README

PublisherSubscriber

说明

一个发布者/订阅者模式的python实现

使用

1. 将MessageController导入你自己的python源码
   `from MessageController import MessageController`
2. 定义消息响应事件

```
def response(info):
    print("response my message whit info %s" % info)
```

3. 注册消息

```
MessageController.registerSubscriberFunc("message1", response)
```

4. 在需要发送消息的时机发送消息

```
MessageController.publishMessage("message1", {"test1": "1", "ts": "ts"})
MessageController.publishMessage("message1")
```

5. 可以看到输出

```
/MessageController.py
response my message whit info {'test1': '1', 'ts': 'ts'}
response my message whit info None
bash-3.2$
```

另一种使用方式

1. 将Task和MessageController导入你自己的python源码
   `from Task import Task`
   `from MessageController import MessageController`
2. 重定义响应任务，继承Task，重写响应事件respose

```
class CustomTask(Task):
    def respone(self, info):
        print("This is My CustomTask %s" % info)
```

3. 注册消息

```
task2 = CustomTask()
MessageController.registerSubscriber("message2", task2)
```

4. 在需要发送消息的时机发送消息

```
MessageController.publishMessage("message2", {"test1": "1", "ts": "ts"})
MessageController.publishMessage("message2")
```

5. 可以看到输出

```
This is My CustomTask {'test1': '1', 'ts': 'ts'}
This is My CustomTask None
```

其他说明

1. 注册一个对所有消息都响应的任务

- 调用MessageController.registerDefaultSubscriber(task)

```
class CustomTask1(Task):
    def respone(self, info):
        print("This is My CustomTask1 %s" % info)
task3 = CustomTask1()
MessageController.registerDefaultSubscriber(task3)
MessageController.publishMessage("message3", {"test1": "1", "ts": "ts"})
MessageController.publishMessage("message4")
```

2. 注册一个对所有消息都响应的事件

- 或调用MessageController.registerDefaultSubscriberFunc(response)

```
def response2(info):
    print("another response my message whit info %s" % info)

MessageController.registerDefaultSubscriberFunc(response2)

MessageController.publishMessage("message5", {"test1": "1", "ts": "ts"})
MessageController.publishMessage("message6")
```

2. 发送一个所有事件或任务都响应的消息，调用MessageController.publishToAll()

```
MessageController.publishToAll()
```

3. 一个事件响应多个消息

```
def response3(info):
    print("the third response my message whit info %s" % info)

MessageController.registerSubscriberFunc("message7", response3)
MessageController.registerSubscriberFunc("message8", response3)

MessageController.publishMessage("message7", {"test1": "1", "ts": "ts"})
MessageController.publishMessage("message8", {"test2": "1", "ts3": "ts"})
```

4. 一个任务响应多个消息

```
class CustomTask2(Task):
    def respone(self, info):
        print("This is My CustomTask2 %s" % info)
task4 = CustomTask2()
MessageController.registerSubscriber("message9", task4)
MessageController.registerSubscriber("message10", task4)
MessageController.publishMessage("message9", {"test1": "1", "ts": "ts"})
MessageController.publishMessage("message10", {"test2": "1", "ts3": "ts"})
```

5. 一个消息发送给多个事件或多个任务

```
class CustomTask3(Task):
    def respone(self, info):
        print("This is My CustomTask3 %s" % info)

class CustomTask4(Task):
    def respone(self, info):
        print("This is My CustomTask4 %s" % info)

task5 = CustomTask3()
task6 = CustomTask4()
MessageController.registerSubscriber("message11", task5)
MessageController.registerSubscriber("message11", task6)

def response4(info):
    print("the fourth response my message whit info %s" % info)

def response5(info):
    print("the fifth response my message whit info %s" % info)

MessageController.registerSubscriberFunc("message11", response4)
MessageController.registerSubscriberFunc("message11", response5)

MessageController.publishMessage("message11", {"test2": "1", "ts3": "ts"})
```