django-mailer2 Documentation

Release master

Tauber, Beaven & APSL

April 22, 2016

Contents

Django mailer is used to queue e-mails. This allows the emails to be sent asynchronously (by the use of a command extension) rather than blocking the response.

Contents:

CHAPTER 1

Installation

An obvious prerequisite of Django Mailer 2 is Django - 1.1 is the minimum supported version.

1.1 Installing django-mailer-2

Download and install from http://github.com/SmileyChris/django-mailer-2.git

If you're using pip and a virtual environment, this usually looks like:

```
pip install -e git+http://github.com/SmileyChris/django-mailer-2.git#egg=django-mailer-2
```

Or for a manual installation, once you've downloaded the package, unpack it and run the `setup.py` installation script:

```
python setup.py install
```

1.2 Configuring your project

In your Django project's settings module, add django_mailer to your `INSTALLED_APPS` setting:

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

Note that django mailer doesn't implicitly queue all django mail (unless you tell it to). More details can be found in the usage documentation.

CHAPTER 2

Usage

django-mailer-2 is asynchronous so in addition to putting mail on the queue you need to periodically tell it to clear the queue and actually send the mail.

The latter is done via a command extension.

2.1 Putting Mail On The Queue (Django 1.2 or higher)

In settings.py, configure Django's EMAIL_BACKEND setting like so:

> EMAIL_BACKEND = 'django_mailer.smtp_queue.EmailBackend'

If you don't need message priority support you can call send_mail like you normally would in Django:

```
send_mail(subject, message_body, settings.DEFAULT_FROM_EMAIL, recipients)
```

If you need prioritized messages, create an instance of EmailMessage and specify {'X-Mail-Queue-Priority': '<value>'} in the `headers` parameter, where <value> is one of:

> 'now' - do not queue, send immediately 'high' - high priority 'normal' - standard priority - this is the default. 'low' - low priority

If you don't specify a priority, the message is sent at 'normal' priority.

2.2 Putting Mail On The Queue (Django 1.1 or earlier)

Because django-mailer currently uses the same function signature as Django's core mail support you can do the following in your code:

```
# favour django-mailer-2 but fall back to django.core.mail
from django.conf import settings

if "django_mailer" in settings.INSTALLED_APPS:
    from django_mailer import send_mail
else:
    from django.core.mail import send_mail
```

and then just call send_mail like you normally would in Django:

```
send_mail(subject, message_body, settings.DEFAULT_FROM_EMAIL, recipients)
```

Additionally you can send all the admins as specified in the `ADMIN` setting by calling:

```
mail_admins(subject, message_body)
```

or all managers as defined in the `MANAGERS` setting by calling:

```
mail_managers(subject, message_body)
```

2.3 Command Extensions

With mailer in your INSTALLED_APPS, there will be four new manage.py commands you can run:

- `send_mail` will clear the current message queue. If there are any failures, they will be marked deferred and will not be attempted again by `send_mail`.
- `retry_deferred` will move any deferred mail back into the normal queue (so it will be attempted again on the next `send_mail`).
- `cleanup_mail` will delete mails created before an X number of days (defaults to 90).
- **`status_mail` the intent of this commant is to allow systems as nagios to** be able to ask the queue about its status. It returns as string with than can be parses as `(?P<queued>\d+)/(?P<deferred>\d+)/(?P<seconds>\d+)`

You may want to set these up via cron to run regularly:

```
* * * * * (cd $PROJECT; python manage.py send_mail >> $PROJECT/cron_mail.log 2>&1)
0,20,40 * * * * (cd $PROJECT; python manage.py retry_deferred >> $PROJECT/cron_mail_deferred.log 2>&
0 1 * * * (cd $PROJECT; python manage.py cleanup_mail --days=30 >> $PROJECT/cron_mail_cleanup.log 2>
```

This attempts to send mail every minute with a retry on failure every 20 minutes and will run a cleanup task every day cleaning all the messaged created before 30 days.

`manage.py send_mail` uses a lock file in case clearing the queue takes longer than the interval between calling `manage.py send_mail`.

Note that if your project lives inside a virtualenv, you also have to execute this command from the virtualenv. The same, naturally, applies also if you're executing it with cron.

CHAPTER 3

Settings

Following is a list of settings which can be added to your Django settings configuration. All settings are optional and the default value is listed for each.

3.1 MAILER_PAUSE_SEND

Provides a way of temporarily pausing the sending of mail. Defaults to `False`.

If this setting is `True`, mail will not be sent when the `send_mail` command is called.

3.2 MAILER_USE_BACKEND

Django 1.2 setting

The mail backend to use when actually sending e-mail. Defaults to `'django.core.mail.backends.smtp.EmailBackend'`

3.3 MAILER_MAIL_ADMINS_PRIORITY

The default priority for messages sent via the `mail_admins` function of Django Mailer 2.

The default value is `constants.PRIORITY_HIGH`. Valid values are `None` or any of the priority from `django_mailer.constants`: `PRIORITY_EMAIL_NOW`, `PRIORITY_HIGH`, `PRIORITY_NORMAL` or `PRIORITY_LOW`.

3.4 MAILER_MAIL_MANAGERS_PRIORITY

The default priority for messages sent via the `mail_managers` function of Django Mailer 2.

The default value is `None`. Valid values are the same as for *MAILER_MAIL_ADMINS_PRIORITY*.

3.5 MAILER_EMPTY_QUEUE_SLEEP

For use with the `django_mailer.engine.send_loop` helper function.

When queue is empty, this setting controls how long to wait (in seconds) before checking again. Defaults to `30`.

3.6 MAILER_LOCK_WAIT_TIMEOUT

A lock is set while the `send_mail` command is being run. This controls the maximum number of seconds the command should wait if a lock is already in place.

The default value is `-1` which means to never wait for the lock to be available.

CHAPTER 4

Django Mailer fork

django-mailer-2 is a fork form Chris Beaven fort to of James Tauber's *django-mailer*.__

This document is readthedocs version of the fork that Chris and James made the original document with some additional information.

comments.

4.1 History

Chris Beaven started a fork of django-mailer and it got to the point when it would be rather difficult to merge back. The fork was then renamed to the completely unimaginative "django mailer 2".

In hindsight, this was a bad naming choice as it wasn't supposed to reflect that this is a "2.0" version or the like, simply an alternative.

The application namespace was changed (django-mailer-2 uses `django_mailer` whereas django-mailer uses `mailer`), allowing the two products to technically live side by side in harmony. One of the motivations in doing this was to make the transition simpler for projects which are using django-mailer (or to transition back, if someone doesn't like this one).

I made an additional fork as I need to correct some bugs related to unicode mail and add some interesting patches as the one which allows you to remove mails.

4.2 Differences

Some of the larger differences in django-mailer-2:

- It saves a rendered version of the email instead - so HTML and other attachments are handled fine
- The models were completely refactored for a better logical separation of data.
- It provides a hook to override (aka "monkey patch") the Django `send_mail`, `mail_admins` and `mail_manager` functions.
- Added a management command to remove old e-mails, so the database does not increase so much.
- Added a new testing procedure, so you can run the tests without having to install and configure a Django application.
- Added some cron templates ein *bin* folder to help you to configure the cron.
- Improved admin configuration.

- Added a demo project, which shows how we can retrieve an email stored in the database and shows django-mailer in the admin.

4.3 Credit

At the time of the fork, the primary authors of django-mailer were James Tauber and Brian Rosner. The additional contributors included Michael Trier, Doug Napoleone and Jannis Leidel.

Original branch and the django-mailer-2 hard work comes from Chris Beaven.