Django WP Admin Documentation

Release 1.8.0

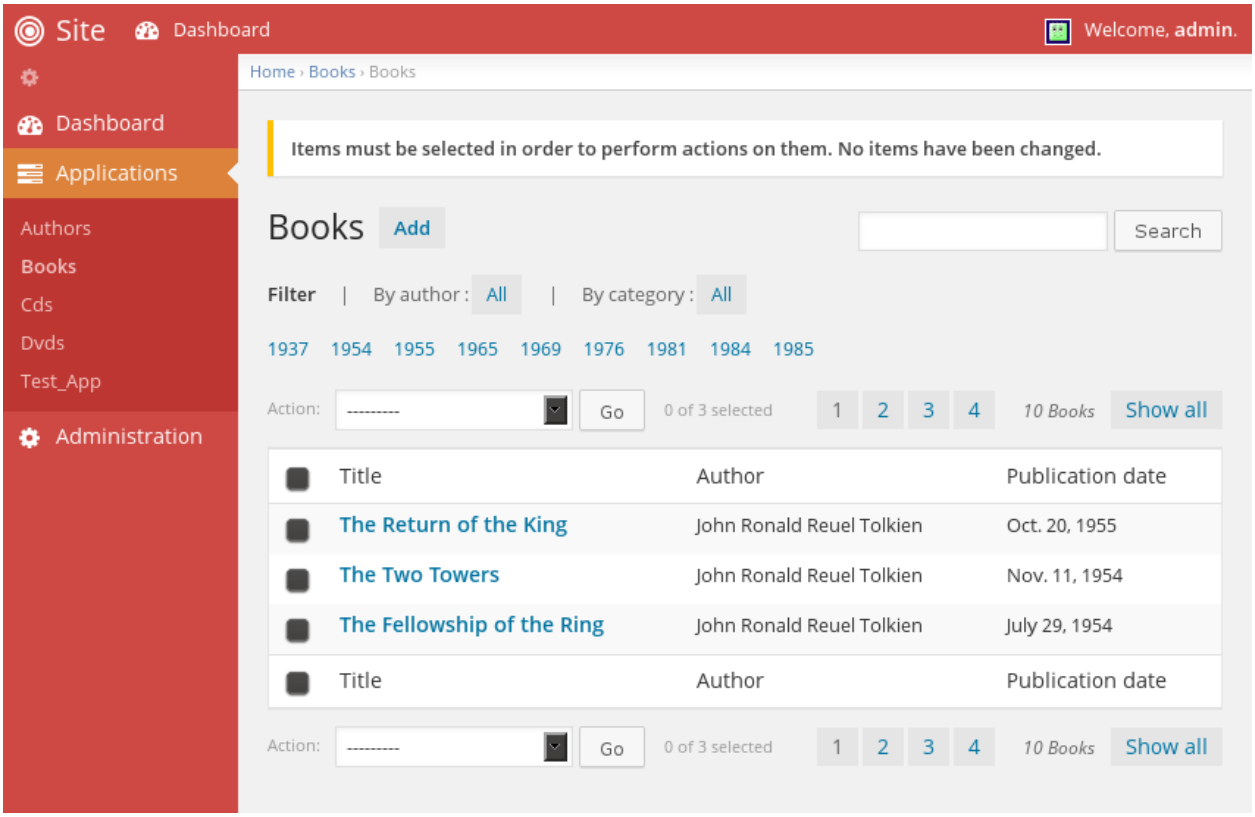
Maciej ’barszcz’ Marczewski

Apr 15, 2020

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
- Python3 compatible

1.2 Django compatibility

On GitHub there is specific branch of Django WP Admin for each major Django version and master branch is always for current stable Django. Releases on PyPI have numbering matching proper Django versions, so for instance for Django 1.7.x you should install Django WP Admin 1.7.y (pip install "django-wpadmin>=1.7,<1.8"). Branches of Django WP Admin for Django older than current stable usually will not have new features added, only bugs will be fixed. Only version for current stable Django will have new features, but any pull requests for older branches are welcome.

1.3 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.

1.4 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
```

1.5 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.

1.6 Advanced topics

1.6.1 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.

1.6.2 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.

1.6.3 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

1.6.4 Changelog

v1.8.0 (2020-04-15)

- merged template changes from Django 1.8
- testing project updated for Django 1.8

v1.7.4 (2015-05-08)

- small updates in styles
- typo fixed in one translation
- requirements in test project updated

v1.7.3 (2015-04-20)

- another fix for small visual regression

v1.7.2 (2015-03-24)

- fixed small visual regression
- added Dutch translation (thanks to Rico Moorman)

v1.7.1 (2015-03-23)

- change list page template modified to include object-tools and object-tools-items blocks
- styles modified to style added blocks
- small cleanup in some templates
- added new translations: Bulgarian (thanks to Metodi Dejanov), French (thanks to qmarlats), Portuguese (Brazil) (thanks to Kaio Henrique).
- updated docs about translations
- Font Awesome updated to version 4.3.0
- Less updated to version 2.4.0

v1.7.0 (2014-11-24)

- merged template changes from Django 1.7
- using new features (site_header and site_title) of AdminSite from Django 1.7
- testing project updated for Django 1.7
- small updates in styles

v1.6.3 (2014-11-20)

- fix for tabular inlines
- added sample tabular and stacked inlines in test project
- added new translations: German (thanks to Silasoa), Indonesian (thanks to Al Firdaus), Italian (thanks to Giuseppe Pignataro), Russian (thanks to Eugene MechanisM).
- updated docs about translations
- Font Awesome updated to version 4.2.0
- Less updated to version 2.0.0
- jQuery Cookie plugin updated to version 1.4.1
- added 2 new color themes: milo and milo-light

v1.6.2 (2014-03-21)

- fix for top menu hover colors in themes
- added color theme chooser on user panel in test project

v1.6.1 (2014-03-13)

- Python3 compatibility
- updated templates for password reset and change (from /registration template path)
- updated template for login page to make it more consistent with rest of the pages
- fixed bug with submenu when left menu is folded
- more fixes for small resolutions screens
- proper login forms in test project used

v1.6.0 (2014-03-11)

- **backward compatibility breaking release!**
- most of things was rewritten
- from now on there will be separate branch for each major Django version and Django WP Admin will be versioned according to Django version it supports...
- ...so this version is compatible only with Django 1.6.x
- all JavaScript libraries updated
- FontAwesome updated to version 4.0.3
- new WordPress admin look
- styles for selector widget
- styles for calendar widget
- styles for timepicker widget
- styles for delete confirmation page
- styles for history page
- collapsible fieldsets can be opened by default
- support for nested submenus
- 7 additional color themes added
- added licenses for all included external files (fonts and JS)

v0.2.0 (2013-04-02)

- styled object's editing pages

v0.1.2 (2013-03-23)

- **Django 1.3 support dropped!** (there's too much differences between Django 1.3 and 1.4)
- installation process slightly changed (there's no need to copy or symlink base.html file for specific Django version)
- added WordPress look and feel for objects lists (change_list.html)
- CHANGELOG added

v0.1.1 (2013-03-19)

- installation scripts
- README and LICENSE files added

v0.1.0 (2013-03-19)

- top and left menu

1.7 Troubleshooting

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

1.8 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.