How to Generate a REST CXF3 Application from a Swagger-Contract

Johannes Fiala, Developer

THE LINUX FOUNDATION

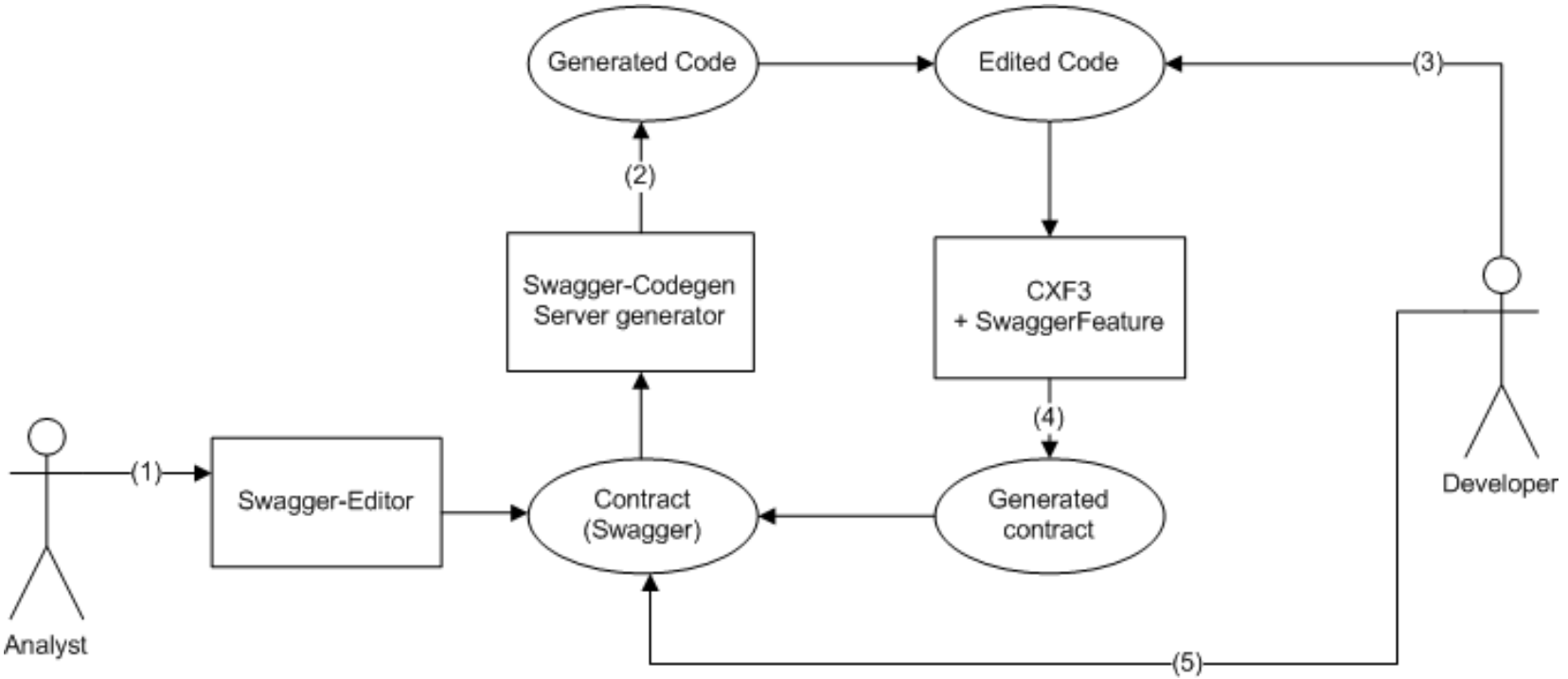
Agenda

- Generate based on contract
- Extend using code first
- Freeze the contract
- Use the REST API
  - Generate client code (Java/Javascript)
  - Access with a browser using a UI
  - View/Share as HTML/PDF
- Migrate between frameworks using the code generator only
- Customize the code generator

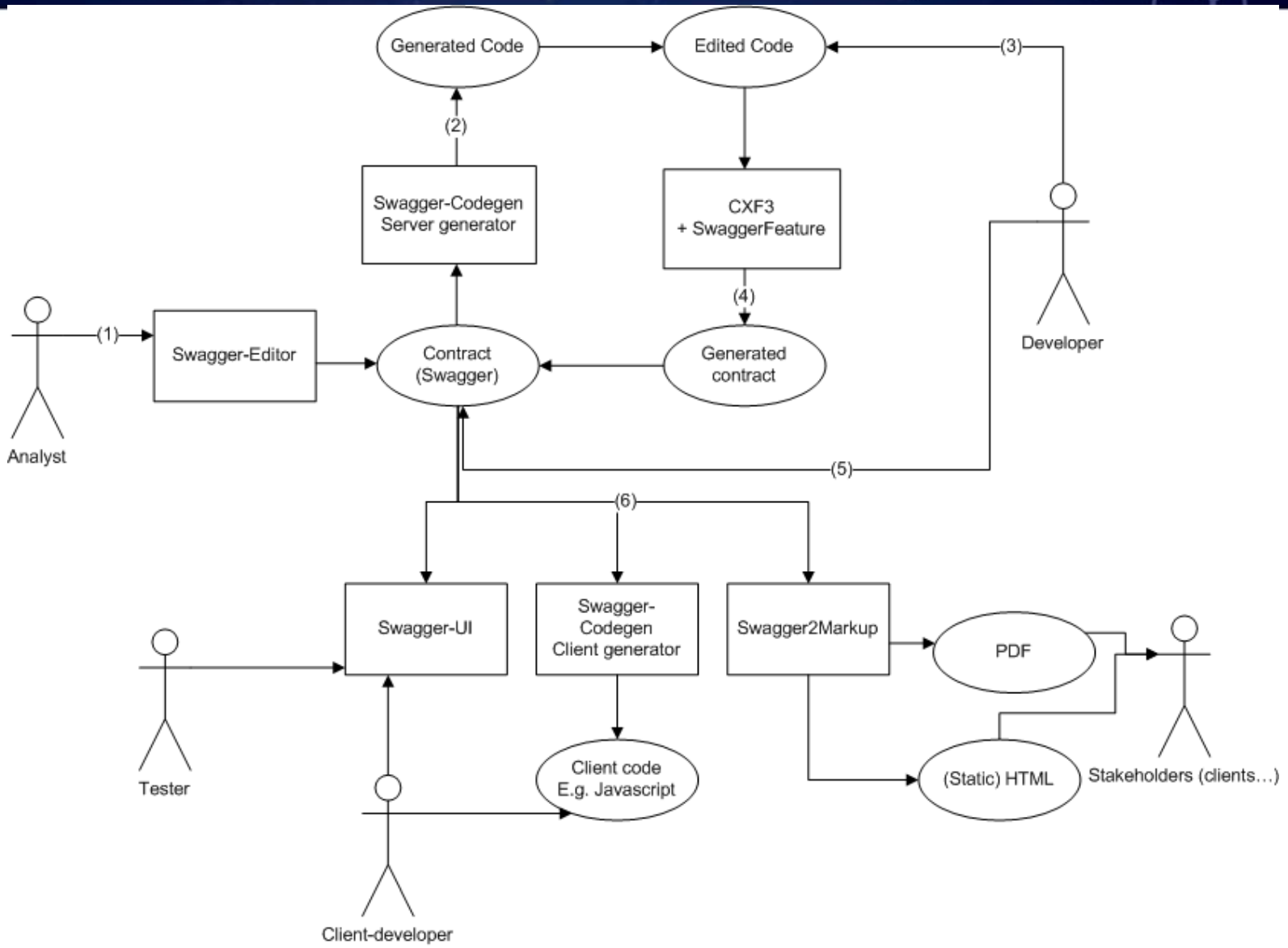
Toolchain

- Apache CXF 3
  - + SwaggerFeature
  - + Spring configuration
  - + Spring Boot integration (start/tests)
- Swagger-Tools
  - Swagger-Editor
  - Swagger-Codegen
  - Swagger-UI
  - Swagger2Markup

Contract first, then code, then contract



Complete process flow



About me…

- Spring REST / Swagger-Springfox
  - Added BeanValidation support
- Swagger-Codegen
  - Created Javascript client
  - Add BeanValidation support for Java
  - Improved CXF server (generate complete server)
  - Created CXF client
- Swagger2Markup
  - Added BeanValidation support

Contract

- WADL (XML-based)
  - By w3c, Last update 2009
- Swagger (Json/Yaml-based)
  - By Linux foundation
  - Version 1.0 – 2011 (Wordnik)
  - Version 1.2 - 2014
  - Version 2.0 – 2015 / transferred to Linux foundation / Open-API initiative
  - Next version 3.0
- Others: Blueprint, RAML, …

Open API / Swagger

- A language-agnostic interface to REST APIs
- allows to discover and understand the capabilities of a service
- Supported Formats: JSON/YAML

https://github.com/OAI/OpenAPI-Specification

Contract editors

- Swagger Editor
  - by SmartBear
- Eclipse SwagEdit
  - By RepreZen API Studio

- Commercial Tools:
  - Restlet Studio
  - RepreZen API Studio

Swagger-Editor (Json)

File ▾ Preferences ▾ Generate Server ▾ Generate Client ▾ Help ▾ ✔ All changes saved

```
{
   "swagger": "2.0",
   "info": {
      "version": "1.0",
      "title": "Hello world"
   },
   "paths": {
      "/hello": {
         "get": {
            "operationId": "sayHello",
            "responses": {
              "200": {
                "description": "OK",
                 "schema": {
                  $ref: "#/definitions/name"
                }
              }
            },
            "parameters": [
               {
                  "name": "name",
                  "in": "query",
                  "required": true,
                  "type": "string",
                  "minLength": 0,
                  "maxLength": 255
               }
            ]
         }
      }
   },
   "definitions": {
      "name": {
        "type": "string",
        "minLength": 0,
        "maxLength": 100
      }
   }
}
```

Hello world

Version 1.0

Paths

/hello

GET /hello

Parameters

Name	Located in	Required	Schema
name	query	Yes	⇄ string

Responses

Code	Description	Schema
200	OK	⇄ ▸name string

Try this operation

Models

name

```
▾Object
  type: "string"
⇄ minLength: 0
  maxLength: 100
  title: "name"
```

Swagger-Editor

- By SmartBear
- Javascript application
  - Run locally using npm
- Edit Json / Yaml format
- Generate server
- Generate client
- Import using URL / copy/paste

https://github.com/swagger-api/swagger-editor

Generate the server stub Swagger-Codegen

- by SmartBear (Apache License)
- Version 2.2.0
- Java program
  - Mustache templating
- Generates server + client code
- 32 Supported languages: Java, C#, Javascript, Dart, Groovy, JaxRS, NodeJS, Objective-C, Perl, PHP, Python, Ruby, Scala, …

https://github.com/swagger-api/swagger-codegen

Generate the server stub Swagger-Codegen

- Supported server stubs for Java:
  - Java JAX RS
    - 2.0 spec
    - Jersey
    - CXF
    - Resteasy
  - Spring MVC
  - Spring Boot

CXF server stub Features

- Scheduled for version 2.2.2
- Generates Interface/Implementation/Models
- Generate a complete web application
  - Context.xml / ApplicationContext.xml
  - Web.xml
  - Jboss: jboss-web.xml
- Spring Boot support
  - Run as Spring-Boot Application
  - Run using Spring Integrationtests using random ports

Swagger-Codegen CLI

- io.swagger.codegen.Codegen
  -i hello_world.json
  -l jaxrs-cxf
  -o c:\hello_world_project
  -c hello_world_config.json

Swagger-Codegen CLI Configuration file

swagger_config_spring_application.json

```
{
"useBeanValidation": "true",
"generateSpringApplication": "true",
"useSwaggerFeature": "true",
"generateSpringBootApplication": "true"
"generateApiTests": "true",
}
```

Demo

- Create hello world service
- Generate CXF server stub (with Spring support enabled)
- Run using Spring Boot
- Run Junit-Tests
- Deploy to application server (Jboss EAP 7)

Spring Boot Hot-Deploy

- Allows hot deploy
  - for adding new methods etc.
- Add JVM parameters:
  - -javaagent:/path_to_spring-loaded/springloaded-1.2.5.RELEASE.jar –noverify
- Used by Grails

- More Info: https://github.com/spring-projects/spring-loaded

CXF server stub Supported Features

- Spring application
  - Swagger generator
  - WADL generator
  - BeanValidation annotations
  - Activate automatic BeanValidation (1.1)
  - Compression (gzip)
  - Logging
  - Integration-Tests (Spring Boot)

Swagger-Codegen CLI
Display all language options

- io.swagger.codegen.SwaggerCodegen config-help -l jaxrs-cxf
  or
- java -jar modules/swagger-codegen-cli/target/swagger-codegen-cli.jar config-help –l jaxrs-cxf

Swagger-Codegen CLI language options for cxf

```
title
    a title describing the application

useBeanValidation
    Use BeanValidation API annotations (Default: false)

generateSpringApplication
    Generate Spring application (Default: false)

useSwaggerFeature
    Use Swagger Feature (Default: false)

useWadlFeature
    Use WADL Feature (Default: false)

useMultipartFeature
    Use Multipart Feature (Default: false)

useGzipFeature
    Use Gzip Feature (Default: false)

useBeanValidationFeature
    Use BeanValidation Feature (Default: false)

useLoggingFeature
    Use Logging Feature (Default: false)

generateSpringBootApplication
    Generate Spring Boot application (Default: false)
```

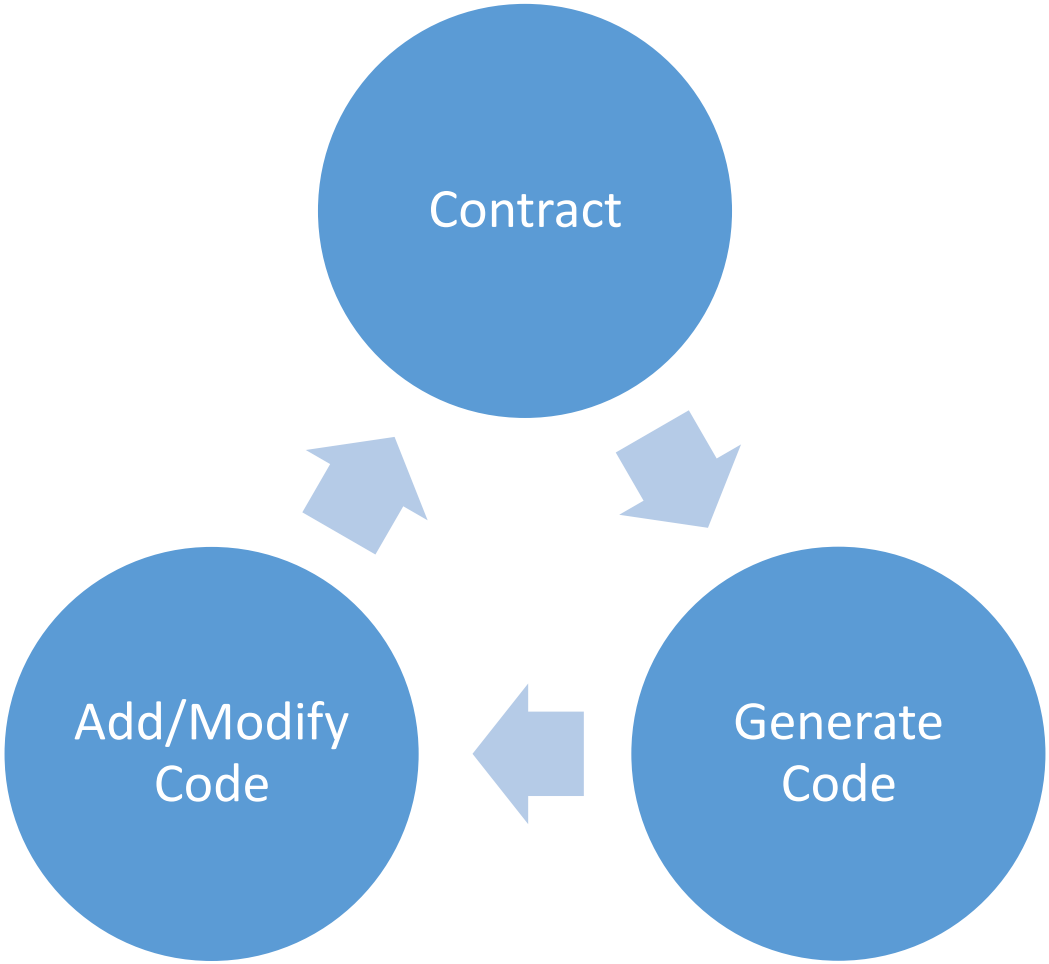
Demo

- Re-Generate with more options
  - BeanValidation
  - Gzip
  - Logging

Further development life cycle

- Extend the API
  - code first
- Use the API
  - Frontend development
- Finalize: Freeze the contract

Further development life cycle



Extend the application

- Modify your API:
  - Add new services (use JAXRS-annotations)
  - Use Swagger annotations
  - Use BeanValidation annotations

- Generated Swagger spec gets updated automatically

Extend the application Swagger annotations

- Service:
  - @Api – activate Swagger for api
- Operations:
  - @ApiOperation - description
  - @ApiResponse – error codes + return types
- Model:
  - @ApiModel - description
  - @ApiModelProperty - description

Freeze your API: contract

- Use the updated contract
- Generate interfaces/models
- Prevent accidential changes of the API
- Integrate in build job (Maven / Gradle)

- https://github.com/swagger-api/swagger-codegen/tree/master/modules/swagger-codegen-maven-plugin

Freeze your API: contract


```
<!-- re-generate interface/model -->
<plugin>
      <groupId>io.swagger</groupId>
      <artifactId>swagger-codegen-maven-plugin</artifactId>
      <version>2.2.1</version>
      <executions>
          <execution>
              <goals>
                  <goal>generate</goal>
              </goals>
              <configuration>
                  <inputSpec>${project.basedir}/src/main/resources/api/hello_world.json</inputSpec>
                  <language>jaxrs-cxf</language>
                  <configOptions>
                      <sourceFolder>src/gen/java</sourceFolder>
                      <useBeanValidation>true</useBeanValidation>
                      <generateSpringApplication>true</generateSpringApplication>
                      <generateSpringBootApplication>true</generateSpringBootApplication>
                      <useWadlFeature>true</useWadlFeature>
                      <useSwaggerFeature>true</useSwaggerFeature>
                  </configOptions>
                  <addCompileSourceRoot>false</addCompileSourceRoot>
                  <output>${project.basedir}</output>
              </configuration>
          </execution>
      </executions>
      <dependencies>
          <dependency>
              <groupId>io.swagger</groupId>
              <artifactId>swagger-codegen</artifactId>
              <version>2.2.2-SNAPSHOT</version>
          </dependency>
      </dependencies>
  </plugin>
```

Demo

- Extend hello world service (+ BeanValidation)
- Access updated specs
  - Swagger spec
  - WADL
- Freeze the contract

Use your API

- Generate client stubs
  - Swagger-Codegen
- Access your API using a browser
  - Swagger-UI
- Generate HTML/PDF documentation
  - Swagger2Markup

Swagger-Codegen

- Server + Client code stub generator
- Integration options
  - Java application
  - Maven
  - Gradle

- https://github.com/swagger-api/swagger-codegen

Why generate client code?

- No more manual api calls
- Ensure consistency of your client code with the API!
- Makes code completion possible!
- Allows developers to read description for your operations and models in the IDE
- You get compilation errors if the API breaks with newer versions!

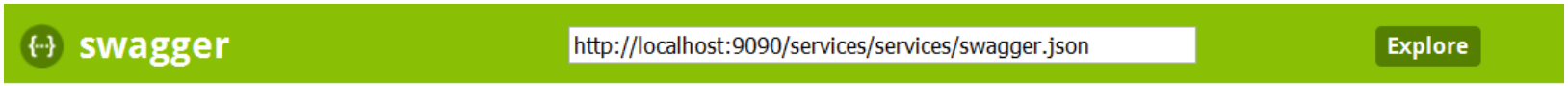
Swagger-UI

- By SmartBear
- Access your API with a browser
- Javascript application
- Can access the generated Swagger spec – always consistent with your code

- Integration options:
  - Copy into your webapp
  - load as Web-Jar

https://github.com/swagger-api/swagger-ui

Swagger-UI



Sample REST Application

The Application

Created by users@cxf.apache.org
Apache 2.0 License

hello

Show/Hide | List Operations | Expand Operations

GET /hello

Response Class (Status 200)
string

Response Content Type application/json

Parameters

Parameter	Value	Description	Parameter Type	Data Type
name	(required)		query	string

Try it out!

[BASE URL: /services/services , API VERSION: 1.0.0]

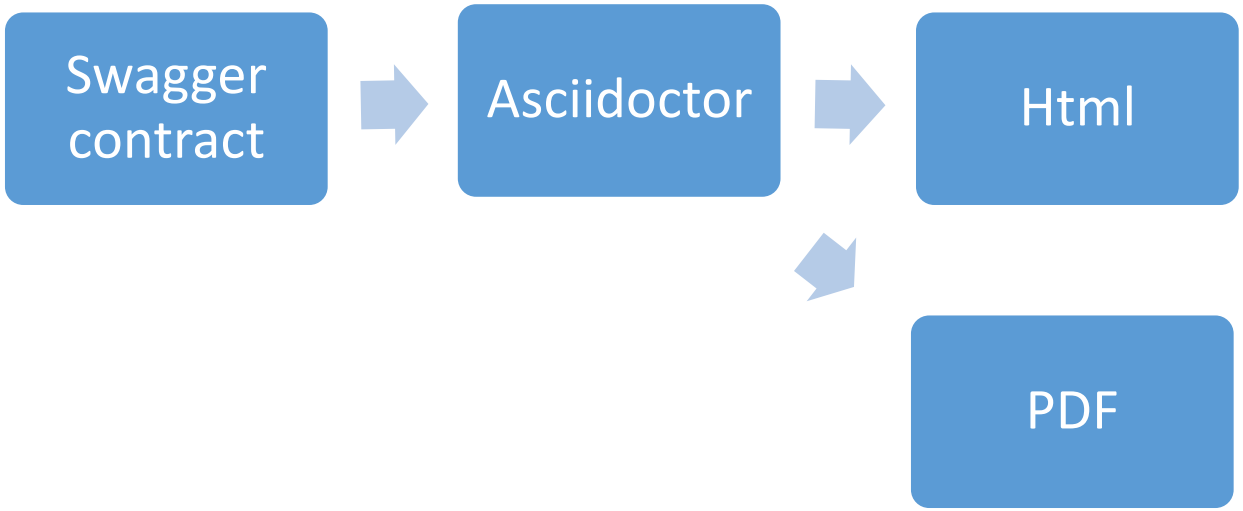
Swagger2Markup

- By Robert Winkler (github)
- Render your API in HTML/PDFs
- Uses Markdown/Asciidoctor files
- Completely customizable

- Integration options:
  - Run as Program/Unittest
  - Maven

https://github.com/Swagger2Markup/swagger2markup

Swagger2Markup



Demo

- Client stub generator:
  - Java
  - Javascript
- Swagger-UI
- Swagger2Markup

Migration – Ensure portability

- Keep method signatures clean
  - Use JaxRS 2 Exceptions
  - Use JaxRS Response only if necessary
- Design implementation classes independently from generated interfaces
  - Do not adhere to generated interfaces strictly, use composition
- Keep framework-specific configuration separately

REST framework alternatives

- Apache CXF
- Jersey
- Jboss RestEasy
- Spring Rest (MVC)

Spring REST (MVC)

- Spring MVC
  - Enable Swagger: Springfox
  - No WADL support
  - No Gzip support
- Swagger-Codegen server stub
  - language: "spring"

- More info: Devoxx BE 2015 "Generate client stubs & document your REST-API using Swagger & Spring"

Demo

- Migrate from Spring REST to CXF
  - only by re-generating using Swagger-Codegen

Customize the generator

- Generator implemented in Java (one class for each language)
- Mustache-files
  - api.mustache
  - apiServiceImpl.mustache
  - pojo.mustache
  - api_test.mustache
  - …
- Jmustache:
  - https://github.com/samskivert/jmustache

Customize the generator

- Use –t flag for your own templates
- Customize only the templates you need
- Examples:
  - Add Maven profile for deployment
  - Add logger declaration
  - Customize generated unit tests
  - …

Jmustache primer / 1

- Print variable: {{mediaType}}
  - Will print as HTML entities by default: <em>…
- Print variable unescaped: {{{mediaType}}}
  - Print "as is" - …
- Section (if): {{#useBeanValidation}}…{{/useBeanValidation}}
- Inverted section (if not): {{^useBeanValidation}}…{{#useBeanValidation}}

Jmustache primer / 2

- Lists:
  {{#repo}} {{name}} {{/repo}}
- Partials (import other template):
  {{>beanValidation}}
- Comments
  {{! My comment }}

- Newline trimming (Jmustache):
  {{#required}}
- @NotNull
- {{/required}}

Customize the generator

- Customize Codegen Languages
  - Extend Language class
  - Add it to io.swagger.codegen.CodegenConfig
    - swagger-codegen\src\main\resources\META-INF\services\io.swagger.codegen.CodegenConfig
  - Copy language templates
  - Also see "Building your own Templates"
    - https://github.com/swagger-api/swagger-codegen/wiki/Building-your-own-Templates

Demo

- Customize the generator
  - E.g. custom pom.xml file for deployment
- Swagger-Codegen project structure

WADL ⇔ Swagger

- Use wadl2java to generate server stub
  - BeanValidation: krasa-jaxb-tools
- Activate CXF3 SwaggerFeature
- Use generated Swagger-file
  - Will include BeanValidation annotations for models

- You can use both WADL first + auto-generate Swagger contract

CXF Caveats

- Impl: return null = no response in browser
- No JaxRS annotations in Impl. Class
- No @Consumes/@Produces Json will fallback to default XML
- Jettison on classpath will overrule all other providers!

Wrapup

- Generate based on contract
  - Swagger-Codegen server stubs
- Extend using code first
  - CXF 3 Swagger Feature
- Freeze using contract
  - Swagger-Codegen build integration (mvn/gradle/cmd)
- Use your application
  - Generate client code (Swagger-Codegen)
  - Use in browser (Swagger-UI)
  - View/Share as HTML/PDF (Swagger2Markup)
- Migrate between frameworks using the code generator only
- Customize the code generator

Contribute to the projects

- Swagger-Codegen
  - Java / JMustache
- Swagger-UI
  - Javascript
- Swagger-Editor
  - Javascript
- Swagger2Markup
  - Java/Asciidoctor

Links & Resources

- Swagger Editor
  - http://editor.swagger.io/
- Swagger Codegen
  - https://github.com/swagger-api/swagger-codegen
- Swagger UI
  - https://github.com/swagger-api/swagger-ui
- CXF
  - http://cxf.apache.org/

Thank you for your attention!

- Demo-Code:
  http://github.com/jfiala/swagger-cxf-demo

- Contact:
  - @johannes_fiala
  - johannes.fiala@fwd.at