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?

Browsable API

Contract
Interface Specification
Documentation
Valid use

Swagger
The World’s Most Popular Framework for APIs.
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
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
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

```python
@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: ...
    """
Json & Jsonschema

```python
{JSON}
```

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

- FromJson
- FromJsonWithSchema
Automated In/Output Validation & Conversion

1. From doc, i.e. type constraint
   ```python
   [ValueError] Parameter `arg0': Constraint is `int|str', got a `float'
   ```

2. From code, i.e. json schema
   ```python
   [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
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
What’s next?

- Open-sourcing:
  - watch [https://github.com/criteo](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