Django自定义中间件及其实例应用

在《Django中间件》介绍了中间件的基概念以及它的执行与响应顺序还有它的作用，我们知道中间件的本质是一个 Python 类，它在视图执行之前与执行之后利用"钩子"函数做一些操作，也就是说所有的请求从开始进来以及最后返回响应都要经过中间件。中间件作为一个轻量级的插件系统，有很多应用场景。比如要通过某个 IP 在一分钟内访问网站的次数来限制一些网络爬虫，就可以通过中间件来实现。在本节我们将一起实现自定义中间件，以及了解它的实际应用。

1. 实现自定义中间的过程

1) 中间件的应用场景

根据之前的介绍，可以把中间件简单地理解为对视图中业务处理逻辑的封装。如果想对请求和响应对象做出修改，就可以考虑使用 Django 的中间件。例如：可以使用中间件对请求做出拦截，限制用户（可以从 HttpRequest 中拿到客户端的IP地址）的访问频率，例如 1 分钟内不允许访问 5 次。很显然，这需要在访问视图函数之前对用户做校验，因此可以利用 process_request 函数完成。

另外一种使用场景：比如说，，在 Web 开发中视图经常需要返回 JSON 数据。但是由于需求的不同，因此数据结构很难统一。此时，可以利用中间件对响应对象再做一层包装，统一数据结构，可以利用 process_response 函数完成。

2) 钩子函数执行过程

下面我们说一说中间件的钩子函数。中间件可以定义 5 个钩子函数，它们的名字是固定的。Django 在处理一个请求时，在调用视图函数之前，会依次从上到下处理 MIDDLEWARE 中声明的各个中间件，这其中会有两个钩子函数会被调用，分别是如下：

```
process_request
process_view
```

在处理响应时，调用视图函数之后，会依次从下到上（和调用正好相反），处理 MDDLEWARE中声明的各个中间件，这其中会有三个钩子函数被执行分别如下：

```
process_exception
process_template_response
process_response
```

五个钩子函数的具体作用介绍如下：

- def process_request(self, request): 在通过路由找到视图函数之前被调用，在每个请求上调用，返回 None 或 HttpResponse 对象；
- def process_view(self, request, view_func, view_args, view_kwargs): 调用视图之前被调用，每个请求上都会调用，返回 None 或 HttpResponse 对象；
- def process_response(self, request, response): 所有响应返回浏览器之前被调用，在每个请求上调用，返回必须是一个 HttpResponse 对象；
- def process_exception(self, request, exception): 当处理过程中抛出异常时调用，返回 None 或者 一个 HttpResponse 对象；
- def process_template_response(self, request, response): 视图函数刚好执行完毕后被调用，必须返回一个实现了 render 方法的响应对象；

当中间件的某个钩子函数返回 None 时，钩子函数会按顺序去执行其他中间件的该钩子函数，执行完毕之后才会进入下一个钩子函数，若当钩子函数返回了 HttpResponse 则该钩子函数不再去执行其他中间件的方法了，将直接进入该中间自己的下一个钩子函数，直到该中间件返回响应为止，才会执行其他中间件的钩子函数！

3)自定义中间件格式

理解了中间的概念以及钩子函数的执行顺序，实现自定义中间件就简单了，自定义中间件类须继承自以下类：

django.utils.deprecation.MiddlewareMixin

并在自定义类中实现适合的钩子函数。中间件类须实现五个钩子函数中的一个或者多个，通常中间件定义在 middleware 目录中，和项目应用属于同级，该目录需要我们自己新建。在目录中定义中间件文件 mymiddleware.py 并将其注册在 settings.py 文件的 MIDDLEWARE 列表中。自定义中间件的格式如下：

```
1. from django.http import HttpResponse
2. from django.utils.deprecation import MiddlewareMixin
3. class MyMiddleWare(MiddlewareMixin):
4.     def process_request(self, request):
5.         print("中间件方法 process_request 被调用")
6.
7.     def process_view(self, request, callback, callback_args, callback_kwargs):
```

```
        print("中间件方法 process_view 被调用")

    def process_response(self, request, response):
        print("中间件方法 process_response 被调用")
        return response

    def process_exception(self, request, exception):
        print("中间件方法 process_exception 被调用")

    def process_template_response(self, request, response):
        print("中间件方法 process_template_response 被调用")
        return response
```

2. 自定义中间件实例应用

1) 限制某IP访问网站次数

下面我们就通过中间件实现限制某个 IP 访问网站次数的功能，如果在 60s 内访问了 5 次就限制该 IP 的访问，代码如下所示：

```
import time
from django.http import HttpResponse
from django.utils.deprecation import MiddlewareMixin
# 限制用户访问次数,每60秒不超过5次
#构建访问者IP池
visit_ip_pool = {}#以'ip'地址为键，以访问的网站的时间戳列表作为值形如{'127.0.0.1':[时间戳,...]}
class VisitLimitMiddleWare(MiddlewareMixin):
    def process_request(self, request):
        #获取用户的访问的ip地址
        ip = request.META.get("REMOTE_ADDR")
        #获取访问时间
        visit_time = time.time()
        if ip not in visit_ip_pool:
            #维护字典,将新的ip地址加入字典
            visit_ip_pool[ip] = [visit_time]
        else:
            #已经存在，则将ip对应值的插入列表开始位置
            visit_ip_pool[ip].insert(0, visit_time)
        #获取ip_list列表
        ip_list = visit_ip_pool[ip]
        #计算访问时间差
        lead_time= ip_list[0] - ip_list[-1]
        print('地址:', ip, '访问次数:', len(ip_list), '时间差', lead_time)
        #两个条件同时成立则，间隔时间在60s内
        while ip_list and lead_time > 60:
            #默认移除列表中的最后一个元素
            ip_list.pop()
```

```
28.         #间隔在60s内判断列表的长度即访问的次数是否大于5次
29.         if len(ip_list) > 5:
30.             return HttpResponse("对不起，访问过于频繁，将终止你的访问请求...")
31.         print('地址:', ip, '访问次数:', len(ip_list), '时间差', lead_time)
```

上述代码完成了对用户访问网站次数的限制。定义完成后，需要在 settings.py 中的 MIDDLEWARE 列表完成注册，如下所示：

'middleware.mymiddleware.VisitLimitMiddleWare'

在运行项目时，中间件会自动加载执，你可以对原来的编写视图函数进行访问，你会发现在 CMD 命令行工具中，有如下显示：

```
System check identified no issues (0 silenced).
July 21, 2020 - 15:36:32
Django version 2.2.10, using settings 'BookStore.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
地址: 127.0.0.1 访问次数: 1 时间差 0.0
地址: 127.0.0.1 访问次数: 1 时间差 0.0
```

当你访问过于频繁，将终止你的访问请求。

3. 中间件使用时注意事项

上面我们完成了自定义中间件，当我们在配置和使用中间件的时候，也有两点需要注意：

- 第一是钩子函数的实现，由于其涉及的内容比较多，因此在接下来的内容中会有详细的介绍；
- 第二是经常提到的中间件的定义顺序。不可以随意更改中间件的定义顺序，因为它们之间可能存在着依赖关系。当你尝试修改它们之间的顺序，这时会发现系统报错了。比如会话中间件必须在用户身份认证之前。与此同时，Django 并没有规定一定需要中间件才能使项目正常工作，如果不需要，可以随时删减中间件。