flask_validator Documentation

Author

Aug 13, 2018

Contents:

Pythonic JSON payload validator for requested JSON payload of Flask

Flask view decorator JSON validation . Flask Large Application Example view decorator , MongoEngine .

CHAPTER 1

Usages

1.1 json_required

```
from flask import Flask
from flask_validator import json_required, Validator

app = Flask(__name__)
Validator(app)


@json_required()
@app.route('/', methods=('POST'))
def index():
    return 'hello!'
```

1.2 validate_keys

```
from flask import Flask
from flask_validator import validate_keys, Validator

app = Flask(__name__)
Validator(app)


@validate_keys(['name', 'age', {'position': ['latitude', 'longitude']}])
@app.route('/', methods=('POST'))
def index():
    return 'hello!'
```

1.3 validate_common

```
from flask import Flask
from flask_validator import validate_common, Validator

app = Flask(__name__)
Validator(app)


@validate_common({'name': str, 'age': int, 'position': {'latitude': float, 'longitude
↪': float}})
@app.route('/', methods=('POST'))
def index():
    return 'hello!'
```

1.4 validate_with_fields

```
from flask import Flask
from flask_validator import validate_with_fields
from flask_validator import StringField, IntField
from flask_Validator import Validator

app = Flask(__name__)
Validator(app)


@validate_with_fields({
    'name': StringField(allow_empty=False, regex='[-]+'),
    'age': IntField(min_value=0),
    'position': {
        'latitude': FloatField(min_value=-90, max_value=90),
        'longitude': FloatField(min_value=-180, max_value=180)
    }
})
@app.route('/', methods=('POST'))
def index():
    return 'hello!'
```

1.5 validate_with_jsonschema

```
from flask import Flask
from flask_validator import validate_with_jsonschema, Validator

app = Flask(__name__)
Validator(app)


@validate_with_jsonschema({
    'type': 'object',
    'properties': {
        'name': {'type': 'string'},
```

(continues on next page)

(continued from previous page)

```
        'age': {'type': 'number'}
    }
})
@app.route('/', methods=('POST'))
def index():
    return 'hello!'
```

CHAPTER 2

API documentation

2.1 Configuring Flask-Validation

class flask_validation.validator.**Validator**(*app=None*)

Bases: object

Create the Validator instance to register config. You can either pass a flask application in directly here to register this extension with the flask app, or call init_app after creating this object (in a factory pattern). :param app: A flask application

init_app(*app*)

2.2 Decorators

flask_validation.decorators.**json_required**()

A decorator to check header type is application/json

if you decorate endpoint with this, it will ensure that the request has a valid payload type before access endpoint if header's content type is not application/json, abort the invalid_content_type_abort_code

flask_validation.decorators.**validate_common**(*key_type_mapping: dict*)

A decorator to check request payload keys and type

If the request payload does not include the key in key_type_mapping, abort the key_missing_code, and if the type is not correct, abort the invalid_type_code.

Nested JSON processing is possible by inserting the dictionary in the required_keys like this {'a': str, 'b': int, 'c': {'d': int, 'e': str}}

Parameters **key_type_mapping** – A dictionary for payload check with this form {<key name>: <type class>}

flask_validation.decorators.**validate_keys**(*required_keys*)

A decorator to check request payload keys

if you decorate endpoint with this, it will ensure that the request's json body includes '*required_keys*'. if request body didn't includes `required_keys` , abort with `key_missing_abort_code`

Nested JSON processing is possible by inserting the dictionary in the `required_keys` like this `['a', 'b', {'c': ['q' ,'z']}]`

Parameters **required_keys** – key list to check request body's JSON

`flask_validation.decorators.`**validate_with_fields**(*key_field_mapping: dict*)

A decorator to check request payload with Field classes in fields.py

If the request payload does not include the key in key_type_mapping, abort `key_missing_code` and abort `validation_failure_code` if field validation fails.

Nested JSON processing is possible by inserting the dictionary in the `required_keys` like this `{'a': StringField(allow_empty=False), 'b': IntField(min_value=0), 'c': {'d': BooleanField()}}`

Parameters **key_field_mapping** – A dictionary for payload check with this form `{<key name>: <field class>}`

`flask_validation.decorators.`**validate_with_jsonschema**(*jsonschema: dict*)

A decorator to check request payload with jsonschema

If validation fails(jsonschema.exceptions.ValidationError raised), abort the `validation_error_abort_code`.

Parameters **jsonschema** – jsonschema

2.3 Fields

class `flask_validation.fields.`**BooleanField**(*validator_function=None, enum=None, required: bool = True, allow_null: bool = False*)

Bases: `flask_validation.fields._BaseField`

Boolean field class

validate(*value*)

class `flask_validation.fields.`**FloatField**(*min_value=None, max_value=None, **kwargs*)

Bases: *`flask_validation.fields.NumberField`*

Float field class

validate(*value*)

class `flask_validation.fields.`**IntField**(*min_value=None, max_value=None, **kwargs*)

Bases: *`flask_validation.fields.NumberField`*

Int field class

validate(*value*)

class `flask_validation.fields.`**ListField**(*min_length: int = None, max_length: int = None, **kwargs*)

Bases: `flask_validation.fields._BaseField`

List field class

validate(*value*)

class flask_validation.fields.**NumberField**(*min_value=None*, *max_value=None*, ***kwargs*)

Bases: flask_validation.fields._BaseField

Number field class

validate(*value*)

class flask_validation.fields.**StringField**(*allow_empty: bool = True*, *min_length: int = None*, *max_length: int = None*, *regex=None*, ***kwargs*)

Bases: flask_validation.fields._BaseField

String field class

validate(*value*)

CHAPTER 3

Configuration Options

You can change many options for how this extension works via

```
app.config[OPTION_NAME] = new_options
```

3.1 Options:

INVALID_CONTENT_TYPE_ABORT_CODE	default is 406
KEY_MISSING_ABORT_CODE	default is 400
INVALID_TYPE_ABORT_CODE	default is 400
VALIDATION_FAILURE_ABORT_CODE	default is 400
VALIDATION_ERROR_ABORT_CODE	default is 400

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index