# Spring Security

## What is Spring Security?

Spring Security是一个提供身份认证、授权和针对常见攻击的保护的框架。Spring Security 通过一系列的**Filter**实现身份认证和授权的。

### Authentication and Authorization

* 认证(Authentication): 身份验证是验证某人身份的过程。A process of validating someone’s identity.
* 授权(Authorization): 是向某人提供访问特定资源的权限的过程。A the process of providing permission to someone to access a particular resource.

### Spring Dependencies

|  |
| --- |
| <**dependency**>  <**groupId**>org.springframework.boot</**groupId**>  <**artifactId**>spring-boot-starter-security</**artifactId**> </**dependency**> |

* spring-security-config
* spring-security-core
* spring-security-crypto
* spring-security-test
* spring-security-web

### Spring Security Auto-Configuration

* + SpringBoot 2.7以前的版本，配置方式为：

META-INF/spring.factories，加上@Configuration注解。

* + 从SpringBoot 2.7 开始，配置方式为

META-INF/spring/org.springframework.boot.autoconfigure.AutoConfiguration.imports, 以及@AutoConfiguration注解。

最主要的自动配置类**SecurityAutoConfiguration，** 通过它注册了

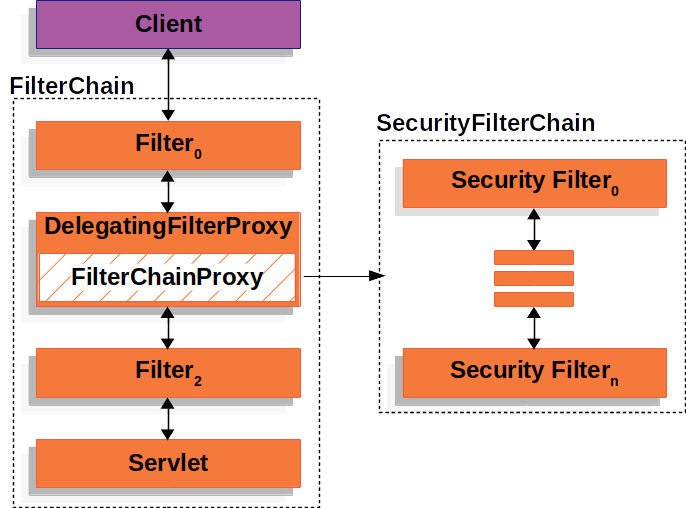
* SecurityFilterChain
* AuthenticationConfiguration

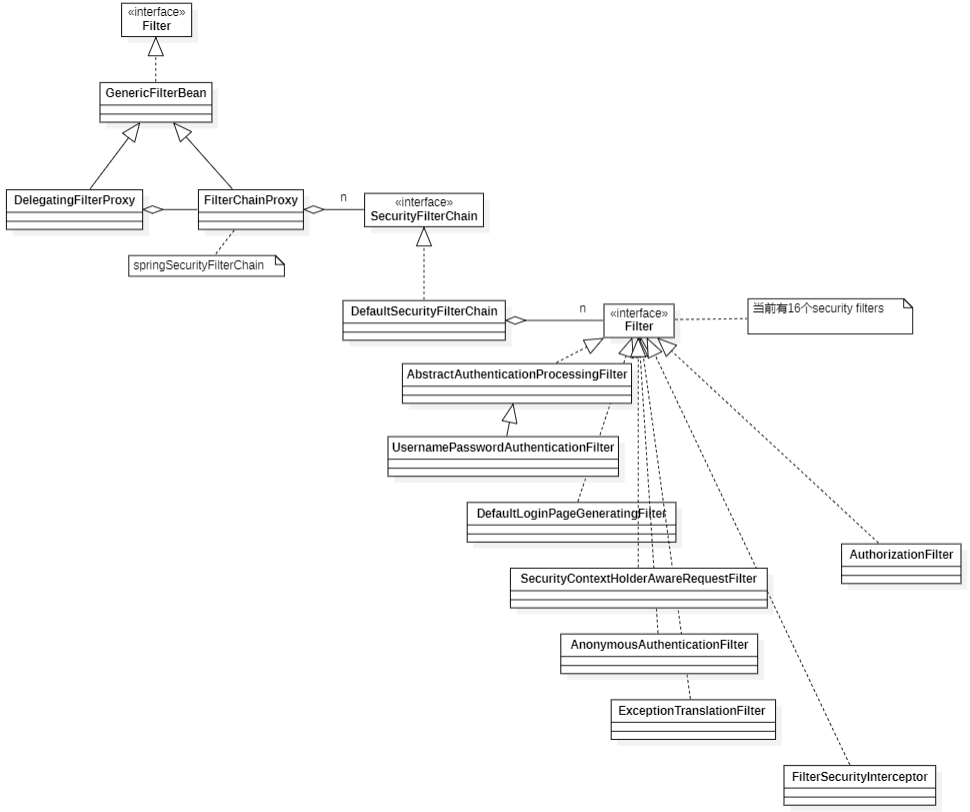
**注意：**

如果加入了Spring Security starter依赖, 而不想启动自动配置，Spring Boot中可以用以下两种方法实现：

* @SpringBootApplication(exclude = SecurityAutoConfiguration.class)
* Spring.autoconfigure.exclude= org.springframework.boot.autoconfigure.security.servlet.SecurityAutoConfiguration

## Spring Security Architecture

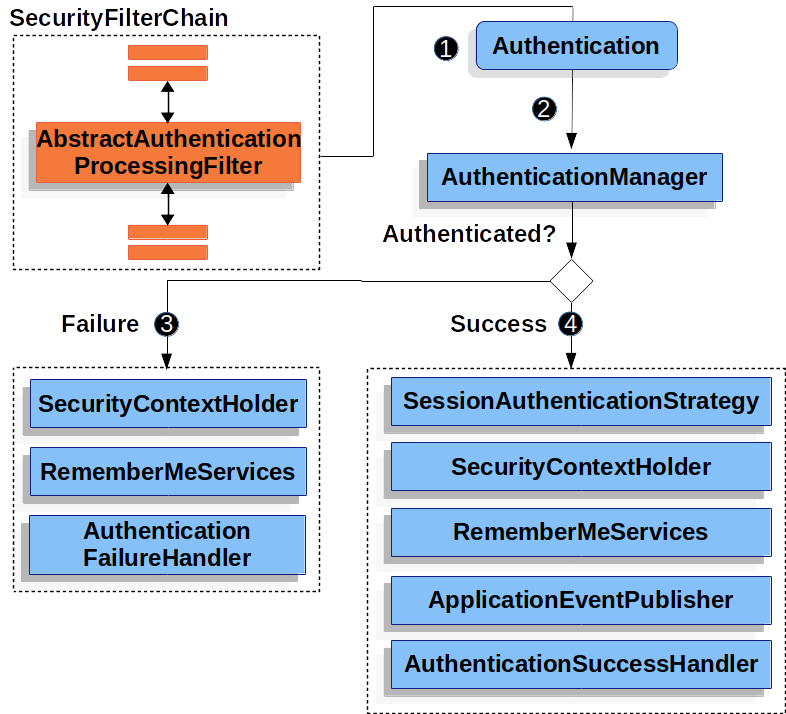


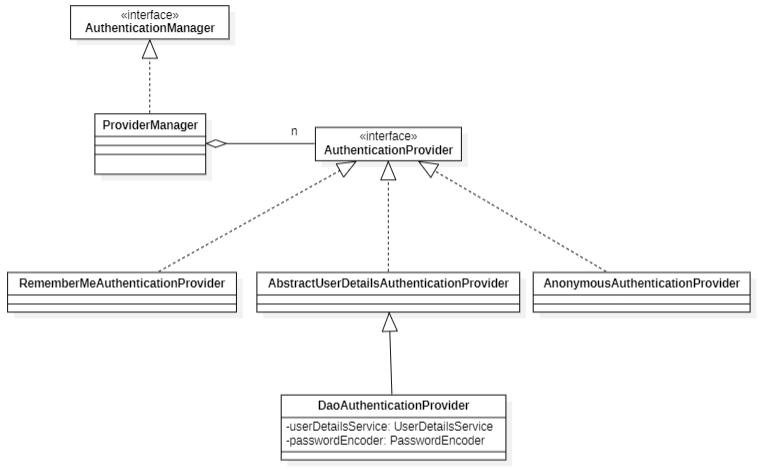


Spring Security 5.7.7版本定义了16个安全过滤器(Security Filter), 其顺序的设置通过**FilterOrderRegistration**类。(HttpSecurityConfiguration==>HttpSecurity==>FilterOrderRegistration)

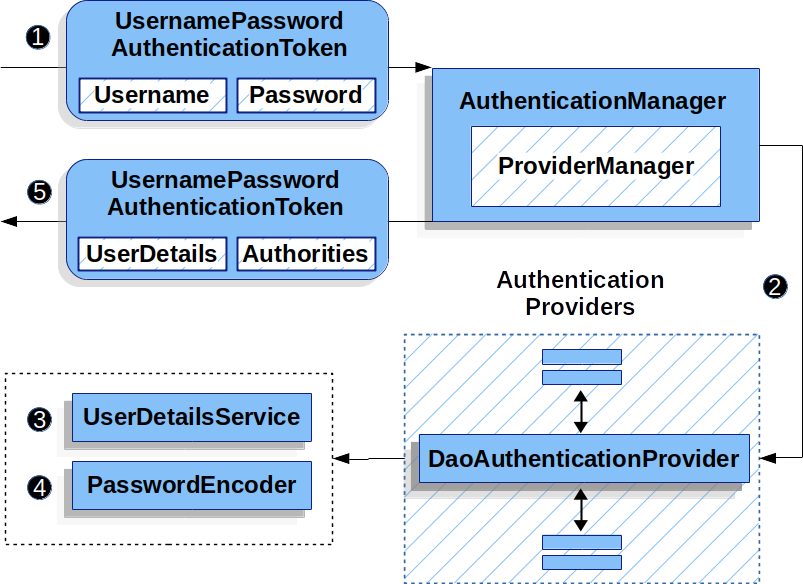
* **安全过滤器及顺序：**
* ForceEagerSessionCreationFilter
* ChannelProcessingFilter
* WebAsyncManagerIntegrationFilter
* SecurityContextPersistenceFilter
* HeaderWriterFilter
* CorsFilter
* CsrfFilter
* LogoutFilter
* OAuth2AuthorizationRequestRedirectFilter
* Saml2WebSsoAuthenticationRequestFilter
* X509AuthenticationFilter
* AbstractPreAuthenticatedProcessingFilter
* CasAuthenticationFilter
* OAuth2LoginAuthenticationFilter
* Saml2WebSsoAuthenticationFilter
* UsernamePasswordAuthenticationFilter 认证操作全靠这个过滤器，默认匹配URL为/login且必须为POST请求。
* DefaultLoginPageGeneratingFilter 默认的登录页面生成过滤器，如果没有在配置文件中指定认证页面，则由该过滤器生成一个默认认证页面。
* DefaultLogoutPageGeneratingFilter
* ConcurrentSessionFilter
* DigestAuthenticationFilter
* BearerTokenAuthenticationFilter
* BasicAuthenticationFilter
* RequestCacheAwareFilter
* SecurityContextHolderAwareRequestFilter
* JaasApiIntegrationFilter
* RememberMeAuthenticationFilter
* AnonymousAuthenticationFilter
* OAuth2AuthorizationCodeGrantFilter
* SessionManagementFilter
* ExceptionTranslationFilter (允许将AccessDeniedException和AuthenticationException 转换为HTTP响应。) 处理过滤器链中抛出的任何AccessDeniedException和AuthenticationException。 这个过滤器是必要的，因为它提供了Java异常和HTTP响应之间的桥梁。它只关心维护用户界面。此过滤器不执行任何实际的安全强制。
* FilterSecurityInterceptor (Performs security handling of HTTP resources via a filter implementation), 将被**AuthorizationFilter** 取代。
* SwitchUserFilter

## Spring Security Authentication





* **DaoAuthenticationProvider**是一个AuthenticationProvider实现，使用UserDetailsService和PasswordEncoder对用户名和密码进行身份验证。可以通过自定义UserDetailsService和PasswordEncoder实现基于数据库的认证。
* **DaoAuthenticationProvider bean**通过AuthenticationConfiguration 注入。



## Spring Security Authorization

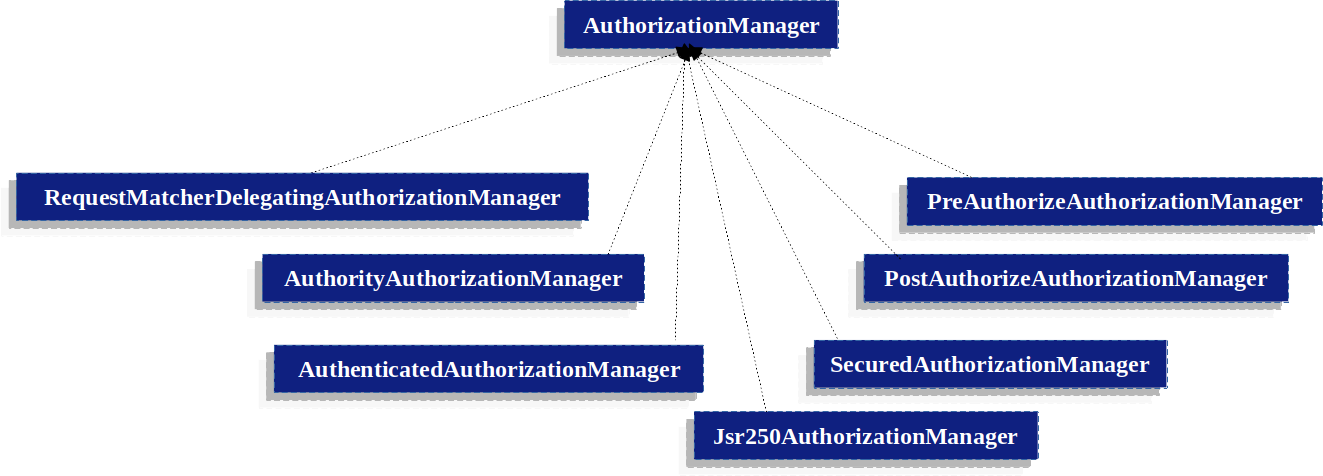
* [Authorization Architecture](https://docs.spring.io/spring-security/reference/5.7.6/servlet/authorization/architecture.html)
* [Authorize HTTP Requests](https://docs.spring.io/spring-security/reference/5.7.6/servlet/authorization/authorize-http-requests.html)
* [Authorize HTTP Requests with FilterSecurityInterceptor](https://docs.spring.io/spring-security/reference/5.7.6/servlet/authorization/authorize-requests.html)
* [Expression-Based Access Control](https://docs.spring.io/spring-security/reference/5.7.6/servlet/authorization/expression-based.html)
* [Secure Object Implementations](https://docs.spring.io/spring-security/reference/5.7.6/servlet/authorization/secure-objects.html)
* [Method Security](https://docs.spring.io/spring-security/reference/5.7.6/servlet/authorization/method-security.html)
* [Domain Object Security ACLs](https://docs.spring.io/spring-security/reference/5.7.6/servlet/authorization/acls.html)
* [Authorization Events](https://docs.spring.io/spring-security/reference/5.7.6/servlet/authorization/events.html)

### Architecture

**Authentication**的实现对象中存储了**GrantedAuthority**对象的列表，这些代表了权限。AuthenticationManager将GrantedAuthority对象插入到Authentication对象中，然后在进行授权决策时由AuthorizationManager读取。

* 老版本中使用AccessDecisionManager 和 AccessDecisionVoter进行授权决策，
* 在新的版本中使用 AuthorizationManager。

**AuthorizationManager**由AuthorizationFilter调用，并负责做出最终的访问控制决策。



**AuthorityAuthorizationManager**

Spring Security提供的最常见的AuthorizationManager是AuthorityAuthorizationManager。它配置了一组给定的权限，以便在当前Authentication上查找。如果Authentication包含任何配置的授权(Authorities)，它将返回正的AuthorizationDecision。

**自定义AuthorizationManager**

可以实现一个自定义AuthorizationManager，并且可以在其中放入您想要的任何访问控制逻辑。

### Authorize HTTP Requests

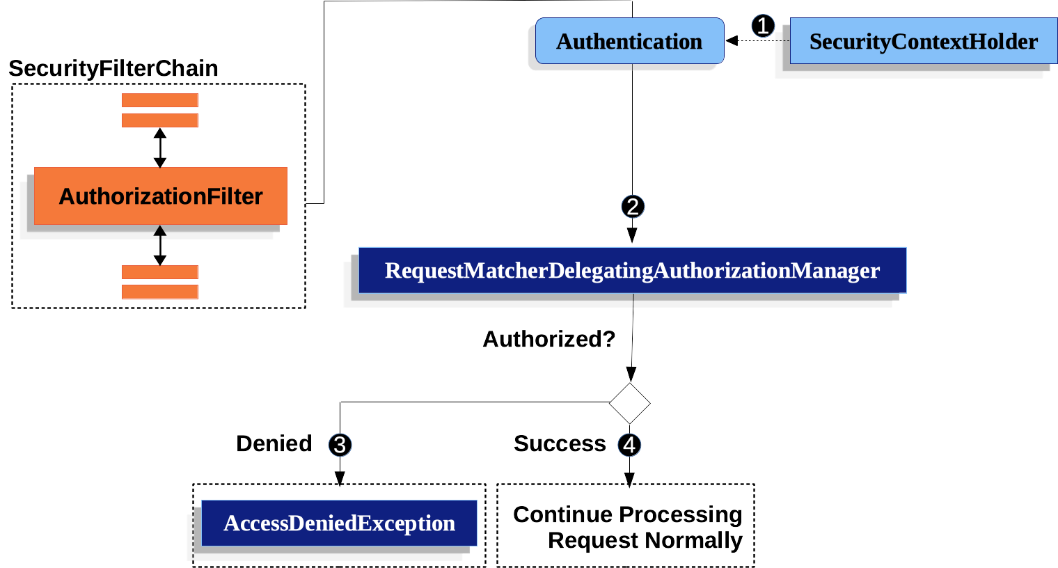
Authorize HttpServletRequests with AuthorizationFilter**。**

AuthorizationFilter取代FilterSecurityInterceptor。

注册一个自定义的SecurityFilterChain, 使用**authorizeHttpRequests** 代替**authorizeRequests**

|  |
| --- |
| @Bean  SecurityFilterChain web(HttpSecurity http) throws AuthenticationException {  http  .authorizeHttpRequests((authorize) -> authorize  .anyRequest().authenticated();  )  // ...  return http.build();  } |

当使用authorizeHttpRequests而不是authorizeRequests时，则使用AuthorizationFilter而不是FilterSecurityInterceptor。

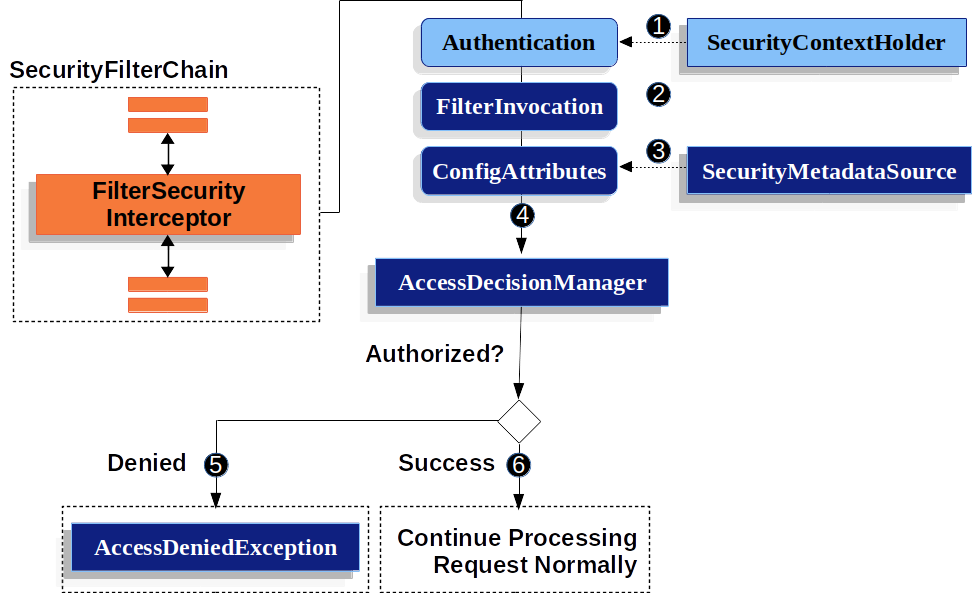


1. First, the AuthorizationFilter obtains an [Authentication](https://docs.spring.io/spring-security/reference/5.7.6/servlet/authentication/architecture.html#servlet-authentication-authentication) from the [SecurityContextHolder](https://docs.spring.io/spring-security/reference/5.7.6/servlet/authentication/architecture.html" \l "servlet-authentication-securitycontextholder). It wraps this in a Supplier in order to delay lookup.
2. Second, it passes the Supplier<Authentication> and the HttpServletRequest to the [AuthorizationManager](https://docs.spring.io/spring-security/reference/5.7.6/servlet/architecture.html" \l "authz-authorization-manager).
3. If authorization is denied, an AccessDeniedException is thrown. In this case the [ExceptionTranslationFilter](https://docs.spring.io/spring-security/reference/5.7.6/servlet/architecture.html" \l "servlet-exceptiontranslationfilter) handles the AccessDeniedException.
4. If access is granted, AuthorizationFilter continues with the [FilterChain](https://docs.spring.io/spring-security/reference/5.7.6/servlet/architecture.html" \l "servlet-filters-review) which allows the application to process normally.

**使用自定义AuthorizationManager**

|  |
| --- |
| @Bean  SecurityFilterChain web(HttpSecurity http) throws Exception {  http  .authorizeHttpRequests((authorize) -> authorize  .anyRequest.access(new **CustomAuthorizationManager**());  )  // ...  return http.build();  } |

### Authorize HttpServletRequest with FilterSecurityInterceptor



1. First, the FilterSecurityInterceptor obtains an Authentication from the SecurityContextHolder.
2. number 2 Second, FilterSecurityInterceptor creates a FilterInvocation from the HttpServletRequest, HttpServletResponse, and FilterChain that are passed into the FilterSecurityInterceptor.
3. number 3 Next, it passes the FilterInvocation to SecurityMetadataSource to get the ConfigAttributes.
4. number 4 Finally, it passes the Authentication, FilterInvocation, and ConfigAttributes to the xref:servlet/authorization.adoc#authz-access-decision-manager`AccessDecisionManager`.
5. number 5 If authorization is denied, an AccessDeniedException is thrown. In this case the ExceptionTranslationFilter handles the AccessDeniedException.
6. number 6 If access is granted, FilterSecurityInterceptor continues with the FilterChain which allows the application to process normally.

By default, Spring Security’s authorization will require all requests to be authenticated.

### Expression-Based Access Control

Spring Security使用Spring EL来支持表达式， Spring Security使用用于web和方法安全性的特定类作为根对象，以便提供内置表达式和对当前主体等值的访问。

* 通用内置表达式(Common Build-in Expressions)

hasRole(String role)

hasAnyRoles(String… roles)

hasAuthority(String authority)

hasAnyAuthority(String…​ authorities)

…

* Web安全表达式(Web Security Expressions)
* 方法安全表达式(Method Security Expressions)

@EnableMethodSecurity, @PreAuthorize, @PreFilter, @PostAuthorize and @PostFilter.

@EnableGlobalMethodSecurity

## 基于数据库的动态认证和授权案例

自定义一下Spring Beans:

* SecurityFilterChain
* UserDetailsService
* BCryptPasswordEncoder
* AuthenticationManager
* AccessDecisionManager
* FilterSecurityInterceptor
* InvocationSecurityMetadataSourceService

**注意：**

新版本中应该使用自定义的**AuthorizationManager**代替FilterSecurityInterceptor，同事不需要自定义InvocationSecurityMetadataSourceService和AccessDecisionManager。

## Spring Security OAuth2