
django-statici18n Documentation

Release 0.9

Sebastien Fievet

May 17, 2014

1	Overview	3
2	Installation	5
3	Contents	7
3.1	Management commands	7
3.2	Settings	7
3.3	Changelog	8

A Django app that provides helper for compiling underscore templates to static files with i18 support.

Overview

This repo an project is forked from `django-static-i18n` github.com/zyegfryed/django-static-i18n to tackle the problem of compiling of Underscore templates to single static js file. The original code was generating static js files for translations.

This app is intended to make life easier when you want to work with your Underscore templates and translate them with Django default `i18n` module (no `jsgettext`). If you are using Underscore templates you can have project directory like:

```
<project_directory>/
...
locale/
+- en/
+- fr/
templates/
+- underscore/
  |
  +- popup.html
+- modals/
+- include/
+- main.html
```

and your `popup.html` can look like:

```
<div>
  {% trans "Hello" %} <% username %>
</div>
```

and you want to render it in something like Backbone:

```
PopupView = Backbone.View.extend({
  template: _.template(popup_variable_name),
});
```

to do this you need to compile your `.html` template to be available in js with `popup_variable_name`. Moreover if you have multiple templates, you can bundle them into single js file and serve it via CDN or nginx omitting django.

With `django-static-underscore-i18n` you can do this by following. Declare dictionary mapping between html files and js variable names:

```
STATIC_UNDERSCORE_TEMPLATES = {'popup_variable_name': 'underscore/popup.html', ... , }
```

and run `python manage.py compilejsunderscoringi18n` which will bundle your html templates into one js file for each locale supporting i18 `{% trans %}` tags.

Installation

1. Use your favorite Python packaging tool to install `django-staticunderscore-i18n` from [PyPI](#), e.g.:

```
pip install django-static-underscore-i18n
```

2. Add `'staticunderscorei18n'` to your `INSTALLED_APPS` setting:

```
INSTALLED_APPS = [
    # ...
    'staticunderscorei18n',
]
```

3. Once you have edited your underscore .html templates or [translated](#) and [compiled](#) your messages, use the `compilejsunderscoringi18n` management command:

```
python manage.py compilejsunderscoringi18n
```

4. Check that the `django.core.context_processors.i18n` context processor is added to your `TEMPLATE_CONTEXT_PROCESSORS` setting - should have already been set by Django. This is needed to resolve request.LANGUAGE_CODE variable during compilation phase:

```
TEMPLATE_CONTEXT_PROCESSORS = (
    # ...
    'django.core.context_processors.i18n',
)
```

and you should have `FileSystemFinder` and `AppDirectoriesFinder` to be available:

```
STATICFILES_FINDERS = (
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
)
```

5. Edit your template(s) and insert .js files those were compiled. Good practice is to serve static files without django server (you can you nginx for that):

```
<script src="{{ STATIC_URL }}jsunderscoringi18n/{{ LANGUAGE_CODE }}/underscore_templates.js"></script>
```

Note: By default, the generated catalogs are stored to `STATIC_ROOT/jsunderscoringi18n`. You can modify the output path and more options by tweaking `django-staticunderscorei18n` settings.

3.1 Management commands

3.1.1 `compileunderscorejsi18n`

Compile JavaScript Underscore templates files from templates to js file in a single location supporting i18n.

Some commonly used options are:

- l LOCALE or --locale=LOCALE** The locale to process. Default is to process all.
- o OUTPUT_DIR or --output=OUTPUT_DIR** Output directory to store generated catalogs. Defaults to the joining path of `STATIC_UNDERSCORE_I18N_ROOT` and `STATIC_UNDERSCORE_I18N_OUTPUT_DIR`.

For a full list of options, refer to the `compilejsi18n` management command help by running:

```
$ python manage.py compilejsunderscoiei18n --help
```

Note: Missing directories will be created on-the-fly by the command when invoked.

3.2 Settings

`django.conf.settings.STATICI_UNDERSCORE_18N_PACKAGES`

Default ('django.conf')

A list of packages to check for translations.

Can be overridden with the `-p/--package` option of `compileunderscorejsi18n` command.

Each string in packages should be in Python dotted-package syntax (the same format as the strings in `INSTALLED_APPS`) and should refer to a package that contains a locale directory. If you specify multiple packages, all those catalogs are merged into one catalog. This is useful if you have JavaScript that uses strings from different applications.

`django.conf.settings.STATIC_UNDERSCORE_TEMPLATES_DOMAIN`

Default underscore_templates

filename of generated static js file.

`django.conf.settings.STATIC_UNDERSCORE_TEMPLATES`

Default {}

Dictionary of variables/template name bindings. Example: .. code-block:: django {

STATIC_UNDERSCORE_TEMPLATES = { 'popup_template': '_templates/popup.html',

} .. note:

django.conf.settings.STATIC_UNDERSCORE_I18N_ROOT

Default STATIC_ROOT

Controls the file path that catalog files will be written into.

django.conf.settings.STATICI_UNDERSCORE_I18N_OUTPUT_DIR

Default 'underscore_templates'

Controls the directory inside `STATIC_UNDERSCORE_I18N_ROOT` that generated files will be written into.

django.conf.settings.STATIC_UNDERSCORE_I18N_FILENAME_FUNCTION

Default 'staticunderscorei18n.utils.default_filename'

The dotted path to the function that creates the filename.

The function receives two parameters:

- locale: a string representation of the locale currently processed
- domain: a string representation of the gettext domain used to check for translations

By default, the function returns the path '`<language_code>/<domain>.js`'.

The final filename is resulted by joining `STATIC_UNDERSCORE_I18N_ROOT`, `STATIC_UNDERSCORE_I18N_OUTPUT_DIR` and `STATIC_UNDERSCORE_I18N_FILENAME_FUNCTION`.

For example, with default settings in place and `STATIC_ROOT = 'static'`, the JavaScript catalog generated for the `en_GB` locale is: `'static/jsunderscorei18n/en-gb/underscore_templates.js'`

3.3 Changelog

3.3.1 v1.9 (2014 May 17)

- Successful Travis build, final version

3.3.2 v0.9 (2014 May 15)

- Initial commit.