7-11 ｜ 基于 netty 搭建 im 系统基本骨架和编解码器

www.imooc.com

启动类的代码编写

实现 Netty 的服务端启动类编写，相关代码如下：

```
Java
package org.qiyu.live.im.core.server;

import io.netty.bootstrap.ServerBootstrap;
import io.netty.channel.ChannelFuture;
import io.netty.channel.ChannelInitializer;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioServerSocketChannel;
import io.netty.channel.socket.nio.NioSocketChannel;

import java.util.concurrent.atomic.AtomicInteger;


/**
 * @Author idea
 * @Date: Created in 15:57 2023/7/4
 * @Description
 */
public class NettyServer {

    private int port;
    private AtomicInteger connectCount = new AtomicInteger(0);

    public int getPort() {
        return port;
    }

    public void setPort(int port) {
        this.port = port;
    }

    //设置启动端口
```

```
    public void startApplication(int port) throws
InterruptedException {
        setPort(port);
        ServerBootstrap bootstrap = new ServerBootstrap();
        NioEventLoopGroup bossGroup = new NioEventLoopGroup();
        NioEventLoopGroup workerGroup = new NioEventLoopGroup();
        bootstrap.group(bossGroup, workerGroup);
        bootstrap.channel(NioServerSocketChannel.class);
        bootstrap.childHandler(new
ChannelInitializer<NioSocketChannel>() {
            @Override
            protected void initChannel(NioSocketChannel ch) throws
Exception {
                System.out.println("连接" +
connectCount.getAndIncrement() + "初始化");
            }
        });
        ChannelFuture channelFuture = bootstrap.bind(port).sync();
        //netty 的优雅关闭并不是很靠谱的机制
        Runtime.getRuntime().addShutdownHook(new Thread(() -> {
            bossGroup.shutdownGracefully();
            workerGroup.shutdownGracefully();
            System.out.println("安全销毁线程池");
        }));
        System.out.println("netty 服务启动成功，绑定端口:" + port);
        channelFuture.channel().closeFuture().sync();
    }

    public static void main(String[] args) throws
InterruptedException {
        NettyServer nettyServer = new NettyServer();
        nettyServer.startApplication(9090);
    }

}
```

自定义处理类

实现自定义的处理类：

```
Java
package org.qiyu.live.im.core.server.handler.impl;
```

www.imooc.com

```
import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.SimpleChannelInboundHandler;
import org.qiyu.live.im.core.server.common.ImMsg;
import org.qiyu.live.im.core.server.handler.ImHandlerFactory;

/**
 * @Author idea
 * @Date: Created in 11:18 2023/7/1
 * @Description
 */
public class ImServerCoreHandler extends
SimpleChannelInboundHandler {

    private ImHandlerFactory imHandlerFactory = new
ImHandlerFactoryImpl();

    @Override
    protected void channelRead0(ChannelHandlerContext ctx, Object
msg) {
        if (!(msg instanceof ImMsg)) {
            throw new IllegalArgumentException("error msg type,msg
is " + msg);
        }
        ImMsg imMsg = (ImMsg) msg;
        imHandlerFactory.doMsgHandler(ctx, imMsg);
    }
}
```

编解码器的实现

```
Java
package org.qiyu.live.im.core.server.common;

import com.alibaba.fastjson.JSON;
import io.netty.buffer.ByteBuf;
import io.netty.channel.ChannelHandlerContext;
import io.netty.handler.codec.ByteToMessageDecoder;

import java.util.List;

import org.qiyu.live.im.interfaces.ImMsgBody;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

www.imooc.com

```
/**
 * @Author idea
 * @Date: Created in 16:40 2023/7/2
 * @Description
 */
public class ImMsgDecode extends ByteToMessageDecoder {

    private static final Logger LOGGER =
LoggerFactory.getLogger(ImMsgDecode.class);
    /**
     * 协议的开头部分的标准长度
     */
    public final int BASE_LENGTH = 2 + 4;

    @Override
    protected void decode(ChannelHandlerContext ctx, ByteBuf
byteBuf, List<Object> out) throws Exception {
        if (byteBuf.readableBytes() >= BASE_LENGTH) {
            if (!(byteBuf.readShort() ==
ImConstants.MAGIC_NUMBER)) {
                ctx.close();
                return;
            }
            int length = byteBuf.readInt();
            int code = byteBuf.readInt();
            if (byteBuf.readableBytes() < length) {
                //数据包有异常
                ctx.close();
                return;
            }
            byte[] body = new byte[length];
            byteBuf.readBytes(body);
            ImMsg imMsg = new ImMsg();
            imMsg.setBody(body);
            imMsg.setCode(code);
            out.add(imMsg);
        }
    }
}
```

```
Java
```

www.imooc.com

```
package org.qiyu.live.im.core.server.common;

import io.netty.buffer.ByteBuf;
import io.netty.channel.ChannelHandlerContext;
import io.netty.handler.codec.MessageToByteEncoder;

/**
 * @Author idea
 * @Date: Created in 16:40 2023/7/2
 * @Description
 */
public class ImMsgEncode extends MessageToByteEncoder {

    @Override
    protected void encode(ChannelHandlerContext ctx, Object msg,
ByteBuf out) throws Exception {
        ImMsg imMsg = (ImMsg) msg;
        out.writeShort(imMsg.getMagic());
        out.writeInt(imMsg.getLen());
        out.writeInt(imMsg.getCode());
        out.writeBytes(imMsg.getBody());
        ctx.writeAndFlush(imMsg);
    }
}
```

IM 消息体的实现

```
Java
package org.qiyu.live.im.interfaces;

/**
 * @Author idea
 * @Date: Created in 16:52 2023/7/2
 * @Description
 */
public class ImMsgBody {

    //不同的产品接入的 code 不同
    private int appId;
    //消息类型
    private int bizType;
    //json 格式的数据
    private String data;
```

www.imooc.com

```
    private Long userId;

    public int getAppId() {
        return appId;
    }

    public void setAppId(int appId) {
        this.appId = appId;
    }

    public String getData() {
        return data;
    }

    public void setData(String data) {
        this.data = data;
    }

    public int getBizType() {
        return bizType;
    }

    public void setBizType(int bizType) {
        this.bizType = bizType;
    }

    public Long getUserId() {
        return userId;
    }

    public void setUserId(Long userId) {
        this.userId = userId;
    }
}
```

消息 code 的实现：

```
Java
package org.qiyu.live.im.interfaces;

/**
 * @Author idea
 * @Date: Created in 17:08 2023/7/2
```

www.imooc.com

```
 * @Description
 */
public enum ImMsgCode {

    IM_LOGIN(0,"登录消息","imLoginHandler"),
    IM_LOGOUT(1,"注销消息","imLogoutHandler");

    int code;
    String desc;
    String handlerName;


    public Integer getCode() {
        return code;
    }

    public String getDesc() {
        return desc;
    }

    public String getHandlerName() {
        return handlerName;
    }

    ImMsgCode(int code, String desc, String handlerName) {
        this.code = code;
        this.desc = desc;
        this.handlerName = handlerName;
    }
}
```

ImMsg 部分的实现：

```
Java
package org.qiyu.live.im.core.server.common;

/**
 * im进行消息发送时候的消息体
 *
 * @Author idea
 * @Date: Created in 16:39 2023/7/2
 * @Description
 */
public class ImMsg {
```

www.imooc.com

```
    private short magic;

    private int len;

    //这个 code 是给 im 的 handlerFactory 里面判断用的
    private int code;

    private byte[] body;

    public short getMagic() {
        return magic;
    }

    public void setMagic(short magic) {
        this.magic = magic;
    }



    public int getCode() {
        return code;
    }

    public void setCode(int code) {
        this.code = code;
    }

    public int getLen() {
        return len;
    }

    public void setLen(int len) {
        this.len = len;
    }

    public byte[] getBody() {
        return body;
    }

    public void setBody(byte[] body) {
        this.body = body;
    }
}
```

www.imooc.com

www.imooc.com