# **Spring REST Docs 설정 가이드: 테스트 기반 API 문서 자동화**

Spring REST Docs는 RESTful API 문서를 효과적으로 작성하고 관리하기 위한 강력한 도구입니다. 이 가이드는 Spring REST Docs의 핵심 철학부터 시작하여 Maven 및 Gradle 프로젝트에서의 상세한 설정 방법, 테스트 코드와의 통합, 문서 조립 및 고급 활용법, 그리고 일반적인 문제 해결 전략까지 포괄적으로 다룹니다. 이 문서를 통해 개발자는 API의 실제 동작과 일치하는 정확하고 유지보수 용이한 문서를 생성할 수 있게 될 것입니다.

## **I. Spring REST Docs 이해하기**

Spring REST Docs를 효과적으로 활용하기 위해서는 그 기본 철학과 핵심 구성 요소, 그리고 주요 이점에 대한 이해가 선행되어야 합니다.

### **A. 철학: 테스트 기반 API 문서화**

Spring REST Docs의 가장 핵심적인 철학은 \*\*테스트 기반 문서화(Test-Driven Documentation)\*\*입니다.1 이는 API의 동작을 검증하는 테스트 코드를 실행함으로써 문서 조각(snippet)들이 자동으로 생성되는 접근 방식을 의미합니다.

이러한 철학은 문서가 API의 실제 동작과 항상 일치하도록 보장하기 때문에 매우 중요합니다.1 만약 테스트가 실패하면 문서는 생성되지 않거나 해당 실패를 반영하게 되어, 문서의 정확성이 자연스럽게 유지됩니다. 이는 어노테이션 기반 도구들에서 발생할 수 있는 실제 구현과 문서 간의 불일치 문제를 해결하는 데 도움을 줍니다. Spring REST Docs는 컨트롤러 메서드를 직접 분석하는 대신, HTTP 요청과 응답의 전체 생명주기(필터, 인터셉터, 메시지 변환기 등 포함)를 포괄하는 테스트를 통해 문서를 생성함으로써 이러한 정확성을 달성합니다.3

개발자들은 종종 문서 작성을 별도의 지루한 작업으로 여기는 경향이 있습니다.4 Spring REST Docs는 문서 생성 과정을 개발자들이 이미 수행하고 있는 테스트 단계에 통합함으로써 1, 문서화 작업의 부담을 크게 줄여줍니다. 테스트로부터 문서가 파생되기 때문에, API를 사용하는 개발자들은 수동으로 작성되거나 순수하게 어노테이션만으로 생성된 문서보다 높은 신뢰도를 가질 수 있습니다.1 이는 API 소비자들의 신뢰를 구축하고 통합 과정에서의 문제를 줄이는 데 기여합니다. 결과적으로 팀은 최신 상태의 문서를 유지할 가능성이 높아지며, 이는 API 소비자의 개발자 경험을 향상시키고 모호성을 감소시킵니다.

### **B. 핵심 아키텍처 구성 요소**

Spring REST Docs는 수동으로 작성된 설명 부분과 테스트를 통해 자동으로 생성된 기술적 세부 사항을 결합하여 문서를 구성합니다.

* **수동 작성 문서 (Asciidoctor/Markdown)**: 문서의 서술적인 부분, 즉 API의 목적, 사용 사례, 비즈니스 로직 등에 대한 설명은 일반적으로 Asciidoctor(기본값) 또는 Markdown을 사용하여 수동으로 작성됩니다.1
  + **Asciidoctor**: 기술 문서 작성에 매우 적합한 강력한 일반 텍스트 마크업 언어로, Markdown보다 더 많은 기능을 제공합니다.8 Spring 프레임워크 자체의 문서도 Asciidoctor를 사용하여 작성됩니다.6
  + **Markdown**: 또 다른 선택지이지만, 표(table) 미지원, 외부 도구(예: Slate) 없이는 파일 포함(include) 기능 부재 등의 한계가 있습니다.9
* **자동 생성 스니펫 (Auto-Generated Snippets)**: API 테스트 실행 시 생성되는 작은 문서 조각들입니다. 예를 들어, cURL 요청 예시, HTTP 응답 본문, 요청/응답 필드 설명 등이 여기에 해당합니다.1 이러한 스니펫은 Spring MVC Test (MockMvc), Spring WebFlux의 WebTestClient, 또는 REST Assured와 같은 테스트 프레임워크를 사용하여 생성됩니다.1
* **결합**: 최종 문서는 이렇게 수동으로 작성된 부분과 자동으로 생성된 스니펫을 결합하여 완성됩니다.1

순수하게 자동으로 생성된 문서(예: 기본적인 Swagger UI 출력)는 종종 맥락, 이야기 흐름, 상세 설명이 부족할 수 있습니다.1 반대로 순수하게 수동으로 작성된 문서는 오류가 발생하기 쉽고 최신 상태를 유지하기 어렵습니다. Spring REST Docs는 풍부한 맥락적 설명(사용 사례, 비즈니스 로직 등)을 수동으로 작성할 수 있게 하면서도, 기술적 세부 사항(요청/응답 구조, 헤더, 경로 등)은 테스트를 통해 자동으로 정확하게 생성되도록 합니다. 이러한 하이브리드 접근 방식은 API 사용자들의 다양한 정보 요구를 충족시키는 동시에 포괄적이고 신뢰할 수 있는 문서를 생산하게 합니다.

### **C. 주요 이점 및 선택 시기**

Spring REST Docs는 여러 가지 장점을 제공하며, 특정 상황에서 특히 유용합니다.

* **정확성**: 테스트된 API 동작을 문서가 반영하므로 매우 정확합니다.1
* **가독성 및 구조**: 단순한 API 레퍼런스가 아닌, 잘 구조화된 가이드를 생성합니다.1 GitHub의 API 문서와 유사한 사용자 가이드를 지향합니다.2
* **유연성**: 어노테이션 기반 도구보다 제약이 적어 풍부한 서술적 콘텐츠 작성이 가능합니다.1
* **형식 지원**: JSON 및 XML 페이로드를 모두 지원합니다.6
* **통합**: 널리 사용되는 테스트 도구 및 빌드 시스템과 잘 통합됩니다.
* **선택 시기**: API 정확성이 매우 중요하고, 단순한 엔드포인트 목록 이상의 상세한 설명이 필요하며, 테스트 주도 개발(TDD) 문화가 있거나 도입하고자 하는 프로젝트에 이상적입니다. 특히 컨텍스트와 링크가 중요한 하이퍼미디어 API에 강력합니다.14

문서 생성을 위해 테스트를 작성하는 과정은 개발자가 API 계약(경로, 매개변수, 요청/응답 본문, 상태 코드 등)에 대해 비판적으로 생각하도록 만듭니다. 만약 API의 특정 부분을 명확하게 문서화하기 위한 테스트를 작성하기 어렵거나, 문서화 과정에서 불일치나 어색함이 드러난다면, 이는 종종 API 자체의 설계 문제를 시사합니다. 필드, 매개변수 등을 명시적으로 문서화해야 한다는 요구사항은 어떤 것도 가정에 맡기지 않도록 합니다.15 이는 단순히 문서를 생성하는 것을 넘어, Spring REST Docs가 더 나은 API 설계를 장려하고 API 계약에 대한 검증 계층 역할을 하여 더 높은 품질과 일관성을 가진 API로 이어지게 함을 의미합니다.

### **D. 필수 선행 조건**

Spring REST Docs를 사용하기 전에 몇 가지 준비 사항이 필요합니다.

* **Java**: Spring REST Docs 자체를 빌드하려면 Java 17 이상이 필요하며 2, 이를 사용하는 프로젝트도 가급적 최신 Java 버전을 사용하는 것이 좋습니다.
* **Spring Boot**: REST Docs 코어의 엄격한 의존성은 아니지만, 일반적으로 Spring Boot와 함께 사용되며, Spring Boot는 자동 설정을 통해 설정을 간소화합니다.1
* **빌드 도구**: Maven 또는 Gradle이 필요합니다.12
* **테스트 프레임워크**: JUnit 4 또는 JUnit 5가 필요하며 (JUnit 5 권장, 예제는 JUnit 5 중심).6
* **기본 이해**: Spring MVC 또는 WebFlux, REST 원칙, 선택한 빌드 도구에 대한 기본적인 이해가 필요합니다.

## **II. Spring REST Docs를 염두에 둔 초기 프로젝트 설정**

일반적인 시작점인 Spring Boot 프로젝트를 생성하는 과정을 안내합니다.

### **A. Spring Initializr (start.spring.io)로 부트스트래핑**

Spring Initializr는 필요한 의존성을 포함한 Spring Boot 프로젝트 생성을 간소화합니다.12

* **단계**:
  1. https://start.spring.io로 이동합니다.
  2. 빌드 도구(Maven 또는 Gradle)와 언어(Java/Kotlin)를 선택합니다. 이 가이드는 Java를 중심으로 설명합니다.
  3. Spring Boot 버전을 선택합니다.
  4. RESTful 서비스를 위해 "Spring Web" 의존성을 추가합니다.
  5. (선택 사항이지만 권장) 보일러플레이트 코드 감소를 위해 "Lombok"을 추가합니다.
  6. 프로젝트를 생성하고 다운로드합니다.

이는 일관된 시작점을 제공하고 기본적인 웹 기능이 포함되도록 보장합니다.

### **B. RESTful 서비스를 위한 핵심 의존성**

Spring Initializr를 사용하지 않거나 기존 프로젝트의 경우, spring-boot-starter-web (Spring MVC용) 또는 spring-boot-starter-webflux (WebFlux용)가 있는지 확인해야 합니다. spring-boot-starter-test는 문서 생성 테스트를 작성하는 데 필수적이며, 일반적으로 JUnit, Spring Test, Mockito 등을 포함합니다.12

Spring REST Docs는 전적으로 테스트에 의존합니다.1 spring-boot-starter-web 및 spring-boot-starter-test 의존성은 테스트 가능한 REST 컨트롤러를 만들고 MockMvc 또는 WebTestClient를 사용하여 이에 대한 통합 테스트를 작성하는 데 필요한 인프라를 제공합니다. 테스트 가능한 애플리케이션 없이는 Spring REST Docs가 작동할 수 없습니다. 따라서 이러한 핵심 의존성을 초기에 강조하는 것은 Spring REST Docs를 효과적으로 사용하기 위해 프로젝트가 테스트 가능하도록 구조화되어야 함을 나타냅니다. 이는 나중에 REST Docs 의존성을 추가하는 것뿐만 아니라 처음부터 테스트 가능한 시스템을 구축하는 것에 관한 문제입니다.

## **III. Spring REST Docs 설정: Maven 프로젝트**

Maven 사용자를 위한 상세 설정 방법을 안내합니다.

### **A. 필수 pom.xml 의존성**

Maven 프로젝트에서 Spring REST Docs를 사용하기 위해서는 pom.xml 파일에 다음과 같은 주요 의존성을 추가해야 합니다.

* **spring-restdocs-mockmvc**: Spring MVC Test의 MockMvc와 통합하기 위한 핵심 의존성입니다.12 test 스코프로 지정하는 것이 중요합니다.  
  XML  
  <dependency>  
   <groupId>org.springframework.restdocs</groupId>  
   <artifactId>spring-restdocs-mockmvc</artifactId>  
   <scope>test</scope>  
  </dependency>
* **대안**:
  + spring-restdocs-webtestclient: WebTestClient를 사용하는 리액티브 애플리케이션용.2
  + spring-restdocs-restassured: REST Assured를 테스트에 사용하는 프로젝트용.2
* **spring-boot-starter-test**: 이미 포함되어 있지 않다면, 일반적인 테스트 기능을 위해 추가해야 합니다.12

다음은 Spring REST Docs를 위한 주요 Maven 의존성을 요약한 표입니다.

| **Group ID** | **Artifact ID** | **Scope** | **설명** |
| --- | --- | --- | --- |
| org.springframework.restdocs | spring-restdocs-mockmvc | test | MockMvc 통합을 위한 핵심 의존성. 12 |
| org.springframework.restdocs | spring-restdocs-webtestclient | test | WebTestClient 통합용 (리액티브 애플리케이션). 4 |
| org.springframework.restdocs | spring-restdocs-restassured | test | REST Assured 통합용. 2 |
| org.springframework.boot | spring-boot-starter-test | test | JUnit, Mockito, Spring Test 등 핵심 테스트 유틸리티 제공. 12 |

이 표는 사용자가 자신의 특정 테스트 설정에 맞는 올바른 의존성을 선택하여 오류를 줄이고 시간을 절약하는 데 도움을 줄 수 있습니다.

### **B. Asciidoctor Maven 플러그인 마스터하기**

Asciidoctor Maven 플러그인은 Asciidoctor (.adoc) 파일을 처리하고 생성된 스니펫을 통합하여 최종 HTML 문서를 생성하는 역할을 합니다.6

* **플러그인 선언**:  
  XML  
  <plugin>  
   <groupId>org.asciidoctor</groupId>  
   <artifactId>asciidoctor-maven-plugin</artifactId>  
   <version>2.2.4</version> <executions>  
   <execution>  
   <id>generate-docs</id>  
   <phase>prepare-package</phase>  
   <goals>  
   <goal>process-asciidoc</goal>  
   </goals>  
   <configuration>  
   <backend>html5</backend>  
   <doctype>book</doctype>  
   <attributes>  
   <snippets>${project.build.directory}/generated-snippets</snippets>  
   </attributes>  
   </configuration>  
   </execution>  
   </executions>  
   <dependencies>  
   <dependency>  
   <groupId>org.springframework.restdocs</groupId>  
   <artifactId>spring-restdocs-asciidoctor</artifactId>  
   <version>${spring-restdocs.version}</version> </dependency>  
   </dependencies>  
  </plugin>  
  이 설정은 매우 중요합니다. prepare-package 단계는 애플리케이션 패키징 전에 문서가 생성되도록 보장하며 7, 이는 문서를 JAR/WAR 내에 번들링하려는 경우 필수적입니다. <snippets> 속성은 Asciidoctor에게 생성된 스니펫을 어디서 찾을지 알려줍니다.6 ${project.build.directory}/generated-snippets 경로는 일반적인 Maven 출력 디렉토리입니다.
* **spring-restdocs-asciidoctor 통합**: Asciidoctor 플러그인 *내부*의 이 의존성은 Spring REST Docs와 Asciidoctor를 연결하는 매크로 및 설정을 제공하여 operation:: 매크로 및 자동 스니펫 경로 확인과 같은 기능을 활성화합니다.7

스니펫은 test 단계에서 생성됩니다. Asciidoctor 처리는 이러한 스니펫이 존재해야 합니다. 최종 문서는 애플리케이션과 함께 패키징되어야 할 수도 있습니다. asciidoctor-maven-plugin에 대한 Maven prepare-package 단계 12는 테스트가 실행되어 스니펫을 생성한 후, Asciidoctor가 .adoc 파일과 스니펫을 처리하도록 보장하며, 이는 실제 패키징 단계 *전에* 이루어집니다. 이 빌드 생명주기 의존성을 이해하는 것은 문서가 올바르게 생성되지 않거나 포함되지 않는 문제를 해결하는 데 핵심입니다. 단순히 플러그인을 추가하는 것뿐만 아니라 실행 순서에 관한 문제입니다.

### **C. 예시 pom.xml 설정**

다음은 Spring REST Docs 및 Asciidoctor에 필요한 모든 구성을 포함하는 <dependencies> 및 <build><plugins> 섹션을 보여주는 완전하고 간결한 pom.xml 스니펫입니다. spring-restdocs.version 및 asciidoctor-maven-plugin.version에 대한 속성을 사용하여 버전을 일관되게 관리하는 것이 중요합니다.

XML

<properties>  
 <java.version>17</java.version>  
 <spring-boot.version>3.1.5</spring-boot.version>  
 <spring-restdocs.version>3.0.1</spring-restdocs.version>  
 <asciidoctor-maven-plugin.version>2.2.4</asciidoctor-maven-plugin.version>  
</properties>  
  
<dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.restdocs</groupId>  
 <artifactId>spring-restdocs-mockmvc</artifactId>  
 <scope>test</scope>  
 </dependency>  
</dependencies>  
  
<build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 <plugin>  
 <groupId>org.asciidoctor</groupId>  
 <artifactId>asciidoctor-maven-plugin</artifactId>  
 <version>${asciidoctor-maven-plugin.version}</version>  
 <executions>  
 <execution>  
 <id>generate-docs</id>  
 <phase>prepare-package</phase>  
 <goals>  
 <goal>process-asciidoc</goal>  
 </goals>  
 <configuration>  
 <backend>html5</backend>  
 <doctype>book</doctype>  
 <attributes>  
 <snippets>${project.build.directory}/generated-snippets</snippets>  
 </attributes>  
 </configuration>  
 </execution>  
 </executions>  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.restdocs</groupId>  
 <artifactId>spring-restdocs-asciidoctor</artifactId>  
 <version>${spring-restdocs.version}</version>  
 </dependency>  
 </dependencies>  
 </plugin>  
 </plugins>  
</build>

## **IV. Spring REST Docs 설정: Gradle 프로젝트**

Gradle 사용자를 위한 상세 설정 방법을 안내합니다.

### **A. 필수 build.gradle 플러그인 및 의존성**

Gradle 프로젝트에서 Spring REST Docs를 사용하기 위해서는 build.gradle 파일에 다음과 같은 주요 플러그인과 의존성을 설정해야 합니다.

* **org.asciidoctor.jvm.convert 플러그인 적용**: Asciidoctor를 위한 최신 Gradle 플러그인입니다.12 최신 호환 버전을 사용하는 것이 중요합니다. 이전 버전인 org.asciidoctor.convert는 최신 Gradle 버전과 호환성 문제가 있을 수 있습니다.19  
  Gradle  
  plugins {  
   id 'java'  
   id 'org.springframework.boot' version '3.1.5' // Spring Boot 버전 명시  
   id 'io.spring.dependency-management' version '1.1.3' // 버전 명시  
   id 'org.asciidoctor.jvm.convert' version '3.3.2' // 또는 최신 호환 버전  
  }
* **spring-restdocs-mockmvc 의존성 선언**:  
  Gradle  
  dependencies {  
   implementation 'org.springframework.boot:spring-boot-starter-web'  
   testImplementation 'org.springframework.boot:spring-boot-starter-test'  
   testImplementation 'org.springframework.restdocs:spring-restdocs-mockmvc' // 버전은 Spring Boot BOM으로 관리되거나 명시적으로 지정  
  }  
  이는 MockMvc 통합을 통해 스니펫 생성을 가능하게 합니다.12
* **spring-restdocs-asciidoctor를 asciidoctorExt에 추가**:  
  Gradle  
  configurations {  
   asciidoctorExt // [18]  
  }  
    
  ext { // 버전 관리를 위한 ext 블록 사용 권장  
   springRestDocsVersion = '3.0.1'  
  }  
    
  dependencies {  
   //... 다른 의존성들  
   asciidoctorExt "org.springframework.restdocs:spring-restdocs-asciidoctor:${springRestDocsVersion}" // [18]  
  }  
  이 설정은 Spring REST Docs Asciidoctor 확장 기능(매크로, 스니펫 경로 확인 등)을 Asciidoctor 태스크에서 사용할 수 있게 합니다.18

다음은 Spring REST Docs를 위한 주요 Gradle 플러그인 및 의존성을 요약한 표입니다.

| **유형** | **이름/식별자** | **설정 블록** | **설명** |
| --- | --- | --- | --- |
| 플러그인 | org.asciidoctor.jvm.convert | plugins {} | Asciidoctor 처리를 위한 핵심 Gradle 플러그인. 12 |
| 의존성 | spring-restdocs-mockmvc | dependencies {} | testImplementation으로 MockMvc 통합. 12 |
| 의존성 | spring-restdocs-webtestclient | dependencies {} | testImplementation으로 WebTestClient 통합. |
| 의존성 | spring-restdocs-restassured | dependencies {} | testImplementation으로 REST Assured 통합. |
| 의존성 | spring-restdocs-asciidoctor | dependencies {} | asciidoctorExt 설정을 통한 Asciidoctor 확장 기능. 18 |

이 표는 Gradle 설정이 Maven과 어떻게 다른지, 특히 Gradle에 특정한 asciidoctorExt 설정과 같은 필요한 플러그인 및 의존성 선언 방법을 명확히 요약하여 올바른 build.gradle 설정에 도움을 줍니다.

### **B. 스니펫 디렉토리 및 테스트 태스크의 전략적 설정**

생성된 스니펫의 위치를 일관되게 관리하고 테스트 태스크와의 연동을 올바르게 설정하는 것이 중요합니다.

* **snippetsDir 정의 및 사용**:  
  Gradle  
  ext {  
   snippetsDir = file("build/generated-snippets")  
  }  
  build/generated-snippets는 Gradle의 일반적인 스니펫 출력 디렉토리 경로입니다.12
* **테스트 태스크 출력을 snippetsDir에 연결**:  
  Gradle  
  tasks.named('test') {  
   outputs.dir snippetsDir  
   useJUnitPlatform()  
  }  
  이 설정은 test 태스크가 snippetsDir에 출력을 생성함을 Gradle에 알립니다. 이는 Gradle의 증분 빌드(incremental build) 지원 및 test 태스크의 출력에 의존하는 다른 태스크들에 중요합니다.12

### **C. asciidoctor 태스크 미세 조정**

asciidoctor 태스크는 생성된 스니펫을 사용하여 최종 문서를 빌드하는 핵심 역할을 합니다.

* **핵심 설정**:  
  Gradle  
  tasks.named('asciidoctor') {  
   configurations 'asciidoctorExt' // [18]  
   inputs.dir snippetsDir // [18]  
   sourceDir file('src/main/asciidoc') // [12] 또는 'src/docs/asciidoc' [20]  
   attributes([  
   'snippets': snippetsDir // [12, 18])  
   dependsOn tasks.named('test') // [12, 17, 18]  
  }  
  + configurations 'asciidoctorExt': Spring REST Docs Asciidoctor 확장을 적용합니다.
  + inputs.dir snippetsDir: 생성된 스니펫을 Asciidoctor 태스크의 입력으로 지정합니다. 이는 증분 빌드와 정확성에 매우 중요합니다.
  + sourceDir: 주 .adoc 파일이 위치한 디렉토리를 지정합니다. 일반적인 경로는 src/main/asciidoc 또는 src/docs/asciidoc입니다.
  + attributes(['snippets': snippetsDir]): snippetsDir 경로를 Asciidoctor 속성 {snippets}로 만들어 include:: 매크로에서 사용할 수 있게 합니다.
  + dependsOn tasks.named('test'): **매우 중요합니다.** 테스트가 실행되어 스니펫이 생성된 *후에* Asciidoctor가 이를 처리하도록 보장합니다.12

Gradle 태스크는 실행 순서를 정의하기 위해 의존성을 가집니다. asciidoctor 태스크는 test 태스크에 의해 생성된 스니펫을 소비합니다. dependsOn test가 없으면 Gradle은 test 태스크 이전 또는 동시에 asciidoctor를 실행하려고 시도할 수 있으며, 이는 스니펫 누락 및 빌드 실패로 이어질 수 있습니다. 이 의존성은 Gradle 설정에서 흔히 발생하는 함정이며, 이를 명시적으로 강조하면 이 설정에 익숙하지 않은 사용자가 자주 겪는 오류를 예방할 수 있습니다. 이는 스니펫 생성과 소비의 순차적 특성을 강조합니다.

### **D. 예시 build.gradle 설정**

다음은 플러그인, 의존성, ext 블록, 그리고 test 및 asciidoctor 태스크 설정을 포함하는 완전하고 간결한 build.gradle 스니펫입니다. Spring Boot 모범 사례에 따라 의존성 버전 관리를 위해 id 'io.spring.dependency-management'를 사용하는 것을 언급합니다.

Gradle

plugins {  
 id 'java'  
 id 'org.springframework.boot' version '3.1.5'  
 id 'io.spring.dependency-management' version '1.1.3'  
 id 'org.asciidoctor.jvm.convert' version '3.3.2'  
}  
  
group = 'com.example'  
version = '0.0.1-SNAPSHOT'  
  
java {  
 sourceCompatibility = '17'  
}  
  
configurations {  
 compileOnly {  
 extendsFrom annotationProcessor  
 }  
 asciidoctorExt // [18]  
}  
  
repositories {  
 mavenCentral()  
}  
  
ext {  
 snippetsDir = file("build/generated-snippets")  
 springRestDocsVersion = '3.0.1' // REST Docs 버전 관리  
}  
  
dependencies {  
 implementation 'org.springframework.boot:spring-boot-starter-web'  
 compileOnly 'org.projectlombok:lombok'  
 annotationProcessor 'org.projectlombok:lombok'  
 testImplementation 'org.springframework.boot:spring-boot-starter-test'  
 testImplementation "org.springframework.restdocs:spring-restdocs-mockmvc:${springRestDocsVersion}"  
 asciidoctorExt "org.springframework.restdocs:spring-restdocs-asciidoctor:${springRestDocsVersion}"  
}  
  
tasks.named('test') {  
 outputs.dir snippetsDir  
 useJUnitPlatform()  
}  
  
tasks.named('asciidoctor') {  
 configurations 'asciidoctorExt'  
 inputs.dir snippetsDir  
 sourceDir file('src/main/asciidoc') // 또는 src/docs/asciidoc  
 attributes([  
 'snippets': snippetsDir  
 ])  
 dependsOn tasks.named('test')  
}  
  
// 생성된 문서를 JAR에 포함시키기 (선택 사항)  
bootJar {  
 dependsOn tasks.named('asciidoctor') // asciidoctor 작업이 먼저 실행되도록 보장  
 from("${tasks.named('asciidoctor').get().outputDir}") { // AsciidoctorTask의 outputDir 사용  
 into 'static/docs'  
 }  
}

## **V. 테스트에 문서화 통합하기**

테스트 코드를 수정하여 문서 스니펫을 생성하는 방법을 상세히 설명합니다.

### **A. 기본 어노테이션 및 JUnit 5 설정**

JUnit 5 기반의 테스트에서 Spring REST Docs를 통합하기 위한 핵심 어노테이션들은 다음과 같습니다.

* **@ExtendWith({RestDocumentationExtension.class, SpringExtension.class}) (JUnit 5)**:
  + SpringExtension.class: JUnit 5 프로그래밍 모델을 Spring TestContext Framework와 통합합니다.
  + RestDocumentationExtension.class: RestDocumentationContextProvider를 설정하고 해제하는 데 필수적이며, 이는 REST Docs에 컨텍스트를 제공합니다.4
* **@AutoConfigureRestDocs (Spring Boot 특정)**:
  + Spring Boot 환경에서 Spring REST Docs 설정을 간소화합니다.1 RestDocumentationContextProvider를 자동으로 설정하고 기본 설정을 적용합니다.
  + 사용 예: @AutoConfigureRestDocs(outputDir = "build/generated-snippets") (Gradle) 또는 target/generated-snippets (Maven). outputDir은 빌드 스크립트에 설정된 경로와 일치해야 합니다.

다음은 Spring REST Docs 테스트 통합(JUnit 5)에 필요한 주요 어노테이션을 요약한 표입니다.

| **어노테이션** | **범위** | **설명** |
| --- | --- | --- |
| @ExtendWith(SpringExtension.class) | 클래스 | JUnit 5를 Spring TestContext Framework와 통합합니다. 4 |
| @ExtendWith(RestDocumentationExtension.class) | 클래스 | REST Docs 설정 및 해제를 위한 RestDocumentationContextProvider를 제공합니다. 4 |
| @SpringBootTest | 클래스 | Spring Boot 통합 테스트를 위해 애플리케이션 컨텍스트를 로드합니다. |
| @WebMvcTest(YourController.class) | 클래스 | 웹 계층(Spring MVC)의 집중 테스트용. @AutoConfigureMockMvc와 함께 사용합니다. 12 |
| @AutoConfigureMockMvc | 클래스 | MockMvc 인스턴스를 자동 설정합니다. |
| @AutoConfigureRestDocs | 클래스 | Spring Boot에서 Spring REST Docs 설정을 간소화하고 출력 디렉토리를 설정합니다. 1 |

이 표는 개발자가 테스트 클래스에 적용해야 하는 어노테이션을 명확히 설명하여 잘못된 구성을 방지하고 테스트 환경이 Spring REST Docs에 맞게 올바르게 준비되도록 돕습니다.

### **B. MockMvc로 문서화하기: 실용적인 예제**

Spring MVC 애플리케이션 테스트에 널리 사용되는 MockMvc를 활용한 문서화 방법을 설명합니다.

* **MockMvc 설정**:
  + MockMvc 주입: @Autowired private MockMvc mockMvc;
  + @AutoConfigureRestDocs를 사용하지 않는 경우, @BeforeEach 메서드에서 RestDocumentationContextProvider의 documentationConfiguration을 사용하여 MockMvc를 설정해야 합니다.  
    Java  
    @BeforeEach  
    void setUp(WebApplicationContext webApplicationContext, RestDocumentationContextProvider restDocumentation) {  
     this.mockMvc = MockMvcBuilders.webAppContextSetup(webApplicationContext)  
     .apply(documentationConfiguration(restDocumentation)) // REST Docs 기능 적용  
     .build();  
    }  
    이 수동 설정은 MockMvc에 REST Docs 기능을 적용합니다.12
* **document("operation-identifier",...) 호출**:
  + 스니펫 생성의 핵심입니다. MockMvc.perform(...).andDo(...)에서 ResultHandler로 사용됩니다.
  + "operation-identifier"는 문서화되는 API 작업의 고유 이름(예: "get-user", "create-item")이며, 생성된 스니펫의 하위 디렉토리 이름이 됩니다.12  
    Java  
    this.mockMvc.perform(get("/api/items/{id}", 1L).accept(MediaType.APPLICATION\_JSON))  
     .andExpect(status().isOk())  
     .andDo(document("items/get-by-id", // operation-identifier  
     pathParameters(parameterWithName("id").description("조회할 아이템의 ID")),  
     responseFields(  
     fieldWithPath("id").description("아이템의 고유 식별자"),  
     fieldWithPath("name").description("아이템의 이름")  
     )  
     ));  
    이 예제는 경로 매개변수와 응답 필드를 문서화하는 방법을 보여줍니다.12
* **전처리기(Preprocessors)**: document()에 추가하여 스니펫 생성 전에 요청/응답을 수정할 수 있습니다 (예: preprocessResponse(prettyPrint())).6
* **기본 생성 스니펫 개요**: document()가 호출되면 curl-request.adoc, http-request.adoc, http-response.adoc, request-body.adoc, response-body.adoc 등 여러 기본 스니펫이 생성됩니다.7

### **C. 간략 가이드: WebTestClient로 문서화하기**

리액티브 Spring WebFlux 애플리케이션의 문서화 방법을 간략히 소개합니다.

* **사용 사례**: 리액티브 Spring WebFlux 애플리케이션.
* **WebTestClient 설정**:
  + spring-restdocs-webtestclient 의존성이 필요합니다.4
  + @BeforeEach에서 ApplicationContext에 바인딩하고 documentationConfiguration 필터를 적용하여 WebTestClient를 설정합니다.  
    Java  
    @BeforeEach  
    void setUp(ApplicationContext applicationContext, RestDocumentationContextProvider restDocumentation) {  
     this.webTestClient = WebTestClient.bindToApplicationContext(applicationContext)  
     .configureClient()  
     .filter(documentationConfiguration(restDocumentation)) // REST Docs 필터 통합  
     .build();  
    }  
    이 설정은 WebTestClient에 특화되어 있으며 REST Docs 필터가 통합되는 방식을 보여줍니다.4
* **문서화 적용**: consumeWith(document("operation-identifier",...)) 사용  
  Java  
  this.webTestClient.get().uri("/api/reactive-items/{id}", 1L)  
   .accept(MediaType.APPLICATION\_JSON)  
   .exchange()  
   .expectStatus().isOk()  
   .expectBody()  
   .consumeWith(document("reactive-items/get-by-id",  
   pathParameters(parameterWithName("id").description("리액티브 아이템의 ID"))  
   //... 다른 디스크립터들  
   ));  
  이는 MockMvc의 andDo(document(...))에 해당하는 WebTestClient 방식입니다.4

Spring REST Docs의 핵심 로직은 HTTP 교환 세부 정보를 캡처하는 것입니다. 이를 위해 특정 통합 모듈(spring-restdocs-mockmvc, spring-restdocs-webtestclient, spring-restdocs-restassured)을 제공합니다.2 문서화 컨텍스트를 구성한 다음 이를 테스트 클라이언트(MockMvc, WebTestClient)에 "적용"하는 일반적인 패턴은 이러한 통합 전반에 걸쳐 유사합니다.4 document() 호출은 구문상 약간 다를 수 있지만(예: andDo() 대 consumeWith()), 동일한 목적을 수행합니다. 따라서 설정 세부 정보가 약간 다르더라도 Spring REST Docs가 작동하는 기본 원칙은 일관되게 유지됩니다. 이를 통해 개발자는 REST Docs를 완전히 다시 배울 필요 없이 테스트 프레임워크를 전환할 수 있습니다.

## **VI. API 문서 조립하기**

생성된 스니펫을 일관성 있는 문서로 만드는 방법을 설명합니다.

### **A. 생성된 스니펫의 구조**

document() 메서드가 호출되면 테스트에서 정의한 operation-identifier를 이름으로 하는 하위 디렉토리에 다양한 스니펫 파일이 생성됩니다. 일반적인 스니펫 파일과 그 내용은 다음과 같습니다.7

* **기본 스니펫 위치**: Maven의 경우 target/generated-snippets/{operation-identifier}/, Gradle의 경우 build/generated-snippets/{operation-identifier}/.

다음은 일반적인 기본 스니펫 개요입니다.

| **스니펫 파일명** | **설명** | **생성 주체** |
| --- | --- | --- |
| curl-request.adoc | 요청에 대한 예제 cURL 명령어. | document() 호출 시 기본 생성 |
| http-request.adoc | 원시 HTTP 요청 세부 정보. | document() 호출 시 기본 생성 |
| http-response.adoc | 원시 HTTP 응답 세부 정보. | document() 호출 시 기본 생성 |
| httpie-request.adoc | 요청에 대한 예제 HTTPie 명령어. | document() 호출 시 기본 생성 |
| request-body.adoc | HTTP 요청 본문 (있는 경우). | 요청 본문이 있을 경우 기본 생성 |
| response-body.adoc | HTTP 응답 본문 (있는 경우). | 응답 본문이 있을 경우 기본 생성 |
| request-fields.adoc | 문서화된 요청 필드를 설명하는 표. | requestFields() 사용 시 |
| response-fields.adoc | 문서화된 응답 필드를 설명하는 표. | responseFields() 사용 시 |
| path-parameters.adoc | 문서화된 경로 매개변수를 설명하는 표. | pathParameters() 사용 시 |
| request-parameters.adoc | 문서화된 쿼리/폼 매개변수를 설명하는 표. | requestParameters(), formParameters(), queryParameters() 사용 시 16 |
| links.adoc | 하이퍼미디어 링크를 설명하는 표. | links() 사용 시 |

이 표는 각 일반 스니펫 유형에 대한 빠른 조회 기능을 제공하여 사용자가 어떤 파일이 생성되고 무엇을 포함하는지 이해하는 데 도움을 줍니다. 어떤 document() 메서드가 어떤 스니펫을 생성하는지 알면 주 Asciidoctor 문서에 올바른 스니펫을 포함하는 과정이 간소화됩니다.

### **B. 마스터 Asciidoctor 문서 (.adoc) 작성하기**

생성된 스니펫들을 포함하고 전체적인 API 설명을 담을 주 Asciidoctor 문서를 작성합니다.5

* **위치**: 일반적으로 Maven의 경우 src/main/asciidoc/index.adoc 12, Gradle의 경우 빌드 설정(asciidoctor 태스크의 sourceDir)에 따라 src/main/asciidoc 또는 src/docs/asciidoc.12
* **표준 Asciidoctor 파일 구조**:
  + 문서 제목: = My API Documentation
  + 저자 정보 (선택 사항)
  + 문서 속성: :doctype: book, :icons: font, :source-highlighter: highlightjs, :toc: left, :toclevels: 4, :sectlinks: 등.5 이러한 속성들은 HTML 출력물의 렌더링(목차, 코드 블록 구문 강조 등)을 제어합니다.
  + **필수 속성**: :snippets: {snippetsDir}. 이 속성은 종종 빌드 플러그인에 의해 설정됩니다 (예: spring-restdocs-asciidoctor는 {snippets}를 설정된 출력 디렉토리를 가리키도록 만듭니다).6
* **콘텐츠 구성**: Asciidoctor 섹션(예: == 리소스 엔드포인트)을 사용하여 문서를 구조화합니다.

### **C. 스니펫 삽입: include:: 및 operation:: 매크로**

Asciidoctor 문서 내에 생성된 스니펫을 삽입하는 두 가지 주요 방법이 있습니다.12

* **include::{snippets}/operation-identifier/snippet-file.adoc**: 표준 Asciidoctor 포함 방식입니다.
  + {snippets}: 생성된 스니펫의 루트를 가리키는 속성 (예: build/generated-snippets).
  + operation-identifier: document() 호출 시 지정한 이름.
  + snippet-file.adoc: 포함할 특정 스니펫 (예: curl-request.adoc). 이 방식은 어떤 스니펫을 어디에 포함할지에 대한 세밀한 제어를 제공합니다.6
* **operation::operation-identifier[snippets='snippet1,snippet2']**: spring-restdocs-asciidoctor가 제공하는 매크로입니다.18
  + 특정 작업에 대해 지정된 여러 스니펫을 포함하며, 종종 기본 제목과 함께 제공됩니다.
  + snippets 속성을 생략하면 해당 작업의 모든 스니펫을 포함합니다: operation::operation-identifier. 이 매크로는 작업에 대한 표준 스니펫 세트를 포함하는 것을 단순화합니다. 22은 제목이 생성되고 사용자 정의될 수 있는 방법을 자세히 설명합니다.

include::는 개별 스니펫을 선택하고 순서를 정하는 데 최대한의 유연성을 제공하여 고도로 사용자 정의된 레이아웃에 적합합니다. 반면, operation::은 작업에 대한 일반적인 스니펫을 묶고 기본 섹션 및 제목을 제공하여 편리함을 제공하며 22, .adoc 파일의 상용구 코드를 줄여 일관성을 유지하는 데 좋습니다. operation:: 매크로를 사용하려면 spring-restdocs-asciidoctor 확장이 필요합니다.18 사용자는 이러한 장단점을 이해해야 합니다. 단순하고 일관된 문서 구조에는 operation::이 효율적이며, 더 복잡하거나 고유하게 구조화된 문서에는 include::가 필요한 제어 기능을 제공합니다.

### **D. 최종 HTML 출력물 생성 (Maven/Gradle 빌드)**

Maven 또는 Gradle 빌드 명령을 실행하여 최종 HTML 문서를 생성합니다.

* **Maven**: mvn package (또는 문서만 원하는 경우 mvn prepare-package) 실행 시 Asciidoctor 플러그인이 트리거됩니다.12 출력물은 일반적으로 target/generated-docs에 위치합니다.
* **Gradle**: gradle build 또는 gradle asciidoctor 실행 시 asciidoctor 태스크가 트리거됩니다.12 출력물은 플러그인 버전 및 설정에 따라 build/docs/asciidoc 또는 build/asciidoc/html5에 위치합니다.20
* **문서 패키징**: 생성된 HTML을 애플리케이션 JAR/WAR에 패키징하여 정적 콘텐츠로 제공할 수 있습니다.23 Gradle의 경우:  
  Gradle  
  bootJar {  
   dependsOn asciidoctor  
   from ("${asciidoctor.outputDir}/html5") { // 실제 출력 경로에 따라 조정  
   into 'static/docs'  
   }  
  }

## **VII. 문서화 기술 향상시키기**

기본 설정을 넘어선 고급 활용법을 다룹니다.

### **A. 심층 분석: 요청/응답 페이로드, 매개변수 문서화**

Spring REST Docs는 API의 다양한 구성 요소를 상세하게 문서화할 수 있는 기능을 제공합니다.

* **요청/응답 필드 (requestFields(), responseFields())**:
  + fieldWithPath("path.to.field").description("설명").type(JsonFieldType.TYPE) 구문을 사용합니다.15
  + 중첩 객체 문서화: fieldWithPath("user.address.street").
  + 배열 문서화: 배열 자체는 fieldWithPath("items"), 배열 내 객체의 필드는 fieldWithPath("items.name").24
  + 필드를 선택 사항으로 표시: .optional().15
  + 재귀적 구조는 beneathPath()와 subsectionWithPath()를 사용하여 문서화할 수 있습니다.25
  + JSON 필드 유형에는 ARRAY, BOOLEAN, OBJECT, NUMBER, NULL, STRING, VARIES 등이 있습니다.15
* **경로 매개변수 (pathParameters())**:
  + parameterWithName("paramName").description("설명").26 MockMvc, WebTestClient, REST Assured에 대한 명확한 예제가 제공됩니다.27
* **쿼리 매개변수 (queryParameters()) / 요청 매개변수 (requestParameters()) / 폼 매개변수 (formParameters())**:
  + 유사한 구문: parameterWithName("paramName").description("설명").4
  + Spring REST Docs 3.0에서는 명확성을 위해 RequestParameterSnippet이 QueryParameterSnippet과 FormParameterSnippet으로 분리되었습니다.16
* **헤더 (requestHeaders(), responseHeaders())**:
  + headerWithName("Header-Name").description("설명").
* **재사용 가능한 디스크립터 (Reusable Descriptors)**:
  + FieldDescriptor 또는 ParameterDescriptor 배열을 정의하고, 중첩 구조나 배열에 대한 접두사와 함께 재사용할 수 있습니다.15 getCustomerFieldDescriptor()와 같은 메서드를 통해 재사용 가능한 필드 디스크립터 세트를 제공할 수 있습니다.5 API는 종종 여러 요청/응답 본문에 나타나는 공통 데이터 구조(예: User 객체, 페이지네이션 메타데이터)를 가집니다. 모든 테스트에서 이러한 공통 구조에 대한 필드 디스크립터를 다시 정의하는 것은 반복적이고 오류가 발생하기 쉽습니다.30 FieldDescriptor 배열을 반환하는 정적 유틸리티 클래스나 메서드를 만들면 5 이러한 정의를 중앙 집중화할 수 있습니다. 공통 구조 변경 시 한 곳에서만 업데이트하면 되므로, 특히 대규모 프로젝트에서 문서화 테스트의 유지보수성과 DRY(Don't Repeat Yourself) 원칙을 크게 향상시킵니다. 이는 상용구를 줄이고 공통 객체가 문서화되는 방식의 일관성을 보장합니다.

### **B. 출력물 맞춤화: 사용자 정의 스니펫 템플릿**

Spring REST Docs는 Mustache 템플릿을 사용하여 스니펫을 생성하므로, 기본 템플릿을 재정의하여 출력물을 세밀하게 제어할 수 있습니다.5

* **메커니즘**: 기본 Mustache 템플릿(.snippet 파일)을 재정의합니다.31
* **위치**: Asciidoctor의 경우 src/test/resources/org/springframework/restdocs/templates/asciidoctor/, Markdown의 경우 .../markdown/.31
* **예제**: 31는 테스트에서 추가 속성을 제공하고 템플릿을 수정하여 request-fields.snippet에 "Constraints" 열을 추가하는 상세한 예제를 제공합니다. 이는 생성된 스니펫 내용과 형식에 대한 완전한 제어를 가능하게 하여 고도로 맞춤화된 문서를 만들 수 있게 합니다.

### **C. 대체 형식: Markdown과 Slate**

Asciidoctor가 기본이지만, Markdown과 Slate를 조합하여 문서를 생성할 수도 있습니다.8

* **Markdown의 한계**: 네이티브 include 기능 부재, 테이블 지원은 Markdown Extra와 같은 확장에 의존.9
* **Slate**: "API 문서화에 특히 적합한" 도구로, Markdown 스니펫을 소비할 수 있습니다.9 REST Docs의 Markdown 출력물과 Slate를 조합하는 것이 좋은 방법으로 언급됩니다.8
* **설정**: REST Docs가 Markdown 출력물을 생성하도록 설정한 후 (예: 템플릿 형식 설정), Slate의 메커니즘을 사용하여 생성된 .md 스니펫을 포함시킵니다.
* **샘플 프로젝트**: spring-restdocs-samples의 restful-notes-slate가 이를 시연합니다.32

Asciidoctor는 기술 문서화에 더 강력하지만 8, 일부 팀이나 프로젝트는 이미 Markdown 및 Slate와 같은 도구를 중심으로 한 도구 체인을 가지고 있을 수 있습니다. Spring REST Docs는 이러한 시나리오를 위해 Markdown 출력물을 제공합니다.9 스니펫을 포함하기 위해 Slate와 같은 외부 도구가 필요하다는 점은 9 spring-restdocs-asciidoctor의 네이티브 기능에 비해 추가 단계를 의미합니다. 따라서 Asciidoctor가 일반적으로 권장되지만, 프로젝트 상황이 이를 선호한다면 Markdown과 Slate도 실행 가능한 옵션입니다.

### **D. 공식 샘플 프로젝트에서 배우기**

spring-projects/spring-restdocs-samples GitHub 저장소는 실제 작동하는 설정과 테스트 예제를 제공하여 학습에 매우 유용합니다.32 MockMvc, WebTestClient, REST Assured, Markdown과 Slate, TestNG 등 다양한 예제가 포함되어 있습니다. 특히 web-test-client 32 및 restful-notes-slate 32 샘플이 관련성이 높습니다.

## **VIII. 일반적인 설정 장애물 해결하기**

Spring REST Docs 설정 시 자주 발생하는 문제들과 해결 방법을 다룹니다.

### **A. "스니펫이 생성되지 않음" 문제 진단**

스니펫이 예상대로 생성되지 않는 경우는 여러 원인이 있을 수 있습니다.33

* **일반적인 원인** 33:
  + 테스트 태스크의 outputs.dir 또는 @AutoConfigureRestDocs의 스니펫 출력 디렉토리 설정 오류.
  + 테스트 내 document() 호출 누락 또는 잘못된 설정.
  + RestDocumentationExtension 또는 documentationConfiguration() 관련 문제.
  + document() 호출 도달 전 테스트 실패.
  + 의존성 문제.
* **디버깅** 33:
  + 스니펫 출력 디렉토리 경로를 주의 깊게 확인합니다.
  + REST Docs 내부 동작을 확인하기 위해 RestDocumentationResultHandler.handle() 메서드를 디버깅합니다.33
  + 테스트가 통과하고 document()가 실행되는지 확인합니다.

### **B. Asciidoctor 플러그인 및 구성 오류 해결**

Asciidoctor 플러그인 설정 오류는 주로 Gradle 버전 호환성 문제나 속성 누락으로 인해 발생합니다.19

* **Gradle 플러그인 버전 비호환성** 19:
  + 이전 org.asciidoctor.convert 플러그인과 최신 Gradle 버전(예: Gradle 7+) 간의 문제.
  + 오류 메시지 예: method 'asGemPath()' should not be annotated with: @Optional, @InputDirectory.
  + **해결책**: Gradle 버전을 6.x로 다운그레이드하거나, Asciidoctor 플러그인을 org.asciidoctor.jvm.convert (예: 3.3.2)로 업그레이드합니다.
* **snippets 속성 누락** 34: IntelliJ AsciiDoc 플러그인에서 IDE 컨텍스트에 {snippets} 속성이 올바르게 설정되지 않으면 operation:: 매크로를 확인하지 못할 수 있습니다. IDE 미리보기를 위해 .asciidoctorconfig 파일이 필요할 수 있습니다.

### **C. "필드가 문서화되지 않음" 또는 "스니펫을 찾을 수 없음" 경고/오류 해결**

이러한 오류는 문서화 과정에서 흔히 발생하며, 명확한 원인 파악과 해결이 필요합니다.

* **"필드가 문서화되지 않음 (Field Not Documented)"** 35:
  + 원인: 실제 페이로드에 필드가 존재하지만 fieldWithPath()로 설명되지 않은 경우.
  + 해결책: 해당 필드를 문서화하거나, 존재 여부가 가변적이라면 .optional()로 표시합니다. 35는 중첩된 선택적 필드가 이 문제를 일으킨 예를 보여줍니다.
* **"스니펫을 찾을 수 없음 (Snippet Not Found)"** 36:
  + 원인 (빌드 시, Asciidoctor 실행 중): .adoc 파일의 include:: 또는 operation:: 매크로가 생성되지 않았거나 다른 경로에 있는 스니펫을 가리키는 경우.
    - 테스트의 operation-identifier가 .adoc 파일의 경로와 일치하는지 확인합니다.
    - {snippets} 속성이 올바르게 확인되는지 검증합니다. 36는 spring-auto-restdocs 컨텍스트에서 상대 경로 대 절대 경로 스니펫 문제에 대해 논의합니다.
  + 원인 (테스트 시, 경고로 덜 일반적): 사용자 정의 템플릿에서 path-parameters와 request-parameters를 혼동하는 경우.37
* **"설정에 해당 스니펫이 없습니다 (No such snippet is present in configuration)"** 37: 이는 일반적으로 document(...)에서 스니펫 유형(예: responseFields())이 요청되었지만 해당 조건이 충족되지 않은 경우(예: 빈 응답 본문)를 의미합니다.

"Field Not Documented"와 같은 오류 35나 문서화된 필드 누락으로 인한 테스트 실패 15는 번거롭게 느껴질 수 있습니다. 그러나 이러한 엄격함은 개발자가 API 문서를 명시적이고 포괄적으로 작성하도록 강제합니다. 이는 문서화되지 않은 필드가 API에 슬며시 포함되거나 문서화된 필드가 조용히 제거되는 것을 방지합니다. 필드를 .optional()로 표시하거나 relaxedResponseFields()를 사용하는 옵션은 필요할 때 유연성을 제공하지만, 기본은 엄격합니다. 이러한 엄격함은 문서 정확성을 보장하는 핵심 부분입니다. 따라서 이러한 "오류"를 단순히 수정해야 할 문제로만 볼 것이 아니라, API 품질과 문서 무결성을 향상시키는 피드백 메커니즘으로 이해해야 합니다.

## **IX. 유지보수 가능하고 효과적인 문서를 위한 모범 사례 (기본 설정 이상)**

대규모 프로젝트를 위한 고급 팁과 전략을 통합하여 설명합니다.

### **A. 스니펫 및 .adoc 파일 구성**

효율적인 문서 관리를 위해서는 스니펫과 Asciidoctor 파일의 체계적인 구성이 중요합니다.6

* **논리적 그룹화**: 주 .adoc 파일에서 Asciidoctor 섹션을 사용하여 관련된 API 엔드포인트를 그룹화합니다.22
* **스니펫 디렉토리 구조**: operation-identifier는 자연스럽게 디렉토리 구조(예: users/create-user, users/get-user)를 만듭니다. 이러한 식별자를 일관되고 설명적으로 유지합니다.
* **마스터 문서**: 문서가 매우 커지면 다른 리소스 그룹에 대한 다른 .adoc 파일을 포함하는 주 index.adoc6을 사용합니다. 22 (Working with Asciidoctor)은 여러 스니펫을 포함하고 제목을 사용자 정의하기 위한 operation:: 매크로 사용과 개별 스니펫을 위한 include:: 사용에 대한 훌륭한 지침을 제공합니다.

### **B. 재사용 가능한 필드 디스크립터 (요약 및 강조)**

VII.A에서 논의한 바와 같이, 이는 공통 DTO를 사용하는 대규모 프로젝트의 유지보수성에 매우 중요합니다.5 공통 FieldDescriptor 객체 세트를 정의하기 위해 유틸리티 클래스나 정적 메서드를 만듭니다.

### **C. API 문서 버전 관리**

REST Docs 설정에서 직접 다루지는 않지만, 일반적인 API 모범 사례에는 버전 관리가 포함됩니다.38 다양한 API 버전에 대한 문서를 관리하기 위한 전략(예: 별도 브랜치, 버전에 따른 스니펫/문서에 대한 다른 출력 디렉토리)을 고려합니다. 이는 빌드 스크립트 수정이 필요할 수 있습니다.

### **D. 제약 조건 및 유효성 검사 문서화**

API 소비자를 위해 데이터 유효성 검사 규칙(예: 최대 길이, 필수 필드)을 명확히 문서화하는 것은 매우 중요합니다.

* Spring REST Docs는 Bean Validation 어노테이션을 활용할 수 있습니다.7
* spring-auto-restdocs 41와 같은 확장은 JSR 303 제약 조건의 문서화를 자동화하는 것을 목표로 합니다.
* 수동으로는 사용자 정의 템플릿의 속성을 통해 제약 조건 정보를 추가할 수 있습니다.31

### **E. 문서 간결하고 집중적으로 유지하기**

REST Docs는 상세한 문서화를 허용하지만, 사용자를 압도하지 않도록 주의해야 합니다. API를 효과적으로 사용하는 데 필요한 정보에 집중합니다.1 필드와 매개변수에 대해 명확하고 설명적인 언어를 사용하고 13, REST Docs가 스니펫을 통해 생성하는 데 도움이 되는 예제를 포함합니다.

### **F. 정기적인 검토 및 유지보수**

문서가 테스트와 연결되어 있으므로 테스트 범위가 양호하고 유지보수되는지 확인합니다. 생성된 HTML 문서를 주기적으로 검토하여 명확성과 완전성을 확인합니다.

### **G. 코드로써의 문서 (Documentation as Code)**

Spring REST Docs는 문서 결과물(스니펫, .adoc 파일)을 코드베이스의 일부로 취급합니다. 이러한 파일은 소스 코드와 함께 버전 관리됩니다. 빌드 프로세스는 코드를 컴파일하는 것과 유사하게 최종 문서를 생성합니다. 테스트 기반 특성은 API 동작 변경이 테스트를 중단시키면 문서 생성도 중단/업데이트됨을 의미합니다. 이는 문서가 애플리케이션 코드 자체와 동일한 개발, 검토 및 CI/CD 프로세스를 따르는 "Docs as Code" 원칙과 일치합니다. 이는 특히 대규모 프로젝트 및 팀에 유익한 보다 강력하고 유지보수 가능하며 통합된 문서화 관행으로 이어집니다.40

## **X. 결론**

Spring REST Docs는 테스트 주도 접근 방식을 통해 API 문서의 정확성과 신뢰성을 획기적으로 향상시키는 강력한 도구입니다. 개발 프로세스에 문서화를 자연스럽게 통합함으로써, API의 실제 동작과 항상 일치하는 최신 문서를 유지할 수 있게 해줍니다. Maven 또는 Gradle 기반 프로젝트에서의 체계적인 설정, MockMvc 및 WebTestClient와의 유연한 연동, Asciidoctor를 통한 풍부한 표현력은 Spring REST Docs를 선택해야 하는 충분한 이유를 제공합니다.

특히, 재사용 가능한 디스크립터, 사용자 정의 템플릿, 그리고 "코드로써의 문서(Docs as Code)" 철학의 적용은 대규모 프로젝트에서도 문서의 유지보수성을 높이고 일관성을 확보하는 데 크게 기여합니다. 초기 설정의 복잡성이나 발생할 수 있는 몇몇 문제점들은 이 가이드에서 제시된 해결책과 모범 사례를 통해 충분히 극복 가능합니다.

궁극적으로 Spring REST Docs는 단순한 문서 생성 도구를 넘어, API 설계의 품질을 높이고 개발팀과 API 사용자 간의 명확한 소통을 가능하게 하는 핵심적인 역할을 수행합니다. 정확하고, 읽기 쉬우며, 신뢰할 수 있는 API 문서를 구축하고자 하는 모든 Spring 개발자에게 Spring REST Docs의 도입을 적극 권장합니다.

#### 참고 자료

1. Spring REST Docs, 5월 7, 2025에 액세스, <https://spring.io/projects/spring-restdocs/>
2. spring-projects/spring-restdocs: Test-driven documentation for RESTful services - GitHub, 5월 7, 2025에 액세스, <https://github.com/spring-projects/spring-restdocs>
3. Spring REST Docs interrogating @RestController? - Stack Overflow, 5월 7, 2025에 액세스, <https://stackoverflow.com/questions/40029481/spring-rest-docs-interrogating-restcontroller>
4. Document Query Parameters with Spring REST Docs - Baeldung, 5월 7, 2025에 액세스, <https://www.baeldung.com/spring-rest-document-query-parameters>
5. API Documentation with Spring REST Docs - mimacom blog, 5월 7, 2025에 액세스, <https://blog.mimacom.com/api-documentation-with-spring-rest-docs/>
6. Introduction to Spring REST Docs - Baeldung, 5월 7, 2025에 액세스, <https://www.baeldung.com/spring-rest-docs>
7. Spring REST Docs, 5월 7, 2025에 액세스, <https://docs.spring.io/spring-restdocs/docs/2.0.5.RELEASE/reference/html5/>
8. Spring I/O 16: Test-driven documentation with Spring REST Docs - Kevin Van Houtte, 5월 7, 2025에 액세스, <https://blog.ordina-jworks.io/conference/2016/06/30/SpringIO16-Spring-Rest-Docs.html>
9. Working with Markdown (Working with Markdown) | Spring REST Docsversion 3.0.2-SNAPSHOT中文文档|Spring官方文档|SpringBoot 教程, 5월 7, 2025에 액세스, <https://www.spring-doc.cn/spring-restdocs/3.0.2-SNAPSHOT/S_6_working-with-markdown.en.html>
10. spring-restdocs/docs/src/docs/asciidoc/working-with-markdown.adoc at main - GitHub, 5월 7, 2025에 액세스, <https://github.com/spring-projects/spring-restdocs/blob/main/docs/src/docs/asciidoc/working-with-markdown.adoc>
11. Spring REST Docs, 5월 7, 2025에 액세스, <https://docs.spring.io/spring-restdocs/docs/current/reference/htmlsingle/#working-with-markdown>
12. Getting Started | Creating API Documentation with Restdocs - Spring, 5월 7, 2025에 액세스, <https://spring.io/guides/gs/testing-restdocs/>
13. What is Spring Rest Docs ? - Apidog, 5월 7, 2025에 액세스, <https://apidog.com/blog/spring-rest-docs/>
14. Spring REST Docs 1.0.0.M1, 5월 7, 2025에 액세스, <https://spring.io/blog/2015/06/25/spring-rest-docs-1-0-0-m1>
15. spring-restdocs/docs/src/docs/asciidoc/documenting-your-api.adoc at main - GitHub, 5월 7, 2025에 액세스, <https://github.com/spring-projects/spring-restdocs/blob/main/docs/src/docs/asciidoc/documenting-your-api.adoc>
16. Releases · spring-projects/spring-restdocs - GitHub, 5월 7, 2025에 액세스, <https://github.com/spring-projects/spring-restdocs/releases>
17. Spring REST Docs, 5월 7, 2025에 액세스, <https://docs.spring.io/spring-restdocs/docs/1.1.0.RELEASE/reference/html5/>
18. Spring REST Docs, 5월 7, 2025에 액세스, <https://docs.spring.io/spring-restdocs/docs/current/reference/htmlsingle/#getting-started-build-configuration-gradle>
19. Configuring asciidoctor when using Spring Restdoc - Stack Overflow, 5월 7, 2025에 액세스, <https://stackoverflow.com/questions/68539790/configuring-asciidoctor-when-using-spring-restdoc>
20. Asciidoctor Gradle Plugin, 5월 7, 2025에 액세스, <https://asciidoctor.github.io/asciidoctor-gradle-plugin/maintenance-1.5/>
21. ePages-de/restdocs-api-spec: Adds API specification support to Spring REST Docs - GitHub, 5월 7, 2025에 액세스, <https://github.com/ePages-de/restdocs-api-spec>
22. spring-restdocs/docs/src/docs/asciidoc/working-with-asciidoctor.adoc at main - GitHub, 5월 7, 2025에 액세스, <https://github.com/spring-projects/spring-restdocs/blob/main/docs/src/docs/asciidoc/working-with-asciidoctor.adoc>
23. Publish spring-restdocs html documentation with application - Stack Overflow, 5월 7, 2025에 액세스, <https://stackoverflow.com/questions/49558157/publish-spring-restdocs-html-documentation-with-application>
24. How to document top-level array as response payload with Spring REST Docs, 5월 7, 2025에 액세스, <https://stackoverflow.com/questions/31162140/how-to-document-top-level-array-as-response-payload-with-spring-rest-docs>
25. How to document recursive data structures in response with Spring REST Docs, 5월 7, 2025에 액세스, <https://stackoverflow.com/questions/59993338/how-to-document-recursive-data-structures-in-response-with-spring-rest-docs>
26. PathParametersSnippet (Spring REST Docs 3.0.3 API), 5월 7, 2025에 액세스, <https://docs.spring.io/spring-restdocs/docs/current/api/org/springframework/restdocs/request/PathParametersSnippet.html>
27. Spring REST Docs, 5월 7, 2025에 액세스, <https://docs.spring.io/spring-restdocs/docs/current/reference/htmlsingle/#documenting-your-api-request-parameters-path-parameters>
28. Spring REST Docs, 5월 7, 2025에 액세스, <https://docs.spring.io/spring-restdocs/docs/current/reference/htmlsingle/>
29. Issue when documenting constraints & reusing field descriptors · Issue #313 · spring-projects/spring-restdocs - GitHub, 5월 7, 2025에 액세스, <https://github.com/spring-projects/spring-restdocs/issues/313>
30. Spring Rest Docs - Documenting REST API | Example | Java Techie - YouTube, 5월 7, 2025에 액세스, <https://www.youtube.com/watch?v=ghn9p6d__Yc>
31. Spring REST Docs, 5월 7, 2025에 액세스, <https://docs.spring.io/spring-restdocs/docs/current/reference/htmlsingle/#customizing-documentation-snippets>
32. spring-projects/spring-restdocs-samples - GitHub, 5월 7, 2025에 액세스, <https://github.com/spring-projects/spring-restdocs-samples>
33. spring rest docs snippet not created - Stack Overflow, 5월 7, 2025에 액세스, <https://stackoverflow.com/questions/78613999/spring-rest-docs-snippet-not-created>
34. Rendering preview for spring-restdocs documentation · Issue #310 - GitHub, 5월 7, 2025에 액세스, <https://github.com/asciidoctor/asciidoctor-intellij-plugin/issues/310>
35. Spring REST Docs - parts not documented: - Stack Overflow, 5월 7, 2025에 액세스, <https://stackoverflow.com/questions/58322539/spring-rest-docs-parts-not-documented>
36. Section snippet does not work with spring-restdocs-asciidoctor · Issue #215 - GitHub, 5월 7, 2025에 액세스, <https://github.com/ScaCap/spring-auto-restdocs/issues/215>
37. Spring Rest Docs snippet template is ignored - java - Stack Overflow, 5월 7, 2025에 액세스, <https://stackoverflow.com/questions/52093601/spring-rest-docs-snippet-template-is-ignored>
38. Top 10 Spring Boot REST API Best Practices You MUST Know - YouTube, 5월 7, 2025에 액세스, <https://www.youtube.com/watch?v=EgQJRB9Vs3Y>
39. Common Mistakes When Designing RESTful APIs with Spring Boot - DEV Community, 5월 7, 2025에 액세스, <https://dev.to/jackynote/common-mistakes-when-designing-restful-apis-with-spring-boot-2l6j>
40. Java Spring Boot Rest Api Best Practices - Restack, 5월 7, 2025에 액세스, <https://www.restack.io/p/java-spring-boot-answer-best-practices>
41. ScaCap/spring-auto-restdocs: Spring Auto REST Docs is an extension to Spring REST Docs - GitHub, 5월 7, 2025에 액세스, <https://github.com/ScaCap/spring-auto-restdocs>
42. arsy786/springboot-best-practices: This repo serves as a guide for Spring Boot best practices! - GitHub, 5월 7, 2025에 액세스, <https://github.com/arsy786/springboot-best-practices>