Flask-Nicely Documentation

Release 0.2.0

Jonathan Evans

May 13, 2015

Contents

Flask-Nicely consists of a simple decorator with which to wrap Flask functions that return data in order to turn them into pretty, consistent JSON responses. It also provides several built in HTTP exceptions which can be raised in a Flask function in order to return the corresponding HTTP error response.

Docs can be found at http://flask-nicely.readthedocs.org/

CHAPTER 1

Dependencies

Flask-Nicely is tested on both Python 2.7 and Python 3.3. It has only one dependency: Flask itself.

CHAPTER 2

Installation

Flask-Nicely is available on PyPI. The best way to install is via pip, as follows:

```
pip install flask-nicely
```

Alternatively, you can download the source code directly from GitHub.

CHAPTER 3

Usage

The Flask-Nicely decorator is used as follows:

```
import flask_nicely

app = Flask(__name__)

@app.route('/hello/<name>')
@flask_nicely.nice_json
def hello(name):

    data = {
        "Name": name,
        "Message": "Hello, {}!".format(name)
    }

    return data
```

Wrapping our view function in the decorator will cause the view to return the following 200 response:

```
{
  "data": data,
  "error": null,
  "status": 200,
}
```

Flask-Nicely also allows you to throw HTTP exceptions which will result in a properly formed HTTP error response. Custom exception messages may be entered:

```
from flask import Flask

import flask_nicely
from flask_nicely.errors import NotFound

app = Flask(__name__)

@app.route('/error/404')
@flask_nicely.nice_json
def throw_404():

    raise NotFound("Could not find the grail!")
```

This will result in the following 404 response:

```
{
  "data": null,
  "error": "Could not find the grail!",
  "status": 404
}
```

Exceptions can accept a payload, which is an arbitrary dictionary to be sent as part of the JSON response. For example:

```
test_payload = {
    'error_detail': "The resource that you requested was not found on the server",
    'documentation': "http://www.flask-nicely.readthedocs.org",
}

raise NotFound(payload=test_payload)
```

Will result in a 404 response containing:

```
{
  "data": null,
  "error": "Not Found",
  "status": 404,
  "error_detail": "The resource that you requested was not found on the server",
  "documentation": "http://www.flask-nicely.readthedocs.org",
}
```

An example app can be found in the examples directory, and can be run from the root directory using:

```
python examples/app.py
```

CHAPTER 4

Testing

To run the tests for the project using py.test, simply run:

```
python setup.py test
```

For multi-version testing, install and run tox:

```
pip install tox
tox
```

CHAPTER 5

API

Flask-Nicely is comprised of two main modules, the `decorator` module and the `errors` module.

5.1 `decorator` Module

The `decorator` module is the core function of the package.

`flask_nicely.decorators.`**`nice_json`**(*func*)

A decorator which returns a pretty jsonified response when wrapped around a Flask view function.

Parameters **`func`** – the Flask view function to be wrapped

Return type `flask.Response`

5.2 `errors` Module

The `errors` module contains a collection of exceptions that will generate corresponding Flask HTTP responses when raised in a function.

exception `flask_nicely.errors.`**`Forbidden`**(*error_message=''*, *payload=None*, **args*, ***kwargs*)

Bases: *flask_nicely.errors.HTTPError*

A 403 Forbidden HTTP error.

`status_code` = 403

exception `flask_nicely.errors.`**`GatewayTimeout`**(*error_message=''*, *payload=None*, **args*, ***kwargs*)

Bases: *flask_nicely.errors.HTTPError*

A 504 Gateway Timeout HTTP error.

`status_code` = 504

exception `flask_nicely.errors.`**`HTTPError`**(*error_message=''*, *payload=None*, **args*, ***kwargs*)

Bases: `exceptions.Exception`

A generic error mixin from which the HTTP error responses inherit.

`get_response`()

`status_code` = None

exception flask_nicely.errors.**NotFound**(*error_message=''*, *payload=None*, **args*, ***kwargs*)
Bases: *flask_nicely.errors.HTTPError*

A 404 Not Found HTTP error.

status_code = 404

exception flask_nicely.errors.**ServerError**(*error_message=''*, *payload=None*, **args*, ***kwargs*)
Bases: *flask_nicely.errors.HTTPError*

A 500 Internal Server Error HTTP error.

status_code = 500

exception flask_nicely.errors.**Unauthorized**(*error_message=''*, *payload=None*, **args*, ***kwargs*)
Bases: *flask_nicely.errors.HTTPError*

A 401 Unauthorized HTTP error.

status_code = 401

CHAPTER 6

Changelog

6.1 Version 0.2

2014-03-11

First 'master' release.

- Greatly improved documentation.
- Added the ability to include arbitrary payloads with an error response.
- Extended tests to cover Python 3.3 compatibility.

6.2 Version 0.1

2014-02-14

Initial development release.

CHAPTER 7

Documentation Index

- genindex

f

F

G

H

N

S

U