**Shiro的学习笔记**

## Shiro概述

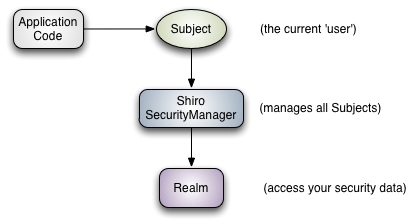
Shiro是一个用来做用户安全认证、授权、加密、会话的安全框架。

## Shiro详细功能

### 2.1 初步流程

1.应用代码ApplicationCode通过**subject**进行认证和授权，而**subject**又委托给**SecurityManager**；

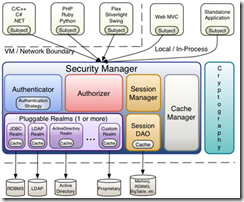
2.我们需要给shiro的**SecurityManager**注入Realm，从而让**SecurityManager**能得到合法的用户以及权限进行判断。



从以上也可以看出：**Shiro不提供维护用户/权限，而是通过Realm让开发人员自己注入。**

### 2.2 shiro的组件

Shiro的组件：



**Subject：**主体，可以看到主体可以是任何可以与应用交互的“用户”

**SecurityManager：**相当于SpringMVC中的DispatcherServlet或者Struts2中的FilterDispatcher；是Shiro的心脏；所有具体的交互都通过SecurityManager进行控制；它**管理着所有Subject、且负责进行认证和授权、及会话、缓存的管理。**实质上SecurityManager是通过Authenticator进行认证，通过Authorizer进行授权，通过SessionManager进行会话管理等。

**Authenticator：**认证器，负责主体认证的，这是一个扩展点，如果用户觉得Shiro默认的不好，可以自定义实现；其需要认证策略（Authentication Strategy），即什么情况下算用户认证通过了；

**Authrizer：**授权器，或者访问控制器，用来决定主体是否有权限进行相应的操作；即控制着用户能访问应用中的哪些功能；

**Realm：可以有1个或多个Realm，可以认为是安全实体数据源，即用于获取安全实体的**；可以是JDBC实现，也可以是LDAP实现，或者内存实现等等；由用户提供；注意：Shiro不知道你的用户/权限存储在哪及以何种格式存储；所以我们一般在应用中都需要实现自己的Realm；**securityManager进行安全认证需要通过Realm获取用户权限数据**，比如：如果用户身份数据在数据库，那么realm就需要从数据库获取用户身份信息。注意：**不要把realm理解成只是从数据源取数据，在realm中还有认证授权校验的相关的代码。**

**SessionDAO：**DAO大家都用过，数据访问对象，用于会话的CRUD，比如我们想把Session保存到数据库，那么可以实现自己的SessionDAO，通过如JDBC写到数据库；比如想把Session放到Memcached中，可以实现自己的Memcached SessionDAO；另外**SessionDAO中可以使用Cache进行缓存，以提高性能**。

**CacheManager：**缓存控制器，来管理如用户、角色、权限等的缓存的；因为这些数据基本上很少去改变，放到缓存中后可以提高访问的性能。

**Cryptography：**密码模块，Shiro提高了一些常见的加密组件用于如密码加密/解密的。

参考地址: https://www.cnblogs.com/flyuphigh/p/8058454.html

### 2.3 资源实例的权限字符串

权限字符串的规则是：**资源标识符：操作：资源实例标识符**，意思是**对哪个资源的哪个实例具有什么操作**，“:”是资源/操作/实例的分割符，权限字符串也可以使用\*通配符。

role73=user:\*:01 表达的是：对user资源的01实例拥有所有权限。

<https://blog.csdn.net/cold___play/article/details/104230574>

### 2.3 授权方式

#### 2.3.1基于角色的访问控制

RBAC基于角色的访问控制（Role-Based Access Control）是**以角色为中心进行访问控制**

if(subject.hasRole("admin")){  
   //操作什么资源  
}

#### 2.3.2**基于资源的访问控制**

RBAC基于资源的访问控制（Resource-Based Access Control）是**以资源为中心进行**访问控制

if(subject.isPermission("user:update:01")){ //资源实例  
  //对01用户进行修改  
}  
if(subject.isPermission("user:update:\*")){  //资源类型  
  //对01用户进行修改

### 2.4 授权方式

**编程式**

Subject subject = SecurityUtils.getSubject();  
if(subject.hasRole(“admin”)) {  
 //有权限  
} else {  
 //无权限  
}

**注解式**

@RequiresRoles("admin")  
public void hello() {  
 //有权限  
}

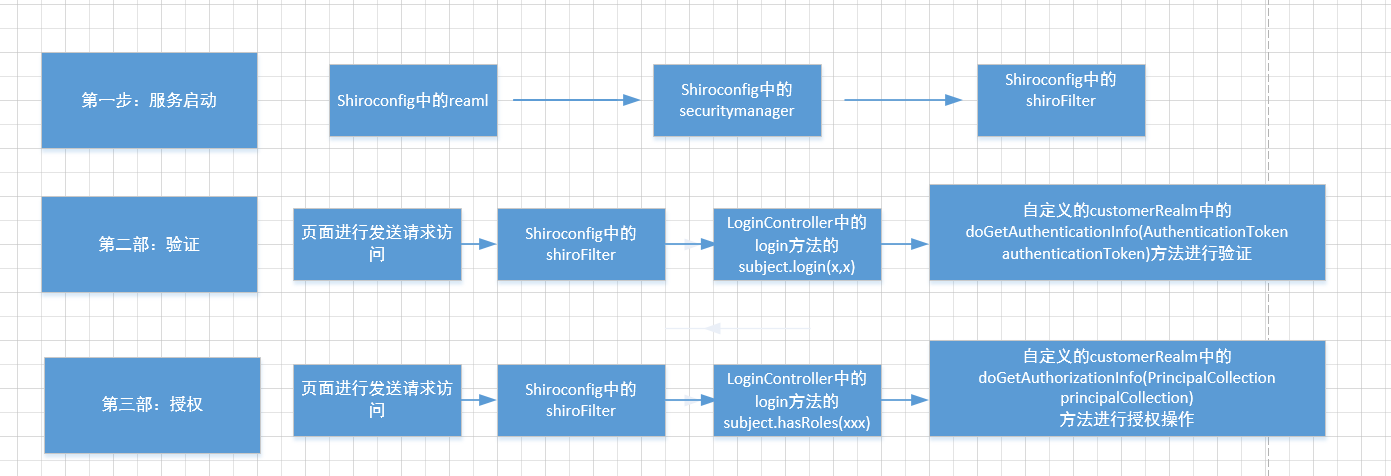
**标签式**

JSP/GSP 标签：在JSP/GSP 页面通过相应的标签完成：  
<shiro:hasRole name="admin">  
 <!— 有权限—>  
</shiro:hasRole>  
注意: Thymeleaf 中使用shiro需要额外集成!

### 2.5 MD5+SALT授权

Md5：  
1.不可逆  
2.相同的内容，经过MD5生成的加密串，结果也是一样的  
3.生成的是一个16进制，32位的字符串。  
注册： md5(password+“salt盐(随机数)”) 存储这个加密后的的密码，同时存储salt盐  
登录: 通过username查询到对应的salt盐，同时和输入明文password， 经过md5(password+salt) ，与之前注册存储到数据库中的密码进行对比。

### 2.6 shiro的执行认证与授权



### 2.7 shiro的执行认证

1.登录页面的地址：http://localhost:8888/shiro/user/login



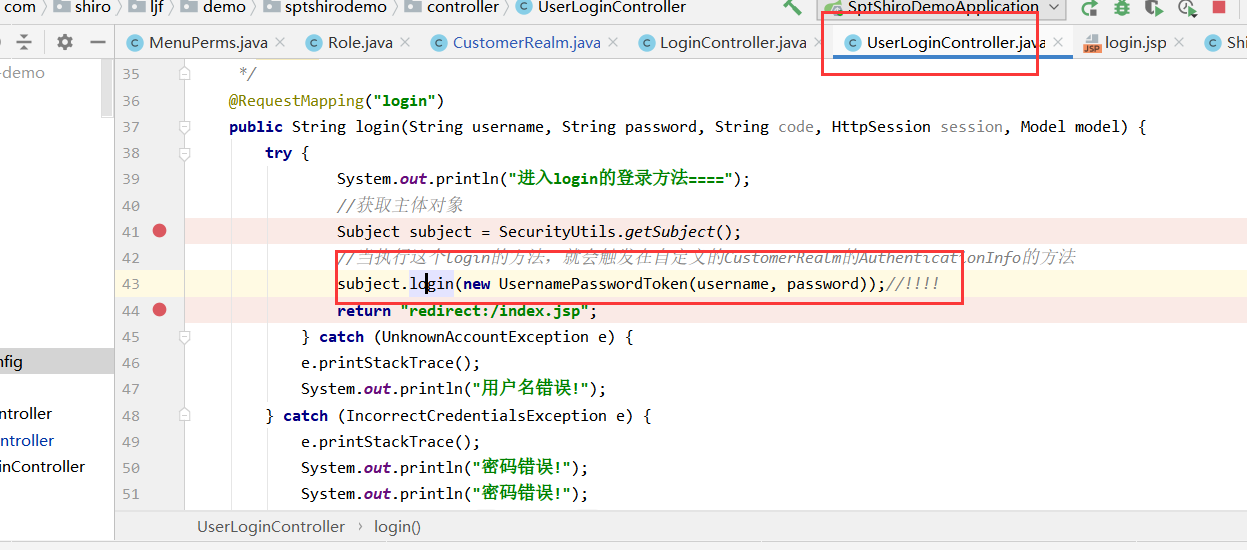
1. 在shiroconfig的filter中，对/user/login进行设置放行过滤器



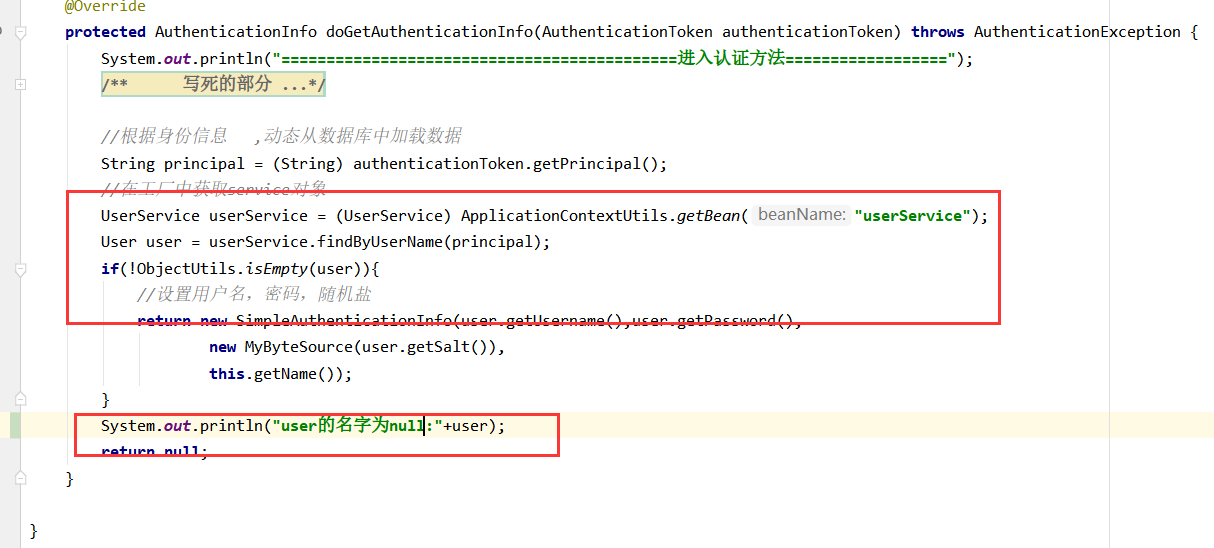
1. 直接在地址栏进行页面访问：



程序访问step1：进入UserLoginController类中的login方法



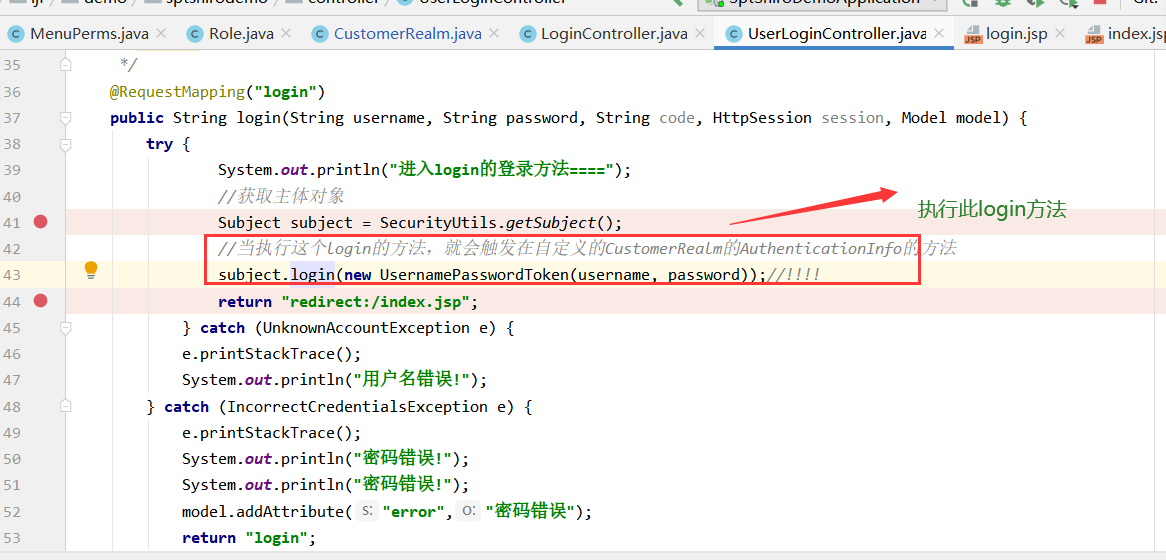
访问step2：执行subject.login方法会调用realm中的dogetAuthentication（）方法



Step3：查看日志：



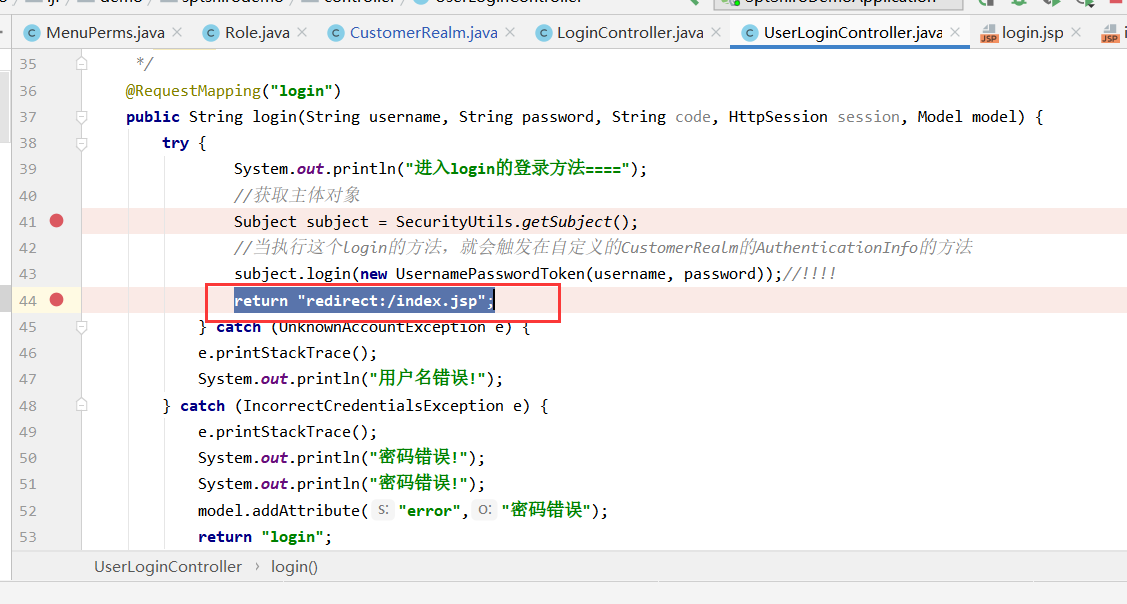
Step4：在登录页面，输入正确的用户名和密码，则再次跳转到进入UserLoginController类中的login方法



Step5：执行subject.login方法会调用realm中的dogetAuthentication（）方法



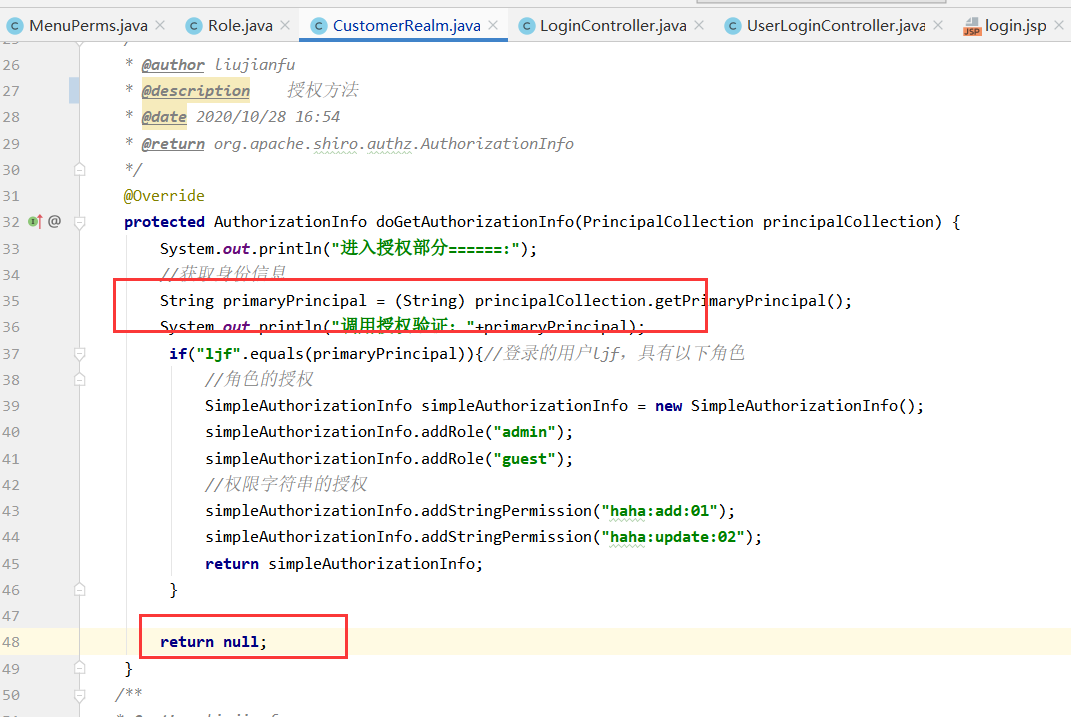
Step6：执行UserLoginControler中的subject.login方法后，执行return "redirect:/index.jsp";



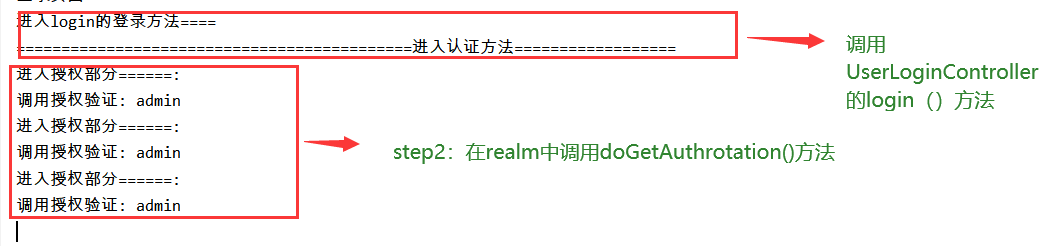
Step7：在index页面，执行hasAnyroles(),方法则调用reaml中的doGetAuthrization方法。



Step9：在reaml中的doGetAuthorizationInfo(PrincipalCollection principalCollection)方法进行授权，但是获取当前登录的用户名为admin，方法的if判断中，不等于“ljf”，则没有进入if块中，则直接返回null



Step9：console的控制台日志：



Step10：页面的显示：

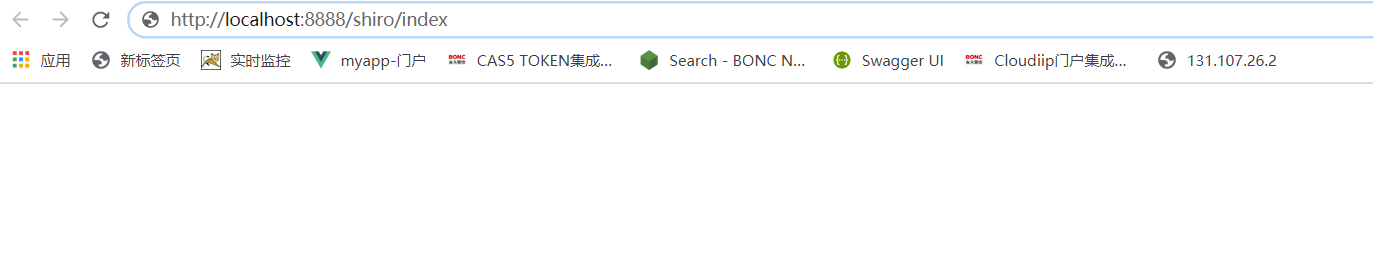


## Shiro的授权

### 3.1 shiro的授权

**####1.情况一：没有登录，直接访问，order/save方法时没有权限的，直接跳转到登录页面**

#访问：<http://localhost:8888/shiro/index>



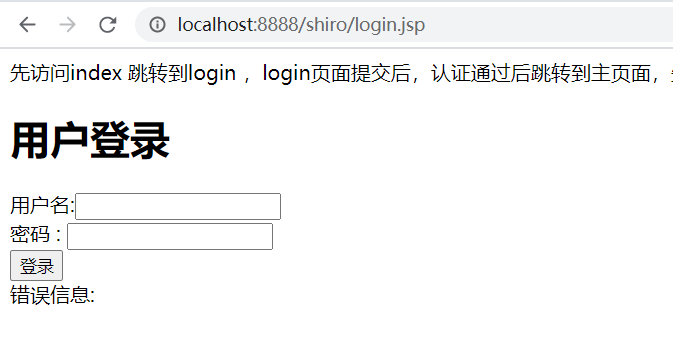
#访问后：



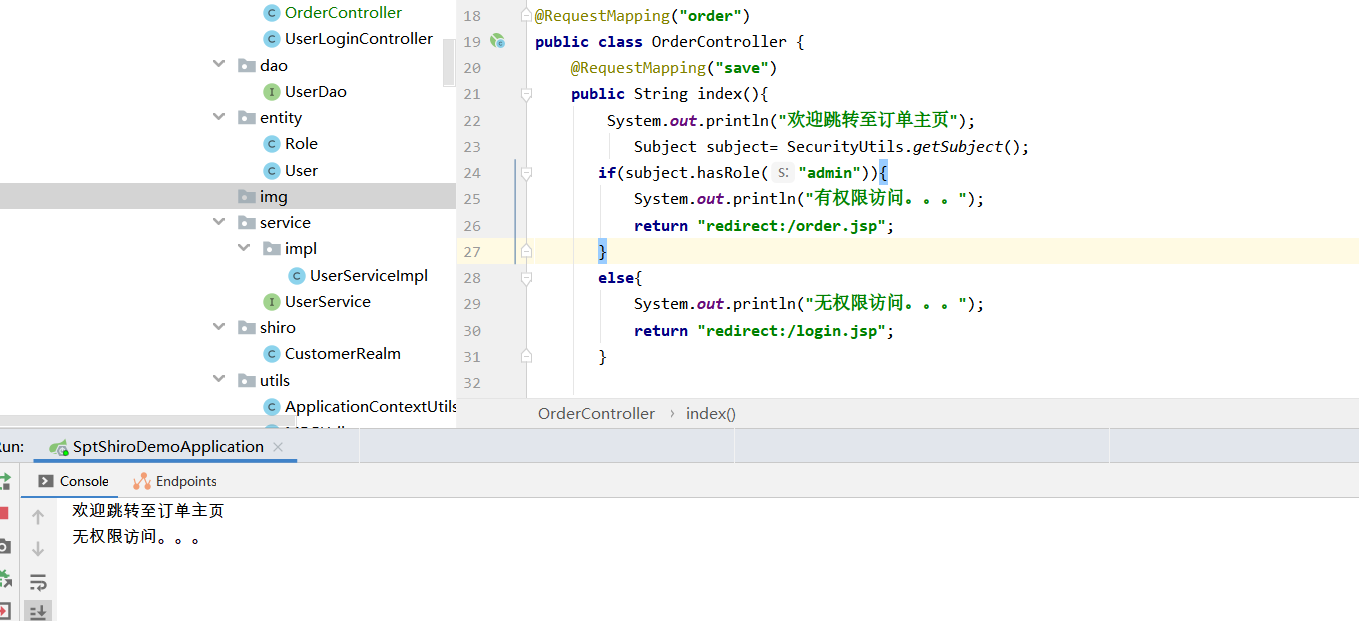
#新打开一个页面：http://localhost:8888/shiro/order/save



#访问后：



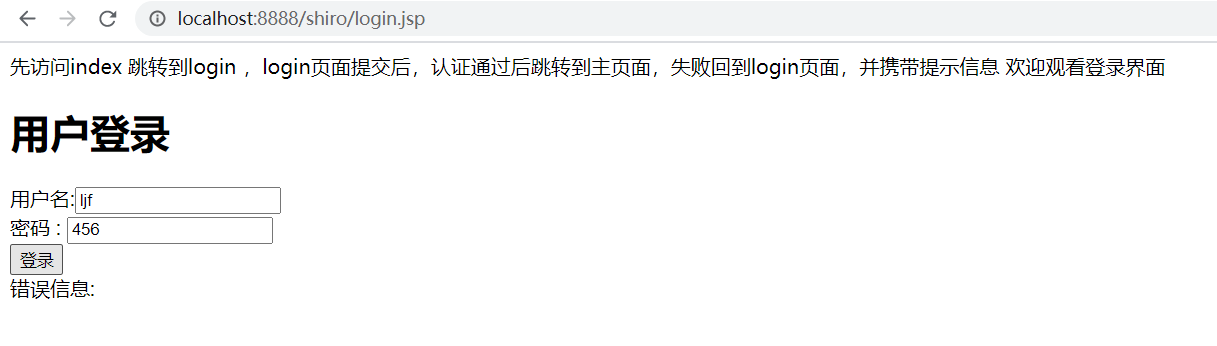
#查看log日志



可以看到：没有登录，直接访问，order/save方法时没有权限的，直接跳转到登录首页

**####1.情况二：用户登录后，直接访问，order/save方法时有权限的，直接跳转到成功的指定的页面**

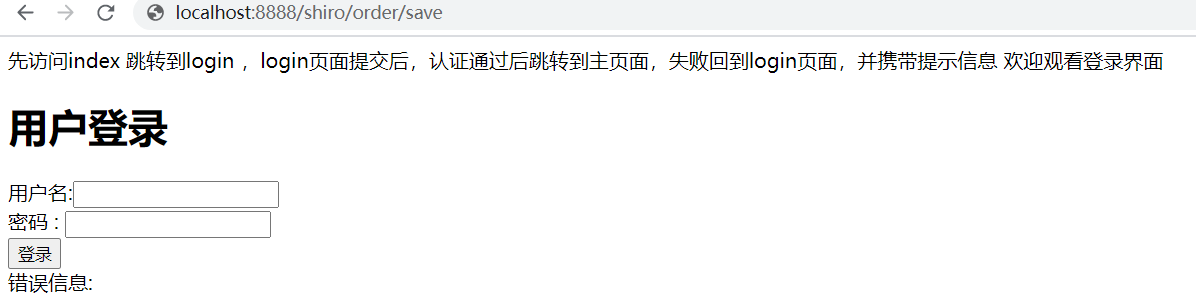
#开始登录



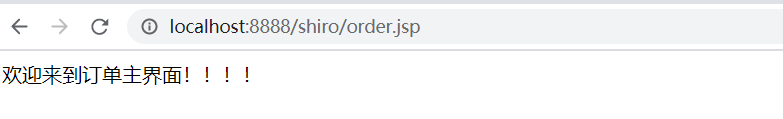
#登录后：



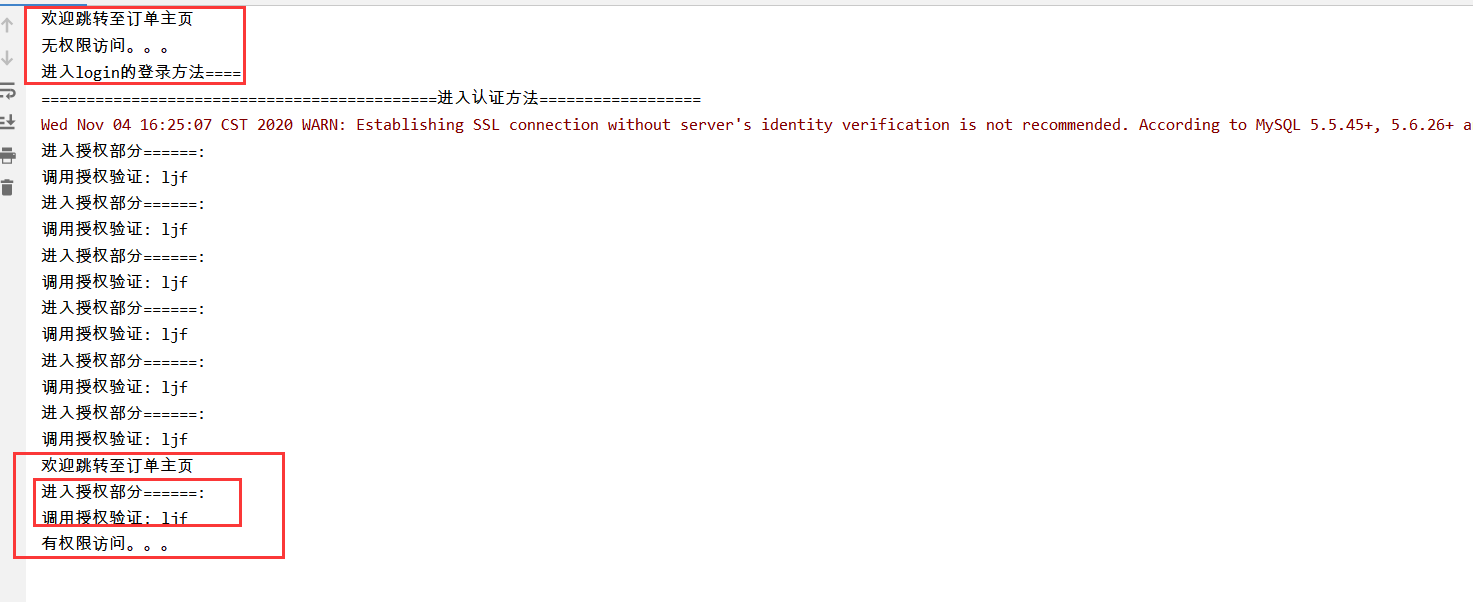
#再次刷新：http://localhost:8888/shiro/order/save



#刷新后：



#后台：



总结：在用户登录之后，再次访问http://localhost:8888/shiro/order/save，便拥有权限，可以访问到指定的订单页面。

**两种情况的总结：**

情况1：用户没有登录，直接访问http://localhost:8888/shiro/order/save，没有权限，按照设定的逻辑，跳转到登录首页（调用subject.hasRole("admin"))判断时，没有登录认证，不会调用customerRealm方法中的doGetAuthorizationInfo(PrincipalCollection principalCollection）；

情况2，用户登录的情况下，直接访问http://localhost:8888/shiro/order/save，有登录认证，调用subject.hasRole("admin"))，然后间接调用customerRealm方法中的doGetAuthorizationInfo(PrincipalCollection principalCollection) 方法，通过String primaryPrincipal = (String) principalCollection.getPrimaryPrincipal();获**取当前登录的用户名，然后按照逻辑判断登录的用户名和设定的逻辑名一致，赋予相应的角色和权限，进行后续的操作。**

猜测：

碰到hasRoes(xxx),@RequiresPermissions ....等等判断方法，只有登录认证成功后，才会调用realm中的doGetAuthorizationInfo的授权方法。如果没有登录认证，或者没有登录认证成功，则不会调用realm中的doGetAuthorizationInfo的授权方法。

### 3.2 shiro的菜单授权

使用admin用户



使用ljf用户：

