到我们指定的页面，所以这个页面也需要在index.js这传入，

### a.我们先修改index.js

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入lowdb库  const low = require('lowdb')  //引入shortid随机自动生成id  let shortid = require('shortid')  const FileSync = require('lowdb/adapters/FileSync')    //创建数据适配器  const adapter = new FileSync(\_\_dirname+'/../data/db.json')  //创建数据库  const db = low(adapter)  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    res.render('list');  });  //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lowdb保存数据    db.get("account").unshift({id:shortid.generate(),...req.body}).write()    let msg = "数据添加成功"    let url = '/account'    res.render('success',{msg,url})  });  module.exports = router; |

### b.然后我们修改success.ejs模板文件

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>      <meta charset="UTF-8">      <meta name="viewport" content="width=device-width, initial-scale=1.0">      <title>添加成功</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>        .h-50{          height: 50px;        }     </style>  </head>  <body>      <div class="container">          <div class="h-50"></div>          <div class="alert alert-success" role="alert">              <h1>:) <%=msg %></h1>              <p> <a href="<%=url %>">点击跳转</a></p>          </div>      </div>  </body>  </html> |

### 效果，添加成功后会出现如下界面，点击后会跳转到指定的页面

|  |
| --- |
|  |
|  |

# 10.实现账单列表功能

## 1）先在index.js中对account的get路由添加实现功能，其实这个满简单，就是在渲染页面的时候给他传递一个数据

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入lowdb库  const low = require('lowdb')  //引入shortid随机自动生成id  let shortid = require('shortid')  const FileSync = require('lowdb/adapters/FileSync')    //创建数据适配器  const adapter = new FileSync(\_\_dirname+'/../data/db.json')  //创建数据库  const db = low(adapter)  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    //获取所有账单信息    let data = db.get('accounts').value()    console.log(data);    res.render('list',{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lowdb保存数据    db.get("accounts").unshift({id:shortid.generate(),...req.body}).write()    let msg = "数据添加成功"    let url = '/account'    res.render('success',{msg,url})  });  module.exports = router; |

## 2）编写list.ejs模板文件，我用的是列表样式，效果和老师的不太一样，不过没有关系，还算可以，我实现了根据type的类型改变显示样式

|  |
| --- |
| <!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">                          <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"><button class="btn btn-warning">删除</button></li>                      </ul>                    </div>                <%})%>      </div>    </body>  </html> |

### 效果

|  |
| --- |
|  |

# 11.删除账单功能

## 1）在index.js里面添加一条删除记录的路由，需要动态路由，删除成功后给一个链接回调账单列表

|  |
| --- |
| var express = require('express');  var router = express.Router();  //引入lowdb库  const low = require('lowdb')  //引入shortid随机自动生成id  let shortid = require('shortid')  const FileSync = require('lowdb/adapters/FileSync')    //创建数据适配器  const adapter = new FileSync(\_\_dirname+'/../data/db.json')  //创建数据库  const db = low(adapter)  /\* GET home page. \*/  // router.get('/', function(req, res, next) {  //   res.render('index', { title: 'Express' });  // });  //显示账本列表的路由  router.get('/account', function(req, res, next) {    //获取所有账单信息    let data = db.get('accounts').value()    console.log(data);    res.render('list',{data});    // res.render('list2',{data}); //不好看  });  //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lowdb保存数据    db.get("accounts").unshift({id:shortid.generate(),...req.body}).write()    let msg = "数据添加成功"    let url = '/account'    res.render('success',{msg,url})  });  //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()    // res.send("数据删除成功：<a href='/account'>返回账本列表</a>")    let msg = "数据删除成功"    let url = '/account'    res.render('success',{msg,url}) //复用上面的success页面  });  module.exports = router; |

## 2）然后在list.ejs里面把删除按钮改为超链接

|  |
| --- |
| <!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">                          <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> |

### 效果：

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

# 功能暂时完成