Atitit 提升扩展性 简化与缩短调用链 提升扩展性 MQ 观察者回调模式

解耦

**点：** 1、观察者和被观察者是抽象耦合的。 2、建立一套触发机制。

目录

[1.1. regConsumer注册监听 1](#_Toc11606)

[1.2. 发送消息 1](#_Toc29315)

[1.3. 工具类消息中心ObserverUtil 2](#_Toc29860)

使用jdk8的消费者函数式接口即可。。。

## regConsumer注册监听

**private** **static** **void** regConsumer() {

// reg consumer

ObserverUtil.*centers*.put("loginEvtSubject", **new** ArrayList<>());

ObserverUtil.*centers*.get("loginEvtSubject").add((Object msg) -> {

System.***out***.println(" show consumer::" + msg);

});

ObserverUtil.*centers*.get("loginEvtSubject").add((Object msg) -> {

System.***out***.println(" filelog consumer::" + msg);

});

}

## 发送消息

List li = **new** ArrayList<>();

li.add(svr2);

li.add(svr1);

ObserverUtil.*sendSubject*("loginEvtSubject", svr1);

ObserverUtil.*sendSubject*("loginEvtSubject", svr2);

## 工具类消息中心ObserverUtil

**package** zuulx;

**import** java.util.HashMap;

**import** java.util.List;

**import** java.util.Map;

**import** java.util.function.Consumer;

**import** com.netflix.loadbalancer.Server;

**public** **class** ObserverUtil {

**public** **static** Map<String, List<Consumer>> *centers* = **new** HashMap();

**public** **static** **void** sendSubject(String msgSubject, Object msg) {

ObserverUtil.*centers*.get(msgSubject).forEach((msgConsumer) -> {

msgConsumer.accept(msg);

});

}

}