استخدام □□□□□ في □□□□

يستخدم □□□□. في الأداء عالية تطبيقات لبناء بالحدث، مدفوع متزامن، قوي، شبكة عمل إطار، □□□□□ استخدام إلى دخول أو الوكالات، الويب، خادمات مثل فعال بشكل الإتصالات من العديد معالجة إلى تحتاج وعميلات خادمات لإنشاء واسع نطاق على شائعة. بداية نقطة إنه حيث بسيط، خادم إعداد على التركيز مع بخطوة، خطوة معك سأكون الفعلي. الوقت في الرسائل أنظمة

---

1. ما هو □□□□□؟

- لبناء نظيفة تطبيقات برمجة واجهة ويوفر المتلقيي□ غير□ الإدخال□الخروج □□□ □□□□ تعقيد عن يخفي عمل □□□□□ الشبكة. تطبيقات
- الميزات الرئيسية: الخروج□الإدخال، المتزامن، حلقات الأحداث، أنابيب معالجة البيانات، والتوسيع.
- افهمه كدستة أدوات لمعالجة الجوازات، البروتوكولات والبيانات بشكل فعال.

---

2. إعداد مشروعك

للاستخدام □□□□□، أضفها إلى مشروعك. إذا كنت تستخدم □□□□□، أضف هذا إلى ملف `pom.xml`:

```
<dependency>
    <groupId>io.netty</groupId>
    <artifactId>netty-all</artifactId>
    <version>4.1.108.Final</version> <!--                2025 -->
</dependency>
```

للمبنى:

```
implementation 'io.netty:netty-all:4.1.108.Final'
```

---

3. المفاهيم الأساسية

متزامن. بشكل المهام وينفذ الإدخال□الخروج عمليات يدير :□□□□□□□□□□ □ الأساسية: النقاط هذه فهم على احرص البرمجة، قبل المحدد□. الإتصال المستلمة، البيانات □مثل الأحداث يعالج :□□□□□□□□□□□□□□ □ جوا□. □مثل اتصالاً يمثل :□□□□□□□□ □ أو خادمك ويبدأ يهيئ :□□□□□□□□□ □ الواردة□الصادرة. البيانات لمعالجة المعالجات من سلسلة :□□□□□□□□□□□□□□□□ □ عميلك.

---

4. بسيط إيكو خادم بناء

لتيتين. كل سيكال مثال هذا العميل. يرسله شيء أي إرسال خادماً لتشغيل

الخطوة 1: إنشاء □□□□□□□□□□□□□□□□□□□□ هيئ هذا يوب الأنبوب لكل الاتصال جديد.

```
import io.netty.channel.ChannelInitializer;
import io.netty.channel.socket.SocketChannel;
import io.netty.channel.ChannelPipeline;
import io.netty.handler.codec.string.StringDecoder;
import io.netty.handler.codec.string.StringEncoder;

public class EchoServerInitializer extends ChannelInitializer<SocketChannel> {
    @Override
    protected void initChannel(SocketChannel ch) throws Exception {
        ChannelPipeline pipeline = ch.pipeline();
        //            /
        pipeline.addLast(new StringDecoder());  //
        pipeline.addLast(new StringEncoder());  //
        pipeline.addLast(new EchoServerHandler());  //
    }
}
```

الخطوة 2: إنشاء □□□□□□□□ ددحي هذا ما حدث عندما تصل البيانات.

```
import io.netty.channel.ChannelHandlerContext;
import io.netty.channel.SimpleChannelInboundHandler;

public class EchoServerHandler extends SimpleChannelInboundHandler<String> {
    @Override
    protected void channelRead0(ChannelHandlerContext ctx, String msg) throws Exception {
        System.out.println("Received: " + msg);
        ctx.writeAndFlush(msg);  //
    }

    @Override
    public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) {
        cause.printStackTrace();
        ctx.close();  //
    }
}
```

```java
import io.netty.bootstrap.ServerBootstrap;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioServerSocketChannel;
import io.netty.channel.EventLoopGroup;

public class EchoServer {
    private final int port;

    public EchoServer(int port) {
        this.port = port;
    }

    public void run() throws Exception {
        // :
        EventLoopGroup bossGroup = new NioEventLoopGroup(1);  //
        EventLoopGroup workerGroup = new NioEventLoopGroup(); //

        try {
            ServerBootstrap bootstrap = new ServerBootstrap()
                .group(bossGroup, workerGroup)
                .channel(NioServerSocketChannel.class)  //          NIO
                .childHandler(new EchoServerInitializer());  //

            //
            bootstrap.bind(port).sync().channel().closeFuture().sync();
        } finally {
            //
            bossGroup.shutdownGracefully();
            workerGroup.shutdownGracefully();
        }
    }

    public static void main(String[] args) throws Exception {
        int port = 8080;  //
        new EchoServer(port).run();
    }
}
```

5. اختبار الخادم

- لتشغيل فئة EchoServer.
- استخدام أداة مثل telnet (telnet localhost 8080) أو كتابة عميل بسيط:

```
import io.netty.bootstrap.Bootstrap;
import io.netty.channel.nio.NioEventLoopGroup;
import io.netty.channel.socket.nio.NioSocketChannel;
import io.netty.channel.EventLoopGroup;

public class EchoClient {
    public static void main(String[] args) throws Exception {
        EventLoopGroup group = new NioEventLoopGroup();
        try {
            Bootstrap bootstrap = new Bootstrap()
                .group(group)
                .channel(NioSocketChannel.class)
                .handler(new EchoServerInitializer());  //

            bootstrap.connect("localhost", 8080).sync().channel().closeFuture().sync();
        } finally {
            group.shutdownGracefully();
        }
    }
}
```

- إعادة الخادم وسيعيد، في telnet، كتابة الرسالة.

---

6. الخيارات التخصيص الرئيسية

- **التشفير**: استخدم StringEncoder، StringDecoder أو قم بتخصيص استخدام ByteBuf الخام للبيانات الثنائية أو مع HttpServerCodec أو مثل الأخرى البروتوكولات.
- **التجميع**: يدعم عدد خيوط NioEventLoopGroup مثل new NioEventLoopGroup(4) لـ 4 خيوط.
- **الأمان**: أضف SslHandler، LoggingHandler من المزيد لتسجيل الحركة من أو مخصص.
- **الخيارات**: تعيين SO_BACKLOG، SO_REUSEADDR، إلخ. عبر bootstrap.option() أو childOption().

---

7. الممارسات أفضل

- **إعادة استخدام [illegible]:** شارك مثيلات الخادم بين العملاء إذا كان ذلك آمنًا. لا تحتفظ بالموارد.
- **استثناءات معالجة:** نفذ دائمًا في معالجات exceptionCaught.
- **إيقاف تشغيل بشكل رقيق:** استدعاء shutdownGracefully() لتنظيف [illegible].
- **استخدام [illegible]:** استخدم ByteBuf لإدارة الذاكرة بشكل فعال في [illegible].

---

8. الخطوات التالية

- **خادم [illegible]:** أضف HttpServerCodec لمعالجة الطلبات والاستجابات.
- **[illegible]:** استخدم WebSocketServerProtocolHandler لترقيات البروتوكول في الوقت الفعلي.
- **الأداء:** قم بضبط حجم EventLoopGroup واستكشف Epoll أو KQueue [illegible] لتحسين الأداء على النظام الأساسي المفضل.

هذا هو الأساس! إذا كنت تريد تعديل مثل [illegible] لإضافة بروتوكول أو معالجة مخصص، البيانات الثنائية، أو خادم الويب، فأخبرني فقط [illegible] من هدفك وسأكون معك خطوة بخطوة!