Dokumentation von RESTful APIs mittels Swagger, AsciiDoc und Swagger2Markup

Im Zeitalter des Internet of Things (IoT) sind öffentliche Web APIs ein wichtiger Wettbewerbsfaktor. Der häufig genutzte Begriff API Economy rückt immer stärker in den Fokus. Sowohl öffentliche als interne APIs benötigen einen verständlichen Developer Guide, damit sie von Entwicklern eingesetzt werden können. Mit einer guten Dokumentation steht und fällt die Verbreitung einer Web API. Dieser Artikel ist in zwei Teile aufgeteilt. Teil 1 zeigt wie man auf Basis einer Swagger Spezifikation mit Hilfe der Open Source Projekte Swagger2Markup und Asciidoctorj einen Developer Guide erstellen kann, der mit handgeschriebener AsciiDoc-Dokumentation kombiniert werden kann und sowohl offline (PDF-Format) als auch online (HTML-Format) lesbar ist. Teil 2 zeigt wie man mit Hilfe des Open Source Projektes Springfox eine Swagger Spezifikation aus einer Spring Boot (MVC) Applikation generieren und wie man mit Hilfe von Spring Rest Docs sinnvolle HTTP Request und Response Beispiele erstellen kann. Mit Swagger2Markup können dann alle Dokumente in eine einzelne Dokumentation vereint werden.

von Robert Winkler

In der letzen Zeit sind viele Spezifikationsformate und Frameworks entstanden, die die Spezifikation und Dokumentation einer HTTP-basierten API vereinfachen und vereinheitlichen sollen. Zu den bekanntesten Spezifikationsformaten gehören Swagger (nun OpenAPI Spezifikation[1]), RAML[2] oder API Blueprint[3]. All diese Projekte haben zum Ziel ein Spezifikationsformat zu entwickeln, dass maschinenlesbar ist, aber in erster Linie von Menschen erstellt werden können soll. Sie nutzen hierfür unterschiedliche Formate wie JSON, YAML oder Markdown. XML-basierte Formate wie die Web Application Description Language (WADL) eignen sich hierfür weniger.Diese Spezifikationsformate für sich alleine stellen aber keinen Developer Guide dar, wie beispielweise der GitHub API Developer Guide[4]. Developer Guides sollten weitere Informationen enthalten, wie beispielsweise ein Tutorial mit echten HTTP Request und Response Beispielen, Code Beispielen und CURL-Kommandos, mit denen die API getestet werden kann.

Swagger

Swagger ist das momentan weitverbreitetste Spezifikationsformat für HTTP-basierte APIs. Um Swagger herum hat sich eine große Tool-Landschaft gebildet. Zum Beispiel die Online-Kollaborationsplattform SwaggerHub, die eine Zusammenarbeit an Swagger Spezifikationen ermöglichen soll. SwaggerHub unterstützt den gesamten Lebenszyklus einer API und bietet hierfür einen Online-Editor für die Erstellung einer Swagger Spezifikationen an, sowie eine Integration mit GitHub, Versionisierung und Codegenerierung.Ursprünglich war Swagger in der Version 1.2 ein Code-First Framework mit Hilfe dessen aus Source Code eine Swagger Spezifikation im JSON Format generiert werden konnte. Seit Version 2 wird auf einen Contract-First Ansatz im YAML-Format gesetzt. Die Version 2 wurde von der Open API Initiative als Basis für die OpenAPI Spezifikation übernommen.

YAML

YAML ist eine Auszeichnungssprache zur Datenserialisierung und eine Übermenge von JSON. Das Akronym YAML steht für „YAML Ain’t Markup Language“. YAML ist leichter von Menschen zu lesen und zu schreiben als beispielsweise JSON oder XML.

Ende Kasten

SwaggerHub bietet einen Online-Editor an, mit dem eine API erstellt und validiert werden kann.

swagger-ui-editor.jpg

Abb. 1: SwaggerHub Online Editor

In Listing 1 wird ein Ausschnitt der Swagger Petstore API Demo Spezifikation dargestellt. Eine Swagger Spezifikation muss immer Meta-Informationen wie einen Titel und eine Version enthalten. Der Hostname und ein Basispfad sind nicht zwingend notwendig. Anschließend können API-Endpunkte, globale Parameter, globale Antwortnachrichten und globale Datentypen spezifiziert werden. Der API-Endpunkt */pets/{petId}* beispielsweise ermöglicht es die Ressource Pet über eine ID zu finden. Innerhalb des API-Endpunktes werden die Input-Parameter des Endpunkts und die möglichen Antwortnachrichten spezifiziert. Die Antwortnachricht mit dem Status-Code 200 referenziert die global Definition für das *Pet* Datenmodell, das in der Antwortnachricht zurückgeliefert wird. Swagger kann im Rahmen dieses Artikels nicht vollständig erläutert werden, da das Spezifikationformat sehr umfangreich ist. Mit Swagger können viele HTTP-basierte APIs abgebildet werden. Lediglich HATEOS als finales Level eine Restful API ist mit Swagger derzeit noch nicht abbildbar.

Listing 1

swagger: "2.0"info: description: This is a sample server Petstore server. version: "1.0.0" title: Swagger Petstorehost: petstore.swagger.wordnik.combasePath: /v2schemes: - httpspaths: /pets/{petId}: get: tags: - pet summary: Find pet by ID description: Returns a pet produces: - application/json parameters: - in: path name: petId description: ID of pet that needs to be fetched required: true type: integer format: int64 responses: "200": description: successful operation schema: $ref: "#/definitions/Pet" "400": description: Invalid ID supplied "404": description: Pet not founddefinitions: Pet: required: - name - photoUrls properties: id: type: integer format: int64 category: $ref: "#/definitions/Category" name: type: string example: doggie status: type: string description: pet status in the store

Ende

Die erstellte Spezifikation wird in dem Online-Editor live dargestellt.

html\_swagger.PNG

Abb. 2: Swagger HTML Dokumentation

Diese interaktive HTML-Dokumentation hat einige Mankos. Zum einen liegt der Fokus der Dokumentation eindeutig auf den API Endpunkten, z.B. *GET /pets/{petId}* und nicht auf einer kurzen Beschreibung der Funktionalität, die von diesem Endpunkt angeboten wird, wie z.B. "Find Pet by ID". Ein Entwickler, dem die API unbekannt ist, würde in erster Linie nicht nach einem konkreten API Endpunkt suchen, sondern nach der von ihm benötigten Funktionalität.Des Weiteren kann die HTML-Dokumentation nicht offline angeschaut oder als PDF exportiert werden. Ein PDF-Format ist aber häufig notwendig, wenn die API-Dokumentation in einer Ausschreibung für ein Software-Projekt mit ausgehändigt werden muss.Ein weiteres Manko ist, dass diese Dokumentation nicht um weitere Informationen erweitert werden kann, wie z.B. um Nutzungsbeispiele in verschiedenen Programmiersprachen.

Es stellen sich also zwei Fragen:

* Wie kann eine Swagger Spezifikation um handgeschriebene Dokumentation erweitert werden?
* Wie kann die Dokumentation sowohl im HTML als auch im PDF Format erstellt werden?

Hierbei können uns die Projekte AsciiDoctorj und Swagger2Markup behilflich sein.

**AsciiDoc und Asciidoctorj**

AsciiDoc[4] ist eine Auszeichnungssprache vergleichbar mit Markdown, allerdings von der Funktionalität deutlich umfangreicher. AsciiDoc Dateien können mit Hilfe von Asciidoctorj in mehrere Formate wie z.B. HTML, PDF, EPUB oder DocBook konvertiert werden. AsciiDoc, aber auch Markdown, eignen sich hervorragend dafür eine handgeschriebene, technische Dokumentation zu erstellen. Ein einfaches AsciiDoc Dokument sieht wie in Listing 2 aus.

Listing 2

= Introduction to AsciiDocDoc Writer <doc@example.com>A preface about http://asciidoc.org[AsciiDoc].

== First Section\* list item 1\* list item 2

\_italic phrase\_\*bold phrase\*`monospace phrase`

[source,ruby]puts "Hello, World!"

image::example.jpg[]include::second\_section.adoc[]

Ende

Mit AsciiDoc kann ein Dokument in Kapitel unterteilt werden. Ein großer Vorteil von AsciiDoc gegenüber Markdown ist, dass man mit Hilfe der Include-Direktive große Dokumente in kleine Dateien aufteilen kann. Mit AsciiDoc lassen sich Texte formatieren und Listen, Tabellen und Source Code darstellen. Man kann auch UML-Diagramme erstellen und Bilder, Videos oder Audio inkludieren. Da AsciiDoc nur Plain-Text ist, eignet es sich, im Gegensatz zu Word-Dateien, mit Git versionisiert zu werden und ermöglicht somit das Arbeiten mit mehreren Leuten an einer Dokumentation. Mit AsciidocFX gibt es einen tollen JavaFX-basierten Editor, um AsciiDoc Dokumente mit Live-Preview zu schreiben. Bekannte Open Source Projekte wie das Spring Framework oder Spring Boot nutzen AsciiDoc für die Erstellung der Dokumentation.

**Swagger2Markup**

Mit Swagger2Markup[5] gibt es ein Open Source Projekt, dass es ermöglicht eine Swagger Spezifikation in AsciiDoc, Markdown or Atlassian Confluence Wiki Markup zu konvertieren. Die Swagger Spezifikation kann sowohl im YAML als auch im JSON Format vorliegen und sowohl lokal als auch remote über HTTP geladen werden. Die Verwendung von Swagger2Markup ist simpel. Listing 3 zeigt wie eine Swagger Spezifikation in AsciiDoc konvertiert werden kann. AsciiDoc ist als Standard-Ausgabeformat vorkonfiguriert. Es muss nur die Quelle und der Zielordner angegeben werden. In diesem Beispiel wird die Swagger Petstore API Demo Spezifikation über HTTP geladen.

Listing 3

URL remoteSwaggerFile = new URL("http://petstore.swagger.io/v2/swagger.json");Path outputDirectory = Paths.get("build/asciidoc");Swagger2MarkupConverter.from(remoteSwaggerFile) .build() .toFolder(outputDirectory);

Ende

Standardmäßig werden aus einer Swagger Spezifikation vier AsciiDoc-Dateien generiert: *overview.adoc*, *security.adoc*, *paths.adoc* und *definitions.adoc*. Diese generierten Dateien können nun mit handgeschriebener Dokumentation (z.B. *hand\_written\_1.adoc* und *hand\_written\_2.adoc*) beliebig kombiniert werden. Es können aber auch einzelne Dateien für jede Operation und Model Definition generiert werden. Die handgeschriebene Dokumentation kann Informationen beinhalten, die mit Swagger nicht abbildbar sind, z.b. ein Tutorial zur Nutzung der API mit Request und Response Beispielen.Hierzu erstellt man eine *index.adoc* Datei, die alle Dateien inkludiert. Der Platzhalter *{generated}* steht für den Ordner in den die Dateien generiert wurden. Siehe Listing 4.

Listing 4

include::{generated}/overview.adoc[]include::manual\_content1.adoc[]include::manual\_content2.adoc[]include::{generated}/security.adoc[]include::{generated}/paths.adoc[]include::{generated}/definitions.adoc[]

Ende

Mit Hilfe des Open Source Projekts Asciidoctorj[6] kann die *index.adoc* Datei in HTML, PDF oder andere Formate konvertiert werden. Listing 5 zeigt beispielsweise eine Konvertierung in HTML mit einer generierten Inhaltsangabe auf der linken Seite und automatischer Kapitelnummerierung.

Listing 5

String asciiDocAsHtml = Asciidoctor.Factory.create().convertFile(Paths.get("build/asciidoc/index.adoc").toFile(), OptionsBuilder.options().backend("html5").headerFooter(true).safe(SafeMode.UNSAFE).attributes(AttributesBuilder.attributes().tableOfContents(true).tableOfContents(Placement.LEFT).sectionNumbers(true)));

Ende

Für Swagger2Markup und Asciidoctorj gibt es aber auch Gradle oder Maven Plugins, um die Konvertierung in einer Build-Pipeline zu automatisieren. Auf GitHub gibt es zwei Template-Projekte, die den Start mit Gradle oder Maven vereinfachen sollen.

Die generierte HTML oder PDF-Dokumentation sieht dann wie folgt aus:

asciidoc\_html.PNG

Abb. 3: AsciiDoc HTML Dokumentation

Der Fokus der generierten Inhaltsangabe (ToC) liegt auf der Funktionalität, die von der API angeboten wird, und nicht auf den konkreten API Endpunkten.Swagger2Markup hat viele Konfigurationsparameter und kann über verschiedene Wege konfiguriert werden, u.a. System Properties, Properties-Datei, Fluent-API oder eine Map. Listing 6 zeigt beispielsweise die Verwendung der Fluent-API, um das Ausgabeformat und die Ausgabesprache zu ändern. Derzeit werden als Sprachen Englisch, Deutsch, Französisch, Russisch und Türkisch unterstützt.

Listing 6

Swagger2MarkupConfig config = new Swagger2MarkupConfigBuilder() .withMarkupLanguage(MarkupLanguage.MARKDOWN) .withOutputLanguage(Language.DE) .withPathsGroupedBy(GroupBy.TAGS) .build();

Swagger2MarkupConverter converter = Swagger2MarkupConverter.from(localSwaggerFile) .withConfig(config) .build();

Ende

Swagger2Markup stellt auch ein Service Provider Interface (SPI) und Extension Points bereit, mit der eigene Extensions entwickelt werden können und somit der Funktionsumfang von Swagger2Markup erweitert werden kann.

Mit Hilfe von Gradle (oder auch Maven) lässt sich eine lokale Swagger Spezifikation, die in dem Ordner *src/docs/swagger* liegt automatisiert konvertieren. Hierfür müssen das Swagger2Markup und das Asciidoctorj Gradle Plugin geladen und die Tasks *convertSwagger2markup* und *asciidoctor* konfiguriert werden. Siehe Listing 7. Standardmäßig sucht das Asciidoctorj Plugin in dem Ordner *src/docs/asciidoc* nach der *index.adoc* Datei. Mit dem Kommando *gradlew asciidoctor* kann nun eine HTML und PDF Dokumentation generiert werden. Die HTML und PDF Dokumentation wird in die Ordner *build/asciidoc/html5* and *build/asciidoc/pdf* generiert.

Listing 7

buildscript { dependencies { classpath 'org.asciidoctor:asciidoctor-gradle-plugin:1.5.3' classpath 'io.github.swagger2markup:swagger2markup-gradle-plugin:1.0.1' }}

apply plugin: 'org.asciidoctor.convert'apply plugin: 'io.github.swagger2markup'

ext { asciiDocOutputDir = file("${buildDir}/asciidoc")}

convertSwagger2markup { swaggerInput file("src/docs/swagger/swagger\_petstore.yaml").getAbsolutePath() outputDir asciiDocOutputDir config = ['swagger2markup.markupLanguage' : 'ASCIIDOC', 'swagger2markup.pathsGroupedBy' : 'TAGS']}

asciidoctor { dependsOn convertSwagger2markup sources { include 'index.adoc' } backends = ['html5', 'pdf'] attributes = [ generated: asciiDocOutputDir ]}

Ende

**Fazit**

Teil 1 dieses Artikels hat gezeigt, dass man bei einem Contract-First Ansatz mit Hilfe von Swagger, AsciiDoc und Swagger2Markup sehr einfach eine Dokumentation für eine Restful API erstellen kann, die Informationen aus der Swagger Spezifikation als auch handgeschriebene Informationen beinhaltet. Ein Schwachpunkt von Swagger ist allerdings, dass HATEOS derzeit noch nicht unterstützt wird. Teil 2 des Artikels wird zeigen wie man mit einem Code-First Ansatz eine Dokumentation aus einer Spring Boot Anwendung mit Hilfe von Springfox und Swagger2Markup generieren kann.

Robert Winkler arbeitet als Software Architekt und Lead Developer bei der Deutsche Telekom AG – Group Innovation in Darmstadt. Seine Schwerpunkte liegen auf den Themen Softewarearchitektur, Continuous Delivery und Entwicklung von Backend-Systemen. Er ist der Ersteller von Swagger2Markup.

Links & Literatur

[1] https://openapis.org/

[2] http://raml.org/

[3] <https://apiblueprint.org/>

[4] <http://asciidoctor.org/docs/asciidoc-writers-guide/>

[5] <https://github.com/Swagger2Markup>

[6] https://github.com/asciidoctor/asciidoctorj