스프링 시큐리티와 OAuth2.0으로 로그인 기능 구현하기

작성일시	@2022년 6월 12일 오후 12:57
파트	5장
자료	
작성자	

Ch05. 스프링 시큐리티와 OAuth2.0으로 로그인 기능 구현하기

☑️스프링 시큐리티

: 막강한 인증(Authentication)과 인가(Authorization) or 권한 부여 기능을 가진 프레임워크임.

확장성을 고려한 프레임워크이기 때문에 손쉽게 추가, 변경 가능하며 1.5에서 2.0으로 넘어오면서 더욱 강력해짐

→ 사실상 스프링 기반의 애플리케이션에서는 **보안을 위한 표준**이라고 보면 됨. (권장)

- 접근 주체(Principal) : 보호된 리소스에 접근하는 대상
- 인증(Authentication) : 보호된 리소스에 접근한 대상에 대해 이 유저가 누구인지, 애플리케이션의 작업을 수행해도 되는 주체인지 확인하는 과정(ex. Form 기반 Login)
- 인가(Authorize) : 해당 리소스에 대해 접근 가능한 권한을 가지고 있는지 확인하는 과정(After Authentication, 인증 이후)

- 권한 : 어떠한 리소스에 대한 접근 제한, 모든 리소스는 접근 제어 권한이 걸려있다. 즉, 인가 과정에서 해당 리소스에 대한 제한된 최소한의 권한을 가졌는지 확인.

5.1 스프링 시큐리티와 스프링 시큐리티 Oauth2 클라이언트

- 스프링 부트 1.5에서의 OAuth2 연동 방법과 2.0에서의 연동방법이 크게 달라졌음에도 불구하고 'spring-security-oauth2-autoconfigure' 라이브러리 덕분에 2.0에서도 1.5에서 쓰던 설정을 그대로 사용할 수 있었음.

- 하지만 이 책에서는 **스프링부트2 방식인 Spring Security Oauth2 Client 라이브러리**를 사용해서 진행할 예정임. 이유는 다음과 같음.

    1) 스프링팀에서 1.5에서 사용되던 'spring-security-oauth' 프로젝트는 신규기능을 추가하지 않겠다고 결정함. → 신규 기능은 새 oauth2 라이브러리에서만 지원하겠다고 선언

    2) 스프링부트용 라이브러리(starter) 출시

    3) 기존 방식과 달리 신규 라이브러리 경우 확장 포인트를 고려해서 설계된 상태

- 스프링 부트 1.5와 2.0의 차이는 spring-security-oauth2-autoconfigure 라이브러리를 썼는지 확인하고 application.properties 혹은 application.yml 정보가 차이가 있는지 비교해야 함

✅로그인 기능을 id/password 방식보다 소셜 로그인 기능을 사용하는 이유는?

- 로그인 시 보안, 비밀번호 찾기, 회원가입 시 이메일 혹은 전화번호 인증, 비밀번호 변경, 회원정보 변경
- 직접 구현 시 앞선 목록을 모두 개발해야 하지만, OAuth 사용 시 이를 소셜에게 맡겨 서비스 개발에 집중 가능

5.2 구글 서비스 등록

- 승인된 리디렉션 URL : http://localhost:8070/login/oauth2/code/google

  - 서비스에서 파라미터로 인증 정보를 주었을 때 인증이 성공하면 구글에서 리다이렉트할 URL
  - 스프링 부트 2버전의 시큐리티에서는 기본적으로 {도메인}/login/oauth2/code/{소셜서비스코드}로 리다이렉트 URL을 지원하고 있음.
  - AWS 서버에 배포하게 되면 localhost 외에 추가로 주소를 추가해야 함. (이후에 진행예정)

➡️리다이렉트(redirect)는 웹 브라우저(사용자)가 어떤 URL로 웹 서버를 요청했을때 다른 URL로 넘겨주는 것을 말함. 예를 들면, Gmail로 접속했을 때 로그인이 되어 있지 않다면, 로그인이 선행되어야 하기 때문에 로그인 페이지로 이동시키는 것을 리다이렉트라고 함.

5.3 구글 로그인 연동하기

User 클래스를 사용하지 않은 이유

- User클래스를 사용하지 않고 따로 Dto 클래스인 SessionUser를 생성하여 사용자 정보를 저장함.

- User 클래스를 그대로 사용하면 다음과 같은 에러가 발생함.

```
Failed to convert from type [java.lang.Object] to type [byte[]] for value
'com.jojoldu.book.springboot.domain.user.User@4a43d6'
```

→ 세션에 저장하기 위해 User 클래스를 세션에 저장하려고 하니 User 클래스에 직렬화를 구현하지 않았다는 의미의 에러임. 그렇다고 User 클래스에 직렬화 코드를 넣으려고 하니, User 클래스가 엔티티이기 때문에 조심스러움. 엔티티 클래스는 언제 다른 엔티티와 관계가 형성될지 모르니 직렬화 기능을 가진 세션 Dto를 하나 추가로 만드는 것이 이후 운영 및 유지보수에 편함.

☑️ 구글 로그인 후 등록시 권한거부(403)에러 뜸 → 권한 변경해주기

: h2-console로 가서 `update user set role = 'USER';`

- 세션에는 이미 GUEST인 정보로 저장되어있으니 로그아웃 후 다시 로그인하여 세션 정보를 최신 정보로 갱신한 후 글 등록을 해야함. → 정상적으로 글이 등록됨.

5.4 어노테이션 기반으로 개선하기

- IndexController 중 세션값을 가져오기

```
SessionUser user = (SessionUser) httpSession.getAttribute("user");
```

- index 메소드 외에 다른 컨트롤러와 메소드에서 세션값이 필요하면 그때마다 직접 세션에서 값을 가져와야 함. =같은 코드가 불필요하게 반복됨.

- 이 부분을 메소드 인자로 세션값을 바로 받을 수 있도록 변경함.

: config.auth패키지에 @LoginUser 어노테이션 생성, 어노테이션을 사용하기 위한 환경 및 설정 추가 (ArgumentResolver 사용)

➡️ 어느 컨트롤러든지 @LoginUser만 사용하면 세션 정보를 가져올 수 있게 됨.

5.5 세션 저장소로 데이터베이스 사용하기

- 현재 우리가 만든 서비스는 애플리케이션을 재실행하면 로그인이 풀림.

- 세션이 내장 톰캣의 메모리에 저장되기 때문.

- 메모리에 저장되다 보니 내장 톰캣처럼 애플리케이션 실행 시 실행되는 구조에선 항상 초기화됨. 즉, 배포할 때마다 톰캣이 재시작되는 것.
- 추가적인 문제로, 2대 이상의 서버에서 서비스하고 있다면 톰캣마다 세션 동기화 설정을 해야만 함.

→ 실제 현업에선 세션 저장소에 대해 다음의 3가지 방법 중 하나를 선택해서 사용함.

(1) 톰캣 세션을 사용한다.

- 일반적으로 별다른 설정을 하지 않을 때 기본적으로 선택되는 방식임.
- 이렇게 될 경우 톰캣(WAS)에 세션이 저장되기 때문에 2대 이상의 WAS가 구동되는 환경에서는 톰캣들 간의 세션 공유를 위한 추가 설정이 필요함.

(2) MySQL과 같은 데이터베이스를 세션 저장소로 사용함.

- 여러 WAS 간의 공용 세션을 사용할 수 있는 가장 쉬운 방법임.
- 많은 설정이 필요 없지만, 결국 로그인 요청마다 DB IO가 발생하여 성능상 이슈가 발생할 수 있음.
- 보통 로그인 요청이 많이 없는 백오피스, 사내 시스템 용도에서 사용함.

(3) Redis, Memcached와 같은 메모리 DB를 세션 저장소로 사용함.

- B2C 서비스에서 가장 많이 사용하는 방식임.
- 실제 서비스로 사용하기 위해서는 Embedded Redis와 같은 방식이 아닌 외부 메모리 서버가 필요함.

- 두번째 방식인 **데이터베이스를 세션 저장소로 사용하는 방식**을 선택하여 진행할 예정

1) 설정이 간단하고 사용자가 많은 서비스가 아니며 비용 절감을 위해서

2) AWS에서 이 서비스를 배포하고 운영할 때를 생각하면 레디스와 같은 메모리 DB를 사용하긴 부담스러움. (별도로 사용료 지불해야하기때문)

3) 사용자가 없는 현재 단계에선 데이터베이스로 모든 기능을 처리하는게 부담이 적음.

- spring-session-jdbc 등록 (build.gradle에 의존성 추가)

: 세션 저장소를 데이터베이스로 교체함. 지금은 기존과 동일하게 스프링을 재시작하면 세션이 풀림.

> ➡️ H2 기반으로 스프링이 재실행될 때 H2도 재시작되기 때문. 이후 AWS로 배포하게 되면 AWS의 데이터베이스 서비스인 RDS를 사용하게 되니 이때부터는 세션이 풀리지 않음.

5.6 네이버 로그인

➕ 💾 카카오 로그인 (찬진님 공유 🤓)

5.7 기존 테스트에 시큐리티 적용하기

☑️ 기존 테스트에 시큐리티 적용으로 문제가 되는 부분들을 해결

- 기존에는 바로 API를 호출할 수 있어 테스트 코드 역시 바로 API를 호출하도록 구성하였으나, 시큐리티 옵션이 활성화되면 인증된 사용자만 API를 호출할 수 있음. 기존의 API 테스트 코드들이 모두 인증에 대한 권한을 받지 못하였으므로, 테스트 코드마다 인증한 사용자가 호출한 것처럼 작동하도록 수정.

배운 내용

- 스프링 부트 1.5와 스프링 부트 2.0에서 시큐리티 설정의 차이점
- 스프링 시큐리티를 이용한 구글/네이버 로그인 연동 방법
- ArgumentResolver를 이용하면 어노테이션으로 로그인 세션 정보를 가져올 수 있다는 것
- 세션 저장소로 톰캣/데이터베이스/메모리DB가 있으며 이 중 데이터베이스를 사용하는 이유
- 스프링 시큐리티 적용 시 기존 테스트 코드에서 문제 해결 방법

📢 **지금까지**
: 인텔리제이로 스프링 부트 통합 개발환경을 만들고 테스트와 JPA로 데이터를 처리하고 머스테치로 화면을 구성했으며 시큐리티와 Oauth로 인증과 권한을 배워보며 간단한 게시판을 모두 완성했음 !

앞으로
: AWS를 이용해 나만의 서비스를 직접 배포하고 운영하는 과정을 진행할 예정!