Django Web Exceptions Documentation

Release 0.1.4

Maks Skorokhod

May 29, 2017

Contents

Contents:

CHAPTER 1

Django Web Exceptions

Throwing web exceptions like in AioHTTP

What and why?

In AioHTTP you can raise any response as exception (this is very cool). But Django can raise only 3+1 web exceptions.

- *400* SuspiciousOperation
- *403* PermissionDenied
- *404* Http404
- *500* Any other non catched exception

This package allow you to raise as exception any of HTTP response.

Documentation

The full documentation is at https://web-exceptions.readthedocs.io.

Quickstart

Install Django Web Exceptions:

```
pip install django-web-exceptions
```

Add it to your *MIDDLEWARE*:

```
# settings.py
MIDDLEWARE = (
    # ...
    'web_exceptions.middleware.WebExceptionsMiddleware',
    # ...
)
```

Features

Import exceptions and raise anywhere

```
# views.py
from web_exceptions import exceptions

# ...

def index(request):
    """ Simple view raise redirectexception """
    raise exceptions.HTTPMovedPermanently('/foo')
```

Also you can customize any kind of exception status code as custom handler, defined in *urls.py* like django error handlers .

```
# urls.py
from myapp import views

handler300 = <callable view>
handler400 = <callable view>
handler<status_code> = <callable view>
```

For more example see example proj

Running Tests

Does the code actually work?

```
source <YOURVIRTUALENV>/bin/activate
(myenv) $ pip install tox
(myenv) $ tox
```

Credits

Tools used in rendering this package:

- Cookiecutter
- cookiecutter-djangopackage

CHAPTER 2

Installation

Install with pip:

```
$ pip install web-exceptions
```

CHAPTER 3

Usage

Configure

To use Django Web Exceptions in a project, add it to your *MIDDLEWARE* settings:

```
# settings.py
MIDDLEWARE = [
    ...
    # add middleware for dj exceptions
    'web_exceptions.middleware.WebExceptionsMiddleware',
    ...
]
```

Simple usage

Import and raise Web Exceptions's:

```
from web_exceptions import exceptions

...

raise exceptions.HTTPOk(
    content="Thist is Http Ok response",
    headers={'X-Extra-Header': 'some value'})
```

Customize response

Self http exception

Declare custom web exception:

```
from web_exceptions import exceptions


class HTTPTeapot(exceptions.HTTPClientError):
    status_code = 418
    reason = "I'm a teapot"

...

raise HTTPTeapot()
```

Self response handler

Also you can customize any kind of exception status code as custom handler, defined in *urls.py* like django error handlers.

```
# urls.py
from myapp import views

handler300 = <callable view>
handler400 = <callable view>
handler<status_code> = <callable view>
```

List of available exceptions

200x status code

- *200* HTTPOk
- *201* HTTPCreated
- *202* HTTPAccepted
- *203* HTTPNonAuthoritativeInformation
- *204* HTTPNoContent
- *205* HTTPResetContent
- *206* HTTPPartialContent

300x status code

- *300* HTTPMultipleChoices
- *301* HTTPMovedPermanently
- *302* HTTPFound

- *303* HTTPSeeOther
- *304* HTTPNotModified
- *305* HTTPUseProxy
- *307* HTTPTemporaryRedirect
- *308* HTTPPermanentRedirect

400x status code

- *400* HTTPBadRequest
- *401* HTTPUnauthorized
- *402* HTTPPaymentRequired
- *403* HTTPForbidden
- *404* HTTPNotFound
- *405* HTTPMethodNotAllowed
- *406* HTTPNotAcceptable
- *407* HTTPProxyAuthenticationRequired
- *408* HTTPRequestTimeout
- *409* HTTPConflict
- *410* HTTPGone
- *411* HTTPLengthRequired
- *412* HTTPPreconditionFailed
- *413* HTTPRequestEntityTooLarge
- *414* HTTPRequestURITooLong
- *415* HTTPUnsupportedMediaType
- *416* HTTPRequestRangeNotSatisfiable
- *417* HTTPExpectationFailed
- *421* HTTPMisdirectedRequest
- *426* HTTPUpgradeRequired
- *428* HTTPPreconditionRequired
- *429* HTTPTooManyRequests
- *431* HTTPRequestHeaderFieldsTooLarge
- *451* HTTPUnavailableForLegalReasons

500x status code

- *500* HTTPInternalServerError
- *501* HTTPNotImplemented
- *502* HTTPBadGateway

- *503* HTTPServiceUnavailable
- *504* HTTPGatewayTimeout
- *505* HTTPVersionNotSupported
- *506* HTTPVariantAlsoNegotiates
- *510* HTTPNotExtended
- *511* HTTPNetworkAuthenticationRequired

CHAPTER 4

Contributing

Contributions are welcome, and they are greatly appreciated! Every little bit helps, and credit will always be given.

You can contribute in many ways:

Types of Contributions

Report Bugs

Report bugs at https://github.com/samael500/web-exceptions/issues.

If you are reporting a bug, please include:

- Your operating system name and version.
- Any details about your local setup that might be helpful in troubleshooting.
- Detailed steps to reproduce the bug.

Fix Bugs

Look through the GitHub issues for bugs. Anything tagged with “bug” is open to whoever wants to implement it.

Implement Features

Look through the GitHub issues for features. Anything tagged with “feature” is open to whoever wants to implement it.

Write Documentation

Django Web Exceptions could always use more documentation, whether as part of the official Django Web Exceptions docs, in docstrings, or even on the web in blog posts, articles, and such.

Submit Feedback

The best way to send feedback is to file an issue at https://github.com/samael500/web-exceptions/issues.

If you are proposing a feature:

- Explain in detail how it would work.
- Keep the scope as narrow as possible, to make it easier to implement.
- Remember that this is a volunteer-driven project, and that contributions are welcome :)

Get Started!

Ready to contribute? Here's how to set up *web-exceptions* for local development.

1. Fork the *web-exceptions* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/web-exceptions.git
```

3. Install your local copy into a virtualenv. Assuming you have virtualenvwrapper installed, this is how you set up your fork for local development:

```
$ mkvirtualenv web-exceptions
$ cd web-exceptions/
$ python setup.py develop
```

4. Create a branch for local development:

```
$ git checkout -b name-of-your-bugfix-or-feature
```

   Now you can make your changes locally.

5. When you're done making changes, check that your changes pass flake8 and the tests, including testing other Python versions with tox:

```
$ flake8 web_exceptions tests
$ python setup.py test
$ tox
```

   To get flake8 and tox, just pip install them into your virtualenv.

6. Commit your changes and push your branch to GitHub:

```
$ git add .
$ git commit -m "Your detailed description of your changes."
$ git push origin name-of-your-bugfix-or-feature
```

7. Submit a pull request through the GitHub website.

Pull Request Guidelines

Before you submit a pull request, check that it meets these guidelines:

1. The pull request should include tests.
2. If the pull request adds functionality, the docs should be updated. Put your new functionality into a function with a docstring, and add the feature to the list in README.rst.
3. The pull request should work for Python 2.6, 2.7, and 3.3, and for PyPy. Check https://travis-ci.org/samael500/web-exceptions/pull_requests and make sure that the tests pass for all supported Python versions.

Tips

To run a subset of tests:

```
$ python -m unittest tests.test_web_exceptions
```

CHAPTER 5

Credits

Development Lead

- Maks Skorokhod <samael500@gmail.com>

Contributors

None yet. Why not be the first?

CHAPTER 6

History

0.1.4 (2017-05-28)

- Set custom reason phrase for exception response.
- Add docs.

0.1.3 (2017-05-15)

- Fix pypi wrong upload issue.

0.1.2 (2017-05-15)

- Clean source code from unused rows.

0.1.1 (2017-05-13)

- Small fixes in source code and readme.

0.1.0 (2017-05-13)

- First release on PyPI.