

---

# **django-fluent-dashboard** **Documentation**

*Release 1.0a1*

**Diederik van der Boor**

June 16, 2016



<b>1</b>	<b>Installation</b>	<b>3</b>
1.1	Django configuration . . . . .	3
<b>2</b>	<b>Configuration</b>	<b>5</b>
2.1	Icon settings . . . . .	6
2.2	Application grouping . . . . .	7
2.3	CMS integration . . . . .	7
<b>3</b>	<b>Optional dependencies</b>	<b>9</b>
3.1	Cache status . . . . .	9
3.2	RSS feeds . . . . .	9
<b>4</b>	<b>Advanced customization</b>	<b>11</b>
4.1	Changing the dashboard layout . . . . .	11
4.2	Changing the menu layout . . . . .	11
4.3	Available classes . . . . .	11
<b>5</b>	<b>Other related applications</b>	<b>13</b>
5.1	django-admin-tools . . . . .	13
5.2	django-admin-user-stats . . . . .	13
5.3	django-admin-tools-stats . . . . .	13
5.4	colloayproject/dashboardmods . . . . .	13
<b>6</b>	<b>API documentation</b>	<b>15</b>
6.1	fluent_dashboard.dashboard . . . . .	15
6.2	fluent_dashboard.modules . . . . .	15
6.3	fluent_dashboard.menu . . . . .	15
6.4	fluent_dashboard.items . . . . .	15
6.5	fluent_dashboard.appgroups . . . . .	15
<b>7</b>	<b>Changelog</b>	<b>17</b>
7.1	Changes in 1.0a1 (2016-05-08) . . . . .	17
7.2	Changes in version 0.6.1 (2016-01-21) . . . . .	17
7.3	Changes in version 0.6 (2015-12-30) . . . . .	17
7.4	Changes in version 0.5.2 (2015-09-02) . . . . .	17
7.5	Changes in version 0.5.2 (2015-09-01) . . . . .	17
7.6	Changes in version 0.5.1 . . . . .	17
7.7	Changes in version 0.5.0 . . . . .	18
7.8	Changes in version 0.4.0 . . . . .	18

7.9	Changes in version 0.3.6	18
7.10	Changes in version 0.3.5	18
7.11	Changes in version 0.3.4	18
7.12	Changes in version 0.3.3	18
7.13	Changes in version 0.3.2	19
7.14	Changes in version 0.3.1	19
7.15	Changes in version 0.3.0	19
7.16	Changes in version 0.2.0	19
7.17	Version 0.1.0	20
<b>8</b>	<b>Preview</b>	<b>21</b>
<b>9</b>	<b>Icon credits</b>	<b>23</b>
9.1	Other icon sets	23
<b>10</b>	<b>Indices and tables</b>	<b>25</b>
	<b>Python Module Index</b>	<b>27</b>

The `fluent_dashboard` module offers a custom admin dashboard, built on top of `django-admin-tools` ([code](#)).

The `django-admin-tools` package provides a default mechanism to replace the standard Django admin homepage with a widget based dashboard. The `fluent_dashboard` module extends this, by providing additional widgets (called “modules”) such as:

- a “icon list” module for the admin homepage.
- a “welcome” module for the admin homepage.
- a configurable module layout for the admin homepage, through `settings.py`.
- a “return to site” link.

Contents:



---

## Installation

---

This package can be installed using pip:

```
pip install django-fluent-dashboard
```

This should automatically install `django-admin-tools` 0.4.1 as well. The 0.4.1 release is required to use Django 1.3.

### 1.1 Django configuration

To enable the dashboard in Django, both the `fluent_dashboard` and the `admin_tools` modules have to be added to `settings.py`:

```
INSTALLED_APPS += (  
    'fluent_dashboard',  
  
    'admin_tools',      # for staticfiles in Django 1.3  
    'admin_tools.theming',  
    'admin_tools.menu',  
    'admin_tools.dashboard',  
    'django.contrib.admin',  
)
```

---

**Note:** The `admin_tools.theming` and `admin_tools.menu` applications are optional.

---

Next, the `django-admin-tools` can be configured to use the `fluent_dashboard` instead, by using:

```
ADMIN_TOOLS_INDEX_DASHBOARD = 'fluent_dashboard.dashboard.FluentIndexDashboard'  
ADMIN_TOOLS_APP_INDEX_DASHBOARD = 'fluent_dashboard.dashboard.FluentAppIndexDashboard'  
ADMIN_TOOLS_MENU = 'fluent_dashboard.menu.FluentMenu'
```

That's all! The Django admin dashboard should open now with the dashboard, configured with sane defaults.

Next:

- The application groups and icons can be customized with additional [configuration](#).
- The Memcache and Varnish statistics can also be displayed by configuring some [optional dependencies](#).





---

## Configuration

---

A quick overview of the available settings:

```
from django.utils.translation import ugettext_lazy as _

# Icon settings

FLUENT_DASHBOARD_ICON_THEME = 'flaticons'

FLUENT_DASHBOARD_APP_ICONS = {
    'cms/page': 'internet-web-browser.png',
    'auth/user': 'system-users.png',
    'auth/group': 'resource-group.png',
    'sites/site': 'applications-internet.png',
    'google_analytics/analytics': 'view-statistics.png',
    'registration/registrationprofile': 'list-add-user.png'
    # ...
}

FLUENT_DASHBOARD_DEFAULT_ICON = 'unknown.png'

FLUENT_DASHBOARD_DEFAULT_MODULE = 'admin_tools.dashboard.modules.AppList'

# Application grouping:

FLUENT_DASHBOARD_APP_GROUPS = (
    (_('CMS'), {
        'models': (
            '*cms.*',
            'fiber.*',
        ),
        'module': 'CmsAppIconList',
        'collapsible': False,
    }),
    (_('Interactivity'), {
        'models': (
            'django.contrib.comments.*',
            'form_designer.*',
            'threadedcomments.*',
            'zinnia.*',
        ),
    }),
    (_('Administration'), {
```

```

        'models': (
            'django.contrib.auth.*',
            'django.contrib.sites.*',
            'google_analytics.*',
            'registration.*',
        ),
    }),
    (_('Applications'), {
        'models': ('*',),
        'module': FLUENT_DASHBOARD_DEFAULT_MODULE,
        'collapsible': True,
    }),
)

# CMS integration:

FLUENT_DASHBOARD_CMS_PAGE_MODEL = ('cms', 'page')

FLUENT_DASHBOARD_CMS_APP_NAMES = (
    '*cms*', # wildcard match; DjangoCMS, FeinCMS
    'fiber', # Django-Fiber
)

FLUENT_DASHBOARD_CMS_MODEL_ORDER = {
    'page': 1,
    'object': 2,
    'layout': 3,
    'content': 4,
    'file': 5,
    'site': 99
}

```

The icon names/paths are parsed in the following way:

- When the icon is an absolute URL, it is used as-is.
- When the icon contains a slash, it is relative from the `STATIC_URL`.
- When the icon has no slashes, it's relative to the theme url folder (typically `STATIC_URL/ecms_dashboard/themename/`),

## 2.1 Icon settings

### 2.1.1 FLUENT\_DASHBOARD\_ICON\_THEME

The icon theme defines which folder is used to find the icons. This allows to easily switch between icon sets without having to update all settings.

The current theme is “flaticons”, which are available from <http://www.flaticon.com/>. You can use these icons if you give attribution for it.

The previous theme was “Oxygen”, which is freely available from KDE. You may use the icons under the [LGPL 3 license](#).

## 2.1.2 FLUENT\_DASHBOARD\_APP\_ICONS

A dictionary of the *app/model*, and the associated icon. For a few commonly know applications, icons are already provided. Any key defined in `settings.py` overrides the default.

## 2.1.3 FLUENT\_DASHBOARD\_DEFAULT\_ICON

In case a suitable icon is not found, this icon is used.

# 2.2 Application grouping

## 2.2.1 FLUENT\_DASHBOARD\_APP\_GROUPS

The application groups to display at the dashboard. Each tuple has a title, and dictionary which can have the following fields:

- **models:** which models should be included. Simple pattern based filtering is provided by `fnmatch()`.
- **collapsible:** whether the group can be collapsed to a single line. Default is `False` for all elements to reduce clutter.
- **module:** which dashboard module can be used. Possible values are:
  - `AppList` (the default from `django-admin-tools`).
  - `ModelList` (the alternative from `django-admin-tools`).
  - `AppIconList`
  - `CmsAppIconList`
  - any other class, specified as `full module.ClassName` syntax.

By default, there is a section for “CMS”, “Interactivity” and “Administration” filled with known Django applications.

The `*` selector without any application name, is special: it matches all applications which are not placed in any other groups.

## 2.2.2 FLUENT\_DASHBOARD\_DEFAULT\_MODULE

The application module used to group the remaining applications. Any of the valued for the **module** field of the `FLUENT_DASHBOARD_APP_GROUPS` setting can be used.

# 2.3 CMS integration

## 2.3.1 FLUENT\_DASHBOARD\_CMS\_PAGE\_MODEL

The model used to display a link to the CMS pages. The value is a tuple of *application name*, and *model name*. This is used in the welcome text of the `PersonalModule`. For some known CMS applications, this value is already set to a sane default.

### 2.3.2 FLUENT\_DASHBOARD\_CMS\_APP\_NAMES

A list of patterns to define which applications should be considered as “CMS” applications. These applications are sorted on top in the `CmsAppIconList` and `CmsModelList` classes. The default `FLUENT_DASHBOARD_APP_GROUPS` also uses this setting to fill the “CMS” category.

### 2.3.3 FLUENT\_DASHBOARD\_CMS\_MODEL\_ORDER

A dictionary of *modelname: ordering* items, to sort the models of CMS applications in a custom order. This can be used for example, to display the pages model first, and the files/images models next.

---

## Optional dependencies

---

The installation of *django-fluent-dasbboard* can be extended with a few modules.

### 3.1 Cache status

When the *collowayproject/dashboardmods* package is installed and configured, the dashboard uses it to display Memcache and Varnish statistics for superusers.

First install the package:

```
pip install dashboardmods
```

An example configurion looks like:

```
INSTALLED_APPS += (
    'dashboardmods',
)

# Example Memcache configuration:
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        'KEY_PREFIX': 'mysite.production',
        'LOCATION': '127.0.0.1:11211',
        'TIMEOUT': 24*3600,
    },
}

# Optional, example Varnish configuration:
VARNISH_MANAGEMENT_ADDRS = ('127.0.0.1:6082',)
```

When a cache is not configured, it will simply not be displayed by the `CacheStatusGroup` module.

### 3.2 RSS feeds

The RSS feeds configured by the *collowayproject/dashboardmods* package will also be displayed at the dashboard. This requires the *feedparser* module. Install it using:

```
pip install feedparser
```



---

## Advanced customization

---

For advanced dashboard or menu layouts beyond the normal [settings](#), the classes provided by this package (and additionally `django-admin-tools`) can be overwritten.

### 4.1 Changing the dashboard layout

To change the widget layout, extend the `FluentIndexDashboard` class and create the new module layout in the `__init__()` or `init_with_context()` function.

The custom dashboard class can be loaded by referencing them in either one of these settings:

- `ADMIN_TOOLS_INDEX_DASHBOARD`
- `ADMIN_TOOLS_APP_INDEX_DASHBOARD`

Any existing classes from the `fluent_dashboard.modules` package could be reused off course.

#### See also:

When customizing the dashboard module layout, dont forget to look at the `django-admin-tools` package and *other applications* for additional modules to use. These packages have modules for RSS feeds, Varnish/Memcache status, and even tabbing/grouping items.

### 4.2 Changing the menu layout

The menu layout can be changed by extending the `FluentMenu` class, and overwriting the `init_with_context()` function.

The custom menu class can be loaded by referencing it in the setting:

- `ADMIN_TOOLS_MENU`

### 4.3 Available classes

The `fluent_dashboard` app provides the following classes, which are suitable for overwriting them:

`fluent_dashboard.dashboard`: the custom dashboard classes:

- `FluentIndexDashboard`: the dashboard for the homepage.
- `FluentAppIndexDashboard`: the dashboard for the application index page.

`fluent_dashboard.items`: menu icons

- `CmsModelList`: a model list, with strong bias of sorting CMS applications on top.
- `ReturnToSiteItem`: a custom `MenuItem` class, with the “Return to site” link.

`fluent_dashboard.menu`: the menu classes.

- `FluentMenu`: a custom `Menu` implementation, which honors the `FLUENT_DASHBOARD_APP_GROUPS` setting, and adds the `ReturnToSiteItem`.

`fluent_dashboard.modules`: custom widgets (called “modules”) to display at the dashboard.

- `AppIconList`: an `AppList` implementation that displays the models as icons.
- `CmsAppIconList`: an `AppIconList` variation with a strong bias towards sorting CMS applications on top.
- `PersonalModule`: a personal welcome text.
- `CacheStatusGroup`: the statistics of Memcache and Varnish.



---

## Other related applications

---

The following packages provide additional modules, which can be displayed at the dashboard:

### 5.1 django-admin-tools

Invaluable to forget, the *django-admin-tools* package also provides modules for:

- A bookmarks menu
- RSS feed module
- Tab grouping module
- Link list module

Website: <https://github.com/django-admin-tools/django-admin-tools>

### 5.2 django-admin-user-stats

The *django-admin-user-stats* package adds graphs to the dashboard, to see the number of registered users in the last month.

Website: <https://github.com/kmike/django-admin-user-stats>

### 5.3 django-admin-tools-stats

Derived from *django-admin-user-stats*, this package adds configurable graphs for any model type. Different model data can be hooked in, for example, new user registrations, number of votes last month, etc..

- Website: <https://github.com/Star2Billing/django-admin-tools-stats>

### 5.4 collwayproject/dashboardmods

Initiated by the The Washington Times, this package adds status modules to the admin dashboard for:

- Varnish
- Memcached

When the `dashboardmods` package is installed, and added to the `INSTALLED_APPS`, the `FluentIndexDashboard` and `CacheStatusGroup` classed will automatically pick it up to display the cache status.

Website: <https://github.com/callowayproject/dashboardmods>

---

## API documentation

---

Contents:

### 6.1 fluent\_dashboard.dashboard

#### 6.1.1 The `FluentIndexDashboard` class

#### 6.1.2 The `FluentAppIndexDashboard` class

### 6.2 fluent\_dashboard.modules

#### 6.2.1 The `AppIconList` class

#### 6.2.2 The `CmsAppIconList` class

#### 6.2.3 The `PersonalModule` class

#### 6.2.4 The `CacheStatusGroup` class

### 6.3 fluent\_dashboard.menu

#### 6.3.1 The `FluentMenu` class

### 6.4 fluent\_dashboard.items

#### 6.4.1 The `CmsModelList` class

#### 6.4.2 The `ReturnToSiteItem` class

### 6.5 fluent\_dashboard.appgroups

Splitting and organizing applications and models into groups. This module is mostly meant for internal use.

`fluent_dashboard.appgroups.get_application_groups()`

Return the applications of the system, organized in various groups.

These groups are not connected with the application names, but rather with a pattern of applications.

`fluent_dashboard.appgroups.get_class(import_path)`

Import a class by name.

`fluent_dashboard.appgroups.get cms_model_order(model_name)`

Return a numeric ordering for a model name.

`fluent_dashboard.appgroups.is cms_app(app_name)`

Return whether the given application is a CMS app

`fluent_dashboard.appgroups.sort cms_models(cms_models)`

Sort a set of CMS-related models in a custom (predefined) order.

---

## Changelog

---

### 7.1 Changes in 1.0a1 (2016-05-08)

- Use new icons, from [www.flaticon.com](http://www.flaticon.com)
- **BACKWARDS INCOMPATIBLE:** Icon defaults changed, please review your `FLUENT_DASHBOARD_...` settings. You can add `FLUENT_DASHBOARD_ICON_THEME = 'oxygen'` to keep using the old settings.

### 7.2 Changes in version 0.6.1 (2016-01-21)

- Fix Python error in `ReturnToSiteItem` when custom admin urls have to url name set.

### 7.3 Changes in version 0.6 (2015-12-30)

- Added Django 1.9 support
- Dropped Django 1.4 support

### 7.4 Changes in version 0.5.2 (2015-09-02)

- Fixed CSS media inclusion for `django-admin-tools 0.5.x` Turns out both version need a different layout.

### 7.5 Changes in version 0.5.2 (2015-09-01)

- Fixed CSS media inclusion for `django-admin-tools 0.6`

### 7.6 Changes in version 0.5.1

- Fixed Python 3 issue
- Added icons for `fluentcms-googlemaps`

## 7.7 Changes in version 0.5.0

- Added Django 1.8 compatibility
- Fixed Python 3 issue

## 7.8 Changes in version 0.4.0

- Allow passing all `DashboardModule` kwargs to `FLUENT_DASHBOARD_APP_GROUPS`.
- Added new Oxygen icons
- Fix assumption that `varnish` is installed because `dashboardmods` is.
- Fix showing disabled application groups in the menu
- Fix 500 error when PK is not an int.
- Fix missing icons for `django-fluent-faq`.
- Fix missing icons for `django-fluent-comments`.

## 7.9 Changes in version 0.3.6

- Fix error when primary keys are not an integer.
- Added more Oxygen icons.

## 7.10 Changes in version 0.3.5

- Added filebrowser icon.
- Fix custom user model support of Django 1.5.
- Fix requirements of `extras_require` `cachestatus`
- Hide cache status group by default.

## 7.11 Changes in version 0.3.4

- Fixed a packaging error, `dashboard.css` was missing in the dist.

## 7.12 Changes in version 0.3.3

- Added more Oxygen icons.
- Added icon for `sharedcontent` plugin of `django-fluent-contents`.
- Fixed `KeyError` when a model has add support, but no edit support.
- Fixed icon layout when a model has no permissions to add/edit.

- Fixed welcome text in personal module, remove pages link if the user has no permission to edit pages.
- Bump required version of `django-admin-tools` to 0.5.1, which has Django 1.4/1.5 support.

## 7.13 Changes in version 0.3.2

- Fix dashboard icons when admin is mounted in a subpath (e.g. `/en/admin/`)
- Added `fluent_dashboard/modules/personal_text.html` template to change the personal text easily.
- Added `*.UserProfile` to the Administration group.
- Added various Oxygen icons to assist other projects.

## 7.14 Changes in version 0.3.1

- Added `FLUENT_DASHBOARD_DEFAULT_MODULE` setting, to switch between `AppList`, `ModelList`, etc..
- Improved support for the `admin_tools.dashboard.modules.ModelList` module.

## 7.15 Changes in version 0.3.0

- Added `dashboardmods` integration, automatically detected.
- Added icons for `django-fluent-pages`, `django-media-tree` and `django-fluent-blogs`.
- Improved README setup
- Fixed requirements for `readthedocs`
- Fixed installation problems on Windows

## 7.16 Changes in version 0.2.0

First public release

- Renamed app to `fluent_dashboard`.
- Added icons for `google_analytics`, Django CMS, FeinCMS, Zinnia, comments, tagging
- Added icon theme switching
- Added documentation
- Added setup files
- Added settings:
  - `FLUENT_DASHBOARD_CMS_PAGE_MODEL`
  - `FLUENT_DASHBOARD_CMS_APP_NAMES`
  - `FLUENT_DASHBOARD_CMS_MODEL_ORDER`
- Improved frontend detection in `ReturnToSiteItem`

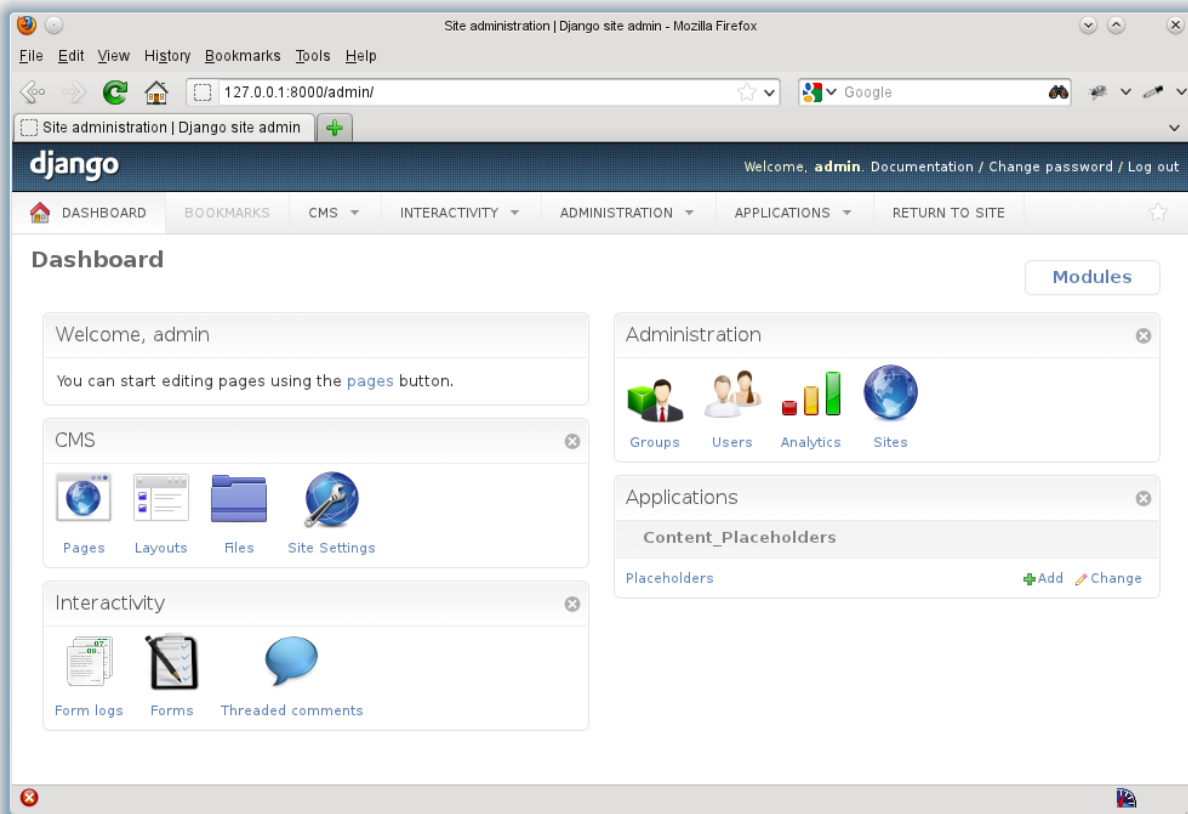
- Changed icon paths to be relative from the `STATIC_URL`.
- Changed `FLUENT_DASHBOARD_APP_GROUPS` items to dictionary layout
- Fixed model name detection when using a subdirectory.
- Fixed sorting in menu.

## **7.17 Version 0.1.0**

Initial internal release



Preview





---

## Icon credits

---

The dashboard uses flat icons, which are available from <http://www.flaticon.com/> and require attribution.

- Icons designed by [Freepik](#) from [www.flaticon.com](http://www.flaticon.com/)
- Icons designed by [SimpleIcon](#) from [www.flaticon.com](http://www.flaticon.com/)

### 9.1 Other icon sets

Previously, the dashboard uses the “Oxygen” icon theme, which is freely available from KDE. You may use the icons under the [LGPL 3](#) license. To use a different icon theme (e.g. [Tango](#)), see the `FLUENT_DASHBOARD_ICON_THEME` setting in the *Configuration* section.

For more information about the Oxygen icon theme, see:

- <http://www.oxygen-icons.org/>
- <http://techbase.kde.org/Projects/Oxygen/Licensing>

The Oxygen icon theme can be downloaded from: <http://download.kde.org/stable/4.10.0/src/oxygen-icons-4.10.0.tar.xz>.



---

**Indices and tables**

---

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



**f**

`fluent_dashboard.appgroups`, 15





## F

fluent\_dashboard.appgroups (module), 15

## G

get\_application\_groups() (in module fluent\_dashboard.appgroups), 15

get\_class() (in module fluent\_dashboard.appgroups), 16

get cms\_model\_order() (in module fluent\_dashboard.appgroups), 16

## I

is cms\_app() (in module fluent\_dashboard.appgroups), 16

## S

sort cms\_models() (in module fluent\_dashboard.appgroups), 16