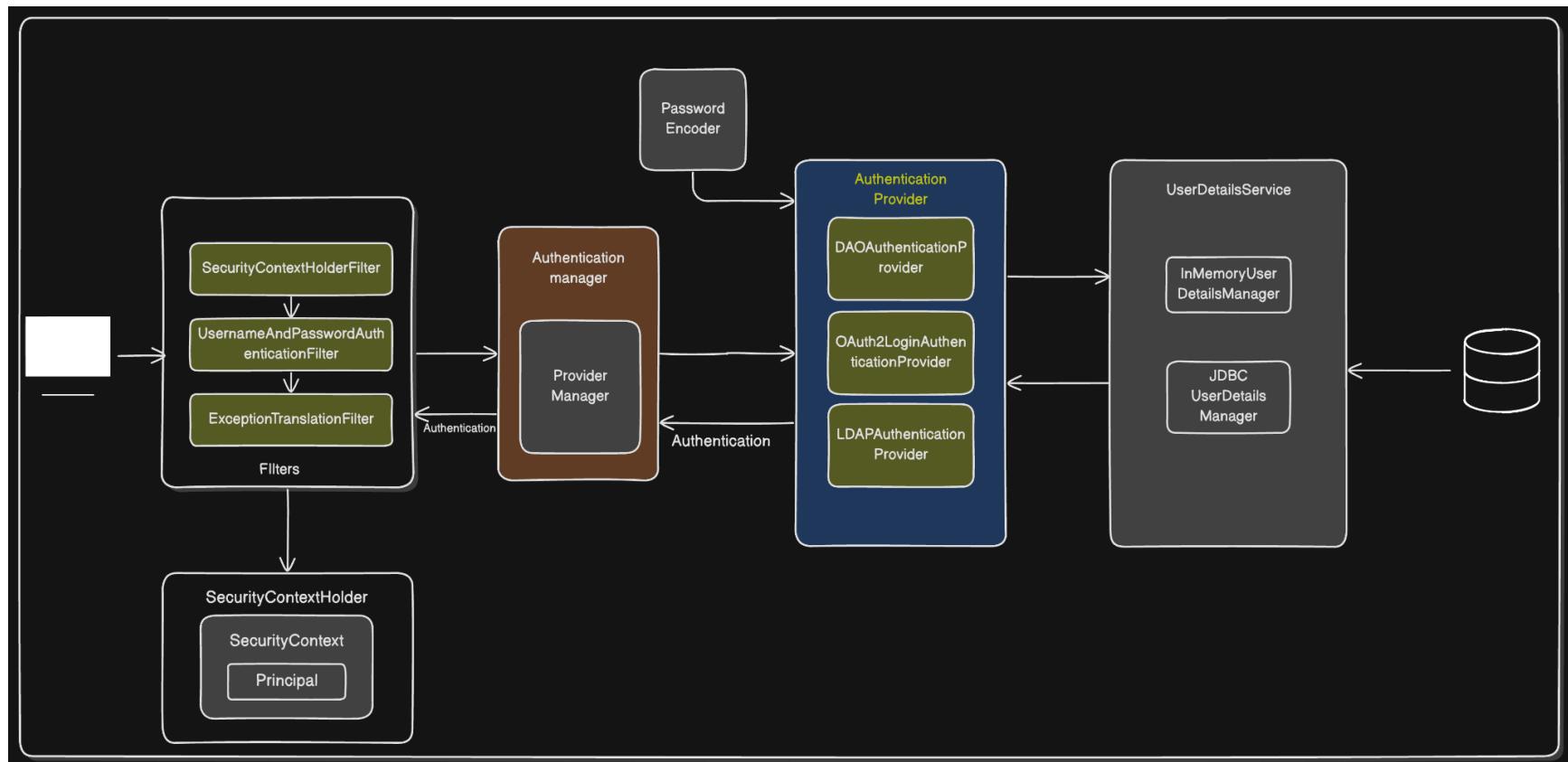


Spring Security Architecture



Video Link : <https://yt.openinapp.co/mkj8x>

Spring Security Architecture Overview

This document explains all components shown in your Spring Security architecture diagram, focusing on the flow of authentication and the role of each module.

1. Filters Layer

Spring Security uses a chain of servlet filters. Each filter performs a specific security-related function.

SecurityContextHolderFilter

- The entry point for security context management.
- Retrieves the current `SecurityContext` (if present) and places it in `SecurityContextHolder`.
- Ensures security context is available for the request lifecycle.

UsernameAndPasswordAuthenticationFilter

- Intercepts login requests (typically `/login`).
- Extracts username and password.
- Creates a `UsernamePasswordAuthenticationToken` and sends it to the `AuthenticationManager`.

ExceptionTranslationFilter

- Catches security-related exceptions thrown during filter processing.
 - Handles `AuthenticationException` and `AccessDeniedException`.
 - Redirects to login page or returns 403 depending on context.
-

2. SecurityContextHolder

The central storage for all security information during a request.

SecurityContext

- Holds the currently authenticated user's details.
- Contains the `Authentication` object.

Principal

- The user identity object extracted from the `Authentication` object.
 - Usually an instance of `UserDetails`.
-

3. Authentication Manager

Responsible for orchestrating authentication.

ProviderManager

- The default implementation of `AuthenticationManager`.
 - Iterates over a list of `AuthenticationProvider` instances.
 - Delegates authentication until one provider successfully authenticates.
-

4. Authentication Providers

Each provider knows how to validate a specific type of authentication.

DAOAuthenticationProvider

- Uses `UserDetailsService` and `PasswordEncoder`.
- Loads user details by username.
- Validates password.

OAuth2LoginAuthenticationProvider

- Handles OAuth2 login flows (Google, GitHub, etc.).
- Validates tokens and retrieves user info from the OAuth2 provider.

LDAPAuthenticationProvider

- Integrates with LDAP directories.
 - Validates credentials and loads user authorities.
-

5. Password Encoder

- Responsible for hashing and verifying passwords.
 - Common implementations include [BCryptPasswordEncoder](#).
 - Always applied by providers like [DAOAuthenticationProvider](#).
-

6. UserDetailsService Layer

Resolves user information during authentication.

InMemoryUserDetailsManager

- Stores user details in memory.
- Useful for testing and simple setups.

JDBCUserDetailsManager

- Fetches users from a relational database.
- Uses predefined or custom SQL queries.

7. Database

- Stores user credentials and authorities when using [JDBCUserDetailsManager](#).
 - Authentication provider fetches user data from here.
-

Authentication Flow Summary

1. Request enters the filter chain.
 2. `UsernamePasswordAuthenticationFilter` extracts credentials.
 3. Filter delegates to `AuthenticationManager`.
 4. `AuthenticationManager` calls each `AuthenticationProvider`.
 5. A matching provider loads the user using `UserDetailsService`.
 6. Password is verified via `PasswordEncoder`.
 7. Successful `Authentication` is stored in `SecurityContextHolder`.
 8. Subsequent requests use the stored context until cleared.
-