(http://baeldung.com)

Introduction to Spring REST Docs

Last modified: July 20, 2017

by baeldung (http://www.baeldung.com/author/baeldung/)

REST (http://www.baeldung.com/category/rest/) Spring (http://www.baeldung.com/category/spring/) + testing (http://www.baeldung.com/category/testing/)

1. Overview

Spring REST Docs (http://projects.spring.io/spring-restdocs/) generates documentation for RESTful services that is both accurate and readable. It combines hand-written documentation with auto-generated document snippets produced with Spring MVC tests.

2. Advantages

One major philosophy behind the project is the use of tests to produce the documentation. This ensures that the documentation generated always accurately matches the actual behavior of the API. Additionally, the output is ready to be processed by Asciidoctor (http://asciidoctor.org/), a publishing toolchain centered around the AsciiDoc syntax. This is the same tool that is used to generate the Spring Framework's documentation.

These approaches reduce the limitations imposed by other frameworks. Spring REST Docs produces documentation that is accurate, concise, and well-structured. This documentation then allows the web service consumers to get the information they need with a minimum of fuss.

The tool has some other advantages, such as:

- curl and http request snippets are generated
- easy to package documentation in projects jar file
- easy to add extra information to the snippets
- supports both JSON and XML

3. Dependencies

The ideal way to get started using Spring REST Docs in a project is by using a dependency management system. Here, we are using Maven as build tool, so the dependency below can be copied and pasted into your POM:

```
<dependency>
    <groupId>org.springframework.restdocs</groupId>
    <artifactId>spring-restdocs-mockmvc</artifactId>
    <version>1.0.1.RELEASE</version>
</dependency>
```

You can also check Maven Central for a new version of the dependency here (http://search.maven.org/#search|gav|1|g%3A%22org.springframework.restdocs%22%20AND%20a%3A%22spring-restdocs-mockmvc%22).

4. Configuration

Spring REST Docs can be created using the Spring MVC Test framework to make requests to the REST services which are to be documented. This produces documentation snippets for the request and the resulting response.

The very first step in generating documentation snippets is to declare a public *RestDocumentation* field that is annotated as a JUnit *@Rule*. The *RestDocumentation* rule is configured with the output directory into which the generated snippets should be saved. For example, this directory can be the build out directory of Maven:

```
@Rule
public RestDocumentation restDocumentation = new RestDocumentation("target/generated-snippets");
```

Next, we set up the *MockMvc* context so that it will be configured to produce documentation.

```
@Autowired
private WebApplicationContext context;

private MockMvc mockMvc;

@Before
public void setUp(){
    this.mockMvc = MockMvcBuilders.webAppContextSetup(this.context)
      .apply(documentationConfiguration(this.restDocumentation))
      .build();
}
```

The *MockMvc* object is configured using a *RestDocumentationMockMvcConfigurer*. An instance of this class can be obtained from the static *documentationConfiguration()* method on *org.springframework.restdocs.mockmvc.MockMvcRestDocumentation*.

Let's create a CRUD RESTful service that we can document:

```
@RestController
@RequestMapping("/crud")
public class CRUDController {

    @RequestMapping(method=RequestMethod.GET)
    public List<CrudInput> read() {
        List<CrudInput> returnList=new ArrayList<CrudInput>();
        return returnList;
    }

    @ResponseStatus(HttpStatus.CREATED)
    @RequestMapping(method=RequestMethod.POST)
    public HttpHeaders save(@RequestBody CrudInput crudInput) {
        HttpHeaders httpHeaders = new HttpHeaders();
        httpHeaders.setLocation(linkTo(CRUDController.class).slash(crudInput.getTitle()).toUri());
        return httpHeaders;
    }

    @RequestMapping(value = "/{id}", method = RequestMethod.DELETE)
    void delete(@PathVariable("id") long id) {
        // delete
    }
}
```

Back in the tests, a *MockMvc* instance has been already created. It can be used to call the service we just created, and will document the request and response. The method below will generate the document in the configured directory:

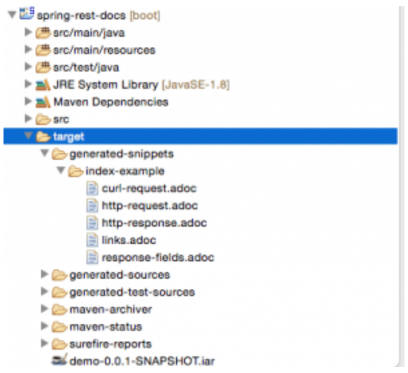
```java
@Test
public void index() throws Exception {
    this.mockMvc.perform(
      get("/")
        .accept(MediaTypes.HAL_JSON))
        .andExpect(status().isOk())
        .andExpect(jsonPath("_links.crud", is(notNullValue())));
}
```

It is time to create the document snippets for the REST service:

```java
@Test
public void indexExample() throws Exception {
    this.document.snippets(
      links(
        linkWithRel("notes").description("The <<resources-notes,Notes resource>>"),
        linkWithRel("tags").description("The <<resources-tags,Tags resource>>")
      ),
      responseFields(
        fieldWithPath("_links").description("<<resources-index-links,Links>> to other resources")
      )
    );
    this.mockMvc.perform(get("/rest/api")).andExpect(status().isOk());
}
```

5. Output

Once the Maven build runs successfully, the output of the REST docs snippets will be generated and will be saved to the *target/generated-snippets* folder.



```
spring-rest-docs [boot]
  src/main/java
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-1.8]
  Maven Dependencies
  src
  target
    generated-snippets
      index-example
        curl-request.adoc
        http-request.adoc
        http-response.adoc
        links.adoc
        response-fields.adoc
    generated-sources
    generated-test-sources
    maven-archiver
    maven-status
    surefire-reports
    demo-0.0.1-SNAPSHOT.jar
```

(/wp-content/uploads/2016/07/Screen-Shot-2016-04-04-at-11.48.52-PM-300x270.png)

The generated output will have the information about the service, how to call the REST service like 'curl' calls, the HTTP request and response from the REST service, and links/endpoints to the service:

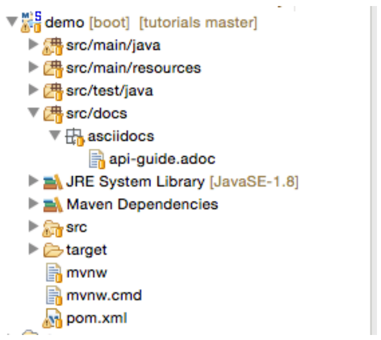
```
<strong>CURL Command</strong>
```

```
----
$ curl 'http://localhost:8080/api (http://localhost:8080/api)' -i
----

<strong>HTTP - REST Response</strong>
```

```
[source,http]
----
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 160

{
  "_links" : {
    "notes" : {
      "href" : "http://localhost:8080/testapi (http://localhost:8080/testapi)"
    },
    "tags" : {
      "href" : "http://localhost:8080/testapi (http://localhost:8080/testapi)"
    }
  }
}
----
```

6. Using Snippets to Create Documentation

In order to use the snippets in a larger document, you can reference them using Asciidoc inlcudes. In our case, we have created a document in *src/docs* called *api-guide.adoc*:



(/wp-content/uploads/2016/07/Screen-Shot-2016-05-01-at-8.51.48-PM.png)

In that document, if we wished to reference the response headers snippet, we can include it, using a placeholder *{snippets}* that will be replaced by Maven when it processes the document:

```
[[overview-headers]]
== Headers

Every response has the following header(s):

include::{snippets}/headers-example/response-headers.adoc[]
```

7. Asciidocs Maven Plugins

To convert the API guide from Asciidoc to a readable format, we can add a Maven plugin to the build lifecycle. There are several steps to enable this:

1. Apply the Asciidoctor plugin to the *pom.xml*
2. Add a dependency on *spring-restdocs-mockmvc* in the *testCompile* configuration as mentioned in the dependencies section
3. Configure a property to define the output location for generated snippets
4. Configure the *test* task to add the snippets directory as an output
5. Configure the *asciidoctor* task
6. Define an attribute named *snippets* that can be used when including the generated snippets in your documentation
7. Make the task depend on the *test* task so that the tests are run before the documentation is created
8. Configure the *snippets* directory as an input. All the generated snippets will be created under this directory

Add the snippet directory as a property in *pom.xml* so the Asciidoctor plugin can use this path to generate the snippets under this folder:

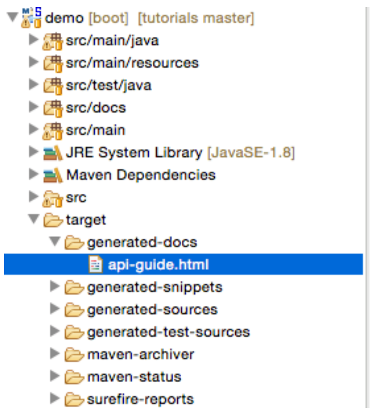
```
<properties>
    <snippetsDirectory>${project.build.directory}/generated-snippets</snippetsDirectory>
</properties>
```

The Maven plugin configuration in the *pom.xml* to generate the Asciidoc snippets from the build is as below:

```xml
<build>
    <plugins>
        <plugin>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-maven-plugin</artifactId>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-surefire-plugin</artifactId>
            <configuration>
            <includes>
                    <include>**/*Documentation.java</include>
            </includes>
        </configuration>
        </plugin>
        <plugin>
            <groupId>org.asciidoctor</groupId>
        <artifactId>asciidoctor-maven-plugin</artifactId>
        <version>1.5.2</version>
        <executions>
            <execution>
                <id>generate-docs</id>
                <phase>package</phase>
                <goals>
                    <goal>process-asciidoc</goal>
                </goals>
                <configuration>
                    <backend>html</backend>
                    <doctype>book</doctype>
                    <attributes>
                        <snippets>${snippetsDirectory}</snippets>
                    </attributes>
                    <sourceDirectory>src/docs/asciidocs</sourceDirectory>
                    <outputDirectory>target/generated-docs</outputDirectory>
                    </configuration>
            </execution>
        </executions>
        </plugin>
    </plugins>
</build>
```

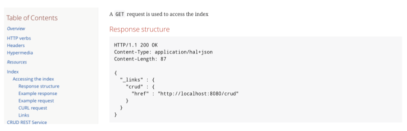
8. API Doc generation process

When the Maven build runs and the tests are executed, all the snippets will be generated in the snippets folder under the configured *target/generated-snippets* directory. Once the snippets are generated, the build process generates HTML output.



(/wp-content/uploads/2016/07/Screen-Shot-2016-05-08-at-11.32.25-PM.png)

The generated HTML file is formatted and readable, so the REST documentation is ready to use. Every time the Maven build runs, the documents also get generated with the latest updates.



A `GET` request is used to access the index

Response structure

```
HTTP/1.1 200 OK
Content-Type: application/hal+json
Content-Length: 87

{
  "_links" : {
    "crud" : {
      "href" : "http://localhost:8080/crud"
    }
  }
}
```

(/wp-content/uploads/2016/07/Screen-Shot-2016-05-08-at-11.36.36-PM.png)

9. Conclusion

Having no documentation is better than wrong documentation, but Spring REST docs will help generate accurate documentation for RESTful services.

As an official Spring project, it accomplishes its goals by using the Spring MVC Test library. This method of generating documentation can help support a test-driven approach to developing and documenting RESTful APIs.

You can find an example project based on the code in this article in the linked GitHub repository (https://github.com/eugenp/tutorials/tree/master/spring-rest-docs).

CATEGORIES

SPRING (HTTP://WWW.BAELDUNG.COM/CATEGORY/SPRING/)
REST (HTTP://WWW.BAELDUNG.COM/CATEGORY/REST/)
JAVA (HTTP://WWW.BAELDUNG.COM/CATEGORY/JAVA/)
SECURITY (HTTP://WWW.BAELDUNG.COM/CATEGORY/SECURITY-2/)
PERSISTENCE (HTTP://WWW.BAELDUNG.COM/CATEGORY/PERSISTENCE/)
JACKSON (HTTP://WWW.BAELDUNG.COM/CATEGORY/JACKSON/)
HTTPCLIENT (HTTP://WWW.BAELDUNG.COM/CATEGORY/HTTP/)

ABOUT

ABOUT BAELDUNG (HTTP://WWW.BAELDUNG.COM/ABOUT/)
THE COURSES (HTTP://COURSES.BAELDUNG.COM)
META BAELDUNG (HTTP://META.BAELDUNG.COM/)
THE FULL ARCHIVE (HTTP://WWW.BAELDUNG.COM/FULL_ARCHIVE)
WRITE FOR BAELDUNG (HTTP://WWW.BAELDUNG.COM/CONTRIBUTION-GUIDELINES)
PRIVACY POLICY (HTTP://WWW.BAELDUNG.COM/PRIVACY-POLICY)
TERMS OF SERVICE (HTTP://WWW.BAELDUNG.COM/TERMS-OF-SERVICE)
CONTACT (HTTP://WWW.BAELDUNG.COM/CONTACT)
COMPANY INFO (HTTP://WWW.BAELDUNG.COM/BAELDUNG-COMPANY-INFO)
ADVERTISE ON THE JAVA WEEKLY (HTTP://WWW.BAELDUNG.COM/JAVA-WEEKLY-SPONSORSHIP)
CONSULTING WORK (HTTP://WWW.BAELDUNG.COM/CONSULTING)