## 引入依赖

1. <dependency>  
    <groupId>insigma.socket.libs.base</groupId>  
    <artifactId>socket-lib-base</artifactId>  
    <version>1.0.1-SNAPSHOT</version>  
   </dependency>

## 2启动sokcet服务

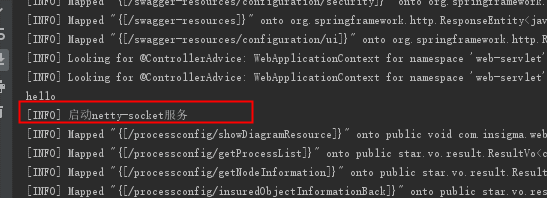
1. 引入该java文件



2.注册该bean

<bean class="com.insigma.config.SocketServerCompoent" name="socketServer" init-method="test">  
 <constructor-arg value="8888"/>  
</bean>

日志中输出如下记录表示成功



## 3发起sokcet请求

public static void main(String[] args) {  
 MyClient.*sendMsg*("127.0.0.1",9999,"word");  
}

使用如下代码即可

## 3 socket包定义及服务器处理逻辑编写

Socket基于tcp/ip协议族，该协议是一个流协议，并未定义明确的包传输规范，我们使用netty搭建socket及发起socket请求，这一过程中，允许我们自定义包规范及处理逻辑，通过重写，ChannelInitializer<SocketChannel>的方法进行，例子如下：

package com.insigma.utils.netty;  
  
import io.netty.channel.ChannelHandler;  
import io.netty.channel.ChannelInitializer;  
import io.netty.channel.ChannelPipeline;  
import io.netty.channel.socket.SocketChannel;  
import io.netty.handler.codec.LineBasedFrameDecoder;  
import io.netty.handler.codec.string.StringDecoder;  
import io.netty.handler.codec.string.StringEncoder;  
import io.netty.util.CharsetUtil;  
  
public class MyServerInitializer extends ChannelInitializer<SocketChannel> {  
 public MyServerInitializer() {  
 }  
  
 protected void initChannel(SocketChannel ch) throws Exception {  
 ChannelPipeline pipeline = ch.pipeline();  
 pipeline.addLast(new ChannelHandler[]{new StringDecoder(CharsetUtil.UTF\_8)});  
 pipeline.addLast(new ChannelHandler[]{new StringEncoder(CharsetUtil.UTF\_8)});  
 pipeline.addLast(new ChannelHandler[]{new LineBasedFrameDecoder(100)});  
 pipeline.addLast(new ChannelHandler[]{new MyServerHandler()});  
 }  
}

本包支持使用者根据自身情况调整包定义及处理逻辑，通过继承该类并注入到SocketServerCompoent的方式实现