세션은 서버에 상태를 저장 stateful

토큰은 서버에 상태를 저장하지 않음 stateless

서버의 속도를 높게 유지시켜 준다는 장점이 있다.

JWT(JSON Web Token) 규격 -> 토큰 기반 인증

Application.properties에

spring.security.oauth2.client.provider.issuer-uri=

여기는 서버 인증 인가를 위한 엔드포인트 설정

spring.security.oauth2.client.registration.keycloak-spring-gateway-client.provider=

spring.security.oauth2.client.registration.keycloak-spring-gateway-client.client-id=

spring.security.oauth2.client.registration.keycloak-spring-gateway-client.client-secret=

spring.security.oauth2.client.registration.keycloak-spring-gateway-client.authorization-grant-type= password

spring.security.oauth2.client.registration.keycloak-spring-gateway-client.scope=

keycloak 입장에서 서버는 클라이언트 이므로 id와 키를 입력 한다.

spring.security.oauth2.resourceserver.jwt.jwk-set-uri=

이것으로 validation에 대한 인증을 확인할 것 (이 설정한 엔드포인트)

Spring Security에서 **AuthenticationProvider**는 실제로 사용자의 인증을 수행하는 클래스입니다. 사용자가 로그인을 시도하면 **AuthenticationFilter**가 해당 요청을 가로채고, **AuthenticationProvider**에게 실제 인증을 위임합니다.

인증이 실패하면 **AuthenticationProvider**는 **AuthenticationException**을 던집니다. 이 예외는 **AuthenticationFilter**로 전달되고, 여기서는 **UsernamePasswordAuthenticationFilter**가 해당 역할을 수행합니다.

Spring Security에서 **.failureHandler()** 메서드를 사용하여 로그인 실패 핸들러를 구성할 수 있습니다. 이 핸들러는 로그인에 실패했을 때 실행할 동작을 정의합니다. 예를 들어 로그인 실패시 실행할 작업이나 실패시 보낼 에러 메시지를 설정할 수 있습니다.

따라서 **AuthenticationProvider**에서 인증이 실패하면 **AuthenticationFilter**로 예외가 전달되고, **AuthenticationFilter**는 설정된 실패 핸들러에게 해당 예외를 전달합니다. 실패 핸들러는 설정된 동작을 실행하고, 예를 들어 실패한 로그인을 기록하거나 실패 메시지를 전송하는 등의 작업을 수행합니다.

결국 이 과정은 사용자의 인증 요청이 **AuthenticationProvider**를 통해 처리되고, 실패했을 경우에는 설정된 실패 핸들러에게 예외가 전달되어 추가적인 처리를 할 수 있도록 합니다.