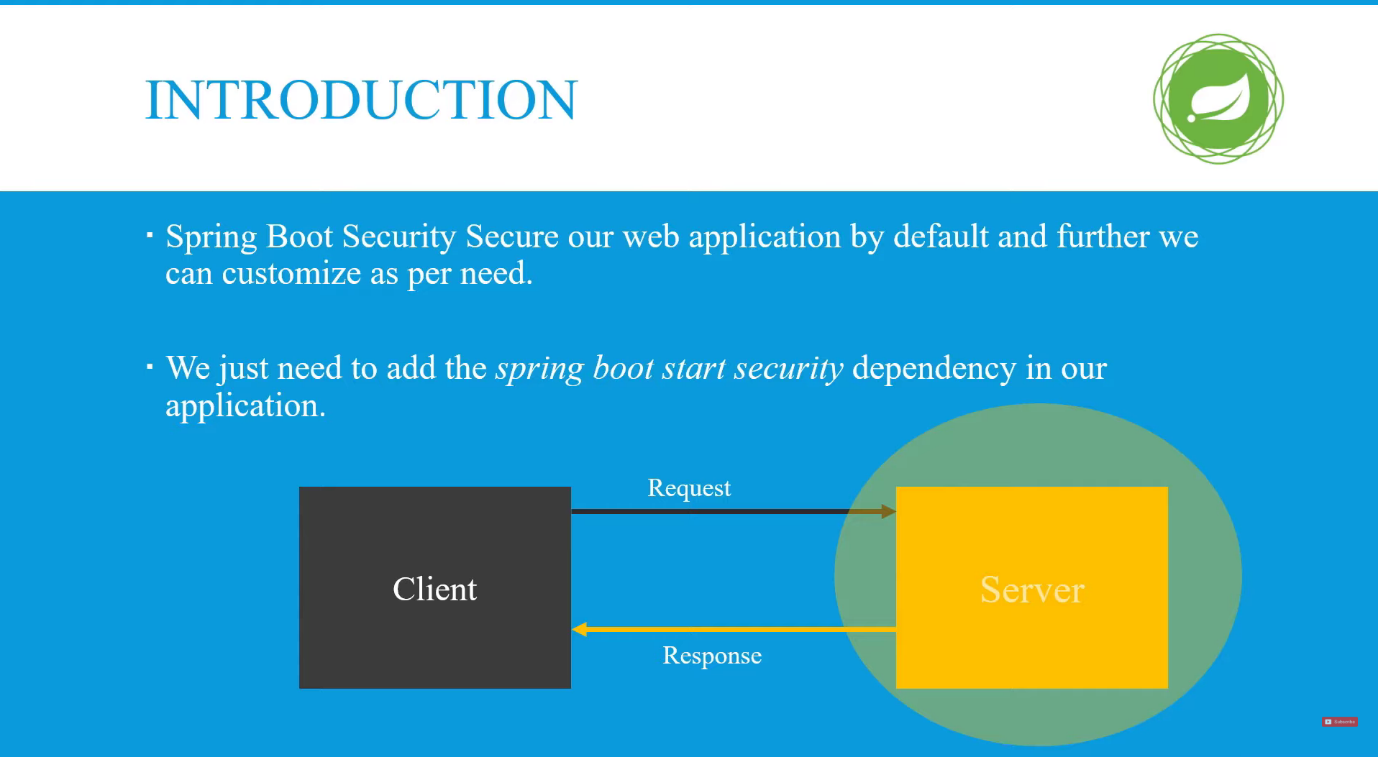
SPRING SECURITY



<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

This dependency is used for auto configure Spring Security in your app.

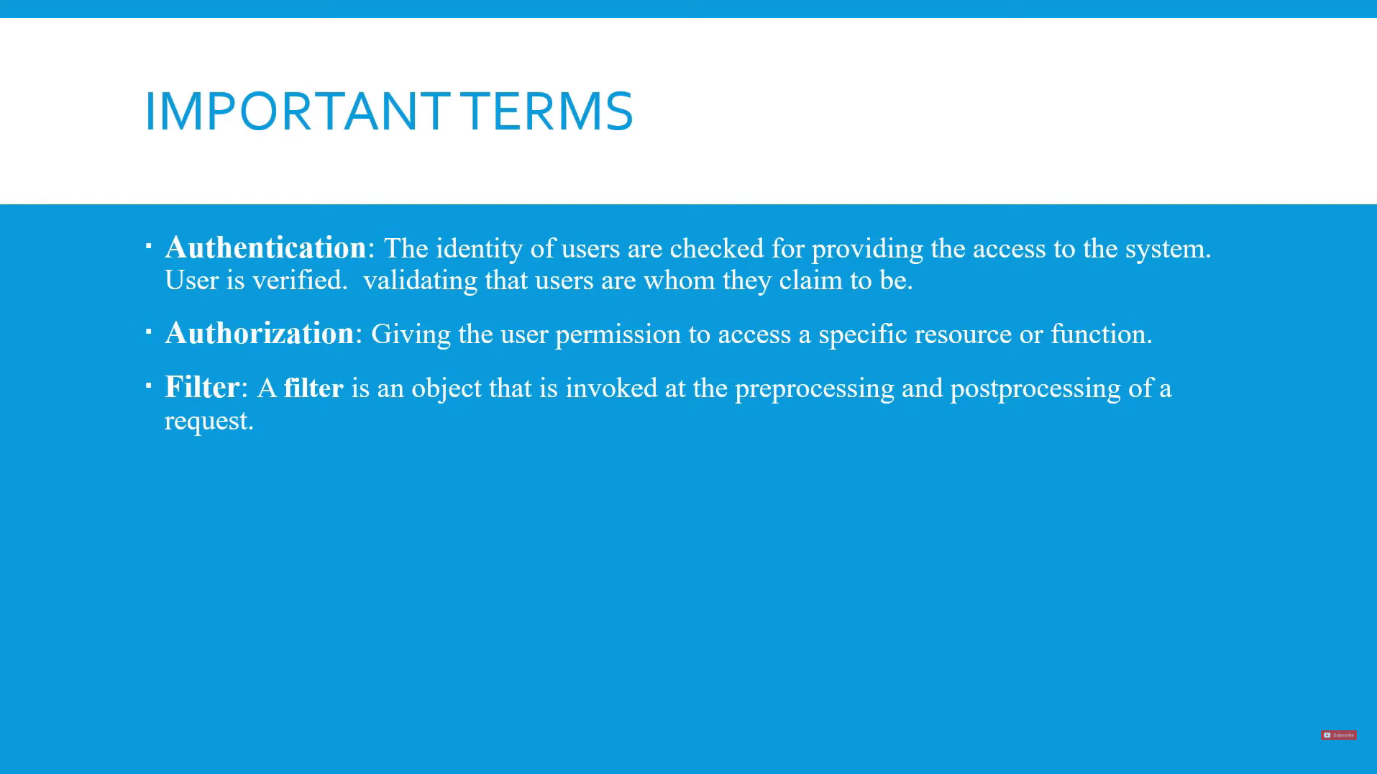
It applies form based Authentication in your app and then we can customize Spring Security as per need.

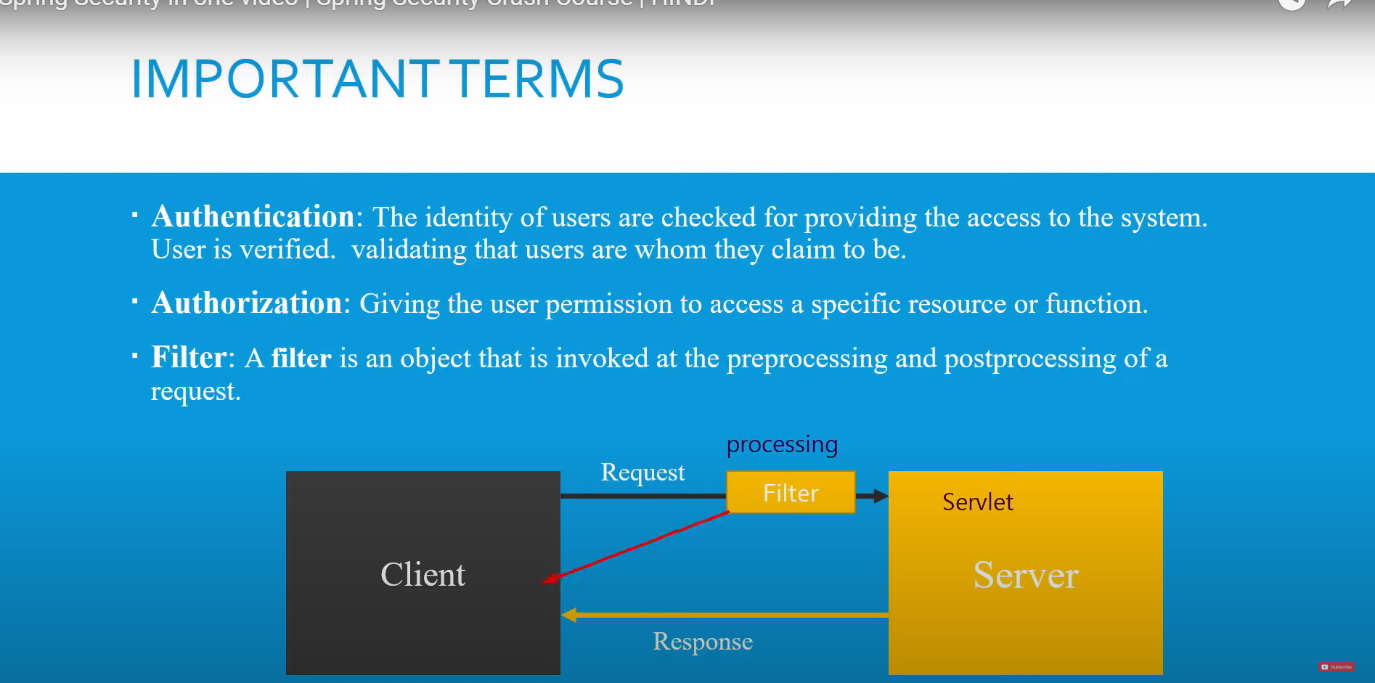
FormBased Authentication means it create a login form when you hit any APIs that form first check valid user then APIs should be run and you can also set

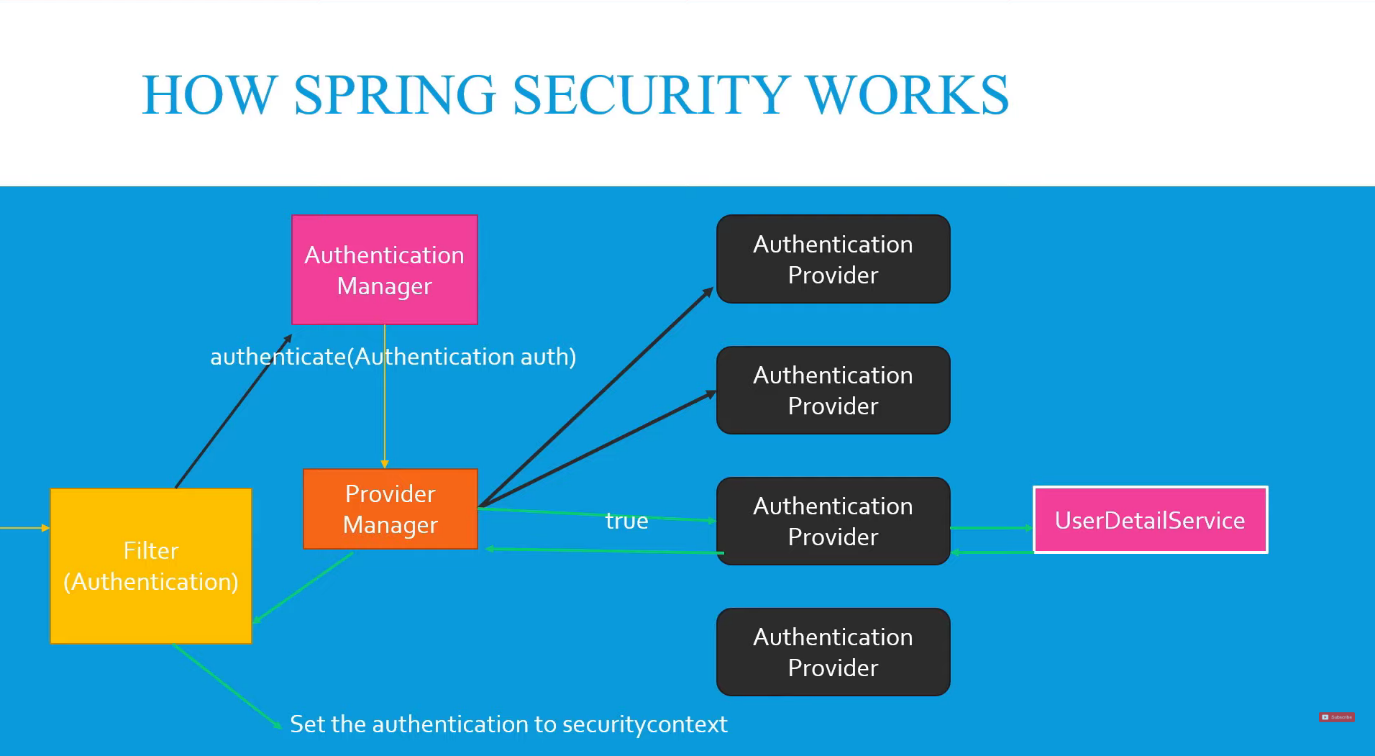
username and password as like in (application.propertise)

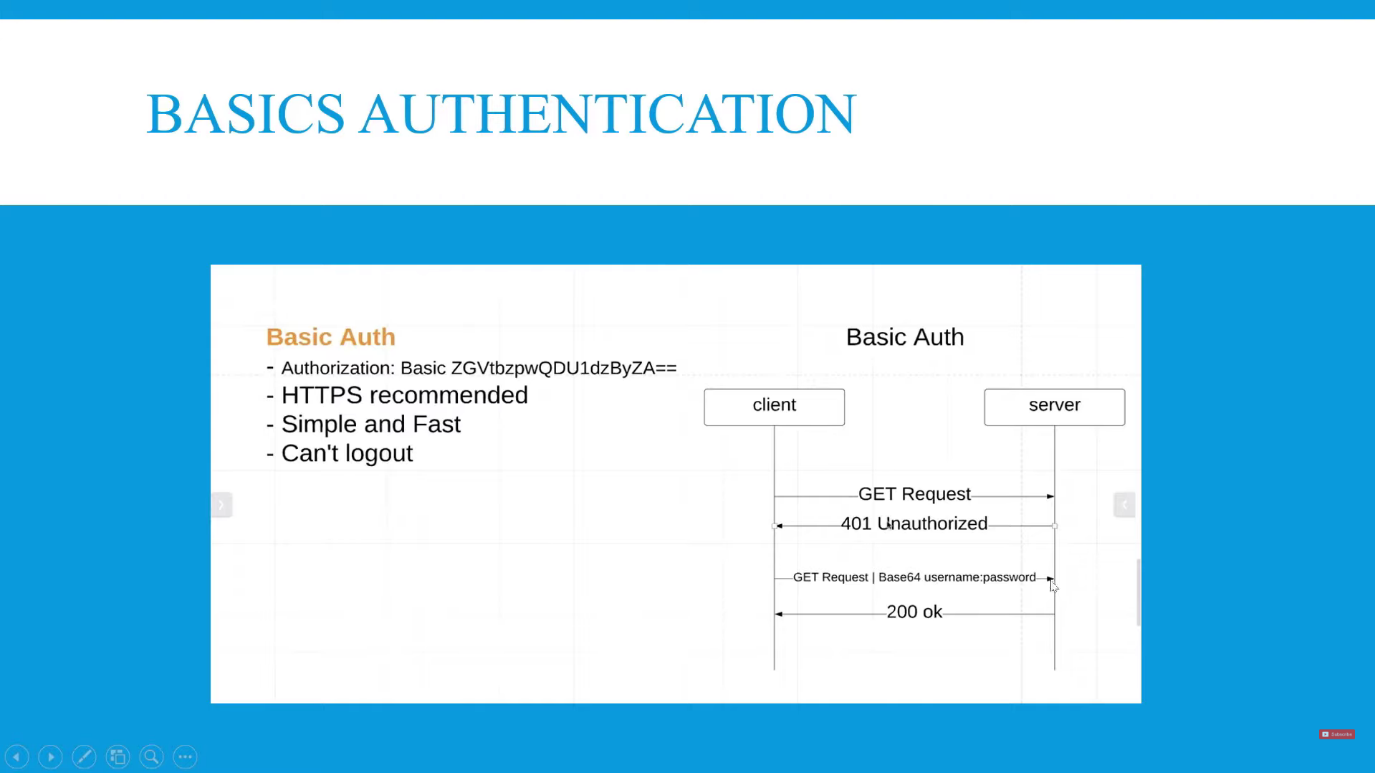
#spring.security.user.name=root

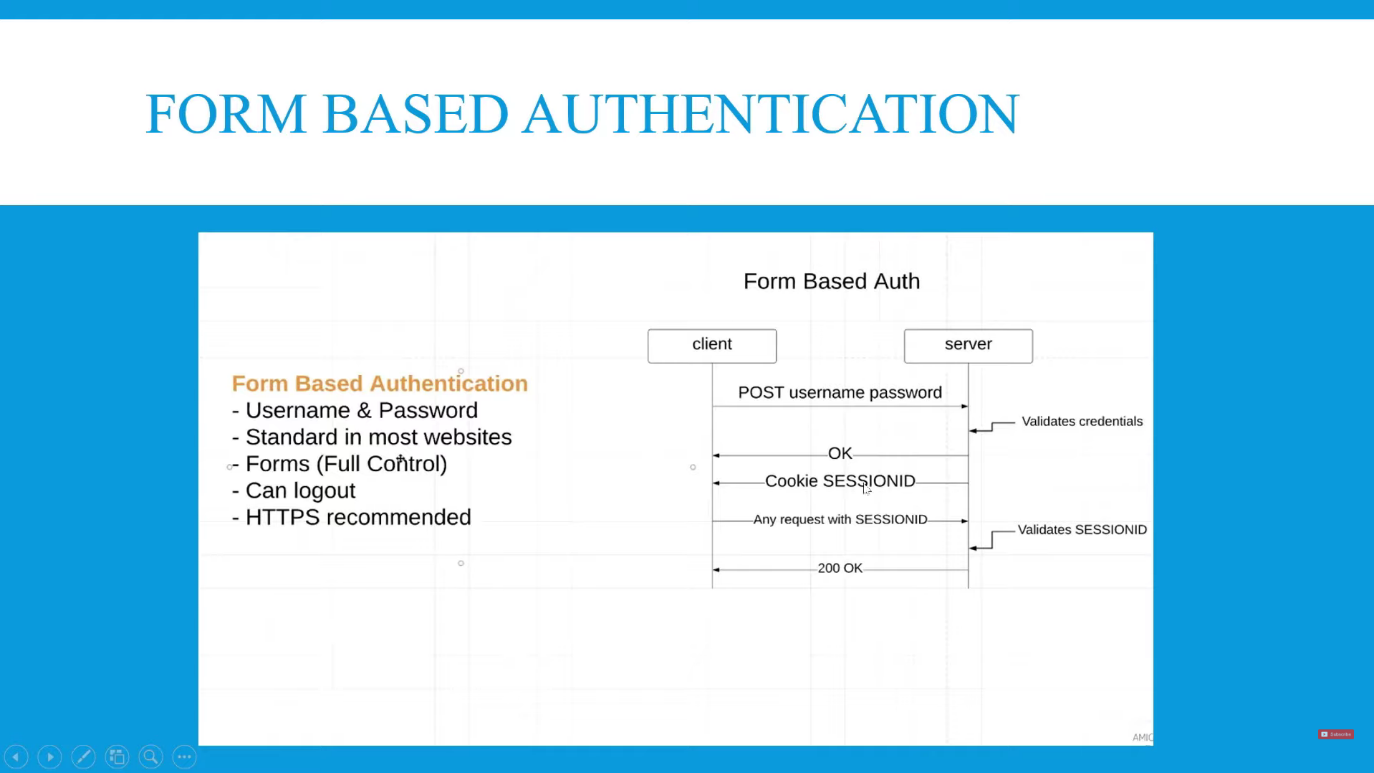
#spring.security.user.password=root











**differnce between antMatchers and requestMatchers**

In Spring Security 5.8, the antMatchers, mvcMatchers, and regexMatchers methods were deprecated in favor of new requestMatchers methods.

The new requestMatchers methods were added to authorizeHttpRequests, authorizeRequests, CSRF configuration, WebSecurityCustomizer and

any other places that had the specialized RequestMatcher methods. The deprecated methods are removed in Spring Security 6.

for details information go to

https://docs.spring.io/spring-security/reference/5.8/migration/servlet/config.html

**Role Based Authentication and Permission(Authorities) Based Authentication**

Role based means a user have role-> USER and their authorities(Permission) -> Read only

Role based means a user have role-> ADMIN and their authorities(Permission) -> Read, Write, Update etc.

for detail info

https://www.geeksforgeeks.org/spring-security-role-based-authentication/

https://www.geeksforgeeks.org/spring-security-role-based-and-permission-based-access-control/

**what is csrf in spring security and why we disable it**

Our recommendation is to use CSRF protection for any request that could be processed by a browser by normal users.

If you are only creating a service that is used by non-browser clients, you will likely want to disable CSRF protection.

It shouldn't impact the performance. A filter (or another component) will be removed from the request processing chain to make the feature unavailable.

What is the reason to disable csrf in a Spring Boot application?

You are using another token mechanism.

You want to simplify interactions between a client and the server.

for more details

https://www.geeksforgeeks.org/csrf-protection-in-spring-security/

https://www.baeldung.com/spring-security-csrf

**here are method to enable and used csrf token in spring security**

@Override

protected void configure(HttpSecurity http) throws Exception {

http

.csrf().csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse())

.and()

// Configure authorization rules

.authorizeRequests()

// Allow access to any URL under /public/ without authentication

.antMatchers("/public/\*\*").permitAll()

// Require authentication for any other request

.anyRequest().authenticated()

.and()

// Configure form-based login

.formLogin()

// Specify the custom login page URL

.loginPage("/login")

// Allow everyone to access the login page

.permitAll()

.and()

// Configure logout functionality

.logout()

// Allow everyone to access the logout functionality

.permitAll();

}

after using this method hit GET api and go to response and check for cookies

In cookies, you will get your csrf token for that api hit

and for using POST api you have to set in header X-XSRF-TOKEN: qiwu4ow-udbwb3-wdejniw2hdwoi

and with this you also have to set basic Auth.