缓存

> Django是动态Web后台框架，需要实时生成用户访问的页面，进行多次的数据库操作，但是多次的数据库访问操作对于整个Web系统来说，会影响效率，尤其是当访问量增大时，数据库的压力也会越来越大。
>
> 相对于磁盘及内存操作，数据库的访问操作付出的成本要大的多

> 浏览器第一次请求时，cache会缓存单个变量或整个网页等内容到硬盘或者内存中，同时设置response头部
>
> 当浏览器再次发起请求时，会与缓存中的过期时间相比较，如果缓存时间比较新，则会重新请求数据，并缓存起来然后返回response给客户端，如果缓存没有过期，则直接从缓存中提取数据，返回给response给客户端

Cache-Control

> HTTP协议头Cache-Control，Cache-Control与Expires的作用一致，都是指明当前资源的有效期，控制浏览器是否直接从浏览器缓存取数据还是重新发请求到服务器取数据。只不过Cache-Control的选择更多，设置更细致，如果同时设置的话，其优先级高于Expires

> 在python中使用memcached需要我们额外安装memcached作为memcache客户端的支持

```
pip3 install python-memcached -i https://pypi.tuna.tsinghua.edu.cn/simple
```

Cache设置

memcached

- 安装memcached

  ```
  apt-get install memcached # debian
  yum install memcached # centos
  ```

- 配置文件：/etc/memcached.conf

  > 配置文件中有两个可能需要修改的参数

  ```
  -m 64 #memcached所能使用的内存大小
  -l 127.0.0.1 #监听的IP地址
  ```

- 开启|关闭memcached服务

  ```
  systemctl start memcached # 开启
  systemctl stop memcached # 关闭
  ```

- 查看服务状态

  ```
  systemctl status memcached
  ```

settings配置

> 使用`memcached`缓存，首先需要在项目的`settings`文件下进行配置

```
CACHES = {
    'default': {
        'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
        # 指定缓存使用的引擎
        'LOCATION':'172.16.19.26:11211',
        # 指定缓存服务器地址，常为本机地址
    }
}
```

单独缓存

> 可以只为某些视图函数进行缓存
>
> 使用`django.views.decorators.cache`下的装饰器`cache_page`进行视图函数装饰即可

- 模型类的表代码

```
class People(models.Model):
    name = models.CharField(max_length=20,verbose_name='名字')
```

- 视图函数代码

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

@cache_page(10) # 缓存10秒
def index(request):
    print('视图函数被调用')
    ss = models.People.objects.all()
    return render(request, 'index.html', locals())
```

- 模板页面代码

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>展示数据</title>
</head>
<body>
    {% for s in ss %}
        <li>{{ s.name }} </li>
    {% endfor %}
</body>
</html>
```

> 在第一次刷新浏览器之后，

> 立即在数据库中添加一个新的数据对象
>
> 接着继续刷新浏览器，前端页面将会读取缓存中的结果，而不会显示刚才添加的用户

> 除了在视图函数上使用装饰器进行缓存设置，还可以在路由匹配部分使用相同装饰器函数进行设置

全站缓存

> 将整站所有视图设置缓存，需要在配置文件的中间件设置首尾部分添加如下内容

```
MIDDLEWARE = [
    'django.middleware.cache.UpdateCacheMiddleware', # 首部要添加的中间件
    # 将response缓存起来
    'django.middleware.security.SecurityMiddleware',
    ...
    'django.middleware.cache.FetchFromCacheMiddleware',# 尾部要添加的中间件
    # 将缓存的response取出来
]
```

> 以及搭配设置当前全站缓存有效时间的全局变量

```
CACHE_MIDDLEWARE_SECONDS = 10 # 每页页面缓存的秒数，默认为600
```

局部缓存

> 局部缓存主要为在模板页面，选择某个区域进行缓存，当用户再次访问相同页面时，如设置缓存未过期，则渲染时局部缓存不会重新生成

```
{% load cache %}
    局部缓存首先需要加载cache标签

{% cache sec key %}
{% endcache %}
```

> 以时间模板变量为例，做一个简单的测试，后台视图函数每次在访问时，返回当前时间

```
import time
now = time.strftime('%H:%M:%S', time.localtime())
```

> 模板页面在使用时的代码

```
{% load cache %}

<p>这里是未缓存的时间:{{ now }}</p>
{% cache 10 time %}
<p>这里是缓存的时间:{{ now }}</p>
{% endcache %}
```