latest

Release 0.3.3

Mar 20, 2017

Contents

Created by Stephen McDonald

django-email-extras is a Django reusable app providing the ability to send PGP encrypted and multipart emails using Django templates. These features can be used together or separately. When configured to send PGP encrypted email, the ability for Admin users to manage PGP keys is also provided.

A tool for automatically opening multipart emails in a local web browser during development is also provided.

CHAPTER 1

Dependencies

- python-gnupg is required for sending PGP encrypted email.

CHAPTER 2

Installation

The easiest way to install django-email-extras is directly from PyPi using pip by running the command below:

```
$ pip install -U django-email-extras
```

Otherwise you can download django-email-extras and install it directly from source:

```
$ python setup.py install
```

CHAPTER 3

Usage

Once installed, first add `email_extras` to your `INSTALLED_APPS` setting and run the migrations. Then there are two functions for sending email in the `email_extras.utils` module:

- `send_mail`
- `send_mail_template`

The former mimics the signature of `django.core.mail.send_mail` while the latter provides the ability to send multipart emails using the Django templating system. If configured correctly, both these functions will PGP encrypt emails as described below.

CHAPTER 4

Sending PGP Encrypted Email

PGP explanation

Using python-gnupg, two models are defined in `email_extras.models` - `Key` and `Address` which represent a PGP key and an email address for a successfully imported key. These models exist purely for the sake of importing keys and removing keys for a particular address via the Django Admin.

When adding a key, the key is imported into the key ring on the server and the instance of the `Key` model is not saved. The email address for the key is also extracted and saved as an `Address` instance.

The `Address` model is then used when sending email to check for an existing key to determine whether an email should be encrypted. When an `Address` is deleted via the Django Admin, the key is removed from the key ring on the server.

CHAPTER 5

Sending Multipart Email with Django Templates

As mentioned above, the following function is provided in the `email_extras.utils` module:

```
send_mail_template(subject, template, addr_from, addr_to,
    fail_silently=False, attachments=None, context=None,
    headers=None)
```

The arguments that differ from `django.core.mail.send_mail` are `template` and `context`. The `template` argument is simply the name of the template to be used for rendering the email contents.

A template consists of both a HTML file and a TXT file each responsible for their respective versions of the email and should be stored in the `email_extras` directory where your templates are stored, therefore if the name `contact_form` was given for the `template` argument, the two template files for the email would be:

- `templates/email_extras/contact_form.html`
- `templates/email_extras/contact_form.txt`

The `attachments` argument is a list of files to attach to the email. Each attachment can be the full filesystem path to the file, or a file name / file data pair.

The `context` argument is simply a dictionary that is used to populate the email templates, much like a normal request context would be used for a regular Django template.

The `headers` argument is a dictionary of extra headers to put on the message. The keys are the header name and values are the header values.

CHAPTER 6

Configuration

There are two settings you can configure in your project's `settings.py` module:

- `EMAIL_EXTRAS_USE_GNUPG` - Boolean that controls whether the PGP encryption features are used. Defaults to `True` if `EMAIL_EXTRAS_GNUPG_HOME` is specified, otherwise `False`.
- `EMAIL_EXTRAS_GNUPG_HOME` - String representing a custom location for the GNUPG keyring.
- `EMAIL_EXTRAS_GNUPG_ENCODING` - String representing a gnupg encoding. Defaults to GNUPG `latin-1` and could be changed to e.g. `utf-8` if needed. Check out python-gnupg docs for more info.
- `EMAIL_EXTRAS_ALWAYS_TRUST_KEYS` - Skip key validation and assume that used keys are always fully trusted.

CHAPTER 7

Local Browser Testing

When sending multipart emails during development, it can be useful to view the HTML part of the email in a web browser, without having to actually send emails and open them in a mail client. To use this feature during development, simply set your email backend as follows in your development `settings.py` module:

```
EMAIL_BACKEND = 'email_extras.backends.BrowsableEmailBackend'
```

With this configured, each time a multipart email is sent, it will be written to a temporary file, which is then automatically opened in a local web browser. Suffice to say, this should only be enabled during development!