Django WP Admin Documentation

Release 1.4.0

Maciej 'barszcz' Marczewski

May 08, 2015

Contents

CHAPTER 1

WordPress look and feel for Django administration panel.

1.1 Features

- WordPress look and feel
- New styles for selector, calendar and timepicker widgets
- More responsive (admin panel should look fine and be usable on displays with minimum 360px width)
- Editable top menu
- Optional fully configurable left menu
- Left menu can be pinned (fixed CSS position) or unpinned and collapsed or expanded
- Awesome Font Awesome icons supported in both menus
- Multiple AdminSite's support with possibility to have different menus, colors and titles for each one
- 9 additional color themes included
- Collapsible fieldsets can be opened by default

1.2 Demo

Try `test_project` here or download django-wpadmin from GitHub and run it on your own machine. `test_project` contains SQLite database file with prepopulated sample data.

1.3 Installation

- Install django-wpadmin from PyPI:

```
pip install django-wpadmin
```

- Or from GitHub:

```
pip install git+https://github.com/barszczmm/django-wpadmin.git#egg=django-wpadmin
```

1.4 Configuration

- Add `wpadmin` to your `INSTALLED_APPS` before `django.contrib.admin`:

```
INSTALLED_APPS = (
    # Django WP Admin must be before django.contrib.admin
    'wpadmin',
)
```

- Add django.core.context_processors.request to TEMPLATE_CONTEXT_PROCESSORS setting.

1.5 Advanced topics

1.5.1 Advanced configuration

Available options

There is only one optional setting for Django WP Admin that can be added to `settings.py` file:

```
WPADMIN = {
    'admin': {
        'admin_site': 'test_project.admin.admin',
        'title': 'Django admin panel',
        'menu': {
            'top': 'wpadmin.menu.menus.BasicTopMenu',
            'left': 'wpadmin.menu.menus.BasicLeftMenu',
        },
        'dashboard': {
            'breadcrumbs': True,
        },
        'custom_style': STATIC_URL + 'wpadmin/css/themes/sunrise.css',
    }
}
```

As you can see this setting is a dictionary and it contains settings for each admin site you want to configure (usually you will have only one admin site under `/admin/` path, so `WPADMIN` dict will only have settings under `admin` key).

Lets explain it a little:

admin This is a key in dict and it **must** be equal to the URL **path** where you have your admin site. So if your admin site is accessible on `http://mydomain.com/mysuperduperadmin/` then all settings for this admin site must be in `WPADMIN['mysuperduperadmin']`.

OK so what settings are available:

admin_site Path to admin site instance. So for example if you created `django.contrib.admin.sites.AdminSite` instance in `admin.py` file in your project's directory (so you have something like `admin = AdminSite(name='admin')` in that file) then you should put `yourproject.admin.admin` here.

title Title of admin site. It will be used in `title` meta tag on site.

menu Dictionary which contains paths to classes for top and left menu to show on admin site. Read more about those classes in next section.

dashboard Dictionary containing settings not related to menus (so related to everything else on page). Currently there is only one setting available here: `breadcrumbs` - set it to True to see breadcrumbs on page, False to hide breadcrumbs.

custom_style Path to custom CSS file to be included on all admin pages. You should use `STATIC_URL` as prefix here. You can create your own custom style or use one of color themes provided with Django WP Admin. Those included themes are in `wpadmin/css/themes/` and here is a list of them: `blue.css`, `coffee.css`, `default.css` (this one is used by default so no need to include it), `ectoplasm.css`, `light.css`, `midnight.css`, `ocean.css`, `sunrise.css`. So if you like coffee then you should probably put `STATIC_URL + 'wpadmin/css/themes/coffee.css'` in this setting ;)

Creating custom menus

By default Django WP Admin mimics Django admin page, so it does not add any custom menus on left or top of the page (you can see such default and simplest setting here (login: staff, password: staff)). If you want to add top or left menu (like those that can be seen here (login: user, password: user)) then you have to create little more complicated setup.

First create `wp.py` file in your project's folder (you can use file from test project as starting template).

Then create menu class which should inherit from `wpadmin.menu.menus.Menu`:

```
from wpadmin.menu.menus import Menu

class MyMenu(Menu):
    """
    My super new menu ;)
    """
(...)
```

In this class you should create method called `init_with_context`:

```
(...)
    def init_with_context(self, context):
```

And populate `children` list with menu items in this method:

```
(...)
        self.children += [
            items.MenuItem(
                title='Back to page',
                url='/',
                icon='fa-bullseye',
                css_styles='font-size: 1.5em;',
            ),
            items.AppList(
                title='Applications',
                icon='fa-tasks',
                exclude=('django.contrib.*',),
            ),
            items.ModelList(
                title='Auth models',
                icon='fa-tasks',
                models=('django.contrib.auth.*',),
            ),
            items.UserTools(
                css_styles='float: right;',
            ),
        ]
```

All menu items must be instance of classes from `wpadmin.menu.items`. Here are available classes and their descriptions:

MenuItem Basic menu item which you would want to use to create menu items for specific urls. Properties this menu item can have:

`title` String that contains the menu item title, make sure you use the django gettext functions if your application is multilingual. Default value: 'Untitled menu item'.

`url` String that contains the menu item URL. Default value: None (will be rendered as 'javascript:;').

`add_url` An optional string that contains second menu item URL. This url allows to have edit and add urls in one menu item. add_url is rendered as a small plus sign in menu, next to normal url. Default value: None.

`icon` An optional string which contains classes for icons from Font Awesome which should be used for this menu item. Note that icons may not show on all levels of menu. They are only supported at top level. Default value: None.

`css_styles` String containing special CSS styling for this menu item. Default value: None.

`description` An optional string that will be used as the `title` attribute of the menu-item `a` tag. Default value: None.

`enabled` Boolean that determines whether the menu item is enabled or not. Disabled items are displayed but are not clickable. Default value: True.

`children` A list of children menu items. All children items must be instances of the `MenuItem` class or its subclasses.

AppList Menu item that lists available applications. It has two additional properties:

`models` List of strings containing paths to applications to be shown.

`exclude` List of strings containing paths to applications to be excluded.

ModelList Menu item that lists available models. It has same properties as **AppList**.

UserTools Special menu item to show "Welcome username" string with Gravatar and basic user options like logging out and changing password. Adding menu items to children property, setting url, title and description does not make sense for this menu item as it will be ignored when rendering.

Please refer to test project's wp.py file for more details and more complicated example.

1.5.2 Changes in Django's ModelAdmin behaviour

Ignored ModelAdmin options

Some options of Django's ModelAdmin are ignored when Django WP Admin is used:

ModelAdmin.actions_on_top Actions and pagination are always visible above objects lists.

ModelAdmin.actions_on_bottom Actions and pagination are always visible below objects lists.

ModelAdmin.save_on_top Save buttons are always displayed at the bottom of the page.

Additional ModelAdmin options

There is one additional class for fieldsets: `collapse-opened` - it tells Django to create collapsible fieldset but opened by default.

1.5.3 Translations

If you want to help to translate this software please join me on Transifex: transifex.com/projects/p/django-wp-admin/
Here is a list of available translations.

English

Source (default) language.

Bulgarian

Thanks to Metodi Dejanov

Dutch (Netherlands)

Thanks to rico moorman

French

Thanks to qmarlats

German

Thanks to Silasoa

Indonesian

Thanks to Al Firdaus

Italian

Thanks to Giuseppe Pignataro

Polish

100% by me (Maciej 'barszcz' Marczewski)

Portuguese (Brazil)

Thanks to Kaio Henrique

Russian

Thanks to Eugene MechanisM

1.5.4 Changelog (specific for 1.4.x branch)

v1.4.0 (2015-03-25)

- based on version 1.7.2
- templates modified to stay close to Django 1.4.x
- jQuery copied from Django 1.4

1.6 Troubleshooting

Please create an issue on GitHub if you have any problems or requests.

1.7 Credits

Python code is based on django-admin-tools app.

WordPress look and feel is of course inspired by WordPress.

Included icons comes from Font Awesome.