

深圳大学实验报告

课程名称： 基于 Web 的编程

实验项目名称： 实验 5 购物网站设计

学院： 计算机软件学院

专业： 软件工程

指导教师： 尹剑飞

报告人： 洪继耀 学号： 2014150120 班级： 02

一、 实验要求

对于购物网站系统的如下关键设计要点，实验者需逐个给出设计方案、源代码、运行结果、设计要求是否达成的分析：

1. 用户注册与登录

设计服务器端用户会话的管理方案，包括：

数据库中用户密码的管理方案

采用的加密算法

JWT

登录验证码的设计与实现。

2. 购物篮的设计与实现

购物篮的存储方案选择

购物篮中的物品与数据库中的相应物品的数据如何保持同步？

在用户提交买单请求时，如何访问支付接口（如支付宝或微信），支付完成后，如何实现页面跳转？

3. REST API 的调用安全性

如何防止非法用户调用服务器端暴露的 REST API？

若采用 JWT(JSON Web Token)，如何设计 Controller、Filter，前端如何存储与管理 JWT？

4. 设计说明

数据库设计

用户表 Schema

User

```
{  
  
  userName:      String,  
  
  userPass:      String // sha1 加密  
  
  userBalance:   Number // 用户余额  
  
  isManager:     Boolean   // 是否是管理员  
  
}
```

产品表 Schema

Product

```
{  
  
  productName:    String  
  
  productPrice:   Number // 价格  
  
  productInventory: Number // 库存  
  
  productDetails: String // 详细描述  
  
}
```

订单表 Schema

Order

```
{
```

```

orderCreator: { // 下单者
  type: mongoose.Schema.Types.ObjectId,
  ref: 'User'
},
orderDetails: [{ // 产品列表
  number: Number,
  product: {
    _id: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Product'
    }
  }
}],
orderPrice: Number, // 总价格
orderTime: Date // 下订单时间
}

```

前后端交互设计

概述

前后端 API 是在 RESTful 风格的, 发送和返回内容均为 json 数据。若请求的操作成功：

- GET 直接返回获取的内容
- POST 和 PUT 返回修改或创建的条目的 ID，以及成功消息

- DELETE 返回成功消息

正常返回数据：

```
{  
  
    code : 0,                // 0 代表成功  
  
    /* 按实际要求封装的 data */  
  
}
```

异常返回：

```
{  
  
    code : -1  
  
    msg : String            // 具体的错误信息  
  
}
```

接口设计

用户相关

1001 登录

PUT /user

```
Params {  
  
    userName : String,  
  
    userPass : String // Hash 过的密码  
  
}
```

```
Response {  
  
    code      : 0
```

```
    _id      : ObjectId

    token    : String

    userMoney : Number

    isManager : Boolean
}
```

1002 注册

POST /user

```
Params {

    userName : String,

    userPass : String // Hash 过的密码
}
```

```
Response {

    code : 0
}
```

产品相关

2001 获取产品列表

GET /product

```
Params {

}

Response {

    code : 0

    products : [Product]
```

```
}
```

2002 更新某产品信息

PUT /product

Params {

token : String

product : Product

```
}
```

Response {

code : 0

```
}
```

2003 删除某产品

DELETE /product

Params {

token : String

productId : ObjectId

```
}
```

Response {

code : 0

```
}
```

2004 增加某产品

POST /product

Params {

```
    token    : String

    product  : Product
}

Response {

    code      : 0

    productId : ObjectId
}
```

订单相关

3001 下订单

POST /order

```
Params {

    token      : String

    orderCreator: ObjectId

    orderDetails: [{

        product    : ObjectId,

        number     : Number

    }]

    orderPrice : Number
}

Response {

    code      : 0

    orderId   : ObjectId
}
```



```
        orderTime : Date
    }

3002 查看某人所有历史订单

GET /order

Params {

    token      : String

    orderCreator: ObjectId
}

Response {

    code      : 0

    orders    : [Order]
}
```

其他设计问题

用户注册与登录

数据库中用户密码的管理方案

用户密码在前端中加密，无论是登录还是注册，ajax 请求发送的是已经用 sha1 加密过的密码，数据库存加密过后的密码，这样就能保证传输和存储安全，只有用户知道原文是什么，即使被抓包或者拖库也不会使用户面临撞库的危险。

采用的加密算法

采用了 sha1，严格来说它不是一种加密算法，因为加密算法是基于密钥的，并且有了密钥是可以解密的，sha1 是一种不可逆的算法，被称为数字签名算法，即使截取了 sha1 处理过的文本，也无从知晓原文，但是认证起来却方便，这种算法适用于数据库中的认证信息存储。

客户端 Cookie 的加密与时间有效期的管理方案

这里采用 token 而不是 session-cookie 机制，token 同样是有有效期的

登录验证码的设计与实现

可以使用 node-ccap 模块来实现，这个模块可以渲染出验证码，然后我们就可以把验证码和答案发送到前端，前端进行验证码验证。

购物篮的设计与实现

购物篮的存储方案选择

购物篮可以通过 Vuex 方便地存储在客户端的 localStorage 里，这样下次打开可以取出

购物篮中的物品与数据库中的相应物品的数据如何保持同步

可以用定时器来不断 ajax 更新，不过我觉得保持同步并不是很必要，因为除了库存之外，物品数据的变化不会太频繁，只要确认订单的时候更新以下就行了。

而对于库存，淘宝也没有能做到同步，可以考虑抢购场景，经常是看到还有，实际上买不到了。

在不同的机器上以相同的用户身份登录，如何保持购物篮同步

将购物栏数据存入数据库，然后登录的时候同时取出就行了。

提交买单请求时如何访问支付接口（如微信），支付完成后如何实现页面跳转？

首先要申请支付接口，这个是需要花钱的，所以我没有实现。

然后就只需要简单地配置一下，调用那个接口，向微信的服务器请求，然后根据返回支付结果，更新自己的数据库。

至于页面跳转，由于我做的是单页应用，并没有跳转这个东西。直接把支付弹框组件 destroy 掉就行了。

抢购商品的应用场景下的订单问题

- 如何将一件商品短期交给一个用户以排它方式持有？

在数据库创建一个临时的未完成订单数据项，保存已经提交的订单信息，给库存减去相应数量，过期或者完成订单后销毁。

- 系统如何能够保证在持有期不超过的情况下，用户买单时该商品没有被卖出？

库存信息加锁，库存如果不大于购买量，其他用户不能买。

- 在持有期超过的情况下，该商品可以被其它用户再次持有？

此时未完成订单数据项销毁，给库存增加相应数量。

- 在秒杀抢购商品的应用场景下，如何将一件商品短期交给多个用户以非排它方式持有，系统如何能够保证一件商品不会卖给两个或以上的用户？

锁有不同机制，使用相应的锁机制可以保证这种情况的正常访问。

REST API 的调用安全性

如何防止非法用户调用服务器端暴露的 REST API ?

使用 JWT 来认证用户的身份，非法用户并不能获取合法的用户的 token

若采用 JWT，如何设计 Controller、Filter，前端如何存储与管理 JWT ?

在后端使用一个路由拦截器，只有检查到合法的未过期的有效 token 才调用 next() 放行，其他情况返回 401 未认证。

代码示例如下：

```
// app.js
/**
 * jwt 验证路由
 */
let jwtAuth = require('./routes/jwtAuth');
app.all('/api/*', [ bodyParser(), jwtAuth ]);

// jwtAuth.js
module.exports = async function(req, res, next) {
  let token = (req.body && req.body.token)
  || (req.query && req.query.token) || req.headers[ 'x-access-token' ];
  if(token) {
    try {
      let decoded = jwt.decode(token, "我是密钥");
      if(decoded.exp <= Date.now()) {
        // 过期 返回 401
      }
      // 访问数据库取出用户信息
      let theUser = await User.fetchById(decoded.iss);
      if(theUser == null) {
        // 无效 返回 401
      }
      // token 有效 放行
      next();
    } catch(err) {
```

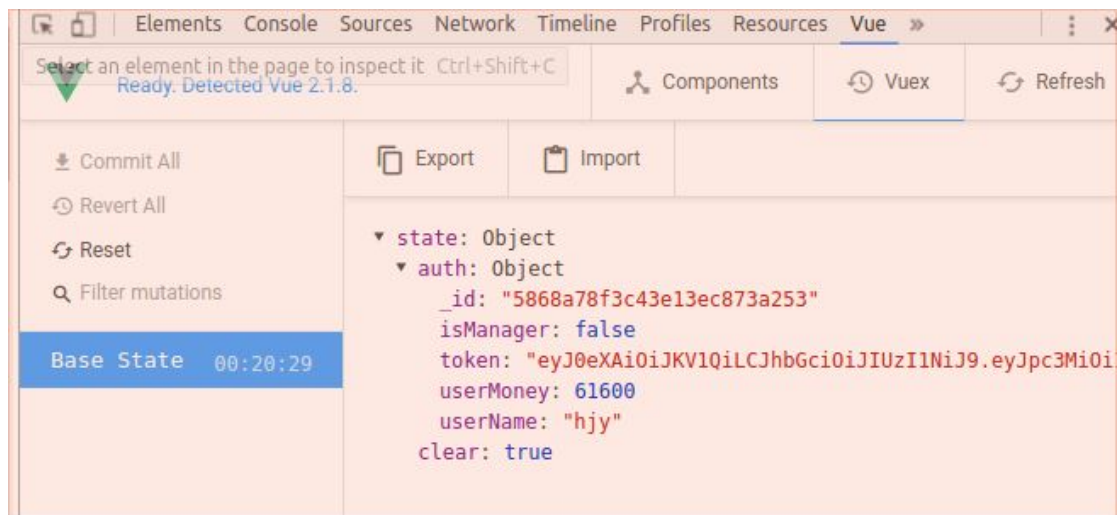
```

        // 数据库访问失败 返回 500
    }
} else {
    // 无 token 返回 401
}
};

```

前端可以通过 Vuex+localStorage 来管理用户数据包括 token ,使得各个组件都能正常访问它。

如图：



会话有效期超过之后 ,在无需用户手工登录的需求下如何自动续期？

安全性有何弱点？

我并不认为自动续期能带来什么好处，反而容易产生如 csrf 这样的安全隐患，用户手工登录一下并不会多麻烦，而且与其自动续期，还不如设置不过期。

关于 Vue 解决了什么

1. 组件化开发

2. 将数据驱动映射到视图（避免复杂易错的 dom 操作）

3. Vue 自带反 XSS 攻击（非常安全）

关于 Vue-router 解决了什么

前端的“路由”类似于后端的“路由”，通过监控 URI 的变化，来动态切换 Vue 组件，达成不必刷新就能产生视图变化，以此构建单页应用

关于 Vuex 解决了什么

Vue 是一个组件树，父子组件通信很方便

但是，有一些场景下用 Vue 原生的组件通信非常麻烦，比如兄弟组件通信必须由子组件通知父组件，父组件通知子组件

这种时候我们要使用 Vuex

关于使用 axios 作为 ajax 库而不是 jquery

使用 Axios 的好处

1. 足够轻量(10kb)，jquery(85kb)太重了
2. 自动转换请求和响应数据为 json
3. 自带支持防止 XSRF（跨站请求伪造）攻击，因为使用了类似于 Referer 检查的技术，可以检查发送源域名

关于使用 JWT 而不是传统的 session-cookie

1. 两者都是解决 http 无状态问题的工具，使用其一就可以了

2.cookie 存在着一定的安全隐患，容易被 xss 攻击获取

3.session 方式存储用户 id 的弊病要占用大量服务器内存，对于较大型应用而言可能还要保存许多的状态

4.cookie-session 存在跨域问题

5.使用 jwt 在时间效率上不如 session-cookie，是因为它的加密算法非常复杂，但是因此不需要浪费大量内存，这是一种时空权衡。

5. 成果展示

打开页面后自动跳转到‘ /login’ 路由，进行登录或者注册操作



带有前端表单验证

A screenshot of a web application interface. At the top, there is a dark blue header bar with the text '★图书商城' in white. Below the header, there is a light gray rectangular box containing a login form. The form has two tabs: '登录' (Login) and '注册' (Register), with '登录' being the active tab. There is a text input field with the value 'hjl'. Below it is a password input field with the placeholder text '请输入密码' (Please enter password) and a red error message '不能留空' (Cannot be empty) below it. At the bottom of the form is a blue button with the text '登录' (Login). Below the login form, there is a dark blue footer bar with the text 'Vue TaoBao v0.1 beta powered by Vue.' and a small circular logo.


登录后自动判断用户是不是管理员，是普通人跳转到‘ product’ 路由，是管理员调到 ‘manager’ 路由，渲染对应组件

★图书商城

产品列表


我的订单

购物车 ¥ 0



商品名称 : CSAPP3
价格(元) : 10000
库存(件) : 1
详细描述 : 23333

购买数量 : 0



商品名称 : CSAPP2
价格(元) : 200
库存(件) : 9910
详细描述 : 23333333

★图书商城

管理商品信息

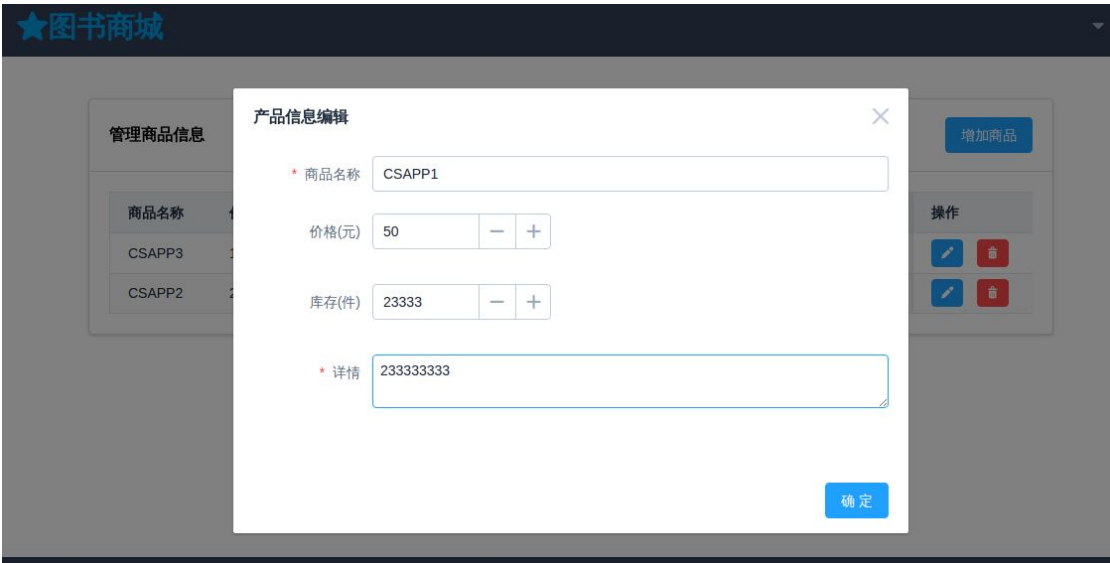
增加商品

商品名称	价格(元)	库存(件)	描述	操作
CSAPP3	10000	1	23333	<div><div></div><div></div></div>
CSAPP2	200	9910	23333333	<div><div></div><div></div></div>

Vue TaoBao v0.1 beta powered by Vue.

管理员页面可以添加商品、操作或者删除

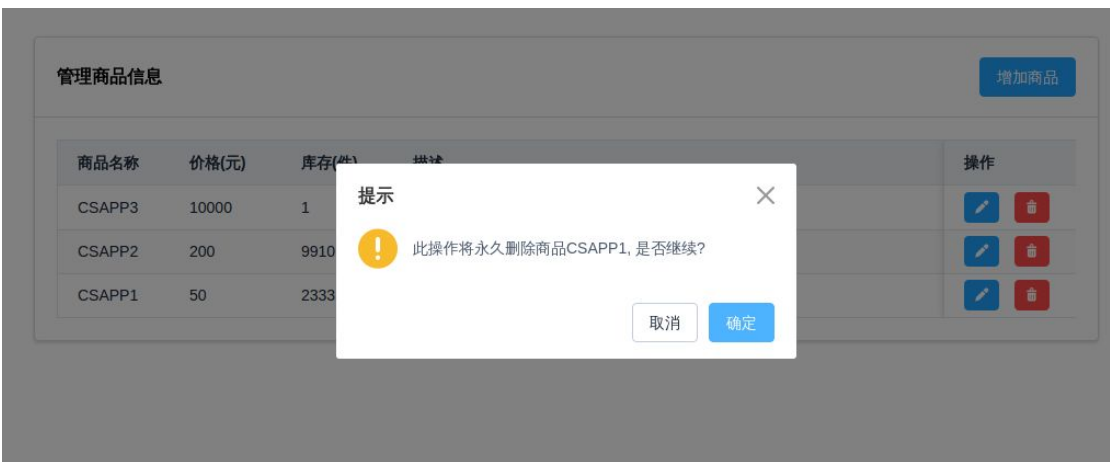
这里展示添加（也有表单验证，不再赘述）



管理商品信息 增加商品

商品名称	价格(元)	库存(件)	描述	操作
CSAPP3	10000	1	23333	编辑 删除
CSAPP2	200	9910	23333333	编辑 删除
CSAPP1	50	23333	233333333	编辑 删除

删除



然后可以退出登录

★图书商城

退出系统

管理商品信息

增加商品

商品名称	价格(元)	库存(件)	描述	操作
CSAPP3	10000	1	23333	<div><div></div><div></div></div>
CSAPP2	200	9910	23333333	<div><div></div><div></div></div>


使用普通人账号登录后调到产品列表页面

★图书商城

产品列表我的订单

产品列表

购物车 ¥ 0




商品名称：CSAPP3

价格(元)：10000

库存(件)：1

详细描述：23333

购买数量：0



商品名称：CSAPP2

价格(元)：200

库存(件)：9910

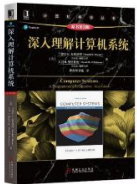
详细描述：23333333

购买数量：0

购物车价格会随着点击变化

产品列表

购物车 ¥ 1200




商品名称：CSAPP3

价格(元)：10000

库存(件)：1

详细描述：23333

购买数量：0



商品名称：CSAPP2

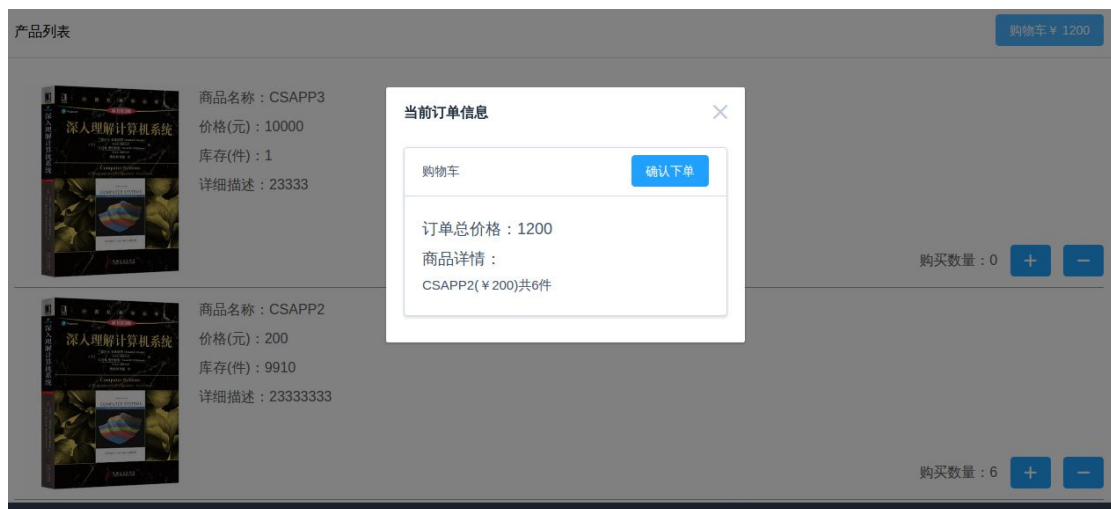
价格(元)：200

库存(件)：9910

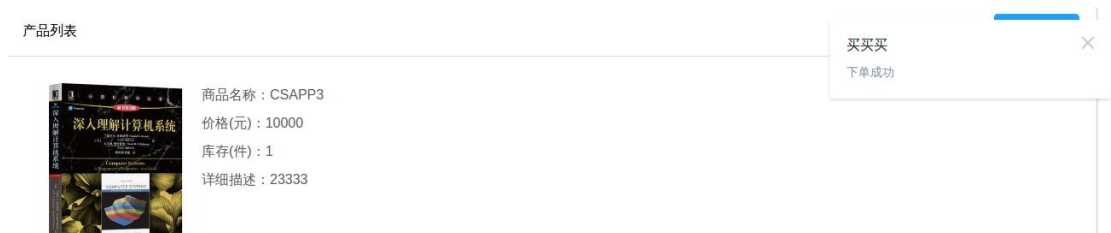
详细描述：23333333

购买数量：6

点击购物车出现订单统计



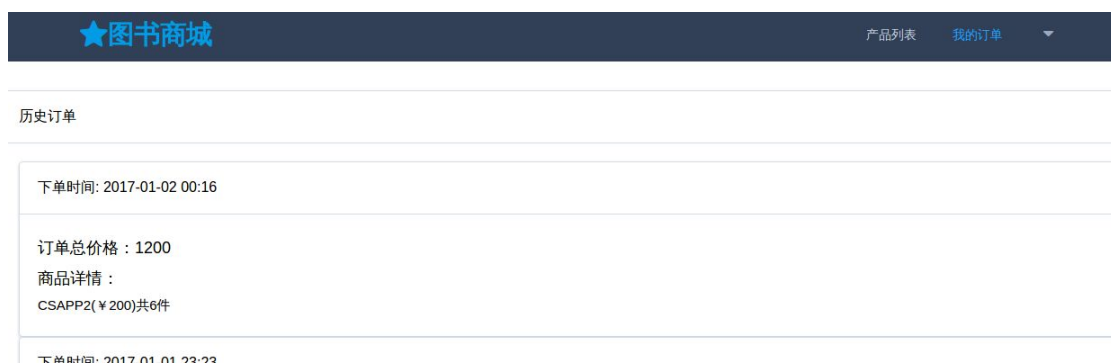
然后可以确认下单



这里的余额会发生变化



点击我的订单可以看到刚刚的历史订单



6. 实验总结

验证码忘记做了，想起来的时候要改已经很麻烦了。

Vue 真好用。

<p>指导教师批阅意见：</p>	
<p>成绩评定：</p>	
<p>指导教师签字：</p>	
<p>年 月 日</p>	
<p>备注：</p>	

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后 10 日内。