后端响应获取题目的Http请求

```java
package com.example.demo.controller;

import com.example.demo.entity.Quest;
import com.example.demo.service.QuestService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;
import java.util.List;

@RestController
public class QuestController {
    @Autowired
    private QuestService questService;


    @RequestMapping("/getQuests")
    public List<Quest> getQuests(@RequestParam("type") String type, @RequestParam("field") Strir
    {
        return questService.getQuests(type,field);

    }
}
```

后端响应不同的websocket请求：

```java
package com.example.demo.controller;

import com.example.demo.config.NettyConfig;
import com.example.demo.message.CompeteMessage;
import com.example.demo.service.MatchService;
import io.netty.bootstrap.ServerBootstrap;
import io.netty.channel.*;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioServerSocketChannel;
import io.netty.channel.socket.nio.NioSocketChannel;
import io.netty.handler.codec.string.StringDecoder;
import io.netty.handler.codec.string.StringEncoder;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RestController;

import java.util.HashMap;
import java.util.List;

@Controller
@RestController
public class StartWebSocket {
    @GetMapping("/action/webSocket")
    public void main(String[] args){
        System.out.println("her");
        MatchService service = new MatchService();
        NioEventLoopGroup boss = new NioEventLoopGroup();
        NioEventLoopGroup worker = new NioEventLoopGroup();
        try{
            ServerBootstrap serverBootstrap = new ServerBootstrap();
            serverBootstrap.channel(NioServerSocketChannel.class);
            serverBootstrap.group(boss,worker);
            serverBootstrap.childHandler(new ChannelInitializer<NioSocketChannel>() {
                protected void initChannel(NioSocketChannel ch) throws Exception{
                    ch.pipeline().addLast(new StringDecoder());
                    ch.pipeline().addLast(new StringEncoder());
                    ch.pipeline().addLast(new ChannelInboundHandlerAdapter() {
                        @Override
                        public void channelActive (ChannelHandlerContext context)throws Exceptic
                            NettyConfig.group.add(context.channel());
                            System.out.println("客户端与服务端连接开启");
                        }

                        @Override
                        public void channelInactive(ChannelHandlerContext context)throws Excepti
                            NettyConfig.group.remove(context.channel());
                            System.out.println("客户端与服务端连接断开");
                        }

                        public void channelRead(ChannelHandlerContext ctx, Object msg){
```

```
                        String command = (String)msg;
                        String []mes = command.split("_");
                        switch(mes[0])
                        {
                            case "1":
                                NettyConfig.allplayer.put(mes[1],ctx.channel());
                                System.out.println(NettyConfig.allplayer.size());
                            case "2":
                                NettyConfig.MatchPool.put(mes[1],ctx.channel());
                                NettyConfig.MatchHelper.put(mes[1],mes[2]);
                                if(service.isIfStart())
                                    break;
                                else
                                {
                                    Thread thread = new Thread()
                                    {
                                        @Override
                                        public void run() {
                                            service.startMatch();
                                        }
                                    };
                                    thread.start();
                                }
                                break;
                            case "3":
                                CompeteMessage mess = new CompeteMessage(mes[1],Integer.pars
                                List<Channel> intr = NettyConfig.GroupPool.get(mes[1]);
                                for(Channel in : intr)
                                {
                                    in.writeAndFlush(mess.MessageString());
                                    System.out.println();
                                    System.out.println(mess.MessageString());
                                }
                            default:
                                break;

                        }
                        System.out.println(msg);
                    }
                });
            }
        });
        Channel channel = serverBootstrap.bind(8888).sync().channel();
        channel.closeFuture().sync();
    } catch (InterruptedException e){

    } finally {
        boss.shutdownGracefully();
        worker.shutdownGracefully();
    }
}
```

前端发送websocket服务：

```
public class CientService {
    private String user;
    private int competeNum = 0;
    private SocketChannel channel;
    private String[] playername = null;
    private HashMap<String, String> gameresult = new HashMap<>();
    boolean ifMatched = false;


    public boolean isIfMatched() {
        return ifMatched;
    }

    public HashMap<String, String> getGameresult()
    {
        return gameresult;
    }

    public void start()
    {
        NioEventLoopGroup group = new NioEventLoopGroup();
        try{
            Bootstrap bootstrap = new Bootstrap();
            bootstrap.channel(NioSocketChannel.class);
            bootstrap.group(group);
            bootstrap.handler(new ChannelInitializer<SocketChannel>() {
                @Override
                protected void initChannel(SocketChannel ch) throws Exception {
                    ch.pipeline().addLast(new StringDecoder());
                    ch.pipeline().addLast(new StringEncoder());
                    ch.pipeline().addLast("client handler",new ChannelInboundHandlerAdapter(){
                        @Override
                        public void channelActive(ChannelHandlerContext ctx) throws Exception{
                            new Thread(()->{
                               System.out.println("连接服务器中");
                            },"system in").start();
                        }

                        public void channelRead(ChannelHandlerContext ctx, Object msg){
                            String retmes = (String) msg;
                            String[] news = retmes.split("_");
                            switch (news[0])
                            {
                                case "1":
                                    break;
                                case "2":
                                    playername = new String[2];
                                    playername[0] = news[1];
                                    playername[1] = news[2];
                                    ifMatched = true;
                                    break;
                                case "3":
                                    if(news[1] != user)
                                        gameresult.put(news[1],news[2]);
                                    break;
                            }
                        }
                    });
                }
            });
            ChannelFuture future =bootstrap.connect("192.168.1.106",8888).sync();
            channel = (SocketChannel) future.channel();

        } catch (Exception e){
            group.shutdownGracefully();
        }
    }

    public void sendMessage(GamerMessage mes)
    {
        String msg = mes.MessageString();
        channel.writeAndFlush(msg);
    }

    public void sendMessage(MatchMessage mes)
    {
        String msg = mes.MessageString();
        channel.writeAndFlush(msg);
    }

    public void sendMessage(CompeteMessage mes)
    {
        String msg = mes.MessageString();
        channel.writeAndFlush(msg);
    }

    public void resetPlayername()
    {
        playername = null;
    }

    public void resetGameresult()
    {
        gameresult = new HashMap<String, String>();
    }
    public void resetIfMatched()
    {
        ifMatched = false;
    }

    public void setCompeteNum(int num)
    {
        competeNum = num;
    }

    public void setUser(String name)
    {
        user = name;
    }
}
```