Spring Framework

Entendiendo el framework

De todos los módulos de Spring, uno de los más utilizados y que más dudas genera es el framework Spring Security ya que a veces parece que es inmenso y muchas personas no son expertas en seguridad.

Con Spring Security podemos gestionar todo lo relativo a la seguridad de nuestra aplicación web, desde el protocolo de seguridad, hasta los roles que necesitan los usuarios para acceder a los diferentes recursos de la aplicación.

Toda la información necesaria, se puede obtener leyendo la documentación oficial de Spring. Por supuesto, la cantidad de información que aparece ahí es enorme, lo que hace tedioso su lectura. Además, para una configuración básica, no nos hace falta saber cómo configurar Spring Security para que, por ejemplo, se autentique contra un LDAP o un CAS.

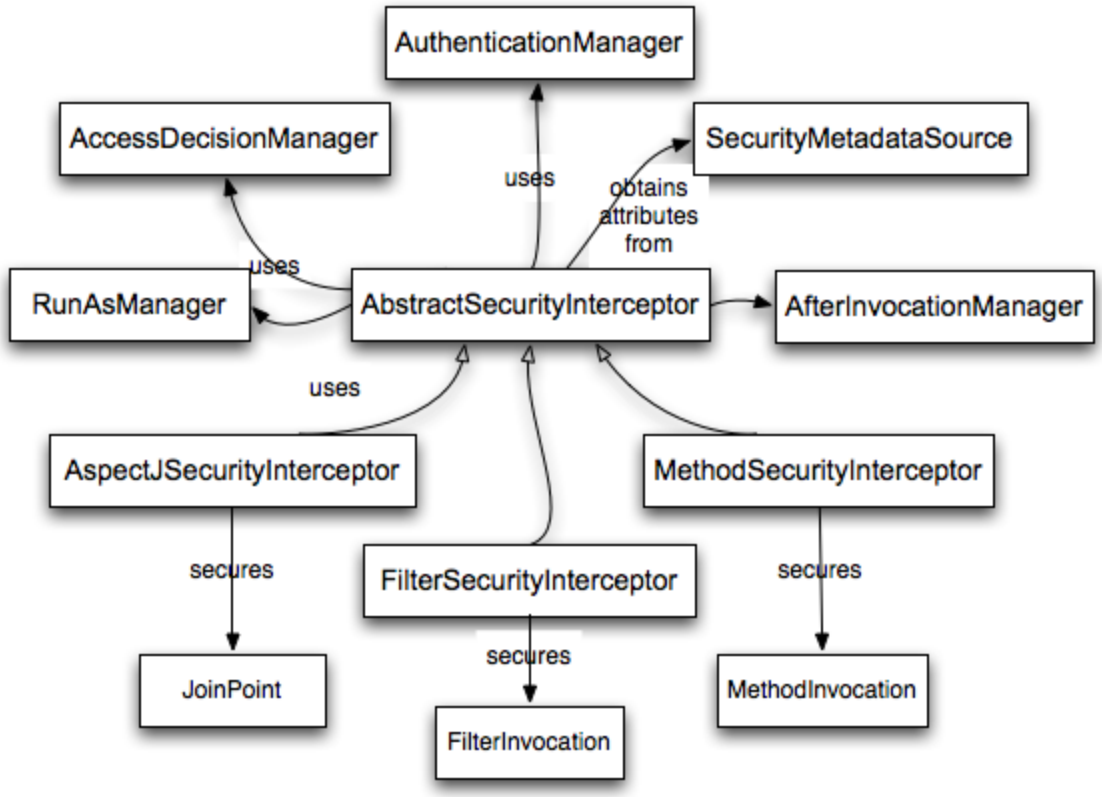
Spring Security puede llegar a facilitar mucho las cosas dependiendo de nuestras necesidades. Se podría decir que Spring Security es declarativo y apenas hace falta programar nada y todo se configura mediante un archivo de configuración o anotaciones. Esto llega a tal punto que incluso no es necesario crear una página de Login, ya que Spring Security lo hace por ti. Lo único que hay que enseñarle es a recuperar los datos del usuario que se está intentando loguear en nuestra aplicación. Si nuestra aplicación lo requiere o necesitamos personalizar algunos aspectos, Spring Security también lo permitirá indicándoselo en las configuraciones.

Existen dos formas de configurar Spring Security: Una 100% manual y otra más automática. La forma manual consiste en definir todos los filtros necesarios para su correcto funcionamiento.

La otra forma, consiste en usar el elemento http. Este elemento configurará por nosotros todos los filtros obligatorios, beans y la cadena de seguridad. Si además se le pone el atributo auto-config a true, configurará todos los filtros restantes necesarios para el funcionamiento.

El primero de los beans que se crea es el springSecurityFilterChain, que es el encargado de gestionar los diferentes filtros de la cadena de seguridad.

Se pueden personalizar los filtros que se configuran automáticamente e incluso añadir los nuestros propios. Esto se realiza anidando un nuevo elemento en el http. Con el custom-filter se pueden añadir nuevos filtros, así como posicionarlos donde sea necesario en la cadena de seguridad. Es posible que algunos de los nuevos filtros entren en conflicto con algunos de los existentes. Esto ocurrirá si estamos sobre escribiendo un filtro existente, con lo cual, habrá que poner el atributo auto-config a false.



Se define también el filtro de autenticación (authenticationProcessingFilter), que será el que permita generar la sesión, el token, etc en el SecurityContext.

El primer y fundamental Bean es el AuthenticationManager. Este es el encargado de recuperar la información de usuario que se está intentando loguear en el sistema. Esto lo consigue mediante un provider que es necesario definir. Este provider es el que proporciona al AuthenticationManager los detalles de nuestro usuario. Así, el provider construye un objeto UsernamePasswordAuthenticationToken si la contraseña es la correcta y se lo pasa al AuthenticationManager. Éste será el que se encargue de rellenar el SecurityContextHolder. Se pueden definir varios providers según sea

necesario, e incluso definir uno propio.

Otra propiedad del AuthenticationManager es authenticationSuccessHandler. De la forma mostrada, indicando SavedRequestAwareAuthenticationSuccessHandler, se redirigirá a la página que se solicitaba originalmente antes de la autenticación. Se proporciona el default por si no estaba pidiendo ninguna página en concreto.

El elemento http es el más cómodo de usar en un principio. Si fuera necesario añadir más elementos de este tipo, para cada uno de ellos será necesario indicarle el atributo pattern, ya que de no existir, se asume el universal (pattern="/**"), lo que capturaría todas las peticiones. En las urls no debe de incluirse el contextPath