**HttpServer.java**

服务端的主程序， HttpServer类中定义静态私有常量logger记录日志，静态私有常量port为服务器端口。启动时如果没有配置端口，则运行在默认端口8080。run() 函数在启动时调用，实现了HttpServer的主要功能。源程序第32-32行定义了两个线程组，bossGroup和workerGroup包含了一组NIO线程，专门用于网络事件的处理。这两个线程组一个用于服务端接收客户端的连接，另一个用于进行SocketChannel的网络读写。第35行创建ServerBootstrap对象，它是Netty用于启动NIO服务端的辅助启动类，第36行调用ServerBootstrap的group方法，将两个NIO线程组当作入参传递到ServerBootstrap中。接着设置创建的Channel为NioServerSocketChannel，并配置TCP参数。在38-44行的匿名内部类中绑定I/O事件的处理类，包括自己实现的 HttpServerhandler类的实例。

**HttpServerHandler.java**

HttpServer中添加HttpServerHandler处理类实现处理逻辑。

HttpServerhandler继承了ChannelHandlerAdapter适配器类。重写的channelRead方法在读取到消息时被调用，从HTTP请求中获取URI和请求方法并将相应信息打印到控制台。第74-76行去除浏览器请求favicon.ico文件带来的干扰。接下来根据请求的方法做相应处理。程序对请求的处理部分在报告书中有详细说明。第61行调用了ChannelHandlerContext的flush方法，它的作用是将消息发送队列中的消息写入到SocketChannel中发送给对方。Netty的write方法并不直接讲消息写入SocketChannel中，而只是把待发送的消息放到发送缓冲数组中，再通过调用flush方法，将缓冲区的消息全部写到SocketChannel中。第117行，当异常发生时，关闭ChannelHandlerContext，释放和ChannelHandlerContext相关联的句柄等资源。

HttpJsonDecoder.java

针对 message body 为 JsonString 的解码器。首先将 msg 转换成 FullHttpRequest对象，然后把 HTTP 请求的 content （即 ByteBuf）的可读字节转换成 JSONObject 对象，加入到对象列表out中。

**HttpClient.java**

客户端的主程序，向服务端发起连接，获取响应。程序的main方法中根据用户输入分别向服务器发起连接并调用sendGet、sendPostJson或sendPostForm方法发送相应格式的消息。发送和接收消息的过程接收消息分别由sendMessage和receiveMessage函数完成。sendMessage根据发送方法、发送内容类型确定发送的消息的首部内容并发送发消息，receiveMessage函数根据响应报文的状态显示相应信息。这些函数的具体功能中在报告书中详细说明。