Django - Caching

To cache something is to save the result of an expensive calculation, so that you don’t perform it the next time you need it. Following is a pseudo code that explains how caching works −

```
given a URL, try finding that page in the cache

if the page is in the cache:
   return the cached page
else:
   generate the page
   save the generated page in the cache (for next time)
   return the generated page
```

Django comes with its own caching system that lets you save your dynamic pages, to avoid calculating them again when needed. The good point in Django Cache framework is that you can cache −

- The output of a specific view.
- A part of a template.
- Your entire site.

To use cache in Django, first thing to do is to set up where the cache will stay. The cache framework offers different possibilities - cache can be saved in database, on file system or directly in memory. Setting is done in the **settings.py** file of your project.

Setting Up Cache in Database

Just add the following in the project settings.py file −

```
CACHES = {
   'default': {
      'BACKEND': 'django.core.cache.backends.db.DatabaseCache',
      'LOCATION': 'my_table_name',
   }
}
```

For this to work and to complete the setting, we need to create the cache table 'my_table_name'. For this, you need to do the following −

```
python manage.py createcachetable
```

Setting Up Cache in File System

Just add the following in the project settings.py file −

```
CACHES = {
   'default': {
      'BACKEND': 'django.core.cache.backends.filebased.FileBasedCache',
      'LOCATION': '/var/tmp/django_cache',
   }
}
```

Setting Up Cache in Memory

This is the most efficient way of caching, to use it you can use one of the following options depending on the Python binding library you choose for the memory cache −

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

Or

```
CACHES = {
   'default': {
      'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
      'LOCATION': 'unix:/tmp/memcached.sock',
   }
}
```

Caching the Entire Site

The simplest way of using cache in Django is to cache the entire site. This is done by editing the MIDDLEWARE_CLASSES option in the project settings.py. The following need to be added to the option −

```
MIDDLEWARE_CLASSES += (
   'django.middleware.cache.UpdateCacheMiddleware',
   'django.middleware.common.CommonMiddleware',
   'django.middleware.cache.FetchFromCacheMiddleware',
)
```

Note that the order is important here, Update should come before Fetch middleware.

Then in the same file, you need to set −

```
CACHE_MIDDLEWARE_ALIAS – The cache alias to use for storage.
CACHE_MIDDLEWARE_SECONDS – The number of seconds each page should be cached.
```

Caching a View

If you don’t want to cache the entire site you can cache a specific view. This is done by using the **cache_page** decorator that comes with Django. Let us say we want to cache the result of the **viewArticles** view −

```
from django.views.decorators.cache import cache_page

@cache_page(60 * 15)

def viewArticles(request, year, month):
   text = "Displaying articles of : %s/%s"%(year, month)
   return HttpResponse(text)
```

As you can see **cache_page** takes the number of seconds you want the view result to be cached as parameter. In our example above, the result will be cached for 15 minutes.

Note − As we have seen before the above view was map to −

```
urlpatterns = patterns('myapp.views',
   url(r'^articles/(?P<month>\d{2})/(?P<year>\d{4})/', 'viewArticles', name = 'articles'),)
```

Since the URL is taking parameters, each different call will be cached separately. For example, request to /myapp/articles/02/2007 will be cached separately to /myapp/articles/03/2008.

Caching a view can also directly be done in the url.py file. Then the following has the same result as the above. Just edit your myapp/url.py file and change the related mapped URL (above) to be −

```
urlpatterns = patterns('myapp.views',
   url(r'^articles/(?P<month>\d{2})/(?P<year>\d{4})/',
   cache_page(60 * 15)('viewArticles'), name = 'articles'),)
```

And, of course, it's no longer needed in myapp/views.py.

Caching a Template Fragment

You can also cache parts of a template, this is done by using the **cache** tag. Let's take our **hello.html** template −

```
{% extends "main_template.html" %}
{% block title %}My Hello Page{% endblock %}
{% block content %}

Hello World!!!<p>Today is {{today}}</p>
We are
{% if today.day == 1 %}

the first day of month.
{% elif today == 30 %}

the last day of month.
{% else %}

I don't know.
{%endif%}

<p>
   {% for day in days_of_week %}
   {{day}}
</p>

{% endfor %}
{% endblock %}
```

And to cache the content block, our template will become −

```
{% load cache %}
{% extends "main_template.html" %}
{% block title %}My Hello Page{% endblock %}
{% cache 500 content %}
{% block content %}

Hello World!!!<p>Today is {{today}}</p>
We are
{% if today.day == 1 %}

the first day of month.
{% elif today == 30 %}

the last day of month.
{% else %}

I don't know.
{%endif%}

<p>
   {% for day in days_of_week %}
   {{day}}
</p>

{% endfor %}
{% endblock %}
{% endcache %}
```

As you can see above, the cache tag will take 2 parameters − the time you want the block to be cached (in seconds) and the name to be given to the cache fragment.

Revision #1
Created 14 December 2022 11:14:58 by Admin
Updated 14 December 2022 11:15:21 by Admin