django-sitemetrics Documentation

Release 1.2.0

Igor 'idle sign' Starikov

Feb 04, 2022

Contents

http://github.com/idlesign/django-sitemetrics

Reusable application for Django providing easy means to integrate site metrics counters into your sites

CHAPTER 1

Requirements

1. Python 3.6+
2. Django 2.0+
3. Django Sites contrib enabled (optional)
4. Django Admin contrib enabled (optional)

CHAPTER 2

Table of Contents

2.1 Getting started

- Add the `sitemetrics` application to `INSTALLED_APPS` in your settings file (usually `settings.py`).
- Add `{% load sitemetrics %}` tag to the top of a template (usually base template, e.g. `_base.html`).
- Use `./manage.py migrate` to install sitemetrics tables into your database.

2.1.1 Quick example

You have two options to add metrics counters code to your pages:

1. Let's add Google Analytics counter without Admin Contrib involvement add a `four arguments` sitemetrics tag into your template:

   ```
   {% sitemetrics by google for "UA-000000-0" %}
   ```

   Here: `google` is a metrics provider alias; `UA-000000-0` - metrics counter keycode.

2. Now let's use `no arguments` sitemetrics tag notation to place metrics counter code associated through the Admin Contrib with the current site (if any):

   ```
   {% sitemetrics %}
   ```

You're done.

Note: Since v0.6.0 counter code in DEBUG mode is replaced with `<!-- sitemetrics counter removed on DEBUG -->` to keep stats clean. If you wish to see counter code in DEBUG set `SITEMETRICS_ON_DEBUG` to `True` in project settings file.

2.2 Customizing sitemetrics

Here we'll talk on how you can customize **sitemetrics** to your needs.

2.2.1 Customizing metric counters

Some metrics providers allows you to adjust metrics settings, thus changing counters code.

In **sitemetrics** counters are described in classes, so you can adjust counters functionality by customizing class attributes values.

Let' try and customize the built-in *Yandex Metrics* counter:

1. First, we define our new customized class somewhere within your project, (let's say, *my_counters.py* inside *my_app* application):

```
from sitemetrics.providers import Yandex

# We inherit from the built-in Yandex Metrics counter class.
class MyYandexProvider(Yandex):

    title = 'My Yandex Metrika'  # This is to differentiate from parent.

    def __init__(self):
        # Parent class has `webvisor` counter param set to True,
        # bu we don't want that functionality and disable it.
        self.params['webvisor'] = False
```

2. Second, we introduce our class to Django, putting *SITEMETRICS_PROVIDERS* tuple into projects' *settings.py*:

```
# Below is a tuple with classes paths to your metrics counters classes.
# We have just one.
SITEMETRICS_PROVIDERS = ('my_app.my_counters.MyYandexProvider',)
```

2.2.2 Implementing new metrics providers

1. Implement a class describing counter aspects, somewhere within your project, (let's say, *my_metrics.py* inside *my_app* application):

```
from sitemetrics.providers import MetricsProvider

# We inherit from the built-in MetricsProvider counter class.
class MyMetrics(MetricsProvider):

    title = 'My Metrics'  # Human-friendly title.
    alias = 'my_metrics'  # Alias to address counter from templates.

    # And here are counter params, which are passed into counter template.
    params = {
        'my_param_1': True,
        'my_param_2': 30,
    }
```

2. Create a counter template (**sitemetrics** will search for it in'{your_app}/templates/sitemetrics/{provider_alias}.html').

   keycode variable will be available within this template. *keycode.keycode* will contain counter identifier:

```
keycode: {{ keycode.keycode }}

{% if keycode.my_param_1 %}
    my_param_1 set to True,
    my_param_2 is {{ keycode.my_param_2 }}
{% endif %}
```

The code above is of course not a real counter code, yet it can give you an idea on how to create a real one.

3. Now if you want to see your counter built into the **sitemetrics** fire an issue or a pull request at https://github.com/idlesign/django-sitemetrics/ or if you want to keep it private use *SITEMETRICS_PROVIDERS* definition approach (described in the previous section) to introduce your class to your Django project.

CHAPTER 3

Get involved into django-sitemetrics

Submit issues. If you spotted something weird in application behavior or want to propose a feature you can do that at https://github.com/idlesign/django-sitemetrics/issues

Write code. If you are eager to participate in application development, fork it at https://github.com/idlesign/django-sitemetrics, write your code, whether it should be a bugfix or a feature implementation, and make a pull request right from the forked project page.

Translate. If want to translate the application into your native language use Transifex: https://www.transifex.com/projects/p/django-sitemetrics/.

Spread the word. If you have some tips and tricks or any other words in mind that you think might be of interest for the others — publish them.

CHAPTER 4

Also

If the application is not what you want for site metrics, you might be interested in considering the other choices — https://www.djangopackages.com/grids/g/analytics/