django-sms

Release 0.6.0

Roald Nefs

Dec 25, 2022

CONTENTS

django-sms is a Django app for sending SMS with interchangeable backends. The module is heavily based upon and structured the same way as the `django.core.mail` module.

CHAPTER

ONE

SENDING SMS

These wrappers are provided to make sending SMS extra quick, to help test SMS sending during development, and to provide additional SMS gateways.

1.1 Quick example

In two lines:

```
from sms import send_sms

send_sms(
    'Here is the message',
    '+12065550100',
    ['+441134960000'],
    fail_silently=False
)
```

The text messages are sent using one of the configured *SMS backends*.

1.2 send_sms()

send_sms(_body, originator, recipients, fail*silently=False, connection=None*)

In most cases, you can send text messages using **sms.send_sms()**.

The **message**, **originator** and **recipients** parameters are required.

- **message**: A string.
- **originator**: A string. If **None**, django-sms will use the **DEFAULT_FROM_SMS** setting.
- **recipients**: A list of strings, each an phone number.
- **fail_silently**: A boolean. When it's **False**, **send_sms()** will raise an exception if an error occurs. See the *SMS backends* documentation for a list of possible exceptions.
- **connection**: The optional SMS backend to use to send the text message. If unspecified, an instance of the default backend will be used. See the documentation of *SMS backends* for more details.

The return value will be the number of successfully delivered text messages.

CHAPTER

TWO

EXAMPLES

This sends a text message to *+44 113 496 0000* and *+44 113 496 0999*:

```
send_sms(
    'Here is the message',
    '+12065550100',
    ['+441134960000', '+441134960999']
)
```

2.1 The Message class

django-sms' **send_sms()** function is actually a thin wrapper that makes use of the **Message** class.

Not all features of the **Message** class will be available though the **send_sms()** and related wrapper functions. If you wish to use advanced features, you'll need to create **Message** instances directly.

Note: This is a design feature. **send_sms()** was originally the only interfaces django-sms provided.

Message is responsible for creating the text message itself. The SMS backend is then responsible for sending the SMS.

For convenience, *Message** provides a **send()** method for sending a single text message. If you need to send multiple text messages, the SMS backend API provides alternatives.

2.1.1 Message Objects

***class* Message**

The **Message** class is initialized with the following parameters (in the given order, if positional arguments are used). All parameters are optional and can be set at any time prior to calling the **send()** method.

- **body**: The body text. This should be a plain text message.
- **originator**: The sender of the text message.
- **recipients**: A list or tuple of recipient phone numbers.
- **connection**: An SMS backend instance. Use this parameter if you want to use the same connection for multiple text messages. If omitted, a new connection is created when **send()** is called.

For example:

```
from sms import Message

message = Message(
    'Here is the message',
    '+12065550100',
    ['+441134960000']
)
```

The class has the following methods:

- **send(fail_silently=False)** sends the text message. If a connection was specified when the text message was constructed, that connection will be used. Otherwise, an instance of the default backend will be instantiated and used. If the keyword argument **fail_silently** is **True**, exceptions raised while sending the text messages will be quashed. An empty list of recipients will not raise an exception.

2.2 SMS backends

The actual sending of an SMS is handled by the SMS backend.

The SMS backend class has the following methods:

- **open()** instantiates a long-lived SMS-sending connection.
- **close()** closes the current SMS-sending connection.

It can also be used as a context manager, which will automatically call **open()** and **close()** as needed:

```
import sms

with sms.get_connection() as connection:
    sms.Message(
        'Here is the message', '+12065550100', ['+441134960000'],
        connection=connection
    ).send()
    sms.Message(
        'Here is the message', '+12065550100', ['+441134960000'],
        connection=connection
    ).send()
```

2.3 Obtaining an instance of an SMS backend

The **sms.get_connection()** function in **sms** returns an instance of the SMS backend that you can use.

get_connection(_backend=None, fail_silently=False, *args, **kwargs_)

By default, a call to **get_connection()** will return an instance of the SMS backend specified in **SMS_BACKEND**. If you specify the **backend** argument, an instance of that backend will be instantiated.

The **fail_silently** argument controls how the backend should handle errors. If **fail_silently** is True, exceptions during the SMS sending process will be silently ignored.

All other arguments are passed directly to the constructor of the SMS backend.

django-sms ships with several SMS sending backends. Some of these backends are only useful during testing and development. If you have special SMS sending requirements, you can *write your own SMS backend*.

2.3.1 Console backend

Instead of sending out real text messages the console backend just writes the text messages that would be sent to the standard output. By default, the console backend writes to **stdout**. You can use a different stream-like object by providing the **stream** keyword argument when constructing the connection.

```
SMS_BACKEND = 'sms.backends.console.SmsBackend'
```

This backend is not intended for use in production - it is provided as a convenience that can be used during development.

2.3.2 File backend

The file backend writes text messages to a file. A new file is created for each session that is opened on this backend. The directory to which the files are written is either taken from the **SMS_FILE_PATH** setting or file the **file_path** keyword when creating a connection with **get_connection()**.

To specify this backend, put the following in your settings:

```
SMS_BACKEND = 'sms.backends.filebased.SmsBackend'
SMS_FILE_PATH = '/tmp/app-messages' # change this to a proper location
```

This backend is not intended for use in production - it is provided as a convenience that can be used during development.

2.3.3 In-memory backend

The **'locmen'** backend stores text messages in a special attribute of the **sms** module. The **outbox** attribute is created when the first message is sent. It's a list with an **Message** instance of each text message that would be sent.

To specify this backend, put the following in your settings:

```
SMS_BACKEND = 'sms.backends.locmem.SmsBackend'
```

This backend is not intended for use in production - it is provided as a convenience that can be used during development.

2.3.4 Dummy backend

As the name suggests the dummy backend does nothing with your text messages. To specify this backend, put the following in your settings:

```
SMS_BACKEND = 'sms.backends.dummy.SmsBackend'
```

This backend is not intended for use in production - it is provided as a convenience that can be used during development.

2.3.5 MessageBird backend

The MessageBird backend sends text messages using the MessageBird SMS API. To specify this backend, put the following in your settings:

```
SMS_BACKEND = 'sms.backends.messagebird.SmsBackend'
MESSAGEBIRD_ACCESS_KEY = 'live_redacted-messagebird-access-key'
```

Make sure the MessageBird Python SDK is installed by running the following command:

```
pip install "django-sms[messagebird]"
```

2.3.6 Twilio backend

The Twilio backend sends text messages using the Twilio SMS API. To specify this backend, put the following in your settings:

```
SMS_BACKEND = 'sms.backends.twilio.SmsBackend'
TWILIO_ACCOUNT_SID = 'live_redacted-twilio-account-sid'
TWILIO_AUTH_TOKEN = 'live_redacted-twilio-auth-token'
```

Make sure the Twilio Python SDK is installed by running the following command:

```
pip install "django-sms[twilio]"
```

2.4 Defining a custom SMS backend

If you need to change how text messages are sent you can write your own SMS backend. The **SMS_BACKEND** setting in your settings file is then the Python import path for you backend class.

Custom SMS backends should subclass **BaseSmsBackend** that is located in the **sms.backends.base** module. A custom SMS backend must implement the **send_messages(messages)** method. This methods receives a list of **Message** instances and returns the number of successfully delivered messages. If your backend has any concept of a persistent session or connection, you should also implement **open()** and **close()** methods. Refer to one of the existing SMS backends for a reference implementation.

2.5 Signals

django-sms provides a set of built-in signals that let user code get notified by Django itself of certain actions. These include some useful notifications:

2.5.1 sms.signals.post_send

Sent after **send()** is called on a **Message** instance. Arguments sent with this signal:

- **sender**: The **Message** class.
- **instance**: The actual **Message** instance being send.

CHAPTER

THREE

ACKNOWLEDGEMENT

This project is heavily based upon the **django.core.mail** module, with the modified work by Roald Nefs. The Django license is included with **django-sms**.