pre-request Documentation

wudong@eastwu.cn

Jan 27, 2022

CONTENTS

pre-requestpre-requestFlask

pre-request

pre-request

CHAPTER

ONE

PRE-REQUEST

pre-request

1.1 PIP

pre-request PyPipip

```
pip install pre-request
```

1.2

githubdevelop

```
python setup.py install
```

```
python setup.py develop
```

1.3 whl

whl

```
python setup.py sdist
pip wheel --wheel-dir=./dist ./
```

CHAPTER

TWO

pre-requestpre-request

2.1 Minimal Usage

pre-request

```
from flask import Flask

from pre_request import pre
from pre_request import Rule

app = Flask(__name__)

req_params = {
   "userId": Rule(type=int, required=True)
}

@app.route("/")
@pre.catch(req_params)
def hello_world(params):
   return str(params)
```

1. *pre-request pre*
2. *userId int*
3. *@pre.catch(req_params)*
4. `g` *params*

2.2 *pre*

pre-request *pre pre* pre-request

- *pre.fuzzy* pre-request
- *pre.sore_key* pre-requestkey
- *pre.content_type* pre-request *application/json text/html*

2.3 *@pre.catch*

pre-request`@pre.catch`

@pre.catch method, *method*

```
@app.route("/get", methods=['get'])
@pre.catch(get=req_params)
def get_handler(params):
    return str(params)

@app.route("/post", methods=['post'])
@pre.catch(post=req_params)
def get_handler(params):
    return str(params)
```

```
# getpost
@app.route("/all", methods=['get', 'post'])
@pre.catch(get=get_field, post=post_field)
def all_handler(params):
    return str(params)
```

CHAPTER

THREE

PRE-REQUEST

pre-requestpre-requestFlask

pre-request

pre-request

3.1 location

location *location* ["args", "form", "values", "headers", "cookies", "json"]

```
#
params = {
  "Access-Token": Rule(location="headers"),
  "userId": Rule(location=["cookies", "headers", "args"])
}
```

3.2 deep

deep rulepre-requestjson *deep=False* pre-request

json . : *userInfo.socialInfo.age=13*

```
#
params = {
  "userInfo": {
      "userId": Rule(type=int, required=False),
      "socialInfo": {
          "gender": Rule(type=int, enum=[1, 2], default=1),
          "age": Rule(type=int, gte=18, lt=80),
          "country": Rule(required=True, deep=False)
      }
  }
}
```

3.3 type

type . str

```
# userIdint
params = {
    "userId": Rule(type=int)
}
```

3.4 skip

skip False

```
#
params = {
    "userName": Rule(skip=True)
}
```

3.5 required

required True

```
#
params = {
    "profile": Rule(required=False)
}
```

3.6 required_with

required_with None

```
#
params = {
     "nickName": Rule(required=False),
     "profile": Rule(required=False, required_with="nickName")
}
```

3.7 default

default default : default required=False

```
#
params = {
   "nickName": Rule(required=False, default="")
}
```

3.8 split

split None

```
# ','
params = {
  "userId": Rule(int, split=",")
}
```

3.9 trim

trim False

```
#
params = {
   "nickName": Rule(trim=True)
}
```

3.10 enum

enum []

```
# 12
params = {
   "gender": Rule(direct_type=int, enum=[1, 2])
}
```

3.11 reg

reg None

```
#
params = {
   "tradeDate": Rule(reg=r"^[1-9]\d{3}-(0[1-9]|1[0-2])-(0[1-9]|[1-2][0-9]|3[0-1])$")
}
```

3.12 email

email ^[A-Za-zd]+([-_.][A-Za-zd]+)@([A-Za-zd]+[-.])+[A-Za-zd]{2,4}$ reg False*

```
# email
params = {
  "email": Rule(email=True)
}
```

3.13 mobile

mobile False

```
# mobile
params = {
  "mobile": Rule(mobile=True)
}
```

3.14 contains

contains []

```
#  ""  ""
params = {
  "content": Rule(contains=["", ""])
}
```

3.15 contains_any

contains_any []

```
#  ""  ""
params = {
  "content": Rule(contains_any=["", ""])
}
```

3.16 excludes

excludes []

```
# "",""
params = {
   "content": Rule(excludes=["", ""])
}
```

3.17 startswith

startswith None

```
#  "CN"
params = {
   "nickName": Rule(startswith="CN")
}
```

3.18 endswith

endswith *None*

```
#  "@eastwu.cn"
params = {
    "email": Rule(endswith="@eastwu.cn")
}
```

3.19 lower

lower *False*

```
#
params = {
  "nickName": Rule(lower=True)
}
```

3.20 upper

upper *False*

```
#
params = {
  "country": Rule(upper=True)
}
```

3.21 ipv4/ipv6

ipv4 IPV4 *False*

ipv6 ipv6 *False*

```
params = {
  "ip4": Rule(ipv4=True)
  "ip6": Rule(ipv6=True)
}
```

3.22 mac

mac MAC *False*

```
params = {
  "macAddress": Rule(mac=True)
}
```

3.23 fmt

`datetime`. `fmt``type=datetime.datetime`

```
params = {
  "birthday": Rule(type=datetime.datetime, fmt="%Y-%m-%d")
}
```

3.24 latitude / longitude

False

```
params = {
  "latitude": Rule(latitude=True),
  "longitude": Rule(longitude=True)
}
```

3.25 eq / eq_key

eq None

eq_key None

```
params = {
  "userId": Rule(eq=10086),
  "userId2": Rule(eq_key="userId")
}
```

3.26 neq / neq_key

neq None

neq_key None

```
params = {
   "userId": Rule(neq=0),
   "forbidUserId": Rule(neq_key="userId")
}
```

3.27 gt / gt_key

gt int str *None*

gt_key None

3.28 gte / gte_key

gt / gt_key

3.29 lt / lt_key

gt / gt_key,

3.30 lte / lte_key

gt / gt_key

3.31 key_map

kay_map None

```
params = {
  "userId": Rule(direct_type=int, key_map="user_id")
}
```

3.32 json

json json *False*

3.33 call_back

call_back

```
def hand(value):
  return value + 100

params = {
  "userId": Rule(direct_type=int, call_back=hand)
}
```

CHAPTER

FOUR

4.1

pre-request pre-request JSON

```
{
    "respCode": 560,
    "respMsg": "",
    "result": {}
}
```

pre-request `BaseResponse`

```
from flask import make_response
from pre_request import BaseResponse

class CusResponse(BaseResponse):

  def __call__(self, fuzzy=False, formatter=None, error=None):
      result = {
          "code": error.code,
          "rst": {}
      }
      return make_response(json.dumps(result))
```

pre-request

```
from pre_request import pre

pre.add_response(CusResponse)
```

4.2

pre-request

```
def custom_formatter(code, msg):
  """
  """
  return {
      "code": code,
      "msg": "hello",
      "sss": "tt",
  }
```

pre-request

```
from pre_request import pre
pre.add_formatter(custom_formatter)
```

4.3

pre-request pre-requestpre-request pre-request

`BaseFilter`

```
from pre_request import BaseFilter

class CustomFilter(BaseFilter):

    def fmt_error_message(self, code):
        if code == 10086:
            return ""

    def filter_required(self):
        """
        """
        return True

    def __call__(self, *args, **kwargs):
        """
        """
        super(CustomFilter, self).__call__()

        if self.rule.direct_type == int and self.key == "number" and self.value !=␣
↪10086:
            raise ParamsValueError(code=10086, filter=self)

        return self.value + 1
```

`fmt_error_message`*filter_required* *__call__* *filter_required* *__call__* *fmt_error_message*

pre-request

```
from pre_request import pre

pre.add_filter(CustomFilter)
```

4.4

pre-request *~flask.g.params* *params* *params*

```
from pre_request import pre

# key
pre.store_key = "pre_params"
```