django-templated-mail Documentation

Release 1.1.1

Sunscrapers

Jan 10, 2019

Contents

CHAPTER 1

Getting started

1.1 Supported Python versions

- Python 2.7
- Python 3.4
- Python 3.5
- Python 3.6

1.2 Supported Django versions

- Django 1.8
- Django 1.11
- Django 2.0

1.3 Installation

```
$ pip install -U django-templated-mail
```

No need to add it to your `INSTALLED_APPS`.

CHAPTER 2

Settings

You may optionally provide following settings:

```
'DOMAIN': 'example.com'
'SITE_NAME': 'Foo Website'
```

2.1 DOMAIN

Used in email template context. In most cases it is used to simplify building URLs, when frontend and backend are hosted under different address'. If not provided the current site's domain will be used.

Required: `False`

2.2 SITE_NAME

Used in email template context. Usually it will contain the desired title of your app. If not provided the current site's name will be used.

Default: `False`

CHAPTER 3

Templates syntax

Email templates can be built using three simple blocks:

- `subject` - used for subject of an email message
- `text_body` - used for plaintext body of an email message (not required)
- `html_body` - used for html body of an email message (not required)

3.1 Examples

```
{% block subject %}Text and HTML mail subject{% endblock %}

{% block text_body %}Foobar email content{% endblock %}
```

```
{% block subject %}Text and HTML mail subject{% endblock %}

{% block text_body %}Foobar email content{% endblock %}
{% block html_body %}<p>Foobar email content</p>{% endblock %}
```

CHAPTER 4

Sample usage

At first let's discuss the simplest possible use case, where you just wish to send an email to a given address and using the given template.

```
from templated_mail.mail import BaseEmailMessage

BaseEmailMessage(template_name='email.html').send(to=['foo@bar.tld'])
```

This one-liner will do all of the work required to render proper template blocks and assign the results to proper email pieces. It will also determine appropriate content type (including support for MIME) and send the output message to provided list of email address'.

You might also wish to define your own subclass of `templated_mail.mail.BaseEmailMessage` to customize a thing or two. What might be most interesting for you is the `get_context_data` method, which returns context used during template rendering.

```
class MyEmailMessage(BaseEmailMessage):
    def get_context_data(self):
        context = super(MyEmailMessage, self).get_context_data()
        context['foo'] = 'bar'
        return context
```

You might also provide custom context data using the `context` parameter.

```
from templated_mail.mail import BaseEmailMessage

BaseEmailMessage(context={'foo': 'bar'}, template_name='email.html').send(to=[
↪'foo@bar.tld'])
```

In other cases you might notice that some of your emails use common `template_name` and so to save some space you might wish to override the base class' attribute.

```
class MyEmailMessage(BaseEmailMessage):
    template_name = 'email.html'
```

CHAPTER 5

Indices and tables

- genindex
- search