django-simple-feedback-form

Release 2.0

Artem Galichkin

Feb 02, 2022

BASIC:

This Django app has for provide simple form for sending mail forom users to admins your django-based site.

CHAPTER

ONE

SETUP

1.1 Installation

Install into your python path using pip or github version:

```
pip install django-simple-feedback-form
```

or:

```
pip install git+https://github.com/DOOMer/django-simple-feedback-form.git
```

1.2 Preparing your project

Add ADMINS list to your settings.py, like that:

ADMIN = [('John', 'john@example.com'), ('Mary', 'mary@example.com')

]

In addition, you can add MANAGERS list to your settings.py, like that^

MANAGERS = [('John', 'john@example.com'), ('Mary', 'mary@example.com')

]

Warning: If you don't, you will get an exception (ImproperlyConfigured with message You must add ADMINS list variable in 'settings' module of your project'.) when sending mail.

CHAPTER TWO

USAGE

2.1 Using in your project

Add *'feedback_form'* to your INSTALLED_APPS in settings.py,

INSTALLED_APPS = (…

'feedback_form',

)

Add *'feedback_form.urls'* in your main urls configuration:

```
path('feedback/', include("feedback_form.urls", namespace="feedback_form")),
```

and add in your template add link to feedback view:

```
<a href="{% url 'feedback_form:feedback-view' %}">Feedback form</a>
```

And override the template on path "feedback_form/feedback.html" for your site.

2.2 Additional settings

You can override these settings in your main settngs.py file:

CONTACT_ADMINS_ONLY - send email only to users which added in ADMINS tuple. By default is enabled. If it is a disabled, send to MANAGERS mails too.

CONTACT_SEND_META_INFO- send meta information about user (IP and user-agent). By default is disabled.

CHAPTER

THREE

ADVANCED USAGE

3.1 Using without including ‘feedback.forms.urls’

You can directly use FeedbackView in your ‘urls.py’ module, without including ‘feedback_form.urls’ in the main urlconf of your project.

from feedback_form.views import FeedbackView

And use it in the urlpattern^

path(‘feedback/’, FeedbackView.as_view(), name=’feedback-view’),

Warning: Name of your urlpattern must be a ‘feedback-view’, because there is a reference to this name in the code of view.

CHAPTER

FOUR

BUILT-IN FORM CLASSES

You can create your own form inherited from the Base Form class. And then pass your new class as the value of the "form_class" parameter to the **FeedbackView.as_view()** call.

4.1 ContactFormBase class

`class feedback_form.forms.ContactFormBase`

The base contact form class from which all contact form classes should inherit. It will collect name, email address and message. You can to override this class attributes:

`subject_template`

A `str`, the name of the template to use when rendering the subject line of the message. By default, this is *feedback_form/email_subject.txt*.

`message_template`

A `str`, the name of the template to use when rendering the body of the message. By default, this is *feedback_form/email_template.txt*.

And two methods are involved in producing the contents of the message to send:

`get_message()`

Returns the body of the message to send. By default, this is accomplished by rendering the template name specified in *`message_template`*.

Return type str

`get_subject()`

Returns the subject line of the message to send. By default, this is accomplished by rendering the template name specified in *`subject_template`*.

Return type str

`**Subject must be a single line**`

The subject of an email is sent in a header (named *Subject:*). Because email uses newlines as a separator between headers, newlines in the subject can cause it to be interpreted as multiple headers; this is the header injection attack. To prevent this, `subject()` will always force the subject to a single line of text, stripping all newline characters. If you override `subject()`, be sure to either do this manually, or use `super()` to call the parent implementation.

`prepare_mail()`

This method loops through *`get_message()`* and *`get_subject()`*, collecting those parts into a dictionary with keys corresponding to the arguments to Django's *mail_admins* function, then returns the dictionary. Overriding this allows essentially unlimited customization of how the message is generated.

Return type dict

`get_request_meta()`
Returns the selected metadata from the `request_meta`. It is used if *CONTACT_SEND_META_INFO* is True in 'settings' module of the your project.

Return type dict

Meanwhile, the following attributes/methods generally should not be overridden; doing so may interfere with functionality, may not accomplish what you want, and generally any desired customization can be accomplished in a more straightforward way through overriding one of the attributes/methods listed above.

`request_meta`
The dict with some meta data from `HttpRequest` object representing the current request. This is set automatically in *__init__()*, and is used if *CONTACT_SEND_META_INFO* is True in 'settings' module of the your project.

CHAPTER

FIVE

BUILT-IN VIEW CLASSES

class feedback_form.views.FeedbackView

The base view class from which most custom contact-form views should inherit. If you don't need any custom functionality, and it is inherited from the `FeedbackBaseView` class.

You can also use it as-is (and the provided URLConf, *feedback_form.urls*, does exactly this).

class feedback_form.views.FeedbackBaseView

The base view class from which most custom contact-form views should inherit. If you don't need any custom functionality, and are content with the default `ContactFormBase` class.

This is a subclass of Django's `FormView`, so refer to the Django documentation for a list of attributes/methods which can be overridden to customize behavior.

The following standard (from `FormView`) methods and attributes are commonly useful to override (all attributes below can also be passed to `as_view()` in the URLconf, permitting customization without the need to write a full custom subclass of this class):

form_class

The form class to use. By default, will be `ContactForm`. This can also be overridden as a method named `form_class()`; this permits, for example, per-request customization (by inspecting attributes of *self.request*).

feedback_form.views.success_message

A `str`, the text, which displayed to user, after successful form submission. By default, will be "Message is sent successfully".

feedback_form.views.success_url

A `str`, the name of urlpattern to redirect to after successful form submission.

feedback_form.views.template_name

A `str`, the template to use when rendering the form. By default, will be *feedback_form/feedback.html*.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

INDEX