

스프링시작하기

스프링 API 11장

spring security 기본 intercepter 기본

1.1 spring security 기본

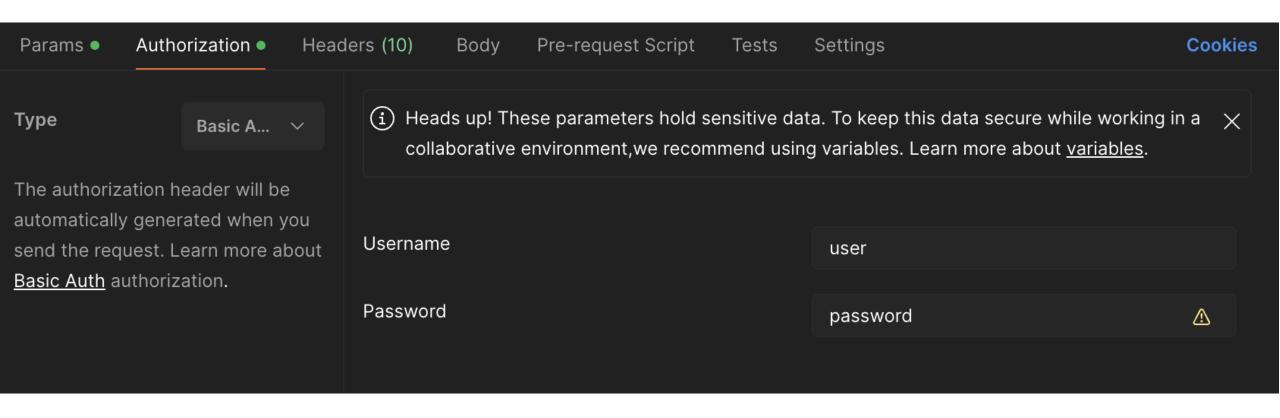
```
@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
```

@Bean

```
UserDetailsService userDetailsService() {
    InMemoryUserDetailsManager userDetailsService = new InMemoryUserDetailsManager();
    UserDetails user = User.withUsername("user")
            .password(passwordEncoder().encode( rawPassword: "password"))
            .authorities("read")
            .build();
    userDetailsService.createUser(user);
    return userDetailsService;
```



1.2 postman Authorization 사용하기



1.3 프로젝트 생성

인텔리 제이에서 Maven 프로젝트를 생성합니다.

UserDetailsService 인터페이스를 구현한 MyUserDetailsService 클래스를 정의합니다. loadUserByUsername 메서드에서는 주어진 사용자 이름에 해당하는 사용자 정보를 가져와서 UserDetails 객체로 반환합니다. 이 예제에서는 하드코딩된 사용자 정보를 반환하도록 하였으며, 실제로는 데이터베이스나 외부 소스에서 사용자 정보를 가져오는 로직을 구현해야 합니다.

```
@Bean
public PasswordEncoder passwordEncoder() {
    return new BCryptPasswordEncoder();
}
```

```
1.4
```

```
@Override
public UserDetails loadUserByUsername(String username) throws UsernameNotFoundExc
   if ("user".equals(username)) {
        // 패스워드만 암호화하여 UserDetails 객체를 생성하여 반환합니다.
       String encodedPassword = passwordEncoder().encode("password");
       return org.springframework.security.core.userdetails.User.builder()
                .username("user")
                .password(encodedPassword)
                .roles("USER")
                .build();
   } else {
       throw new UsernameNotFoundException("User not found with username: " + us
private PasswordEncoder passwordEncoder() {
   return PasswordEncoderFactories.createDelegatingPasswordEncoder();
```

1.5 intercepter 사용하기

```
@Component
public class MyInterceptor implements HandlerInterceptor {
    no usages
    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws Exception {
        String id = request.getHeader( s: "id");
        if(id == null)
            throw new Exception();
        else {
            System.out.println(id);
        return true;
```

1.5 intercepter 사용하기

```
@RequiredArgsConstructor
public class MySecurityConfig implements WebMvcConfigurer {
    private final MyInterceptor myInterceptor;
    no usages
    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        System.out.println("되나");
        registry.addInterceptor(myInterceptor)
                .addPathPatterns("/**");
```



1.5 intercepter 사용하기

```
@ControllerAdvice
public class UserControllerAdvice extends ResponseEntityExceptionHandler {
    @ExceptionHandler(Exception.class)
    public final ResponseEntity<Object> AllException(Exception ex, WebRequest webRequest){
        return new ResponseEntity<>( body: "AllException", HttpStatus.INTERNAL_SERVER_ERROR);
```

1.5 Cors 설정하기

```
@Override
public void addCorsMappings(CorsRegistry registry) {
    registry.addMapping( pathPattern: "/**")
            .allowedOrigins("*")
            .allowedMethods(
                    HttpMethod.GET.name(),
                    HttpMethod.POST.name(),
                    HttpMethod.PUT.name(),
                    HttpMethod.PATCH.name(),
                    HttpMethod.DELETE.name(),
                    HttpMethod.OPTIONS.name()
            .maxAge(3600)
```

1.5 Cors 설정하기

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8" />
    <title>CORS Test</title>
    <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
</head>
<body>
<script>
    $.ajax({
                 : "http://localhost:8080/users",
        url
                 : "GET",
        type
        beforeSend: function(xhr) {
            xhr.setRequestHeader("custom-header", "custom value");
        },
        success : function(result){
            console.log(result);
    });
</script>
</body>
```

고맙습니다.