
Markdownify Documentation

Release 1.0.0

Erwin Matijssen

Jun 13, 2022

Table of contents

1	Requirements	1
2	Installation	3
3	Usage	5
4	Settings	7
5	Tests	13
6	Django Markdownify - A Django Markdown filter	15

CHAPTER 1

Requirements

Django Markdownify requires [Django](#) (obviously), as well as [Markdown](#) and [Bleach](#) version 5 or higher. When installing Django Markdownify, dependencies will be installed automatically.

CHAPTER 2

Installation

Install Django Markdownify with pip:

```
pip install django-markdownify
```

Or add `django-markdownify` to your `requirements.txt` and run `pip install -r requirements.txt`

Finally add `markdownify` to your installed apps in `settings.py`:

```
INSTALLED_APPS = [  
    ...  
    'markdownify.apps.MarkdownifyConfig',  
]
```


CHAPTER 3

Usage

Load the tag in your template:

```
{% load markdownify %}
```

Then you can change markdown to html as follows:

```
{{ 'text'|markdownify }}
```

Use Markdown in your template directly:

```
{% load markdownify %}
{{ 'Some *test* [link] (#)'|markdownify }}
```

Or use the filter on a variable passed to the template via your views. For example:

```
# views.py
class Markdown(TemplateView):
    template_name = 'index.html'

    def get_context_data(self, **kwargs):
        markdowntext = open(os.path.join(os.path.dirname(__file__), 'templates/test.md
→')).read()

        context = super().get_context_data(**kwargs)
        context['markdowntext'] = markdowntext

        return context

# index.html
{% load markdownify %}
{{ markdowntext|markdownify }}
```

You probably want to add some extra allowed tags and attributes in the *Settings*, because the defaults are rather sparse.

It is possible to have different settings for different use cases, for example:

```
# page1.html
{{ markdowntext|markdownify }} <!-- uses the default settings -->

# page2.html
{{ markdowntext|markdownify:"restricted" }} <!-- uses the 'restricted' settings -->
```

See [Settings](#) for a more detailed explanation.

You can change the behavior of Markdownify by adding them to your `settings.py`. All settings are optional and will fall back to default behavior if not specified.

Warning: The settings described here are for version 0.9 and up. The old style settings are deprecated and will be removed in an upcoming release. For reference, you can find the deprecated settings here: [oldsettings](#)

4.1 Setup

Define a dictionary `MARKDOWNIFY` in your `settings.py` with one or more keys:

```
MARKDOWNIFY = {
    "default": {
        ...
    },
    "other": {
        ...
    }
}
```

The keys can be used in the `markdownify` template filter to choose which settings to use. If you define a default key, you don't have to specify it in the filter.:

```
# page1.html
{{ markdowntext|markdownify }} <!-- uses the default key -->

# page2.html
{{ markdowntext|markdownify:"other" }} <!-- uses the 'other' settings -->
```

If you don't define a `MARKDOWNIFY` dict at all, all settings will fall back to defaults as described below.

4.2 Whitelist tags

Add whitelisted tags with the `WHITELIST_TAGS` key and a list of tags as the value. For example:

```
MARKDOWNIFY = {
    "default": {
        "WHITELIST_TAGS": [
            'a',
            'abbr',
            'acronym',
            'b',
            'blockquote',
            'em',
            'i',
            'li',
            'ol',
            'p',
            'strong',
            'ul'
        ]
    }
}
```

`WHITELIST_TAGS` defaults to `bleach.sanitizer.ALLOWED_TAGS`

4.3 Whitelist attributes

Add whitelisted attributes with the `WHITELIST_ATTRS` key and a list of attributes as the value. For example:

```
MARKDOWNIFY = {
    "default": {
        "WHITELIST_ATTRS": [
            'href',
            'src',
            'alt',
        ]
    }
}
```

`WHITELIST_ATTRS` defaults to `bleach.sanitizer.ALLOWED_ATTRIBUTES`

4.4 Whitelist styles

Add whitelisted styles with the `WHITELIST_STYLES` key and a list of styles as the value. For example:

```
MARKDOWNIFY = {
    "default": {
        "WHITELIST_STYLES": [
            'color',
            'font-weight',
        ]
    }
}
```

WHITELIST_STYLES defaults to `bleach.css_sanitizer.ALLOWED_CSS_PROPERTIES`

4.5 Whitelist protocols

Add whitelisted protocols with the `WHITELIST_PROTOCOLS` key and a list of protocols as the value. For example:

```
MARKDOWNIFY = {
    "default": {
        "WHITELIST_PROTOCOLS": [
            'http',
            'https',
        ]
    }
}
```

MARKDOWNIFY_WHITELIST_PROTOCOLS defaults to `bleach.sanitizer.ALLOWED_PROTOCOLS`

4.6 Enable Markdown Extensions

`Python-Markdown` is extensible with extensions. To enable one or more extensions, add extensions with the `MARKDOWN_EXTENSIONS` key and a list of extensions as the value. For example:

```
MARKDOWNIFY = {
    "default": {
        "MARKDOWN_EXTENSIONS": [
            'markdown.extensions.fenced_code',
            'markdown.extensions.extra',
        ]
    }
}
```

`MARKDOWN_EXTENSIONS` defaults to an empty list (so no extensions are used). To read more about extensions and see the list of official supported extensions, go to [the markdown documentation](#).

4.7 Strip markup

Choose if you want to `strip` or `escape` tags that aren't allowed. `STRIP: True` (default) strips the tags. `STRIP: False` escapes them.:

```
MARKDOWNIFY = {
    "default": {
        "STRIP": False
    }
}
```

4.8 Disable sanitation (bleach)

If you just want to markdownify your text, not sanitize it, add `BLEACH: False`. Defaults to `True`.:

```
MARKDOWNIFY = {
    "default": {
        "BLEACH": False
    }
}
```

4.9 Linkify text

Use `LINKIFY_TEXT` to choose which - if any - links you want automatically to be rendered to hyperlinks. See next example for the default values::

```
MARKDOWNIFY = {
    "default": {
        "LINKIFY_TEXT": {
            "PARSE_URLS": True,

            # Next key/value-pairs only have effect if "PARSE_URLS" is True
            "PARSE_EMAIL": False,
            "CALLBACKS": [],
            "SKIP_TAGS": [],
        }
    }
}
```

Use the following settings to change the linkify behavior:

4.9.1 Linkify email

Set `PARSE_EMAIL` to `True` to automatically linkify email addresses found in your text. Defaults to `False`.

4.9.2 Set callbacks

Set `CALLBACKS` to use `callbacks` to modify your links, for example setting a title attribute to all your links.:

```
def set_title(attrs, new=False):
    attrs[(None, u'title')] = u'link in user text'
    return attrs

# settings.py
...
"CALLBACKS": [set_title, ]
...
```

`CALLBACKS` defaults to an empty list, so no callbacks are used. See the [bleach documentation](#) for more examples.

4.9.3 Skip tags

Add tags with `SKIP_TAGS` to skip linkifying links within those tags, for example `<pre>` blocks. For example:

```
...  
"SKIP_TAGS": ['pre', 'code', ]  
...
```


CHAPTER 5

Tests

Django Markdownify comes with tests to check if settings and defaults produce the expected output. To run the tests, make sure Django Markdownify is *installed*. Then go to your project where Django Markdownify is installed, and run the tests.

```
>>> cd /path/to/your/project
>>> python manage.py test markdownify
```

Django Markdownify - A Django Markdown filter

Django Markdownify is a template filter to convert Markdown to HTML in Django. Markdown is converted to HTML and sanitized.

Example:

```
{% load markdownify %}
{{ 'Some test [link] (#)' | markdownify }}
```

Is transformed to:

```
<p>
  Some <em>test</em> <a href="#">link</a>
</p>
```

The filter is a wrapper around [Markdown](#) and [Bleach](#) and as such supports their settings. It is possible to define multiple settings for multiple usecases.

For example:

```
# settings.py

MARKDOWNIFY = {
    "default": {
        "WHITELIST_TAGS": ["a", "p", "h1", ],
    },
    "alternative": {
        "WHITELIST_TAGS": ["a", "p", ],
        "MARKDOWN_EXTENSIONS": ["markdown.extensions.fenced_code", ]
    }
}
```

And in your templates:

```
<!-- page1.html -->
{{ mytext|markdownify }} <!-- Uses your default settings -->

<!-- page2.html -->
{{ mytext|markdownify:"alternative" }} <!-- Uses your alternative settings -->
```

The code can be found on [Github](#).