Memcached for Django

Courtney Meyerhofer
May 29, 2017 · 2 min read

A friend recently told me that premature optimization is the root of all evil. I set up caching for my most recent project because an API essential to the application rate limits at 200 requests per endpoint per day, and I was hitting that limit near daily. The information also isn't updated frequently, making it a great candidate for caching. On some pages, 4 AJAX calls are made when the page is loaded, and they took 4 seconds total to complete before setting up caching. They're now at about 20–100 ms each.

This is a guide for setting up Memcached for Django locally and in a production environment. Memcached is an open source caching system that has support for Python bindings through python-memcached and can run as a daemon.

System Setup

First, download the dependencies:

`brew install memcached` or `sudo apt-get install memcached`

`pip install python-memcached`

You can check memcached is on your system by running `memcached -vv`, and you'll see some server-like output. In order to run it as a daemon (I recommend):

`brew services start memcached` or `sudo service memcached start`

If you choose not to run it as a daemon, you'll need to manually start it in a dedicated terminal window by running `memcached`.

Django Setup

```
# settings.py
```

```
CACHES = {
    'default': {
        'BACKEND':
'django.core.cache.backends.memcached.MemcachedCache',
        'LOCATION': '127.0.0.1:11211',
    }
}
```

Memcached runs through localhost port 11211 by default, so there's no further configuration here. There are other options to have a dedicated Memcached server or have Memcached store files locally. Django documentation has more information on how to set that up.

Do the Thing

Since I'm interested in caching responses from an API, all of my cache-related code is inside of a view (what Django calls a controller, for other MVC folks). The strategy used below is to see if the data is already stored in the cache. If the data isn't there, perform the API request through the service and set the data in the cache with its expiration time and key in order to request it later.

```
...

from django.core.cache import cache

def my_view(request):
    cache_key = 'my_unique_key' # needs to be unique
    cache_time = 86400 # time in seconds for cache to be valid
    data = cache.get(cache_key) # returns None if no key-value pair

    if not data:
        my_service = Service()
        data = service.get_data()

    cache.set(cache_key, data, cache_time)

return JsonResponse(data, safe=False)

...
```

Python Django Caching Web Development Memcached