django-mailmate Documentation

Release 1.3.0

Chris McKenzie

Jul 24, 2018

Contents

Mailmate is a Django app comprised of tools to make dealing with emails easier. Its main feature is a simple, class-based way to define email messages using Django templates. Here's a quick sales pitch:

```
from mailmate import TemplatedEmailMessage

class MyEmail(TemplatedEmailMessage):
    to = ['some-user@some-email.com']
    from_email = 'no-reply@some-email.com'
    subject = 'Hello, {{ name }}!'
    template_name = 'emails/template.txt'

MyEmail(extra_context={'name': 'Jerry'}).send()
```

CHAPTER 1

Installation

```
pip install django-mailmate
```

CHAPTER 2

TemplatedEmailMessage

Extend `TemplatedEmailMessage`, and set class attributes. You can override any of those attributes by passing keyword arguments to the constructor.

```
from mailmate import TemplatedEmailMessage

class MyEmail(TemplatedEmailMessage):
    to = ['some-user@some-email.com']
    from_email = 'no-reply@some-email.com'
    subject = 'Hello!'
    template = 'emails/template.txt'

MyEmail(to=['somebodyelse@somewhereelse.com']).send()
```

You can use a template to define your email body (like in the above example), or define it as a string:

```
from mailmate import TemplatedEmailMessage

class MyEmail(TemplatedEmailMessage):
    to = ['some-user@some-email.com']
    subject = "The subject is parsed as a {{ what }}"
    body = "The body's also parsed as a {{ what }}."

MyEmail(extra_context={'what': 'Django template!'}).send()
```

`TemplatedEmailMessage` also makes it simple to create HTML emails. Simply add an `html_template_name` attribute to your class (or pass it to the constructor):

```
from mailmate import TemplatedEmailMessage

class MyEmail(TemplatedEmailMessage):
    to = ['some-user@some-email.com']
    subject = "The subject is parsed as a {{ what }}"
    body = "The body's also parsed as a {{ what }}."
    html_template_name = 'emails/my_email.html'
```

The `TemplatedEmailMessage` class extends `django.core.mail.EmailMultiAlternatives`, so you don't have to do anything special to use it with your favorite backend.

If you install markdownify, you can omit the plaintext version of your message; mailmate will generate one automatically from the HTML version. Otherwise, omitting both `body` and `template_name` will cause a MissingBody exception to be raised. If you want to send an email without a plaintext body, you must set `body` to an empty string explicitly.

CHAPTER 3

CleanEmailBackend

Mailmate also includes a special backend to help you debug your emails. It's like Django's `django.core.mail.backends.filebased.EmailBackend`, but in addition to the *.log file, it will also save files containing the message body for each version of the message. For example, if you send an email that has both a plaintext and HTML version, it will save a *.log file (with the entire message), a *.txt file (with the plaintext body) and a *.html file (with the body of the HTML alternative).

To use it, set your `EMAIL_BACKEND` and `EMAIL_FILE_PATH` settings in settings.py:

```
EMAIL_BACKEND = 'mailmate.backends.CleanEmailBackend'
EMAIL_FILE_PATH = '/path/to/messages/'
```

Contents:

CHAPTER 4

Indices and tables

- genindex
- modindex
- search