RESTful API설계/구현 가이드

2022.10.01

김동석

1. url : [redacted]>설계 > URL 네이밍 참고
2. path : 첫번째 path에는 버전 표기. 그 이후부터 설계 (ex. https://api.sett[redacted]co.kr/v1.0)
3. RESTful 원칙을 최대한 지킨다 (https://quickref.me/http-status-code#1xx-information)
    a. HTTP 상태 코드 : 표준 HTTP와 REST 규약을 따름
        i. 200 OK : 요청을 성공적으로 처리함
        ii. 201 Created : 새 리소스를 성공적으로 생성함. 응답의 Location 헤더에 해당 리소스의 URI가 담겨있다.
        iii. 204 No Content : 기존 리소스를 성공적으로 수정함. PUT에 대한 응답
        iv. 400 Bad Request : 잘못된 요청을 보낸 경우. 응답 본문에 더 오류에 대한 정보가 담겨있다.
        v. 403 *TODO*설명추가
        vi. 404 Not Found : 요청한 리소스가 없음.
        vii. 500 서버오류
    b. 동작에 해당하는 http method 사용
        i. GET : 조회
        ii. POST : 신규 저장
        iii. PUT : 기존 데이터 대체
        iv. PATCH : 일부 수정(특정 컬럼들만 수정)
        v. DELETE : 삭제
    c. 복수형 조회시 반드시 리소스명도 복수로 한다(주소에 표현됨)
        i. user 👉 users
        ii. album 👉 albums
        iii. history 👉 histories
    d. 리소스의 상하/분류 혹은 소유관계가 엔드포인트 주소에 표현되어야 함
       왼쪽 👉 상위/부모, 오른쪽 👉 하위/자식 *TODOjava controller*의 패키지를 기본 *url*로 사용
        i. 소유/선후관계 표현 샘플
            1. Good😀
               ~/owners/1/albums/2/tracks
               ~/albums/2/tracks
            2. Bad😓 : ~/albums/2/owner
        ii. 그루핑의 목적으로 중간 경로를 추가하는 경우 샘플
            1. 자산 : ~/**assets**/albums, ~/**assets**/videos
            2. 인증/인가 관련 : ~/**secu**/auths, ~/**secu**/groups
            3. 공통 : ~/**comm**/codes, ~/**comm**/dspDomains
    e.
4. 단순CRUD가 아닌 복잡한 업무에 대한 end point가 필요할 때 url 규칙
   동사(+'/' +명사)형태로 작성 👉 java controller method명과 가능하면 일치시킴
    a. 하위가 있는 경우 동사를 path에 넣을 수 있다 ex) ~/users/1/**change**/status, ~/users/1/**change**/position,
    b. 하위가 없는 경우 동사로 주소가 끝날 수 있다 ex) ~/messages/**sendAlimTalk**, ~/messages/**download**
5. 목록 페이징 api
    a. pageNum : 페이지 번호(기본값은 1)
    b. limit : 리스트 중 가져올 개수(기본값은 10)
    c. orderBy : 정렬옵션. 정렬 컬럼명 + "," + 정렬방향 ex) …&**orderBy=id,down**&**orderBy=name,up**…
        i. enum으로 정렬문제
    d. searchWord : 기본 키워드 검색용 파라미터
    e. 기타 검색용 파라미터들은 api별로 정의해서 사용
       ex) albums?pageNum=12&limit=10&searchWord=asdf&searchAlbumName=asdf&orderBy=id,desc
6. 데이터 모델 제약 사항
    a. 모델을 담는 용도로 hashmap은 적절하지 않음.

b. 가능하면 java객체(DTO)를 사용하는 것이 오타에 의한 실수를 줄일 수 있고 가독성을 높여 이해하기 쉬움
c. 가능하면 도메인 모델을 전송용으로 사용하지 않도록 함 👉 참고 : DTO vs Domain Models 특히 수정요청용 DTO는 도메인 모델과 필드는 동일하더라도 유효성검사가 틀린 경우가 많아 같이 쓸 수 없음
d. 모델, DTO의 멤버들은 가능하면 객체를 사용 : null일 때 처리를 다르게 할 수 있음

7. patch(Partial Update)전략(참고 :테이블 데이터 부분 갱신 가이드)
객체 정보 모두를 갱신하지 않고 부분적으로 갱신하는 경우 해당 필드를 반드시 포함해야 함.
* undefined면 갱신하지 않음
* null로 넘겨줬다면 null로 갱신
* empty로 넘겨줬다면 empty(“”)로 갱신
※ 참고
   a. https://datatracker.ietf.org/doc/html/rfc7386
   b. https://github.com/OpenAPITools/jackson-databind-nullable
   c. https://kdrozd.pl/how-to-perform-a-partial-update-patch-with-explicit-null/
   d. https://www.mscharhag.com/api-design/rest-partial-updates-patch
8. 리턴 코드/메시지 TODO
   a. 메시지 형식 정의
      i. 메시지 코드, 메시지 : 시스템과 업무별로 정의
   b. 공통 오류(Exception) 목록
      i. 권한 문제
      ii. 오류
      iii. …
9. 고려할 사항
   a. ~~파일 업로드/다운로드~~ 👉 독립 인스턴스 구현 예정
      ~~i. 어떻게 하면 가장 깔끔하게 구현 할 수 있을까~~
      ~~ii. 3rd party업로더를 사용한다면 어떻게 연동해야 할까~~
      ~~iii. 파일 다운로드 어뷰징 방지 방안 필요~~
10. swagger3 + spring boot
   a. @NotBlank 인 것들은 모두 @NotNull 혹은 @NotEmpty 추가(collection인 경우)
      그래야 api-docs 의 required에 뜸

```
- PrepaidContractPostReq: {
  - required: [
      "name",
      "payments",
      "rightId",
      "targetAssetList",
      "term",
      "typeAdvancePay"
    ],
    type: "object",
  - properties: {
    - name: {
        maxLength: 16,
        minLength: 1,
        type: "string",
```

   b. 유효성 검사
      i. 메시지 샘플
         1. "기간 시작시점은 필수입니다"
         2. "권리자 ID는 1이상이어야 합니다"
         3. "담당자 (매니저) 최대 길이는 45자 입니다"
         4. @Size(min=4, max=200, message="이메일 최소 길이는 4자, 최대 길이는 200자 입니다")
         5. “이메일 형식이 아닙니다”
         6. @Size(min=4, max=4, message="DSP (판매처) 코드 길이는 4자 입니다")
11. 부록
   a. Spring Initializr

b. swagger3 참고

RestController의 get method에서 모델을 파라미터로 받기

1. 모델 파라미터 샘플

```
public class RightHolderListReq {
  private String brandCode;
  private String arr[];
  private List<String> list;
  Pagination pag ;
}
```

2. Controller 메소드 샘플

```
  @GetMapping("rightsHolders")
  ResponseEntity<RightHolderListReq> settRuntimeExceptionTest(RightHolderListReq
rightHolderListReq) {
      return new ResponseEntity<>(rightHolderListReq, HttpStatus.OK);
  }
```

3. 호출 샘플
http://10.1.40.53:9111/v0.1/rightsHolders?brandCode=br1&arr=arr1&arr=arr2&list=list1&list=list2&pag.limit=99&pag.searchWord=asdf

4. 결과

```
{
  "brandCode": "br1",
  "arr": [
    "arr1",
    "arr2"
  ],
  "list": [
    "list1",
    "list2"
  ],
  "pag": {
    "limit": 99,
    "dataCnt": 0,
    "searchWord": "asdf",
    "orderColumn": null,
    "orderDirection": null
  }
}
```

Spring Boot 2.6.7 + springdoc-openapi-ui 1.6.9

1. 설정
    a. build.gradle에 추가

    ```
    implementation group: 'org.springdoc', name: 'springdoc-openapi-ui', version: '1.6.9'
    ```

    b. ~~운영환경에서는 swagger띄우지 않기 : application-prod.yml에 추가~~ 👉 클라이언트 측 유효성 검사를 위해 openapi 정의가 필요해서 운영에서도 띄워야 함

    ```
    springdoc:
      api-docs:
        enabled: false
    ```

2. 기동 확인 : http://localhost:9101/swagger-ui/index.html
3. springdoc-openapi는 JSR-303을 지원하므로 openapi 전용 어노테이션 보다는 JSR-303 어노테이션을 최대한 활용한다(그러나 복잡한 비즈니스 로직 제약사항은 springdoc-openapi에서 추출 불가 -.-)
4. 주요 어노테이션
    a. @NotNull
    b. @Min(202208)
    c. @Size(min=1, max=60, message="이름을 입력하세요.")

파일 압축 & 다운로드

1. 한글 파일명 인코딩 문제로 java.util.zip을 사용하지 말고 zip4j사용
2. 압축하여 바로 다운로드가 필요한 경우 파일을 저장하지 말고 stream으로 바로 전송해야 파일이 서버에 남지 않음

```
try(ZipOutputStream zipOutputStream = new
ZipOutputStream(servletResponse.getOutputStream())) {
    for(File fileToAdd : new File(resFolderPath).listFiles()) {
       ZipParameters zipParameters = new ZipParameters();
       zipParameters.setFileNameInZip(fileToAdd.getName());
       zipOutputStream.putNextEntry(zipParameters);

       try (InputStream inputStream = new FileInputStream(fileToAdd)) {
          int readLen;
          while ((readLen = inputStream.read(ZIP_FILE_READ_BUFFER)) != -1) {
             zipOutputStream.write(ZIP_FILE_READ_BUFFER, 0, readLen);
          }
       }
       zipOutputStream.closeEntry();
    }
}
```

3.

참고

1. RESTful API문서를 pdf로 만들기