Netty 实现简单的HTTP服务 ☆

尹吉欢 · 2018-04-08 · 0条评论 · 2623人阅读

netty (http://cxytiandi.com/article/search/netty) java (http://cxytiandi.com/article/search/java)

本篇文章是Netty专题的第八篇，前面七篇文章如下：

- 高性能NIO框架Netty入门篇 (http://cxytiandi.com/blog/detail/17345)
- 高性能NIO框架Netty-对象传输 (http://cxytiandi.com/blog/detail/17403)
- 高性能NIO框架Netty-整合kryo高性能数据传输 (http://cxytiandi.com/blog/detail/17436)
- 高性能NIO框架Netty-整合Protobuf高性能数据传输 (http://cxytiandi.com/blog/detail/17469)
- Netty4自带编解码器详解 (http://cxytiandi.com/blog/detail/17547)
- Netty粘包拆包解决方案 (http://cxytiandi.com/blog/detail/17641)
- Netty 断线重连解决方案 (http://cxytiandi.com/blog/detail/18044)

超文本传输协议（HTTP，HyperText Transfer Protocol)是互联网上应用最为广泛的一种网络协议。

在后端开发中接触HTTP协议的比较多，目前大部分都是基于Servlet容器实现的Http服务，往往有一些核心子系统对性能的要求非常高，这个时候我们可以考虑采用NIO的网络模型来实现HTTP服务，以此提高性能和吞吐量，Netty除了开发网络应用非常方便，还内置了HTTP相关的编解码器，让用户可以很方便的开发出高性能的HTTP协议的服务，Spring Webflux默认是使用的Netty。

接下来我们简单的介绍下如何使用Netty来构建一个简单的Http服务

- 创建一个NettyHttpServer来启动服务

```
    public static void main(String[] args) {
        int port = 2222;
        new NettyHttpServer().run(port);
    }

    public void run(int port) {
        EventLoopGroup bossGroup = new NioEventLoopGroup();
        EventLoopGroup workerGroup = new NioEventLoopGroup();

        ServerBootstrap bootstrap = new ServerBootstrap();
        bootstrap.group(bossGroup, workerGroup).channel(NioServerSocketChannel.class)
                .childHandler(new ChannelInitializer<SocketChannel>() {
                    @Override
                    public void initChannel(SocketChannel ch) throws Exception {
                        ch.pipeline().addLast(
                                new HttpResponseEncoder(),
                                new HttpRequestDecoder(),
                                new NettyHttpServerHandler());
                    }
                }).option(ChannelOption.SO_BACKLOG, 128)
                .childOption(ChannelOption.SO_KEEPALIVE, true);
        try {
            ChannelFuture f = bootstrap.bind(port).sync();
            f.channel().closeFuture().sync();
        } catch (InterruptedException e) {
            e.printStackTrace();
        } finally {
            workerGroup.shutdownGracefully();
            bossGroup.shutdownGracefully();
        }
    }
```

需要关注的是下面的这行代码：

```
ch.pipeline().addLast(
    new HttpResponseEncoder(),
    new HttpRequestDecoder(),
    new NettyHttpServerHandler());
```

- HttpResponseEncoder： 服务端往客户端发送数据的行为是Response，所以这边要使用HttpResponseEncoder将数据进行编码操作
- HttpRequestDecoder：服务端接收到数据的行为是Request，所以要使用HttpRequestDecoder进行解码操作

- NettyHttpServerHandler：自定义的数据处理类

```
1. public class NettyHttpServerHandler extends ChannelInboundHandlerAdapter {
2. 
3.     @Override
4.     public void channelRead(ChannelHandlerContext ctx, Object msg) throws Exception {
5.         FullHttpResponse response = new DefaultFullHttpResponse(
6.                 HttpVersion.HTTP_1_1,
7.                 HttpResponseStatus.OK,
8.                 Unpooled.wrappedBuffer("欢迎来到猿天地".getBytes("utf-8")));
9.         response.headers().set(Names.CONTENT_TYPE, "text/plain;charset=UTF-8");
10.         response.headers().set(Names.CONTENT_LENGTH, response.content().readableBytes());
11.         response.headers().set(Names.CONNECTION, Values.KEEP_ALIVE);
12.         ctx.write(response);
13.         ctx.flush();
14.     }
15. 
16.     @Override
17.     public void channelReadComplete(ChannelHandlerContext ctx) throws Exception {
18.         ctx.flush();
19.     }
20. 
21.     @Override
22.     public void exceptionCaught(ChannelHandlerContext ctx, Throwable cause) throws Excepti
    on {
23.         ctx.close();
24.         cause.printStackTrace();
25.     }
26. }
```

通过DefaultFullHttpResponse构建了返回的对象，设置了HTTP版本，返回的状态码，返回的内容。

返回的响应头通过response.headers().set()进行设置。

到此为止，一个简单的HTTP服务就实现好了，我们启动服务，在浏览器中输入http://localhost:2222/ (http://localhost:2222/) 就可以看到页面中显示的内容是：**欢迎来到猿天地**

上面演示的是一个典型的请求响应模式，一般我们开发接口的时候通常都是需要根据请求的参数进行对应的数据返回，如何在Netty中获取请求的参数呢?

channelRead方法中的msg参数就是请求信息，通过msg可以获取到请求的所有信息，有请求头信息（包括请求的地址，GET请求的参数），请求体（POST请求的数据）。

下面已GET请求的方式来获取请求的参数信息，代码如下：

```
if (msg instanceof HttpRequest) {
        DefaultHttpRequest request = (DefaultHttpRequest) msg;
        System.out.println("URI:" + request.getUri());
        System.err.println(msg);
}

if (msg instanceof HttpContent) {
        LastHttpContent httpContent = (LastHttpContent) msg;
        ByteBuf byteData = httpContent.content();
        if (byteData instanceof EmptyByteBuf) {
            System.out.println("Content：无数据");
        } else {
            String content = new String(ByteUtils.objectToByte(byteData));
            System.out.println("Content:" + content);
        }
}
```

重启服务，访问地址加上参数进行访问：http://localhost:2222/?name=yjh (http://localhost:2222/?name=yjh)

可以看到控制台输出的内容就是一个完整的HTTP请求包含的信息：

```
URI:/?name=yjh
DefaultHttpRequest(decodeResult: success, version: HTTP/1.1)
GET /?name=yjh HTTP/1.1
Host: localhost:2222
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko ) Chrome/64.0.3282.186 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8
Accept-Encoding: gzip, deflate, br
Accept-Language: zh-CN,zh;q=0.9
Cookie: _ga=GA1.1.939107719.1520393952; JSESSIONID=EE205236911D5BBA145E3021DB472D90
Content：无数据
```

本文只是简单的介绍了如何在Netty中去实现HTTP服务，如果想要做成Spring MVC这样的框架那后面的路还很长，请求响应Netty内置了编解码器，还是有很多工作需要自己去做的。比如参数的获取，请求的路由，参数映射成对象等....

源码参考：https://github.com/yinjihuan/netty-im (https://github.com/yinjihuan/netty-im)

欢迎加入我的知识星球，一起交流技术，免费学习猿天地的课程（http://cxytiandi.com/course） (http://cxytiandi.com/course））