

---

# **django-soapbox Documentation**

***Release 1.2***

**James Bennett**

September 05, 2015



<b>1</b>	<b>Documentation contents</b>	<b>3</b>
1.1	Installation guide . . . . .	3
1.2	Usage overview . . . . .	4
<b>2</b>	<b>Indices and tables</b>	<b>7</b>



This application provides a simple mechanism for creating and displaying messages – such as announcements or site information – on a [Django](#)-powered site. Messages can be turned on or off, and can be set to display globally or only on a subset of a site's URLs.



---

## Documentation contents

---

### 1.1 Installation guide

Before installing `django-soapbox`, you'll need to have a copy of [Django](#) already installed. For information on obtaining and installing Django, consult the [Django download page](#), which offers convenient packaged downloads and installation instructions.

The 1.2 release of `django-soapbox` supports Django 1.7 and 1.8, on any of Python 2.7, 3.3 or 3.4. Older versions of Django and/or Python may work, but are not tested or officially supported.

#### 1.1.1 Normal installation

The preferred method of installing `django-soapbox` is via `pip`, the standard Python package-installation tool. If you don't have `pip`, instructions are available for [how to obtain and install it](#).

Once you have `pip`, simply type:

```
pip install django-soapbox
```

#### 1.1.2 Manual installation

It's also possible to install `django-soapbox` manually. To do so, obtain the latest packaged version from [the listing on the Python Package Index](#). Unpack the `.tar.gz` file, and run:

```
python setup.py install
```

Once you've installed `django-soapbox`, you can verify successful installation by opening a Python interpreter and typing `import soapbox`.

If the installation was successful, you'll simply get a fresh Python prompt. If you instead see an `ImportError`, check the configuration of your install tools and your Python import path to ensure `django-soapbox` installed into a location Python can import from.

#### 1.1.3 Installing from a source checkout

The development repository for `django-soapbox` is at [<https://github.com/ubernostrum/django-soapbox>](https://github.com/ubernostrum/django-soapbox). Presuming you have [git](#) installed, you can obtain a copy of the repository by typing:

```
git clone https://github.com/ubernostrum/django-soapbox.git
```

From there, you can use normal git commands to check out the specific revision you want, and install it using `python setup.py install`.

## 1.2 Usage overview

The goal of django-soapbox is to provide a simple way to display persistent messages on either all pages, specific pages, or a subset of pages on a Django-powered site. To begin using django-soapbox, simply *install it*, then add `soapbox` to your `INSTALLED_APPS` setting and run `manage.py migrate` to install the `Message` model.

You can then begin creating `Message` instances through the admin interface, indicating which URLs you'd like them to appear on.

### 1.2.1 Provided models

#### class `Message`

The core of django-soapbox is the `Message` model, which represents messages to be displayed on your site. This model has four fields and one important custom method:

##### `message`

The actual text of the message to display. This can be plain text, or it can include HTML.

##### `is_active`

A `BooleanField` (defaults to `True`) indicating whether the message is currently active; only active messages will be retrieved by the standard helpers built in to django-soapbox.

##### `is_global`

A `BooleanField` (defaults to `False`) indicating whether the message is global; a global message does not need to have `url` (see below) set, and will match any URL.

##### `url`

A field to indicate which URL on your site this message should be associated with. Not needed if `is_global` is `True`.

##### `match(url)`

Return `True` if this `Message` matches `url`, `False` otherwise. If `is_global` is `True`, will always return `True`.

#### class `MessageManager`

Also provided on `Message` is a custom manager, accessible as `Message.objects`, which defines two useful methods:

##### `active()`

Returns a `QuerySet` of all `Message` instances which have `is_active` set to `True`. This is defined as a custom `QuerySet` method, so it can also be “chained” onto other `QuerySets`. For example, the following would retrieve all `Message` instances which are both global and active:

```
Message.objects.filter(is_global=True).active()
```

##### `match(url)`

Return a list – *not* a `QuerySet` – of all `Message` instances which match `url`.

### 1.2.2 Validation requirements

While `Message` instances are relatively freeform, there are two requirements you must abide by; failure to do so will result in validation errors being raised when trying to save the `Message`:

1. Each `Message` must either have `is_global` set to `True`, or specify some URL prefix to match in `url`.
2. A `Message` cannot have both `is_global` set to `True` and simultaneously have a URL prefix to match specified in `url` (in other words, a `Message` can be global, or “local” to some URL prefix, but never both at the same time).

### 1.2.3 Message URL matching

The message-retrieval helpers provided in `django-soapbox` will only retrieve messages which are active and which match a particular URL you pass to them; typically, this will be the URL of the current request. The matching process is case-sensitive and uses the following algorithm, implemented in the `match()` method of `Message`.

1. If the `Message` has `is_global` set to `True`, immediately return `True`.
2. Strip leading and trailing slashes from the URL, and from the `url` field of the `Message`, and split each on internal slashes to yield a list of path components.
3. If the list of components from the `url` field of the `Message` is longer than the list from the passed-in URL, immediately return `False`.
4. Return `True` if the list of components from the `url` field, and the corresponding list of components from the beginning of the passed-in URL, are equal. Otherwise, return `False`.

This means that a `Message` will match not only a URL which is an exact match for its own `url`, but also any URL of which its `url` is a prefix. So, for example, if the `url` field contained `/foo/`, it would match on `/foo/` and on `/foo/bar/`.

### 1.2.4 Retrieving and displaying messages

There are two helpers built in to `django-soapbox` for retrieving and displaying messages in templates.

One is a context processor, which will add a variable `soapbox_messages` to the context of any template rendered with a `RequestContext` (required in order to have access to the request path to determine the URL). To enable it, simply add `soapbox.context_processors.soapbox_messages` to your `TEMPLATE_CONTEXT_PROCESSORS` setting.

If you prefer to have more fine-grained control of where messages will be retrieved and displayed, `django-soapbox` provides a template tag, `get_soapbox_messages` which can retrieve messages for a given URL and place them into a variable in the context. The syntax of the tag is:

```
{% get_messages_for_page [url] as [varname] %}
```

To use the tag, first add `{% load soapbox %}` to the template to load the `django-soapbox` template tag library, then call the `get_messages_for_page` tag, passing a URL – either a string, or a template variable which the tag will resolve – and the name of the context variable you’d like the message to be placed into. For example:

```
{% load soapbox %}
{% get_messages_for_page request.path as soapbox_messages %}

{% for message in soapbox_messages %}
  <p>Important message: {{ message }}</p>
{% endfor %}
```

### 1.2.5 What django-soapbox is not

Importantly, django-soapbox is not a system for displaying one-time “flash”-type notifications to an individual user; for that, use Django’s [built-in message framework](#). It also is not a system for users to send messages to each other; for that, email or a custom user-message tool is more appropriate.

Instead, django-soapbox is for displaying messages to *all* users, on any URLs the messages match, each time they visit those URLs. Most often this is useful for site-wide or section-specific announcements all users need to see.

### 1.2.6 Security considerations

The tools provided in django-soapbox are designed around the assumption that only trusted administrators of your site will be permitted to create [Message](#) instances. In particular, a `Message` will, by default, mark its contents as safe for display, and so the Django template system will *not* perform autoescaping of the contents. This is useful for allowing HTML messages – for example, containing links to longer announcements on their own pages – but if opened to arbitrary or untrusted users would be a serious [cross-site scripting vulnerability](#).

Because of this, it is recommended that you only use the Django administrative interface to create `Message` instances, and that you carefully restrict the `soapbox.add_message` permission to only a small number of trusted administrators.

---

## Indices and tables

---

- `genindex`
- `modindex`
- `search`



## A

`active()` (MessageManager method), 4

## I

`is_active` (Message attribute), 4

`is_global` (Message attribute), 4

## M

`match()` (Message method), 4

`match()` (MessageManager method), 4

Message (built-in class), 4

`message` (Message attribute), 4

MessageManager (built-in class), 4

## U

`url` (Message attribute), 4