# RAML - RESTful API Modeling Language

Olivier Liecthi & Laurent Prévost COMEM Web Services



Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

### **API** Documentation



- When you are designing and implementing a REST API, you are most often doing it for **third-party developers**:
  - Think about Twitter, Instagram or Amazon exposing services to external developers.
  - Think about an enterprise (e.g. car manufacturer) exposing services to business partners (e.g. suppliers, subcontractors, distributors).
- The documentation of your API is the first thing that third-party developers (your **customers**) will see. You want to **seduce** them.
- The documentation of your API will have a big impact on its learnability and ease of use.
- Best practices and tools have emerged. Evaluate and apply them!

## RAML by their authors



(Mulesoft but not only - OpenSource - Apache License v2)

RESTful API Modeling Language (RAML) is a simple and succinct way of describing practically-RESTful APIs. It encourages reuse, enables discovery and pattern-sharing, and aims for merit-based emergence of best practices. The goal is to help our current API ecosystem by solving immediate problems and then encourage ever-better API patterns. RAML is built on broadly-used standards such as YAML and JSON and is a non-proprietary, vendor-neutral open spec.

#### heig-vd Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

```
#%RAML 0.8
title: World Music API
baseUri: http://example.api.com/{version}
version: v1
traits:
  - paged:
      queryParameters:
        pages:
          description: The number of pages to return
          type: number
  - secured: !include http://raml-example.com/secured.yml
/sonas:
  is: [ paged, secured ]
  get:
    queryParameters:
      genre:
        description: filter the songs by genre
  post:
  /{songId}:
    get:
      responses:
        200:
          body:
            application/json:
              schema:
                { "$schema": "http://json-schema.org/schema",
                  "type": "object",
                  "description": "A canonical song",
                  "properties": {
                    "title": { "type": "string" },
                    "artist": { "type": "string" }
                  "required": [ "title", "artist" ]
            application/xml:
    delete:
      description: |
        This method will *delete* an **individual song**
```

#### heig-vol Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Preamble

```
#%RAML 0.8
title: World Music API
baseUri: http://example.api.com/{version}
version: v1
traits:
  - paged:
      queryParameters:
        pages:
          description: The number of pages to return
          type: number
  - secured: !include http://raml-example.com/secured.yml
/sonas:
  is: [ paged, secured ]
  get:
    queryParameters:
      genre:
        description: filter the songs by genre
  post:
  /{songId}:
    get:
      responses:
        200:
          body:
            application/json:
              schema:
                { "$schema": "http://json-schema.org/schema",
                   "type": "object",
                   "description": "A canonical song",
                   "properties": {
                    "title": { "type": "string" },
                    "artist": { "type": "string" }
                   "required": [ "title", "artist" ]
            application/xml:
    delete:
      description: |
        This method will *delete* an **individual song**
```

#### heig-vd Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud

Preamble

Traits

(reusability)

```
#%RAML 0.8
 title: World Music API
 baseUri: http://example.api.com/{version}
 version: v1
traits:
    paged:
        queryParameters:
          pages:
            description: The number of pages to return
            type: number
    - secured: !include http://raml-example.com/secured.yml
  /sonas:
    is: [ paged, secured ]
    get:
      queryParameters:
        genre:
          description: filter the songs by genre
    post:
    /{songId}:
      get:
        responses:
          200:
            body:
              application/json:
                schema:
                  { "$schema": "http://json-schema.org/schema",
                    "type": "object",
                    "description": "A canonical song",
                    "properties": {
                      "title": { "type": "string" },
                      "artist": { "type": "string" }
                    "required": [ "title", "artist" ]
              application/xml:
      delete:
        description: |
          This method will *delete* an **individual song**
```

Preamble

Traits

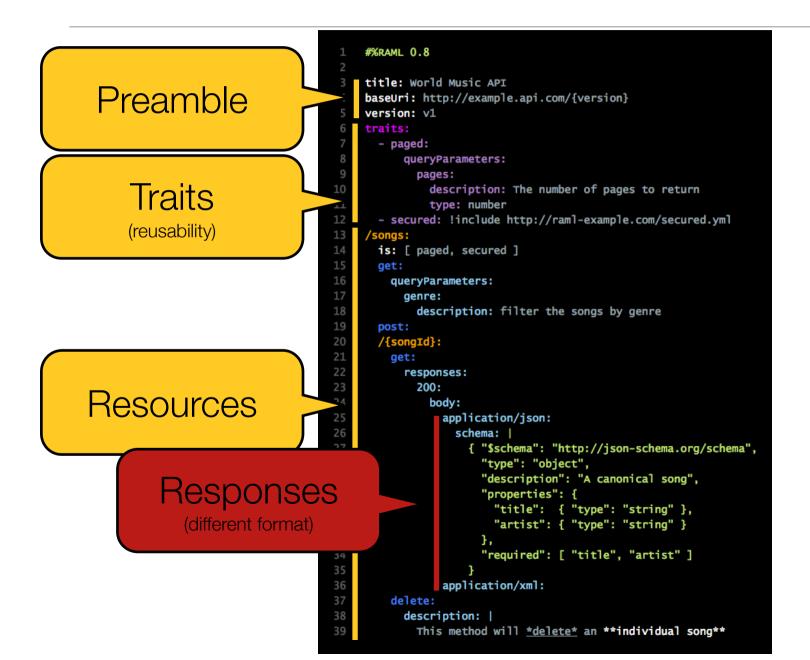
(reusability)

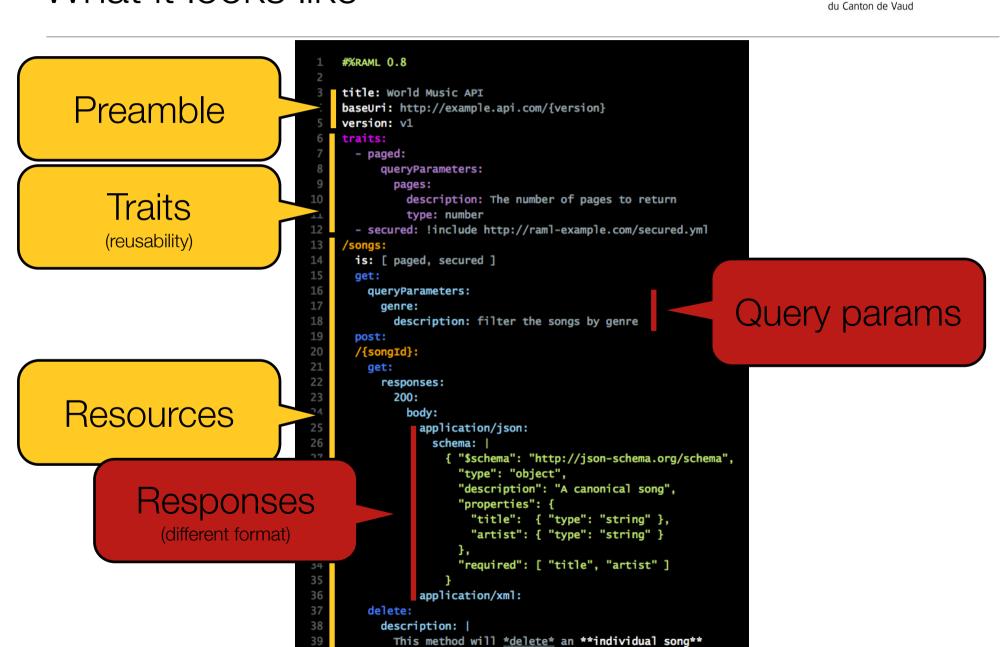
Resources

```
#%RAML 0.8
    title: World Music API
    baseUri: http://example.api.com/{version}
    version: v1
6 traits:
       - paged:
           queryParameters:
             pages:
               description: The number of pages to return
               type: number
      - secured: !include http://raml-example.com/secured.yml
      is: [ paged, secured ]
      get:
        queryParameters:
          genre:
            description: filter the songs by genre
       post:
      /{songId}:
        get:
           responses:
            200:
               body:
                application/json:
                   schema:
                     { "$schema": "http://json-schema.org/schema",
                       "type": "object",
                       "description": "A canonical song",
                       "properties": {
                         "title": { "type": "string" },
                         "artist": { "type": "string" }
                       "required": [ "title", "artist" ]
34
                application/xml:
        delete:
          description: |
            This method will *delete* an **individual song**
```

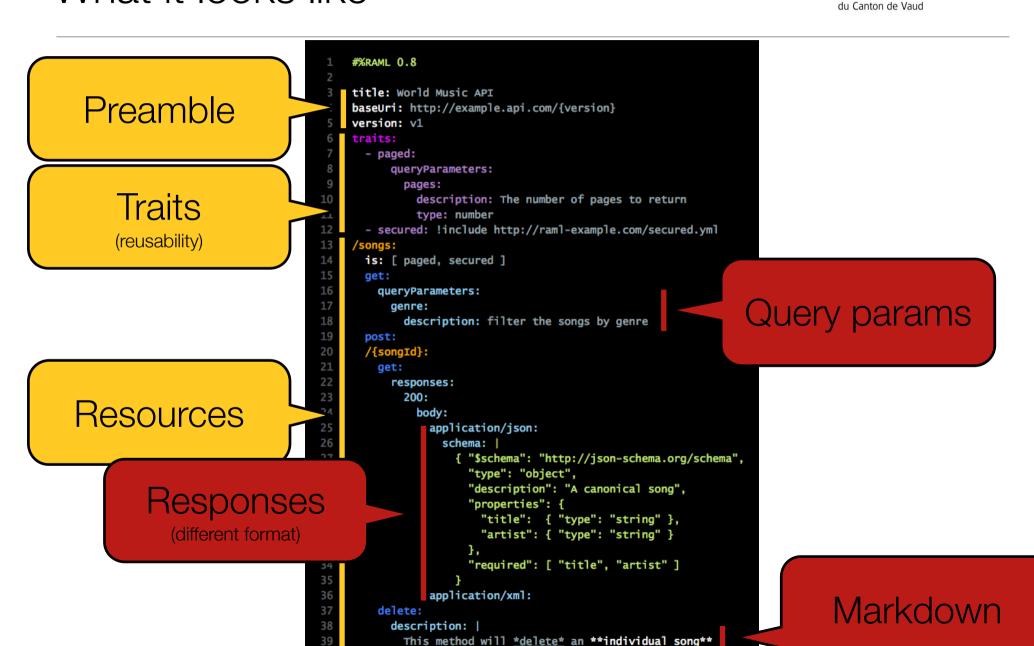
## heig-vd

Haute Ecole d'Ingénierie et de Gestion du Canton de Vaud









#### RAML



- RAML is pretty straightforward to use:
  - You describe the list of resources managed by your application, document the support HTTP verbs, enlist the query parameters, etc.
  - Most of modern IDE and text editors supports YAML syntax highlighting. As RAML is based on YAML.
  - If you use **Sublime Text**, you can take advantage of an extension that provides **syntax highlighting** specifically designed for RAML.
  - See RAML 100 tutorial (http://raml.org/docs.html)
- RAML has advanced features that can make your specifications less verbose (by abstracting and reusing common elements):
  - includes
  - resource types and schemas
  - traits
  - See RAML 200 tutorial (http://raml.org/docs-200.html)

### **RAML** Alternatives





- Swagger (Reverb Open Source Apache License v2)
  - Swagger is a simple yet powerful representation of your RESTful API. With the largest ecosystem of API tooling on the planet, thousands of developers are supporting Swagger in almost every modern programming language and deployment environment. With a Swagger-enabled API, you get interactive documentation, client SDK generation and discoverability.



- API Blueprint (APlary Open Source MIT)
  - Connecting the dots in API development
    - Web API Language Pure Markdown Designed for Humans -Understandable by Machines - Powerful Tooling - Easy Lifecycle



API Doc Home Blog API Reference

#### Welcome

### **Smart City**

#### Citizen Engagement

Welcome on board! You are about to discover the wonderful world where people are participating in the city life everyday.

Each citizen is able to submit any issue he disovers in the city and then a staff member can take care of this issue to solve it.

You will discover on this website the marvelous API documentation and the epic stories



API Doc Home Blog API Reference

#### **Overview**

This pages contains general documentation about the API. Use the links on the right to navigate to specific resources.

#### Content-type

The API uses the JSON format. Unless specified otherwise, all requests and response should have the Content-Type: application/json header.

#### **HTTP** verbs

The API uses the standard HTTP verbs to perform CRUD operations (Create, Retrieve, Update, Delete) on resources, following standard RESTful API practices.

Find below a quick summary of how HTTP verbs are used in the API:

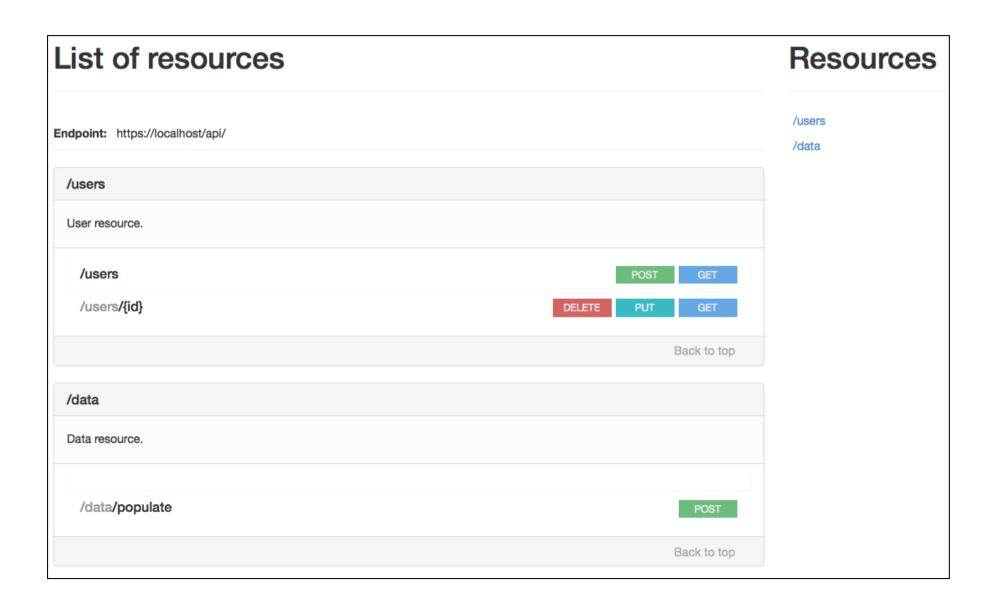
Verb	Description
GET	Used for retrieving a resource or a collection of resources.
POST	Used for creating a new resource or performing a non-CRUD operation on a resource.
PUT	Used to perform a full update of a resource (replacing the resource by the JSON data provided in the request).
DELETE	Used for deleting resources.

HEAD and PATCH are not currently used.

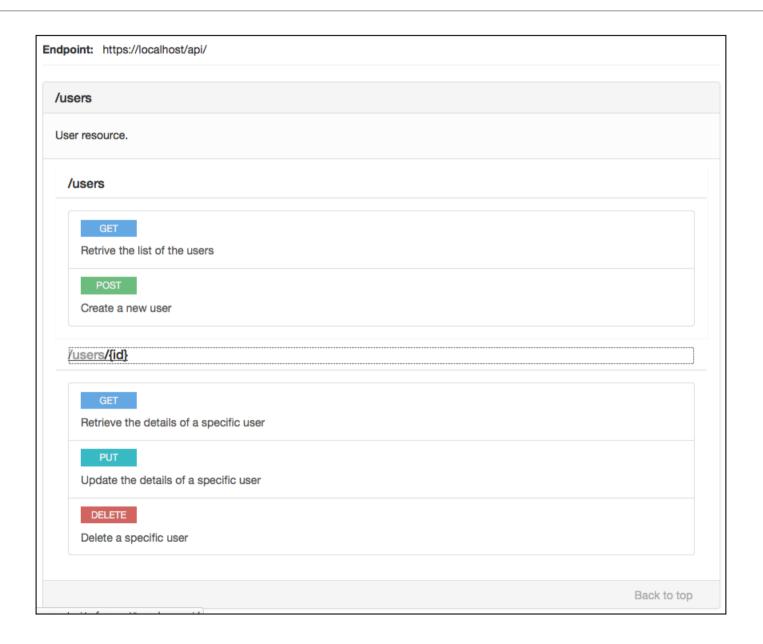
#### Resources

/users /data

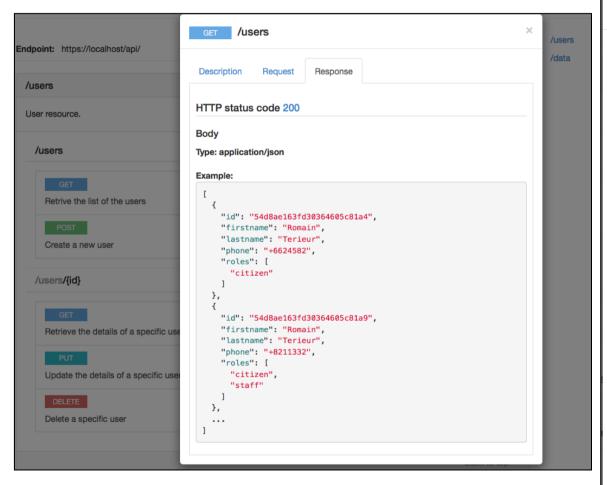


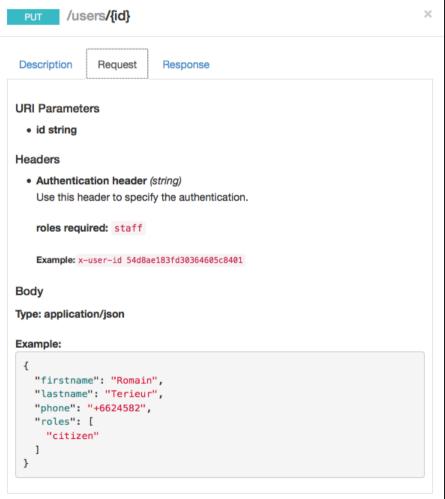














- apidoc-seed is an open source tool, which makes it easy to generate a complete HTML site for documenting your REST API.
- To use the tool, **clone the GitHub repo** (<a href="https://github.com/lotaris/apidoc-seed">https://github.com/lotaris/apidoc-seed</a>) and follow the readme instructions.
  - edit/create jade templates and markdown documents to provide documentation for your service (general service information, usage guides, support information, etc.).
  - edit/create RAML files to document your REST APIs. Depending on the complexity of your APIs, you can split the documentation into several files.
- The tool supports a notion of "**private**" API elements. This is used if your API has resources, methods or parameters that you don't want to publicly expose yet (note that this is documentation level only, nothing will prevent a user to send a request!).

#### Individual work



- Clone the repo, build and test it in your browser.
  - https://github.com/lotaris/apidoc-seed
- Follow the instructions in the **RAML 100 tutorial** (http://raml.org/docs.html), but do it in the /src/api/raml/index.raml file (keep a copy of the original file).
- Modify the Welcome page in the documentation site (when you deliver your project, I expect to find a description of your platform here).
- Add an article in the blog that describe, if any, your difficulties to setup the project.
- Customize the page footer.

#### Resources



- http://control.laneworks.net/admin/project/jobs/api-design/white-paper.php?
   Project ID=17
- http://swagger.io/
- https://apiblueprint.org/
- http://raml.org/licensing.html
- https://github.com/SoftEng-HEIGVD/Teaching-HEIGVD-CM\_WEBS-2015-Labo-Doc
- https://github.com/lotaris/apidoc-seed