

2025년 상반기 K-디지털 트레이닝

Spring Web Security 소개

[KB] IT's Your Life

- ✓ 다음과 같이 프로젝트를 생성하고, Spring Security 의존성을 추가하세요.
 - Template: MyBatisSpringLegacy
 - name: secserver

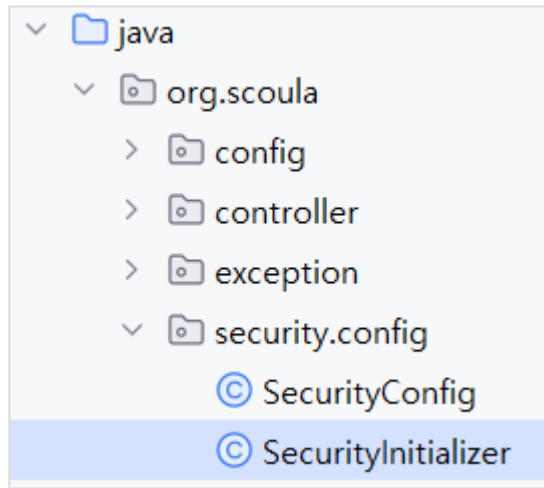
✓ build.gradle

```
...
ext {
    junitVersion = '5.9.2'
    springVersion = '5.3.37'
    ...
    springSecurityVersion='5.8.13'
}

dependencies {
    ...

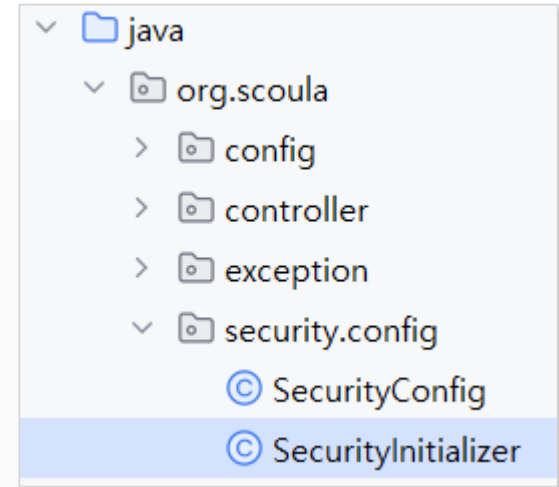
    // 보안
    implementation("org.springframework.security:spring-security-web:${springSecurityVersion}")
    implementation("org.springframework.security:spring-security-config:${springSecurityVersion}")
    implementation("org.springframework.security:spring-security-core:${springSecurityVersion}")
    implementation("org.springframework.security:spring-security-taglibs:${springSecurityVersion}")
    ...
}
```

- ✔ 다음과 같이 SecurityInitializer 클래스를 추가하고, 기본 코드를 작성하세요.
 - AbstractSecurityWebApplicationInitializer 상속

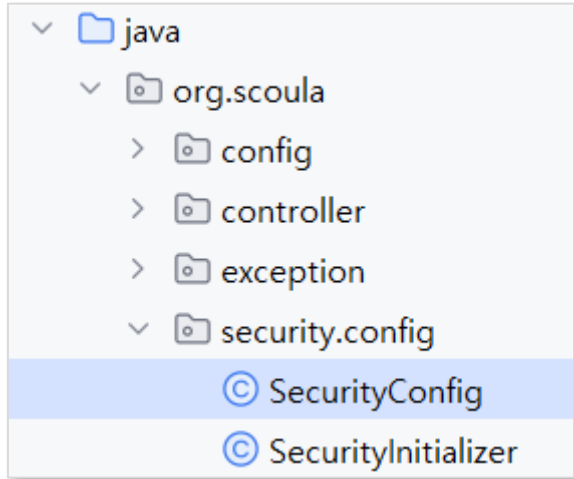


✓ SecurityInitializer.java

```
package org.scoula.security.config;  
  
import org.springframework.security.web.context.AbstractSecurityWebApplicationInitializer;  
  
public class SecurityInitializer extends AbstractSecurityWebApplicationInitializer {  
  
}
```



- ✓ 다음과 같이 SecurityConfig 클래스를 추가하고, 기본 골격을 작성하세요.



- ✓ WebConfig에 SecurityConfig 설정을 추가하세요.

✓ SecurityConfig.java

```
package org.scoula.security.config;

@Configuration
@EnableWebSecurity
@Log4j2
public class SecurityConfig extends WebSecurityConfigurerAdapter {

}
```

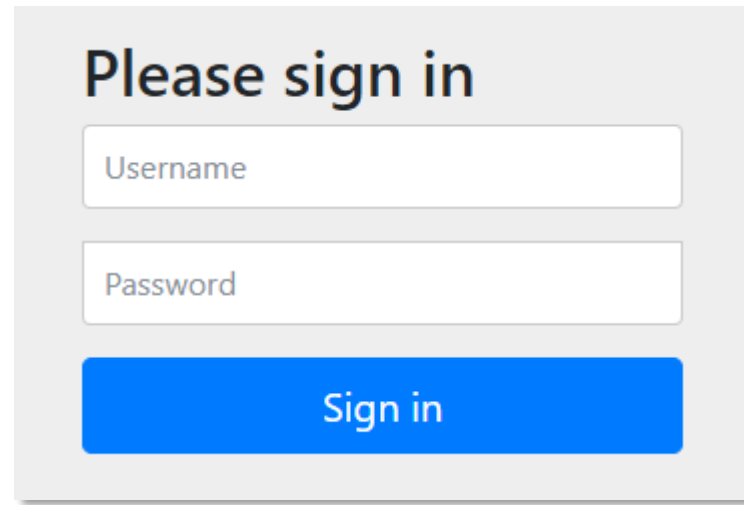
✓ config/WebConfig.java

```
@Override
public Class<?>[] getRootConfigClasses() {
    return new Class[] { RootConfig.class, SecurityConfig.class };
}
```

- ✓ **http://localhost:8080/ 접속시 로그인 페이지로 리다이렉트되는지 확인하세요.**

- ✓ <http://localhost:8080> 실행

<http://localhost:8080/login>



Please sign in

Username

Password

Sign in

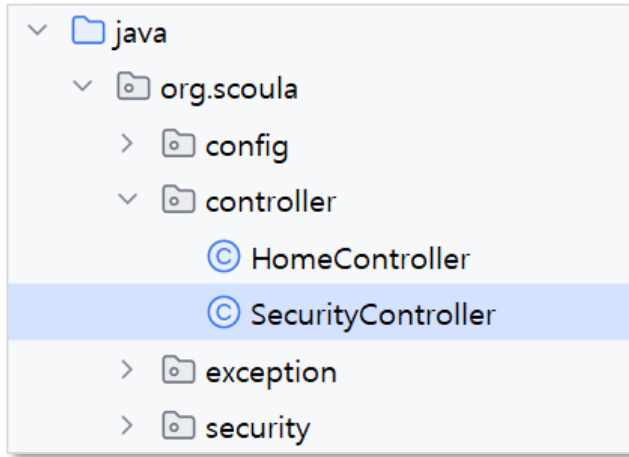
- ✔ SecurityConfig에서 UTF-8 문자셋 인코딩 필터를 작성하고 CSRF 필터 앞에 배치하세요.

✔ SecurityController.java

```
...
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    // 문자셋 필터
    public CharacterEncodingFilter encodingFilter() {
        CharacterEncodingFilter encodingFilter = new CharacterEncodingFilter();
        encodingFilter.setEncoding("UTF-8");
        encodingFilter.setForceEncoding(true);
        return encodingFilter;
    }

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.addFilterBefore(encodingFilter(), CsrfFilter.class);
    }
    ...
}
```

- ✔ 다음과 같이 SecurityController를 추가하고, 다음을 처리하도록 작성하세요.



- /security/all 접속시 로그 출력 및 sercurity/all 뷰로 이동
- /security/member 접속시 로그 출력 및 sercurity/member 뷰로 이동
- /security/admin 접속시 로그 출력 및 sercurity/admin 뷰로 이동
- all, member, admin 뷰 jsp 작성

✓ controller/SecurityController.java

```
package org.scoula.controller;

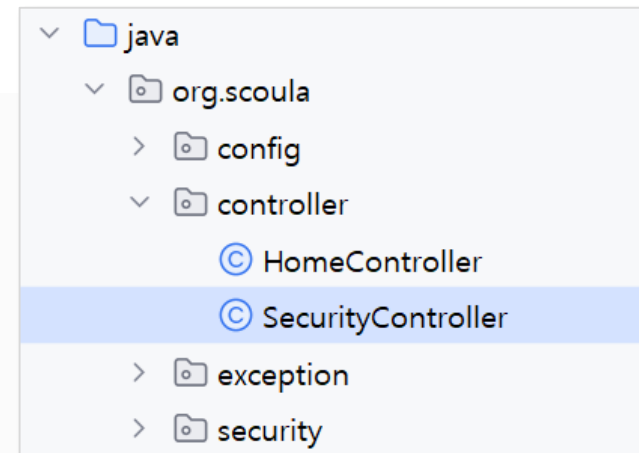
...

@Log4j2
@RequestMapping("/security")
@Controller
public class SecurityController {

    @GetMapping("/all")    // 모두 접근 가능
    public void doAll() {
        log.info("do all can access everybody");
    }

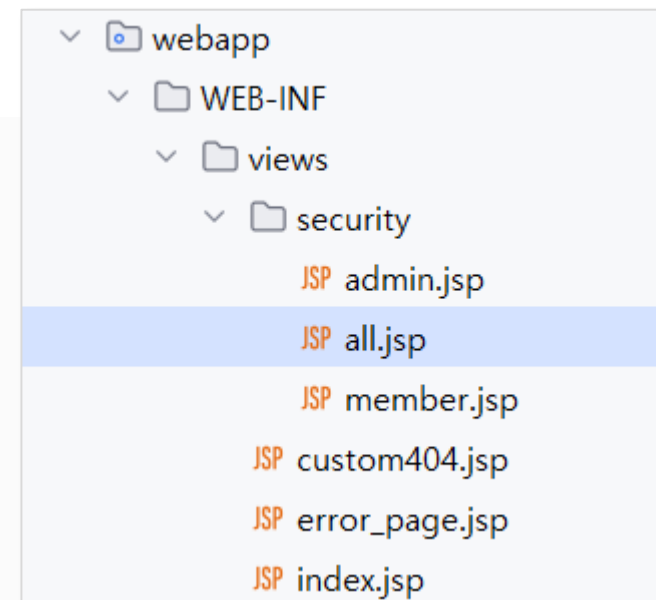
    @GetMapping("/member") // MEMBER 또는 ADMIN 권한 필요
    public void doMember() {
        log.info("logged member");
    }

    @GetMapping("/admin") // ADMIN 권한 필요
    public void doAdmin() {
        log.info("admin only");
    }
}
```



✓ views/security 폴더에 all.jsp, member.jsp, admin.jsp 생성

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
    <h1>/security/all page</h1>
</body>
</html>
```



- <h1> 부분은 각 페이지에 맞게 수정

- ✔ SecurityConfig에서 다음 경로에 대해서 접근 권한을 설정하세요.
 - /security/all : 모두 접근 가능
 - /security/member: ROLE_MEMBER 권한 이상인 경우 접근 가능
 - /security/admin: ROLE_ADMIN 권한인 경우 접근 가능

✓ SecurityConfig.java

```
...

@Configuration
@EnableWebSecurity
@Log4j2
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    public void configure(HttpSecurity http) throws Exception {

        // 경로별 접근 권한 설정
        http.authorizeRequests()
            .antMatchers("/security/all").permitAll()
            .antMatchers("/security/admin").access("hasRole('ROLE_ADMIN')")
            .antMatchers("/security/member").access("hasAnyRole('ROLE_MEMBER', 'ROLE_ADMIN')");
    }
}
```


✓ <http://localhost:8080/security/all>

/security/all page

✓ <http://localhost:8080/security/member>

✓ <http://localhost:8080/security/admin>

- 에러 발생보다는 로그인 페이지로 리다이렉트하는 것이 좋음
--> 로그인 설정 필요

HTTP 상태 403 – 금지됨

탭 상태 보고

메시지 Access Denied

설명 서버가 요청을 이해했으나 승인을 거부합니다.

VMware tc Runtime 9.0.33.A.RELEASE

- ✔ 로그인 안한 상태에서 접근 권한이 없는 경우 로그인 페이지로 이동하도록 설정하세요.

✔ SecurityConfig.java

```
...
public class SecurityConfig extends WebSecurityConfigurerAdapter {

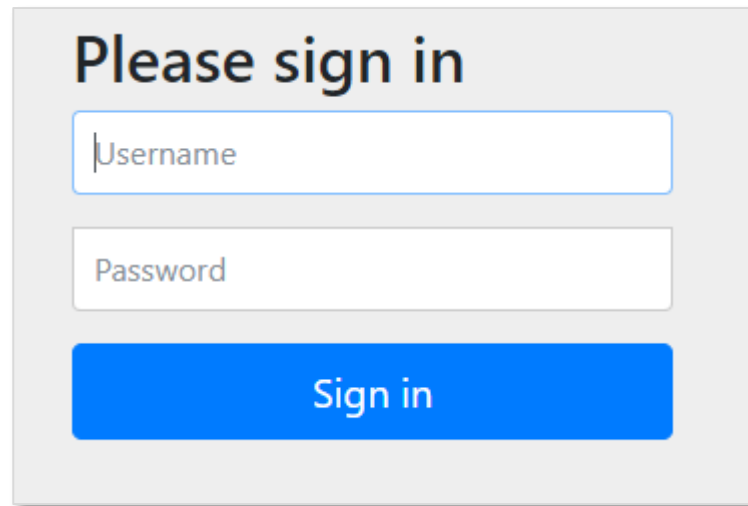
    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/security/all").permitAll()
            .antMatchers("/security/admin").access("hasRole('ROLE_ADMIN')")
            .antMatchers("/security/member").access("hasAnyRole('ROLE_MEMBER', 'ROLE_ADMIN')");

        http.formLogin(); // form 기반 로그인 활성화, 나머지는 모두 디폴트
    }
}
```

- ✓ <http://localhost:8080/security/member>
- ✓ <http://localhost:8080/security/admin>

→ /login으로 리다이렉트

<http://localhost:8080/login>



Please sign in

Username

Password

Sign in

✓ 다음과 같은 메모리 기반의 사용자 정보를 등록하세요.

- username: admin
- password: 1234(암호화 없음)
- 권한: ROLE_ADMIN, ROLE_MEMBER

- username: member
- password: 1234(암호화 없음)
- 권한: ROLE_MEMBER

✓ SecurityConfig.java

```
@Override
protected void configure(AuthenticationManagerBuilder auth)
    throws Exception {
    log.info("configure .....");

    auth.inMemoryAuthentication()
        .withUser("admin")
        .password("{noop}1234")
        .roles("ADMIN","MEMBER");    // ROLE_ADMIN

    auth.inMemoryAuthentication()
        .withUser("member")
        .password("{noop}1234")
        .roles("MEMBER"); // ROLE_MEMBER
    }
}
```

- admin 또는 member로 로그인 가능

- ✔ SecurityController에 /security/login 요청을 처리하는 메서드와 뷰를 작성하고, 로그인 페이지를 자신이 작성한 페이지로 운영하세요.

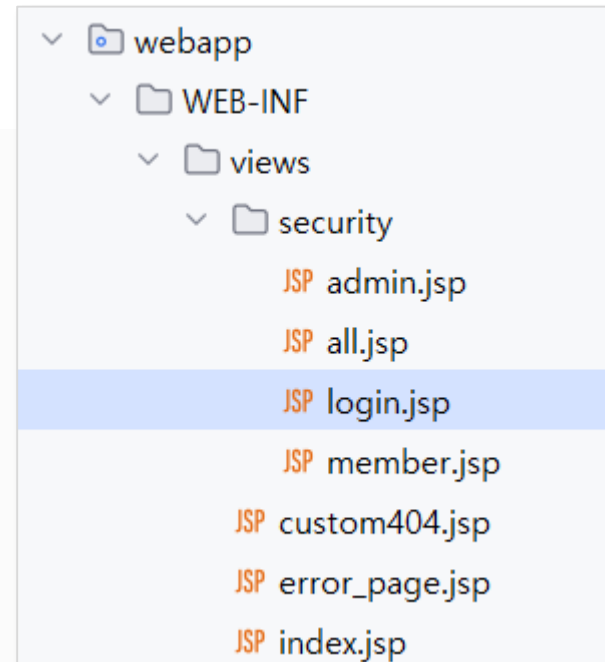
✓ controller/SecurityController.java

```
...
@RequestMapping("/security")
public class SecurityController {
    ...

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


✓ security/login.jsp

```
<body>
  <h1>login</h1>
  <form name='f' action='/security/login' method='POST'>
    <input type="hidden" name="_csrf.parameterName" value="_csrf.token" />
    <table>
      <tr>
        <td>User:</td>
        <td><input type='text' name='username' value=''></td>
      </tr>
      <tr>
        <td>Password:</td>
        <td><input type='password' name='password' /></td>
      </tr>
      <tr>
        <td colspan='2'>
          <input name="submit" type="submit" value="Login" />
        </td>
      </tr>
    </table>
  </form>
</body>
</html>
```



```
<form name='f' action='/security/login' method='POST'>
  <input type="hidden" name="_csrf" value="d5bab99f-8a5d-46ad-990e-8b5504bce41b" />
  <table>
    ...
```

✓ config/SecurityConfig.java

```
...
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/security/all").permitAll()
            .antMatchers("/security/admin").access("hasRole('ROLE_ADMIN')")
            .antMatchers("/security/member").access("hasAnyRole('ROLE_MEMBER', 'ROLE_ADMIN')");

        http.formLogin()
            .loginPage("/security/login")
            .loginProcessingUrl("/security/login")
            .defaultSuccessUrl("/");
    }
}
```

✓ 다음과 같이 로그아웃 처리되도록 설정하세요.

- 로그아웃 요청 POST URL: /security/logout
- 로그아웃 성공시 리다이렉트 URL: /security/logout

- SecurityController에 /security/logout GET 요청처리 메서드 추가
- security/logout 뷰 작성

✓ config/SecurityConfig.java

```
...
public class SecurityConfig extends WebSecurityConfigurerAdapter {
    ...
    @Override
    public void configure(HttpSecurity http) throws Exception {
        http.authorizeRequests()
            .antMatchers("/security/all").permitAll()
            .antMatchers("/security/admin").access("hasRole('ROLE_ADMIN')")
            .antMatchers("/security/member").access("hasAnyRole('ROLE_MEMBER', 'ROLE_ADMIN')");

        http.formLogin()
            .loginPage("/security/login")
            .loginProcessingUrl("/security/login")
            .defaultSuccessUrl("/");

        http.logout()                // 로그아웃 설정 시작
            .logoutUrl("/security/logout") // POST: 로그아웃 호출 url
            .invalidateHttpSession(true)   // 세션 invalidate
            .deleteCookies("remember-me", "JSESSION-ID") // 삭제할 쿠키 목록
            .logoutSuccessUrl("/security/logout"); // GET: 로그아웃 이후 이동할 페이지
    }
    ...
}
```

✓ controller.SercurityController.java

```
@RequestMapping("/security")
public class SecurityController {

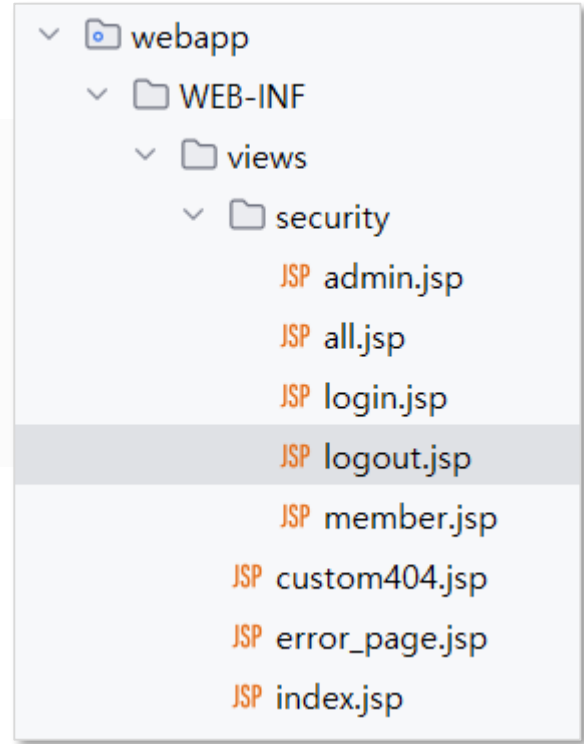
    ...

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

✓ security/logout.jsp

```
<body>
  <h1>
    로그 아웃됨
  </h1>

  <a href="/">홈으로</a>
</body>
```



- ✓ member.jsp, admin.jsp 페이지에 로그아웃 폼을 추가하고, 실제 로그아웃 되는지 확인하세요.

✓ member.jsp, admin.jsp

```
..  
  
<body>  
  <h1>/security/member page</h1>  
  
  <form action="/security/logout" method="post">  
    <input type="hidden" name="${_csrf.parameterName}" value="${_csrf.token}" />  
    <input type="submit" value="로그아웃"/>  
  </form>  
</body>  
</html>
```

/security/member page

로그아웃

redirect: /security/logout

http://localhost:8080/security/logout

로그 아웃됨

홈으로

- ✓ PasswordEncoder 빈을 SecurityConfig에 등록하고, 단위 테스트를 이용하여 비밀번호 암호화 및 검증을 테스트하세요.

✓ config/SecurityConfig.java

```
...
public class SecurityConfig extends WebSecurityConfigurerAdapter {

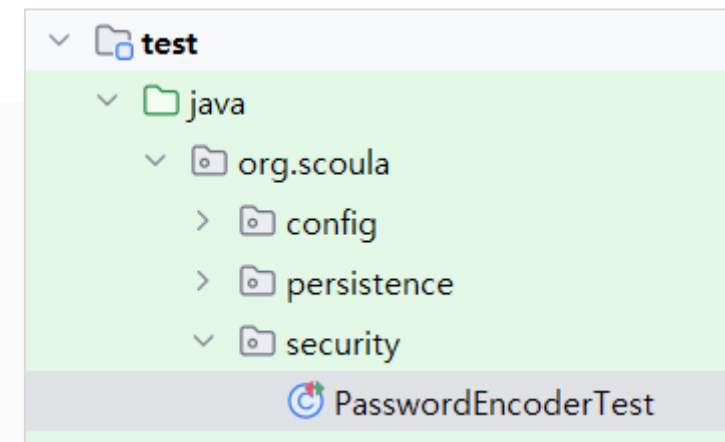
    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    ...
}
```

✓ tests:: PasswordEncoderTest.java

```
@ExtendWith(SpringExtension.class)
@ContextConfiguration(classes = {
    RootConfig.class,
    SecurityConfig.class
})
@Log4j2
public class PasswordEncoderTest {

    @Autowired
    private PasswordEncoder pwEncoder;
```



✓ tests: PasswordEncoderTests.java

```
@Test
public void testEncode() {
    String str = "1234";

    String enStr = pwEncoder.encode(str);          // 암호화
    log.info("password: " + enStr);

    String enStr2 = pwEncoder.encode(str);         // 암호화
    log.info("password: " + enStr2);

    log.info("match :" + pwEncoder.matches(str, enStr)); // 비밀번호 일치 여부 검사
    log.info("match :" + pwEncoder.matches(str, enStr2)); // 비밀번호 일치 여부 검사
}
}
```

```
INFO : org.galapagos.security.SecurityTest -
password: $2a$10$VJK.3K/W3PhSu53.FVm7W0EzFZPlGTw5.iiCZXgKTHPkhK419Jdz2
INFO : org.galapagos.security.SecurityTest -
password: $2a$10$ME3YMFVYP.Wi1YTL5ghU9.yPyuEkEBXpQl9FHBsZ9BpP0tef8hj72
INFO : org.galapagos.security.SecurityTest - match :true
INFO : org.galapagos.security.SecurityTest - match :true
```

같은 문자열이어도
매번 다르게 암호화됨

- ✓ 메모리 등록 유저의 비밀번호를 암호화된 비밀번호로 대체하세요.

✓ config/SecurityConfig.java

```
...
public class SecurityConfig extends WebSecurityConfigurerAdapter {
...
    @Override
    protected void configure(AuthenticationManagerBuilder auth)
        throws Exception {
        auth.inMemoryAuthentication()
            .withUser("admin")
            // .password("{noop}1234")
            .password("$2a$10$EsIMfxbJ6NuvwX7MDj4Wq0YFzLU9U/lldCyn0nic5dFo3VfJYrXYC")
            .roles("ADMIN","MEMBER");    // ROLE_ADMIN, ROLE_MEMBER

        auth.inMemoryAuthentication()
            .withUser("member")
            // .password("{noop}1234")
            .password("$2a$10$EsIMfxbJ6NuvwX7MDj4Wq0YFzLU9U/lldCyn0nic5dFo3VfJYrXYC")
            .roles("MEMBER"); // ROLE_MEMBER
    }
...
}
```