CH5 스프링 시큐리티

정승혜

Oauth 2.0

- 인증을 위한 오픈 스탠다드 프로토콜
- 어플리케이션에서 타사의 API를 사용하고 싶을 때 권한 획득을 위해 사용

- Authentication : 인증
- Authorization : 허가

- 회원가입 과정을 거쳐 ID/Password를 통한 로그인을 인증(Authentication)

- Oauth 2.0은 A서비스(google,네이버,카카오,페이스북…)의
  이메일 정보에 B 서비스의 접근을 허가(Authorization)하여
  사용자를 인증(Authentication) 함

스프링 부트와 Oauth 2.0

- Spring-security-oauth2-autoconfigure

```
compile('org.springframework.boot:spring-boot-starter-oauth2-client')
```

- 스프링 부트용 라이브러리
- 신규기능, 상속/오버라이딩에 대한 확장포인트 사용을 위해 spring Security Oauth2 Client 라이브러리 사용

< 1.5 버전과의 차이점 >

- Oauth2 정보의 url 주소로 명시되던 것들이 enum 으로 대체됨
- CommonOAuthProvider

구글 로그인을 사용하기 - 1

- **Google cloud Platform에서 신규 서비스 생성**

- API 및 서비스 - 사용자 인증 정보/동의화면 구성
  - 어플리케이션 이름
  - 지원 이메일
  - 사용할 Google API의 범위

- Oauth 클라이언트 ID 생성
  - 어플리케이션 유형
  - 인증 성공시 리다이렉션 URI
  - Client ID / Client Secret 발급

구글 로그인을 사용하기 - 2

- **Application-oauth.properties 파일 생성**

⇒Client Id, Secret 등 관련 정보 등록

```
spring.security.oauth2.client.registration.google.client-id=client ID
spring.security.oauth2.client.registration.google.client-secret=client secret
spring.security.oauth2.client.registration.google.scope=profile,email
```

- **Oauth의 scope**

- 구글은 profile/email/openID 제공사
- 네이버/카카오는 OpenID provider가 아님

=> 한번에 서비스를 관리하기 위해서 profile,email로 등록

구글 로그인을 사용하기 - 2

- 스프링부트와 properties 파일

⇒ 스프링부트에서 application-xxx.properties 라는 파일은 xxx라는 이름의 Profile로 관리됨

그래서 profile=xxx와 같은 식으로 프로퍼티 설정을 가져올 수 있음

교재에서 호출 방식은 application.properties에 해당 프로퍼티 파일을 등록

```
spring.profiles.include=oauth
```

구글 로그인을 사용하기 - 3

- Domain
  - User.java - entity class
  - Role.java - enum
  - UserRepository.java - User 클래스의 CRUD

- config.auth
  - SecurityConfig.java - configure() 서비스 등록
  - CustomOAuth2UserService - loadUser(), saveOrUpdate() : 넘어온 사용자 정보를 업데이트 혹은 등록, 세션에 유저 등록
  - DTO
    - OAuthAttributes - 사용자 정보 DTO
    - SessionUser - 인증된 사용자 정보를 담는 DTO

- Index.mustache 버튼 및 경로 연동 코드 추가
- IndexController에 UserName을 사용할 수 있게 모델 저장 코드 추가

구글 로그인을 사용하기 - 3

```
public class CustomOAuth2UserService implements OAuth2UserService<OAuth2UserRequest,
OAuth2User> {
    private final UserRepository userRepository;
    private final HttpSession httpSession;


    private User saveOrUpdate(OAuthAttributes attributes){
        User user = userRepository.findByEmail(attributes.getEmail())
                .map(entity->entity.update(attributes.getName(), attributes.getPicture()))
                .orElse(attributes.toEntity());

        return userRepository.save(user);
    }
```

구글 로그인을 사용하기 - 3

```
    @Override
    public OAuth2User loadUser(OAuth2UserRequest userRequest) throws
OAuth2AuthenticationException{
        OAuth2UserService<OAuth2UserRequest, OAuth2User> delegate = new
DefaultOAuth2UserService();
        OAuth2User oAuth2User = delegate.loadUser(userRequest);

        // 현재 로그인 진행 중인 서비스를 구분하는 코드
        String registrationId = userRequest.getClientRegistration().getRegistrationId();
        // OAuth2 로그인 진행 시 키가 되는 필드 값, 구글의 기본코드는 "sub", 네이버/구글 로그인 동시
지원시 사용됨
        String userNameAttributeName = userRequest.getClientRegistration().getProviderDetails()
                .getUserInfoEndpoint().getUserNameAttributeName();
```

구글 로그인을 사용하기 - 3

```
        // OAuth2UserService를 통해 가져온 OAuth2User의 attribute를 담을 클래스
        OAuthAttributes attributes = OAuthAttributes.of(registrationId,
userNameAttributeName,
                oAuth2User.getAttributes());

        // 세션에 사용자 정보를 저장하기 위한 Dto 클래스
        User user = saveOrUpdate(attributes);
        httpSession.setAttribute("user",new SessionUser(user));

        return new DefaultOAuth2User(
                Collections.singleton(new SimpleGrantedAuthority(user.getRoleKey())),
                attributes.getAttributes(),
                attributes.getNameAttributeKey());
    }
```

구글 로그인을 사용하기 - 3

- SessionUser - 인증된 사용자 정보를 담는 DTO

- **세션 유지에 User(entity) 클래스를 사용하지 않는 이유**

- 직렬화 구현이 안 되어있음
- 엔티티 클래스는 언제 관계가 생길지 모르기에 직렬화시
  성능 이슈, 부수 효과 발생 확률 증가

=> 직렬화 기능을 가진 세션 dto 작성

구글 로그인을 사용하기 - 3

왜 직렬화가 필요한가?

- 자바 시스템 개발에 최적화, 시스템간 객체의 변환이 용이해짐
- JVM 메모리에만 상주된 객체 데이터를 영속화 할 수 있음
- 시스템이 종료되더라도 없어지지 않고, 네트워크 전송이 가능
- 필요할 때 직렬화된 객체 데이터를 역직렬화하여 바로 사용 가능

- **직렬화의 사용**

1. 서블릿 세션(session)
2. 캐시
   - 퍼포먼스를 위해 캐시를 사용할 때, 특정 데이터 객체를 저장해두고 동일한 요청이 잦을 때 객체로 응답, 이때 직렬화된 데이터로 저장
3. 자바 RMI (원격 시스템 간 메시지 교환)