스프링시작하기

스프링 API 10장

HATEOAS
Swagger Documentation

박명회

1.1 HATEOAS

HATEOAS(Hypermedia as the Engine of Application State)는 RESTful API에서 하이퍼미디어를 통해 애플리케이션의 상태를 관리하는 방법론입니다. Spring HATEOAS는 Spring Framework의 하위 프로젝트로, HATEOAS를 구현하는 데 도움이 되는 클래스와 유틸리티를 제공합니다. 이를 사용하여 간단한 예제를 만들어보겠습니다.
에 추가하여 Spring HATEOAS를 사용할 것입니다.
그런 다음, 사용자를 나타내는 도메인 클래스를 만듭니다.

```
<!-- Spring HATEOAS -->
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-hateoas</artifactId>
</dependency>
```

1.2 HATEOAS

이 코드에서는 엔드포인트로 들어오는 GET 요청에 대한 핸들러를 정의합니다. 이 핸들러는 사용자 정보를 EntityModel에 담아 반환하고, 해당 사용자에 대한 self 링크를 추가합니다. 이렇게 함으로써 클라이언트가 반환된 링크를 통해 해당 사용자의 자세한 정보를 요청할 수 있습니다.

```
@RestController
@RequestMapping("/api")
public class UserController {

    @GetMapping("/users/{id}")
    public EntityModel<User> getUser(@PathVariable Long id) {
        User user = new User(id, "John");

        EntityModel<User> entityModel = EntityModel.of(user);
        entityModel.add(
            WebMvcLinkBuilder.linkTo(UserController.class)
                             .slash("/users")
                             .slash(id)
                             .withSelfRel()
        );

        return entityModel;
    }
}
```

1.2 HATEOAS

```
EntityModel entityModel = EntityModel.of(user);

WebMvcLinkBuilder linkTo = linkTo(methodOn(this.getClass()).retrieveAllUsers());
entityModel.add(linkTo.withRel("all-users")); // http://127.0.0.1:8080/users

try {
    return entityModel;
} catch (Exception ex) {
    throw new RuntimeException();
}
```

1.2 HATEOAS

```
@GetMapping("users")
public EntityModel<Users> users(){

    EntityModel<Users> entityModel = EntityModel.of(new Users( username: "홍길동", email: "aaa@naver.com"));

    entityModel.add(
            WebMvcLinkBuilder.linkTo(WebMvcLinkBuilder.methodOn(this.getClass()).user())
                    .slash( object: "/user")
                    .slash( object: "1")
                    .withSelfRel()
    );

    return entityModel;
}
```

```
{
  "username": "홍길동",
  "email": "aaa@naver.com",
  "_links": {
    "self": {
      "href": "http://localhost:8080/user/user/1"
    }
  }
}
```

1.3 HATEOAS

링크가 포함되어 있으며, 해당 링크는 사용자 자체를 나타냅니다. 이 링크를 따라가면 같은 사용자에 대한 자세한 정보를 요청할 수 있습니다.
이렇게 함으로써 HATEOAS를 사용하여 API의 응답에 하이퍼미디어 링크를 포함시켜 클라이언트가 쉽게 API를 탐색하고 상호작용할 수 있도록 할 수 있습니다.

```
{
  "id": 1,
  "name": "John",
  "_links": {
    "self": {
      "href": "http://localhost:8080/api/users/1"
    }
  }
}
```

1.4 Swagger Documentation

https://springdoc.org/ 공식홈페이지 방문!!

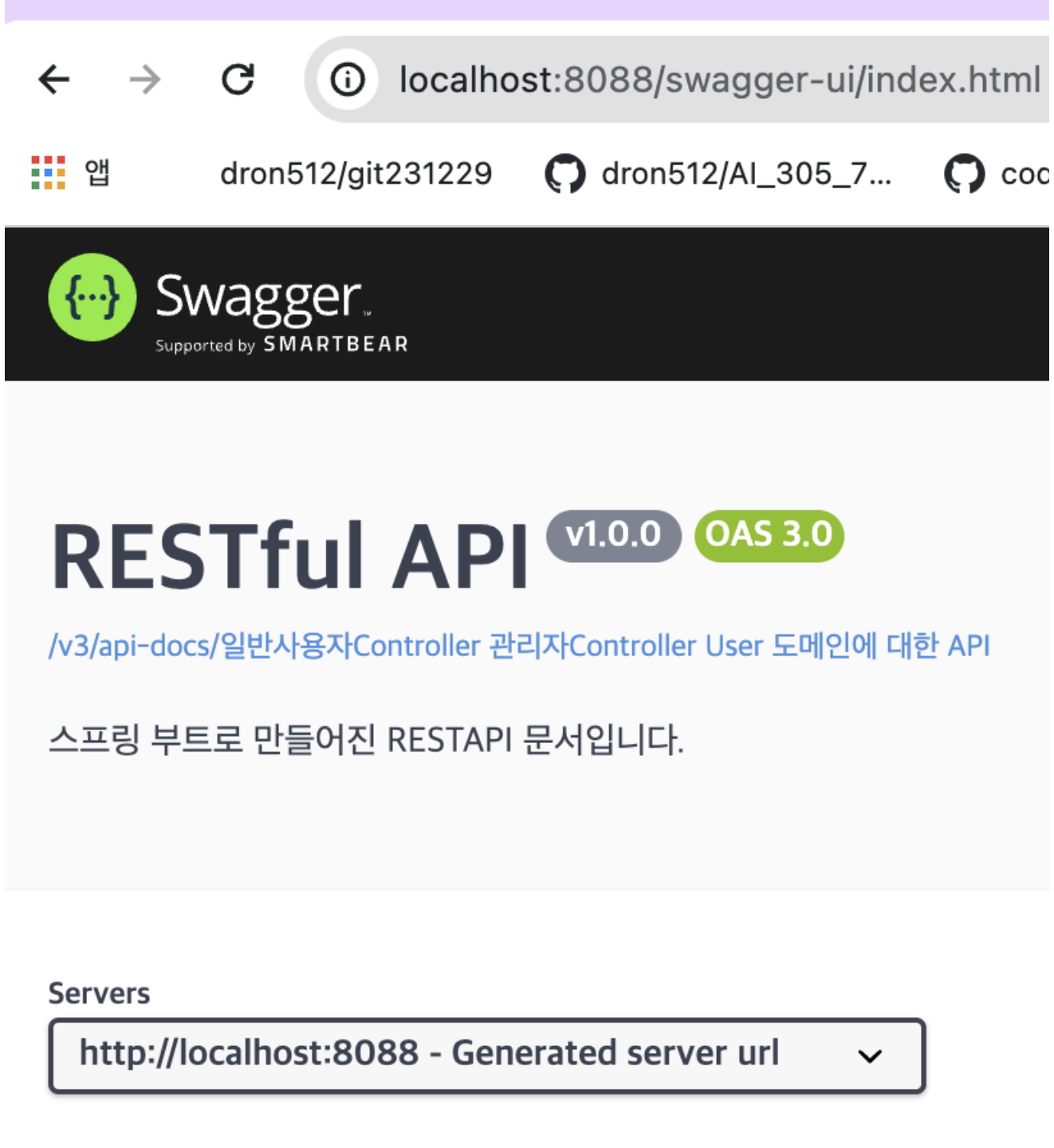
1. dependency 추가 2. @OpenAPIDefinition 추가

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.3.0</version>
</dependency>
```

```
@OpenAPIDefinition(
        info = @Info(title = "RESTful API",
                description = "스프링 부트로 만들어진 RESTAPI 문서입니다.",
                version = "v1.0.0"))
@Configuration
@RequiredArgsConstructor
public class NewSwaggerConfig {
```

1.4 Swagger Documentation

Swagger는 RESTful API를 문서화하고 브라우징할 수 있는 강력한 오픈 소스 프레임워크입니다. Swagger는 API의 설계, 빌드, 문서화, 테스트 및 배포를 위한 다양한 도구를 제공하여 개발자들이 API를 쉽게 이해하고 사용할 수 있도록 돕습니다.



1.4 Swagger Documentation

/v3/api-docs/일반사용자Controller 관리자Controller User 도메인에 대한 API

```java
@Bean
public GroupedOpenApi customTestOpenAPi() {
    String[] paths = {"/users/**", "/admin/**"};

    return GroupedOpenApi
            .builder()
            .group(group:"일반사용자Controller 관리자Controller User 도메인에 대한 API")
            .pathsToMatch(paths)
            .build();
}
```

1.4 Swagger Documentation

@Sechma

```java
@Schema(description = "사용자 상세 정보를 위한 도메인 객체")
@Entity
@Table(name = "users")
public class User {
    @Schema(title = "사용자 ID", description = "사용자 ID는 자동 생성됩니다.")
    @Id
    @GeneratedValue
    private Integer id;
```

1.4 Swagger Documentation

User ⌄ {

description:	사용자 상세 정보를 위한 도메인 객체
id	사용자 ID › [...]
username	› [...]
email	› [...]
name	사용자 이름 › [...]
joinDate	등록일 › [...]
password	

1.4 Swagger Documentation

```
@RestController
@Tag(name = "user-controller", description = "일반 사용자 서비스를 위한 컨트롤러")
public class UserController {
    5 usages
```

```
@Operation(summary = "사용자 정보 조회 API", description = "사용자 ID를 이용하여 사용자 상세 정보 조회를 합니다.")
@ApiResponses({
        @ApiResponse(responseCode = "200", description = "OK !!",
                content = @Content(array = @ArraySchema(schema = @Schema(implementation = User.class)))),
        @ApiResponse(responseCode = "400", description = "BAD REQUEST !!"),
        @ApiResponse(responseCode = "404", description = "NOT FOUND !!"),
        @ApiResponse(responseCode = "500", description = "INTERNAL SERVER ERROR !!"),
})
@GetMapping("/users/{id}")
public EntityModel<User> retrieveUser(
```

1.4 Swagger Documentation

Contoller에 PathVariable에 대한 설명 입력

```
@Parameter(description = "사용자 ID", required = true, example = "1") @PathVariable int id) {
```

고맙습니다.