django-static-precompilers Documentation

Release 1.1

Andrey Fedoseev

May 24, 2017

Contents

1 Installation **3**

2 `compile` template filter **5**
2.1 Example 5

3 {% inlinecompile %} tag **7**
3.1 Example 7

4 General settings **9**

5 Compiler specific settings **11**
5.1 CoffeeScript 11
5.2 Babel 11
5.3 LiveScript 12
5.4 Handlebars 12
5.5 SASS / SCSS 12
5.6 Libsass 13
5.7 LESS 14
5.8 Stylus 14

6 Usage with forms media **15**

7 `compilestatic` management command **17**

8 Troubleshooting **19**

Django Static Precompiler provides template tags and filters to compile CoffeeScript, SASS / SCSS, LESS, Stylus, Babel and Handlebars. It works with both inline code and external files.

CHAPTER 1

Installation

`django-static-precompiler` is avaiable through pip:

```
$ pip install django-static-precompiler
```

1. Add "static_precompiler" to INSTALLED_APPS setting.
2. Initialize DB:
   - On Django < 1.7 run `syncdb` or `migrate static_precompiler` if you use South (1.0 is required).
   - On Django >= 1.7 run `migrate static_precompiler`.
3. Make sure that you have necessary compilers installed.
4. Optionally, you can specify the full path to compilers (see below).
5. In case you use Django's staticfiles contrib app you have to add static-precompiler's file finder to the `STATICFILES_FINDERS` setting, for example:

```
STATICFILES_FINDERS = (
    'django.contrib.staticfiles.finders.FileSystemFinder',
    'django.contrib.staticfiles.finders.AppDirectoriesFinder',
    # other finders..
    'static_precompiler.finders.StaticPrecompilerFinder',
)
```

Note that by default compiled files are saved into `COMPILED` folder under your `STATIC_ROOT` (or `MEDIA_ROOT` if you have no `STATIC_ROOT` in your settings). You can change this folder with `STATIC_PRECOMPILER_ROOT` and `STATIC_PRECOMPILER_OUTPUT_DIR` settings.

Note that all relative URLs in your stylesheets are converted to absolute URLs using your `STATIC_URL` setting.

CHAPTER 2

`compile` template filter

`compile` is a template filter that allows to compile any source file supported by compilers configured with `STATIC_PRECOMPILER_COMPILERS` settings.

Example

```
{% load compile_static %}
{% load static %}

<script src="{% static "path/to/script.coffee"|compile %}"></script>
<link rel="stylesheet" href="{% static "path/to/styles1.less"|compile %}" />
<link rel="stylesheet" href="{% static "path/to/styles2.scss"|compile %}" />
```

renders to:

```
<script src="/static/COMPILED/path/to/script.js"></script>
<link rel="stylesheet" href="/static/COMPILED/path/to/styles1.css" />
<link rel="stylesheet" href="/static/COMPILED/path/to/styles2.css" />
```

CHAPTER 3

{% inlinecompile %} tag

Compiles everything between `{% inlinecompile %}` and `{% endinlinecompile %}` with compiler specified by name. Compiler must be specified in `STATIC_PRECOMPILER_COMPILERS` setting. Names for default compilers are:

- `coffeescript`
- `babel`
- `less`
- `sass`
- `scss`
- `stylus`

Example

```
{% load compile_static %}

<script type="text/javascript">
  {% inlinecompile "coffeescript" %}
    console.log "Hello, World!"
  {% endinlinecompile %}
</script>
```

renders to:

```
<script type="text/javascript">
  (function() {
    console.log("Hello, World!");
  }).call(this);
</script>
```

CHAPTER 4

General settings

STATIC_PRECOMPILER_COMPILERS List of enabled compilers. You can modify it to enable your custom compilers. Default:

```
STATIC_PRECOMPILER_COMPILERS = (
    'static_precompiler.compilers.CoffeeScript',
    'static_precompiler.compilers.Babel',
    'static_precompiler.compilers.Handlebars',
    'static_precompiler.compilers.SASS',
    'static_precompiler.compilers.SCSS',
    'static_precompiler.compilers.LESS',
    'static_precompiler.compilers.Stylus',
)
```

You can specify compiler options using the following format:

```
STATIC_PRECOMPILER_COMPILERS = (
    ('static_precompiler.compilers.CoffeeScript', {"executable": "/usr/bin/
↪coffeescript"}),
    ('static_precompiler.compilers.SCSS', {"compass_enabled": True}),
)
```

STATIC_PRECOMPILER_ROOT Controls the absolute file path that compiled files will be written to. Default: `STATIC_ROOT`.

STATIC_PRECOMPILER_OUTPUT_DIR Controls the directory inside `STATIC_PRECOMPILER_ROOT` that compiled files will be written to. Default: `"COMPILED"`.

STATIC_PRECOMPILER_USE_CACHE Whether to use cache for inline compilation. Default: `True`.

STATIC_PRECOMPILER_CACHE_TIMEOUT Cache timeout for inline styles (in seconds). Default: 30 days.

STATIC_PRECOMPILER_MTIME_DELAY Cache timeout for reading the modification time of source files (in seconds). Default: 10 seconds.

STATIC_PRECOMPILER_CACHE_NAME Name of the cache to be used. If not specified then the default django cache is used. Default: `None`.

STATIC_PRECOMPILER_PREPEND_STATIC_URL Add `STATIC_URL` to the output of template tags and filters. Default: `False`.

STATIC_PRECOMPILER_DISABLE_AUTO_COMPILE Disable automatic compilation from template tags or `compile_static` utility function. Files are compiled only with `compilestatic` command (see below). Default: `False`.

STATIC_PRECOMPILER_FINDER_LIST_FILES Whether or not `static_precompiler.finders.StaticPrecompilerFinder` will list compiled files when `collectstatic` command is executed. Set to `True` if you want compiled files to be found by `collectstatic`. Default: `False`.

CHAPTER 5

Compiler specific settings

CoffeeScript

`executable` Path to CoffeeScript compiler executable. Default: `"coffee"`.

`sourcemap_enabled` Boolean. Set to `True` to enable source maps. Default: `False`.

Example:

```
STATIC_PRECOMPILER_COMPILERS = (
    ('static_precompiler.compilers.CoffeeScript', {
        "executable": "/usr/bin/coffee",
        "sourcemap_enabled": True,
    }),
)
```

Babel

`executable` Path to Babel compiler executable. Default: `"babel"`.

`sourcemap_enabled` Boolean. Set to `True` to enable source maps. Default: `False`.

`plugins` Babel plugins command line option. Default: `None` (uses Babel's default option).

`presets` Babel presets command line option. Default: `None` (uses Babel's default option).

Example:

```
STATIC_PRECOMPILER_COMPILERS = (
    ('static_precompiler.compilers.Babel', {
        "executable": "/usr/bin/babel",
        "sourcemap_enabled": True,
        "plugins": "transform-react-jsx",
        "presets": "es2015,react",
```

```
    }),
)
```

LiveScript

executable Path to LiveScript compiler executable. Default: `"lsc"`.

sourcemap_enabled Boolean. Set to `True` to enable source maps. Default: `False`.

Example:

```
STATIC_PRECOMPILER_COMPILERS = (
    ('static_precompiler.compilers.LiveScript', {
        "executable": "/usr/bin/lsc",
        "sourcemap_enabled": True,
    }),
)
```

Handlebars

executable Path to Handlebars compiler executable. Default: `"handlebars"`.

sourcemap_enabled Boolean. Set to `True` to enable source maps. Default: `False`.

known_helpers List of known helpers (`-k` compiler option). Default: `None`.

namespace Template namespace (`-n` compiler option). Default: `None`.

simple Output template function only (`-s` compiler option). Default: `False`.

Example:

```
STATIC_PRECOMPILER_COMPILERS = (
    ('static_precompiler.compilers.Handlebars', {
        "executable": "/usr/bin/handlebars",
        "sourcemap_enabled": True,
        "simple": True,
    }),
)
```

SASS / SCSS

executable Path to SASS compiler executable. Default: “sass”.

sourcemap_enabled Boolean. Set to `True` to enable source maps. Default: `False`.

compass_enabled Boolean. Whether to use compass or not. Compass must be installed in your system. Run `sass --compass` and if no error is shown it means that compass is installed.

load_paths List of additional directories to look imported files (`--load-path` command line option). Default: `None`.

precision How many digits of precision to use when outputting decimal numbers. Default: `None`. Set this to 8 or more if you compile Bootstrap.

output_style Output style. Default: `None`. Can be nested, compact, compressed, or expanded.

Example:

```
STATIC_PRECOMPILER_COMPILERS = (
    ('static_precompiler.compilers.SCSS', {
        "executable": "/usr/bin/sass",
        "sourcemap_enabled": True,
        "compass_enabled": True,
        "load_paths": ["/path"],
        "precision": 8,
        "output_style": "compressed",
    }),
)
```

Libsass

Libsass is a C/C++ implementation of SASS. `django-static-precompiler` uses libsass-python bindings for `libsass`

To use SASS / SCSS compiler based on `libsass` install `django-static-precompiler` with `libsass` flavor:

```
pip install django-static-precompiler[libsass]
```

Note: Libsass compiler is disabled by default. See how to enable it in the example below.

Options:

sourcemap_enabled Boolean. Set to `True` to enable source maps. Default: `False`.

load_paths List of additional paths to find imports. Default: `None`.

precision How many digits of precision to use when outputting decimal numbers. Default: `None`. Set this to 8 or more if you compile Bootstrap.

output_style Output style. Default: `None`. Can be nested, compact, compressed, or expanded.

Example:

```
STATIC_PRECOMPILER_COMPILERS = (
    ('static_precompiler.compilers.libsass.SCSS', {
        "sourcemap_enabled": True,
        "load_paths": ["/path"],
        "precision": 8,
    }),
    ('static_precompiler.compilers.libsass.SASS', {
        "sourcemap_enabled": True,
        "load_paths": ["/path"],
        "precision": 8,
        "output_style": "compressed",
    }),
)
```

Note: Libsass compiler doesn't support Compass extension, but you can replace it with compass-mixins.

LESS

executable Path to LESS compiler executable. Default: `"lessc"`.

sourcemap_enabled Boolean. Set to `True` to enable source maps. Default: `False`.

include_path List of additional directories to look for imported files (`--include-path` command line option). Default: `None`.

global_vars Dictionary of global variables (`--global-var` command line option). Default: `None`.

Example:

```
STATIC_PRECOMPILER_COMPILERS = (
    ('static_precompiler.compilers.LESS', {
        "executable": "/usr/bin/lessc",
        "sourcemap_enabled": True,
        "global_vars": {"link-color": "red"},
    }),
)
```

Stylus

executable Path to Stylus compiler executable. Default: `"stylus"`.

sourcemap_enabled Boolean. Set to `True` to enable source maps. Default: `False`.

Example:

```
STATIC_PRECOMPILER_COMPILERS = (
    ('static_precompiler.compilers.Stylus', {"executable": "/usr/bin/stylus",
↪"sourcemap_enabled": True),
)
```

CHAPTER 6

Usage with forms media

If you want to use `static_precompiler` in form media definitions, you can use the following approach:

```
from django import forms
from static_precompiler.utils import compile_static

class MyForm(forms.Form):

    @property
    def media(self):
        return forms.Media(
            css={"all": (
                compile_static("styles/myform.scss"),
            )},
            js=(
                compile_static("scripts/myform.coffee"),
            )
        )
```

CHAPTER 7

`compilestatic` management command

Django Static Precompiler includes a management command `compilestatic`. It will scan your static files for source files and compile all of them.

You can use this command in conjunction with `STATIC_PRECOMPILER_DISABLE_AUTO_COMPILE` setting if you use custom `STATICFILES_STORAGE` such as S3 or some CDN. In that case you can should run `compilestatic` every time when your source files change and then run `collectstatic`.

You can run `compilestatic` in watch mode (`--watch` option). In watch mode it will monitor the changes in your source files and re-compile them on the fly. It can be handy if you use tools such as LiveReload.

You should install Watchdog to use watch mode or install `django-static-precompiler` with the `watch` extra:

```
$ pip install django-static-precompiler[watch]
```

CHAPTER 8

Troubleshooting

If you get `[Errno 2] No such file or directory` make sure that you have the required compiler installed. For all compilers you can specify the path to executable file using the `executable` option, see examples above.

If you run `migrate` and get `ImportError: cannot import name migrations` then most likely you use Django < 1.7 and South < 1.0. You should either upgrade to Django 1.7+ or use South 1.0.