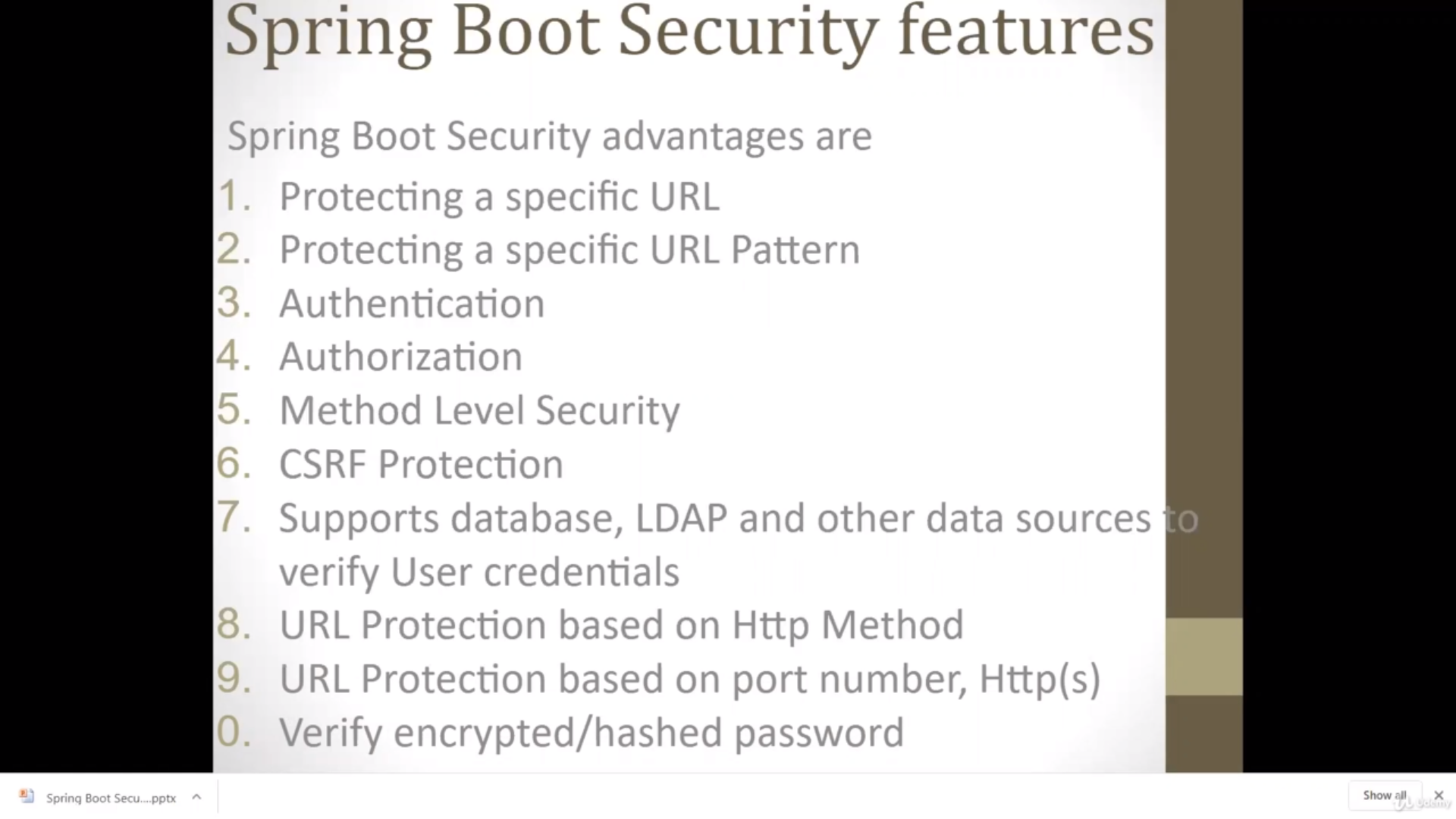
Spring Security

Spring Security is a framework that focuses on providing both authentication and authorization to Java applications. Like all Spring projects, the real power of Spring Security is found in how easily it can be extended to meet custom requirements

spring security features :



What is the difference between authentication and authorizing ?

**Authentication** and **authorization** might sound similar, but they are distinct security processes **in the** world of identity and access management (IAM). **Authentication** confirms that users are who they say they are. **Authorization** gives those users permission to access a

Note: if you want to write any security related code, you need to extended to WebSecurityConfigureAdapter Class and annotated @Configuration

Allows customization to the [WebSecurity](eclipse-javadoc:%E2%98%82=SpringSecurity/C:%5C/Users%5C/q1035274%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-config%5C/5.4.2%5C/spring-security-config-5.4.2.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-config=/=/maven.version=/5.4.2=/=/maven.scope=/compile=/%3Corg.springframework.security.config.annotation.web(WebSecurityConfigurer.class%E2%98%83WebSecurityConfigurer%E2%98%82WebSecurity). In most instances users will use [EnableWebSecurity](eclipse-javadoc:%E2%98%82=SpringSecurity/C:%5C/Users%5C/q1035274%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-config%5C/5.4.2%5C/spring-security-config-5.4.2.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-config=/=/maven.version=/5.4.2=/=/maven.scope=/compile=/%3Corg.springframework.security.config.annotation.web(WebSecurityConfigurer.class%E2%98%83WebSecurityConfigurer%E2%98%82EnableWebSecurity) and either create a [Configuration](eclipse-javadoc:%E2%98%82=SpringSecurity/C:%5C/Users%5C/q1035274%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-config%5C/5.4.2%5C/spring-security-config-5.4.2.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-config=/=/maven.version=/5.4.2=/=/maven.scope=/compile=/%3Corg.springframework.security.config.annotation.web(WebSecurityConfigurer.class%E2%98%83WebSecurityConfigurer%E2%98%82Configuration) that extends [WebSecurityConfigurerAdapter](eclipse-javadoc:%E2%98%82=SpringSecurity/C:%5C/Users%5C/q1035274%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-config%5C/5.4.2%5C/spring-security-config-5.4.2.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-config=/=/maven.version=/5.4.2=/=/maven.scope=/compile=/%3Corg.springframework.security.config.annotation.web(WebSecurityConfigurer.class%E2%98%83WebSecurityConfigurer%E2%98%82WebSecurityConfigurerAdapter) or expose a [SecurityFilterChain](eclipse-javadoc:%E2%98%82=SpringSecurity/C:%5C/Users%5C/q1035274%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-config%5C/5.4.2%5C/spring-security-config-5.4.2.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-config=/=/maven.version=/5.4.2=/=/maven.scope=/compile=/%3Corg.springframework.security.config.annotation.web(WebSecurityConfigurer.class%E2%98%83WebSecurityConfigurer%E2%98%82SecurityFilterChain) bean. Both will automatically be applied to the [WebSecurity](eclipse-javadoc:%E2%98%82=SpringSecurity/C:%5C/Users%5C/q1035274%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-config%5C/5.4.2%5C/spring-security-config-5.4.2.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-config=/=/maven.version=/5.4.2=/=/maven.scope=/compile=/%3Corg.springframework.security.config.annotation.web(WebSecurityConfigurer.class%E2%98%83WebSecurityConfigurer%E2%98%82WebSecurity) by the [EnableWebSecurity](eclipse-javadoc:%E2%98%82=SpringSecurity/C:%5C/Users%5C/q1035274%5C/.m2%5C/repository%5C/org%5C/springframework%5C/security%5C/spring-security-config%5C/5.4.2%5C/spring-security-config-5.4.2.jar=/maven.pomderived=/true=/=/maven.pomderived=/true=/=/maven.groupId=/org.springframework.security=/=/maven.artifactId=/spring-security-config=/=/maven.version=/5.4.2=/=/maven.scope=/compile=/%3Corg.springframework.security.config.annotation.web(WebSecurityConfigurer.class%E2%98%83WebSecurityConfigurer%E2%98%82EnableWebSecurity) annotation.

hasRole, hasAnyRole :

These expressions are responsible for defining the access control or authorization to specific URLs or methods in your application.

AntMatcher():-

antMatcher is the method(), it will tells to spring configure HttpSecurity, if path matches this pattern is authorizeRequests().antMatcher

is used to apply one or more paths to specify in antMatcher such as permitAll() or hasRole("USER3")

AntMatcher Rules

PathMatcher implementation for Ant-style path patterns.

Part of this mapping code has been kindly borrowed from Apache Ant.

The mapping matches URLs using the following rules:

? matches one character

\* matches zero or more characters

\*\* matches zero or more directories in a path

{spring:[a-z]+} matches the regexp [a-z]+ as a path variable named "spring"

Authentication type :

inMemoryAuthentication() : - Here is the source is the inMemoryAuthentication , that means here we are going to hardcoding the source code of authentication Username and Password and Roles

exe : -

// Authentication : set user/password details and mention the role

protected void configure (AuthenticationManagerBuilder auth) throws Exception { auth.inMemoryAuthentication()

.passwordEncoder(org.springframework.security.crypto.password.NoOpPasswordEncoder.getInstance())

.withUser("someone1").password("pass").roles("USER")

.and().withUser("someone2").password("pass").roles("USER", "ADMIN")

.and().withUser("slr").password("pass").roles("SELR");

}

// Authorization : mention which role can access which URL

protected void configure(HttpSecurity http) throws Exception {

http.httpBasic().and().authorizeRequests()

.antMatchers("/userlogin?", "/xyz").hasRole("USER")

.antMatchers("/adminlogin").hasRole("ADMIN")

.antMatchers("/slrlogin").hasAnyRole("USER", "SELR", "ADMIN")

.antMatchers("/\*\*").denyAll()

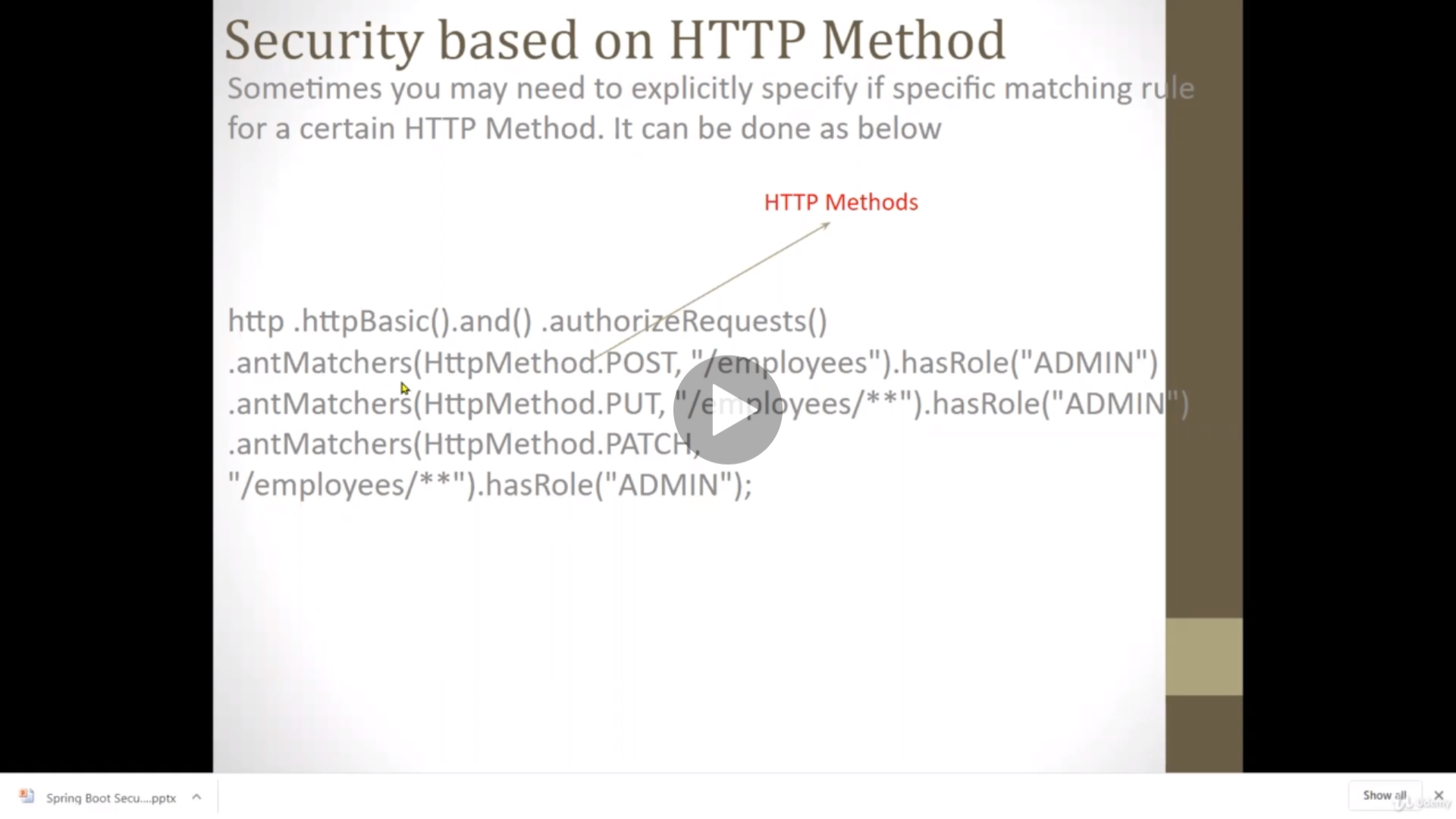
.and()

.logout().logoutUrl("/logout").deleteCookies("remove").invalidateHttpSession(false).and().csrf()

.disable().headers().frameOptions().disable();

}

**Note** : In the Realtime application no one hardcode to username and password, rather it will picked from some data source , like database or LDAP



@ConfigurationProperties(“spring.datasource”) : this is the annotation, which is the read the all the spring configurations, this configuration read data from the application.properties file to all DB configurations