第三次迭代报告

舒步清 5130379018

docker镜像

dockerhub: MonkeyDAsce/reins-homework:3
部署后根据浏览器输入 https://ip:8443/bookstore 即可访问

> 网页运行说明可以见下方【业务功能说明】。如果出现不能跑的情况，可以联系手机13567409937/qq405752182，我也可以到实验直接演示。助教第二次迭代批阅时候说我有一个bug，但是我自己找不到是什么错。。。自己跑了几遍没发现，就没修正。

开发环境

- 工具： IDEA 2016.01
- 系统： Ubuntu 15.04 x64
- 建议浏览器： chrome，vivaldi，firefox
- 容器： jboss EAP6.4
- 数据库: mysql 5.6.281
- 缓存服务：redis 3.2.0
- jdk： 1.7

主要文件结构说明

ejb3/*

ejb模块类，后端逻辑，数据库访问，日志全在这

- src/business/*
  逻辑业务bean
- src/aspectj/AutoLog.aj
  利用aspectj对business内的所有业务bean设置切点做日志
- src/cache/CacheManager.java
  缓存交互的无状态bean，就是一个reids的client
- src/data/DataManager.java
  数据库交互bean，DAO的作用
- src/entity
  数据实体类

enviroment/*

- modules/* 需要往EAP里面放的mysql数据库
- .keystore 我的SSL证书
- databaseFile-iter2.sql sql文件
- standalone.xml： jboss配置文件

lib/*

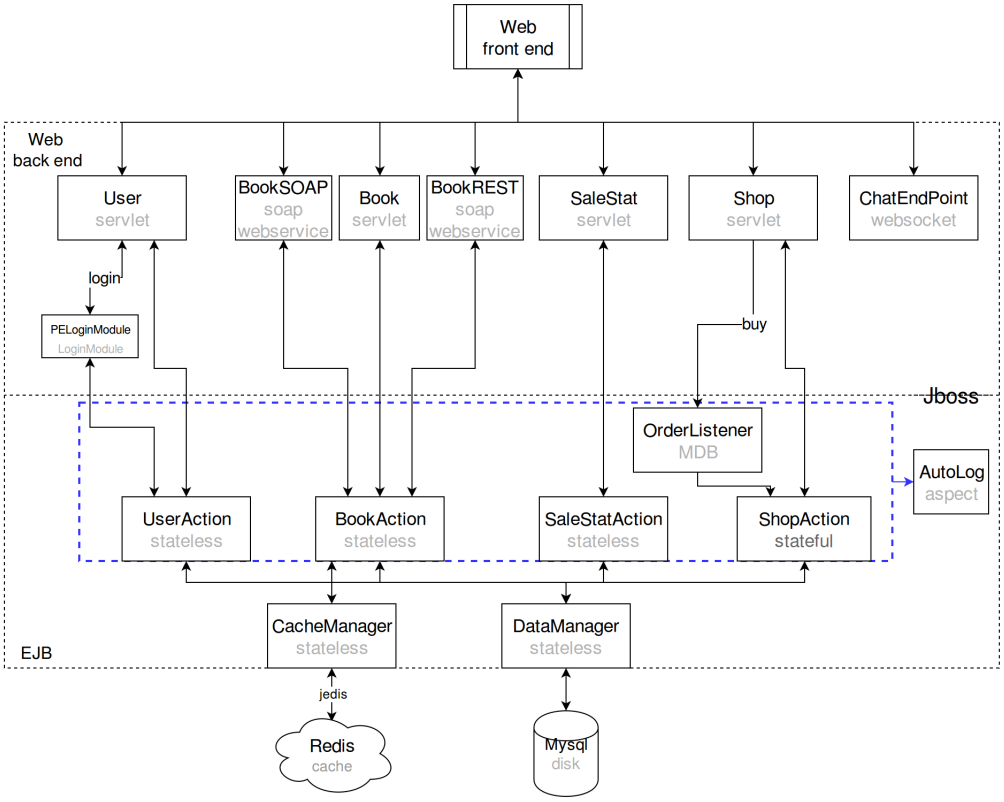
添加的json-lib，jboss-websocket，javax.ws.rs,aspectj等外部包

WebClient/*

直接与网页前端交互的部分

- src/main/java/servlet/*
  负责接受请求，然后针对请求分别调用相应的EJB，并返回结果到网页
- src/main/java/webSocket/*
  负责聊天室后端
- src/main/java/service/*
  负责soap和rest服务
- src/main/java/security/*
  负责loginmodule安全验证，角色分配
- web/*
  网页前端相关代码

系统模块交互示意图



模块说明

一些辅助类不在此说明

- Web Front End
  前端页面，通过get，post与后台服务器（内的servlet）交互，通过socket与websocket交互
- Web Back End
  在jboss容器
  - 网络接口
    - WebSocket

`ChatEndPoint`
处理聊天室相关操作

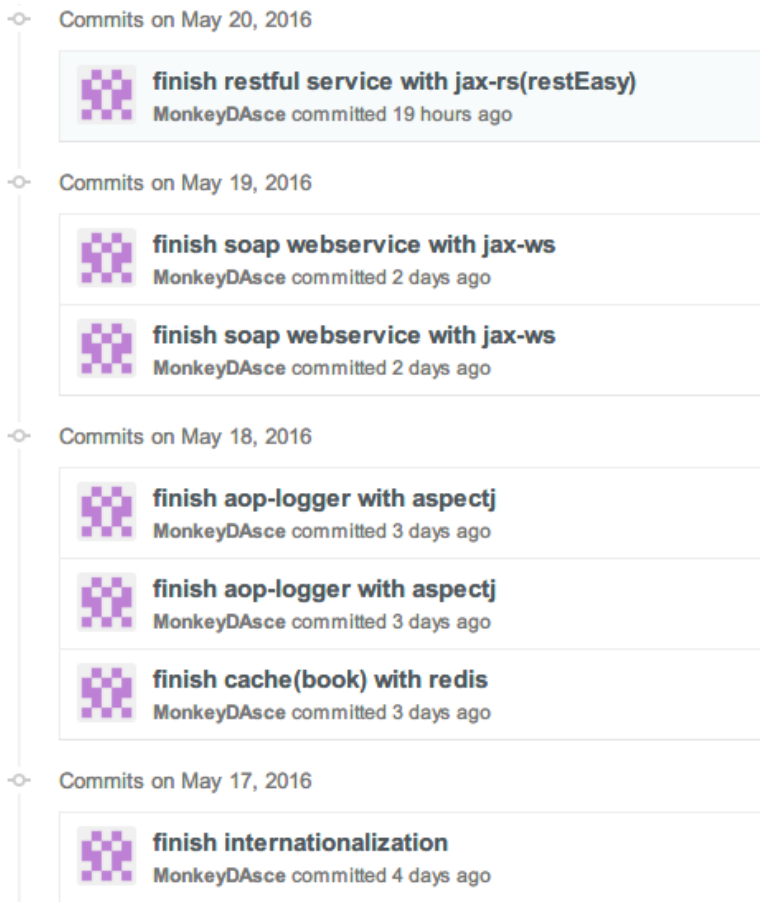
- WebService
  `BookSOAP` `BookREST`
  为获取书籍表格和详情分别提供soap和rest接口
  接受请求之后都会区调用后台BookAction执行业务逻辑
- Servlets
  `User` `Book` `SaleStat` `Shop`
  这边每个servlet都是接受一类请求，解析参数，然后分发给对应EJB做真正的业务逻辑处理，比如User就是接受、用户登陆、注册请求，然后调用对应的EJB（对于login操作会通过 `request.login` 调用loginmodule，然后loginmodule再调用UserAction的角色获取，最终分配角色）

- 内部业务-EJBs
  - 业务bean
    `UserAction` `BookAction` `SaleStatAction` `ShopAction` 分类处理业务逻辑，内部不同接口处理同一类下不同的业务
    `OrderListener` 监听购买消息，然后再调用ShopAction执行业务逻辑
  - 数据交互bean
    `DataManager` 充当一个DAO，所有的数据库交互都要通过他
    `CacheManager` 所有与缓存的交互都通过他
  - 日志
    `AutoLog` 对业务类的所有方法调用（蓝色虚线框内）都加了joint点，记录业务方法的调用参数和返回结果

- disk
  - mysql 存储数据
  - redis 提供缓存服务

各任务代码修改查看说明

为了方便批阅，我在github(https://github.com/MonkeyDAsce/SimpleJ2EEBookStore)里对每个要求commit都做了注释，对于每个小任务我更改的代码可以点击commit信息很方便查看具体修改/添加了哪些代码。如下

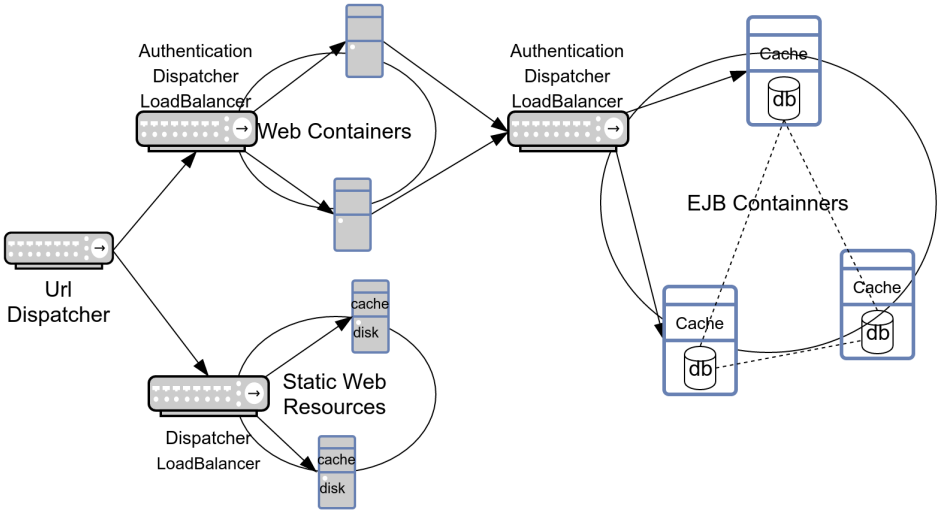


迭代三完成情况说明

这次迭代对业务功能没有进行任何修改。

- Applying MemCached or Redis in your application.
  我的书店当中没有一项数据是静态不变的，对比之下，选择了更改频率最低的书本信息（只有管理员能更改的数据），也就是对书籍信息查阅做了缓存，用户查阅书籍会先访问缓存，没有访问到的话就是去数据库找，返回信息同时把结果写入缓存，管理员一旦更改数据，就会清空缓存。缓存访问也是一个无状态bean，支持三个基本方法(set,get,clear)，隶属于ejb模块。
  与缓存的交互代码在ejb3/src/cache/cacheManager，他是一个statless bean，然后会内部通过jedis与外部redis交互。至于为什么不在业务类里面直接使用jedis，是为了以后如果要换缓存模型，我只要修改cacheManger的代码就好了，不需要修改业务逻辑的代码。
- Refactoring your application to support internationalization.
  对首页(登陆页面做了国际化s)，首页下方可以选英文还是中文。
  在java后端(webClient/src/main/java/Servlet/Language)去解析resourcebundle，前端对一些需要修改的内容其class以”inter_”开头来做标记，通过jquery_ajax获取词典做动态替换。

- Writing a design doc to describe the clustering solution you will adopt to scale up your Book Store.
    - 概述：从性能角度上看用Collocated architecture会好一些，但是考虑到这是一个交易类网站，安全问题应当是首要的，应当对每一个层次都做安全检查，这个时候就就需要做到严格分层，每个层次都是一个小集群，所以我会选择Distributed architecture。以下为我的集群示意图，如果有不好的地方，希望助教在评语中指出。



    - Url Dispatcher：
      可以用nginx根据request的url的不同分配不同的资源获取地址，如果是静态（网页）资源，直接从叫静态页面的集群获取；如果是获取动态数据资源/执行业务逻辑，再交给Web Container去做业务分发。其他的dispatcher都具有负载平衡的作用。
    - Static Web Resources
      用于获取静态网页，从磁盘返回静态网页，网页访问十分频繁，每台机器有自己的缓存。因为静态网页不设计业务，所以不需要涉及安全检查。
    - Web Containers
      解析请求内容，保存用户session状态（Disaptcher需要保证同一个用户的请求每次发到同一个server），调用指定EJB执行业务，需要有安全认证。
    - EJB Containers
      执行具体的业务或者数据获取，无状态业务为主，需要有安全认证，能和数据库交互，对频繁访问的数据设有缓存。几个相同业务的之间的数据库要保证一致性。
- Developing an Aspect to implement logging.

监视所有业务类(ejb/bussiness内)的方法调用，在日志内记录每个方法调用时候的参数和返回值：

```
1.
2.  public aspect AutoLog
3.  {
4.      private static final Logger logger = Logger.getLogger(AutoLog.class.getName()
    );
5.      pointcut businessCall(): execution(* business..*(..) );
6.      before(): businessCall(){
7.                  logger.info("enter: " + getClassName(thisJoinPoint) + getParaInfo
    (thisJoinPoint));
8.              }
9.
10.     after() returning(Object o): businessCall(){
11.                 logger.info("return: " + getClassName(thisJoinPoint) + ": " + (o
    == null ? "void" : o.toString()));
12.             }
13.    ....
14. }
```

以为没有设置Logger的输出路径，默认输出在server的log里，输出格式如下

```
1.  ...
2.  02:42:30,611 INFO  [aspectj.AutoLog] (http-/0.0.0.0:8443-113) enter: business.Use
    rActionBean.table()
3.  02:42:30,616 INFO  [aspectj.AutoLog] (http-/0.0.0.0:8443-113) return: business.Us
    erActionBean.table: [{"name":"admin","userId":1},{"name":"user1","userId":2},{"na
    me":"user2","userId":3},{"name":"user3","userId":4},{"name":"user4","userId":5},{
    "name":"user5","userId":6},{"name":"usermogo","userId":8},{"name":"user10","userI
    d":9},{"name":"guest","userId":10}]
4.  ...
```

- Respectively developing a SOAP and a REST web service for a query.
  实现对书籍的查询操作的两种服务，并且在前端的购书页面js中有相应的调用代码。
  - SOAP
    - 服务创建： 代码见webClient/src/main/java/service/BookSOAP.java，提供了获取书籍表和书籍详情的服务。
    - 服务调用：在网页中利用js建立一个简易的客户端用作测试，购物页面中的获取书本列表的方式将替换为通过访问soap webservice获取。（其中访问webservice借助了一个jquery.soap.js开源库，其github:https://github.com/doedje/jquery.soap）
  - RESTful
    - 服务创建：代码见webClient/src/main/java/service/BookREST.java，同样提供了获取书籍表和书籍详情的服务。
    - 服务调用：在网页中通过ajax对目标RESTurl进行访问，购物页面中的获取书籍详情的方法将替换为访问REST获取。

迭代二完成情况说明

- Applying transaction control on order processing
  为容器托管的模式 `@TransactionManagement(TransactionManagementType.CONTAINER)` ，在订单操作在本项目体现在ejb中的shopAciotn中，方法类型均为required，出错就调用setRollbackOnly。其他ejb因为都是单一操作，对数据库都只有一次访问，作业里也没有要

求，就没写了。

```
@TransactionManagement(TransactionManagementType.CONTAINER)
@Stateless(name = "ShopActionEJB")
public class ShopActionBean implements ShopAction
{
   //...
   @Resource
   private SessionContext context;

   //..
   @TransactionAttribute(TransactionAttributeType.REQUIRED)
   public boolean buy(String cartData, int userId)
   public boolean buy(String cartData, String username)
   public String getOrder(String username)
   public boolean delOrder(int orderId)
}
```

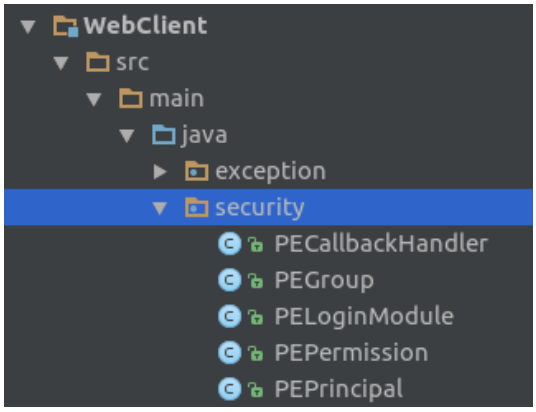
- Utilizing JMS(MDB) to process orders.
  对购物车内多本书的批量购买这个操作利用MDB交互，因为这个操作在本系统中占用时间较长，但是用户又不需要马上得到反馈，只有等到用户点击订单之后才会显示自己的订单，所以就把购买当做异步；像其余的删除订单，查看订单都是点击之后界面马上做出变化，同步要求较高，就不做消息驱动了。调用流程是 broswer(发送请求)->servlet（分发消息）->messageListener（监听消息）->ejb（真正处理）
- Utilizing Ajax to implement book browsing page. When user clicks on a book, the details of this book will be retrieved from server with Ajax and displayed on this page.
  除了书本详情之外，关于用户登陆，登出，其余数据加载也都是通过ajax实现异步交互的。
- Utilizing WebSocket to implement an on-line chat room for customer group.
  进入shop.jsp后点击charroom可以进入了聊天室，我设置了cookie保存用户名，所以多人聊天室操作请使用多个浏览器访问。聊天界面如下，右边是聊天记录，所有新消息都会置顶，左边将会实时显示当前聊天室内的人和名称，有人退出页面或者切换页面都将触发这个函数，上面是消息输入栏。

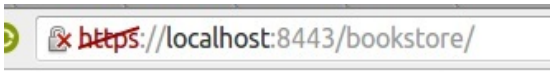
- JSON processing in Browser and Server. In the ajax-based book list page, the book details should be transferred as JSONs.
  里面的用户、书本、订单都是通过json动态填充页面。

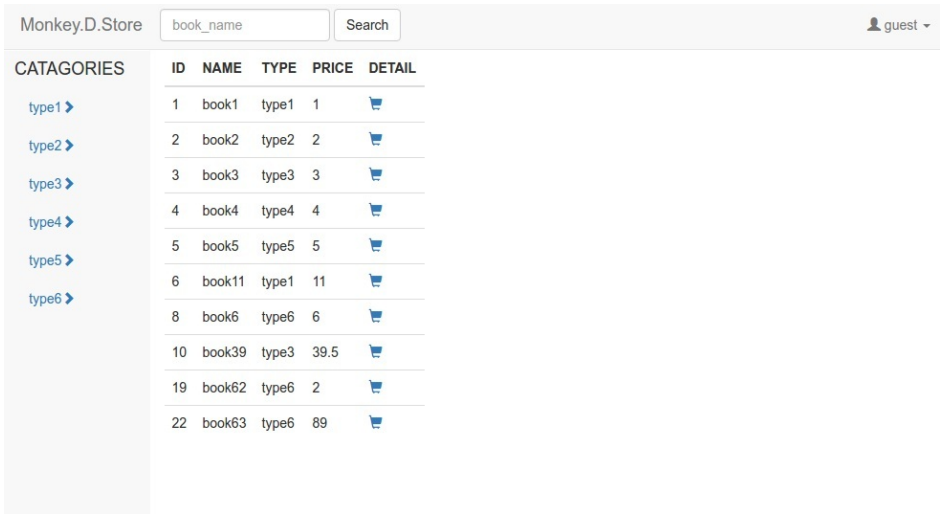
迭代一完成情况说明

- To build your business logics with:
  - At least one stateful session bean, such as shopping cart
  - At least one stateless session bean, such as login

目前将 `shopAction` 定义为有状态的，其余EJB均为无状态，整体架构可详见之前的架构图。

- To design your security system of your project
  - Develop the SimpleLoginModule in Courseware V, and use it as the authentication method in your project
    参见目录 `WebCLient/src/main/java/security/*` ，如下，里面有关于的loginmodule的定义、角色、权限、callbackhandler
    

  - Develop the billing function in a security way (Encrypt financial information)
    全局ssl登陆，http端口被设置为拒绝访问，只用使用https和端口8443访问,使用时浏览器设置信任才能访问
    

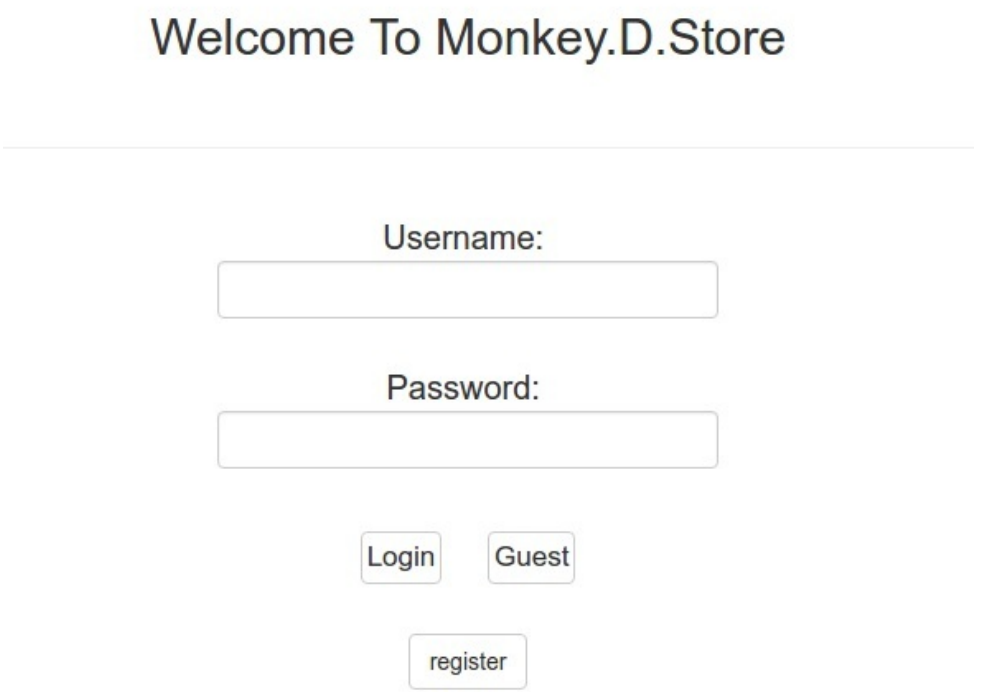
  - Develop a customized permission to view categories of books. The categories are modeled as a tree. You can grant permission to codebase(or principal) to describe which code(or who) can view which categories of book.
    根据角色选择性的展示书本类型数据，guest用户无法看到vip书籍，且没有购买，进入聊天室，管理用户等等权限，只有查看普通书本（详情），搜索的功能，如下图，正常用户的界面和功能可参见【业务功能说明-购物界面】
    

  - Add security constraints in your web.xml in order to constrain only Manager can browse profiles of users.
    一共是三个角色（user，guest），在web.xml中添加了管理页面的权限限制，只有管理员(账号密码都是admin/admin)可以进入manage.jsp页面管理用户、浏览数据等，只有user能进入聊天室

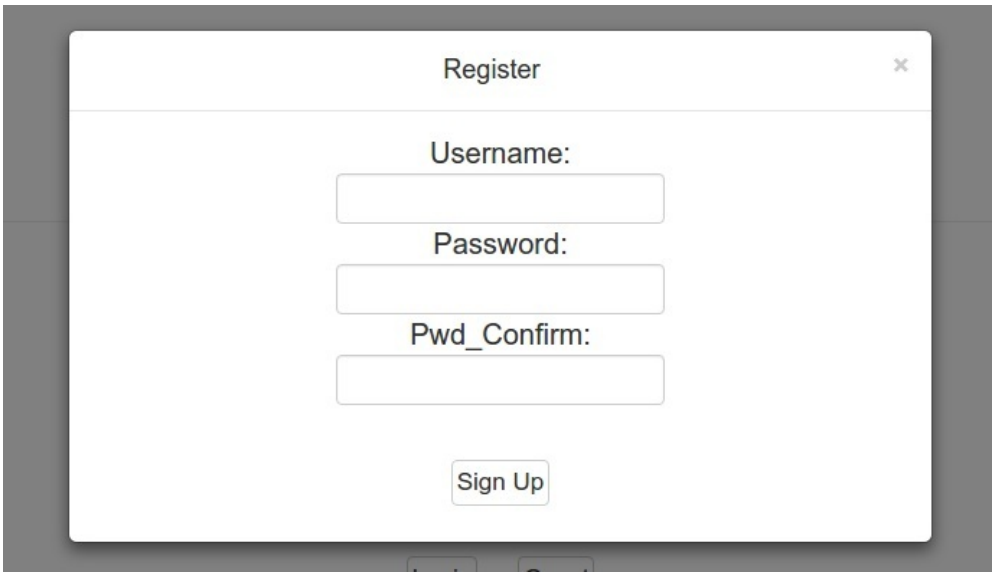
业务功能说明

index.jsp 初始界面

- 三个按钮分别是支持登陆、注册、游客身份浏览。可以用预设用户user1/user1登陆，也可以自己注册。

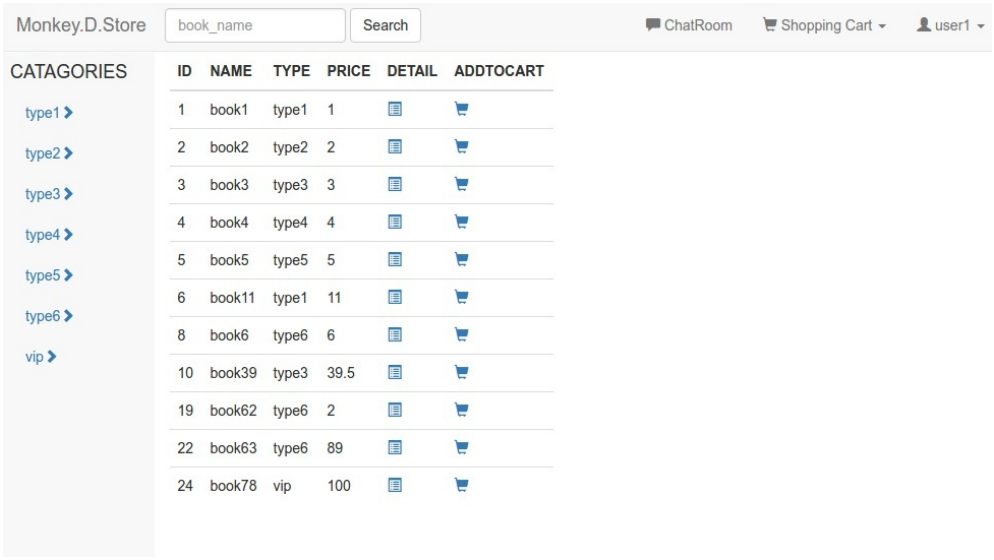


- 注册 ～弹出框的形式注册

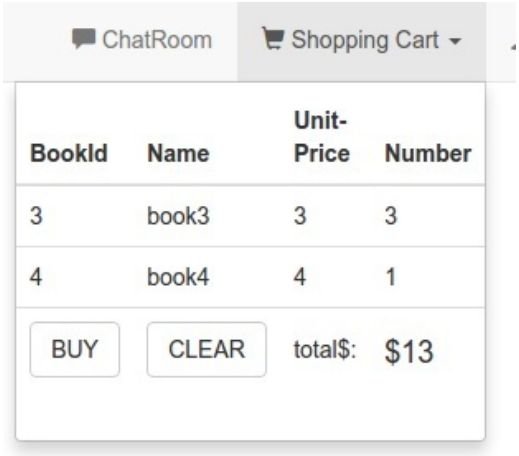


shop.jsp 购物界面

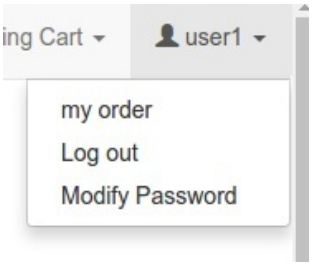
- 浏览书籍，支持按书名关键词搜索，点击右边类型按类型筛选

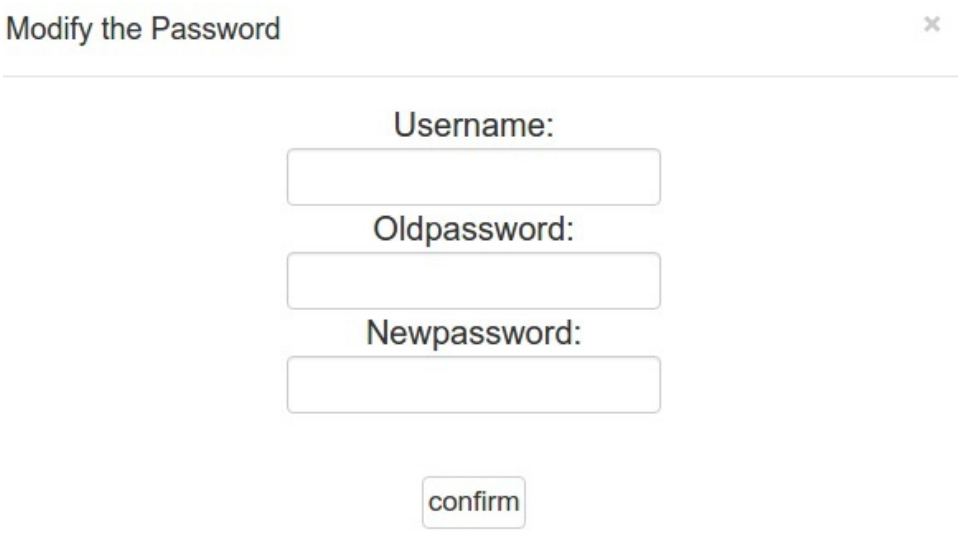


- 右上角为功能区，点击chatRoom进入聊天室，点击Shopping Cart 查看购物车，点击user1可以做用户信息相关操作

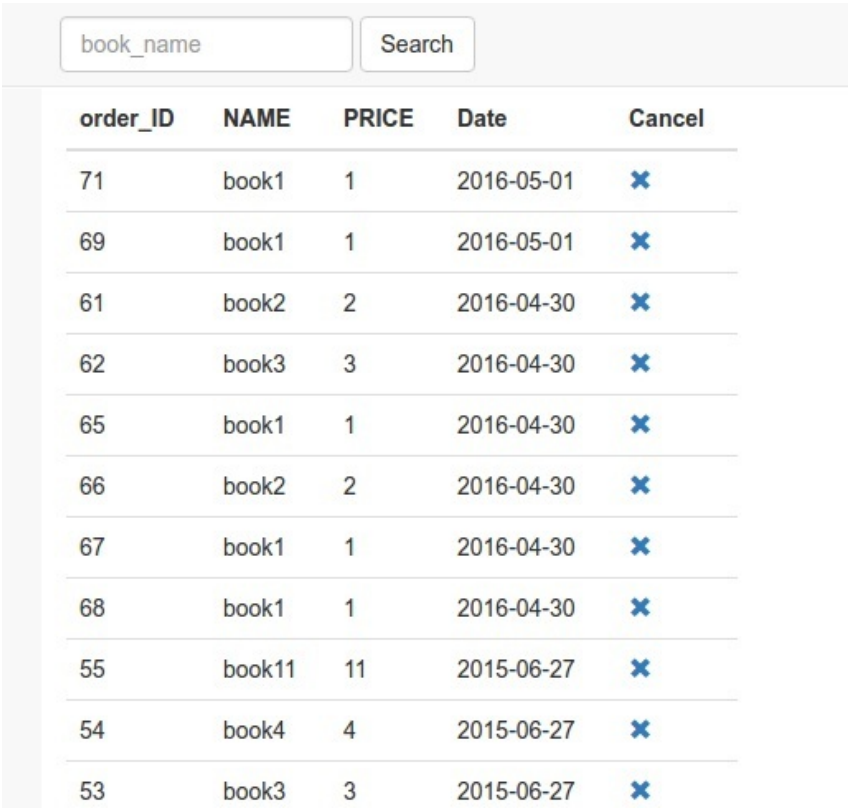


- 点击右上角用户可注销、修改密码、查看订单





- 订单查看，点击search可回到原来书本浏览模式，右边的叉是取消订单按钮

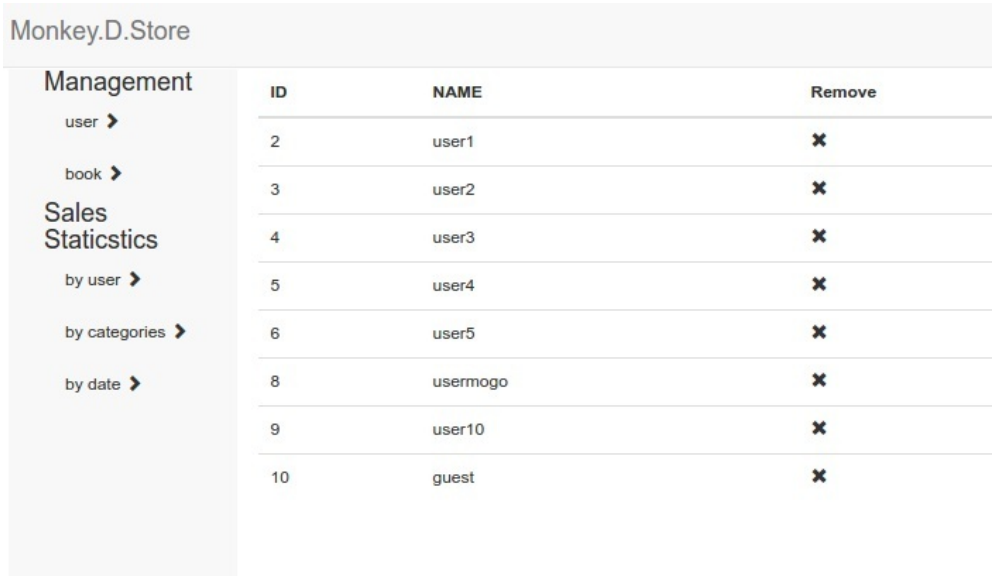


chatRoom.html 聊天窗口

具体说明可见迭代二聊天窗口任务完成情况。

manage.jsp 管理页面，可用admin/admin登陆到该页面

- 管理用户， 又是删除用户，点击左边不同标签可以选择管理/统计不同的内同

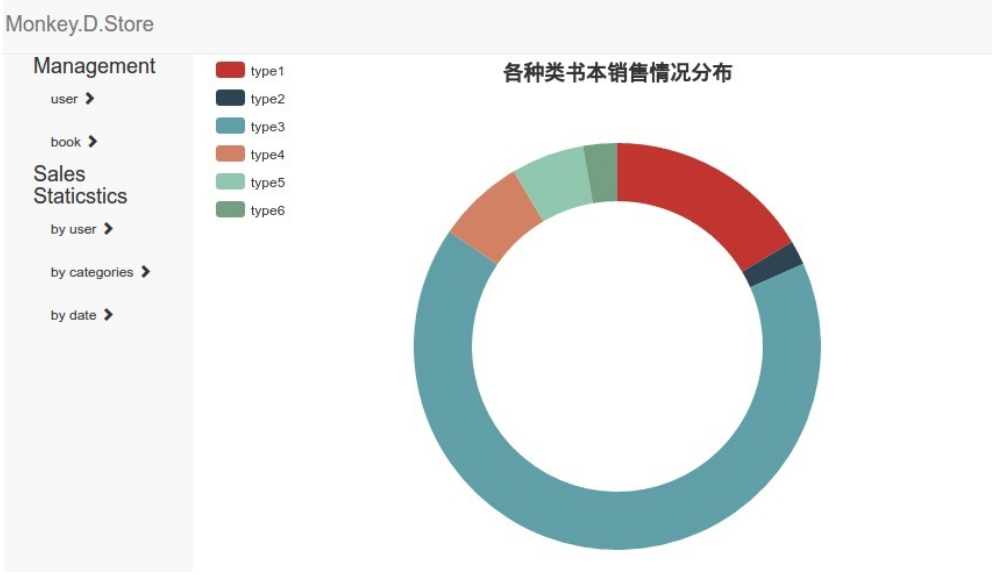


- 管理书籍，增加/删除/修改

BOOK ID	NAME	TYPE	PRICE	REMOVE
1	book1	type1	1	✖
2	book2	type2	2	✖
3	book3	type3	3	✖
4	book4	type4	4	✖
5	book5	type5	5	✖
6	book11	type1	11	✖
8	book6	type6	6	✖
10	book39	type3	39.5	✖
19	book62	type6	2	✖
22	book63	type6	89	✖
24	book78	vip	100	✖

ADD/MODIFY BOOK

- 数据统计 - （echarts展示图表）



- 日期销量曲线

参考链接

- redis：http://www.runoob.com/redis/redis-java.html
- 国际化：http://zhwj184.iteye.com/blog/1872380
- JPA entitymanager注入方法：http://www.blogjava.net/jesson2005/articles/380880.html
- EJB3 事务管理 http://fansofjava.iteye.com/blog/350119
- 消息驱动MDB http://blog.csdn.net/xiaoping8411/article/details/6457805， 要切换到standlone-full.xml，同时要把自己以前的sercurity-domain，ssl，datasource配置代码拷贝过来。新的配置standalone.xml过程：http://theopentutorials.com/examples/java-ee/ejb3/remote-jms-client-and-ejb3-mdb-consumer-eclipse-jboss-7-1/
- jboss websocket: https://access.redhat.com/documentation/en-US/JBoss_Enterprise_Application_Platform/6.4/html/Development_Guide/Create_a_Websocket_Application.html
- jboss 官方文档（主要！）：https://access.redhat.com/documentation/en-US/JBoss_Enterprise_Application_Platform/6.4/html/Development_Guide/chap-Get_Started_Developing_Applications.html