

올인원 패키지 Online.

안녕하세요

Spring RestDocs

강사 김남윤 입니다.

PART1 | 프로젝트 구성하기

- 프로젝트 구성하기 1
- 프로젝트 구성하기 2
- 프로젝트 구성하기 3

PART2 | REST Docs 문서 만들기

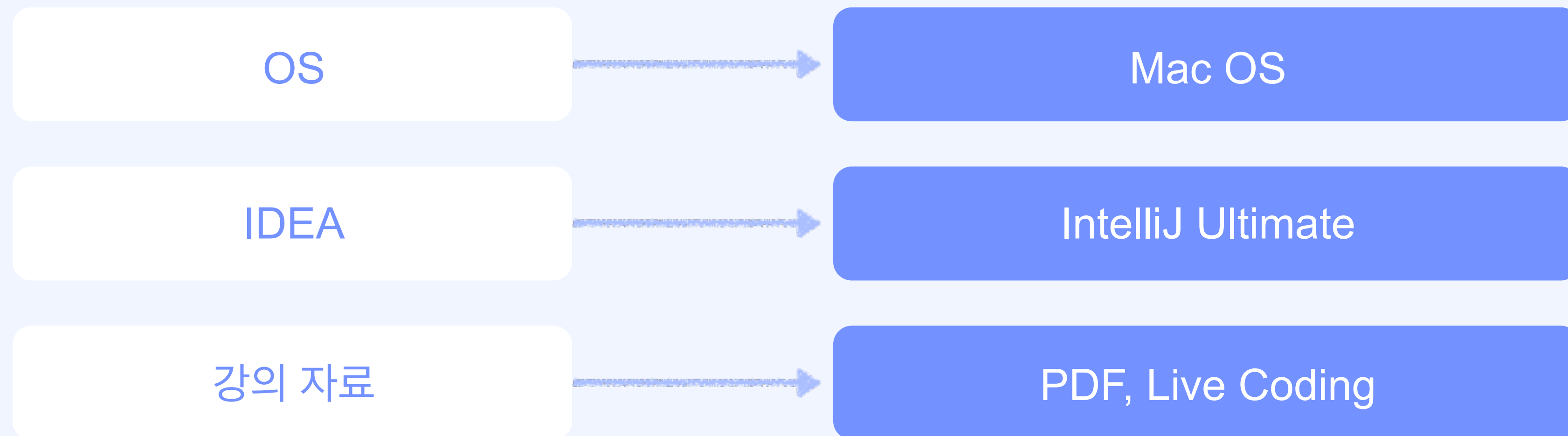
- REST Docs 살펴보기
- 문서화에 필요한 API 만들기
- Member API REST Docs 만들기

PART3 | REST Docs 커스텀하기

- 테스트 코드 개선
- 문서 커스텀

강의 소개

강의 환경



선행 지식

- Spring Web MVC 프레임 워크로 간단한 Restful API 개발 가능한 수준
- JPA 기초
- 테스트 코드에 대한 기초적인 지식

기술 스택

- Java8 & Lombok 1.18.20
- Spring Boot 2.5.2
- Spring Web MVC 5.3.8
- Spring Restdocs Mockmvc 2.0.5
- Spring Data JPA 2.5.2
- Spring Test 5.3.8
- H2 1.4.200
- Junit5
- Greadle 7.x.x

REST Docs 살펴보기 Part 1

RestDocs 간단 소개

- * Rest Docs는 테스트 코드 기반으로 Restfull API 문서를 돕는 도구입니다.
- * AsciiDoctor를 이용해서 HTML 등등 다양한 포맷으로 문서를 자동으로 출력할 수 있습니다.
- * RestDocs의 가장 큰 장점이자 다른 문서화 도구의 차이점은 테스트 코드 기반으로 문서를 작성한다는 것입니다.
- * API Spec과 문서화를 위한 테스트 코드가 일치하지 않으면 테스트 빌드를 실패하게 되어 테스트 코드로 검증된 문서를 보장할 수 있습니다.

RestDocs VS Swagger

```
@ApiOperation(
    value = "members 조회",
    response = Member.class,
    responseContainer = "List"
)
@ApiResponses(
    value = {
        @ApiResponse(code = 200, message = "응답 성공"),
        @ApiResponse(code = 400, message = "유효하지 하지 않은 입력")
    }
)
@GetMapping("/swagger")
public List<Member> getMember2(
    @ApiParam(name = "page", value = "page", defaultValue = "0")
    @RequestParam Integer page,
    @ApiParam(name = "size", value = "size", defaultValue = "10")
    @RequestParam Integer size
) {
    final List<Member> members = new ArrayList<>();
    members.add(new Member(email: "yun@asd.com", name: "yun"));
    return members;
}
```

Swagger Code

```
@GetMapping
public PageResponse<MemberResponse> getMembers(
    @PageableDefault(sort = "id", direction = Direction.DESC) Pageable pageable
) {
    return new PageResponse<>(memberRepository.findAll(pageable).map(MemberResponse::new));
}
```

REST Docs Code

REST Docs 문서를 신뢰할 수 있는 이유

```
@Test
public void member_get() throws Exception {
    mockMvc.perform(
        get(urlTemplate: "/api/members/{id}", ...urlVariables: "1")
        .contentType(MediaType.APPLICATION_JSON)
    )
    .andExpect(status().isOk())
    .andDo(
        restDocs.document(
            pathParameters(
                parameterWithName("id").description("Member id")
            ),
            responseFields(
                fieldWithPath("id").description("Member id"),
                fieldWithPath("email").description("Email"),
                fieldWithPath("name").description("Name"),
                fieldWithPath("status").description("회원상태"),
                fieldWithPath("createdAt").description("생성 일시"),
                fieldWithPath("updatedAt").description("수정 일시")
            )
        )
    );
}
```

```
Tests failed: 1 of 1 test - 428 ms
Body = [{"id":null,"email":"yun@asd.com","nickname":"yun","createdAt":"2021-07-05T21:53:16.195655","updatedAt":null}]
Forwarded URL = null
Redirected URL = null
Cookies = []

The following parts of the payload were not documented:
[ {
  "nickname" : "yun"
} ]
Fields with the following paths were not found in the payload: [[0].name]
org.springframework.restdocs.snippet.SnippetException: The following parts of the payload were not documented:
[ {
  "nickname" : "yun"
} ]
Fields with the following paths were not found in the payload: [[0].name]
at org.springframework.restdocs.payload.AbstractFieldsSnippet.validateFieldDocumentation(AbstractFieldsSnippet.java:218)
at org.springframework.restdocs.payload.AbstractFieldsSnippet.createModel(AbstractFieldsSnippet.java:160)
at org.springframework.restdocs.snippet.TemplateSnippet.document(TemplateSnippet.java:78)
at org.springframework.restdocs.generate.RestDocumentationGenerator.handle(RestDocumentationGenerator.java:191)
at org.springframework.restdocs.mockmvc.RestDocumentationResultHandler.handle(RestDocumentationResultHandler.java:100)
at org.springframework.test.web.servlet.MockMvc$1.andDo(MockMvc.java:201)
```

REST Docs 최종 결과물

REST Docs

Andy Wilkinson – Version 0.0.1-SNAPSHOT

Member API

member 조회

```
$ curl 'http://localhost:8080/api/members/restdocs?size=10&page=0' -i -X GET \
-H 'Accept: application/json'
```

```
GET /api/members/restdocs?size=10&page=0 HTTP/1.1
Accept: application/json
Host: localhost:8080
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 130
```

```
[{"id":null,"email":"yun@asd.com","name":"yun","createdAt":"2021-07-05T02:44:33.132037","updatedAt"
```

```
$ http GET 'http://localhost:8080/api/members/restdocs?size=10&page=0' \
'Accept:application/json'
```

BASH

Parameter	Description
size	size
page	page

```
": "yun@asd.com", "name": "yun", "createdAt": "2021-07-05T02:44:33.132037", "updatedAt": "2021-07-05T02:44:3
```

REST Docs 문서만들기

1 프로젝트 구성하기

Github



- <https://github.com/cheese10yun/spring-rest-docs>
- branch: init-project

build.gradle.kts

```
dependencies {
    implementation("org.springframework.boot:spring-boot-starter-web")
    implementation("org.springframework.boot:spring-boot-starter-actuator")
    implementation("org.springframework.boot:spring-boot-starter-validation")
    implementation("org.springframework.boot:spring-boot-starter-data-jpa")
    compileOnly("org.projectlombok:lombok")
    runtimeOnly("com.h2database:h2")
    annotationProcessor("org.projectlombok:lombok")
    testImplementation("org.springframework.boot:spring-boot-starter-test")
    testImplementation("org.springframework.restdocs:spring-restdocs-mockmvc")
}

tasks.withType<Test> {
    useJUnitPlatform()
}
```

```
plugins {
    id("org.springframework.boot") version "2.5.2"
    id("io.spring.dependency-management") version "1.0.11.RELEASE"
    id("org.asciidoctor.jvm.convert") version "3.1.0"
    id("java")
}

group = "com.rest.docs"
version = "0.0.1-SNAPSHOT"

java.sourceCompatibility = JavaVersion.VERSION_1_8

configurations {
    compileOnly {
        extendsFrom(configurations.annotationProcessor.get())
    }
}

repositories {
    mavenCentral()
}
```

build.gradle.kts



- [Gist: build.gradle.kts](#)

REST Docs 문서만들기

2 문서화에 필요한 API만들기

- 1. Data Set Up
 - 1. properties 설정
 - 2. Member 엔티티 만들기
 - 3. repositroy 만들기
 - 4. application runner 만들기
- 2. Member API
 - 1. Member 등록
 - 2. Member 수정
 - 3. Member 단일 조회
 - 4. Member 페이징 조회

REST Docs 문서만들기

3 Member API REST Docs 만들기

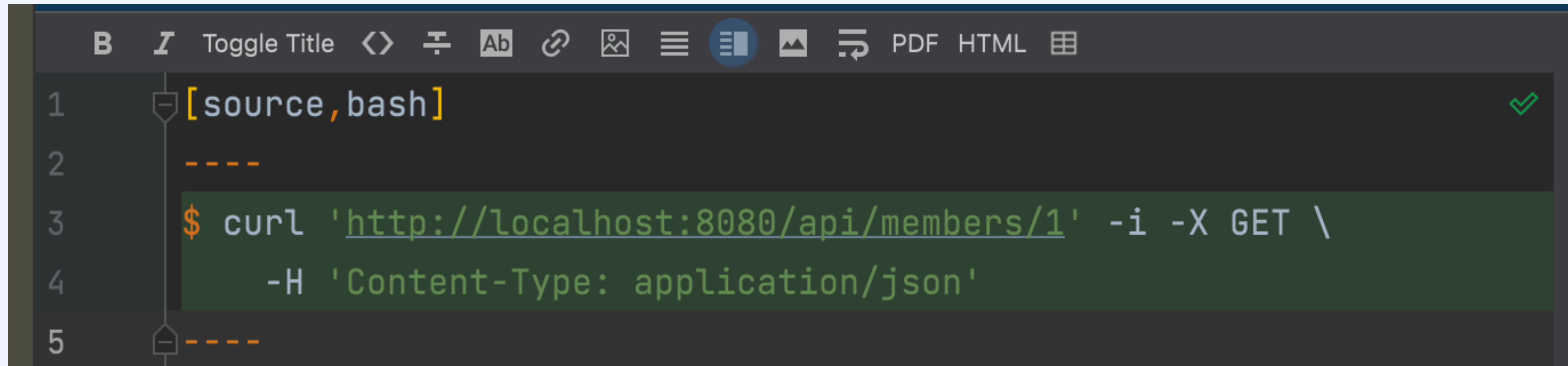
자동으로 생성되는 Snippets

- curl-request.adoc
- http-request.adoc
- http-response.adoc
- httpie-request.adoc
- request-body.adoc
- response-body.adoc

curl-request.adoc

2.

REST Docs 문서만들기



The screenshot shows a code editor with a dark theme. The editor has a toolbar at the top with various icons for editing and viewing. The main content area displays a document with line numbers 1 through 5 on the left. The document content is as follows:

```
1 [source, bash] ✓
2 -----
3 $ curl 'http://localhost:8080/api/members/1' -i -X GET \
4     -H 'Content-Type: application/json'
5 -----
```

http-request.adoc

```
1  [source,http,options="nowrap"]
2  ----
3  POST /api/members HTTP/1.1
4  Content-Type: application/json
5  Content-Length: 45
6  Host: localhost:8080
7
8  {
9      "email": "bbb@bbb.com",
10     "name": "bbb"
11 }
12  ----
```

http-response.adoc

```

1  [source,http,options="nowrap"]
2  -----
3  HTTP/1.1 200 OK
4  Content-Type: application/json
5  Content-Length: 47
6
7  {"id":1,"email":"yun@bbb.com","name":"new-yun"}
8  -----

```

httpie-request.adoc

2.

REST Docs 문서만들기

```
Writing AsciiDoc works best with soft wrap enabled. Do you want to enable it by default?
```

B
I

Toggle Title

<>

≡

Ab

🔗

🖼️

☰

☰

🖼️

↺

PDF

HTML

📄

1

2

3

4

5

6

7

8

[-]

```
[source, bash]

-----

$ echo '{
  "email": "bbb@bbb.com",
  "name": "bbb"
}' | http POST 'http://localhost:8080/api/members' \
  'Content-Type:application/json'

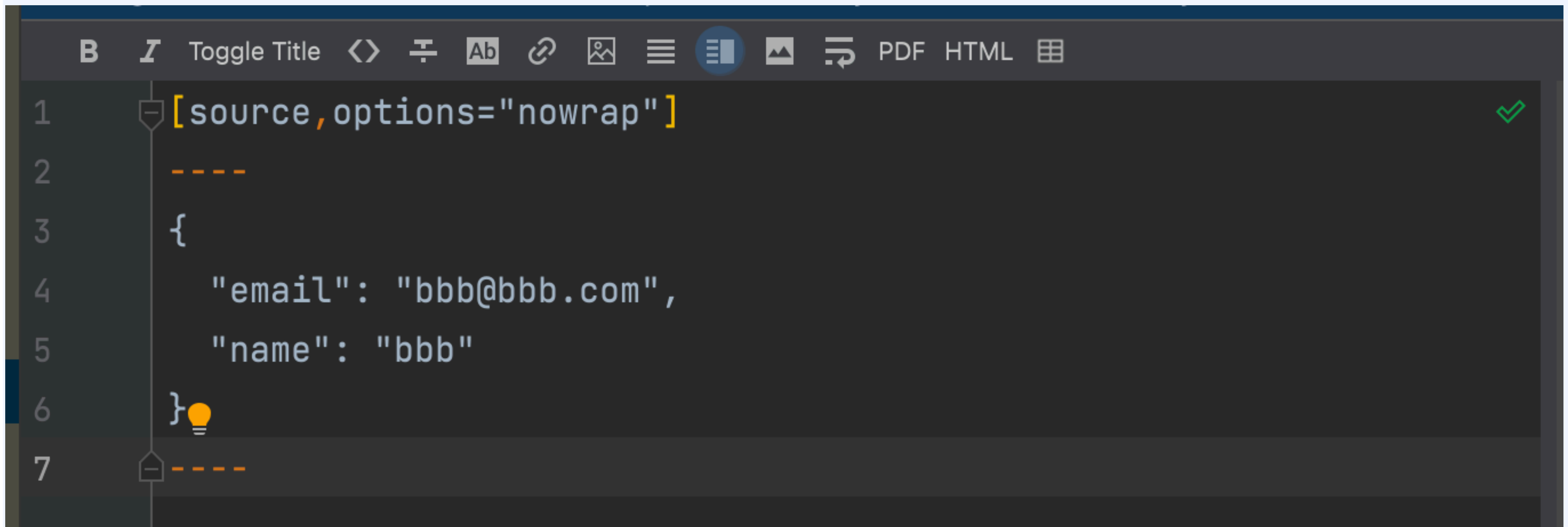
-----
```

[✓]

request-body.adoc

2.

REST Docs 문서만들기



```
1  [source,options="nowrap"]
2  -----
3  {
4      "email": "bbb@bbb.com",
5      "name": "bbb"
6  }
7  -----
```

The screenshot shows a code editor with a dark theme. The top toolbar includes icons for bold (B), italic (I), toggle title, code blocks, undo, redo, and various view modes (PDF, HTML, etc.). The code is written in a monospaced font. Line 1 shows a YAML directive [source,options="nowrap"] with a green checkmark icon to its right. Line 2 is a dashed separator line. Line 3 starts a YAML object with {. Line 4 and 5 contain two key-value pairs: "email": "bbb@bbb.com" and "name": "bbb". Line 6 ends the object with a closing brace }, followed by a yellow lightbulb icon indicating a suggestion or warning. Line 7 is another dashed separator line.

response-body.adoc

```
B I Toggle Title <> ÷ Ab 🔗 🖼️ ☰ ☷ 🖼️ ↺ PDF HTML 📄
1  [source,options="nowrap"] ✓
2  -----
3  {"id":1,"email":"yun@bbb.com","name":"new-yun"}
4  -----
```


REST Docs 커스텀하기

1 테스트 코드 개선

반복적인 코드 제거

```
@SpringBootTest
@AutoConfigureMockMvc
@ExtendWith(RestDocumentationExtension.class)
class MemberApiTest {
    5 usages
    @Autowired cheese10yun, Today • Step 1
    private MockMvc mockMvc;

    @BeforeEach
    void setUp(
        final WebApplicationContext context,
        final RestDocumentationContextProvider provider
    ) {
        this.mockMvc = MockMvcBuilders.webAppContextSetup(context)
            .apply(MockMvcRestDocumentation.documentationConfiguration(provider))
            .build();
    }

    @Test
    public void member_page_test() throws Exception {
        mockMvc.perform(
            get(urlTemplate: "/api/members")
        )
    }
}
```

```
@SpringBootTest
@AutoConfigureMockMvc
@ExtendWith(RestDocumentationExtension.class)
public class OrderApiController {

    1 usage
    @Autowired
    private MockMvc mockMvc;

    @BeforeEach
    void setUp(
        final WebApplicationContext context,
        final RestDocumentationContextProvider provider
    ) {
        this.mockMvc = MockMvcBuilders.webAppContextSetup(context)
            .apply(MockMvcRestDocumentation.documentationConfiguration(provider))
            .build();
    }

    @Test
    public void test() {
    }
}
```

반복적인 코드 제거

```
@Test
public void member_get() throws Exception {
    // 조회 API -> 대상의 데이터가 있어야 합니다.
    mockMvc.perform(
        get(urlTemplate: "/api/members/{id}", ...uriVars: 1L)
        .contentType(MediaType.APPLICATION_JSON)
    )
    .andDo(print())
    .andDo(MockMvcRestDocumentation.document(identifier: "{class-name}/{method-name}"))
    .andExpect(status().isOk());
}
```

```
@Test
public void member_page_test() throws Exception {
    mockMvc.perform(
        get(urlTemplate: "/api/members")
        .param(name: "size", ...values: "10")
        .param(name: "page", ...values: "0")
        .contentType(MediaType.APPLICATION_JSON)
    )
    .andDo(print())
    .andDo(MockMvcRestDocumentation.document(identifier: "{class-name}/{method-name}"))
    .andExpect(status().isOk());
}
```

Json 표현 개선

```
mockMvc.perform(
    put("/api/members/{id}", 1)
    .contentType(MediaType.APPLICATION_JSON)
    .content("")
        + "{\n"
        + "  \"name\": \"new-yun\"\n"
        + "}")
)
.andDo(print())
```

```
mockMvc.perform(
    post("/api/members")
    .contentType(MediaType.APPLICATION_JSON)
    .content("")
        + "{\n"
        + "  \"email\": \"bbb@bbb.com\", \n"
        + "  \"name\": \"bbb\"\n"
        + "}")
)
```

Json Formatting

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json
```

```
Content-Length: 539
```

```
{"content":[{"id":5,"email":"bbb@bbb.com","name":"bbb"}, {"id":4,"email":"jo@bbb.com","name":"jo"}]}
```

REST Docs 커스텀하기

2 문서 커스텀

최종 문서 형식

member 등록

Request fields

Field	Type	Required	Description	Length
name	String	true	name	10
email	String	true	email	123

HTTP request

```
POST /api/members HTTP/1.1
Content-Type: application/json
Content-Length: 50
Host: localhost:8080
```

```
{
  "name" : "yun",
  "email" : "writer@asd.com"
}
```

HTTP

HTTP response

```
HTTP/1.1 200 OK
```


최종 문서 형식

member 조회

HTTP request

```
GET /api/members/1 HTTP/1.1
Content-Type: application/json
Host: localhost:8080
```

Path parameters

Table 1. /api/members/{id}

Parameter	Description
id	Member id

HTTP response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 63

{
  "id" : 1,
  "nickname" : "yun",
  "email" : "yun@asd.com"
}
```

Response fields

Path	Type	Required	Description
id	Number	true	Member id
email	String	true	Email
nickname	String	true	Name

REST Docs 커스텀하기

2 문서 커스텀

최종 문서 형식

member 등록

Request fields

Field	Type	Required	Description	Length
name	String	true	name	10
email	String	true	email	123

HTTP request

```
POST /api/members HTTP/1.1
Content-Type: application/json
Content-Length: 50
Host: localhost:8080
```

HTTP

```
{
  "name" : "yun",
  "email" : "writer@asd.com"
}
```

HTTP response

```
HTTP/1.1 200 OK
```

최종 문서 형식



- <https://gist.github.com/cheese10yun/c30acbb71cb52533ae7b967b5e15526c>

Error Response 문서화

Error Response

HTTP response

```
HTTP/1.1 400 Bad Request
Content-Type: application/json
Content-Length: 326

{
  "message" : " Invalid Input Value",
  "status" : 400,
  "errors" : [ {
    "field" : "email",
    "value" : "asd",
    "reason" : "must be a well-formed email address"
  }, {
    "field" : "name",
    "value" : "",
    "reason" : "must not be empty"
  } ],
  "code" : "C001",
  "timestamp" : "2021-09-11T22:07:04.721359"
}
```

Response fields

Path	Type	Required	Description
message	String	true	에러 메시지
status	Number	true	Http Status Code
code	String	true	Error Code
timestamp	String	true	에러 발생 시간
errors	Array	true	Error 값 배열 값
errors[0].field	String	true	문제 있는 필드
errors[0].value	String	true	문제가 있는 값
errors[0].reason	String	true	문제가 있는 이유

END

Github: <https://github.com/cheese10yun>

Email: cheese10yun@gmail.com

Blog: <https://cheese10yun.github.io/>