

---

# **django-admin-tools Documentation**

*Release 0.6.0*

**David Jean Louis**

May 21, 2016



<b>1</b>	<b>Quick start guide</b>	<b>3</b>
1.1	Installing django-admin-tools . . . . .	3
1.2	Basic configuration . . . . .	3
1.3	Testing your new shiny admin interface . . . . .	4
<b>2</b>	<b>Installation guide</b>	<b>5</b>
2.1	Requirements . . . . .	5
2.2	Installing django-admin-tools . . . . .	5
<b>3</b>	<b>Configuring django-admin-tools</b>	<b>7</b>
3.1	Basic configuration . . . . .	7
3.2	Available settings variables . . . . .	8
<b>4</b>	<b>Customization of the django-admin-tools modules</b>	<b>9</b>
4.1	Introduction . . . . .	9
4.2	Customizing the navigation menu . . . . .	9
4.3	Customizing the dashboards . . . . .	10
4.4	Customizing the theme . . . . .	10
<b>5</b>	<b>Working with multiple admin sites</b>	<b>11</b>
5.1	Introduction . . . . .	11
5.2	Setting up a different dashboard and menu for each admin site instance . . . . .	11
<b>6</b>	<b>The django-admin-tools menu and menu items API</b>	<b>13</b>
6.1	The Menu class . . . . .	13
6.2	The MenuItem class . . . . .	13
6.3	The AppList class . . . . .	13
6.4	The ModelList class . . . . .	13
6.5	The Bookmarks class . . . . .	13
<b>7</b>	<b>The django-admin-tools dashboard and dashboard modules API</b>	<b>15</b>
7.1	The Dashboard class . . . . .	15
7.2	The AppIndexDashboard class . . . . .	15
7.3	The DashboardModule class . . . . .	15
7.4	The Group class . . . . .	15
7.5	The LinkList class . . . . .	15
7.6	The AppList class . . . . .	15
7.7	The ModelList class . . . . .	15
7.8	The RecentActions class . . . . .	15

7.9	The <code>Feed</code> class . . . . .	15
<b>8</b>	<b>Integration with third party applications</b>	<b>17</b>
<b>9</b>	<b>Contributing to <code>django-admin-tools</code></b>	<b>19</b>
<b>10</b>	<b>Testing of <code>django-admin-tools</code></b>	<b>21</b>
10.1	Running tests . . . . .	21
10.2	Code coverage report . . . . .	21
10.3	Where tests live . . . . .	21
	<b>Python Module Index</b>	<b>23</b>

This documentation covers the latest release of django-admin-tools, a collection of extensions and tools for the Django administration interface, django-admin-tools includes:

- a full featured and customizable dashboard (for the admin index page and the admin applications index pages),
- a customizable menu bar,
- tools to make admin theming easier.

To get up and running quickly, consult the *quick-start guide*, which describes all the necessary steps to install django-admin-tools and configure it for the default setup. For more detailed information about how to install and how to customize django-admin-tools, read through the documentation listed below.

Contents:



---

## Quick start guide

---

Before installing `django-admin-tools`, you'll need to have a copy of `Django` already installed. For the 0.6 release, `Django` 1.7 or newer is required.

---

**Note:** *Important note to users of `django` 1.6 or below:* starting from 0.6.0, `django-admin-tools` is *NOT* compatible with `django`  $\leq$  1.6. If you want, you can still use the 0.5.2 version that will always be available on Pypi.

---

### 1.1 Installing `django-admin-tools`

`django-admin-tools` requires `Django` version 1.3 or superior, optionally, if you want to display feed modules, you'll also need the [Universal Feed Parser](#) module.

There are several ways to install `django-admin-tools`, this is explained in *the installation section*.

For the impatient, the easiest method is to install `django-admin-tools` via `easy_install` or `pip`.

Using `easy_install`, type:

```
easy_install -Z django-admin-tools
```

Note that the `-Z` flag is required, to tell `easy_install` not to create a zipped package; zipped packages prevent certain features of `Django` from working properly.

Using `pip`, type:

```
pip install django-admin-tools
```

### 1.2 Basic configuration

For a more detailed guide on how to configure `django-admin-tools`, please consult *the configuration section*.

#### 1.2.1 Prerequisite

In order to use `django-admin-tools` you obviously need to have configured your `Django` admin site. If you didn't, please refer to the [relevant `django` documentation](#).

## 1.2.2 Configuration

First make sure you have the `django.core.context_processors.request` template context processor in your `TEMPLATE_CONTEXT_PROCESSORS`.

---

**Note:** Starting from django 1.8, `TEMPLATE_CONTEXT_PROCESSORS` is deprecated, you must add the request context processor in your `TEMPLATES` variable instead, please refer to the [relevant django documentation](#).

---

Then, add `admin_tools` and its modules to the `INSTALLED_APPS` like this:

```
INSTALLED_APPS = (
    'admin_tools',
    'admin_tools.theming',
    'admin_tools.menu',
    'admin_tools.dashboard',
    'django.contrib.auth',
    'django.contrib.sites',
    'django.contrib.admin'
    # ...other installed applications...
)
```

---

**Important:** it is very important that you put the `admin_tools` modules **before** the `django.contrib.admin` module, because `django-admin-tools` overrides the default Django admin templates, and this will not work otherwise.

---

Then, just add `django-admin-tools` to your `urls.py` file:

```
urlpatterns = patterns('',
    url(r'^admin_tools/', include('admin_tools.urls')),
    #...other url patterns...
)
```

Finally simply run:

```
python manage.py migrate
```

## 1.3 Testing your new shiny admin interface

Congrats! At this point you should have a working installation of `django-admin-tools`. Now you can just login to your admin site and see what changed.

`django-admin-tools` is fully customizable, but this is out of the scope of this quickstart. To learn how to customize `django-admin-tools` modules please read [the customization section](#).



---

## Installation guide

---

### 2.1 Requirements

Before installing `django-admin-tools`, you'll need to have a copy of [Django](#) already installed. For the 0.6 release, Django 1.7 or newer is required.

---

**Note:** *Important note to users of django 1.6 or below:* starting from 0.6.0, `django-admin-tools` is *NOT* compatible with `django <= 1.6`. If you want, you can still use the 0.5.2 version that will always be available on Pypi.

---

For further information, consult the [Django download page](#), which offers convenient packaged downloads and installation instructions.

---

**Note:** If you want to display feeds in the admin dashboard, using the `FeedDashboardModule` you need to install the [Universal Feed Parser module](#).

---

### 2.2 Installing `django-admin-tools`

There are several ways to install `django-admin-tools`:

- Automatically, via a package manager.
- Manually, by downloading a copy of the release package and installing it yourself.
- Manually, by performing a Mercurial checkout of the latest code.

It is also highly recommended that you learn to use [virtualenv](#) for development and deployment of Python software; `virtualenv` provides isolated Python environments into which collections of software (e.g., a copy of Django, and the necessary settings and applications for deploying a site) can be installed, without conflicting with other installed software. This makes installation, testing, management and deployment far simpler than traditional site-wide installation of Python packages.

#### 2.2.1 Automatic installation via a package manager

Several automatic package-installation tools are available for Python; the most popular are [easy\\_install](#) and [pip](#). Either can be used to install `django-admin-tools`.

Using `easy_install`, type:

```
easy_install -Z django-admin-tools
```

Note that the `-Z` flag is required, to tell `easy_install` not to create a zipped package; zipped packages prevent certain features of Django from working properly.

Using `pip`, type:

```
pip install django-admin-tools
```

It is also possible that your operating system distributor provides a packaged version of `django-admin-tools`. Consult your operating system's package list for details, but be aware that third-party distributions may be providing older versions of `django-admin-tools`, and so you should consult the documentation which comes with your operating system's package.

### 2.2.2 Manual installation from a downloaded package

If you prefer not to use an automated package installer, you can download a copy of `django-admin-tools` and install it manually. The latest release package can be downloaded from [django-admin-tools's listing on the Python Package Index](#).

Once you've downloaded the package, unpack it (on most operating systems, simply double-click; alternately, type `tar zxvf django-admin-tools-X-Y-Z.tar.gz` at a command line on Linux, Mac OS X or other Unix-like systems). This will create the directory `django-admin-tools-X-Y-Z`, which contains the `setup.py` installation script. From a command line in that directory, type:

```
python setup.py install
```

---

**Note:** On some systems you may need to execute this with administrative privileges (e.g., `sudo python setup.py install`).

---

### 2.2.3 Manual installation from a git checkout

If you'd like to try out the latest in-development code, you can obtain it from the `django-admin-tools` repository, which is hosted on [Github](#). To obtain the latest code and documentation, you'll need to have Git installed, at which point you can type:

```
git clone https://github.com/django-admin-tools/django-admin-tools.git
```

This will create a copy of the `django-admin-tools` Git repository on your computer; you can then add the `django-admin-tools` directory to your Python import path, or use the `setup.py` script to install as a package.

---

## Configuring django-admin-tools

---

### 3.1 Basic configuration

Once installed, you can add django-admin-tools to any Django-based project you're developing.

django-admin-tools is composed of several modules:

- `admin_tools.theming`: an app that makes it easy to customize the look and feel of the admin interface;
- `admin_tools.menu`: a customizable navigation menu that sits on top of every django administration index page;
- `admin_tools.dashboard`: a customizable dashboard that replaces the django administration index page.

#### 3.1.1 Prerequisite

In order to use django-admin-tools you obviously need to have configured your django admin site, if you didn't, please refer to the [relevant django documentation](#).

#### 3.1.2 Required settings

First make sure you have the `django.core.context_processors.request` template context processor in your `TEMPLATE_CONTEXT_PROCESSORS`.

---

**Note:** Starting from django 1.8, `TEMPLATE_CONTEXT_PROCESSORS` is deprecated, you must add the `request` context processor in your `TEMPLATES` variable instead, please refer to the [relevant django documentation](#).

---

Then, add the django-admin-tools modules to the `INSTALLED_APPS` like this:

```
INSTALLED_APPS = (  
    'admin_tools.theming',  
    'admin_tools.menu',  
    'admin_tools.dashboard',  
    'django.contrib.auth',  
    'django.contrib.sites',  
    'django.contrib.admin'  
    # ...other installed applications...  
)
```

**Note:** it is very important that you put the `admin_tools` modules **before** the `django.contrib.admin` module, because `django-admin-tools` overrides the default `django` admin templates, and this will not work otherwise.

---

`django-admin-tools` is modular, so if you want to disable a particular module, just remove or comment it in your `INSTALLED_APPS`. For example, if you just want to use the dashboard:

```
INSTALLED_APPS = (
    'admin_tools.dashboard',
    'django.contrib.auth',
    'django.contrib.sites',
    'django.contrib.admin'
    # ...other installed applications...
)
```

### 3.1.3 Setting up the database

To set up the tables that `django-admin-tools` uses you'll need to type:

```
python manage.py migrate
```

### 3.1.4 Adding `django-admin-tools` to your `urls.py` file

You'll need to add `django-admin-tools` to your `urls.py` file:

```
urlpatterns = patterns('',
    url(r'^admin_tools/', include('admin_tools.urls')),
    #...other url patterns...
)
```

## 3.2 Available settings variables

**ADMIN\_TOOLS\_MENU** The path to your custom menu class, for example “`yourproject.menu.CustomMenu`”.

**ADMIN\_TOOLS\_INDEX\_DASHBOARD** The path to your custom index dashboard, for example “`yourproject.dashboard.CustomIndexDashboard`”.

**ADMIN\_TOOLS\_APP\_INDEX\_DASHBOARD** The path to your custom app index dashboard, for example “`yourproject.dashboard.CustomAppIndexDashboard`”.

**ADMIN\_TOOLS\_THEMING\_CSS** The path to your theming css stylesheet, relative to your `MEDIA_URL`, for example:

```
ADMIN_TOOLS_THEMING_CSS = 'css/theming.css'
```

---

## Customization of the django-admin-tools modules

---

### 4.1 Introduction

django-admin-tools is very easy to customize, you can override the admin menu, the index dashboard and the app index dashboard.

**For this django-admin-tools provides two management commands:**

- `custommenu`
- `customdashboard`

### 4.2 Customizing the navigation menu

To customize the admin menu, the first step is to do the following:

```
python manage.py custommenu
```

This will create a file named `menu.py` in your project directory. If for some reason you want another file name, you can do:

```
python manage.py custommenu somefile.py
```

The created file contains a class that is a copy of the default menu, it is named `CustomMenu`, you can rename it if you want but if you do so, make sure you put the correct class name in your `ADMIN_TOOLS_MENU` settings variable.

---

**Note:** You could have done the above by hand, without using the `custommenu` management command, but it's simpler with it.

---

Now you need to tell django-admin-tools to use your custom menu instead of the default one, open your `settings.py` file and add the following:

```
ADMIN_TOOLS_MENU = 'yourproject.menu.CustomMenu'
```

Obviously, you need to change “yourproject” to the real project name, if you have chosen a different file name or if you renamed the menu class, you'll also need to change the above string to reflect your modifications.

At this point the menu displayed in the admin is your custom menu, now you can read [the menu and menu items API documentation](#) to learn how to create your custom menu.

## 4.3 Customizing the dashboards

To customize the index and app index dashboards, the first step is to do the following:

```
python manage.py customdashboard
```

This will create a file named `dashboard.py` in your project directory. If for some reason you want another file name, you can do:

```
python manage.py customdashboard somefile.py
```

**The created file contains two classes:**

- The `CustomIndexDashboard` class that corresponds to the admin index page dashboard;
- The `CustomAppIndexDashboard` class that corresponds to the index page of each installed application.

You can rename these classes if you want but if you do so, make sure adjust the `ADMIN_TOOLS_INDEX_DASHBOARD` and `ADMIN_TOOLS_APP_INDEX_DASHBOARD` settings variables to match your class names.

---

**Note:** You could have done the above by hand, without using the `customdashboard` management command, but it's simpler with it.

---

Now you need to tell django-admin-tools to use your custom dashboard(s). Open your `settings.py` file and add the following:

```
ADMIN_TOOLS_INDEX_DASHBOARD = 'yourproject.dashboard.CustomIndexDashboard'  
ADMIN_TOOLS_APP_INDEX_DASHBOARD = 'yourproject.dashboard.CustomAppIndexDashboard'
```

If you only want a custom index dashboard, you would just need the first line. Obviously, you need to change “yourproject” to the real project name, if you have chosen a different file name or if you renamed the dashboard classes, you’ll also need to change the above string to reflect your modifications.

At this point the dashboards displayed in the index and the app index should be your custom dashboards, now you can read [the dashboard and dashboard modules API documentation](#) to learn how to create your custom dashboard.

## 4.4 Customizing the theme

**Warning:** The theming support is still very basic, do not rely too much on it for the moment.

This is very simple, just configure the `ADMIN_TOOLS_THEMING_CSS` to point to your custom css file, for example:

```
ADMIN_TOOLS_THEMING_CSS = 'css/theming.css'
```

A good start is to copy the `admin_tools/media/admin_tools/css/theming.css` to your custom file and to modify it to suits your needs.

---

## Working with multiple admin sites

---

### 5.1 Introduction

Django supports custom admin sites, and of course you can have as many admin sites as you want, django-admin-tools provides basic support for this, you can setup a custom dashboard or menu for each admin site.

### 5.2 Setting up a different dashboard and menu for each admin site instance

In the following example we will assume that you have two admin site instances: the default django admin site and a custom admin site of your own. In your urls, you should have something like this:

```
from django.conf.urls.defaults import *
from django.contrib import admin
from yourproject.admin import admin_site

admin.autodiscover()

urlpatterns = patterns('',
    (r'^admin/', include(admin.site.urls)),
    (r'^myadmin/', include(admin_site.urls)),
)
```

Now to configure your dashboards, you could do:

```
python manage.py customdashboard django_admin_dashboard.py
python manage.py customdashboard my_admin_dashboard.py
```

And to tell django-admin-tools to use your custom dashboards depending on the admin site being used, you just have to add the following to your project settings file:

```
ADMIN_TOOLS_INDEX_DASHBOARD = {
    'django.contrib.admin.site': 'yourproject.django_admin_dashboard.CustomIndexDashboard',
    'yourproject.admin.admin_site': 'yourproject.my_admin_dashboard.CustomIndexDashboard',
}
```

Note that the same applies for the ADMIN\_TOOLS\_APP\_INDEX\_DASHBOARD settings variable.

Finally do the same thing for menu:

```
python manage.py custommenu django_admin_menu.py
python manage.py custommenu my_admin_menu.py
```

And to tell django-admin-tools to use your custom menu depending on the admin site being used:

```
ADMIN_TOOLS_MENU = {
    'django.contrib.admin.site': 'yourproject.django_admin_menu.CustomMenu',
    'yourproject.admin.admin_site': 'yourproject.my_admin_menu.CustomMenu',
}
```



---

## The django-admin-tools menu and menu items API

---

This section describe the API of the django-admin-tools menu and menu items. Make sure you read this before creating your custom menu.

### 6.1 The Menu class

### 6.2 The MenuItem class

### 6.3 The AppList class

### 6.4 The ModelList class

### 6.5 The Bookmarks class



---

## The django-admin-tools dashboard and dashboard modules API

---

This section describe the API of the django-admin-tools dashboard and dashboard modules. Make sure you read this before creating your custom dashboard and custom modules.

**..note::** If your layout seems to be broken or you have problems with included javascript files, you should try to reset your dashboard preferences (assuming a MySQL backend, the truncate command also works in postgress):

```
python manage.py dbshell
mysql> truncate admin_tools_dashboard_preferences;
```

For more information see [this issue](#).

### 7.1 The Dashboard class

### 7.2 The AppIndexDashboard class

### 7.3 The DashboardModule class

### 7.4 The Group class

### 7.5 The LinkList class

### 7.6 The AppList class

### 7.7 The ModelList class

### 7.8 The RecentActions class

### 7.9 The Feed class



---

## Integration with third party applications

---

todo: write doc for “Integration with third party applications” section.



---

## Contributing to django-admin-tools

---

You are very welcome to contribute to the project! django-admin-tools is on [Github](#), which makes collaborating very easy.

There are various possibilities to get involved, for example you can:

- [Report bugs](#), preferably with patches if you can
- [Discuss new features ideas](#)
- Fork the project, implement those features and send a pull request
- [Enhance the documentation](#)
- [Translate django-admin-tools in your language](#)





---

## Testing of django-admin-tools

---

This is information for developers of django-admin-tools itself.

### 10.1 Running tests

First, cd the test\_proj directory:

```
$ cd test_proj
```

And to run the tests, just type:

```
$ python manage.py test
```

### 10.2 Code coverage report

Install the coverage.py library and the django-coverage app:

```
$ pip install coverage django-coverage
```

Then run tests and open test\_proj/\_coverage/index.html file in browser.

### 10.3 Where tests live

Unit tests should be put into appropriate module's tests.py. Functional/integration tests should be put somewhere into test\_proj.



**a**

`admin_tools.dashboard`, [15](#)

`admin_tools.dashboard.modules`, [15](#)



## A

[admin\\_tools.dashboard \(module\)](#), [15](#)

[admin\\_tools.dashboard.modules \(module\)](#), [15](#)