9.django中间件

一. django中间件的使用场景

在http请求到达视图函数之前和视图函数return之后，django会根据自己的规则在合适的时机执行中间件中相应的方法.
使用场景:
1)如果你想修改请求，例如被传送到view中的HttpRequest对象或者修改view返回的HttpResponse对象，这些都可以通过中间件来实现.
2)可能你还想在view执行之前做一些操作，这种情况就可以用中间件来实现.
3)比如我们写一个判断浏览器信息，是pc还是手机，我们不想把这个逻辑加到视图函数里，想作为一个通用服务，作为一个可插拔的组件被使用，最好　的方法就是实现中间件.

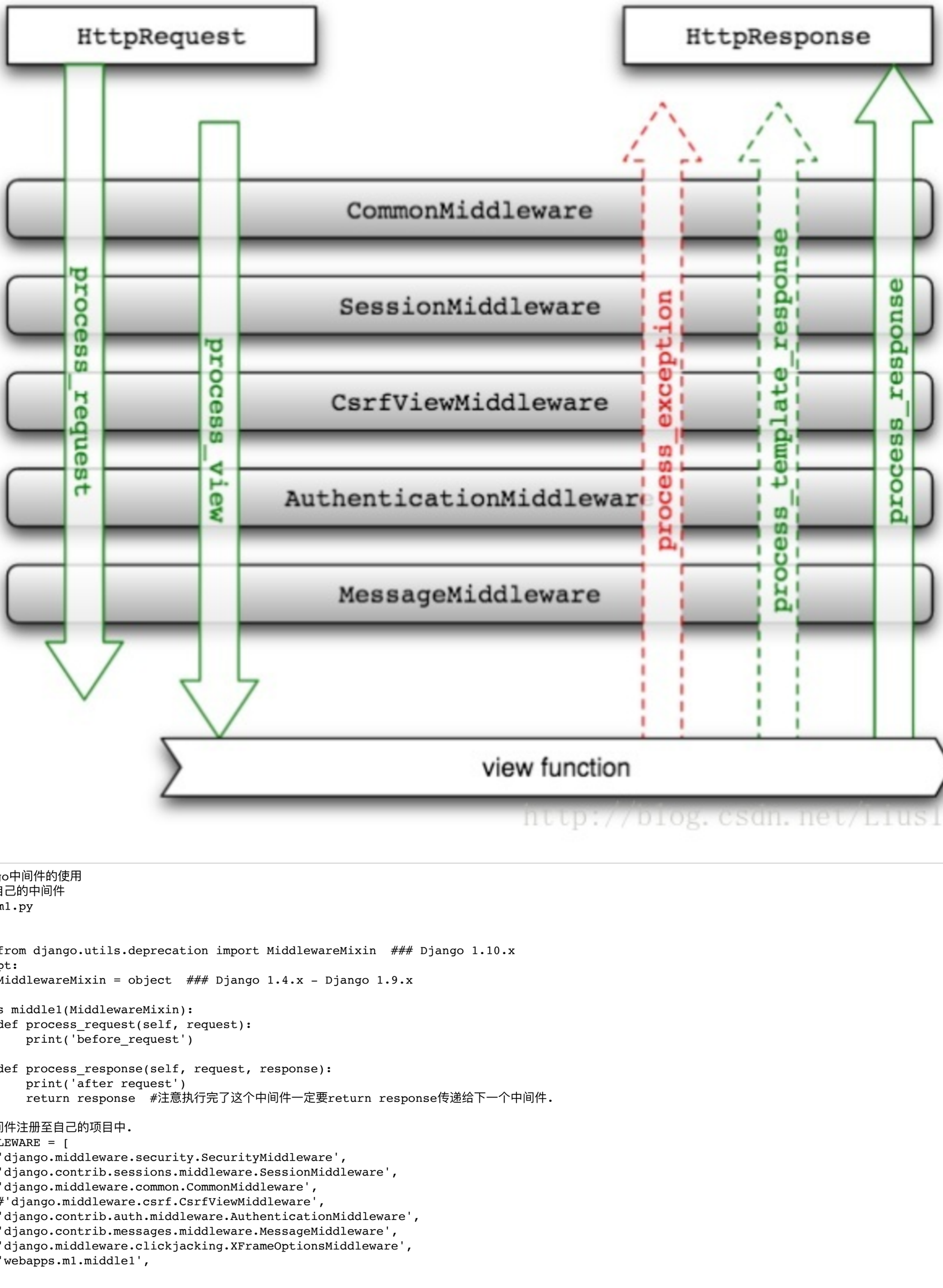
二. 中间件的使用方法
Django中间件必须是一个类，不需要继承任何类，并提供五个接口:
1. process_request(self,request)
　　每一个请求都是先通过中间件中的process_request函数，这个函数返回None或者HttpResponse对象，如果返回前者则继续处理其它中间件，如果返回一个HttpResponse就处理中止，返回到网页上.

2. process_response(self, request, response)
　　请求离开视图函数views之后所执行的内容，该函数返回response，因为要把http响应发给下一个中间件处理.

3. process_view(self, request, callback, callback_args, callback_kwargs)
　　执行完所有中间件的request方法，url匹配成功，拿到视图函数的名称、参数(注意不执行)，再执行process_view()方法，最后去执行视图函数.

4. process_exception(self, request, exception)
　　执行完所有process_request方法，再执行所有process_view方法，　如果视图函数出错则执行process_exception(最终response，process_exception的return值)，如果process_exception方法有了返回值就不再执行其他中间件的process_exception，直接执行response方法响应，执行所有response方法,最后返回process_exception的返回值.

5. process_template_response(self,request,response) #较少用
　　只有在视图函数的返回对象中有render方法才会执行.



三. django中间件的使用
3.1 编写自己的中间件
```
vim m1.py

try:
    from django.utils.deprecation import MiddlewareMixin  ### Django 1.10.x
except:
    MiddlewareMixin = object  ### Django 1.4.x - Django 1.9.x

class middle1(MiddlewareMixin):
    def process_request(self, request):
        print('before_request')

    def process_response(self, request, response):
        print('after request')
        return response  #注意执行完了这个中间件一定要return response传递给下一个中间件.
```

3.2 将中间件注册至自己的项目中.
```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    #'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
    'webapps.m1.middle1',
]
```

3.3 测试中间件，随便请求一个项目url，在django的工作日志中可看到如下内容.

```
January 31, 2019 - 09:55:15
Django version 2.1.5, using settings 'webapps.settings'
Starting development server at http://0.0.0.0:8899/
Quit the server with CONTROL-C.
before_request  #在请求开始之前执行 process_request中的代码
this is view home  #视图函数中的代码
after request #视图函数执行完成后, 离开视图函数到达客户端之前执行的代码.
[31/Jan/2019 09:55:17] "GET /home/ HTTP/1.1" 200 56
```

四. 继续使用其他中间件.

4.1 在上述中间件中增加以下两个方法:

```
def process_view(self, request, callback, callback_args, callback_kwargs):
    print('this is process_view')

def process_exception(self, request, exception):
    print('this process_exception')
```

4.2 请求项目中的一个url, 会得到如下输出.

```
January 31, 2019 - 10:01:44
Django version 2.1.5, using settings 'webapps.settings'
Starting development server at http://0.0.0.0:8899/
Quit the server with CONTROL-C.
before_request
this is process_view  #执行完所有的request, 在请求到达views函数之前, 执行process_view
this is view home
after request
[31/Jan/2019 10:01:46] "GET /home/ HTTP/1.1" 200 56
```

4.3 将视图函数人为抛出异常, 再次请求该视图, 得到以下输出.

```
January 31, 2019 - 10:03:30
Django version 2.1.5, using settings 'webapps.settings'
Starting development server at http://0.0.0.0:8899/
Quit the server with CONTROL-C.
before_request
this is process_view
this is view home
this process_exception  #代码发生错误时才执行 process_exception下的代码, 代码正常执行时不运行此处代码.
Internal Server Error: /home/
Traceback (most recent call last):
  File "/app_shell/python/webapps/lib/python3.6/site-packages/django/core/handlers/exception.py", line 34, in inner
    response = get_response(request)
  File "/app_shell/python/webapps/lib/python3.6/site-packages/django/core/handlers/base.py", line 126, in _get_response
    response = self.process_exception_by_middleware(e, request)
  File "/app_shell/python/webapps/lib/python3.6/site-packages/django/core/handlers/base.py", line 124, in _get_response
    response = wrapped_callback(request, *callback_args, **callback_kwargs)
  File "/app_shell/python/webapps/webapps/login/views.py", line 27, in home
    print(abc)
NameError: name 'abc' is not defined
after request
[31/Jan/2019 10:03:32] "GET /home/ HTTP/1.1" 500 63189
```

五. 中间件的使用场景

1做IP限制: 放在中间件类的列表中阻止某些IP访问了.

2) URL访问过滤: 如果用户访问的是login视图则放过, 如果访问其他视图(需要检测是不是有session已经存在登录状态, 如果没有则返回login), 这样就省得在多个视图函数上写装饰器了.

3) 缓存 (还记得CDN吗?)

客户端请求来了, 中间件去缓存看看有没有数据, 有直接返回给用户, 没有再去逻辑层 执行视图函数