django-trending Documentation

Release 0.1.2

Eldarion

Aug 03, 2017

Contents

CHAPTER 1

Development

The source repository can be found at https://github.com/eldarion/django-trending

Contents

ChangeLog

0.1

- initial release

Installation

- To install

```
pip install django-trending
```

- Add `trending` to your `INSTALLED_APPS` setting:

```
INSTALLED_APPS = (
    # other apps
    "trending",
)
```

Usage

This app is pretty simple but requires a certain amount of integration with your project. Also while not required, it is recommended that you use soemthing like celery to background some of the operations like updating the summary log.

In order to best illustrate how to use this, let's see how to integrate this into a site that has a content app called `articles` that is just a simple blog-like app.

First we want to record the views. We do that through putting signal handlers in for the articles.signals.article_viewed signal:

```
@receiver(article_viewed)
def handle_article_viewed(sender, **kwargs):
    request = kwargs.get("request")
    article = kwargs.get("article")
    if request and article:
        request.session["article-%s" % article.pk] = "viewed"
        task_update_article_trending.delay(
            session_key=request.session.session_key,
            article_pk=article.pk
        )
```

You'll noticed that I am calling `task_update_article_trending`. This is a celery task that has been written to do the business of recording the trending view count and updating summaries in the background to get that work out of the request/response cycle:

```
def create_view_log(session_key, obj, kind=""):
    view_log, created = ViewLog.objects.get_or_create(
        session_key = session_key,
        viewed_content_type = ContentType.objects.get_for_model(obj),
        viewed_object_id = obj.pk,
        kind = kind
    )
    if created:
        DailyViewSummary.summarize(
            for_date=datetime.date.today(),
            view_log=view_log
        )

@task
def task_update_article_trending(session_key, article_pk):
    article = Article.objects.get(pk=article_pk)
    if article.is_car_article():
        create_view_log(session_key, article, kind="CAR")
    else:
        create_view_log(session_key, article)
```

Now that we are recording views and updating summaries, let's think about what happens when a user decides to delete an article. We don't want those summaries laying around. So let's quickly wire up a signal handler to remove the appropriate entries when an article is deleted:

```
@receiver(post_delete, sender=Article)
def handle_article_delete(sender, **kwargs):
    deleted_article = kwargs.get("instance")
    if deleted_article is not None:
        ct = ContentType.objects.get_for_model(Article)
        ViewLog.objects.filter(viewed_content_type=ct, viewed_object_id=deleted_
↪article.pk).delete()
        DailyViewSummary.objects.filter(viewed_content_type=ct, viewed_object_
↪id=deleted_article.pk).delete()
```

Okay, now that view counts and summaries are being recorded as well as removed properly, all the bookkeeping is taken care adn we can move on to seeing the fruits of our labor by displaying the "top 5 articles for the past 7 days".

Let's start with a standard view, well a snippet of a view where we get a queryset of trending objects:

```
ctx = {
    "trending_articles": DailyViewSummary.objects.trending(Article, days=7),
    "trending_car_articles": DailyViewSummary.objects.trending(Article, days=7, kind=
↪"CAR")
}
```

Now, finally to display this content in your template:

```
<h2>Top 5 Articles for Past 7 Days</h2>

<ol>
    {% if trending_articles %}
        {% for trending_stat in trending_articles|slice:":5" %}
            {% with trending_stat.object as article %}
                <li>
                    <a href="{{ article.get_absolute_url }}">
                        {{ article.title }}
                    </a>
                </li>
            {% endwith %}
        {% endfor %}
    {% endif %}
</ol>

<h2>Top 5 Car Reports for Past 7 Days</h2>

<ol>
    {% if trending_car_articles %}
        {% for trending_stat in trending_car_articles|slice:":5" %}
            {% with trending_stat.object as article %}
                <li>
                    <a href="{{ article.get_absolute_url }}">
                        {{ article.title }}
                    </a>
                </li>
            {% endwith %}
        {% endfor %}
    {% endif %}
</ol>
```