thymeleaf 사용하기

1. 의존성 및 설정 추가

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
<dependency>
    <groupId>org.thymeleaf.extras</groupId>
    <artifactId>thymeleaf-extras-springsecurity5</artifactId>
</dependency>
```

thymeleaf와 security를 html에서 사용하기 위한 의존성 추가!

```
spring.thymeleaf.prefix=classpath:templates/
spring.thymeleaf.check-template-location=true
spring.thymeleaf.suffix=.html
spring.thymeleaf.mode=HTML5
spring.thymeleaf.cache=false
spring.thymeleaf.order=0
```

```java
@Configuration
public class ThymeleafViewResolverConfig {

    @Value("${thymeleaf.cache}")
    private boolean isCache;

    @Bean
    public SpringResourceTemplateResolver templateResolver() {
        SpringResourceTemplateResolver  templateResolver = new SpringResourceTemplateResolver ();
        templateResolver.setPrefix("classpath:templates/");
        templateResolver.setCharacterEncoding("UTF-8");
        templateResolver.setSuffix(".html");
        templateResolver.setTemplateMode("LEGACYHTML5");
        templateResolver.setCacheable(isCache);
        return templateResolver;
    }

    @Bean
    public SpringTemplateEngine templateEngine(MessageSource messageSource) {
        SpringTemplateEngine templateEngine = new SpringTemplateEngine();
        templateEngine.setTemplateResolver(templateResolver());
        templateEngine.setTemplateEngineMessageSource(messageSource);
        templateEngine.addDialect(layoutDialect());

        return templateEngine;
    }

    @Bean
    public LayoutDialect layoutDialect() {
        return new LayoutDialect();
    }

    @Bean
    @Autowired
    public ViewResolver viewResolver(MessageSource messageSource) {
        ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();
        viewResolver.setTemplateEngine(templateEngine(messageSource));
        viewResolver.setCharacterEncoding("UTF-8");
        viewResolver.setOrder(0);
        return viewResolver;
    }
}
```

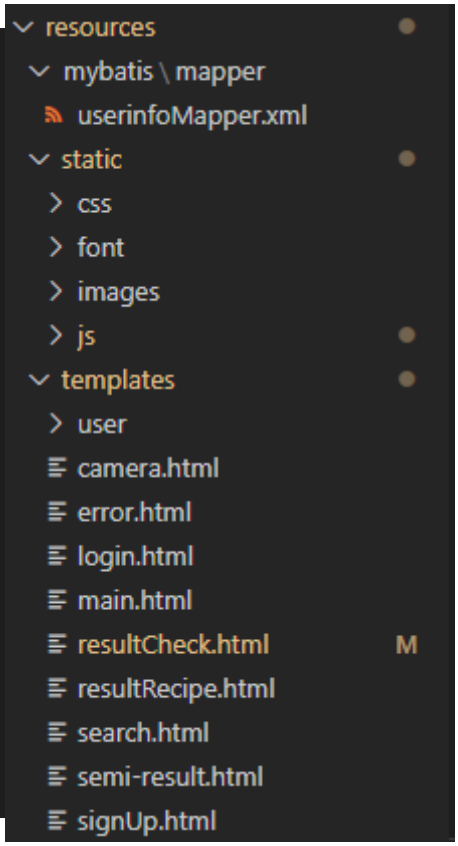

이와 같은 설정은 application.properties 또는 application.yml에 정의하여 설정을 변경할 수 있다.
또는 아래 사진과 같이 java파일로 설정할 수 있다.

다만! 설정을 하지 않아도 default값으로 설정이 되므로 문제는 없다.
필요한 사항만 변경을 하도록한다.

2. Controller작성 및 HTML 파일 경로

```java
@RequestMapping("/")
public String home(){
    return "main";
}
@RequestMapping("/search")
public String search(){
    return "search";
}
@RequestMapping("/guide")
public String guide(){
    return "guide";
}

@RequestMapping("/camera")
public String camera(){
    return "camera";
}
```



이와 같이 Controller에서 경로를 생략하고 입력할 경우

resource/templates경로로 자동 매핑된다.
이는 thymeleaf default설정에 의한 값이다. 변경이 필요하면 변경해도 좋다.

또한 css,font,images,js 파일은 여타 프로젝트와 같이 정적데이터를 저장하는 static경로에 저장해 사용하도록 한다.

3. HTML에서의 사용

HTML에서 spring security의 권한정보에 따라 다른 컨텐츠를 출력해야하는 경우가 있다. 예를들면 로그인 시 '로그인버튼 -> 로그아웃버튼' 으로 변경한다.

xmlns:sec="http://www.thymeleaf.org/extras/spring-security"

이를 HTML태그 내에 사용한다. spring security의 자원을 사용하겠다는 뜻이다.

```
<li sec:authorize="hasRole('ROLE_ADMIN')"><a th:href="@{/admin}">관리자 페이지</a></li>
//ROLE_USER 권한이 있는 경우만 표시됨
<li sec:authorize="hasRole('ROLE_USER')"><a th:href="@{/books}">책 목록</a></li>
//로그인이 되지 않은 경우 표시
<li sec:authorize="!isAuthenticated()"><a th:href="@{/users}">회원가입</a></li>
//로그인 된 경우의 username을 출력
<li sec:authorize="isAuthenticated()"><span sec:authentication="principal.username"></span></li>
//변수처럼 사용하고 싶은 경우
<a th:href="@{'/users/'+ ${#authentication.principal.id}}">마이페이지</a>
```

이러한 형식으로 security의 정보에 따라 View에 대한 권한 설정을 할 수 있다.

Made by 장경석