t-im 开发文档

目录

[t-im简介 3](#_Toc99461975)

[1.1 认识t-im 3](#_Toc99461976)

[t-im开始使用 3](#_Toc99461977)

[2.1 目录结构 3](#_Toc99461978)

[2.2 t-im 的使用 3](#_Toc99461979)

[2.2.1 引入t-im-server jar包 4](#_Toc99461980)

[2.2.2 新建启动类 4](#_Toc99461981)

[2.2.3 启动效果 4](#_Toc99461982)

[2.3 客户端开发 4](#_Toc99461983)

[2.3.1 jar包的引入 4](#_Toc99461984)

[2.3.2新建启动类 4](#_Toc99461985)

[2.3.3 API的使用 4](#_Toc99461986)

[2.4 SpringBoot项目使用方式 4](#_Toc99461987)

[2.4.1 jar包引入 4](#_Toc99461988)

# 第一章 t-im简介

## 1.1 认识t-im

t-im使用Java语言开发，站在巨人肩膀上制作了异步非阻塞的IM，采用官方协议适配器解决一台服务器可解决http、websocket、tcp（可以私有定制）的多协议通讯。t-im通讯内核采用tio，t-im的设计思想(一套业务逻辑代码解决所有协议的业务逻辑)， 采用了t-io的websocket生态,服务器处理多协议是监听了多端口，与监听相同端口再做协议判别其实是殊途同归， 采用JAVA语言开发的轻量、高性能的IM服务器，Linux OS一核心CPU、2G内存可支持30万用户在线， 主要目标降低即时通讯门槛，快速打造低成本接入在线IM系统， 通过统一的的消息格式就可以实现不同协议间的消息发送，如内置(Http、Websocket、Tcp自定义IM协议)， 并提供通过http协议的api接口进行消息发送无需关心接收端属于什么协议，一套后端业务逻辑处理器搞定一切！

## 1.2 主要特点

1、高性能服务器，采用异步非阻塞AIO通讯

2、轻量、可水平扩展

3、支持集群多机部署

4、支持SSL/TLS加密传输

5、消息格式为(JSON)

6、支持可插拔多种协议(Socket自定义IM协议、Websocket、Http)，各协议可分别独立部署。

7、内置消息持久化(离线、历史、漫游)，保证消息可靠性，高性能存储

8、提供springboot依赖，方便使用spring生态的便利性。

9、部署简单，一键轻松启动。

# 第二章 t-im开始使用

## 2.1 目录结构

各个文件夹概要：

t-im-common：t-im的核心，把客户端和服务器公共部分提取出来组成单独模块，供它们引用。

t-im-server：t-im服务器开发核心框架，引入它即可快速开发IM服务器，使项目轻松获取即时通讯能力。

t-im-client：t-im客户端开发核心框架，引入它即可快速开发IM客户端，轻松连接使用t-im-server开发的服务器。

t-im-server-spring-boot-starter：t-im为springboot生态提供了相应的使用方式，可以通过注解的方式快速形成IM服务器。

t-im-server-demo：一个快速使用t-im-server的demo，用于快速掌握t-im服务器。

t-im-client-demo：一个快速使用t-im-client的demo，用于快速使用t-im客户端。



## 2.2 t-im 的使用

接下来我们来看下怎么基于 jim-server 开发一个属于自己的 im 服务端程序,本教程中以笔者开发的 jim-server-demo 来作为例子讲解如何快速开发一个 im 服务器。 在开发之前我们先来了解一下 j-im 的消息结构是什么样子,因为 j-im通讯都是基于这些个消息结构及命令 cmd 来处理用户消息的，也可以帮助更好的理解后面的 jim-server 端是如何处理消息逻辑以及如何扩展实现自己的 cmd 命令及消息结构。

SpringBoot 开发：只需要引入springboot依赖，在启动类上添加注解@EnableIM，并将t-im-server-demo下的tim.properties文件拷贝至springboot项目的resources目录下。maven 坐标，普通开发引入它即可

### 2.2.1 引入t-im-server jar包

<dependency>

<groupId>io.github.mxd888</groupId>

<artifactId>t-im-server</artifactId>

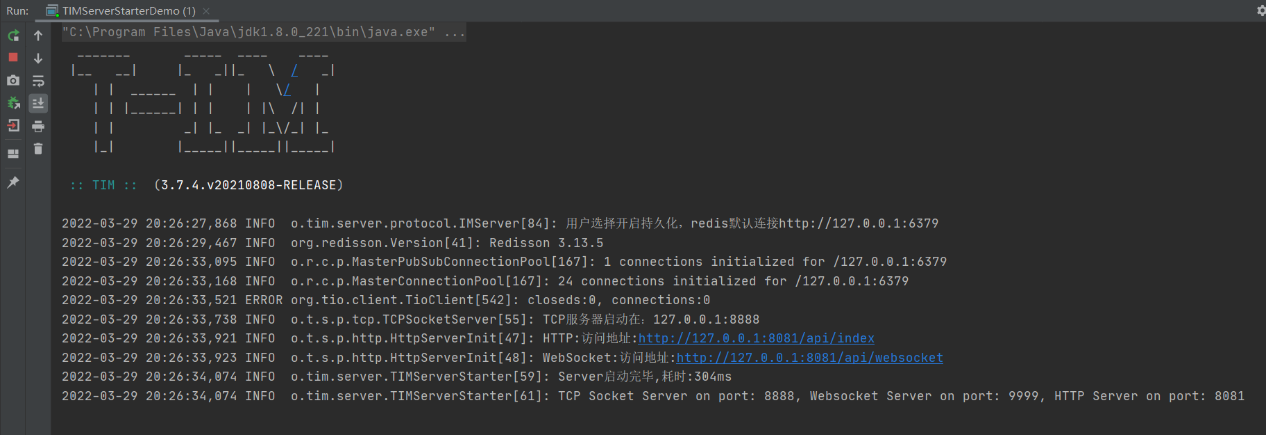
<version>2.1.7.v20220120-RELEASE</version>

</dependency>

### 2.2.2 新建启动类

// 启动方式：推荐使用  
TIMServerStarter timServerStarter = Singleton.get(TIMServerStarter.class);  
// 初始化服务器  
timServerStarter.init();  
// 启动  
timServerStarter.start();

### 2.2.3 启动效果



## 2.3 客户端开发

### 2.3.1 jar包的引入

<dependency>

<groupId>io.github.mxd888</groupId>

<artifactId>t-im-client</artifactId>

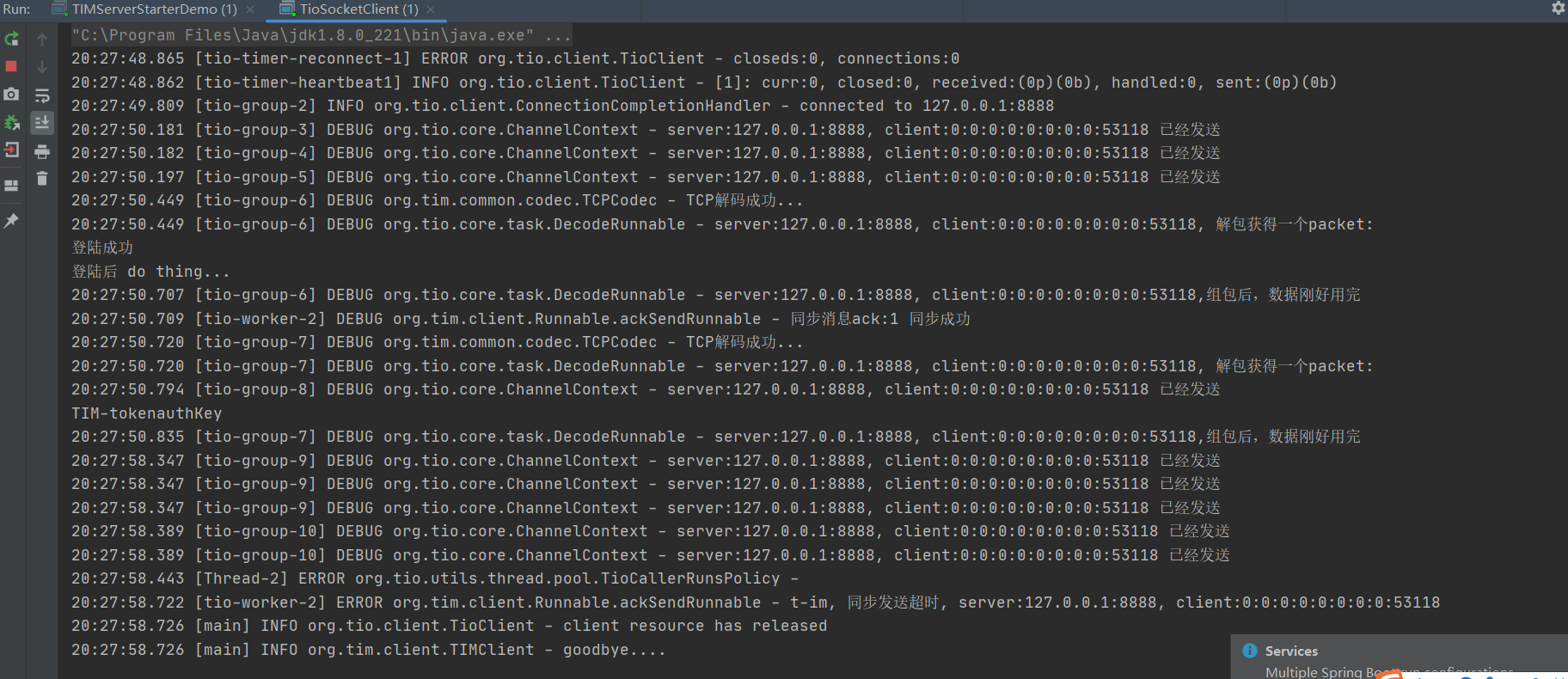
<version>2.1.7.v20220120-RELEASE</version>

</dependency>

### 2.3.2新建启动类

// 初始化并登录  
Options options = new Options(new Node("mixiaodong.xyz", 8888)); // mixiaodong.xyz 192.168.79.1  
TIMClient.*start*(options);  
TIMClient.*getInstance*().login("888", "mi191919", new Callback() {  
 @Override  
 public void success() {  
 System.*out*.println("登陆成功");  
 }  
  
 @Override  
 public void fail() {  
 System.*out*.println("登陆失败");  
 }  
});

TIMClient.*getInstance*().authReq();  
 TIMClient.*getInstance*().joinGroup("200");  
 ChatBody.Builder builder = ChatBody.*newBuilder*();  
 ChatBody body = builder.from("888")  
 .to("999")  
 .setId("120")  
 .setIsSyn(true)  
 .content("集群私聊消息")  
 .msgType(0)  
 .chatType(2)  
 .setCreateTime(new Date().getTime())  
 .groupId(null)  
 .build();  
// TIMClient.sendChatBody(body);  
 TIMClient.*getInstance*().ackSend(new ImPacket(Command.*COMMAND\_CHAT\_REQ*, body.toByte()), 120);  
 try {  
 Thread.*sleep*(8000);  
 TIMClient.*getInstance*().messageReq("119119");  
 TIMClient.*getInstance*().onlineUserId();  
 TIMClient.*getInstance*().logout();  
 } catch (InterruptedException e) {  
 e.printStackTrace();  
 }



### 2.3.3 API的使用

## 2.4 SpringBoot项目使用方式

### 2.4.1 jar包引入

<dependency>

<groupId>io.github.mxd888</groupId>

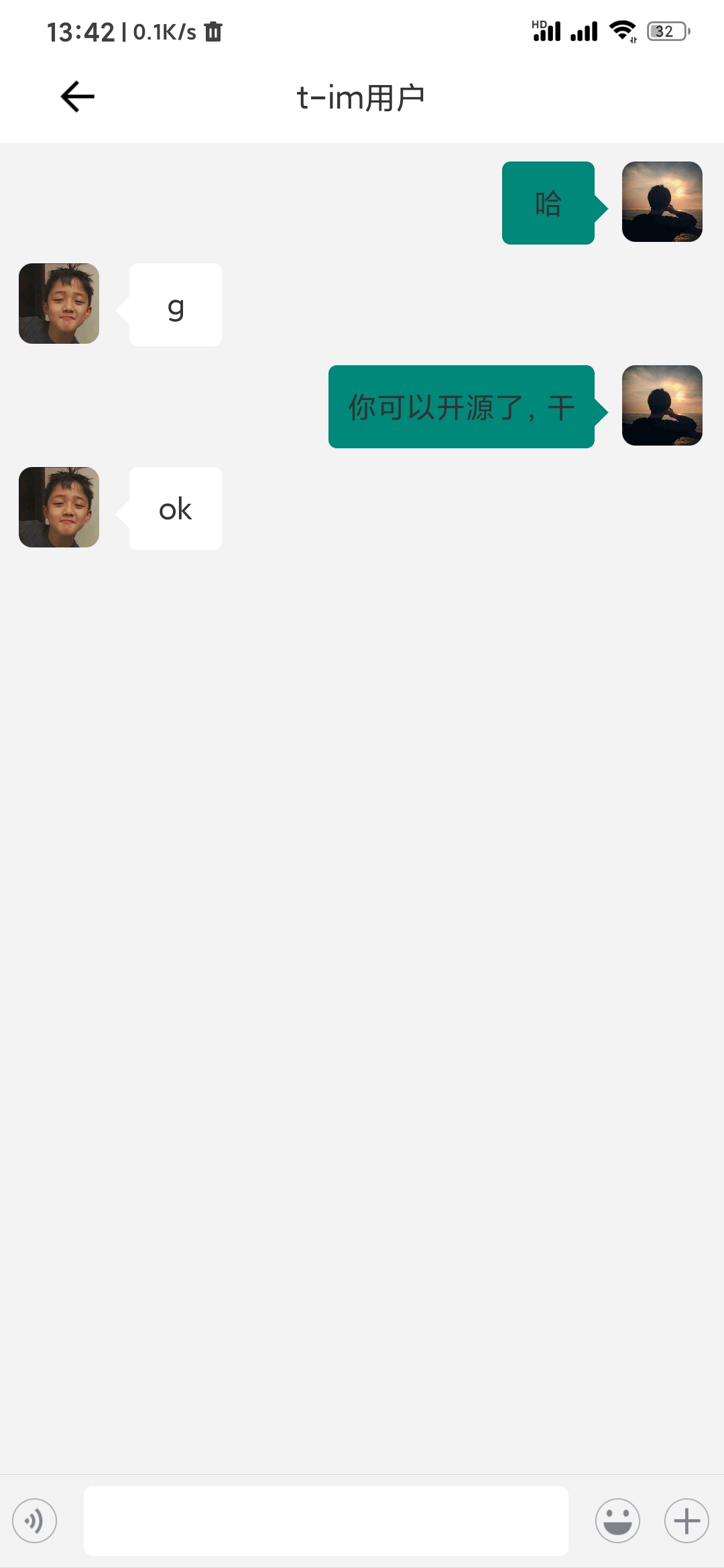
<artifactId>t-im-server-spring-boot-starter</artifactId>

<version>2.1.7.v20220120-RELEASE</version>

</dependency>

# 第三章 效果图

通讯效果图：



感谢您的使用

邮箱：[1191998028@qq.com](mailto:1191998028@qq.com)