Spring Security and OAuth 2.0

V.V.S.K Chaitanya

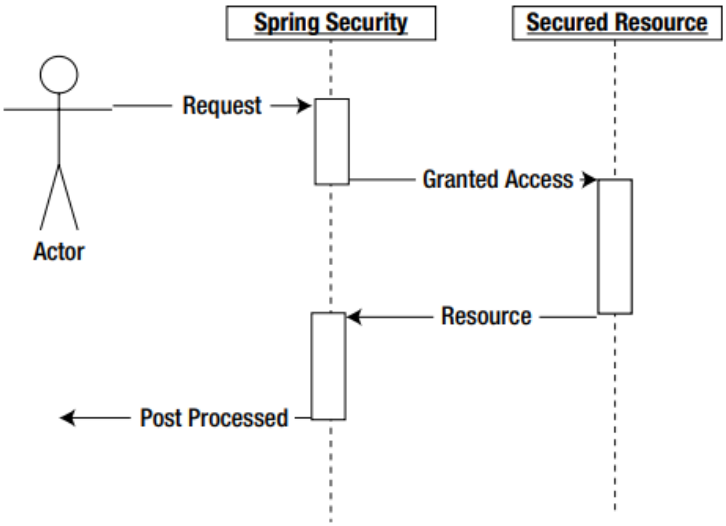
Introduction

Spring Security is a powerful and highly customizable authentication and access-control framework. It provides a comprehensive security solution for Java EE based Enterprise Software Applications.

Architecture

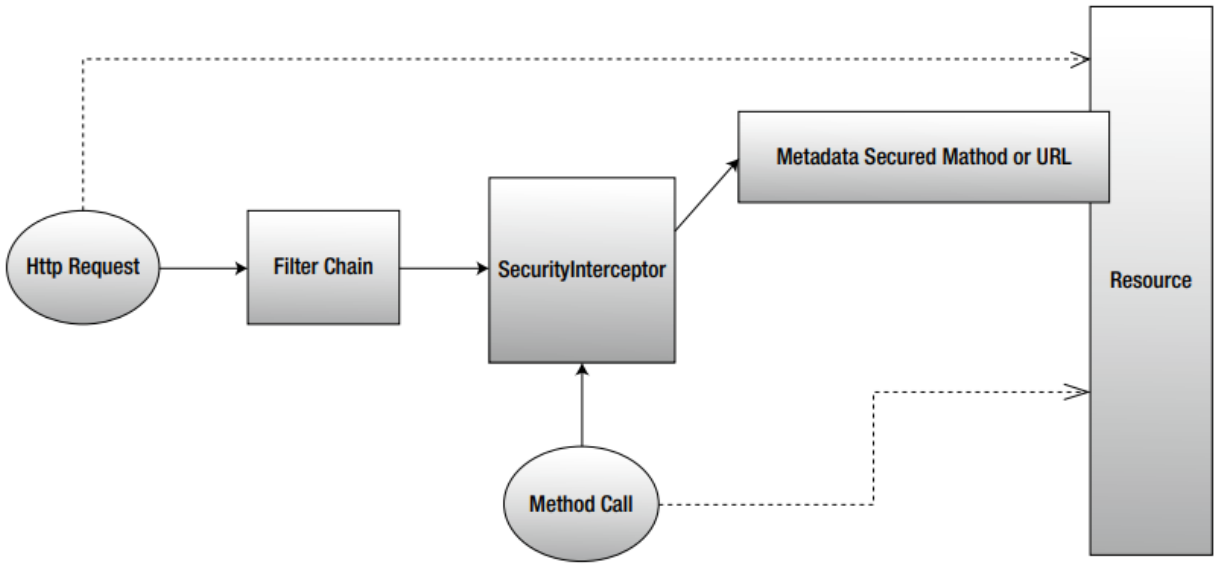
10,000 foot overview

In this view, we can assume the spring security as an extra layer built on top of our application which secures the access of the application based on a set of logics and security rules.



1,000 foot overview

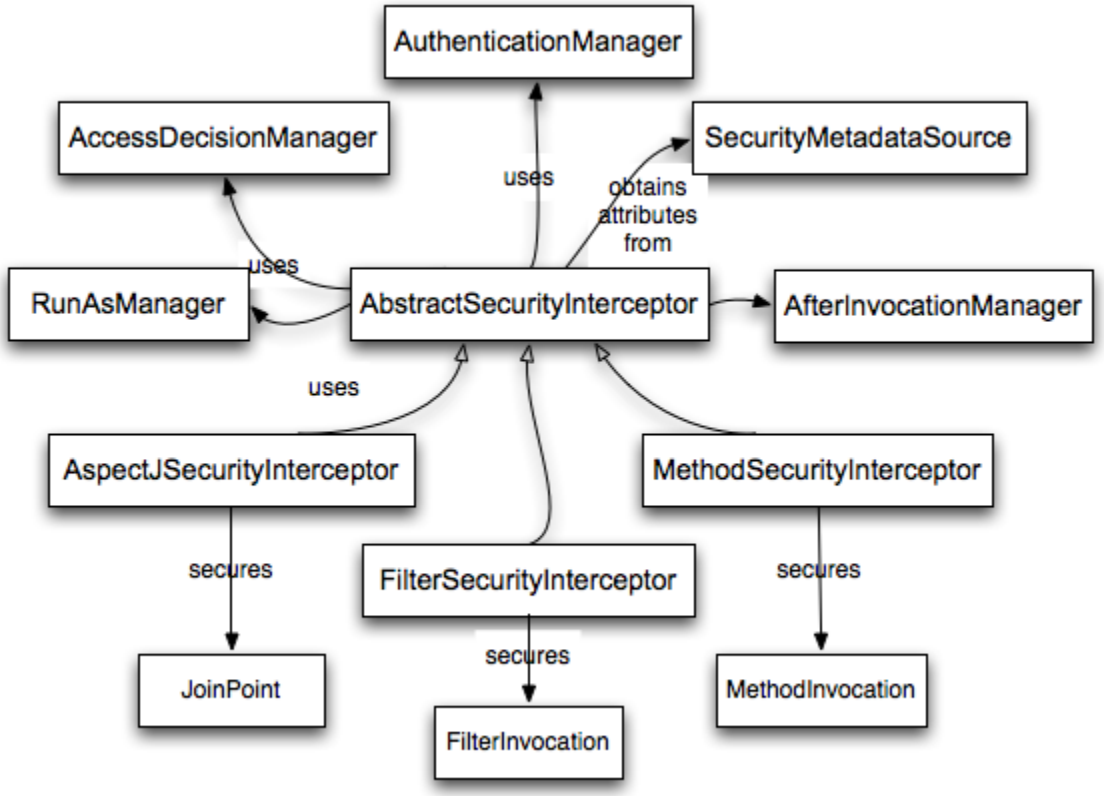
We can imagine this view as the basic architecture of the spring security layer. Any application built using spring security contains the mentioned basic security components.After this level, we can configure our own security architecture for the application. The architecture may vary depending on the security mechanism used by the application and the level of security. Spring security supports various security mechanisms which are standardized by IETF(Internet Engineering Task Force).



100 foot overview

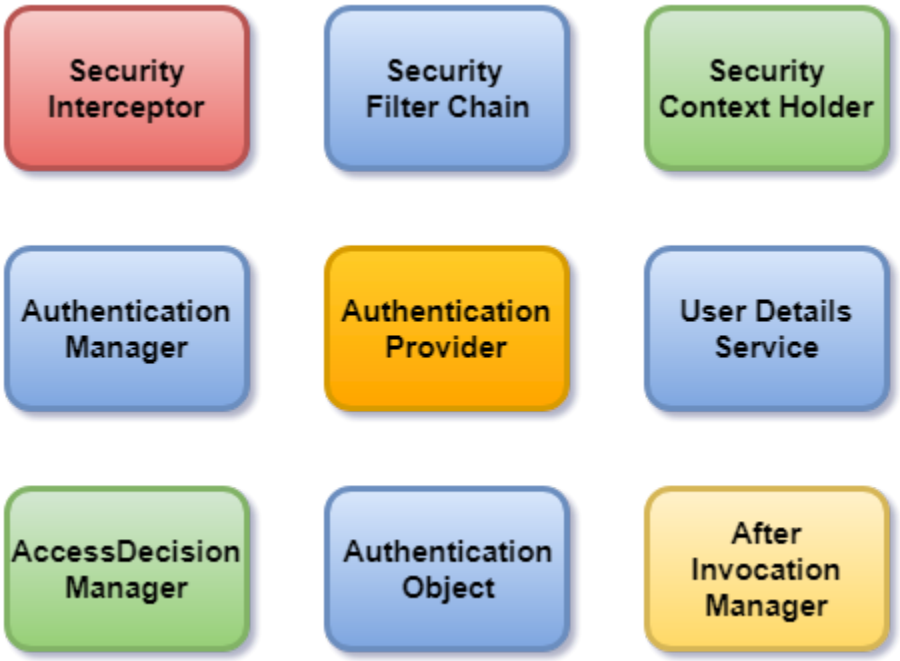
Initial version of spring security was released during 2006, it was revised many times considering the real time use scenarios and the trending spring security has a strong architecture consisting of useful and highly configurable components.

The below view describes the Abstract Security Interceptor architecture which is one of the core components of spring security.



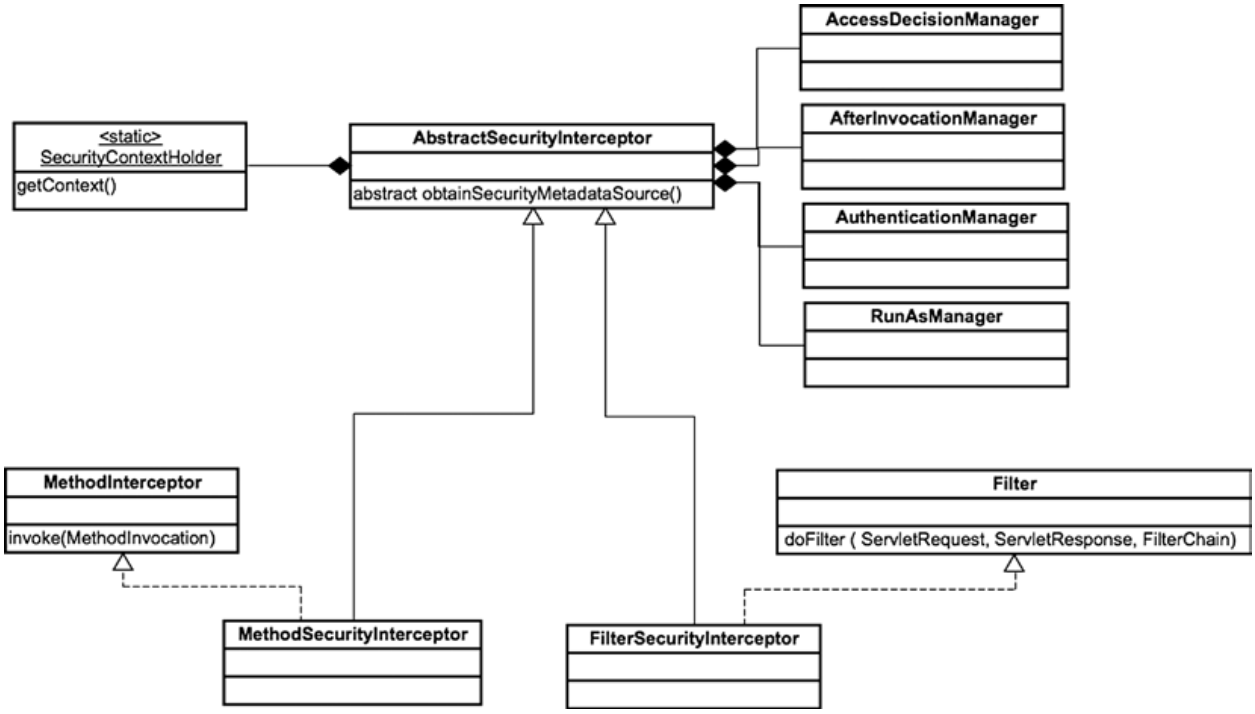
Key Components

These are the basic components which enable us to understand the authentication and authorization flow when spring security is added to the application.



Security Interceptor

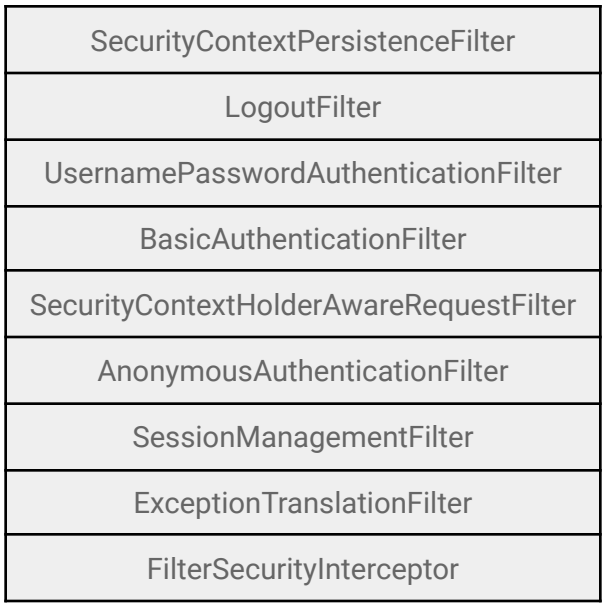
One of the most important components of the framework is the Security Interceptor. With the main logic implemented in AbstractSecurityInterceptor and with two concrete implementations in the form of FilterSecurityInterceptor and MethodSecurityInterceptor, the Security Interceptor is in charge of deciding whether a particular petition should be allowed to go through to a secured resource.



Security Filter Chain

The filter chain model is what Spring Security uses to secure web applications. This model is built on top of the standard servlet filter functionality. Working as an Intercepting Filter Pattern, the filter chain in Spring Security is built of a few single-responsibility filters that cover all the different security constraints required by the application. The filter chain in Spring Security preprocesses and post processes all the HTTP requests that are sent to the application and then applies security to URLs that require it.

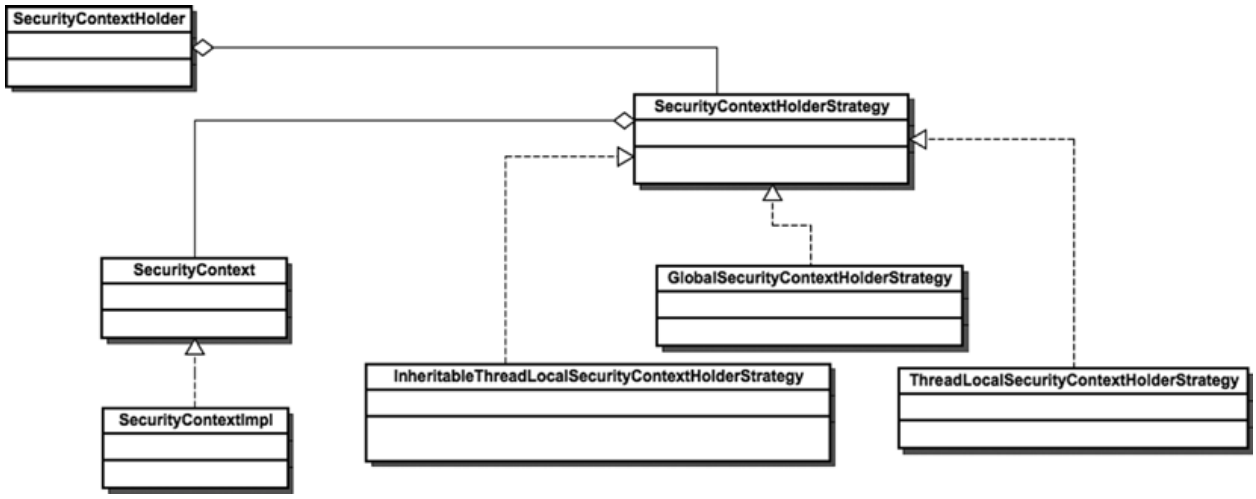
When we enable http security auto configuration in our application, the most commonly required security filters are included into our application which is called FilterChainProxy.



Security Context and Security Context Holder

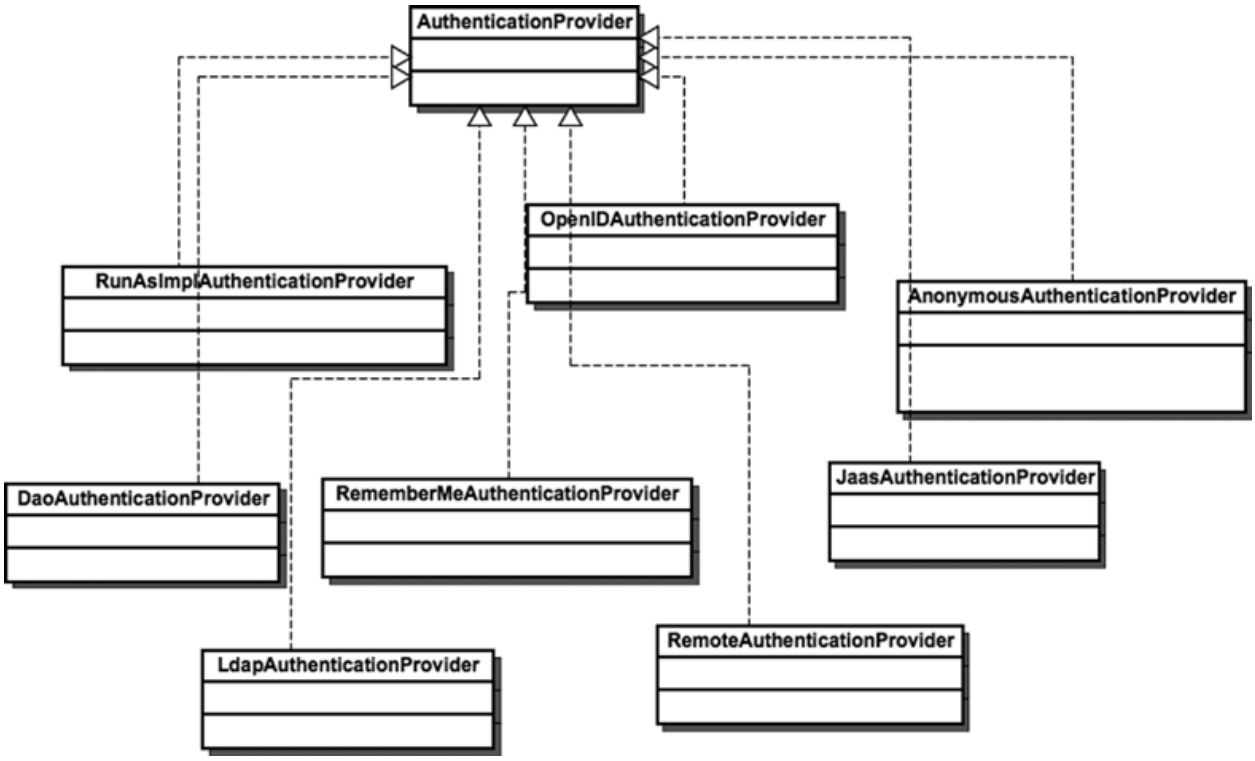
SecurityContext is the place where Spring Security stores the valid Authentication object, associating it with the current thread.

SecurityContextHolder is the class used to access SecurityContext from many parts of the framework. It is built mainly of static methods to store and access SecurityContext, delegating to configurable strategies the way to handle this SecurityContext.



Authentication Manager and Authentication Provider

The main interface which provides authentication services in Spring Security is the Authentication Manager (instance of Spring Security ProviderManager). We can build an Authentication Manager by registering Authentication Provider(s). Some of the Authentication Providers which are supported by spring are shown below..



Access Decision Manager

AccessDecisionManager is in charge of deciding if a particular Authentication object is allowed or not allowed to access a particular resource. In its main implementations, it delegates to AccessDecisionVoter objects, which basically compares the Granted Authorities in the Authentication object against the ConfigAttribute(s) required by the resource that is being accessed, deciding whether or not access should be granted. They emit their vote whether to allow access or not. The AccessDecisionManager implementations take the output from the voters into consideration and apply a determined strategy on whether or not to grant access.

User Details Service

UserDetailsService is responsible for loading the user information from the underlying user store (in-memory, database, and so on) when an authentication request arrives in the application. UserDetailsService makes use of the provided user name for looking up the rest of the required user data from the datastore.

public interface UserDetailsService {

UserDetails loadUserByUsername(String username) throws UsernameNotFoundException;

}

OAuth 2.0

OAuth 2 is an authorization framework that enables applications to obtain limited access to user accounts.OAuth 2.0 focuses on client developer simplicity while providing specific authorization flows such as Authorization Code, Implicit, Client Credentials, Password Credentials Flow for web, desktop and mobile apps.

Authorization Code Flow

The Authorization Code Grant Type is probably the most common of the OAuth 2.0 grant types that you'll encounter. It is used by both web apps and native apps to get an access token after a user authorizes an app.

Client Credentials Flow

The Client Credentials grant is used when applications request an access token to access their own resources, not on behalf of a user.

UserName Password Flow

The OAuth 2.0 Password Grant Type is a way to get an access token given a username and password. It's typically used only by a service's own mobile apps.

References

https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/

https://the-eye.eu/public/Books/IT%20Various/pro_spring_security.pdf

https://projects.spring.io/spring-security-oauth/docs/oauth2.html

https://spring.io/guides/tutorials/spring-boot-oauth2/

https://oauth.net/2/

https://tools.ietf.org/html/rfc6749