## 回顾：

查询所有产品

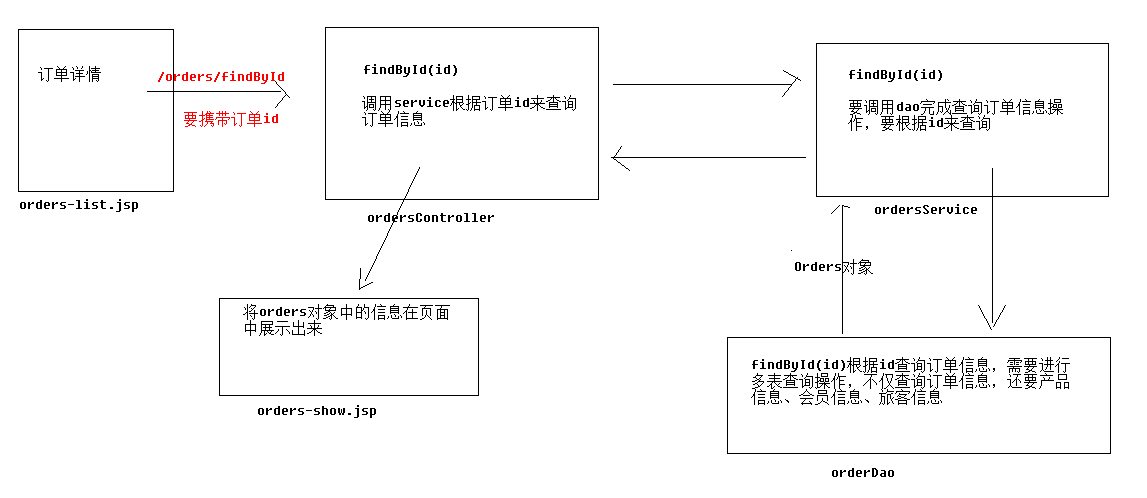
添加产品

查询所有订单

## 今日重点：

1. 订单详情：查询某个id的订单信息（多表连接）
2. **SpringSecurity框架！**

## 订单操作-订单详情查询流程分析



## 2.订单操作-订单详情查询代码实现1

问题？ 查询游客 多对多 many

Orders 属性 List<Traxxxxx>

订单 多对多 旅客 @Many

订单 一对一 产品 @One

订单 一对一 会员 @One

## 3.订单操作-订单详情查询代码实现2

一对一的查询：

One = @one(“select= …”)

一对多查询：

Many = @many(select = ….)

## 4.订单操作-订单详情查询代码实现3

@Select("select \* from traveller where id in (select travellerId from order\_traveller where orderId=#{ordersId})")

public List<Traveller> findByOrdersId(String ordersId) throws Exception;

总结：

1. 流程
2. Dao中sql语句的映射

订单 多对多 旅客 @Many

订单 一对一 产品 @One

订单 一对一 会员 @One

产品的新增，产品的查询，订单的查询分页，订单的详情

Main.jsp 前进行登录

## 5.权限操作-表结构分析与创建表

三张表：

权限表，角色表，用户表

多对多的关系

## 6.权限操作-springSecurity概述及快速入门步骤分析

SpringSecurity 框架是一个安全认证框架

认证登录！！！

配置

使用：

1. 导入依赖
2. Web.xml 配置filter
3. 核心配置 springsecurity.xml

## 7.权限操作-springSecurity快速入门

认证过程：帮我们做好了登陆拦截！！！

修改登陆的页面和跳转的路径！！！

个性化

## 8.权限操作-springSecurity快速入门-使用自定义页面

<security:http security="none" pattern="/login.html" />

<security:http security="none" pattern="/failer.html" />

配置自定义登陆页面，登陆失败页面：

<security:form-login login-page="/login.html"

login-processing-url="/login" username-parameter="username"

password-parameter="password" authentication-failure-url="/failer.html"

default-target-url="/success.html"

/>

## 9.用户操作-登录流程分析

查询数据库，登陆校验

特点：1.不需要controller

2.只需要写service,必须要实现UserDetailsService

public interface UserDetailsService {

调用dao,查询用户信息

**UserDetails** loadUserByUsername(String username) throws UsernameNotFoundException;

}

将dao查询的user对象变成**UserDetails对象！**

## 10.用户操作-登录代码实现1

<security:authentication-manager>

去spring容器中查找一个叫userService的service

<security:authentication-provider user-service-ref="userService">

<!-- 配置加密的方式

<security:password-encoder ref="passwordEncoder"/>

-->

</security:authentication-provider>

</security:authentication-manager>

在自己service工程中创建一个service

要求：实现UserDetailsService，名字

@Service(**"userService"**)

@Transactional

public class UserServiceImpl implements UserDetailsService{

@Autowired

private IUserDao userDao;

@Override

public UserDetails **loadUserByUsername**(String username) throws UsernameNotFoundException {

订单详情查询！

SpringSecurity 安全框架，权限方面的验证的

步骤：

1. 导入依赖
2. Web.xml 配置Filter 名字不能修改！
3. Spring-security.xml 核心配置

拷贝，修改！

## 11.用户操作-登录代码实现2

1.查询dao

2.userInfo 用户 与UserDetails对象关系？

UserDetails是接口，实现类 User,构造方法（用户名，密码）

public UserDetails **loadUserByUsername**(String username) throws UsernameNotFoundException {

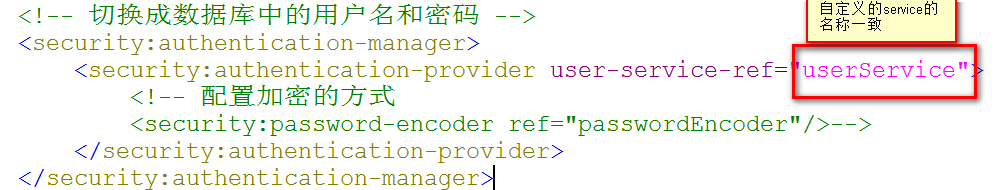
User user = new User(userInfo.getUserName() ,……, 角色集合);

Return user;

}

总结：springSecurity通过调用service—dao访问数据库校验登陆

1. 导入依赖
2. 配置web.xml 过滤器，过滤器名字不能改
3. 拷贝spring-security.xml



4.写一个service类，实现UserDetailsService接口

@Service(**"userService"**)

@Transactional

public class UserServiceImpl implements UserDetailsService{

@Autowired

private IUserDao userDao;

@Override

public UserDetails **loadUserByUsername**(String username) throws UsernameNotFoundException {

连接数据库查询校验用户名和密码

User user = new User(userInfo.getUserName() ,……, 角色集合);

Return user;

UserDetails 是user 父亲

}

## 12.用户操作-登录代码实现3

User user = new User(userInfo.getUserName() ,……, 角色集合);

1.关于角色的问题

List<SimpleGrantedAuthority> 角色的集合

new SimpleGrantedAuthority(角色的名称)

2.密码的问题

明文！MD5 加密算法 数据库中存储的密码都应该是加密后的密文！！！

做法："{noop}" + userInfo.getPassword()

## 13.用户操作-登录代码实现4

用户名，密码，账户状态一起验证登陆！

User user = new User(userInfo.getUsername(), "{noop}" + userInfo.getPassword(),

userInfo.getStatus() == 0 ? false : true, true, true, true, authoritys);

## 14.用户操作-退出

<security:logout invalidate-session="true" logout-url="/logout.do" logout-success-

url="/login.jsp" />

总结：实现登陆，实现退出

1. 导入依赖
2. Web.xml配置filter,名称不能改变

<filter>

<filter-name>springSecurityFilterChain</filter-name>

<filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>

</filter>

<filter-mapping>

<filter-name>springSecurityFilterChain</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

1. 配置spring-security.xml,配置service调用dao查询验证用户名密码

<security:authentication-manager>

<security:authentication-provider user-service-ref="userService">

1. 写业务层登陆实现，不需要controller, springsecurity框架自动调用service

@Service(**"userService"**)

@Transactional

public class UserServiceImpl implements UserDetailsService{

@Autowired

private IUserDao userDao;

@Override

public UserDetails **loadUserByUsername**(String username) throws UsernameNotFoundException {

调用dao,查询数据库，用户信息和角色

User user = new User(userInfo.getUsername(), "{noop}" + userInfo.getPassword(),

userInfo.getStatus() == 0 ? false : true, true, true, true,角色集合);

return user;

}

退出：/logout.do

## 15.springSecurity源码分析-springSecurityFilterChain

## 问题：springSecurityFilterChain为什么不能改变？

答案：因为SpringSecurity框架默认会从spring容器中读取一个springSecurityFilterChain名词的实例对象

springSecurityFilterChain名字的实例类型：

FilterChainProxy类型：加载所有的filter

## 16.springSecurity源码分析-spring-security.xml文件配置

1.加载配置文件的类：SecurityNamespaceHandler

Init 初始化

Parser方法解析xml文件