Aldryn Categories Documentation

Release 1.1.0

Divio AG

December 17, 2018

Contents

Aldryn Categories provides Aldryn-compatible hierarchical categories/taxonomies for your Django project.

Aldryn Categories allows you to organise arbitrary objects in a hierarchical fashion using common Django fields and widgets.

Aldryn Categories is open-source software.

CHAPTER 1

Introduction

Installation

Installing packages

Then run either:

```
pip install aldryn-categories
```

or to install from the latest source tree:

```
pip install -e git+https://github.com/aldryn/aldryn-categories.git#egg=aldryn-categories
```

settings.py

In your project's `settings.py` make sure you have all of:

```
'parler',
'treebeard',
'aldryn_categories',
```

listed in `INSTALLED_APPS`.

Prepare the database and run

Now run `python manage.py migrate` to prepare the database for the new application, then `python manage.py runserver`.

For Aldryn users

On the Aldryn platform, the Addon is available from the Marketplace.

You can also install Aldryn Categories into any existing Aldryn project.

Basic usage

Aldryn Categories is a tool for developers to allow attaching of arbitrary models to a hierarchical taxonomy of categories.

To use this in your Django project, ensure that this project has been properly installed, then, add one of these field types to your model:

- `CategoryManyToManyField`
- `CategoryOneToOneField`
- `CategoryForeignKey`

Each acts exactly like the corresponding non-Category Django version with two differences. First, you won't need to specify the `to` argument on the field declaration as this automatically defaults to `Category`. Second, each presents the category choices in a hierarchical manner.

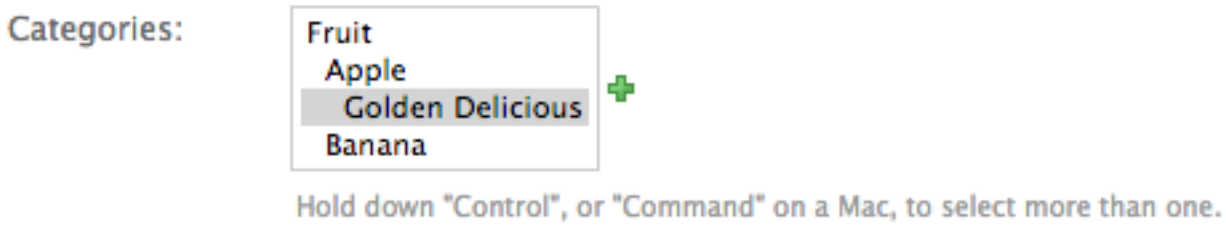
For example, if you would like to "attach" any number of categories to your `Thing` model:

```
# -*- coding: utf-8 -*-

from django.db import models
from aldryn_categories.fields import CategoryManyToManyField

class Thing(models.Model):
    my_field = models.CharField(...)
    ...
    categories = CategoryManyToManyField()
```

This usage of the `CategoryManyToManyField` simply allows your categories to be displayed hierarchically in the otherwise normal `MultipleSelectWidget` like so:



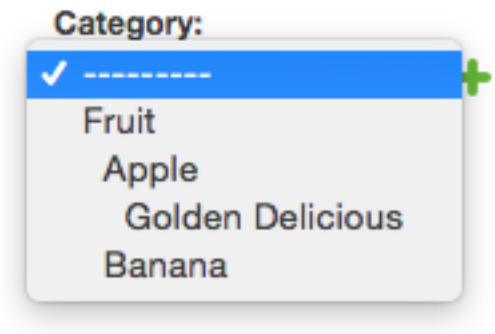
Similarly, the `CategoryModelChoiceField` provides similar presentation changes to support the `CategoryForeignKey` and `CategoryOneToOneField`. For example:

```
# -*- coding: utf-8 -*-

from django.db import models
from aldryn_categories.fields import CategoryForeignKey

class Thing(models.Model):
    my_field = models.CharField(...)
    ...
    category = CategoryForeignKey()
```

The widget produced would look like this:


Category:

Fruit
Apple
Golden Delicious
Banana

CHAPTER 2

Release notes & upgrade information

The CHANGELOG is maintained and updated within the repository.

CHAPTER 3

Development & community

Aldryn Categories is an open-source project.

You don't need to be an expert developer to make a valuable contribution - all you need is a little knowledge, and a willingness to follow the contribution guidelines.

Divio AG

Aldryn Categories was created by Divio AG and is released under a BSD licence.

Aldryn Categories is compatible with Divio's Aldryn cloud-based django CMS hosting platform, and therefore with any standard django CMS installation. The additional requirements of an Aldryn application do not preclude its use with any other django CMS deployment.

Divio is committed to Aldryn Categories as a high-quality application that helps set standards for others in the Aldryn/django CMS ecosystem, and as a healthy open source project.

Divio maintains overall control of the Aldryn Categories repository.

Standards & policies

Although Aldryn Categories is not strictly dependent on django CMS, it is maintained under the same standards and policies of django CMS.

These include:

- guidelines and policies for contributing to the project, including standards for code and documentation
- standards for managing the project's development
- a code of conduct for community activity

Please familiarise yourself with this documentation if you'd like to contribute to Aldryn Categories.

Running tests

Aldryn Categories uses django CMS Helper to run its test suite.

Backend Tests

To run the tests, in the aldryn-categories directory:

```
virtualenv env  # create a virtual environment
source env/bin/activate  # activate it
python setup.py install  # install the package requirements
pip install -r test_requirements/django-1.11.txt  # install the test requirements
python test_settings.py  # run the tests
```

You can run the tests against a different version of Django by using the appropriate value in `django-xx.txt` when installing the test requirements.

Frontend Tests

Follow the instructions in the aldryn-boilerplate-bootstrap3 documentation and setup the environment through the *Backend Tests* section.

Instead of using `python test_settings.py` described above, you need to execute `python test_settings.py server` to get a running local server. You can open the development server locally through `http://127.0.0.1:8000/`. The database is added within the root of this project `local.sqlite`. You might want to delete the database from time to time to start with a fresh installation. Don't forget to restart the server if you do so.