



# OpenAPIs are everywhere

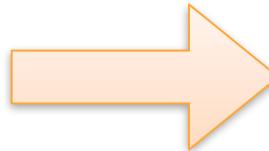
EclipseCon 2018 - Jérémie Bresson - 2018-10-23

# OpenAPI



# Swagger vs OpenAPI: it is the same!

Swagger



OpenAPI



# Jérémie Bresson

Java developer



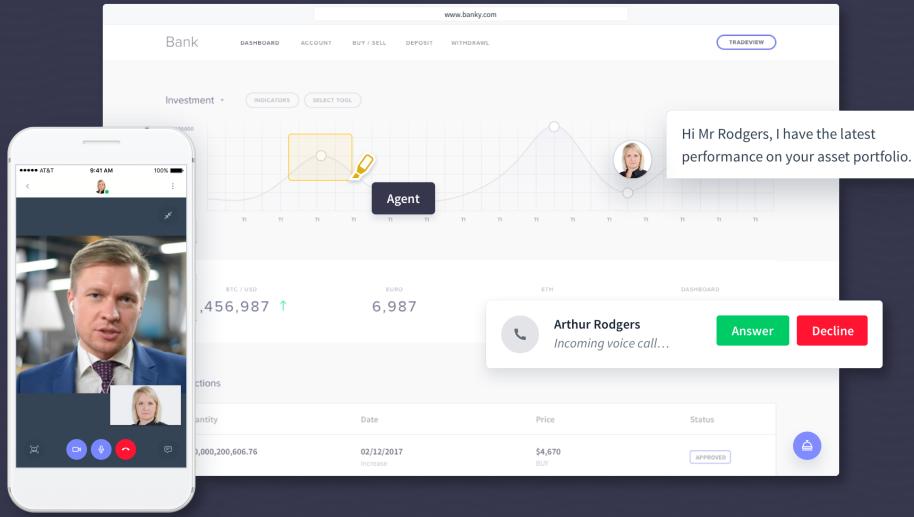
@j2r2b



jmini

Open-source contributor





# Conversational Banking made real

Switzerland | Germany | United Kingdom | Czech Republic | Bulgaria | US

[info@unblu.com](mailto:info@unblu.com)

# OpenAPIs are everywhere



# OpenAPIs are everywhere



Jens Reimann

## Apache Camel Java DSL in combination Eclipse Kura Wires

by Jens Reimann at September 19, 2018 08:30 AM

In part #1 and part #2, we saw how easy it is to interface Apache Camel with Kura Wires. Simply by re-using some existing functionality. A few lines of XML, Groovy and you can already build an IoT solution based on the Camel ecosystem and the Eclipse Kura runtime. This part will focus on the Java DSL of Apache Camel.

It will also take into account, that when you develop and deploy an application, you need some kind of development, test and integration environment. When you build something, no matter how big, based on Camel or Kura Wires, you do want to test it. You want to have unit tests, and the capability to automatically test if your solution works, or still works after you made changes.

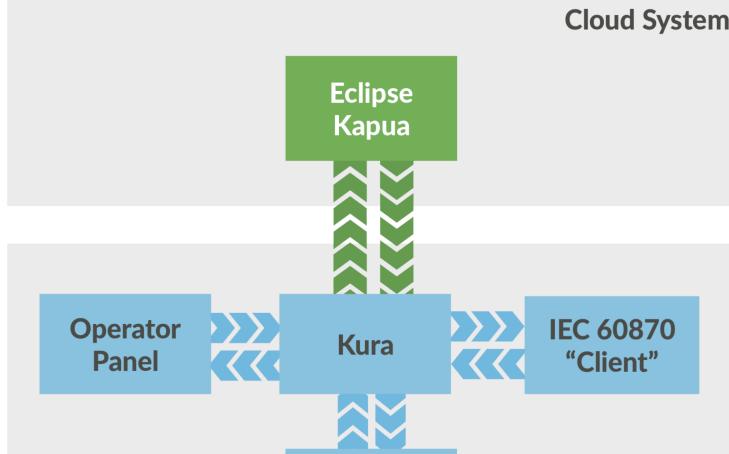
Using Kura Wires alone, this can be a problem. But Camel offers you a way to easily run your solution in a local IDE, debugging the whole process. You can have extra support for debugging Camel specific constructs like routes and endpoints. Camel has support for JUnit and e.g. using the "seda" endpoints, you can create an abstraction layer between Camel and Wires.

### The goal

I'll make this one up (and yes, let's try to keep it realistic). We have a device, and his device allows to set two parameters for its operation (P1 and P2, both floating points). Now we already have the device connection set up in Kura. Maybe using Modbus, or something else. Kura can talk to it using Kura Wires and that is all that counts.

Now we do get two additional requirements. There is some kind of operating panel next to the device, which should allow viewing and setting those parameters locally. Also, those parameters should be accessible, using IEC 60870-5-104, for an additional device, right next to the Kura gateway.

All of those operations have to be local only, and still work when no connection to the cloud is possible. But of course, we don't want to lose the ability to monitor the parameters from our cloud system.



# OpenAPIs are everywhere

additional you will need to install the following dependencies:

- <https://repo1.maven.org/maven2/de/dentrassi/kura/addons/de.dentrassi.kura.addons.camel.iec60870/0.6.1/de.dentrassi.kura.addons.camel.iec60870-0.6.1.dp>
- <https://repo1.maven.org/maven2/de/dentrassi/kura/addons/de.dentrassi.kura.addons.camel.jetty/0.6.1/de.dentrassi.kura.addons.camel.jetty-0.6.1.dp>
- <https://repo1.maven.org/maven2/de/dentrassi/kura/addons/de.dentrassi.kura.addons.camel.swagger/0.6.1/de.dentrassi.kura.addons.camel.swagger-0.6.1.dp>

This will install the support for REST APIs, backed by Jetty. As Kura already contains Jetty, it only makes sense to re-use those existing components.

Once the component is deployed and started, you can navigate your web browser to <http://:8090/api>. This should bring up the Swagger UI, showing the API of the routes:

The screenshot shows the Swagger UI interface for the `/parameters/current` endpoint. The top navigation bar indicates the endpoint is for local interfacing. Below the navigation, there are two main sections: `GET /parameters/current` and `PUT /parameters/current`.

**GET /parameters/current**: Described as "Get the current parameters". It includes a "Try it out" button and a table for parameters. A red note says "body \* required" and describes the parameter as "The new parameters to set. This may contain null values, which are then filled with the currently active parameters". An example value is provided as a JSON object:

```
{  
    "setpoint1": 0,  
    "setpoint2": 0  
}
```

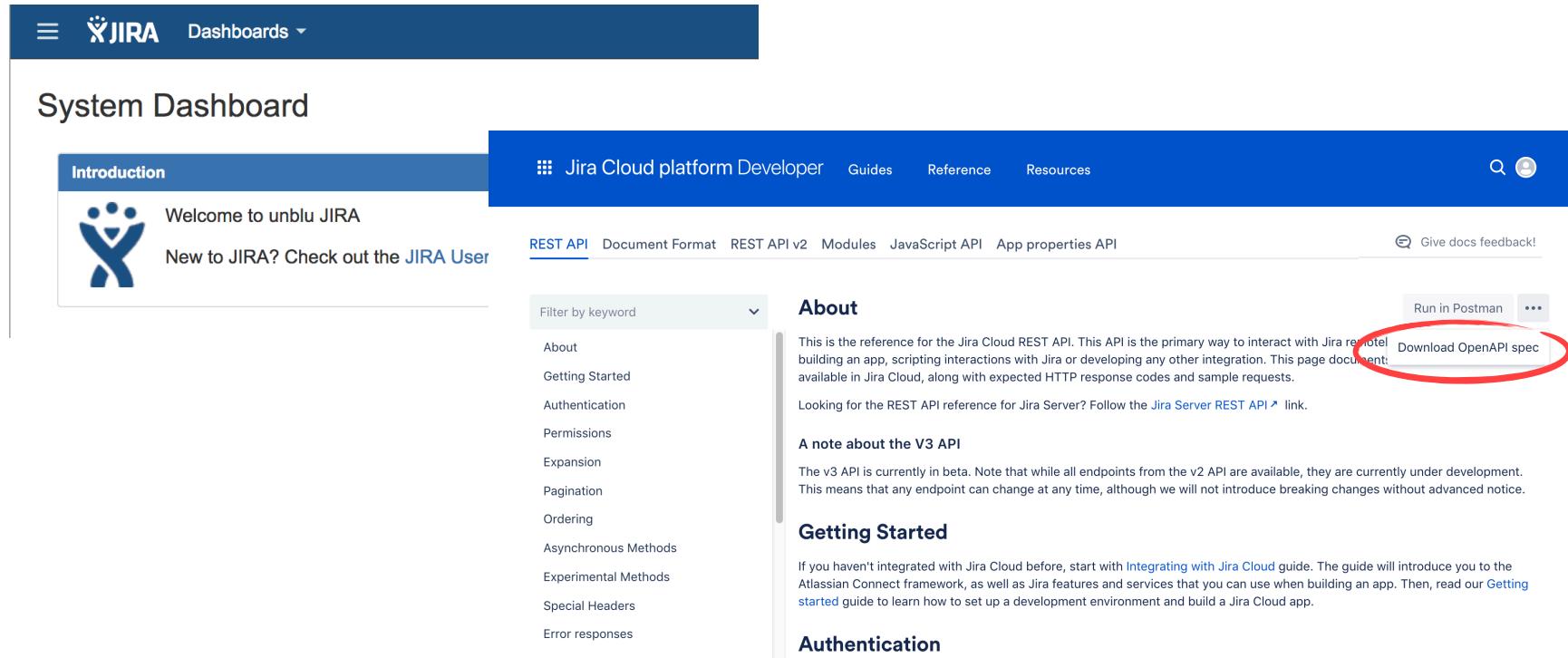
**PUT /parameters/current**: Described as "The newly apply parameters, merged with the current parameters". It includes a "Try it out" button and a table for parameters. A red note says "body \* required" and describes the parameter as "The new parameters to set. This may contain null values, which are then filled with the currently active parameters". An example value is provided as a JSON object:

```
{  
    "setpoint1": 0,  
    "setpoint2": 0  
}
```

**Responses**: Shows a 200 response with the description "Applied new parameters". An example value is provided as a JSON object:

```
{  
    "setpoint1": 0,  
    "setpoint2": 0  
}
```

# OpenAPIs are everywhere



The screenshot shows the Jira Cloud platform Developer documentation. At the top, there's a navigation bar with the Jira logo and a 'Dashboards' dropdown. Below it is the 'System Dashboard'. On the left, there's an 'Introduction' sidebar with a logo of three stylized human figures and text welcoming users to unblu JIRA and linking to the JIRA User guide.

The main content area has a blue header with the title 'Jira Cloud platform Developer' and tabs for 'Guides', 'Reference', and 'Resources'. Below the header, there's a search bar and a 'Give docs feedback!' link. The 'REST API' tab is selected, showing a sub-navigation bar with 'Document Format', 'REST API v2', 'Modules', 'JavaScript API', and 'App properties API'.

The 'About' section contains a brief introduction to the REST API, mentioning it's the primary way to interact with Jira. It includes a note about the V3 API being beta and links to the 'Getting Started' and 'Authentication' guides. On the right side of the 'About' section, there are buttons for 'Run in Postman' and 'Download OpenAPI spec', with the latter being the one highlighted by a red circle.

# OpenAPIs are everywhere

The screenshot shows the OpenShift Container Platform interface. On the left, a sidebar navigation bar includes links for Overview, Applications, Builds, Resources, Storage, Monitoring, and Catalog. The main area shows the 'unblu-testing' namespace. Under 'APPLICATIONS', there is a single application named 'unblu'. Under 'DEPLOYMENTS', three deployments are listed: 'collaboration-server', 'haproxy', and 'mariadb'. A modal window is open over the deployment list, displaying the contents of the URL <https://raw.githubusercontent.com/openshift/origin/master/api/swagger-spec/openshift-openapi-spec.json>. The modal content is a large JSON document describing the OpenShift API's schema and behavior.

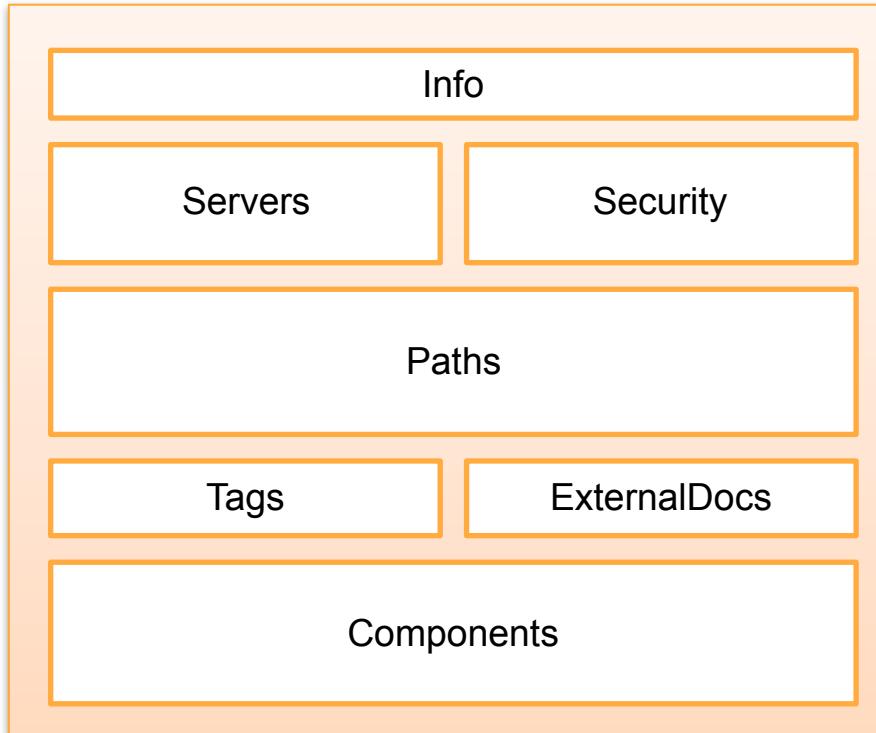
```
{
  "swagger": "2.0",
  "info": {
    "description": "OpenShift provides builds, application lifecycle, image content management, and administrative policy on top of Kubernetes. The API allows consistent management of those objects. All API operations are authenticated via an Authorization bearer token that is provided for service accounts as a generated secret (in JWT form) or via the native OAuth2 mechanism. The API uses standard Kubernetes resource types and follows the standard REST conventions. Most objects return a 'resourceVersion' string that represents the version of the object in the underlying storage. The standard LIST operation performs a snapshot read of the underlying objects, returning all observed by a client. By listing and beginning a watch from the returned resourceVersion, clients may observe a consistent view of the state of one or more objects. Note that WATCH always returns the update after the provided resourceVersion. Watch may be extended a limited time in the past, using netcd 2 the watch window is 100ms events (which on a large cluster may take a few seconds). All objects will be explicitly handled by the client, reflecting which ones are added to the cluster. Objects with a lifecycle have their state reflect the state of the cluster, and those that have no state (no objects with lifecycle typically have three main sections):\n\n  * 'metadata' common to all objects\n  * 'spec' that represents the desired state\n  * 'status' that represents how much of the desired state is reflected on the cluster at the current time\n\nObjects that have no state have 'metadata' but may lack a 'spec' or 'status' section.\n\nNo objects are divided into those that are namespace scoped (only exist inside/nof a namespace) and those that are cluster scoped (exist outside/nf a namespace). A namespace scoped resource will be deleted when the namespace is deleted and cannot be created if the namespace has not yet been created nor is it the primary namespace for the object type. A schema is a combination of the 'kind' and 'apiVersion' fields. This schema is additive only for any given version - backwards incompatible changes are allowed without incrementing the apiVersion. The server will return and accept a number of standard responses that share a common schema - for instance, the common error type is 'metav1.Status' (described below) and will be returned on any error from the API server.\n\nThe API is available in multiple serialization formats - the default is JSON (Accept: application/json and Content-Type: application/json) but clients may also use YAML (application/yaml) or the native Protobuf (application/vnd.kubernetes.protobuf). Note that the format of the WATCH event is slightly different for JSON it returns newline-delimited objects while for Protobuf it returns length-delimited frames (4 bytes in network-order) that contain a 'versioned.WatchEvent' object. See the OpenShift documentation at https://docs.openshift.org for more information.\n",
  "title": "OpenShift API (with Kubernetes)",
  "license": {
    "name": "Apache 2.0 (ASL2.0)",
    "url": "http://www.apache.org/licenses/LICENSE-2.0"
  },
  "version": "latest",
  "paths": {
    "/api": {
      "get": {
        "description": "get available API versions",
        "consumes": [
          "application/json",
          "application/yaml",
          "application/vnd.kubernetes.protobuf"
        ],
        "produces": [
          "application/json",
          "application/yaml",
          "application/vnd.kubernetes.protobuf"
        ],
        "schemes": [
          "https"
        ],
        "tags": [
          "core"
        ],
        "operationId": "getCoreLegacyAPIVersions",
        "responses": {
          "200": {
            "description": "OK"
          }
        }
      }
    }
  }
}
```

# Content of an OpenAPI Specification



# Content of an OpenAPI Specification

OpenAPI v3



JSON  
or  
YAML

# Content of an OpenAPI Specification

```
1  openapi: 3.0.1
2  info:
3    title: A TODO-Task list application
4    description: A simple application to handle tasks.
5    version: 1.0.0
6    license:
7      name: Apache-2.0
8      url: 'http://www.apache.org/licenses/LICENSE-2.0.html'
9    servers:
10   - url: 'http://localhost:8080/todoapp'
11   tags:
12     - name: task
13     |   description: Task management
14   paths:
15     /Task:
16       get:
17         operationId: getAllTasks
18         description: Get the list of all tasks
19         tags:
20           - task
21         responses:
22           200:
23             |   description: a list of all tasks
```

# Editors for OpenAPI specifications



# Swagger Online Editor

**Swagger Editor**

```

1  openapi: 3.0.1
2  info:
3    title: Unblu
4    description: Unblu Server
5    version: 5.0.0
6    servers:
7      - url: http://localhost:7777/co-unblu/rest/v2
8    security:
9      - basicAuth: []
10   tags:
11     - name: AccountSecrets
12       description: Service to read account secret of current account<br>@since 4.3.0
13     - name: Accounts
14       description: With this service the accounts of the unblu system can be managed.
15         Most of the provided interface needs super admin permissions. Especially if the
16         edited account is not the one of the current user.<br><br> The Account object
17         can be expanded. If the query parameter expand is set to contactAddressId and/or
18         billingAddressId (e.g ?expand=contactAddressId,billingAddressId) the address id's
19         will be automatically resolved and wrapped into the object. This can also be done
20         when sending the object<br>@since 4.3.0
21     - name: Addresses
22       description: With this service, the addresses of accounts can be managed.<br>@since
23         4.3.0
24     - name: ApiKeys
25       description: With this service the api keys can be managed<br>@since 4.3.0
26     - name: CannedResponses
27       description: 'Service to manage canned responses.<br> The canned responses a user
28         can edit and the once he can use for chat can differ. The once he can edit are
29         depending on his role. The ones he can use in a chat depends on the team setup
30         and his role. See the following policies for the list of chat messages available
31         for chat: <ul><li>All canned responses of the own user.</li><li>All canned responses
32         of the team cascade (the users team and all of the parent teams).</li><li>All
33         canned responses of the sub-teams if role is: SUPERVISOR or higher.</li><li>All canned
34         responses of all teams if role is: ADMIN or higher.</li><li>All canned
35         responses of the users account.</li></ul> In comparison to this the creation/editing
36         /deleting
37         of canned responses follow the following policies: <ul><li>All canned responses
38         of the own user.</li><li>All canned responses of the sub-teams if role is: SUPERVISOR
39         or higher.</li><li>All canned responses of all teams if role is: ADMIN or higher.</li>
40         >
41     - name: Contacts
42       description: Service to access the contact data for an account<br>@since 4.3.0
43     - name: Domains
44       description: Service to manage the domains of the account. A domain needs to be
45         defined to use unblu on it.<br>@since 4.3.0
46     - name: NamedAreas

```

**Unblu** 5.0.0 OAS3

Unblu Server

[Authorize](#)

Server

**AccountSecrets** Service to read account secret of current account @since 4.3.0

**GET** /accountsecrets/getCurrentAccountSecret

With this service the accounts of the unblu system can be managed. Most of the provided interface needs super admin permissions. Especially if the edited account is not the one of the current user.

**Accounts** The Account object can be expanded. If the query parameter expand is set to contactAddressId and/or billingAddressId (e.g ?expand=contactAddressId,billingAddressId) the address id's will be automatically resolved and wrapped into the object. This can also be done when sending the object @since 4.3.0

**POST** /accounts/create

**GET** /accounts/delete

<http://editor.swagger.io/>

# Eclipse IDE plugin: KaiZen-OpenAPI-Editor



```
unblu-openapi.yaml
openapi: 3.0.1
info:
  title: Unblu
  description: Unblu Server
  version: 5.0.0
servers:
- url: http://localhost:7777/unblu/rest/v1
security:
- basicAuth: []
tags:
- name: AccountSecrets
  description: Service to read account secret of current account<br>@since 4.3.0
- name: Accounts
  description: With this service the accounts of the unblu system can be managed.
    Most of the provided interface needs super admin permissions. Especially if the
    edited account is not the one of the current user.<br> <br> The Account object
    can be expanded. If the query parameter expand is set to contactAddressId and/or
    billingAddressId (e.g ?expand=contactAddressId,billingAddressId) the address id's
    will be automatically resolved and wrapped into the object. This can also be done
    when sending the object<br>@since 4.3.0
- name: Addresses
  description: With this service, the addresses of accounts can be managed.<br>@since
    4.3.0
- name: ApiKeys
  description: With this service the api keys
- name: CannedResponses
  description: 'Service to manage canned responses. The ones he can use for chat
    depend on his role. The ones he can use in a chat depends on the team setup
    and his role. See the following policies for the list of chat messages available
    for chat: <ul> <li>All canned responses of the own user.</li> <li>All canned responses
    of the team cascade (the users team and all of the parent teams)</li> <li>All
    canned responses of the sub teams if role is SUPERVISOR or higher</li> <li>All'
```

<https://github.com/RepreZen/KaiZen-OpenAPI-Editor>

# Postman client

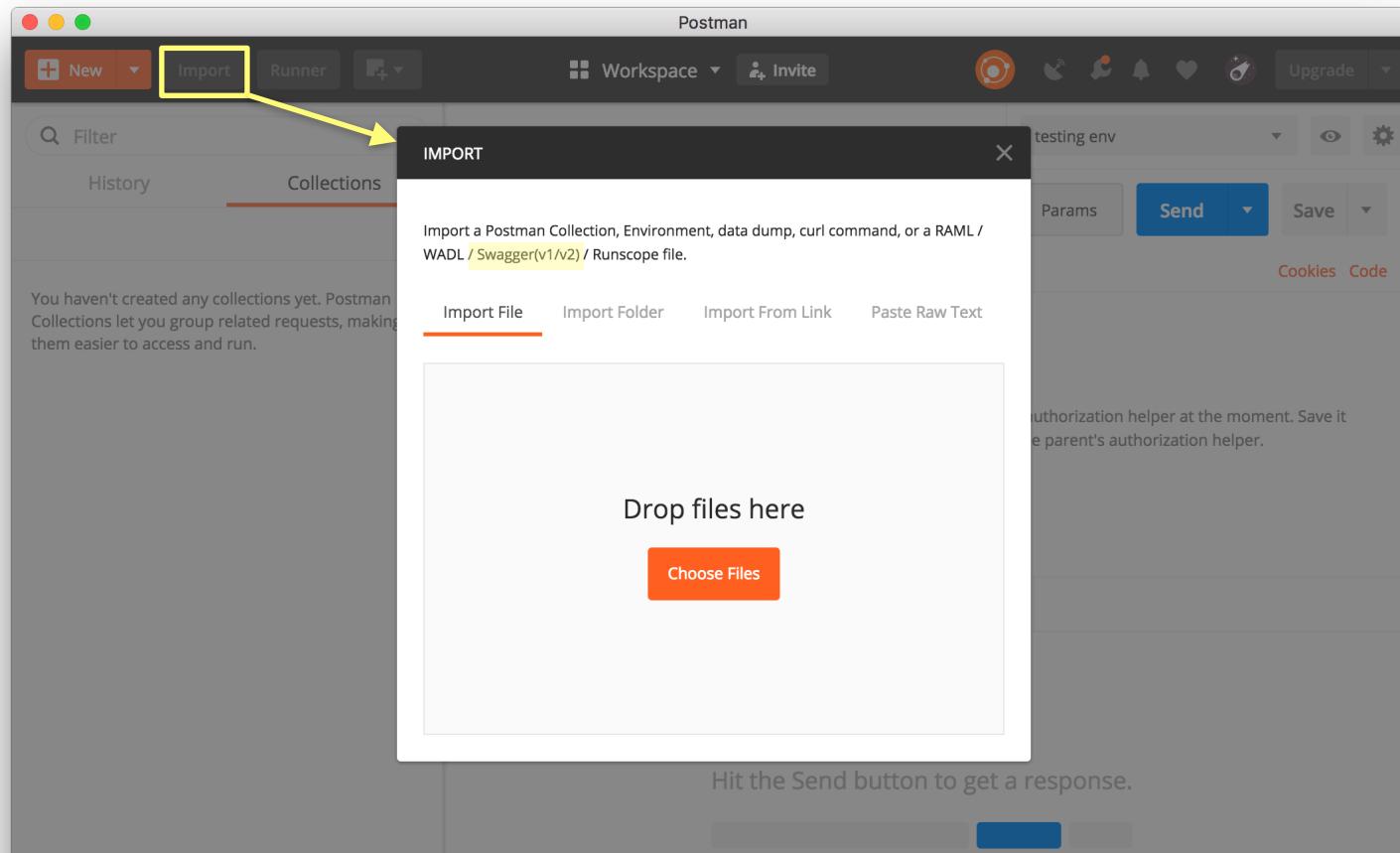


# Postman



The screenshot shows the Postman application interface. The top navigation bar includes 'New', 'Import', 'Runner', 'Workspace' (with a dropdown menu), 'Invite', and 'Upgrade'. The workspace shows a collection named 'getAllTasks' selected. The left sidebar displays a tree structure for a 'A TODO-Task list application' with four requests under the 'task' folder: 'getAllTasks' (GET), 'createTask' (POST), 'updateTask' (PUT), and 'deleteTask' (DEL). The main panel shows the 'getAllTasks' request details: method 'GET', URL 'http://localhost:8080/todoapp/task', and an 'Authorization' tab selected. The authorization type is set to 'Inherit auth from pa...'. A note states that the authorization header will be automatically generated when sending the request, with a link to learn more about authorization. Below the request details is a 'Response' section containing the URL <https://www.getpostman.com/>. At the bottom, a message says 'Hit the Send button to get a response.'

# Postman: import OpenAPI (v2)



# Server-side



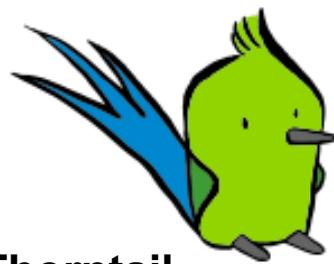
# Support for OpenAPI (Java Server)



VERT.X



# Adding OpenAPI to a JAX-RS project



**Thorntail**

(a.k.a WildFly Swarm)  
<https://thorntail.io/>



<https://openliberty.io/>

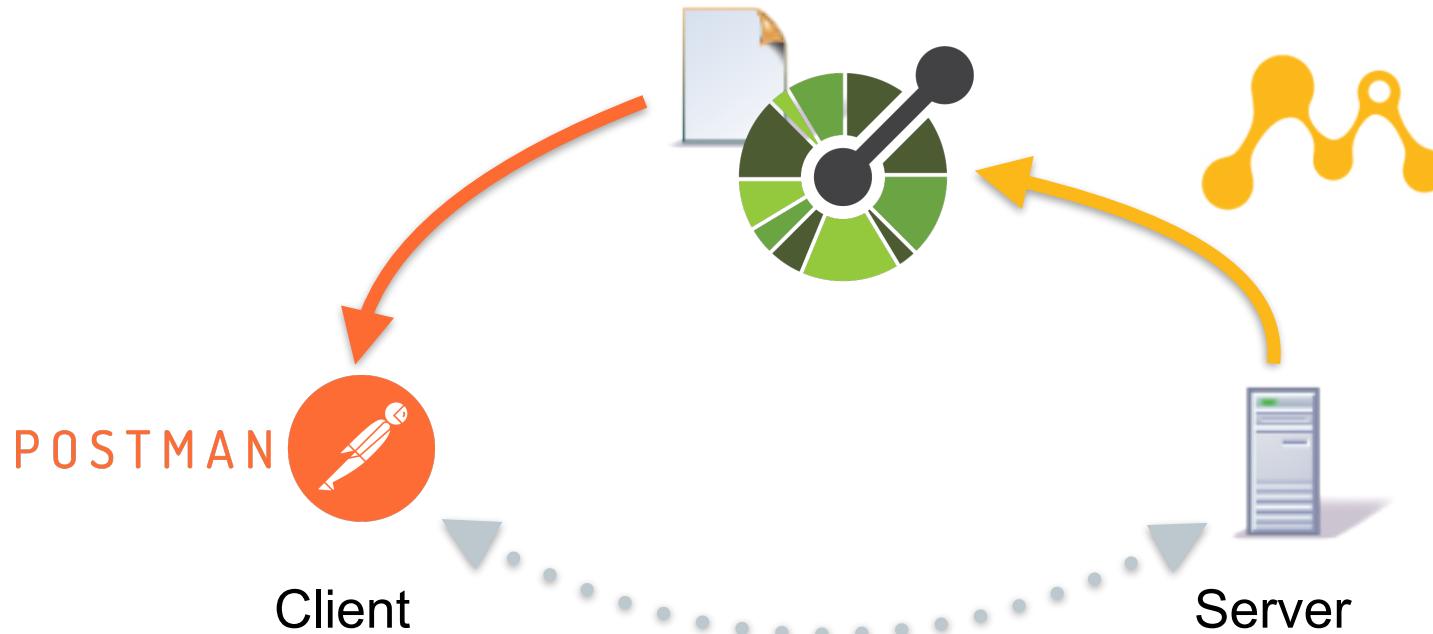
# Adding OpenAPI to a JAX-RS project

- Add the additional maven dependency:

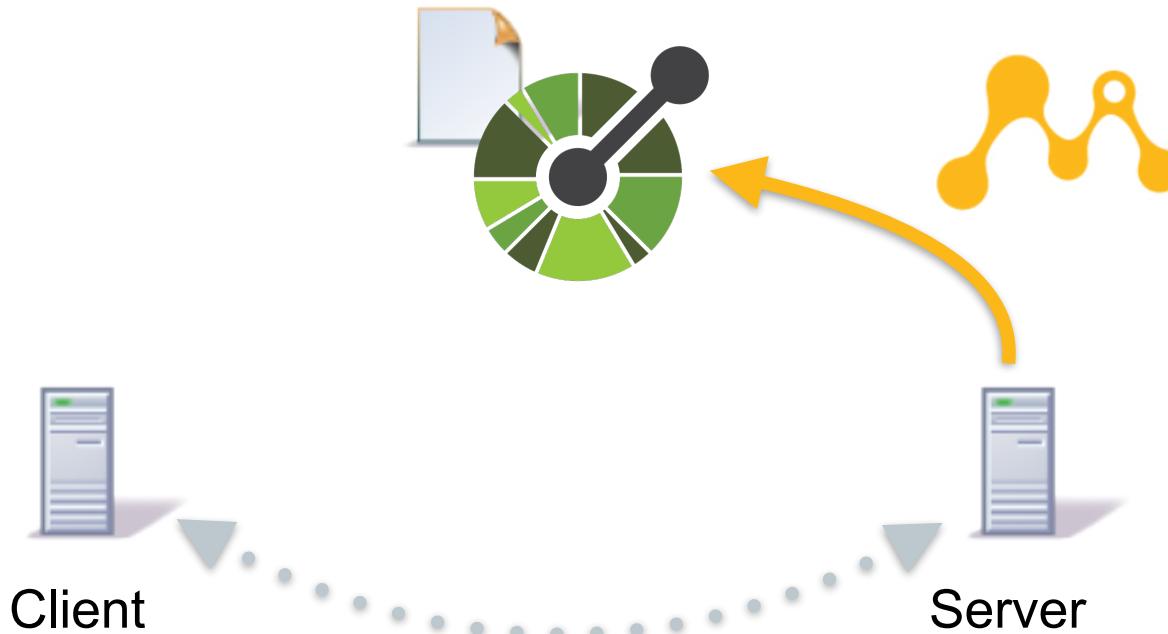
```
<dependency>
  <groupId>io.thorntail</groupId>
  <artifactId>microprofile-openapi</artifactId>
</dependency>
```

- Add OpenAPI annotations (next to the JAX-RS ones)

# Code first approach



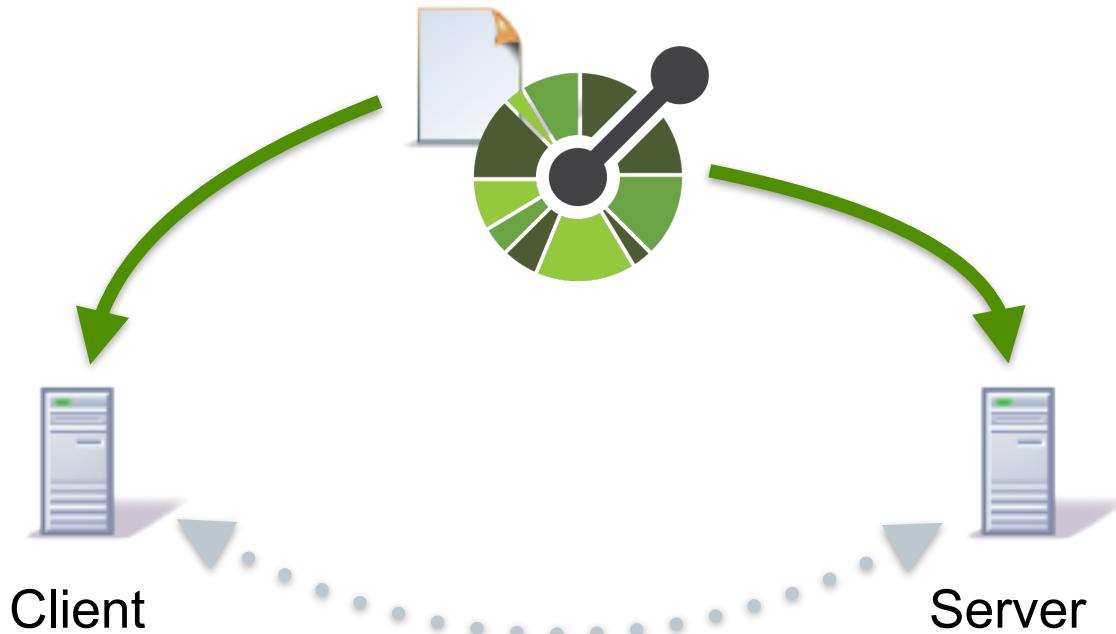
# Code first approach



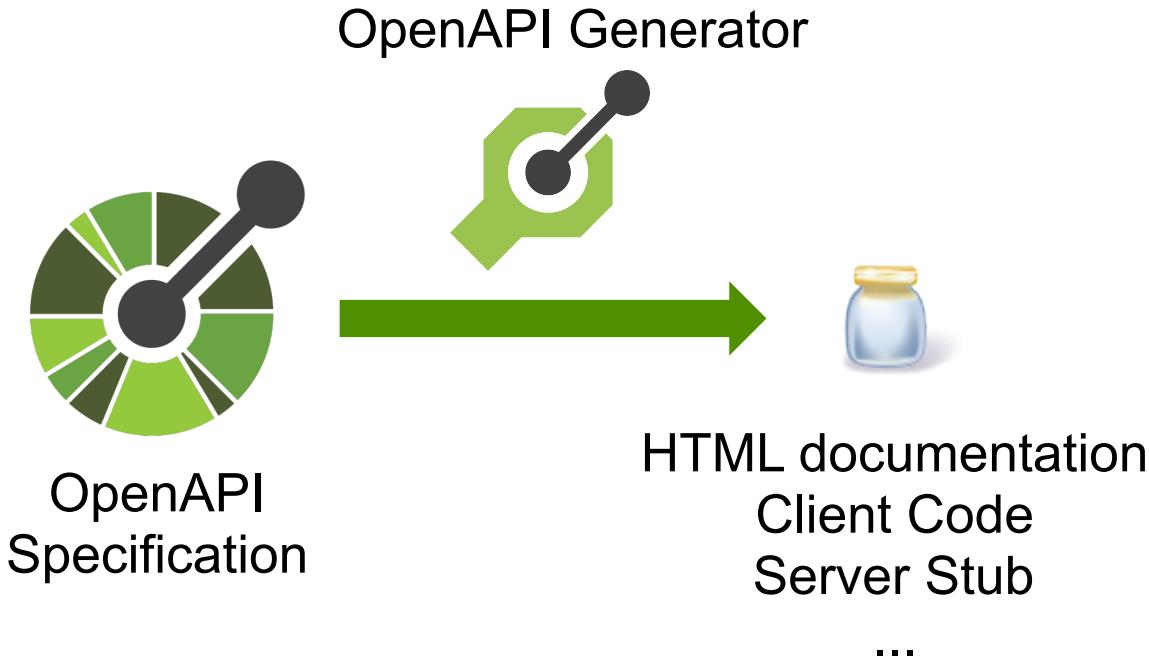
# Specification first approach



# Start with a Specification



# Code generator: OpenAPI-Generator



# OpenAPI-Generator



- **Open Source** (Apache 2.0 License)
- Hosted on **GitHub**:  
<https://github.com/OpenAPITools/openapi-generator>
- Java code & mustache templates
- Fork of Swagger-Codegen



A large word cloud centered around the word "Java", which is rendered in a large, bold, dark font. Surrounding "Java" are numerous other words representing different programming languages and frameworks, all in green. These include: PowerShell, Go, Server, Dart, Play Framework, Confluence, NodeJS, Apache2, Kotlin, Erlang, Server, Scala, Apex, MySQL, Erlang, Scala, Ruby on Rails, C++, REST, Bash, Go, Groovy, JMeter, Elm, Spring, Boot, Rust, clojure, R, Haskell, Eiffel, C++, Perl, Ada, Haskell, Elixir, Lua, Javascript, C#, Haskell, Servant, HTML, Swift, JAX-RS, PHP, TypeScript.

# OpenAPI-Generator - Usage

## Generate a java client for the example application

```
java -jar ./openapi-generator-cli-3.3.1.jar generate  
    -i ../OpenAPI-Spec/todoapp.yaml  
    -g java  
    -o out/
```

# OpenAPI: an ecosystem



# A lot of tools are available for OpenAPI

- Editors
- Tool to generate documentation
- Breaking change detection
- Integration with existing tools (Postman, ...)
- Code generation
  - (OpenAPI-Generator, AutoRest, Swagger-Codegen, ... )



Thank you for  
your attention



@j2r2b



jmini

Code Examples:

<https://github.com/jmini/ece2018-openapi>



eclipsecon  
Europe 2018



## Evaluate the Sessions

Sign in and vote at [eclipsecon.org](https://eclipsecon.org)

-1

0

+1