# 1. 系统概述

## 1.1 设计目的

本设计旨在开发一个功能完善的网络商城系统，专注于二次元周边商品的管理和销售。  
系统需具备吸引二次元文化爱好者的界面设计，同时实现高效的商品管理、订单处理以及用户交互功能，以提升用户体验和促进商品流通。

## 1.2 项目概况

* 用户管理：支持用户注册、登录、个人信息修改及删除账户等功能。
* 商品管理：包括商品添加、编辑、删除及分类展示，支持上传商品图片、描述及库存管理。

# 2. 系统分析

## 2.1 系统规划

前后端分离，总体功能需要支持用户注册、登录，支持管理员（商家）对商品数据进行增删改查。

## 2.2 运行环境规定

| 依赖项 | 版本 |
| --- | --- |
| JUnit - 单元测试库 | 3.8.1 |
| Java Servlet JSP API | 2.3.3 |
| Java Servlet API | 4.0.1 |
| Lombok | 1.18.26 |
| MySQL Connector/J | 8.0.32 |
| MariaDB Java Client | 3.3.3 |
| MyBatis | 3.5.13 |
| Logback Classic | 1.4.6 (test) |
| Spring Context | 5.2.10.RELEASE |
| Druid | 1.1.22 |
| Spring JDBC | 5.2.10.RELEASE |
| MyBatis-Spring | 2.0.7 |
| Java Annotation API | 1.2 |

## 2.3 需求分析

关于用户数据方面，允许新用户注册为普通用户，且允许管理员管理所有用户数据，包括新增用户、删除用户、修改用户权限等。在商品数据方面，只有管理员有权添加新商品、修改商品数据、删除已存在的商品。

## 2.4 可行性分析

通过建立2个数据表对用户数据和商品数据进行管理。同时使用前端页面供非程序员用户对数据库进行操作。由后端通过封装好的DAO层与数据库进行交互，

# 3. 概要（总体）设计

## 3.1 系统功能设计

### 用户登录验证

用户登录验证模块是系统的入口，确保只有经过身份验证的用户才能访问系统的功能。需要实现用户名和密码输入、身份验证逻辑以及登录成功与失败的反馈。

### 商品管理

商品管理模块允许管理员用户对商品信息进行增加、删除、修改和查询，无需进入代码层面进行修改。

### 用户管理

用户管理模块允许管理员对用户信息进行增加、删除、修改和查询。

### 数据库连接模块

数据库连接模块负责建立和管理系统与数据库的连接。包括数据库连接的创建、数据库连接的关闭、数据库连接池的管理。

### Servlet处理模块

Servlet处理模块是系统与用户交互的接口，负责处理用户的HTTP请求。

## 3.2 系统结构设计

系统采用MVC架构模式，分为表现层、业务逻辑层和数据访问层。

* **表现层（Presentation Layer）**：负责向用户展示信息和接收用户输入。
* **业务逻辑层（Service Layer）**：包含业务逻辑和处理用户请求。
* **数据访问层（Data Access Layer, DAL）**：负责与数据库交互，执行数据的增删改查操作。
* **持久层（Persistence Layer）**：由数据库和数据库连接逻辑组成。

表现层组件包括：

* **JSP页面**：用于展示用户界面和表单。
* **Servlet**：处理HTTP请求和响应，调用业务逻辑层。

业务逻辑层组件包括：

* **Service接口**：定义业务逻辑操作的接口。
* **ServiceImpl类**：实现Service接口的具体业务逻辑。

数据访问层组件包括：

* **Mapper接口**：使用MyBatis框架定义的数据库操作接口。
* **XML映射文件**：MyBatis的映射配置文件，定义SQL语句。

持久层组件包括：

* **数据库表**：存储数据的数据库表。
* **数据库连接配置**：数据库连接的配置信息。

### 用户请求流程

1. 用户通过浏览器发送HTTP请求到服务器。
2. Servlet接收请求，并根据请求类型调用相应的Service。
3. Service调用DAO层执行数据库操作。
4. DAO层通过MyBatis框架与数据库交互。
5. 数据库返回结果给DAO层。
6. Service处理结果并返回给Servlet。
7. Servlet生成响应并发送给用户。

### 数据库操作流程

1. DAO层请求数据库连接。
2. 数据库连接池提供连接。
3. DAO层执行SQL语句。
4. 数据库返回查询结果或更新状态。
5. DAO层关闭数据库连接。

## 3.3 数据库设计

数据库设计包括用户表和商品表，下面是主要数据表的结构。

* 商品数据表

CREATE TABLE `t\_product` (  
 `pid` INT(11) NOT NULL,  
 `productName` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4\_bin',  
 `productDesc` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4\_bin',  
 `productCat` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4\_bin',  
 `productCount` INT(11) NULL DEFAULT NULL,  
 PRIMARY KEY (`pid`) USING BTREE  
)

* 用户数据表

CREATE TABLE `t\_user` (  
 `uid` INT(11) NOT NULL AUTO\_INCREMENT,  
 `user\_name` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4\_bin',  
 `password` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4\_bin',  
 `permission` INT(11) NULL DEFAULT NULL,  
 PRIMARY KEY (`uid`) USING BTREE  
)

# 4. 系统后端主要模块实现

## 4.1 登录验证模块

1. **AccountService 接口** (AccountService.java)：

* 定义了登录服务的契约，声明了登录方法 login。
* public interface AccountService {
    
   /\*\*
    
   \* 登录，查询数据库中是否有该和用户名和密码
    
   \* @param userName
    
   \* @param password
    
   \* @return true，登录成功；false，登录失败
    
   \*/
    
   Boolean login(String userName, String password);
    
    
   /\*\*
    
   \* 注册，用户数据数据存入数据库
    
   \* @param userName
    
   \* @param password
    
   \* @return true，注册成功；false，注册失败
    
   \*/
    
   Boolean signup(String userName, String password) throws RuntimeException;
    
  }

1. **AccountServiceImpl 类** (AccountServiceImpl.java)：

* 实现了 AccountService 接口，具体实现了登录逻辑，通过调用 UserDao 来验证用户名和密码。
* @Service
    
  public class AccountServiceImpl implements AccountService {
    
   @Resource
    
   private UserDao userDao;
    
    
   @Override
    
   public Boolean login(String userName, String password) {
    
   User user=userDao.selectByUnameAndPwd(userName,password);
    
   if(user == null) { // 没查询到该用户，登陆失败，返回 false
    
   System.out.println("Login failed.");
    
   return false;
    
   } else { // 登录成功，返回 true
    
   System.out.println("User [" + user.getUserName() + "] Login successful.");
    
   return true;
    
   }
    
    
   }
    
    
   @Override
    
   public Boolean signup(String userName, String password) throws RuntimeException {
    
   User user = userDao.selectUserByName(userName);
    
   if (user != null) { // 用户已存在 返回 false 注册失败
    
   System.out.println("User [" + user.getUserName() + "] already exists.");
    
   return false;
    
   } else { // 将注册表单中的用户名和密码插入数据库，同时设置权限为普通人
    
   User userFrom = new User(null, userName, password, 1);
    
   if (userDao.insertUser(userFrom) == 1) { // 插入成功，返回 true 注册成功
    
   System.out.println("User [" + userFrom.getUserName() + "] signup successful.");
    
   return true;
    
   } else { // 插入失败，发生异常
    
    
   throw new RuntimeException("Unknown err.");
    
   }
    
    
   }
    
   }
    
   }

1. **UserDao 接口** (UserDao.java)：

* 定义了与用户数据相关的数据库操作，比如 insertUser 方法，虽然它在这里是用于添加用户，但通常还会有查询用户的方法，比如 findByUsername 来根据用户名查询用户信息。
* public interface UserDao {
    
   /\*\*
    
   \* 新增用户
    
   \* @param user
    
   \* @return 1 成功；0 失败
    
   \*/
    
   @Insert("insert into t\_user(user\_name, password, permission) " +
    
   "values (#{userName}, #{password}, #{permission}) ")
    
   int insertUser(User user);
    
    
   /\*\*
    
   \* 根据 uid 删除用户
    
   \* @param uid
    
   \* @return 1 成功；0 失败
    
   \*/
    
   @Delete("delete from t\_user where uid = #{uid}")
    
   int deleteById(int uid);
    
    
   /\*\*
    
   \* 修改用户信息
    
   \* @param user
    
   \* @return 1 成功；0 失败
    
   \*/
    
   @Update("update t\_user set user\_name=#{userName}, password=#{password}, permission=#{permission} where uid=#{uid}")
    
   int updateUser(User user);
    
    
   /\*\*
    
   \* 根据 uid 查询用户
    
   \* @param userId
    
   \* @return 用户对象
    
   \*/
    
   @Select("select \* from t\_user where uid = #{userId}")
    
   @Result(column = "user\_name", property = "userName")
    
   User selectById(int userId);
    
    
   /\*\*
    
   \* 根据用户名和密码查询用户
    
   \* @param userName
    
   \* @param password
    
   \* @return 用户对象
    
   \*/
    
   @Select("select \* from t\_user where user\_name = #{uname} and password = #{pwd}")
    
   @Result(column = "user\_name", property = "userName")
    
   User selectByUnameAndPwd(@Param("uname")String userName, @Param("pwd")String password);
    
    
   /\*\*
    
   \* 查询所有用户
    
   \* @return 用户对象列表
    
   \*/
    
   @Select("select \* from t\_user")
    
   @Results(@Result(column = "user\_name", property = "userName"))
    
   List<User> selectAll();
    
    
   /\*\*
    
   \* 根据用户名查询用户，支持模糊搜索
    
   \* @param userName
    
   \* @return 用户对象列表
    
   \*/
    
   @Select("select \* from t\_user where user\_name like concat('%', #{userName}, '%')")
    
   @Result(column = "user\_name", property = "userName")
    
   List<User> selectUsersByName(String userName);
    
    
   /\*\*
    
   \* 根据用户名查询单个用户
    
   \* @param userName
    
   \* @return 用户对象
    
   \*/
    
   @Select("select \* from t\_user where user\_name = #{userName}")
    
   @Result(column = "user\_name", property = "userName")
    
   User selectUserByName(String userName);
    
    
   @Select("select permission from t\_user where user\_name = #{userName}")
    
   int selectPermissionByName(String userName);
    
  }

1. **AccountServlet** (AccountServlet.java)：

* 作为 Servlet，它处理来自客户端的登录请求，调用 AccountService 的 login 方法，并根据返回结果决定是重定向到管理页面还是显示错误信息。
* @WebServlet({"/account/login", "/account/logout", "/account/signup"})
    
  public class AccountServlet extends HttpServlet {
    
   ApplicationContext ctx=new AnnotationConfigApplicationContext(SpringConfig.class);
    
   AccountService accountService=ctx.getBean(AccountService.class);
    
   @Override
    
   protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    
   String contextPath = req.getContextPath();
    
   // 模板方法
    
   String servletPath = req.getServletPath();
    
   System.out.println(contextPath + servletPath);
    
   if ("/account/login".equals(servletPath)) {
    
   if(login(req, resp)) { // 登录成功，跳转管理页面
    
   resp.sendRedirect(contextPath + "/jsp/Manager\_page\_productList.jsp");
    
   } else { // 登录失败
    
   resp.sendRedirect(contextPath + "/jsp/unameOrPwdErr.jsp");
    
   }
    
   }
    
   else if ("/account/logout".equals(servletPath)) {
    
   logout(req, resp);
    
   // 退出登录后回到主页，重定向到主页
    
   resp.sendRedirect(contextPath + "/index.jsp");
    
   }
    
   else if ("/account/signup".equals(servletPath)) {
    
   if (signup(req, resp)) { // 注册成功，跳转到登陆页面
    
   resp.sendRedirect(contextPath + "/jsp/login.jsp");
    
   } else { // 注册失败
    
   resp.sendRedirect(contextPath + "/jsp/registerErr.jsp");
    
   }
    
   }
    
   }
    
    
   private Boolean login(HttpServletRequest request, HttpServletResponse response) throws IOException {
    
   request.setCharacterEncoding("UTF-8");
    
   // 获取请求的登录信息
    
   String userName = request.getParameter("userName");
    
   String password = request.getParameter("userPwd");
    
    
    
   if (accountService.login(userName, password)) { // 登陆成功，重定向到管理页面
    
   // 创建 cookie 并返回给浏览器
    
   Cookie cookie1 = new Cookie("userName", userName); // 创建 cookie 对象
    
   Cookie cookie2 = new Cookie("userPwd", password);
    
   cookie1.setMaxAge(60 \* 60 \* 8); // 设置 cookie 在 8h 后失效
    
   cookie2.setMaxAge(60 \* 60 \* 8);
    
   cookie1.setPath(request.getContextPath()); // 设置 cookie 的关联路径
    
   cookie2.setPath(request.getContextPath());
    
   response.addCookie(cookie1); // 将 cookie 响应到浏览器
    
   response.addCookie(cookie2);
    
    
   HttpSession session = request.getSession(); // 获取当前会话
    
   session.setAttribute("userName", userName); // 将 userName 存入会话域
    
   return true;
    
   }
    
   return false;
    
   }
    
    
   private void logout(HttpServletRequest request, HttpServletResponse response) throws IOException {
    
   System.out.println("==============logout(): current cookies==============");
    
   for (Cookie cookie : request.getCookies()) {
    
   System.out.println(cookie.getName() + "=" + cookie.getValue());
    
   }
    
   System.out.println("=====================================================");
    
   // 防止之前登录过的 cookie 影响，清除 userName 和 userPwd
    
   Cookie cookie1 = new Cookie("userName", null);
    
   Cookie cookie2 = new Cookie("userPwd", null);
    
   cookie1.setMaxAge(0); // 设置有效时间为 0
    
   cookie2.setMaxAge(0);
    
   cookie1.setPath(request.getContextPath()); // 设置 cookie 的关联路径
    
   cookie2.setPath(request.getContextPath());
    
   response.addCookie(cookie1); // cookie 下发到浏览器
    
   response.addCookie(cookie2);
    
    
   HttpSession session = request.getSession();
    
   if(session != null) {
    
   session.invalidate(); // 销毁 session 对象
    
   System.out.println("logout");
    
   }
    
   }
    
    
   private Boolean signup(HttpServletRequest request, HttpServletResponse response) throws IOException {
    
   request.setCharacterEncoding("UTF-8");
    
   // 获取请求的注册信息
    
   String userName = request.getParameter("userName");
    
   String password = request.getParameter("userPwd");
    
    
   if(accountService.signup(userName, password)) { // 注册成功后跳转到登录页面
    
   logout(request, response); // 防止注册前登录过的账号的 cookie 影响
    
   return true;
    
   }
    
   return false;
    
   }
    
  }

1. **LoginCheckFilter** (LoginCheckFilter.java)：

* 这是一个过滤器，用于检查用户是否已经登录，如果是，则允许访问受限页面，否则重定向到登录页面。
* package com.itheima.filter;
    
    
    
  import com.itheima.config.SpringConfig;
    
  import com.itheima.service.AccountService;
    
  import com.itheima.service.UserService;
    
  import com.itheima.service.impl.AccountServiceImpl;
    
  import com.itheima.service.impl.UserServiceImpl;
    
  import org.springframework.context.ApplicationContext;
    
  import org.springframework.context.annotation.AnnotationConfigApplicationContext;
    
    
  import javax.servlet.FilterChain;
    
  import javax.servlet.ServletException;
    
  import javax.servlet.annotation.WebFilter;
    
  import javax.servlet.http.\*;
    
  import java.io.IOException;
    
    
  // 在登录的前提下才可进入下面三个路径
    
  @WebFilter({"/jsp/Manager\_page\_productList.jsp", "/jsp/Manager\_page\_userList.jsp", "/user", "/product"})
    
  public class LoginCheckFilter extends HttpFilter {
    
   ApplicationContext ctx=new AnnotationConfigApplicationContext(SpringConfig.class);
    
   UserService userService=ctx.getBean(UserService.class);
    
    
   @Override
    
   protected void doFilter(HttpServletRequest req, HttpServletResponse resp, FilterChain chain) throws IOException, ServletException {
    
   System.out.println("-------------------doFilter-------------------");
    
   for (Cookie cookie : req.getCookies()) {
    
   System.out.println(cookie.getName() + "=" + cookie.getValue());
    
   }
    
   System.out.println("-----------------------------------------------");
    
   // 若 cookies 中的登录信息有效就直接用该信息自动登录
    
    
   HttpSession session = req.getSession(); // 获取会话
    
   String userName = (String) session.getAttribute("userName"); // 从会话域获取用户名
    
   // 从会话域中获取 userName，若获取到对应用户名代表已登录
    
   if(session != null && userName != null) {
    
   String servletPath = req.getServletPath();
    
   System.out.println(session.getAttribute("userName") + " logged in, accessing " + servletPath);
    
   // 如果想进入用户管理页面或就行用户管理需要判断权限
    
   if("/jsp/Manager\_page\_userList.jsp".equals(servletPath) || servletPath.contains("/user/")) {
    
   if(userService.isAdmin(userName)) {
    
   chain.doFilter(req, resp); // 是 admin 放行
    
   } else { // 权限不足
    
   String ctxPath = req.getServletContext().getContextPath();
    
   resp.sendRedirect(ctxPath + "/jsp/permissionErr.jsp");
    
   }
    
   } else { // 若不是访问 userList.jsp 不做处理，直接放行
    
   chain.doFilter(req, resp);
    
   }
    
   } else { // 未登录，跳转到登陆页面
    
   String ctxPath = req.getServletContext().getContextPath();
    
   resp.sendRedirect(ctxPath + "/jsp/login.jsp");
    
   }
    
   System.out.println("filter over");
    
   }
    
    
    
   static void autoLoginByCookie(HttpServletRequest req) {
    
   System.out.println("Using cookie");
    
   Cookie[] cookies = req.getCookies();
    
   String userName = null;
    
   String password = null;
    
   if (cookies != null) {
    
   for (Cookie cookie : cookies) {
    
   if("userName".equals(cookie.getName())) {
    
   userName = cookie.getValue();
    
   } else if("userPwd".equals(cookie.getName())) {
    
   password = cookie.getValue();
    
   }
    
   }
    
   }
    
   ApplicationContext ctx=new AnnotationConfigApplicationContext(SpringConfig.class);
    
   AccountService accountService=ctx.getBean(AccountService.class);
    
   if(accountService.login(userName, password)) {
    
   HttpSession session = req.getSession(); // 获取会话
    
   session.setAttribute("userName", userName); // 将 userName 存入会话域
    
   }
    
   }
    
  }

1. **LoginCheckFilter2** (LoginCheckFilter2.java)：

* 另一个过滤器，特别用于检查用户在尝试访问登录页面时是否已经登录，如果是，则自动重定向到管理页面。
* package com.itheima.filter;
    
    
  import javax.servlet.FilterChain;
    
  import javax.servlet.ServletException;
    
  import javax.servlet.annotation.WebFilter;
    
  import javax.servlet.http.HttpFilter;
    
  import javax.servlet.http.HttpServletRequest;
    
  import javax.servlet.http.HttpServletResponse;
    
  import javax.servlet.http.HttpSession;
    
  import java.io.IOException;
    
    
  // 在登录的前提下访问登录页面时直接到管理页面
    
  @WebFilter("/jsp/login.jsp")
    
  public class LoginCheckFilter2 extends HttpFilter {
    
   @Override
    
   protected void doFilter(HttpServletRequest req, HttpServletResponse resp, FilterChain chain) throws IOException, ServletException {
    
   // 若 cookies 中的登录信息有效就直接用该信息自动登录
    
   //LoginCheckFilter.autoLoginByCookie(req);
    
    
   HttpSession session = req.getSession(); // 获取会话
    
   String ctxPath = req.getServletContext().getContextPath();
    
   // 从会话域中获取 userName，若获取到对应用户名代表已登录
    
   if(session != null && session.getAttribute("userName") != null) {
    
   System.out.println(session.getAttribute("userName") + " logged in.");
    
   resp.sendRedirect(ctxPath + "/jsp/Manager\_page\_productList.jsp");
    
   }
    
   chain.doFilter(req, resp);
    
   System.out.println("filter2 over");
    
   }
    
  }

这些文件和组件协同工作，实现了登录验证模块的功能。用户提交登录信息后，AccountServlet 处理请求，调用 AccountService 的 login 方法进行验证，如果验证成功，则允许用户访问系统的功能模块；如果失败，则显示错误信息或重定向回登录页面。同时，LoginCheckFilter 和 LoginCheckFilter2 确保用户在访问受限资源时已经通过了身份验证。

## 4.2 商品管理模块

1. **ProductService 接口** (ProductService.java)：

* 定义了商品服务的契约，声明了商品管理所需的方法，如 save。
* package com.itheima.service;
    
    
  import com.itheima.pojo.Product;
    
    
  import java.util.List;
    
    
  public interface ProductService {
    
   int save(Product product);
    
    
   int delete(int pid);
    
    
   int update(Product product);
    
    
   Product getById(int pid);
    
    
   List<Product> getAll();
    
    
   List<Product> getByName(String productName);
    
  }

1. **ProductServiceImpl 类** (ProductServiceImpl.java)：

* 实现了 ProductService 接口，包含具体的商品管理逻辑，如添加、删除、更新商品信息。
* package com.itheima.service.impl;
    
    
  import com.itheima.pojo.Product;
    
  import com.itheima.dao.ProductDao;
    
  import com.itheima.service.ProductService;
    
  import org.springframework.stereotype.Service;
    
    
  import javax.annotation.Resource;
    
  import java.util.List;
    
  @Service
    
  public class ProductServiceImpl implements ProductService{
    
   @Resource
    
   private ProductDao productDao;
    
   @Override
    
   public int save(Product product) {
    
   int flag = productDao.insertProduct(product);
    
   return flag;
    
   }
    
    
   @Override
    
   public int delete(int pid) {
    
   int flag = productDao.deleteById(pid);
    
   return flag;
    
   }
    
    
   @Override
    
   public int update(Product product) {
    
   int flag = productDao.updateProduct(product);
    
   return flag;
    
   }
    
    
   @Override
    
   public Product getById(int pid) {
    
   Product product = productDao.selectById(pid);
    
   return product;
    
   }
    
    
   @Override
    
   public List<Product> getAll() {
    
    
   List<Product> productList = productDao.selectAll();
    
   return productList;
    
   }
    
    
   @Override
    
   public List<Product> getByName(String productName) {
    
   List<Product> productList = productDao.selectProductsByName(productName);
    
   return productList;
    
   }
    
  }

1. **ProductDao 接口** (ProductDao.java)：

* 定义了与商品数据相关的数据库操作，使用 MyBatis 注解来映射 SQL 语句。
* package com.itheima.service;
    
    
  import com.itheima.pojo.Product;
    
    
  import java.util.List;
    
    
  public interface ProductService {
    
   int save(Product product);
    
    
   int delete(int pid);
    
    
   int update(Product product);
    
    
   Product getById(int pid);
    
    
   List<Product> getAll();
    
    
   List<Product> getByName(String productName);
    
  }

1. **Product 类** (Product.java)：

* 作为 POJO（Plain Old Java Object），代表商品数据模型，包含商品属性和对应的 getter 和 setter 方法。
* package com.itheima.pojo;
    
    
  import lombok.AllArgsConstructor;
    
  import lombok.Data;
    
  import lombok.NoArgsConstructor;
    
    
  @Data
    
  @NoArgsConstructor
    
  @AllArgsConstructor
    
  public class Product {
    
   private Integer pid;
    
   private String productName; // 商品名
    
   private String productDesc; // 商品描述
    
   private String productCat; // 商品类别 productCategory
    
   private Integer productCount; // 商品数量
    
  }

1. **ProductServlet** (ProductServlet.java)：

* 作为 Servlet，处理来自客户端的商品管理请求，调用 ProductService 的方法来执行具体的商品管理操作。
* package com.itheima.web;
    
    
  import com.itheima.config.SpringConfig;
    
  import com.itheima.pojo.Product;
    
  import com.itheima.service.ProductService;
    
  import com.itheima.service.impl.ProductServiceImpl;
    
  import org.springframework.context.ApplicationContext;
    
  import org.springframework.context.annotation.AnnotationConfigApplicationContext;
    
    
  import javax.servlet.\*;
    
  import javax.servlet.http.\*;
    
  import javax.servlet.annotation.\*;
    
  import java.io.IOException;
    
  import java.util.List;
    
    
  @WebServlet({"/product/add", "/product/delete", "/product/update", "/product/getById", "/product/getAll"})
    
  public class ProductServlet extends BaseServlet {
    
   ApplicationContext ctx=new AnnotationConfigApplicationContext(SpringConfig.class);
    
   private ProductService productService = ctx.getBean(ProductService.class);
    
    
   @Override
    
   protected void service(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    
   String contextPath = req.getContextPath();
    
   // 模板方法
    
   String servletPath = req.getServletPath();
    
   System.out.println(contextPath + servletPath);
    
   if ("/product/add".equals(servletPath)) {
    
   add(req, resp);
    
   } else if ("/product/delete".equals(servletPath)) {
    
   delete(req, resp);
    
   } else if ("/product/update".equals(servletPath)) {
    
   update(req, resp);
    
   } else if ("/product/getById".equals(servletPath)) {
    
   getById(req, resp);
    
   } else if ("/product/getAll".equals(servletPath)) {
    
   getAll(req, resp);
    
   }
    
   // 执行完对应操作后出现打开商品管理页面
    
   resp.sendRedirect(contextPath + "/jsp/Manager\_page\_productList.jsp");
    
   }
    
    
   /\*\*
    
   \* 新增商品
    
   \*/
    
   public void add(HttpServletRequest request,HttpServletResponse response) throws IOException {
    
   //1.获取参数,商品的id 名称 描述 类别 库存
    
   int id = Integer.parseInt(request.getParameter("id"));
    
   String name = request.getParameter("p\_name");
    
   String desc = request.getParameter("p\_desc");
    
   String category = request.getParameter("p\_cat");
    
   int count = Integer.parseInt(request.getParameter("p\_count"));
    
   //2.调用productService 的方法进行进行将 商品入库
    
   Product product = new Product(id,name,desc,category,count);
    
   productService.save(product);
    
   }
    
    
   /\*\*
    
   \* 删除商品
    
   \*/
    
   public void delete(HttpServletRequest request,HttpServletResponse response) throws IOException {
    
   //1. 获取 需要删除参数的 请求id
    
   int id = Integer.parseInt(request.getParameter("id"));
    
   //2. 调用service 层的方法进行删除
    
   productService.delete(id);
    
   }
    
    
   /\*\*
    
   \* 更新商品
    
   \* @param request
    
   \* @param response
    
   \* @throws IOException
    
   \*/
    
   public void update(HttpServletRequest request ,HttpServletResponse response) throws IOException {
    
   //1.获取参数,商品的id 名称 描述 类别 库存
    
   int id = Integer.parseInt(request.getParameter("id"));
    
   String name = request.getParameter("p\_name");
    
   String desc = request.getParameter("p\_desc");
    
   String category = request.getParameter("p\_cat");
    
   int count = Integer.parseInt(request.getParameter("p\_count"));
    
    
   //2.调用productService 的方法进行进行将 更新商品
    
   productService.update(new Product(id,name,desc,category,count));
    
   }
    
    
   /\*\*
    
   \* 根据 id 查询商品
    
   \*/
    
   public void getById(HttpServletRequest request ,HttpServletResponse response)throws IOException{
    
   //1. 获取 需要删除参数的 请求id
    
   int id = Integer.parseInt(request.getParameter("id"));
    
   //2. 调用service 层的方法进行删除
    
   productService.getById(id);
    
   }
    
   /\*\*
    
   \* 获取全部的商品
    
   \*/
    
   public void getAll(HttpServletRequest request ,HttpServletResponse response) throws IOException, ServletException {
    
   // 获取所有商品数据
    
   List<Product> productList = productService.getAll();
    
   // 将商品列表放入request作用域中
    
   request.setAttribute("productList", productList);
    
   }
    
  }

这些文件和组件共同构成了商品管理模块的架构，它们协同工作以实现商品的增删改查等功能。用户通过前端页面进行操作，请求被 ProductServlet 捕获并处理，然后调用 ProductService 和 ProductDao 来执行具体的数据库操作。

## 4.3 用户管理模块

在您提供的代码和文档中，用户管理模块涉及的文件和组件主要包括：

1. **UserService 接口** (UserService.java)：

* 定义了用户服务的契约，声明了用户管理所需的方法，例如 save 方法。
* package com.itheima.service;
    
    
  import com.itheima.pojo.User;
    
    
  import java.util.List;
    
    
  public interface UserService {
    
   int save(User user);
    
    
   int delete(int uid);
    
    
   int update(User user);
    
    
   User getById(int pid);
    
    
   List<User> getAll();
    
    
   List<User> getByName(String userName);
    
    
   boolean isAdmin(String userName);
    
  }

1. **UserServiceImpl 类** (UserServiceImpl.java)：

* 实现了 UserService 接口，包含具体的用户管理逻辑，如添加、删除、更新用户信息。
* package com.itheima.service.impl;
    
    
  import com.itheima.pojo.User;
    
  import com.itheima.dao.UserDao;
    
  import com.itheima.service.UserService;
    
  import org.springframework.stereotype.Service;
    
    
    
  import javax.annotation.Resource;
    
  import java.util.List;
    
  @Service
    
  public class UserServiceImpl implements UserService {
    
   @Resource
    
   private UserDao userDao;
    
   @Override
    
   public int save(User user) {
    
    
   int flag = userDao.insertUser(user);
    
   return flag;
    
   }
    
    
   @Override
    
   public int delete(int uid) {
    
   int flag =userDao.deleteById(uid);
    
   return flag;
    
   }
    
    
   @Override
    
   public int update(User user) {
    
   int flag = userDao.updateUser(user);
    
   return flag;
    
   }
    
    
   @Override
    
   public User getById(int pid) {
    
   User user = userDao.selectById(pid);
    
   return user;
    
   }
    
    
   @Override
    
   public List<User> getAll() {
    
   List<User> productList = userDao.selectAll();
    
   return productList;
    
   }
    
    
   @Override
    
   public List<User> getByName(String userName) {
    
   List<User> productList =userDao.selectUsersByName(userName);
    
   return productList;
    
   }
    
    
   /\*\*
    
   \* 判断是否是超级管理员 Administrator
    
   \* @param userName
    
   \* @return true/false
    
   \*/
    
   @Override
    
   public boolean isAdmin(String userName) {
    
    
   if(userDao.selectPermissionByName(userName) == 0) {
    
   return true;
    
   }
    
   return false;
    
   }
    
  }

1. **UserDao 接口** (UserDao.java)：

* 定义了与用户数据相关的数据库操作，使用 MyBatis 注解来映射 SQL 语句。
* package com.itheima.dao;
    
    
    
  import com.itheima.pojo.User;
    
  import org.apache.ibatis.annotations.\*;
    
    
  import java.util.List;
    
    
  public interface UserDao {
    
   /\*\*
    
   \* 新增用户
    
   \* @param user
    
   \* @return 1 成功；0 失败
    
   \*/
    
   @Insert("insert into t\_user(user\_name, password, permission) " +
    
   "values (#{userName}, #{password}, #{permission}) ")
    
   int insertUser(User user);
    
    
   /\*\*
    
   \* 根据 uid 删除用户
    
   \* @param uid
    
   \* @return 1 成功；0 失败
    
   \*/
    
   @Delete("delete from t\_user where uid = #{uid}")
    
   int deleteById(int uid);
    
    
   /\*\*
    
   \* 修改用户信息
    
   \* @param user
    
   \* @return 1 成功；0 失败
    
   \*/
    
   @Update("update t\_user set user\_name=#{userName}, password=#{password}, permission=#{permission} where uid=#{uid}")
    
   int updateUser(User user);
    
    
   /\*\*
    
   \* 根据 uid 查询用户
    
   \* @param userId
    
   \* @return 用户对象
    
   \*/
    
   @Select("select \* from t\_user where uid = #{userId}")
    
   @Result(column = "user\_name", property = "userName")
    
   User selectById(int userId);
    
    
   /\*\*
    
   \* 根据用户名和密码查询用户
    
   \* @param userName
    
   \* @param password
    
   \* @return 用户对象
    
   \*/
    
   @Select("select \* from t\_user where user\_name = #{uname} and password = #{pwd}")
    
   @Result(column = "user\_name", property = "userName")
    
   User selectByUnameAndPwd(@Param("uname")String userName, @Param("pwd")String password);
    
    
   /\*\*
    
   \* 查询所有用户
    
   \* @return 用户对象列表
    
   \*/
    
   @Select("select \* from t\_user")
    
   @Results(@Result(column = "user\_name", property = "userName"))
    
   List<User> selectAll();
    
    
   /\*\*
    
   \* 根据用户名查询用户，支持模糊搜索
    
   \* @param userName
    
   \* @return 用户对象列表
    
   \*/
    
   @Select("select \* from t\_user where user\_name like concat('%', #{userName}, '%')")
    
   @Result(column = "user\_name", property = "userName")
    
   List<User> selectUsersByName(String userName);
    
    
   /\*\*
    
   \* 根据用户名查询单个用户
    
   \* @param userName
    
   \* @return 用户对象
    
   \*/
    
   @Select("select \* from t\_user where user\_name = #{userName}")
    
   @Result(column = "user\_name", property = "userName")
    
   User selectUserByName(String userName);
    
    
   @Select("select permission from t\_user where user\_name = #{userName}")
    
   int selectPermissionByName(String userName);
    
  }

1. **User 类** (User.java)：

* 作为 POJO，代表用户数据模型，包含用户属性和对应的 getter 和 setter 方法。
* package com.itheima.pojo;
    
    
  import lombok.AllArgsConstructor;
    
  import lombok.Data;
    
  import lombok.NoArgsConstructor;
    
    
  @Data
    
  @NoArgsConstructor
    
  @AllArgsConstructor
    
  public class User {
    
   private Integer uid;
    
   private String userName; // 用户名
    
   private String password; // 密码
    
   private Integer permission; // 权限 超管 0，普通管理员 1
    
  }

1. **UserServlet** (UserServlet.java)：

* 作为 Servlet，处理来自客户端的用户管理请求，调用 UserService 的方法来执行具体的用户管理操作。
* package com.itheima.service;
    
    
  import com.itheima.pojo.User;
    
    
  import java.util.List;
    
    
  public interface UserService {
    
   int save(User user);
    
    
   int delete(int uid);
    
    
   int update(User user);
    
    
   User getById(int pid);
    
    
   List<User> getAll();
    
    
   List<User> getByName(String userName);
    
    
   boolean isAdmin(String userName);
    
  }

1. **LoginCheckFilter 类** (LoginCheckFilter.java)：

* 过滤器，用于检查用户是否已经登录，如果是管理员，则允许访问用户管理页面。
* package com.itheima.filter;
    
    
    
  import com.itheima.config.SpringConfig;
    
  import com.itheima.service.AccountService;
    
  import com.itheima.service.UserService;
    
  import com.itheima.service.impl.AccountServiceImpl;
    
  import com.itheima.service.impl.UserServiceImpl;
    
  import org.springframework.context.ApplicationContext;
    
  import org.springframework.context.annotation.AnnotationConfigApplicationContext;
    
    
  import javax.servlet.FilterChain;
    
  import javax.servlet.ServletException;
    
  import javax.servlet.annotation.WebFilter;
    
  import javax.servlet.http.\*;
    
  import java.io.IOException;
    
    
  // 在登录的前提下才可进入下面三个路径
    
  @WebFilter({"/jsp/Manager\_page\_productList.jsp", "/jsp/Manager\_page\_userList.jsp", "/user", "/product"})
    
  public class LoginCheckFilter extends HttpFilter {
    
   ApplicationContext ctx=new AnnotationConfigApplicationContext(SpringConfig.class);
    
   UserService userService=ctx.getBean(UserService.class);
    
    
   @Override
    
   protected void doFilter(HttpServletRequest req, HttpServletResponse resp, FilterChain chain) throws IOException, ServletException {
    
   System.out.println("-------------------doFilter-------------------");
    
   for (Cookie cookie : req.getCookies()) {
    
   System.out.println(cookie.getName() + "=" + cookie.getValue());
    
   }
    
   System.out.println("-----------------------------------------------");
    
   // 若 cookies 中的登录信息有效就直接用该信息自动登录
    
    
   HttpSession session = req.getSession(); // 获取会话
    
   String userName = (String) session.getAttribute("userName"); // 从会话域获取用户名
    
   // 从会话域中获取 userName，若获取到对应用户名代表已登录
    
   if(session != null && userName != null) {
    
   String servletPath = req.getServletPath();
    
   System.out.println(session.getAttribute("userName") + " logged in, accessing " + servletPath);
    
   // 如果想进入用户管理页面或就行用户管理需要判断权限
    
   if("/jsp/Manager\_page\_userList.jsp".equals(servletPath) || servletPath.contains("/user/")) {
    
   if(userService.isAdmin(userName)) {
    
   chain.doFilter(req, resp); // 是 admin 放行
    
   } else { // 权限不足
    
   String ctxPath = req.getServletContext().getContextPath();
    
   resp.sendRedirect(ctxPath + "/jsp/permissionErr.jsp");
    
   }
    
   } else { // 若不是访问 userList.jsp 不做处理，直接放行
    
   chain.doFilter(req, resp);
    
   }
    
   } else { // 未登录，跳转到登陆页面
    
   String ctxPath = req.getServletContext().getContextPath();
    
   resp.sendRedirect(ctxPath + "/jsp/login.jsp");
    
   }
    
   System.out.println("filter over");
    
   }
    
    
    
   static void autoLoginByCookie(HttpServletRequest req) {
    
   System.out.println("Using cookie");
    
   Cookie[] cookies = req.getCookies();
    
   String userName = null;
    
   String password = null;
    
   if (cookies != null) {
    
   for (Cookie cookie : cookies) {
    
   if("userName".equals(cookie.getName())) {
    
   userName = cookie.getValue();
    
   } else if("userPwd".equals(cookie.getName())) {
    
   password = cookie.getValue();
    
   }
    
   }
    
   }
    
   ApplicationContext ctx=new AnnotationConfigApplicationContext(SpringConfig.class);
    
   AccountService accountService=ctx.getBean(AccountService.class);
    
   if(accountService.login(userName, password)) {
    
   HttpSession session = req.getSession(); // 获取会话
    
   session.setAttribute("userName", userName); // 将 userName 存入会话域
    
   }
    
   }
    
  }

这些文件和组件共同构成了用户管理模块的架构，它们协同工作以实现用户的增删改查等功能。管理员通过前端页面进行操作，请求被 UserServlet 捕获并处理，然后调用 UserService 和 UserDao 来执行具体的数据库操作。过滤器 LoginCheckFilter 确保只有经过身份验证且具有适当权限的用户才能访问用户管理功能。

# 5. 前端主要功能模块设计与实现

## 商品列表展示



<%@ page import="java.util.List" %>
  
<%@ page import="com.itheima.service.impl.ProductServiceImpl" %>
  
<%@ page import="com.itheima.pojo.Product" %>
  
<%@ page import="org.springframework.context.ApplicationContext" %>
  
<%@ page import="org.springframework.context.annotation.AnnotationConfigApplicationContext" %>
  
<%@ page import="com.itheima.config.SpringConfig" %>
  
<%@ page import="com.itheima.service.ProductService" %>
  
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
  
<!DOCTYPE html>
  
<html lang="en">
  
  
<head>
  
 <meta charset="UTF-8">
  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
  
 <title>后台管理</title>
  
 <link rel="stylesheet" href="../css/Manager.css">
  
 <link href="https://fonts.googlefonts.cn/css?family=Poppins" rel="stylesheet">
  
 <link rel="stylesheet" href="//at.alicdn.com/t/c/font\_4014732\_5xyegniznbv.css">
  
 <!-- <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'> -->
  
 <link rel="stylesheet" href="//at.alicdn.com/t/c/font\_4018417\_cvfx9zef9m8.css">
  
  
 <link rel="stylesheet" href="../css/Administrator\_list.css">
  
 <link rel="stylesheet" href="//at.alicdn.com/t/c/font\_4014732\_gv1qekurcl7.css">
  
 <link rel="stylesheet" href="../icon-font/font\_4014732\_pacxm9kefzo/iconfont.css">
  
 <style>
  
 input[type="search"] {
  
 position: relative;
  
 top: 30px;
  
 left: 40px;
  
 }
  
 li.add-box {
  
 position: relative;
  
 top: 18px;
  
 left: 34px;
  
 }
  
 </style>
  
</head>
  
  
<body>
  
 <nav class="sidebar close">
  
 <header>
  
 <div class="image-text">
  
 <span class="image">
  
 <img src="../img/apple.png" alt="logo">
  
 </span>
  
  
 <div class="text header-text">
  
 <%-- 从会话中获取当前用户名 --%>
  
 <span class="name"><%=session.getAttribute("userName")%></span>
  
 <span class="profession">Web developer</span>
  
 </div>
  
 </div>
  
  
 <i class="iconfont icon-arrow-right-bold toggle"></i>
  
 </header>
  
  
 <div class="menu-bar">
  
 <div class="menu">
  
 <li class="search-box">
  
 <i class="iconfont icon-search icon"></i>
  
 <form id="searchForm" action="<%=application.getContextPath()%>/jsp/Manager\_page\_searchProduct.jsp">
  
 <input type="search" placeholder="Search..." name="name">
  
 </form>
  
 </li>
  
 <ul class="menu-links">
  
 <li class="nav-link">
  
 <a href="#" onclick="history.back()">
  
 <i class="iconfont icon-fanhuishangyiji icon"></i>
  
 <span class="text nav-text">返回</span>
  
 </a>
  
 </li>
  
 <li class="nav-link">
  
 <a href="<%=application.getContextPath()%>/jsp/Manager\_page\_productList.jsp">
  
 <i class="iconfont icon-shouye icon"></i>
  
 <span class="text nav-text">商品管理</span>
  
 </a>
  
 </li>
  
 <li class="nav-link">
  
 <a href="https://matrix.to/#/#general:mozilla.org">
  
 <i class="iconfont icon-yijianfankui icon"></i>
  
 <span class="text nav-text">加入群组</span>
  
 </a>
  
 </li>
  
 <li class="nav-link">
  
 <a href="#" onclick="alert('敬请期待')">
  
 <i class="iconfont icon-cangguan icon"></i>
  
 <span class="text nav-text">数据分析</span>
  
 </a>
  
 </li>
  
 <li class="nav-link">
  
 <a href="<%=application.getContextPath()%>/jsp/Manager\_page\_userList.jsp">
  
 <i class="iconfont icon-quanzi icon"></i>
  
 <span class="text nav-text">管理员</span>
  
 </a>
  
 </li>
  
<%-- <li class="nav-link">--%>
  
<%-- <a href="#" onclick="confirm('已经是最新版本')">--%>
  
<%-- <i class="iconfont icon-banben icon"></i>--%>
  
<%-- <span class="text nav-text">版本更新</span>--%>
  
<%-- </a>--%>
  
<%-- </li>--%>
  
 </ul>
  
 </div>
  
  
 <div class="bottom-content">
  
 <li class="">
  
 <a href="<%=application.getContextPath()%>/account/logout">
  
 <i class="iconfont icon-24gl-extractLeft icon"></i>
  
 <span class="text nav-text">离开</span>
  
 </a>
  
 </li>
  
  
 <li class="mode">
  
 <div class="moon-sun">
  
 <i class="iconfont icon-yueliang icon moon"></i>
  
 <i class="iconfont icon-taiyang-copy icon sun"></i>
  
 </div>
  
  
 <span class="mode-text text">Dark Mode</span>
  
  
 <div class="toggle-switch">
  
 <span class="switch"></span>
  
 </div>
  
 </li>
  
 </div>
  
 </div>
  
 </nav>
  
  
 <!-- 注意：引入其他网页放置在这个section标签里添加引入代码 -->
  
 <section class="main-home">
  
 <div class="top-box">
  
 <li class="search-box">
  
 <form action="<%=application.getContextPath()%>/jsp/Manager\_page\_searchProduct.jsp">
  
 <input type="search" name="name" placeholder="Search...">
  
 <button type="submit"> <!--放大镜-->
  
 <i class="iconfont icon">&#xe610;</i>
  
 </button>
  
 </form>
  
 </li>
  
 <li class="add-box">
  
 <a href="<%=application.getContextPath()%>/jsp/Manager\_page\_addProduct.jsp"><!--增加-->
  
 <i class="iconfont icon-pinleizengjia\_o icon"></i>
  
 </a>
  
 </li>
  
<%-- <li class="choose-box">--%>
  
<%-- <a href="#"><!--筛选-->--%>
  
<%-- <i class="iconfont icon-shaixuan icon"></i>--%>
  
<%-- </a>--%>
  
<%-- </li>--%>
  
 </div>
  
 <div class="table-box">
  
 <table>
  
 <tbody>
  
 <tr>
  
 <th>pid</th>
  
 <th>商品名称</th>
  
 <th>商品描述</th>
  
 <th>商品类别</th>
  
 <th>库存数</th>
  
 <th>操作</th>
  
 </tr>
  
 <%
  
 ApplicationContext ctx=new AnnotationConfigApplicationContext(SpringConfig.class);
  
 ProductService productService=ctx.getBean(ProductService.class);
  
 List<Product> products = productService.getAll();
  
 for(int i = 0; i < products.size(); i++) {
  
 %>
  
 <tr>
  
 <td><%=products.get(i).getPid()%></td>
  
 <td><%=products.get(i).getProductName()%></td>
  
 <td><%=products.get(i).getProductDesc()%></td>
  
 <td><%=products.get(i).getProductCat()%></td>
  
 <td><%=products.get(i).getProductCount()%></td>
  
 <td>
  
 <a href="<%=application.getContextPath()%>/jsp/Manager\_page\_changeProduct.jsp?id=<%=products.get(i).getPid()%>">
  
 <i class="iconfont icon-xiugai"></i></a>
  
 &emsp;<span>|</span>&emsp;
  
 <a href="<%=application.getContextPath()%>/product/delete?id=<%=products.get(i).getPid()%>">
  
 <i class="iconfont icon-shanchu"></i>
  
 </a>
  
 </td>
  
 </tr>
  
 <%}%>
  
 </tbody>
  
 </table>
  
 </div>
  
 </section>
  
  
 <script src="../js/script.js"></script>
  
</body>
  
  
</html>

## 用户列表展示



<%@ page import="java.util.List" %>
  
<%@ page import="com.itheima.service.UserService" %>
  
<%@ page import="com.itheima.service.impl.UserServiceImpl" %>
  
<%@ page import="com.itheima.pojo.User" %>
  
<%@ page import="com.itheima.config.SpringConfig" %>
  
<%@ page import="org.springframework.context.annotation.AnnotationConfigApplicationContext" %>
  
<%@ page import="org.springframework.context.ApplicationContext" %>
  
  
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
  
<!DOCTYPE html>
  
<html lang="en">
  
  
<head>
  
 <meta charset="UTF-8">
  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
  
 <title>后台管理</title>
  
 <link rel="stylesheet" href="../css/Manager.css">
  
 <link href="https://fonts.googlefonts.cn/css?family=Poppins" rel="stylesheet">
  
 <link rel="stylesheet" href="//at.alicdn.com/t/c/font\_4014732\_5xyegniznbv.css">
  
 <!-- <link href='https://unpkg.com/boxicons@2.1.4/css/boxicons.min.css' rel='stylesheet'> -->
  
 <link rel="stylesheet" href="//at.alicdn.com/t/c/font\_4018417\_cvfx9zef9m8.css">
  
  
 <link rel="stylesheet" href="../css/Administrator\_list.css">
  
 <link rel="stylesheet" href="//at.alicdn.com/t/c/font\_4014732\_gv1qekurcl7.css">
  
 <link rel="stylesheet" href="../icon-font/font\_4014732\_pacxm9kefzo/iconfont.css">
  
</head>
  
  
<body>
  
 <nav class="sidebar close">
  
 <header>
  
 <div class="image-text">
  
 <span class="image">
  
 <img src="../img/apple.png" alt="logo">
  
 </span>
  
  
 <div class="text header-text">
  
 <%-- 从会话中获取当前用户名 --%>
  
 <span class="name"><%=session.getAttribute("userName")%></span>
  
 <span class="profession">Web developer</span>
  
 </div>
  
 </div>
  
  
 <i class="iconfont icon-arrow-right-bold toggle"></i>
  
 </header>
  
  
 <div class="menu-bar">
  
 <div class="menu">
  
 <li class="search-box">
  
 <i class="iconfont icon-search icon"></i>
  
 <form id="searchForm" action="<%=application.getContextPath()%>/jsp/Manager\_page\_searchUser.jsp">
  
 <input type="search" placeholder="Search..." name="name">
  
 </form>
  
 </li>
  
 <ul class="menu-links">
  
 <li class="nav-link">
  
 <a href="#" onclick="history.back()">
  
 <i class="iconfont icon-fanhuishangyiji icon"></i>
  
 <span class="text nav-text">返回</span>
  
 </a>
  
 </li>
  
 <li class="nav-link">
  
 <a href="<%=application.getContextPath()%>/jsp/Manager\_page\_productList.jsp">
  
 <i class="iconfont icon-shouye icon"></i>
  
 <span class="text nav-text">商品管理</span>
  
 </a>
  
 </li>
  
 <li class="nav-link">
  
 <a href="https://matrix.to/#/#general:mozilla.org">
  
 <i class="iconfont icon-yijianfankui icon"></i>
  
 <span class="text nav-text">加入群组</span>
  
 </a>
  
 </li>
  
 <li class="nav-link">
  
 <a href="#" onclick="alert('敬请期待')">
  
 <i class="iconfont icon-cangguan icon"></i>
  
 <span class="text nav-text">数据分析</span>
  
 </a>
  
 </li>
  
 <li class="nav-link">
  
 <a href="#">
  
 <i class="iconfont icon-quanzi icon"></i>
  
 <span class="text nav-text">管理员</span>
  
 </a>
  
 </li>
  
<%-- <li class="nav-link">--%>
  
<%-- <a href="#">--%>
  
<%-- <i class="iconfont icon-banben icon"></i>--%>
  
<%-- <span class="text nav-text">版本更新</span>--%>
  
<%-- </a>--%>
  
<%-- </li>--%>
  
 </ul>
  
 </div>
  
  
 <div class="bottom-content">
  
 <li class="">
  
 <a href="<%=application.getContextPath()%>/account/logout">
  
 <i class="iconfont icon-24gl-extractLeft icon"></i>
  
 <span class="text nav-text">离开</span>
  
 </a>
  
 </li>
  
  
 <li class="mode">
  
 <div class="moon-sun">
  
 <i class="iconfont icon-yueliang icon moon"></i>
  
 <i class="iconfont icon-taiyang-copy icon sun"></i>
  
 </div>
  
  
 <span class="mode-text text">Dark Mode</span>
  
  
 <div class="toggle-switch">
  
 <span class="switch"></span>
  
 </div>
  
 </li>
  
 </div>
  
 </div>
  
 </nav>
  
  
 <!-- 注意：引入其他网页放置在这个section标签里添加引入代码 -->
  
 <section class="main-home">
  
 <div class="top-box">
  
 <li class="search-box">
  
 <form action="<%=application.getContextPath()%>/jsp/Manager\_page\_searchUser.jsp">
  
 <input type="search" name="name" placeholder="Search...">
  
 <button type="submit"> <!--放大镜-->
  
 <i class="iconfont icon">&#xe610;</i>
  
 </button>
  
 </form>
  
 </li>
  
 <li class="add-box">
  
 <a href="<%=application.getContextPath()%>/jsp/Manager\_page\_addUser.jsp"><!--增加-->
  
 <i class="iconfont icon-pinleizengjia\_o icon"></i>
  
 </a>
  
 </li>
  
<%-- <li class="choose-box">--%>
  
<%-- <a href="#"><!--筛选-->--%>
  
<%-- <i class="iconfont icon-shaixuan icon"></i>--%>
  
<%-- </a>--%>
  
<%-- </li>--%>
  
 </div>
  
 <div class="table-box">
  
 <table>
  
 <tbody>
  
 <tr>
  
 <th>uid</th>
  
 <th>用户名</th>
  
 <th>用户密码</th>
  
 <th>账户类型</th>
  
 <th>选项</th>
  
 </tr>
  
 <%
  
 ApplicationContext ctx=new AnnotationConfigApplicationContext(SpringConfig.class);
  
 UserService userService=ctx.getBean(UserService.class);
  
 List<User> users = userService.getAll();
  
 for(int i = 0; i < users.size(); i++) {
  
 %>
  
 <tr>
  
 <td><%=users.get(i).getUid()%></td>
  
 <td><%=users.get(i).getUserName()%></td>
  
 <td><%=users.get(i).getPassword()%></td>
  
 <td>
  
 <%if(users.get(i).getPermission() == 0) { // 如果权限是 0
  
 out.print("超级管理员");
  
 } else {
  
 out.print("仓管员");
  
 }%>
  
 </td>
  
 <td>
  
 <a href="<%=application.getContextPath()%>/jsp/Manager\_page\_changeUser.jsp?id=<%=users.get(i).getUid()%>">
  
 <i class="iconfont icon-xiugai"></i></a>
  
 &emsp;<span>|</span>&emsp;
  
 <a href="<%=application.getContextPath()%>/user/delete?id=<%=users.get(i).getUid()%>">
  
 <i class="iconfont icon-shanchu"></i>
  
 </a>
  
 </td>
  
 </tr>
  
 <%}%>
  
 </tbody>
  
 </table>
  
 </div>
  
 </section>
  
  
 <script src="../js/script.js"></script>
  
</body>
  
  
</html>

## 登录模块



<%@ page contentType="text/html;charset=UTF-8" language="java" %>
  
<!DOCTYPE html>
  
<html lang="en">
  
<head>
  
 <meta charset="UTF-8">
  
 <meta http-equiv="X-UA-Compatible" content="IE=edge">
  
 <meta name="viewport" content="width=device-width, initial-scale=1.0">
  
 <title>登录界面</title>
  
 <link rel="stylesheet" href="//at.alicdn.com/t/c/font\_4014732\_seb9wnhm9ap.css">
  
 <link rel="stylesheet" href="../css/login.css">
  
</head>
  
<body>
  
 <div class="top">
  
 <div class="logo">
  
 <a href="#">
  
 <h1>
  
 <span class="iconfont icon-bird"></span>
  
 </h1>
  
 </a>
  
 </div>
  
 <div class="help">
  
 <p>用户协议</p><span>|</span><p>隐私政策</p><span>|</span><p>帮助中心</p>
  
 </div>
  
 </div>
  
 <div class="login-box">
  
 <form action="<%=application.getContextPath()%>/account/login" method="post" name="login\_item">
  
 <div class="tit">
  
 <h2>Login in</h2>
  
 </div>
  
 <div class="input-box">
  
 <label for="uname">用户名:</label>
  
 <input id="uname" type="text" name="userName" maxlength="16">
  
 </div>
  
 <div class="input-box">
  
 <label for="psw">密码:</label>
  
 <input id="psw" type="password" name="userPwd" maxlength="16">
  
 </div>
  
 <div class="btn-box" onclick="">
  
 <button type="button" onclick="submitForm()">登录</button>
  
 </div>
  
 <script>
  
 function submitForm() {
  
 const unameInput = document.getElementById("uname");
  
 const pswInput = document.getElementById("psw");
  
 const uname = unameInput.value;
  
 const psw = pswInput.value;
  
 if (uname.length == 0 || psw.length == 0) {
  
 alert("用户名和密码不能为空");
  
 return;
  
 }
  
  
 // 表单提交逻辑
  
 const form = document.forms["login\_item"];
  
 form.submit();
  
 }
  
 </script>
  
 </form>
  
 </div>
  
 <div class="signin">
  
 <a href="<%=application.getContextPath()%>/jsp/signup.jsp">
  
 <p>前往注册</p>
  
 </a>
  
 </div>
  
</body>
  
</html>