django-sitetree Documentation

Release 0.9

Igor 'idle sign' Starikov

March 29, 2015

Contents

django-sitetree is a reusable application for Django, introducing site tree, menu and breadcrumbs navigation elements.

Site structure in django-sitetree is described through Django admin interface in a so called site trees. Every item of such a tree describes a page or a set of pages through the relation of URI or URL to human-friendly title. Eg. using site tree editor in Django admin:

```
URI             Title
  /             - Site Root
  |_users/      - Site Users
    |_users/13/ - Definite User
```

Alas the example above makes a little sense if you have more than just a few users, that's why django-sitetree supports Django template tags in item titles and Django named URLs in item URIs.

If we define a named URL for user personal page in urls.py, for example, 'users-personal', we could change a scheme in the following way:

```
URI                           Title
  /                           - Site Root
  |_users/                    - Site Users
    |_users-personal user.id  - User Called {{ user.first_name }}
```

After setting up site structure as a sitetree you should be able to use convenient and highly customizable site navigation means (menus, breadcrumbs and full site trees).

User access to certain sitetree items can be restricted to authenticated users or more accurately with the help of Django permissions system (Auth contrib package).

CHAPTER 1

Requirements

1. Django 1.2+
2. Admin site Django contrib package
3. Auth Django contrib package
4. South 0.7.1+ for Django (required for version upgrades)

Table of Contents

2.1 Getting started

1. Add the **sitetree** application to INSTALLED_APPS in your settings file (usually 'settings.py').
2. Check that *django.core.context_processors.request* is enabled in TEMPLATE_CONTEXT_PROCESSORS in your settings file.
3. Check that *django.contrib.auth.context_processors.auth* is enabled in TEMPLATE_CONTEXT_PROCESSORS too.
4. Run './manage.py syncdb' to install sitetree tables into database.
5. Go to Django Admin site and add some trees and tree items (see *Making tree* section).
6. Add *{% load sitetree %}* tag to the top of a template.

Now you can use the following template tags:

- *sitetree_menu* - to render menu based on sitetree;
- *sitetree_breadcrumbs* - to render breadcrumbs path based on sitetree;
- *sitetree_tree* - to render site tree;
- *sitetree_page_title* - to render current page title resolved against definite sitetree.

2.1.1 Upgrade hint

When switching from older version of SiteTree to newer do not forget to upgrade your database schema.

That could be done with the following command issued in your Django project directory:

```
./manage.py migrate
```

Note that the command **requires** South.

2.1.2 Making tree

Taken from StackOverflow.

In this tutoral we create sitetree that could handle URI like */categoryname/entryname*.

To create a tree:

1. Go to site administration panel;
2. Click +Add near 'Site Trees';
3. Enter alias for your sitetree, e.g. 'maintree'. You'll address your tree by this alias in template tags;
4. Push 'Add Site Tree Item';
5. Create first item:

```
Parent - As it is root item that would have no parent.
Title - Let it be 'My site'.
URL - This URL is static, so put here '/'.
```

6. Create second item (that one would handle 'categoryname' from your 'categoryname/entryname'):

```
Parent - Choose 'My site' item from step 5.
Title - Put here 'Category #{{ category.id }}'.
URL - Put named URL 'category-detailed category.name'.

In 'Additional settings': check 'URL as Pattern' checkbox.
```

7. Create third item (that one would handle 'entryname' from your 'categoryname/entryname'):

```
Parent - Choose 'Category #{{ category.id }}' item from step 6.
Title - Put here 'Entry #{{ entry.id }}'.
URL - Put named URL 'entry-detailed category.name entry.name'.

In 'Additional settings': check 'URL as Pattern' checkbox.
```

8. Put '{% load sitetree %}' into yor template to have access to sitetree tags.
9. Put '{% sitetree_menu from "maintree" %}' into your template to render menu.
10. Put '{% sitetree_breadcrumbs from "maintree" %}' into your template to render breadcrumbs.

Steps 6 and 7 clarifications:

- In titles we use Django template variables, which would be resolved just like they do in your templates.

  E.g.: You made your view for 'categoryname' (let's call it 'detailed_category') to pass category object into template as 'category' variable. Suppose that category object has 'id' property. In your template you use '{{ category.id }}' to render id. And we do just the same for site tree item in step 6.

- In URLs we use Django's named URL patterns (documentation). That is almost idential to usage of Django 'url' tag in templates.

  Your urls configuration for steps 6, 7 supposed to include:

```
url(r'^(?P<category_name>\S+)/(?P<entry_name>\S+)/$', 'detailed_entry', name='entry-detailed'),
url(r'^(?P<category_name>\S+)/$', 'detailed_category', name='category-detailed'),
```

  Consider 'name' argument values of 'url' function.

  So, putting 'entry-detailed category.name entry.name' in step 7 into URL field we tell sitetree to associate that sitetree item with URL named 'entry-detailed', passing to it category_name and entry_name parameters.

2.2 SiteTree template tags

To use template tags available in SiteTree you should add **{% load sitetree %}** tag to the top of chosen template.

Tree tag argument (part in double quotes, following '**from**' word) of SiteTree tags should containt tree alias.

Hints:

- Tree tag argument could be a template variable (do not use quotes for those).
- Optional **template** argument could be supplied to all SitetTree tags except *sitetree_page_title* to render using different templates. It should contain path to template file.

  Examples:

```
{% sitetree_menu from "mytree" include "trunk,topmenu" template "mytrees/mymenu.html" %}
{% sitetree_breadcrumbs from "mytree" template "mytrees/mybreadcrumbs.html" %}
```

2.2.1 sitetree_menu

This tag renders menu based on sitetree.

Usage example:

```
{% sitetree_menu from "mytree" include "trunk,topmenu" %}
```

This command renders as a menu sitetree items from tree named 'mytree', including items **under** 'trunk' and 'topmenu' aliased items. That means that 'trunk' and 'topmenu' themselves won't appear in a menu, but rather their ancestors. If you need item filtering behaviour please use *tree hooks*.

Aliases are given to items through Django's admin site.

Note that there are some reserved aliases. To illustrate how do they work, take a look at the sample tree:

```
Users
  |-- Moderators
  |-- Ordinary

Articles
  |-- About cats
        |-- Good
        |-- Bad
        |-- Ugly
  |-- About dogs
  |-- About mice

Contacts
  |-- Russia
        |-- Web
              |-- Public
              |-- Private
        |-- Postal
  |-- Australia
  |-- China
```

- **trunk** - get items without parents (root items);

  Renders:

```
Users
Articles
Contacts
```

- **this-children** - get items under item resolved as current for the current page;

  Considering that we are now at *Articles* renders:

```
About cats
About dogs
About mice
```

- **this-siblings** - get items under parent of item resolved as current for the current page (current item included);

  Considering that we are now at *Bad* renders:

  ```
  Good
  Bad
  Ugly
  ```

- **this-ancestor-children** - items under grandparent item (closest to root) for the item resolved as current for the current page.

  Considering that we are now at *Public* renders:

  ```
  Web
  Postal
  ```

Thus in the template tag example above 'trunk' is reserved alias, and 'topmenu' alias is given to an item through admin site.

Sitetree items could be addressed not only by aliases but also by IDs:

```
{% sitetree_menu from "mytree" include "10" %}
```

2.2.2 sitetree_breadcrumbs

This tag renders breadcrumbs path (from tree root to current page) based on sitetree.

Usage example:

```
{% sitetree_breadcrumbs from "mytree" %}
```

This command renders breadcrumbs from tree named 'mytree'.

2.2.3 sitetree_tree

This tag renders entire site tree.

Usage example:

```
{% sitetree_tree from "mytree" %}
```

This command renders sitetree from tree named 'mytree'.

2.2.4 sitetree_page_title

This tag renders current page title resolved against definite sitetree. Title is taken from sitetree item title resolved for current page.

Usage example:

```
{% sitetree_page_title from "mytree" %}
```

This command renders current page title from tree named 'mytree'.

2.3 Internationalization

With django-sitetree it is possible to render different trees for different active locales still addressing them by the same alias from a template.

`register_i18n_trees(aliases)` function registers aliases of internationalized sitetrees. Internationalized sitetrees are those, which are dubbed by other trees having locale identifying suffixes in their aliases.

Lets suppose `my_tree` is the alias of a generic tree. This tree is the one that we call by its alias in templates, and it is the one which is used if no i18n version of that tree is found.

Given that `my_tree_en`, `my_tree_ru` and other `my_tree_{locale-id}`-like trees are considered internationalization sitetrees. These are used (if available) in accordance with current locale used in project.

Example:

```
# First import the register function.
from sitetree.sitetreeapp import register_i18n_trees

# Now register i18n trees.
register_i18n_trees(['my_tree', 'my_another_tree'])

# After that you need to create trees for languages supported
# in your project, e.g.: `my_tree_en`, `my_tree_ru`.

# Then when we address ``my_tree`` from a template django-sitetree will render
# an appropriate tree for locale currently active in your project.
# See ``activate`` function from ``django.utils.translation``
# and https://docs.djangoproject.com/en/dev/topics/i18n/internationalization
# for more information.
```

2.4 Management commands

SiteTree comes with two management commands which can facilitate development and deployment processes.

2.4.1 sitetreedump

Sends sitetrees from database as a fixture in JSON format to output.

Output all trees and items into *treedump.json* file example:

```
python manage.py sitetreedump > treedump.json
```

You can export only trees that you need by supplying their aliases separated with spaces:

```
python manage.py sitetreedump my_tree my_another_tree > treedump.json
```

If you need to export only tree items without trees use `--items_only` command switch:

```
python manage.py sitetreedump --items_only my_tree > items_only_dump.json
```

Use `--help` command switch to get quick help on the command:

```
python manage.py sitetreedump --help
```

2.4.2 sitetreeload

This command loads sitetrees from a fixture in JSON format into database.

Command makes use of `--mode` command switch to control import strategy.

> a) *append* (default) mode should be used when you need to extend sitetree data that is now in DB with that from a fixture.
>
> **Note:** In this mode trees and tree items identifiers from a fixture will be changed to fit existing tree structure.
>
> b) *replace* mode should be used when you need to remove all sitetree data existing in DB and replace it with that from a fixture.
>
> **Warning:** Replacement is irreversible. You should probably dump sitetree data if you think that you might need it someday.

Using *replace* mode:

```
python manage.py sitetreeload --mode=replace treedump.json
```

Import all trees and items from *treedump.json* file example:

```
python manage.py sitetreeload treedump.json
```

Use `--items_into_tree` command switch and alias of target tree to import all tree items from a fixture there. This will not respect any trees information from fixture file - only tree items will be considered. **Keep in mind** also that this switch will automatically change *sitetreeload* commmand into *append* mode:

```
python manage.py sitetreeload --items_into_tree=my_tree items_only_dump.json
```

Use `--help` command switch to get quick help on the command:

```
python manage.py sitetreeload --help
```

2.5 Notes on built-in templates

Default templates shipped with SiteTree created to have as little markup as possible in a try to fit most common website need.

2.5.1 Styling built-in templates

Use CSS to style default templates for your needs. Templates are deliberately made simple, and only consist of *ul*, *li* and *a* tags.

Nevertheless pay attention that menu template also uses two CSS classes marking tree items:

- **current_item** — marks item in the tree, corresponding to current page;
- **current_branch** — marks all ancestors of current item, and current item itself.

2.5.2 Overriding built-in templates

To customize visual representation of navigation elements you should override the built-in SiteTree templates as follows:

1. Switch to sitetree folder
2. Switch further to 'templates/sitetree'
3. There you'll find the following templates:

- breadcrumbs.html
- menu.html
- tree.html

4. Copy whichever of them you need into your project templates directory and feel free to customize it.
5. See *Advanced SiteTree tags section* for clarification on two advanced SiteTree template tags.

2.5.3 Templates for Foundation CSS Framework

Information about Foundation CSS Framework is available at http://foundation.zurb.com

The following templates are bundled with SiteTree:

- *sitetree/menu_foundation.html*

  This template can be used to construct Foundation Nav Bar (classic horizontal top menu) from a sitetree.

  Note: The template renders no more than two levels of a tree with hover dropdowns for root items having children.

- *sitetree/menu_foundation-vertical.html*

  This template can be used to construct a vertical version of Foundation Nav Bar, suitable for sidebar navigation.

  Note: The template renders no more than two levels of a tree with hover dropdowns for root items having children.

- *sitetree/sitetree/menu_foundation_sidenav.html*

  This template can be used to construct a Foundation Side Nav.

  Note: The template renders only one tree level.

You can take a look at Foundation navigation elements examples at http://foundation.zurb.com/docs/navigation.php

2.5.4 Templates for Bootstrap CSS Framework

Information about Bootstrap CSS Framework is available at http://twitter.github.com/bootstrap/

The following templates are bundled with SiteTree:

- *sitetree/breadcrumbs_bootstrap.html*

  This template can be used to construct a breadcrumb navigation from a sitetree.

- *sitetree/menu_bootstrap.html*

  This template can be used to construct *menu contents* for Boostrap Navbar.

Warning: To widen the number of possible use-cases for which this template can be applied, it renders only menu contents, but not Navbar container itself.
This means that one should wrap *sitetree_menu* call into the appropriately styled divs (i.e. having classes *navbar*, *navbar-inner*, etc.).
Please see Boostrap Navbar documentation for more information on subject.

Note: The template renders no more than two levels of a tree with hover dropdowns for root items having children.

- *sitetree/menu_bootstrap_navlist.html*

  This template can be used to construct a Boostrap Nav list.

  Note: The template renders only one tree level.

You can find Bootstrap navigation elements examples at http://twitter.github.com/bootstrap/components.html#navbar

2.6 Advanced SiteTree tags

SiteTree introduces two advanced template tags which you have to deal with in case you override the built-in sitetree templates.

2.6.1 sitetree_children

Implements down the tree traversal with rendering.

Usage example:

```
{% sitetree_children of someitem for menu template "sitetree/mychildren.html" %}
```

Used to render child items of specific sitetree item 'someitem' for 'menu' navigation type, using template "sitetree/mychildren.html".

Allowed navigation types: 1) *menu*; 2) *sitetree*.

Basically template argument should contain path to current template itself.

2.6.2 sitetree_url

Resolves site tree item's url or url pattern.

Usage example:

```
{% sitetree_url for someitem params %}
```

This tag is much the same as Django built-in 'url' tag. The difference is that after 'for' it should get site tree item object.

And, yes, you can pass some params after that object.

2.7 Tree hooks

What to do if a time comes and you need some fancy stuff done to tree items that django-sitetree does not support? It might be that you need some special tree items ordering in a menu, or you want to render in a huge site tree with all articles titles that are described by one tree item in Django admin, or god knowns what else.

django-sitetree can facilitate on that as it comes with `register_items_hook(callable)` function which registers a hook callable to process tree items right before they are passed to templates.

Note that callable should be able to:

1. **handle `tree_items` and `tree_sender` key params.** `tree_items` will contain a list of extended TreeItem objects ready to pass to template.

   `tree_sender` will contain navigation type identifier (e.g.: *menu*, *sitetree*, *breadcrumbs*, *menu.children*, *sitetree.children*)

2. return a list of extended TreeItems objects to pass to template.

Example:

```
# First import the register function.
from sitetree.sitetreeapp import register_items_hook

# The following function will be used as items processor.
def my_items_processor(tree_items, tree_sender):
    # Suppose we want to process only menu child items.
    if tree_sender == 'menu.children':
        # Lets add 'Hooked: ' to resolved titles of every item.
        for item in tree_items:
            item.title_resolved = 'Hooked: %s' % item.title_resolved
    # Return items list mutated or not.
    return tree_items

# And we register items processor.
register_items_hook(my_items_processor)
```

Note: You might also be interested in the notes on *Overriding SiteTree Admin representation*.

2.8 Overriding SiteTree Admin representation

SiteTree allows you to override tree and tree item representation in Django Admin interface.

That could be used not only for the purpose of enhancement of visual design but also for integration with other applications, using admin inlines. The following functions from *sitetree.admin* could be used to override tree and tree item representation:

- *override_tree_admin()* is used to customize tree representation.
- *override_item_admin()* is used to customize tree item representation.

Example:

```
# Supposing we are in admin.py of your own application.

# Import two helper functions and two admin models to inherit our custom model from.
from sitetree.admin import TreeItemAdmin, TreeAdmin, override_tree_admin, override_item_admin
```

```
# This is our custom tree admin model.
class CustomTreeAdmin(TreeAdmin):
    exclude = ('title',)  # Here we exclude 'title' field from form.

# And our custom tree item admin model.
class CustomTreeItemAdmin(TreeItemAdmin):
    # That will turn a tree item representation from the default variant
    # with collapsible groupings into a flat one.
    fieldsets= None

# Now we tell the SiteTree to replace generic representations with custom.
override_tree_admin(CustomTreeAdmin)
override_item_admin(CustomTreeItemAdmin)
```

Note: You might also be interested in using *Tree hooks*.

2.8.1 Inlines override example

In the example below we'll use django-seo application from https://github.com/willhardy/django-seo

According to django-seo documentation it allows an addition of custom metadata fields to your models, so we use it to connect metadata to sitetree items.

That's how one might render django-seo inline form on sitetree item create and edit pages:

```
from rollyourown.seo.admin import get_inline
from sitetree.admin import TreeItemAdmin, TreeAdmin, override_tree_admin, override_item_admin
# Let's suppose our application contains seo.py with django-seo metadata class defined.
from myapp.seo import CustomMeta


class CustomTreeItemAdmin(TreeItemAdmin):
    inlines = [get_inline(CustomMeta)]

override_item_admin(CustomTreeItemAdmin)
```

CHAPTER 3

Get involved into django-sitetree

Submit issues. If you spotted something weird in application behavior or want to propose a feature you can do that at https://github.com/idlesign/django-sitetree/issues

Write code. If you are eager to participate in application development, fork it at https://github.com/idlesign/django-sitetree, write your code, whether it should be a bugfix or a feature implementation, and make a pull request right from the forked project page.

Translate. If want to translate the application into your native language use Transifex: https://www.transifex.net/projects/p/django-sitetree/.

Spread the word. If you have some tips and tricks or any other words in mind that you think might be of interest for the others — publish it.

CHAPTER 4

The tip

If the application is not what you want for site navigation, you might be interested in considering the other choices — http://djangopackages.com/grids/g/navigation/