Python_采用协程的责任链模式

```
#coding=utf8

import collections
from functools import wraps

ALL_EVENTS = "click","rightmousedown","dbclick","leftmousedown"
EVENT_DEF = collections.namedtuple("event",ALL_EVENTS)
EVENTS = EVENT_DEF(*ALL_EVENTS)

#定义一个事件发生器
def Events():
    index = 0
    while(True):
        index += 1
        yield ALL_EVENTS[index%(len(ALL_EVENTS))]

def coroutine(generator):
    @wraps(generator) #特别注意：如果没有functools.wraps，包装之后函数名字会改变
    def gen(*args,**kwargs):
        g = generator(*args,**kwargs)
        next(g)
        return g
    return gen


#定义事件处理链
@coroutine
def clickHandler(successor = None):
    while True:
        event = yield
        if event == EVENTS.click: #每个判断是否是自己能处理的类型
            print("Event {} handled by {}".format(event,clickHandler.__name__))
        elif successor: #不是就看有没有后继，传给后继进行处理
            successor.send(event)
@coroutine
def DBClickHandler(successor = None):
    while True:
        event = yield
        if event == EVENTS.dbclick:
            print("Event {} handled by {}".format(event,DBClickHandler.__name__))
        elif successor:
            successor.send(event)
@coroutine
def lefMouseDownHandler(successor = None):
    while True:
        event = yield
        if event == EVENTS.leftmousedown:
            print("Event {} handled by {}".format(event,lefMouseDownHandler.__name__))
        elif successor:
            successor.send(event)
@coroutine
def rightMouseDownHandler(successor = None):
    while True:
        event = yield
        if event == EVENTS.rightmousedown:
            print("Event {} handled by {}".format(event,rightMouseDownHandler.__name__))
        elif successor:
            successor.send(event)



if __name__=="__main__":
    processor = clickHandler(DBClickHandler(lefMouseDownHandler(rightMouseDownHandler())))
```

```
events = Events()
for _ in range(10):
    event =  next(events)
    processor.send(event)
```