django-request

Release 1.5.1

June 23, 2017

Contents

CHAPTER 1

django-request

django-request is a statistics module for django. It stores requests in a database for admins to see, it can also be used to get statistics on who is online etc.

As well as a site statistics module, with the active_users template tag and manager method you can also use django-request to show who is online in a certain time.

First steps

- *Installation*
- *Settings*
- *Template tags*

Quick start guide

Note: To find the request overview page, please click on Requests inside the admin, then "Overview" on the top right, next to "add request".

Once you have installed `django-request` you can add it to a django project by following these steps;

1. Install the blog app by adding `'request'` to `INSTALLED_APPS`.
2. Run `manage.py migrate` so that Django will create the database tables.
3. Add `request.middleware.RequestMiddleware` to `MIDDLEWARE`. If you use `django.contrib.auth.middleware.AuthenticationMiddleware`, place `RequestMiddleware` after it. If you use `django.contrib.flatpages.middleware.FlatpageFallbackMiddleware` place `request.middleware.RequestMiddleware` before it else flatpages will be marked as error pages in the admin panel.

4. Make sure that the domain name in django.contrib.sites admin is correct. This is used to calculate unique visitors and top referrers.

django-admin.py

purgerequests

This command can be used to purge old requests, it takes two arguments amount and duration. Example:

```
$ python manage.py purgerequests 1 month
$ python manage.py purgerequests 2 weeks
```

It also has a option called `--noinput`, if this is supplied, it will not ask you to confirm. With this option you can use this command in a cron.

Valid durations: `hour(s)`, `day(s)`, `week(s)`, `month(s)`, `year(s)`

CHAPTER 2

Settings

REQUEST_IGNORE_AJAX

Default: `False`

If this is set to True, then ajax requests will be ignored and not added to the database. To determine if a request was ajax, we use HttpRequest.is_ajax(), see Django documentation for more information.

REQUEST_IGNORE_IP

Default: `None`

Any requests from a IP Address in this list will be ignored.

REQUEST_LOG_IP

Default: `True`

If set to False, ip addresses are replaced with REQUEST_IP_DUMMY.

REQUEST_IP_DUMMY

Default: `1.1.1.1`

Used dummy address, if REQUEST_LOG_IP is set to False.

REQUEST_ANONYMOUS_IP

Default: `False`

If set to True, last octet of the ip is set to 1.

REQUEST_LOG_USER

Default: `True`

If set to False, user are not logged (set to None).

REQUEST_IGNORE_USERNAME

Default: `None`

Any requests from users in this list will be ignored.

REQUEST_IGNORE_PATHS

Default: `None`

Any requests which match these paths will be ignored. This setting should be a tuple filled with regex paths.

Example:

```
REQUEST_IGNORE_PATHS = (
    r'^admin/',
)
```

REQUEST_IGNORE_USER_AGENTS

Default: `None`

Any request with a user agent that matches any pattern in this list will be ignored.

Example:

```
REQUEST_IGNORE_USER_AGENTS = (
    r'^$', # ignore requests with no user agent string set
    r'Googlebot',
    r'Baiduspider',
)
```

REQUEST_TRAFFIC_MODULES

Default:

```
(
'request.traffic.UniqueVisitor',
'request.traffic.UniqueVisit',
'request.traffic.Hit',
)
```

These are all the items in the traffic graph and table on the overview page. If you wish to remove or add a item you can override this setting and set what you want to see. There are also many more options you can add from the following list;

- `'request.traffic.Ajax'`: To show the amount of requests made from javascript.
- `'request.traffic.NotAjax'`: To show the amount of requests that are NOT made from javascript.
- `'request.traffic.Error'`: To show the amount of error's, this includes error 500 and page not found.
- `'request.traffic.Error404'`: To show the amount of page not found.
- `'request.traffic.Hit'`: To show the total amount of requests.
- `'request.traffic.Search'`: To display requests from search engines.
- `'request.traffic.Secure'`: To show the amount of requests over SSL.
- `'request.traffic.Unsecure'`: To show the amount of requests NOT over SSL.
- `'request.traffic.UniqueVisit'`: To show visits based from outsider referrals.
- `'request.traffic.UniqueVisitor'`: To show the amount of requests made from unique visitors based upon IP address.
- `'request.traffic.User'`: To show the amount of requests made from a valid user account.
- `'request.traffic.UniqueUser'`: To show the amount of users.

REQUEST_PLUGINS

Default:

```
(
    'request.plugins.TrafficInformation',
    'request.plugins.LatestRequests',
    'request.plugins.TopPaths',
    'request.plugins.TopErrorPaths',
    'request.plugins.TopReferrers',
    'request.plugins.TopSearchPhrases',
    'request.plugins.TopBrowsers',
)
```

These are all the plugins you can see on the overview page. If you wish to remove or add a plugin you can override this setting and set what you want to see. Here is a list of all the plugins and what they do;

- `'request.plugins.TrafficInformation'`: This is a plugin to show a table of the traffic modules.
- `'request.plugins.LatestRequests'`: The last 5 requests.
- `'request.plugins.TopPaths'`: A list of all the paths (not including errors).
- `'request.plugins.TopErrorPaths'`: A list of the paths which error, this can be useful for finding bugs.
- `'request.plugins.TopReferrers'`: Shows a list of top referrals to your site.

- `'request.plugins.TopSearchPhrases'`: Shows a list of all the search phrases used to find your site.
- `'request.plugins.TopBrowsers'`: Shows a graph of the top browsers accessing your site.
- `'request.plugins.ActiveUsers'`: Show a list of active users in the last 5 minutes. This may not be a good idea to use on a large website with lots of active users as it will generate a long list.

REQUEST_BASE_URL

Default: `'http://%s' % Site.objects.get_current().domain`

This setting should only be set if you use SSL or do not use django.contrib.sites. This is the base url for detecting referral from within the same site.

REQUEST_ONLY_ERRORS

Default: `False`

If this is set to True, django-request will ONLY store error returning request/responses. This can be useful to use django-request purely as a error detection system.

REQUEST_VALID_METHOD_NAMES

Default: ('get', 'post', 'put', 'delete', 'head', 'options', 'trace')

Any request which is not in this tuple/list will be ignored.

CHAPTER 3

Installing django-request

You can install django-request from the following;

Via tar.gz / zip archive

You can download the latest version in a archive created directly from the git repository via the following links:

- tar.gz: https://github.com/django-request/django-request/tarball/master
- zip: https://github.com/django-request/django-request/zipball/master

Once you have downloaded and extracted django-request, you can use the following command to install.

```
$ python setup.py install
```

Via git repository

Git is one of the best ways to download django-request, it allows you to easily pull the latest version, just by typing `git pull` within the django-request directory. To download git or how to use it. See: http://git-scm.com/ .

You can use the following command to clone the git repository:

```
$ git clone git://github.com/django-request/django-request.git
$ ln -s django-request/request <PYTHONPATH>
```

Using a package-management tool

This is one of the easiest ways to install django-request. There are two different package-management tools which you could use:

pip

pip is one of the more popular package-management systems for python. You can find documentation, and how to install pip itself here. Once you have pip installed and running, simply type:

```
$ pip install django-request
```

easy_install

Another option is to use easy_install, first you need to install easy_install. You can find documentation and how to install easy_install here. Once you have easy_install up and running, just type:

```
$ easy_install django-request
```

CHAPTER 4

Template tags

active_users

This template tag will get a list of active users based on time, if you do not supply a time to the tag, the default of 15 minutes will be used. With the 'as' clause you can supply what context variable you want the user list to be. There is also a 'in' clause, after in you would specify a amount and a duration. Such as 2 hours, of 10 minutes.

```
{% active_users in [amount] [duration] as [varname] %}
{% active_users as [varname] %}
{% active_users %}
```

Example usage:

```
{% load request_tag %}
{% active_users in 10 minutes as user_list %}
{% for user in user_list %}
    {{ user.username }}
{% endfor %}
```