# Algunas clases e interfaces de Spring Security

Implementa la seguridad de tu API Rest con Spring Boot

# Clases e interfaces

- En la lección anterior hemos nombrado algunas clases o interfaces.
- Dada la amplitud de Spring Security, no podemos conocer todas.
- Nos centramos en las más comunes o las que más usaremos.

# WebSecurityConfigurerAdapter

- Clase base para nuestra clase de configuración de seguridad web.
- Suele venir acompañada de @Configuration + @EnableWebSecurity.
- Métodos convenientes para configurar la autenticación y la autorización.

# @EnableWebSecurity

- Sirve para conmutar (apagar) la configuración por defecto aplicada por Spring Boot, y añadir la nuestra.
- Se utiliza anotando una clase que extienda a *WebSecurityConfigurerAdapter*.

# Authentication

- Interfaz que extiende a Principal
- Representa un token para realizar la autenticación, o para un principal una vez autenticado.
- Normalmente se almacena en el contexto de seguridad (*SecurityContext*) manejadop por el (*SecurityContextHolder*)
- Spring Security tiene decenas de clases que lo implementan
- **Interfaz nuclear en la autenticación**.

# AuthenticationManagerBuilder

- *Builder* utilizado para construir un AuthenticationManager.
- Permite construir, fácilmente, un AuthenticationManager en memoria, LDAP, JDBC o con UserDetailsService.
- Se suele configurar sobrescribiendo el método *configure(AuthenticationManagerBuilder)* de la clase *WebSecurityConfigurerAdapter*.

# UserDetails

- Interfaz que representa la información nuclear de un usuario.
- Almacena la información que posteriormente será encapsulada en un objeto de tipo *Authentication*.
- Las implementaciones de esta interfaz deben verificar bien cada método, para saber qué atributos no deben ser nulos.
- Es implementado por la clase *org.springframework.security.core.userdetails.User*.

# User

- Objeto modelo que incluye la información de un usuario obtenido por un UserDetailsService.
- Se puede usar directamente, extenderla o implementar la interfaz *UserDetails*.
- La implementación de equals y hashCode se basa en el atributo *username*.
- Incluye las *Authorities* del usuario.

# GrantedAuthority

- Representa un privilegio individual.
- Se pueden usar con perspectiva de *grano fino*.
    - CAN_READ_SOME_ENTITY_PROPERTY
- El nombre es arbitrario.
- Solo un método, que devuelve la representación como String.
- En ocasiones, pueden representar a un rol con el prefijo *ROLE_*

# SimpleGrantedAuthority

- Implementación concreta y muy básica de *GrantedAuthority*.
- Almacena una representación en un String de una *authority* concedida a un objeto de tipo *Authentication*.

# UserDetailsService

- Interfaz que es capaz de cargar la información de un usuario.
- Se puede utilizar como DAO (Data Access Object).
- Es utilizado por DaoAuthenticationProvider (un *AuthenticationProvider* que obtiene la información de los usuarios a través de un *UserDetailsService*).
- Un solo método, *UserDetails loadUserByUsername(String)*.
- Se suele utilizar cuando almacenamos la información de los usuarios a través de Spring Data y el uso de entidades.