python_设计模式——责任链模式

```
#coding=utf8
import collections

#创建事件的枚举类型——注意创建枚举类型的技巧
ALL_EVENTS = "click","rightmousedown","dbclick","leftmousedown"
EVENT_DEF = collections.namedtuple("event",ALL_EVENTS)
EVENTS = EVENT_DEF(*ALL_EVENTS)

#事件基类
class Event(str):
    __slots__ =()

#动态创建类的new方法
def makeNewMethod(clsName):
    def new(cls):
        return Event.__new__(cls,clsName)
    return new

#动态创建定义的事件类
for clsName in  ALL_EVENTS:
    _new = makeNewMethod(clsName)
    cls = type(clsName,(Event,),dict(__slots__=(),__new__ = _new))
    globals()[clsName] = cls

#采用责任链模式动态处理事件
class ChainObject(object):

    def __init__(self,successor=None):
        self.successor = successor

    def handle(self,event):
        assert(isinstance(event,Event)) #之所以创建事件类是为了便于我们判断是不是事件
        if self.successor:
            self.successor.handle(event)
        else:
            print("Drop event {} for no handler".format(event))

class ClickHandler(ChainObject):

    def handle(self,event):
        if event== EVENTS.click:
            print("{} Event handled by:{}".format(event,ClickHandler.__name__))
        else:
            super().handle(event)

class RightmousedownHandler(ChainObject):

    def handle(self,event):
        if event== EVENTS.rightmousedown:
            print("{} Event handled by:{}".format(event,RightmousedownHandler.__name__))
        else:
            super().handle(event)

if __name__=="__main__":
    #handler Chain
    proessor = ClickHandler(RightmousedownHandler())
    clickEvent = globals()[EVENTS.click]()
    rightmousedownEvent = globals()[EVENTS.rightmousedown]()
    leftmousedownEvent = globals()[EVENTS.leftmousedown]()
    proessor.handle(clickEvent)
    proessor.handle(rightmousedownEvent)
    proessor.handle(leftmousedownEvent)
```