dj-hitcount Documentation

Release 1.1.0

Abhyudai

Jul 17, 2021

CONTENTS

Dj-Hitcount allows you to track the number of hits/views for a particular object.

CHAPTER

ONE

OVERVIEW

Dj-Hitcount allows you to track the number of hits (views) for a particular object. This isn't meant to be a full-fledged tracking application or a real analytic tool; it's just a basic hit counter.

How one tracks a "hit" or "view" of a web page is not such a simple thing as it might seem. That's why folks rely on Google Analytics or similar tools. It's tough! This is a simple app with some settings and features that should suit the basic needs of smaller sites.

It comes ready to track hits with a `HitCountDetailView` and a `HitCountJSONView` (to use the out-of-the-box JavaScript method, you will need jQuery – although writing your own JavaScript implementation may not be hard).

1.1 Migrating from `django-hitcount`

Please see the *guidelines for migration*.

CHAPTER

TWO

INSTALLATION AND USAGE

Install using `pip`.

```
$ python -m pip install dj-hitcount
```

If you want, you may install it from the source, grab the source code and run `setup.py`.

```
$ git clone git://github.com/abhiabhi94/dj-hitcount.git
$ cd dj-hitcount
$ python setup.py install
```

Add `dj-hitcount` to your `INSTALLED_APPS`:

```
# settings.py
INSTALLED_APPS = (
    ...
    'hitcount'
)
```

View the *additional settings section* for a list of the available settings that can be configured.

For a working implementation, you can view the example project on Github.

2.1 Counting Hits

The main business-logic for evaluating and counting a *Hit* is done in `hitcount.mixins.HitCountViewMixin.hit_count()`. You can use this static method directly in your own Views or you can use one of the Views packaged with this app.

- *HitCountJSONView*: a JavaScript implementation which moves the business-logic to an Ajax View and hopefully speeds up page load times and eliminates some bot-traffic
- *HitCountDetailView*: which provides a wrapper from Django's generic `DetailView` and allows you to process the Hit as the view is loaded

2.1.1 HitCountMixin

This mixin can be used in your own class-based views or you can call the `hit_count()` method directly. The method takes two arguments, a `HttpRequest` and `HitCount` object it will return a namedtuple: `UpdateHitCountResponse(hit_counted=Boolean, hit_message='Message')`.

`hit_counted` will be `True` if the hit was counted and `False` otherwise. `hit_message` will indicate by what means the Hit was either counted or ignored.

It works like this.

```
from hitcount.models import HitCount
from hitcount.mixins import HitCountViewMixin

# first get the related HitCount object for your model object
hit_count = HitCount.objects.get_for_object(your_model_object)

# next, you can attempt to count a hit and get the response
# you need to pass it the request object as well
hit_count_response = HitCountViewMixin.hit_count(request, hit_count)

# your response could look like this:
# UpdateHitCountResponse(hit_counted=True, hit_message='Hit counted: session key')
# UpdateHitCountResponse(hit_counted=False, hit_message='Not counted: session key has␣
↪active hit')
```

To see this in action see the views.py code.

2.1.2 HitCountJSONView

The `hitcount.views.HitCountJSONView` can be used to handle an AJAX POST request. Dj-hitcount comes with a bundled jQuery plugin for speeding up the `$.post` process by handling the retrieval of the CSRF token for you.

If you wish to use the `HitCountJSONView` in your project you first need to update your `urls.py` file to include the following:

```
# urls.py
from django.urls import path

urlpatterns = [
    ...
    path('hitcount/', include('hitcount.urls', namespace='hitcount')),
]
```

Next, you will need to add the JavaScript Ajax request to your template. To do this, use the `{% get_hit_count_js_variables for post as [var_name] %}` template tag to get the `ajax_url` and `hitcount_pk` for your object. The `hitcount_pk` is needed for POST-ing to the `HitCountJSONView`.

Here is an example of how all this might work together with the bundled jQuery plugin. It is taken from the example project and the jQuery can be modified to suit your needs. In the example below it simply updates the template with the `HitCountJSONView` response after the Ajax call is complete.

```
{% load staticfiles %}
<script src="{% static 'hitcount/jquery.postcsrf.js' %}"></script>
```

(continues on next page)

(continued from previous page)

```
{% load hitcount_tags %}
{% get_hit_count_js_variables for post as hitcount %}
<script type="text/javascript">
jQuery(document).ready(function($) {
  // use the template tags in our JavaScript call
  $.postCSRF("{{ hitcount.ajax_url }}", { hitcountPK : "{{ hitcount.pk }}" })
    .done(function(data){
      $('<i />').text(data.hit_counted).attr('id','hit-counted-value').appendTo('#hit-
↪counted');
      $('#hit-response').text(data.hit_message);
  }).fail(function(data){
      console.log('POST failed');
      console.log(data);
  });
});
</script>
```

2.1.3 HitCountDetailView

The `HitCountDetailView` can be used to do the business-logic of counting the hits by setting `count_hit=True`. See the views section for more information about what else is added to the template context with this view.

Here is an example implementation from the example project:

```
from hitcount.views import HitCountDetailView

class PostCountHitDetailView(HitCountDetailView):
    model = Post        # your model goes here
    count_hit = True    # set to True if you want it to try and count the hit
```

Note: Unlike the JavaScript implementation (above), this View will do all the HitCount processing *before* the content is delivered to the user; if you have a large dataset of Hits or exclusions, this could slow down page load times. It will also be triggered by web crawlers and other bots that may not have otherwise executed the JavaScript.

2.2 Displaying Hits

There are different methods for *displaying* hits:

- *Template Tags*: provide a robust way to get related counts.
- Views: allows you to wrap a class-based view and inject additional context into your template.
- *Models*: can have a generic relation to their respective `HitCount`.

2.2.1 Template Tags

For a more granular approach to viewing the hits for a related object you can use the `get_hit_count` template tag.

```
# remember to load the tags first
{% load hitcount_tags %}

# Return total hits for an object:
{% get_hit_count for [object] %}

# Get total hits for an object as a specified variable:
{% get_hit_count for [object] as [var] %}

# Get total hits for an object over a certain time period:
{% get_hit_count for [object] within ["days=1,minutes=30"] %}

# Get total hits for an object over a certain time period as a variable:
{% get_hit_count for [object] within ["days=1,minutes=30"] as [var] %}
```

2.2.2 Views

The `hitcount.views.HitCountDetailView` extends Django's generic `DetailView` and injects an additional context variable `hitcount`.

```
{# the primary key for the hitcount object #}
{{ hitcount.pk }}

{# the total hits for the object #}
{{ hitcount.total_hits }}
```

If you have set `count_hit=True` (see: *HitCountDetailView*) two additional variables will be set.

```
{# whether or not the hit for this request was counted (true/false) #}
{{ hitcount.hit_counted }}

{# the message form the UpdateHitCountResponse #}
{{ hitcount.hit_message }}
```

CHAPTER

THREE

MODELS

Note: You are not *required* to do anything specific with your models; dj-hitcount relies on a `GenericForeignKey` to create the relationship to your model's `HitCount`.

If you would like to add a reverse lookup in your own model to its related `HitCount` you can utilize the `hitcount.mixins.HitCountModelMixin`.

```
from django.db import models

from hitcount.mixins import HitCountModelMixin
from hitcount.conf import settings as hitcount_settings

# here is an example model with a GenericRelation
class MyModel(models.Model, HitCountMixin):

  # adding a generic relationship makes sorting by Hits possible:
  # MyModel.objects.order_by("hit_count_generic__hits")
  hit_count_generic = GenericRelation(
      hitcout_settings.HITCOUNT_HITCOUNT_MODEL,
      object_id_field='object_pk',
      related_query_name='hit_count_generic_relation'
  )

# you would access your hit_count like so:
my_model = MyModel.objects.get(pk=1)
my_model.hit_count.hits                 # total number of hits
my_model.hit_count.hits_in_last(days=7) # number of hits in last seven days
```

3.1 Customization

`dj-hitcount` allows you to customize `HitCount` model.

1. Define your own `hitcount` model inherited from `HitCountBase`.
2. Now when *models.py* in your application has the definition of a custom hitcount model, you need to instruct Django to use it for your project instead of a built-in one:

```
# Somewhere in your settings.py do the following.
# Here `myapp` is the name of your application, `MyHitCount` is the names of your␣
↪customized model.
```

(continues on next page)

(continued from previous page)

```
HITCOUNT_HITCOUNT_MODEL = 'myapp.MyHitCount'
```

3. Run *manage.py migrate* to install your customized models into DB.

CHAPTER

FOUR

ADDITIONAL SETTINGS

There are a few additional settings you can use to customize `dj-hitcount` by using them in your `settings.py` file.

4.1 HITCOUNT_USE_IP

This tells whether to store the IP address from the request. Defaults to `True` for backwards compatibility with `django-hitcount`.

> **Warning:** The default value for `HITCOUNT_USE_IP` will change to `False` in the next major version bump(if a lot of things go well, then in `v2.0.0`). Explicitly set it to `True` to use IP.

4.2 HITCOUNT_KEEP_HIT_ACTIVE

This is the number of days, weeks, months, hours, etc (using a `timedelta` keyword argument), that an `Hit` is kept **active**. If a `Hit` is **active** a repeat viewing will not be counted. After the **active** period ends, however, a new `Hit` will be recorded. You can decide how long you want this period to last and it is probably a matter of preference.:

```
# default value
HITCOUNT_KEEP_HIT_ACTIVE = { 'days': 7 }
```

4.3 HITCOUNT_HITS_PER_IP_LIMIT

Limit the number of **active** `Hits` from a single IP address. 0 means that it is unlimited.:

```
# default value
HITCOUNT_HITS_PER_IP_LIMIT = 0
```

4.4 HITCOUNT_EXCLUDE_USER_GROUP

Exclude `Hits` from all users in the specified user groups. By default, this is set to an empty list (all users counted). In the example, below, it will exclude all your 'Editors'.:

```
# example value, default is empty tuple
HITCOUNT_EXCLUDE_USER_GROUP = ( 'Editor', )
```

4.5 HITCOUNT_KEEP_HIT_IN_DATABASE

`Hits` remain in the database indefinitely unless you run the `hitcount_cleanup` management command. This setting specifies a `timedelta` within which to keep/save `Hits`. Any `Hit` older than the time specified will be removed from the `Hits` table.:

```
# default value
HITCOUNT_KEEP_HIT_IN_DATABASE = { 'days': 30 }
```

CHAPTER

FIVE

MANAGEMENT COMMANDS

By default, your `Hits` remain in the database indefinitely. If you would like to periodically prune your stale `Hits` you can do so by running the the management command `hitcount_cleanup`.:

```
./manage.py hitcount_cleanup
```

The command relies on the setting `HITCOUNT_KEEP_HIT_IN_DATABASE` to determine how far back to prune. See the *additional settings section* for more information.

CHAPTER

SIX

REAL WORLD EXAMPLE

There is an example project that tries to demonstrate the functionality of this app. You can get it working using the Django development server. Be sure to run this inside your own `virtualenv`:

```
$ git clone git@github.com:abhiabhi94/dj-hitcount.git
$ pip install -r dev-requirements.txt
$ python -m django --settings=test.settings migrate  # will load some data fixtures for␣
↪you
$ python -m django --settings=test.settings createsuperuser  # for access to the admin␣
↪portion
$ python -m django --settings=test.settings runserver        # should be all set!
```

When you are ready to work on your own site, check out the *Installation and Usage* and *Additional Settings* sections.

CHAPTER

SEVEN

CONTRIBUTING TO DJ HITCOUNT

There are many ways to contribute to the project. You may improve the documentation, address a bug, add some feature to the code or do something else. All sort of contributions are welcome.

7.1 Development

To start development on this project, fork this repository and follow the following instructions.

```
# clone the forked repository
$ git clone YOUR_FORKED_REPO_URL

# create a virtual environment
$ python3 -m venv venv
# activate the virtual environment(unix users)
$ . venv/bin/activate
# activate the virtual environment(window users)
$ venv\Scripts\activate
# install dependencies
(venv) $ pip install -e . Django -r dev-requirements.txt pre-commit
# migrate the migrations to the database and also creates some placeholder data
(venv) $ python manage.py migrate
# start the development server
(venv) $ python manage.py runserver
```

7.2 Testing

To run tests against a particular `python` and `django` version installed inside your virtual environment, you may use:

```
(venv) $ pytest
```

This skips the `selenium` tests which are a bit slow to run. To run them as well use

```
(venv) $ pytest --runslow
```

You may have to install `firefox` and gecko-driver to run these successfully.

To run tests against all supported `python` and `django` versions, you may run:

```
# install dependency
(venv) $ pip install tox
# run tests
(venv) $ tox
```

CHAPTER

EIGHT

CHANGELOG

8.1 1.1.0

- Confirm support for python `3.10-dev` (#11).
- Optimize query for saving of increased and decreased hits(#10).
- Prevent compiling of `regex` for IP detection on every import of `hitcount.utils` (#8)

8.2 1.0.1

- Fix saving of anonymous sessions(#4).

8.3 1.0.0

- Bring back project to life.
- **Change in project structure**
  - **`signals`**
    - **`delete_hit_count` from `hitcount.models` has been moved to `hitcount.signals`.**
      - The argument `save_hitcount` to the function `delete_hit_count_handler` (this process the signal `delete_hit_count`) is now `keyword-only`. The earlier design pattern was a case of boolean-trap.
  - **`mixins`**
    - `HitCountMixin` from `hitcount.models` has been renamed as `HitCountModelMixin` and moved to `hitcount.mixins`.
    - `HitCountMixin` from `hitcount.views` has been renamed as `HitCountViewMixin` and moved to `hitcount.mixins`.
  - **`models`**
    - `BlackListIP` renamed to `BlockedIP`.
    - `BlackListUserAgent` renamed to `BlockedUserAgent`.
    - **The `ip` field for `Hit` model has been made optional. This hopefully makes the project GDPR compliant.**

        - To maintain backwards compatibility with `django-hitcount`, an additional setting *HITCOUNT_USE_IP* has been added.

- **views**
    - `hitcount.views.update_hit_count_ajax` that was to be removed in `django-hitcount 1.2` has been removed. Use `hitcount.views.HitCountJSONView` instead.
    - `hitcount.views._update_hit_count` that was to be removed in `django-hitcount 1.2` has been removed. Use `hitcount.mixins.HitCountViewMixin.hit_count` instead.
- removed additional dependency of `django-etc`.
- added additional unit tests. Test coverage is now `100%`.

CHAPTER

NINE

MIGRATING FROM DJANGO-HITCOUNT

The project has been built with the idea to make the migrations as smooth as possible. After installation of `dj-hitcount`, run the migrations command to make the appropriate changes to the database (see the `models` section in this list for specifics).

```
python manage.py migrate hitcount
```

You will also have to make some changes if you were using any one of the following:

- **`signals`**
  - **`delete_hit_count` from `hitcount.models` has been moved to `hitcount.signals`.**
    - The argument `save_hitcount` to the function `delete_hit_count_handler` (this process the signal `delete_hit_count`) is now `keyword-only`. The earlier design pattern was a case of boolean-trap.
- **`mixins`**
  - `HitCountMixin` from `hitcount.models` has been renamed as `HitCountModelMixin` and moved to `hitcount.mixins`.
  - `HitCountMixin` from `hitcount.views` has been renamed as `HitCountViewMixin` and moved to `hitcount.mixins`.
- **`models`**
  - `BlackListIP` renamed to `BlockedIP`.
  - `BlackListUserAgent` renamed to `BlockedUserAgent`.
  - **The `ip` field for `Hit` model has been made optional. This hopefully makes the project GDPR compliant. Please**
    - To maintain backwards compatibility with `django-hitcount`, an additional setting *HITCOUNT_USE_IP* has been added.
- **`views`**
  - `hitcount.views.update_hit_count_ajax` that was to be removed in `django-hitcount 1.2` has been removed. Use `hitcount.views.HitCountJSONView` instead.
  - `hitcount.views._update_hit_count` that was to be removed in `django-hitcount 1.2` has been removed. Use `hitcount.mixins.HitCountViewMixin.hit_count` instead.
- removed additional dependency of `django-etc`.

CHAPTER

TEN

ISSUES

Use the GitHub issue tracker for dj-hitcount to submit bugs, issues, and feature requests.