django-cache-tools Documentation

Release 0.1.0

Diego Lapiduz

January 03, 2017

Contents

Django Cache tools is a basic set of tools to built on top of the django cache framework to make it easier to use and add caching related features.

Contents:

CHAPTER 1

KeyableModel

The concept of a keyable model was "inspired" by the Ruby on Rails cache_key system where you cache partials based on the updated timestamp of the instance. This leverages memcached LRU algorithm where unused cache items are not a problem and are discarded after a while of not being used.

1.1 Installation

To make a model a Keyable Model you need to inherit that class in your model:

```
from cache_tools.models import KeyableModel
# ...
class Profile(KeyableModel):
  # Your model stuff
```

Then sync your db or create your south migration:

```
python manage.py schemamigration front add_keyable_model --auto
```

1.2 Usage

To use it in your templates you must pass the cache key as a parameter to the cache block:

```
{% load cache %}
{% cache 86400 cache_tools profile.cache_key %}
    <p>
        Lots of very time consuming code.
    </p>
{% endcache %}
```

CHAPTER 2

Group Cache

The idea behind group caching is to have a set of pages that you group under a certain name allowing you to expire the whole set instead of having to look for the individual cache entries.

2.1 View Cache

To cache a page in a group you just use the cache_page_in_group decorator:

```
#views.py
from cache_tools.tools import cache_page_in_group

@cache_page_in_group('profiles')
def show(req, slug):
  # ...
```

When you need to expire that group you use the expire_cache_group method:

```
from cache_tools.tools import expire_cache_group
# ...
expire_cache_group('profiles')
```

2.2 Fragment Cache

You can also use group caching to cache fragments using the "get_group_key" template tag.

```
# cacheable.html
{% get_group_key group_name as group_key %}

{% cache 600 page_title group_key %}
  <!-- Long running code -->
{% endcache %}
```

CHAPTER 3

expire_page method

Django has a great cache framework but doesn't provide an easy way to expire a view that has been cached.

This method does just that.

3.1 Example

If you have a view like:

```
#views.py
@cache_page(60 * 10)
def show(req, slug):
  # ...

#urls.py
# ...
url(r'^profile/(?P<stub>.*)/$', 'show', name='show_profile'),
# ...
```

Then you can use expire_page like:

```
from cache_tools.tools import expire_page
# ...
expire_page(reverse('show_profile', args=(stub,)))
```

CHAPTER 4

Indices and tables

- genindex
- modindex
- search