

January
Last
Week

0124

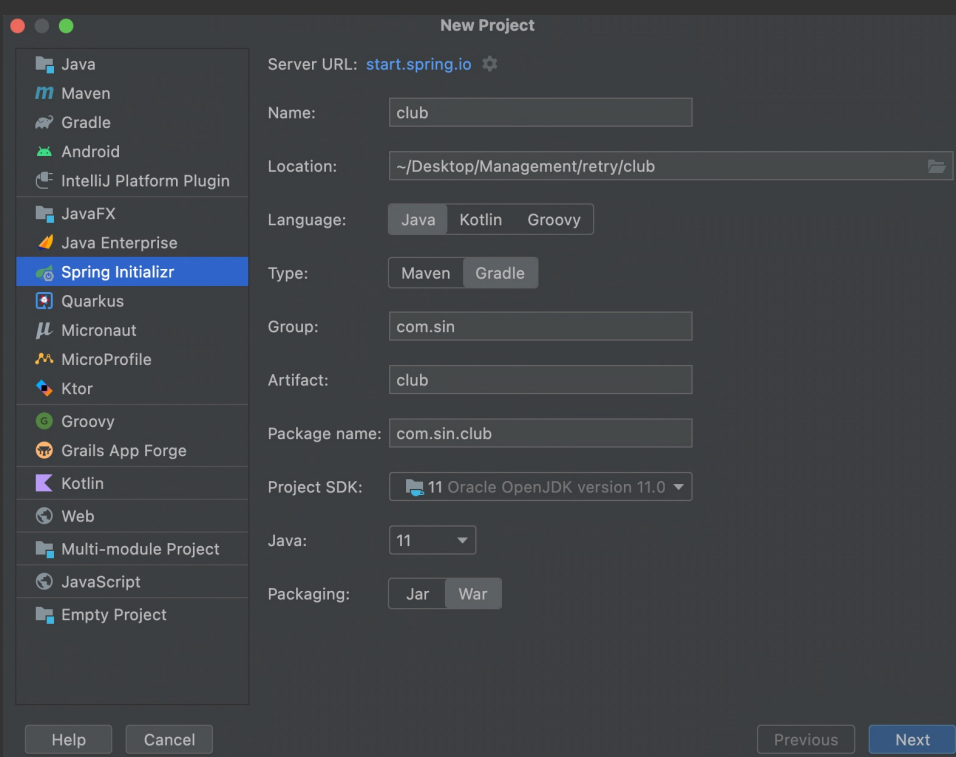
파일업로드

- 30MB. 한번에 업로드 할 수 있는 크기
- 10 MB 하나의 파일 크기

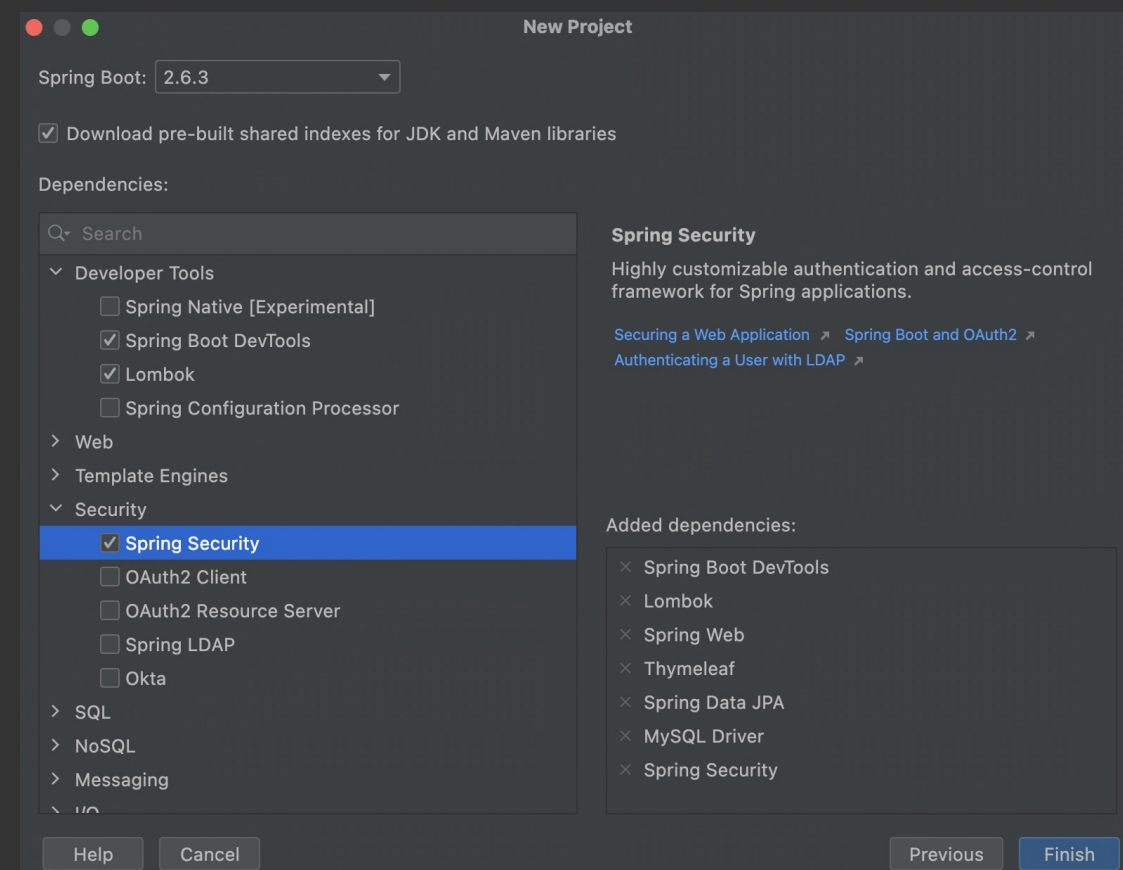
— 업로드 하는 파일의 확장자 제한

- 1). html 파일 : input type = file 에서 accept 속성을 이용
 - 지정하지 않으면 모든 종류의 파일 업로드 가능

- 2). Server 의 Controller 에서 처리



OAuth2 : 로그인 표준안



build.gradle 에서 의존성을 추가

- thymeleaf 에서 시간 포맷을 사용하는 것
- Security 의 내용을 사용하기 위한 의존성

build.gradle

```
plugins {  
    id 'org.springframework.boot' version '2.6.3'  
    id 'io.spring.dependency-management' version '1.0.11.RELEASE'  
    id 'java'  
    id 'war'  
}  
  
group = 'com.sin'  
version = '0.0.1-SNAPSHOT'  
sourceCompatibility = '11'  
  
configurations {  
    compileOnly {  
        extendsFrom annotationProcessor  
    }  
}  
  
repositories {  
    mavenCentral()  
}  
  
dependencies {  
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'  
    implementation 'org.springframework.boot:spring-boot-starter-security'  
    implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'  
    implementation 'org.springframework.boot:spring-boot-starter-web'  
    implementation 'org.thymeleaf.extras:thymeleaf-extras-springsecurity5'  
    compileOnly 'org.projectlombok:lombok'  
    developmentOnly 'org.springframework.boot:spring-boot-devtools'  
    runtimeOnly 'mysql:mysql-connector-java'  
    annotationProcessor 'org.projectlombok:lombok'  
    providedRuntime 'org.springframework.boot:spring-boot-starter-tomcat'  
    testImplementation 'org.springframework.boot:spring-boot-starter-test'  
    testImplementation 'org.springframework.security:spring-security-test'  
  
    implementation group: 'org.thymeleaf.extras', name: 'thymeleaf-extras-java8time'  
    implementation group: 'org.thymeleaf.extras', name: 'thymeleaf-extras-springsecurity5'  
}  
  
tasks.named('test') {  
    useJUnitPlatform()  
}
```

application.properties

```
server.port=2002
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.url=jdbc:mysql://localhost:3306/SpringTest
spring.datasource.username=singsiuk
spring.datasource.password=ssw0304

spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.show-sql=true

spring.thymeleaf.cache=false

spring.servlet.multipart.enabled=true
spring.servlet.multipart.location=/Users/adam/Documents/data
spring.servlet.multipart.max-request-size=30MB
spring.servlet.multipart.max-file-size=10MB

kr.co.adamsoft.upload.path=/Users/adam/Documents/data

logging.level.org.springframework.security.web=debug
logging.level.kr.co.adamsoft.security=debug
```



log 레벨 설정

spring security 에서 로그를 출력할 때 레벨 설정

실행클래스의 Annotation 을 추가

ClubApplication.java

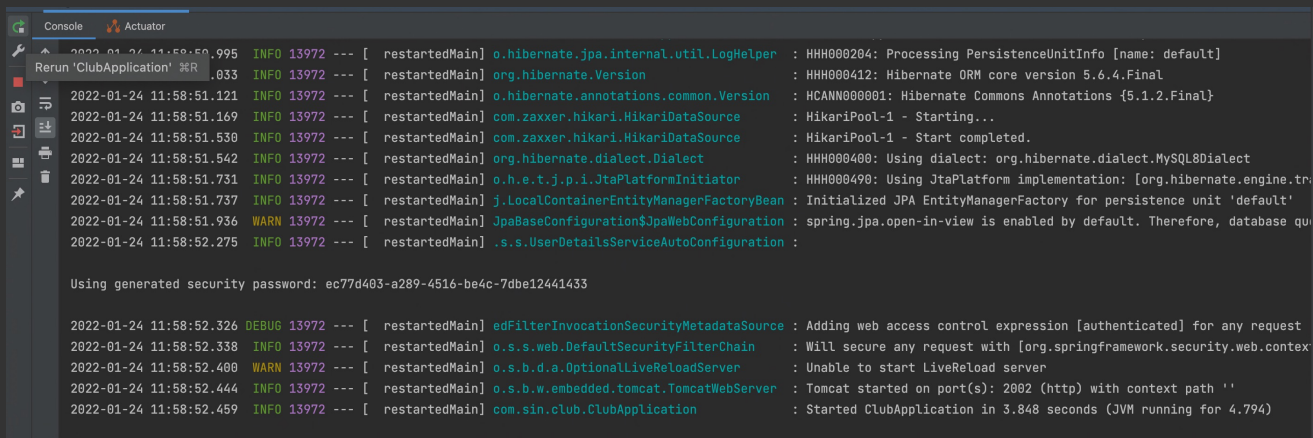
```
package com.sin.club;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

@SpringBootApplication
@EnableJpaAuditing
public class ClubApplication {

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

Project 실행



```
2022-01-24 11:58:50.995 INFO 13972 --- [ restartedMain] o.h.j.p.i.u.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
Rerun 'ClubApplication' #R...033 INFO 13972 --- [ restartedMain] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.6.4.Final
2022-01-24 11:58:51.121 INFO 13972 --- [ restartedMain] o.h.a.c.Version : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2022-01-24 11:58:51.169 INFO 13972 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2022-01-24 11:58:51.530 INFO 13972 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2022-01-24 11:58:51.542 INFO 13972 --- [ restartedMain] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.MySQL8Dialect
2022-01-24 11:58:51.731 INFO 13972 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2022-01-24 11:58:51.737 INFO 13972 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2022-01-24 11:58:51.936 WARN 13972 --- [ restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may only be executed within the application context.
2022-01-24 11:58:52.275 INFO 13972 --- [ restartedMain] .s.s.UserDetailsServiceAutoConfiguration :

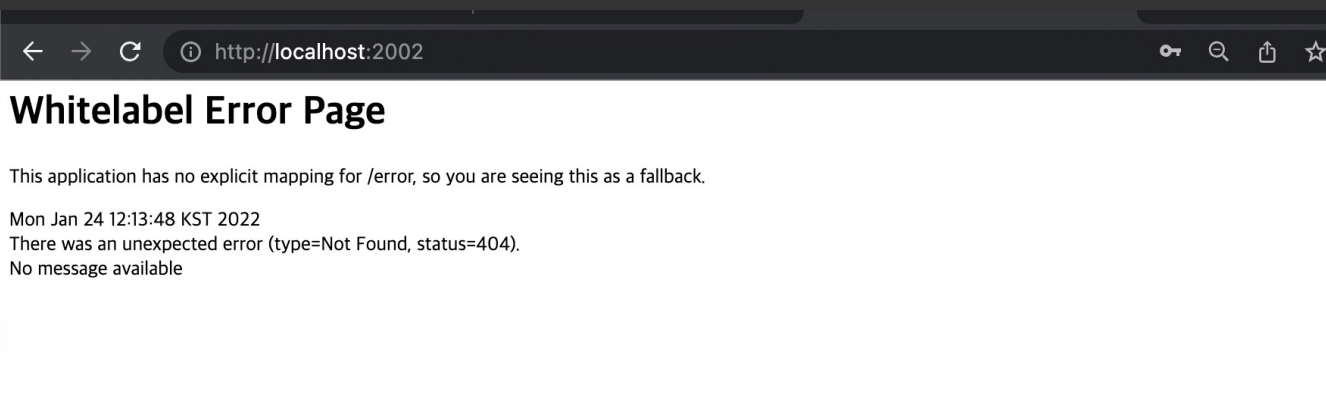
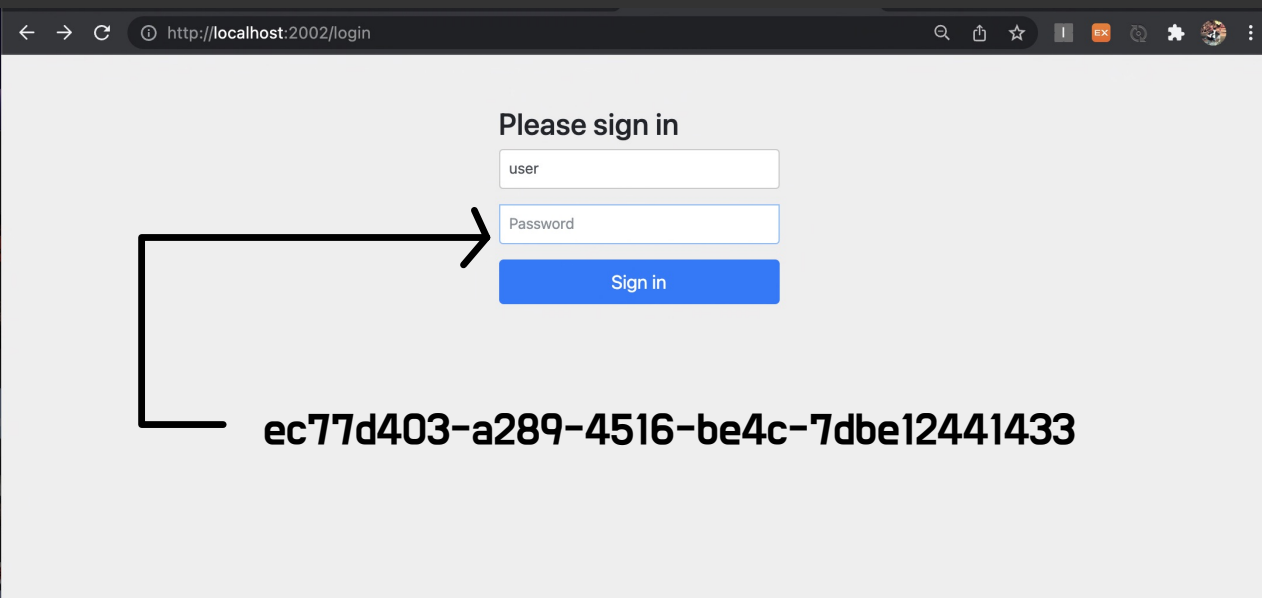
Using generated security password: ec77d403-a289-4516-be4c-7dbe12441433

2022-01-24 11:58:52.326 DEBUG 13972 --- [ restartedMain] edFilterInvocationSecurityMetadataSource : Adding web access control expression [authenticated] for any request
2022-01-24 11:58:52.338 INFO 13972 --- [ restartedMain] o.s.s.web.DefaultSecurityFilterChain : Will secure any request with [org.springframework.security.web.context.request.async.WebAsyncManagerIntegrationFilter, org.springframework.security.web.csrf.CsrfFilter, org.springframework.security.web.header.processor.HeaderWriterFilter, org.springframework.security.web.savedrequest.RequestCacheAwareFilter, org.springframework.security.web.servletapi.SecurityContextHolderAwareRequestFilter, org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter, org.springframework.security.web.authentication.logout.LogoutFilter]
2022-01-24 11:58:52.400 WARN 13972 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : Unable to start LiveReload server
2022-01-24 11:58:52.444 INFO 13972 --- [ restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 2002 (http) with context path ''
2022-01-24 11:58:52.459 INFO 13972 --- [ restartedMain] com.sin.club.ClubApplication : Started ClubApplication in 3.848 seconds (JVM running for 4.794)
```

Using generated security password: ec77d403-a289-4516-be4c-7dbe12441433

Spring boot Security 가 기본적으로 제공하는 user 계정의 password

application을 실행 시킬 때 마다 바뀔

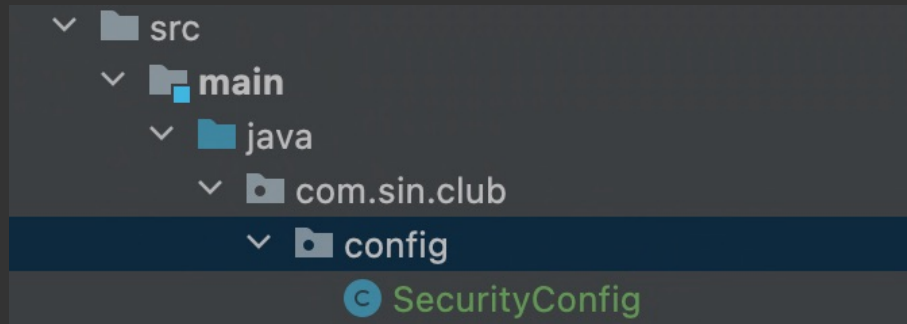


Spring Boot 설정 클래스

——> Web SercurityConfigure Adapter class 로 부터 상속

설정 class 생성 시 상단에 @Configuration 추가

기본package 에 config. securityconfig



SecurityConfig.java

```
package com.sin.club.config;

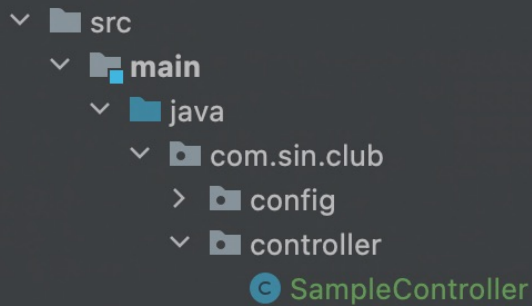
import lombok.extern.log4j.Log4j2;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration
@Log4j2
public class SecurityConfig extends WebSecurityConfigurerAdapter {
}
```

화면 이동 과 인증 설정

Controller class 생성

기본package. controller . SampleController



SampleController.java

```
package com.sin.club.controller;

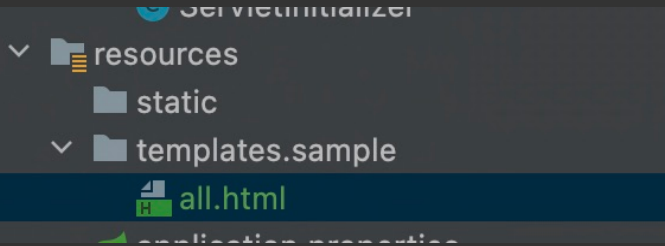
import lombok.extern.log4j.Log4j2;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestMapping;

@Controller
@Log4j2
// 공통 URL 설정
@RequestMapping("/sample")
public class SampleController {
    // 누구나 접속할 수 있는 page 생성
    @GetMapping("/all")
    public void exAll(){
        log.info("all user can access");
    }

    @GetMapping("/member")
    public void onlyMember(){
        log.info("Member Only");
    }
    @GetMapping("/admin")
    public void onlyAdmin(){
        log.info("Admin Only");
    }
}
```

templates 에 view 를 설정

sample directory 생성



all.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>All Access</title>
</head>
<body>
<h1>all User can access in here</h1>
</body>
</html>
```

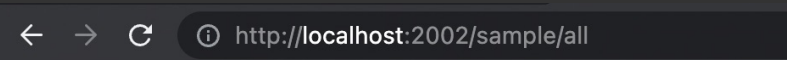
member.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>only member</title>
</head>
<body>
  <h1> only member who login our page </h1>
</body>
</html>
```

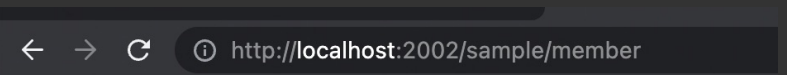
admin.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>admin</title>
</head>
<body>
  <h1>only Supervisor in here</h1>

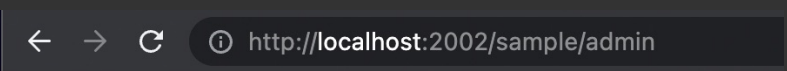
</body>
</html>
```



all User can access in here



only member who login our page



only Supervisor in here

Authentication(인증)

: 자신을 증명하는 것. 무언가 자신을 증명할 만한 자료를 제시하는 것

Authorization(인가, 권한 부여)

: 남에 의해서 부여된 자격

필터의 핵심적인 동작은 AuthenticationManager를 통해서
인증(Authentication)이라 는 타입의 객체로 작업

필터 중에서 UsernamePasswordAuthenticationFilter 클래스 코드 중 일부

```
String username = obtainUsername(request);
username = (username != null) ? username :
username = username.trim();
String password = obtainPassword(request);
password = (password != null) ? password : "";
UsernamePasswordAuthenticationToken authRequest = new
UsernamePasswordAuthenticationToken(username, password);
// Allow subclasses to set the "details" property
setDetails(request, authRequest);
return this.getAuthenticationManager().authenticate(authRequest);
```

Roles(권한에 대한 정보가 안에 들어있다)

PasswordEncoder : 비밀번호 암호화

— > 복호화는 불가능

(Spring에서는 기본적으로 bcrypt 라는 함수를 제공해 준다)

Spring security 설정 클래스에

Bean 을 생성하는 코드 작성

SecurityConfig.java

```
package com.sin.club.config;

import lombok.extern.log4j.Log4j2;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@Log4j2
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean
    public PasswordEncoder passwordEncoder(){
        // 암호화해서 평문과 비교는 할 수 있다.
        // 그러나 복호화를 할 수 없는 클래스의
        // 인스턴스를 리턴해서 생성
        return new BCryptPasswordEncoder();
    }
}
```

TEST

```
test
├── java
│   └── com.sin.club
│       ├── ClubApplicationTests
│       └── SpringPasswordTest
```

SpringPasswordTest.java

```
package com.sin.club;

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

@SpringBootTest
public class SpringPasswordTest {
    @Autowired
    private PasswordEncoder passwordEncoder;

    @Test
    public void testEncoding(){
        String password = "pickchu";
        //암호화
        String enPw=passwordEncoder.encode(password);

        // 출력
        System.out.println("Encoding 된 피카츄 : "+enPw);

        // 평문과 비교// 로그인 할 때는 이런 형태로 비교
        System.out.println("비교 : "+ passwordEncoder.matches(password,enPw));
    }
}
```

RESULT

```
2022-01-24 14:42:49.344 INFO 2921 --- [Test worker] .S.S.USERDETAILSServiceAutocor

Using generated security password: a97fef75-d55b-4986-a583-e41f5edc2b2a

2022-01-24 14:42:49.489 DEBUG 2921 --- [Test worker] edFilterInvocationSecurityMeta
2022-01-24 14:42:49.520 INFO 2921 --- [Test worker] o.s.s.web.DefaultSecurityFilter
2022-01-24 14:42:49.676 INFO 2921 --- [Test worker] com.sin.club.SpringPasswordTest
Encoding 된 피카츄 : $2a$10$EDfYFSUlsWkWiQF/0KXk.pm3TkiurL8kc148FLGgJ0TeCe6op6EK
비교 : true
```

Encoding 된 피카츄 : \$2a\$10\$EDfYFSUlsWkWiQF/
OKXk.pm3TkiurL8kc148FLGgJ0TeCe6op6EK
비교 : true

In Memory DB

: 데이터 주기억장치에 저장시키는 방법

— 실행 속도가 빨라짐

회원에 수가 많아지면 많은 데이터에서
데이터를 찾아서 로그인을 해야함으로 속도가 느려짐

해결방법

1. 테이블 분할

: 자주 로그인 하는 유저와 그렇지 않은 유저를 분할해서 다른 테이블에 저장해서 해결

2. 정말 자주 로그인 하는 유저라면

: Memory Database를 이용

인가(접근 권한)에 대한 설정

—> `HttpSecurity`를 parameter로 받는 `configure`라는 method를 Overriding 할 수 있고 annotation 으로도 가능하다

`SecurityConfigure.java` 에서 method Override —————

SecurityConfigure.java

```
package com.sin.club.config;

import lombok.extern.log4j.Log4j2;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

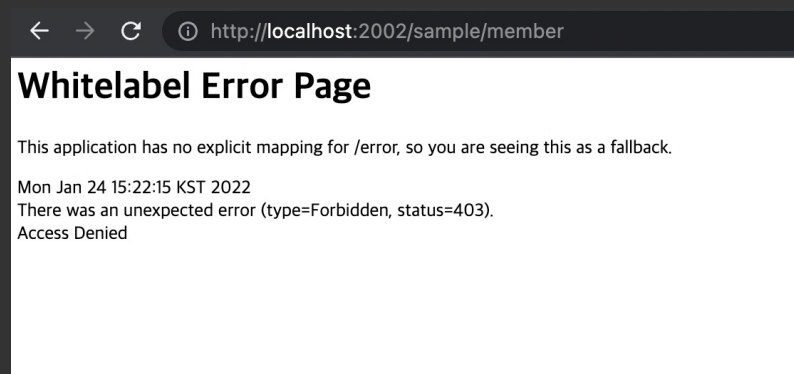
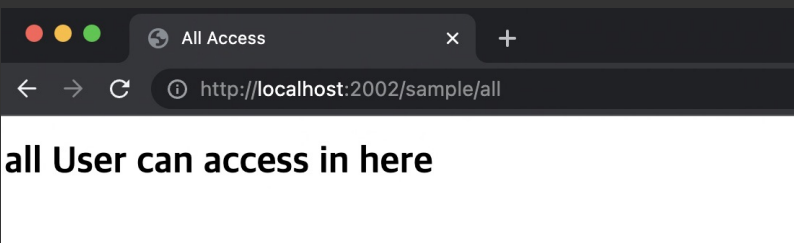
@Configuration
@Log4j2
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean
    public PasswordEncoder passwordEncoder(){
        // 암호화해서 평문과 비교는 할 수 있다.
        // 그러나 복호화를 할 수 없는 클래스의
        // 인스턴스를 리턴해서 생성
        return new BCryptPasswordEncoder();
    }

    // 설정하는 method
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception{
        // in memory 에 유저 생성
        auth.inMemoryAuthentication()
            .withUser("user1")
            .password("$2a$10$EDfYFSUlsWkWiQF/0KXk.pM3TkiurL8kc148FLGgJ0TeCe6op6EK")
            .roles("USER");
    }

    // 인가 설정 Override
    @Override
    protected void configure(HttpSecurity http) throws Exception{
        //sample / all 은 로그인 여부와 상관없이 접근 가능
        // sample/member 는 USER 권한이 있어야만 접근 가능
        http.authorizeRequests()
            .antMatchers("/sample/all").permitAll()
            .antMatchers("/sample/member").hasRole("USER");
    }
}
```

RESULT



Spring Security 설정파일의 configure method 에서 ——> Parameter 가 formLogin() 을 호출하도록 하면 권한이 없는 경우 로그인 page 로 이동한다.

SecurityConfig.java

```
package com.sin.club.config;

import lombok.extern.log4j.Log4j2;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@Log4j2
public class SecurityConfig extends WebSecurityConfigurerAdapter {

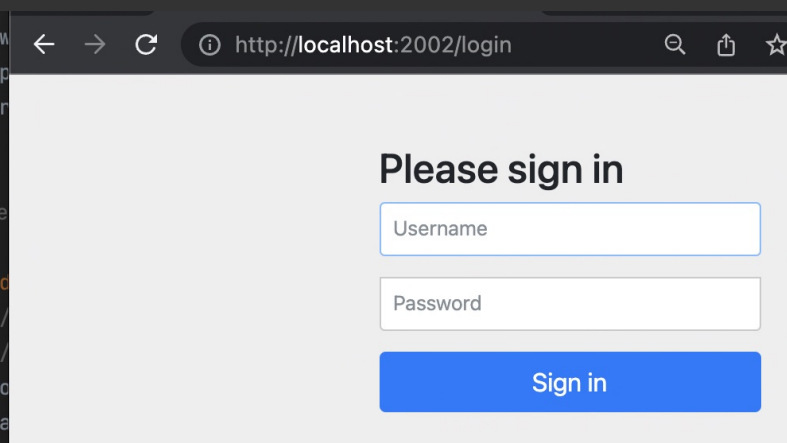
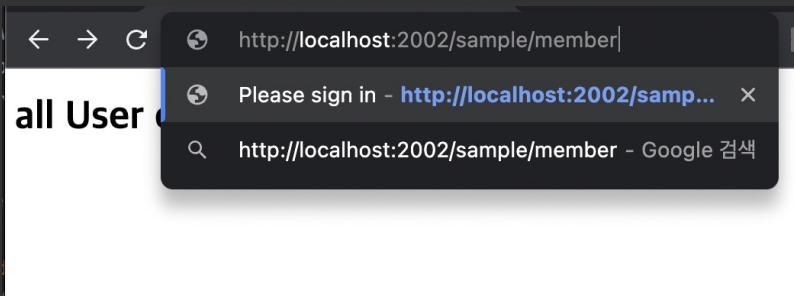
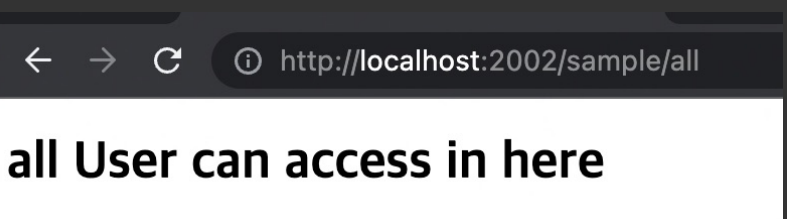
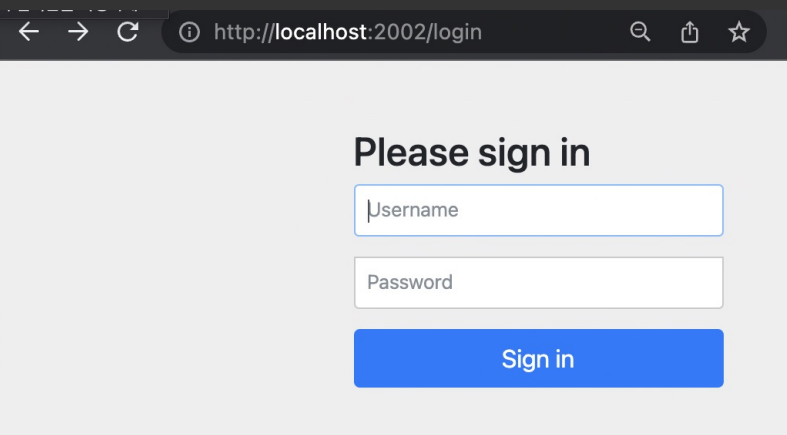
    @Bean
    public PasswordEncoder passwordEncoder(){
        // 암호화해서 평문과 비교는 할 수 있다.
        // 그러나 복호화를 할 수 없는 클래스의
        // 인스턴스를 리턴해서 생성
        return new BCryptPasswordEncoder();
    }

    // 설정하는 method
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception{
        // in memory 에 유저 생성
        auth.inMemoryAuthentication()
            .withUser("user1")
            .password("$2a$10$EDfYFSUlsWkWiQF/0KXk.pm3TkiurL8kc148FLGgJ0TeCe6op6EK")
            .roles("USER");
    }

    // 인가 설정 Override
    @Override
    protected void configure(HttpSecurity http) throws Exception{
        //sample / all 은 로그인 여부와 상관없이 접근 가능
        // sample/member 는 USER 권한이 있어야만 접근 가능
        http.authorizeRequests()
            .antMatchers("/sample/all").permitAll()
            .antMatchers("/sample/member").hasRole("USER");

        // 권한이 없는 경우 로그인 page 로 이동
        http.formLogin();
    }
}
```

RESULT



CUSTOM LOGING PAGE — — — —

사용자가 직접 작성한 login page 를 사용하는 것

→ formLogin() method 를 호출 한 후

→ loginPage("로그인 Page URL") 을 호출하면

: 사용자가 작성한 로그인 페이지를 출력할 수 있고

loginProcessUrl("로그인 처리 URL") 을 호출하면 로그인 처리

URL 을 직접 설정할 수 있음

이 경우 ID 에 해당하는 name 은 username 이다 .

Spring Security 에서는 username 이 Id 이다.

SecurityController.java

Config 의 method 수정

```
package com.sin.club.config;

import lombok.extern.log4j.Log4j2;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@Log4j2
public class SecurityConfig extends WebSecurityConfigurerAdapter {

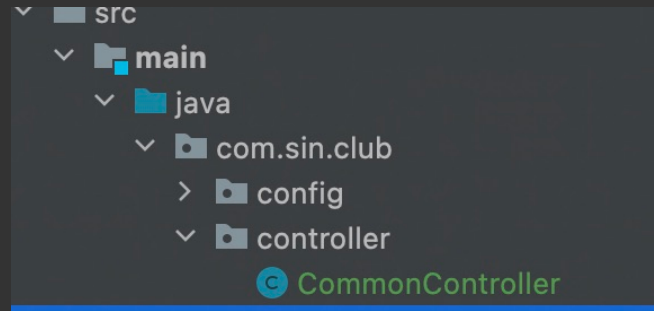
    @Bean
    public PasswordEncoder passwordEncoder(){
        // 암호화해서 평문과 비교는 할 수 있다.
        // 그러나 복호화를 할 수 없는 클래스의
        // 인스턴스를 리턴해서 생성
        return new BCryptPasswordEncoder();
    }

    // 설정하는 method
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception{
        // in memory 에 유저 생성
        auth.inMemoryAuthentication()
            .withUser("user1")
            .password("$2a$10$EDfYFSUlsWkWiQF/0KXk.pm3TkiurL8kc148FLGgJ0TeCe6op6EK")
            .roles("USER");
    }

    // 인가 설정 Override
    @Override
    protected void configure(HttpSecurity http)throws Exception{
        //sample / all 은 로그인 여부와 상관없이 접근 가능
        // sample/member 는 USER 권한이 있어야만 접근 가능
        http.authorizeRequests()
            .antMatchers("/sample/all").permitAll()
            .antMatchers("/sample/member").hasRole("USER");

        // 권한이 없는 경우 로그인 page 로 이동
        // 로그인 요청 URL 은 Custmlogin 이고 URL 처리는 login
        http.formLogin()
            .loginPage("/customlogin")
            .loginProcessingUrl("/login");
    }
}
```

Controller package 에 PageController 클래스를 생성하고 customlogin 요청을 처리하는 method 생성 (controller. CommonController)



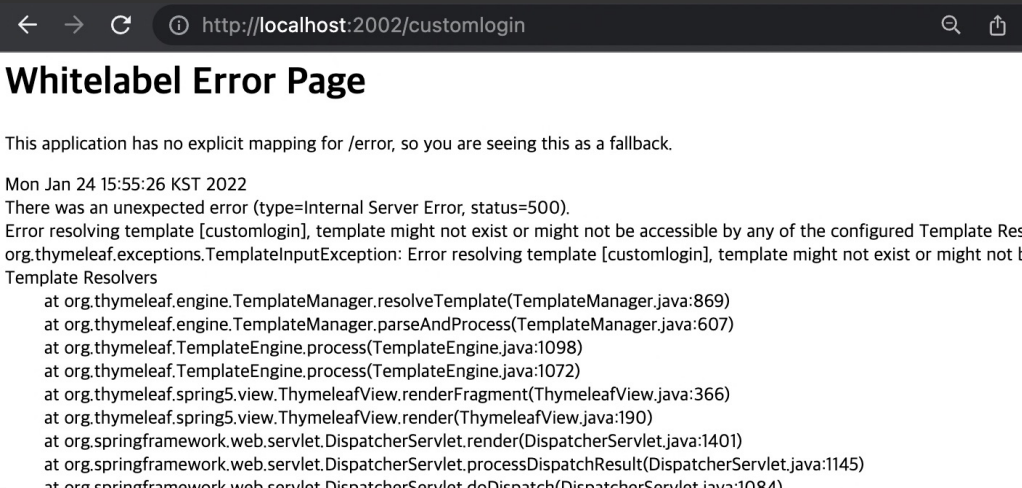
CommonController.java

```
package com.sin.club.controller;

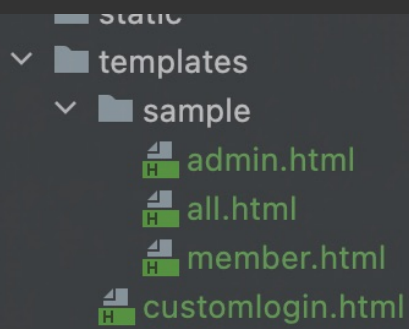
import lombok.extern.log4j.Log4j2;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
@Log4j2
public class CommonController {
    @GetMapping("/customlogin")
    public void customlogin(){
        log.info("----- Customer Login Page----- ");
    }
}
```

RESULT



costomlogin 에 대해서 생성하지 않았기 때문에
에러 발생



costomlogin.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>customlogin</title>
</head>
<body>
<h2>this is a CustomLogin Page</h2>
</body>
</html>
```

Result

A web browser address bar with navigation icons (back, forward, refresh) on the left. The address is 'http://localhost:2002/customlogin'.

this is a CustomLogin Page

CSRF(Cross Site Request Forgery. 크로스 사이트 요청 위조)

—— CSRF 공격

: SERVER 에서 요청을 해석하고 처리할 때
어떤 출처에서 호출이 진행되었는지 확인하지 않기 때문에
발생하는 허접을 노리는 공격 방식

- 다른 User 가 사용자의 Grade(등급) 을 변경하는 URI를 알고
필요한 Parameter를 안다면
직접 로그인 하지 않고
img 태그나 form 태그를 이용해서
URI 와 parameter 를 기록해둔 상태에서 관리자가 링크를 클릭하면
공격자가 관리자 등급으로 변경

해결책

1. 사용자의 요청에 대한 출처를 의미하는 referrer header를 체크
2. 모든 요청을 PUT 이나 DELETE 또는 POST 를 이용
3. CSRF TOKEN 을 이용해서 해결
——> Spring 에서는 이 방식으로 해결
(Server 가 클라이언트에게 데이터를 전송시 CSRF TOKEN 을 같이 전송)

: 이 토큰의 값을 Server 에 저장해 두었다가
클라이언트가 요청을 전송하면
일치하는 토큰이 있을 경우에만 처리한다.

—— Spring 은 GET 을 제외한 모든 요청 방식에 대해서 CSRF TOKEN을 이용해서 처리

CSRF 를 비활성화 한다(보안상 권장이지 되기도 한다)

- Server 의 데이터를 수정하지 않는 REST API 서버의 경우
매번 CSRF TOKEN 을 확인하는 것은 번거롭기 때문에
이를 확인하지 않도록 설정하기도 한다 .

(CSRF 가 위험한 경우 —> 관리자의 권한으로 데이터를 수정하는 곳에서는 위험)

- 그 외의 곳은 비활성화 시킨다

SecurityConfigure 클래스에 method 추가

SecurityConfig.java

```
package com.sin.club.config;

import lombok.extern.log4j.Log4j2;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@Log4j2
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean
    public PasswordEncoder passwordEncoder(){
        // 암호화해서 평문과 비교는 할 수 있다.
        // 그러나 복호화를 할 수 없는 클래스의
        // 인스턴스를 리턴해서 생성
        return new BCryptPasswordEncoder();
    }

    // 설정하는 method
    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception{
        // in memory 에 유저 생성
        auth.inMemoryAuthentication()
            .withUser("user1")
            .password("$2a$10$EDfYFSUlsWkWiQF/0KXk.pm3TkiurL8kc148FLGgJ0TeCe6op6EK")
            .roles("USER");
    }

    // 인가 설정 Override
    @Override
    protected void configure(HttpSecurity http) throws Exception{
        //sample / all 은 로그인 여부와 상관없이 접근 가능
        // sample/member 는 USER 권한이 있어야만 접근 가능
        http.authorizeRequests()
            .antMatchers("/sample/all").permitAll()
            .antMatchers("/sample/member").hasRole("USER");

        // 권한이 없는 경우 로그인 page 로 이동
        // 로그인 요청 URL 은 Custmlogin 이고 URL 처리는 login
        http.formLogin()
            .loginPage("/customlogin")
            .loginProcessingUrl("/login");

        // csrf 토큰을 비교하는 작업을 수행하지 않음
        http.csrf().disable();
    }
}
```

logout 설정

: logout method 만 호출하면 설정 된다.

로그아웃이 되면

1. session
2. login 정보
3. cookie 삭제

— > logout 이후에는 login page 로 이동

— logoutUrl method 를 이용해서 logout url 을 변경할 수 있고

— deleteCookies 를 이용해서 쿠키 삭제 가능

— invalidateHttpSession 을 호출해서 Session 을 초기화 할 수 있다 .

JPA 를 이용한 LOGIN 처리

— Entity 를 설계

데이터 삽입 날짜와 수정날짜를 갖는 BaseEntity 를 생성

- 회원정보 Entity : ClubMember
(email: 아이디역할)
(password : 복호화가 불가능한 암호화를 적용)
- 이름 (nick name)
- 소셜 가입 여부 (google 로 로그인 했는지 확인)
- regdate
- modDate

- 권한 : clubMemberRole

User, Manager, Admin 으로 구분

— 우리꺼 사용

BaseEntity.java

```
package com.sin.club;

import lombok.Getter;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.data.jpa.repository.config.EnableJpaAuditing;

@SpringBootApplication
@EnableJpaAuditing
@Getter
public class ClubApplication {

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

}
```

권한의 종류를 상수로 갖는 enum 을 생성

entity.ClubMemberRole

ClubMemberRole.java(enum)

```
package com.sin.club.entity;  
  
public enum ClubMemberRole {  
    USER, MANAGER, ADMIN  
}
```

ClubMember.java

```
package com.sin.club.entity;

import lombok.*;
import org.hibernate.annotations.DynamicInsert;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import javax.persistence.*;
import java.time.LocalDate;
import java.util.HashSet;
import java.util.Set;

@Entity
@Builder
@AllArgsConstructor
@NoArgsConstructor
@Getter
@ToString
public class ClubMember extends BaseEntity{
    @Id
    private String email;
    private String password;
    private String name;
    private boolean fromSocial;
    //권한을 하나만 가지는 경우
    //private ClubMemberRole rols;

    //권한을 여러개 가질 수 있는 경우
    @Builder.Default
    @ElementCollection(fetch = FetchType.LAZY)
    private Set<ClubMemberRole> roleSet = new HashSet<>();

    //권한을 추가하는 메서드
    public void addMemberRole(ClubMemberRole clubMemberRole){
        roleSet.add(clubMemberRole);
    }
}
```

ClubMemberEntity를 사용하는 Repository 를 생성

club_member 와 club_member_role_set 생성

```
package com.sin.club.repository;

import com.sin.club.entity.ClubMember;
import org.springframework.data.jpa.repository.EntityGraph;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;

import java.util.Optional;

public interface ClubMemberRepository extends JpaRepository<ClubMember,String> {
    //이메일을 가지고 조회하는 메서드
    @EntityGraph(attributePaths = {"roleSet"}, type = EntityGraph.EntityGraphType.LOAD)
    @Query("select m from ClubMember m where m.fromSocial = :social and m.email=:email")
    Optional<ClubMember> findByEmail(String email, boolean social);
}
```

TEST

```
package com.sin.club;

import com.sin.club.entity.ClubMember;
import com.sin.club.entity.ClubMemberRole;
import com.sin.club.repository.ClubMemberRepository;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.security.crypto.password.PasswordEncoder;

import java.time.LocalDate;
import java.util.Optional;
import java.util.stream.IntStream;

@SpringBootTest
public class ClubMemberTest {
    @Autowired
    private ClubMemberRepository clubMemberRepository;

    @Autowired
    private PasswordEncoder passwordEncoder;

    @Test
    public void insertDummitData(){
        for(int i=1; i<=100; i++){
            ClubMember clubMember = ClubMember.builder().email("user"+ i +"@gmail.com").name("사용자" +
i).fromSocial(false).password(passwordEncoder.encode("1234")).build();
            clubMember.addMemberRole(ClubMemberRole.USER);
            if(i>80){
                clubMember.addMemberRole(ClubMemberRole.MANAGER);
            }
            if(i>90){
                clubMember.addMemberRole(ClubMemberRole.ADMIN);
            }
            clubMemberRepository.save(clubMember);
        }
    }
    @Test
    public void testEmail(){
        Optional<ClubMember> result = clubMemberRepository.findByEmail("user88@gmail.com",false);
        System.out.println(result.get());
    }
}
```

TEST -result

```
@Test
public void insertDummitData(){
    for(int i=1; i<=100; i++){
        ClubMember clubMember = ClubMember.builder().email("user"+ i +"@gmail.com").name("사용자" +
i).fromSocial(false).password(passwordEncoder.encode("1234")).build();
        clubMember.addMemberRole(ClubMemberRole.USER);
        if(i>80){
            clubMember.addMemberRole(ClubMemberRole.MANAGER);
        }
        if(i>90){
            clubMember.addMemberRole(ClubMemberRole.ADMIN);
        }
        clubMemberRepository.save(clubMember);
    }
}
```

select * from club_member_role_set cmrs | Enter a

Ctrl+click to open SQL console

123 role_set

122	user98@gmail.com	2
123	user98@gmail.com	1
124	user98@gmail.com	0
125	user99@gmail.com	2
126	user99@gmail.com	1
127	user99@gmail.com	0
128	user100@gmail.com	2
129	user100@gmail.com	1
130	user100@gmail.com	0

select * from club_member | Enter a SQL expression to filter results (use Ctrl+Space)

	email	moddate	regdate	from_social	name	password
50	user53@gmail.com	2022-01-24 17:35:14.375630000	2022-01-24 17:35:14.375630000	0	사용자53	\$2a\$10\$eaj.G/Pg7XR779OKgNxbgWuqkLLsOPD1qPmLwIA2sM
51	user54@gmail.com	2022-01-24 17:35:14.467040000	2022-01-24 17:35:14.467040000	0	사용자54	\$2a\$10\$1CED.tWbhEaPLcJ8.WxH7OXLwXdFxmfmfJuoXLZlrHG:
52	user55@gmail.com	2022-01-24 17:35:14.560278000	2022-01-24 17:35:14.560278000	0	사용자55	\$2a\$10\$uCSTt0.EhkWcVikAOEL.zi.ZbVAEdwKVDWYpxDI3cns.t
53	user56@gmail.com	2022-01-24 17:35:14.652225000	2022-01-24 17:35:14.652225000	0	사용자56	\$2a\$10\$YojGde1DwfJi4Fd6T.6vperYF0H0Ek4INNHRvnlg8aY0x
54	user57@gmail.com	2022-01-24 17:35:14.752741000	2022-01-24 17:35:14.752741000	0	사용자57	\$2a\$10\$F5ojL37/N9slnFgOP4cEOwyNWa4JUPRdgZXOnrsRM
55	user58@gmail.com	2022-01-24 17:35:14.842724000	2022-01-24 17:35:14.842724000	0	사용자58	\$2a\$10\$.Hm8PawMr2OIdp.Wwv3QE06cub3PLNcODEj5EjwbS
56	user59@gmail.com	2022-01-24 17:35:14.936023000	2022-01-24 17:35:14.936023000	0	사용자59	\$2a\$10\$Xij2SLnSCWVsrhgBBqKKpeq3hetECgHhbGH0rjOuhk5
57	user6@gmail.com	2022-01-24 17:35:09.545461000	2022-01-24 17:35:09.545461000	0	사용자6	\$2a\$10\$Zt3Y41Xhb4g0kAt0zOxIYOIAZJdlg5q46hNFr7/L78OI5
58	user60@gmail.com	2022-01-24 17:35:15.026903000	2022-01-24 17:35:15.026903000	0	사용자60	\$2a\$10\$4K4FoPRsIF7ixX0lPtz8pO7lQldYOhgn1qYzp5u40oP/1t
59	user61@gmail.com	2022-01-24 17:35:15.125122000	2022-01-24 17:35:15.125122000	0	사용자61	\$2a\$10\$jt1nJVWsX9G7cOJKx7peMeNsnasNEgplVuhnrrTikFCst
60	user62@gmail.com	2022-01-24 17:35:15.218400000	2022-01-24 17:35:15.218400000	0	사용자62	\$2a\$10\$6ZyCnUOyCThihnm5AjJ1uqAPTJR1om2naR8fDJcPA5

```
@Test
public void testEmail(){
    Optional<ClubMember> result = clubMemberRepository.findByEmail("user88@gmail.com",false);
    System.out.println(result.get());
}
```

```
club_member_role_set roleset1_
on clubmember0_.email=roleset1_.club_member_email
where
clubmember0_.from_social=?
and clubmember0_.email=?
ClubMember(email=user88@gmail.com, password=$2a$10$309Ij3R.i8r5GK5Wa0KE6eIP2qM8YtdizW/KbQ3V62EQwaBBEoKq, name=사용자88, fr
```

시큐리티를 위한 UserDetailsService

— > 하나의 method 만 소유

: loadUserByUsername

- 이 method 는 username 을 이용해서 회원정보를 가져온다 .
리턴 타입은 UserDetails.

UserDetails 의 method

- getAuthorities() : 권한 정보

- getPassword() : 비밀번호

- getUsername : 아이디 정보

구현 방법

: 기존 DTO 클래스에 UserDetails interface 를 구현하는 방법
(DTO 와 같은 별도의 클래스를 구성하고 이를 활용)

— DTO Package 에 ClubAuthMember DTO 클래스를 생성

ClubAuthMember.java

```
package com.sin.club.dto;

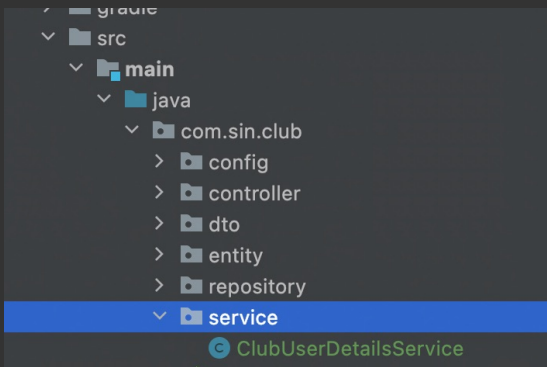
import lombok.*;
import lombok.extern.log4j.Log4j2;
import org.springframework.security.core.GrantedAuthority;
import org.springframework.security.core.userdetails.User;

import java.util.Collection;

@Log4j2
@Getter
@Setter
@ToString
public class ClubAuthMember extends User {
    private String email;
    private String name;
    private boolean fromSocial;

    public ClubAuthMember(String username , String password ,
                           Collection<? extends GrantedAuthority> authorities){
        //상위 클래스의 생성자 호출
        super(username,password,authorities);
        this.email = username;
        this.fromSocial= isFromSocial();
    }
}
```


service package 에 UserDetailsService 를 상속받는 Class 를 생성.



ClubUserDetailsService.java

```
package com.sin.club.service;

import com.sin.club.repository.ClubMemberRepository;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

@Log4j2
@Service
@RequiredArgsConstructor
public class ClubUserDetailsService implements UserDetailsService {
    private final ClubMemberRepository clubMemberRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        log.info("loadByUserName : "+username);
        return null;
    }
}
```

SecurityConfig.java

```
package com.sin.club.config;

import lombok.extern.log4j.Log4j2;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;

@Configuration
@Log4j2
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Bean
    public PasswordEncoder passwordEncoder(){
        // 암호화해서 평문과 비교는 할 수 있다.
        // 그러나 복호화를 할 수 없는 클래스의
        // 인스턴스를 리턴해서 생성
        return new BCryptPasswordEncoder();
    }

    // --- JPA 를 사용하면 주석처리
    // 설정하는 method
    /*@Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception{
        // in memory 에 유저 생성
        auth.inMemoryAuthentication()
            .withUser("user1")
            .password("$2a$10$EDfYFSUlsWkWiQF/0KXk.pm3TkiurL8kc148FLGgJ0TeCe6op6EK")
            .roles("USER");
    }*/

    // 인가 설정 Override
    @Override
    protected void configure(HttpSecurity http)throws Exception{
        //sample / all 은 로그인 여부와 상관없이 접근 가능
        // sample/member 는 USER 권한이 있어야만 접근 가능
        http.authorizeRequests()
            .antMatchers("/sample/all").permitAll()
            .antMatchers("/sample/member").hasRole("USER");

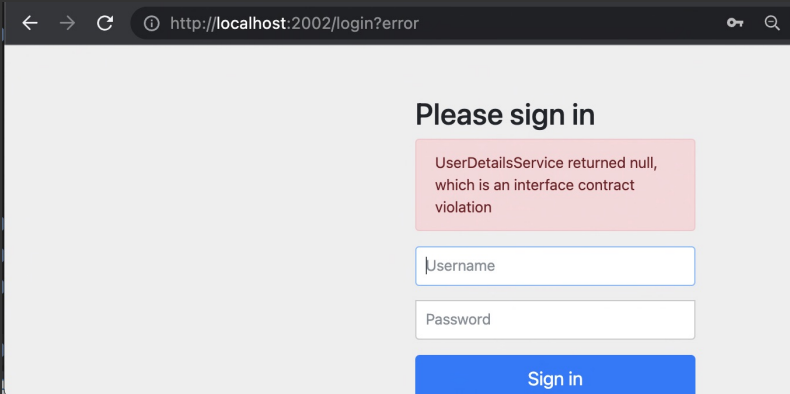
        // 권한이 없는 경우 로그인 page 로 이동
        // 로그인 요청 URL 은 Custmlogin 이고 URL 처리는 login
        // http.formLogin()
        //     .loginPage("/customlogin")
        //     .loginProcessingUrl("/login");

        http.formLogin();

        // csrf 토큰을 비교하는 작업을 수행하지 않음
        http.csrf().disable();
    }
}
```

Application 을 실행

———— UserDetailsService interface를 구현한 클래스의 bean이 있으면
이 bean 을 UserDetailsService로 인식해서 자동으로 사용



id: user1@gmail.com
pw : 1234

————> 로그인 되지 않음

```
er : Cleared SecurityContextHolder to complete request
   : Securing POST /login
er : Set SecurityContextHolder to empty SecurityContext
e  : loadByUsername : user1@gmail.com
er : An internal error occurred while trying to authenticate the user.
```

ClubUserDetailsService.java 를 수정

ClubUserDetailsService.java

```
package com.sin.club.service;

import com.sin.club.dto.ClubAuthMember;
import com.sin.club.entity.ClubMember;
import com.sin.club.repository.ClubMemberRepository;
import lombok.RequiredArgsConstructor;
import lombok.extern.log4j.Log4j2;
import org.springframework.security.core.authority.SimpleGrantedAuthority;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.core.userdetails.UsernameNotFoundException;
import org.springframework.stereotype.Service;

import java.util.stream.Collectors;

@Log4j2
@Service
@RequiredArgsConstructor
public class ClubUserDetailsService implements UserDetailsService {
    private final ClubMemberRepository clubMemberRepository;

    @Override
    public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {
        log.info("loadByUsername : "+username);
        // 데이터 베이스에서 username 에 해당하는 데이터 찾아오기
        ClubMember clubMember
            =clubMemberRepository.findByEmail(
                username, false).get();

        // 인증을 위한 class 의 instance를 생성
        ClubAuthMember clubAuthMember = new ClubAuthMember(
            clubMember.getEmail(),
            clubMember.getPassword(),
            clubMember.getRoleSet().stream()
                .map(role->new SimpleGrantedAuthority(
                    "ROLE_"+role.name()))
                .collect(Collectors.toSet())
        );
        clubAuthMember.setName(clubAuthMember.getName());
        clubAuthMember.setFromSocial(clubAuthMember.isFromSocial());
        log.info(clubMember);
        log.info(clubAuthMember);
        return clubAuthMember;
    }
}
```

OPERATE

tenorshare.kr

Please sign in

user1@gmail.com

....

Sign in

← → ↺ ⓘ http://localhost:2002/sample/member

only member who login our page

애플리케이션

메니페스트Service Workers저장용량

저장용량

로컬 스토리지세션 스토리지IndexedDB웹 SQL쿠키

http://localhost:2002

이름	값	Domain	P.	E.	크	Http..	Secure	S.	Priority
JSESSIONID	C6D52...	localhost	/	세..	42	✓			Medium

```
: ClubMember(email=user1@gmail.com, password=$2a$10$L56MTY2LT34VWEhEr7107u5S9/sH0sPDwLdyN5pUQD7sWM06V0WTO, name=사용자1, fromSocial=false, roleSet=[USER])
: ClubAuthMember(email=user1@gmail.com, name=null, fromSocial=false)
: Changed session id from EFAD98A607BF9A4EEE26A6FC602FB940
```