HOL : Spring Security in Spring Boot

Task1. 간단한 인증처리하기
1. Spring Boot Project 생성
  1)Package Explorer > right-click > New > Spring Starter Project
  2)다음의 각 항목의 값을 입력후 Next 클릭
    -Service URL :http://start.spring.io
    -Name : SpringBootSecurityDemo
    -Type : Maven
    -Packaging : Jar
    -Java Version : 8
    -Language : Java
    -Group : com.example
    -Artifact : SpringBootSecurityDemo
    -Version : 0.0.1-SNAPSHOT
    -Description : Demo project for Spring Boot
    -Package : com.example.biz

  3)다음 각 항목을 선택후 Finish 클릭
    -Spring Boot Version : 2.2.6
    -Select
      --Web > Spring Web
      --Developer Tools > Spring Boot DevTools, Lombok
      --Template Engines > Thymeleaf
    -Finish

  4)Project version up
    -Java Build Path > JRE System Library [jdk 1.8.0_251] > Apply
    -Java compiler > JDK Compliance > 1.8 > Apply
    -Project Facets > Java > 1.8
    -Apply and Close

2. Security를 적용하지 않았을 때
  1)com.example.biz.HomeController 생성
  2)HomeController.java

```
    package com.example.biz;

    import org.springframework.stereotype.Controller;
    import org.springframework.web.bind.annotation.GetMapping;

    @Controller
    public class HomeController {
      @GetMapping("/hello")
      public void hello() {}
    }
```

  3)templates/hello.html

```
    <body>
      <h3>안녕하세요. ^^</h3>
    </body>
```

  4)Project 실행
    -http://localhost:8080/hello 요청
    -문제없이 페이지를 잘 볼 수 있다.

5)Spring Boot로 Project를 생성시, Security Starter를 추가하지 않으면 Security 관련 자동 설정이 동작하지 않는다.
6)따라서 사용자는 Web Application이 제공하고 있는 모든 자원에 아무런 제약 없이 접근할 수 있다.

3. Spring Security를 적용했을 때
1)pom.xml에서 Ctrl + Spacebar를 통해 [Edit Starter...]를 선택한다.
2)Security > Spring Security를 check
3)pom.xml에 보면 아래와 같은 dependency가 추가된 것을 알 수 있다.

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-test</artifactId>
  <scope>test</scope>
</dependency>
```

4)Spring Boot Application을 다시 시작한다.
5)localhost:8080/hello 요청시 인증 폼이 보인다.
6)username은 'user', password는 Console 창에 보이는 [Using generated security password] 값이다.
7)Sign in하면 정상적으로 hello.html에 접근하는 것을 알 수 있다.
8)Security Starter를 추가하면 당연히 Security 관련 의존성들이 추가되고 관련되는 자동설정들도 동작한다.
9)현재 이 방식은 Memory에 인증에 필요한 사용자가 자동으로 등록되는 방식이다.

4. Security의 기본 설정 추가하기
1)가장 먼저 생성할 파일은 Spring Security 관련 설정을 위한 환경 설정 Class이다.
2)이 Class는 org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter Abstract class를 상속받는다.
3)com.example.security.SecurityConfig class 생성
4)src/main/java > right-click > New > Package
5)Name : com.example.security
6)Finish
7)com.example.security > right-click > New > Class
8)Name : SecurityConfig
9)Superclass : org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter
10)Finish

11)생성한 SecurityConfig는 어떤 annotation이 적용되지 않은 상태이기 때문에 Bean으로 인식되기 위해 @EnableWebSecurity annotation을 추가한다.

```
package com.example.security;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurer
Adapter;

import lombok.extern.slf4j.Slf4j;
```

```
107
108     @Slf4j
109     @EnableWebSecurity
110     public class SecurityConfig extends WebSecurityConfigurerAdapter{
111       @Override
112       protected void configure(HttpSecurity http) throws Exception{
113         log.info("security config...");
114       }
115     }
116
```

12)@EnableWebSecurity annotation은 이 Class로부터 생성된 객체가 Security 설정 파일임을 의미한다.
13)WebSecurityConfigurerAdapter Abstract class는 Web Security와 관련된 다양한 설정을 추가할 수 있는 configure() method가 있는데, 이 method를 재정의해서 Security 관련 설정을 Customizing 할 수 있다.
14)configure() method는 parameter로 HttpSecurity를 사용하는데, Web Application 자원에 대한 인증과 인가를 제어할 수 있다.

5. SpringBootSecurityDemoApplication.java 수정하기
1)새로 생성한 com.example.security package를 인식하기 위해 @ComponentScan의 scan 범위를 com.example로 조정한다.
2)src/main/java/com.example.biz.SpringBootSecurityDemoApplication.java

```
package com.example.biz;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan(basePackages = "com.example")
public class SpringBootSecurityDemoApplication {

  public static void main(String[] args) {
    SpringApplication.run(SpringBootSecurityDemoApplication.class, args);
  }
}
```

6. log level을 DEBUG로 맞춘다.
1)Spring Boot는 기본적으로 INFO level이상의 log만 출력하도록 구성되어 있기 때문에 더 많은 Security 관련 Log message를 확인하려면 log level을 DEBUG로 변경해야 한다.
2)src/main/resources/application.properties

```
spring.thymeleaf.cache=false
logging.level.org.springframework.web=debug
logging.level.org.springframework.security=debug
```

7. 실행
1)Project > right-click > Run As > Spring Boot App

```
Using generated security password: 0b67a25d-d2f4-490a-b9b3-ab7f0da17896 <--결과값은 다르다.
[  restartedMain] com.example.security.SecurityConfig      : security config...
```

2)문제가 없으면 log.info()처리를 확인할 수 있다.
3)이때 Spring Security가 자동으로 생성한 패스워드를 확인할 수 있다.

4)기본적으로 Spring Security는 하나의 사용자 정보를 가지고 있는데, 사용자 이름은 user, password는 위에 출력결과에서 보이는 값이다.
5)WebSecurityConfigurerAdapter Abstarct class를 상속한 Security 설정 class가 Bean으로 등록되면 더 이상 Application에서는 login을 강제하지 않는다.
6)http://localhost:8080/hello를 요청하면 바로 요청한 page로 이동할 것이다.

8. Security 화면 구성하기
1)실습을 위한 시나리오는 다음과 같다.
-/ : 인증을 하지 않은 모든 사용자가 접근할 수 있다.
-/member : 인증을 통과한 사용자만 접근할 수 있다.
-/manager : 인증을 통과했고, MANAGER 권한을 가진 사용자만 접근할 수 있다.
-/admin : 인증을 통과했고, ADMIN 권한을 가진 사용자만 접근할 수 있다.

2)com.example.security.SecurityController 작성

```
package com.example.security;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Controller
public class SecurityController {
  @GetMapping("/")
  public String index() {
    log.info("called Index page");
    return "index";
  }

  @GetMapping("/member")
  public void forMember() {
    log.info("called Member page");
  }

  @GetMapping("/manager")
  public void forManager() {
    log.info("called Manager page");
  }

  @GetMapping("/admin")
  public void forAdmin() {
    log.info("called Admin page");
  }
}
```

3)templates page 작성
-templates/index.html

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
```

```
      <h1>Index page</h1>
    </body>
    </html>

-templates/member.html

    <!DOCTYPE html>
    <html xmlns:th="http://thymeleaf.org">
    <head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
    </head>
    <body>
      <h1>Member page : Login 성공한 분만 보입니다.</h1>
      <a th:href="@{/loginSuccess}">뒤로 가기</a>
    </body>
    </html>

-templates/manager.html

    <!DOCTYPE html>
        <html xmlns:th="http://thymeleaf.org">
    <head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
    </head>
    <body>
      <h1>Manager page : Manager 권한을 가진 분만 보입니다.</h1>
      <a th:href="@{/loginSuccess}">뒤로 가기</a>
    </body>
    </html>

-templates/admin.html

    <!DOCTYPE html>
    <html xmlns:th="http://thymeleaf.org">
    <head>
    <meta charset="UTF-8">
    <title>Insert title here</title>
    </head>
    <body>
      <h1>Admin page : Admin 권한을 가진 분만 보입니다.</h1>
      <a th:href="@{/loginSuccess}">뒤로 가기</a>
    </body>
    </html>


9. 특정 권한을 가진 사람만이 특정 URI 접근 허용하기
  1)com.example.security.SecurityConfig.java 수정

    package com.example.security;

    import org.springframework.security.config.annotation.web.builders.HttpSecurity;
    import
    org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
    import
    org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurer
    Adapter;
```

```
270
271     import lombok.extern.slf4j.Slf4j;
272
273     @Slf4j
274     @EnableWebSecurity
275     public class SecurityConfig extends WebSecurityConfigurerAdapter {
276       @Override
277       protected void configure(HttpSecurity http) throws Exception{
278         log.info("security config...");
279         http.authorizeRequests().antMatchers("/").permitAll();
280         http.authorizeRequests().antMatchers("/member/**").authenticated();
281         http.authorizeRequests().antMatchers("/manager/**").hasRole("MANAGER");
282         http.authorizeRequests().antMatchers("/admin/**").hasRole("ADMIN");
283       }
284     }
285
286   2)특정한 경로에 특정한 권한을 가진 사용자만 접근하게 하려면 authorizeRequests()와 antMatches()를 이용
    해서 경로를 설정할 수 있다.
287   3)authorizeRequests() method는 Security 처리에서 HttpServletRequest를 이용한다.
288   4)antMatches() method는 특정한 경로를 지정한다.
289   5)permitAll()은 모든 사용자가 접근할 수 있다는 것을 의미한다.
290   6)hasRole()은 System상에서 특정 권한을 가진 사람만이 접근할 수 있다는 것을 의미한다.
291
292
293 10. Test
294   1)localhost:8080/
295     Index page
296
297   2)localhost:8080/manager
298     Whitelabel Error Page
299     This application has no explicit mapping for /error, so you are seeing this as a fallback.
300
301     Tue Apr 28 22:45:38 KST 2020
302     There was an unexpected error (type=Forbidden, status=403).
303     Access Denied      <---권한 거부
304
305   3)위에서 작성한 페이지('/member', '/admin', '/manager')를 실제로 들어가보면 Whitelabel Error
    Page(...status=403) Access Denied의 에러 페이지를 만나게 된다.
306   4)즉, 인증된 회원(login에 성공한 사용자)에게만 특정 page를 보여주기 위해서는 사용자에게 login 화면을 제공해
    야 한다.
307
308
309 11. Login page 처리하기
310   1)SecurityConfig.java 수정
311
312     @Override
313     protected void configure(HttpSecurity http) throws Exception{
314       log.info("security config...");
315       http.authorizeRequests().antMatchers("/").permitAll();
316       http.authorizeRequests().antMatchers("/member/**").authenticated();
317       http.authorizeRequests().antMatchers("/manager/**").hasRole("MANAGER");
318       http.authorizeRequests().antMatchers("/admin/**").hasRole("ADMIN");
319
320       http.formLogin();
321     }
322
323   2)method마지막 줄에 사용한 http.formLogin()은 Form tag 기반의 login을 지원한다는 의미이다.
324   3)이것은 별도로 login page를 제작하지 않아도 login 정보를 입력하는 page를 보여준다.
```

4)http://localhost:8080/member 을 호출하면 인증이 필요하기 때문에 Spring Security가 제공하는 기본 login 화면을 볼 수 있다.
5)username은 'user', password는 console에서 확인한 Using generated security password: 06dfc851-2125-43b8-965c-f3895a7af31f의 값을 입력한다.
6)member page인 다음과 같은 화면을 보게 된다.

Member page : Login 성공한 분만 보입니다.
뒤로 가기

7)하지만 여전히 /manager, /admin은 인증이 되지 않는다.
8)인증이 되기 위해서는 SecurityConfig.java에 AuthenticationManagerBuilder를 주입해서 인증에 대한 처리를 해야 한다.

```
@Override
public void configure(AuthenticationManagerBuilder auth) throws Exception{
  log.info("build Auth global...");

  auth.inMemoryAuthentication().withUser("manager")
  .password("{noop}12345678")
  .roles("MANAGER");

  auth.inMemoryAuthentication().withUser("admin")
  .password("{noop}12345678")
  .roles("ADMIN");
}
```

9)AuthenticationManagerBuilder는 인증에 대한 다양한 설정을 생성할 수 있다.
10)예를 들면 Memory 상의 정보를 이용하거나, JDBC나 LDAP 등의 정보를 이용해서 인증 처리를 할 수 있다.
11)여기서는 Memory상의 정보를 이용하고 있다.
12)localhost:8080/manager를 요청하면 /login으로 이동이 된다.
11)User : manager, password : 12345678을 입력하면 성공적으로 /manager로 이동하여 manager page를 볼 수 있다.

Manager page : Manager 권한을 가진 분만 보입니다.
뒤로 가기

12)또한 localhost:8080/admin을 요청하면 역시 /login으로 이동한다.
13)User : admin, password : 12345678을 입력하면 성공적으로 /admin로 이동하여 admin page를 볼 수 있다.

Admin page : Admin 권한을 가진 분만 보입니다.
뒤로 가기

12. Custom Login Page 만들기

1)SecurtyConfig.java에서 수정한다.

```
@Override
protected void configure(HttpSecurity http) throws Exception{
  log.info("security config...");
  http.authorizeRequests().antMatchers("/").permitAll();
  http.authorizeRequests().antMatchers("/member/**").authenticated();
  http.authorizeRequests().antMatchers("/manager/**").hasRole("MANAGER");
  http.authorizeRequests().antMatchers("/admin/**").hasRole("ADMIN");

  http.formLogin().loginPage("/login");   <--수정
}
```

2)formLogin() 이후에 loginPage()를 이용해서 URI를 지정하면 된다.
3)SecurityController.java 코드 추가

```
@GetMapping("/login")
public void login() {
}
```

4)templates/login.html

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Login Page</title>
</head>
<body>
    <h2>Custom Login Page</h2>
    <div th:if="${param.error ne null}">
        <h2>Invalid Username or Password.</h2>
        <h2 th:value="${param.error}"></h2>
    </div>

    <div th:if="${param.logout ne null}">
        <h2>You have been logged out.</h2>
    </div>

    <form method="post">
        <p>
            <label for="username">Username</label>
            <input type="text" id="username" name="username"/>
        </p>
        <p>
            <label for="password">Password</label>
            <input type="password" name="password" id="password" />
        </p>
        <input type="hidden" th:name="${_csrf.parameterName}"
        th:value="${_csrf.token}" />
        <button type="submit" class="btn">Login</button>
    </form>
</body>
</html>
```

5)form에서 action을 지정하지 않았기 때문에 인증이 마치면 index page로 이동한다.
6)form 내부에는 hidden으로 _csrf 속성이 존재한다.
7)이 속성은 '사이트 간 요청 위조(Cross-site Request Forgery, CSRF, XSRF)'를 방지하기 위한 것이다.
8)이것은 요청을 보내는 URL에서 Server가 가진 동일한 값과 같은 값을 가지고 data를 전송할 때에만 신뢰하기 위한 방법이다.

13. Test

1)localhost:8080/admin으로 요청하면 /login으로 이동하게 되고 Custom Login page를 보여준다.
2)만일 잘못된 id와 password를 입력하면 Invalid Username or Password.가 보여진다.
3)제대로 입력하면 admin page로 이동한다.

14. 접근 권한 없음 페이지 처리하기

1)접근 권한이 없는 사용자가 특정 페이지를 요청하면 error(403)가 발생한다.
2)접근 권한이 없는 특정 경로로 접근하려 할 때 사용자에게 권한이 없음을 알려주고, login 화면으로 이동할 수 있도록 안내 페이지를 만든다.
3)이 설정은 HttpSecurity에서 exceptionHandling()을 이용해서 지정한다.

```
@Override
protected void configure(HttpSecurity http) throws Exception{
  log.info("security config...");
  http.authorizeRequests().antMatchers("/").permitAll();
  http.authorizeRequests().antMatchers("/member/**").authenticated();
  http.authorizeRequests().antMatchers("/manager/**").hasRole("MANAGER");
  http.authorizeRequests().antMatchers("/admin/**").hasRole("ADMIN");

  http.formLogin().loginPage("/login");
  http.exceptionHandling().accessDeniedPage("/accessDenied");
}
```

4)SecurityController.java 코드 추가

```
@GetMapping("/accessDenied")
public void accessDenied() {
  log.info("called accessDenied page");
}
```

5)templates/accessDenied.html
-static/css/bootstrap-theme.min.css
-static/css/bootstrap.min.css

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://thymeleaf.org">
<head>
   <meta charset="UTF-8" />
   <meta name="viewport" content="width=device-width, initial-scale=1.0" />
   <title>Document</title>
   <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />
   <link rel="stylesheet" th:href="@{/css/bootstrap-theme.min.css}" />
</head>
<body>
   <div class="panel panel-default">
      <div class="panel-body">
         <div class="alert alert-danger" role="alert">
            <h2>Sorry. You do not have permission to view this page.</h2>
            Click Login at <a th:href="@{/login}">here</a>
         </div>
      </div>
   </div>
</body>
</html>
```

15. Test

1)localhost:8080/admin으로 요청한다.
2)manager/12345678을 입력하면 ADMIN권한이 없기 때문에 accessDenied.html로 이동한다.

16. login 성공시 이동할 page 만들기

1)SecurityConfig.java 수정

```
491     @Override
492     protected void configure(HttpSecurity http) throws Exception{
493       log.info("security config...");
494       http.authorizeRequests().antMatchers("/").permitAll();
495       http.authorizeRequests().antMatchers("/member/**").authenticated();
496       http.authorizeRequests().antMatchers("/manager/**").hasRole("MANAGER");
497       http.authorizeRequests().antMatchers("/admin/**").hasRole("ADMIN");
498
499       http.formLogin().loginPage("/login").defaultSuccessUrl("/loginSuccess", true);
500       http.exceptionHandling().accessDeniedPage("/accessDenied");
501     }
502
503   2)defaultSuccessUrl() method는 login에 성공했을 때 이동할 URL을 지정한다.
504
505   3)SecurityController.java 코드 추가
506
507     @GetMapping("/loginSuccess")
508       public void loginSuccess() {
509         log.info("called loginSuccess page");
510       }
511
512   4)templates/loginSuccess.html
513
514     <!DOCTYPE html>
515     <html xmlns:th="http://www.thymeleaf.org">
516     <head>
517     <meta charset="UTF-8">
518     <title>Insert title here</title>
519     </head>
520     <body>
521       <h3><span style="color:red">로그인 인증 성공</span></h3>
522       <h5><a th:href="@{/}">Index Page로 이동</a></h5>
523       <h5><a th:href="@{/member}">Member Page로 이동</a></h5>
524       <h5><a th:href="@{/manager}">Manager 전용 Page로 이동</a></h5>
525       <h5><a th:href="@{/admin}">Admin 전용 Page로 이동</a></h5>
526     </body>
527     </html>
528
529
530 17. Logout 처리하기
531   1)Spring Security가 Web을 처리하는 방식의 기본은 HttpSession이기 때문에 Browser를 닫으면 login한 정
      보를 잃는다.
532   2)Browser가 종료되지 않으면 logout을 수행해서 사용자가 login했던 모든 정보를 삭제해야 한다.
533   3)HttpSession의 정보를 무력화시키고, 모든 Cookie를 삭제해야 한다.
534   4)SecuryConfig.java 수정
535
536     @Override
537     protected void configure(HttpSecurity http) throws Exception{
538       log.info("security config...");
539       http.authorizeRequests().antMatchers("/").permitAll();
540       http.authorizeRequests().antMatchers("/member/**").authenticated();
541       http.authorizeRequests().antMatchers("/manager/**").hasRole("MANAGER");
542       http.authorizeRequests().antMatchers("/admin/**").hasRole("ADMIN");
543
544       http.formLogin().loginPage("/login").defaultSuccessUrl("/loginSuccess", true);
545       http.exceptionHandling().accessDeniedPage("/accessDenied");
546       http.logout().logoutUrl("/logout").invalidateHttpSession(true).logoutSuccessUrl("/login"
          );
```

```
    }

  5)logout을 특정한 page에서 수행하기
  6)SecurityController.java 코드 추가

    @GetMapping("/logout")
    public void logout() {
      log.info("called Logout page");
    }

  7)templates/logout.html

    <!DOCTYPE html>
    <html lang="en" xmlns:th="http://www.thymeleaf.org">
    <head>
      <meta charset="UTF-8" />
      <meta name="viewport" content="width=device-width, initial-scale=1.0" />
      <title>Document</title>
      <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />
      <link rel="stylesheet" th:href="@{/css/bootstrap-theme.min.css}" />
    </head>
    <body>
      <h2>Custom Logout Page</h2>
      <form method="POST">
        <h3>Logout</h3>
        <input type="hidden" th:name="${_csrf.parameterName}"
        th:value="${_csrf.token}" />
        <button type="submit" class="btn btn-primary">Logout</button>
      </form>
    </body>
    </html>

  8)templates/manager.html 수정

    <body>
      <h1>Manager page : Manager 권한을 가진 분만 보입니다.</h1>
      <a th:href="@{/loginSuccess}">뒤로 가기</a>
      <form action="logout" method="get">
        <input type="submit" value="Logout" />
      </form>
    </body>

  9)templates/admin.html 수정

    <body>
      <h1>Admin page : Admin 권한을 가진 분만 보입니다.</h1>
        <a th:href="@{/loginSuccess}">뒤로 가기</a>
        <form action="logout" method="get">
          <input type="submit" value="Logout" />
        </form>
    </body>


18. Test
  1)http://localhost:8080/admin
  2)admin/12345678 -> loginSuccess page로 이동
  3)click [Admin 전용 페이지로 이동]
  4)Admin page에서 [logout] click
```

5)/logout? page로 이동
6)[Logout] button click
7)/login page로 이동 확인

Task2. Database 인증하기

1. Database 준비
1)Member Table 생성

```
DROP TABLE Member;

CREATE TABLE Member
(
    userid   VARCHAR2(20) PRIMARY KEY,
    passwd   VARCHAR2(100) NOT NULL,
    name     VARCHAR2(20) NOT NULL,
    role     VARCHAR2(45) NOT NULL,
    enabled  NUMBER(1) DEFAULT 1
);

INSERT INTO Member VALUES('member', '12345678', '회원', 'ROLE_MEMBER', 1);
INSERT INTO Member VALUES('manager', '12345678', '매니저', 'ROLE_MANAGER', 1);
INSERT INTO Member VALUES('admin', '12345678', '어드민', 'ROLE_ADMIN', 1);
COMMIT;
```

2)password column의 길이가 유난히 길게 잡혀있는 이유는 이후에 설정할 비밀번호에 대한 암호화 처리 때문이다.
3)enabled column은 해당 계정을 활성화여부에 대한 컬럼이다.

2. Spring Boot Project 생성
1)Package Explorer > right-click > New > Spring Starter Project
2)다음의 각 항목의 값을 입력후 Next 클릭
-Service URL :http://start.spring.io
-Name : SpringBootSecurityDemo1
-Type : Maven
-Packaging : Jar
-Java Version : 8
-Language : Java
-Group : com.example
-Artifact : SpringBootSecurityDemo1
-Version : 0.0.1-SNAPSHOT
-Description : Demo project for Spring Boot
-Package : com.example.biz

3)다음 각 항목을 선택후 Finish 클릭
-Spring Boot Version : 2.2.6
-Select
--Developer Tools > Spring Boot DevTools, Lombok, Spring Configuration Processor
--SQL > Oracle Driver, JDBC API
--Security > Spring Security
--Template Engines > Thymeleaf
--Web > Spring Web
-Finish

4)Project version up
-Java Build Path > JRE System Library [jdk 1.8.0_251] > Apply

-Java compiler > JDK Compliance > 1.8 > Apply
-Project Facets > Java > 1.8
-Apply and Close

3. src/main/resources/application.properties

```
spring.thymeleaf.cache=false

logging.level.org.springframework.web=debug
logging.level.org.springframework.security=debug

spring.datasource.url=jdbc:oracle:thin:@localhost:1521:XE
spring.datasource.driver-class-name=oracle.jdbc.driver.OracleDriver
spring.datasource.username=membership
spring.datasource.password=membership
```

4. com.example.security.SecurityConfig.java 생성하기
1)Database 인증을 하기 위해 SecurityConfig.java를 생성한다.
2)src/main/java > right-click > New > Package
3)Name : com.example.security
4)Finish
5)com.example.security > right-click > New > Class
6)Name : SecurityConfig
7)Superclass : org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter
8)Finish

```
package com.example.security;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

import com.example.service.MemberService;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
  @Autowired
  private MemberService memberService;

  @Override
  protected void configure(HttpSecurity http) throws Exception{
    log.info("security config...");
    http.authorizeRequests().antMatchers("/").permitAll();
    http.authorizeRequests().antMatchers("/member/**").authenticated();
    http.authorizeRequests().antMatchers("/manager/**").hasRole("MANAGER");
    http.authorizeRequests().antMatchers("/admin/**").hasRole("ADMIN");

```

```
715             http.formLogin().loginPage("/login").defaultSuccessUrl("/loginSuccess", true);
716             http.exceptionHandling().accessDeniedPage("/accessDenied");
717             http.logout().logoutUrl("/logout").invalidateHttpSession(true).logoutSuccessUrl("/logi
                n");
718
719             http.userDetailsService(memberService);
720           }
721         }
722
723       9)Autowired로 DataSource를 선언하고, 의존성 주입한다.
724
725
726   5. SpringBootSecurityDemo1Application.java 수정하기
727       1)@ComponentScan의 scan 범위를 com.example로 조정한다.
728
729         package com.example.biz;
730
731         import org.springframework.boot.SpringApplication;
732         import org.springframework.boot.autoconfigure.SpringBootApplication;
733         import org.springframework.context.annotation.ComponentScan;
734
735         @SpringBootApplication
736         @ComponentScan(basePackages = "com.example")
737         public class SpringBootSecurityDemo1Application {
738
739           public static void main(String[] args) {
740             SpringApplication.run(SpringBootSecurityDemo1Application.class, args);
741           }
742         }
743
744
745   6. Security 화면 구성하기
746       1)실습을 위한 시나리오는 다음과 같다.
747         -/ : 인증을 하지 않은 모든 사용자가 접근할 수 있다.
748         -/member : 인증을 통과한 사용자만 접근할 수 있다.
749         -/manager : 인증을 통과했고, MANAGER 권한을 가진 사용자만 접근할 수 있다.
750         -/admin : 인증을 통과했고, ADMIN 권한을 가진 사용자만 접근할 수 있다.
751
752
753   7. Controller 작성
754       1)com.example.security.SecurityController 작성
755       2)com.example.security > right-click > New > Class
756       3)Name : SecurityController
757       4)Finish
758
759         package com.example.security;
760
761         import org.springframework.stereotype.Controller;
762         import org.springframework.web.bind.annotation.GetMapping;
763
764         import lombok.extern.slf4j.Slf4j;
765
766         @Slf4j
767         @Controller
768         public class SecurityController {
769           @GetMapping("/")
770           public String index() {
771             log.info("called Index page");
```

```
772         return "index";
773       }
774
775       @GetMapping("/member")
776       public void forMember() {
777         log.info("called Member page");
778       }
779
780       @GetMapping("/manager")
781       public void forManager() {
782         log.info("called Manager page");
783       }
784
785       @GetMapping("/admin")
786       public void forAdmin() {
787         log.info("called Admin page");
788       }
789
790       @GetMapping("/login")
791       public void login() {
792         log.info("called Login page");
793       }
794
795       @GetMapping("/accessDenied")
796       public void accessDenied() {
797       }
798
799       @GetMapping("/loginSuccess")
800       public void loginSuccess() {}
801
802       @GetMapping("/logout")
803       public void logout() {
804         log.info("called Logout page");
805       }
806     }
807
808
809 8. templates page 작성
810   1)templates/index.html
811
812     <!DOCTYPE html>
813     <html>
814     <head>
815     <meta charset="UTF-8">
816     <title>Insert title here</title>
817     </head>
818     <body>
819       <h1>Index page</h1>
820     </body>
821     </html>
822
823   2)templates/member.html
824
825     <!DOCTYPE html>
826     <html xmlns:th="http://thymeleaf.org">
827     <head>
828     <meta charset="UTF-8">
829     <title>Insert title here</title>
```

```
  </head>
  <body>
    <h1>Member page : Login 성공한 분만 보입니다.</h1>
    <a th:href="@{/loginSuccess}">뒤로 가기</a>
    <form action="logout" method="get">
      <input type="submit" value="Logout" />
    </form>
  </body>
  </html>
```

3)templates/manager.html

```
  <!DOCTYPE html>
  <html xmlns:th="http://thymeleaf.org">
  <head>
  <meta charset="UTF-8">
  <title>Insert title here</title>
  </head>
  <body>
    <h1>Manager page : Manager 권한을 가진 분만 보입니다.</h1>
    <a th:href="@{/loginSuccess}">뒤로 가기</a>
    <form action="logout" method="get">
      <input type="submit" value="Logout" />
    </form>
  </body>
  </html>
```

4)templates/admin.html

```
  <!DOCTYPE html>
  <html xmlns:th="http://thymeleaf.org">
  <head>
  <meta charset="UTF-8">
  <title>Insert title here</title>
  </head>
  <body>
    <h1>Admin page : Admin 권한을 가진 분만 보입니다.</h1>
    <a th:href="@{/loginSuccess}">뒤로 가기</a>
    <form action="logout" method="get">
      <input type="submit" value="Logout" />
    </form>
  </body>
  </html>
```

5)templates/login.html

```
  <!DOCTYPE html>
  <html lang="en" xmlns:th="http://www.thymeleaf.org">
  <head>
     <meta charset="UTF-8" />
     <meta name="viewport" content="width=device-width, initial-scale=1.0" />
     <title>Login Page</title>
  </head>
  <body>
     <h2>Custom Login Page</h2>
     <div th:if="${param.error ne null}">
        <h2>Invalid Username or Password.</h2>
        <h2 th:value="${param.error}"></h2>
```

```
888         </div>
889
890         <div th:if="${param.logout ne null}">
891            <h2>You have been logged out.</h2>
892         </div>
893
894         <form method="post">
895            <p>
896               <label for="username">Username</label>
897               <input type="text" id="username" name="username"/>
898            </p>
899            <p>
900               <label for="password">Password</label>
901               <input type="password" name="password" id="password" />
902            </p>
903            <input type="hidden" th:name="${_csrf.parameterName}"
               th:value="${_csrf.token}" />
904            <button type="submit" class="btn">Login</button>
905         </form>
906      </body>
907      </html>
908
909   6)templates/accessDenied.html
910      -static/css/bootstrap-theme.min.css
911      -static/css/bootstrap.min.css
912
913      <!DOCTYPE html>
914      <html lang="en" xmlns:th="http://thymeleaf.org">
915      <head>
916         <meta charset="UTF-8" />
917         <meta name="viewport" content="width=device-width, initial-scale=1.0" />
918         <title>Document</title>
919         <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />
920         <link rel="stylesheet" th:href="@{/css/bootstrap-theme.min.css}" />
921      </head>
922      <body>
923         <div class="panel panel-default">
924            <div class="panel-body">
925               <div class="alert alert-danger" role="alert">
926                  <h2>Sorry. You do not have permission to view this page.</h2>
927                  Click Login at <a th:href="@{/login}">here</a>
928               </div>
929            </div>
930         </div>
931      </body>
932      </html>
933
934   7)templates/loginSuccess.html
935
936      <!DOCTYPE html>
937      <html xmlns:th="http://www.thymeleaf.org">
938      <head>
939      <meta charset="UTF-8">
940      <title>Insert title here</title>
941      </head>
942      <body>
943        <h3><span style="color:red">로그인 인증 성공</span></h3>
944        <h5><a th:href="@{/}">Index Page로 이동</a></h5>
```

```
945         <h5><a th:href="@{/member}">Member Page로 이동</a></h5>
946         <h5><a th:href="@{/manager}">Manager 전용 Page로 이동</a></h5>
947         <h5><a th:href="@{/admin}">Admin 전용 Page로 이동</a></h5>
948       </body>
949       </html>
950
951     8)templates/logout.html
952
953       <!DOCTYPE html>
954       <html lang="en" xmlns:th="http://www.thymeleaf.org">
955       <head>
956         <meta charset="UTF-8" />
957         <meta name="viewport" content="width=device-width, initial-scale=1.0" />
958         <title>Document</title>
959         <link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />
960         <link rel="stylesheet" th:href="@{/css/bootstrap-theme.min.css}" />
961       </head>
962       <body>
963         <h2>Custom Logout Page</h2>
964         <form method="POST">
965           <h3>Logout</h3>
966           <input type="hidden" th:name="${_csrf.parameterName}"
              th:value="${_csrf.token}" />
967           <button type="submit" class="btn btn-primary">Logout</button>
968         </form>
969       </body>
970       </html>
971
972
973   9. com.example.vo.MemberVO.java 생성
974     1)src/main/java > right-click > New > Package
975     2)Name : com.example.vo
976     3)com.example.vo > right-click > New > Class
977     4)Name : MemberVO
978
979       package com.example.vo;
980
981       import lombok.AllArgsConstructor;
982       import lombok.Getter;
983       import lombok.NoArgsConstructor;
984       import lombok.Setter;
985       import lombok.ToString;
986
987       @Setter
988       @Getter
989       @ToString
990       @AllArgsConstructor
991       @NoArgsConstructor
992       public class MemberVO {
993         private String userid;
994         private String passwd;
995         private String name;
996         private String role;
997         private int enabled;
998       }
999
1000
1001  10. com.example.dao.MemberDao.java 생성
```

1)src/main/java > right-click > New > Package
2)Package name : com.example.dao
3)com.example.dao > right-click > New > Class
4)Name : MemberDao

```
package com.example.dao;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;

import com.example.vo.MemberVO;

@Repository
public class MemberDao {
  @Autowired
  private JdbcTemplate jdbcTemplate;

  public MemberVO getUserByID(String username) {
    String sql = "SELECT userid, passwd, name, role, enabled FROM Member WHERE
    userid=?";
    return this.jdbcTemplate.queryForObject(sql, new String[] {username}, new
    RowMapper<MemberVO>() {
      @Override
      public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
        MemberVO member = new MemberVO();
        member.setUserid(rs.getString("userid"));
        member.setPasswd("{noop}" + rs.getString("passwd"));
        member.setName(rs.getString("name"));
        member.setRole(rs.getString("role"));
        member.setEnabled(rs.getInt("enabled"));
        return member;
      }
    });
  }
}
```

11. com.example.service.MemberService.java 생성하기
1)UserDetailService 구현하기
-Spring security에서는 사용자에 대한 정보를 UserDetails와 UserDetailsService interface로 참조한다.
-Database에 연동해서 사용자 정보를 가져올 수 있는 Service Class가 있으면 UserDetailService interface를 상속받아서 loadUserByUsername() 을 구현하면 된다.

2)src/main/java > right-click > New > Package
3)Package name : com.example.service
4)com.example.service > right-click > New > Class
5)Name : MyUserService
6)Interfaces : org.springframework.security.core.userdetails.UserDetails

```
package com.example.service;

import java.util.ArrayList;
import java.util.Collection;
```

```
1057
1058      import org.springframework.beans.factory.annotation.Autowired;
1059      import org.springframework.security.core.authority.SimpleGrantedAuthority;
1060      import org.springframework.security.core.userdetails.User;
1061      import org.springframework.security.core.userdetails.UserDetails;
1062      import org.springframework.security.core.userdetails.UserDetailsService;
1063      import org.springframework.security.core.userdetails.UsernameNotFoundException;
1064      import org.springframework.stereotype.Service;
1065
1066      import com.example.dao.MemberDao;
1067      import com.example.vo.MemberVO;
1068
1069      import lombok.extern.slf4j.Slf4j;
1070
1071      @Slf4j
1072      @Service
1073      public class MemberService implements UserDetailsService {
1074        @Autowired
1075        private MemberDao memberDao;
1076
1077        @Override
1078        public UserDetails loadUserByUsername(String username) throws
          UsernameNotFoundException {
1079          log.info("called loadUserByUsername() user id {}", username);
1080          MemberVO member = this.memberDao.getUserByID(username);
1081          log.debug("memer = {}", member.toString());
1082
1083          Collection<SimpleGrantedAuthority> roles = new
            ArrayList<SimpleGrantedAuthority>();
1084          roles.add(new SimpleGrantedAuthority(member.getRole()));
1085          log.debug("Role : {}", roles.toString());
1086          UserDetails user = new User(username, member.getPasswd(), roles);
1087          return user;
1088        }
1089      }
1090
1091
1092  12. Test
1093    1)Project > right-click > Run As > Spring Boot App
1094    2)http://localhost:8080/ -> Index page
1095    3)http://localhost:8080/member --> http://localhost:8080/login
1096    4)Username : member, Password : 12345678 -> Login 성공 페이지
1097    5)Member page로 이동 click --> 정상적으로 Member page로 이동됨.
1098    6)Click Logout --> Logout 처리 --> Login page
1099    7)Username : admin, Password : 12345678 -> Login 성공 페이지
1100    8)Admin 전용 Page로 이동 click --> 정상적으로 Admin page로 이동됨.
1101    9)뒤로 가기 click
1102    10)Login 성공 페이지에서 Manager 전용 Page로 이동 click --> Access denined
1103
1104
1105  ----------------------------------------------------------
1106  Task3. PasswordEncoder 사용하기
1107  1. 비밀번호 암호화
1108    1)이제까지 우리는 사용자 정보를 조회할 때 비밀번호를 암호화하지 않고 사용하기 위해 비밀번호 앞에 "{noop}"이라
        는 접두사를 붙여서 사용해왔다.
1109    2)비밀번호를 암호화하지 않고 사용하면 비밀번호 노출시 심각한 문제(법적으로도 개인정보 보호법 위배)가 발생할 수
        있다.
1110    3)사용자가 회원 가입할 때 사용자가 입력한 비밀번호는 반드시 암호화해서 다른 사람들이 알아볼 수 없도록 해야 한다.
```

4)Spring Security에서는 패스워드를 쉽게 암호화할 수 있도록 PasswordEncoder라는 interface를 구현한 class들을 제공한다.
5)따라서 몇 가지 암호화 관련 설정만 추가하면 사용자가 입력한 비밀번호를 암호화하여 처리할 수 있다.
6)Spring Security에서는 BCryptPasswordEncoder를 사용하는데 비밀번호 암호화에 특화되어 있으면서 가장 안전한 Hash Algorithm인 BCrypt를 사용한다.

2. Member Table에서 기존의 모든 Member 삭제하기
-TRUNCATE TABLE Member;

3. 암호화 적용하기
1)SecurityConfig.java에 단지 BCryptPasswordEncoder 객체를 return하는 passwordEncoder() method만 추가한다.

```
import org.springframework.security.crypto.factory.PasswordEncoderFactories;
...
...
@Bean
public PasswordEncoder passwordEncoder() {
  return PasswordEncoderFactories.createDelegatingPasswordEncoder();
}
```

2)MemberDao.java 코드 추가

```
public int insertMember(MemberVO member) {
  String sql = "INSERT INTO Member(userid, passwd, name, role, enabled)
  VALUES(?,?,?,?,?)";
  return this.jdbcTemplate.update(sql, member.getUserid(), member.getPasswd(),
  member.getName(), member.getRole(), member.getEnabled());
}
```

4. Tester class 작성
1)이제 PasswordEncoder를 이용하여 회원가입 기능을 테스트하도록 PasswordEncoderTest class를 작성한다.
2)src/test/java > right-click > New > JUnit Test Case
3)Select New JUnit Jupiter test
4)Name : PasswordEncoderTest
5)Finish

```
package com.example.biz;

import static org.junit.jupiter.api.Assertions.assertEquals;

import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.test.context.junit.jupiter.SpringExtension;

import com.example.dao.MemberDao;
import com.example.vo.MemberVO;

@SpringBootTest
@ExtendWith(SpringExtension.class)
class PasswordEncoderTest {
```

```
1163        @Autowired
1164        private MemberDao memberDao;
1165
1166        @Autowired
1167        private PasswordEncoder encoder;
1168
1169        @Test
1170        void test1() {
1171          MemberVO member = new MemberVO();
1172          member.setUserid("member");
1173          member.setPasswd(encoder.encode("12345678"));
1174          member.setName("정회원");
1175          member.setRole("ROLE_MEMBER");
1176          member.setEnabled(1);
1177          int row = this.memberDao.insertMember(member);
1178          assertEquals(1, row);
1179        }
1180
1181        @Test
1182        void test2() {
1183          MemberVO member = new MemberVO();
1184          member.setUserid("manager");
1185          member.setPasswd(encoder.encode("12345678"));
1186          member.setName("매니저");
1187          member.setRole("ROLE_MANAGER");
1188          member.setEnabled(1);
1189          int row = this.memberDao.insertMember(member);
1190          assertEquals(1, row);
1191        }
1192
1193        @Test
1194        void test3() {
1195          MemberVO member = new MemberVO();
1196          member.setUserid("admin");
1197          member.setPasswd(encoder.encode("12345678"));
1198          member.setName("어드민");
1199          member.setRole("ROLE_ADMIN");
1200          member.setEnabled(1);
1201          int row = this.memberDao.insertMember(member);
1202          assertEquals(1, row);
1203        }
1204      }
1205
1206    6)실행
1207      -right-click > Run As > JUnit Test
1208      -Green Bar
1209
1210    7)MemberDao.java에서 "{noop}" 제거
1211
1212      public MemberVO getUserByID(String username) {
1213        String sql = "SELECT userid, passwd, name, role, enabled FROM Member WHERE
          userid=?";
1214        return this.jdbcTemplate.queryForObject(sql, new String[] { username }, new
          RowMapper<MemberVO>() {
1215          @Override
1216          public MemberVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1217            MemberVO member = new MemberVO();
1218            member.setUserid(rs.getString("userid"));
```

```
            member.setPasswd(rs.getString("passwd"));   <---여기 수정
            member.setName(rs.getString("name"));
            member.setRole(rs.getString("role"));
            member.setEnabled(rs.getInt("enabled"));
            return member;
          }
        });
      }
```

8)각 계정별로 Test

5. JDBC를 이용한 더 간단한 인증처리
1)이 작업은 JdbcUserDetailsManagerConfigurer이라는 객체를 이용하여 보다 쉽게 처리할 수 있다.
2)Spring Security가 Database를 연동하는 방법은 아래의 2가지를 사용한다.
-직접 SQL 등을 지정해서 처리하는 방법
-기존에 작성된 Repository나 Service를 이용해서 처리하는 방법

3)우리는 위에서 사용했던 방법은 2번째 방법이고, 이번에는 첫 번째 방법을 사용해 본다.
4)즉, 이 방법은 DataSource와 SQL을 이용하는 방법니다.
5)SecurityConfig.java를 다음과 같이 수정한다.

```
package com.example.security;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.context.annotation.Bean;
import
org.springframework.security.config.annotation.authentication.builders.AuthenticationMan
agerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurer
Adapter;
import org.springframework.security.crypto.factory.PasswordEncoderFactories;
import org.springframework.security.crypto.password.PasswordEncoder;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter {
  @Autowired
  private DataSource dataSource;        <--추가
  //private MemberService memberService;

  @Override
  protected void configure(HttpSecurity http) throws Exception {
    log.info("security config...");
    http.authorizeRequests().antMatchers("/").permitAll();
    http.authorizeRequests().antMatchers("/member/**").authenticated();
    http.authorizeRequests().antMatchers("/manager/**").hasRole("MANAGER");
    http.authorizeRequests().antMatchers("/admin/**").hasRole("ADMIN");

    http.formLogin().loginPage("/login").defaultSuccessUrl("/loginSuccess", true);
```

```
1272          http.exceptionHandling().accessDeniedPage("/accessDenied");
1273          http.logout().logoutUrl("/logout").invalidateHttpSession(true).logoutSuccessUrl("/logi
              n");
1274
1275          //http.userDetailsService(memberService);    <---주석 처리
1276        }
1277
1278        @Bean
1279        public PasswordEncoder passwordEncoder() {
1280          return PasswordEncoderFactories.createDelegatingPasswordEncoder();
1281        }
1282
1283        @Override
1284        protected void configure(AuthenticationManagerBuilder auth) throws
          java.lang.Exception{
1285          String query1 = "SELECT userid username, passwd password, enabled FROM Member
              WHERE userid = ?";
1286          String query2 = "SELECT userid, role FROM Member WHERE userid = ?";
1287          auth.jdbcAuthentication().dataSource(dataSource).usersByUsernameQuery(query1).a
              uthoritiesByUsernameQuery(query2);
1288        }
1289      }
1290
1291    6)사용자에 대한 계정 정보와 권한을 체크하는 부분은 DataSource를 이용하고, SQL을 직접 지정해서 작성한다.
1292    7)이때 SQL은
1293      -query1 : 사용자의 계정 정보를 이용해서 필요한 정보를 가져오는 SQL
1294      -query2 : 해당 사용자의 권한을 확인하는 SQL
1295
1296    8)auth.jdbcAuthentication() method는 JdbcUserDetailsManagerConfigurer 객체를 반환한다.
1297    9)이 객체를 이용해서 DataSource를 주입하고, 필요한 SQL문을 parameter로 전달하는 방식을 이용하면 간단한
        몇 가지 설정만으로도 인증 매니저를 생성하게 된다.
1298    10)이때, usersByUsernameQuery()와 authoritiesByUsernameQuery()는 username을 이용해서 특정
        한 인증 주체(사용자)정보를 세팅하고, username을 이용해서 권한에 대한 정보를 조회한다.
1299    11)Spring Security에서는 내부적으로 username과 password를 사용하기 때문에 table 설계시 username
        을 userid로, password를 passwd로 변경해서 사용한다.
1300    12)usersByUsernameQuery()를 이용하는 경우에는 username과 password, enabled라는 컬럼의 데이터
        가 필요하다.
1301    13)실제 테이블의 칼럼과 다를 경우에는 alias를 사용한다.
1302    14)이 3가지 데이터를 성공적으로 가져온 후에는 authoritiesByUsernameQuery() method를 사용해서 실제
        권한에 대한 정보를 가져오는 SQL이다.
1303    15)실제로 /member로 요청하면 아래의 log를 확인할 수 있다.
1304
1305      DEBUG 21136 --- [io-8080-exec-10] o.s.s.w.u.matcher.AntPathRequestMatcher  :
          Checking match of request : '/member'; against '/member/**'
1306      DEBUG 21136 --- [io-8080-exec-10] o.s.s.w.a.i.FilterSecurityInterceptor    : Secure
          object: FilterInvocation: URL: /member; Attributes: [authenticated]
1307      DEBUG 21136 --- [io-8080-exec-10] o.s.s.w.a.i.FilterSecurityInterceptor    : Previously
          Authenticated:
          org.springframework.security.authentication.UsernamePasswordAuthenticationToken@8f7
          4b9: Principal: org.springframework.security.core.userdetails.User@bfc28a9a: Username:
          member; Password: [PROTECTED]; Enabled: true; AccountNonExpired: true;
          credentialsNonExpired: true; AccountNonLocked: true; Granted Authorities:
          ROLE_MEMBER; Credentials: [PROTECTED]; Authenticated: true; Details:
          org.springframework.security.web.authentication.WebAuthenticationDetails@b364:
          RemoteIpAddress: 0:0:0:0:0:0:0:1; SessionId: A97803E21013A8A3CE8E0801B70A7D09;
          Granted Authorities: ROLE_MEMBER
1308      DEBUG 21136 --- [io-8080-exec-10] o.s.s.access.vote.AffirmativeBased       : Voter:
          org.springframework.security.web.access.expression.WebExpressionVoter@2eb504d5,
```

returned: 1
DEBUG 21136 --- [io-8080-exec-10] o.s.s.w.a.i.FilterSecurityInterceptor : Authorization successful

6. Thymeleaf에서 Spring Security 처리하기
-ref : https://velog.io/tags/spring-security

1)Thymeleaf에서 작성된 page에서는 다음과 같은 순서로 Spring Security를 이용할 수 있다.
-Security 관련 namespace 추가
-Namespace를 이용한 tag 작성

2)Thymelaf Extras Springsecurity5 추가하기
-mvnrepository 사이트에서 'thymeleaf-extras-springsecurity5'로 검색
-Thymelaf Extras Springsecurity5 click
-3.0.4.RELEASE click
-다음의 dependency를 복사하여 pom.xml에 붙여넣기

```
<dependency>
  <groupId>org.thymeleaf.extras</groupId>
  <artifactId>thymeleaf-extras-springsecurity5</artifactId>
  <version>3.0.4.RELEASE</version>
</dependency>
```

3)pom.xml > right-click > Run As > Maven Install

4)templates/loginSuccess.html 수정하기

```
<!DOCTYPE html>
<html xmlns:th="http://thymeleaf.org"
xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
<meta charset="UTF-8" />
<title>Insert title here</title>
</head>
<body>
  <h3><span style="color:red">로그인 인증 성공</span></h3>
  <h5><a th:href="@{/}">Index Page로 이동</a></h5>
  <h5 sec:authorize="isAuthenticated()"><a th:href="@{/member}">Member Page로 이동</a></h5>
  <h5 sec:authorize="hasRole('ROLE_MANAGER')"><a
  th:href="@{/manager}">Manager 전용 Page로 이동</a></h5>
  <h5 sec:authorize="hasRole('ROLE_ADMIN')"><a th:href="@{/admin}">Admin 전용
  Page로 이동</a></h5>
</body>
</html>
```

5)Thymeleaf에서 Spring Security를 사용하기위해 "sec: " 함수를 사용한다.

6)isAuthenticated( )
-Login한 사용자에게만 보인다.

7)hasRole('ROLE_ADMIN')
-ADMIN 권한을 가진 사용자에게만 보인다.

8)hasAnyRole('ROLE_ADMIN','ROLE_MANAGER')
-ADMIN이나 MANAGER 권한을 가진 사용자에게만 보인다.

9)templates/admin.html

```
<!DOCTYPE html>
<html xmlns:th="http://thymeleaf.org"
xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
<meta charset="UTF-8" />
<title>Insert title here</title>
</head>
<body>
  <h1>Admin page : Admin 권한을 가진 분만 보입니다.</h1>
  <ul>
    <li>User name : <span sec:authentication="name"></span></li>
    <li>Authorities : <span sec:authentication="principal.authorities"></span></li>
  </ul>
  <a th:href="@{/loginSuccess}">뒤로 가기</a>
  <form th:action="@{/logout}" method="get">
    <button sec:authorize="isAuthenticated()">Logout</button>
    <button sec:authorize="isAnonymous()" th:href="@{/login}">Login</button>
  </form>
</body>
</html>
```

10)isAnonymous( )
-Login하지 않은 사용자에게만 보인다.

11)"sec:authentication"은 Login한 사용자만 사용할 수 있는 반면, "sec:authorize"와 달리 만약 Login하지 않은 사용자가 접근할 경우 Error가 발생한다.

12)name은 User의 username을 return한다.
13)principal.authorities는 User의 전체 authorities를 return한다.
14)principal 객체를 사용하면 authorities 이외의 다른 User 정보도 사용할 수 있다.

15)templates/manager.html

```
<!DOCTYPE html>
<html xmlns:th="http://thymeleaf.org"
xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
<meta charset="UTF-8" />
<title>Insert title here</title>
</head>
<body>
  <h1>Manager page : Manager 권한을 가진 분만 보입니다.</h1>
  <ul>
    <li>User name : <span sec:authentication="name"></span></li>
    <li>Authorities : <span sec:authentication="principal.authorities"></span></li>
  </ul>
  <a th:href="@{/loginSuccess}">뒤로 가기</a>
  <form th:action="@{/logout}" method="get">
    <button sec:authorize="isAuthenticated()">Logout</button>
    <button sec:authorize="isAnonymous()" th:href="@{/login}">Login</button>
  </form>
</body>
</html>
```

16)templates/member.html

```
<!DOCTYPE html>
<html xmlns:th="http://thymeleaf.org"
xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
<head>
<meta charset="UTF-8" />
<title>Insert title here</title>
</head>
<body>
  <h1>Member page : Login 성공한 분만 보입니다.</h1>
  <ul>
    <li>User name : <span sec:authentication="name"></span></li>
    <li>Authorities : <span sec:authentication="principal.authorities"></span></li>
  </ul>
  <a th:href="@{/loginSuccess}">뒤로 가기</a>
  <form th:action="@{/logout}" method="get">
    <button sec:authorize="isAuthenticated()">Logout</button>
    <button sec:authorize="isAnonymous()" th:href="@{/login}">Login</button>
  </form>
</body>
</html>
```

17)test

-Spring Boot Stop

-Project > right-click > Run As > Spring Boot App

-http://localhost:8080/member

-member의 계정으로 login하면 loginSuccess page에서

로그인 인증 성공
Index Page로 이동
Member Page로 이동

-Click [Member Page로 이동]

Member page : Login 성공한 분만 보입니다.
User name : admin
Authorities : [ROLE_ADMIN]
뒤로 가기
Logout

-admin의 계정으로 login하면 loginSuccess page에서

로그인 인증 성공
Index Page로 이동
Member Page로 이동
Admin 전용 Page로 이동

-Click [Admin 전용 Page로 이동]

-http://localhost:8080/admin

Admin page : Admin 권한을 가진 분만 보입니다.
User name : admin
Authorities : [ROLE_ADMIN]
뒤로 가기
Logout

-manager의 계정으로 login하면 loginSuccess page에서

로그인 인증 성공
Index Page로 이동
Member Page로 이동
Manager 전용 Page로 이동

-Click [Manager 전용 Page로 이동]

-http://localhost:8080/manager

Manager page : Manager 권한을 가진 분만 보입니다.
User name : manager
Authorities : [ROLE_MANAGER]
뒤로 가기
Logout

7. @AuthenticationPrincipal과 org.springframework.security.core.userdetails.User를 사용하여 template page에 user 정보 출력하기

1)pom.xml에 가서 위의 6번에서 추가했던 다음의 dependency를 제거한다.

```
<dependency>
   <groupId>org.thymeleaf.extras</groupId>
   <artifactId>thymeleaf-extras-springsecurity5</artifactId>
   <version>3.0.4.RELEASE</version>
</dependency>
```

2)pom.xml > right-click > Run As > Maven Install

3)com.example.security.SecurityConfig.java 수정

```
-//private DataSource dataSource;  <--주석 처리

@Autowired private MemberService memberService;  <--주석 해제

@Override
protected void configure(HttpSecurity http) throws Exception {
  log.info("security config...");
  http.authorizeRequests().antMatchers("/").permitAll();
  http.authorizeRequests().antMatchers("/member/**").authenticated();
  http.authorizeRequests().antMatchers("/manager/**").hasRole("MANAGER");
  http.authorizeRequests().antMatchers("/admin/**").hasRole("ADMIN");

  http.formLogin().loginPage("/login").defaultSuccessUrl("/loginSuccess", true);
  http.exceptionHandling().accessDeniedPage("/accessDenied");
  http.logout().logoutUrl("/logout").invalidateHttpSession(true).logoutSuccessUrl("/login"
  );

  http.userDetailsService(memberService);      <--주석 해제
}

...
...
// @Override    <---주석 처리
// protected void configure(AuthenticationManagerBuilder auth) throws
java.lang.Exception{
//   String query1 = "SELECT userid username, passwd password, enabled FROM Member
WHERE userid = ?";
//   String query2 = "SELECT userid, role FROM Member WHERE userid = ?";
//
```

```
    auth.jdbcAuthentication().dataSource(dataSource).usersByUsernameQuery(query1).autho
    ritiesByUsernameQuery(query2);
1527    //}
1528
1529  4)com.example.service.MemberService.java 수정
1530
1531    @Override
1532    public UserDetails loadUserByUsername(String username) throws
        UsernameNotFoundException {
1533      log.info("called loadUserByUsername() user id {}", username);
1534      MemberVO member = this.memberDao.getUserByID(username);
1535      log.debug("memer = {}", member.toString());
1536
1537      Collection<SimpleGrantedAuthority> roles = new
          ArrayList<SimpleGrantedAuthority>();
1538      String [] array = member.getRole().split(",");
1539      for(String str : array) roles.add(new SimpleGrantedAuthority(str));
1540      //log.debug("Role : {}", roles.toString());
1541      UserDetails user = new User(username, member.getPasswd(), roles);
1542      return user;
1543    }
1544
1545  5)com.example.security.SecurityController.java 수정
1546
1547    ...
1548    import org.springframework.security.core.annotation.AuthenticationPrincipal;
1549    import org.springframework.security.core.userdetails.User;
1550    ...
1551    ...
1552    ...
1553      @GetMapping("/member")
1554      public void forMember(@AuthenticationPrincipal User principal, Model model) {
1555        log.info("called Member page user's role = {}", principal.getAuthorities());
1556      }
1557
1558      @GetMapping("/manager")
1559      public void forManager(@AuthenticationPrincipal User principal, Model model) {
1560        log.info("called Manager page");
1561        model.addAttribute("username", principal.getUsername());
1562        model.addAttribute("roles", principal.getAuthorities());
1563      }
1564
1565      @GetMapping("/admin")
1566      public void forAdmin(@AuthenticationPrincipal User principal, Model model) {
1567        log.info("called Admin page");
1568        model.addAttribute("username", principal.getUsername());
1569        model.addAttribute("roles", principal.getAuthorities());
1570      }
1571      ...
1572      @GetMapping("/loginSuccess")
1573      public void loginSuccess(@AuthenticationPrincipal User principal, Model model) {
1574      }
1575
1576  6)templates/loginSuccess.html
1577
1578    <!DOCTYPE html>
1579    <html xmlns:th="http://thymeleaf.org">
1580    <head>
```

```
1581    <meta charset="UTF-8" />
1582    <title>Insert title here</title>
1583    </head>
1584    <body>
1585      <h3><span style="color:red">로그인 인증 성공</span></h3>
1586      <h5><a th:href="@{/}">Index Page로 이동</a></h5>
1587      <div th:each="authority : ${#authentication.principal.authorities}">
1588      <h4 th:if="${#strings.trim(authority)} eq 'ROLE_MEMBER'"><a
          th:href="@{/member}">Member Page로 이동</a></h4>
1589      <h4 th:if="${#strings.trim(authority)} eq 'ROLE_MANAGER'"><a
          th:href="@{/member}">Manager 전용 Page로 이동</a></h4>
1590      <h4 th:if="${#strings.trim(authority)} eq 'ROLE_ADMIN'"><a
          th:href="@{/admin}">Admin 전용 Page로 이동</a></h5>
1591      </div>
1592    </body>
1593    </html>
1594
1595  7)templates/member.html
1596
1597    <!DOCTYPE html>
1598    <html xmlns:th="http://thymeleaf.org">
1599    <head>
1600    <meta charset="UTF-8" />
1601    <title>Insert title here</title>
1602    </head>
1603    <body>
1604      <h1>Member page : Login 성공한 분만 보입니다.</h1>
1605      <ul th:if="${#authentication.principal.username ne null}">
1606        <li>User name : <span
          th:text="${#authentication.principal.username}"></span></li>
1607        <li>Authorities : <span
          th:text="${#authentication.principal.authorities}"></span></li>
1608      </ul>
1609      <form th:action="@{/logout}" method="get"
          th:if="${#authentication.principal.username ne null}">
1610        <button>Logout</button>
1611      </form>
1612    </body>
1613    </html>
1614
1615  8)template/manager.html
1616
1617    <!DOCTYPE html>
1618    <html xmlns:th="http://thymeleaf.org">
1619    <head>
1620    <meta charset="UTF-8" />
1621    <title>Insert title here</title>
1622    </head>
1623    <body>
1624      <h1>Manager page : Manager 권한을 가진 분만 보입니다.</h1>
1625      <ul th:if="${username ne null}">
1626        <li>User name : <span th:text="${username}"></span></li>  <--위 처럼 길게 쓰지 않
          아도 됨 -->
1627        <li>Authorities : <span th:text="${roles}"></span></li>   <--그 이유는 Controller 에
          서 이미 정제되서 들어오기 때문 -->
1628      </ul>
1629      <form th:action="@{/logout}" method="get"  th:if="${username ne null}">
1630        <button>Logout</button>
```

```
      </form>
    </body>
    </html>

  9)templates/admin.html

    <!DOCTYPE html>
    <html xmlns:th="http://thymeleaf.org"
    xmlns:sec="http://www.thymeleaf.org/extras/spring-security">
    <head>
    <meta charset="UTF-8" />
    <title>Insert title here</title>
    </head>
    <body>
      <h1>Admin page : Admin 권한을 가진 분만 보입니다.</h1>
      <ul th:if="${username ne null}">
        <li>User name : <span th:text="${username}"></span></li>
        <li>Authorities : <span th:text="${roles}"></span></li>
      </ul>
      <form th:action="@{/logout}" method="get"  th:if="${username ne null}">
        <button>Logout</button>
      </form>
    </body>
    </html>

  10)Test



--------------------------------
Task4. Naver, Kakao, Google, Facebook 인증 설정하기
1. Facebook
  1)Facebook 개발자 페이지(https://developers.facebook.com/) 방문
  2)오른쪽 상단의 내 앱 > 앱 만들기
  3)새 앱 ID 만들기
    -표시 이름 : oauth2-demo
    -연락처 이메일 : javaexpert@nate.com
    -Click 앱 ID 만들기
    -Check 로봇이 아닙니다.
    -제출

  4)왼쪽 Sidebar의 설정 > 기본 설정
    -앱 ID :
    -앱 시크릿 코드 :

  5)왼쪽 Sidebar에서 제품 Click
    -Facebook 로그인 > 설정
    -클라이언트 OAuth 로그인 : 예
    -웹 OAuth 로그인 : 예
    -HTTPS 적용 : 예
    -웹 OAuth 재인증 사용 : 아니요
    -포함된 브라우저 OAuth 로그인 : 아니요
    -리디렉션 URI에 Strict 모드 사용 : 예
    -유효한 OAuth 리디렉션 URI
      --http://localhost:8080/login/oauth2/code/facebook
    -기기에서 로그인 : 아니요
    -변경 내용 저장 Click

```

2. Google
  1)Google Developer Console(https://console.developers.google.com/)
  2)Click [NewMyMapProject] > [새 프로젝트]
  3)프로젝트 이름 : spring-boot-oauth2-demo
  4)프로젝트 ID : spring-boot-oauth2-demo-276311
  5)만들기 Click
  6)상단 Bar에 [spring-boot-oauth2-demo] 선택 확인 > 왼쪽 사이드 바 > 사용자 인증 정보
  7)[사용자 인증 정보 만들기] Click > OAuth 클라이언트 ID 만들기
  8)애플리케이션 유형 > 웹 애플리케이션 선택
  9)이름 : spring-boot-oauth2-demo
  10)제한사항 > 승인된 자바스크립트 원본 : http://localhost:8080
  11)승인된 리디렉션 URI : http://localhost:8080/login/oauth2/code/google
  12)생성 버튼 Click
  13)OAuth 클라이언트 생성됨
    -클라이언트 ID :
    -클라이언트 보안 비밀번호 :
  14)확인

3. Kakao
  1)카카오 개발자 페이지(https://developers.kakao.com/)
  2)내 애플리케이션 > 애플리케이션 추가하기
  3)애플리케이션 추가
    -앱 이름 : spring-boot-oauth2-demo
    -회사 이름 : freelancer
    -저장
  4)앱 키
    -네이티브 앱 키 :
    -REST API 키 : <--여기서 사용할 키
    -JavaScript 키 :
    -Admin 키 :
  5)왼쪽 Sidebar > 카카오 로그인 > 동의항목
    -profile : 필수 동의
    -카카오계정(이메일) : 이용중 동의
  6)카카오 로그인을 Off에서 ON으로 변경

  7)왼쪽 Sidebar > 플랫폼 > Web 플랫폼 등록
    -http://localhost:8080

  8)왼쪽 Sidebar > 카카오 로그인 > Redirect URI
    -http://localhost:8080/login/oauth2/code/kakao

  9)왼쪽 Sidebar > 보안 >
    -Client Secret > 코드 생성
    -활성화 상태 : 사용함으로 설정

4. Naver
  1)Naver Developer(https://developers.naver.com/apps)
  2)[Application 등록]
    -애플리케이션 이름 : spring-boot-oauth2-demo
    -사용 API : 네아로(네이버 아이디로 로그인)
      --회원이름과 이메일만 체크
    -로그인 오픈 API 서비스 환경 : PC 웹
      -서비스 URL : http://localhost:8080
      -네이버아이디로로그인 Callback URL : http://localhost:8080/login/oauth2/code/naver
    -등록하기

```
1746    3)애플리케이션 정보
1747      -Client ID :
1748      -Client Secret :
1749
1750
1751  ------------------------------
1752  Task5. OAuth2 인증하기
1753  1. Spring Boot Project 생성
1754    1)Package Explorer > right-click > New > Spring Starter Project
1755    2)다음의 각 항목의 값을 입력후 Next 클릭
1756      -Service URL :http://start.spring.io
1757      -Name : SpringBootOAuth2Demo
1758      -Type : Maven
1759      -Packaging : Jar
1760      -Java Version : 8
1761      -Language : Java
1762      -Group : com.example
1763      -Artifact : SpringBootOAuth2Demo
1764      -Version : 0.0.1-SNAPSHOT
1765      -Description : Demo project for Spring Boot
1766      -Package : com.example.biz
1767
1768    3)다음 각 항목을 선택후 Finish 클릭
1769      -Spring Boot Version : 2.2.6
1770      -Select
1771        --Web > Spring Web
1772        --Developer Tools > Lombok, Spring Boot DevTools
1773        --Template Engines > Thymeleaf
1774        --Security > Spring Security, OAuth2 Client
1775      -Finish
1776
1777    4)Project version up
1778      -Java Build Path > JRE System Library [jdk 1.8.0_251] > Apply
1779      -Java compiler > JDK Compliance > 1.8 > Apply
1780      -Project Facets > Java > 1.8
1781      -Apply and Close
1782
1783
1784  2. Thymeleaf page
1785    1)templates/hello.html
1786
1787      <!DOCTYPE html>
1788      <html xmlns:th="http://thymeleaf.org">
1789      <head>
1790      <meta charset="UTF-8" />
1791      <title>Insert title here</title>
1792      </head>
1793      <body>
1794      <h1>인증 성공</h1>
1795      <ul>
1796      <li>User name : <span
          th:text="${#authentication.principal.attributes.email}"></span></li>
1797      <li>Authorities : <span
          th:text="${#authentication.principal.authorities}"></span></li>
1798      </ul>
1799      <form th:action="@{/logout}" method="get">
1800        <button th:if="${#authentication ne null}">Logout</button>
1801      </form>
```

```
  </body>
  </html>

2)templates/home.html

  <!DOCTYPE html>
  <html xmlns:th="http://www.thymeleaf.org">
  <head>
  <meta charset="UTF-8" />
  <title>Insert title here</title>
  </head>
  <body>
    <h1>Home Page</h1>
    <a th:href="@{/login}">로그인</a>
  </body>
  </html>

3)templates/login.html

  <!DOCTYPE html>
  <html lang="en" xmlns:th="http://www.thymeleaf.org">
  <head>
  <meta charset="UTF-8" />
  <title>Insert title here</title>
  <script th:src="@{/js/jquery-3.5.1.min.js}"></script>
  <script>
    $(function(){
      $('a').bind("click", function(){
        location.href = "/oauth2/authorization/" + $(this).attr("data-social");
      });
    });
  </script>
  </head>
  <body>
    <h1>Login Page</h1>
    <a href="#" class="btn_social" data-social="facebook">Facebook</a><br />
    <a href="#" class="btn_social" data-social="google">Google</a><br />
    <a href="#" class="btn_social" data-social="kakao">Kakao</a><br />
    <a href="#" class="btn_social" data-social="naver">Naver</a><br />
  </body>
  </html>

4)templates/loginFailure.html

  <!DOCTYPE html>
  <html xmlns:th="http://www.thymeleaf.org">
  <head>
  <meta charset="UTF-8" />
  <title>Insert title here</title>
  </head>
  <body>
    <h1>인증 실패</h1>
    <a th:href="@{/login}">Login Again</a>
  </body>
  </html>

5)templates/logout.html
```

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
<meta charset="UTF-8" />
<meta name="viewport" content="width=device-width, initial-scale=1.0" />
<title>Document</title>
<link rel="stylesheet" th:href="@{/css/bootstrap.min.css}" />
<link rel="stylesheet" th:href="@{/css/bootstrap-theme.min.css}" />
</head>
<body>
  <h2>Custom Logout Page</h2>
  <form method="POST">
    <h3>Logout</h3>
    <input type="hidden" th:name="${_csrf.parameterName}"
      th:value="${_csrf.token}" />
    <button type="submit" class="btn btn-primary">Logout</button>
  </form>
</body>
</html>
```

6)static/js/jquery-3.5.1.min.js
-src/main/resources > right-click > New > Folder
-Name : js
-Paste jquery-3.5.1.min.js to static/js folder

7)static/css/bootstrap-theme.min.css, bootstrap.min.css
-src/main/resources > right-click > New > Folder
-Name : css
-Paste bootstrap-theme.min.css, bootstrap.min.css to static/css folder

8)static/fonts/glyphicons-halflings-regular.*
-src/main/resources > right-click > New > Folder
-Name : fonts
-Paste glyphicons-halflings-regular.eot등 5개 to static/fonts folder

3. src/main/resources/application.properties

```
spring.thymeleaf.cache=false

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

#registration
spring.security.oauth2.client.registration.kakao.client-id=
spring.security.oauth2.client.registration.kakao.client-secret=
spring.security.oauth2.client.registration.kakao.redirect-uri={baseUrl}/{action}/oauth2/code/{registrationId}
spring.security.oauth2.client.registration.kakao.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.kakao.scope=profile,email
spring.security.oauth2.client.registration.kakao.client-name=Kakao

spring.security.oauth2.client.registration.naver.client-id=
spring.security.oauth2.client.registration.naver.client-secret=
```

```
spring.security.oauth2.client.registration.naver.redirect-uri={baseUrl}/{action}/oauth2/code/{registrationId}
spring.security.oauth2.client.registration.naver.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.naver.scope=name,email
spring.security.oauth2.client.registration.naver.client-name=Naver

#Provider
spring.security.oauth2.client.provider.kakao.authorization-uri=https://kauth.kakao.com/oauth/authorize
spring.security.oauth2.client.provider.kakao.token-uri=https://kauth.kakao.com/oauth/token
spring.security.oauth2.client.provider.kakao.user-info-uri=https://kapi.kakao.com/v2/user/me
spring.security.oauth2.client.provider.kakao.user-name-attribute=response

spring.security.oauth2.client.provider.naver.authorization-uri=https://nid.naver.com/oauth2.0/authorize
spring.security.oauth2.client.provider.naver.token-uri=https://nid.naver.com/oauth2.0/token
spring.security.oauth2.client.provider.naver.user-info-uri=https://openapi.naver.com/v1/nid/me
spring.security.oauth2.client.provider.naver.user-name-attribute=response
```

4. com.example.biz.SpringBootOAuth2DemoApplication.java

```
package com.example.biz;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.ComponentScan;

@SpringBootApplication
@ComponentScan(basePackages = "com.example")
public class SpringBootOAuth2DemoApplication {

   public static void main(String[] args) {
      SpringApplication.run(SpringBootOAuth2DemoApplication.class, args);
   }

}
```

5. Package 생성
   1)src/main/java > right-click > New > Package
   2)Name : com.example.oauth2.security

6. Controller 생성
   1)com.example.oauth2.security > right-click > New > Class
   2)Name : OAuth2Controller

```
   package com.example.oauth2.security;

   import java.security.Principal;

   import org.springframework.stereotype.Controller;
```

```
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

import lombok.extern.slf4j.Slf4j;

@Slf4j
@Controller
public class OAuth2Controller {

  @GetMapping({ "", "/" })
  public String getAuthorizationMessage() {
    return "home";
  }

  @GetMapping("/login")
  public String login() {
    return "login";
  }

  @GetMapping({ "/loginSuccess", "/hello" })
  public String loginSuccess(Principal principal, Model model) {
    return "hello";
  }

  @GetMapping("/loginFailure")
  public String loginFailure() {
    return "loginFailure";
  }

  @GetMapping("/logout")
  public void logout() {
    log.info("called Logout page");
  }
}
```

7. OAuthAttributes.java 생성하기
  1)com.example.oauth2.security > right-click > New > Class
  2)Name : OAuthAttributes

```
package com.example.oauth2.security;

import java.util.Map;

import lombok.Builder;
import lombok.Getter;
import lombok.extern.slf4j.Slf4j;

@Slf4j
@Getter
public class OAuthAttributes {
  private Map<String, Object> attributes;
  private String nameAttributeKey;
  private String name;
  private String email;

  @Builder
  public OAuthAttributes(Map<String, Object> attributes, String nameAttributeKey,
```

```
        String name, String email) {
2024      this.attributes = attributes;
2025      this.nameAttributeKey = nameAttributeKey;
2026      this.name = name;
2027      this.email = email;
2028    }
2029
2030    public static OAuthAttributes of(String registrationId, String userNameAttributeName,
2031        Map<String, Object> attributes) {
2032      if ("naver".equals(registrationId)) {
2033        return ofNaver("id", attributes);
2034      }
2035      return ofGoogle(userNameAttributeName, attributes);
2036    }
2037
2038    private static OAuthAttributes ofGoogle(String userNameAttributeName, Map<String,
        Object> attributes) {
2039      return OAuthAttributes.builder().name((String)
          attributes.get("name")).email((String) attributes.get("email"))
2040          .attributes(attributes).nameAttributeKey(userNameAttributeName).build();
2041    }
2042
2043    private static OAuthAttributes ofNaver(String userNameAttributeName, Map<String,
        Object> attributes) {
2044      Map<String, Object> response = (Map<String, Object>) attributes.get("response");
2045      return OAuthAttributes.builder().email((String) response.get("email")).name((String)
          response.get("name"))
2046          .attributes(response).nameAttributeKey(userNameAttributeName).build();
2047    }
2048  }
2049
2050
2051 8. CustomOAuth2UserService.java 생성하기
2052   1)src/main/java > right-click > New > Package
2053   2)Name : com.example.service
2054   3)com.example.service > right-click > New > Class
2055   4)Name : CustomOAuth2UserService
2056   5)Interfaces : org.springframework.security.oauth2.client.userinfo.OAuth2UserService;
2057
2058    package com.example.service;
2059
2060    import java.util.LinkedHashSet;
2061    import java.util.Set;
2062
2063    import org.springframework.security.core.GrantedAuthority;
2064    import org.springframework.security.core.authority.SimpleGrantedAuthority;
2065    import org.springframework.security.oauth2.client.userinfo.DefaultOAuth2UserService;
2066    import org.springframework.security.oauth2.client.userinfo.OAuth2UserRequest;
2067    import org.springframework.security.oauth2.client.userinfo.OAuth2UserService;
2068    import org.springframework.security.oauth2.core.OAuth2AccessToken;
2069    import org.springframework.security.oauth2.core.OAuth2AuthenticationException;
2070    import org.springframework.security.oauth2.core.user.DefaultOAuth2User;
2071    import org.springframework.security.oauth2.core.user.OAuth2User;
2072    import org.springframework.security.oauth2.core.user.OAuth2UserAuthority;
2073
2074    import com.example.oauth2.security.OAuthAttributes;
2075
2076    import lombok.extern.slf4j.Slf4j;
```

```
2077
2078      @Slf4j
2079      public class CustomOAuth2UserService implements
          OAuth2UserService<OAuth2UserRequest, OAuth2User> {
2080        @Override
2081        public OAuth2User loadUser(OAuth2UserRequest userRequest) throws
          OAuth2AuthenticationException {
2082          OAuth2UserService delegate = new DefaultOAuth2UserService();
2083          OAuth2User oAuth2User = delegate.loadUser(userRequest);
2084          String registrationId = userRequest.getClientRegistration().getRegistrationId();
2085          String userNameAttributeName =
          userRequest.getClientRegistration().getProviderDetails().getUserInfoEndpoint()
2086              .getUserNameAttributeName();
2087          OAuthAttributes attributes = OAuthAttributes.of(registrationId,
          userNameAttributeName,
2088              oAuth2User.getAttributes());
2089          Set<GrantedAuthority> authorities = new LinkedHashSet<>();
2090          authorities.add(new OAuth2UserAuthority(attributes.getAttributes()));
2091          OAuth2AccessToken token = userRequest.getAccessToken();
2092          for (String authority : token.getScopes()) {
2093            authorities.add(new SimpleGrantedAuthority("SCOPE_" + authority));
2094          }
2095          return new DefaultOAuth2User(authorities, attributes.getAttributes(),
          attributes.getNameAttributeKey());
2096        }
2097      }
2098
2099
2100  9. SocialType Enum 생성하기
2101    1)com.example.oauth2.security > right-click > New > Enum
2102    2)Name : SocialType
2103
2104      package com.example.oauth2.security;
2105
2106      public enum SocialType {
2107        FACEBOOK("facebook"),
2108        GOOGLE("google"),
2109        KAKAO("kakao"),
2110        NAVER("naver");
2111
2112        private final String ROLE_PREFIX = "ROLE_";
2113        private String name;
2114
2115        SocialType(String name) { this.name = name; }
2116        public String getRoleType() {
2117          return ROLE_PREFIX + name.toUpperCase();
2118        }
2119
2120        public String getValue() {
2121          return name;
2122        }
2123
2124        public boolean isEquals(String authority) {
2125          return this.getRoleType().equals(authority);
2126        }
2127      }
2128
2129
```

10. SecurityConfig.java 생성
  1)com.example.oauth2.security > right-click > New > Class
  2)Name : SecurityConfig
  3)Superclass : org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter

```
package com.example.oauth2.security;

import static com.example.oauth2.security.SocialType.FACEBOOK;
import static com.example.oauth2.security.SocialType.GOOGLE;
import static com.example.oauth2.security.SocialType.KAKAO;
import static com.example.oauth2.security.SocialType.NAVER;

import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.web.authentication.LoginUrlAuthenticationEntryPoint;

import com.example.service.CustomOAuth2UserService;

@Configuration
@EnableWebSecurity
public class SecurityConfig extends WebSecurityConfigurerAdapter{

  @Override
  protected void configure(HttpSecurity http) throws java.lang.Exception{
    http.authorizeRequests().antMatchers("/", "/oauth2/**", "/login/**", "/css/**",
                                         "/images/**", "/js/**", "/console/**", "/favicon.ico/**")
        .permitAll()
        .antMatchers("/facebook").hasAuthority(FACEBOOK.getRoleType())
        .antMatchers("/google").hasAuthority(GOOGLE.getRoleType())
        .antMatchers("/kakao").hasAuthority(KAKAO.getRoleType())
        .antMatchers("/naver").hasAuthority(NAVER.getRoleType())
        .anyRequest().authenticated()
        .and()
        .oauth2Login()
        .userInfoEndpoint().userService(new CustomOAuth2UserService())
        .and()
        .defaultSuccessUrl("/loginSuccess")
        .failureUrl("/loginFailure")
        .and()
        .exceptionHandling()
        .authenticationEntryPoint(new LoginUrlAuthenticationEntryPoint("/login"));
    http.logout().logoutUrl("/logout").invalidateHttpSession(true).logoutSuccessUrl("/login");
  }
}
```