REST API 구현 & Swagger 적용

URL	
간단 설명	기존 Product 프로젝트에 REST API 적용 & Swagger Test
과제 주내용	Spring
날짜	@2021/05/03
문제 난이도	
문제 번호	
문제 출처	
복습 필요	☐
키워드	REST API Spring

!? Task

💯 기존 Product 프로젝트에 REST API를 적용시키고 Swagger 테스트를 진행해보자

1. 우선 pom.xml에 필요한 jar 파일 jackson-databind, Swagger-ui, Swagger2 jar 파일들을 가져오자

```xml
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>2.12.3</version>
    </dependency>

    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger2</artifactId>
        <version>2.4.0</version>
    </dependency>
    <dependency>
        <groupId>io.springfox</groupId>
        <artifactId>springfox-swagger-ui</artifactId>
        <version>2.4.0</version>
    </dependency>
```

pom.xml

2. servlet-context.xml 파일에 <resource> 태그에 swagger config 파일 Mapping 정보를 넣어주고 경로를 설정해준다.
    - swagger-ui.html 과 webjars의 위치를 설정해준다.

```xml
<resources mapping="/swagger-ui.html" location="classpath:/META-INF-resources/" />
<resources mapping="/webjars/**" location="classpath:/META-INF/resources/" />
```

servlet-context.xml

3. java 패키지에 com.ssafy.product.config 패키지를 생성한 다음 SwaggerConfig.java 파일을 복사해온다.
    - 세부 내용은 현재 프로젝트와 맞게 바꾸어준다.
    - groupName과 path 같은 것들을 수정해준다.

```java
@Configuration
@EnableSwagger2          //스웨거를 위한 설정
public class SwaggerConfig {
    private static final Logger logger = LoggerFactory.getLogger(SwaggerConfig.class);

    @Bean
    public Docket api() {
        logger.debug("swagger ready........{}", "준비중");
        return new Docket(DocumentationType.SWAGGER_2)
                .groupName("product")        //service할 project 이름
                .apiInfo(info())
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.ssafy.product.controller"))
                 // 서비스할 경로 phone으로 통일  서비스 구분은 요청 method로 구분
                .paths(PathSelectors.ant("/product/**"))
//                 //서비스할 경로를 통일하지 않고 경로명으로 구분해서 사용
//                .paths(PathSelectors.any())
                .build();
    }

    /** 프로젝트의 정보를 만들어 주는 함수 */
    private ApiInfo info() {
        return new ApiInfoBuilder().title("Product Management API")
                .description("제품 관리를 위한 <b>CRUD</b>")
                .license("Swagger 테스트!!! ")
                .version("3.0")
                .build();
    }
```

SwaggerConfig.java

- logger.debug를 사용하여 console 창에 출력하기 위해 resource 파일 안에 log4j.xml 파일에서 debug로 수정해준다.

log4j.xml

4. 기존의 Controller 들을 복사하여 RestController로 만들고 세부 내용을 코딩한다.
    - 우선 Annotation을 `@RestController`로 바꾸고 `@CrossOrigin` , `@Api` 등을 사용한다.
    - 비동기 방식을 사용하는 메서드들의 구현을 바꾼다.
    - 기존 Method 방식을 GET,POST 만 있던 것에서 GET,POST,PUT,DELETE 등 다양하게 바꾼다.
    - `@ApiOperation` 은 해당 메서드에 대한 설명, 주석을 입력하는 Annotation이다.

```java
@RestController
@CrossOrigin("*")
@Api(value="SSAFY", description="SSAFY Product Swagger Main Controller")
public class MainControllerRest {

    @Autowired
    ProductService pService;

    @ExceptionHandler
    public ModelAndView handler(Exception e) {
        ModelAndView mav = new ModelAndView("/error/error");
        mav.addObject("msg", e.getMessage());
        e.printStackTrace();
        return mav;
    }

//  @GetMapping("mvInsert")
//  public String mvInsert() {
//      return "product/insert";
//  }

    @PutMapping("product")
    @ApiOperation("제품 등록")
    public ResponseEntity insertProduct(@RequestBody Product product) {
        pService.insertProduct(product);
        return new ResponseEntity(HttpStatus.NO_CONTENT);
    }


    @GetMapping("product")
    @ApiOperation("제품 목록 조회")
    public ResponseEntity<List<Product>> listProduct(Model model, @RequestBody PageBean bean) {
        model.addAttribute("list", pService.searchAll(bean));
        return new ResponseEntity<List<Product>>(pService.searchAll(bean),HttpStatus.OK);

    }

    @GetMapping("product/{productno}")
    @ApiOperation("제품 조회")
    public ResponseEntity<Product> search(Model model, @PathVariable int productno) {
        model.addAttribute("product", pService.getProduct(productno));
        return new ResponseEntity<Product>(pService.getProduct(productno), HttpStatus.OK);
    }
}
```

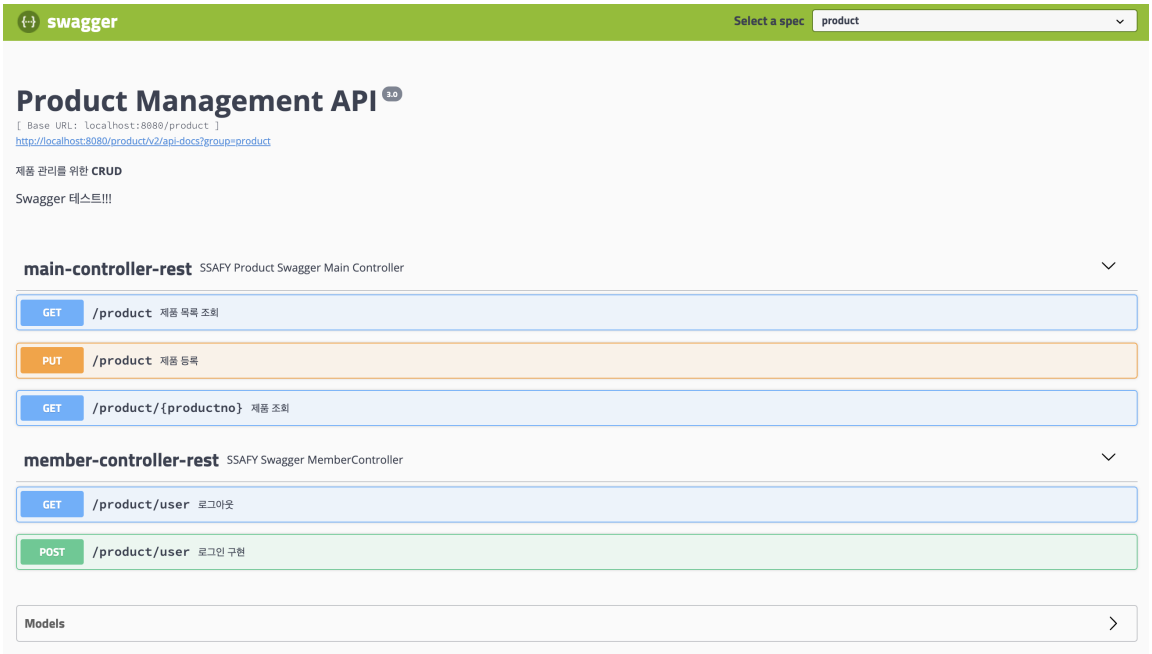
MainControllerRest.java

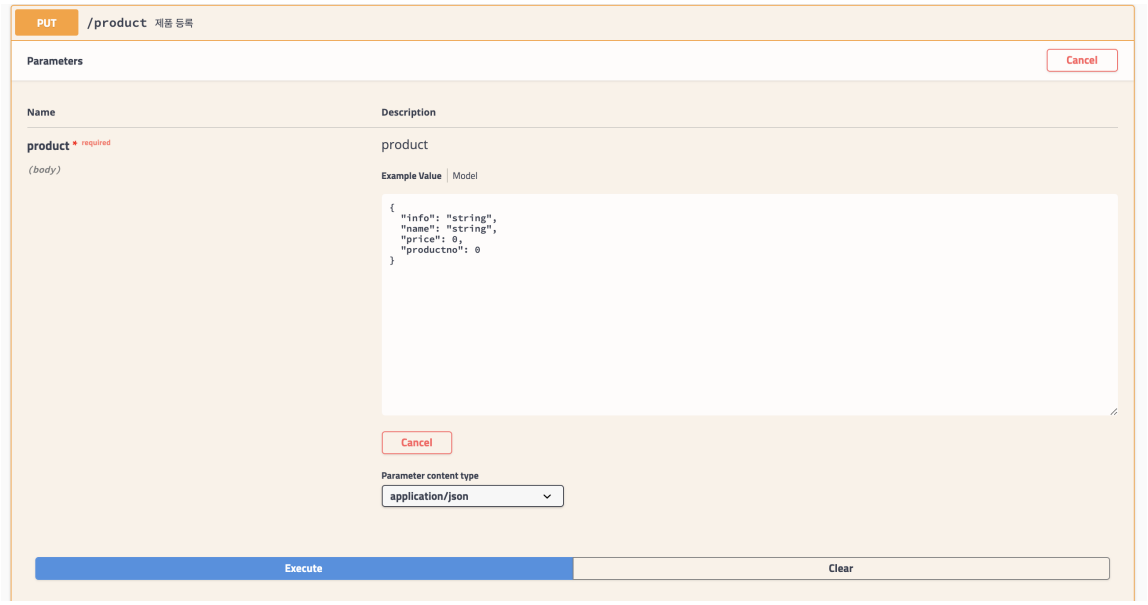
- MemberController 또한 바꾸어준다.

```java
@RestController
@CrossOrigin("*")
@Api(value="SSAFY", description="SSAFY Swagger MemberController")
public class MemberControllerRest {

    @Autowired
    private LoginService loginService;

    @ExceptionHandler
    public ModelAndView handler(Exception e) {
        ModelAndView mav = new ModelAndView("/error/error");
        mav.addObject("msg", e.getMessage());
        e.printStackTrace();
        return mav;
    }

    @PostMapping("product/user")
    @ApiOperation("로그인 구현")
    public ResponseEntity<MemberDto> login(@RequestBody MemberDto member, HttpServletRequest request) {
        try {
            MemberDto mem = loginService.login(member);
            HttpSession session = request.getSession();
            session.setAttribute("userinfo", mem);
            if(mem != null) {
                return new ResponseEntity<MemberDto>(mem, HttpStatus.OK);
            }

        } catch (ProductException e) {
            request.setAttribute("msg", e.getMessage());
            return new ResponseEntity(HttpStatus.INTERNAL_SERVER_ERROR);
        }

        return new ResponseEntity(HttpStatus.INTERNAL_SERVER_ERROR);
    }

    @GetMapping("product/user")
    @ApiOperation("로그아웃")
    public ResponseEntity logout(HttpServletRequest request) {
        HttpSession session = request.getSession();
        session.invalidate();
        return new ResponseEntity(HttpStatus.OK);
    }
```

MemberControllerRest.java

5. 마지막으로 실행 후 product/swagger-ui.html 로 접속해보니 정상적으로 보이는 것을 확인할 수 있다.





⚠ 주의 사항

- 처음에는 swagger-ui.html 페이지를 찾을 수 없는 404 에러가 발생했었다.
- 따라서 나는 mapping이 안된 것을 확인하고 swagger-ui 등의 jar 파일의 버전을 2.9.2 버전으로 바꾸어 봤지만 역시나 같은 상황이었다.
- 분명히 resource의 mapping 문제일 것이라 생각하고 다시 확인해보았지만 잘 보이지 않았다.
- 하지만 제일 첫 줄의 resource mapping을 확인해보니 /** 로 되어있어서 모든 resource들이 WEB-INF/resources 로 경로가 연결된 것을 보고 원인을 찾았다.

```
<resources mapping="/**" location="/WEB-INF/resources/" />
<resources mapping="/swagger-ui.html"    location="classpath:/META-INF/resources/" />
<resources mapping="/webjars/**"         location="classpath:/META-INF/resources/webjars/" />
<resources mapping="/*.html"         location="/" />
```

- 이렇게 되면 swagger-ui.html 또한 저 경로로 mapping 되기 때문에 찾을 수 없었던 것이다.
- 따라서 아래처럼 바꾸어주고 다시 실행해보니 정상 실행됨을 확인할 수 있었다.

```
<!-- Handles HTTP GET requests for /resources/** by efficiently serving up static resources in
<resources mapping="/resources/**" location="/resources/" />
<resources mapping="/swagger-ui.html"    location="classpath:/META-INF/resources/" />
<resources mapping="/webjars/**"         location="classpath:/META-INF/resources/webjars/" />
<resources mapping="/*.html"         location="/" />
```