目 录

[1 概述分析 1](#_Toc374177561)

[2 实现步骤 3](#_Toc374177562)

[3 数据库及框架搭建 4](#_Toc374177563)

[4 Dao层设计 5](#_Toc374177564)

[5 Service层设计 7](#_Toc374177565)

[6 Controller层设计 9](#_Toc374177567)

[7 实践心得体会 1](#_Toc374177568)0

1 概述分析

本系统是的要求是实现一个简单的内容销售系统。在此系统中，存在两种类型的用户，卖家和买家。

针对买家：买家用户可以登录系统进行内容浏览，内容购买，以及查看个人账务的操作。

针对卖家：买家用户可以登录系统对发布的内容进行编辑，发布新内容，和查看内容售出情况。

功能分析：

1. 内容展示（首页）

系统的基本界面为卖家所有内容的展示，展示的细节为内容的标题、图片和价格。

1. 用户登陆前，展示界面有登陆按钮。
2. 买家登陆后，展示界面显示用户昵称，有退出、账务两个功能按钮。
3. 买家登陆后，已购买的内容上有特殊标识表明已购买。
4. 买家登陆后，可以查看全部内容和未购买的内容。
5. 卖家登陆后，展示界面显示用户昵称，有退出、发布两个功能按钮。
6. 卖家登陆后，可以在展示界面删除未出售的内容。
7. 用户登陆

用户登陆时，需要输入用户名和密码，前端将密码用 MD5 加密后传输；登陆后，所有界面上要显示用户的昵称和退出按钮。

1. 内容查看

点击展示中的每个内容的网格，进入查看界面，可以查看内容。

1. 买家对于没有购买的内容，看到的是标题、摘要文字、当前的价格、购买按钮、全文。
2. 买家对于已经购买的内容，看到的是标题、摘要文字、购买时的价格、购买按钮不可用、全文。
3. 卖家看到的是标题、摘要文字、价格、全文、编辑按钮。
4. 内容购买

买家在查看界面点击购买按钮，弹出确认窗口，购买成功添加进入账务。

1. 账务

在展示界面点击“账务”按钮，进入账务界面；在账务界面以列表方式列出所有已购买的内容，每一项内容列出：

1. 标题；
2. 图片；
3. 购买的时间；
4. 购买时的价格；
5. 列表的最后计算出总的金额。
6. 内容发布

卖家在展示界面点击“发布”按钮，进入发布界面；在发布界面可以输入内容的标题、摘要、图片、正文和价格，点击发布按钮即发布，

1. 内容编辑

卖家在查看界面可以点击编辑按钮进入编辑界面；可以修改内容的全部细节：标题、摘要、图片、正文和价格，点击提交后回到查看界面。

非功能需求：

相关数据格式要求：

1. 用户
2. 买家用户名为 buyer，密码为 reyub
3. 卖家用户名为 seller，密码为 relles。
4. 内容
5. 标题长度在[2,80]字符内；
6. 图片采用 URL地址存储；
7. 摘要长度在[2,140]字符内；
8. 正文长度在[2,1000]字符内；

# 2 实现步骤

1. 准备数据库及其数据表

根据给定的文档中的数据库文件，建立数据库及其数据表，并添加相关记录。

1. 工程目录设置：

Java目录下建立：

Controller层：存放数据交互的相关controller

Dao层：存放数据库访问接口，采用Mybatis和Spring整合后进行数据库访问。

Service层：业务逻辑处理，承上启下，Controller层请求Service层，Service层调用Dao层。采用接口实现类的方法进行编写。

Pojo层：简单的实体类，对应数据库中的数据表。

1. 整合Spring，Mybatis框架，配置文件归纳

a .建立db.properties 文件，存储数据源的配置。

b .建立spring-mybatis.xml 文件，进行数据源配置和mybatis的相关配置。

c .spring-mvc.xml 文件，freemarker相关配置和返回json数据的配置

d . web.xml文件，前端控制器，配置文件的相关声明。

1. 根据数据交互文档构建Controller

控制层将异步数据请求接口和常规请求分开编写。

1. 部署运行

部署到tomcat服务器上调试运行。

# 3 数据库及框架搭建

1. 在本机上使用MySQL创建数据库，数据库名称为：ss
2. 进入ss数据库，根据给定的数据表信息，创建数据表分别为：

content表：存储内容信息，这里由于icon字段的blob类型不便于后期读取，所以这里的icon字段类型设置成了varchar类型，用来存储内容的URL地址。

person表：存储用户信息，给定了买家和卖家两个用户，根据userType字段区分这两种用户，用户的密码均采用md5加密后进行存储。

trx表：存储账务信息，内置字段contentId和personId进行绑定，表示该用户购买了该商品。

1. 框架搭建

本项目采用SSM框架+freemarker模板引擎搭建，这个地方要说明freemarker模板引擎并不是静态的页面，向模板中填充数据后，才可以生成输出文本或者网页等。

Spring：利用Spring框架进行bean的注入和配置。

SpringMVC：主要用于Controller层的数据交互和传递。

Mybatis：关联数据库，建立数据库记录和对象的关系映射，操纵对象既可以达到操纵数据库记录的效果。

1. 相关依赖导入
2. XML相关文件配置

4 DAO层设计

DAO层设计采用了Mybatis框架，整合了Spring框架进行搭建。

1. 创建db.properties文件，用来存储数据源相关参数，主要参数包括driveClassName，url,username,password。
2. 创建spring-mybatis.xml文件，主要内容包括：
3. 自动扫描：Spring框架会自动扫描该包下添加注解的类，并进行生成。

这个地方由于整合完成后进行了单元测试，用到了Service层的对象，所以扫描的service层所在的包。

<context:component-scan base-package="com.ex.service"></context:component-scan>

1. Mybatis整合Spring：关键代码，采用了Mybatis框架，DAO层均采用接口形式，不存在实现类。整合Spring和Mybatis

<mybatis:scan base-package="com.ex.dao"/>

1. 数据源设置

获取数据源配置文件：

<context:property-placeholder location="classpath:db.properties"/>

设置数据源及其相关信息

<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">

1. sqlSessionFactory配置：mybatis的核心类

<bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">

自动扫描mapper文件

<property name="mapperLocations" value="classpath:com/ex/mapping/\*.xml">

1. Mapper文件导入配置：采用注解方式不用配置。

<bean class="org.mybatis.spring.mapper.MapperScannerConfigurer">

1. 在web.xml中声明spring-mybatis.xml的路径。

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-mybatis.xml</param-value>

</context-param>

1. 创建POJO类：

简单的实体类，对应数据库中的数据表，数据表对应对象集合，一条数据记录，对应一个对象。

根据建立数据表情况分别建立Content，Transaction，Person的pojo类，此外，还创建了一个关系pojo类Product，继承Content类，添加了isBuy，isSell等属性，用来标示内容这一实体的状态变化。例如，isBuy == true 时，表示内容已经被购买等。

1. 建立DAO层接口

对应POJO类建立了ContentDao，TransactionDao，PersonDao，ProductDao接口，并创建了所需要的方法。

例如：按照指定ID删除内容的SQL语句

@Delete("delete from content where id = #{id}")

public void deleteContent(int id);

1. 测试数据库连接：测试整合结果。

@RunWith(SpringJUnit4ClassRunner.class)

@ContextConfiguration(locations = {"classpath:spring-mybatis.xml"})

public class Test {

@Resource

private PersonService personService;

@org.junit.Test

public void test1(){

Person person = personService.getPersonById(1);

System.out.println(person);

}

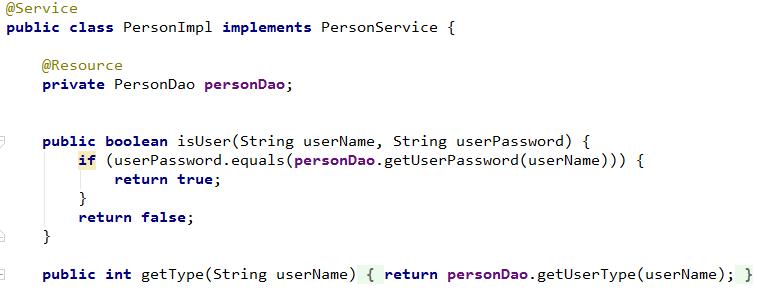
}

5 Service层设计

Service层设计采用接口+实现类的方式进行编写。

分别建立了ContentService，PersonService，ProductService，TransactionService及其对应的实现类。在Service层中注入DAO层对象，调用DAO层对象的方法进行业务逻辑处理。

PersonService实现类部分内容：

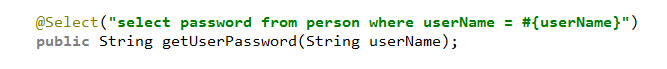


需要注意的地方：

@Service注解，标注Service层实现类，Spring实现自动注入。

@Resource注解，注入DAO层对象。

isUser()，根据Controller层传递的用户的账号和密码进行账号核查，方法中调用了personDao的验证方法进行数据库查询，如下所示：



由Service层去验证用户名对应密码是否存在，而Service再调用DAO层，DAO层跟底层的数据库打交道，查询出结果返回Service层，Service层返回结果给Controller层。Controller层根据结果返回对应的资源。如果涉及到到了DAO的操作，都是由Service去调用DAO的。

6 Controller层设计

Controller层采用的框架技术是SpringMVC。MVC控制层的作用：接收客户端的请求，然后调用Service层业务逻辑，获取到数据，传递数据给视图层（客户端）用于视图渲染，视觉呈现。

配置文件：

web.xml：声明前端控制器及其控制层配置文件。

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:spring-mvc.xml</param-value>

</init-param>

</servlet>

spring-mvc.xml：配置freemarker及其JSON数据配置

由于该系统前端页面采用的是freemarker的模板引擎，所以需要在配置文件中声明配置，以及freemarker的视图解析器。

<bean id="freemarkerConfig"

class="org.springframework.web.servlet.view.freemarker.FreeMarkerConfigurer">

<bean class="org.springframework.web.servlet.view.ContentNegotiatingViewResolver">

前端交互数据采用了ajax技术，并且在异步数据文档中明确指出返回数据为JSON类型。设置返回的JSON数据的呈现格式，需要导入fastjson+jackson-core+jackson-databind依赖。

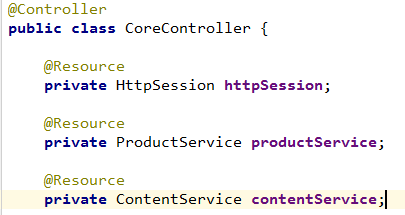
<bean id="fastJsonHttpMessageConverter" class="com.alibaba.fastjson.support.spring.FastJsonHttpMessageConverter">

Controller类具体实现：

根据数据接口文档，将Controller层划分为两部分，常规Controller和进行异步数据交互的Controller

根据异步数据接口文档构造了：Login\_a，Buy\_a，Delete\_a三个Controller，进行数据交互。

常规的请求合并写为了CoreController。



下面就登录这一操作进行详细解释：

首先，Login\_a这一异步数据的控制器。注入了PersonService的对象和HttpSessin对象(用来创建和保存会话)。

@RequestMapping注解：声明请求URL和请求方法POST。

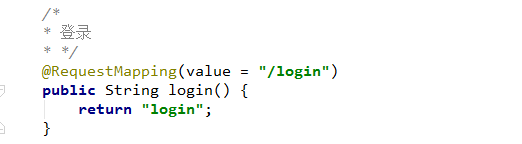
@RequestBody注解：该注解常用来处理Content-Type，例如application/json， application/xml等；

在判断用户操作成功后，建立该用户的会话，实现浏览器保存用户相应信息，实现会话跟踪。最终将结果按照数据文档规定的格式通过map返回。



CoreController中：

根据文档定义请求路径，请求发出，根据请求地址的不同会在CoreController进行匹配，并返回字符串，在spring-mvv.xml 定义的freemarker的视图解析器进行视图解析，添加后缀名，返回页面。



Spring可以很便捷地和其他MVC框架集成，如Struts，Spring 的MVC框架用控制反转把业务对象和控制逻辑清晰地隔离。SprinMVC是一个基于MVC的web框架。SprinMVC是Spring框架的一个模块，SpringMVC和Spring无需通过中间整合层进行整合。

7 实践心得体会

在本学期，我们学习了JavaEE中的三大框架，Spring，SpringMVC，Mybatis，这次的实践项目是将三大框架进行整合，前端页面采用了freemarker静态模板引擎，开始看到这个项目的时候，我对freemarker的理解不够深刻，在进行完数据库层面的整合后，Controller层的建立出现的很多问题。

freemarker的静态模板是需要用数据来进行填充的，再次我转换了思路，就index.ftl这个页面所需要的数据，以及请求的方式ajax，都已经规定好了再JS文件中。我首先熟悉了JS文件在中的ajax请求事项，尤其是URL和所携带的数据等重要数据，结合网络的博文解释，渐渐的实现了页面的显示问题。

另外一个问题就是关于内容的icon字段的blob大文件存储类型问题，如果使用大文件类型进行存储，在读取等操作的时候很繁琐，对应ftl文件中的要求，内容显示图片是在<image src=”url”> 中进行展示的，和同学进行了讨论和网上搜索，发现常规的图片存储都是采用URL地址进行存储，无论是网络图片还是本机图片，于是对数据库进行了修改，最终实现了图片的访问。

在真正的实践过程中，收获到了平时上课收获不到的知识，404这样的错误不停的出现，出现错误不害怕，重要的是尽力去解决，去完善自己薄弱的环节，集思广益，和同学老师多交流去解决问题。整合框架进行解决实际问题，从中收获更多的知识，这才是实践项目的重要意义。

感谢这次实践项目，让我更加深刻的了解了框架的内容及其深刻含义，这将是我以后的学习和工作中很宝贵的经验。