```
# 中间件
    #介于request与response处理之间的一道处理过程，相对比较轻量级
    #在全局上改变django的输入与输出

  process_request(self, request)
    return response  # 不往后走，直接回去,被中间件直接挡回，返回response
    return none      # 继续走

   process_response(self, request, response)
    return response    #必须return，否则报错

  '''中间件方法'''
    *process_request(self,request)
    *process_response(self, request, response)

    # 几乎不用
    # process_view(self, request, callback, callback_args, callback_kwargs)
    # process_template_response(self,request,response)
    # process_exception(self, request, exception)

  '''自定义中间件'''
      #视图函数
      def index(request):
        print("view函数...")
        return HttpResponse("OK")

      #自定义的中间件
      from django.utils.deprecation import MiddlewareMixin  # 重要
      from django.shortcuts import HttpResponse

      class Md1(MiddlewareMixin):

        def process_request(self,request):
          print("Md1请求")

        def process_response(self,request,response):
          print("Md1返回")
          return response

      class Md2(MiddlewareMixin):

        def process_request(self,request):
          print("Md2请求")
          #return HttpResponse("Md2中断")
        def process_response(self,request,response):
          print("Md2返回")
          return response

      # 中间件注册
        '类的路径'

       #结果:
        Md1请求
        Md2请求
        view函数...
        Md2返回
        Md1返回
```

```
        #  return HttpResponse("Md2中断") 结果:
           Md1请求
           Md2请求
           Md2返回
           Md1返回

'''了解Django生命周期，此处可以较快理解 '''
    '''process_rquest 总结'''
        # 中间件的process_request方法是在执行视图函数之前执行的
        #
        当配置多个中间件时，会按照MIDDLEWARE中的注册顺序，也就是列表的索引值，从前到后依次执行
        # 不同中间件之间传递的request都是同一个对象

    '''process_response 总结'''
        # 按照MIDDLEWARE中的注册顺序倒序执行
        # 不同中间件之间传递的request都是同一个对象
```