

# 스프링 기반 REST API 개발

이번 강좌에서는 다양한 스프링 기술을 사용하여 Self-Descriptive Message와 HATEOAS(Hypermedia as the engine of application state)를 만족하는 REST API를 개발합니다.

이응준님께서 2017년 네이버가 주관한 Devview라는 개발자 컨퍼런스에서 '[그런 REST API로 관찮은가](#)'라는 발표를 했습니다. 발표의 핵심은 현재 REST API로 불리우는 대부분의 API가 실제로는 로이 필딩이 정의한 REST를 따르고 있지 않으며, 그 중에서도 특히 Self-Descriptive Message와 HATEOAS가 지켜지고 있지 않음을 지적했고 그에 대한 대안을 제시했습니다.

이번 강좌는 해당 발표에 영감을 받아 만들었으며, KSUG에서 동일한 이름으로 2018년 11월 세미나를 진행한 경험이 있습니다. 4시간이라는 짧지 않은 발표였지만, 준비한 내용에 비해 시간이 부족할 수 있었기 때문에 진행을 빨리 하느라 충분히 설명하지 못하고 넘어갔던 부분을 이번 강좌에서 보충하고 또 해결하려는 문제에 대한 여러 선택지를 제공하는 것이 좋을것 같아 강좌를 만들었습니다. 그리고 제가 주로 사용하는 인텔리J 단축키 역시 코딩하는 중에 설명합니다.

이번 강좌에서는 다음의 다양한 스프링 기술을 사용하여 REST API를 개발합니다.

- 스프링 프레임워크
- 스프링 부트
- 스프링 데이터 JPA
- 스프링 HATEOAS
- 스프링 REST Docs
- 스프링 시큐리티 OAuth2

또한 개발은 테스트 주도 개발(TDD)로 진행하기 때문에 평소 테스트 또는 TDD에 관심있던 개발자에게도 이번 강좌가 도움이 될 것으로 기대합니다.

## 사전 학습

- 스프링 프레임워크 핵심 기술 (필수)
- 스프링 부트 개념과 활용 (필수)
- 스프링 데이터 JPA (선택)

## 학습 목표

- Self-Descriptive Message와 HATEOAS를 만족하는 REST API를 이해합니다.
- 다양한 스프링 기술을 활용하여 REST API를 개발할 수 있습니다.
- 스프링 HATEOAS와 스프링 REST Docs 프로젝트를 활용할 수 있습니다.
- 테스트 주도 개발(TDD)에 익숙해 집니다.

## 1. 강좌 소개

첫 페이지 참고

소스 코드

- <https://github.com/keesun/study/tree/master/rest-api-with-spring>

## 2. 강사 소개

백기선

- 현재 마이크로소프트 미국 본사에 근무 중. (그전에는 네이버와 아마존에서 일을 했습니다.)
- 2007년부터 개발자로 일했으며 이제 막 경력 10년이 조금 넘었네요.
- 자바, 스프링 프레임워크, JPA, 하이버네이트를 주로 공부하고 공유해 왔습니다.
- Youtube/백기선 채널에서 코딩 관련 정보를 영상으로 공유하고 있습니다.
- (예전에는 Whiteship.me 라는 블로그에 글도 많이 올렸지만 요즘은 잘 안써요.)
- (더 예전에는 책도 쓰고 번역도 하고 발표도 많이 했었지만 역시나.. 요즘은 안합니다.)

### 3. REST API

#### API

- **A**pplication **P**rogramming **I**nterface

#### REST

- **RE**presentational **S**tate **T**ransfer
- 인터넷 상의 시스템 간의 상호 운용성(interoperability)을 제공하는 방법중 하나
- 시스템 제각각의 **독립적인 진화**를 보장하기 위한 방법
- REST API: REST 아키텍처 스타일을 따르는 API

#### REST 아키텍처 스타일 ([발표 영상](#) 11분)

- Client-Server
- Stateless
- Cache
- **Uniform Interface**
- Layered System
- Code-On-Demand (optional)

#### Uniform Interface ([발표 영상](#) 11분 40초)

- Identification of resources
- manipulation of resources through representations
- **self-describe messages**
- **hypermedia as the engine of appliaction state (HATEOAS)**

#### 두 문제를 좀 더 자세히 살펴보자. ([발표 영상](#) 37분 50초)

- Self-descriptive message
  - 메시지 스스로 메시지에 대한 설명이 가능해야 한다.
  - 서버가 변해서 메시지가 변해도 클라이언트는 그 메시지를 보고 해석이 가능하다.
  - **확장 가능한 커뮤니케이션**
- HATEOAS
  - 하이퍼미디어(링크)를 통해 애플리케이션 상태 변화가 가능해야 한다.
  - **링크 정보를 동적으로 바꿀 수 있다.** (Versioning 할 필요 없이!)

#### Self-descriptive message 해결 방법

- 방법 1: 미디어 타입을 정의하고 IANA에 등록하고 그 미디어 타입을 리소스 리턴할 때 Content-Type으로 사용한다.
- **방법 2: profile 링크 헤더를 추가한다. ([발표 영상](#) 41분 50초)**
  - [브라우저들이 아직 스펙 지원을 잘 안해](#)
  - 대안으로 [HAL](#)의 링크 데이터에 [profile 링크](#) 추가

#### HATEOAS 해결 방법

- 방법1: 데이터에 링크 제공

- 링크를 어떻게 정의할 것인가? HAL

- 방법2: 링크 헤더나 Location을 제공

## 4. “Event” REST API

이벤트 등록, 조회 및 수정 API

### GET /api/events

이벤트 목록 조회 REST API (로그인 안 한 상태)

- 응답에 보여줘야 할 데이터
  - 이벤트 목록
  - 링크
    - self
    - profile: 이벤트 목록 조회 API 문서로 링크
    - get-an-event: 이벤트 하나 조회하는 API 링크
    - next: 다음 페이지 (optional)
    - prev: 이전 페이지 (optional)
- 문서?
  - 스프링 REST Docs로 만들 예정

이벤트 목록 조회 REST API (로그인 한 상태)

- 응답에 보여줘야 할 데이터
  - 이벤트 목록
  - 링크
    - self
    - profile: 이벤트 목록 조회 API 문서로 링크
    - get-an-event: 이벤트 하나 조회하는 API 링크
    - **create-new-event: 이벤트를 생성할 수 있는 API 링크**
    - next: 다음 페이지 (optional)
    - prev: 이전 페이지 (optional)
- 로그인 한 상태???? (stateless라며..)
  - 아니, 사실은 Bearer 헤더에 유효한 AccessToken이 들어있는 경우!

### POST /api/events

- 이벤트 생성

### GET /api/events/{id}

- 이벤트 하나 조회

### PUT /api/events/{id}

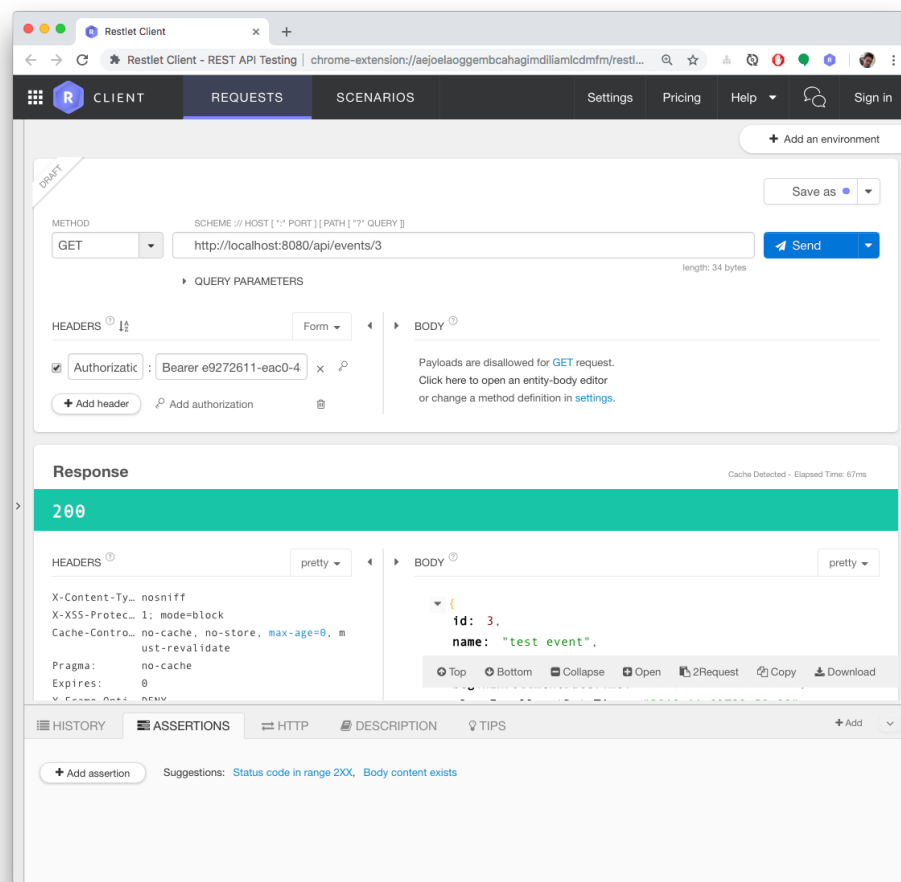
- 이벤트 수정

## 5. Events API 사용 예제

1. (토큰 없이) 이벤트 목록 조회
  - a. create 안 보임
2. access token 발급 받기 (A 사용자 로그인)
3. (유효한 A 토큰 가지고) 이벤트 목록 조회
  - a. create event 보임
4. (유효한 A 토큰 가지고) 이벤트 만들기
5. (토큰 없이) 이벤트 조회
  - a. update 링크 안 보임
6. (유효한 A 토큰 가지고) 이벤트 조회
  - a. update 링크 보임
7. access token 발급 받기 (B 사용자 로그인)
8. (유효한 B 토큰 가지고) 이벤트 조회
  - a. update 안 보임

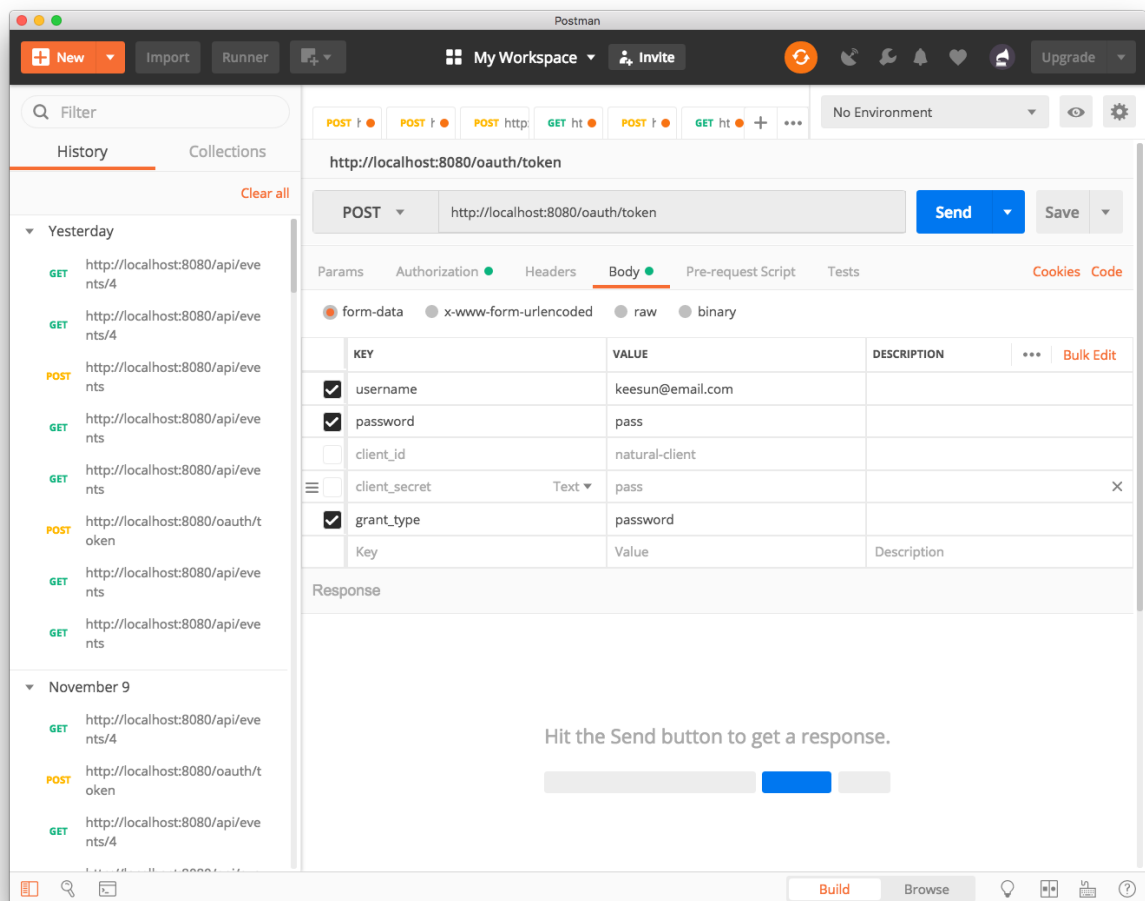
### REST API 테스트 클라이언트 애플리케이션

- 크롬 플러그인
  - Restlet



- 애플리케이션

## ○ Postman



## 6. 스프링 부트 프로젝트 만들기

추가할 의존성

- Web
- JPA
- HATEOAS
- REST Docs
- H2
- PostgreSQL
- Lombok

자바 버전 11로 시작

- [자바는 여전히 무료다.](#)

스프링 부트 핵심 원리

- 의존성 설정 (pom.xml)
- 자동 설정 (@EnableAutoConfiguration)
- 내장 웹 서버 (의존성과 자동 설정의 일부)
- 독립적으로 실행 가능한 JAR (pom.xml의 플러그인)



## 7. Event 생성 API 구현: 비즈니스 로직

### Event 생성 API

- 다음의 입력 값을 받는다.
  - name
  - description
  - beginEnrollmentDateTime
  - closeEnrollmentDateTime
  - beginEventDateTime
  - endEventDateTime
  - location (optional) 이게 없으면 온라인 모임
  - basePrice (optional)
  - maxPrice (optional)
  - limitOfEnrollment

basePrice와 maxPrice 경우의 수와 각각의 로직

basePrice	maxPrice	
0	100	선착순 등록
0	0	무료
100	0	무제한 경매 (높은 금액 낸 사람이 등록)
100	200	제한가 선착순 등록  처음 부터 200을 낸 사람은 선 등록.  100을 내고 등록할 수 있으나 더 많이 낸 사람에 의해 밀려날 수 있음.

- 결과값
  - **id**
  - name
  - ...
  - **eventStatus: DRAFT**, PUBLISHED, ENROLLMENT\_STARTED, ...
  - offline
  - free
  - \_links
    - profile (for the self-descriptive message)
    - self
    - publish
    - ...

## 8. Event 생성 API 구현: Event 도메인 구현

```
public class Event {  
  
    private String name;  
    private String description;  
    private LocalDateTime beginEnrollmentDateTime;  
    private LocalDateTime closeEnrollmentDateTime;  
    private LocalDateTime beginEventDateTime;  
    private LocalDateTime endEventDateTime;  
    private String location; // (optional) 이게 없으면 온라인 모임  
    private int basePrice; // (optional)  
    private int maxPrice; // (optional)  
    private int limitOfEnrollment;  
  
}
```

### 추가 필드

```
private Integer id;  
private boolean offline;  
private boolean free;  
private EventStatus eventStatus = EventStatus.DRAFT;
```

### EventStatus 이넘 추가

```
public enum EventStatus {  
  
    DRAFT, PUBLISHED, BEGAN_ENROLLMENT, CLOSED_ENROLLMENT, STARTED, ENDED  
  
}
```

### 롬복 애노테이션 추가

```
@Getter @Setter @EqualsAndHashCode(of = "id")  
@Builder @NoArgsConstructor @AllArgsConstructor  
public class Event {
```

- 왜 @EqualsAndHashCode에서 of를 사용하는가
- 왜 @Builder를 사용할 때 @AllArgsConstructor가 필요한가
- @Data를 쓰지 않는 이유
- 애노테이션 줄일 수 없나

## 9. Event 생성 API 구현: 테스트 만들자

### 스프링 부트 슬라이스 테스트

- @WebMvcTest
  - MockMvc 빈을 자동 설정 해준다. 따라서 그냥 가져와서 쓰면 됨.
  - 웹 관련 빈만 등록해 준다. (슬라이스)

### MockMvc

- 스프링 MVC 테스트 핵심 클래스
- 웹 서버를 띄우지 않고도 스프링 MVC (DispatcherServlet)가 요청을 처리하는 과정을 확인할 수 있기 때문에 컨트롤러 테스트용으로 자주 쓰임.

### 테스트 할 것

- 입력값들을 전달하면 JSON 응답으로 201이 나오는지 확인.
  - Location 헤더에 생성된 이벤트를 조회할 수 있는 URI 담겨 있는지 확인.
  - id는 DB에 들어갈 때 자동생성된 값으로 나오는지 확인
- 입력값으로 누가 id나 eventStatus, offline, free 이런 데이터까지 같이 주면?
  - Bad\_Request로 응답 vs **받기로 한 값 이외는 무시**
- 입력 데이터가 이상한 경우 Bad\_Request로 응답
  - 입력값이 이상한 경우 에러
  - 비즈니스 로직으로 검사할 수 있는 에러
  - 에러 응답 메시지에 에러에 대한 정보가 있어야 한다.
- 비즈니스 로직 적용 됐는지 응답 메시지 확인
  - offline과 free 값 확인
- 응답에 HATEOA와 profile 관련 링크가 있는지 확인.
  - self (view)
  - update (만든 사람은 수정할 수 있으니까)
  - events (목록으로 가는 링크)
- API 문서 만들기
  - 요청 문서화
  - 응답 문서화
  - 링크 문서화
  - profile 링크 추가

## 10. Event 생성 API 구현: 201 응답 받기

@RestController

- @ResponseBody를 모든 메소드에 적용한 것과 동일하다.

ResponseEntity를 사용하는 이유

- 응답 코드, 헤더, 본문 모두 다루기 편한 API

Location URI 만들기

- HATEOS가 제공하는 linkTo(), methodOn() 사용

객체를 JSON으로 변환

- ObjectMapper 사용

테스트 할 것

- 입력값들을 전달하면 JSON 응답으로 201이 나오는지 확인.
  - Location 헤더에 생성된 이벤트를 조회할 수 있는 URI 담겨 있는지 확인.
  - id는 DB에 들어갈 때 자동생성된 값으로 나오는지 확인

## 11. Event 생성 API 구현: EventRepository 구현

스프링 데이터 JPA

- JpaRepository 상속 받아 만들기

Enum을 JPA 맵핑시 주의할 것

- @Enumerated(EnumType.STRING)

@MockBean

- Mockito를 사용해서 mock 객체를 만들고 빈으로 등록해 줌.
- (주의) 기존 빈을 테스트용 빈이 대체 한다.

테스트 할 것

- 입력값들을 전달하면 JSON 응답으로 201이 나오는지 확인.
  - Location 헤더에 생성된 이벤트를 조회할 수 있는 URI 담겨 있는지 확인.
  - id는 DB에 들어갈 때 자동생성된 값으로 나오는지 확인

## 12. Event 생성 API 구현: 입력값 제한하기

### 입력값 제한

- id 또는 입력 받은 데이터로 계산해야 하는 값들은 입력을 받지 않아야 한다.
- EventDto 적용

### DTO -> 도메인 객체로 값 복사

- ModelMapper

```
<dependency>
  <groupId>org.modelmapper</groupId>
  <artifactId>modelmapper</artifactId>
  <version>2.3.1</version>
</dependency>
```

### 통합 테스트로 전환

- @WebMvcTest 빼고 다음 애노테이션 추가
  - @SpringBootTest
  - @AutoConfigureMockMvc
- Repository @MockBean 코드 제거

### 테스트 할 것

- 입력값으로 누가 id나 eventStatus, offline, free 이런 데이터까지 같이 주면?
  - Bad\_Request로 응답 vs **받기로 한 값 이외는 무시**

## 13. Event 생성 API 구현: 입력값 이외에 에러 발생

### ObjectMapper 커스터마이징

- spring.jackson.deserialization.fail-on-unknown-properties=true

### 테스트 할 것

- 입력값으로 누가 id나 eventStatus, offline, free 이런 데이터까지 같이 주면?
  - **Bad\_Request로 응답** vs 받기로 한 값 이외는 무시

## 14. Event 생성 API 구현: Bad Request 처리하기

@Valid와 BindingResult (또는 Errors)

- BindingResult는 항상 @Valid 바로 다음 인자로 사용해야 함. (스프링 MVC)
- @NotNull, @NotEmpty, @Min, @Max, ... 사용해서 입력값 바인딩할 때 에러 확인할 수 있음

도메인 Validator 만들기

- [Validator](#) 인터페이스 없이 만들어도 상관없음

테스트 설명 용 애노테이션 만들기

- @Target, @Retention

테스트 할 것

- 입력 데이터가 이상한 경우 Bad\_Request로 응답
  - 입력값이 이상한 경우 에러
  - 비즈니스 로직으로 검사할 수 있는 에러
  - 에러 응답 메시지에 에러에 대한 정보가 있어야 한다.

## 15. Event 생성 API 구현: Bad Request 응답 본문 만들기

### 커스텀 JSON Serializer 만들기

- extends JsonSerializer<T> (Jackson JSON 제공)
- @JsonComponent (스프링 부트 제공)

### BindingError

- FieldError 와 GlobalError (ObjectError)가 있음
- objectName
- defaultMessage
- code
- field
- rejectedValue

### 테스트 할 것

- 입력 데이터가 이상한 경우 Bad\_Request로 응답
  - 입력값이 이상한 경우 에러
  - 비즈니스 로직으로 검사할 수 있는 에러
  - 에러 응답 메시지에 에러에 대한 정보가 있어야 한다.



## 16. Event 생성 API 구현: 비즈니스 로직 적용

테스트 할 것

- 비즈니스 로직 적용 됐는지 응답 메시지 확인
  - offline과 free 값 확인

## 17. Event 생성 API 구현: 매개변수를 이용한 테스트

테스트 코드 리팩토링

- 테스트에서 중복 코드 제거
- 매개변수만 바꿀 수 있으면 좋겠는데?
- JUnitParams

JUnitParams

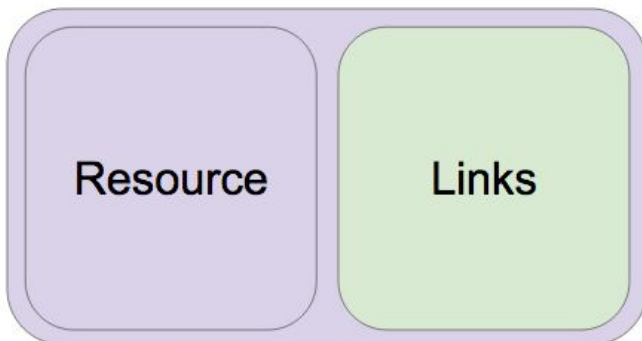
- <https://github.com/Pragmatists/JUnitParams>

```
<!-- https://mvnrepository.com/artifact/pl.pragmatists/JUnitParams -->
<dependency>
  <groupId>pl.pragmatists</groupId>
  <artifactId>JUnitParams</artifactId>
  <version>1.1.1</version>
  <scope>test</scope>
</dependency>
```

## 18. 스프링 HATEOAS 소개

### 스프링 HATEOAS

- <https://docs.spring.io/spring-hateoas/docs/current/reference/html/>
- 링크 만드는 기능
  - 문자열 가지고 만들기
  - 컨트롤러와 메소드로 만들기
- 리소스 만드는 기능
  - 리소스: 데이터 + 링크
- 링크 찾아주는 기능
  - Traverson
  - LinkDiscoverers
- 링크
  - HREF
  - REL
    - self
    - profile
    - update-event
    - query-events



## 19. 스프링 HATEOAS 적용

### EvnetResource 만들기

- extends ResourceSupport의 문제
  - @JsonUnwrapped로 해결
  - extends Resource<T>로 해결

### 테스트 할 것

- 응답에 HATEOA와 profile 관련 링크가 있는지 확인.
  - self (view)
  - update (만든 사람은 수정할 수 있으니까)
  - events (목록으로 가는 링크)

## 20. 스프링 REST Docs 소개

<https://docs.spring.io/spring-restdocs/docs/2.0.2.RELEASE/reference/html5/>

### REST Docs 코딩

- andDo(document("doc-name", snippets))
- snippets
  - links()
  - requestParameters() + parameterWithName()
  - pathParameters() + parametersWithName()
  - requestParts() + partWithName()
  - requestPartBody()
  - requestPartFields()
  - requestHeaders() + headerWithName()
  - requestFields() + fieldWithPath()
  - responseHeaders() + headerWithName()
  - responseFields() + fieldWithPath()
  - ...
- Relaxed\*
- Processor
  - preprocessRequest(prettyPrint())
  - preprocessResponse(prettyPrint())
  - ...

### Constraint

- <https://github.com/spring-projects/spring-restdocs/blob/v2.0.2.RELEASE/samples/rest-notes-spring-hateoas/src/test/java/com/example/notes/ApiDocumentation.java>

## 21. 스프링 REST Docs 적용

REST Docs 자동 설정

- `@AutoConfigureRestDocs`

RestDocMockMvc 커스터마이징

- `RestDocsMockMvcConfigurationCustomizer` 구현한 빈 등록
- `@TestConfiguration`

테스트 할 것

- API 문서 만들기
  - 요청 본문 문서화
  - 응답 본문 문서화
  - 링크 문서화
    - profile 링크 추가
  - 응답 헤더 문서화

## 22. 스프링 REST Docs: 링크, (Req, Res) 필드와 헤더 문서화

### 요청 필드 문서화

- requestFields() + fieldWithPath()
- responseFields() + fieldWithPath()
- requestHeaders() + headerWithName()
- responseHedaers() + headerWithName()
- links() + linkWithRel()

### 테스트 할 것

- API 문서 만들기
  - 요청 본문 문서화
  - 응답 본문 문서화
  - 링크 문서화
    - self
    - query-events
    - update-event
    - profile 링크 추가
  - 요청 헤더 문서화
  - 요청 필드 문서화
  - 응답 헤더 문서화
  - 응답 필드 문서화

### Relaxed 접두어

- 장점: 문서 일부분만 테스트 할 수 있다.
- 단점: 정확한 문서를 생성하지 못한다.

## 23. 스프링 REST Docs: 문서 빌드

### 스프링 REST Docs

- <https://docs.spring.io/spring-restdocs/docs/2.0.2.RELEASE/reference/html5/>
- pom.xml에 메이븐 플러그인 설정

```
<plugin>
  <groupId>org.asciidoctor</groupId>
  <artifactId>asciidoctor-maven-plugin</artifactId>
  <version>1.5.3</version>
  <executions>
    <execution>
      <id>generate-docs</id>
      <phase>prepare-package</phase>
      <goals>
        <goal>process-asciidoc</goal>
      </goals>
      <configuration>
        <backend>html</backend>
        <doctype>book</doctype>
      </configuration>
    </execution>
  </executions>
  <dependencies>
    <dependency>
      <groupId>org.springframework.restdocs</groupId>
      <artifactId>spring-restdocs-asciidoctor</artifactId>
      <version>2.0.2.RELEASE</version>
    </dependency>
  </dependencies>
</plugin>
<plugin>
  <artifactId>maven-resources-plugin</artifactId>
  <version>2.7</version>
  <executions>
    <execution>
      <id>copy-resources</id>
      <phase>prepare-package</phase>
      <goals>
        <goal>copy-resources</goal>
      </goals>
      <configuration>
        <outputDirectory>
          ${project.build.outputDirectory}/static/docs
        </outputDirectory>
      </configuration>
    </execution>
  </executions>
</plugin>
```

```

        <resources>
            <resource>
                <directory>

${project.build.directory}/generated-docs
                </directory>
            </resource>
        </resources>
    </configuration>
</execution>
</executions>
</plugin>

```

- 템플릿 파일 추가
  - src/main/asciidoc/index.adoc

#### 문서 생성하기

- mvn package
  - test
  - prepare-package :: process-asciidoc
  - prepare-package :: copy-resources
- 문서 확인
  - /docs/index.html

#### 테스트 할 것

- API 문서 만들기
  - 요청 본문 문서화
  - 응답 본문 문서화
  - 링크 문서화
    - self
    - query-events
    - update-event
    - profile 링크 추가
  - 요청 헤더 문서화
  - 요청 필드 문서화
  - 응답 헤더 문서화
  - 응답 필드 문서화

## 24. PostgreSQL 적용

테스트 할 때는 계속 H2를 사용해도 좋지만 애플리케이션 서버를 실행할 때 PostgreSQL을 사용하도록 변경하자.

/scripts.md 참고

### 1. PostgreSQL 드라이버 의존성 추가

```
<dependency>
  <groupId>org.postgresql</groupId>
  <artifactId>postgresql</artifactId>
</dependency>
```

### 2. 도커로 PostgreSQL 컨테이너 실행

```
docker run --name ndb -p 5432:5432 -e POSTGRES_PASSWORD=pass -d postgres
```

### 3. 도커 컨테이너에 들어가보기

```
docker exec -i -t ndb bash
su - postgres
psql -d postgres -U postgres
\l
\dt
```

### 4. 데이터소스 설정

application.properties

```
spring.datasource.username=postgres
spring.datasource.password=pass
spring.datasource.url=jdbc:postgresql://localhost:5432/postgres
spring.datasource.driver-class-name=org.postgresql.Driver
```

### 5. 하이버네이트 설정

application.properties

```
spring.jpa.hibernate.ddl-auto=create-drop
spring.jpa.properties.hibernate.jdbc.lob.non_contextual_creation=true
spring.jpa.properties.hibernate.format_sql=true
logging.level.org.hibernate.SQL=DEBUG
```



```
logging.level.org.hibernate.type.descriptor.sql.BasicBinder=TRACE
```

애플리케이션 설정과 테스트 설정 중복 어떻게 줄일 것인가?

- 프로파일과 @ActiveProfiles 활용

application-test.properties

```
spring.datasource.username=sa
spring.datasource.password=
spring.datasource.url=jdbc:h2:mem:testdb
spring.datasource.driver-class-name=org.h2.Driver

spring.datasource.hikari.jdbc-url=jdbc:h2:mem:testdb

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.H2Dialect
```

## 25. 인덱스 핸들러 만들기

### 인덱스 핸들러

- 다른 리소스에 대한 링크 제공
- 문서화

```
@GetMapping("/api")
public ResourceSupport root() {
    ResourceSupport index = new ResourceSupport();
    index.add(linkTo(EventController.class).withRel("events"));
    return index;
}
```

### 테스트 컨트롤러 리팩토링

- 중복 코드 제거

### 여러 리소스

- 인덱스로 가는 링크 제공

## 26. Event 목록 조회 API

페이징, 정렬 어떻게 하지?

- 스프링 데이터 JPA가 제공하는 Pageable

Page<Event>에 안에 들어있는 Event 들은 리소스로 어떻게 변경할까?

- 하나씩 순회하면서 직접 EventResource로 매핑을 시킬까..
- PagedResourceAssembler<T> 사용하기

테스트 할 때 Pageable 파라미터 제공하는 방법

- page: 0부터 시작
- size: 기본값 20
- sort: property,property(,ASC|DESC)

테스트 할 것

- Event 목록 Page 정보와 함께 받기
  - content[0].id 확인
  - pageable 경로 확인
- Sort과 Paging 확인
  - 30개를 만들고, 10개 사이즈로 두번째 페이지 조회하면 이전, 다음 페이지로 가는 링크가 있어야 한다.
  - 이벤트 이름순으로 정렬하기
  - page 관련 링크
- Event를 EventResource로 변환해서 받기
  - 각 이벤트 마다 self
- 링크 확인
  - self
  - profile
  - (create)
- 문서화

## 27. Event 조회 API

테스트 할 것

조회하는 이벤트가 있는 경우 이벤트 리소스 확인

- 링크
  - self
  - profile
  - (update)
- 이벤트 데이터

조회하는 이벤트가 없는 경우 404 응답 확인

## 28. Events 수정 API

테스트 할 것

수정하려는 이벤트가 없는 경우 404 NOT\_FOUND  
입력 데이터 (데이터 바인딩)가 이상한 경우에 400 BAD\_REQUEST  
도메인 로직으로 데이터 검증 실패하면 400 BAD\_REQUEST  
(권한이 충분하지 않은 경우에 403 FORBIDDEN)  
정상적으로 수정한 경우에 이벤트 리소스 응답

- 200 OK
- 링크
- 수정한 이벤트 데이터

## 29. 테스트 코드 리팩토링

여러 컨트롤러 간의 중복 코드 제거하기

- 클래스 상속을 사용하는 방법
- @Ignore 애노테이션으로 테스트로 간주되지 않도록 설정

## 30. Account 도메인 추가

OAuth2로 인증을 하려면 일단 Account 부터

- id
- email
- password
- roles

AccountRoles

- ADMIN, USER

JPA 맵핑

- @Table("Users")

JPA enumeration collection mapping

```
@ElementCollection(fetch = FetchType.EAGER)
@Enumerated(EnumType.STRING)
private Set<AccountRole> roles;
```

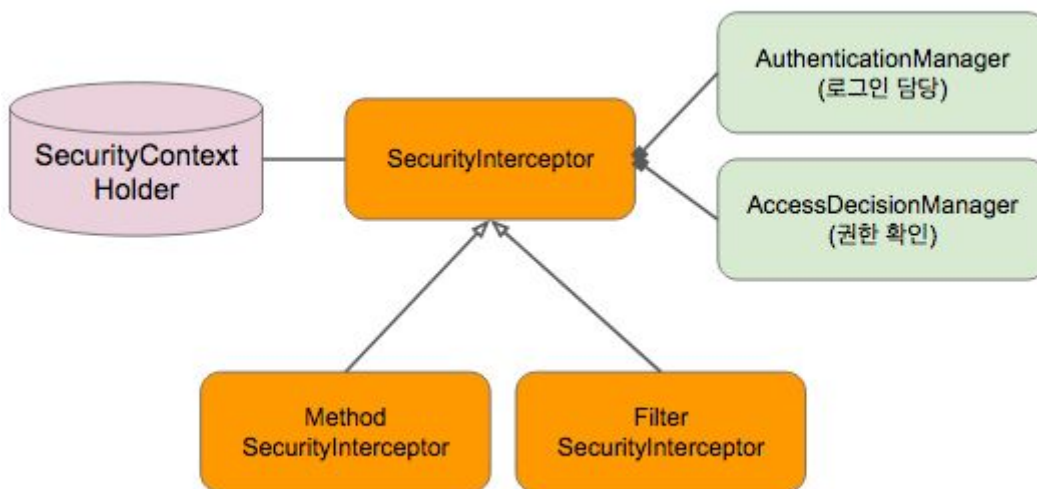
Event에 owner 추가

```
@ManyToOne
Account manager;
```

## 31. 스프링 시큐리티

### 스프링 시큐리티

- 웹 시큐리티 (Filter 기반 시큐리티)
- 메소드 시큐리티
- 이 둘 다 Security Interceptor를 사용합니다.
  - 리소스에 접근을 허용할 것이냐 말 것이냐를 결정하는 로직이 들어있음.



### 의존성 추가

```
<dependency>
  <groupId>org.springframework.security.oauth.boot</groupId>
  <artifactId>spring-security-oauth2-autoconfigure</artifactId>
  <version>2.1.0.RELEASE</version>
</dependency>
```

### 테스트 다 깨짐 (401 Unauthorized)

- 깨지는 이유는 스프링 부트가 제공하는 스프링 시큐리티 기본 설정 때문.



## 32. 예외 테스트

### 1. @Test(expected)

예외 타입만 확인 가능

### 2. try-catch

예외 타입과 메시지 확인 가능.  
하지만 코드가 다소 복잡.

### 3. @Rule ExpectedException

코드는 간결하면서 예외 타입과 메시지 모두 확인 가능

## 33. 스프링 시큐리티 기본 설정

시큐리티 필터를 적용하기 않음...

- /docs/index.html

로그인 없이 접근 가능

- GET /api/events
- GET /api/events/{id}

로그인 해야 접근 가능

- 나머지 다...
- POST /api/events
- PUT /api/events/{id}
- ...

스프링 시큐리티 OAuth 2.0

- AuthorizationServer: OAuth2 토큰 발행(/oauth/token) 및 토큰 인증(/oauth/authorize)
  - Oder 0 (리소스 서버 보다 우선 순위가 높다.)
- ResourceServer: 리소스 요청 인증 처리 (OAuth 2 토큰 검사)
  - Oder 3 (이 값은 현재 고칠 수 없음)

스프링 시큐리티 설정

- @EnableWebSecurity
- @EnableGlobalMethodSecurity
- extends WebSecurityConfigurerAdapter
- PasswordEncoder: PasswordEncoderFactories.createDelegatingPasswordEncoder()
- TokenStore: InMemoryTokenStore
- AuthenticationManagerBean
- configure(AuthenticationManagerBuidler auth)
  - userDetailsService
  - passwordEncoder
- configure(HttpSecurity http)
  - /docs/\*\*: permitAll
- configure(WebSecurty web)
  - ignore
    - /docs/\*\*
    - /favicon.ico
- PathRequest.toStaticResources() 사용하기

## 34. 스프링 시큐리티 폼 인증 설정

```
@Override
protected void configure(HttpSecurity http) throws Exception {
    http
        .anonymous()
        .and()
        .formLogin()
        .and()
        .authorizeRequests()
            .mvcMatchers(HttpMethod.GET, "/api/**").authenticated()
            .anyRequest().authenticated();
}
```

- 익명 사용자 사용 활성화
- 폼 인증 방식 활성화
  - 스프링 시큐리티가 기본 로그인 페이지 제공
- 요청에 인증 적용
  - /api 이하 모든 GET 요청에 인증이 필요함. (permitAll())을 사용하여 인증이 필요없이 익명으로 접근이 가능케 할 수 있음
  - 그밖에 모든 요청도 인증이 필요함.

## 35. 스프링 시큐리티 OAuth 2 설정: 인증 서버 설정

```
<dependency>
  <groupId>org.springframework.security</groupId>
  <artifactId>spring-security-test</artifactId>
  <version>${spring-security.version}</version>
  <scope>test</scope>
</dependency>
```

### 토큰 발행 테스트

- User
- Client
- POST /oauth/token
  - HTTP Basic 인증 헤더 (클라이언트 아이디 + 클라이언트 시크릿)
  - 요청 매개변수 (MultiValuMap<String, String>)
    - grant\_type: password
    - username
    - password
  - 응답에 access\_token 나오는지 확인

### Grant Type: Password

- Granty Type: 토큰 받아오는 방법
- 서비스 오너가 만든 클라이언트에서 사용하는 Grant Type
- <https://developer.okta.com/blog/2018/06/29/what-is-the-oauth2-password-grant>

### AuthorizationServer 설정

- @EnableAuthorizationServer
- extends AuthorizationServerConfigurerAdapter
- configure(AuthorizationServerSecurityConfigurer security)
  - PasswordEncode 설정
- configure(ClientDetailsServiceConfigurer clients)
  - 클라이언트 설정
  - grantTypes
    - password
    - refresh\_token
  - scopes
  - secret / name
  - accessTokenValiditySeconds
  - refreshTokenValiditySeconds
- AuthorizationServerEndpointsConfigurer
  - tokenStore
  - authenticationMaanger
  - userDetailsService

## 36. 스프링 시큐리티 OAuth 2 설정: 리소스 서버 설정

테스트 수정

- GET을 제외하고 모두 액세스 토큰을 가지고 요청 하도록 테스트 수정

ResourceServer 설정

- `@EnableResourceServer`
- `extends ResourceServerConfigurerAdapter`
- `configure(ResourceServerSecurityConfigurer resources)`
  - 리소스 ID
- `configure(HttpSecurity http)`
  - `anonymous`
  - `GET /api/** : permit all`
  - `POST /api/**: authenticated`
  - `PUT /api/**: authenticated`
  - 에러 처리
    - `accessDeniedHandler(OAuth2AccessDeniedHandler())`

## 37. 문자열을 외부 설정으로 빼내기

기본 유저 만들기

- `ApplicationRunner`
  - `Admin`
  - `User`

외부 설정으로 기본 유저와 클라이언트 정보 빼내기

- `@ConfigurationProperties`

## 38. 이벤트 API 점검

### 토큰 발급 받기

- POST /oauth/token
- BASIC authentication 헤더
  - client Id(myApp) + client secret(pass)
- 요청 본문 폼
  - username: admin@email.com
  - password: admin
  - grant\_type: password

METHOD: POST  
SCHEME // HOST [ : PORT ] [ PATH [ ? QUERY ] ]  
http://localhost:8080/oauth/token  
length: 33 bytes  
Send

QUERY PARAMETERS

HEADERS

- Authorization: Basic bmF0dXJhbC1jbGllbnQ6cGFz
- Content-Type: application/x-www-form-urlencoded

BODY

- grant\_type: password
- username: admin@email.com
- password: admin

### 토큰 갱신하기

- POST /oauth/token
- BASIC authentication 헤더
  - client Id(myApp) + client secret(pass)
- 요청 본문 폼
  - token: 처음에 발급받았던 refresh 토큰
  - grant\_type: refresh\_token

METHOD: POST  
SCHEME // HOST [ : PORT ] [ PATH [ ? QUERY ] ]  
http://localhost:8080/oauth/token  
length: 33 bytes  
Send

QUERY PARAMETERS

HEADERS

- Authorization: Basic bmF0dXJhbC1jbGllbnQ6cGFz
- Content-Type: application/x-www-form-urlencoded

BODY

- grant\_type: refresh\_token
- refresh\_token: 87d1eee9-c3c7-486a-9726-bed5dd89a38d

### 이벤트 목록 조회 API

- 로그인 했을 때
  - 이벤트 생성 링크 제공

### 이벤트 조회

- 로그인 했을 때
  - 이벤트 Manager인 경우에는 이벤트 수정 링크 제공

## 39. 스프링 시큐리티 현재 사용자

### SecurityContext

- 자바 ThreadLocal 기반 구현으로 인증 정보를 담고 있다.
- 인증 정보 꺼내는 방법:

```
Authentication authentication = SecurityContextHolder.getContext().getAuthentication();
```

### @AuthenticationPrincipal spring.security.User user

- 인증 안한 경우에 null
- 인증 한 경우에는 username과 authorities 참조 가능

### spring.security.User를 상속받는 클래스를 구현하면

- 도메인 User를 받을 수 있다.
- @AuthenticationPrincipal `me.whiteship.user.UserAdapter`
- `Adatepr.getUser().getId()`

### SpEL을 사용하면

- @AuthenticationPrincipal(expression="account") `me.whiteship.user.Account`

```
@Target(ElementType.PARAMETER)
@Retention(RetentionPolicy.RUNTIME)
@AuthenticationPrincipal(expression = "account")
public @interface CurrentUser {
}
```

### 커스텀 애노테이션을 만들면

- @CurrentUser Account account
- 엇? 근데 인증 안하고 접근하면..?

```
expression = "#{this == 'anonymousUser' ? null : account}"
```

- 현재 인증 정보가 anonymousUse 인 경우에는 null을 보내고 아니면 "account"를 꺼내준다.

### 조회 API 개선

- 현재 조회하는 사용자가 owner인 경우에 update 링크 추가 (HATEOAS)

### 수정 API 개선

현재 사용자가 이벤트 owner가 아닌 경우에 403 에러 발생



## 40. Events API 개선: 출력값 제한하기

생성 API 개선

- Event owner 설정
- 응답에서 owner의 id만 보내 줄 것.

```
{
  "id" : 4,
  "name" : "test 3PISM1Ju",
  "description" : "test event",
  ...
  "free" : false,
  "eventStatus" : "DRAFT",
  "owner" : {
    "id" : 3,
    "email" : "keesun@email.com",
    "password" : "{bcrypt}$2a$10$3z/rHmeYsKpoOQR3aUq38OmZjZNsrGfRZxSnmpLfl3lpLxjD5/JZ6",
    "roles" : [ "USER", "ADMIN" ]
  },
}
```

- JsonSerializer<User> 구현
- @JsonSerialize(using) 설정

## 41. 깨진 테스트 살펴보기

EventControllerTests.updateEvent()

- 깨지는 이유
- 해결 방법

EventControllerTests.getEvent()

- 깨지는 이유
- 해결 방법

DemoApplicationTests

- 깨지는 이유
- 해결 방법

## 42. 스프링 부트 2.2.5 버전으로 업데이트

현재 스프링 부트 최신 권장 버전은 2.2.5

- <https://spring.io/projects/spring-boot>

스프링 부트 버전을 올리면 바뀔 수 있는 일

- 기본 (자동) 설정 값 변경
- 의존성 변경
- 기존 애플리케이션의 동작이 바뀌거나 컴파일 에러가 발생할 수 있다.

스프링 부트 2.2.\* 주요 변화

- JUnit 4 -> JUnit 5
- 스프링 HATEOAS 버전이 올라가면서 스프링 HATEOAS의 API가 바뀌었다.

스프링 HATEOAS

- Resource -> EntityModel
- Resources -> CollectionModel
- PagedResources -> PagedModel
- ResourceSupport -> RepresentationModel
- assembler.toResource -> assembler.toModel
- org.springframework.hateoas.mvc.ControllerLinkBuilder -> org.springframework.hateoas.server.mvc.WebMvcLinkBuilder
- MediaTypes 중에 (UTF8)인코딩이 들어간 상수 제거.

JUnit 5

- org.junit -> org.junit.jupiter
- @Ignore -> @Disabled
- @Before, @After -> @BeforeEach, @AfterEach
- @TestDescription (우리가 만든거) -> @DisplayName

스프링 MVC 변경

- MediaType 중에 (UTF8)인코딩이 들어간 상수 deprecation.

지금까지 HATEOAS와 Self-Descriptive Message를 만족하는 REST API를 스프링 및 여러 오픈 소스 프로젝트를 사용해 구현했습니다.

감사합니다.