스프링부트로 RestFulAPI 구현하기

12장 cors JWT interceptor

- cors
- JWT
- interceptor
- @JsonIgnore

박명회

12장 Cors 해결 filter 사용하기

```
Configuration
@RequiredArgsConstructor
public class WebConfig implements WebMvcConfigurer {

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

}
```

12장 Cors 해결 preflight

```
@Component
@RequiredArgsConstructor
public class AuthInterceptor implements HandlerInterceptor {

    private final TokenManager tokenManager;

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws
Exception {

        if (request.getMethod().equals("OPTIONS")) {
            return true;
        }


       ...
   }
```

12장 JWT 란

액세스 토큰(access token)과 리프레시 토큰(refresh token)은 인증 및 권한 부여를 위해 사용되는 보안 토큰입니다. 이 두 가지 토큰은 주로 OAuth 2.0 프로토콜에서 사용되며, 사용자의 인증 및 권한 부여를 관리하는 데 중요한 역할을 합니다.

액세스 토큰(Access Token):

액세스 토큰은 사용자가 리소스에 접근할 때 사용되는 토큰입니다.
주로 OAuth 2.0 인증 서버에 의해 발급됩니다.
액세스 토큰은 일반적으로 짧은 유효 기간을 가지며(예: 몇 분 또는 몇 시간), 발행 후 일정 시간이 지나면 만료됩니다.
만료된 경우, 사용자는 새로운 액세스 토큰을 받기 위해 인증 서버에 다시 인증해야 합니다.

리프레시 토큰(Refresh Token):

리프레시 토큰은 액세스 토큰을 갱신하기 위해 사용되는 토큰입니다.
주로 OAuth 2.0 인증 서버에 의해 발급됩니다.
리프레시 토큰은 액세스 토큰보다 긴 유효 기간을 가집니다(예: 일반적으로 몇 일 또는 몇 주).
액세스 토큰이 만료되었을 때, 사용자는 리프레시 토큰을 사용하여 새로운 액세스 토큰을 요청할 수 있습니다.
리프레시 토큰은 보안 상의 이유로 액세스 토큰보다 안전한 저장소에 보관되어야 합니다.

12장 JWT dependency추가

```
runtimeOnly 'io.jsonwebtoken:jjwt-impl:0.12.5'
runtimeOnly 'io.jsonwebtoken:jjwt-jackson:0.12.5'
implementation 'io.jsonwebtoken:jjwt-api:0.12.5'
```

12장 TokenManager.class

```
@Component
public class TokenManager {

    @Value("${mh.jwt.secret}")  //application.yml 에 값 가져오기
    private String mykey;


    public String generateToken(Member member) {

        return Jwts.builder()
                .subject("mhToken")
                .claim("id", member.getMemberId())
                .claim("username", member.getUsername())
                .claim("role", member.getRole())
                .claim("email", member.getEmail())
                .expiration(new Date(System.currentTimeMillis() + 1000 * 60 * 15)) // 유효시간 15분 설정
                .signWith(hmacShaKeyFor(mykey.getBytes()))
                .compact();

    }
```

12장 TokenManager.class

```
// 토큰 검증해주는 함수.
public Jws<Claims> validateToken(String token) {

    try {
          Jws<Claims> jws = Jwts.parser()
                .setSigningKey(hmacShaKeyFor(mykey.getBytes()))
                .build()
                .parseClaimsJws(token);
          return jws;
   } catch (ExpiredJwtException e) {
         log.info("token 만료", e);
         throw new AuthenticationException(ErrorCode.TOKEN_EXPIRED);
   } catch (Exception e) {
         log.info("유효하지 않은 token", e);
         throw new AuthenticationException(ErrorCode.NOT_VALID_TOKEN);
   }

   return null

}
```

12장 intercepter 추가하기

```
@Component
@RequiredArgsConstructor
public class AuthenticationInterceptor implements HandlerInterceptor {

    private final TokenManager tokenManager;

    @Override
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler) throws
Exception {

        String authorizationHeader = request.getHeader("Authorization");
         String token = authorizationHeader.split(" ")[1];

        Jws<Claims> claims = tokenManager.validateToken(token);

        String tokenType = tokenClaims.getSubject();

        if(!TokenType.eqauls("mhToken")) {
            throw new AuthenticationException(ErrorCode.MY_TOKEN_TYPE);
        }
        return true;
    }
}
```

12장 Cors 해결 filter 사용하기

```
Configuration
@RequiredArgsConstructor
public class WebConfig implements WebMvcConfigurer {

    private final AuthenticationInterceptor authenticationInterceptor;

    ...cors 생략...

    @Override
    public void addInterceptors(InterceptorRegistry registry) {
        registry.addInterceptor(authenticationInterceptor)
                .addPathPatterns("/**")
                .excludePathPatterns("/user/login",
                        "/user/logout");
    }
}
```

감사합니다