Spring Security

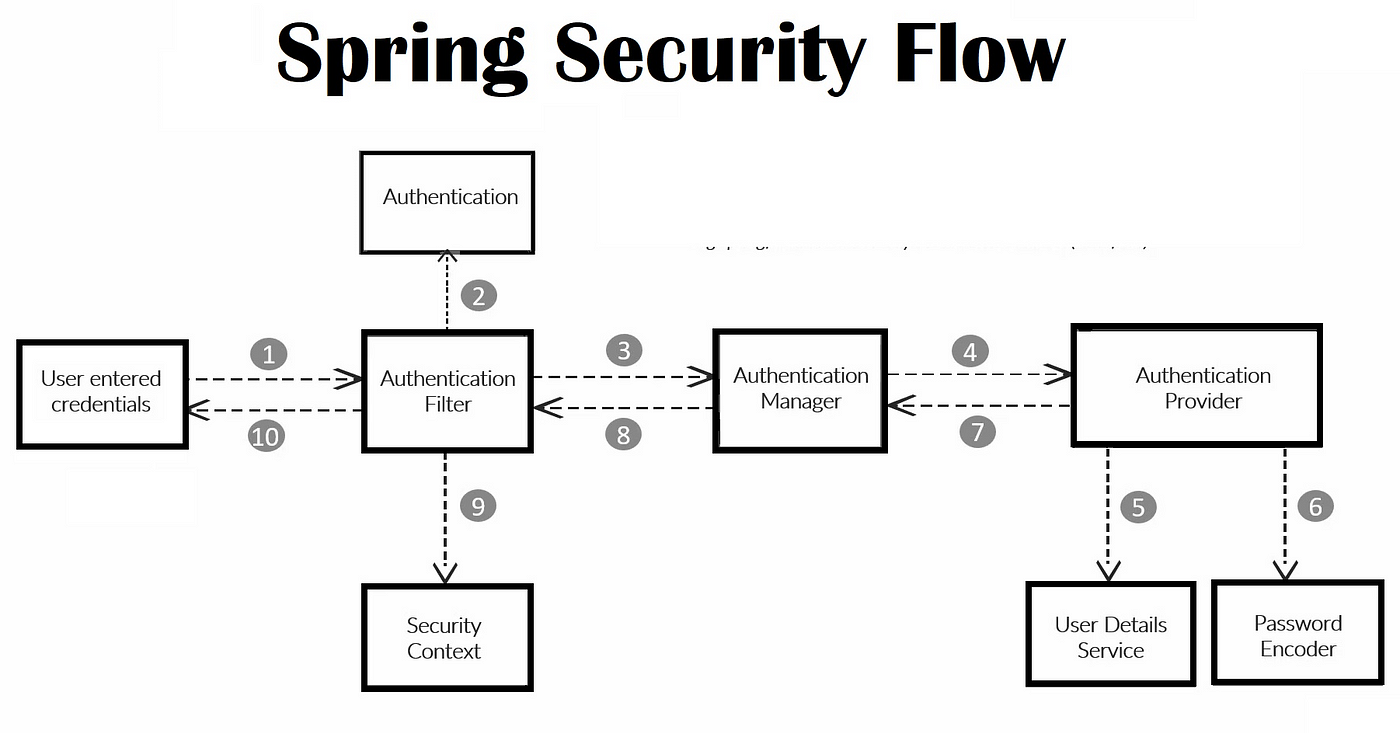
Spring Security is a powerful and customizable authentication and access control framework for Java applications built on the Spring Framework

Some of the key features of Spring Security include:  
1. Authentication  
2. Authorization  
3. Protection against common security vulnerabilities  
4. Session Management  
  
**Authentication:** Authentication is the process of recognizing a user's identity; i.e, verifying that if the user is who he claims to be, typically done with username and password**.**

**Authorization:** Authorization is the security mechanism to establish the access level/privileges for users to access resources like files, services, data etc.

# How Spring Security Works Internally

**Filter Chain -> Authentication Manager -> Authentication Provider**



Spring Security Flow:

1. Filter Chain intercepts incoming request before forwarding it to the Dispatcher Servlet.
2. Request goes to the authentication filters
3. The filter extracts the username and password from the request
4. A UsernamePasswordAuthenticationToken is created with the credentials, and request sent to authentication manager
5. The AuthenticationManager authenticate() method is invoked
6. The AuthenticationManager authenticate() method implementation will then try to authenticate using one of the AuthenticationProvider it has
7. If one of the authentication providers is able to authenticate successfully, it returns a complete UsernamePasswordAuthenticationToken holding the credentials and authorities
8. This token which is returned from the provider is used to set the user as authenticated in the Spring Security Context.

## Different Main Components of Spring Security Architecture

### Filters:

The Security Filter Chain is a key component of Spring Security responsible for processing incoming HTTP requests and applying various security measures. It consists of a series of filters organized in a specific order to handle different aspects of security such as authentication, authorization, CSRF protection, session management, and more. Each filter performs a specific task in the authentication and authorization process

1. SecurityContextPersistenceFilter
2. UsernamePasswordAuthenticationFilter
3. BasicAuthenticationFilter
4. SessionManagementFilter
5. CsrfFilter
6. LogoutFilter

### Authentication Manager

The AuthenticationManager is an interface which processes the Authentication request. It has an authenticate() method which takes Authentication object as a parameter.

The implementation class of AuthenticationManager is the ProviderManager class which provides the logic for authenticate() method

### Authentication Provider Manager

This class implements the AuthenticationManager interface and overrides the authenticate() method. It uses a list of AuthenticationProviders to authenticate the sent credentials in the Authentication object. If one of the AuthenticationProvider supports the type of authentication, it will be used to authenticate the user. If none of the providers support the authentication type, an AuthenticationException will be thrown.

### Authentication Provider

An authentication provider in Spring Security is responsible for verifying the credentials of a user and authenticating them. It's the component that performs the actual authentication process. Spring Security supports various authentication mechanisms, and different authentication providers are available to handle different types of authentication.

The AuthenticationProvider interface has two methods:

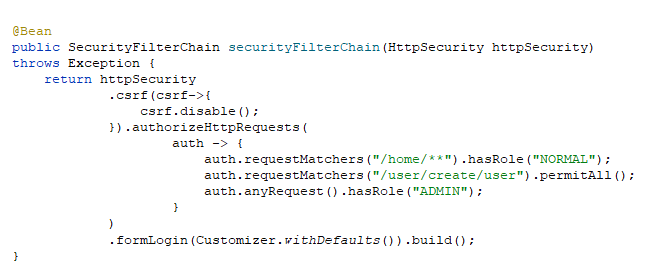
**authenticate**() : This method takes an Authentication object as input and returns an authenticated Authentication object if the credentials are valid. If the credentials are invalid, the AuthenticationProvider should throw an AuthenticationException.

**supports**() : This method takes an Authentication object as input and returns true if the AuthenticationProvider can authenticate the object. If the AuthenticationProvider cannot authenticate the object, it should return false

**Note:** In Spring Security, the default authentication provider is typically the **DaoAuthenticationProvider**. This provider is commonly used for authenticating users against a data source, such as a database.

## Security Filter Chain

It is bean in which all the authentication and authorization is done with the help of HttpSecurity class



## Password Encoder

## UserDetails Object

## UserDetailsService Interface

## Roles