django-exo-mentions Documentation

Release 1.1.2

Jose M. Marfil

Jul 24, 2019

Contents

Contents:

CHAPTER 1

django-exo-mentions

1.1 Documentation

The purpose of this package is to handle in some way mentions to users in a text field of a model. You can choose the model you want, the field you want to listen to mentions, the pattern you use to codify the mention and the callback to notify to your app.

The package will notify to callback function each time there is a mention in this field of the model. Then you can act accordingly on your application requisites.

The full documentation is at https://exo-mentions.readthedocs.io.

1.2 Quickstart

Install django-exo-mentions:

```
pip install django-exo-mentions
```

CHAPTER 2

Usage

To use django-exo-mentions in a project, add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'exo_mentions',
    ...
)
```

Define a signal for the callback

```
from django.dispatch import receiver
from django.core.signals import request_finished

@receiver(request_finished)
def post_detect_mention_callback(sender, **kwargs):
    """ You will receive information of the mention
    user_from: kwargs.get('user_from')
        User that mentions
    object_pk: kwargs.get('object_pk')
        User's Pk that has been mentioned
    target: kwargs.get('target')
        The object where the mention was made
    """

    # Your code here
```

Register a model and field in which you want to detect mentions. You can override the pattern if you want.

```
from django.apps import AppConfig
from exo_mentions.registry import register

class MyAppConfig(AppConfig):
    name = 'myapp'
```

(continues on next page)

(continued from previous page)

```
    def ready(self):
        model = Post
        field = 'description'
        callback = post_detect_mention_callback

        register(model, field, callback)
```

At this point the library will notify to the callback each time there is a mention in the field of the registered model. Thats all! :)

```
def register(model, field, callback, pattern):
"""
This method handles the mentions about the model in the field and notify to the␣
↪callback when there is any mention

Parameters
----------
model : Models
    The model to register for detect mentions
field : str
    Field of the model to detect mentions
callback : function
    Callback function to notify when there are mentions
pattern : regular expression
    The pattern to codify the mentions (default r'class="mention" data-user=[\'"]?([^\
↪'" >]+)')

"""
```

2.1 Running Tests

Does the code actually work?

Docker, Compose, and Tox are used to approximate the environment that Travis CI, Code Climate, and Coveralls all run when you push. This will allow you to test your code against multiple versions of Python (3.4, 3.5, 3.6, 3.7) locally before pushing it or even committing it. For more information about how to get Docker, please visit **'documentation https://docs.docker.com/install/linux/docker-ce/ubuntu/'_**.

To run everything (this will take a while the first time you run it, but subsequent runs will be quick):

$ docker build -t django-exo-mentions/tox:latest .

CHAPTER 3

Installation

At the command line:

```
$ pip install django-exo-mentions
```

CHAPTER 4

Usage

To use django-exo-mentions in a project, add it to your *INSTALLED_APPS*:

```
INSTALLED_APPS = (
    ...
    'exo_mentions',
    ...
)
```

Add this url to your api urls:

Define a signal for the callback

```
from django.dispatch import receiver
from django.core.signals import request_finished

@receiver(request_finished)
def post_detect_mention_callback(sender, **kwargs):
    """ You will receive information of the mention
    user_from: kwargs.get('user_from')
        User that mentions
    object_pk: kwargs.get('object_pk')
        User's Pk that has been mentioned
    target: kwargs.get('target')
        The object where the mention was made
    """

    # Your code here
```

Register a model and field in which you want to detect mentions. If you want to get trace about who is the user that has made the mention, ensure the model registered define a property or a model field called created_by. You can override the pattern if you want.

```
from django.apps import AppConfig
from exo_mentions.registry import register
```

(continues on next page)

(continued from previous page)

```
class MyAppConfig(AppConfig):
    name = 'myapp'

    def ready(self):
        model = Post
        field = 'description'
        callback = post_detect_mention_callback

        register(model, field, callback)
```

At this point the library will notify to the callback each time there is a mention in the field of the registered model. Thats all! :)

```
def register(model, field, callback, pattern):
"""
This method handles the mentions about the model in the field and notify to the␣
↪callback when there is any mention

Parameters
----------
model : Models
    The model to register for detect mentions
field : str
    Field of the model to detect mentions
callback : function
    Callback function to notify when there are mentions
pattern : regular expression
    The pattern to codify the mentions (default r'class="mention" data-user=[\'"]?([^\
↪'" >]+)')

"""
```

CHAPTER 5

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:

5.1 Types of Contributions

5.1.1 Report Bugs

Report bugs at https://github.com/exolever/django-exo-mentions/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.

5.1.2 Fix Bugs

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

5.1.3 Implement Features

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

5.1.4 Write Documentation

django-exo-mentions could always use more documentation, whether as part of the official django-exo-mentions docs, in docstrings, or even on the web in blog posts, articles, and such.

5.1.5 Submit Feedback

The best way to send feedback is to file an issue at https://github.com/exolever/django-exo-mentions/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 :)

5.2 Get Started!

Ready to contribute? Here's how to set up *django-exo-mentions* for local development.

1. Fork the *django-exo-mentions* repo on GitHub.
2. Clone your fork locally:

```
$ git clone git@github.com:your_name_here/django-exo-mentions.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 django-exo-mentions
$ cd django-exo-mentions/
$ 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 exo_mentions 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.

5.3 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/exolever/django-exo-mentions/pull_requests and make sure that the tests pass for all supported Python versions.

5.4 Tips

To run a subset of tests:

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

CHAPTER 6

Credits

6.1 Development Lead

- Jose M. Marfil <josemarfyl@gmail.com>
- Tomás Garzón <tomasgarzonhervas@gmail.com>
- Javier Sújar <javier.sujar@gmail.com>

6.2 Contributors

None yet. Why not be the first?

CHAPTER 7

History

7.1 1.0.0 (2018-09-25)

- First release on PyPI.