Autenticación y Autorización

Implementa la seguridad de tu API Rest con Spring Boot

Autenticación y autorización

- La seguridad de una aplicación suele reducirse a dos problemas más o menos independientes
  - Autenticación: ¿quién es usted?
  - Autorización: ¿qué se le permite hacer?
- En ocasiones se llama control de acceso a la autorización.

Autenticación

Autenticación

- Spring Security proporciona un interfaz, *AuthenticationManager*, que implementa el patrón estrategia.

```
public interface AuthenticationManager {

  Authentication authenticate(Authentication authentication)
    throws AuthenticationException;

}
```

Autenticación

- Un *AuthenticationManager* puede hacer tres cosas con su único método:
  - Devolver un *Authentication* (normalmente con *authenticated=true*)
  - Lanzar una excepción de tipo *AuthenticationException*
  - Devolver *null*

Autenticación

- La implementación más usada de *AuthenticationManager* es *ProviderManager*, el cual delega en una cadena de instancias de tipo *AuthenticationProvider*.
- Un *AuthenticationProvider* se parece a un *AuthenticationManager*, ya que solo añade un nuevo método que permite verificar si la instancia soporta un determinado tipo de *Authentication*.
- Un *ProviderManager* puede soportar diferentes mecanismos de autenticación en una sola aplicación.

Autenticación

- Un *ProviderManager* puede tener un padre, que puede consultar si todos sus *provider* han devuelto null.
- Si no hay un padre disponible, una respuesta null se transforma en una excepción (AuthenticationException).
- En ocasiones, se puede tener grupos de recursos (por ejemplo, recursos web en un determinado path), y cada grupo tener su propio *AuthenticationManager*. Si establecemos una jerarquía, algunos grupos podrían compartir un padre como mecanismo global de autenticación.

AuthenticationManagerBuilder

- Spring Security ofrece algunos mecanismos rápidos de configuración de un *AuthenticationManager*.
- El más común es el uso de un *AuthenticationManagerBuilder*. Este permite configurar rápidamente
  - Autenticación en memoria
  - JDBC
  - LDAP
  - Un servicio de UserDetailsServices personalizado

Autorización o control de acceso

Autorización

- Una vez que la autenticación ha sido exitosa, pasamos al control de acceso, a través de la interfaz *AccessDecisionManager*.
- Hay 3 implementaciones de esta interfaz, y todas delegan en una cadena de *AccessDecisionVoter* (algo así como el *ProviderManager*).
- Un *AccessDecisionVoter* considera un *Authentication* y un *objeto* seguro (este objeto es genérico y puede representar cualquier cosa, como un recurso web). El objeto seguro es decorado a través de una colección de *ConfigAttributes*.

Autorización

- Los *ConfigAttributes* decoran un objeto con metadatos para determinar el nivel de permisos requeridos para acceder a él.
- La interfaz *ConfigAttribute* es muy sencilla, y tiene solo un método que devuelve un String.
- Esta cadena codifica, de alguna forma, la intención del propietario del recurso. Por ejemplo, es típico el nombre de un rol, como *ROLE_ADMIN* o *ROLE_AUDIT*.

Autorización

- También es común usar *ConfigAttributes* basados en expresiones SpEL, como *isAuthenticated()* o *hasRole('THEROLE')*.

Configuración de la autorización

- Se pueden configurar estos *AccessDecisionVoter* a través de diferentes mecanismos, que utilizaremos a lo largo del curso. Entre otros:
    - Extendiendo la clase WebSecurityConfigurerAdapter y el uso de *AntMatchers* (patrones de rutas)
    - A través de anotaciones (*@PreAuthorize* y *@PostAuthorize*)