<https://blog.csdn.net/monkeyking1987/article/details/16828059>

# Spring Security 与 Oauth2 整合 步骤

2013年11月19日 22:31:56 [monkeyking1987](https://me.csdn.net/monkeyking1987) 阅读数：128474

 版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/monkeyking1987/article/details/16828059

spring-security-oauth2的项目地址为 <https://github.com/spring-projects/spring-security-oauth/tree/master/spring-security-oauth2>

spring-security-oauth2的demo 地址为 <https://github.com/spring-projects/spring-security-oauth/tree/master/samples/oauth2>

**为什么要写**

1. spring-security-oauth2的demo 不容易让开发者理解, 配置的内容很多, 没有分解的步骤; 我曾经试着按照文档(<https://github.com/spring-projects/spring-security-oauth/blob/master/docs/oauth2.md>) 配置了几次, 结果全失败, 无一成功(说实话, 这是第二次在实际项目中使用spring & oauth,但还是花了不少时间才完全弄清楚);甚至有时候找错误的原因都不好找.

2.Oauth应该属于security的一部分, 但demo并没有将二者分开, 混在一起

3.总结现在, 方便未来

**相比于demo,作了哪些改进**

1. 将Spring MVC配置与Oauth的配置分开, 互不影响

2.将用户信息存放数据库

3.将ClientDetails数据存放于数据库,并能对数据进行管理

4.扩展ClientDetails基本属性, 添加trusted属性,用于判断Client是否是可信任的

5.取消掉demo中一些不必要的配置

6.针对不同的资源配置不同的权限

7.token存入数据库而不是内存

**开始**

>>前提:  使用Maven来管理项目; spring-security-oauth的版本号为 1.0.5.RELEASE

1. 添加Maven dependencies; 以下只列出了主要的

1. *<!--spring security-->*
2. <dependency>
3. <groupId>org.springframework.security</groupId>
4. <artifactId>spring-security-core</artifactId>
5. <version>${spring.security.version}</version>
6. </dependency>
7. <dependency>
8. <groupId>org.springframework.security</groupId>
9. <artifactId>spring-security-web</artifactId>
10. <version>${spring.security.version}</version>
11. </dependency>
12. <dependency>
13. <groupId>org.springframework.security</groupId>
14. <artifactId>spring-security-taglibs</artifactId>
15. <version>${spring.security.version}</version>
16. </dependency>
17. <dependency>
18. <groupId>org.springframework.security</groupId>
19. <artifactId>spring-security-acl</artifactId>
20. <version>${spring.security.version}</version>
21. </dependency>
22. <dependency>
23. <groupId>org.springframework.security</groupId>
24. <artifactId>spring-security-crypto</artifactId>
25. <version>${spring.security.version}</version>
26. </dependency>
27. <dependency>
28. <groupId>org.springframework.security</groupId>
29. <artifactId>spring-security-config</artifactId>
30. <version>${spring.security.version}</version>
31. </dependency>
33. <dependency>
34. <groupId>org.springframework.security.oauth</groupId>
35. <artifactId>spring-security-oauth2</artifactId>
36. <version>1.0.5.RELEASE</version>
37. </dependency>

2. web.xml配置; 这一步与只使用Spring Security的配置一样.

1. </pre><pre code\_snippet\_id="73897" snippet\_file\_name="blog\_20131119\_2\_2257675" name="code" class="html"> <filter>
2. <filter-name>springSecurityFilterChain</filter-name>
3. <filter-class>org.springframework.web.filter.DelegatingFilterProxy</filter-class>
4. </filter>
6. <filter-mapping>
7. <filter-name>springSecurityFilterChain</filter-name>
8. <url-pattern>/\*</url-pattern>
9. </filter-mapping>
11. *<!--contextConfigLocation -->*
12. <context-param>
13. <param-name>contextConfigLocation</param-name>
14. <param-value>classpath:spring/\*.xml</param-value>
15. </context-param>
17. *<!-- Spring context listener -->*
18. <listener>
19. <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
20. </listener>
22. *<!--hy mvc-->*
23. <servlet>
24. <servlet-name>hy</servlet-name>
25. <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
26. <load-on-startup>2</load-on-startup>
27. </servlet>
28. <servlet-mapping>
29. <servlet-name>hy</servlet-name>
30. <url-pattern>/</url-pattern>
31. </servlet-mapping>

对于Spring MVC, 需要配置文件hy-servlet.xml, 该文件不是这儿关注的(忽略);

在classpath创建spring目录, 在该目录里创建 ***security.xml***文件, 这是所有步骤配置的重点.

3.security.xml的配置; 重点开始.

3.1 起用注解; TokenEndpoint与AuthorizationEndpoint需要

1. <mvc:annotation-driven/>
2. <mvc:default-servlet-handler/>

3.2  TokenServices 配置

   1). TokenStore, 使用JdbcTokenStore, 将token信息存放数据库, 需要提供一个dataSource对象; 也可使用InMemoryTokenStore存于内存中

1. *<!--<beans:bean id="tokenStore" class="org.springframework.security.oauth2.provider.token.InMemoryTokenStore"/>-->*
2. <beans:bean id="tokenStore" class="org.springframework.security.oauth2.provider.token.JdbcTokenStore">
3. <beans:constructor-arg index="0" ref="dataSource"/>
4. </beans:bean>

**注**: 可以在spring-security-oauth2中找到对应的SQL脚本, 地址为<https://github.com/spring-projects/spring-security-oauth/tree/master/spring-security-oauth2/src/test/resources>, 目录中的[schema.sql](https://github.com/spring-projects/spring-security-oauth/blob/master/spring-security-oauth2/src/test/resources/schema.sql) 即是. (以下不再说明SQL脚本的问题)

  2).TokenServices; 需要注入TokenStore

1. <beans:bean id="tokenServices" class="org.springframework.security.oauth2.provider.token.DefaultTokenServices">
2. <beans:property name="tokenStore" ref="tokenStore"/>
3. <beans:property name="supportRefreshToken" value="true"/>
4. </beans:bean>

      如果允许刷新token 请将supportRefreshToken 的值设置为true, 默认为不允许

3.3 ClientDetailsService 配置, 使用JdbcClientDetailsService, 也需要提供dataSource, 替换demo中直接配置在配置文件中

1. <beans:bean id="clientDetailsService" class="org.springframework.security.oauth2.provider.JdbcClientDetailsService">
2. <beans:constructor-arg index="0" ref="dataSource"/>
3. </beans:bean>

3.4 ClientDetailsUserDetailsService配置, 该类实现了Spring security中 UserDetailsService 接口

1. <beans:bean id="oauth2ClientDetailsUserService"
2. class="org.springframework.security.oauth2.provider.client.ClientDetailsUserDetailsService">
3. <beans:constructor-arg ref="clientDetailsService"/>
4. </beans:bean>

3.5 OAuth2AuthenticationEntryPoint配置

1. <beans:bean id="oauth2AuthenticationEntryPoint"
2. class="org.springframework.security.oauth2.provider.error.OAuth2AuthenticationEntryPoint"/>

3.6 oauth2 AuthenticationManager配置; 在整个配置中,有两个AuthenticationManager需要配置

1. <authentication-manager id="oauth2AuthenticationManager">
2. <authentication-provider user-service-ref="oauth2ClientDetailsUserService"/>
3. </authentication-manager>

第二个AuthenticationManager用于向获取UserDetails信息, 

1. <authentication-manager alias="authenticationManager">
2. <authentication-provider user-service-ref="userService">
3. <password-encoder hash="md5"/>
4. </authentication-provider>
5. </authentication-manager>

userService是一个实现UserDetailsService的Bean

3.7 OAuth2AccessDeniedHandler配置, 实现AccessDeniedHandler接口

1. <beans:bean id="oauth2AccessDeniedHandler"
2. class="org.springframework.security.oauth2.provider.error.OAuth2AccessDeniedHandler"/>

3.8 UserApprovalHandler配置, 这儿使用DefaultUserApprovalHandler, 这里是实现client是否可信任的关键点,你可以扩展该接口来自定义approval行为

1. <beans:bean id="oauthUserApprovalHandler" class="org.springframework.security.oauth2.provider.approval.DefaultUserApprovalHandler">
2. </beans:bean>

3.9 authorization-server配置, 核心

1. <oauth2:authorization-server client-details-service-ref="clientDetailsService" token-services-ref="tokenServices"
2. user-approval-handler-ref="oauthUserApprovalHandler">
3. <oauth2:authorization-code/>
4. <oauth2:implicit/>
5. <oauth2:refresh-token/>
6. <oauth2:client-credentials/>
7. <oauth2:password/>
8. </oauth2:authorization-server>

该元素里面的每个标签可设置每一种authorized-grant-type的行为. 如disable refresh-token的配置为

<oauth2:refresh-token disabled="true"/>

3.10 Oauth2 AccessDecisionManager配置, 这儿在默认的Spring Security AccessDecisionManager的基础上添加了ScopeVoter

1. <beans:bean id="oauth2AccessDecisionManager" class="org.springframework.security.access.vote.UnanimousBased">
2. <beans:constructor-arg>
3. <beans:list>
4. <beans:bean class="org.springframework.security.oauth2.provider.vote.ScopeVoter"/>
5. <beans:bean class="org.springframework.security.access.vote.RoleVoter"/>
6. <beans:bean class="org.springframework.security.access.vote.AuthenticatedVoter"/>
7. </beans:list>
8. </beans:constructor-arg>
9. </beans:bean>

3.11 resource-server配置, 这儿定义两咱不同的resource

1. *<!--unity resource server filter-->*
2. <oauth2:resource-server id="unityResourceServer" resource-id="unity-resource" token-services-ref="tokenServices"/>
4. *<!--mobile resource server filter-->*
5. <oauth2:resource-server id="mobileResourceServer" resource-id="mobile-resource" token-services-ref="tokenServices"/>

**注意**: 每个resource-id的值必须在对应的ClientDetails中resourceIds值中存在

3.12 ClientCredentialsTokenEndpointFilter配置, 该Filter将作用于Spring Security的chain 链条中

1. <beans:bean id="clientCredentialsTokenEndpointFilter"
2. class="org.springframework.security.oauth2.provider.client.ClientCredentialsTokenEndpointFilter">
3. <beans:property name="authenticationManager" ref="oauth2AuthenticationManager"/>
4. </beans:bean>

3.13 /oauth/token 的http 配置, 用于监听该URL的请求, 核心

1. <http pattern="/oauth/token" create-session="stateless" authentication-manager-ref="oauth2AuthenticationManager"
2. entry-point-ref="oauth2AuthenticationEntryPoint">
3. <intercept-url pattern="/oauth/token" access="IS\_AUTHENTICATED\_FULLY"/>
4. <anonymous enabled="false"/>
5. <http-basic entry-point-ref="oauth2AuthenticationEntryPoint"/>
7. <custom-filter ref="clientCredentialsTokenEndpointFilter" before="BASIC\_AUTH\_FILTER"/>
8. <access-denied-handler ref="oauth2AccessDeniedHandler"/>
9. </http>

3.14 针对不同resource的http配置, 由于上面配置了两个resource, 这儿也配置两个

1. *<!--unity http configuration-->*
2. <http pattern="/unity/\*\*" create-session="never" entry-point-ref="oauth2AuthenticationEntryPoint"
3. access-decision-manager-ref="oauth2AccessDecisionManager">
4. <anonymous enabled="false"/>
6. <intercept-url pattern="/unity/\*\*" access="ROLE\_UNITY,SCOPE\_READ"/>
8. <custom-filter ref="unityResourceServer" before="PRE\_AUTH\_FILTER"/>
9. <access-denied-handler ref="oauth2AccessDeniedHandler"/>
10. </http>
12. *<!--mobile http configuration-->*
13. <http pattern="/m/\*\*" create-session="never" entry-point-ref="oauth2AuthenticationEntryPoint"
14. access-decision-manager-ref="oauth2AccessDecisionManager">
15. <anonymous enabled="false"/>
17. <intercept-url pattern="/m/\*\*" access="ROLE\_MOBILE,SCOPE\_READ"/>
19. <custom-filter ref="mobileResourceServer" before="PRE\_AUTH\_FILTER"/>
20. <access-denied-handler ref="oauth2AccessDeniedHandler"/>
21. </http>

注意每一个http对应不同的resourceServer. access-decison-manager-ref对应Oauth的AccessDecisionManager

3.15 默认的http配置,给/oauth/\*\* 设置权限

1. <http access-denied-page="/login.jsp?authorization\_error=2" disable-url-rewriting="true"
2. authentication-manager-ref="authenticationManager">
3. <intercept-url pattern="/oauth/\*\*" access="ROLE\_USER,ROLE\_UNITY,ROLE\_MOBILE"/>
4. <intercept-url pattern="/\*\*" access="IS\_AUTHENTICATED\_ANONYMOUSLY"/>
6. <form-login authentication-failure-url="/login.jsp?authentication\_error=1" default-target-url="/index.jsp"
7. login-page="/login.jsp" login-processing-url="/login.do"/>
8. <logout logout-success-url="/index.jsp" logout-url="/logout.do"/>
9. <anonymous/>
10. </http>

到此, securiy.xml 配置完毕.

当然,还有些额外的工作你需要做, 如配置dataSource, 创建数据库, 添加用户用户信息, 管理ClientDetails等等.

Oauth相关的数据都是存放在数据库, 我们就可以根据表结果创建domain来实现管理.

为了方便大家学习与讨论, 我现在把该项目放在了GIT OSC上, 访问地址: <http://git.oschina.net/shengzhao/spring-oauth-server>, 欢迎大家关注.

与文章相关的配置,扩展请访问: <http://andaily.com/blog/?cat=19>

与文章相关的讨论请访问: <http://andaily.com/blog/?page_id=182>