Spring boot

Spring Security

Introduction

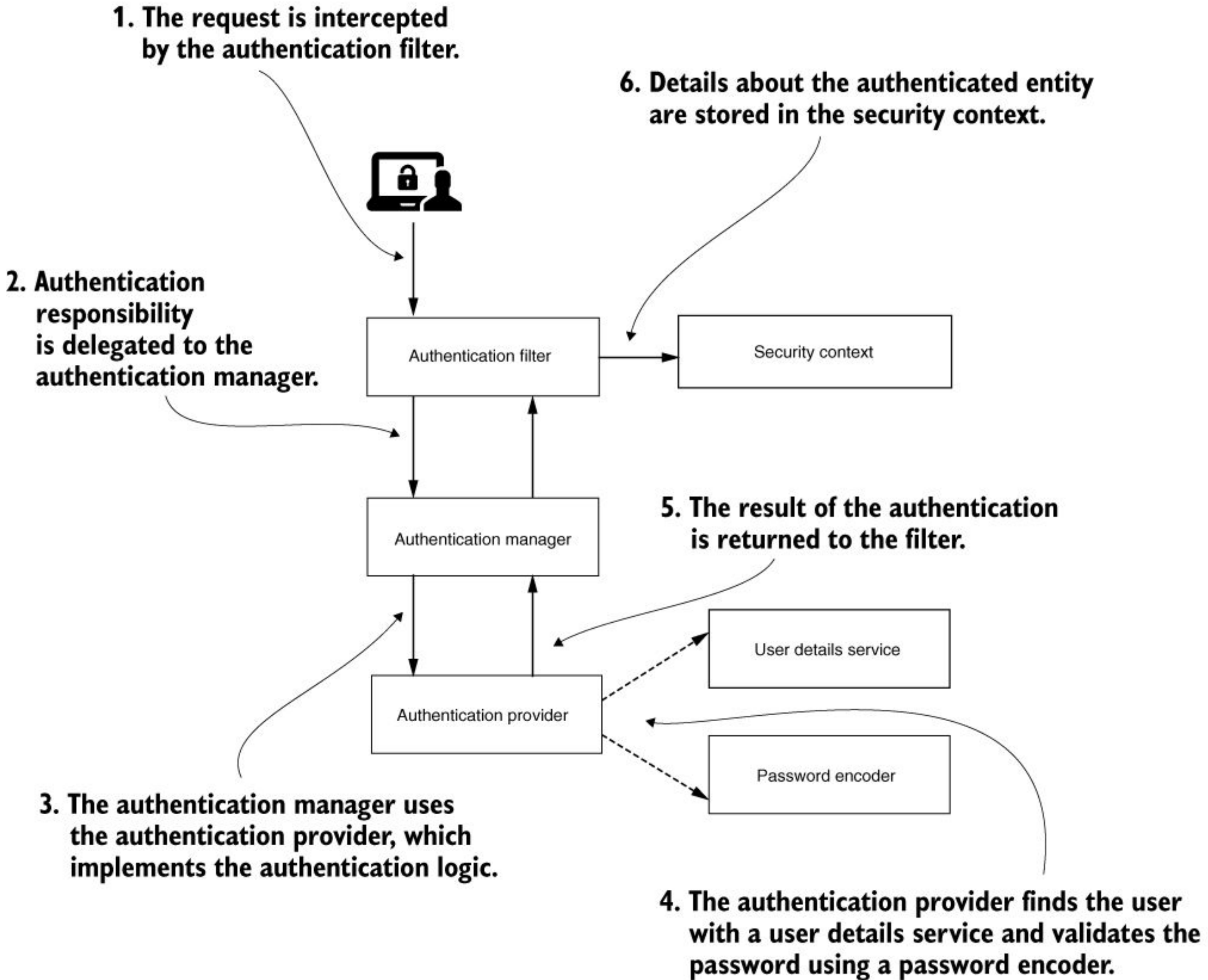
- What is software security ?
- Why is security important ?
- Spring security.
    - Source: https://github.com/spring-projects/spring-security/

Spring Security

- Architecture.
- Default Configuration.
- Overriding default configuration.
- Managing users.
- Dealing with password.

Architecture

- Authentication Filter
- Authentication Manager
- Authentication Provider
- Security Context
- User Details Service
- Password Encoder


1. The request is intercepted by the authentication filter.
2. Authentication responsibility is delegated to the authentication manager.
3. The authentication manager uses the authentication provider, which implements the authentication logic.
4. The authentication provider finds the user with a user details service and validates the password using a password encoder.
5. The result of the authentication is returned to the filter.
6. Details about the authenticated entity are stored in the security context.
Authentication filter
Security context
Authentication manager
Authentication provider
User details service
Password encoder

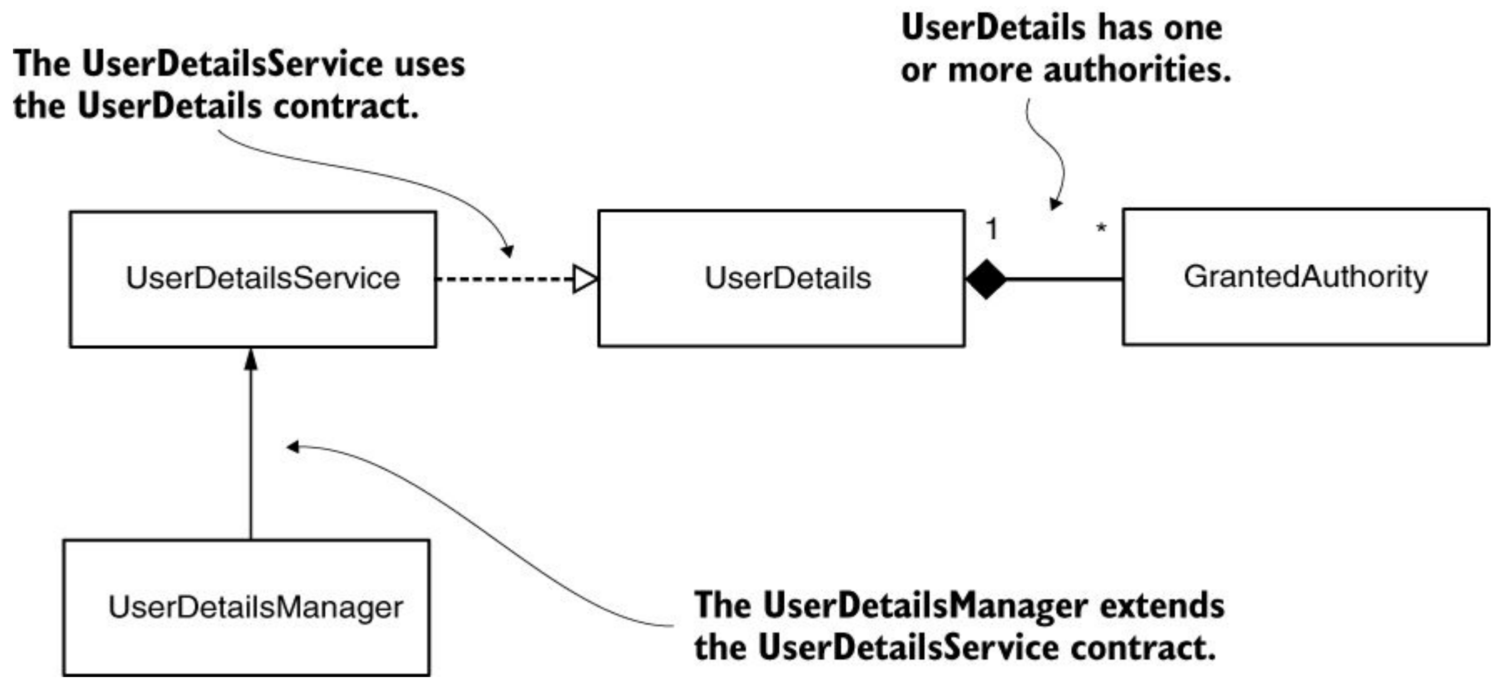
Default configuration

- The authentication filter delegates the authentication request to authentication manager.
- The authentication manager uses the authentication provider to process authentication.
- The authentication provider implements the authentication logic.
- The user details service implements user management.
- The password encoder implements password management.
- The security context keeps the authentication data after the authentication process.

Overriding default configuration

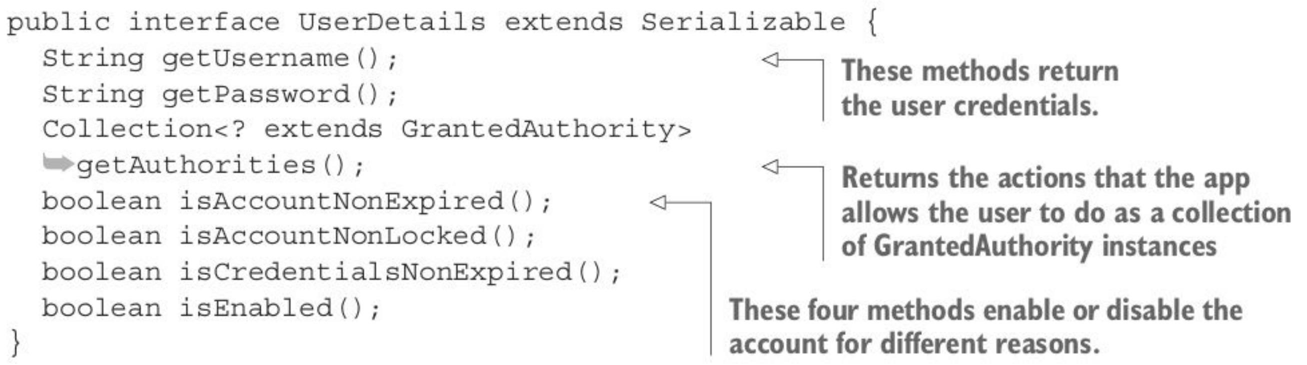
- Overriding UserDetailsService component.
- Overriding endpoint authorization configuration.
- Overriding AuthenticationProvider implementation.

Managing users


The UserDetailsService uses the UserDetails contract.
UserDetails has one or more authorities.
UserDetailsService
UserDetails
1
*
GrantedAuthority
UserDetailsManager
The UserDetailsManager extends the UserDetailsService contract.

Managing users

- UserDetails



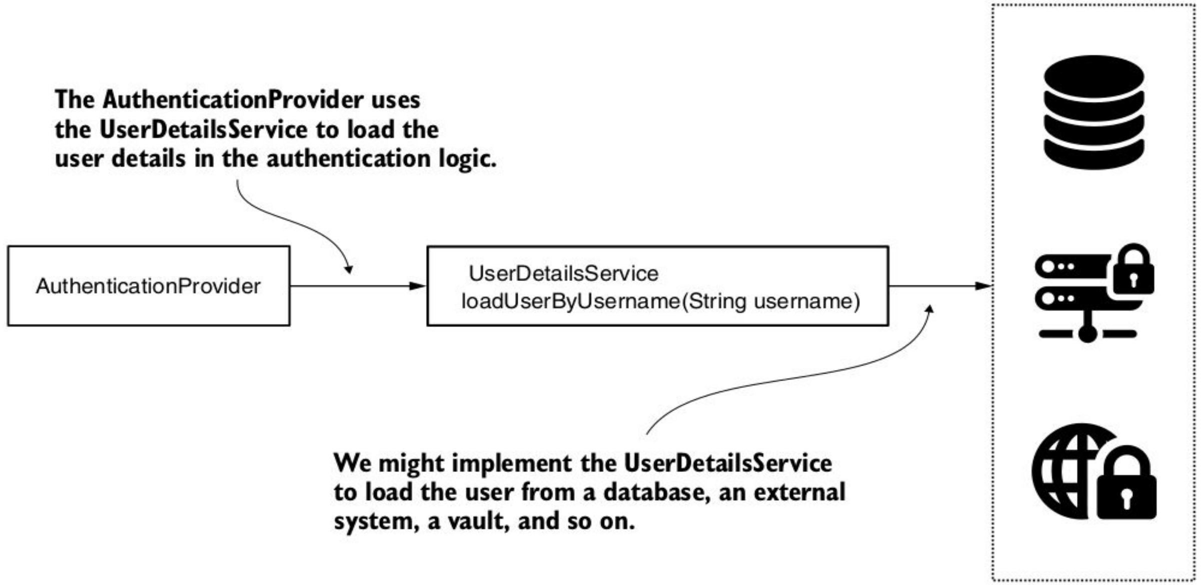
```
public interface UserDetails extends Serializable {
  String getUsername();
  String getPassword();
  Collection<? extends GrantedAuthority>
  ➥getAuthorities();
  boolean isAccountNonExpired();
  boolean isAccountNonLocked();
  boolean isCredentialsNonExpired();
  boolean isEnabled();
}
```

These methods return the user credentials.
Returns the actions that the app allows the user to do as a collection of GrantedAuthority instances
These four methods enable or disable the account for different reasons.

Managing users

- GrantedAuthority
  - The authorities represent what the user can do in your application.
  - Without authorities, all users would be equal.
- How to manage users
  - UserDetailsService.
  - UserDetailsManager.
  - JdbcUserDetailsManager.

UserDetailsService


The AuthenticationProvider uses the UserDetailsService to load the user details in the authentication logic.
AuthenticationProvider
UserDetailsService
loadUserByUsername(String username)
We might implement the UserDetailsService to load the user from a database, an external system, a vault, and so on.

Dealing with password

- PasswordEncoder
- PasswordEncoder Implementations.
  - NoOpPasswordEncoder
  - StandardPasswordEncoder
  - Pbkdf2PasswordEncoder
  - BCryptPasswordEncoder
  - SCryptPasswordEncoder
- DelegatingPasswordEncoder