# 基于SSM的网上商城系统设计与实现

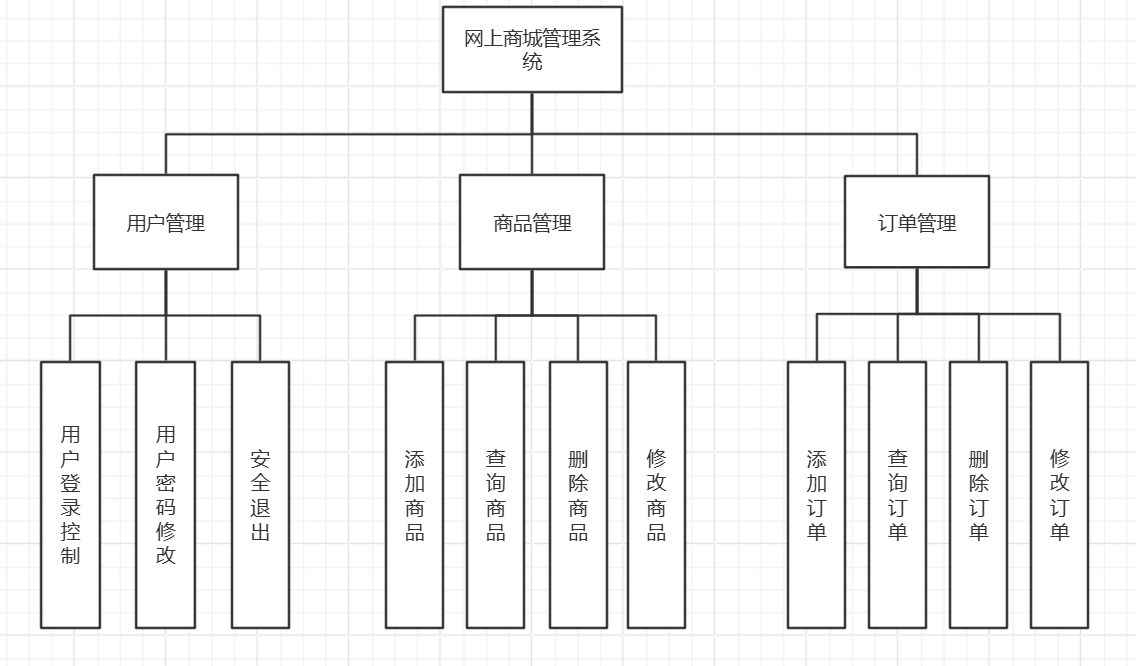
## 系统设计

### 1.1 系统功能需求

网上商城管理系统是一套针对用户网上购物的管理系统。系统提供的功能如下:

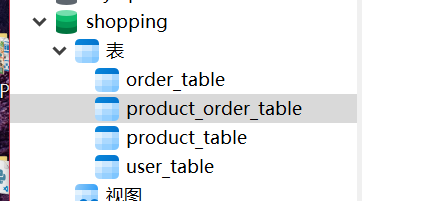
* 用户管理，用户只有注册后才能登录系统，每个用户可以查看相关的信息。
* 商品管理，用户可以添加、修改、删除以及浏览相应的商品信息。
* 订单管理，用户可以添加、修改、删除以及浏览自己的订单信息。
* 成功登录的用户，可以修改密码。

### 1.2 系统模块划分



## 2. 数据库设计

### 2.1 数据库概念结构设计



User\_table保存了用户信息,包括用户名字,密码;

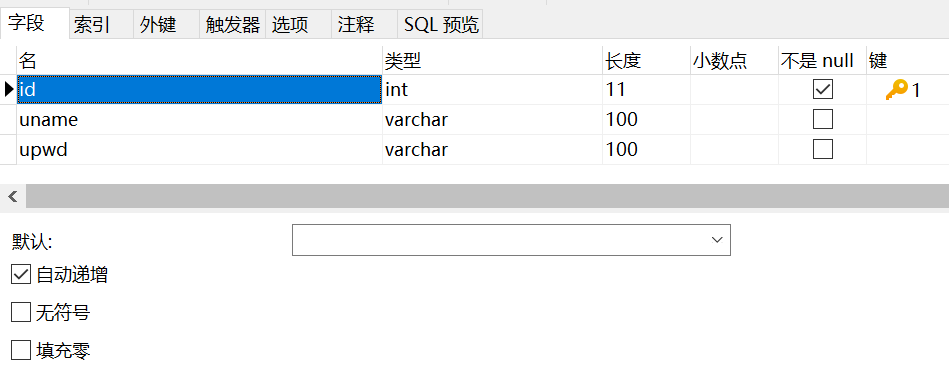
Product\_table保存商品信息,包括商品名字,价格,折扣价格,库存,图片;

Order\_table保存了订单信息,包括订单编号,用户编号;

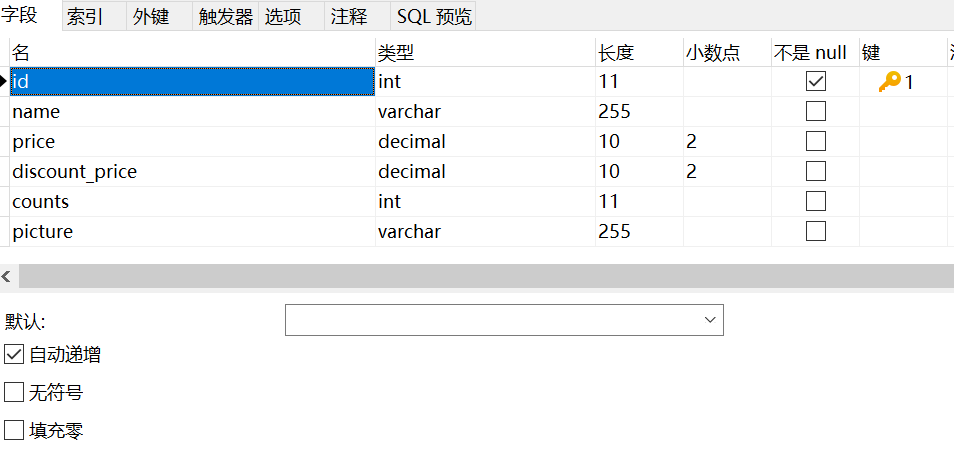
Product\_order\_table保存了订单详细信息,这张表用来存储订单和商品的关联关系,包括了订单编号和商品名字;

### 2.2 数据库逻辑结构设计

##### User\_table

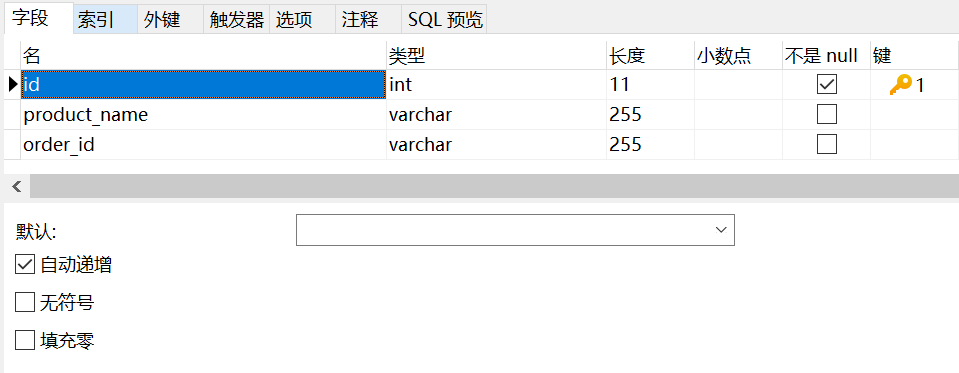


##### Product\_table



##### Order\_table

##### Product\_order\_table



## 3.子系统与模块设计

### 3.1用户管理

1.用户登录。输入用户名和密码后，系统将对用户名和密码进行验证。如果用户名和密码同时正确，则成功登录，如果用户名或密码有误,则提示错误。

2.用户注册。未注册的用户登录前需要注册一个新用户。注册的用户名如果存在会提示“已存在”。

3.用户密码修改。用户登录后可以修改自己的密码。

#### 3.1.1 Controller

*@Controller*

*@RequestMapping*("/user")

public class UserController {

*@Autowired*

private UserService userService;

*@RequestMapping*("/toLogin")

public String toLogin(*@ModelAttribute* User user) {

return "user\_LoginPage";

}

*@RequestMapping*("/login")

public String login(*@ModelAttribute* User user, Model model, HttpSession session) {

return userService.login(user, model, session);

}

*@RequestMapping*("/toRegister")

public String toRegister(*@ModelAttribute* User user) {

return "user\_RegistPage";

}

*@RequestMapping*("/register")

public String register(*@ModelAttribute* User user, Model model) {

return userService.register(user);

}

*@RequestMapping*("/loginOut")

public String loginOut(Model model, HttpSession session) {

return userService.loginOut(model, session);

}

*@RequestMapping*("/toMainPage")

public String toMainpage(*@ModelAttribute* User user) {

return "redirect:/user/mainpage?currentPage=1";

}

*@RequestMapping*("/mainpage")

public String mainpage(*@ModelAttribute* User user, Model model, int currentPage, HttpSession session) {

return userService.mainpage(user, model, currentPage, session);

}

*@RequestMapping*("/toUpdatePwdPage")

public String toUpdatePwd(*@ModelAttribute* User user,Model model, HttpSession session) {

UserTable myuser = (UserTable)session.getAttribute("userLogin");

model.addAttribute("update\_user", myuser);

return "user\_UpdatePwdPage";

}

*@RequestMapping*("/updatePwd")

public String updatePwd(*@ModelAttribute* User user,Model model, HttpSession session) {

return userService.updatePwd(user);

}

*@RequestMapping*("/checkUname")

*@ResponseBody*

public String checkUname(*@RequestBody* User user) {

return userService.checkUname(user);

}

}

#### 3.1.2 Service

*@Service*

public class UserServiceImpl implements UserService {

*@Autowired*

private UserMapper userMapper;

*@Autowired*

private ProductMapper productMapper;

*@Autowired*

private OrderMapper orderMapper;

*@Override*

public String checkUname(User user) {

List<UserTable> userList = userMapper.selectByUname(user);

if (userList.size() > 0)

return "no";

return "ok";

}

*@Override*

public String register(User user) {

if (userMapper.register(user) > 0)

return "user\_LoginPage";

return "user\_RegistPage";

}

*@Override*

public String login(User user, Model model, HttpSession session) {

List<UserTable> list = userMapper.login(user);

if (list.size() > 0) {

UserTable usertable = list.get(0);

session.setAttribute("userLogin", usertable);

// 获取商品

List<Map<String, Object>> allproducts = productMapper.selectAllProducts();

int totalCount = allproducts.size();

int currentPage = 1;

// 计算共多少页

int pageSize = 3;

int totalPage = (int) Math.*ceil*(totalCount \* 1.0 / pageSize);

List<Map<String, Object>> productsByPage = productMapper

.selectAllProductsByPage((currentPage - 1) \* pageSize, pageSize);

//

model.addAttribute("page\_products", productsByPage);

model.addAttribute("totalPage", totalPage);

model.addAttribute("currentPage", currentPage);

// 获取订单

List<OrderTable> allorder = orderMapper.getOrdersByUserID(usertable.getId());

model.addAttribute("order\_list", allorder);

model.addAttribute("order\_size", allorder.size());

return "user\_MainPage";

} else {

model.addAttribute("errorMessage", "用户名或密码错误！");

return "user\_LoginPage";

}

}

*@Override*

public String loginOut(Model model, HttpSession session) {

session.invalidate();

model.addAttribute("user", new User());

return "user\_LoginPage";

}

*@Override*

public String updatePwd(User user) {

user.setUpwd(user.getUpwd());

userMapper.updatePwd(user);

return "user\_LoginPage";

}

*@Override*

public String mainpage(User user, Model model, int currentPage, HttpSession session) {

UserTable myuser = (UserTable)session.getAttribute("userLogin");

// 获取商品

List<Map<String, Object>> allproducts = productMapper.selectAllProducts();

int totalCount = allproducts.size();

// 计算共多少页

int pageSize = 3;

int totalPage = (int) Math.*ceil*(totalCount \* 1.0 / pageSize);

List<Map<String, Object>> productsByPage = productMapper.selectAllProductsByPage((currentPage - 1) \* pageSize,

pageSize);

//

model.addAttribute("page\_products", productsByPage);

model.addAttribute("totalPage", totalPage);

model.addAttribute("currentPage", currentPage);

// 获取订单

List<OrderTable> allorder = orderMapper.getOrdersByUserID(myuser.getId());

model.addAttribute("order\_list", allorder);

model.addAttribute("order\_size", allorder.size());

return "user\_MainPage";

}

}

#### 3.1.3 数据访问

*@Repository*

public interface UserMapper {

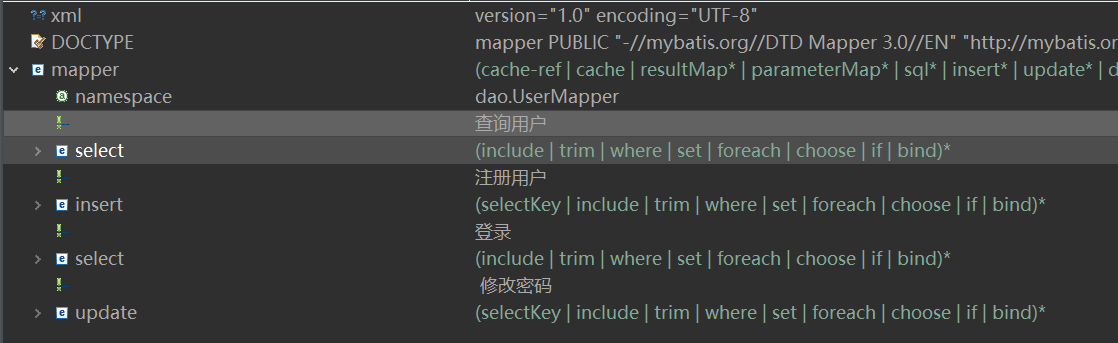
public List<UserTable> selectByUname(User myUser);

public int register(User myUser);

public List<UserTable> login(User myUser);

public int updatePwd(User myUser);

}



### 3.2商品管理

#### 3.2.1控制器

*@Controller*

*@RequestMapping*("/product")

public class ProductController {

*@Autowired*

private ProductService productService;

*@RequestMapping*("/toAddProduct")

public String toAddProduct(*@ModelAttribute* Product product) {

return "product\_addPage";

}

*@RequestMapping*("/addProduct")

public String addProduct(*@ModelAttribute* Product product, Model model, HttpServletRequest request, String act, HttpSession session) throws IllegalStateException, IOException {

return productService.addProduct(product, model, request, act, session);

}

*@RequestMapping*("/selectAllProductsByPage")

public String selectAllCardsByPage(Model model, int currentPage, HttpSession session) {

return productService.selectAllProductsByPage(model, currentPage, session);

}

*@RequestMapping*("/toDetail")

public String detail(Model model, HttpSession session, int id, String act) {

return productService.detail(model, session, id, act);

}

}

#### 3.2.2 Service

*@Service*

public class ProductServiceImpl implements ProductService {

*@Autowired*

private ProductMapper productMapper;

*@Override*

public String selectAllProductsByPage(Model model, int currentPage, HttpSession session) {

List<Map<String, Object>> allproducts = productMapper.selectAllProducts();

int totalCount = allproducts.size();

// 计算共多少页

int pageSize = 5;

int totalPage = (int) Math.*ceil*(totalCount \* 1.0 / pageSize);

List<Map<String, Object>> productsByPage = productMapper.selectAllProductsByPage((currentPage - 1) \* pageSize,

pageSize);

model.addAttribute("allProducts", productsByPage);

model.addAttribute("totalPage", totalPage);

model.addAttribute("currentPage", currentPage);

return "product\_MainPage";

}

*@Override*

public String addProduct(Product product, Model model, HttpServletRequest request, String act, HttpSession session)

throws IllegalStateException, IOException {

MultipartFile myfile = product.getPicture\_file();

// 如果选择了上传文件，将文件上传到指定的目录static/images

if (!myfile.isEmpty()) {

// 上传文件路径（生产环境）

String path = request.getServletContext().getRealPath("/static/images/");

// 获得上传文件原名

String fileName = myfile.getOriginalFilename();

// 对文件重命名

String fileNewName = MyUtil.*getNewFileName*(fileName);

File filePath = new File(path + File.***separator*** + fileNewName);

// 如果文件目录不存在，创建目录

if (!filePath.getParentFile().exists()) {

filePath.getParentFile().mkdirs();

}

// 将上传文件保存到一个目标文件中

myfile.transferTo(filePath);

// 将重命名后的图片名存到card对象中，添加时使用

product.setPicture(fileNewName);

}

// 确认是增加操作

if ("add".equals(act)) {

int n = productMapper.addProduct(product);

if (n > 0)// 成功

return "redirect:/product/selectAllProductsByPage?currentPage=1";

// 失败

return "product\_addPage";

} else {

int id = (int) session.getAttribute("product\_id");

product.setId(id);

// 修改

int n = productMapper.updateProduct(product);

if (n > 0)// 成功

return "redirect:/product/selectAllProductsByPage?currentPage=1";

// 失败

return "product\_updatePage";

}

}

*@Override*

public String detail(Model model, HttpSession session, int id, String act) {

ProductTable ct = productMapper.selectAProduct(id);

model.addAttribute("product", ct);

if ("detail".equals(act)) {

return "product\_DetailPage";

} else if ("update".equals(act)) {

session.setAttribute("product\_id", ct.getId());

return "product\_updatePage";

} else {

int n = productMapper.deleteAProduct(id);

if (n > 0) {

return "redirect:/product/selectAllProductsByPage?currentPage=1";

}

return "product\_MainPage";

}

}

}

#### 3.2.3 数据访问

*@Repository*

public interface ProductMapper {

public List<Map<String, Object>> selectAllProducts();

public List<Map<String, Object>> selectAllProductsByPage(*@Param*("startIndex") int startIndex, *@Param*("perPageSize") int perPageSize);

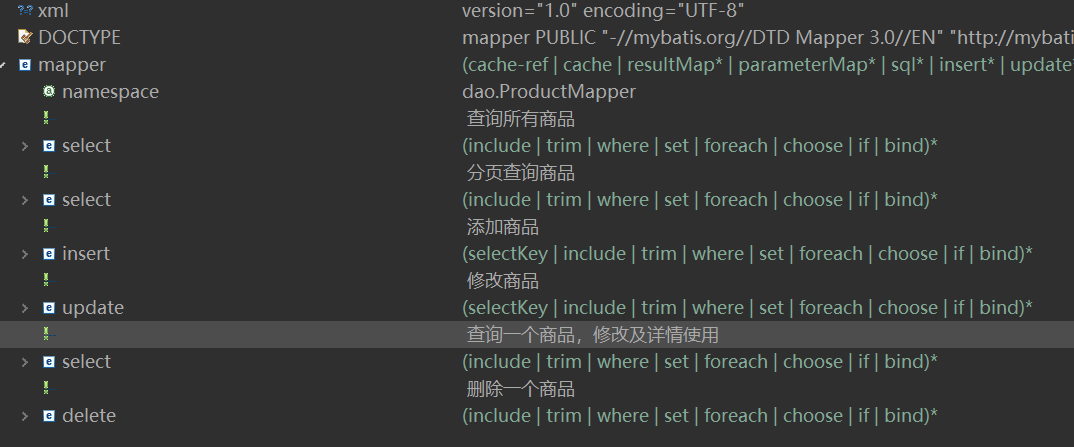
public int addProduct(Product product);

public int updateProduct(Product product);

public ProductTable selectAProduct(int id);

public int deleteAProduct(int id);

}



### 3.3 订单管理

#### 3.3.1 Controller

*@Controller*

*@RequestMapping*("/order")

public class OrderController {

*@Autowired*

private OrderService orderService;

*@RequestMapping*("/toOrderMainPage")

public String toOrderMainPage(Model model, HttpSession session) {

return orderService.selectAllOrders(model, session);

}

*@RequestMapping*("/toOrderAddPage")

public String toOrderAddPage(Model model, HttpSession session) {

return orderService.selectAllProducts(model, session);

}

*@RequestMapping*("/deleteOrder")

public String deleteOrder(Model model, HttpSession session, int id, String order\_id) {

return orderService.deleteOrder(model, session, id, order\_id);

}

*@PostMapping*("/createOrder")

public ResponseEntity<?> createOrder(*@RequestBody* OrderRequest orderRequest, Model model, HttpSession session) {

try {

List<String> selectedProducts = orderRequest.getProducts();

String orderId = orderService.createOrder(model,session, selectedProducts);

return ResponseEntity.*ok*(new OrderResponse(orderId));

} catch (Exception e) {

return ResponseEntity.*status*(*HttpStatus*.***INTERNAL\_SERVER\_ERROR***)

.body("创建订单时出错: " + e.getMessage());

}

}

*@RequestMapping*("/showDetail")

public String detail(Model model, HttpSession session, String order\_id) {

return orderService.showDetail(model, session, order\_id);

}

}

#### 3.3.2 Service

*@Service*

public class OrderServiceImpl implements OrderService {

*@Autowired*

private OrderMapper orderMapper;

*@Autowired*

private ProductMapper productMapper;

*@Autowired*

private ProductOrderMapper productOrderMapper;

*@Override*

public String selectAllOrders(Model model, HttpSession session) {

UserTable user = (UserTable)session.getAttribute("userLogin");

List<OrderTable> allorder = orderMapper.getOrdersByUserID(user.getId());

model.addAttribute("order\_list", allorder);

model.addAttribute("order\_size", allorder.size());

return "order\_MainPage";

}

*@Override*

public String selectAllProducts(Model model, HttpSession session)

{

List<Map<String, Object>> allproducts = productMapper.selectAllProducts();

model.addAttribute("allProducts", allproducts);

return "order\_add";

}

public String createOrder(Model model, HttpSession session, List<String> products) {

// 生成随机订单编号

String orderId = generateRandomOrderId();

UserTable user = (UserTable)session.getAttribute("userLogin");

Order new\_order = new Order();

new\_order.setOrder\_id(orderId);

new\_order.setUser\_id(user.getId());

orderMapper.addOrder(new\_order);

// 保存每个产品与订单编号的关联

for (String product : products) {

ProductOrder productOrder = new ProductOrder();

productOrder.setOrder\_id(orderId);

productOrder.setProduct\_name(product);

productOrderMapper.addProductOrder(productOrder);

}

return orderId; // 返回生成的订单编号

}

public String deleteOrder(Model model, HttpSession session, int id, String order\_id)

{

orderMapper.deleteAOrder(id);

productOrderMapper.deleteAOrder(order\_id);

return "redirect:/order/toOrderMainPage";

}

public String showDetail(Model model, HttpSession session, String order\_id)

{

List<ProductOrder> results = productOrderMapper.selectAllProductsByOrder(order\_id);

model.addAttribute("product\_order\_list", results);

return "order\_DetailPage";

}

private String generateRandomOrderId() {

return UUID.*randomUUID*().toString(); // 生成唯一标识符作为订单编号

}

}

#### 3.3.3 数据访问

商品数据查询

*@Repository*

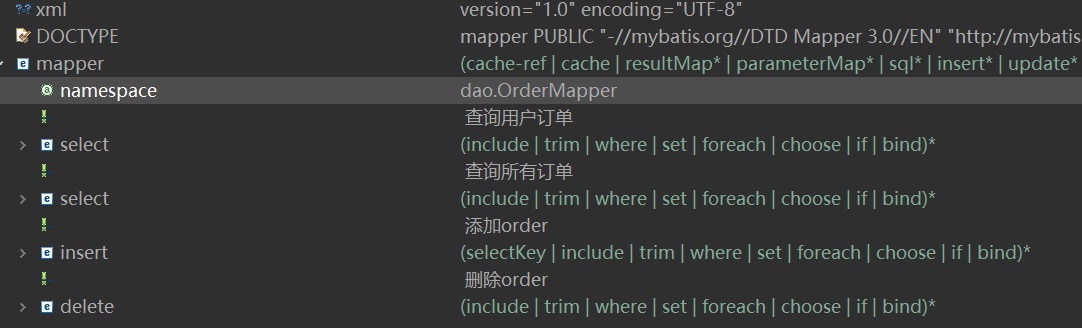
public interface OrderMapper {

public List<OrderTable> getOrdersByUserID(int user\_id);

public int addOrder(Order order);

public int deleteAOrder(int id);

}



商品和订单关联数据查询

*@Repository*

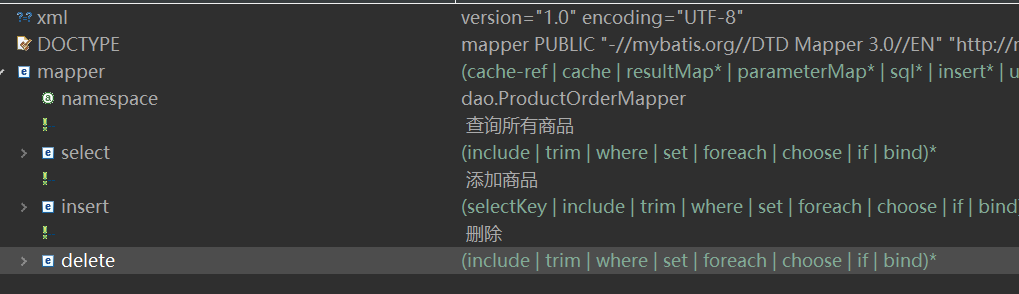
public interface ProductOrderMapper {

public List<ProductOrder> selectAllProductsByOrder(String order\_id);

public void addProductOrder(ProductOrder productOrder);

public void deleteAOrder(String order\_id);

}



## 4.系统演示截图

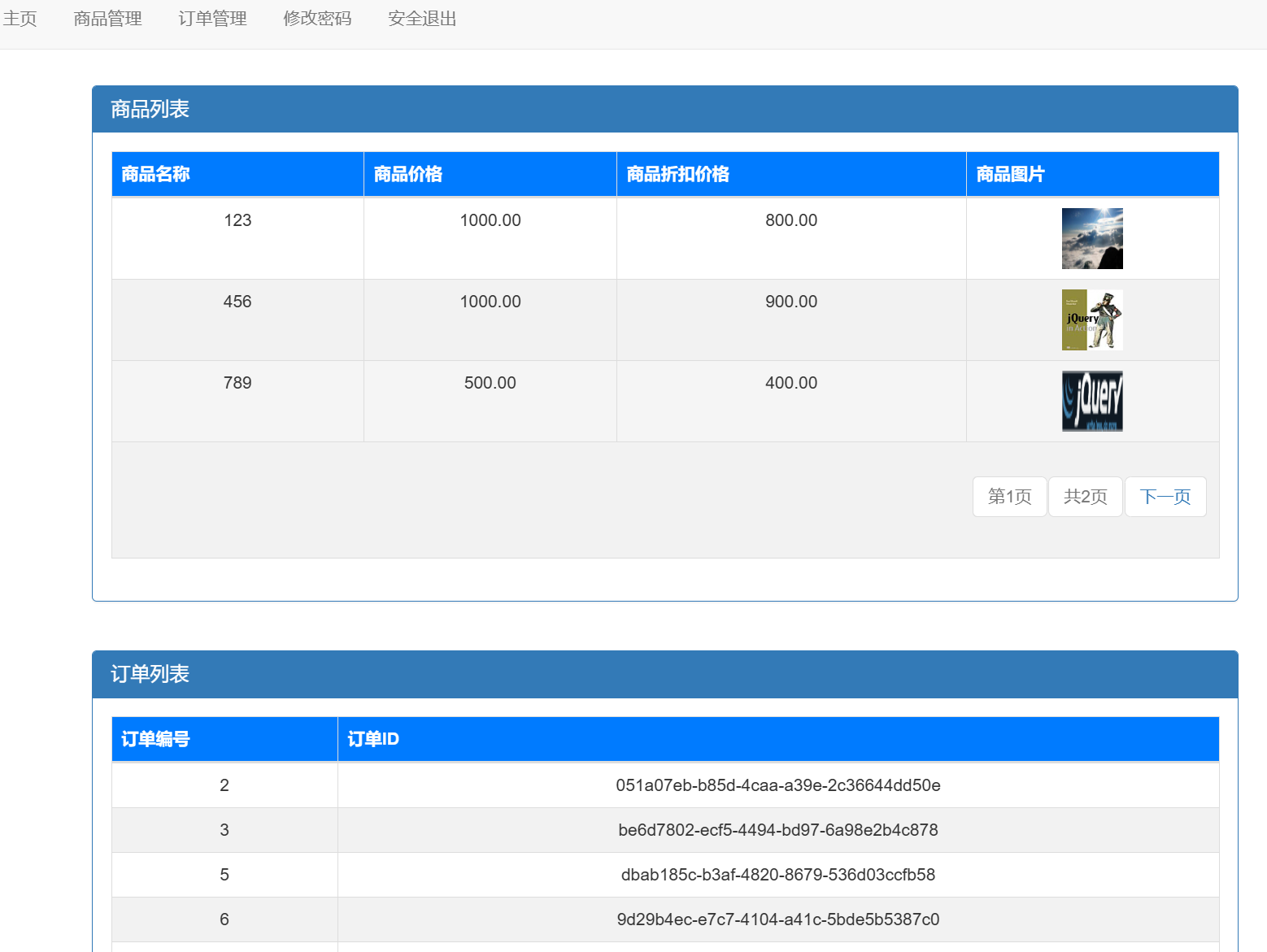
### 4.1用户登录演示



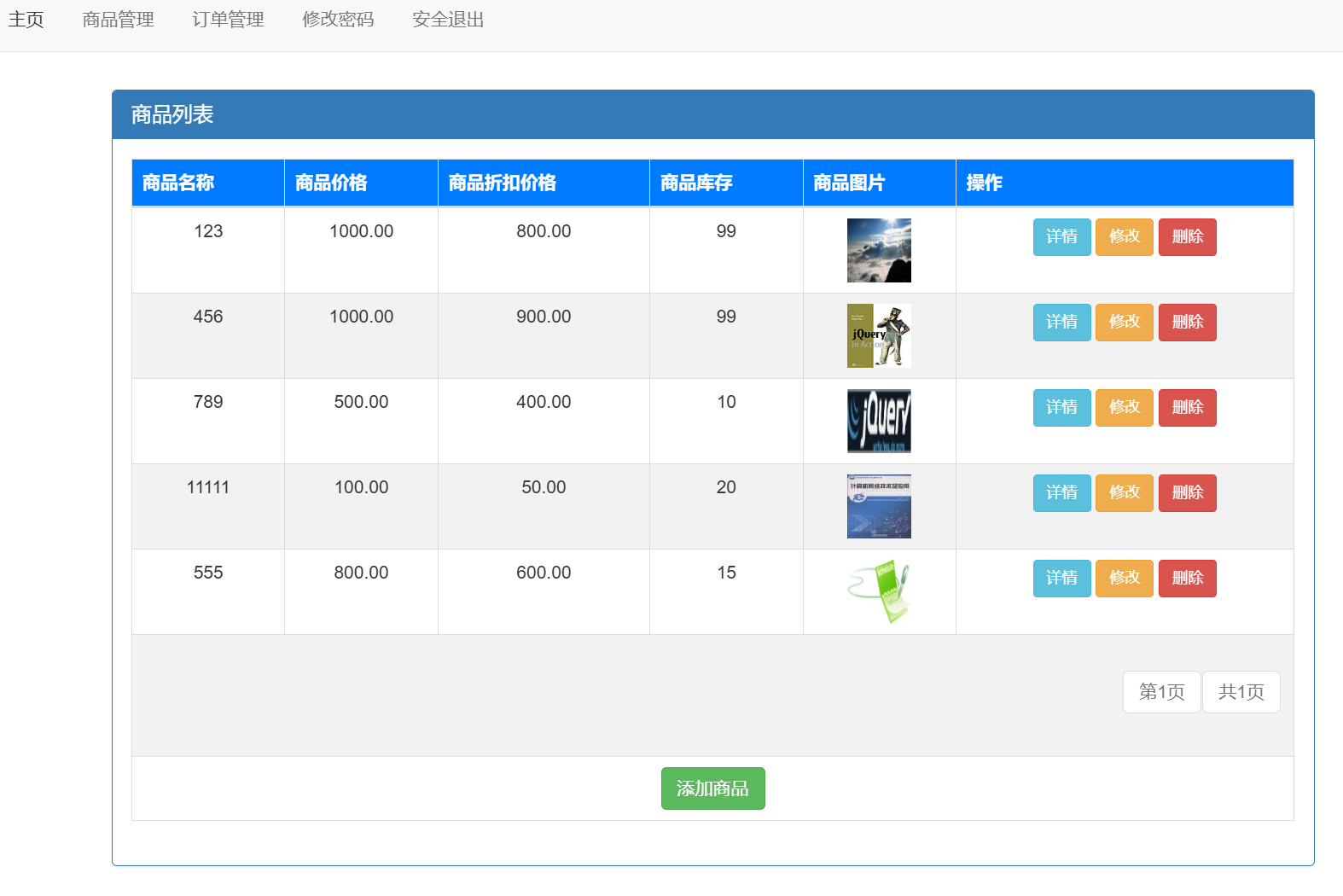
### 4.2用户注册演示



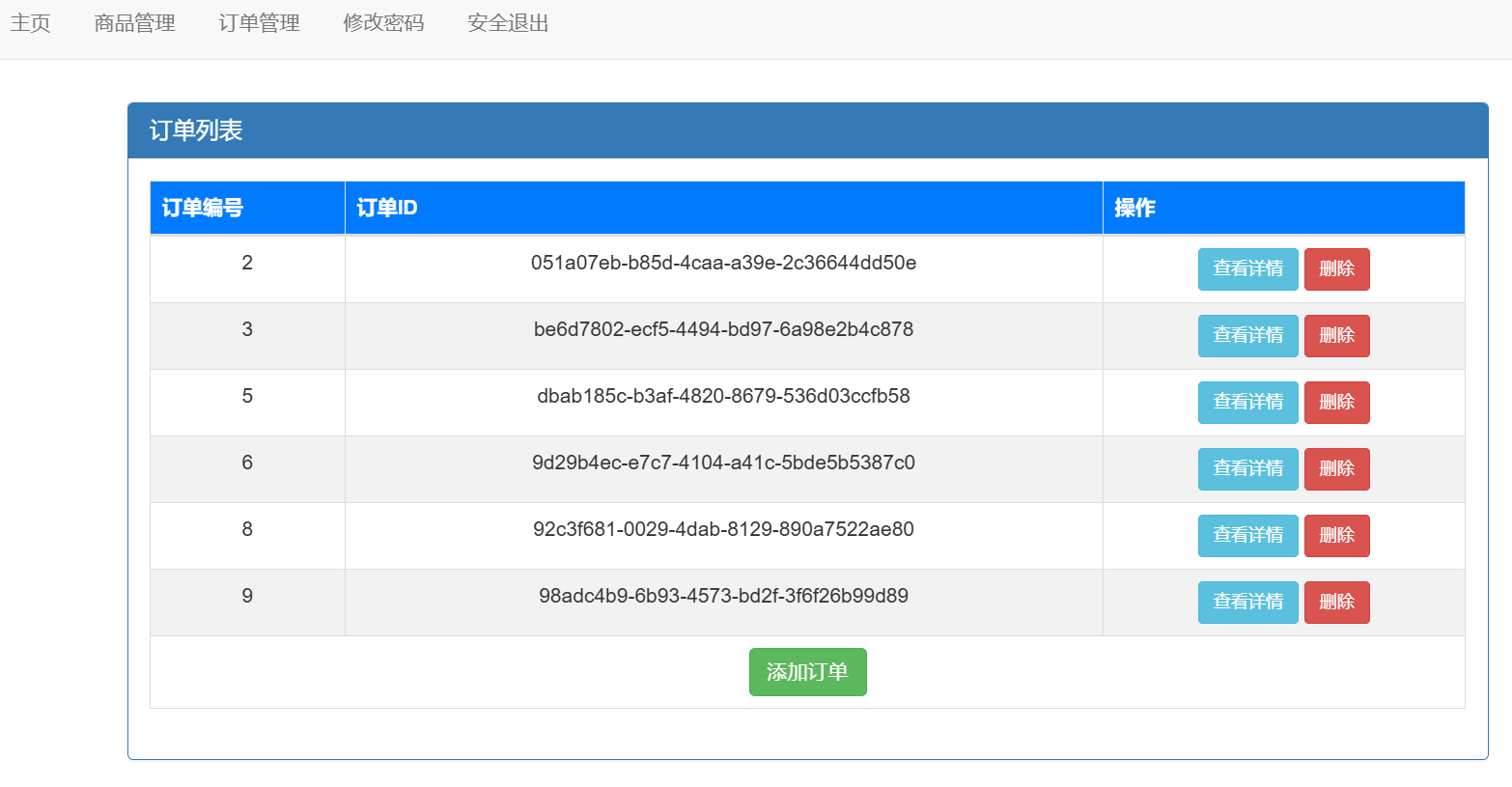
### 4.3 主页内容演示



### 4.4 商品管理界面演示



### 4.5 订单管理界面演示



### 4.6 用户密码修改界面演示

