# ByteBuf

在进行数据传输使用缓存区时，常用的缓冲区是java.nio.Buffer。除了boolean，其他7中基本数据类型都有相对应的缓冲区实现，比如IntBuffer、DoubleBuffer、ByteBuffer等，其中最常用的是ByteBuffer。但是ByteBuffer具有一定的局限性。

1. ByteBuffer长度固定，一旦分配完成后，无法动态扩展和收缩。当需要编码的POJO（Plain Ordinary Java Object）对象大小大于ByteBuffer的容量时，会发生越界异常；
2. ByteBuffer只有一个标识位置的指针position，读写时需要调用flip()方法、rewind()方法等来修改position的值，需要格外的小心；
3. ByteBuffer的API功能有限，一些高级和实用的功能需要自己编程实现；

为了解决这些不足，Netty提供了自己的ByteBuffer实现——ByteBuf。

ByteBuffer只有一个位置指针用于处理读写操作，因此每次读写的时候都需要调用方法来设置指针的值，否则将会出错。

ByteBuffer buffer = ByteBuffer.**allocate**(88);

buffer.**put**("Hello".**getBytes**());

out.**println**(buffer.**toString**()); //java.nio.HeapByteBuffer[pos=5 lim=88 cap=88]

buffer.**flip**();

out.**println**(buffer.**toString**()); //java.nio.HeapByteBuffer[pos=0 lim=5 cap=88]

byte[] array = new byte[buffer.**remaining**()];

buffer.**get**(array);

out.**println**(buffer.**toString**()); //java.nio.HeapByteBuffer[pos=5 lim=5 cap=88]

out.**println**(new **String**(array)); //Hello

return;

ByteBuf通过两个位置指针来协助缓冲区的读写操作。读操作使用readerIndex，写操作使用writerIndex。

# Channel