

# Crawlers Limiter Middleware

**Author :** Frederic De Zorzi  
**Contact :** [fredz@nospam.pimentech.fr](mailto:fredz@nospam.pimentech.fr)  
**Revision :** 1  
**Date :** 2009-05-29  
**Copyright :** This document has been placed in the public domain.  
**Tags :** django

## Status

In production.

## Abstract

This little middleware has been written to prevent system overload caused by aggressive crawlers on huge django websites. In conjunction with **memcached** cache, this system is really fast and could be extended for other purposes. Regular hosts like bots are configured in a white-list.

## Installation

- get PimenTech libcommonDjango :  
`svn checkout http://svn.pimentech.org/pimentech/libcommonDjango`
- install it with "make install"

## Activation and configuration

In your settings.py :

- **Activate memcached cache backend.** Remember that each django hit is logged and other cache systems would be too costly.

```
::: CACHE_BACKEND = 'memcached://127.0.0.1:11211/'
```

- the best is to put the middleware in first position of MIDDLEWARE\_CLASSES

```
MIDDLEWARE_CLASSES = (  
    'django.pimentech.middleware.crawler_limiter.CrawlerLimiterMiddleware',  
    'django.middleware.common.CommonMiddleware',  
  
    ...other middlewares...  
)
```

- configure the following parameters :

```
import re
```

```
CRAWL_WHITE_LIST = re.compile("127\.0\.0\.1|192\.168\.1\.*|66\.249\.65\.*|66\.249")  
CRAWL_CACHE_DURATION = 2
```

```
CRAWL_SITE_COUNT = 50
CRAWL_CACHE_BANNED_DURATION = 60 * 5
```

- CRAWL\_WHITE\_LIST : these ips are not concerned with the crawler limiter (66.249.65.\*, 66.249.66.\* : Google ; 74.6.8.\* : Yahoo)
- CRAWL\_CACHE\_DURATION : duration in seconds of cache per ip. For each django connection, the client ip is stored. If the client hits the site before the expiration, the ipcounter is incremented, with a new expiration of CRAWL\_CACHE\_DURATION. Otherwise the ip key expires from the cache with its counter.
- CRAWL\_SITE\_COUNT : if the ip counter reach this value, a mail is sent to the site admins, a 403 forbidden http response is returned, and the ip counter is incremented with the CRAWL\_CACHE\_BANNED\_DURATION.

## Source

```
# -*- coding: utf-8 -*-
from django.http import HttpResponseForbidden
from django.conf import settings
from django.core.mail import mail_admins
from django.core.cache import cache

import socket

class CrawlerLimiterMiddleware(object):
    """
    Forbids access to violent crawlers
    see http://garage.pimentech.net/libcommonDjango_django_pimentech_middleware_c
    for configuration instructions.
    """
    def process_request(self, request):
        ip = request.META['REMOTE_ADDR']
        if settings.CRAWL_WHITE_LIST.match(ip):
            return

        try:
            n = cache.get(ip)
        except ValueError:
            n = None
            mail_admins("Memcached error for host %s" % ip)
        if n is None:
            cache.set(ip, 1, settings.CRAWL_CACHE_DURATION)
        elif n >= settings.CRAWL_SITE_COUNT:
            cache.set(ip, n+1, settings.CRAWL_CACHE_BANNED_DURATION)
            if n == settings.CRAWL_SITE_COUNT:
                try:
                    host = socket.gethostbyaddr(ip)
                except:
                    host = 'Unknown'
                else:
                    host = host[0]
                mail_admins(' Bad crawler detected',
                    '%s hits limit reached for %s (host %s).\n'
                    'Forbidden response will be returned until it stops f
                    % (n, ip, host, settings.CRAWL_CACHE_BANNED_DURAT
            return HttpResponseForbidden('<h1>Forbidden</h1>')
        else:
            cache.set(ip, n+1, settings.CRAWL_CACHE_DURATION + n/10)
```

## Note

We would be please to receive your feedbacks, corrections and improvement suggestions. If you have Denial Of Service attacks problems, you could also try **mod\_evasive** apache module.