Advanced Programming with Python. Session 2

Pepe García

2020-04-20

Advanced Programming with Python. Session 2

Plan for today

- returning different status codes

Plan for today

- returning different status codes
- using request bodies

Status codes

Status codes

We all know the infamous **404 Not Found** HTTP status code. Apart of it, there are a lot more that are used when designing APIs. Some of the most used are:

Status codes

200 OK

Used whenever everything went correctly

Status Codes

201 Created

Used to give the user feedback so they know the resource has been created.

Status Codes

400 Bad request

A general error in the received request

Status Codes

404 Not found

Whenever the resource requested by the user is not found

Status Codes

Using status codes in Flask

Flask allows returning a tuple in any route, in which the first parameter is the response body, and the second the status code:

```
@app.route("/users/<user_id>")
def get_user(user_id):
    if user_not_found():
        return jsonify({"error": "not found"}), 404
```

Status Codes

Practice

Create a flask server exposing just one route that receives two numbers and divides the first by the second.

Validate the data and be sure to return a meaningful status code

Error handling

Whenever something crashes in our application, flask shows the error in a non-very-nice way.

Error handling

Flask provides a nice way of handling errors that may happen in our application, such as **404** or **500**.

```
@app.errorhandler(500)
def handle_500_error(error):
    return jsonify({"error": "500 Internal Server Error"}), 500
```

Error handling

Practice

Let's see the default behaviour and the effect we get when adding the error handler

HTTP request bodies

So far, we've been sending data in response bodies but haven't yet seen how to receive data from requests.

Something we'll need to consider is that not all HTTP verbs allow us to set request bodies:

verb	has body?
GET	no
HEAD	no
DELETE	no
PUT	yes
PATCH	yes
POST	yes

HTTP request bodies

Getting request body (server)

```
from flask import request

@app.route("/get-body")
def get_body():
    body = request.get_json()
    print(body)
    return "body received!"
```

HTTP request bodies

Using request body (client)

```
import requests

dictionary = {
  "name": "dict",
  "purpose": "none at all"
}

request.post("http://localhost:5000/get_body", json=dictionary)
```

HTTP request bodies

Practice

Create a API that allows tht user to:

- submit **tweets** ({"user": "pepe", "tweet": "Hello world"})

HTTP request bodies

Practice

Create a API that allows tht user to:

- submit **tweets** ({"user": "pepe", "tweet": "Hello world"})
- List all tweets

Homework

https://classroom.github.com/a/NTrXJTLv