中间件实现技术开源代码研究报告

学院：信息学院

专业：软件工程

姓名：祝佩文

学号：2017104143

目录

**[1、课题背景](#_Toc14634_WPSOffice_Level1)** **[1](#_Toc14634_WPSOffice_Level1)**

**[2、研究目的](#_Toc25504_WPSOffice_Level1)** **[1](#_Toc25504_WPSOffice_Level1)**

**[3、项目环境](#_Toc7885_WPSOffice_Level1)** **[1](#_Toc7885_WPSOffice_Level1)**

**[4、源码分析](#_Toc30173_WPSOffice_Level1)** **[2](#_Toc30173_WPSOffice_Level1)**

[4.1项目简介](#_Toc25504_WPSOffice_Level2) [2](#_Toc25504_WPSOffice_Level2)

[4.2源码分析](#_Toc7885_WPSOffice_Level2) [2](#_Toc7885_WPSOffice_Level2)

**[5、 总结与思考](#_Toc3484_WPSOffice_Level1)** **[10](#_Toc3484_WPSOffice_Level1)**

# 1、课题背景

中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源。中间件位于客户机/ 服务器的操作系统之上，管理计算机资源和网络通讯。是连接两个独立应用程序或独立系统的软件。相连接的系统，即使它们具有不同的接口，但通过中间件相互之间仍能交换信息。执行中间件的一个关键途径是信息传递。通过中间件，应用程序可以工作于多平台或OS环境。

# 2、研究目的

通过对中间件课程的学习，能够对中间件的原理、实现以及定位有一定的了解，在此基础上，已部署并运行开源项目的形式，对所学知识加深理解。从开源平台获取开源项目，搭建运行项目所需要的环境，成功运行项目后，对项目源码加以分析，进一步学习中间件相关知识。

# 3、项目环境

JDK：1.8

OS：windows10

DataBase：MySQL

编译器：eclipse

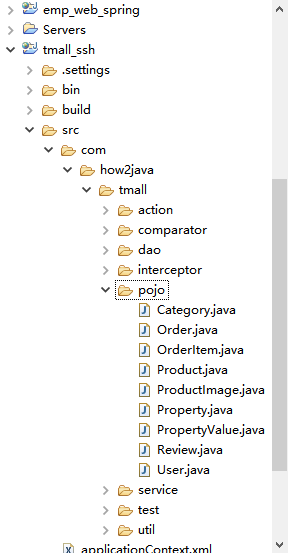
# 4、源码分析

## 4.1项目简介

本项目是从开源社区上下载的SSH项目，其模拟业务场景为天猫网上商城。分为用户端和管理端，用户可在用户端浏览商品，选中商品加入购物车，并下单，管理员端可对货物进行补给，能够监控用户提交的下单请求，能够监控商品数量等。

下载项目源码后，完善其数据库结构，模拟商品数据，调试项目，搭建项目运行需要的环境，成功运行项目后，以调试模式监控项目运行。

## 4.2源码分析



后端pojo类：后端有9个pojo类，对应数据库的9张数据表

@Entity

@Table(name = "user")

**public** **class** User {

@Id

@GeneratedValue(strategy = GenerationType.***IDENTITY***)

@Column(name = "id")

**private** **int** id;

**private** String password;

**private** String name;

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

}

**public** String getPassword() {

**return** password;

}

**public** **void** setPassword(String password) {

**this**.password = password;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getAnonymousName(){

**if**(**null**==name)

**return** **null**;

**if**(name.length()<=1)

**return** "\*";

**if**(name.length()==2)

**return** name.substring(0,1) +"\*";

**char**[] cs =name.toCharArray();

**for** (**int** i = 1; i < cs.length-1; i++) {

cs[i]='\*';

}

**return** **new** String(cs);

}

}

数据表内容：

CREATE TABLE user (

id int(11) NOT NULL AUTO\_INCREMENT,

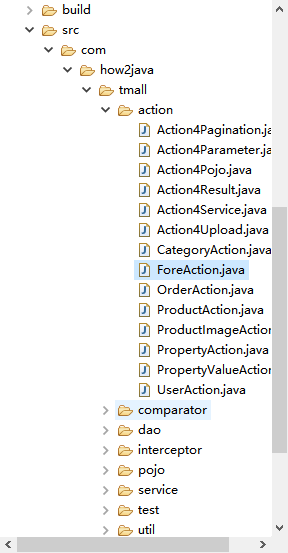
name varchar(255) DEFAULT NULL,

password varchar(255) DEFAULT NULL,

PRIMARY KEY (id)

)

Pojo类与数据库形成对应关系，便于永久化数据的存储。



@Action("forereview")

**public** String review() {

t2p(order);

orderItemService.fill(order);

product = order.getOrderItems().get(0).getProduct();

reviews = reviewService.listByParent(product);

productService.setSaleAndReviewNumber(product);

**return** "review.jsp";

}

Action中使用@Action注解的代理方式，有Spring框架进行请求代理。

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

xmlns:tx=*"http://www.springframework.org/schema/tx"*

xmlns:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"*

*http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/aop*

*http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>

<context:annotation-config />

<context:component-scan base-package=*"com.how2java"* />

<tx:annotation-driven transaction-manager=*"transactionManager"*/>

<bean name=*"sf"*

class=*"org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"*>

<property name=*"dataSource"* ref=*"ds"* />

<!-- 注解扫描 -->

<property name=*"packagesToScan"*>

<list>

<value>com.how2java.\*</value>

</list>

</property>

<property name=*"schemaUpdate"*>

<value>true</value>

</property>

<property name=*"hibernateProperties"*>

<value>

hibernate.dialect=org.hibernate.dialect.MySQLDialect

hibernate.show\_sql=false

hbm2ddl.auto=update

</value>

</property>

</bean>

<bean name=*"ds"*

class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"*>

<property name=*"driverClassName"* value=*"com.mysql.jdbc.Driver"* />

<property name=*"url"* value=*"jdbc:mysql://localhost:3306/tmall\_ssh?characterEncoding=UTF-8"* />

<property name=*"username"* value=*"root"* />

<property name=*"password"* value=*"960229"* />

</bean>

<!-- 配置事务管理器（声明式的事务） -->

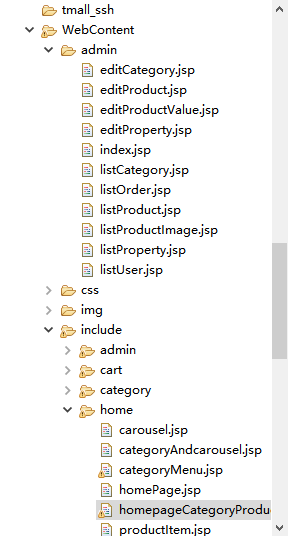
<bean id=*"transactionManager"* class=*"org.springframework.orm.hibernate3.HibernateTransactionManager"*>

<property name=*"sessionFactory"* ref=*"sf"*></property>

</bean>

</beans>

applicationContext.xmlSpring配置文件是用于指导Spring工厂进行Bean生产、依赖关系注入（装配）及Bean实例分发的“图纸”。Java EE程序员必须学会并灵活应用这份“图纸”准确地表达自己的“生产意图”。Spring配置文件是一个或多个标准的XML文档，ApplicationContext.xml是Spring的默认配置文件，当容器启动时找不到指定的配置文档时，将会尝试加载这个默认的配置文件。



<!DOCTYPE html>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"* isELIgnored=*"false"*%>

<%@ taglib prefix=*"fn"* uri=*"http://java.sun.com/jsp/jstl/functions"* %>

<c:if test="${**empty** param.categorycount}">

<c:set var=*"categorycount"* scope=*"page"* value=*"100"*/>

</c:if>

<c:if test="${!**empty** param.categorycount}">

<c:set var=*"categorycount"* scope=*"page"* value="${param.categorycount}"/>

</c:if>

<div class=*"homepageCategoryProducts"*>

<!--循环遍历-->

<c:forEach items="${categorys}" var=*"c"* varStatus=*"stc"*>

<c:if test="${stc.count<=categorycount}">

<div class=*"eachHomepageCategoryProducts"*>

<div class=*"left-mark"*></div>

<span class=*"categoryTitle"*>${c.name}</span>

<br>

<c:forEach items="${c.products}" var=*"p"* varStatus=*"st"*>

<c:if test="${st.count<=5}">

<div class=*"productItem"* >

<a href=*"foreproduct?product.id=*${p.id}*"*><img width=*"100px"* src=*"img/productSingle\_middle/*${p.firstProductImage.id}*.jpg"*></a>

<a class=*"productItemDescLink"* href=*"foreproduct?product.id=*${p.id}*"*>

<span class=*"productItemDesc"*>[热销]

${fn:substring(p.name, 0, 20)}

</span>

</a>

<span class=*"productPrice"*>

<fmt:formatNumber type=*"number"* value="${p.promotePrice}" minFractionDigits=*"2"*/>

</span>

</div>

</c:if>

</c:forEach>

<div style="clear:*both*"></div>

</div>

</c:if>

</c:forEach>

<img id=*"endpng"* class=*"endpng"* src=*"img/site/end.png"*>

<!--imgb标签-->

</div>

前段代码为JSP代码，动态数据以JSP表单形式绑定到页面，与后台交互。

# 总结与思考

中间件是位于平台(硬件和操作系统)和应用之间的通用服务，这些服务具有标准的程序接口和协议。针对不同的操作系统和硬件平台，它们可以有符合接口和协议规范的多种实现。由于本次课程研究时间较为紧张，花费了较多时间从开源社区上找合适项目，找到项目后，项目的部署又花费了较多时间，所以对项目代码的分析不够深入。

通过本次从开源社区搜索项目，并且把项目成功部署的经历，对中间件有了更宏观的认识，不再局限于某一段代码或者某一种技术方式，而是更关注各个组件之间的关联，对老师提及的WXML微信小程序开发部分也有了新的认识，三天的课程，收益良多。