
django-messages Documentation

Release 0.5.0

Arne Brodowski

Aug 16, 2019

Contents

1	Contents	3
1.1	Installing django-messages	3
1.2	Using django-messages	4
1.3	Customizing django-messages	6

Django-messages is a Django app which provides a user-to-user messaging system. It enables your users to send private messages to each other. It provides a basic set of functionality you would expect from such a system. Every user has an inbox, an Outbox and a Trash. Messages can be composed and there is an easy url-based approach to preloading the compose-form with the recipient user, which makes it extremely easy to put “send user a message” links on profile pages.

Currently django-messages comes with these translations:

- ar
- da
- de
- el
- es
- es_AR
- fa
- fr
- it
- lt
- ko
- nl
- pl
- pt_BR
- ru
- zh_CN

1.1 Installing django-messages

Basically all you have to do is get the `messages` folder somewhere on the Python path. There are multiple ways to achieve this.

1.1.1 Quickstart

If you already downloaded the package change into the `django-messages` directory and run:

```
python setup.py install
```

Otherwise you will find more information in the remainder of this document.

Django-messages is available via PyPi, so the following command will download and install django-messages on your system in one step:

```
easy_install django-messages
```

If you prefer using pip, you may achieve the same result by running:

```
pip install django-messages
```

1.1.2 Download

You will always find and download the latest packaged version at: <http://code.google.com/p/django-messages/downloads/list>

If you prefer to use the current development version to get earlier access to new features you can checkout the code from the GIT repository:

```
git clone https://github.com/arneb/django-messages.git
```

1.1.3 Install

If you downloaded the tar-ball extract it with (change the version number if required):

```
tar -xcvf django-messages-0.4.tar.gz
```

After extracting the tar-ball or checking out the code from the repository, change into the `django-messages` directory and install the code:

```
cd django-messages
python setup.py install
```

1.1.4 Manual Install

Instead of using `setup.py install` it is also possible to copy or symlink the `django_messages` folder inside the toplevel `django-messages` folder to your Python path. This will be enough to make `django-messages` available to your system.

1.1.5 Dependencies

Django-messages has no external dependencies except for Django. Starting with version 0.4 Django 1.1 or later is required. Version 0.3 works with Django 1.0. If you have to use Django 0.96.x you might still use version 0.2 (unsupported).

Django-messages has some features which may use an external app if it is present in the current Django project. Please note, that these apps have to be listed in `INSTALLED_APPS` to be used by `django-messages`.

- If you use [pinax-notifications](#) `django-messages` will use it for sending notifications to users about new messages instead of using the built-in mechanism
- If [django-mailer](#) is used the built-in messages sending code will use it instead of the django built-in `send_mail` function.

Please note that we try to support Python 2.4, but as we develop `django-messages` on Python 2.5 sometimes code might slip in which breaks Python 2.4 compatibility. This can be considered a bug and you can help us by reporting the incompatibility.

1.2 Using django-messages

To enable `django-messages` in your Django project make sure it is *installed*. You can check if `django-messages` was successfully installed by opening a python shell and running:

```
>>> import django_messages
>>>
```

If no error occurred, you can assume that the app was installed correctly.

1.2.1 Edit settings

The next step is to add `django_messages` to the `INSTALLED_APPS` setting:

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

1.2.2 Add urls

To make django-messages available to your users you should include the bundled url-conf in your root url-conf. One example would be to edit your main `urls.py` and add a line like this:

```
urlpatterns = patterns('
    ...
    (r'^messages/', include('django_messages.urls')),
    ...
)
```

1.2.3 Templates

Django-messages provides some simple default templates which will get you started quickly. The templates make the assumption that a base template with the name `base.html` exists which defines a block `content` and a block `sidebar`. If this is not the case, or the template doesn't fit due to other concerns, it's very easy to provide your own templates. Please see the [customization docs](#) for more details.

1.2.4 Templatetags and Context-Processors

Django-messages provides a Templatetag and a Template Context Processor to make it easy to print the number of unread messages of a user in the templates.

To use the Templatetag simply add this to your template:

```
{% load inbox %}
```

Now you can either print the number of unread messages in the users inbox by using:

```
{% inbox_count %}
```

Or you can assign the count to a variable to further process it in the template:

```
{% inbox_count as my_var %}
{{ my_var }}
```

If you want to show the inbox count on every page of your site you could also use the bundled Context Processor to add the value to every Template Context instead of loading the Templatetag. Simply add the Context Processor to the `TEMPLATE_CONTEXT_PROCESSORS` settings in your `settings.py`:

```
TEMPLATE_CONTEXT_PROCESSORS = (
    ...
    'django_messages.context_processors.inbox',
)
```

And now every Template Context will contain a variable named `messages_inbox_count`, if the user is logged in:

```
{{ messages_inbox_count }}
```

1.2.5 Settings Options

If you do want to disable django-messages from sending either a ‘pinax-notifications’ notice or an email (fallback if ‘pinax-notifications not installed’ then set the following in your django settings:

```
DJANGO_MESSAGES_NOTIFY = False
```

1.3 Customizing django-messages

There are multiple levels at which you can customize django-messages without altering the code directly.

1.3.1 Templates

Django-messages comes with a set of built-in templates which you can use. If these templates don’t fit your project you can override any or all of them by putting files with the same filenames in one the directories listed in `TEMPLATES_DIRS` in your `settings.py`.

Django-messages uses the following templates:

- `django_messages/base.html` - A base template from which all the following templates inherit. Maybe it’s enough to customize this template for your project.
- `django_messages/compose.html` - This template is rendered, when a user composes a new messages.
- `django_messages/inbox.html` - This template lists the users inbox.
- `django_messages/new_messages.html` - This template is used to construct the notification mail sent to a user, whenever a new message is received.
- `django_messages/outbox.html` - This template lists the users outbox aka sent messages.
- `django_messages/trash.html` - This template lists the users trash.
- `django_messages/view.html` - This template renders a single message with all details.

Additionally django-message provides a set of template for pinax-notifications. These template can be found in `django_messages/templates/notification/` and can also be overwritten in one of your project’s `TEMPLATE_DIRS`.

1.3.2 URL-conf

If you want to further customize how django-messages works it is possible to write your own url-conf instead of including `django_messages.urls` in your root url-conf. This not only allows changing the url structure but also allows modifying the kwargs passed to the views and therefore modifying some behaviour.

Please note: If you provide your own url-conf, or urlpatterns directly embedded in your root url-conf, you shouldn't include `django_messages.urls`.

Three common customizations are described in more detail below.

Modifying template names

If overwriting templates in your project's `TEMPLATE_DIRS` does not provide enough freedom, you can change the names of the used templates by providing a *template_name* keyword argument to the views. Every view which renders a template accepts this keyword-argument.

If you want to change the template the `inbox` view uses to `my_inbox.html` instead of the default `django_messages/inbox.html` you can use this line in your own url-conf:

```
url(r'^inbox/$',
    inbox,
    {'template_name': 'my_inbox.html'},
    name='messages_inbox'),
```

Modifying form classes

If you want to use your own form for composing messages, for example to add new features, you can simply pass the form-class to the views via kwargs. Every view which renders a form accepts a *form_class* keyword argument to specify the form-class.

If you want to use Your own `MyComposeForm` you can pass it to the view by using a line like the following in your own url-conf:

```
from somewhere import MyComposeForm
...
url(r'^compose/$',
    compose,
    {'form_class': MyComposeForm},
    name='messages_compose'),
```

Modifying success urls

All views, which will redirect the user after a successful action accept a *success_url* keyword argument to specify the destination url. The `delete` and `undelete` views will additionally check if a *next* parameter is provided in the querystring appended to the url.

If you don't want to append the next target to the url, or want to change the redirecting behaviour of other views, you can pass a *success_url* parameter in your own url-conf, for example like this:

```
url(r'^delete/(?P<message_id>[\d]+)/$',
    delete,
    {'success_url': '/profile/'},
    name='messages_delete'),
```