创新点报告

本系统是一个售货机系统。无人售货机凭借着没有收银员、结账无需排队、全天候售货等特点，已经成为了大众关注的焦点。本系统就是这样一个方便用户生活的系统。

一、功能创新

1. 基于CSV文件的批量用户导入功能

在设计之初，系统需要确定面向哪些对象进行使用。例如，如果将系统投放在校园内进行使用，要求只有在本校内的学生和老师才能使用；如果将系统投放在学员队内进行使用，要求只有本队的人员才能使用，以此来保证对学员队的供货能够充足，满足学员的需求，提高学员们的使用体验。

基于此设计，在系统前期即将投入使用时，需要批量导入可使用该系统的用户。因此，本系统实现了基于CSV文件的导入大批量用户的功能。可以通过线上请用户填写问卷或者线下到营业厅统计信息等方式，将绝大部分潜在用户提前录入系统，方便系统投放后用户的使用。系统已经投入使用时，再导入个别新用户则不需要通过此功能。

本功能会在后台用户管理页面提供一个按钮以供管理员使用，生成.csv文件后，管理员点击“批量导入用户”按钮，即可将.csv文件中的所有数据统一导入数据库中。即管理员只需要将.csv文件准备好即可。

1. 基于关键字的货品筛选功能

在使用一般售货机时，面对繁多的货品，一时间用户可能忘记了自己需要购买的货品，或是一时半会儿找不到自己需要的货品。因此，为了提高用户的使用体验、方便用户对需要的物品进行查找，系统投入使用时，提供了货品筛选功能。

首先是基于货品的种类进行筛选，实现了分类购物的功能，用户可以缩小自己需要查找的范围，节省了大量的搜索时间；其次是基于货品名的精确筛选，采取了模糊匹配，待用户输入关键词后，将与之匹配的货品全部搜索出，让用户能够及时找到自己需要的货品。

本功能会在前台用户使用界面提供一个导航栏给用户，方便其基于货品的种类进行筛选；也会在货品展示的上方显示一个搜索框，让用户进行基于关键字的搜索。

1. 基于Echarts的可视化功能

后台管理员可以对用户登记在系统的信息进行可视化操作。例如，管理员可以对用户的性别进行可视化，分析用户的性别组成；也可以对用户数量的变化进行可视化，分析用户量在系统投入使用的增速。

后台管理员可以对用户在系统进行购物后，生成的订单记录进行可视化，分析日订单数目的变化、总订单数目的增速等。

用户也可以对自己在此系统的消费记录进行可视化操作。例如，某用户可以对自己的购物数目进行可视化。

这些功能的实现，需要借助JFreeChart和Echarts。并且在后台管理页面中提供了可点击的按钮，直接让管理员进行使用，非常方便。

1. 基于用户购买记录的查看功能

“大数据杀熟”是个常被提起的词汇，指同样的商品或服务，[老客户](https://baike.baidu.com/item/%E8%80%81%E5%AE%A2%E6%88%B7/10750368)看到的价格反而比新客户要贵出许多的现象。针对此问题，本系统对用户经常购买的商品，提供再次购买的功能，并且确保针对各用户提供的服务是相同的。

可以给用户提供一个按钮，使其点击后，可以查看自己购买过的所有货品，选择回购。另外，基于此，用户还可以查看自己是否需要购买与之前不同的货品，来进行不同的尝试，提高用户的使用体验。

二、技术创新

1. 基于Maven的外部依赖管理

在JavaWeb开发中，需要使用大量的jar包，手动导入过于麻烦，且有的jar包之间会有依赖，只导入一个，同样无法运行。

因此本系统使用了基于Maven的外部依赖管理，导入jar包时，只需要在.pom文件中进行配置即可。而且，导入一个jar包后，MAVEN会自动导入其相关的依赖，极其方便。

IDEA内置了Maven插件，可以直接使用其进行开发，也可以在官网下载后，将jar包下载到本地的maven-repo进行开发。

1. 基于DBCP数据源的数据库连接池

在获取数据库连接时，封装了基于DBCP数据源的数据库连接池技术。

DBCP数据库连接池是 apache 上的一个Java连接池项目。

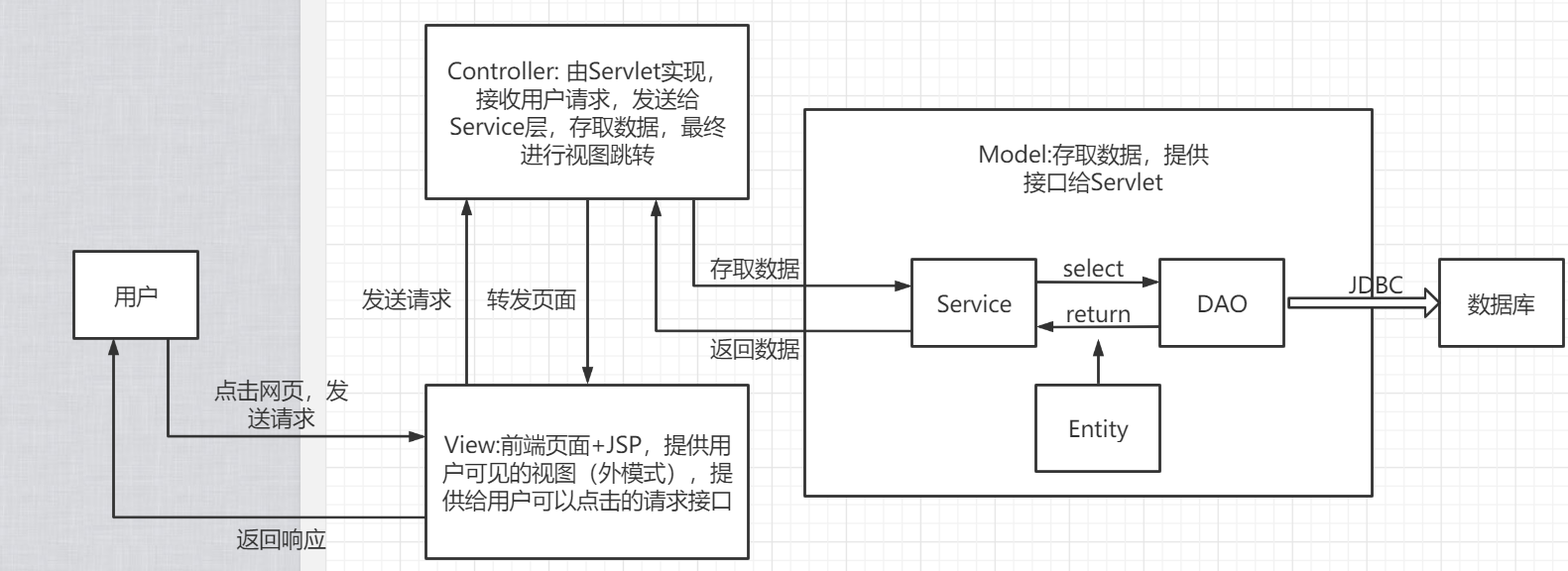
通过连接池预先同数据库建立一些连接放在内存中(即连接池中)，应用程序需要建立数据库连接时直接到从接池中申请一个连接使用，用完后由连接池回收该连接，从而达到连接复用，减少资源消耗的目的，使系统更加稳定。

具体开发时，导入jar包就用到了Maven的包管理技术。

<!-- https://mvnrepository.com/artifact/commons-dbcp/commons-dbcp -->  
<dependency>  
 <groupId>commons-dbcp</groupId>  
 <artifactId>commons-dbcp</artifactId>  
 <version>1.4</version>  
</dependency>

使用DBCP连接池时，需要配置dbcpconfig.properties文件，在DbUtil.java工具类中进行导入并封装进getConnection方法中。

1. 基于MVC框架的后端实现技术



Model是业务模型，完成了存取数据、提供接口给Servlet层的作用。具体实现中，DAO部分负责通过JDBC与数据库进行交互，每个数据库表都对应一个实体类，DAO通过DHCP数据库连接池获取连接后，使用嵌入式sql语句读取数据库中的数据，将实体类返回。Service部分负责接收DAO返回的实体类，并根据自己需要的服务，提供接口给上层，以供上层进行调用。

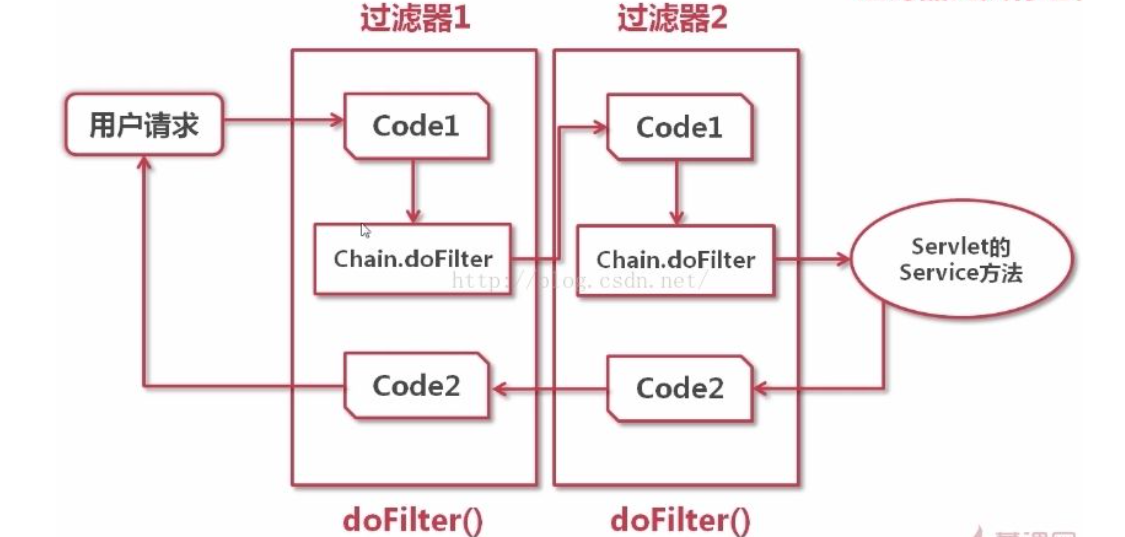
View是用户界面，即用户可见的外模式。前端页面是由JSP+HTML+CSS+JS进行编写的，页面中会提供给便于用户操作的可点击的按钮，这些按钮绑定了对应的请求，可以发送给控制层，控制层再调用业务层进行处理，最终返回给用户响应，使得用户获得自己需要的页面。

Controller是控制层，主要负责将用户请求与业务模型进行对接。这一层通过编写对应的Servlet进行实现。每个Servlet都是继承自Java的HttpServlet类，只需重写doGet和doPost函数即可。用户提交的请求会作为参数HttpServletRequest传入doGet或doPost函数，业务模型返回的响应会作为参数HttpServletResponse传回，最终进行页面重定向或转发后，用户所得的新页面即是所请求的页面，一个功能也就此实现。

1. 基于Filter实现的登录页面拦截
2. **public** **void** doFilter(ServletRequest req, ServletResponse resp, FilterChain chain) **throws** IOException, ServletException {
3. HttpServletRequest request = (HttpServletRequest) req;
4. HttpServletResponse response = (HttpServletResponse) resp;
5. User user = (User) request.getSession().getAttribute(Constants.USER\_SESSION);
6. **if**(user==**null**){
7. //已经被移除（注销）或者未登录
8. response.sendRedirect(request.getContextPath()+"/err/getback.jsp");
9. }
10. chain.doFilter(req, resp);
12. }

过滤器可以实现对客户和资源之间的请求和响应进行过滤与处理。当用户通过浏览器访问服务器中的目标资源时，会被Filter过滤器拦截，在Filter过滤器中可以对用户请求进行预处理，然后再将请求转发给目标资源。

其工作原理如下图所示。



1. 基于JFreeChart实现的可视化操作
2. **protected** **void** doPost(HttpServletRequest req, HttpServletResponse resp) **throws** ServletException, IOException {
3. Connection connection = BaseDao.getConnection();
4. **try** {
5. JDBCPieDataset jdbcPieDataset = **new** JDBCPieDataset(connection, "select gender,userRole from smbms\_user");
6. JFreeChart test = ChartFactory.createPieChart("Test", jdbcPieDataset);
7. HttpSession session = req.getSession();
8. String filename = ServletUtilities.saveChartAsJPEG(test, 400, 250, session);
9. String url = req.getContextPath() + "/servlet/DisplayChart?filename=" + filename;
10. req.setAttribute("url", url);
11. req.setAttribute("Gname", "饼图");
12. req.getRequestDispatcher("showJFC.jsp").forward(req, resp);

15. } **catch** (SQLException e) {
16. e.printStackTrace();
17. }
18. }

JFreeChart是JAVA平台上的一个开放的图表绘制类库。它完全使用JAVA语言编写，是为applications, applets, servlets 以及JSP等使用所设计。JFreeChart的缺陷就是只适合生成图片，而且文字、图片都不是特别清晰，还需要进行一些调整，比较麻烦。

本系统目前使用了此开源库作为图像绘制的工具，如上代码所示，使用此库能够方便地与数据库和JSP页面进行交互。

1. 基于MyBatis实现的DAO层

MyBatis 免除了几乎所有的 JDBC 代码以及设置参数和获取结果集的工作，它可以通过简单的XML文件来配置与数据库进行交互的Mapper（DAO层）。

使用时，首先通过Maven导入对应的依赖。

然后编写mybatis-config.xml配置文件

1. **<?xml** version="1.0" encoding="UTF-8" **?>**
2. <!DOCTYPE configuration
3. PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
4. "http://mybatis.org/dtd/mybatis-3-config.dtd"**>**
5. **<configuration>**
6. **<environments** default="development"**>**
7. **<environment** id="development"**>**
8. **<transactionManager** type="JDBC"**/>**
9. **<dataSource** type="POOLED"**>**
10. **<property** name="driver" value="com.mysql.jdbc.Driver"**/>**
11. **<property** name="url" value="jdbc:mysql://localhost:3306/mybatis?serverTimezone=UTC&useUnicode=true&characterEncoding=UTF-8"**/>**
12. **<property** name="username" value="root"**/>**
13. **<property** name="password" value="123456"**/>**
14. **</dataSource>**
15. **</environment>**
16. **</environments>**
18. **</configuration>**

再构建SqlSessionFactory对象，获取SqlSession实例。

DAO层的接口类不变，增加一个Mapper.xml配置文件。

1. **<?xml** version="1.0" encoding="UTF-8" **?>**
2. <!DOCTYPE mapper
3. PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
4. "http://mybatis.org/dtd/mybatis-3-mapper.dtd"**>**
5. <!--绑定一个对应的Mapper接口-->
6. **<mapper** namespace="com.Ray.dao.UserDao"**>**
7. <!--   查询语句 -->
8. **<select** id="getUserList" resultType="com.Ray.pojo.User"**>**
9. select \* from mybatis.user
10. **</select>**
11. **</mapper>**

pom文件中需要增加资源导出的配置。

1. **<build>**
2. **<resources>**
3. **<resource>**
4. **<directory>**src/main/resources**</directory>**
5. **<includes>**
6. **<include>**\*\*/\*.properties**</include>**
7. **<include>**\*\*/\*.xml**</include>**
8. **</includes>**
9. **<filtering>**true**</filtering>**
10. **</resource>**
11. **<resource>**
12. **<directory>**src/main/java**</directory>**
13. **<includes>**
14. **<include>**\*\*/\*.properties**</include>**
15. **<include>**\*\*/\*.xml**</include>**
16. **</includes>**
17. **<filtering>**true**</filtering>**
18. **</resource>**
19. **</resources>**
20. **</build>**

之后便可以轻松编写代码，实现功能。