

How to get data received in Flask request



539



195

I want to be able to get the data sent to my Flask app. I've tried accessing `request.data` but it is an empty string. How do you access request data?

```
@app.route('/', methods=['GET', 'POST'])
def parse_request():
    data = request.data # data is empty
    # need posted data here
```

The answer to this question led me to ask [Get raw POST body in Python Flask regardless of Content-Type header](#) next, which is about getting the raw data rather than the parsed data.

python

flask

edited Apr 4 '17 at 10:35



Cœur

20.4k

10

119

160

asked May 3 '12 at 15:31



ddinchev

17.4k

17

71

114

16 Answers



881



The [docs](#) describe the attributes available on the request. In most common cases `request.data` will be empty because it's used as a fallback:

`request.data` Contains the incoming request data as string in case it came with a mimetype Flask does not handle.

- `request.args` : the key/value pairs in the URL query string
- `request.form` : the key/value pairs in the body, from a HTML post form, or JavaScript request that isn't JSON encoded
- `request.files` : the files in the body, which Flask keeps separate from `form` . HTML forms must use `enctype=multipart/form-data` or files will not be uploaded.
- `request.values` : combined `args` and `form` , preferring `args` if keys overlap

All of these are [MultiDict](#) instances. You can access values using:

- `request.form['name']` : use indexing if you know the key exists
- `request.form.get('name')` : use `get` if the key might not exist
- `request.form.getlist('name')` : use `getlist` if the key is sent multiple times and you want a list of values. `get` only returns the first value.

edited Jan 23 '18 at 15:18



davidism

answered May 21 '13 at 7:25

Robert



68.7k

13

193

200



17.5k

7

27

29

155 Note: In order to get the payload as JSON you could use `request.get_json()` . – [albert](#) Feb 8 '16 at 9:51

28 Adding on to @albert's comment, `request.get_json(force=True)` can be used. From the documentation: `force` – if set to `True` the `mimetype` is ignored. , so it won't return `None` for requests that don't have `application/json` set in the `contentType` header. – [plsnoban](#) Jul 5 '16 at 16:48

1 @fujianjin6471: see [docs here](#) – [albert](#) Jan 21 '18 at 18:10

2 Also you can -or should- verify if a key exists before try to get it: `if 'some_value' in request.get_json(): actualValue = request.get_json()['some_value']` – [hestellez](#) Mar 8 '18 at 18:33

4 This must be the most annoying feature for me in Flask: I just want to get the request body, and that's all. And flask as a web server cannot do this, no, it will put it in the different place with the different format that I have to guess using a lot of `if` – [Tyler Temp](#) Jul 17 '18 at 13:40

`from flask import request`
`request.data`

180

answered May 3 '12 at 15:38

[clyfish](#)

7,330

2

24

23

13 For docs on this see [flask.pocoo.org/docs/quickstart/#accessing-request-data](#) and [flask.pocoo.org/docs/api/#incoming-request-data](#). – [Steven Rumbalski](#) May 3 '12 at 15:41

3 this works if you specify the `contentType` (eg., `'applications/json'`) in the request – [tldr](#) Oct 15 '14 at 21:13

@clyfish Hi I want to forward complete request object(which includes **request.files** and `request.form`) to third party API using `http requests.post`. Can you please guide me? – [Naisarg Parmar](#) Feb 4 at 3:15

It is simply as follows

148

For **URL Query parameter**, use **request.args**

```
search = request.args.get("search")
page = request.args.get("page")
```

For **Form input**, use **request.form**

```
email = request.form.get('email')
password = request.form.get('password')
```

For **data type application/json**, use **request.data**

```
# data in string format and you have to parse into dictionary
data = request.data
```

```
dataDict = json.loads(data)
```

answered Aug 12 '14 at 15:22

**Fizer Khan****36.6k** 24 119 135

40 Flask has a shortcut for JSON: [request.get_json\(\)](#) – Mark E. Haase Jan 3 '15 at 22:05

1 if data sent via POST you must use request.form.get("") – mcolak May 24 '17 at 11:21

1 update to link in @MarkE.Haase comment: [request.get_json\(\)](#) – ssc Dec 30 '18 at 11:21

▲ I give a full example of **application/json**:

70

```
from flask import Flask, abort, request
import json
```

```
app = Flask(__name__)
```

```
@app.route('/foo', methods=['POST'])
def foo():
    if not request.json:
        abort(400)
    print request.json
    return json.dumps(request.json)
```

```
if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000, debug=True)
```

use [Postman](#) for post request:

The screenshot shows a REST client interface with a tab for 'localhost:5000/foo'. The request method is 'POST'. The 'Body' tab is selected, and the content type is set to 'JSON (application/json)'. The request body is a JSON object: `{ "userId": 1, "username": "fizz bizz" }`. Below the request, the response is shown in the 'Body' tab, displaying the same JSON object: `{ "username": "fizz bizz", "userId": 1 }`.

use **curl** command:

```
curl -i -H "Content-Type: application/json" -X POST -d '{"userId": "1", "username": "fizz bizz"}' http://localhost:5000/foo
```

P.S. For URL Query parameter example, you can see my answer in [Multiple parameters in in Flask approute](#)

edited Mar 29 '18 at 12:49



kaushal agrawal

170 15

answered Nov 16 '16 at 3:55



Little Roys

2,161 2 17 24

Why would you use `request.json` if you convert it back with `json.dumps()` anyway? – [nyuszika7h](#) Dec 11 '16 at 21:21

2 [@nyuszika7h](#), not for any practical application, but just for showing the result. – [Jochem Schenklopper](#) Nov 10 '17 at 15:32

Flask has another shortcut for JSON:

Header:

29

`{Content-Type: application/json}`

```
@app.route("/something", methods=["POST"])
def do_something():
    data = request.get_json()
```

edited Nov 10 '17 at 16:54



Jochem Schenklopper
2,623 23 43

answered Jul 27 '15 at 13:06



Amit Karnik
560 10 14

-
- 1 Just a little note: If the /something end point is called other than json format, `get_json()` will fail. Do not forget handling the exceptions. – [cell-in](#) Dec 2 '18 at 13:21
-

if you want the raw post body regardless of the content type, you should use `request.get_data()` , because `request.form` is converted to `werkzeug.ImmutableMultiDict` format.

answered Aug 14 '15 at 7:29



Xiao
9,274 2 18 33

```
@app.route('/', methods=['POST'])
def process_data():
    req_data = request.get_json(force=True) # force=True will make sure this works even
    if a client does not specify application/json
    language = req_data['language'] # or whatever key you have in your json

    return '''The language value is: {}'''.format(language)
```

edited Mar 25 '18 at 20:53

answered Mar 25 '18 at 14:45



Tarik Fojnica
286 2 8

-
- 2 You're right about the documenting it, I slightly updated my comments. Why is my answer different than the other ones? Simply because it gets straight to the point. I wrote only what the OP asked, and it works without any additional question. – [Tarik Fojnica](#) Mar 25 '18 at 21:00
-

Using `request.form` .

Instead of getting a single form data (`request.form["field_name"]`), you can obtain all posted data, by parsing the `ImmutableDict` provided by `request.form` object, like this:

Flask (Route)

```
@app.route('/data', methods=['POST'])
def f_data():
    if request.method == "POST":
```

```

fields = [k for k in request.form]
values = [request.form[k] for k in request.form]
data = dict(zip(fields, values))
return jsonify(data)

```

Shell

```

$ curl http://127.0.0.1:5000/data -d "name=ivanleonz&role=Software Developer"
{
  "name": "ivanleonz",
  "role": "Software Developer"
}

```

For more details, this [Gist](#).

edited Jun 7 '18 at 18:22

answered Jun 6 '18 at 19:54



[ivanleonz](#)

2,461 3 27 34

1 I love this one ! – [Mouldri](#) Oct 18 '18 at 18:20

Thanks :)! I believe that it is clear enough, with some comprehensions and dynamic, for exposing as a JSON API (depending on your necessity) or having this object for database purposes. – [ivanleonz](#) Oct 18 '18 at 18:23

Simply speaking, you can get data by the way below:

8

```

@app.before_request
def before_request():
    g.data = request.get_json() or request.values

```

Now, `g.data` is an instance of `werkzeug.ImmutableMultiDict`. Then you can use `g.data` which can handle most of your requirements. For example, you can use it just like this:

```

@app.route("/something", methods=["POST"])
def do_something():
    result = handle(g.data)
    return jsonify(data=result)

```

Of course, you can use `blueprint` instead of `app` ~~

answered Aug 22 '17 at 12:46



[zhangqy](#)

295 3 7

```

length = request.headers["Content-Length"]
data=request.stream.read()

```

6

Now, data is the request body

edited Feb 24 at 9:48

answered Jul 11 '16 at 15:46



Daniel

191 2 6

This answer is fine, but please don't use built in functions as variable names, consider changing `len` to `length` . – Purple Ice Feb 13 at 9:39

Thanks for your suggestion, I have changed it. – Daniel Feb 24 at 9:48

4

If the mime type is recognized, then both `request.data` and `request.get_data()` will return empty strings.

To get the full contents regardless, you need to call `request.get_data(as_text=True)` .

See http://flask.pocoo.org/docs/1.0/api/#flask.Request.get_data

answered Aug 21 '18 at 22:52



Zavec

85 8

3

This is kind of a dirty hack to get all the request data regardless of how it was sent, but I seriously just use:

```
def get_request_info():
    args = str(request.args)
    form = str(request.form)
    files = str(request.files)
    maybe_json = request.get_json(silent=True, cache=False)
    if maybe_json:
        thejson = json.dumps(maybe_json)
    else:
        thejson = "no json"
    return # whatever you want
```

and then I just return either a string that concatenates these, or, if I feel fancy, I skip the string calls/json dump and merge all the dicts. then this can be logged, returned in a view function, whatever and you can actually see the whole request no matter what it includes.

answered Dec 1 '18 at 5:04



Paul Gowder

1,074 1 12 21

2

For those like me who have forgotten (a bit) about HTML, be sure `<input>` in your `<form>` has a `name=""` attribute!

```

from flask import Flask, request

app = Flask(__name__)

@app.route('/', methods=['GET', 'POST'])
def index():
    print("Posted data : {}".format(request.form))

    return """
<form method="post">
  <input type="text">
  <input type="text" id="idtxt2">
  <input type="text" name="txt3" id="idtxt3">
  <input type="submit" Value="Hopla!">
</form>
"""

if __name__ == "__main__":
    app.run()

```

Result on console :

```

freezed@machine % python3 run.py
* Serving Flask app "flaskstuff.views" (lazy loading)
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 268-634-781
127.0.0.1 - - [20/Aug/2018 16:52:59] "POST / HTTP/1.1" 200 -
Posted data : ImmutableMultiDict([('txt3', 'text 3')])

```

No *name* attribute = no data in `ImmutableMultiDict([])` !

edited Sep 2 '18 at 15:50

answered Aug 20 '18 at 15:45



freezed

497 1 8 21

In javascript:

2

```

var value_data = [1,2,3,4];

$.ajax({
  type: 'POST',
  url: '/',
  data:JSON.stringify(value_data),
  success: function (response) {
    alert("Data added successfully");
  },
});

```

In python:


```
client_data = request.get_data()
```

edited May 11 '18 at 14:01



Syscall

14.3k 5 11 32

answered May 11 '18 at 13:58



vaishali chaudhari

111 1 4



```
from flask import request
```

2

```
content = request.get_json()
name = content.get('name', '')
```



get data if request type json and you can also mention default parameters along with it

```
from flask import request
```

```
content = request.form
name = content.get('name', '')
```

get data if request type form

```
from flask import request
```

```
request.args.get("name", "")
```

to fetch parameters from url with a GET request

answered Mar 10 at 7:32



Ravin Gupta

177 1 9



```
from flask import Flask, request, jsonify
```

-1

```
@app.route('/added', methods=['POST'])
def add():
    data = request.get_json(force=True)
    l = {'name': data['name']}
    lingual.append(l)

    return jsonify({'lang': lingual})
```



answered Jul 21 '18 at 22:49



aakash gupta

1

when you get raw data using post method in flask use request.get_json(force= True) – aakash gupta Jul 21 '18 at 22:51

3 You should explain your code and how it solves the OP's question. – Nic3500 Jul 22 '18 at 1:33

protected by [eyllanesc](#) Aug 21 '18 at 22:53

Thank you for your interest in this question. Because it has attracted low-quality or spam answers that had to be removed, posting an answer now requires 10 [reputation](#) on this site (the [association bonus does not count](#)).

Would you like to answer one of these [unanswered questions](#) instead?