



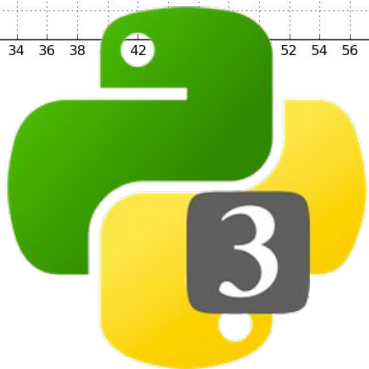
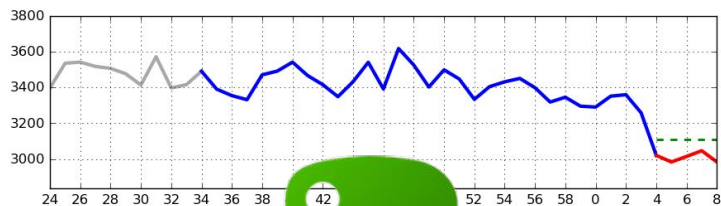
# Leveraging documentation power for better web APIs

Rudy Sicard

*Europython 2016*

# Context

---



## Monitoring as a Service

### Browsable APIs

Hugues Lerebours

Rudy Sicard

What ?

---

# Browsable API

# What ?

---

Browser

Explorable

# Browsable API

Experimentable



# What ?

---

## Browsable API

Contract

Interface Specification

Documentation

Valid use



# Features

---

- Use documentation
- Explorable & Experimentable
- In/Output Validation & Conversion
- Html Form & Validation
- Automated Tests
- User/Dev friendliness
- Generate a spec, e.g. Swagger



# Features

---

- Use documentation
- Explorable & Experimentable
- In/Output Validation & Conversion
- Html Form & Validation
- Automated Tests
- User/Dev friendliness
- Generate a spec, e.g. Swagger

## /usage

- ▶ DESCRIPTION
- ▶ HEADERS
- ▶ JSON RESPONSE
- ▼ PRETTIER RESPONSE

```
[
  /acknowledge (POST),
  /acknowledge/(?P[0-9]{10}) (POST),
  /acknowledgements/([-a-f0-9]{32,36}) (DELETE),
  /acknowledgements/active (GET),
  /acknowledgements/active/([-a-f0-9]{32,36}) (GET),
  /acknowledgements/archived (GET),
  /alerts (POST),
  /alerts/search (POST),
  /apidoc/json_schema (GET),
  /downtime (POST),
  /downtimes/([-a-f0-9]{32,36}) (DELETE),
  /downtimes/active (GET),
  /downtimes/active/([-a-f0-9]{32,36}) (GET),
  /downtimes/archived (GET),
  /usage (GET)
]
```

# Features

- Use documentation
- Explorable & Experimentable
- In/Output Validation & Conversion
- Html Form & Validation
- Automated Tests
- User/Dev friendliness
- Generate a spec, e.g. Swagger

The screenshot shows a Swagger API interface for the `/alerts` endpoint. It includes a description, parameters, and HTTP codes for known errors.

**/alerts**

▼ DESCRIPTION

Push an alert, like the response of a check.

PARAMETERS

<b>series_name</b>	<i>str</i>	name identifying alert series, must be unique and 255 char max
<b>level</b>	<i>int or str</i>	among 0 'NORMAL' 1 'UNKNOWN' 2 'WARNING' 3 'CRITICAL'
<b>tags</b>	<i>Tags</i>	list of tags, or typed tags in a dictionary, to be able to efficiently filter alerts by tags. Note: one c
	<i>default:</i>	For the time being, all alerts come from Nagios and are silenced in Nagios; we need Nagios to
	<b>None</b>	named hostname and service. Note: all tag types and values will be converted to lower-case.
<b>description</b>	<i>str default:</i>	description of the check that creates this alert
	<b>None</b>	

HTTP CODES FOR KNOWN ERRORS

<b>422</b>	<i>ValueError</i>	e.g. if there is not both hostname and service tags. Error responses are of the form: {"message": "[ErrorType] verbose details"}
------------	-------------------	---

▼ play with me!

POST



# Features

---

- Use documentation
- Explorable & Experimentable
- In/Output Validation & Conversion
- Html Form & Validation
- Automated Tests
- User/Dev friendliness
- Generate a spec, e.g. Swagger

A stylized logo for JSON, featuring the word "JSON" in blue capital letters with a 3D effect, enclosed in large blue curly braces. The letter "O" is a solid black circle.

# Features

---

- Use documentation
- Explorable & Experimentable
- In/Output Validation & Conversion
- **Html Form & Validation**
- Automated Tests
- User/Dev friendliness
- Generate a spec, e.g. Swagger

▼ play with me!

PARAMETERS ▼ JSON Properties

Please fill in the following parameters

series\_name

level integer ▼

tags array ▼

▼

+ item

description

POST

# Features

---

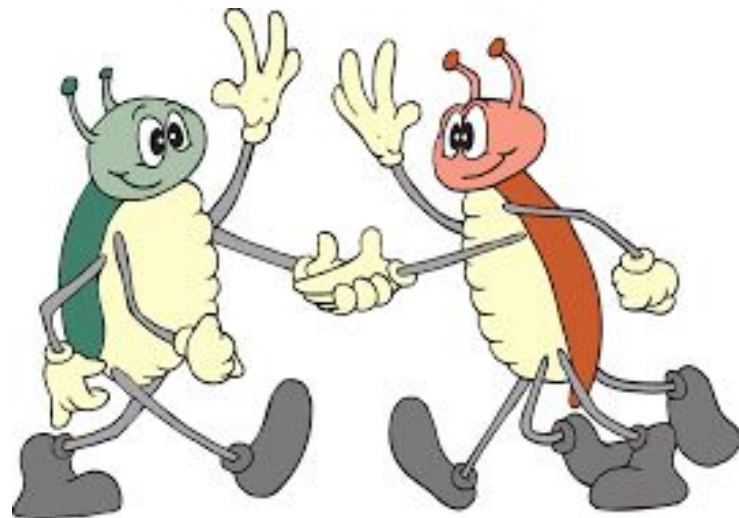
- Use documentation
- Explorable & Experimentable
- In/Output Validation  
& Conversion
- Html Form & Validation
- **Automated Tests**
- User/Dev friendliness
- Generate a spec, e.g. Swagger



# Features

---

- Use documentation
- Explorable & Experimentable
- In/Output Validation & Conversion
- Html Form & Validation
- Automated Tests
- **User/Dev friendliness**
- Generate a spec, e.g. Swagger



# Features

---

- Use documentation
- Explorable & Experimentable
- In/Output Validation  
& Conversion
- Html Form & Validation
- Automated Tests
- User/Dev friendliness
- Generate a spec, e.g. Swagger



# Plan

---

- **Use documentation**
- Explorable & Experimentable
- **In/Output Validation & Conversion**
- **Html Form & Validation**
- **Automated Tests**
- **User/Dev friendliness**
- Generate a spec, e.g. Swagger



# Use documentation

---

```
@myapi.publish(...)
def my_operation(arg0, arg1, arg2):
    """
    :param int|str arg0: ...
    :param dict[str, int] arg1: ...
    :param MyTypeA[str] arg2: ...
    :rtype: list[MyTypeB]
    :raise ValueError: ...
    :raise MyException: ...
    """
```

- Parsing & Introspection
- Input/Output/Exception
- Comments ...
- **Type constraints**
- Conversion & json schema

# Use documentation

---

```
@myapi.publish(...)
def my_operation(arg0, arg1, arg2):
    """
    :param int|str arg0: ...
    :param dict[str, int] arg1: ...
    :param MyTypeA[str] arg2: ...
    :rtype: list[MyTypeB]
    :raise ValueError: ...
    :raise MyException: ...
    """
```

- Parsing & Introspection
- Input/Output/Exception
- Comments ...
- **Type constraints**
- Conversion & json schema



# Use documentation

---

```
@myapi.publish(...)
def my_operation(arg0, arg1, arg2):
    """
    :param int|str arg0: ...
    :param dict[str, int] arg1: ...
    :param MyTypeA[str] arg2: ...
    :rtype: list[MyTypeB]
    :raise ValueError: ...
    :raise MyException: ...
    """
```

- Parsing & Introspection
- [Input/Output/Exception](#)
- Comments ...
- **Type constraints**
- Validation & Conversion

# Json & Jsonschema

 python ↔ {JSON}

None	null
bool	boolean
str	string
int	integer
float	number
list[json]	array
dict[str, json]	object

```
{  
  "type": ...,  
  "minimum": ...,  
  "items": ...,  
  "properties": ...,  
  ...  
}
```



- FromJson
- FromJsonWithSchema

# Automated In/Output Validation & Conversion

---

## 1. From doc, i.e. type constraint

```
[ValueError] Parameter `arg0`: Constraint is `int|str`, got a `float`
```

## 2. From code, i.e. json schema

```
[ValueError] Parameter `apples`: under `weight`, `two` is not of type `number`
```

## 3. Implicit & hand-crafted conversion

## 4. Implementation is called

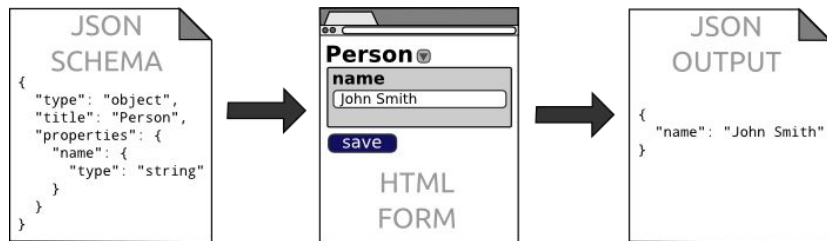
## 5. Exception

- hide undocumented exception

## 6. Warn on non validated returned value

# Automated Html Form & Validation

- Javascript [json-editor](#)
- Form & Validation w.r.t. Json schema
- Plug into the html template



Source: <https://github.com/jdorn/json-editor>

▼ play with me!

▼ play with me!

PARAMETERS ▼

Please fill in the following parameters

series\_name

level

tags

▼

description

Valid input

# Automated testing

---

- hypothesis
  - Property-based testing
  - Strategies to generate inputs
  - Fantastic tool, try it
- jsonschema strategy
- Checked properties
  - Input conversion
  - Undocumented or unused exceptions
  - Output type & conversion to json

```
Error in "addition", undocumented exceptions:
```

```
* for instance "TypeError" from 1 locations:
```

```
1. TypeError("unsupported operand type(s) for +:
'int' and 'str'",)
Traceback (most recent call last):
  File "fruit/api.py", line 59, in addition
    1 + ''
```

```
Warning in "addition", the documented exceptions cannot
be generated: NeedMoreApples
```

```
Error in "addition", must return a "list[Fruit]" but
returned something else:
```

```
* for instance a "list[int]", e.g. [1]
```

# User "Friendliness"

- Pretty view



- Input-Link
  - Json data via anchors
  - Useful in:
    - Pretty view
    - Tutorial
    - Sharing

```
▼ JSON RESPONSE
[
  {
    "series_name": "Eggs",
    "last_update_time": 1469044902.045574,
    "count_at_current_level": 1,
    "level": 0,
    "last_level_change_time": 1469044902.045574,
    "description": "Spam",
    "tags": ":hostname:py :service:thon "
  }
]
```

```
▼ PRETTIER RESPONSE
[
  {
    "active acknowledgements": [] + acknowledge + acknowledge last level change,
    "active downtimes": [] + downtime me,
    "count_at_current_level": 1,
    "description": "Spam",
    "last_level_change_time": 2016-07-20 20:01:42.045574+00:00,
    "last_update_time": 2016-07-20 20:01:42.045574+00:00,
    "level": OK,
    "series_name": "Eggs",
    "tags": {
      "hostname": "py",
      "service": "thon"
    }
  }
]
```

**play with me!**

**PARAMETERS** [dropdown] [edit icon]

Please fill in the following parameters

**comment**

**alert\_series\_name**

# What's next ?

---

- Open-sourcing:
  - watch <https://github.com/criteo>
  - contribution to hypothesis
  - type\_validation
  - json\_api
- Enum support
- Integration with py.test
- Swagger spec generation
  - Code generation for client library
- Type annotations

# Summary

---

- Browsable APIs are user & dev friendly
- Documentation & types enables automation
  - Conversion
  - Validation
  - Forms
  - Testing
- Json & Json schema are good for such tasks



# Q & A, don't be shy

---

- Related talks you may like:
  - **Friday 22 July** at 14:00 [building-beautiful-restful-apis-using-flask](#)
  - **Friday 22 July** at 14:30 [restful-api-best-practices](#)
- Criteo is hiring
  - Curious ? Come to our booth ! We've got cookies
  - Locations: Paris & Palo Alto
  - With relocation packages, family included
- Sprint: around hypothesis, this week-end
  - jsonschema strategy & other introspection strategies



# Resources

---

- Specifications
  - <http://json-schema.org>
  - <http://modeling-languages.com/modeling-web-api-comparing>
  - <http://www.mikestowe.com/2014/12/api-spec-comparison-tool.php>
  - <https://github.com/marcgibbons/django-rest-swagger>
- Forms
  - <http://jeremydorn.com/json-editor>
  - <https://github.com/joshfire/jsonform>
  - <https://github.com/marcgibbons/django-rest-swagger>
- Testing: <http://hypothesis.readthedocs.io/en/latest/index.html>