flask-request-params Documentation

Release 0.2.3

bluele

Nov 06, 2017

Contents

Flask-request-params provides Rails-like interface to HTTP Request Parameters for Flask.

Supports mixed parameters(GET, POST, POST-JSON), Rails Strong_Parameters.

CHAPTER 1

Installation

To install Flask-request-params, simply:

```
pip install flask-request-params
```

Or alternatively, you can download the repository and install manually by doing:

```
git clone git@github.com:bluele/flask-request-params.git
cd flask-request-params
python setup.py install
```

CHAPTER 2

Examples

See examples for more code.

Client code

```
# support hash type
$ curl -X POST http://localhost:5000/echo/user -d 'user[name]=john&user[password]=pass
↪'
{
  "path": "user",
  "user": {
    "name": "john",
    "password": "pass"
  }
}

# support array type
$ curl -X POST http://localhost:5000/echo/lang -d 'languages[]=python&
↪languages[]=golang'
{
  "path": "lang",
  "languages": [
    "python",
    "golang"
  ]
}

# support strong_parameters
$ curl -X POST http://localhost:5000/user -d 'user[name]=bluele&
↪user[password]=password'
{
  "name": "bluele",
  "password": "password"
}
```

Server code

```
from flask import Flask, request, render_template, jsonify
from flask_request_params import bind_request_params


app = Flask(__name__)
app.secret_key = 'secret'
# bind rails like params to request.params
app.before_request(bind_request_params)

# just return request.params
@app.route('/echo/<path>', methods=['GET', 'POST'])
def echo(path):
    return jsonify(request.params)

@app.route('/user', methods=['POST'])
def create_user():
    user = request.params.require('user').permit('name', 'password')
    # do something
    return jsonify(user)

# serve at localhost:5000
app.run(debug=True)
```

CHAPTER 3

Link

PyPI - https://pypi.python.org/pypi/Flask-request-params

Github - https://github.com/bluele/flask-request-params