## Spring security

[樊冬]

拦截器url拦截，管理权限。

用户<>角色<>权限

## 1.Jar包依赖，version:4.2.2.RELEASE

<!-- Spring Security -->

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-web</artifactId>

<version>${spring.security.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-config</artifactId>

<version>${spring.security.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.security</groupId>

<artifactId>spring-security-core</artifactId>

<version>${spring.security.version}</version>

</dependency>

## 2.Spring security配置

**1.**定义哪些资源是受保护的，哪些可以不受保护的

**2.**使用时的一些个人配置:登陆，登出，无权限返回 是否使用默认配置， csrf

**3.**拦截的管理和提供，拦截的规则（Filter）

**4.**单点登录

pattern属性的: 通过正则表达式来匹配资源文件。

security="none" 不受spring security保护的资源

< url可以允许访问的话，图片不会出现>

<http pattern=*"/framework/\*\*"* security=*"none"* />

<http pattern=*"/auto/createVMTask"* security=*"none"* /><!-- 创建虚拟机 -->

<http pattern=*"/component/\*\*"* security=*"none"* />

<http pattern=*"/css/\*\*"* security=*"none"* />

<http pattern=*"/js/\*\*"* security=*"none"* />

<http pattern=*"/static/\*\*"* security=*"none"* />

<http pattern=*"/main/\*\*"* security=*"none"* />

**一个简单的配置**

<http auto-config='true'>

<!--对login.jsp放行，不用保护-->

<intercept-url pattern="/login.jsp" filters="none" />

<!-- 允许访问的uri -->

<intercept-url pattern="/\*\*" access="ROLE\_USER" />

<!-- 登陆页面配置 -->

<form-login login-page="/login.jsp" default-target-url="/index.jsp" authentication-failure-url="/login.jsp?error=true"/>

</http>

**升级配置**

<http use-expressions=*"true"*

auto-config=*"false"*

entry-point-ref=*"paasEntryPoint"*>

<access-denied-handler error-page=*"/main/noPermissions"*/>

##允许访问的url

\* <intercept-url pattern=*"/\*\*"* access=*"isAuthenticated()"* />

##注销

\* <logout

logout-success-url=*"/j\_spring\_security\_logout"* logout-url=*"/main/logout"*

invalidate-session=*"true"*

delete-cookies=*"JSESSIONID"*/>

<custom-filter

ref=*"paasSecurityFilter"*

before=*"FILTER\_SECURITY\_INTERCEPTOR"*/>

<csrf disabled=*"true"* />

<headers>

<frame-options policy=*"SAMEORIGIN"* />

</headers>

</http>

## 权限管理者

<authentication-manager alias=*"authenticationManager"*>

##权限提供者

<authentication-provider user-service-ref=*"paasUserDetailServiceImpl"* />

</authentication-manager>

##配置的单点登录

<beans:bean

id=*"sessionRegistry"* class=*"org.springframework.security.core.session.SessionRegistryImpl"* />

<beans:bean class=*"org.springframework.security.web.session.ConcurrentSessionFilter"*>

<beans:constructor-arg ref=*"sessionRegistry"*/>

</beans:bean>

## 3.web.xml

1. 加载Spring；

2、加载Spring Security；

3、添加Spring Security Session监听器(用于控制登录)

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>

classpath:/spring/spring-context.xml,

classpath:/spring/spring-datasource.xml,

classpath:/spring/spring-security.xml

</param-value>

</context-param>

## Spring Security会话控制

<listener>

<listener-class>

org.springframework.security.web.session.HttpSessionEventPublisher

</listener-class>

</listener>

<!-- Spring Security -->

<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>

## 4. JAVA

一个自定义的过滤器

包含authenticationManager,accessDecisionManager,securityMetadataSource三个属性

**public** **void** init(){

**super**.setAuthenticationManager(authenticationManager);

**super**.setAccessDecisionManager(userDecisionManager);

}

1.PaasSecurityMetadataSource 实现FilterInvocationSecurityMetadataSource

1.加载所有资源与权限的关系

2.返回所请求资源所需要的权限

2.PaasUserDecisionManager 实现AccessDecisionManager

DECIDE方法

所请求的资源拥有的权限

访问所请求资源所需要的权限

用户所拥有的权限authentication

Iterator<ConfigAttribute> iterator = configAttributes.iterator();

**while**(iterator.hasNext()) {

ConfigAttribute configAttribute = iterator.next();

String needPermission = configAttribute.getAttribute();

**for**(GrantedAuthority ga : authentication.getAuthorities()) {

**if**(needPermission.equals(ga.getAuthority())) {

**return**;

}

}

}

参考：

<http://blog.csdn.net/ydj7501603/article/details/9049663>

<http://www.cnblogs.com/Beyond-bit/p/SpringMVC_And_SpringSecurity.html>

demo代码

BASE java