django-terms Documentation

Release 1.2.0

Bertrand Bordage

February 28, 2016

Contents

Site-wide adds a definition or a link for specialized terms.

Source:

https://github.com/BertrandBordage/django-terms

Code statistics: PyPI status: Continuous integration status:

CHAPTER 1

Live example

https://criminocorpus.org/musee/282/

(To learn more about the connection between django-terms and Criminocorpus, read the *Bounty* paragraph)

CHAPTER 2

Table of contents

Contents:

2.1 Requirements

2.1.1 Mandatory

- Python 2.7, 3.3, 3.4, or 3.5
- Django 1.7, 1.8, or 1.9
- lxml

2.1.2 Optional

- django-tinymce (tested with 1.5.1b2) to type the definition in a beautiful GUI (see *TERMS_DEFINITION_WIDGET*)
- django-ckeditor (tested with 3.6.2.1) to type the definition in another beautiful GUI (see *TERMS_DEFINITION_WIDGET*)
- django-reversion (tested with 1.6.0) to recover changes and deletions
- django-CMS (tested with 2.3) because django-terms has an apphook, a menu, a plugin processor and a plugin
- django-haystack (tested with 1.2.7 and 2.1.0) because django-terms has a search index
- django.contrib.sitemaps because django-terms has a sitemap
- South (tested with 0.8.1) because django-terms has migrations

2.2 Installation

1. `[sudo] pip install django-terms`
2. Add `'terms',` to your `INSTALLED_APPS`
3. `./manage.py syncdb` (`./manage.py migrate terms` if you use South)
4. Add terms to your urls:
    - add `url(r'^terms/', include('terms.urls')),` to your *urls.py*

- or, if you are using django-CMS, add a page and use the apphook and menu

2.3 Usage

1. Add some terms in the admin
2. Choose how django-terms should apply to your website:
   - *Middleware* (to give django-terms a try or for development)
   - *Template filter* (for production)
   - *With django-CMS*

The added terms should now be automatically linked to their definitions.

2.3.1 Middleware

A middleware is available to automatically add links on all your website. It is not recommended to use it in production because it parses and rebuilds whole pages, which can be an overkill in most cases (even though django-terms has excellent performances).

It is also perfect for development: it never fails silently, unlike filters (see *Exceptions* for more details).

1. Add `'terms.middleware.TermsMiddleware',` to your `MIDDLEWARE_CLASSES`
2. If the middleware applies to unwanted Django applications, HTML tags, classes, or IDs, set the corresponding *Common settings*

2.3.2 Template filter

A template filter is available to add links only on desired parts of your website.

1. Choose one of your existing templates
2. Add `{% load terms %}` to the beginning of the file (just after `{% extends '[file]' %}` if you have one)
3. Use the filter `replace_terms` like every normal filter
4. If the filter applies to unwanted HTML tags, classes, or IDs, set the corresponding *:ref:Common settings*

Example:

Suppose you have such a template:

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

{% block article_header %}
  {{ article.header }}
{% endblock %}

{% block article_content %}
  {{ article.section1 }}
  {{ article.section2 }}
{% endblock %}
```

Here is how you can modify it:

```
{% extends 'base.html' %}
{% load terms %}

{% block article_header %}
  {{ article.header|replace_terms }}
{% endblock %}

{% block article_content %}
  {% filter replace_terms %}
    {{ article.section1 }}
    {{ article.section2 }}
  {% endfilter %}
{% endblock %}
```

Now, suppose you have an HTML class `code-snippet` in `article.section2` where you do not want to add links on terms. Go to *Common settings*, and you will find the solution:

Add this line in *settings.py*:

```
TERMS_ADDITIONAL_IGNORED_CLASSES = ['code-snippet']
```

2.3.3 With django-CMS

A few tools are available to make your life easier if you use django-CMS.

Plugin processor

It will automatically apply the *Template filter* on every plugin.

To use it, add or modify `CMS_PLUGIN_PROCESSORS` in *settings.py*:

```
CMS_PLUGIN_PROCESSORS = (
    ...
    'terms.cms_plugin_processors.TermsProcessor',
    ...
)
```

Glossary plugin

This plugin displays all terms and their definitions.

Don't forget to update `CMS_PLACEHOLDER_CONF` in your *settings.py* if you defined it, otherwise this plugin will not be available from your placeholders.

Apart from this, nothing to do to make it work.

App hook and menu

You can use the the app hook and the menu to integrate the complete glossary to your CMS architecture.

Nothing to do to make it work.

2.4 Settings

2.4.1 Common settings

TERMS_ENABLED

Default `True`

Definition If set to `False`, globally disabled django-terms.

TERMS_DEBUG

Default `DEBUG`

Definition If set to `True`, allows django-terms to raise minor exceptions (see *Exceptions*).

TERMS_REPLACE_FIRST_ONLY

Default `True`

Definition If set to `True`, adds a link only on the first occurrence of each term

Used by *Middleware*, *Template filter*

TERMS_ADDITIONAL_IGNORED_APPS

Default `()`

Definition A list or tuple of ignored Django applications (expressed as strings)

Used by *Middleware*

Extends *TERMS_IGNORED_APPS*

Syntax example `['cms']`

TERMS_ADDITIONAL_IGNORED_TAGS

Default `()`

Definition A list or tuple of ignored HTML tags (expressed as strings)

Used by *Middleware*, *Template filter*

Extends *TERMS_IGNORED_TAGS*

Syntax example `['h1', 'h2', 'h3', 'footer']`

TERMS_ADDITIONAL_IGNORED_CLASSES

Default `()`

Definition A list or tuple of ignored HTML classes (expressed as strings)

Used by *Middleware*, *Template filter*

Extends *TERMS_IGNORED_CLASSES*

Syntax example `['footnote', 'text-caption']`

TERMS_ADDITIONAL_IGNORED_IDS

Default `()`

Definition A list or tuple of ignored HTML IDs (expressed as strings)

Used by *Middleware*, *Template filter*

Extends *TERMS_IGNORED_IDS*

Syntax example `['article-footer', 'side-content']`

TERMS_DEFINITION_WIDGET

Default `'auto'`

Definition Explicitly tells django-terms which text widget to choose for the definition of a term. Accepted values are `'auto'`, `'basic'`, `'tinymce'`, and `'ckeditor'`.

2.4.2 Advanced settings

These settings should not be used, unless you know perfectly what you are doing.

TERMS_IGNORED_APPS

Default see *terms/settings.py*

Definition A list or tuple of ignored Django applications (expressed as strings)

Used by *Middleware*

TERMS_IGNORED_TAGS

Default see *terms/settings.py*

Definition A list or tuple of ignored HTML tags (expressed as strings)

Used by *Middleware*, *Template filter*

TERMS_IGNORED_CLASSES

Default see *terms/settings.py*

Definition A list or tuple of ignored HTML classes (expressed as strings)

Used by *Middleware*, *Template filter*

TERMS_IGNORED_IDS

Default see *terms/settings.py*

Definition A list or tuple of ignored HTML IDs (expressed as strings)

Used by *Middleware*, *Template filter*

2.5 Troubleshooting

2.5.1 Side effects

Why?

When using django-terms, your HTML pages are totally or partially reconstructed:

- totally reconstructed if you use the *Middleware*
- partially reconstructed if you use the *Template filter* or *With django-CMS*

The content is parsed and rebuilt with lxml. See `terms/html.py` to understand exactly how.

List of known side effects

A few side effects are therefore happening during HTML reconstruction:

- Entity names and numbers (e.g. `&eacute;`, `&#233;`, ...) are unescaped. This means they are replaced with their unicode characters (e.g. `&eacute;` -> é)
- Additional spaces inside HTML tags are stripped:
  - Start tags `<a href = "url" >` -> `<a href="url">`
  - End tags `</ a >` -> `</a>`
  - "Start-end" tags `<input style = "text" />` -> `<input style="text"/>`

Warning: This implies one bad side effect: the unescaping may break the special characters rendering in some complex form fields like django-ckeditor. *django.contrib.admin* is already ignored, so you should not encounter any problem. Otherwise, using filters instead of the middleware and/or ignore the correct apps/tags/classes/ids using *Common settings* will ensure a proper rendering.

2.5.2 Performance

Good news

django-terms nearly never hits the database. After each change in your terms table, the database is hit just one time in order to build a regular expression that's saved into your cache (assuming you set up the cache). If you never change your terms and if your cache is never emptied, there will be zero database hit.

Considering memory, no particular leak has been found.

Bad news

Unfortunately, django-terms has a significant impact on speed, especially if you use the *Middleware*. That's why we recommend using the *Template filter*.

What is important is the number of HTML tags wrapped by the filter or the middleware. Then comes the complexity of your HTML tree. The amount of flat text, luckily, has no impact.

To give you an idea, *terms/tests/terms/performance_test_before.html* contains 263 tags and takes 11 ms to be parsed and rebuilt on my computer with the middleware. That gives an average of 40 µs per tag. If you use the template tag

only on the content of the page (124 tags), it takes 9 ms. Quite slow, but if you cache the part of the template that's filtered, this issue should be negligible.

2.5.3 Exceptions

Resolver404

Raised by *Middleware* only.

Raised in *TERMS_DEBUG* mode. Otherwise the page is ignored by django-terms.

Reason This happens when django-terms is unable to resolve the current `request.path` to determine whether the application of the current page is in TERMS_IGNORED_APPS.

Encountered In django-CMS 2.3, when adding a plugin in frontend editing.

2.6 Translations

2.6.1 Status

2.6.2 Write your translation

Localization is done directly on our Transifex page. There is no access restriction, so feel free to spend two minutes translating django-terms to your language :o)

2.6.3 Get & Compile

1. Make sure you have transifex-client installed: `[sudo] pip install transifex-client`
2. Pull all translations from Transifex: `tx pull -a`
3. Compile them: `cd terms && django-admin.py compilemessages`

2.7 Bounty & donations

2.7.1 Bounty

django-terms was originally developed for a french website, Criminocorpus. Now that it is finished, I don't have credits to continue development on django-terms.

Maybe you want a new feature in django-terms but don't want to code anything. First, open an issue on github so that we can publicly decide whether it's a desirable feature. Then send me an email to privately discuss the cost of such a demand.

2.7.2 No donations...

I don't accept donations for the time being.

Others would ask for donations in such cases, but in my opinion that would be quite dishonest because:

- I've been paid to develop something that became django-terms, even though I was not supposed to make it open source, nor to make docs and tests.
- I'm not a street mime, I can't develop properly on the basis of a few bucks donated from time to time.
- I don't believe in Santa Claus.

If you're still willing to help with money, see the next paragraph.

2.7.3 . . . but you can still help

The obvious help is to contribute to the code.

If you're really willing to help but you can't code, you can send me an email to say that you like django-terms, that's always heartwarming :)

If you want to help me with money, please donate to PyPy (especially STM/AME). I'm not part of it, but that's an awesome project that probably is the future of Python. It's Python, but way faster. In a few years, maybe PyPy will become the main Python implementation, as was attempted with Unladen Swallow. Then django-terms will probably be much faster.

2.8 Getting help

If you have a question or need help, ask on our Google group.

If you spotted a bug or if you want to propose a new feature, open a new github issue.