Django Addendum Documentation

Release 0.3.4

Ben Lopatin

October 29, 2015

Contents

1 Installation **3**
1.1 Upgrading . . . 3

2 Basic Usage **5**

3 Richtext snippets **7**

4 Translations **9**
4.1 Setup . . . 9
4.2 Models . . . 9
4.3 Cache . . . 10
4.4 Templates . . . 11

5 Caching **13**

6 Management commands **15**

7 License **17**

8 Indices and tables **19**

Change snippets of copy on your site, on the fly, for any application, and without a full-fledged CMS.

Solving queries like:

> Hey, we need to change the greeting on login from "Hi!" to "Sup?"

And:

> The footer copy needs to be udpated.

And:

> The marketing team would really like to be able to change that message on a monthly basis. I don't care that that's a third-party appliwhoozitz!

This is all simple stuff and it's probably coded right into your templates. Changing it is easy enough, but requires a developer and then a release. Boo!

CHAPTER 1

Installation

Install the package from PyPI:

```
pip install django-addendum
```

Add it to your *INSTALLED_APPS* tuple:

```
INSTALLED_APPS += ('addendum')
```

Sync your database or migrate if you have South installed.

Note: Django Addendum is not compatible with South versions prior to 1.0.

1.1 Upgrading

If you are upgrading from a version prior to 0.3.0 you will need to flush your cache, as the cached values have change. You can reload your cache layer or simply run the refresh management command immediately after updating your project.:

```
python manage.py refresh_snippet_cache
```

Older versions of Django Addendum stored the entire snippet object in the cache; from 0.3.0 onwards a dictionary of snippet values (including translations) are stored against the snippet cache key.

CHAPTER 2

Basic Usage

Just add *addendum_tags* to your templates and reference a snippet by name.

```
{% load addendum_tags %}

{% snippet 'home:greeting' %}Hi!{% endsnippet %} {{ user.first_name }}

<footer>
  {% snippet 'home:footer' %}&copy; 2011 by Acme Corp.{% endsnippet %}
</footer>
```

You can use template variables for snippet names, too.

```
{% snippet greeting %}Hi!{% endsnippet %} {{ user.first_name }}
```

Now you can edit content for these placeholders from the admin interface. If you don't add anything or you delete text, the site text will always revert to what is in the template.

Use it for small bits of user modifiable text from any template on your site, and for swapping out *lorem ipsum* text when prototyping.

CHAPTER 3

Richtext snippets

Text from snippets is escaped by default. If you want to render non-escaped HTML, use the *safe* keyword argument:

```
{% load addendum_tags %}

{% snippet 'greeting' safe=True %}Bienvenidos!{% endsnippet %}
```

Plaintext by default is a good way of keeping site editors from getting carried away by adding differently formatted content.

Note: Previous versions used the *richtext* keyword. This will work as well, but is deprecated.

CHAPTER 4

Translations

Django Addendum version 0.3.0 introduced multilingual support.

4.1 Setup

If you plan on using snippet translations, presumably you already have internationalization enabled in your project.:

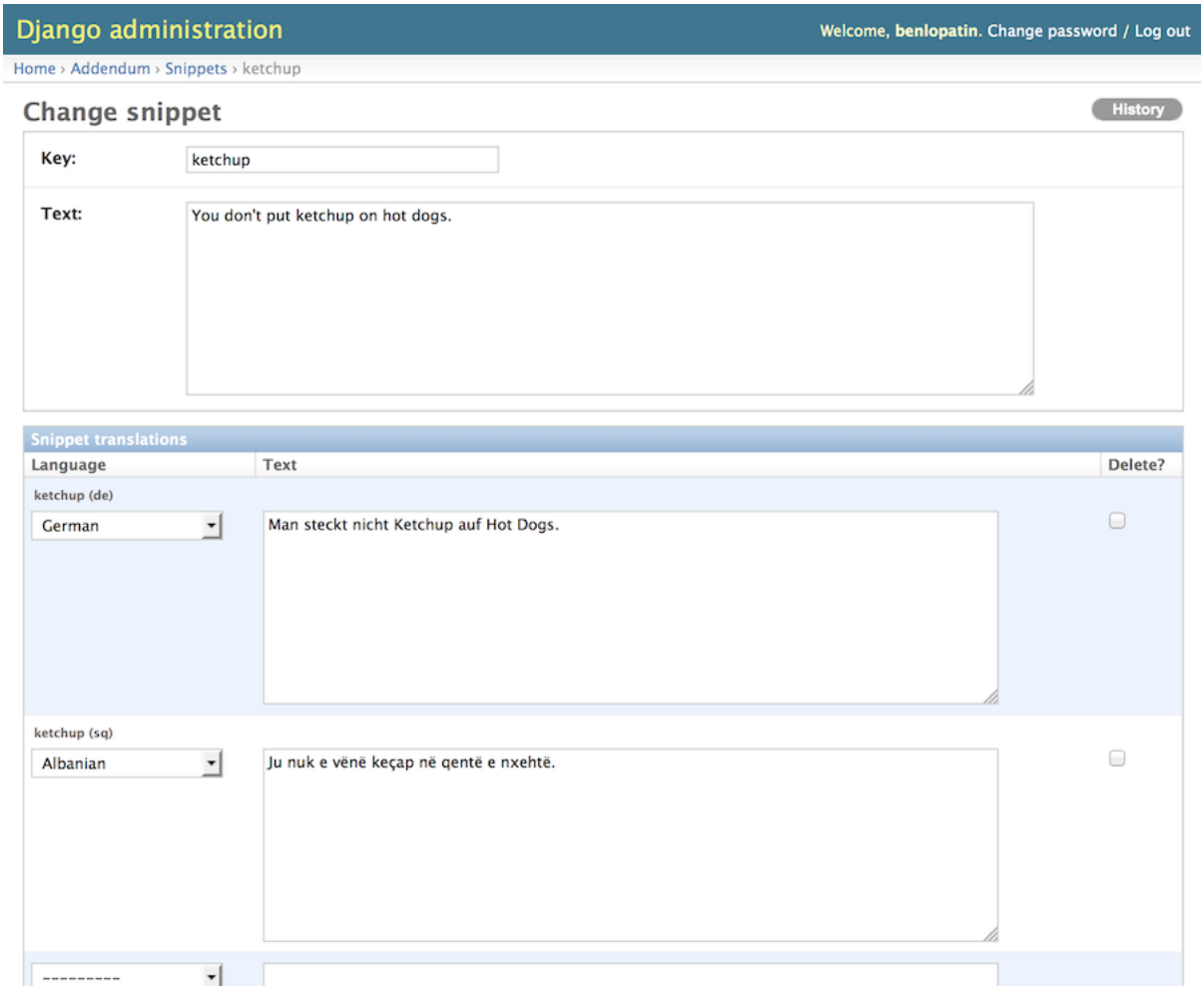
```
USE_I18N = True
```

Note: If you do not have internationalization enabled in your Django project you will not notice any changes.

4.2 Models

A new *SnippetTranslation* model will add a table to your database whether you use internationalization support or not.

If you do not have *i18n* enabled in your project, you won't notice any other changes other than this new table. If you do have *i18n* support enabled, each Snippet will have inline SnippetTranslation instances availabe in the admin.



The available language options are provided by your *LANGUAGES* settings tuple.

4.3 Cache

Translations are stored with the original snippet in the cache store for the given key.:

```
{
  "": "Original snippet",
  "es": "Fragmento original",
}
```

The interface for updating and retrieving these values is provided by the *set_cached_snippet* and *get_cached_snippet* functions.

The *set_cached_snippet* function takes a snippet key and updates the complete cached dictionary from the snippet and its translations. The *get_cached_snippet* function takes a key and optional language code and returns the text for the snippet by translation, if available, otherwise it returns the default snippet (or *None* if none is found).

4.4 Templates

Provided you are using the *i18n* context processor and a RequestContext, then the language will be accessible from the context without any changes.:

```
{% load addendum_tags %}
{% snippet 'greeting' %}Welcome!{% endsnippet %}
```

If you've added a translation for Spanish and your users are accessing your content with 'es' in their requests:

Bienvenidos

You can also specify the language via a variable of your choosing:

```
{% load addendum_tags %}
{% snippet 'greeting' language=language_template_var %}Bienvenidos!{% endsnippet %}
```

Or a string:

```
{% load addendum_tags %}
{% snippet 'greeting' language='es-mx' %}Bienvenidos!{% endsnippet %}
```

The specified language string will probably be less helpful for most use cases.

Note: The language code checks are strict. If you want to support alternate dialects, e.g. American, Australian, British English, then you will need to provide translations for each. The lookup will not fall back on the parent language.

CHAPTER 5

Caching

Every template tag instance represents a database lookup, so snippets are cached by default, *as are missing keys*.

Snippets are cached indefinitely in the default cache using a key of this format: *snippet:<snippet_key>*. If a key is absent in both the cache and the database, the key is stored with the integer value *-1*. This ensures that an obvious and non *NoneType* value is stored to prevent subsequent database lookups. Cache return values of *-1* are treated immediately as absent keys and the base text in the template is rendered directly.

CHAPTER 6

Management commands

The *refresh_snippet_cache* command will cycle through all snippets and update the cached value.

CHAPTER 7

License

BSD licensed.

CHAPTER 8

Indices and tables

- genindex
- search