https://www.cnblogs.com/mushroom/p/4470006.html

[Redis系列(三)-Redis发布订阅及客户端编程](https://www.cnblogs.com/mushroom/p/4470006.html)

**阅读目录**

1. [发布订阅模型](https://www.cnblogs.com/mushroom/p/4470006.html#one)
2. [Redis中的发布订阅](https://www.cnblogs.com/mushroom/p/4470006.html#two)
3. [客户端编程示例](https://www.cnblogs.com/mushroom/p/4470006.html#three)
4. [0.3版本Hredis](https://www.cnblogs.com/mushroom/p/4470006.html#four)

**发布订阅模型**

在应用级其作用是为了减少依赖关系，通常也叫观察者模式。主要是把耦合点单独抽离出来作为第三方，隔离易变化的发送方和接收方。

发送方：只负责向第三方发送消息。(杂志社把读者杂志交给邮局)  
接收方：被动接收消息。(1：向邮局订阅读者杂志，2：门口去接邮过来的杂志)  
第三方作用是：存储订阅杂志的接收方，并在杂志过来时送给接收方。 (邮局)

C#示例，发送方把杂志放到邮局里面：

if (QA.AddBug())

EmailNotify();

接收方到邮局登记地址，有杂志过来时送货上门：

EmailNotify += () => { Console.WriteLine("A君"); };

EmailNotify += () => { Console.WriteLine("B君"); };

第三方邮局接受读者杂志订阅，收到杂志时进行派送：

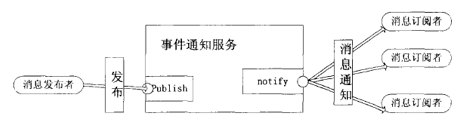
public delegate void MessageHandler();

public static event MessageHandler EmailNotify;

if (QA.AddBug())

EmailNotify();

当我们把观察者模式放大到系统级时，就是发布订阅(pub/sub)了。 主要是用来降低发布者和订阅者的耦合，提高前端系统吞吐量。结构如图：



**Redis中的发布订阅**

Redis实现完整的发布订阅范式，就是说任何一台redis服务器，启动后都可以当做发布订阅服务器。

**普通订阅**

启动订阅者client。

redis-cli.exe -h 127.0.0.1 -p 6379

订阅bar频道。格式：SUBSCRIBE name1 name2。   
成功订阅回复，分别对应订阅类型、订阅频道、订阅数量。

127.0.0.1:6379> SUBSCRIBE bar

Reading messages... (press Ctrl-C to quit)

1) "subscribe"

2) "bar"

3) (integer) 1

新起个发布者client，发送消息。格式：publish channelName Message。

127.0.0.1:6379> publish bar val

(integer) 1

订阅client回复，分别对应消息类型，频道，消息。

1) "message"

2) "bar"

3) "val"

图例



**模式订阅**

Redis支持模式匹配订阅，\*为模糊匹配符。  
订阅所有频道的消息

PSUBSCRIBE \*

订阅以news.开头的所有频道。

PSUBSCRIBE news.\*

**取消订阅**

取消普通订阅和取消模式订阅的命令。

UNSUBSCRIBE bar

PUNSUBSCRIBE ba\*

取消在官方提供的连接工具中无法模拟的。

**查看订阅信息**

查看订阅消息是redis在2.8中心增加的命令之一。

**pubsub channels [pattern] 。**

返回当前服务器被订阅的所有频道。

127.0.0.1:6379> pubsub channels

1) "bar"

指定匹配参数，返回与模式匹配的所有频道。

127.0.0.1:6379> pubsub channels ba\*

1) "bar"

**pubsub numsub [channel-1 channel-2 ...channel-n]**

接受任意多个频道作为输入参数，返回这些频道的订阅者数量。

127.0.0.1:6379> pubsub numsub bar bar2

1) "bar"

2) (integer) 1

3) "bar2"

4) (integer) 0

**客户端编程示例**

[复制代码](javascript:void(0);)

RedisPubSub client = new RedisPubSub("127.0.0.1", 6381);

client.OnUnSubscribe += (obj) => {

Console.WriteLine();

};

client.OnMessage = (sender, arcgs) =>{

Console.WriteLine(arcgs);

};

client.OnError = (Exception) => {

Console.WriteLine(Exception.Message);

};

client.Subscribe("bar");

Console.ReadLine();

[复制代码](javascript:void(0);)

**0.3版本HRedis**

**基本使用**

using (RedisClient client = new RedisClient("127.0.0.1", 6381))

{

client.Set("key", "value");

client.Get("key");

}

**使用连接池，自动回收连接。**

PoolRedisClient prc = new PoolRedisClient(new PoolConfiguration());

prc.Single.Set("key", "value");

prc.Single.Get("key");

**及上面的订阅。**

开源地址 <https://github.com/mushroomsir/HRedis>