spring security实现动态配置url权限...

缘起

标准的RABC，权限需要支持动态配置，spring security默认是在代码里约定好权限，真实的业务场景通常需要可以支持动态配置角色访问权限，即在运行时去配置url对应的访问角色。

基于spring security，如何实现这个需求呢?

最简单的方法就是自定义一个Filter去完成权限判断，但这脱离了spring security框架，如何基于spring security优雅的实现呢?

spring security 授权回顾

spring security 通过FilterChainProxy作为注册到web的filter，FilterChainProxy里面一次包含了内置的多个过滤器，我们首先需要了解spring security内置的各种filter：

Alias	Filter Class	Namespace Element

		or Attribute
CHANNEL_FILTER	ChannelProcessingFilter	http/intercept-url@requires-channel
SECURITY_CONTEXT_FILTER	SecurityContextPersistenceFilter	http
CONCURRENT_SESSION_FILTER	ConcurrentSessionFilter	session-management/concurrency-control
HEADERS_FILTER	HeaderWriterFilter	http/headers
CSRF_FILTER	CsrfFilter	http/csrf
LOGOUT_FILTER	LogoutFilter	http/logout
X509_FILTER	X509AuthenticationFilter	http/x509
PRE_AUTH_FILTER	AbstractPreAuthenticatedProcessingFilter Subclasses	N/A
CAS_FILTER	CasAuthenticationFilter	N/A
FORM_LOGIN_FILTER	UsernamePasswordAuthenticationFilter	http/form-login
BASIC_AUTH_FILTER	BasicAuthenticationFilter	http/http-basic
SERVLET_API_SUPPORT_FILTER	SecurityContextHolderAwareRequestFilter	http/@servlet-api-provision
JAAS_API_SUPPORT_FILTER	JaasApiIntegrationFilter	http/@jaas-api-provision
REMEMBER_ME_FILTER	RememberMeAuthenticationFilter	http/remember-me
ANONYMOUS_FILTER	AnonymousAuthenticationFilter	http/anonymous
SESSION_MANAGEMENT_FILTER	SessionManagementFilter	session-management
EXCEPTION_TRANSLATION_FILTER	ExceptionTranslationFilter	http
FILTER_SECURITY_INTERCEPTOR	FilterSecurityInterceptor	http
SWITCH_USER_FI		

3.基于ambassador实现K8S灰度发布
4.JAVA 使用jgit管理git仓库
5.JAVA使用SnakeYAML解析与序列化YAML
6.通过 Drone Rest API 获取构建记录日志
7.fastText训练word2vec并用于训练任务
8.NLP标注工具brat 配置文件说明
9.教程 —— 如何在自己的应用集成superset
10.知识图谱推理与实践(3) -- jena自定义builtin

我的标签

java(21)
docker(15)
JavaScript(13)
c#(8)
Spring Boot(8)

LTER	SwitchUserFilter	N/A

最重要的是`FilterSecurityInterceptor`，该过滤器实现了主要的鉴权逻辑，最核心的代码在这里：

```
protected InterceptorStatusToken beforeInvocation(Object
object) {

        // 获取访问URL所需权限
        Collection<ConfigAttribute> attributes =
this.obtainSecurityMetadataSource()
                .getAttributes(object);


        Authentication authenticated =
authenticateIfRequired();

        // 通过accessDecisionManager鉴权
        try {

this.accessDecisionManager.decide(authenticated, object,
attributes);
        }
        catch (AccessDeniedException accessDeniedException)
{
            publishEvent(new
AuthorizationFailureEvent(object, attributes,
authenticated,
                    accessDeniedException));

            throw accessDeniedException;
        }

        if (debug) {
            logger.debug("Authorization successful");
        }

        if (publishAuthorizationSuccess) {
            publishEvent(new AuthorizedEvent(object,
attributes, authenticated));
        }

        // Attempt to run as a different user
        Authentication runAs =
this.runAsManager.buildRunAs(authenticated, object,
                attributes);
```

算法(6)
jenkins(6)
angular(5)
python(5)
知识图谱(5)
更多

积分与排名

积分 - 238166
排名 - 1748

随笔分类 (115)

ASP.NET编程(10)
java编程(39)
jenkins X实践系列(4)
编程周边(14)
机器学习(5)
架构与选型(4)
前端开发(12)
移动开发(6)
云原生开发(16)
知识图谱(5)

随笔档案 (141)

```
        if (runAs == null) {
            if (debug) {
                logger.debug("RunAsManager did not change
Authentication object");
            }

            // no further work post-invocation
            return new
InterceptorStatusToken(SecurityContextHolder.getContext(),
false,
                    attributes, object);
        }
        else {
            if (debug) {
                logger.debug("Switching to RunAs
Authentication: " + runAs);
            }

            SecurityContext origCtx =
SecurityContextHolder.getContext();

SecurityContextHolder.setContext(SecurityContextHolder.crea
teEmptyContext());

SecurityContextHolder.getContext().setAuthentication(runAs)
;

            // need to revert to token.Authenticated post-
invocation
            return new InterceptorStatusToken(origCtx,
true, attributes, object);
        }
    }
```

从上面可以看出，要实现动态鉴权，可以从两方面着手：

- 自定义SecurityMetadataSource，实现从数据库加载ConfigAttribute
- 另外就是可以自定义accessDecisionManager，官方的UnanimousBased其实足够使用，并且他是基于AccessDecisionVoter来实现权限认证的，因此我们只需要自定义一个AccessDecisionVoter就可以了

下面来看分别如何实现。

自定义AccessDecisionManager

官方的三个AccessDecisionManager都是基于AccessDecisionVoter来实现权限认证的，因此我们只需要自定义一个AccessDecisionVoter就可以了。

自定义主要是实现 `AccessDecisionVoter` 接口，我们可以仿照官方的RoleVoter实现一个：

```
public class RoleBasedVoter implements
AccessDecisionVoter<Object> {

    @Override
    public boolean supports(ConfigAttribute attribute) {
        return true;
    }

    @Override
    public int vote(Authentication authentication, Object
object, Collection<ConfigAttribute> attributes) {
        if(authentication == null) {
            return ACCESS_DENIED;
        }
        int result = ACCESS_ABSTAIN;
        Collection<? extends GrantedAuthority> authorities
= extractAuthorities(authentication);

        for (ConfigAttribute attribute : attributes) {
            if(attribute.getAttribute()==null){
                continue;
            }
            if (this.supports(attribute)) {
                result = ACCESS_DENIED;

                // Attempt to find a matching granted
authority
                for (GrantedAuthority authority :
authorities) {
                    if
(attribute.getAttribute().equals(authority.getAuthority()))
{
                        return ACCESS_GRANTED;
                    }
                }
```

2012年4月(1)
2012年3月(3)
2011年11月(2)
2011年10月(2)
2011年4月(2)
2011年3月(4)
2011年1月(6)
2010年12月(2)
2010年9月(2)
2010年6月(3)
2010年1月(2)
2009年5月(1)
2009年2月(1)
2008年9月(2)
2008年8月(2)

阅读排行榜

1. Spring Bo...
2. \x 开头编...
3. 开源APM...
4. Docker+J...
5. 使用Sprin...

推荐排行榜

1. asp.net MV C 权限设计(续) (22)

```
            }
        }

        return result;
    }

    Collection<? extends GrantedAuthority>
extractAuthorities(
        Authentication authentication) {
        return authentication.getAuthorities();
    }

    @Override
    public boolean supports(Class clazz) {
        return true;
    }
}
```

如何加入动态权限呢?

```
vote(Authentication authentication, Object object,
Collection<ConfigAttribute> attributes)
```

里的 `Object object` 的类型是 `FilterInvocation`，可以通过 `getRequestUrl` 获取当前请求的URL:

```
  FilterInvocation fi = (FilterInvocation) object;
  String url = fi.getRequestUrl();
```

因此这里扩展空间就大了，可以从DB动态加载，然后判断URL的ConfigAttribute就可以了。

如何使用这个RoleBasedVoter呢？在configure里使用accessDecisionManager方法自定义，我们还是使用官方的 `UnanimousBased`，然后将自定义的RoleBasedVoter加入即可。

```
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true,
securedEnabled = true)
public class SecurityConfiguration extends
WebSecurityConfigurerAdapter {


    @Override
    protected void configure(HttpSecurity http) throws
Exception {
```

2. 10款精选的用于构建良好易用性网站的jQuery插件(14)
3. Spring Boot配置文件放在jar外部(14)
4. 开源小工具酷狗、网易音乐缓存文件转mp3工具(14)
5. Solr vs. Elasticsearch谁是开源搜索引擎王者(13)

```
        http
            .addFilterBefore(corsFilter,
UsernamePasswordAuthenticationFilter.class)
            .exceptionHandling()
            .authenticationEntryPoint(problemSupport)
            .accessDeniedHandler(problemSupport)
        .and()
            .csrf()
            .disable()
            .headers()
            .frameOptions()
            .disable()
        .and()
            .sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS)
        .and()
            .authorizeRequests()
            // 自定义accessDecisionManager
            .accessDecisionManager(accessDecisionManager())

        .and()
            .apply(securityConfigurerAdapter());

    }


    @Bean
    public AccessDecisionManager accessDecisionManager() {
        List<AccessDecisionVoter<? extends Object>>
decisionVoters
            = Arrays.asList(
            new WebExpressionVoter(),
            // new RoleVoter(),
            new RoleBasedVoter(),
            new AuthenticatedVoter());
        return new UnanimousBased(decisionVoters);
    }
```

自定义SecurityMetadataSource

自定义FilterInvocationSecurityMetadataSource只要实现接口即可，在接口里从DB动态加载规则。

为了复用代码里的定义，我们可以将代码里生成的

SecurityMetadataSource 带上，在构造函数里传入默认的FilterInvocationSecurityMetadataSource。

```
public class AppFilterInvocationSecurityMetadataSource
implements
org.springframework.security.web.access.intercept.FilterInv
ocationSecurityMetadataSource {

    private FilterInvocationSecurityMetadataSource
superMetadataSource;

    @Override
    public Collection<ConfigAttribute>
getAllConfigAttributes() {
        return null;
    }

    public
AppFilterInvocationSecurityMetadataSource(FilterInvocationS
ecurityMetadataSource
expressionBasedFilterInvocationSecurityMetadataSource){
         this.superMetadataSource =
expressionBasedFilterInvocationSecurityMetadataSource;

         // TODO 从数据库加载权限配置
    }

    private final AntPathMatcher antPathMatcher = new
AntPathMatcher();

    // 这里的需要从DB加载
    private final Map<String,String> urlRoleMap = new
HashMap<String,String>(){{
        put("/open/**","ROLE_ANONYMOUS");
        put("/health","ROLE_ANONYMOUS");
        put("/restart","ROLE_ADMIN");
        put("/demo","ROLE_USER");
    }};

    @Override
    public Collection<ConfigAttribute> getAttributes(Object
object) throws IllegalArgumentException {
        FilterInvocation fi = (FilterInvocation) object;
        String url = fi.getRequestUrl();

        for(Map.Entry<String,String>
entry:urlRoleMap.entrySet()){
```

```
            if(antPathMatcher.match(entry.getKey(),url)){
                return
SecurityConfig.createList(entry.getValue());
            }
        }

        //  返回代码定义的默认配置
        return superMetadataSource.getAttributes(object);
    }



    @Override
    public boolean supports(Class<?> clazz) {
        return
FilterInvocation.class.isAssignableFrom(clazz);
    }
}
```

怎么使用？和`accessDecisionManager`不一样，`ExpressionUrlAuthorizationConfigurer`并没有提供set方法设置`FilterSecurityInterceptor`的`FilterInvocationSecurityMetadataSource`，how to do?

发现一个扩展方法`withObjectPostProcessor`，通过该方法自定义一个处理`FilterSecurityInterceptor`类型的`ObjectPostProcessor`就可以修改`FilterSecurityInterceptor`。

```
@EnableWebSecurity
@EnableGlobalMethodSecurity(prePostEnabled = true,
securedEnabled = true)
public class SecurityConfiguration extends
WebSecurityConfigurerAdapter {



    @Override
    protected void configure(HttpSecurity http) throws
Exception {
        http
            .addFilterBefore(corsFilter,
UsernamePasswordAuthenticationFilter.class)
            .exceptionHandling()
            .authenticationEntryPoint(problemSupport)
            .accessDeniedHandler(problemSupport)
        .and()
```

```
            .csrf()
            .disable()
            .headers()
            .frameOptions()
            .disable()
        .and()
            .sessionManagement()

.sessionCreationPolicy(SessionCreationPolicy.STATELESS)
        .and()
            .authorizeRequests()
            // 自定义FilterInvocationSecurityMetadataSource
            .withObjectPostProcessor(new
ObjectPostProcessor<FilterSecurityInterceptor>() {
                @Override
                public <O extends
FilterSecurityInterceptor> O postProcess(
                    O fsi) {

fsi.setSecurityMetadataSource(mySecurityMetadataSource(fsi.
getSecurityMetadataSource()));
                    return fsi;
                }
            })
        .and()
            .apply(securityConfigurerAdapter());

    }


    @Bean
    public AppFilterInvocationSecurityMetadataSource
mySecurityMetadataSource(FilterInvocationSecurityMetadataSo
urce filterInvocationSecurityMetadataSource) {
        AppFilterInvocationSecurityMetadataSource
securityMetadataSource = new
AppFilterInvocationSecurityMetadataSource(filterInvocationS
ecurityMetadataSource);
        return securityMetadataSource;
}
```

小结

本文介绍了两种基于spring security实现动态权限的方法，一是自定义accessDecisionManager，二是自定义

FilterInvocationSecurityMetadataSource。实际项目里可以根据需要灵活选择。

延伸阅读:

Spring Security 架构与源码分析

分类: java编程

标签: Spring Boot, java, Spring Security

« 上一篇： Spring Security 架构与源码分析
» 下一篇： 基于spring security 实现前后端分离项目权限控制

posted @ 2018-06-07 15:33 JadePeng 阅读(21105) 评论(3) 编辑 收藏

评论列表

#1楼 2018-06-07 15:40 来如风

准备放弃ss，url动态扩展这方便及其不方便

支持(0) 反对(0)

#2楼 [楼主] 2018-06-07 17:05 JadePeng

@ 来如风

了解了SS的机制，扩展也挺方便

支持(0) 反对(0)

#3楼 2018-09-06 14:50 EverKnown

请问怎么动态刷新权限信息呢

支持(0) 反对(0)

刷新评论 刷新页面 返回顶部

注册用户登录后才能发表评论，请 登录 或 注册， 访问 网站首页。

6