

Spring Security

जय श्री गणेश



→ What is Spring security ?

- Spring security is a powerful and highly customizable security framework that is often used in SpringBoot applications to handle "authentication" and "Authorization"

① Authentication → The process of verifying a user's identity.

Ex → username & Password.

② Authorization → The process of granting ^{or} and denying access to specific resources or actions based on the authenticated user's role and permission.

"Once the dependency is added, Spring Boot auto-configuration feature will automatically apply security to the application"

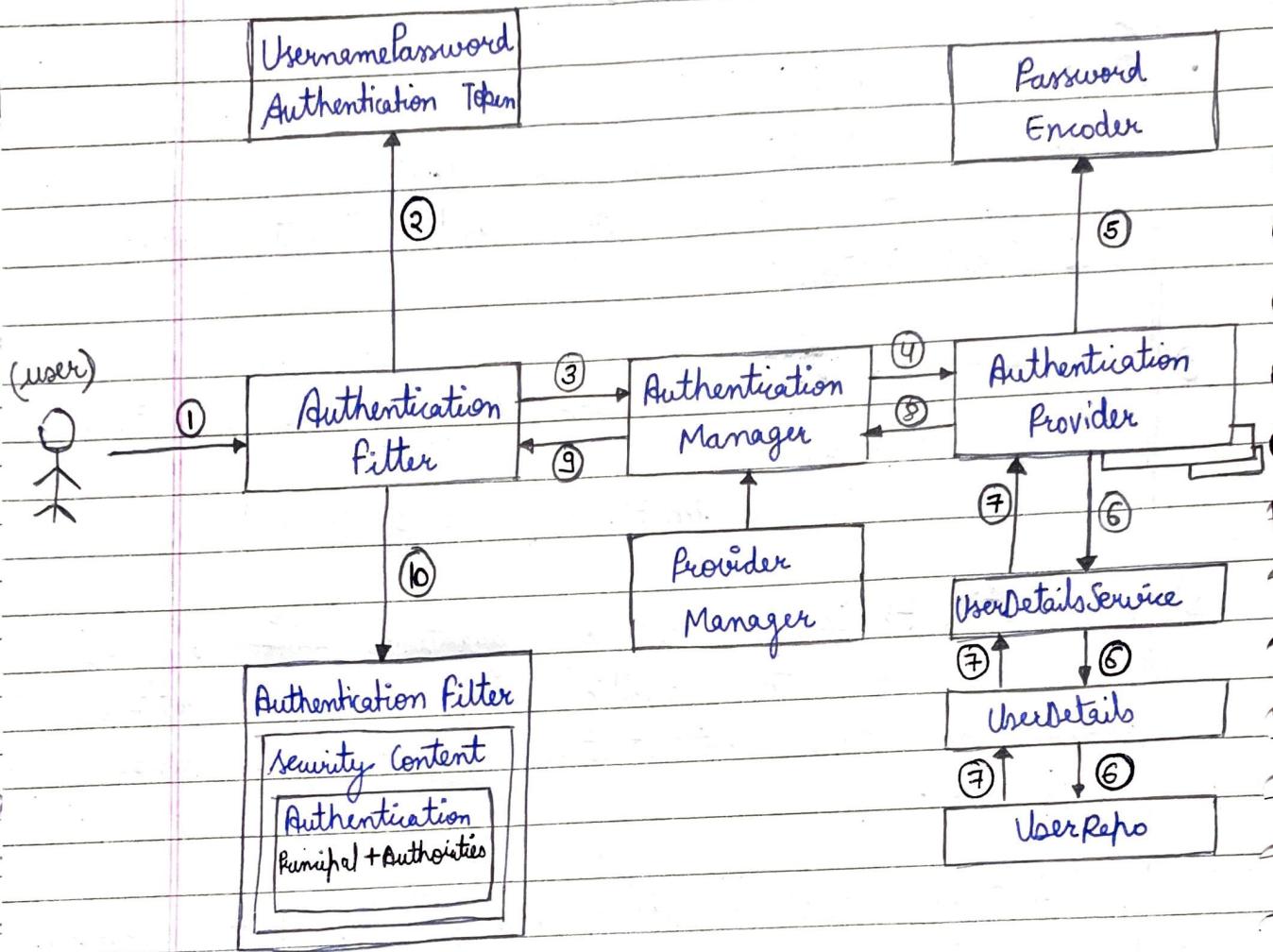
dependency → Spring-boot-starter-security

• By default, spring security uses HTTP Basic Authentication.

• By default, all end points will be secured. Spring security will generate a default user with a random password that is printed in the console.

"जिस तरह हम धर के खिलकी बैताजे lock कर देते हैं की कोई अंदर नहीं आ जाये। उसी तरह spring security पुरी Application को lock कर देता है"

Internal working of Spring security



Working of above diagram :-

- शब्द से पहले एक user के पिस्तो username + Password डाल के request भेजा। तो शब्द से पहले वो request आये गी **AuthenticationFilter** के पास। तो अब **AuthenticationFilter** के पास username and Password आ गया। अब UserNamePasswordAuthenticationToken का object बनेगा और **AuthenticationFilter** उस object में username and Password store करा देगा। यह object अब **AuthenticationManager** को दें देगा। तो Basically **AuthenticationManager** ने **ProviderManager** का interface के तो एक class के **UserDetailsService**



पी की इस interface को extend कर रही है।
तो ProviderManager ने check करने वाली है
की username & Password को validate करने के लिए
कौन-सा AuthenticationProvider suitable है ज्याकि उचित सारे
AuthenticationProvider होते हैं।

“

तो Basically ProviderManager के पास एक support() method
रहती है जिसकी help से वो Decide करता है की
कौन-सा AuthenticationProvider suitable है ”
→ like DaoAuthenticationProvider etc..

AuthenticationProvider का main role यह है कि वो check
करता है की जो Incoming username & Password आ
रहा है वो हमारे Database में exist कर रहा है या
नहीं। Inshort ये Validation का काम कर रहा है।

“ Suppose ProviderManager ने DaoAuthenticationProvider choose किया
ये क्या करेगा की ये UserDetailsService Interface को बोलेगा
की इस check करे की ये username & Password का user
Database में exist कर रहा है या नहीं। ”

तो UserDetailsService Interface के पास एक method
है loadUserByUsername जिसकी help से ये check करेगा की user
exist कर रहा है या नहीं।
यदि exist कर रहा है तो UserDetails का object
return कर देगा।

अब AuthenticationProvider ने object को ProviderManager
को देंगा वो अब ProviderManager ने object को AuthenticationFilter
को देंगा।

“ तो अब AuthenticationFilter securityContent की help से
role set कर देगा। ”

1st Basic Program of spring

security"



→ Create a new Project → Day1 Spring Security

- Dependencies →
- Spring web
 - Spring security
 - spring dev-tools

→ Create a one controller class → Package → com.security.Controller
class → MyController

@RestController

```
public class MyController {
    @GetMapping("/")
    public String demo() {
        return "ram ji";
    }
}
```

Run the project → One Password will be generated on console.

Open chrome → localhost:8080/

→ Press Enter

Username	user
Password	*****
login	

O/P → ram ji



→ जैसे ही इस login हो जाते हैं तो एक SessionId generate हो जाती है जो की हमारे system के Browser में store हो जाती है तो user New window open करेगा तो उसे again sign in नहीं करता पड़ेगा व्याकि उसका session अब्दी valid है।

यदि user chrome पूरा close कर देता हो तो session Id destroy हो जायेगी फिर वो chrome से again request करेगा तो अब उसे sign ^{again} करना पड़ेगा।

Ques: अब user जो चाहता है कि वो बिना chrome close करे logout होना पाएता है तो उसे कैसे होगा?

Ans: तो Basically user chrome पर /logout लिखेगा तो वो logout हो जायेगा और उसकी sessionid destroy हो जायेगी।

→ अब हमें sessionid ढैखता है :> ① Inspect पर जाके ढैख लो।

② HttpServletRequest

GetMapping("/")

public String demo(HttpServletRequest req)

return "ram" + req.getSession().getId();

3

→ Now we want custom username & Password :-

go to Application.properties :-

spring.security.user.name = ram

spring.security.user.password = ramji123

अब हमें custom username & Password डाल दिया तो console पर नहीं आयेगा Password"

→ Now we use Postman ⇒

Step → 1 Open Postman

Step → 2 Select GET → `http://localhost:8080/`

Step → 3 Click on Auth
Select → BasicAuth

username = root

password = root

[Send] → Click on send Button

O/P → ram-ji 192.168.1.23

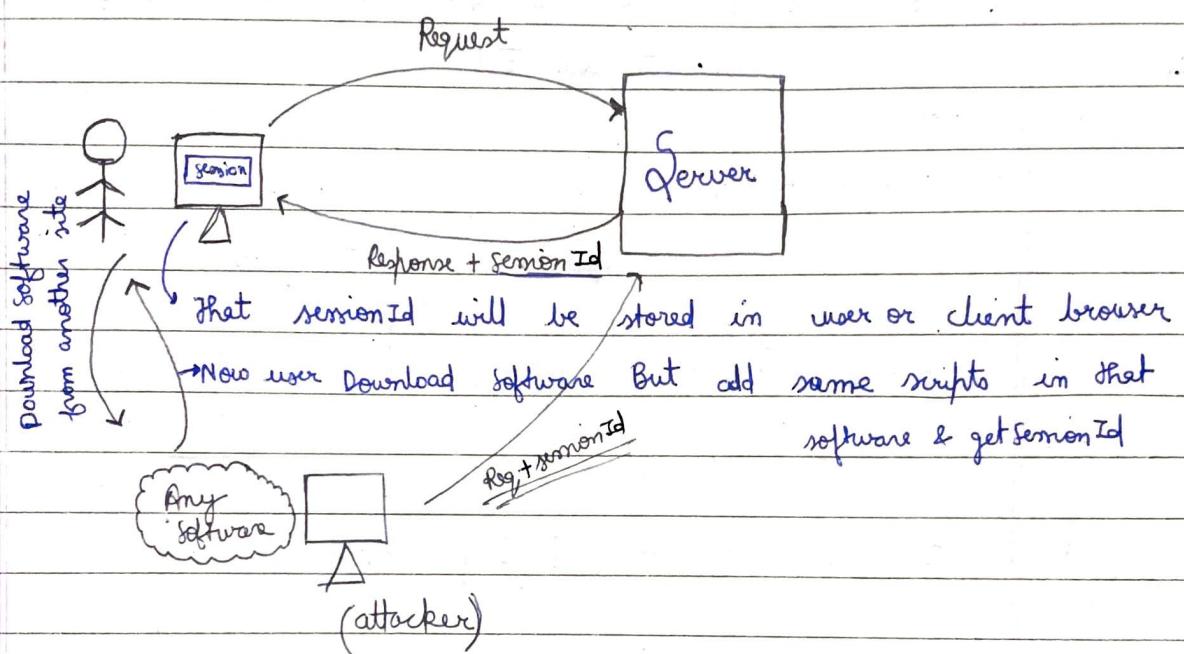
sessionid

“CSRF”



- CSRF stands for Cross-site Request forgery.
- It is a type of cyber attack that tricks a user's web browser into performing unwanted actions on a trusted site.

→



Basically यह हुआ कि client ने request करी username और password डालके तो वह user valid user निकला तो उसको response में एक sessionid मिल गयी कि आप यह website पर again लगाते हो तो आपको login करते की ज़रूरत नहीं है क्योंकि आपके पास sessionid है इसने आपको पहले ही authenticate कर लिया। तो यह sessionid user के browser में store होती है। अब attacker ने एक gmail भेजा की यह STS का new version है आप download कर ले तो user ने download कर लिया तो attacker ने उस software में कुछ script डाली थी अब यह script client के system में Background में चलने लगी। अब client की sessionid अब attacker के पास आयी अब attacker यह भेज के साथ user का Data access कर लेंगा।

→ तो इस attack से बचने के लिए spring security ने एक concept दिया → CSRF token

→ Spring security में Bydefault CSRF token enable होता है।

→ Ex: अब भी हम form से request भेजते हैं तो request के साथ CSRF token भी जाता है और sever पर यह check होता है कि क्या ये Token valid है यदि है तो successfully login हो जाता है।

Right click on Default spring security login Page ⇒
click on inspect source

We can see this,

<input type="csrf" type="hidden" value="_____"/>
↳ csrf token no.

→ Create a new project ⇒ Day1SpringSecurity - 2

Dependencies →

- spring web
- thymeleaf
- spring security

→ Create a class ⇒ HomeController
↳ com.security.controller

@Controller

public class MyController

{

@GetMapping("/")

public String home()

{

return "home";

}



```
@PostMapping("/savedata")
public String home (String uname)
{
    s.o.hn(uname);
    return "home";
}
```

→ Create home.html Page :

```
<body>
<form action = "/savestate" method = "post">
    Enter Email <input type = "text" name = "uname" /> <br>
    <input type = "submit" value = "save Record" />
</form>
</body>
```

"Run the Project"

→ Open chrome : localhost:8080/

↓

username	<input type="text" value="user"/>
password	<input type="password" value="*****"/>

print on console

If it is correct

Enter Email	<input type="text" value="Pujish@"/>
<input type="button" value="Save Record"/>	

जैसे ही email डर्क के

उसने इस Button पर click किया तो Data Post Method की help से
server पर गया और उसने उभारे form में CSRF token भी किया
जाएगा तो white label error page आएगा।

- तो 2 तरीके से हम इस error को solve कर सकते हैं :-
- ① हमारे form में CSRF token generate करें।
 - ② या csrf को disable कर दें।
- तो हम अब csrf को disable कर देंगे।
- Create a new class : SecuritySpring

@Configuration

@EnableWebSecurity

public class SecuritySpring

{

@Bean

public SecurityFilterChain func (HttpSecurity http)

{

http.csrf (csrf → csrf.disable());

return http.build();

}

}

- अब हम Data भेजेंगे तो PostMapping चलेगी हमारा DataServer पर पायेगा और हम उसको console पर देखेंगे वो print हो पायेगा।

“ तो अब attacker इस चीज का फ़ायदा उठाएगा वो हमारे URL को copy करके हमारे URL से Data भेजेगा तो वो शी sewer पर चले पायेगा और उसी console पर show हो पायेगा ”

“ तो अब finally CSRF token का concept माया ”



→ तो अब हम HTML Page में CSRF Token generate करेंगे।

same code

```
<input type="hidden" th:name="${_csrf.parameterName}"  
       th:value="${_csrf.token}"/>  
      ↓  
      predefined interface
```

आइए हम अब csrf को git enable कर दें।

→ अब इस अब csrf की depth में आगे पढ़ें।

- How do generate custom CSRT Token
- Use token with filters etc....