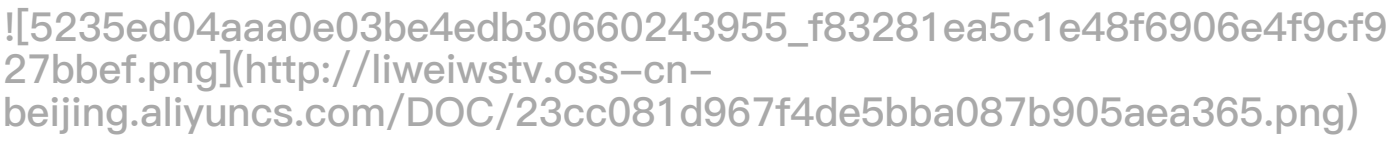
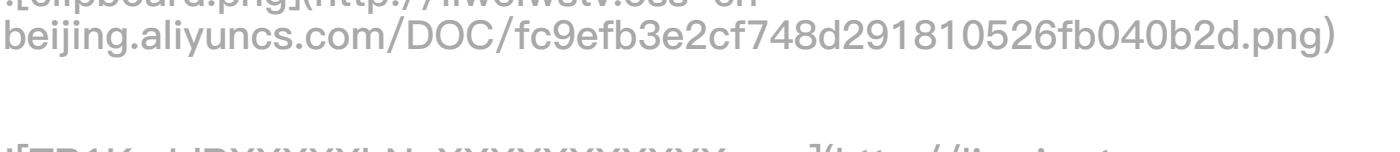
ActiveMQ之MQTT实现即时聊天

==========





MQTT简介（[mqtt.org/](http://cxyroad.com/ "http://mqtt.org/")）

[docs.oasis-open.org/mqtt/mqtt/v...](http://cxyroad.com/ "http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html")

MQTT协议中文文档（[mcxiaoke.gitbooks.io/mqtt-cn/con...](http://cxyroad.com/ "https://mcxiaoke.gitbooks.io/mqtt-cn/content/%EF%BC%89")

MQTT是基于二进制消息的发布/订阅编程模式的消息协议（MQTT 协议是建立在 TCP 协议之上的），最早由IBM提出的，如今已经成为OASIS规范。由于规范很简单，非常适合需要低功耗和网络带宽有限的IoT场景，比如：

```
- 遥感数据
- 汽车
- 智能家居
```

– 智慧城市

– 医疗医护

```

详细介绍请移步至：[dataguild.org/?p=6817](http://cxyroad.com/”http://dataguild.org/?p=6817”)

mqtt 能做什么？（物联网、即时通讯还可以用到其他适应的场景）

mqtt 简单来讲就是一个协议 是基于tcp之上的协议， 现在好多消息中间件都支持 mqtt协议 （mosquitto、activemq...等）

那么问题来了，为什么要在 TCP 协议之上再封装一层 MQTT 协议呢?

就好比汽车地盘上有四个轮胎就能跑了，但是想要大家开车更舒服，我们还是得给底盘加个壳。

举个例子，如果你要用 TCP 协议做一个即时聊天室，那么你：

```

1. 首先写一个服务器程序，监听某个端口，这样客户端就能连接上来了。然后大家就能向你的服务器发送数据了。

2. 但是你不希望随便谁都能连接你的服务器并且往你的服务器里发东西，于是你写了一个注册页面，让别人先去注册账号，然后他们在客户端创建连接时发送的第一个TCP报文必须包含了账号密码。这样当你收到第一个报文之后，你就能判断这个连接是否合法了。

3. 但是发过来的账号密码，具体摆在报文什么位置，也得事先约定好吧，不然怎么知道哪几位是账号哪几位是密码，于是你要求：第一个字节为1，用来告诉你这是一个请求连接的报文。然后第二个字节是报文剩下的长度（这个没啥异议，粘包处理必备），然后是“饼干熊最帅”这样一个固定的字符串（没啥意思，就是开心），后面紧跟着就是账号，然后是密码。但是账号有多长呢，所以账号第一个字节是账号的长度，剩下才是账号内容，这样就解决了“账号有多长”的问题了，密码同理。

4. 如果账号密码不匹配就断开连接并且返回一句“xxxx off”，如果创建连接后

半天不发送任何东西也断开连接。

5. 现在大家的客户端都连接上来了，你在服务器保存了一个`账号:socket`的map，能通过任何一个账号找到这个人的socket并向他发送信息。接下来大家要开始聊天了。

6. 在聊天室中，一个人发送的消息其他人都能收到，所以你添加了聊天室的概念，用户首先去你的web页面创建聊天室，然后会得到一个聊天室的ID。然后用户要加入聊天室，必须先发送一个加入聊天室的报文。然后你要求报文第一个字节为2代表加入聊天室的请求报文，然后是报文剩余长度，然后是聊天室ID，聊天室ID有多长？我们还是用第一个字节代表长度，剩下的内容为具体ID的形式来搞定。

7. 现在有多个用户加入了聊天室，其中每个用户又都加入了多个聊天室。假设现在大家在聊天室A中开始聊天。用户甲向服务器发送消息，然后你发现你不知道这条消息是请求连接还是要加入聊天室还是干啥，所以发送消息也应该定义一种报文类型。于是你要求发送消息的报文的第一个字节为3，那么这个报文是发送到哪个聊天室的呢？于是你要求报文后面跟上聊天室ID，最后是具体的消息内容。

8. 服务器收到第一个字节为3的消息，就知道这是一个聊天消息。然后根据上面带的聊天室ID以及你在服务器存储的用户和聊天室的关系，找到了这个聊天室里的所有人，然后你就把消息发给这里面的所有人了。

```

上面这个例子，在连接服务器（检查合法性、断开连接）、订阅主题（加入聊天室）、发布消息这些过程中，你约定的报文格式和设计的服务器处理逻辑就是 MQTT 协议的内容，当然我举的例子非常粗糙

真正的MQTT协议要求的处理逻辑和报文格式都完善很多，但是协议本身还是很简单的，具体内容去看 MQTT 的文档吧。

文章底部会附上关于MQTT的一些文档链接。

要了解MQTT需要了解：jms规范的（发布/订阅模式）

发布/订阅模式

与请求/回答这种同步模式不同，发布/订阅模式解耦了发布消息的客户（发布者）与订阅消息的客户（订阅者）之间的关系，这意味着发布者和订阅者之间并不需要直接建立联系。打个比方，你打电话给朋友，一直要等到朋友接电话
```

了才能够开始交流，是一个典型的同步请求/回答的场景；而给一个好友邮件列表发电子邮件就不一样，你发好电子邮件该干嘛干嘛，好友们到有空了去查看邮件就是了，是一个典型的异步发布/订阅的场景。

熟悉编程的同学一定非常熟悉这种设计模式了，因为它带来了这些好处：

```
– 发布者与订阅者不必了解彼此，只要认识同一个消息代理即可。

– 发布者和订阅者不需要交互，发布者无需等待订阅者确认而导致锁定。

– 发布者和订阅者不需要同时在线，可以自由选择时间来消费消息。
```

具体通讯详情请移步至
：[blog.hyaroma.com/articles/20…](http://cxyroad.com/”https://blog.hyaroma.com/articles/2016/08/31/1472612162615.html”)

第一个链接中笔者讲到了使用 mosquitto 来实现mqtt协议 ， 下文我就围绕activemq来做实践操作了。

activemq 简单介绍和安装使用请移步至：

activemq 入门介绍
：[blog.hyaroma.com/articles/20…](http://cxyroad.com/”https://blog.hyaroma.com/articles/2016/08/31/1472609577131.html”)

activemq 安装与使用
：[blog.hyaroma.com/articles/20…](http://cxyroad.com/”https://blog.hyaroma.com/articles/2016/08/31/1472610347453.html”)

在activemq5.9 之后就有mqtt 和 websoket的支持了。搭建好activemq之后

vim /usr/soft/activemq/conf/activemq.xml

找到：transportConnectors 节点 注意：“mqtt” 和 “ws”

mqtt(Message Queuing Telemetry Transport)

ws(WebSocket)

mqtt（java c .net） 则使用mqtt协议进行链接即可

ws 主要针对 html5前端 利用websocket链接来和后台进行通讯。

两者的应用场景不一样：

```
– MQTT是为了物联网场景设计的基于TCP的Pub/Sub协议，有许多为物联网优化的特性，比如适应不同网络的QoS、层级主题、遗言等等。

– WebSocket是为了HTML5应用方便与服务器双向通讯而设计的协议，HTTP握手然后转TCP协议，用于取代之前的Server Push、Comet、长轮询等老旧实现。
```

两者之所有有交集，是因为一个应用场景：如何通过HTML5应用来作为MQTT的客户端，以便接受设备消息或者向设备发送信息，那么MQTT over WebSocket自然成了最合理的途径了。

activemq 也提供了对websocket配置方式的说明

MQ 提供的 MQTT 服务严格遵循 MQTT3.1.1 协议设计，理论上能够适配所有的 MQTT 客户端，但不排除部分客户端存在细节上的兼容性问题。针对 MQTT 用户常用的平台，推荐对应的三方包如下（paho 客户端）：

使用平台推荐的第三方 SDK相关链接

使用平台	推荐的第三方 SDK	相关链接
Java	Eclipse Paho SDK	[www.eclipse.org/paho/client...](http://cxyroad.com/”http://www.eclipse.org/paho/clients/java/”)
iOS	MQTT-Client-Framework	[github.com/ckrey/MQTT-...](http://cxyroad.com/ ”https://github.com/ckrey/MQTT-Client-Framework”)
Android	Eclipse Paho SDK	[github.com/eclipse/pah...](http://cxyroad.com/”https://github.com/eclipse/paho.mqtt.android”)
JavaScript	Eclipse Paho JavaScript	[www.eclipse.org/paho/client...](http://cxyroad.com/”http://www.eclipse.org/paho/clients/js/”)
Python	Eclipse Paho Python SDK	[pypi.python.org/pypi/paho-m...](http://cxyroad.com/ ”https://pypi.python.org/pypi/paho-mqtt/”)
C	Eclipse Paho C SDK	[eclipse.org/paho/client...](http://cxyroad.com/”https://eclipse.org/paho/clients/c/”)
C#	Eclipse Paho C# SDK	[github.com/eclipse/pah...](http://cxyroad.com/”https://github.com/eclipse/paho.mqtt.m2mqtt”)

JavaScript 就使用的是websoket方式

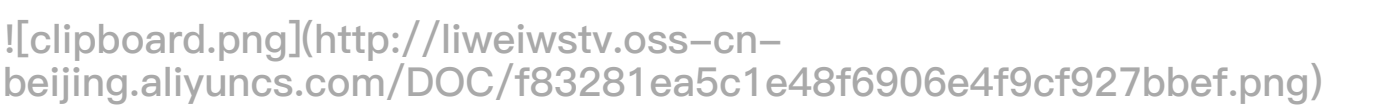
其他语言的客户端 SDK 如 PHP 等暂时没有提供测试。如有需要可以访问 [www.eclipse.org/paho/downlo...](http://cxyroad.com/”http://www.eclipse.org/paho/downloads.php”) 进行下载。

先拿前端（js）进行测试，下载官方包：

![clipboard.png](http://liweiwstv.oss-cn-beijing.aliyuncs.com/DOC/7953332d3acc4ef78b1249e01efb9507.png)

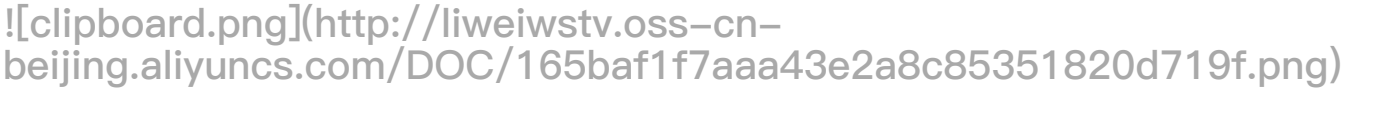
下载下来之后 运行包下面的：utility/index.html

![clipboard.png](http://liweiwstv.oss-cn-beijing.aliyuncs.com/DOC/3e459fcbf165424385f659463912fa2e.png)



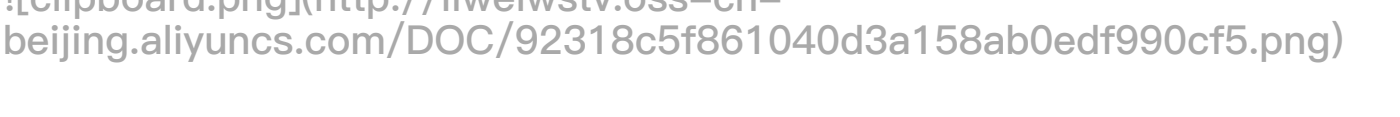
> 原文链接 [www.hanyuanhun.cn](http://cxyroad.com/”http://www.hanyuanhun.cn”) | node.hanyuanhun.cn

注意观察activemq connection（来自websocket的链接）



再看看topic主题：

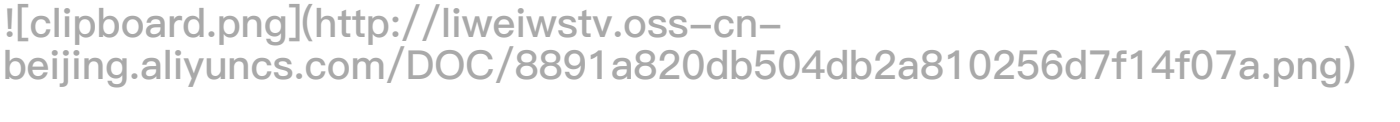
自定义js使用websocket链接 请参见对应github demo



上面步骤属于是 前端和前端的一个通讯方式 包括：发布topic消息和订阅topic消息 接下来是使用java 客户端进行 topic的订阅与发布

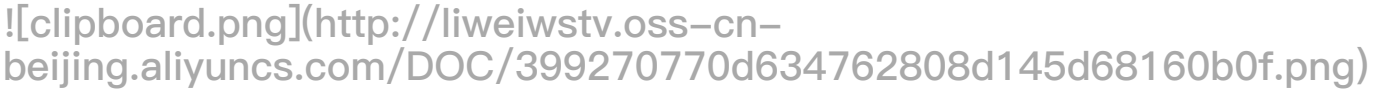
实现 java客户端与 前端js 进行实时聊天功能

下载paho for java：



由于官方太多繁琐，我单独抽取了demo进行交互操作，可以去这里下载简单的demo：

[github.com/liweigithub...](http://cxyroad.com/”https://github.com/liweigithub/paho-mqtt-java.git”)



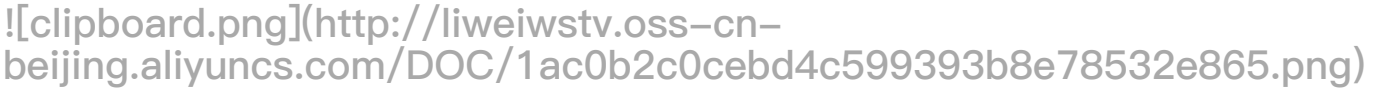
JAVA客户端使用的就是mqtt协议进行链接的 注意看端口是：1883

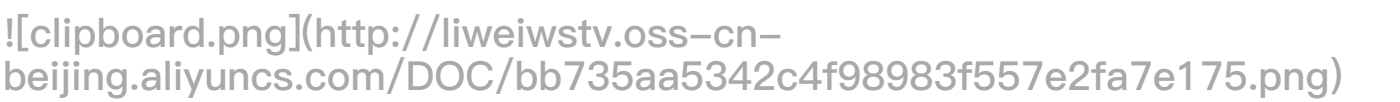
对应的mq端口也是 1883

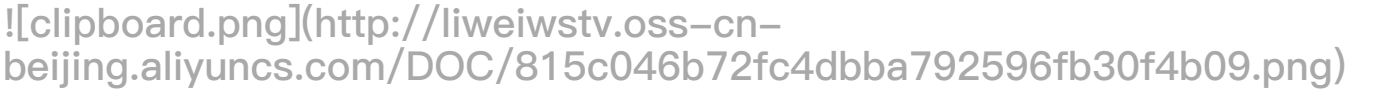
由于我的activemq设置了访问密码 官方demo并没有设置密码 所以需要在此处稍作修改：

如果没有用户名密码 注释掉即可

接下来运行 MQTTfram







原文链接: https://juejin.cn/post/7381741857708670985