django-pagelets Documentation

Release 0.10.0

Caktus Consulting Group

February 04, 2017

Contents

1 Features **3**

2 Required Dependencies **5**

3 Optional Dependencies **7**

4 Installation and Setup **9**

4.1 Running the Sample Project 10

4.2 Optional Additional Setup 10

4.3 Built-in Template Tags 11

4.4 List of Available Settngs 12

4.5 Release History 13

5 Indices and tables **17**

django-pagelets is a simple, flexible app for integrating static, unstructured content in a Django site.

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

CHAPTER 1

Features

- “Pagelets” for adding small pieces of content to otherwise static templates
- CMS “pages” which include any number of pagelets and, if needed, attachments
- Different pagelet content types including HTML and Markdown
- An integrated WYSIWYG editor (WYMeditor) which can be selectively enabled/disabled

CHAPTER 2

Required Dependencies

- Django >= 1.8
- A Python version supported by your chosen Django version
- Django admin site
- django-taggit 0.12.1 or greater
- django-selectable 0.9.0 or greater
- The *django.template.context_processors.request* context processor

CHAPTER 3

Optional Dependencies

- jQuery 1.7
- WYMeditor (included in pagelets media)

CHAPTER 4

Installation and Setup

1. django-pagelets is available on PyPI, so the easiest way to install it is to use pip:

```
pip install django-pagelets
```

2. Add *pagelets*, *selectable* and *taggit* to INSTALLED_APPS in settings.py and run migrate:

```
INSTALLED_APPS = (
    ...,
    'pagelets',
    'selectable',
    'taggit'
    ...
)
```

3. Make sure *django.template.context_processors.request* is loaded and that you have a template directory with a "base.html" template in it:

```
TEMPLATES=[
    {
        ...
        'DIRS': ['/home/user/projects/myproject/templates'], # <- should have 'base.html' inside
        ...
        'OPTIONS': {
            'context_processors': [
                ...
                'django.template.context_processors.request',
            ]
        },
    },
],
```

4. Add the pagelets URLs to urls.py, e.g.:

```
urlpatterns += [
    url(r'^selectable/', include('selectable.urls')),
    url(r'^pagelets-management/', include('pagelets.urls.management')),
    url(r'^', include('pagelets.urls.content')),
]
```

5. Visit the admin site, add and save a new page, and click the View on site link. If everything is setup correctly, you should be able to see and edit the content you just added.

Development sponsored by Caktus Consulting Group, LLC..

Contents:

4.1 Running the Sample Project

django-pagelets includes a sample project to use as an example setup. You can run the sample project like so:

```
~$ mkvirtualenv --distribute pagelet-test
(pagelet-test)~$ pip install -U django
(pagelet-test)~$ git clone git://github.com/caktus/django-pagelets.git
(pagelet-test)~$ cd django-pagelets/
(pagelet-test)~/django-pagelets$ python setup.py develop
(pagelet-test)~/django-pagelets$ cd sample_project/
(pagelet-test)~/django-pagelets/sample_project$ ./manage.py syncdb
(pagelet-test)~/django-pagelets/sample_project$ ./manage.py runserver
```

4.2 Optional Additional Setup

4.2.1 Sitemap

If you are using the *contrib.sitemaps* application to generate your sitemap you can make use of the *PageSiteMap*, e.g.:

```
from django.conf.urls.defaults import *
from pagelets.sitemaps import PageSiteMap

sitemaps = {
    'pagelets': PageSiteMap(priority=0.6),
    # Your other sitemaps
    # ...
}

# Site url patterns would go here
# ...

urlpatterns += patterns('',

    # the sitemap
    (r'^sitemap\.xml$', 'django.contrib.sitemaps.views.sitemap', {'sitemaps': sitemaps}),
)
```

4.2.2 Auto template tag loading

To automatically load a custom template tag on every pagelet, add a `PAGELET_TEMPLATE_TAGS` list to settings.py:

```
PAGELET_TEMPLATE_TAGS = (
    'myapp_tags',
    'myotherapp_tags',
)
```

4.2.3 Custom base templates and content areas

By default, django-pagelets uses a simplified setup for rendering pages in a uniform way. However, pages can be modified to extend from different base templates for greater customization. Pagelets can also specify custom content areas to allow for special grouping and positioning within pages.

Base templates and content areas can be customized via 2 settings: PAGELET_BASE_TEMPLATES and PAGELET_CONTENT_AREAS. For example, if you'd like to add an alternative 2-column layout, you could define the settings like so:

```
PAGELET_BASE_TEMPLATES = (
    ('pagelets/two_column_page.html', 'Two Column'),
)

PAGELET_CONTENT_AREAS = (
    ('main', 'Main'),
    ('sidebar', 'Sidebar'),
)
```

The page admin will now include an additional form field to select a base template and pagelets will allow the specification of content areas. The *Two Column* template could look something like this:

```
{% extends "base.html" %}

{% load pagelet_tags %}

{% block title %}{{ page.title }}{% endblock %}

{% block content %}
    <div id="main-panel">
        {% render_content_area page 'main' %}
    </div>
    <div id="sidebar-panel">
        {% render_content_area page 'sidebar' %}
    </div>
{% endblock %}
```

Note the `render_content_area` template tags with `main` and `sidebar` specified.

4.3 Built-in Template Tags

4.3.1 Render Pagelet

```
{% render_pagelet pagelet %}
```

This takes in either the slug of a pagelet or the pagelet object itself, then outputs the content of the pagelet along with some divs to wrap the content, and adds the administration links when a logged in user has permission to edit.

4.3.2 Create Page

```
{% create_page "Link Text" %}
```

Uses the first argument to create the text in the link and builds a simple link to create a new page based on the current URL.

This is meant to be used on 404s.

4.3.3 Render Content Area

```
{% render_content_area 'content_area_name' %}
```

This is used to create content areas on the PAGELET_BASE_TEMPLATES. For more information see, *Custom base templates and content areas*

4.3.4 Pagelink Ifexists

```
{% pagelink_ifexists pagelet %}
```

Creates a link to the input pagelet (or pagelet with slug) if it exists; otherwise, it renders nothing.

4.3.5 Page Content Teaser

```
{% page_content_teaser page number_of_words %}
```

Uses the content from the pagelets related to the page (or page with slug) and truncates the texts to the number_of_words.

4.3.6 Page Teaser

```
{% page_teaser page number_of_words %}
```

If the page has a description, then it is truncated to the number_of_words and returned, after removing the HTML. If there is no description, then the related pagelets are searched for <p> tags and the inner HTML is combined then truncated.

4.4 List of Available Settngs

4.4.1 PAGELET_TEMPLATE_TAGS

Default: *[]*

This setting also you to specify a list of template tags to include when rendering pagelet content. The names of the template tag libraries should be given as strings. *pagelet_tags* are defined and available when rendering pagelet content.

4.4.2 PAGELET_BASE_TEMPLATES

Default: *[]*

By default pages all render using the *pagelets/view_page.html* template. If you wish to define additional templates which pages can use, then you can define them here. This should be a list of tuples i.e.:

```
[("pagelets/two_col_page.html", "Two Column Template"), ]
```

4.4.3 PAGELET_CONTENT_AREAS

Default: *(('main', 'Main'),)*

Pagelets use content areas to define when they render on a given page. The default *pagelets/view_page.html* only uses one content area called *main*. If you define additional content areas in your page templates, then you need to add them into the choices using this setting.

4.4.4 PAGELET_CONTENT_DEFAULT

Default: *html*

This defines the default content type used when creating new pagelets. It can be either a built-in content type or a user defined content type as described above.

4.4.5 PAGE_ATTACHMENT_PATH

Default: *attachments/pages/*

This defines the location where page attachments will be stored relative to the media directory.

4.5 Release History

4.5.1 0.10.0 (Released 2014-11-19)

- Support for Django 1.7
- Moved to django-taggit dropping support for django-tagging
- Dropped South migrations

4.5.2 0.9.1 (Released 2014-11-03)

- Fixed Python 3 compatibility bug #59
- Removed test coverage for unsupported Python 2.6 and Django 1.3

4.5.3 0.9.0 (Released 2014-02-24)

- Setup tox
- Support Django's custom user model
- Python 3 support
- Flake8 improvements

4.5.4 0.8.0 (Released 2014-02-05)

- Fix migrations to run on MySQL
- Configurable content types with included JS/CSS requirements
- Updated message API to be compatible with Django 1.4+
- Updated template url syntax to be compatible with Django 1.5+
- Add missing CSRF token to attach form (see #38)
- Allow use of *{% render_content_area "by-slug-string" %}*
- Provide proper eTag and Last Modified checks to *edit_pagelet* view, avoiding edits that get lost by overzealous caching
- Add *truncate_html_words* shim for Django 1.6+ compatibility
- Fix urls.py imports to work with Django 1.6+

4.5.5 0.7.2 (Released 2012-03-28)

- Updated migration 0003 to be a data migration
- Made Page.tags field always exist, and add migration for it

4.5.6 0.7.1 (Released 2011-09-29)

- Add Read the Docs reference to README
- Update sample_project to work with Django 1.3
- Use CDN for external jQuery dependencies
- Add migration to install sequences properly on PostgreSQL

4.5.7 0.7.0 (Released 2011-04-01)

- Add docs and publish on Read the Docs
- Update media references to work better with staticfiles
- Make PageletBase an abstract base class
- Begin using South for migrations
- Update sample_project to use Django 1.2

4.5.8 0.6.2 (Released 2010-12-03)

- Remove use of .format() to support earlier versions of Python
- Fix license reference and URL endpoint in setup.py
- Include sample_project in MANIFEST.in
- Update license date

4.5.9 0.6.0 (Released 2010-09-12)

- First official release

Development sponsored by Caktus Consulting Group, LLC..

CHAPTER 5

Indices and tables

- genindex
- modindex
- search