django-snakeoil

Aug 09, 2020

Contents:

CHAPTER 1

Setup

1.1 Installation

To install:

```
pip install django-snakeoil
```

1.2 Configuration

If you're using Django templates, add `snakeoil` to your installed apps:

```
INSTALLED_APPS = [
    "myapp",
    "snakeoil",
    # ...
]
```

If you're using Jinja2, you need to add the `get_meta_tags` function to your environment:

```
from jinja2 import Environment
from snakeoil.jinja2 import get_meta_tags

def environment(**options):
    env = Environment(**options)
    env.globals.update(
        {
            "get_meta_tags": get_meta_tags,
            # ...
        }
    )
    return env
```

CHAPTER 2

Usage

There are several ways to use snakeoil.

2.1 Global metadata

Usually, you will have some metadata that will be the same on every page, or at least a majority of them. You can set these in your `settings.py`:

```
SNAKEOIL_DEFAULT_TAGS={
    "default": [
        {"name": "author", "content": "Tom Carrick"},
        {"property": "og:site_name", "content": "My Website"},
        {"property": "og:type", "content": "website"}
    ]
    "eo": [
        {"property": "og:site_name", "content": "Mia Ratejo"},
    ]
}
```

Note: The `default` key here is for the language. If you're not using internationalization, you need only use the `default` key.

This will set the Open Graph site name property to `Mia Ratejo` if the page is requested in Esperanto, or `My Website` for any other language. Additionally, the `meta author` tag will be set to `Tom Carrick` on every page.

It's possible to set both the meta `name` and `property` for the same tag:

```
{
    "default": [
        {
```

(continues on next page)

(continued from previous page)

```
            "name": "description",
            "property": "og:description",
            "content": "My meta description.",
        }
    ]
```

2.2 Per-object metadata

Metadata can be set and overridden per-object.

First, inherit your model from `SEOModel`:

```
from snakeoil.models import SEOModel

class Article(SEOModel):
    title = models.CharField(max_length=200)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    main_image = models.Imagefield()

    def get_absolute_url(self):
        # Snakeoil will use this to find the object
        # if you don't pass it in.
        return reverse("...")

    @property
    def author_name(self):
        return self.author.get_full_name()
```

Then you can build a JSON object (perhaps in a code-editor) and add this into the `meta_tags` field of the object in the Django Admin, or with code. For example:

```
{
    "default": [
        {"property": "og:type", "content": "article"}
    ]
}
```

This will override the type from `website` (defined in the global config) to `article`.

2.2.1 Setting metadata from object attributes

You can also set metadata from object attributes with the `attribute` key:

```
{
    "default": [
        {"name": "author", "attribute": "author"},
        {"property": "og:image", "attribute": "main_image"},
        {"property": "og:title", "attribute": "title"}
    ]
}
```

For images using `ImageField` in an `og:image`, this will automatically populate the `og:image:width` and `og:image:height` properties.

2.2.2 Setting metadata dynamically

Usually, your models will have some metadata stored as model fields or attributes, and it's a lot of effort to override this for every obejct. To set per-model metadata based on object attributes, you can define a property called `snakeoil_metadata` on your model:

```
from snakeoil.models import SEOModel

class Article(SEOModel):
    title = models.CharField(max_length=200)
    author = models.ForeignKey(User, on_delete=models.CASCADE)
    main_image = models.Imagefield(blank=True, null=True)

    @property
    def author_name(self):
        return

    @property
    def snakeoil_metadata(self):
        metadata = {
            "default": [
                {
                    "name": "author",
                    "content": self.author.get_full_name(),
                },
                {"property": "og:title", "content": self.title},
            ]
        }
        if self.main_image:
            metadata.append(
                {"property": "og:image", "attribute": "main_image"}
            )
        return metadata
```

Note: It's important to use `attribute` for `og:image` so the height and and width can be set automatically.

2.3 Per-URL metadata

Sometimes you don't have an object, or can't add anything to it, if for example you're using `django.contrib.flatpages` or are using static views. For this, you can use the `SEOPath` model, added to the Django admin.

2.4 Using static files

You can also get files by their static path. However, this won't auatomatically add `og:image:width` and `og:image:height` properties, so these need to be added manually if needed:

```
{
    "default": [
        {"property": "og:image", "static": "img/default_image.jpg"},
        {"property": "og:image:width", "content": "600"},
        {"property": "og:image:height", "content": "480"},
```

(continues on next page)

(continued from previous page)

```
    ]
}
```

CHAPTER 3

Indices and tables

- genindex
- modindex
- search