API First Web Development with Python

a tutorial for falsy

author: dameng

Python 快速入门

- 内建类型
- 流程控制
- 函数与方法

```python
a = 1+1.0
print(a, type(a))
```

```
2.0 <class 'float'>
```

```python
b = 1+1j
print(b, type(b))
```

```
(1+1j) <class 'complex'>
```

```python
c = 0x10 + 10_00 + 0o10
print(c, type(c))
```

```
1024 <class 'int'>
```

```python
print(1-0.8)
print(1-0.5-0.25-0.125,1-0.875)
from decimal import Decimal as Dec
a = Dec('1')
b = Dec('0.8')
print(a-b)
```

```
0.19999999999999996
0.125 0.125
0.2
```

```python
print("hello"+' '*2+'world')
a = "hello"
b = 123.456
print("{} {}".format(a, b))
print("{a} {b}".format(a=a, b=b))
print(f"{a} {b}")
print("{} {:.2f}".format(a,b))
import math
print('PI = {0.pi}'.format(math))
```

```
hello  world
hello 123.456
hello 123.456
hello 123.456
hello 123.46
PI = 3.141592653589793
```

```python
a = [1,2,3]
b = ["4", "5", "6"]
c = a + b
print(c, type(c))
c.append(7)
print(c)
```

```
[1, 2, 3, '4', '5', '6'] <class 'list'>
[1, 2, 3, '4', '5', '6', 7]
```

```python
a = {1: 'a', 2: 'b'}
print(a, type(a))
b = (1,2,3)
print(b, type(b))
c = {1, 2, 3}
print(c, type(c))
print('*'*20)
a[3] = 'c'
print(a)
c.add(2)
print(c)
```

```
{1: 'a', 2: 'b'} <class 'dict'>
(1, 2, 3) <class 'tuple'>
{1, 2, 3} <class 'set'>
********************
{1: 'a', 2: 'b', 3: 'c'}
{1, 2, 3}
```

```python
s = 90
if s >= 60:
    print('passed')
else:
    print('failed')

ans = 'passed' if s >= 60 else 'failed'
print(ans)
ans = s >= 60 and 'passed' or 'failed'
print(ans)
```

```
passed
passed
passed
```

```python
for i in [1,2]:
    print(i)
print('*'*20)
for k,v in {1:'a', 2:'b', 3:'c'}.items():
    print(k, v)
print('*'*20)
for i,v in enumerate([1, '2']):
    print(i,v)
else:
    print('done')
```

```
1
2
********************
1 a
2 b
3 c
********************
```

```python
m = int(input('number 1: '))
n = int(input('number 2: '))
while n != 0:
    r = m % n
    m = n
    n = r
print("GCD:", m)
```

```
number 1: 360
number 2: 128
 GCD: 8
```

```python
a = 10
while a > 0:
    print(a, end='.')
    a -= 1
else:
    print('done')
```

```
10.9.8.7.6.5.4.3.2.1.done
```

```python
a = [i for i in range(5)]
print(a)
a = list(map(lambda x: x*2, a))
print(a)
a = list(filter(lambda x: x%3 == 0, a))
print(a)
import functools as fn
a = fn.reduce(lambda x,y: x+y, range(11))
print(a)
```

```
[0, 1, 2, 3, 4]
[0, 2, 4, 6, 8]
[0, 6]
55
```

```python
a = {x:x*100 for x in range(10)}
print(a)
```

```
{0: 0, 1: 100, 2: 200, 3: 300, 4: 400, 5: 500,
```

```python
a = {1, 2, 3}
b = {1, 4, 7}
print(a&b)
print(a|b)
print(a^b)
print(a-b)
print(b-a)
```

```
{1}
{1, 2, 3, 4, 7}
{2, 3, 4, 7}
{2, 3}
{4, 7}
```

```python
def show():
    print('test')

def hello(people):
    print('hello {}'.format(people))
show()
hello('john')
```

```
test
hello john
```

```python
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age
    def hello(self):
        print('hello {}'.format(self.name))
    def show(self):
        print('{} is {} years old'.format(self.name, self.age))

p = Person('jack', 18)
p.hello()
p.show()
```

```
hello jack
jack is 18 years old
```

```python
class Person:
    pass

def constructor(self, name, age):
    self.name = name
    self.age = age
def hello(self):
    print('hello {}'.format(self.name))
def show(self):
    print('{} is {} years old'.format(self.name, self.age))

Person.__init__ = constructor
Person.hello = hello
Person.show = show

p = Person('jack', 18)
p.hello()
p.show()
```

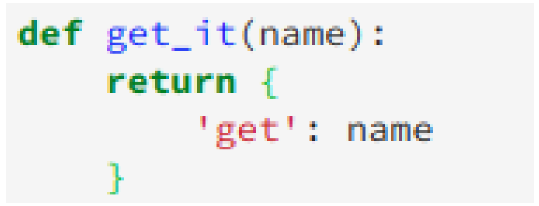
```
hello jack
jack is 18 years old
```

Save the Cheerleader, save the World

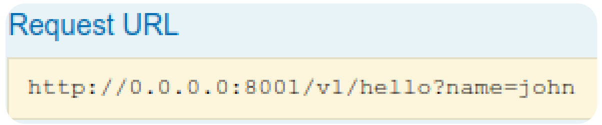
Who saves the CRUD Boy?

HTTP API

- 设计API
- 实现业务逻辑
- 测试相应的逻辑
- 编写对应的文档
- 部署

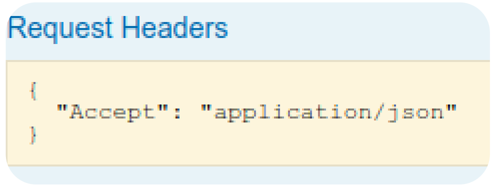


```python
def get_it(name):
    return {
        'get': name
    }
```



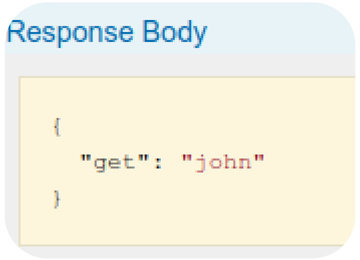
Request URL

```
http://0.0.0.0:8001/v1/hello?name=john
```



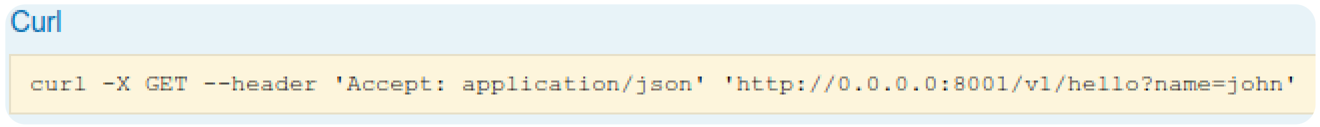
Request Headers

```
{
  "Accept": "application/json"
}
```



Response Body

```
{
  "get": "john"
}
```



Curl

```
curl -X GET --header 'Accept: application/json' 'http://0.0.0.0:8001/v1/hello?name=john'
```

HTTP Restful API is Easy

- Get
- Post
- Delete
- Put
- .....

- Get
- Post

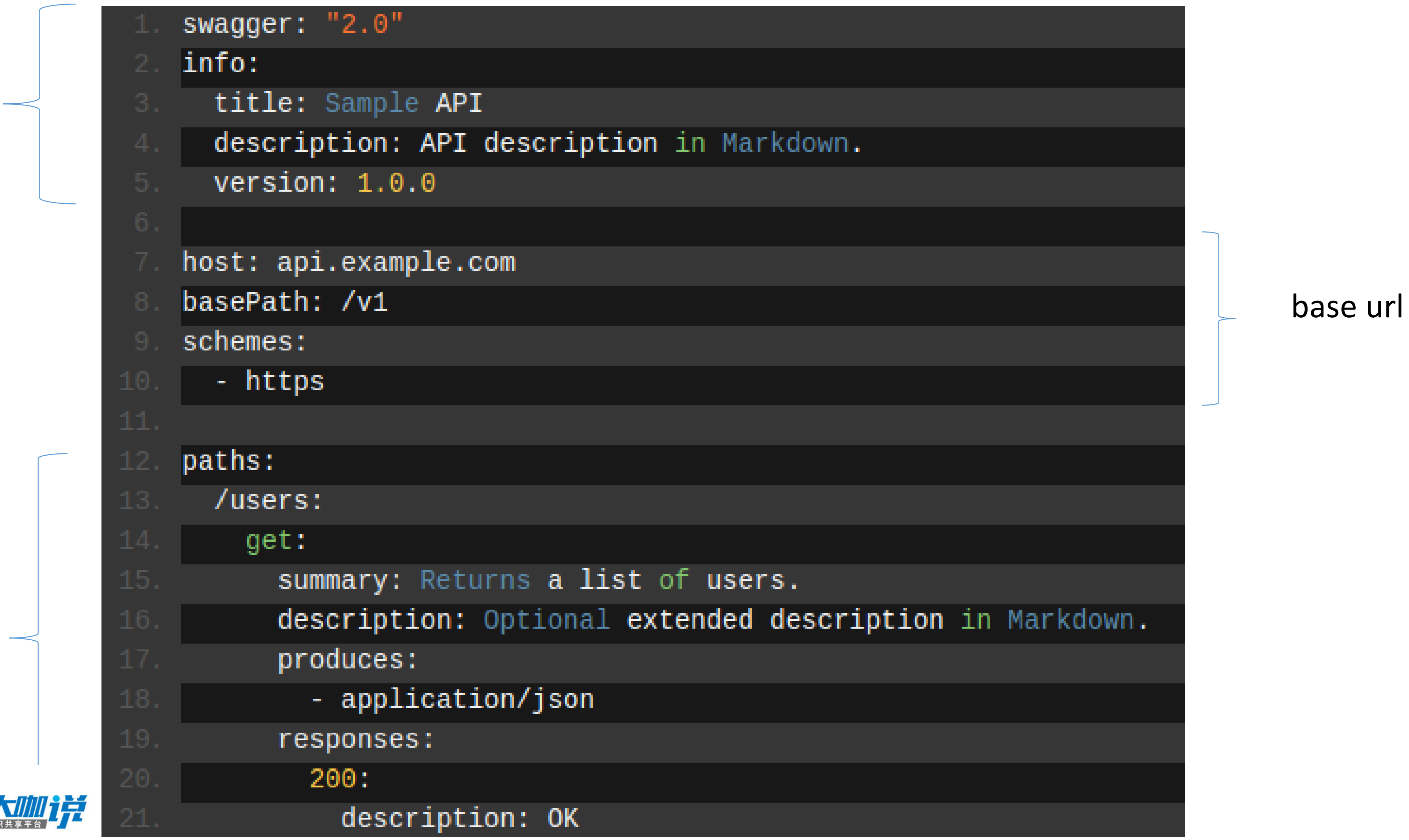
While the biz logics depends,

this tutorial is only about the API ...

有没有统一的规范?

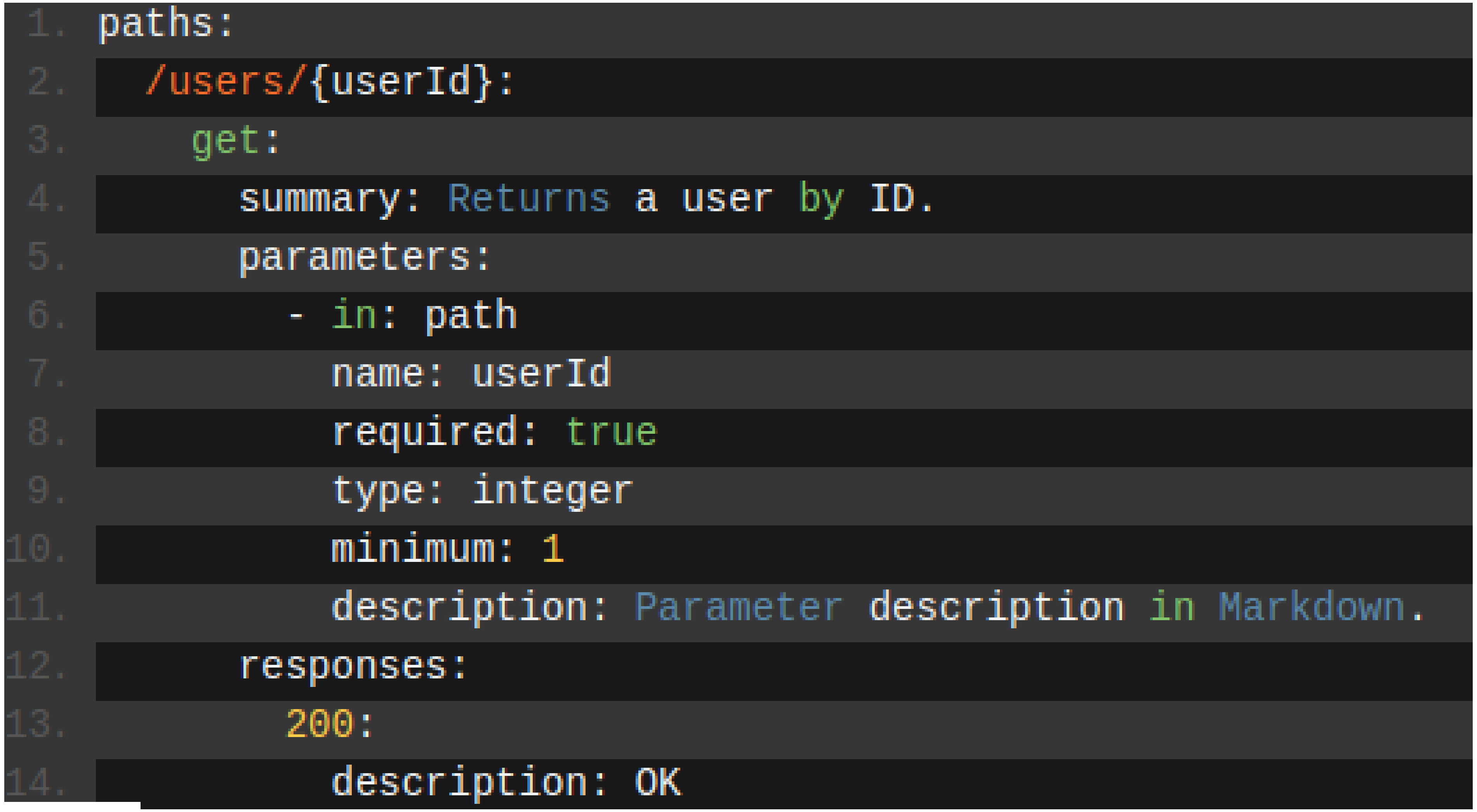
- Swagger
- Raml
- Blueprint

Demo of OAI(Swagger) 2.0



```yaml
swagger: "2.0"
info:
  title: Sample API
  description: API description in Markdown.
  version: 1.0.0

host: api.example.com
basePath: /v1
schemes:
  - https

paths:
  /users:
    get:
      summary: Returns a list of users.
      description: Optional extended description in Markdown.
      produces:
        - application/json
      responses:
        200:
          description: OK
```

add parameters



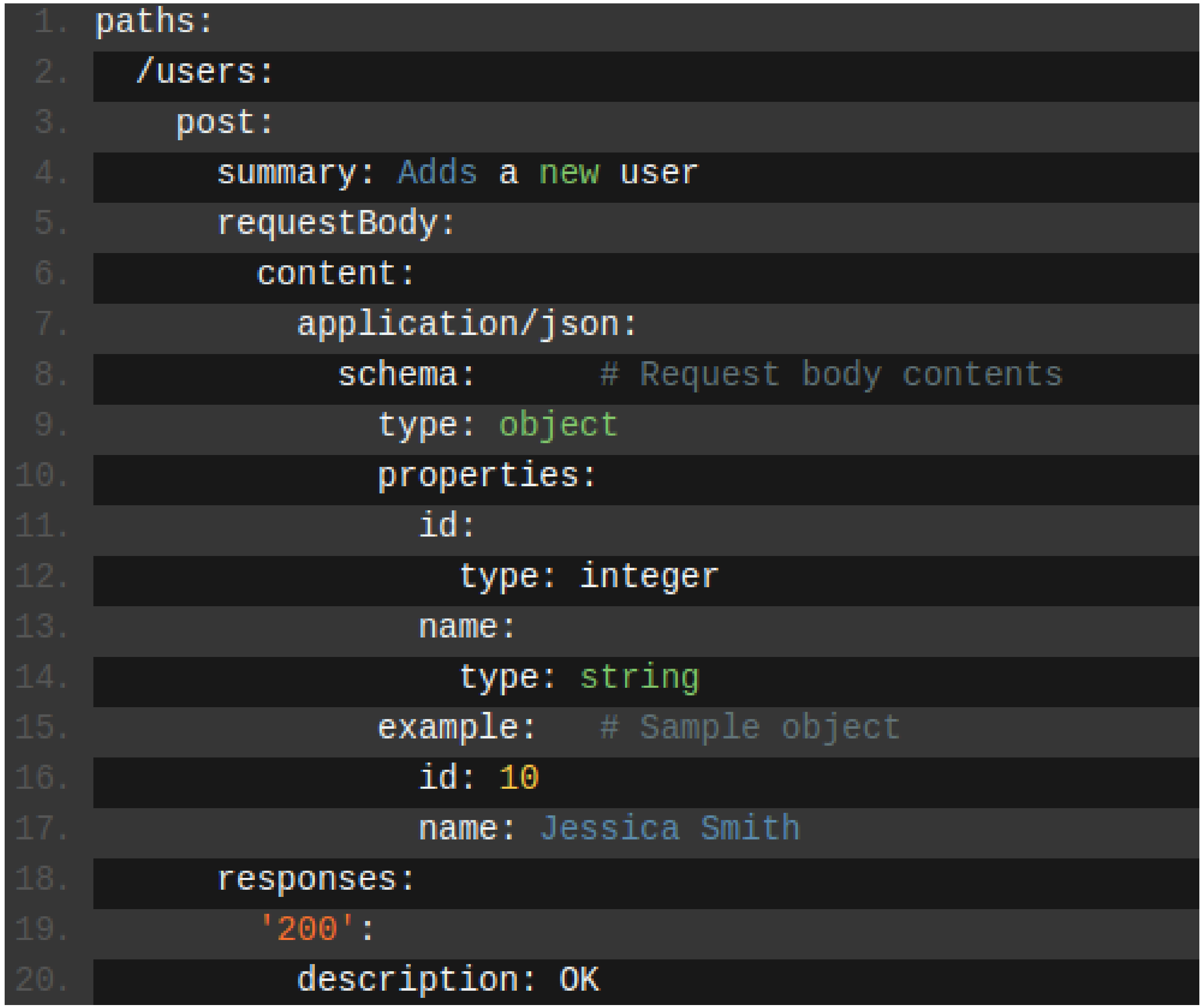
```yaml
paths:
  /users/{userId}:
    get:
      summary: Returns a user by ID.
      parameters:
        - in: path
          name: userId
          required: true
          type: integer
          minimum: 1
          description: Parameter description in Markdown.
      responses:
        200:
          description: OK
```

paths

params

post example

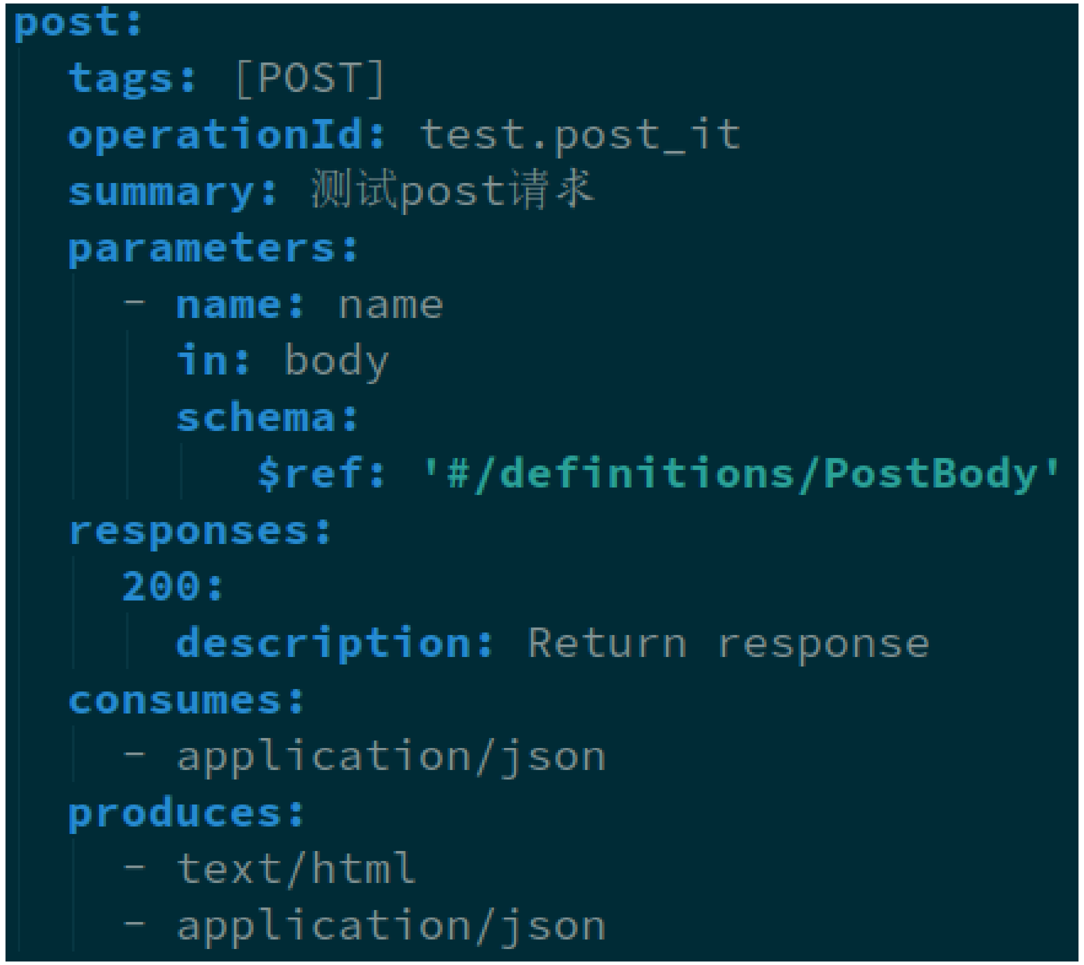
paths



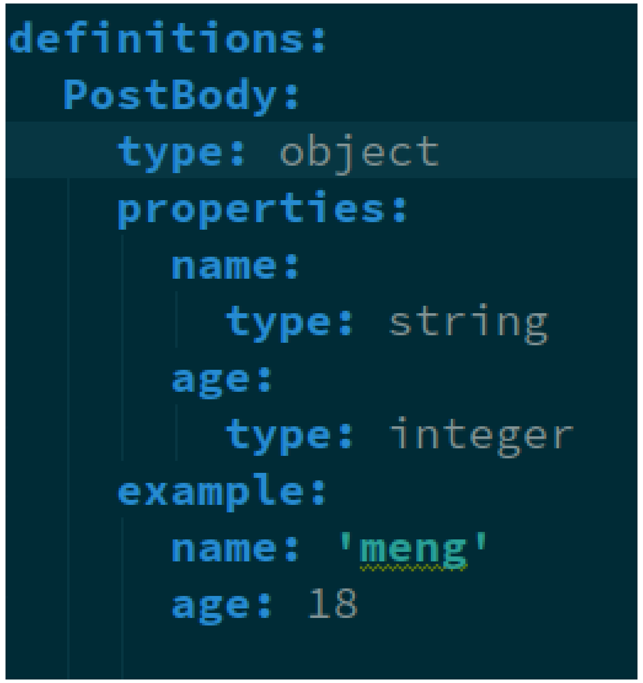
```yaml
paths:
  /users:
    post:
      summary: Adds a new user
      requestBody:
        content:
          application/json:
            schema:      # Request body contents
              type: object
              properties:
                id:
                  type: integer
                name:
                  type: string
              example:   # Sample object
                id: 10
                name: Jessica Smith
      responses:
        '200':
          description: OK
```

params

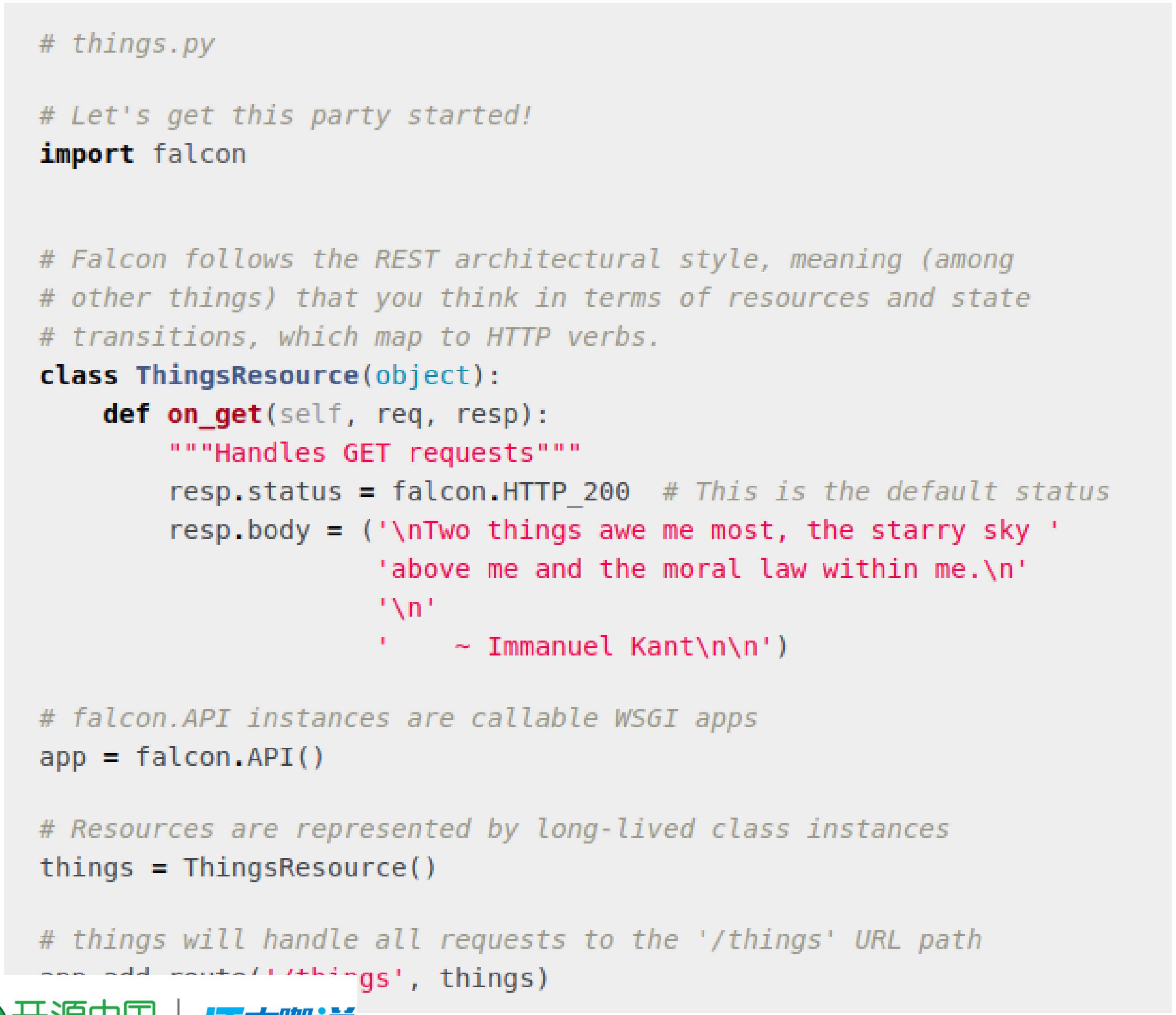
post with ref obj



```
post:
  tags: [POST]
  operationId: test.post_it
  summary: 测试post请求
  parameters:
    - name: name
      in: body
      schema:
         $ref: '#/definitions/PostBody'
  responses:
    200:
      description: Return response
  consumes:
    - application/json
  produces:
    - text/html
    - application/json
```



```
definitions:
  PostBody:
    type: object
    properties:
      name:
        type: string
      age:
        type: integer
    example:
      name: 'meng'
      age: 18
```

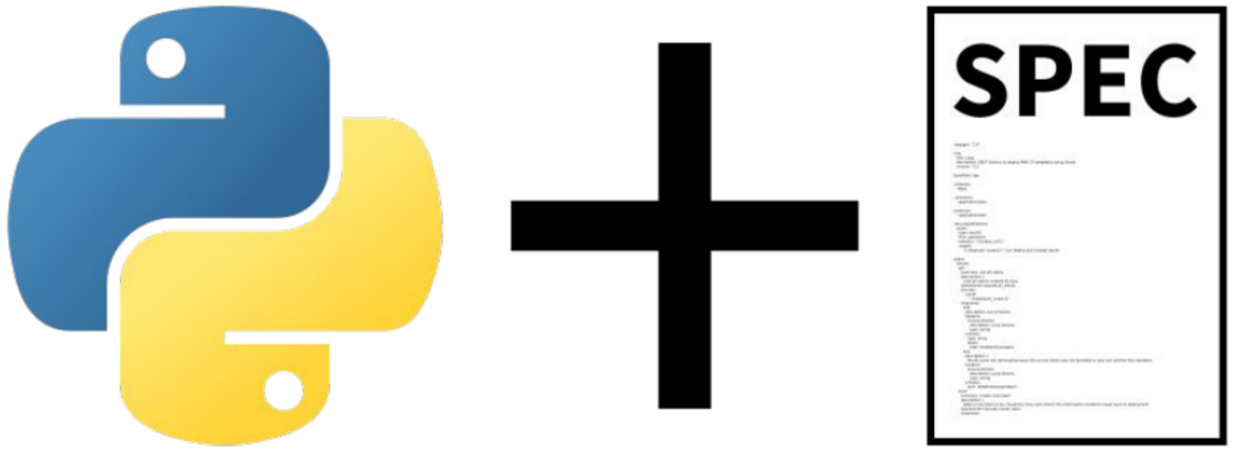


```python
# things.py

# Let's get this party started!
import falcon


# Falcon follows the REST architectural style, meaning (among
# other things) that you think in terms of resources and state
# transitions, which map to HTTP verbs.
class ThingsResource(object):
    def on_get(self, req, resp):
        """Handles GET requests"""
        resp.status = falcon.HTTP_200  # This is the default status
        resp.body = ('\nTwo things awe me most, the starry sky '
                     'above me and the moral law within me.\n'
                     '\n'
                     '    ~ Immanuel Kant\n\n')

# falcon.API instances are callable WSGI apps
app = falcon.API()

# Resources are represented by long-lived class instances
things = ThingsResource()

# things will handle all requests to the '/things' URL path
app.add_route('/things', things)
```

Introduce the Falcon Framework

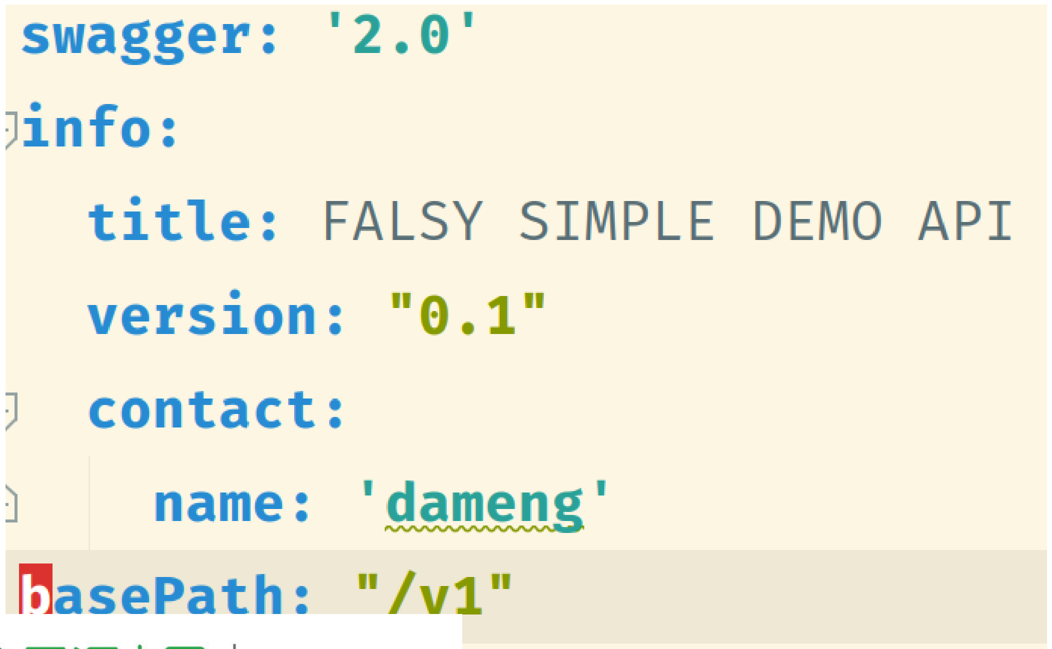
Fal.s.y

- Falcon(as the backend)
- swagger
- yml

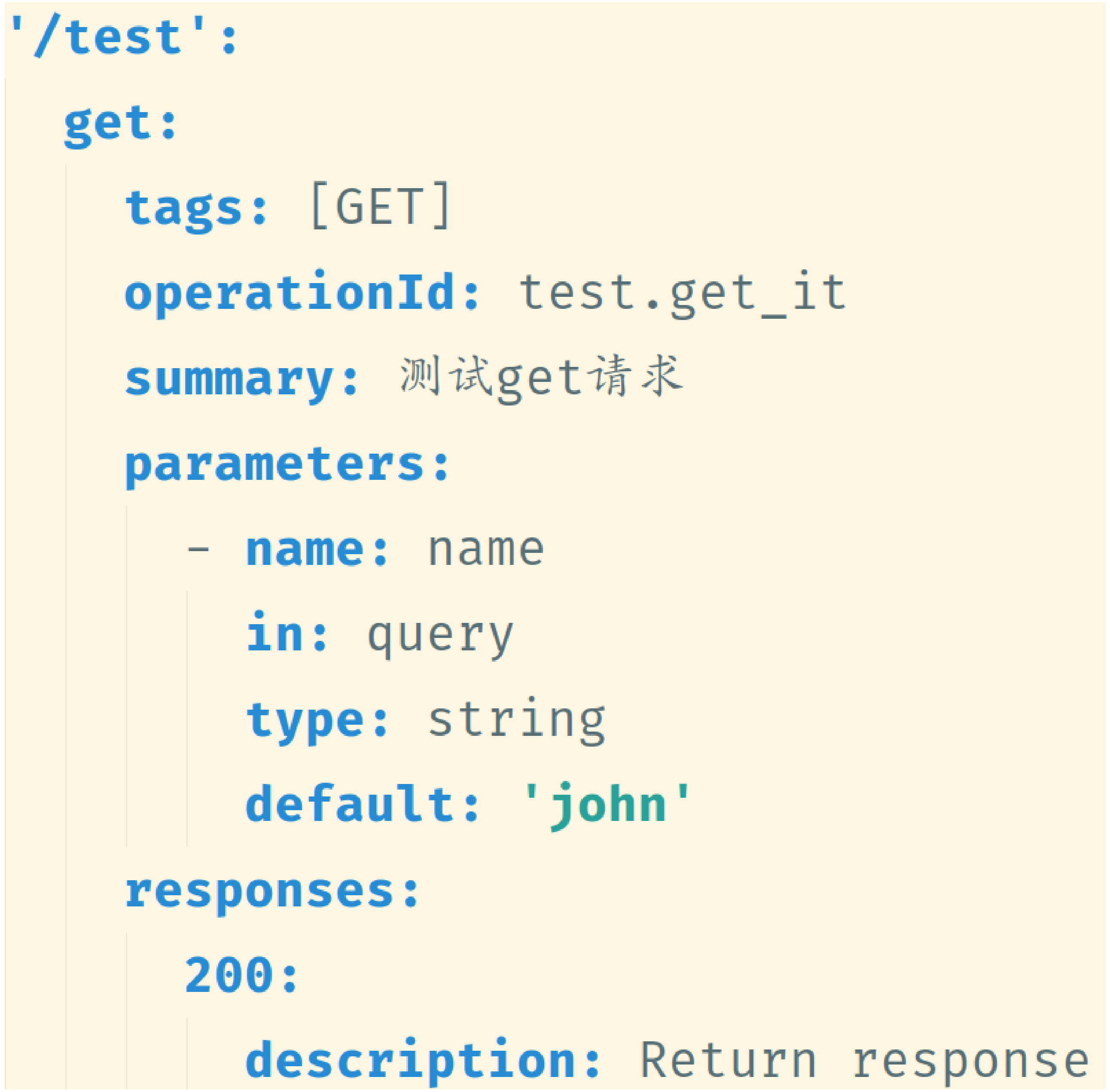


Step 1

- 编写Spec 文件



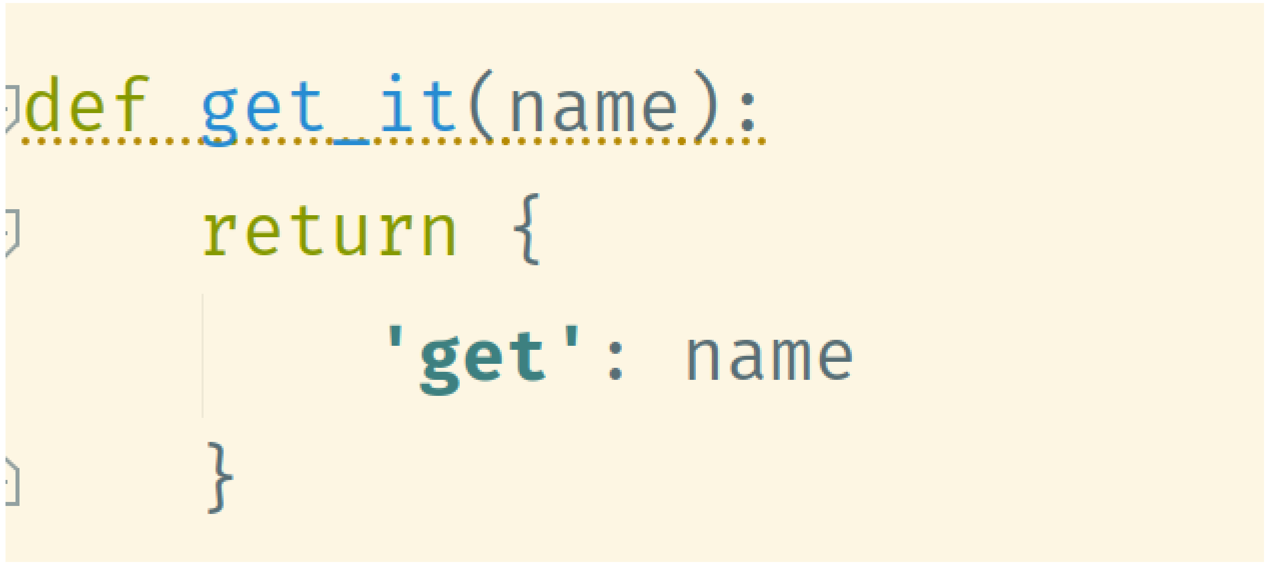
```
swagger: '2.0'
info:
  title: FALSY SIMPLE DEMO API
  version: "0.1"
  contact:
    name: 'dameng'
basePath: "/v1"
```



```
'/test':
  get:
    tags: [GET]
    operationId: test.get_it
    summary: 测试get请求
    parameters:
      - name: name
        in: query
        type: string
        default: 'john'
    responses:
      200:
        description: Return response
```

Step 2

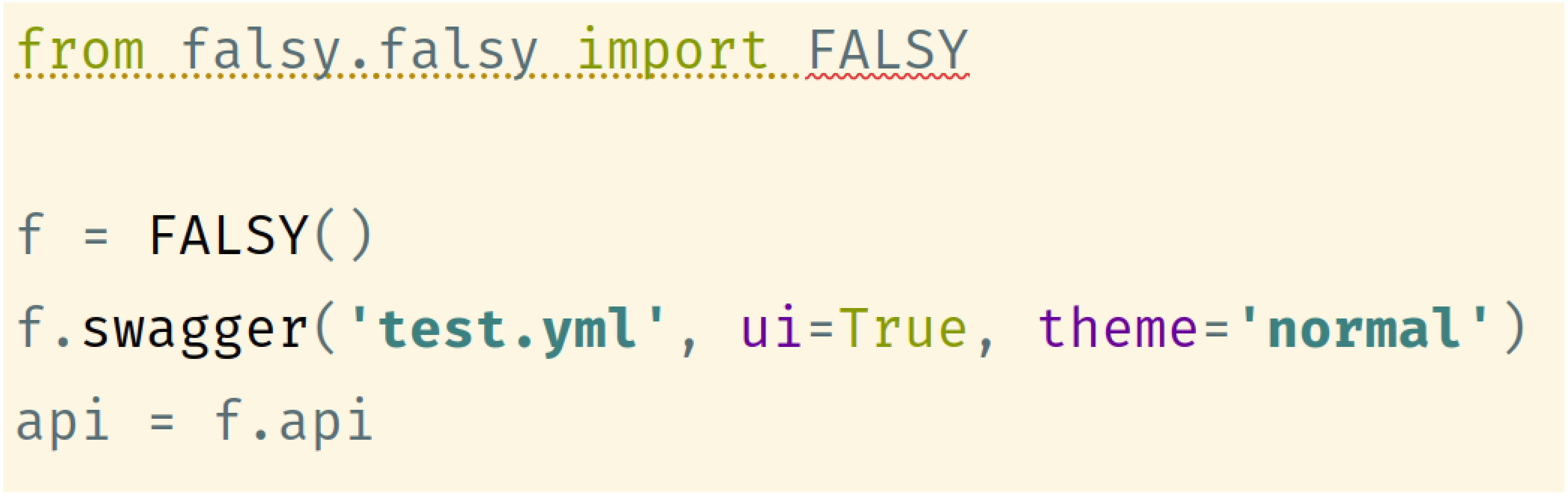
- 编写python handler



```
def get_it(name):
    return {
        'get': name
    }
```

Step 3

- 将spec和python函数绑定



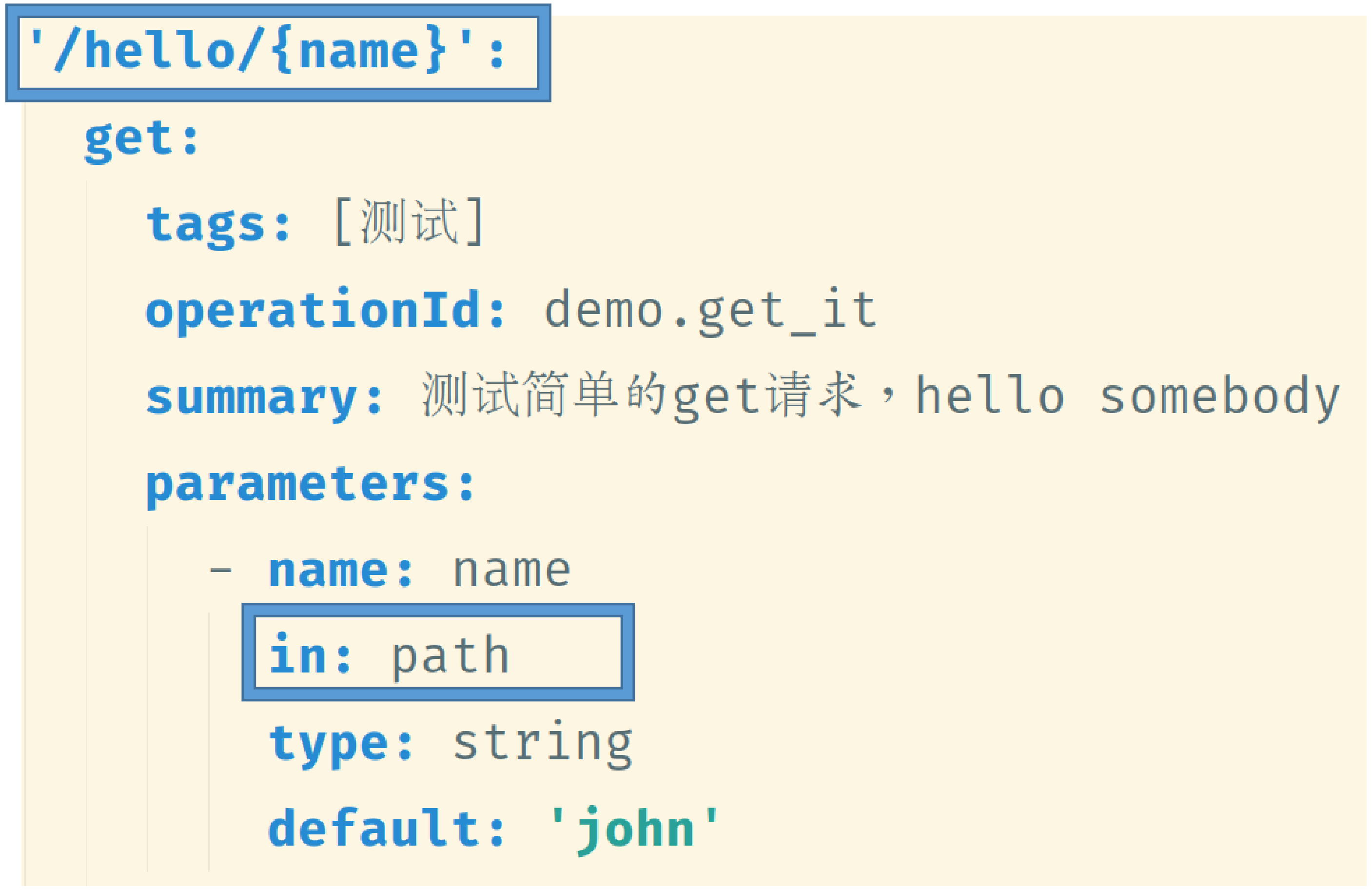
```
from falsy.falsy import FALSY


f = FALSY()
f.swagger('test.yml', ui=True, theme='normal')
api = f.api
```

Step 4

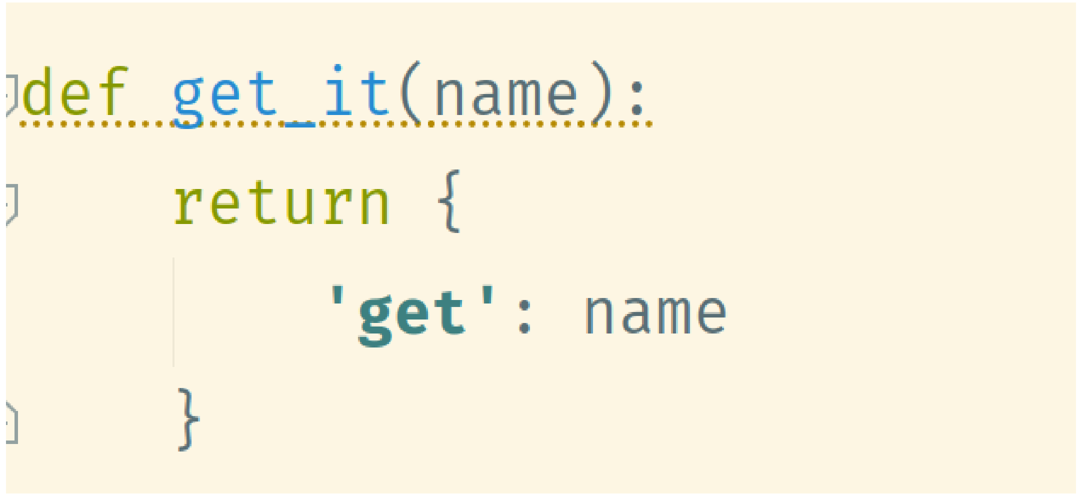
- 运行server
- gunicorn serve:api --bind 127.0.0.1:8181

需求变更了？

- 如果需要将传入的参数从querymeter变为pathmeter？



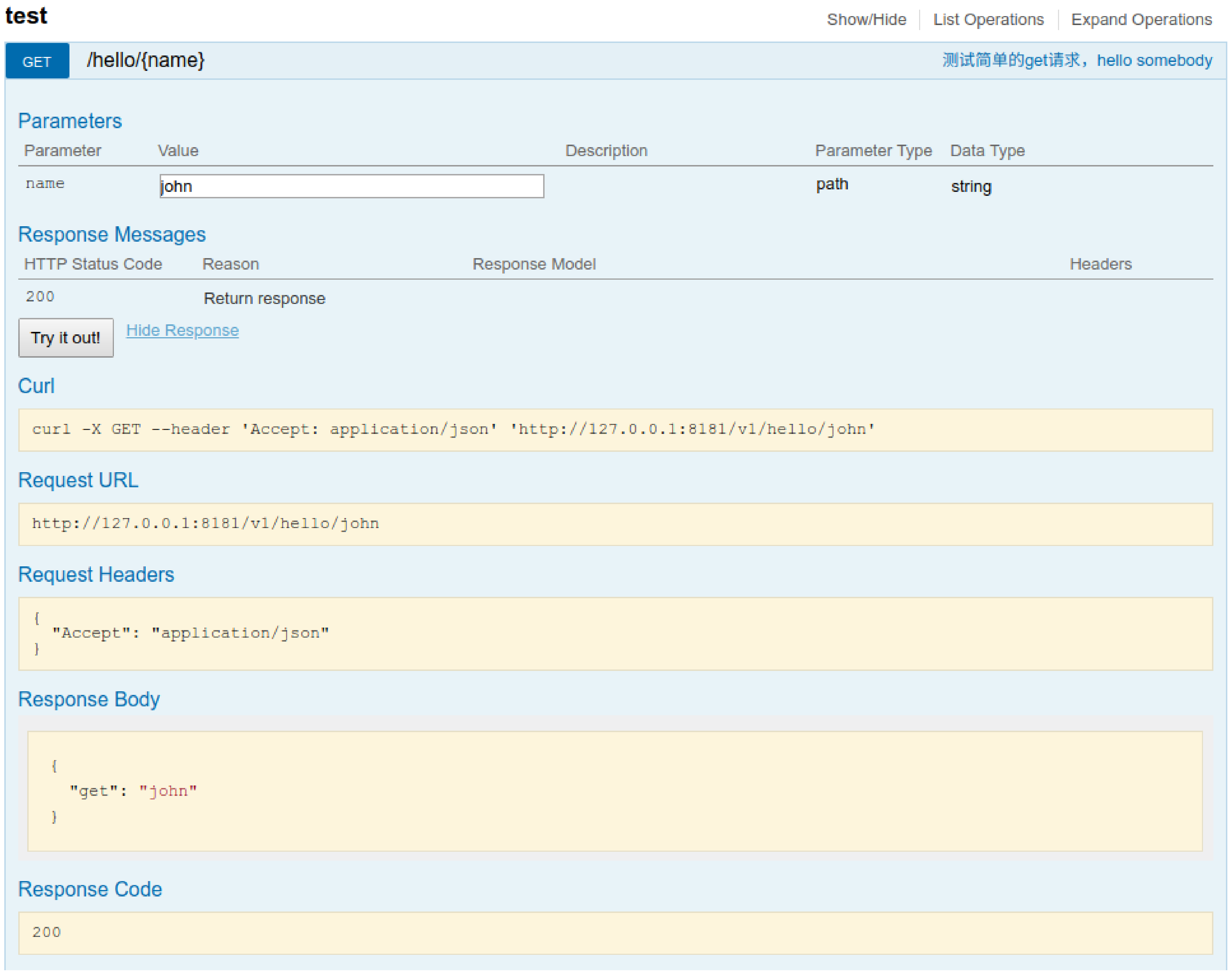
```
'/hello/{name}':
  get:
    tags: [测试]
    operationId: demo.get_it
    summary: 测试简单的get请求，hello somebody
    parameters:
      - name: name
        in: path
        type: string
        default: 'john'
```



```
def get_it(name):
    return {
        'get': name
    }
```

代码不变

UI



GET /hello/{name} 测试简单的get请求，hello somebody

Parameters

Parameter	Value	Description	Parameter Type	Data Type
name	john		path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	Return response		

Curl

```
curl -X GET --header 'Accept: application/json' 'http://127.0.0.1:8181/v1/hello/john'
```

Request URL

```
http://127.0.0.1:8181/v1/hello/john
```

Request Headers

```
{
  "Accept": "application/json"
}
```

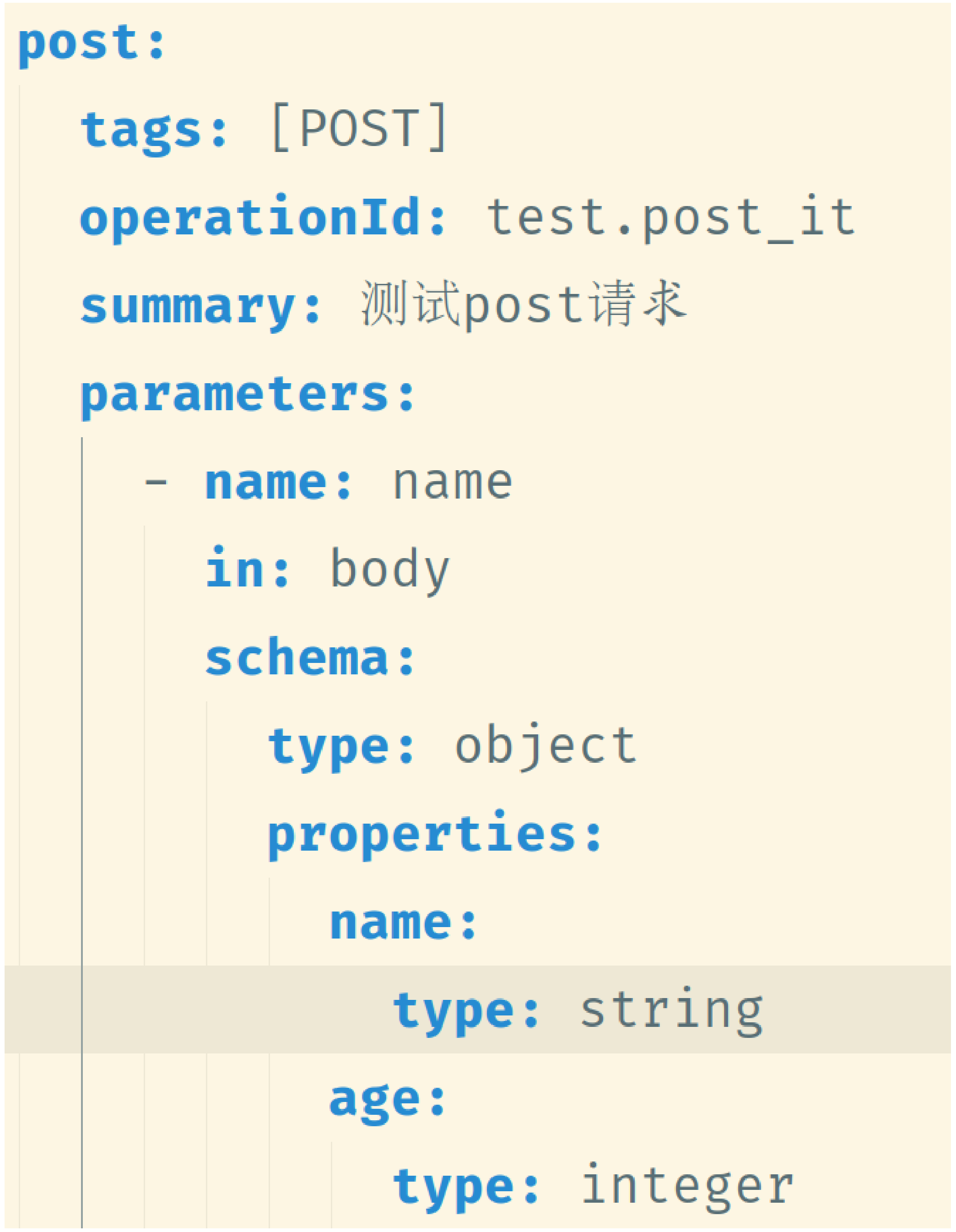
Response Body

```
{
  "get": "john"
}
```

Response Code

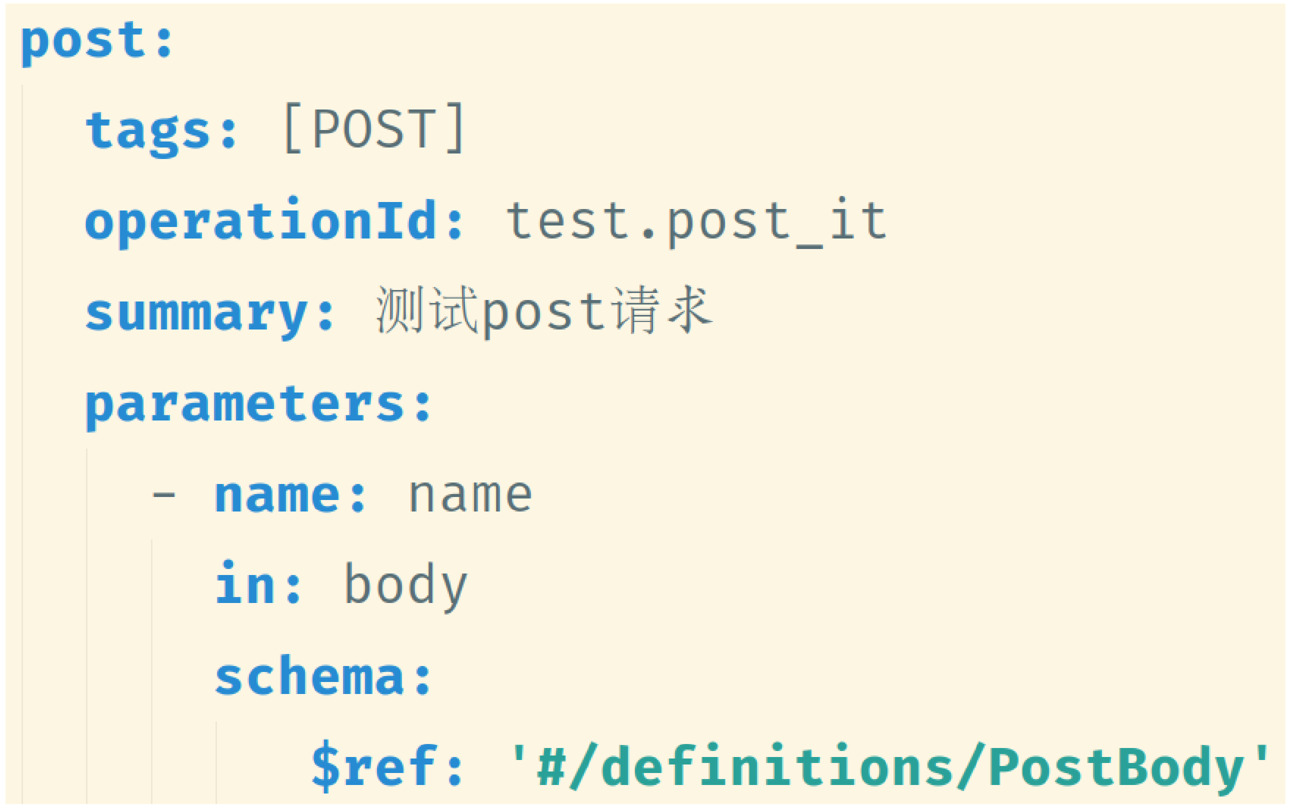
```
200
```

不仅仅是get请求，
其它的
带有body的请求呢？

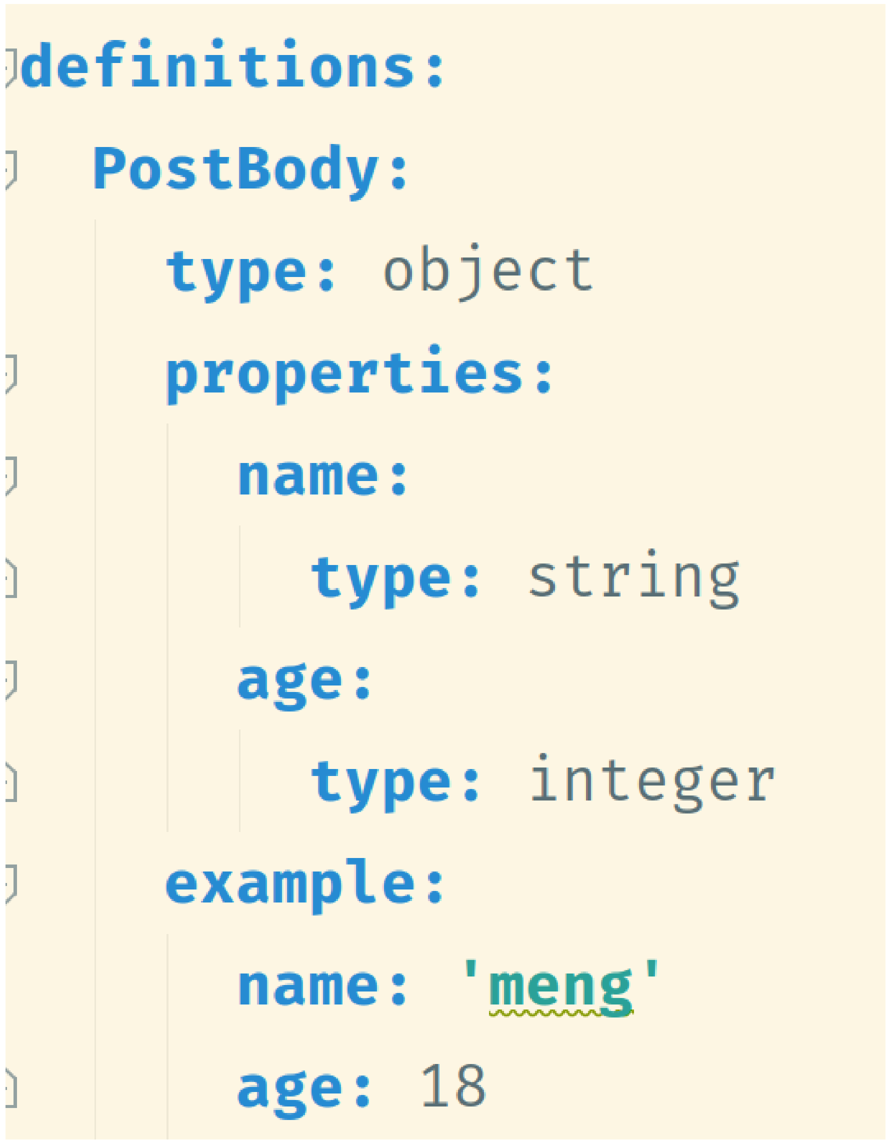


```yaml
post:
  tags: [POST]
  operationId: test.post_it
  summary: 测试post请求
  parameters:
    - name: name
      in: body
      schema:
        type: object
        properties:
          name:
            type: string
          age:
            type: integer
```

body中数据的另一种呈现方式



```yaml
post:
  tags: [POST]
  operationId: test.post_it
  summary: 测试post请求
  parameters:
    - name: name
      in: body
      schema:
        $ref: '#/definitions/PostBody'
```



```yaml
definitions:
  PostBody:
    type: object
    properties:
      name:
        type: string
      age:
        type: integer
    example:
      name: 'meng'
      age: 18
```

讲到这里。。。

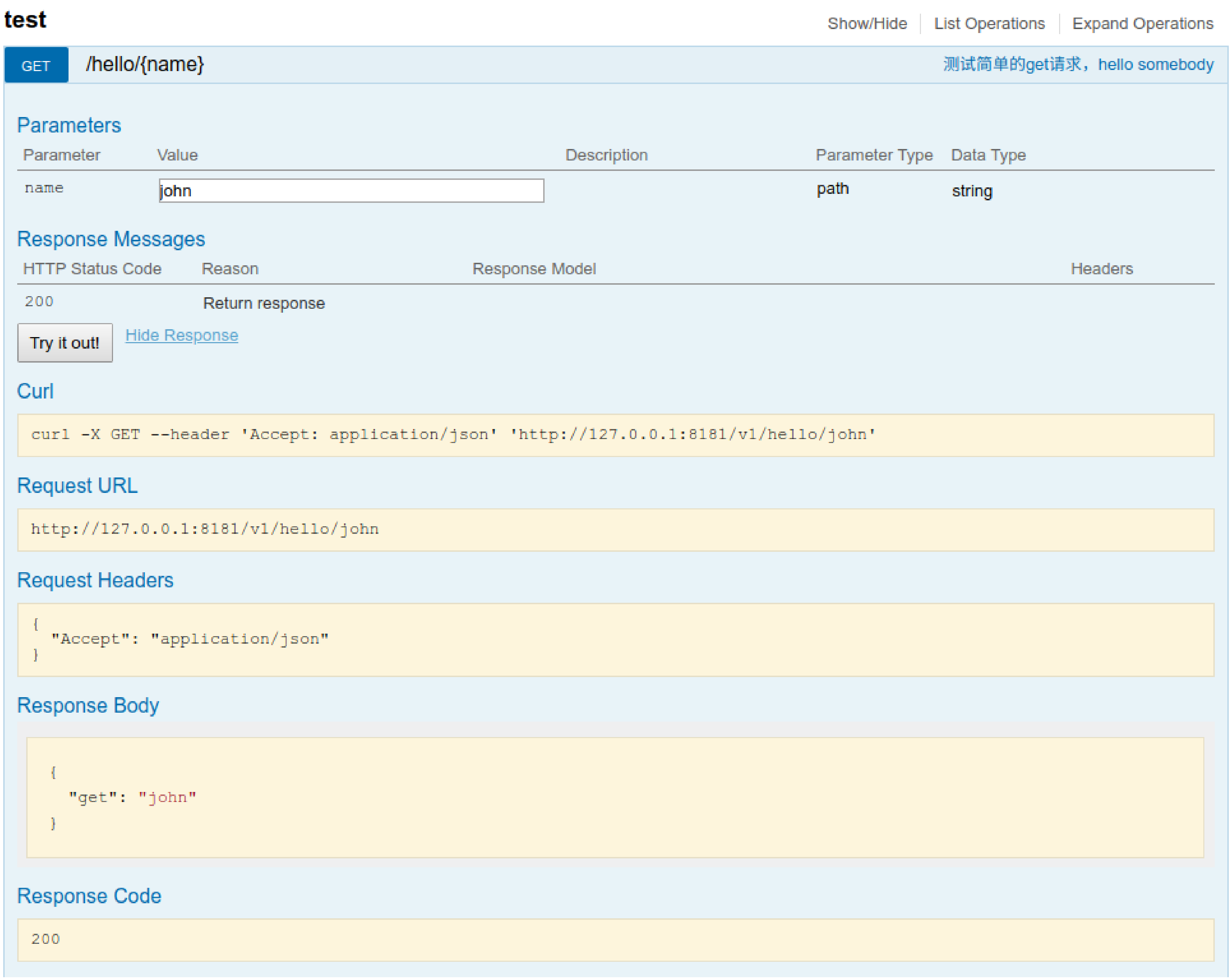
- 基本的CRUD已经可以使用Falsy完成
- 但是这还远远不够。。。

世界是丰富多彩的

```
f = FALSY()
f.swagger('test.yml', ui=True, theme='normal')
api = f.api
```

- normal
- impress
- material

Normal UI



test — Show/Hide | List Operations | Expand Operations

GET /hello/{name} — 测试简单的get请求，hello somebody

Parameters

Parameter	Value	Description	Parameter Type	Data Type
name	john		path	string

Response Messages

HTTP Status Code	Reason	Response Model	Headers
200	Return response		

Try it out! Hide Response

Curl

```
curl -X GET --header 'Accept: application/json' 'http://127.0.0.1:8181/v1/hello/john'
```

Request URL

```
http://127.0.0.1:8181/v1/hello/john
```

Request Headers

```
{
  "Accept": "application/json"
}
```

Response Body

```
{
  "get": "john"
}
```

Response Code

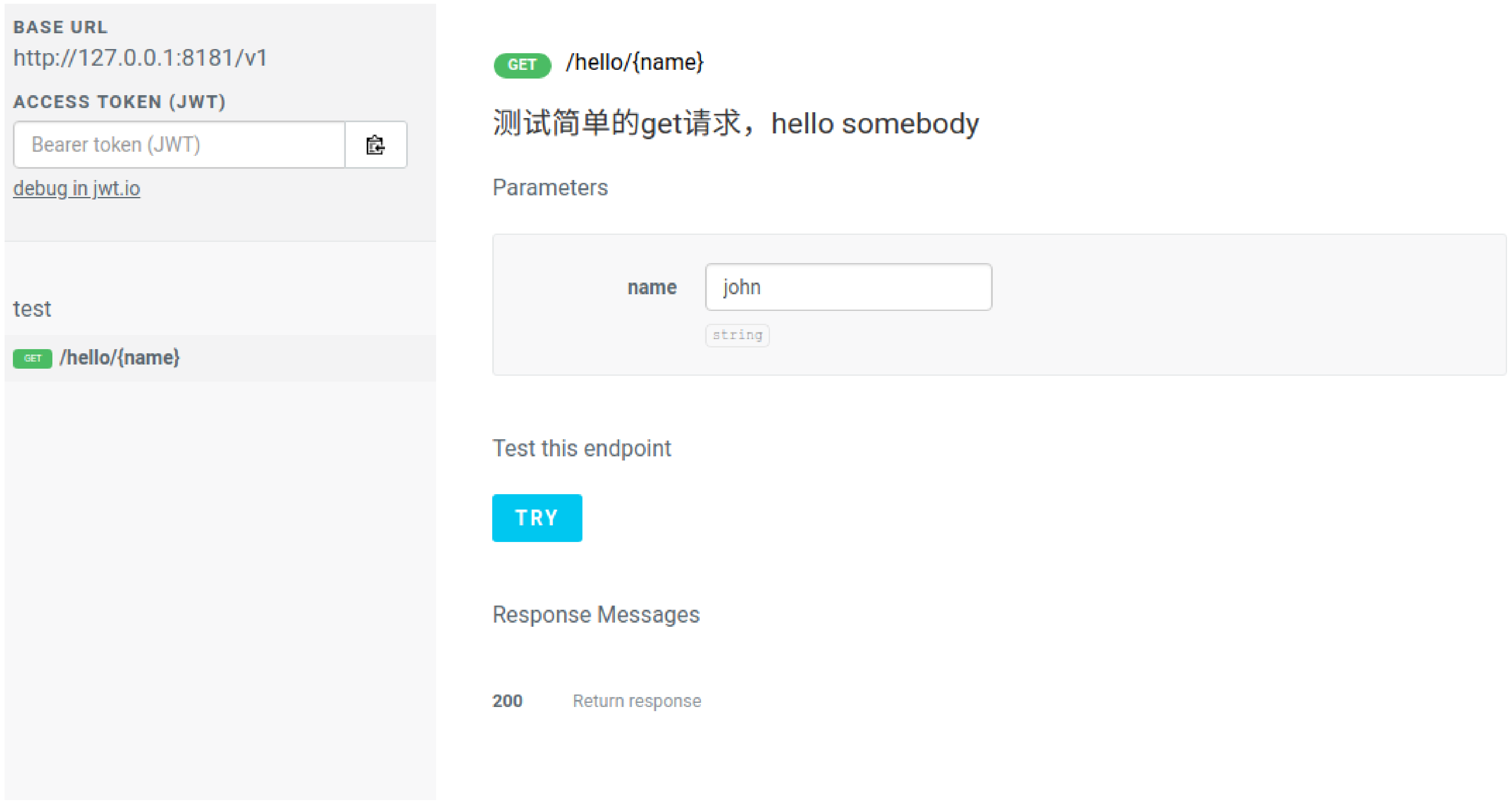
```
200
```

开源中国 oschina.net | IT大咖说 知识共享平台

Bonus

- Curl script
- multi-language support
- vertical view

Impress UI



BASE URL
http://127.0.0.1:8181/v1

ACCESS TOKEN (JWT)

Bearer token (JWT)

debug in jwt.io

test

GET /hello/{name}

GET /hello/{name}

测试简单的get请求，hello somebody

Parameters

name: john
string

Test this endpoint

TRY

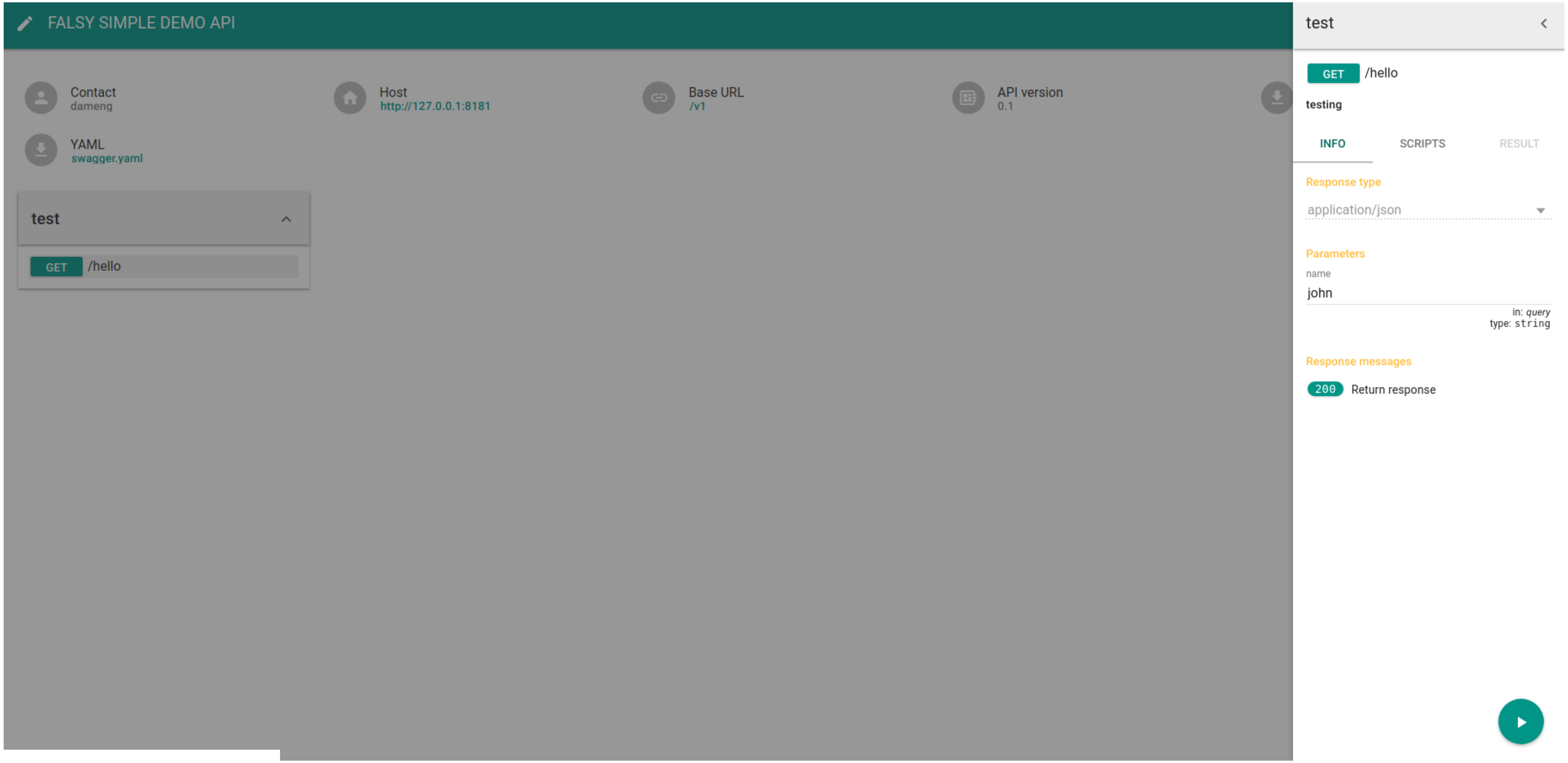
Response Messages

200 Return response

Bonus

- most comfortable UI
- json highlights
- popup window
- horizontal view

Material UI



FALSY SIMPLE DEMO API

Contact
dameng

Host
http://127.0.0.1:8181

Base URL
/v1

API version
0.1

YAML
swagger.yaml

test

GET /hello

test

GET /hello

testing

Response type
application/json

Parameters
name
john
in: *query*
type: `string`

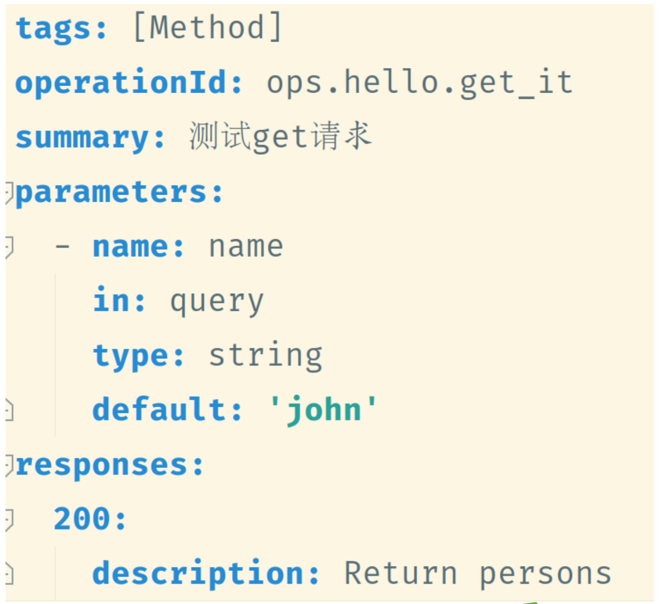
Response messages
200 Return response

Bonus

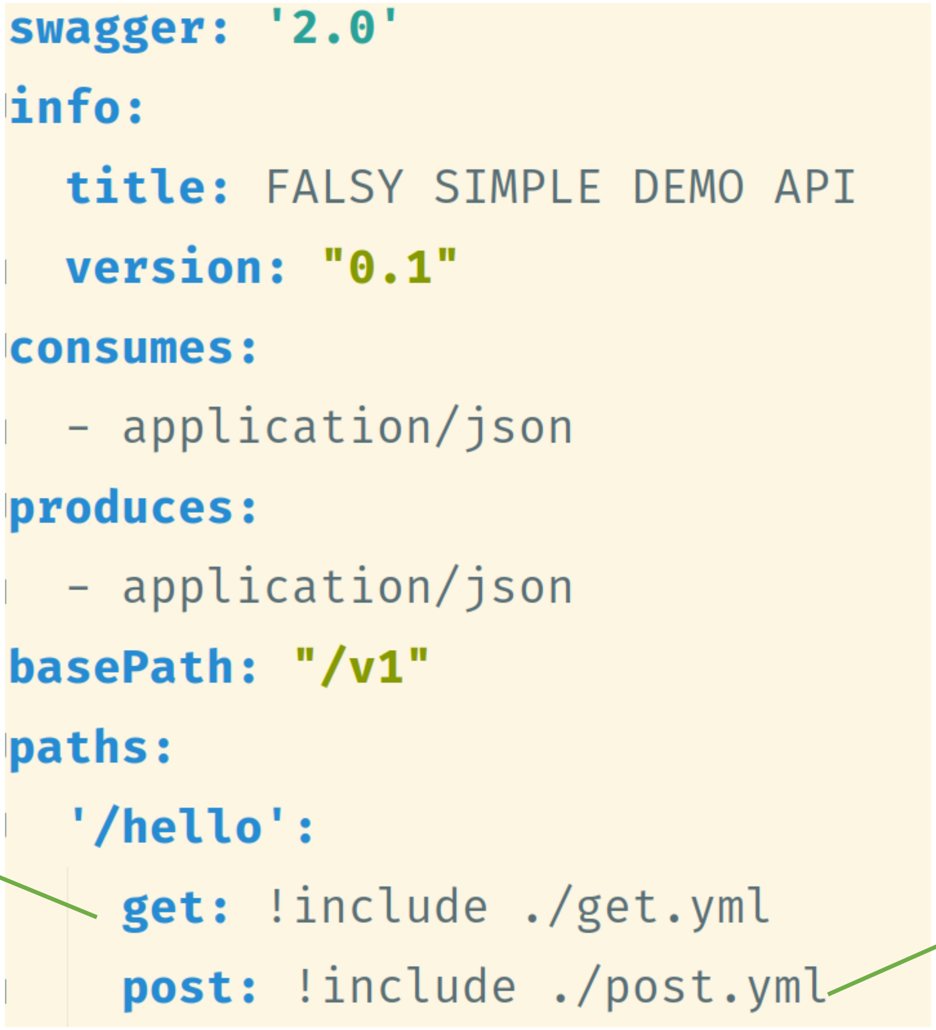
- Angular script
- material style

当我们的api变得越来越多

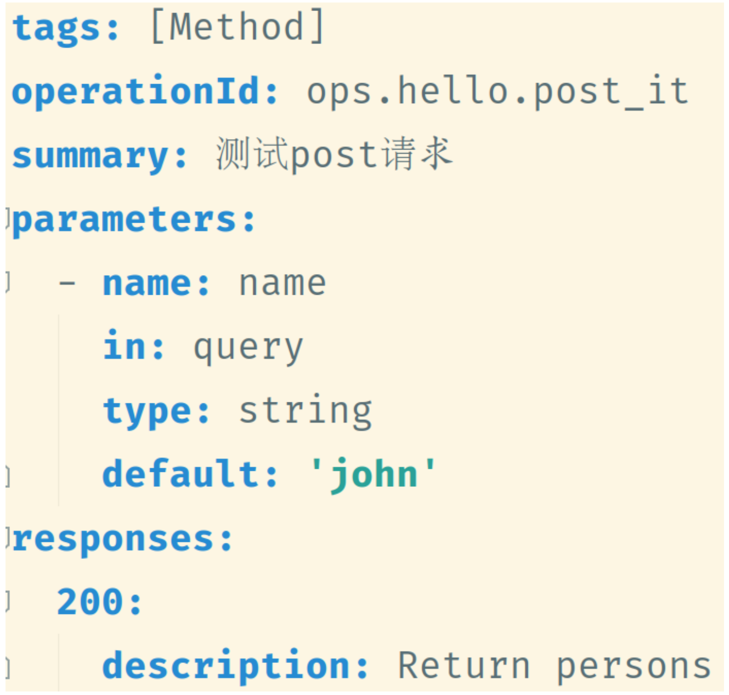
- 对spec文件进行拆分



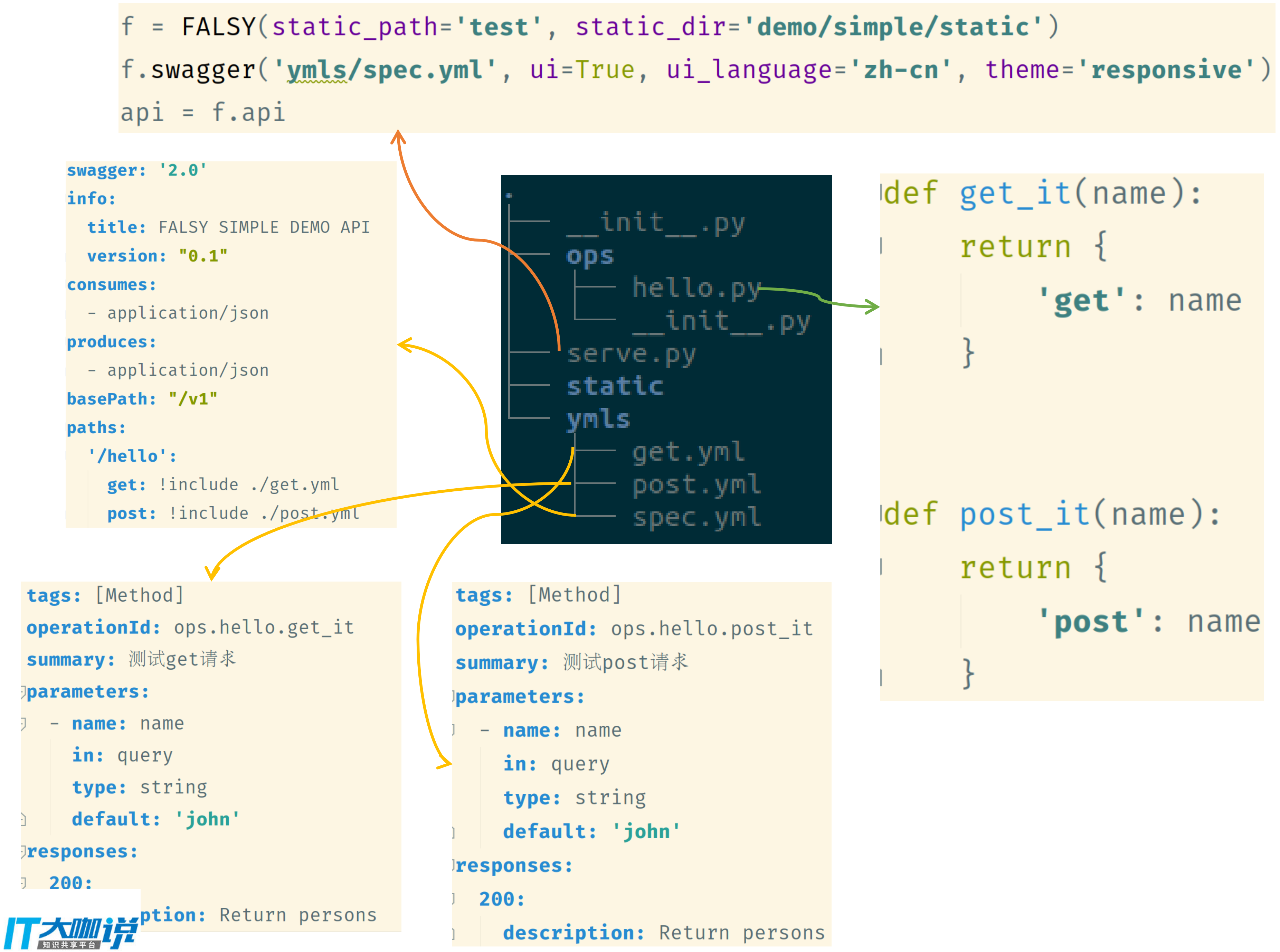
```yaml
tags: [Method]
operationId: ops.hello.get_it
summary: 测试get请求
parameters:
  - name: name
    in: query
    type: string
    default: 'john'
responses:
  200:
    description: Return persons
```



```yaml
swagger: '2.0'
info:
  title: FALSY SIMPLE DEMO API
  version: "0.1"
consumes:
  - application/json
produces:
  - application/json
basePath: "/v1"
paths:
  '/hello':
    get: !include ./get.yml
    post: !include ./post.yml
```



```yaml
tags: [Method]
operationId: ops.hello.post_it
summary: 测试post请求
parameters:
  - name: name
    in: query
    type: string
    default: 'john'
responses:
  200:
    description: Return persons
```



```python
f = FALSY(static_path='test', static_dir='demo/simple/static')
f.swagger('ymls/spec.yml', ui=True, ui_language='zh-cn', theme='responsive')
api = f.api
```

```yaml
swagger: '2.0'
info:
  title: FALSY SIMPLE DEMO API
  version: "0.1"
consumes:
  - application/json
produces:
  - application/json
basePath: "/v1"
paths:
  '/hello':
    get: !include ./get.yml
    post: !include ./post.yml
```

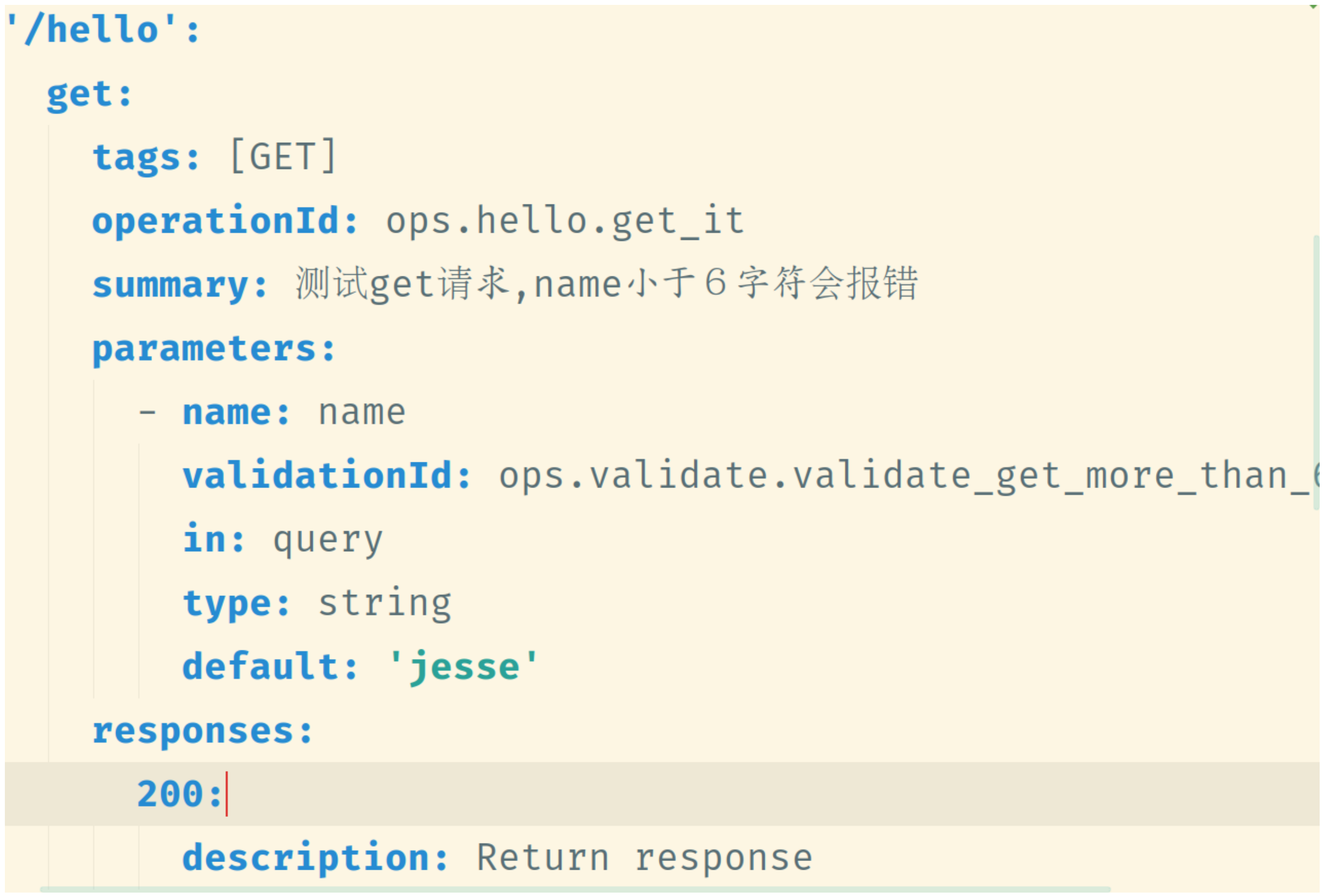
```
.
├── __init__.py
├── ops
│   ├── hello.py
│   └── __init__.py
├── serve.py
├── static
└── ymls
    ├── get.yml
    ├── post.yml
    └── spec.yml
```

```python
def get_it(name):
    return {
        'get': name
    }


def post_it(name):
    return {
        'post': name
    }
```

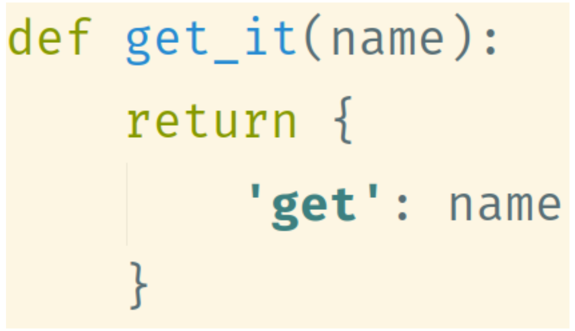
```yaml
tags: [Method]
operationId: ops.hello.get_it
summary: 测试get请求
parameters:
  - name: name
    in: query
    type: string
    default: 'john'
responses:
  200:
      ption: Return persons
```

```yaml
tags: [Method]
operationId: ops.hello.post_it
summary: 测试post请求
parameters:
  - name: name
    in: query
    type: string
    default: 'john'
responses:
  200:
    description: Return persons
```

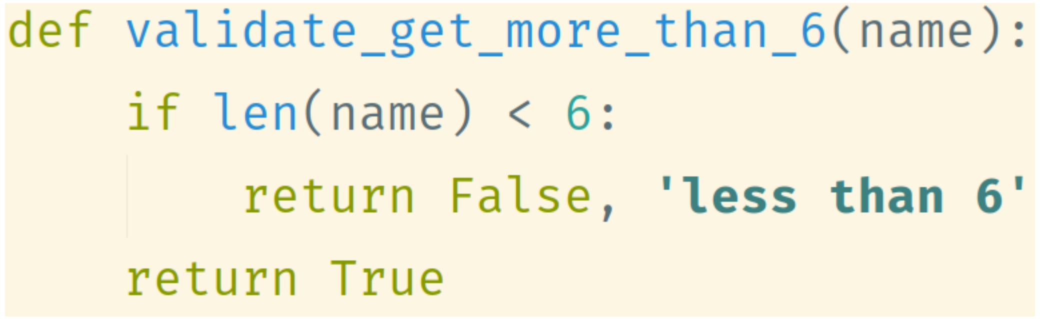
validationId



```yaml
'/hello':
  get:
    tags: [GET]
    operationId: ops.hello.get_it
    summary: 测试get请求,name小于6字符会报错
    parameters:
      - name: name
        validationId: ops.validate.validate_get_more_than_
        in: query
        type: string
        default: 'jesse'
    responses:
      200:
        description: Return response
```

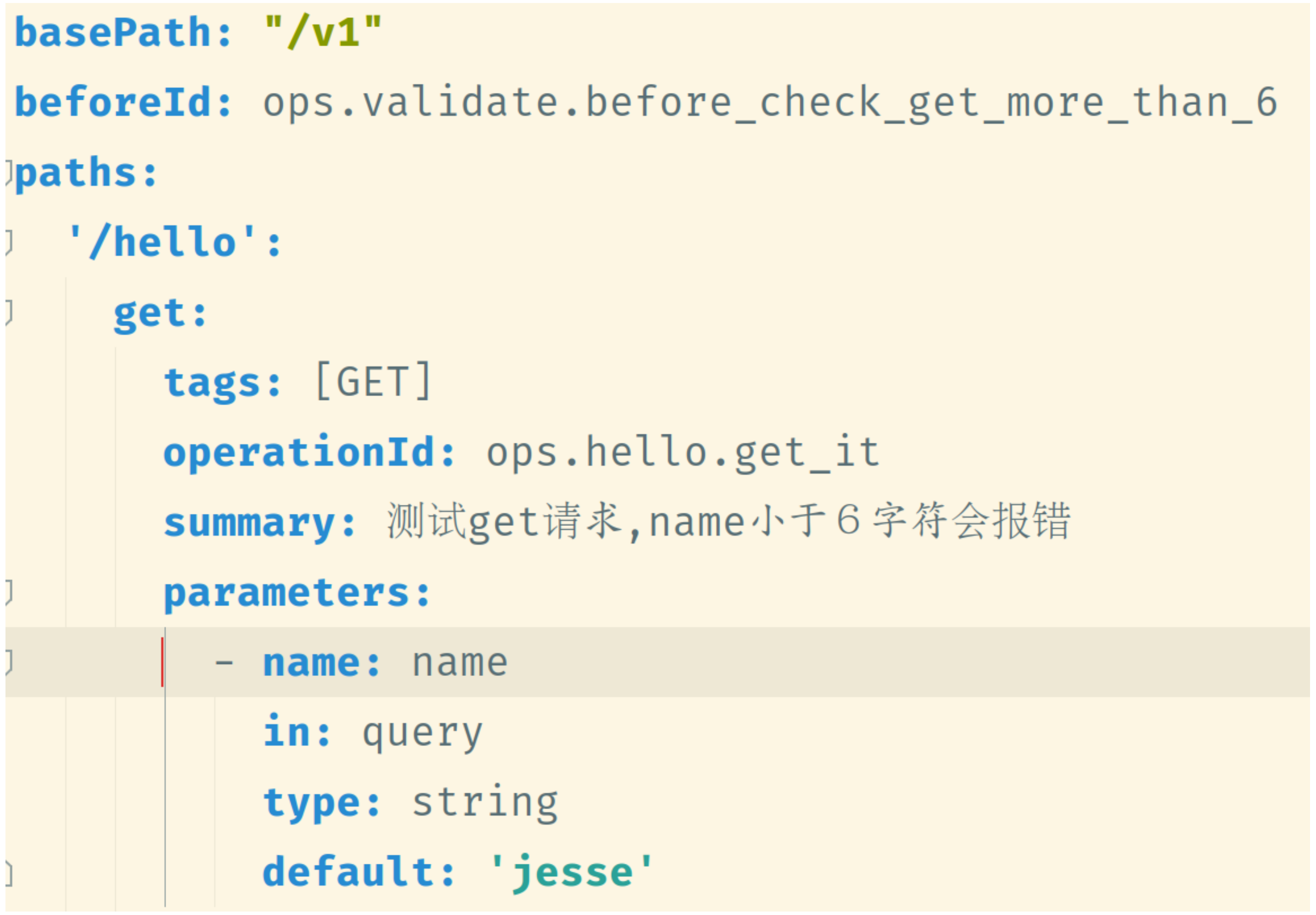


```python
def get_it(name):
    return {
        'get': name
    }
```

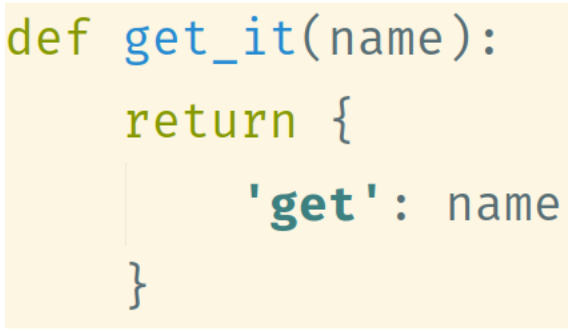


```python
def validate_get_more_than_6(name):
    if len(name) < 6:
        return False, 'less than 6'
    return True
```

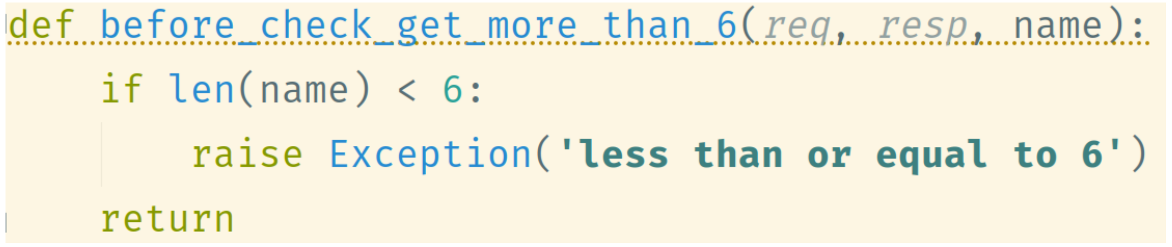
beforeId



```yaml
basePath: "/v1"
beforeId: ops.validate.before_check_get_more_than_6
paths:
  '/hello':
    get:
      tags: [GET]
      operationId: ops.hello.get_it
      summary: 测试get请求,name小于6字符会报错
      parameters:
        - name: name
          in: query
          type: string
          default: 'jesse'
```

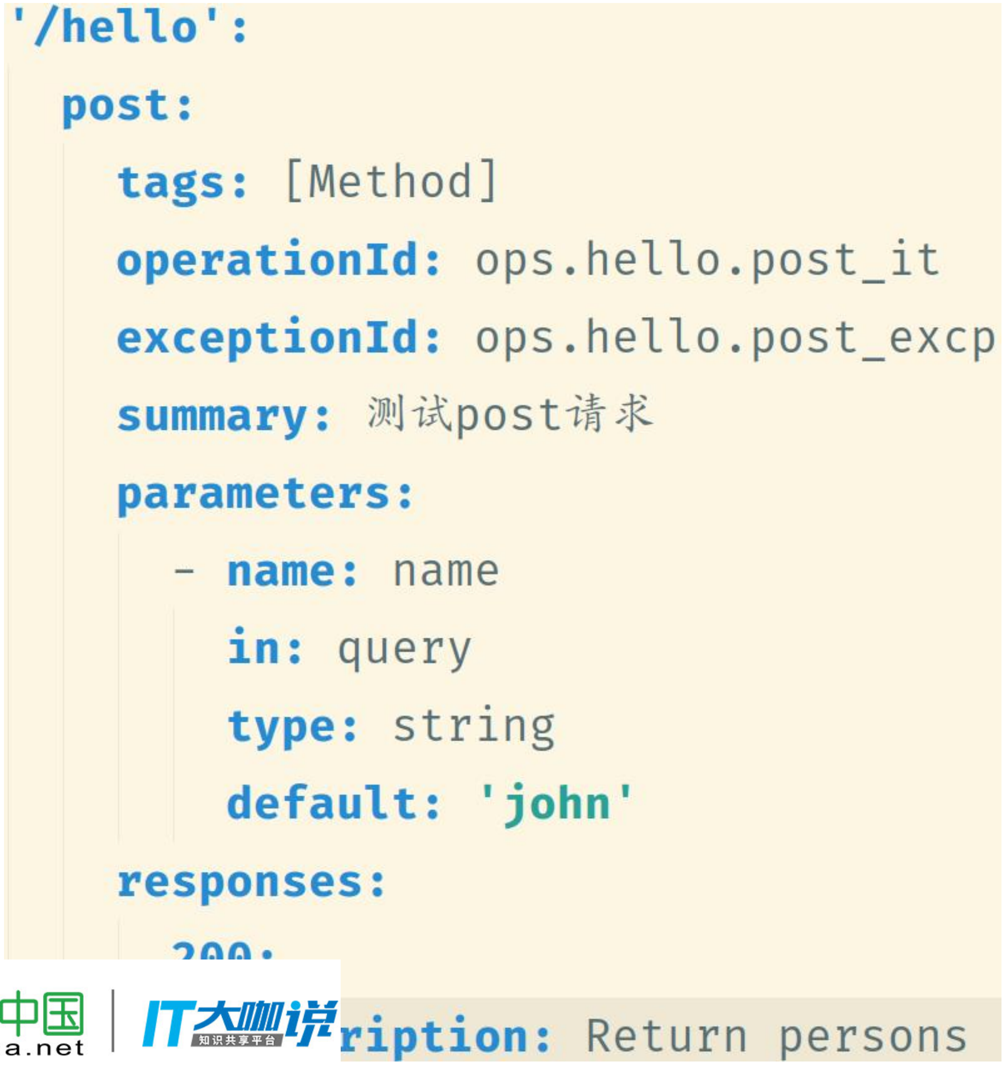


```python
def get_it(name):
    return {
        'get': name
    }
```

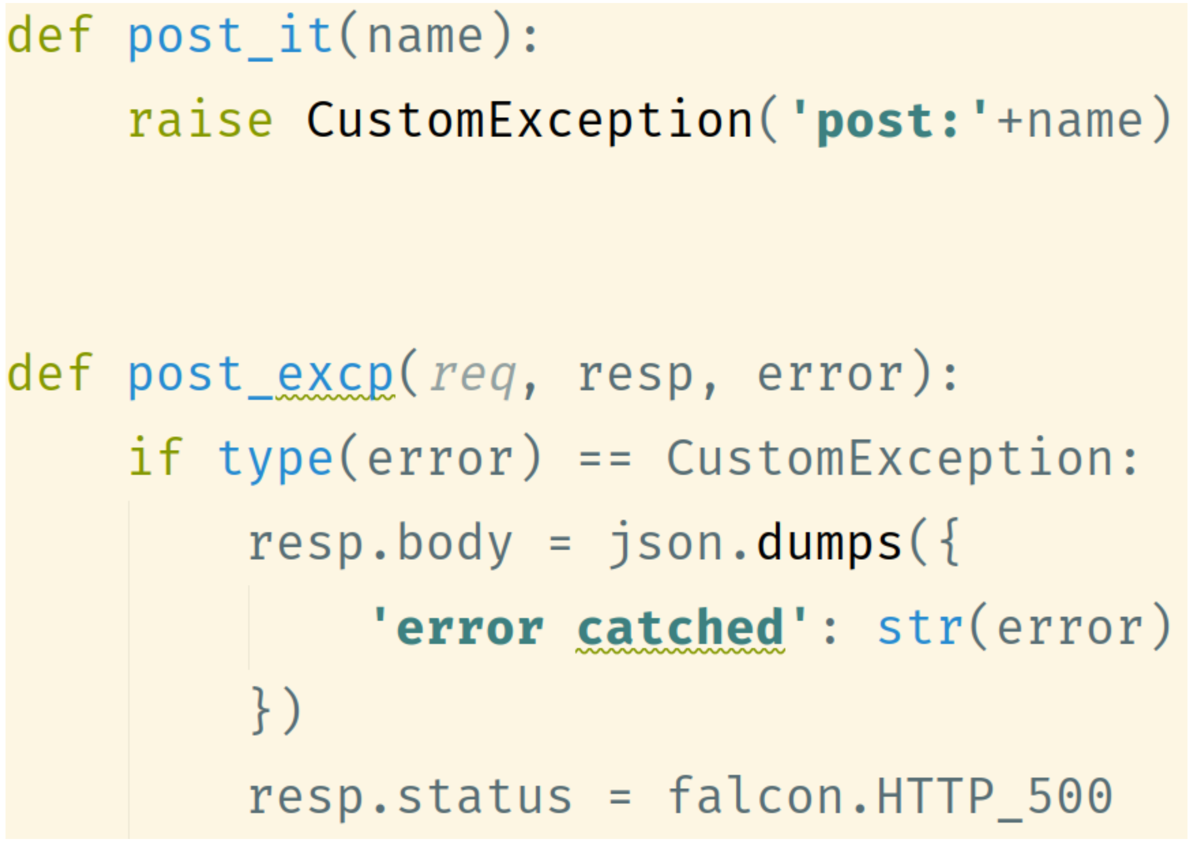


```python
def before_check_get_more_than_6(req, resp, name):
    if len(name) < 6:
        raise Exception('less than or equal to 6')
    return
```

exceptionId



```yaml
'/hello':
  post:
    tags: [Method]
    operationId: ops.hello.post_it
    exceptionId: ops.hello.post_excp
    summary: 测试post请求
    parameters:
      - name: name
        in: query
        type: string
        default: 'john'
    responses:
      200:
        ription: Return persons
```



```python
def post_it(name):
    raise CustomException('post:'+name)


def post_excp(req, resp, error):
    if type(error) == CustomException:
        resp.body = json.dumps({
            'error catched': str(error)
        })
        resp.status = falcon.HTTP_500
```

all about the Id ?

- operationId
- validationId
- beforeId
- afterId
- exceptionId

Live Show

顺带把这张图献给那些
showcase必败的team

Q&A

Thanks!