# Spring-security使用流程

#### 导依赖 spring-security-web、spring-security-config

#### 配置spring-security.xml文件

**配置url白名单：**

<security:http pattern="/login.html" security="none"/>  
<security:http pattern="/css/\*\*" security="none"/>  
<security:http pattern="/img/\*\*" security="none"/>  
<security:http pattern="/js/\*\*" security="none"/>  
<security:http pattern="/plugins" security="none"/>

**是否开启spel表达式以及过滤内容**

<security:http use-expressions="false">  
 <security:intercept-url pattern="/\*\*" access="ROLE\_USER"/>  
 <security:form-login login-page="/login.html" authentication-success-forward-url="/admin/index.html" always-use-default-target="true"

login-processing-url="/login.do" //登录页面的action请求

authentication-failure-forward-url="/login.html"/>  
 <security:headers>

<security:logout logout-url="/logout" invalidate-session="true" logout-success-url="/login.html" />//配置登出  
 <security:frame-options policy="SAMEORIGIN"></security:frame-options>  
 </security:headers>  
 <!—开启CSRF,默认是开启的 -->  
 <security:csrf disabled="true" />  
</security:http>

配置认证管理

<security:authentication-manager>  
 <security:authentication-provider >  
 <security:user-service >  
 <security:user name="admin" authorities="ROLE\_USER"></security:user>  
 <security:user name="user" authorities="ROLE\_ADMIN"></security:user>  
 </security:user-service>  
 </security:authentication-provider>  
</security:authentication-manager>

#### 配置web.xml文件

1. 过滤器及全局变量

<filter>  
 <filter-name>springSecurityFilterChain</filter-name>  
 <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>  
</filter>  
<context-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:spring/spring-security.xml</param-value>  
</context-param>  
<filter-mapping>  
 <filter-name>springSecurityFilterChain</filter-name>  
 <url-pattern>/\*</url-pattern>  
</filter-mapping>

1. 监听器

<listener>  
 <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>  
</listener>

3.可通过SecurityContextHolder.*getContext*()获取到认证用户，并可获取用户名

1. 基于数据库认证

通过基于数据库认证spring-security

1. 配置文件中拦截白名单需添加注册资源路径
2. 配置文件中的认证管理器中的认证提供者 <provider>
3. 认证类的实现需实现UserServiceDetail接口并重写loaduserByUsername方法
4. 返回一个spring-security提供的User类
5. 远程注入xxxService 根据username查找xxx,根据返回对象重写User对象
6. 提供xxxService的setter方法
7. 在spring-security.xml文件连接注册中心，将远程bean纳入到spring容器；方案二是用@reference注解，并且配置dubbo的包扫描
8. <dubbo:application name="pinyougou-shop-web" />  
   <dubbo:registry address="zookeeper://192.168.25.12:2181"/>  
   <dubbo:annotation package="com.itcast.service" />

#### 密码加密：

1. Bcrypt 加盐

配置加密密码方式后，需在spring-security的认证管理器配置<password-encoder> 且需配置加密类的bean

<bean id="passwordEncoder" class="org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder"></bean>

Spring提供的加密类

BCryptPasswordEncoder passwordEncoder = new BCryptPasswordEncoder(); String password = passwordEncoder.encode(seller.getPassword()); seller.setPassword(password);