django-mailrobot Documentation

Release v0.3

kaleissin

November 26, 2013

Contents

Stores and sends canned email responses.

Ever had to change the signature or add a recipient to N hardcoded emails spread all throughout your code? Hardcode no more! Use mailrobot instead.

Contents:

CHAPTER 1

Installation

1. Install library, for instance with pip:

   ```
   pip install django-mailrobot
   ```

2. Add library to your INSTALLED_APPS in your settings:

   ```
   INSTALLED_APPS += ['mailrobot']
   ```

3. Add the tables.

   Prior to django 1.7:

   ```
   $ ./manage.py syncdb
   ```

   With South:

   ```
   $ ./manage.py schemamigration --initial mailrobot
   $ ./manage.py migrate mailrobot
   ```

CHAPTER 2

Usage

Add mails and addresses through the django admin.

2.1 In code

Fetch a mail-template:

```
template = Mail.objects.get(name='hello-world').
```

Fill it:

```
mail = template.make_message(
    sender='Yep <overridden-from@example.com'>,
    recipients=('extra1@example.com', u'Blåbærsyltetøy <extra2@example.com>'),
    context={'world': 'Mailrobot'}
)
```

Have a look:

```
print mail.message
```

Send it:

```
mail.send()
```

CHAPTER 3

Niceties

In case you need to send an email somewhere else for testing/debugging, clone an existing email in the admin:

1. Select it
2. Choose “Clone selected mails” in the action list
3. Hit “Go”

The clone will share everything with its original except the name, which will be suffixed with a timestamp.

Edit the name of the clone to what you need, change recipients, CCs, BCCs. Then, where you send the mail from, choose the clone if

CHAPTER 4

Models

class mailrobot.models.**AbstractNamedModel**(**args*, ***kwargs*)
Bases: django.db.models.base.Model

class Meta

abstract = False

AbstractNamedModel.**NAME_MAX_LENGTH = 40**

AbstractNamedModel.**clone**()
Clone and return a Named model

AbstractNamedModel.**natural_key**()

class mailrobot.models.**Address**(**args*, ***kwargs*)
Bases: django.db.models.base.Model

Address(id, address, comment)

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception Address.**MultipleObjectsReturned**
Bases: django.core.exceptions.MultipleObjectsReturned

Address.**bcc**

Address.**cc**

Address.**natural_key**()

Address.**recipients**

Address.**reply_to**

Address.**sender**

class mailrobot.models.**MailBody**(**args*, ***kwargs*)
Bases: mailrobot.models.AbstractNamedModel

Subject and bodytext of the email

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception MailBody.**MultipleObjectsReturned**
Bases: django.core.exceptions.MultipleObjectsReturned

MailBody.**mail**

class mailrobot.models.**Mail**(**args*, ***kwargs*)
Bases: mailrobot.models.AbstractNamedModel

Canned Mail with default sender, Reply-To and recipients

Verifies that there is a sender and at least one recipient.

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception Mail.**MultipleObjectsReturned**
Bases: django.core.exceptions.MultipleObjectsReturned

Mail.**attach_signature**(*context=None*)
Attach signature, if any

Mail.**bccs**

Mail.**body**

Mail.**ccs**

Mail.**clone**()
Clone and return a Mail

Use this to send the same MailBody to several sets of recipients.

Mail.**content**

Mail.**get_bccs**(*additional=()*)
Get recipients for BCC:

May be empty.

Mail.**get_ccs**(*additional=()*)
Get recipients for CC:

May be empty.

Mail.**get_recipients**(*additional=()*, *required=True*)
Get recipients for To: and ensures there is at least one.

Email lacking anything in To: is likely spam.

Mail.**get_reply_to**(*reply_to=u''*)
Reply-To may be empty

Mail.**get_sender**(*sender=None*)
Sender may not be empty

Mail.**make_content**(*context=None*)
Generate the content (body + signature) from django templates and context

Mail.**make_message**(*sender=None*, *recipients=()*, *ccs=()*, *bccs=()*, *reply_to=None*, *headers=None*, *context=None*)
Generate a django.core.mail.EmailMessage

sender and **reply_to** may be overridden. **recipients**, **ccs** and **bccs** may be supplemented.

Verfies that there is a sender and at least one recipient.

Mail.**make_subject**(*context=None*)
Generate the subject from a django template and context

Mail.**recipients**

Mail.**reply_to**

Mail.**send**(***kwargs*)
Use django's email backend system to send the Mail

Mail.**sender**

Mail.**signature**

Mail.**subject**

Mail.**validate_addresses**(*sender=None*, *recipients=()*, *ccs=()*, *bccs=()*)

class mailrobot.models.**Signature**(**args*, ***kwargs*)
Bases: mailrobot.models.AbstractNamedModel

Email signature

No validation as to size.

exception DoesNotExist
Bases: django.core.exceptions.ObjectDoesNotExist

exception Signature.**MultipleObjectsReturned**
Bases: django.core.exceptions.MultipleObjectsReturned

Signature.**attach**(*context=None*)
Signature is attached so:

the final line of content

– signature

Signature.**mail**

CHAPTER 5

Indices and tables

- *genindex*
- *modindex*
- *search*