Spring Boot Security

**SSL - Secure Socket Layer**

* This communicates over HTTPS protocol.

**TLS - Transport Layer Security**

* An advanced layer of SSL
* This communicates over HTTPS protocol

**TrustStore**

* Whenever we want to store any public certificate, we use Truststore to store them.

Eg: Google.com, facebook.com, load balancer certificates

**KeyStore**

* Whenever we want to store any private certificates, we use Keystore to store them.

Eg: A private WebServer certificate

**Generate JKS Self Signed Certificate**

* Keytool software is used to generate JKS certificate

Command to generate :

**Keytool -genkey -alias <https-example> -storetype JKS -keyalg RSA -keysize 2048 -validity 365 -keystore <https-example>.jks**

* **Enter Keystore Password and ReEnter**
* **First & LastName**
* **Name of the organisational Unit**
* **Name of the organisation**
* **Name of the city or locality**
* **Name of the state or province**
* **Two letter country code for this unit**
* **Verify all data**

* Command to view certificate using keytool

**keytool -printcert -v -file mydomain.crt.**

**Spring Boot security**

Dependency :

spring-boot-started-security

Working:

Once the above dependency gets added in the pom , automatically spring activates its security .

This is achieved through filters in spring .

Filter:

* A filter is a construct in server that intercept any request .
* It has the ability to manipulate the request
* Generally servlet and request is mapped as OneToOne mapping , whereas filters can be applied to a wide range based on regex declared.
* By adding spring starter security in app , by default all the request is mapped to spring security own filter known as **DelegatingFilterProxy.**
* **‘/\*’** this in below url-pattern listen to intercept everything.

Example:

In non spring boot app mapping has to be added in web.xml :

**<filter>**

**<filter-name>**springSecurityFilterChain**</filter-name>**

**<filter-class>**org.springframework.web.filter.DelegatingFilterProxy**</filter-class>**

**</filter>**

**<filter-mapping>**

**<filter-name>**springSecurityFilterChain**</filter-name>**

**<url-pattern>**/\***</url-pattern>**

**</filter-mapping>**

**DelegatingFilterProxy**

* Spring’s own security Filter
* As the name suggests, it delegates the request to a minimum of 5- 6 sub-filters in a default context .
* Authentication / Authorization some among them.

**Authentication**

Authentication is an interface in spring security that is solely responsible for authenticating requests.

Authentication Provider : is an interface in spring’s Authentication that authenticates a request by taking input as **Authentication(credentials)** and pass it to **authenticate()** methods and if the response is true ,i.e if the user is authenticated successfully, then it return **Authentication(Principle)** which contains info about logged in user.

Below table contains different constituents for Authentication interfaces.

Yellow : Before Authentication

Green : After Authentication

Basically :

 if (Authentication.getIsAuthenticated)

Contains Principle

else

   Contains Credentials

***Method Summary***

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| java.util.Collection<? extends [**GrantedAuthority**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/GrantedAuthority.html)> | [**getAuthorities**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/Authentication.html#getAuthorities--)()  Set by an AuthenticationManager to indicate the authorities that the principal has been granted. |
| java.lang.Object | [**getCredentials**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/Authentication.html#getCredentials--)()  The credentials that prove the principal is correct. |
| java.lang.Object | [**getDetails**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/Authentication.html#getDetails--)()  Stores additional details about the authentication request. |
| java.lang.Object | [**getPrincipal**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/Authentication.html#getPrincipal--)()  The identity of the principal being authenticated. |
| boolean | [**isAuthenticated**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/Authentication.html#isAuthenticated--)()  Used to indicate to AbstractSecurityInterceptor whether it should present the authentication token to the AuthenticationManager. |
| void | [**setAuthenticated**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/Authentication.html#setAuthenticated-boolean-)(boolean isAuthenticated)  See [**isAuthenticated()**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/Authentication.html#isAuthenticated--) for a full description. |

**Authentication Manager**

There can multiple ways to authenticate. Some ways are :

1. OAuth2
2. Basic Filter
3. JWT

Provider Manager is the class implements AuthenticationManager .

* AuthenticationManager does have an authenticate**()** method which takes authentication as input and provides authentication as output.
* Authentication manager makes use of a different **AuthentcationProviders** which has authenticate method .
* The kind of provider chosen by AuthenticationManager is selected based on the method in AuthenticationProvider ,i.e **supports()**.

***Method Summary***

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| [**Authentication**](https://docs.spring.io/spring-security/site/docs/4.2.20.RELEASE/apidocs/org/springframework/security/core/Authentication.html) | [**authenticate**](https://docs.spring.io/spring-security/site/docs/4.2.20.RELEASE/apidocs/org/springframework/security/authentication/AuthenticationProvider.html#authenticate-org.springframework.security.core.Authentication-)([**Authentication**](https://docs.spring.io/spring-security/site/docs/4.2.20.RELEASE/apidocs/org/springframework/security/core/Authentication.html) authentication)  Performs authentication with the same contract as [**AuthenticationManager.authenticate(Authentication)**](https://docs.spring.io/spring-security/site/docs/4.2.20.RELEASE/apidocs/org/springframework/security/authentication/AuthenticationManager.html#authenticate-org.springframework.security.core.Authentication-) . |
| boolean | [**supports**](https://docs.spring.io/spring-security/site/docs/4.2.20.RELEASE/apidocs/org/springframework/security/authentication/AuthenticationProvider.html#supports-java.lang.Class-)([**Class**](https://download.oracle.com/javase/6/docs/api/java/lang/Class.html?is-external=true)<?> authentication)  Returns true if this AuthenticationProvider supports the indicated Authentication object. |

**Authentication Provider**

There can be various AuthenticationProvider implementations based on the type of authentication required .

How do Authentication Providers provide Authenticate :

They connect to IdentitiyStore to extract user data and thereafter it does various checks and validation to identify the authenticity of the users.

However there can be different mechanisms to get the data, but once the data is there in the Authentication Provider context , it does the same checks and validations with the user data .

So Spring has extracted these checks and validations into an interface :

**UserDetailsService**

**It has a method called loadUserByUserName()**

***loadUserByUserName()***

This method returns an Interface : **UserDetails,** which contains all the relevant information regarding User for any subsequent Authorization.

***Method Summary***

|  |  |
| --- | --- |
| **Modifier and Type** | **Method and Description** |
| java.util.Collection<? extends [**GrantedAuthority**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/GrantedAuthority.html)> | [**getAuthorities**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetails.html#getAuthorities--)()  Returns the authorities granted to the user. |
| java.lang.String | [**getPassword**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetails.html#getPassword--)()  Returns the password used to authenticate the user. |
| java.lang.String | [**getUsername**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetails.html#getUsername--)()  Returns the username used to authenticate the user. |
| boolean | [**isAccountNonExpired**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetails.html#isAccountNonExpired--)()  Indicates whether the user's account has expired. |
| boolean | [**isAccountNonLocked**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetails.html#isAccountNonLocked--)()  Indicates whether the user is locked or unlocked. |
| boolean | [**isCredentialsNonExpired**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetails.html#isCredentialsNonExpired--)()  Indicates whether the user's credentials (password) has expired. |
| boolean | [**isEnabled**](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetails.html#isEnabled--)()  Indicates whether the user is enabled or disabled. |

**Sessions (After Authentication)**

* Once the successful Authentication is done, the Principal gets stored in ThreadLocal , a thread instance. And so in the securityContext.
* So Basically that ThreadLocal can be attached to a session and hence the user needn’t be authenticated until the session expires.
* Session is responsible to share the threadLocal Principal ,i.e. SecurityContext should be passed to subsequent filters.

**WebSecurityConfigurerAdapter**

* This class enables user to override Spring Security Defaults and implements custom security configuration
* This class should be annotated woth @EnableWebSecurity

This class does have different implementation of configure method - >

***Method Summary***

|  |  |
| --- | --- |
| Modifier and Type | Method and Description |
| protected [AuthenticationManager](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/authentication/AuthenticationManager.html) | [authenticationManager](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#authenticationManager--)()  Gets the [AuthenticationManager](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/authentication/AuthenticationManager.html) to use. |
| [AuthenticationManager](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/authentication/AuthenticationManager.html) | [authenticationManagerBean](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#authenticationManagerBean--)()  Override this method to expose the [AuthenticationManager](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/authentication/AuthenticationManager.html) from [configure(AuthenticationManagerBuilder)](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#configure-org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder-) to be exposed as a Bean. |
| protected void | [configure](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#configure-org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder-)([AuthenticationManagerBuilder](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/authentication/builders/AuthenticationManagerBuilder.html) auth)  Used by the default implementation of [authenticationManager()](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#authenticationManager--) to attempt to obtain an [AuthenticationManager](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/authentication/AuthenticationManager.html). |
| protected void | [configure](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#configure-org.springframework.security.config.annotation.web.builders.HttpSecurity-)([HttpSecurity](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/builders/HttpSecurity.html) http)  Override this method to configure the [HttpSecurity](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/builders/HttpSecurity.html). |
| void | [configure](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#configure-org.springframework.security.config.annotation.web.builders.WebSecurity-)([WebSecurity](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/builders/WebSecurity.html) web)  Override this method to configure [WebSecurity](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/builders/WebSecurity.html). |
| protected org.springframework.context.ApplicationContext | [getApplicationContext](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#getApplicationContext--)()  Gets the ApplicationContext |
| protected [HttpSecurity](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/builders/HttpSecurity.html) | [getHttp](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#getHttp--)()  Creates the [HttpSecurity](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/builders/HttpSecurity.html) or returns the current instance |
| void | [init](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#init-org.springframework.security.config.annotation.web.builders.WebSecurity-)([WebSecurity](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/builders/WebSecurity.html) web)  Initialize the [SecurityBuilder](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/SecurityBuilder.html). |
| void | [setApplicationContext](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#setApplicationContext-org.springframework.context.ApplicationContext-)(org.springframework.context.ApplicationContext context) |
| void | [setAuthenticationConfiguration](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#setAuthenticationConfiguration-org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration-)([AuthenticationConfiguration](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/authentication/configuration/AuthenticationConfiguration.html) authenticationConfiguration) |
| void | [setContentNegotationStrategy](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#setContentNegotationStrategy-org.springframework.web.accept.ContentNegotiationStrategy-)(org.springframework.web.accept.ContentNegotiationStrategy contentNegotiationStrategy) |
| void | [setObjectPostProcessor](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#setObjectPostProcessor-org.springframework.security.config.annotation.ObjectPostProcessor-)([ObjectPostProcessor](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/ObjectPostProcessor.html)<java.lang.Object> objectPostProcessor) |
| void | [setTrustResolver](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#setTrustResolver-org.springframework.security.authentication.AuthenticationTrustResolver-)([AuthenticationTrustResolver](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/authentication/AuthenticationTrustResolver.html) trustResolver) |
| protected [UserDetailsService](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetailsService.html) | [userDetailsService](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#userDetailsService--)()  Allows modifying and accessing the [UserDetailsService](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetailsService.html) from [userDetailsServiceBean()](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#userDetailsServiceBean--) without interacting with the ApplicationContext. |
| [UserDetailsService](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetailsService.html) | [userDetailsServiceBean](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#userDetailsServiceBean--)()  Override this method to expose a [UserDetailsService](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/core/userdetails/UserDetailsService.html) created from [configure(AuthenticationManagerBuilder)](https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/config/annotation/web/configuration/WebSecurityConfigurerAdapter.html#configure-org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder-) as a bean. |

How does configure Works:

**configure()** can be used to override default spring configuration with customized one.

Examples of types of configure method for authorization and authentication.

Eg 1:

**@EnableWebSecurity**

Public class SecurityConfig extends WebSecurityConfigurerAdapter{

//authentication

@Override

**Protected void configure(AuthenticationManagerBuilderAuth auth){**

auth.inMemoryAuthentication().withUser(“user”).password(“password”).roles(“USER”);

auth.inMemoryAuthentication().withUser(“admin”).password(“password”).roles(“ADMIN”);

**}**

//authorization

@Override

**Protected void configure(HttpsSecurity http) throws Exception{**

http.csrf().disable();

http.antMatcher(“/\*\*”).authorizeRequests().anyRequest().fullyAuthenticated().and().httpBasic();

**}**

//for password hash

@Bean

**Public static NoOpPasswordEncoder passwordEncoder{**

return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();

**}**

}

Eg:2- Inlined classpath database to be used as data source:

If the dataSource is changed , Automatically point will be changed.

Default Schema is used , that’s created by Spring

**@EnableWebSecurity**

Public class SecurityConfig extends WebSecurityConfigurerAdapter{

//authentication

@Autowired

**DataSource** dataSource;

@Override

**Protected void configure(AuthenticationManagerBuilderAuth auth){**

auth.jdbcAuthentication()

.dataSource(dataSource)

.withDefaultSchema()

.withUser(

User.withUsername(“user”)

.password(“pass”)

.roles(“USER”)

).withUser(

            User.withUsername(“admin”)

.password(“pass”)

.roles(“ADMIN”)

);

**}**

//authorization

@Override

**Protected void configure(HttpsSecurity http) throws Exception{**

http.authorizeRequests()

 .antMatchers(“/admin”).hasRole(“ADMIN”)

 .antMatchers(“/user”).hasAnyRole(“ADMIN”,”USER”)

 .antMatchers(“/”).permitAll()

 .and().formLogin();

**}**

//for password hash

@Bean

**Public static NoOpPasswordEncoder passwordEncoder{**

return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();

**}**

}

*Default Schema In Spring*

Default schema gets created by Spring boot , and the same is pointed by DataSource.

create table users(

      username varchar\_ignorecase(50) not null primary key,

      password varchar\_ignorecase(50) not null,

      enabled boolean not null);

  create table authorities (

      username varchar\_ignorecase(50) not null,

      authority varchar\_ignorecase(50) not null,

      constraint fk\_authorities\_users foreign key(username) references users(username));

      create unique index ix\_auth\_username on authorities (username,authority);

Eg:3- Inlined classpath database to be used as data source:

If the dataSource is changed , Automatically point will be changed.

Custom architecture - as default schema used

**@EnableWebSecurity**

Public class SecurityConfig extends WebSecurityConfigurerAdapter{

//authentication

@Autowired

**DataSource** dataSource;

@Override

**Protected void configure(AuthenticationManagerBuilderAuth auth){**

auth.jdbcAuthentication()

.dataSource(dataSource);

**}**

//authorization

@Override

**Protected void configure(HttpsSecurity http) throws Exception{**

http.authorizeRequests()

 .antMatchers(“/admin”).hasRole(“ADMIN”)

 .antMatchers(“/user”).hasAnyRole(“ADMIN”,”USER”)

 .antMatchers(“/”).permitAll()

 .and().formLogin();

**}**

//for password hash

@Bean

**Public static NoOpPasswordEncoder passwordEncoder{**

return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();

**}**

}

Eg:4- Inlined classpath database to be used as data source:

If the dataSource is changed , Automatically point will be changed.

Custom Schema is used , reference is provided to Spring through two sql expect methods:

* **usersByUsernameQuery(“select username,password,enabled from my\_user where user\_name = ? “)**

* **authoritiesByUsernameQuery(“select username,authority from custom\_authorities where user\_name = ? ”)**

**@EnableWebSecurity**

Public class SecurityConfig extends WebSecurityConfigurerAdapter{

//authentication

@Autowired

**DataSource** dataSource;

@Override

**Protected void configure(AuthenticationManagerBuilderAuth auth){**

auth.jdbcAuthentication()

.dataSource(dataSource)

.usersByUsernameQuery(“select username,password,enabled from my\_user where user\_name = ? “)

.authoritiesByUsernameQuery(“select username,authority from custom\_authorities where user\_name = ? ”);

**}**

//authorization

@Override

**Protected void configure(HttpsSecurity http) throws Exception{**

http.authorizeRequests()

 .antMatchers(“/admin”).hasRole(“ADMIN”)

 .antMatchers(“/user”).hasAnyRole(“ADMIN”,”USER”)

 .antMatchers(“/”).permitAll()

 .and().formLogin();

**}**

//for password hash

@Bean

**Public static NoOpPasswordEncoder passwordEncoder{**

return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();**}**}

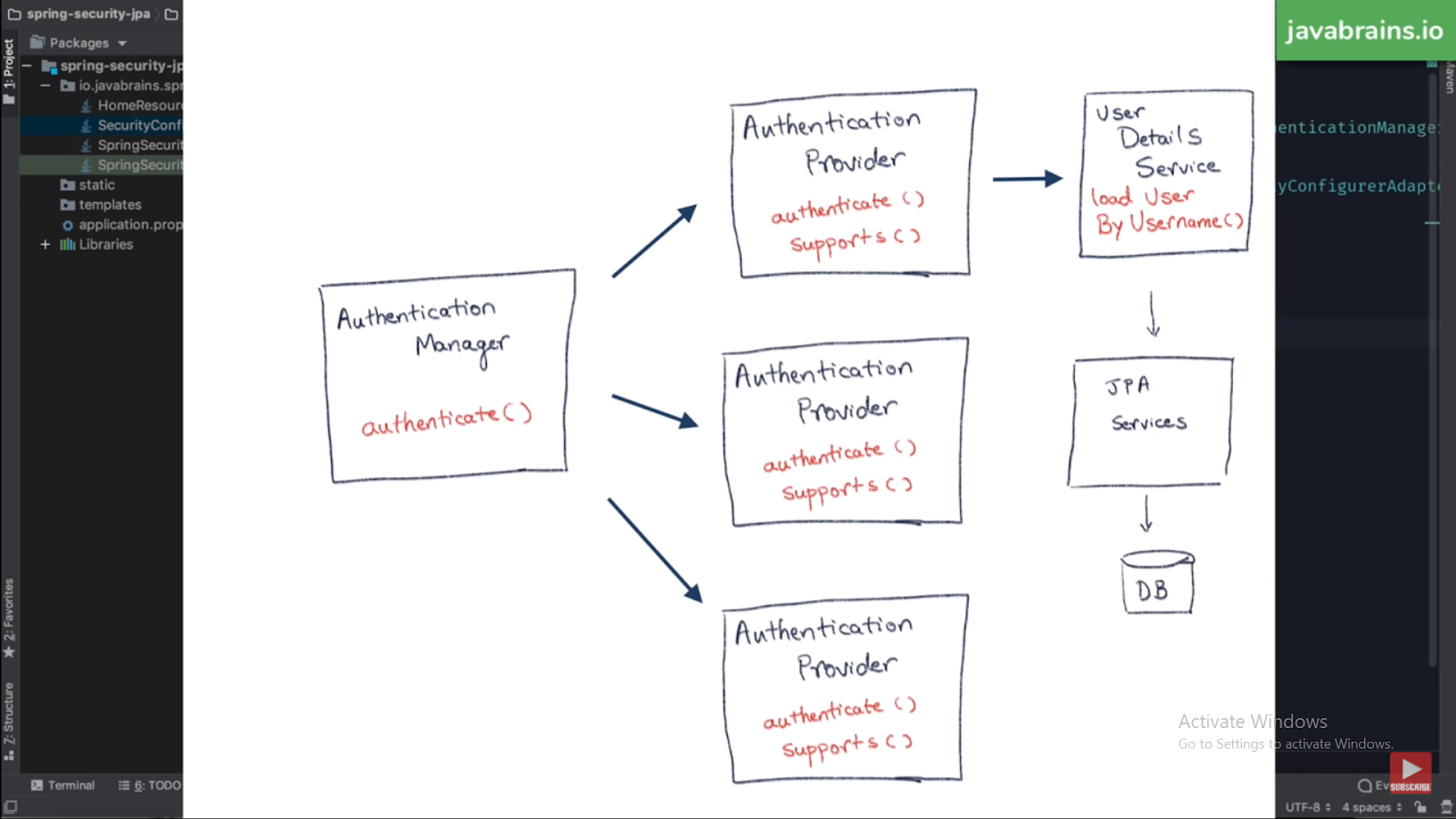
**JPA Authentication with MySQL**

Key Points to be considered.

* In this case, Spring handles manipulation till AuthenticationProvider, however custom implementation of **UserDetailsService -> loadUserByUserName()** has to bedone.
* MyUserDetailsService implements UserDetailsService

* Whenever spring requires DataSource for Authentication, **configure(AuthenticationManagerBuilderAuth auth)** is used, and Custom Bean for UserDetailsService is consumed to get connected through custom DataSource.

* Similarly the interface “UserDetails” is also Implemented to a custom class
* MyUserDetails implements UserDetails



Code Implementation

**@EnableWebSecurity**

Public class SecurityConfig extends WebSecurityConfigurerAdapter{

//authentication

@Autowired

UserDetailsService userDetailsService;

@Override

**protected void configure(AuthenticationManagerBuilderAuth auth){**

   auth.userDetailsService(userDetailsService);

**}**

//authorization

@Override

**Protected void configure(HttpsSecurity http) throws Exception{**

http.authorizeRequests()

 .antMatchers(“/admin”).hasRole(“ADMIN”)

 .antMatchers(“/user”).hasAnyRole(“ADMIN”,”USER”)

 .antMatchers(“/”).permitAll()

 .and().formLogin();

**}**

//for password hash

@Bean

**Public static NoOpPasswordEncoder passwordEncoder{**

return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();

**}**

}

-------------------------------------------------------------------------------------------

@Service

Public **MyUserDetailsService** implements **UserDetailsService**{

@Override

Public UserDetails **loadUserByUserName**(String s){

Return new **MyUserDetails**(s);

}

}

-------------------------------------------------------------------------------------------

Public **MyUserDetail** implements **UserDetails**{

// implements all the methods and return appropriate data ;

}

**X509 certificate Authentication using SSL**

app

**@EnableWebSecurity**

**Public class SecurityConfig extends WebSecurityConfigurerAdapter{**

@Autowired

private final AccountUserDetailsService accountUserDetailsService ;

@Override

protected UserDetailsService userDetailsService() {

        return this.accountUserDetailsService ;

}

@Override

protected void configure(HttpSecurity http) throws Exception {

  http.authorizeRequests().anyRequest().authenticated()

                .and()

                    .x509()

                .and()

                    .sessionManagement().sessionCreationPolicy(SessionCreationPolicy.NEVER)

                .and()

                    .csrf().disable();

**}**

//for password hash

@Bean

**Public static NoOpPasswordEncoder passwordEncoder{**

return (NoOpPasswordEncoder) NoOpPasswordEncoder.getInstance();

**}**

}

@Service

**class AccountUserDetailsService implements UserDetailsService** **{**

   private final AccountRepository accountRepository;

   @Autowired

    public AccountUserDetailsService(AccountRepository accountRepository) {

        this.accountRepository = accountRepository;

    }

  @Override

**public UserDetails loadUserByUsername(String username) throws       UsernameNotFoundException** {

        return this.accountRepository.findByUsername(username)

                .map(account -> new User(account.getUsername(),

                        account.getPassword(),

                        account.isActive(), account.isActive(), account.isActive(), account.isActive(),

                        AuthorityUtils.createAuthorityList("ROLE\_ADMIN", "ROLE\_USER")

                ))

                .orElseThrow(() -> new UsernameNotFoundException("couldn't find " + username + "!"));

    }

**}**

**//Rest CALL**

**@RestController**

**class GreetingsRestController {**

    @RequestMapping(method = RequestMethod.GET, value = "/hi")

    public Map<String, String> greetings(Principal p) {

        return Collections.singletonMap("content", "Hello, " + p.getName());

    }

**}**

**//Entity & repo**

**interface AccountRepository extends JpaRepository<Account, Long> {**

Optional<Account> findByUsername(String username);

**}**

**@Entity**

**class Account {**

    @Id

    @GeneratedValue

    private Long id;

    private String username, password;

    private boolean active;

    public Account() {// why JPA why?

    }

    public Account(String username, String password, boolean active) {

        this.username = username;

        this.password = password;

        this.active = active;

    }

    public Long getId() {

        return id;

    }

    public String getUsername() {

        return username;

    }

    public String getPassword() {

        return password;

    }

    public boolean isActive() {

        return active;

    }

**}**