REST API Documentation

Olivier Liechti & Simon Oulevay
COMEM Web Services 2016

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

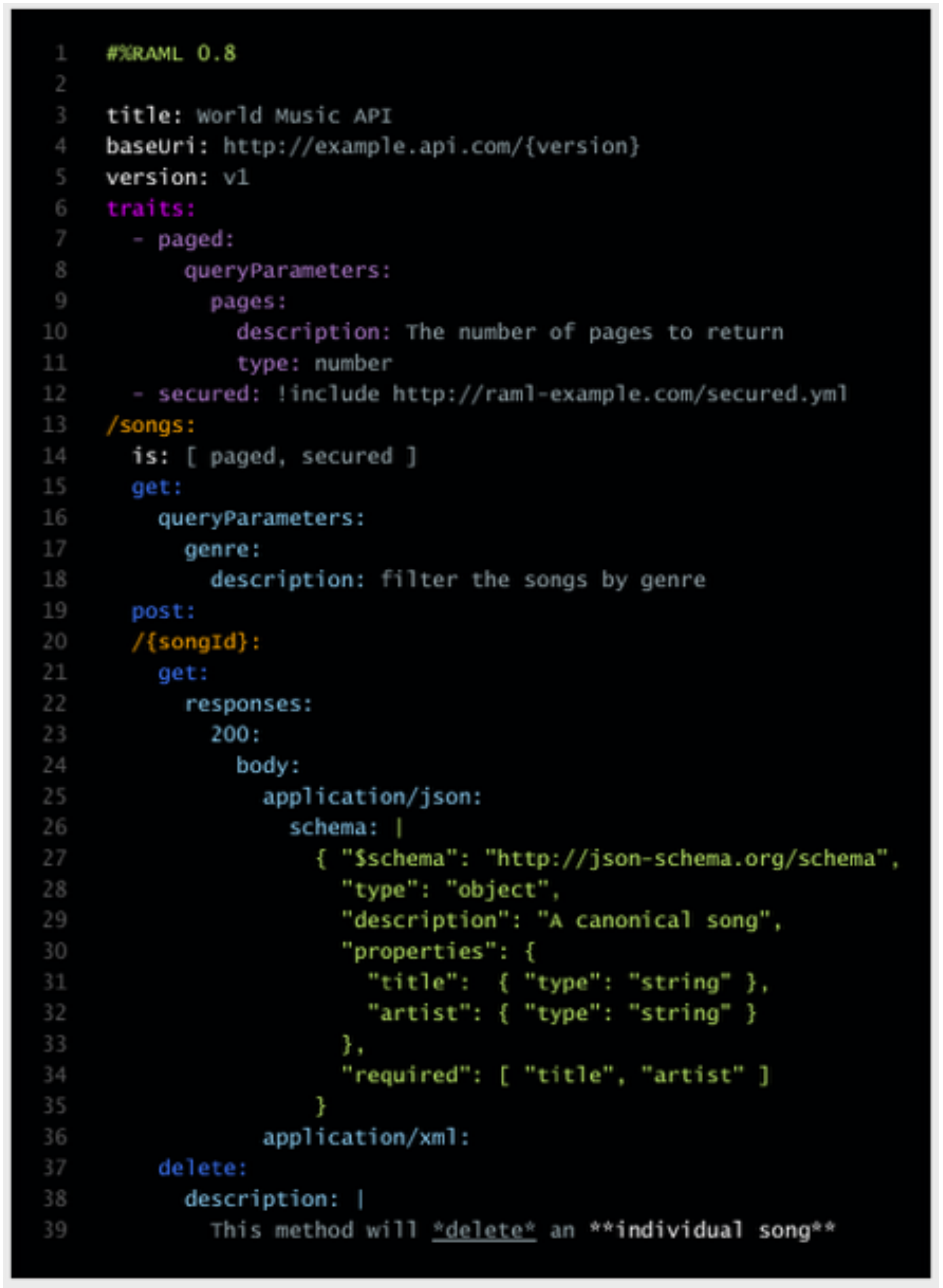
API Documentation

- 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

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

- **RE**STful **A**PI **M**odeling **L**anguage
- RAML is a language that has been developed to facilitate the **design** and **documentation** of REST APIs.
- It allows you to describe **resources**, **methods**, **parameters**, **headers** and **payloads** in a succinct manner (support for abstraction and reuse).
- From a RAML file, it is possible to **generate a user-friendly documentation** (e.g. in HTML) with various **tools**.
- Other tools support import/export exchange with REST frameworks (e.g. JAX-RS).



```
#%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
/songs:
  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**
```

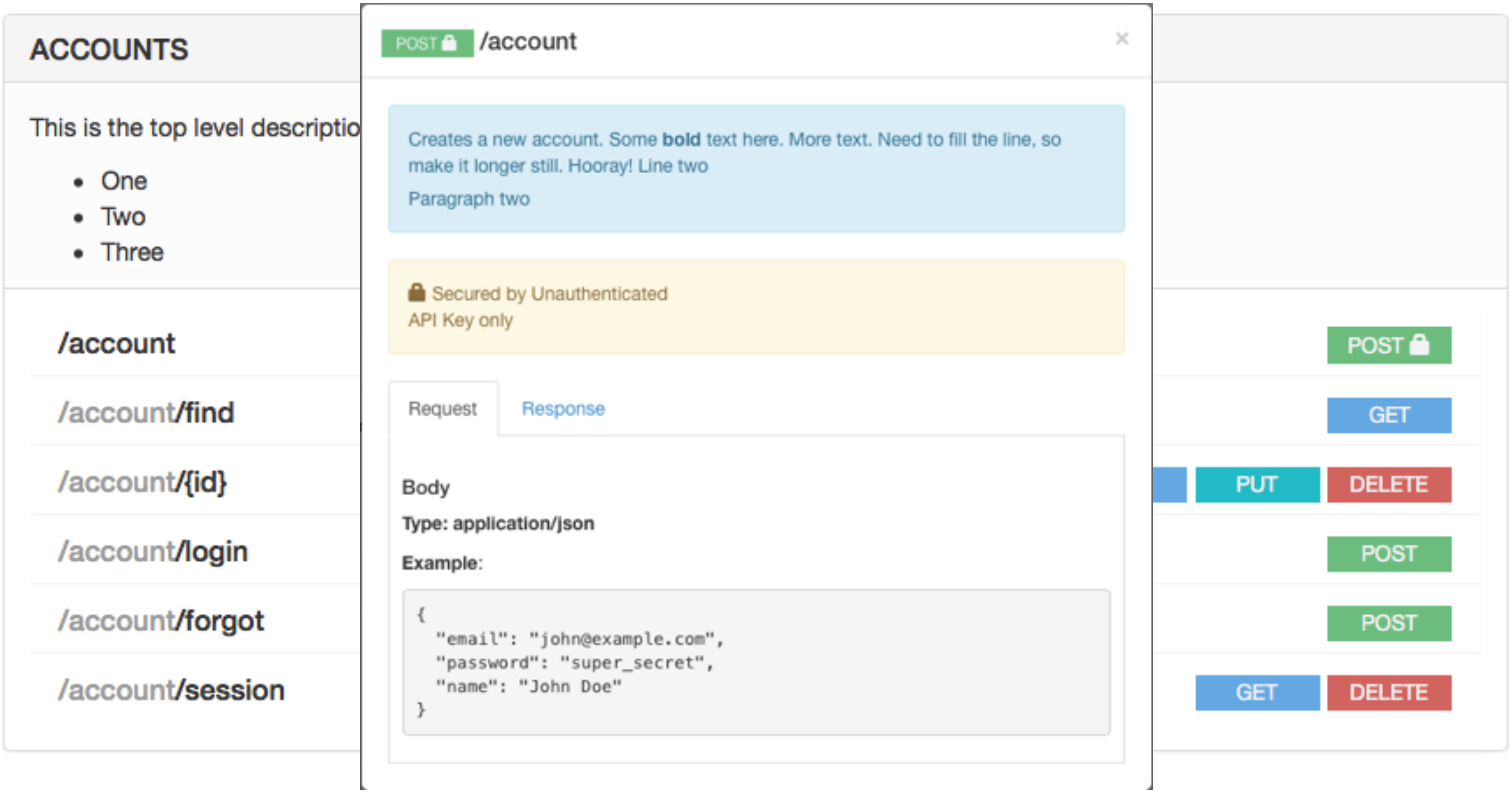
RAML

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

- 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.
  - If you use **Sublime Text**, you can take advantage of an extension that provides **syntax highlighting**.
  - 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)

Convert RAML to HTML

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


ACCOUNTS

This is the top level descriptio

- One
- Two
- Three

/account — POST
/account/find — GET
/account/{id} — PUT DELETE
/account/login — POST
/account/forgot — POST
/account/session — GET DELETE

POST /account

Creates a new account. Some **bold** text here. More text. Need to fill the line, so make it longer still. Hooray! Line two

Paragraph two

Secured by Unauthenticated
API Key only

Request | Response

Body

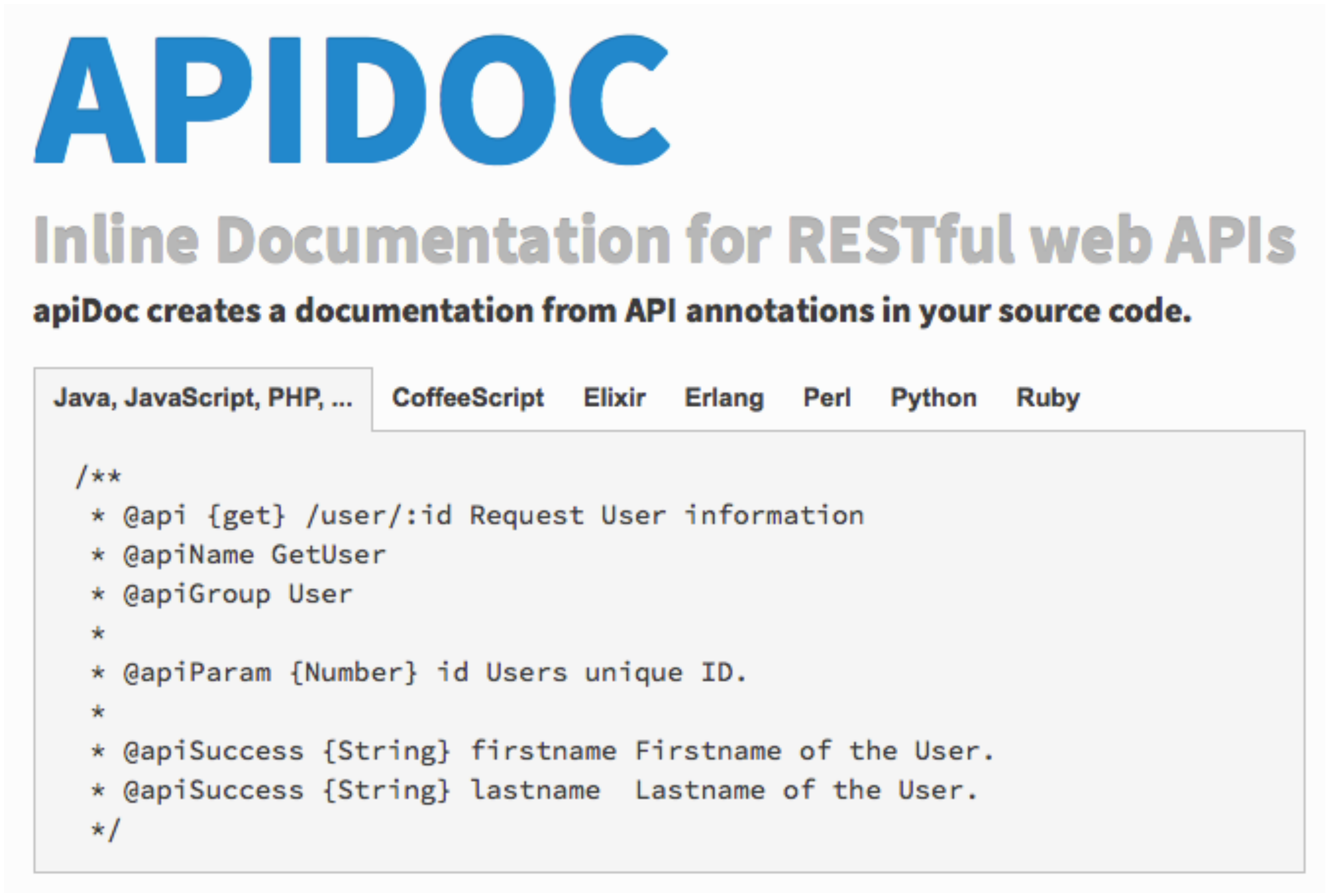
Type: application/json

Example:

```
{
  "email": "john@example.com",
  "password": "super_secret",
  "name": "John Doe"
}
```

https://rawgit.com/raml2html/raml2html/master/examples/example.html

APIDOC


APIDOC
Inline Documentation for RESTful web APIs

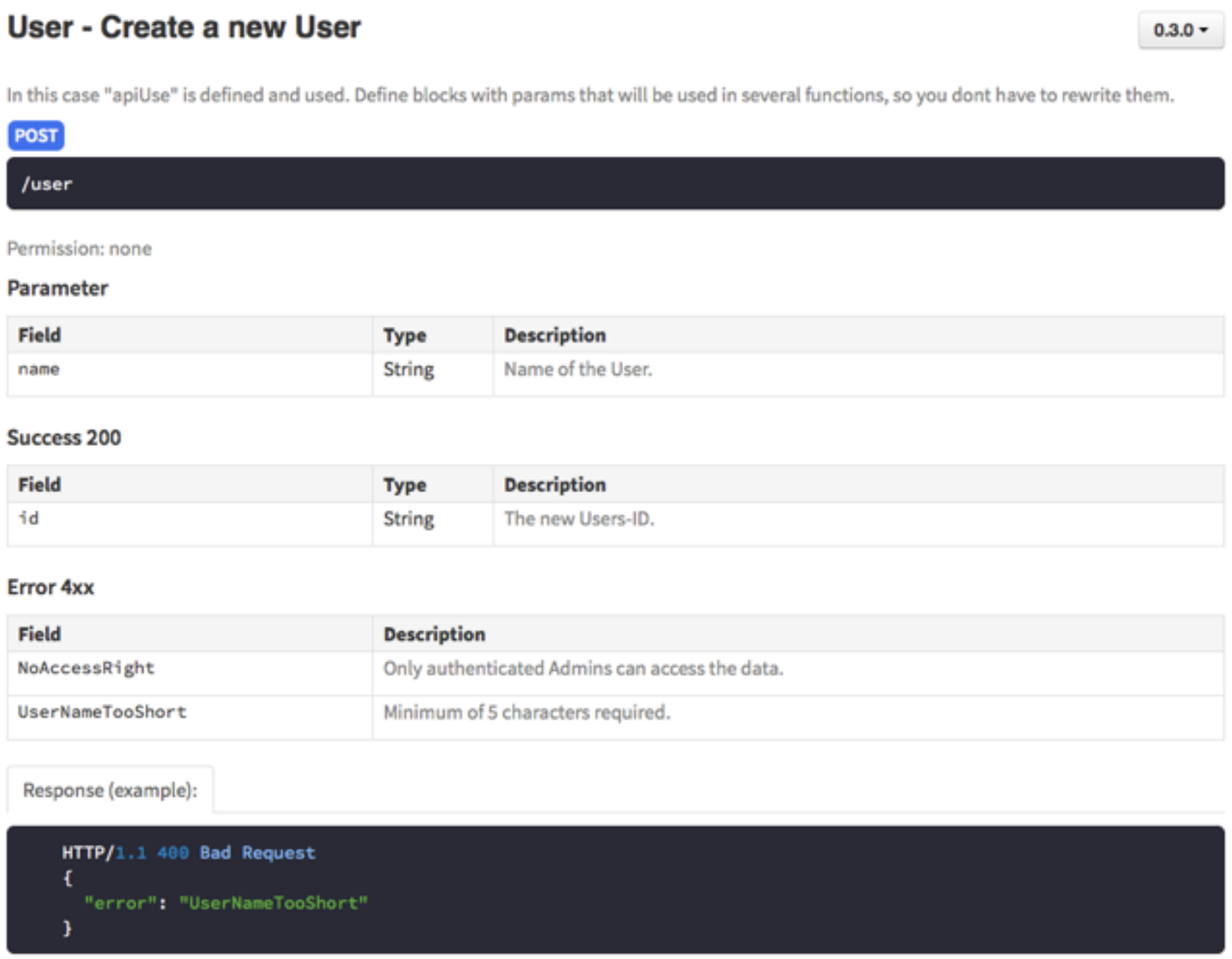
apiDoc creates a documentation from API annotations in your source code.

Java, JavaScript, PHP, ...
CoffeeScript
Elixir
Erlang
Perl
Python
Ruby

```
/**
 * @api {get} /user/:id Request User information
 * @apiName GetUser
 * @apiGroup User
 *
 * @apiParam {Number} id Users unique ID.
 *
 * @apiSuccess {String} firstname Firstname of the User.
 * @apiSuccess {String} lastname  Lastname of the User.
 */
```

Generate HTML from APIDOC

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



User - Create a new User

0.3.0 ▾

In this case "apiUse" is defined and used. Define blocks with params that will be used in several functions, so you dont have to rewrite them.

POST

```
/user
```

Permission: none

Parameter

Field	Type	Description
`name`	String	Name of the User.

Success 200

Field	Type	Description
`id`	String	The new Users-ID.

Error 4xx

Field	Description
`NoAccessRight`	Only authenticated Admins can access the data.
`UserNameTooShort`	Minimum of 5 characters required.

Response (example):

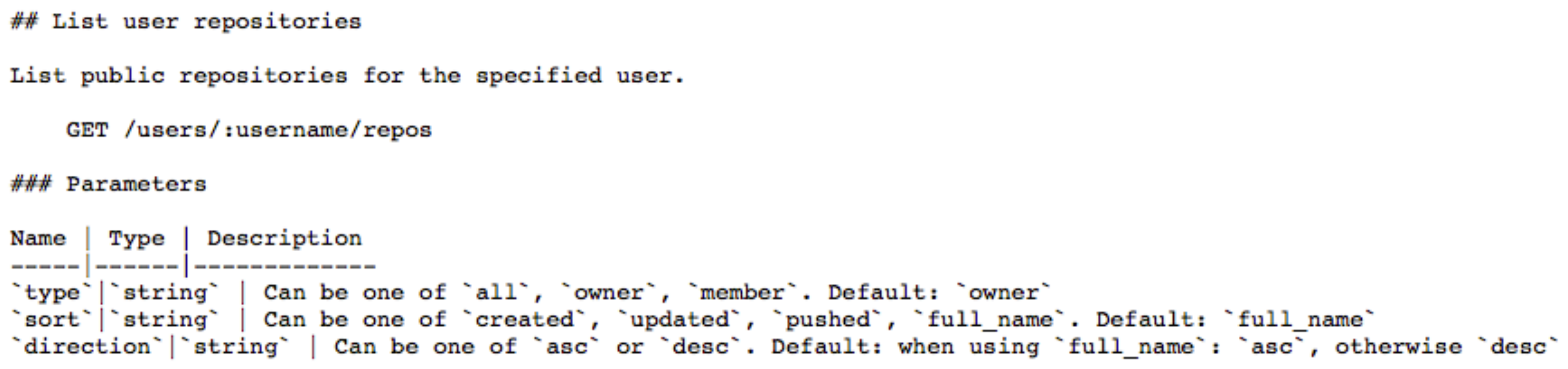
```
HTTP/1.1 400 Bad Request
{
  "error": "UserNameTooShort"
}
```

http://apidocjs.com/example/

Custom example: GitHub API

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

Written in Markdown, an easy-to-read, easy-to-write plain text format.

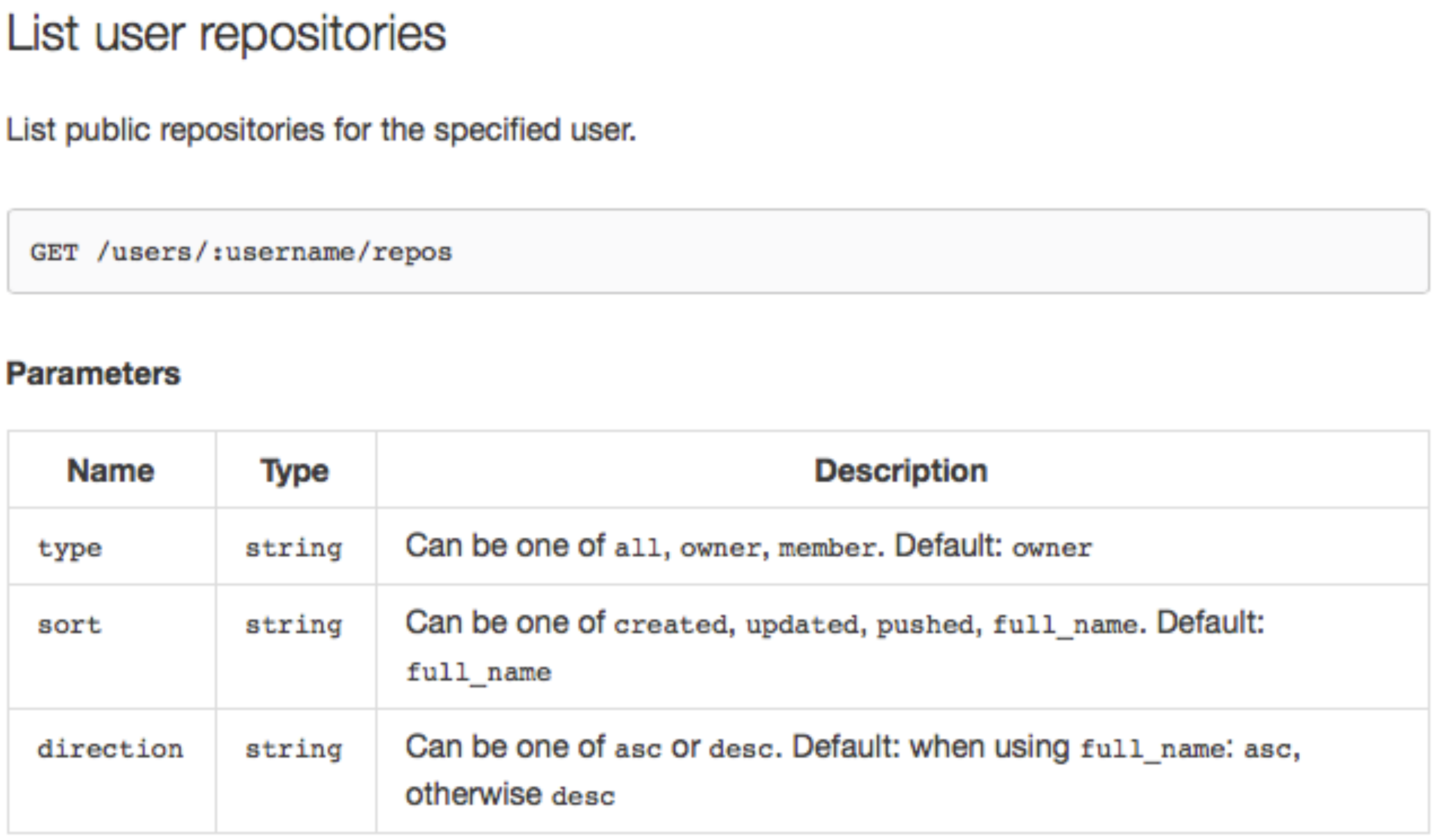


```
## List user repositories

List public repositories for the specified user.

    GET /users/:username/repos

### Parameters

Name | Type | Description
-----|------|--------------
`type`|`string` | Can be one of `all`, `owner`, `member`. Default: `owner`
`sort`|`string` | Can be one of `created`, `updated`, `pushed`, `full_name`. Default: `full_name`
`direction`|`string` | Can be one of `asc` or `desc`. Default: when using `full_name`: `asc`, otherwise `desc`
```

Rendered to HTML with nanoc, a static website generator written in Ruby.

Custom example: GitHub API

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



List user repositories

List public repositories for the specified user.

```
GET /users/:username/repos
```

Parameters

Name	Type	Description
`type`	`string`	Can be one of `all`, `owner`, `member`. Default: `owner`
`sort`	`string`	Can be one of `created`, `updated`, `pushed`, `full_name`. Default: `full_name`
`direction`	`string`	Can be one of `asc` or `desc`. Default: when using `full_name`: `asc`, otherwise `desc`

https://developer.github.com/v3/

Don't forget Grunt

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

RAML to HTML with Grunt

https://github.com/cybertk/grunt-raml2html

APIDOC to HTML with Grunt

https://github.com/apidoc/grunt-apidoc

API documentation examples

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

https://developer.github.com/v3/

https://dev.twitter.com/rest/public

https://developers.facebook.com/docs/graph-api

https://www.instagram.com/developer/