django-fluent-dashboard Documentation

Release 1.0.1

Diederik van der Boor

Feb 10, 2022

Contents

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:

CHAPTER 1

Installation

This package can be installed using pip:

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

This should automatically install django-admin-tools as well.

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',
    'admin_tools.theming',
    'admin_tools.menu',
    'admin_tools.dashboard',
    'django.contrib.admin',
)

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': (),
        'OPTIONS': {
            'loaders': (
                ...
                'admin_tools.template_loaders.Loader',  # Add this line!
            ),
        }
```

(continues on next page)

(continued from previous page)

```
    }
]
```

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*.

CHAPTER 2

Configuration

A quick overvivew 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,
    }),
```

(continues on next page)

(continued from previous page)

```
    (_('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.

CHAPTER 3

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 configurtion 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
```

CHAPTER 4

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 bios towards sorting CMS applications on top.
- *PersonalModule*: a personal welcome text.
- *CacheStatusGroup*: the statistics of Memcache and Varnish.

CHAPTER 5

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 collowayproject/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

CHAPTER 6

API documentation

Contents:

6.1 fluent_dashboard.dashboard

Custom dashboards for Django applications.

This package defines the following classes:

- *FluentIndexDashboard*
- *FluentAppIndexDashboard*

These classes need to be linked in `settings.py` to be loaded by *django-admin-tools*. Off course, you can also extend the classes, and use those names in the settings instead.

6.1.1 The `FluentIndexDashboard` class

class fluent_dashboard.dashboard.**FluentIndexDashboard**(***kwargs*)

A custom home screen for the Django admin interface.

It displays the application groups based on with *FLUENT_DASHBOARD_APP_GROUPS* setting. To activate the dashboard add the following to your settings.py:

```
ADMIN_TOOLS_INDEX_DASHBOARD = 'fluent_dashboard.dashboard.FluentIndexDashboard'
```

The dashboard modules are instantiated by the following functions, which can be overwritten:

- *get_personal_module()*
- *get_application_modules()*
- *get_recent_actions_module()*
- *get_rss_modules()*

- *get_cache_status_modules()*

To have a menu which is consistent with the application groups displayed by this module, use the *FluentMenu* class to render the *admin_tools* menu.

When overwriting this class, the elements can either be added in the *__init__()* method, or the *init_with_context()* method. For more information, see the *django-admin-tools* documentation.

__init__(***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

get_application_modules()

Instantiate all application modules (i.e. AppList, *AppIconList* and *CmsAppIconList*) for use in the dashboard.

get_cache_status_modules(*request*)
Instantiate the *CacheStatusGroup* module for use in the dashboard.

This module displays the status of Varnish and Memcache, if the *collowayproject/dashboardmods* package is installed and the caches are configured. By default, these modules are only shown for the superuser.

get_personal_module()
Instantiate the *PersonalModule* for use in the dashboard.

get_recent_actions_module()
Instantiate the RecentActions module for use in the dashboard.

get_rss_modules()
Instantiate the RSS modules for use in the dashboard. This module displays the RSS feeds of the *collowayproject/dashboardmods* package, if it is installed, and configured.

init_with_context(*context*)
Sometimes you may need to access context or request variables to build your dashboard, this is what the `init_with_context()` method is for. This method is called just before the display with a `django.template.RequestContext` as unique argument, so you can access to all context variables and to the `django.http.HttpRequest`.

6.1.2 The `FluentAppIndexDashboard` class

class fluent_dashboard.dashboard.**FluentAppIndexDashboard**(*app_title*, *models*, ***kwargs*)

A custom application index page for the Django admin interface.

This dashboard is displayed when one specific application is opened via the breadcrumb. It displays the models and recent actions of the specific application. To activate the dashboards add the following to your settings.py:

```
ADMIN_TOOLS_APP_INDEX_DASHBOARD = 'fluent_dashboard.dashboard.
↪FluentAppIndexDashboard'
```

__init__(*app_title*, *models*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

get_model_list_module()
Instantiate a standard ModelList class to display the models of this application.

get_recent_actions_module()
Instantiate the RecentActions module for use in the appliation index page.

6.2 fluent_dashboard.modules

Custom modules for the admin dashboard.

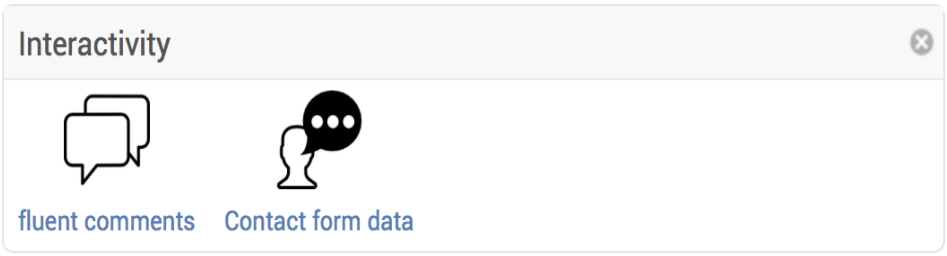
This package adds the following classes:

- *AppIconList*
- *CmsAppIconList*
- *PersonalModule*
- *CacheStatusGroup*

6.2.1 The AppIconList class

class fluent_dashboard.modules.**AppIconList**(*title=None*, ***kwargs*)

The list of applications, icon style.



It uses the FLUENT_DASHBOARD_APP_ICONS setting to find application icons.

get_icon_for_model(*app_name*, *model_name*, *default=None*)

Return the icon for the given model. It reads the *FLUENT_DASHBOARD_APP_ICONS* setting.

get_icon_url(*icon*)

Replaces the "icon name" with a full usable URL.

- 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.
- Otherwise, it's relative to the theme url folder.

init_with_context(*context*)

Initializes the icon list.

icon_static_root = '/static/'

The current static root (considered read only)

icon_theme_root = '/static/fluent_dashboard/flaticons/'

The current theme folder (considerd read only)

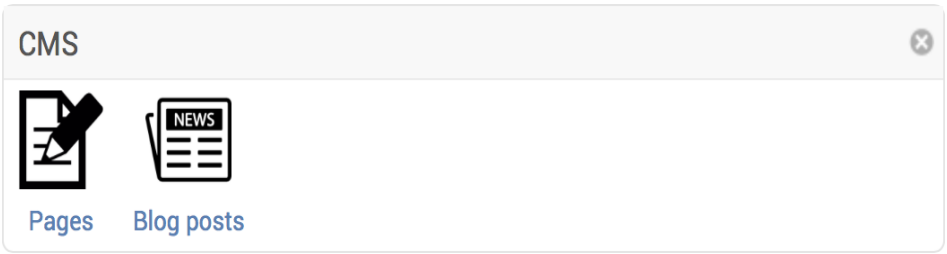
template = 'fluent_dashboard/modules/app_icon_list.html'

Specify the template to render

6.2.2 The CmsAppIconList class

class fluent_dashboard.modules.**CmsAppIconList**(*title=None*, ***kwargs*)

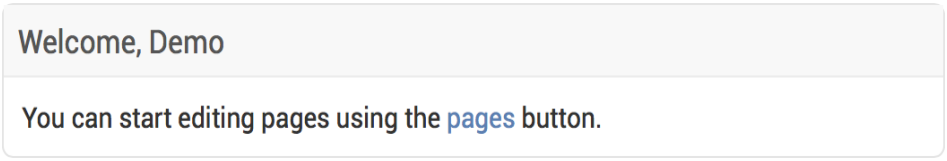
A variation of the *AppIconList* class with a strong bias towards sorting CMS apps on top.



init_with_context(*context*)
Initializes the icon list.

6.2.3 The `PersonalModule` class

class fluent_dashboard.modules.**PersonalModule**(*title=None*, ***kwargs*)
A simple module to display a welcome message.



It renders the template `fluent_dashboard/modules/personal.html`, unless the `template` variable is overwritten. The module overrides `LinkList`, allowing links to be added to the element. The *FLUENT_DASHBOARD_CMS_PAGE_MODEL* setting is used to display a link to the pages module. If this setting is not defined, a general text will be displayed instead.

init_with_context(*context*)
Initializes the link list.

is_empty()
Return True if the module has no content and False otherwise.

```
>>> mod = DashboardModule()
>>> mod.is_empty()
True
>>> mod.pre_content = 'foo'
>>> mod.is_empty()
False
>>> mod.pre_content = None
>>> mod.is_empty()
True
>>> mod.children.append('foo')
>>> mod.is_empty()
False
>>> mod.children = []
>>> mod.is_empty()
True
```

cms_page_model = None
The model to use for the CMS pages link.

template = 'fluent_dashboard/modules/personal.html'
Define the template to render

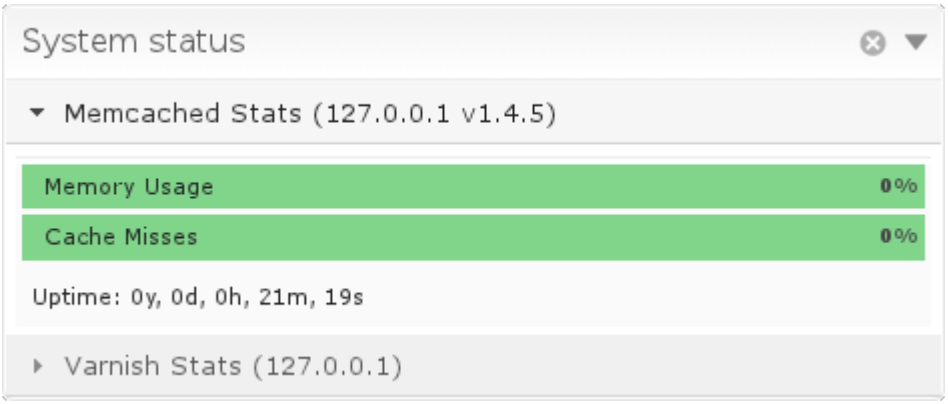
title = 'Welcome,'
Define the title to display

6.2.4 The `CacheStatusGroup` class

class `fluent_dashboard.modules.`**`CacheStatusGroup`**(*title=None*, ***kwargs*)

Display status modules for Varnish en Memcache, in a `Group` module.

This module is only displayed when the *collowayproject/dashboardmods* package is installed, added to the `INSTALLED_APPS`, and the caches are configured and reachable. For more information, see the *optional dependencies* page.



init_with_context(*context*)

Initializes the status list.

display = 'stacked'

The default display mode, can be "tabs", "stacked" or "accordion"

enabled = False

Hide by default

title = 'System status'

The default title

6.3 fluent_dashboard.menu

Custom menu for fluent_dashboard apps.

6.3.1 The `FluentMenu` class

class `fluent_dashboard.menu.`**`FluentMenu`**(***kwargs*)

Custom Menu for admin site.

The top level menu items created by this menu reflect the application groups defined in *FLUENT_DASHBOARD_APP_GROUPS*. By using both the *`FluentIndexDashboard`* and this class, the menu and dashboard modules at the admin index page will consistent. The *`ReturnToSiteItem`* is also added at the end of the menu.

To activate the menu add the following to your settings.py:

```
ADMIN_TOOLS_MENU = 'fluent_dashboard.menu.FluentMenu'
```

init_with_context(*context*)

Initialize the menu items.

6.4 fluent_dashboard.items

Additional menu items.

6.4.1 The CmsModelList class

class fluent_dashboard.items.**CmsModelList**(*title=None*, *models=None*, *exclude=None*, ***kwargs*)

A custom ModelList that displays menu items for each model. It has a strong bias towards sorting CMS apps on top.

init_with_context(*context*)

Initialize the menu.

is_item_visible(*model*, *perms*)

Return whether the model should be displayed in the menu. By default it checks for the `perms['change']` value; only items with change permission will be displayed. This function can be extended to support "view permissions" for example.

Parameters

- **model** – The model class
- **perms** – The permissions from `ModelAdmin.get_model_perms()`.

6.4.2 The ReturnToSiteItem class

class fluent_dashboard.items.**ReturnToSiteItem**(*title=None*, *url=None*, ***kwargs*)

A "Return to site" button for the menu. It redirects the user back to the frontend pages.

By default, it attempts to find the current frontend URL that corresponds with the model that's being edited in the admin 'change' page. If this is not possible, the default URL (`/`) will be used instead.

The menu item has a custom `returntosite` CSS class to be distinguishable between the other menu items.

get_edited_object(*request*)

Return the object which is currently being edited. Returns `None` if the match could not be made.

get_object_by_natural_key(*app_label*, *model_name*, *object_id*)

Return a model based on a natural key. This is a utility function for *get_edited_object()*.

init_with_context(*context*)

Find the current URL based on the context. It uses *get_edited_object()* to find the model, and calls `get_absolute_url()` to get the frontend URL.

title = 'Return to site'

Set the default title

url = '/'

Set the default URL

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.

CHAPTER 7

Changelog

7.1 Changes in 2.0 (2021-11-17)

- Added Django 4.0 support.
- Dropped Django 1.11 support.
- Dropped Python 2.7 - 3.5 support.

7.2 Changes in 1.0.1 (2019-07-15)

- Fixed crash in `ReturnToSiteItem` when visiting the admin password change form.
- Fixed early gettext calls on module loading.
- Fixed wheel package.
- Fixed setup classifiers.
- Fixed building docs.
- Bump django-admin-tools to proper minimal version.
- Reformat code with isort and black.

7.3 Changes in 1.0 (2018-01-22)

- Added Django 2.0 support.
- Removed Django 1.5 / 1.6 compatibility.
- Removed Python 2.6 support.
- Use new icons from www.flaticon.com

Backwards incompatible: The icon defaults have changed to use flat icons. Please review and update your `FLUENT_DASHBOARD_APP_ICONS` settings. To keep using the old icon theme, add `FLUENT_DASHBOARD_ICON_THEME = 'oxygen'` to your settings.

7.3.1 Released as 1.0a1 (2016-05-08)

- Use new icons from www.flaticon.com

7.4 Changes in version 0.6.1 (2016-01-21)

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

7.5 Changes in version 0.6 (2015-12-30)

- Added Django 1.9 support
- Dropped Django 1.4 support

7.6 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.

7.7 Changes in version 0.5.2 (2015-09-01)

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

7.8 Changes in version 0.5.1

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

7.9 Changes in version 0.5.0

- Added Django 1.8 compatibility
- Fixed Python 3 issue

7.10 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.

7.11 Changes in version 0.3.6

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

7.12 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.

7.13 Changes in version 0.3.4

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

7.14 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.

7.15 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.

7.16 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.

7.17 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

7.18 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.

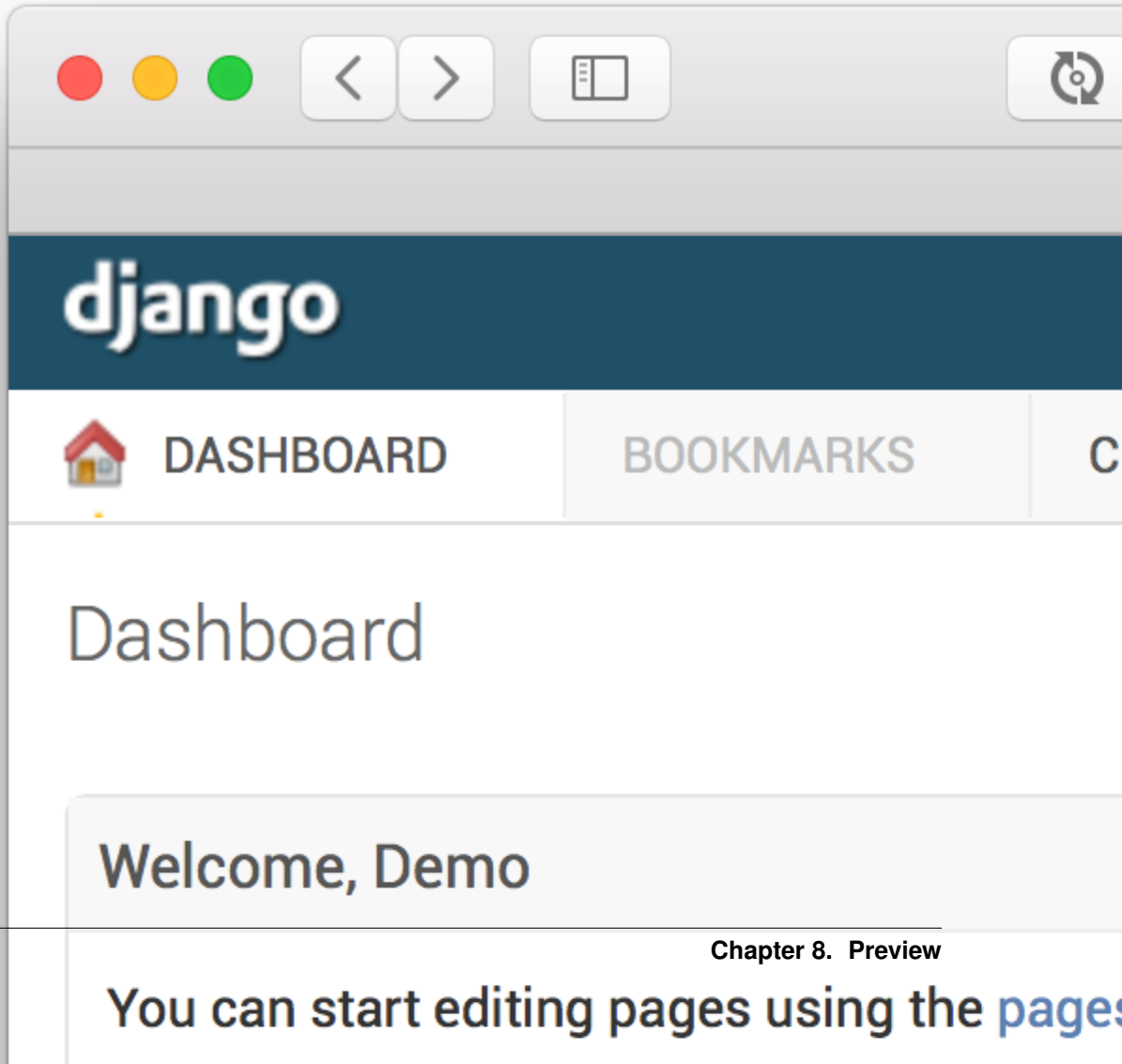
- Fixed sorting in menu.

7.19 Version 0.1.0

Initial internal release

CHAPTER 8

Preview



CHAPTER 9

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
- Icons designed by SimpleIcon from 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.

CHAPTER 10

Indices and tables

- genindex
- modindex
- search

Python Module Index

f

Index

Symbols

A

C

D

E

F

G

I

P

R

S

T

U