django-tinycontent Documentation

Release 0.7.0

Dominic Rodger

Apr 19, 2017

Contents

django-tinycontent is a simple Django application for re-usable content blocks, much like django-boxes.

Blocks are configured via the Django admin, and are available to use in templates:

```
{% load tinycontent_tags %}

{% tinycontent_simple 'welcome' %}
```

That template fragment will look for a content block called `welcome`, and display the content of it.

Optionally, you can post-process the output with *Filters*.

CHAPTER 1

Contents

Installation

Installation is simple:

```
pip install django-tinycontent
```

Then, add `tinycontent` to your `INSTALLED_APPS`.

Version Support

Python 2.7, 3.4 and 3.5 are supported. Django versions from 1.5 upwards are supported with Python 2.7 and 3.4. Django versions from 1.8 upwards are supported with Python 3.5.

Using Blocks in Templates

Usage in templates is simple - to show the content of a block called `content_name` you can just use the template tag `tinycontent_simple`:

```
{% load tinycontent_tags %}

{% tinycontent_simple 'content_name' %}
```

Or, to specify a value if a content block by the given name cannot be found, use the `tinycontent` tag:

```
{% load tinycontent_tags %}

{% tinycontent 'content_name' %}
This will be shown if no matching object is found.
{% endtinycontent %}
```

The name of the content block can also be a context variable, using both the simple and the complex variants.

Optionally, you can post-process the output with *Filters*.

Passing Multiple Arguments

New in version 0.5.

You can pass multiple arguments to the django-tinycontent template tags, like this:

```
{% load tinycontent_tags %}

{% tinycontent_simple 'content_name' 'extra' %}
```

Extra arguments are concatenated together before looking up the content block - the above example will look for a content block called `content_name:extra`.

The main use case for this is internationalisation - each argument can either be a string literal (as in our example above), or a context variable. For example - to include the language code as part of your block name, you could use:

```
{% load tinycontent_tags %}

{% tinycontent_simple 'content_name' request.LANGUAGE_CODE %}
```

For those of us running websites in Great Britain, that would result in fetching the content block `content_name:en-gb`.

This feature is available both for `tinycontent_simple`, and `tinycontent`.

Adding and Editing Blocks

Content blocks themselves can be added and edited using Django's admin interface. If a block with the name given in the template tag cannot be found, either nothing is rendered (if using `tinycontent_simple`), or the text between `tinycontent` and `endtinycontent` is rendered (if using the more complex variant).

If you're logged in as a user with permission to add or edit content blocks (which you can set via permissions in the Django admin), you'll see links to the admin page for adding blocks (if there's no block set up yet) or editing (if the content block already exists).

Filters

By default, no transformations are applied to the content blocks - they're just displayed as they were entered in the admin. Since you probably want to display HTML, you'll probably want to set up a filter to apply before displaying content blocks, such as Markdown.

- *File-upload Handler*

Specifying Filters

You can configure what filter is applied using the setting `TINYCONTENT_FILTER`, which should be set to a dotted path to a function to call to filter the content (for example, to convert Markdown to HTML).

Warning: If the given path is invalid, any use of tinycontent tags will raise `ImproperlyConfigured`. If this setting is not provided, the content will be returned exactly as stored.

For example, if your project has a file called `utils.py`, you might have a function in it called `tinycontent_transform` that would look something like this:

```
def tinycontent_transform(content):
    return do_something_to(content)
```

To get the tinycontent templates to use that function, in your `settings.py` file, you'd write something like:

```
TINYCONTENT_FILTER = 'myproj.utils.tinycontent_transform'
```

Chaining Filters

You can optionally set `TINYCONTENT_FILTER` to a list of dotted paths - filters will be applied in the order in which you provide them.

For example, to use *Markdown* with tinycontent's *built-in file support*, you could set `TINYCONTENT_FILTER` like this:

```
TINYCONTENT_FILTER = [
    'tinycontent.filters.md.markdown_filter',
    'tinycontent.filters.builtin.uploaded_file_filter',
]
```

Built-in Filters

Markdown

django-tinycontent ships with a filter for Markdown. You can enable this by setting `TINYCONTENT_FILTER` like this:

```
TINYCONTENT_FILTER = 'tinycontent.filters.md.markdown_filter'
```

File-upload Handler

The file-upload filter replaces instances of `@file:slug` (where `slug` is the slug of a TinyContentFileUpload) with the URL to the file.

You can enable this filter by setting `TINYCONTENT_FILTER` like this:

```
TINYCONTENT_FILTER = 'tinycontent.filters.builtin.uploaded_file_filter'
```

Release Notes

v0.7.0

- Compatibility changes for Django 1.11 - dropped support for versions of Django earlier than 1.8, and Python 3.4 (Python 2.7 and Python 3.5 are still supported).

v0.6.1

- Modify cache name, to prevent warnings for non-ASCII characters or whitespace (thanks @ad-m).

v0.6.0

- Compatibility changes for Django 1.10.

v0.5.1

- Added a Polish translation and locale (thanks @ad-m).

v0.5.0

- Add support for multiple arguments to both the `tinycontent` and the `tinycontent_simple` template tags. See the documentation about *Passing Multiple Arguments*.
- Start caching database queries - fetching a TinyContent block by name (as the template tags do), will only hit the database the first time that content block is loaded (unless the content block is changed).

v0.4.0

- Require at least django-autoslug 1.8.0, to fix a warning about unapplied migrations.

v0.3.0

- Drop support for Django 1.4 (it's quite hard to support Django 1.4 and 1.9 in a single release - since Django 1.4 requires `{% load url from future %}`, and Django 1.9 doesn't support it).
- Ensure the wheel we upload to PyPI is universal.
- Forward compatibility for Django 1.9 - remove the `{% load url from future %}` from tinycontent templates.

v0.2.1

- Forwards compatibility change for Django 1.9 - which will remove the version of `importlib` bundled with Django. All supported versions of Python (2.7, 3.3 and 3.4) have `importlib`.

v0.2.0

- Dropped support for Python 2.6.
- Added a built-in markdown filter - you can use it by setting `TINYCONTENT_FILTER` to `'tinycontent.filters.md.markdown_filter'`.
- Added the ability to include links to files which you can upload via the admin.
- Added support for setting `TINYCONTENT_FILTER` to a list of dotted paths, to allow chaining filters.

v0.1.8

- Added the `TINYCONTENT_FILTER` setting for controlling the way content is output.
- Improved testing with Travis (we now test all supported Python versions and Django versions).