django-treenav Documentation

Release 1.1.0

Caktus Consulting Group LLC

May 25, 2017

Contents

Build Status

master

develop

coverage

An extensible, hierarchical, and pluggable navigation system for Django sites.

django-treenav was designed from the start to live independent of a CMS implementation. As a separate application, treenav can easily be integrated into existing, custom setups and does not enforce or require users to use a particular content management system.

Sharing the same principles, django-pagelets integrates seamlessly with treenav and can be used together to create a flexible CMS product.

For complete documentation checkout, http://django-treenav.readthedocs.org

CHAPTER 1

Features

- Generic functionality with multiple URL specifications: *get_absolute_url()*, *reverse()*, or raw URLs
- Packaged with templates to render the tree hierarchy with nested <*ul*>'s, but can easily be overridden with custom templates
- Useful CSS classes for flexible UI customization
- Automatically sets "active" on item and item's parents if *PATH_INFO* is equal to *item.href*
- Efficient: minimizes database access with django-mptt functionality
- Caches the tree so that repeated page views do not hit the database.
- Simple links in the *MenuItem* list view for refreshing the cache and href from the database.

CHAPTER 2

Requirements

- django >= 1.8
- django-mptt >= 0.8.6

CHAPTER 3

Using the demo

For a quick demo, follow these steps:

```
$ mkvirtualenv django-treenav
(django-treenav)$ git clone git://github.com/caktus/django-treenav.git
(django-treenav)$ cd django-treenav/
(django-treenav)~/django-treenav$ python setup.py develop
(django-treenav)~/django-treenav$ cd sample_project/
(django-treenav)~/django-treenav/sample_project$ pip install -r requirements.txt
(django-treenav)~/django-treenav/sample_project$ ./manage.py migrate
(django-treenav)~/django-treenav/sample_project$ ./manage.py runserver
```

Visit http://localhost:8000/ in your browser and follow the instructions.

CHAPTER 4

Installation

1. Install the app with pip:

```
pip install django-treenav
```

2. Add to your *INSTALLED_APPS* and run migrate:

```
INSTALLED_APPS = (
    ...,
    'mptt',
    'treenav',
)
```

3. Include these context processors:

```
TEMPLATES = [
  {
    'OPTIONS': {
      'context_processors': [
        "django.template.context_processors.request",
        "treenav.context_processors.treenav_active",
      ],
    },
  },
]
```

4. Add these urls:

```
urlpatterns = [
    url(r'^treenav/', include('treenav.urls')),
]
```

Development sponsored by Caktus Consulting Group, LLC.

CHAPTER 5

Contents

Setup

1. Login to the admin and build your menu item hierarchy
2. Load the `treenav_tags` in your template and render the menu where the *show_treenav argument* is the slug of the top level menu item e.g:

```
{% load treenav_tags %}
{% show_treenav 'primary-nav' %}
```

3. Apply CSS to your navigation by adding it to your base template and media directory using the classes described in *Menu HTML Example*.

Builtin Template Tags

Generally you will get access to the menus created by treenav through template tags. Currently treenav offers four template tags to use.

Show Treenav

```
{% show_treenav "top-level-slug" full_tree="False" %}
```

This is the canonical tag you will be using. It outputs nested lists starting with all the children of the menu item whose slug matches `top-level-slug`. Then it builds the next level down if one exists for the active leg of the tree and so on. Only the active portion of the tree will be shown unless *full_tree* is True, i.e. `full_tree="True"`. To see the HTML output go to *Menu HTML Example*.

Render Menu Children

```
{% render_menu_children model-object %}
```

This takes a MenuItem model instance and builds a nested list like *Menu HTML Example*. This is mostly used internally and is the same as `show_treenav` except it takes a model object instead of its slug and does not have a *full_tree* option.

Menu HTML Example

The above two templates tags use the same template to generate lists.

Note that Tests is the page currently active. It as well as its parents have the class *active-menu-item* attached to them. The links on active nodes are marked with the class *active-menu-item-link*. The depth-N class, where N starts at 0 and increases for each nested UL, it is often used to help apply custom CSS depending on how deeply a UL is nested.

The topmost list items and links are *top-level* and *top-level-link* respectively. Deeper lists have the class *parent-slug-child* which can be seen on the Test item.

```
<ul id="menu-primary-nav" class="menu depth-0">
    <li id="menu-item-home" class="inactive-menu-item top-level">
        <a class="inactive-menu-item-link top-level-link" href="/" title="Home">Home</
↪a>
    </li>
    <li id="menu-item-about" class="active-menu-item top-level">
        <a class="active-menu-item-link top-level-link" href="/about/" title="About">
↪About</a>
        <ul id="menu-about" class="menu depth-1">
            <li id="menu-item-test" class="active-menu-item about-child">
                <a class="active-menu-item-link about-child-link" href="/test/" title=
↪"Test">Test</a>
            </li>
        </ul>
    </li>
</ul>
```

Single Level Menu

```
{% single_level_menu "top-level-slug" 6 %}
```

This is used to build single-level navigations. The first argument is used to define the *MenuItem* that will be used as the top of the tree and the second argument defines the level to be displayed. Often this will be used when a page has multiple menus where one menu is the top level, and another is the first level of the active page.

For HTML output go to *Menu HTML Example*. Although the example is two levels, this can only output one. One thing to note is the if you select a depth of 3, then the depth class mentioned above will be depth-3, not 0 even though its the top(and only) UL.

Show Menu Crumbs

```
{% show_menu_crumbs "top-level-slug" %}
```

Builds a list of the active MenuItems between the *top-level-slug* and the current active menu item so as to build a breadcrumb stucture.

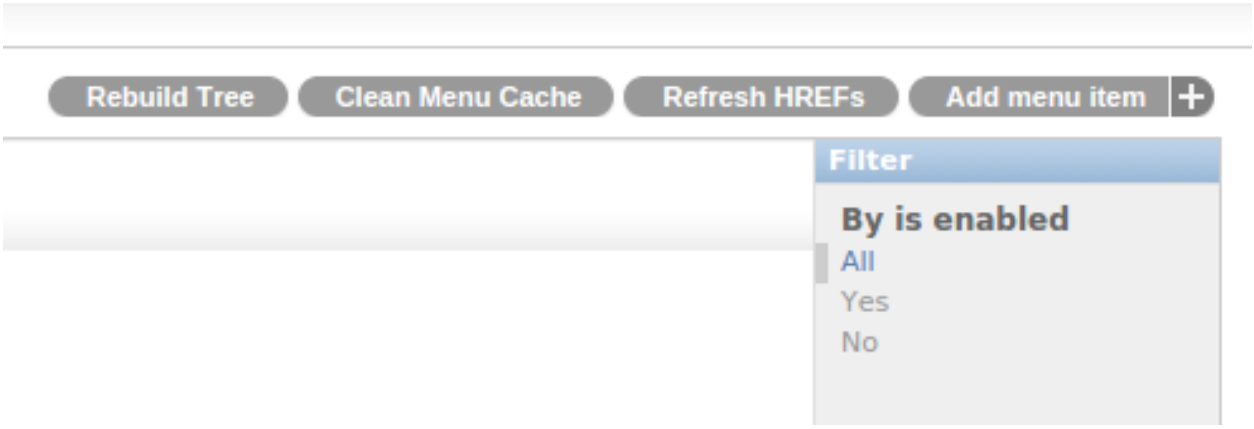
HTML Output

The HTML generated by *show_menu_crumbs*.

```
<ul>
    <li>
        <a href="/" title="Home">
            Home
        </a>
    </li>
    <li>
        <a href="/admin" title="Test">
            Test
        </a>
    </li>
    <li>
        <a href="/test/" title="New Test">
            New Test
        </a>
    </li>
</ul>
```

Troubleshooting

There have been reports of menu items becoming improperly ordered in various situations.[1][2][3][4] If that occurs, you may be able to rebuild your trees in the database by going to the MenuItem admin and clicking 'Rebuild Tree'.



Release History

Release and change history for django-treenav

[1] https://github.com/caktus/django-treenav/issues/42
[2] https://github.com/caktus/django-treenav/issues/2
[3] https://github.com/caktus/django-treenav/issues/7
[4] https://github.com/caktus/django-treenav/issues/17

vNEXT (Released DATE)

- To be determined

v1.1.0 (Released 2016-12-14)

- Django 1.9 and 1.10 support (#67)
- Dropped support for Django 1.7

v1.0.0 (Released 2015-10-01)

This is a stable release supporting Django 1.8 and Python 3.

- Python 3 support (#28)
- Django 1.8 support (#40)
- Confirmed support for django-mptt 0.7
- Confirmed support for Django 1.7
- Dropped support for Django prior to 1.7
- Dropped support for Python 2 prior to 2.7
- Dropped support for Python 3 prior to 3.3
- Dropped support for django-mptt prior to 0.7
- Setup Travis (#27)
- Add docs for 'Rebuild Tree' admin action.
- Updated Tox (#45) and Travis (#50) to work with Tox 2.0
- Updated sample project to work with Django 1.8 and django-mptt 0.7 (#25)
- Fixed bug that prevented deletion of items from admin changelist (#54)
- Fixed bug where reordering items in the admin would disorder tree (#42)
- Switched to Django migrations (#63)

Backwards Incompatible Changes

- `MenuItem` model no longer uses `.tree` for `TreeManager` methods. Instead use `.objects` (change from django-mptt 0.5)
- If you are upgrading from 0.9.2 (or earlier) and are currently using Django 1.8, then you will need to run `python manage.py migrate --fake-initial` since we have converted from using South to Django migrations. Django 1.7 does "fake-initial" behavior by default.
- The `post_save` signal which updates MenuItems when their corresponding object changes will NOT be active during data migrations. If you update any objects during a data migration, and that object has a corresponding MenuItem, you will need to manually update the HREF of the MenuItem.

v0.9.2 (Released 2015-09-02)

- Add 'Rebuild Tree' button for MenuItem admin

v0.9.1 (Released 2012-10-26)

- Allow MenuItem.parent to be blank
- Fixed bug that prevented display of more than 2 levels of tree when `full_tree` was set to True

v0.9.0 (Released 2012-10-26)

This release is a clean up release for the API and internals. This should be considered a release candidate for an offical v1.0.

- Dropped support for Django prior to 1.3
- Added tox support for testing

Backwards Incompatible Changes

- `treenav_refresh_hrefs` and `treenav_clean_cache` were moved under the admin and require staff access
- Including url patterns has been simplified and `treenav.urls.admin` were removed

0.6.1 (Released 2012-04-12)

- Fixed bug that prevented adding parent and child menu items simultaneously

0.6.0 (Released 2011-12-02)

- Moved to GitHub
- Add Sphinx-powered documentation
- Update to Django 1.3.x and django-mptt 0.5.2
- Provide more order choices by default
- Fix a few documentation related bugs
- Cleaned up sample project for an easier demo

0.5.0 (Released 2011-03-11)

- Initial release

CHAPTER 6

Indices and tables

- genindex
- modindex
- search