108.Introduction to Spring Security

SPRING SECURITY

- *Spring Security is a powerful and highly customizable authentication and access-control framework. It is the de-facto standard for securing Spring-based applications.*
- *Below is the maven dependency that we can add to implement security using Spring Security project in any of the SpringBoot projects,*

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Spring Security is a framework that provides authentication, authorization, and protection against common attacks.

Spring Security helps developers with easier configurations to secure a web application by using standard username/password authentication mechanism.

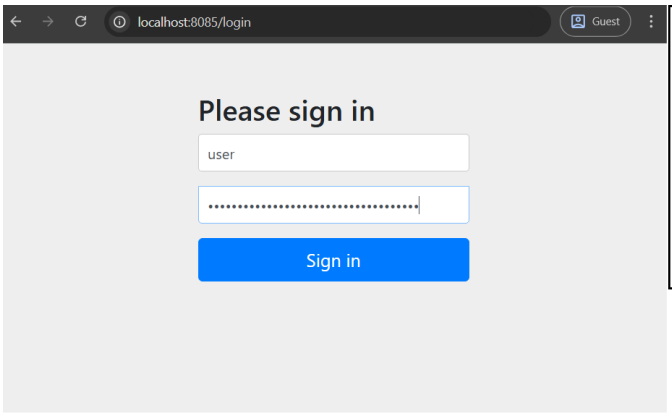
Spring Security provides out of the box features to handle common security attacks like CSRF, CORs. It also has good integration with security standards like JWT, OAUTH2 etc.

JWT JSON Web Token

109: Deep dive of Authentication vs Authorization

AUTHENTICATION	AUTHORIZATION
In authentication, the identity of users are checked for providing the access to the system.	In authorization, person's or user's authorities are checked for accessing the resources.
Authentication done before authorization	Authorization always happens after authentication.
It needs usually user's login details	it needs user's privilege or roles
If authentication fails usually we will get 401 error response	If authorization fails usually we will get 403 error response
For example as a Bank customer/employee in order to perform actions in the app, we need to prove our identity	Once logged into the application, my roles, authorities will decide what kind of actions I can do

110: Demo of Spring Security inside Eazy School WebApp with Default Behavior



As soon we add the Spring Security dependency to the project our project will be provided with login page and the user name for that will be user and password we can view in the logs .

Using generated security password: 311297ec-7c2c-4427-97d8-8f0990931a7d

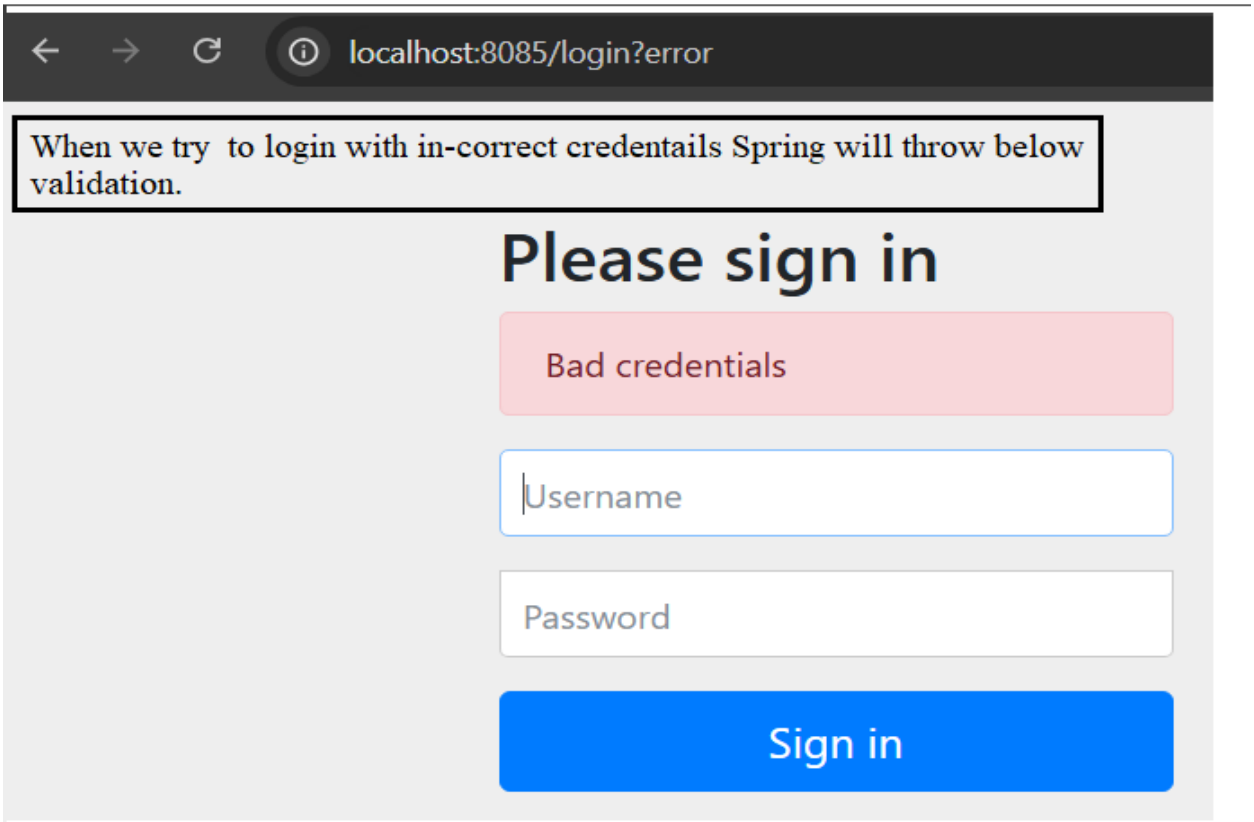
This generated password is for development use only. Your security configuration must be updated before running your application in production.

we can't access any pages in the project/application until we login with details.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

Once we login it will be valid until the browser will be closed.Once we close the browser we need to provide the details again for login.

111: Configure custom credentials inside Spring Security



When we try to login with in-correct credentails Spring will throw below validation.

```
spring.security.user.name=chinmay
spring.security.user.password=987654321
```

As we have provided the login details spring will not generate any security password

Do you know,

- ✓ As soon as we add spring security dependency to a web application, be default it protects all the pages/APIs inside it. It will redirect to the inbuilt login page to enter credentials.
- ✓ The default credentials are user and password is randomly generated & printed on the console.
- ✓ We can configure custom credentials using the below properties to get started for POCs etc. But for PROD applications, Spring Security supports user credentials configuration inside DB, LDAP, OAUTH2 Server etc.

```
spring.security.user.name = eazybytes
spring.security.user.password = 12345
```

113: Understanding default security configurations inside Spring Security Framework.

```java
@Configuration(proxyBeanMethods = false)
@ConditionalOnDefaultWebSecurity
static class SecurityFilterChainConfiguration {

    @Bean
    @Order(SecurityProperties.BASIC_AUTH_ORDER)
    SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Exception {
        http.authorizeHttpRequests(( AuthorizationManagerRequestMat... requests) -> requests.anyRequest().authenticated());
        http.formLogin(withDefaults());
        http.httpBasic(withDefaults());
        return http.build();
    }

}
```

As of now, we have added the Spring Security related maven dependency inside our web application.
As soon as we add the dependency, we observe that the Spring Security framework is securing all my web pages.

But in real world, we may want to secure only few pages, whereas other pages we want to let them available for everyone so that even without entering any credentials, my end users should be able to access them.

Spring Security Framework Also let developers to define their own custom security configurations.

So to define our own Spring Security custom configurations.

First, we need to understand what is the piece of code inside the Spring Security framework which is responsible to secure my web application, like we discussed before, by default, Spring Security is going to protect all the paths present inside your web application.
This behavior is due to the code present inside the method default security filter chain and this method is present inside the class SpringBootWebSecurityConfiguration.
You can see there is a method called default security filter chain.
This method is accepting an input parameter of type HttpSecurity and it is returning an object of type SecurityFilterChain and since we have @Bean on top of this method, my spring framework is going to return a bean of type security filter chain whenever this method is executed.
we can also see there is @Configuration on top of this class, which means during the startup of my web application, this method is going to be executed and bean of SecurityFilterChain will be initialized by the Spring IOC Container.
The line http.authorizeHttpRequest().anyRequest().authenticated() is responsible to protect all the paths available inside our web application.
Whenever we are using .authenticated(), that means we want to secure the API paths.
Next we have code like

http.formLogin(withDefaults());
http.httpBasic(withDefaults());

With these two lines, my Spring Security framework is going to enforce the security requirements which is defined in the above line for both the formLogin() and httpBasic() request.
For any web application, the request can come in two different forms. The very first approach with the help of formLogin(), where you have an application, where you enter
some data and send request to the backend server.
The other approach is with the help of rest APIs or Http protocol. So all the requests that we are sending without the help of UI application are without the help of HTML
forms will come under the Http basic.
At last we are also invoking a build method available inside this Http Security. This build method is going to return an object of security filter chain based upon the configurations that we have mentioned here.
We can customize our security requirements so that few pages will be secured and a few other pages will be publicly accessible.
Whenever you have such custom requirements, you need to define a method inside your web application, which is going to return a bean of security filter chain.

114:Configure PermitAll() inside the WebApp using Spring Security

```java
@Configuration  no usages  new *
public class ProjectSecurityConfig {

    @Bean  no usages  new *
    SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Exception {
        http.authorizeHttpRequests(( AuthorizationManagerRequestMat... requests) -> requests.anyRequest().permitAll());
        http.formLogin(withDefaults());
        http.httpBasic(withDefaults());
        return http.build();
    }
}
```

As we have made it as permitAll() even though we have we have provided the authentication details in the application.properties all the pages will be dispalyed without authentication.

- *Using permitAll() configurations we can allow full/public access to a specific resource/path or all the resources/paths inside a web application.*
- *Below is the sample configuration that we can do in order to allow any requests in a Web application with out security,*

```java
@Configuration
public class ProjectSecurityConfig {

  @Bean
  SecurityFilterChain defaultSecurityFilterChain(HttpSecurity http) throws Exception {

    http.authorizeHttpRequests((requests) -> requests.anyRequest().permitAll())
          .formLogin(Customizer.withDefaults())
          .httpBasic(Customizer.withDefaults());
    return http.build();

  }
```

- Form Login provides support for username and password being provided through an html form
- HTTP Basic Auth uses an HTTP header in order to provide the username and password when making a request to a server.

115