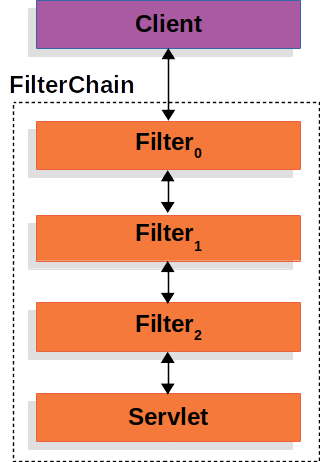
**Spring Security**

Spring Security is a framework which provides various security features like: authentication, authorization to create secure Java Enterprise Applications.

This framework targets two major areas of application are authentication and authorization. Authentication is the process of knowing and identifying the user that wants to access.

**Authorization** is the process to allow authority to perform actions in the application.

Architecture : -



The client sends a request to the application, and the container creates a FilterChain, which contains the Filter instances and Servlet that should process the HttpServletRequest, based on the path of the request URI. In a Spring MVC application, the Servlet is an instance of [DispatcherServlet](https://docs.spring.io/spring-framework/docs/6.0.5/reference/html/web.html" \l "mvc-servlet). At most, one Servlet can handle a single HttpServletRequest and HttpServletResponse.

Servlet container (tomcat) will convert the Http message that they receive from the brower into an Http servlet request object and the same object will be given to the java core and vice versa.

Filters are special kind of servlets which we can use to intercept each and every request that is coming towards our web application. So inside these filters we can define any pre logic or pre-work that we want to do before our actual business logic executes.

@Configuration when we define this annotation, it is an indication to the spring boot framework that we have certain configurations defined inside this class that is especially @Bean.

SecurityFilterChain is very much important whenever we want to define our own custom requirements.

Whenever we want to define our own custom requirements, we need to invoke a method called requestMatchers.

RequestMatchers method is going to accept any number of Api patterns.

Whenenver we invoke authenticator method it’s an indication to the spring security framework that I want to protect these API paths.

.and is a method which is going to combine different configurations of spring security.

In order to deny all the request from the users we can use the concept called profile in the spring security. With the help of the profiles we can create the beans conditionally based upon the current environment where our application is deployed.

.denyAll() – will denies all the requests made by the user.

permitAll requests – In order to accomplish this we will create the two beans of security filter chain. One bean is going to activate for the production environment whereas other bean is going to activate for all the non-production environments.

NoOpPasswordEncoder will treat our passwords as a plain text.

PasswordEncoder is only to deal with the passwords whether you want to deal them with plain text or hashing or encrypted values.

By default there is an Authentication Provider provided by the spring security with the name DaoAuthenticationProvider and this provider is going to take help from the list of UserDetailsManager.

The methods which was in UserDetails can have only getters but not setters.

InMemory is a most common scenario which we want to use for no-production applications or for some demo applications.

MySql disadvantages :-

1.The very first one is it require lot of memory and our system processing time leads to slow down our system.

2.If corruption is happened then all the data that we have inside our database will gone forever.

create table users(id int not null auto\_increment, username varchar(45) not null, password varchar(45) not null, enabled int not null, primary key(id));

create table authorities(id int not null auto\_increment, username varchar(45) not null, authority varchar(45) not null, primary key(id));

spring-boot-starter-jdbc - this dependency we are adding since we are leveraging JDBC framework to fetch the records from the database.

Mysql-connector-j – this dependency is related to Mysql.

Spring-boot-starter-data-jpa – we are going to use spring boot jpa framework.

PasswordEncoder bean is useful to communicate to spring security how our passwords are stored, whether they are stored in plain text password or whether they are stored with the help of encryption or hashing.

@Entity – we can try to fetch the records from this table with the help of spring data JPA framework.

@Entity is an annotation from the spring data JPA framework, which represents that this project class that we are creating exactly represents that database table.

@GeneratedValue(strategy = GenerationType.***AUTO***) – this means we are telling to the spring that id is autogenerating by itself.

CrudRepository is the interface available inside the Spring Data JPA framework which helps us to automatically generate the code for all CRUD operations.

@Repository represents the spring framework that this class is a repository class which deals completely with the logics related to the database interaction.

Based upon the method name our spring data JPA is going to generate the business logic at the runtime.

When we are using the findBy, we are telling to the spring data JPA please fetch the record with the help of select query.

findByEmail – is called as derived method name query.

If we are using the spring boot the we need not to define the @EnableWebSecurity, else we have to define then only all the features of spring security will be enabled.

List is a interface in the java library.

ArrayList is a class from the collections.

User is an implementation of the UserDetails interface.

SimpleGrantedAuthority is the class that implementing the GrantedAuthority.

@Service – inside this class we have written some business logic so it is acting as a service layer.

Now spring security framework whenever it is trying to perform authentication it will handover the request to DAO authentication provider and DAO authentication provider can look what are the implementation classes of user details service that my developer want me to use.

Constructor of the User class return back to the DAO authentication provider.

Inside the DAO authentication provider it is going to compare the password that we are sending from the database and the password that it receives from the end user.

@GenericGenerator(name = "native", strategy = "native") – using this our spring jpa will not worry of creating the id values instead it depends on the backend database server.

**Password Encoders:-**

As of now we had used the Default PasswordEncoder which handles the passwords in a plaintext format.

When we enter the login details and hit enter spring security framework will execute all the logic present inside the authentication provider and it will try to load the details from the storage system with the help of the method loadUserByUsername which is available inside the implementation classes of User details manager.

Once user details are loaded our spring security framework will try to compare the passwords that are provided by end user and the password that is loaded from the database.

Inside the DAOAuthenticationProvider class there is a authenticate method first we will try to load the username details from the storage system or from the database.

PreAuthenticationChecks checks whether my account is expired, whether my account is locked or whether my account is disabled.

Spring security filters will convert the username and password into the authentication object.

PasswordEncoder is an interface which has two abstract methods and one default method.

First abstract method that we have is encode, this method we can use during the registration process of an end user.

Second abstract method is matches, this method we can use at the during the login operation to compare the user-entered password.

UpgradeEncoding is the default method and it has some default logic of returning false all the times.this will give more security to our passwords.

**Different implementations of PasswordEncoder inside Spring Security :-**

If some hackers have strong GPU machine then Pbkdf2PasswordEncoder is not recommended.

A brute force attack is a hacking method that uses trial and error to crack passwords, login credentials, and encryption keys.

BcryptPasswordEncoder we can safely use for our production web applications because it demands lot of computation power from the hacker missions.

ScryptPasswordEncoder is an advanced version of BcryptPasswordEncoder because it demands two parameters whenever someone is trying to utilize hashing functionality or the matches functionality of this algorithm.

The first parameter is computation power.

The second parameter is memory inside RAM.

Argon2PasswordEncoder this is even latest hashing algorithm which has three dimensions.

The first parameter is computation power.

The second parameter is memory inside RAM.

The third parameter is number of threads or multiple cores od CPU.

The first three characters in the hash value of a password indicates the Bcrypt version - $2a.

The next three characters indicates what is the workload factor or the number of rounds that we have considered for our hashing alogorithm - $10. By default spring considers the 10 as the rounds.

**Inside the AuthenticationProvider Interface :-**

We have two abstract methods.

the very first one is authenticate method. This is the critical method where we need to write all our authentication logic. For this method the input object is authentication which will have the username and the credentials of the end user.

The second abstract method we have is the supports, so using this supports method we need to tell to the spring security framework what kind of authentications that I want to support with the help of authentication provider.

**CORS and CSRF**

**@Query – it is a JPQL query. This is not a direct hard cored sql. JPQL is a SQL format which will allow us to write the queries with the help of our entity class.**

Different types of security:-

1.firewalls

2.HTTPS

3.SSL

4.Authentication

5.Authorization.

The communication between an UI application and the backend application is not happening – CORS policy.

Origin is an URL, which is the combination of URL HTTP protocol and what is a domain name or host name and what is a port number where the web application is deployed. The combination of three parameters like HTTP protocol, domain and the port is called as origin.

Cross Origin Resource Sharing (CORS) two different origins they are trying to share their resources. In our case two different origins are client application and the backend application.

CORS is not a security thread it is a protection layer from the security vulnerabilities that we have inside the web.

CORS is the default security layer available in the inside the browsers, which will stop the communication between two different origins.

CSRF is an security attack it is not a protection layer inside our browser. Rather it is an security vulnerability which hackers is going to use this technique to steal the data from our web application.

By default spring security provides the CSRF protection. It will not allow any POST operations or PUT operations that are going to update or create new data inside our database.

CROSS-SITE REQUEST FORGERY (CSRF) is an security attack or security vulnerabilities which used by hackers. So this Is a technique whenever a hacker is performing this attack, he will not steal our credentials or our session ID or cookies. Instead he will trick us to carryout some action without our will.

CsrfTokenRequestAttributeHandler this class is implementing the interface called CsrfTokenRequestHandler. This class is capable of making the CsrfToken available as a request attribute and resolving the token value.

withHttpOnlyFalse() – With the help of this we are telling to the spring security framework, please create a CSRF cookie with a configuration as HttpOnlyFalse so that my JavaScript code deployed.