django-pagetree Documentation

Release 1.4.1

Anders Pearson

Dec 06, 2017

Contents

1 Installation **3**

2 Configuration **5**

3 Custom Pageblocks **7**

4 API **9**
4.1 Models 9
4.2 Views 9

5 Testing **11**

6 Glossary **13**

django-pagetree is a helper for building sites that are organized as a hierarchy of pages which the user/visitor goes through in (depth-first) order.

The pages can then each have 'blocks' attached to them which are content or interactive things.

See django-pageblocks for a basic set of these blocks.

django-pagetree is designed to allow this kind of site to be built by an editor through the web. It aims to provide the minimum amount of functionality possible and stay out of the way as much as possible.

CHAPTER 1

Installation

You can install `django-pagetree` through `pip`:

```
$ pip install django-pagetree
```

In your project, add `django-pagetree` to your `requirements.txt`.

Add to `INSTALLED_APPS` in your `settings.py`:

```
'pagetree',
```

The `PAGEBLOCKS` variable in your `settings.py` determines which pageblocks will be available on your site:

```
PAGEBLOCKS = [
    'pageblocks.TextBlock',
    'pageblocks.HTMLBlock',
]
```

To use these pageblocks, you'll need to put `django-pageblocks` in your `requirements.txt`, and add `'pageblocks'` to your `INSTALLED_APPS`.

django-pagetree provides a set of generic views that you can use to build a barebones site out of the box. In your `urls.py`, you will need to import the generic views:

```
from pagetree.generic.views import PageView, EditView, InstructorView
```

Then add the following URL routes:

```
(r'^pagetree/', include('pagetree.urls')),
(r'^pages/edit/(?P<path>.*)$',
 EditView.as_view(hierarchy_name="main", hierarchy_base="/pages/"),
 {}, 'edit-page'),
(r'^pages/instructor/(?P<path>.*)$',
 InstructorView.as_view(
     hierarchy_name="main", hierarchy_base="/pages/")),
```

```
(r'^pages/(?P<path>.*)$',
 PageView.as_view(hierarchy_name="main", hierarchy_base="/pages/")),
```

CHAPTER 2

Configuration

PAGETREE_CUSTOM_CACHE_CLEAR Use this as a hook to clear any custom caches you've set up. It will get called whenever Pagetree's internal cache is called. The function should take one argument: the *Section* whose cache is getting cleared.

CHAPTER 3

Custom Pageblocks

You might want to define custom pageblock types specific to your application.

It's possible to define a custom pageblock from scratch by defining a model with all the necessary hooks and a `GenericRelation` to django-pagetree's `PageBlock` class. For convenience, django-pagetree provides `BasePageBlock` that contains the basics you'll need for making a custom pageblock.

Here's an example of a custom pageblock:

```
from django import forms
from pagetree.generic.models import BasePageBlock


class MyBlock(BasePageBlock):
    display_name = 'My Block Name'
    template_file = 'main/my_block.html'
    css_template_file = 'main/my_block.css'
    js_template_file = 'main/my_block.js'

    @staticmethod
    def add_form():
        return MyBlockForm()

    def edit_form(self):
        return MyBlockForm(instance=self)

    @staticmethod
    def create(request):
        form = MyBlockForm(request.POST)
        return form.save()

    @classmethod
    def create_from_dict(cls, d):
        return cls.objects.create(**d)

    def edit(self, vals, files):
```

```
        form = MyBlockForm(data=vals, files=files, instance=self)
        if form.is_valid():
            form.save()


class MyBlockForm(forms.ModelForm):
    class Meta:
        model = MyBlock
        fields = '__all__'
```

Here's a list of methods and properties you can override in your `BasePageBlock` subclass:

CHAPTER 4

API

4.1 Models

4.2 Views

CHAPTER 5

Testing

It can be useful to programmatically set up a pagetree site for testing purposes. If you have custom pageblocks that rely on JavaScript for essential functionality, you won't be able to test that code with django's built-in testing features. You can use Selenium with Behave or Lettuce to do this kind of testing. This page shows how to mock a version of your pagetree site in code. It can then be used for Selenium tests or for Django-style unittests.

Here's an example of a factory that you can put alongside `factory_boy` factories:

```
from pagetree.tests.factories import HierarchyFactory


class CustomPagetreeModuleFactory(object):
    def __init__(self):
        hierarchy = HierarchyFactory(name='main', base_url='/pages/')
        root = hierarchy.get_root()
        root.add_child_section_from_dict({
            'label': 'Welcome to the Intro Page',
            'slug': 'intro',
            'children': [
                {
                    'label': 'Step 1',
                    'slug': 'step-1',
                    'pageblocks': [{
                        'block_type': 'Text Block'
                    }]
                },
                {
                    'label': 'Step 2',
                    'slug': 'step-2',
                    'pageblocks': [{
                        'block_type': 'My Block Name'
                    }]
                },
            ]
        })
```

```
        self.root = root
```

To instantiate a custom pageblock in this way, you set `block_type` to the custom pageblock's `display_name` property.

Then, if you're writing a behave test, you can call this factory in `environment.py`:

```
def before_all(context):
    CustomPagetreeModuleFactory()
```

And navigate the hierarchy in the feature file:

```
Feature: Navigate the pagetree hierarchy
  Scenario: Access custom block on Step 2
    When I visit "/pages/"
    Then I see the text "Welcome to the Intro Page"

    When I click the next button
    Then I see the text "Step 1"

    When I click the next button
    Then I see the text "Step 2"
    Then I see the text "My Block Name"
```

CHAPTER 6

Glossary

This is a glossary of terms used in pagetree.

section A "section" in pagetree refers to a node in the tree. A section can be thought of as a page in your hierarchy. Keep in mind that each section can contain any number of child sections.

locked / unlocked If a section is "locked", that means the user can't navigate past it with the "next" button.

gating / is_gated Gating is a feature that prevents users from visiting a section if they haven't visited all the preceding sections. If a pagetree site is "gated" and you try to visit a section in the middle of the tree with a new user, you will be redirected to the first node (or "section") in the tree.