Spring Security & JWT

FISA 기술세미나

클라우드 엔지니어링 1조

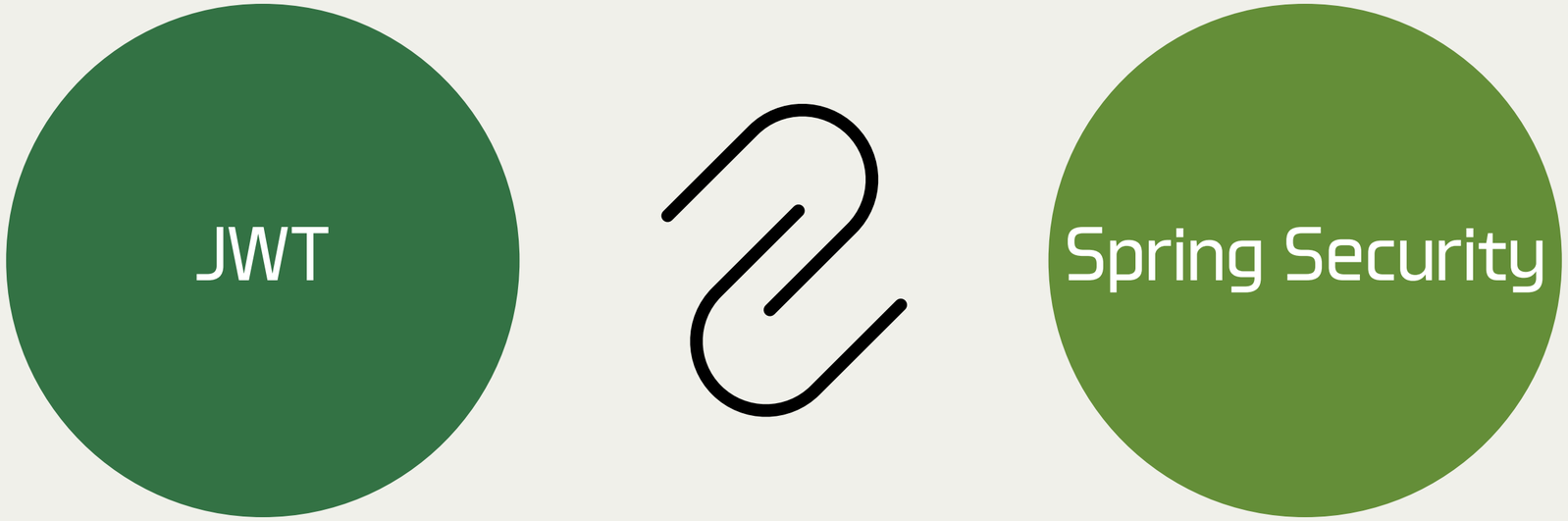
목 차

목표 및 팀 소개

01. 중간 목표

쿠키/세션	해당 인증 방식의 장·단점
JWT	JWT의 구조와 토큰의 특징
Spring Security	인증/인가의 구조와 SecurityChainFilter

01. 최종 목표



01. 프로젝트 팀 구성 소개

유호준

김지훈

박재희

이성빈

쿠키

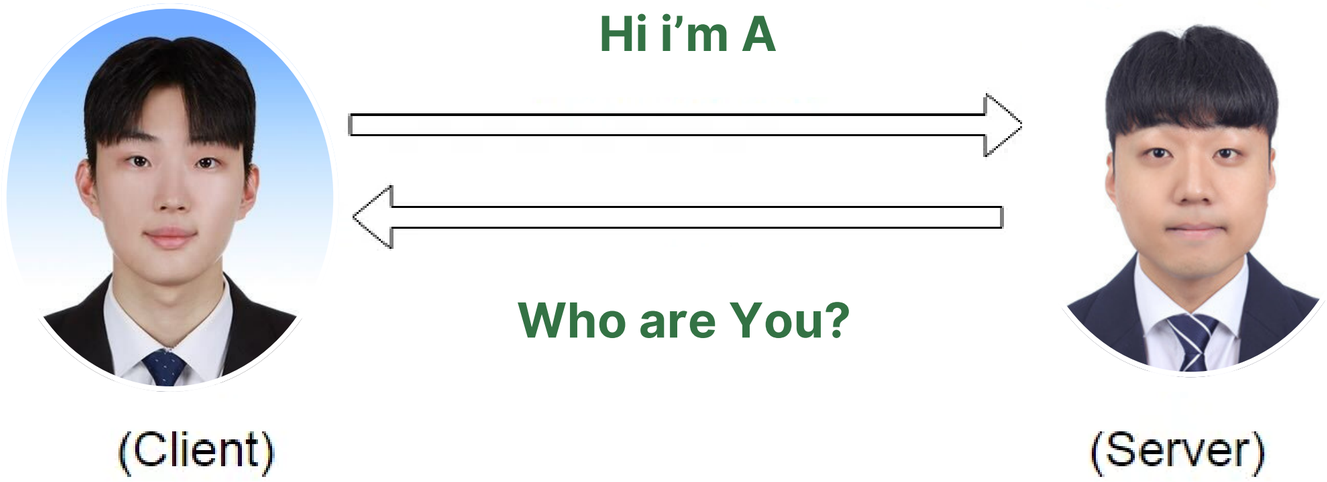
02. 쿠키

정의

웹 서버가 생성하여 웹 브라우저로 전송하는
작은 정보 파일

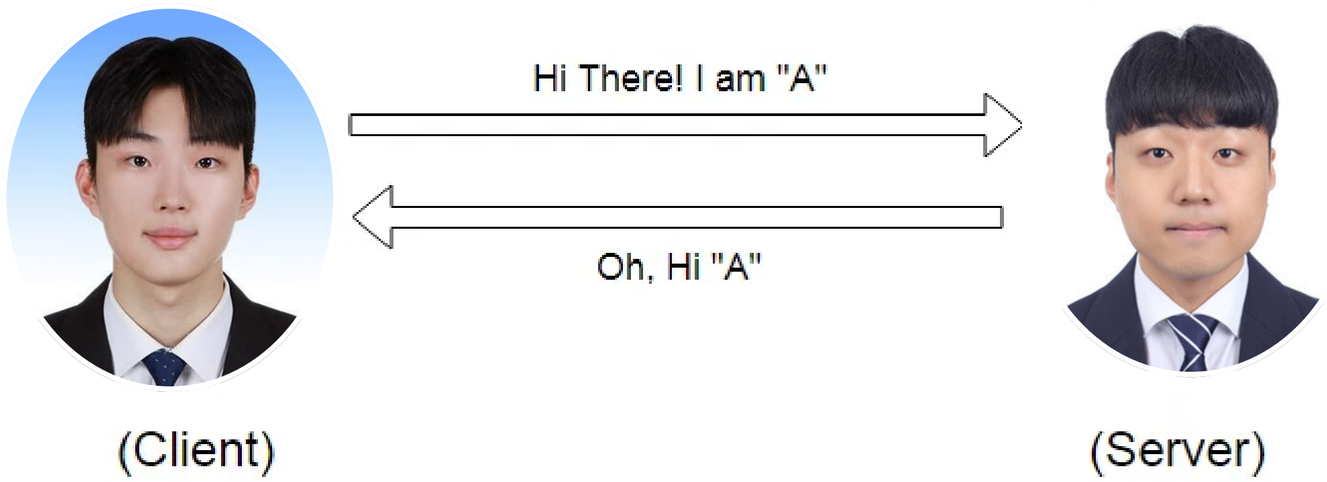
02. 쿠키 사용 이유

HTTP STATELESS



02. 쿠키 사용 이유

HTTP STATEFUL



02. 쿠키

장점

빠른 인증

클라이언트 중심 서버 관리

간단한 구현

단점

탈취 위험성 - CSRF 공격 취약

데이터 크키 제한
(브라우저 허용 4KB 이하)

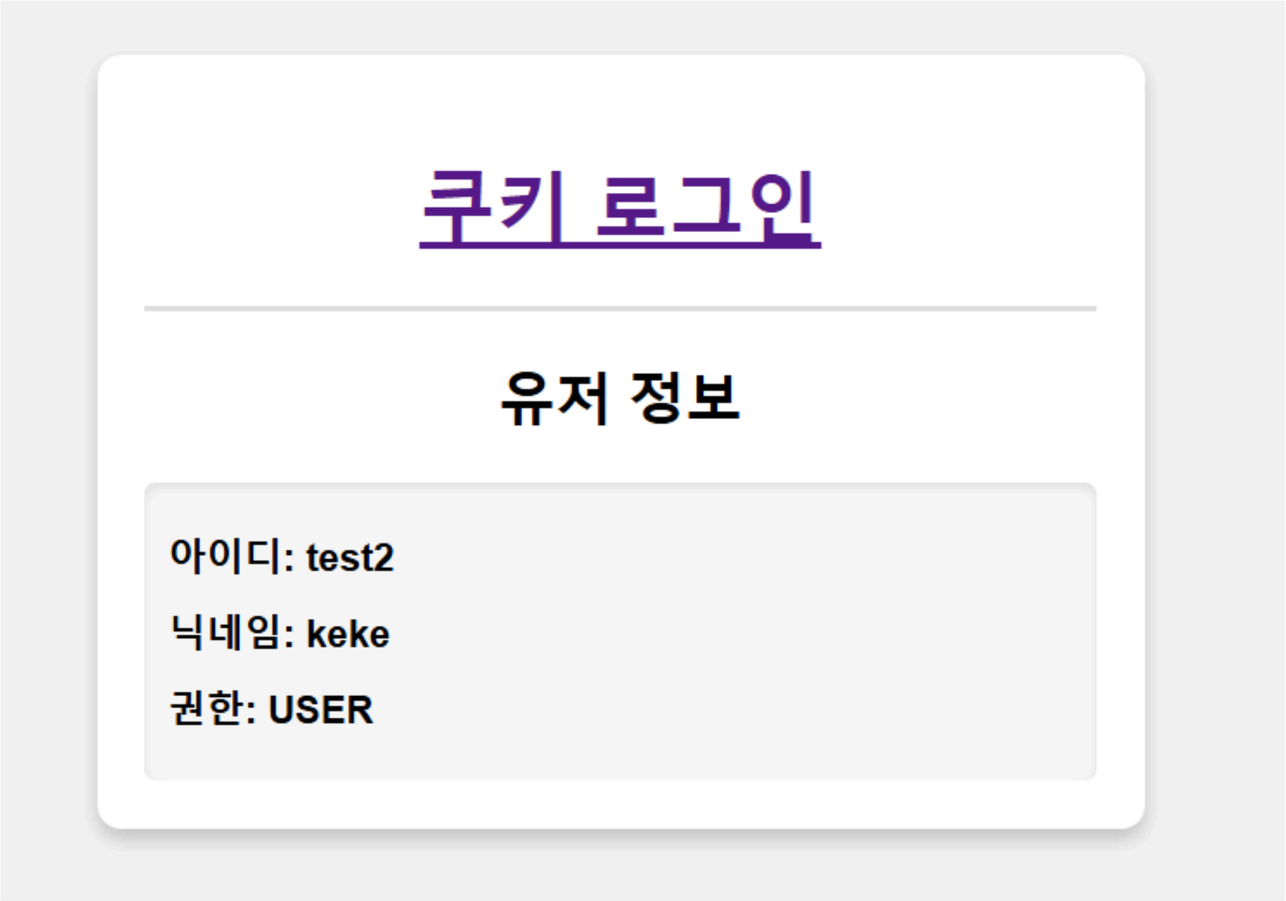
클라이언트의 쿠키 삭제 또는 수정

02. 실제 공격

로그인

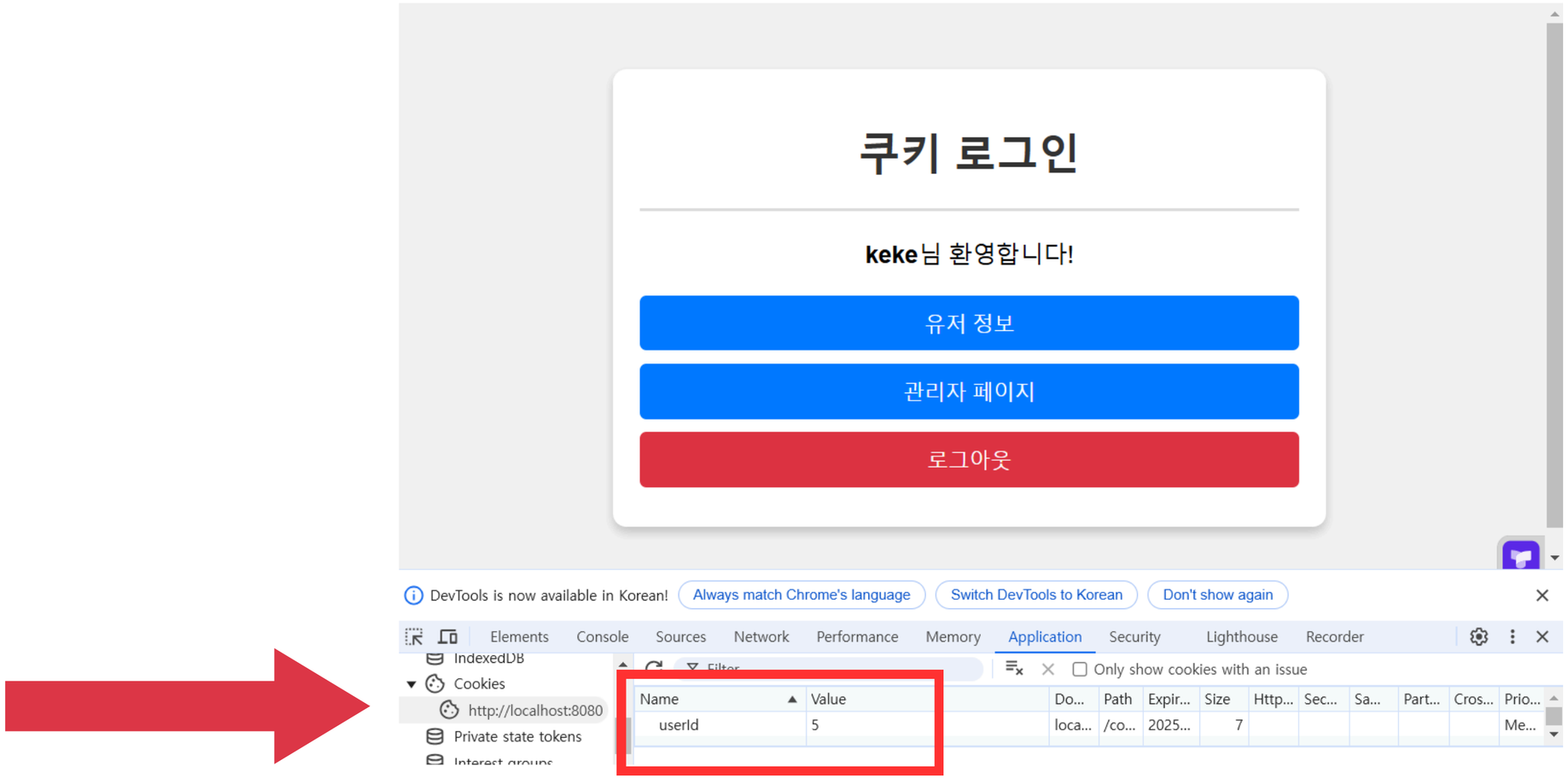
02. 실제 공격

유저 로그인



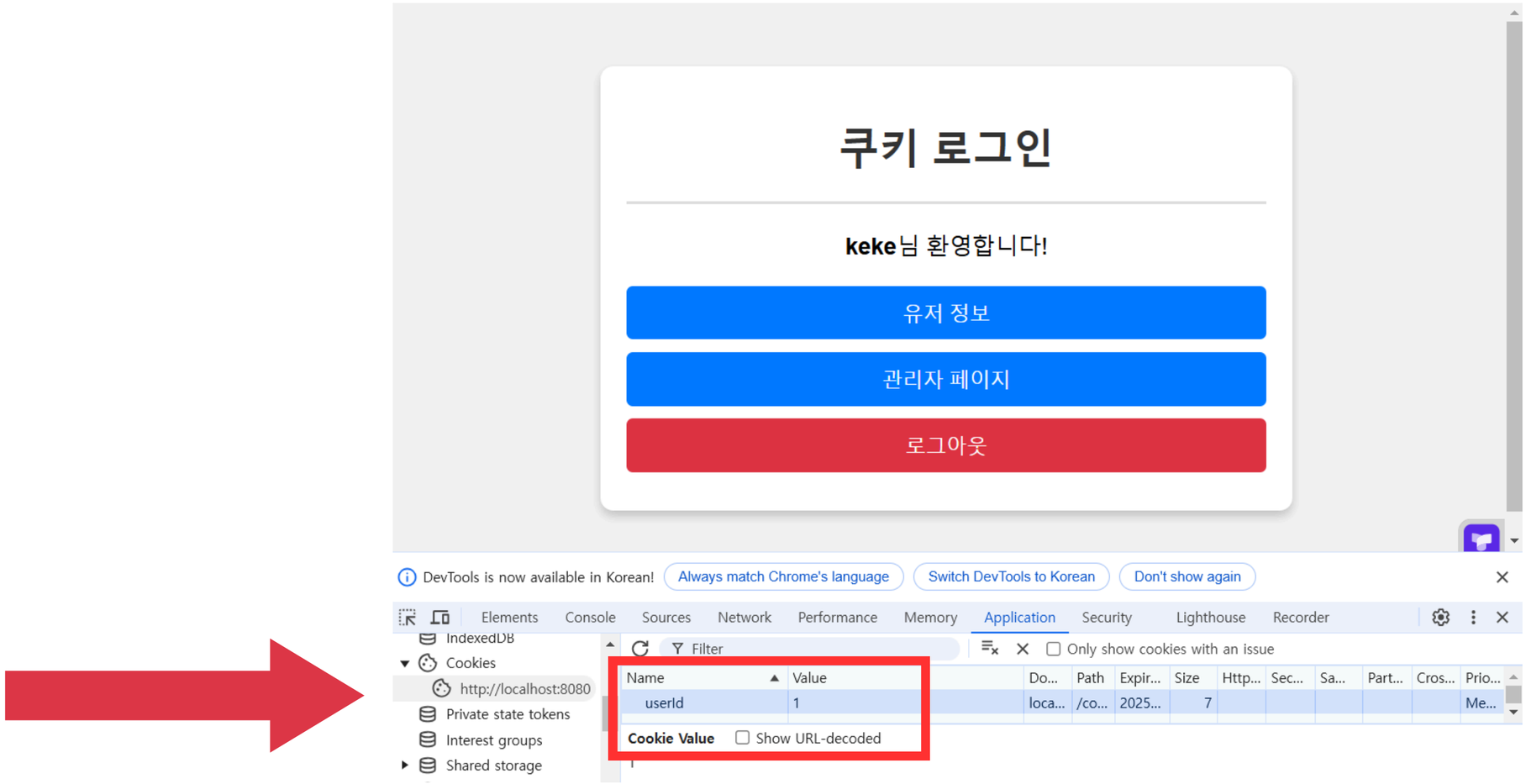
02. 실제 공격

쿠키 확인


쿠키 로그인
keke님 환영합니다!
유저 정보
관리자 페이지
로그아웃
DevTools is now available in Korean!
Always match Chrome's language
Switch DevTools to Korean
Don't show again
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
IndexedDB
Cookies
http://localhost:8080
Private state tokens
Only show cookies with an issue
Name
Value
Do...
Path
Expir...
Size
Http...
Sec...
Sa...
Part...
Cros...
Prio...
userId
5
loca...
/co...
2025...
7
Me...

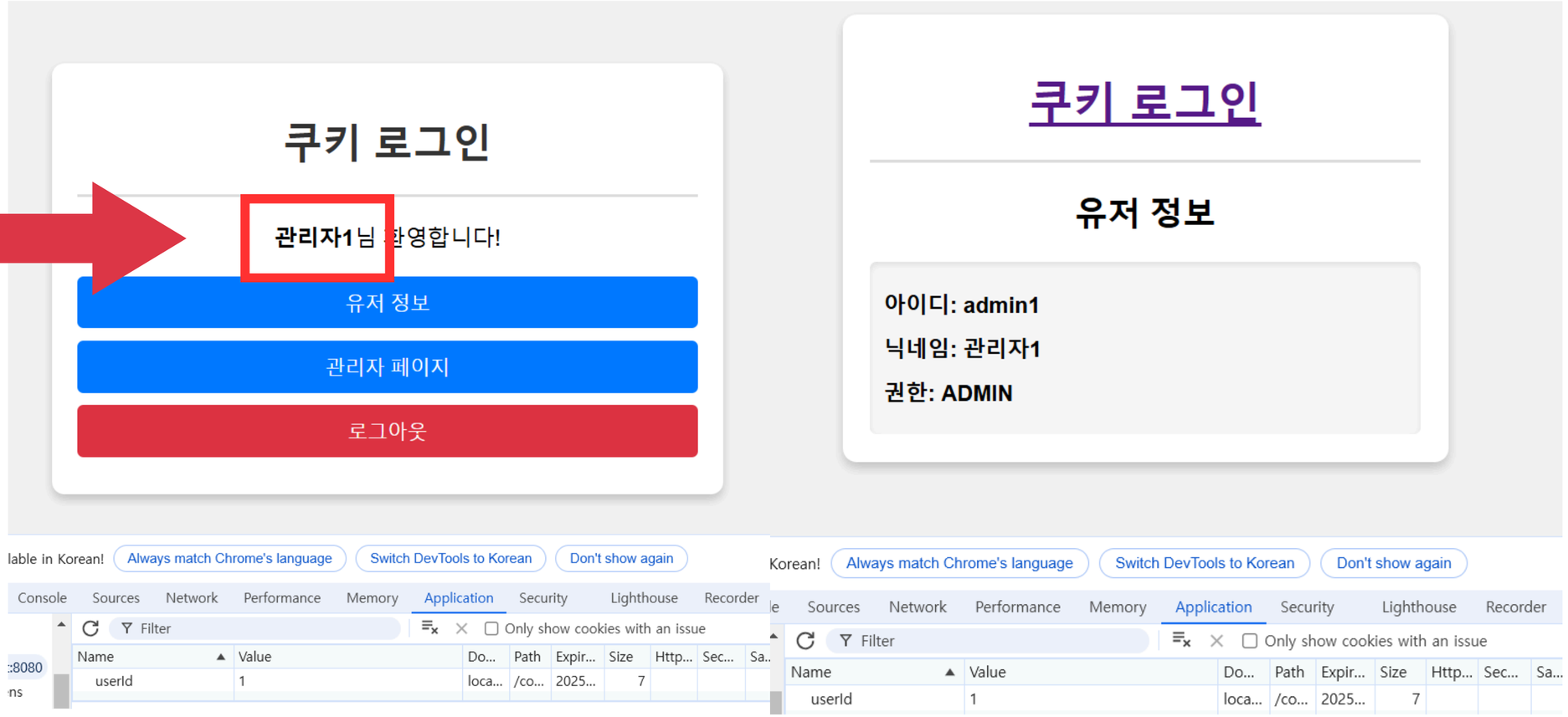
02. 실제 공격

쿠키 값 변경



02. 실제 공격

공격 성공


쿠키 로그인
관리자1님 환영합니다!
유저 정보
관리자 페이지
로그아웃
쿠키 로그인
유저 정보
아이디: admin1
닉네임: 관리자1
권한: ADMIN
lable in Korean!
Always match Chrome's language
Switch DevTools to Korean
Don't show again
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Recorder
Filter
Only show cookies with an issue
Name
Value
Do...
Path
Expir...
Size
Http...
Sec...
Sa...
userId
1
loca...
/co...
2025...
7

세션

03. 세션

정의

상호작용적인 정보 교환을 전제하는 둘 이상의
통신 장치나, 컴퓨터와 사용자의 송수신 연결상태를 의미

03. 세션

장점

Session ID (데이터 노출 X)

서버 제어 가능

클라이언트의 쿠키 크기 제한(4KB) 문제 해결

단점

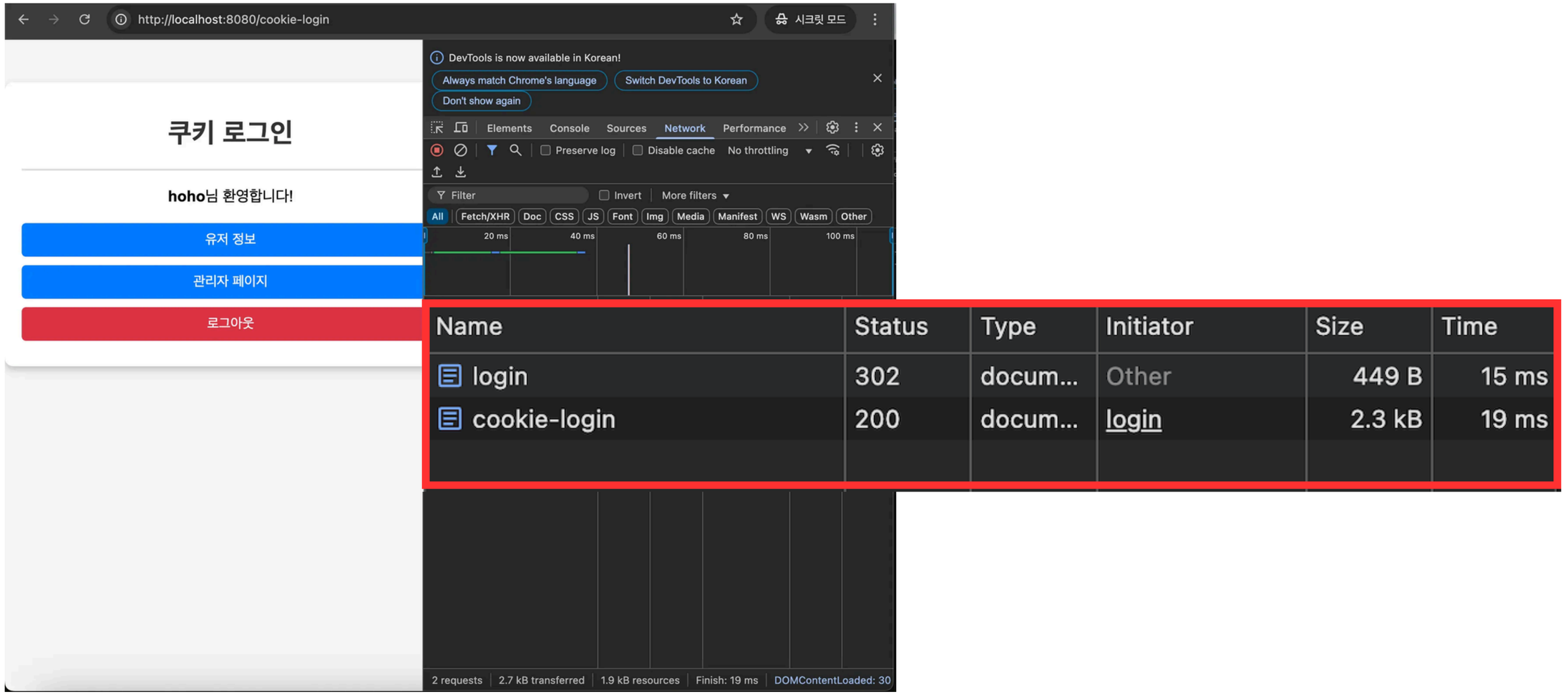
보안 취약점
XSS, CSRF 공격 대상

확장성

속도 저하

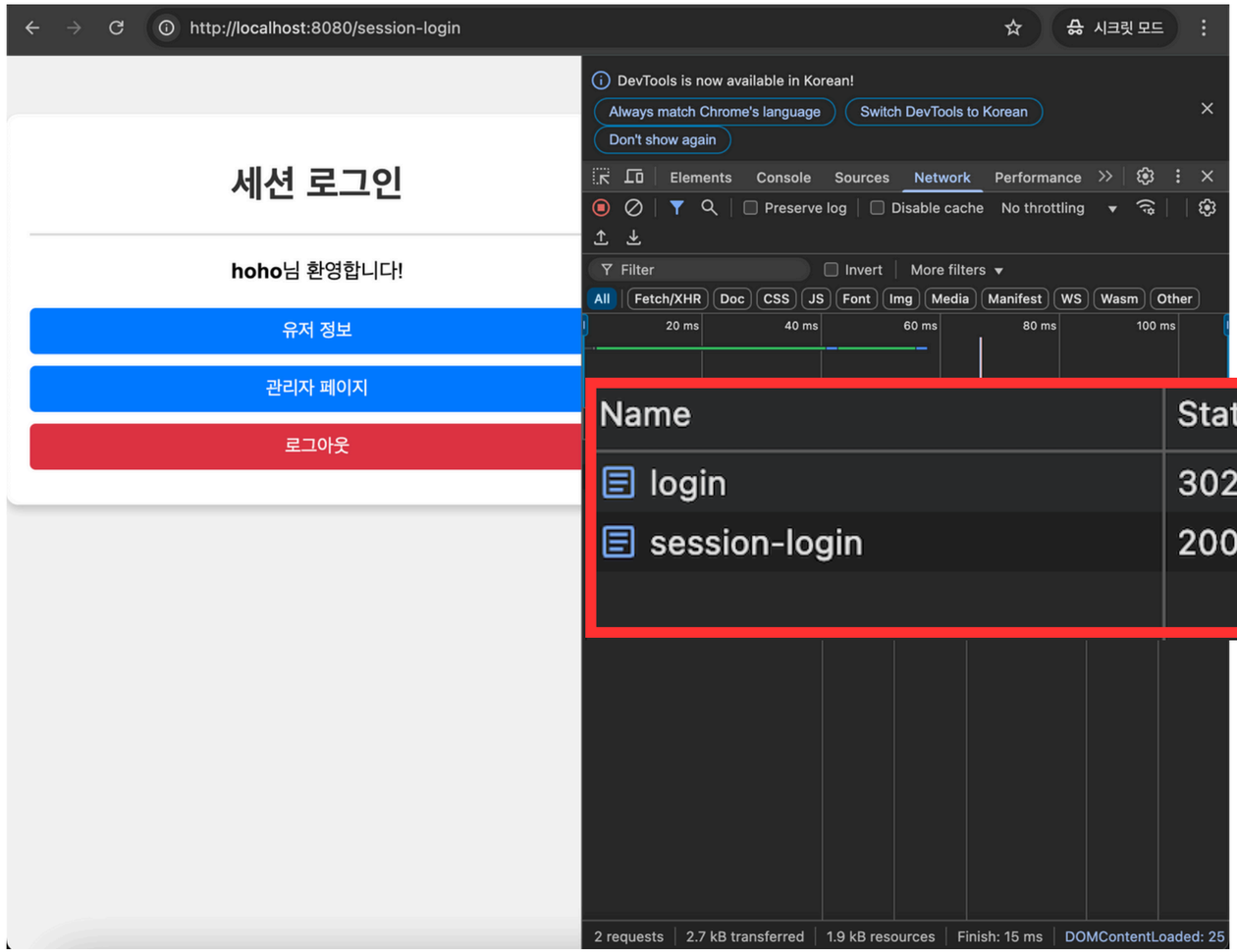
03. 속도 저하

쿠키 로그인



03. 속도 저하

세션 로그인



Name	Status	Type	Initiator	Size	Time
login	302	docum...	Other	450 B	40 ms
session-login	200	docum...	login	2.3 kB	15 ms

03. 중간 정리

쿠키/세션

항목	쿠키	세션
저장위치	클라이언트(브라우저)	서버
보안성	위변조, 탈취 위험	탈취 위험
HTTP 요청시	자동으로 쿠키 전송	세션 ID를 통해 서버에서 관리


토큰

04. 토큰

정의

인증 및 보안 목적으로 사용되는 **고유한 문자열**로,
사용자의 인증 정보나 권한을 포함하여 **세션 상태를**
유지하거나 접근 권한을 검증하는 데 활용되는 데이터

종류

Access

Refresh

JWT

04. 중간 정리

Access / Refresh

항목	Access 토큰	Refresh 토큰
저장 위치	클라이언트	보안이 강화된 저장소 (HTTP-only Secure 쿠키)
사용 목적	API 요청 인증	Access Token 갱신
보안성	탈취 시 즉시 API 사용 가능	탈취 시 Access Token을 지속적으로 발급 받을 위험


JWT

05. JWT

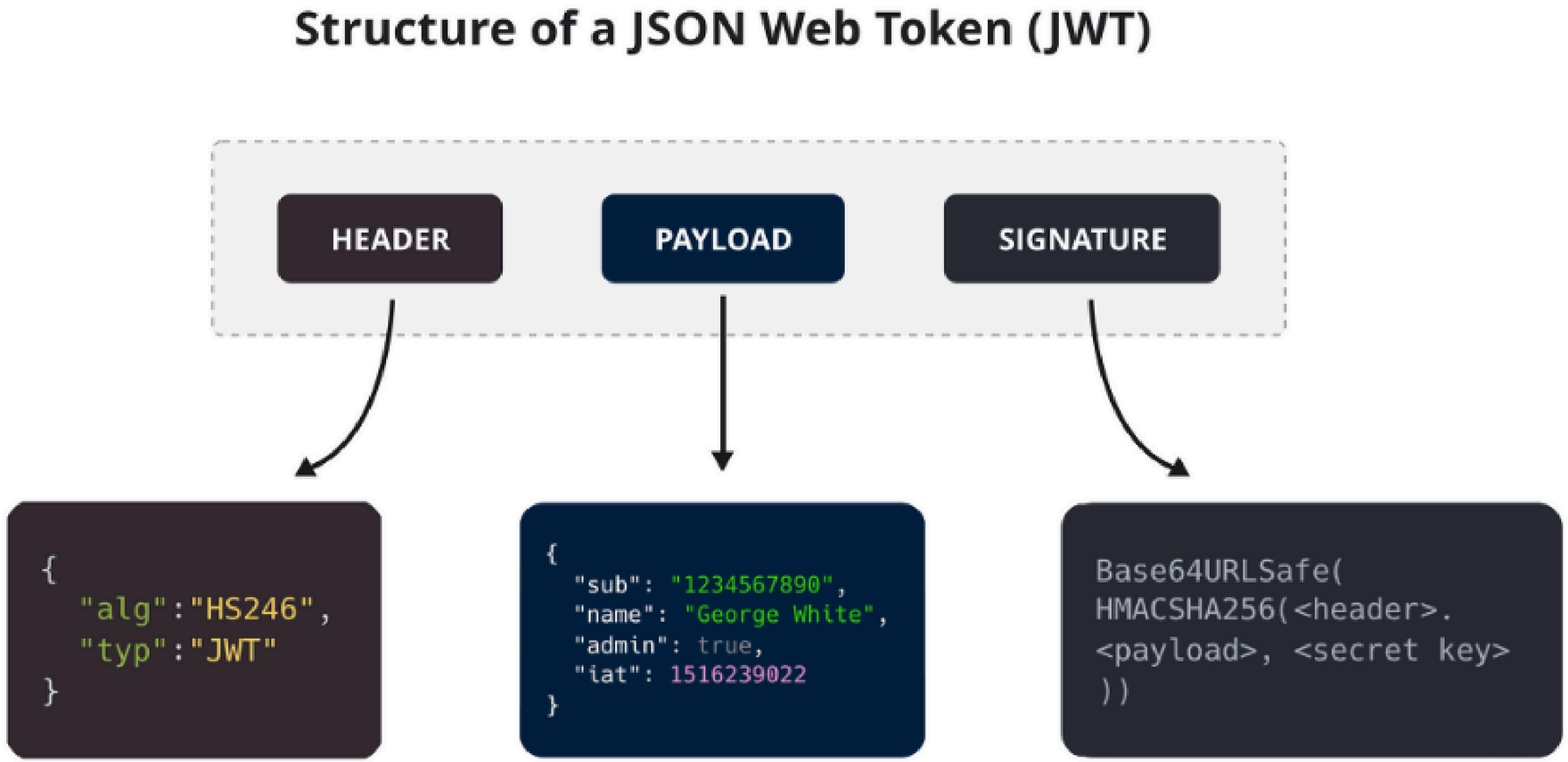
정의

JWT(JSON Web Token)는 **사용자 인증**과 **정보 교환**을 위한 토큰 기반 인증 방식

역할

클라이언트와 서버 간 **인증 및 권한 부여**를 위해 사용되며, 서버가 상태를 저장하지 않는 **무상태(Stateless) 인증 방식**을 가능하게 합니다.

05. JWT 구조


Structure of a JSON Web Token (JWT)
HEADER
PAYLOAD
SIGNATURE
{
"alg":"HS246",
"typ":"JWT"
}
{
"sub": "1234567890",
"name": "George White",
"admin": true,
"iat": 1516239022
}
Base64URLSafe(
HMACSHA256(<header>.
<payload>, <secret key>
))

05. JWT

장점

Stateless - 확장성, 서버 저장 공간

Access, Refresh 토큰의 조합
보안성, 사용자 편의성

단점

Payload 노출 위험성

민감한 정보 포함 금지

05. 중간 정리

JWT

항목	JWT 단점	해결 방법
민감한 정보	Base64URL의 단순한 인코딩 방식	암호화된 저장소, DB에서 저장 API에서 별도로 조회
PayLoad	정보 노출	JWE 방식의 암호화
서명	취약한 알고리즘, 서명이 없는 경우	비대칭키 알고리즘 (RSA) alg: none 옵션

Spring Security

06. Spring Security

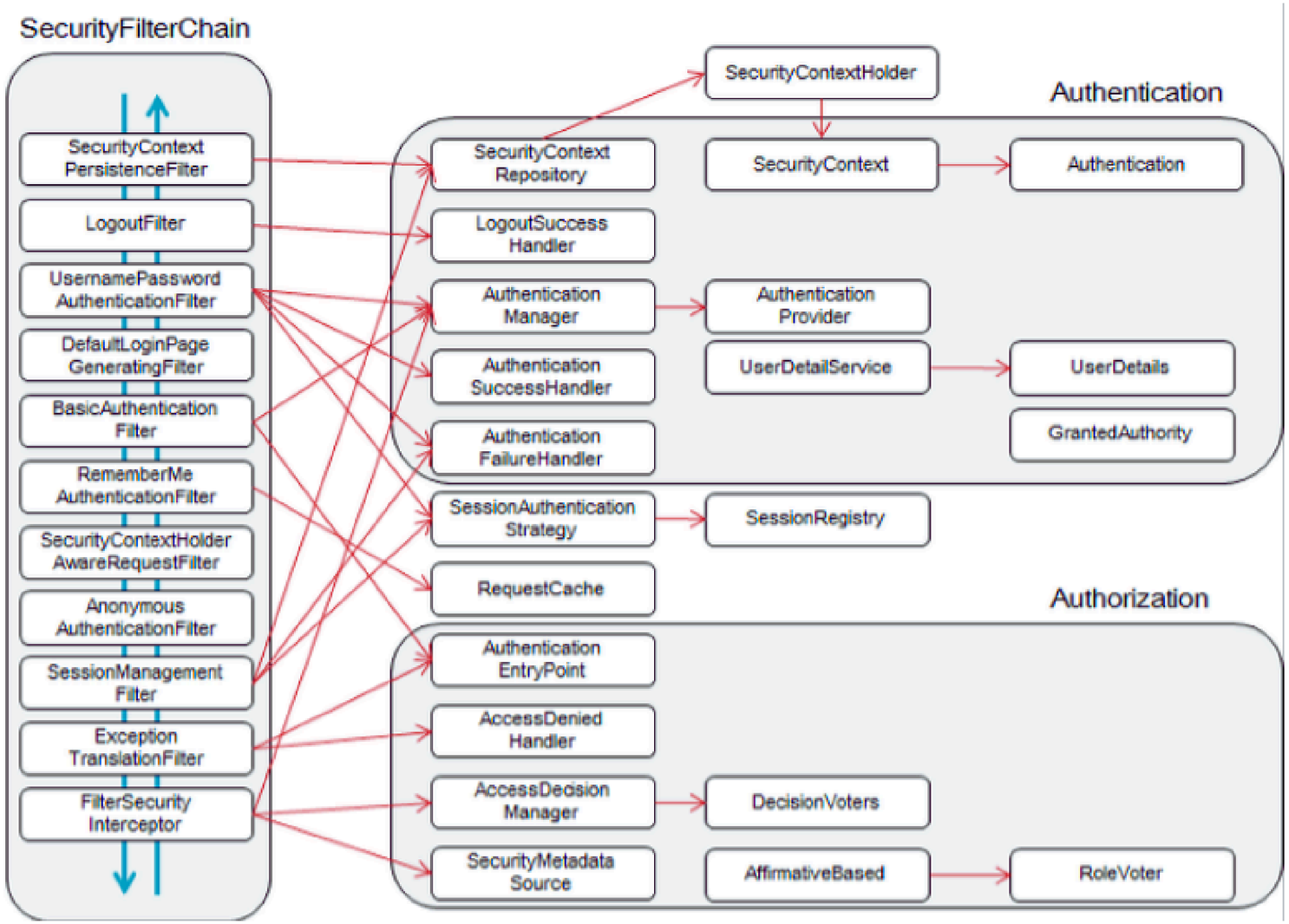
정의

스프링 기반 애플리케이션의 **보안**을
담당 하는 스프링 하위 **프레임워크**

역할

인증과 권한 부여를 효율적으로
처리하고, **보안을 강화**하는
다양한 기능을 제공하여
웹 애플리케이션 보호

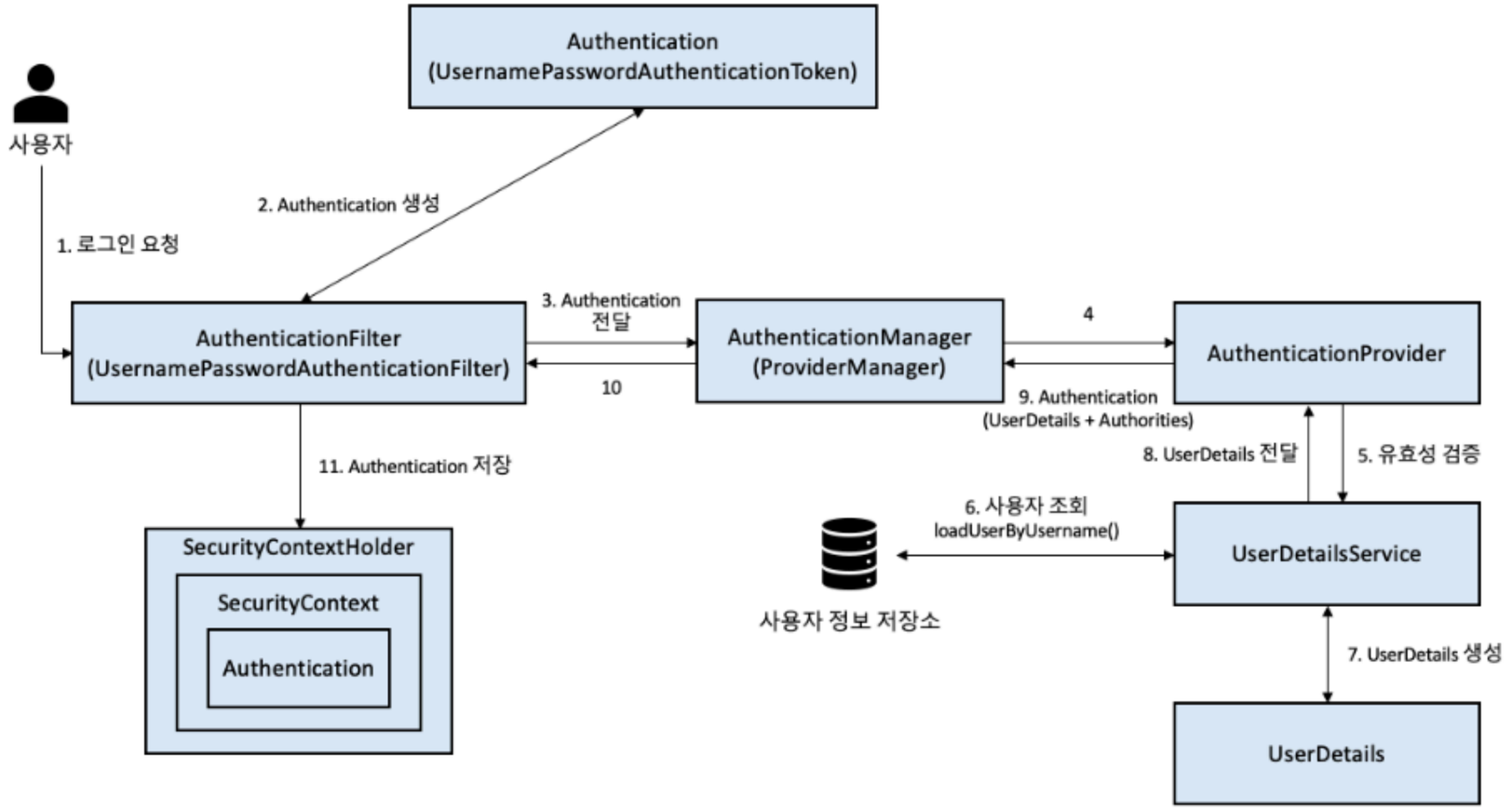
06. Filter Architecture



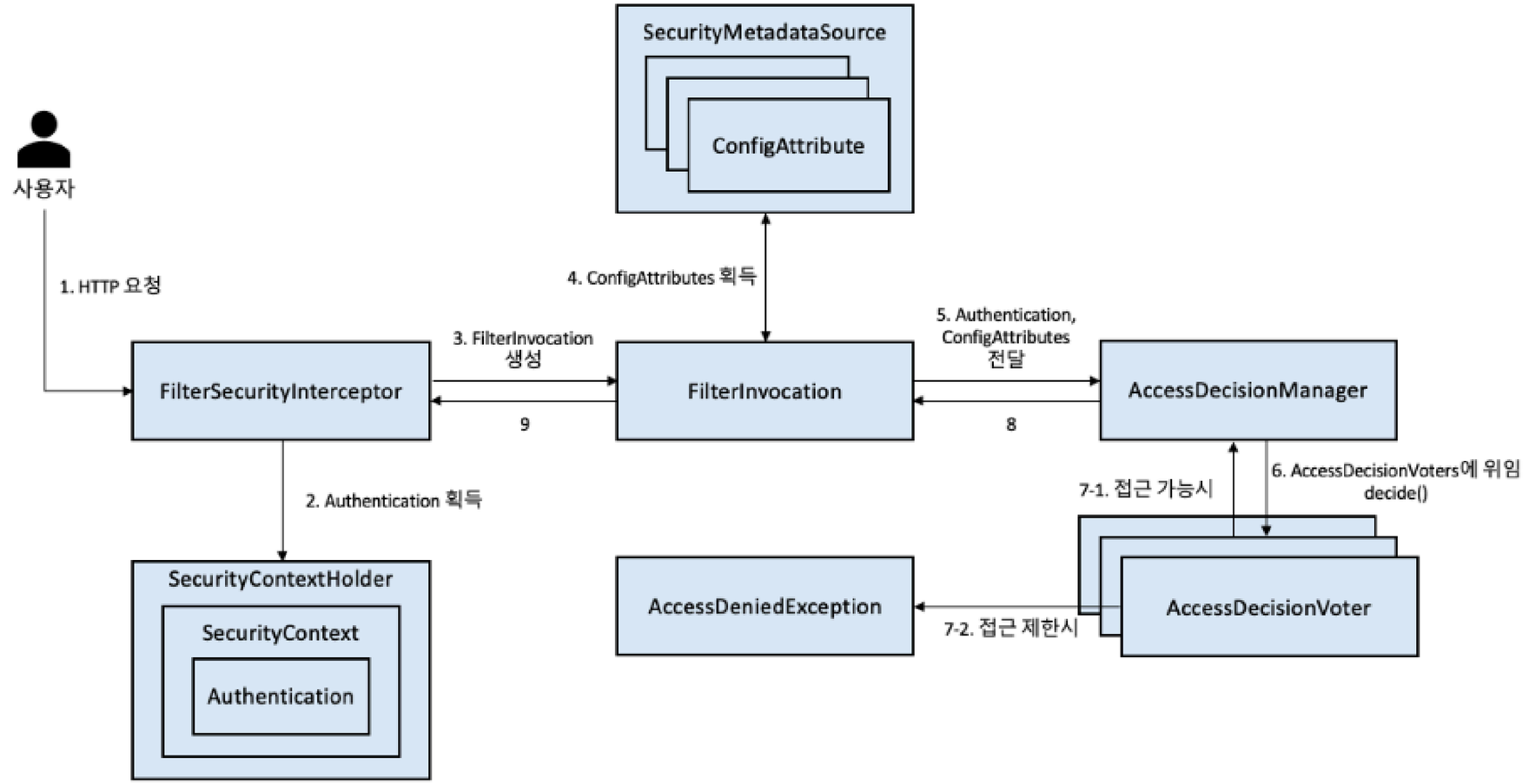
06. SecurityFilterChain - 주요 필터

No	필터	역할
1	Security_Context_Persistence_Filter	SecurityContext 객체 생성, 저장, 조회
2	Username_Password_Authentication_Filter	인증 관리자(ID/PW 파싱 후 인증 요청)
3	Basic_Authentication_Filter	HTTP basic 인증을 처리
4	Bearer_Token_Authentication_Filter	인가 헤더에 포함된 Bearer 토큰 인증
5	Exception_Translation_Filter	요청 중 발생하는 예외 처리
6	Filter_Security_Interceptor	인증된 사용자의 접근 권한을 검사

06. 인증 구조

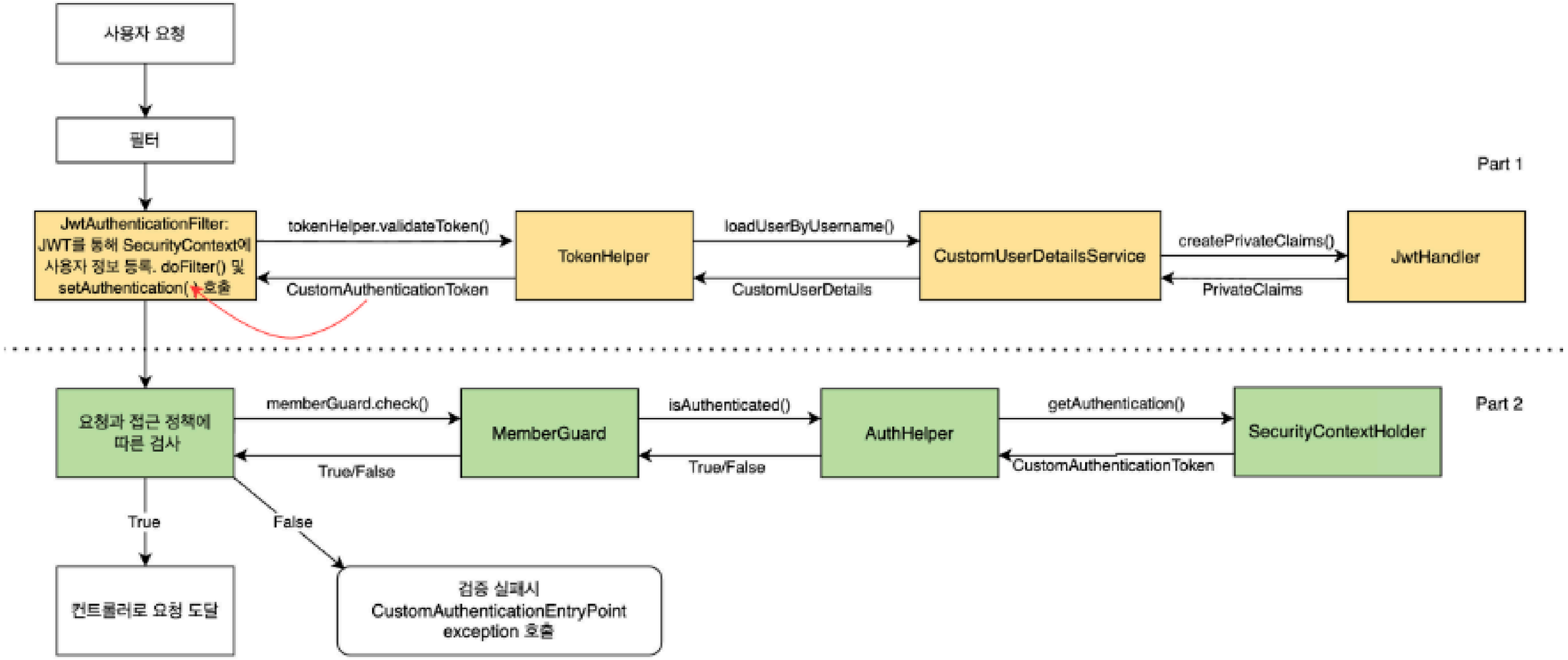

Authentication
(UsernamePasswordAuthenticationToken)
사용자
2. Authentication 생성
1. 로그인 요청
AuthenticationFilter
(UsernamePasswordAuthenticationFilter)
3. Authentication
전달
10
AuthenticationManager
(ProviderManager)
4
AuthenticationProvider
9. Authentication
(UserDetails + Authorities)
8. UserDetails 전달
5. 유효성 검증
11. Authentication 저장
SecurityContextHolder
SecurityContext
Authentication
6. 사용자 조회
loadUserByUsername()
사용자 정보 저장소
UserDetailsService
7. UserDetails 생성
UserDetails

06. 인가 구조


SecurityMetadataSource
ConfigAttribute
사용자
1. HTTP 요청
4. ConfigAttributes 획득
3. FilterInvocation 생성
5. Authentication, ConfigAttributes 전달
FilterSecurityInterceptor
FilterInvocation
AccessDecisionManager
9
8
2. Authentication 획득
7-1. 접근 가능시
6. AccessDecisionVoters에 위임 decide()
SecurityContextHolder
SecurityContext
Authentication
AccessDeniedException
AccessDecisionVoter
7-2. 접근 제한시

Spring Security & JWT

07. 인증 / 인가 구조


사용자 요청
필터
Part 1
JwtAuthenticationFilter:
JWT를 통해 SecurityContext에
사용자 정보 등록. doFilter() 및
setAuthentication() 호출
tokenHelper.validateToken()
CustomAuthenticationToken
TokenHelper
loadUserByUsername()
CustomUserDetails
CustomUserDetailsService
createPrivateClaims()
PrivateClaims
JwtHandler
요청과 접근 정책에
따른 검사
memberGuard.check()
True/False
MemberGuard
isAuthenticated()
True/False
AuthHelper
getAuthentication()
CustomAuthenticationToken
SecurityContextHolder
Part 2
True
False
컨트롤러로 요청 도달
검증 실패시
CustomAuthenticationEntryPoint
exception 호출

최종 정리

08. 최종 정리

프로젝트 단계별 진행 상황

쿠키 → 세션 → JWT → Spring Security → 최종 결론

초기 단계

- 쿠키를 활용한 로그인
- 세션을 활용한 로그인
- 기존 인증 방식의 한계

중간 단계

- JWT, Spring Security 란?
- JWT, Spring Security의 필요성
- Spring Security를 활용한 로그인

마지막 단계

- Spring Security 적용하기
- Spring Security에 JWT 적용

08. 추가 학습 자료 및 다음 단계 제안

추가 학습자료

https://github.com/PersonIn8/Tech_Seminar_01/blob/main/README.md

다음 단계

1. JWT & Spring Security 구현
2. Outh 2.0 구현
3. CSRF (XSRF) / CORS / XSS 설정법

08. 참고문헌

No	Title	URL
1	Spring Security Authentication, Authorization	https://velog.io/@kwj1830/codestates29
2	Spring Security Filter Architecture	https://docs.spring.io/spring-security/reference/servlet/architecture.html#servlet-security-filters
3	Spring boot Modern API	https://github.com/PacktPublishing/Modern-API-Development-with-Spring-6-and-Spring-Boot-3/tree/main/Chapter06
4	Spring Security & JWT	https://velog.io/@tanggu01/Spring-Security-JWT로-인증인가-구현하기

감사합니다.

Spring Security & JWT

클라우드 엔지니어링 1조

유호준, 김지훈, 박재희, 이성빈

Q&A