一、OrderManager电子商城 --- 知识的整合,加入一些真实开发中的思想 面向切面编程 利用注解加本地线程的方式实现事务管理

用户注册(发送激活邮件/JS前台实现数据校验/验证码)

用户激活

用户登录(记住用户名/30天内自动登陆)

用户注销

添加商品(文件上传)

查看商品列表

查看商品详情

加入购物车(Cookie \*session 数据库)

增删改查购物车

生成订单(多表设计)

订单查询(多表查询)

订单删除(事务管理/注解+本地线程+动态代理实现事务管理,AOP--面向切面编程)

在线支付(体验调用第三方接口实现特定功能)

销售榜单下载(利用程序生成excel数据/文件下载)

权限过滤器

全站乱码过滤器

用户:id 用户名 密码 昵称 邮箱 激活状态 激活码 角色 注册时间

商品:id 商品名称 商品种类 商品库存数量 商品单价 图片url 描述信息

订单:id 下单时间 收货地址 支付状态 订单金额 用户编号(外键)

订单项: 订单id 商品id 购买数量

用户 1 -- \* 订单

商品 \* -- \*订单

create database OrderManager;

用户：

create user OrderManager identified by 'OrderManager';

授权：

grant all on OrderManager.\* to OrderManager;

use OrderManager;

create table users (

id int primary key auto\_increment,

username varchar(100),

password varchar(100),

telnum varchar(40),

role\_id int, #0---管理员，1-----快递员，2------普通用户

foreign key(role\_id) references roles(id)

);

create table roles (

id int primary key,

roleName varchar(20)

);

create table function (

id int primary key,

funcName varchar(100),

path varchar(255)

);

create table privilege (

role\_id int,

func\_id int,

primary key(role\_id, func\_id),

foreign key(role\_id) references roles(id),

foreign key(func\_id) references function(id)

);

create table products(

id varchar(100) primary key ,

name varchar(40),

imgurl varchar(100),

price double,

pnum int ,

startTime date,

endTime date

);

create table orders(

id varchar(100) primary key,

money double,

address varchar(255),

payState int,

orderTime timestamp,

sendTime timestamp,

signTime timestamp,

admin\_id int,

courier\_id int,

user\_id int ,

foreign key(user\_id) references users(id)

);

create table orderitem(

order\_id varchar(100),

product\_id varchar(100),

buynum int ,

primary key(order\_id,product\_id), #联合主键,两列的值加在一起作为这张表的主键使用

foreign key(order\_id) references orders(id),

foreign key(product\_id) references products(id)

);

javaee经典三层架构+工厂类实现解耦

包结构:

com.itheima.web

.service

.dao

.util

.domian

.factory

.exception

.test

.filter

.listener

导入第三方jar包

\*junit

\*JSTL

beanutils

mysql驱动

c3p0

dbutils

commns-fileupload

配置文件

c3p0-config.xml

config.properties

配置虚拟主机:

在tomcat/conf/server.xml中配置

<Host name="www.OrderManager.com" appBase="E:\resource\resource\20140105\OrderManager">

<Context path="" docBase="E:\resource\resource\20140105\OrderManager\WebRoot" />

</Host>

在Hosts文件中配置

127.0.0.1 www.OrderManager.com

前置开发:

工厂类实现解耦:

全站乱码解决

~权限控制

javabean

工具类

1.用户注册

index.jsp -- 如果用户没有登录,则提示 欢迎光临游客 注册 登录 如果用户已经登录 提示 欢迎回来xxx,注销

regist.jsp -- 提供注册表单 (验证码/js实现表单校验)

RegistServlet -- 校验验证码 封装数据校验数据 调用Service注册用户 重定向到主页

UserService -- 注册用户 检验用户名是否已经存在 如果存在则提示 如果不存在则调用dao中添加用户的方法添加用户 发送激活邮件

UserDao -- 添加客户方法

2.用户激活:用户点击邮箱中的激活链接时触发此Servlet

ActiveServlet: 获取请求参数中的激活码,调用Service中激活用户的方法,回到主页

UserSerivce: 激活用户 调用dao根据激活码查找用户 如果找不到用户提示激活码无效 如果用户已经激活过,则提示不要重复激活 检查如果激活码已经超时(24小时)则提示激活码超时要求重新注册此时应该删除此用户记录

UserDao: 根据激活码查找用户 根据用户id删除用户的方法

3.用户登录(记住用户名 30天内自动登陆)

index.jsp -- 如果用户没有登录,则提示 欢迎光临游客 注册 登录 如果用户已经登录 提示 欢迎回来xxx,注销

login.jsp -- 提供用户登录表单输入用户名密码

LoginServlet -- 1.获取用户名密码,2.调用Service中根据用户名密码查找用户的方法 3.检查用户的激活状态,如果没有激活则提示 4.登录用户重定向到主页

UserService -- 提供根据用户名 密码查找用户的方法 调用dao中对应方法

UserDao -- 提供根据用户名密码查找用户的方法

4.用户注销

5.添加商品(文件上传)

index.jsp -- 提供 添加商品

addProd.jsp -- 提供添加商品的表单,这个表单应该是文件上传的表单,其中允许上传商品图片

AddProdServlet -- 实现文件上传,将商品的图片上传到服务器中.并且向数据库的商品表中增加一条记录

Service -- 添加商品

Dao -- 添加商品信息的方法

6.商品列表

index.jsp -- 提供商品列表

ProdListServlet -- 调用Service中查询所有商品的方法,查到后存入request域带到页面展示

prodList.jsp -- 从request域中拿出所有的商品做展示

7.查看商品详情

在商品列表页面中点击图片时,查看商品的详情

ProdInfoServlet 根据商品id查询商品信息,带到页面显示

ProdService 提供根据id查询商品的方法

ProdDao 提供根据id查询商品的方法

prodInfo.jsp页面展示

8.加入购物车

在商品详细信息页面中点击加入购物车,将商品加入购物车

设置监听器,在Session创建时,就将cartmap加入到session中

AddOrderServlet -- 根据id查找商品,存入购物车,如果购物车中还不存在这个商品,则存入,数量为1,如果已经存在则在原有数量上+1

cart.jsp -- 遍历购物车map,遍历cartmap显示当前用户所有的购物车信息

9.删除购物车

在购物车页面,中点击删除时,触发

DelCartServlet,根据id找到要删除的商品后,从购物车map中删除

10.修改购买数量

在购物车页面,修改购物数量时触发,利用js控制输入的数字必须正整数

ChangeCartServlet,根据id找到要删除的商品后,修改购物车中商品的数量

11.清空购物车

找到购物车map,清空map

12.生成订单

在购物车中,当购物完成后,用户点击 生成订单 生成订单

访问一个addOrder.jsp -- 列出订单的基本信息,要求用户输入收货地址和支付方式

AddOrderServlet -- 创建Order对象设置基本值,其中Money需要在后台根据购物车实时计算出来 调用OrderService中生成订单的方法生成订单, 清空购物车 回到主页

OrderService -- 中生成订单的方法 在订单表中插入一条记录 在订单项表中插入记录保存此订单和商品之间的关系 从商品表中的库存数量中扣除购买数量 需要进行事务管理

OrderDao -- 增加订单的方法 增加订单项的方法

ProductDao -- 增加扣除商品数量的方法

面向切面编程 (AOP)

~注解:

注释--给人看的提示信息就叫做注释 // /\*\*/ /\*\* \*/

注解--给程序看的提示信息就叫做注解 @xxxxx(...)

java中提供的原生的注解:

@Override: 限定重写父类方法, 该注解只能用于方法

@Deprecated: 用于表示某个程序元素(类, 方法等)已过时

@SuppressWarnings: 抑制编译器警告.

自定义注解

@interface 定义一个注解

定义出来的注解可以被元注解修饰,确定其基本的特性:

@Retention: 只能用于修饰一个 Annotation 定义, 用于指定该 Annotation 可以保留的域

RetentionPolicy.SOURCE: 编译器直接丢弃这种策略的注释

RetentionPolicy.CLASS: 编译器将把注解记录在 class 文件中. 当运行 Java 程序时, JVM 不会保留注解. 这是默认值

RetentionPolicy.RUNTIME:编译器将把注释记录在 class 文件中. 当运行 Java 程序时, JVM 会保留注解. 程序可以通过反射获取该注释

@Target:指定注解用于修饰类的哪个成员

@Documented: 用于指定被该元 Annotation 修饰的 Annotation 类将被 javadoc 工具提取成文档.

@Inherited: 被它修饰的 Annotation 将具有继承性.如果某个类使用了被 @Inherited 修饰的 Annotation, 则其子类将自动具有该注解

注解中还可以包含属性:

属性的定义类似于在接口中定义一个方法

String name();

String addr() default "xxx";

如果注解中只有一个属性并且名字为value,则在定义注解时可以直接写值而省略value= 部分

反射注解:通过反射注解,来确定某个类方法属性上是否有注解从而控制程序的流转.

--希望在Service和dao中都不用操心事务,是否管理事务只需要在Service接口中的方法上写或不写 @Tran就可以控制了

ThreadLocal + 注解 实现事务管理

工厂类+动态代理 实现面向切面编程

利用动态代理,使dao中不需要区分是否开启过事务

13.订单查询

在主页中 提供 订单查询

OrderListServlet 获取当前客户的用户号,调用Service中查询指定用户订单的方法,查询出这个用户所有订单,存入request域,带到页面显示

OrderService 提供根据用户id查询订单的方法

OrderDao 提供根据用户id查询订单的方法

orderList.jsp 展示所有订单

订单号 ok

用户名称 --- 用户表中查

订单金额 ok

支付状态 ok

收货地址 ok

下单时间 ok

------------------------------------------------------

商品名称 购买数量 单价 总金额 --------------------------------订单项 表 商品表

------------------------------------------------------

海尔冰箱 1 999 999

美的笔记本 1 888 888

....

...

------------------------------------------------------

14.订单删除

在订单列表 对于未支付的订单 提供 订单删除

DelOrderServlet 获取订单id 调用service中根据订单id删除订单的方法,删除后再重新回到订单列表页面

OrderService 提供根据订单id删除订单的方法 将订单相关商品的库存数量加回去 删除订单项表中的记录 删除订单记录 -- 事务控制

OrderDao中提供根据订单编号删除订单项和 订单的方法

ProdDao中提供修改库存数量的方法

15.在线支付(在真正的开发中,一些功能可能不是我们自己开发的,而是调用第三方接口来实现)

在订单列表页面 对于未支付的订单 提供 在线支付

pay.jsp 页面中选择要支付的银行

PayServlet 准备访问易宝所需要的所有的参数,将这些参数保存到request域中,再转发到confirm.jsp页面

confirm.jsp 将request域中的所有访问易宝的参数取出,组织表单 提交到易宝

CallBackServlet处理支付完成后的回调

16.销售榜单下载:管理员可能需要查看商品销售的数据-- 多表查询 csv格式的文件 文件下载

index.jsp中提供销售榜单下载连接

SaleListServlet 调用Service中查询销售榜单的方法 将查询到的销售榜单组织成csv格式的文件后提供下载

OrderService 查询销售榜单

OrderDao 查询销售榜单:查询所有已支付订单找对应的订单项信息找到销售出去的商品,做一个销售数量的sum()操作

select products.id prod\_id,products.name prod\_name,sum(orderitem.buynum ) sale\_num

from orders ,orderitem ,products

where

orders.id=orderitem.order\_id

and

orderitem.product\_id=products.id

and

orders.paystate = 1

group by products.id

order by sale\_num desc;

商品编号 商品名称 总销售数量

1 xxx 100

2 yyy 10

17.权限控制