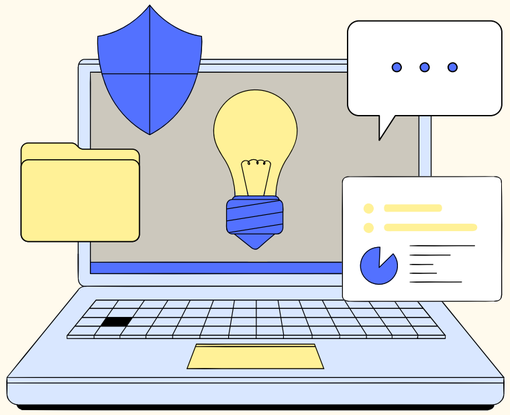
센드빌 KOSTA 교육

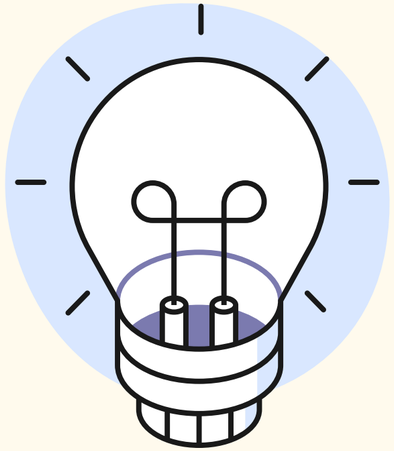
Spring Boot 기반 REST API 구현과 JWT인증

2024.3.20 ~ 3.21



목차

REST API 란?



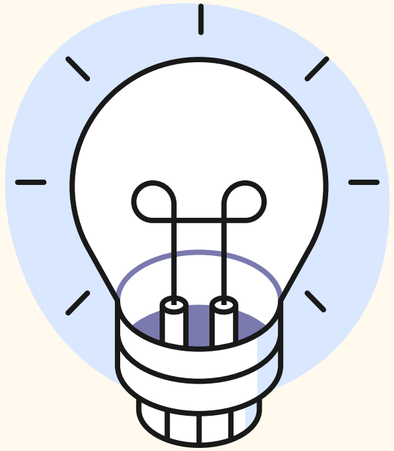
REST (REpresentational State Transfer)

REST는 분산 시스템 설계를 위한 아키텍처 스타일이다. 아키텍처 스타일이라는 것은 제약 조건의 집합이라고 보면 됩니다.

RESTful 이란?

RESTful은 위의 제약 조건의 집합 (아키텍처 원칙)을 모두 만족하는 것을 의미합니다.
REST라는 아키텍처 스타일이 있고,
RESTful API라는 말은 REST 아키텍처 원칙을 모두 만족하는 API 이라는 뜻입니다.

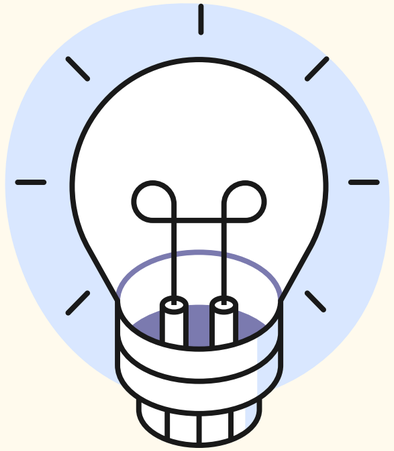
REST API 란?



API (Application Programming Interface)

어떠한 응용 프로그램에서 데이터를 주고 받기 위한 방법을 의미합니다.
어떤 특정 사이트에서
데이터를 공유할 경우 어떠한 방식으로 정보를 요청해야 하는지,
그리고 어떠한 데이터를 제공
받을 수 있을지에 대한 규격들을 API라고 합니다.

REST API 란?



REST가 필요한 이유

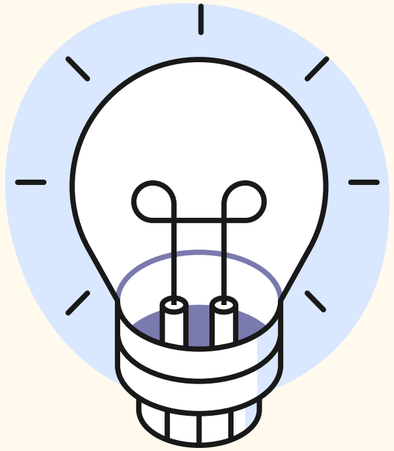
1. 분산시스템

거대한 어플리케이션을 모듈, 기능별로 분리하기 쉬워졌으므로.
RESTful API를 서비스 하면 다른 모듈 또는 어플리케이션들이라도
RESTful API를 통해서 상호간에 통신을 할 수 있기 때문입니다.

2. Web 브라우저 이외의 클라이언트

웹페이지를 위한 HTML 및 이미지 등을 보내던 것과는 다르게,
데이터만 보내면 여러 클라이언트에서 해당 데이터를 주고 받기 때문에
자유롭고 부담없이 데이터를 이용할 수 있으며
서버도 요청한 데이터를 보내기 만 하므로 가벼워지고
유지 보수성도 좋아질 수 있습니다.

HATEOAS & SDM적용



HATEOAS 란?

1. 링크에 사용 가능한 URL을 리소스로 전달하여 client가 참고하여 사용할 수 있도록 하는 것
2. 하이퍼미디어(링크)를 통해 애플리케이션 상태 변화가 가능해야 합니다.
3. Hypermedia (링크)에 자기 자신에 대한 정보가 포함 되어야 한다

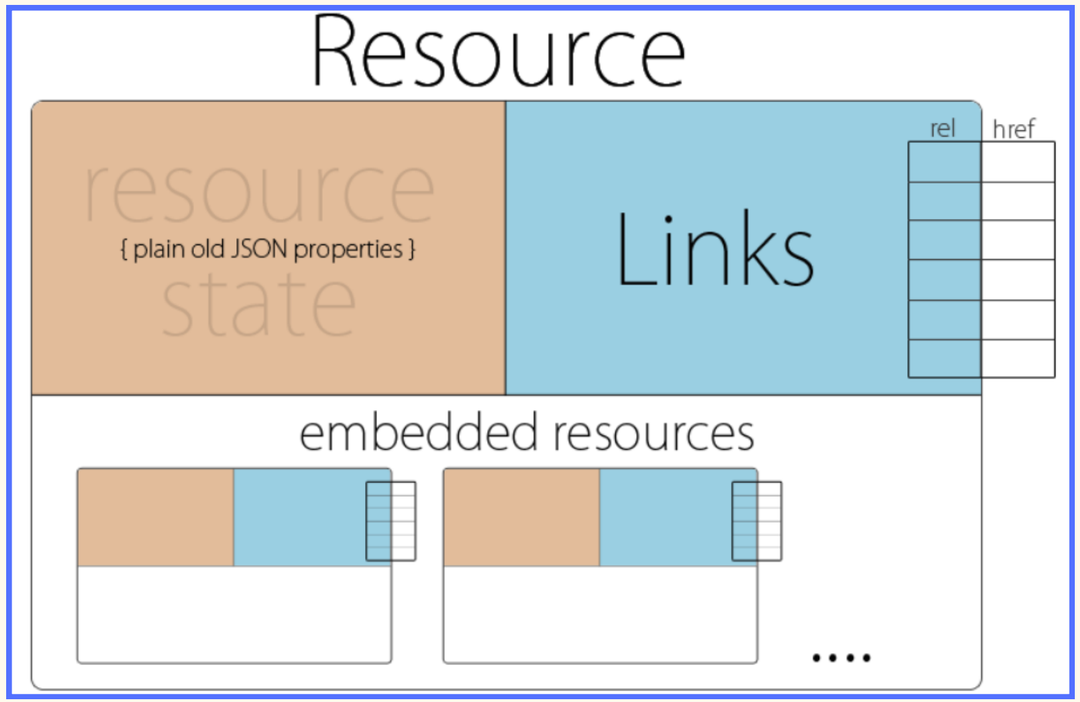
Self-Descriptive Messages

1. 메시지 스스로 메시지에 대한 설명이 가능해야 합니다
2. API 문서가 REST API 응답 본문에 존재해야ㅁ 한다

HATEOAS & SDM적용

(Hypermedia As The Engine Of Application State)

HAL Model 구조



Resources(data 필드)

1. **Links :** 웹사이트에서 하이퍼 링크 역할을 하는것과 비슷하고 이링크는 URL 과 rel 로 이루어져 있으며 선택적 속성을포함 할수도 있습니다.
2. **Embedded Resources :** 큰 리소스 안에 작은리소스가 들어가있는 구조
3. **State :** 리소스의 실제 데이터 리소스의 속성과 값을 나타냄

Links(self 필드)

1. **target :** 링크가 연결되는 리소스의 위치를 정확하게 식별합니다
2. **rel(관계) :** 링크의 종류를 나타냅니다 예를 들어
   상세보기 rel=detail 다음 페이지 rel=next,
   부모리소스 rel=parent
3. **optional properties(선택적 속성) :** 앞의 두가지 속성 외 몇가지 선택적 속성이 포함될수있고 이속성들은 주로 링크 사용의 상태를 조정하는 역할을 합니다.

HATEOAS & SDM적용

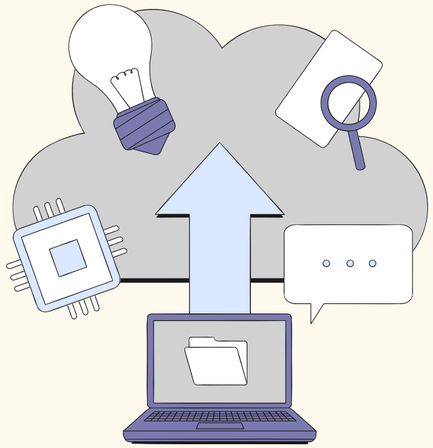
기본 REST API 응답 데이터


```
{
   "id": 4,
   "name": "2 Lecture ",
   "description": "Test Lecture",
   "beginEnrollmentDateTime": "2022-11-23 14:21",
   "closeEnrollmentDateTime": "2022-11-24 14:21",
   "beginLectureDateTime": "2022-11-25 14:21",
   "endLectureDateTime": "2022-11-26 14:21",
   "location": "2 강의장",
   "basePrice": 100,
   "maxPrice": 200,
   "limitOfEnrollment": 100,
   "offline": true,
   "free": false,
   "email": null
}
```


HATEOAS 도입 REST API 응답 데이터


```
{
   "id": 4,
   "name": "2 Lecture ",
   "description": "Test Lecture",
   "beginEnrollmentDateTime": "2022-11-23 14:21",
   "closeEnrollmentDateTime": "2022-11-24 14:21",
   "beginLectureDateTime": "2022-11-25 14:21",
   "endLectureDateTime": "2022-11-26 14:21",
   "location": "2 강의장",
   "basePrice": 100,
   "maxPrice": 200,
   "limitOfEnrollment": 100,
   "offline": true,
   "free": false,
   "email": null,
   "_links": {
      "self": {
         "href": "http://localhost:8089/api/lectures/4"
      }
   }
}
```

REST API 구현



사용환경

1. JAVA Version 17
2. 스프링 부트 프로젝트 생성 (https://start.spring.io/)
3. 추가 의존성
   - Web
   - Data JPA
   - HATEOAS
   - Validation
   - Lombook
   - Devtolls
   - Configuration Processor
4. 사용 툴
   - Inteli J
5. 사용 DB
   - H2 DB
   - Maria DB

REST API 구현

프로젝트 구조

Lecture
Entity

Lombok 어노테이션 Builder 를 사용하여 복잡한 Builder API를 자동으로 생성해줌 Entity 추가

LectureController
ResponseEntity

•ResponseEntity를 사용하는 이유
응답 코드, 헤더, 본문을 모두 저장 해주는 편리한 API
•Location URI 만들기
Spring HATEOAS WebMvcLinkBuilder의 linkTo(), methodOn() 사용

Lecture Repository
담당업무 : 디자인 총괄

Spring Data JPA는 "JpaRepository" 라는 기능을 사용하면 매우 간단히 데이터를 검색/등록할 수 있다.

LectureReqDto
ModelMapper

DB Layer에는 Entity 클래스, View Layer에서 DTO 클래스를 사용하여 역할을 분리하는 것이 좋습니다
Entity와 DTO를 연결할 때 ModelMapper 를 사용합니다

LectureResDto

응답을 처리하는 DTO 클래스

LectureValidator
@Valid

• @NotNull, @NotEmpty, @Min, @Max, @Size(min=, max=) 사용해서 입력 값을 바인딩할 때 에러를 확인할 수 있습니다.

REST API 보안 적용

1.스프링 Security 란?

스프링 Security는 스프링 기반의 어플리케이션의 보안(인증과 권한)을 담당하는 프레임워크이다

*** 보안 관련 용어**

- 접근 주체(Principal) : 보호된 대상에 접근하는 user
- 인증(Authenticate) : 현재 user가 누구인지 확인, 애플리케이션의 작업을 수행할 수 있는주체임을 증명함
- 인가(Authorize) : 현재 user가 어떤 서비스, 페이지에 접근할 수 있는 권한이 있는지 검사

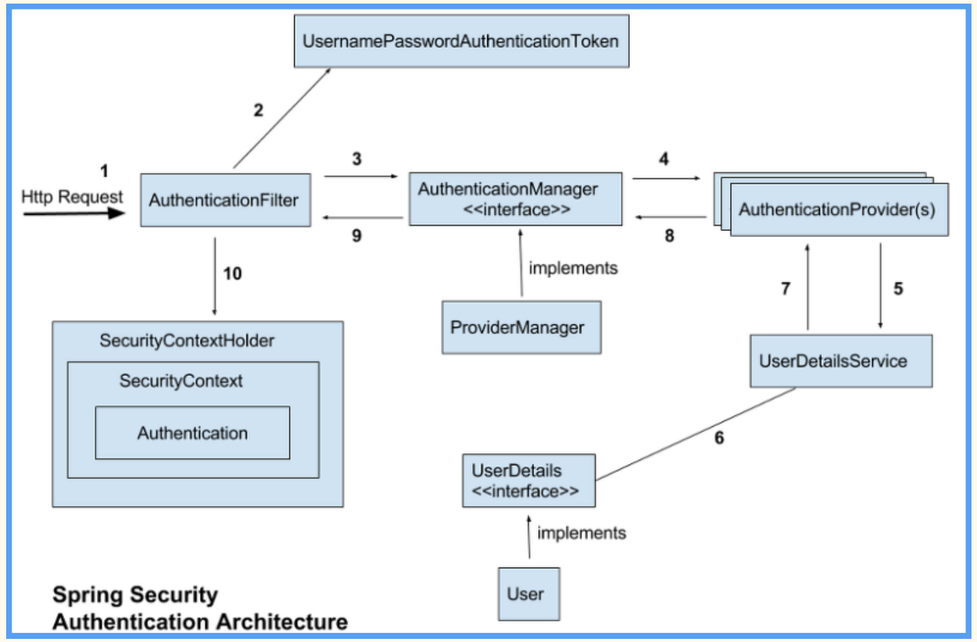
*** 스프링 시큐리티가 제공하는 기능**

- 웹 시큐리티 (Filter 기반 시큐리티)
- 메소드 시큐리티
- 두가지 방법 모두 Security Interceptor를 사용합니다.

: 리소스에 접근을 허용할 것이냐 말 것 이냐를 결정하는 로직이 들어 있음

REST API 보안 적용

2. 인증관련 아키텍쳐

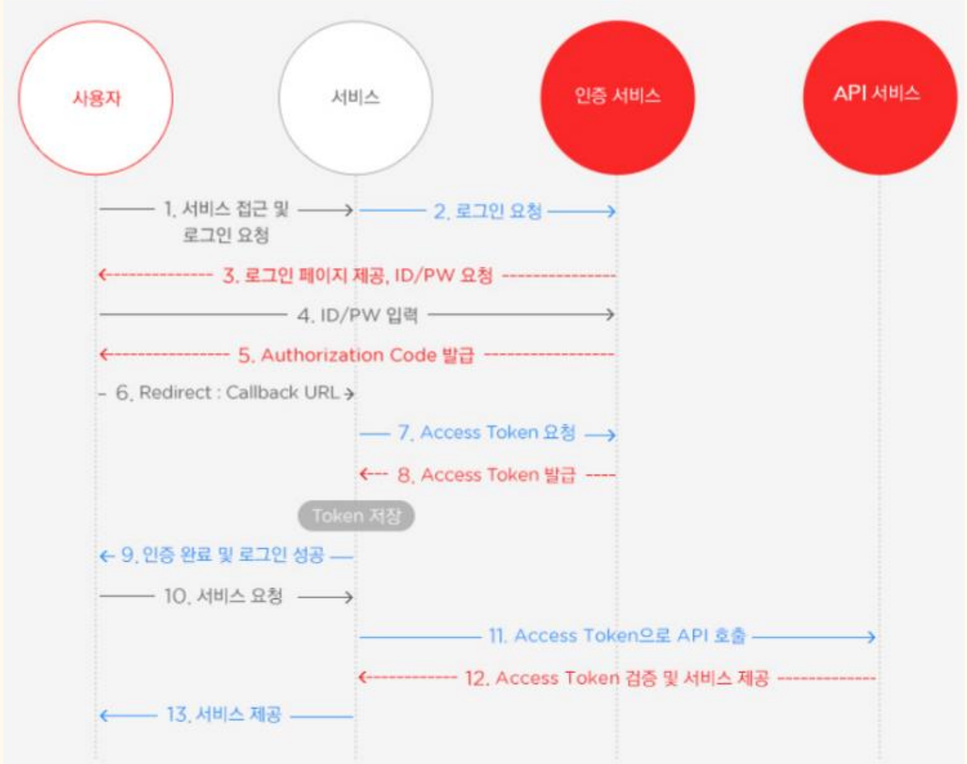


7. 7,8,9,10. 인증이 완료되면 사용자 정보를 가진 Authentication 객체를 SecurityContextHolder에 담은 이후 AuthenticationSuccessHandler를 실행한다. (실패시 AuthenticationFailureHandler를 실행한다.)

1. 사용자가 Form을 통해 로그인 정보를 입력하고 인증 요청을 보낸다
2. UsernamePasswordAuthenticationFilter가 사용자가 보낸 아이디와 패스워드를 인터셉트한다. HttpServletRequest에서 꺼낸 아이디와 패스워드로 실제 인증을 담당 할 AuthenticationManager (구현체-ProviderManager) 에게 인증용 객체 (UsernamePasswordAuthenticationToken)로 만들어 주어 위임한다.
3. AuthenticationFilter에게 인증용 객체 (UsernamePasswordAuthenticationToken)을 전달받는다.
4. 실제 인증을 할 AuthenticationProvider에게 Authentication객체 (UsernamePasswordAuthenticationToken)을 다시 전달한다.
5. DB에서 사용자 인증 정보를 가져올 UserDetailsService 객체에게 사용자 아이디를 넘겨주고 DB에서 인증에 사용할 사용자 정보(사용자 아이디, 암호화된 패스워드, 권한 등)를 UserDetails라는 객체로 전달 받는다.
6. AuthenticationProvider는 UserDetails 객체를 전달 받은 이후 실제 사용자의 입력정보와 UserDetails 객체를 사용하여 인증을 시도한다.

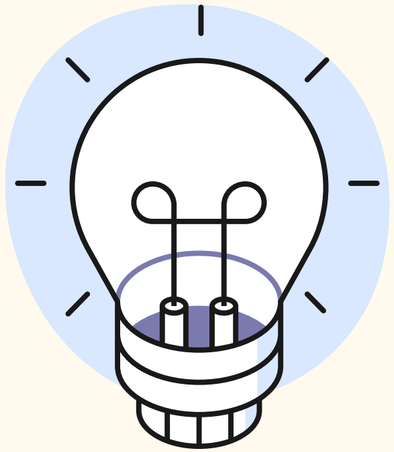
REST API 보안 적용

3. OAuth 2.0 와 JWT(Json Web Token)



- OAuth는 토큰을 발급하고 인증하는 오픈 스탠다드 프로토콜이며 프레임워크이다. JWT는 이러한 프레임워크에서 발생하는 산출물로 볼 수 있다.
- OAuth2.0는 하나의 플랫폼의 권한(아무 의미 없는 무작위 문자열 토큰)으로 다양한 플랫폼에서 권한을 행사할 수 있게 해줌으로써 리소스 접근이 가능하게 하는데 목적을 두고 있다.
- JWT는 Cookie, Session을 대신하여 의미 있는 문자열 토큰으로써 권한을 행사할 수 있는 토큰의 한 형식이다. (로그인 세션이나 주고받는 값이 유효한지 검증할 때 주로 쓰인다.)

REST API 보안 적용



JWT(Json Web Token) 적용 시 고려사항

1.Self-contained

토큰 자체에 정보를 담고 있으므로 토큰의 길이가 길어 질 수 있다.
토큰 길이: 토큰의 페이로드(Payload)에 3종류의 클레임을 저장하기 때문에, 정보가 많아질수록 토큰의 길이가 늘어나
네트워크에 부하를 줄 수 있다.

2.Payload 인코딩

페이로드(Payload) 자체는 암호화된 것이 아니라, BASE64로 인코딩 된 것이다. 중간에 Payload를 탈취하여 디코딩
하면 데이터를 볼 수 있으므로, JWE로 암호화 하거나 Payload에 중요 데이터를 넣지 않아야 한다.

3.Stateless

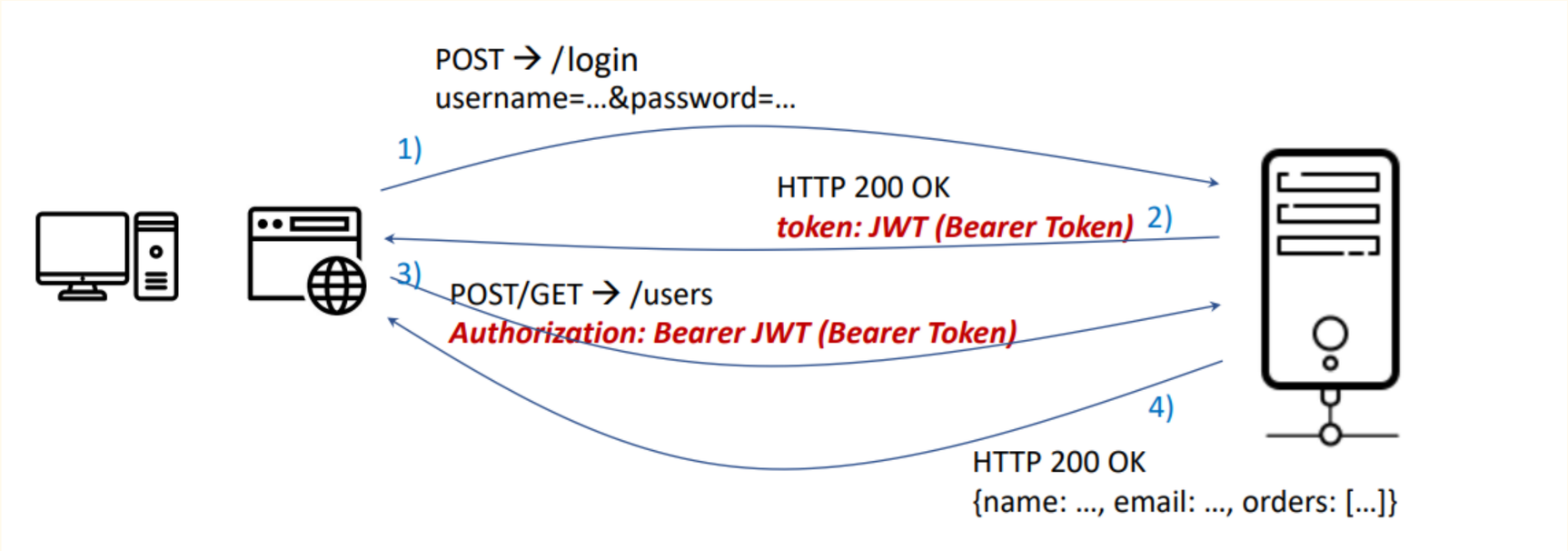
JWT는 상태를 저장하지 않기 때문에 한 번 만들어지면 제어가 불가능하다. 즉, 토큰을 임의로 삭제하는 것이 불가능
하므로 토큰 만료 시간을 꼭 넣어주어야 한다.

4.Tore Token

토큰은 클라이언트 측에서 관리해야 하기 때문에, 토큰을 저장해야 한다

REST API 보안 적용

4. 로그인 인증 처리과정



REST API Docs 활용

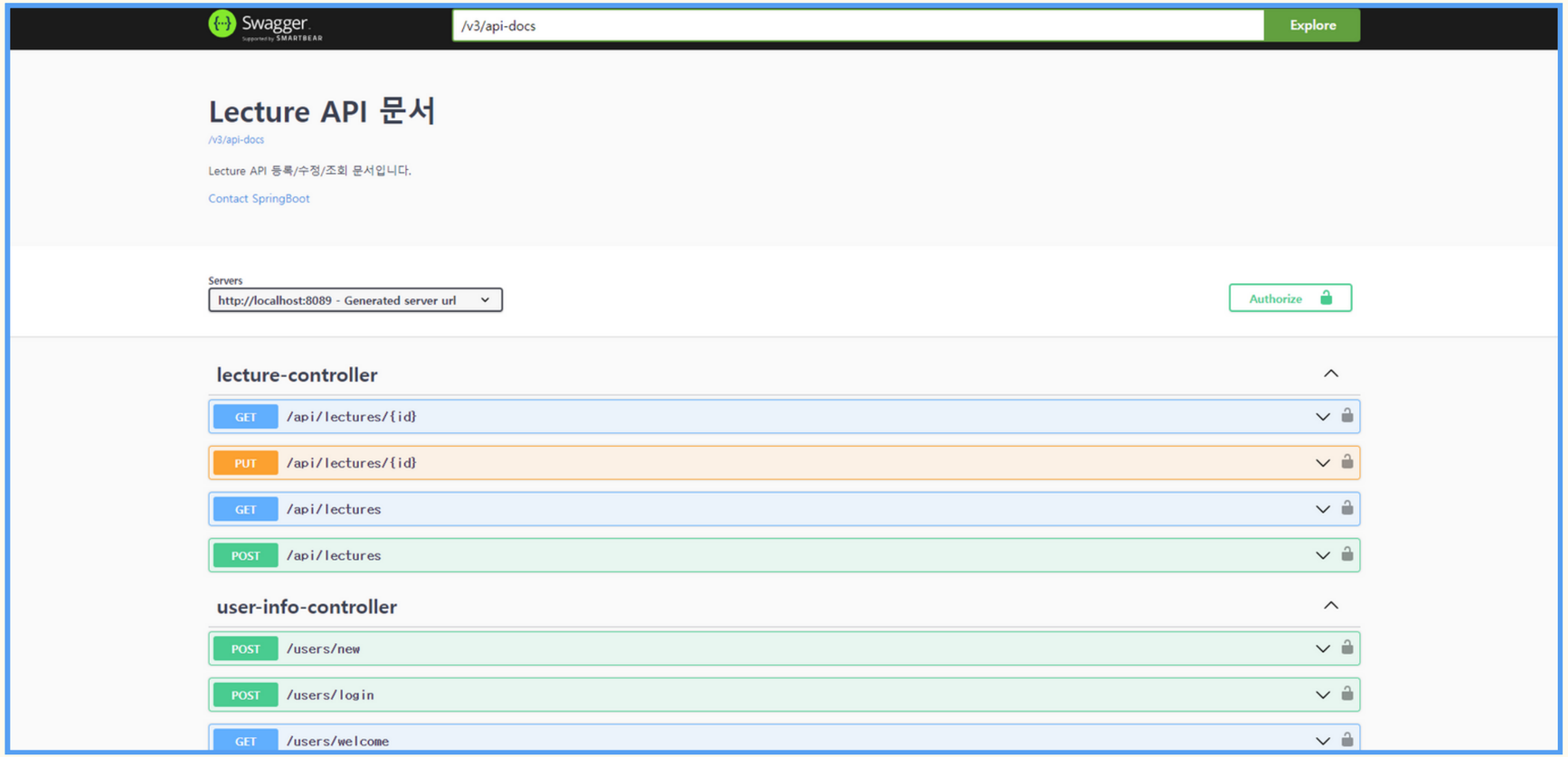
Swagger 로 API 문서 자동화

1. **springdoc-openapi 란?**

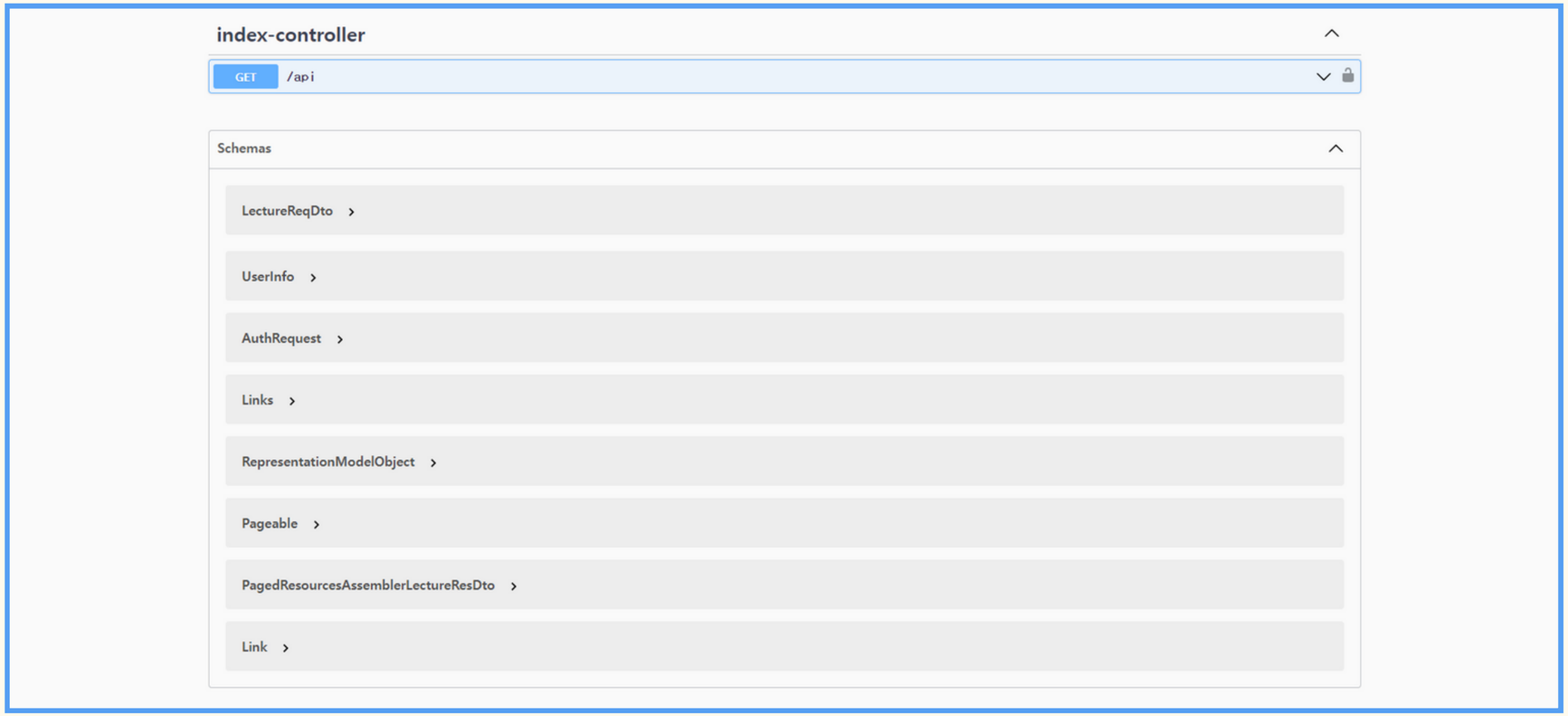
   Springdoc은 OpenAPI 3.0 spec을 이용하여 구현 하였다.

   REST API Docs을 생성 해주는 Open Source는 Springfox Swagger와 SpringDoc 이 있다.

REST API Docs 활용



REST API Docs 활용



감사합니다

Spring Boot 기반 REST API 구현과 JWT인증

2024.3.20 ~ 3.21

박진석

https://github.com/ParkJinTol/RESTAPI