



链滴

ByteBufferDemo测试 (转载)

作者: [oldcaptain](#)

原文链接: <https://ld246.com/article/1399292760680>

来源网站: [链滴](#)

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)

```

<pre class="brush: java">import java.io.IOException;
import java.nio.ByteBuffer;

/*****
 * com.lin.channel FileChannelDemo.java Created on 2013-10-25
 *
 * @Author: linfenliang
 * @Description:
 * @Version: 1.0
 *****/
public class ByteBufferDemo {

    /**
     * @param args
     *      void
     * @date 2013-10-25
     * @version V1.0.0
     * @author linfenliang
     * @throws IOException
     */
    public static void main(String[] args) throws IOException {
        ByteBuffer buffer = ByteBuffer.allocate(16);
        System.out.println("&quot;ByteBuffer :&quot;");
        System.out.println("&quot;capacity:&quot; + buffer.capacity());
        buffer.put(new byte[] { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
            14, 15 });
        System.out
            .println("&quot;put byte[]{0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15} into buffer.&quot;");
        System.out.println("&quot;limit:&quot; + buffer.limit());
        System.out.println("&quot;position:&quot; + buffer.position());
        buffer.flip();// 数据由写转为读取
        System.out.println("&quot;ByteBuffer 执行flip, 转为读取&quot;");
        byte[] dst = new byte[10];
        buffer.get(dst, 0, dst.length);
        System.out.println(String.format(
            "&quot;byte[:%s,%s,%s,%s,%s,%s,%s,%s,%s,%s&quot;, dst[0], dst[1], dst[2],
            dst[3], dst[4], dst[5], dst[6], dst[7], dst[8], dst[9]));
        System.out.println("&quot;读取完10个字节的数据后:&quot;");
        System.out.println("&quot;limit:&quot; + buffer.limit());
        System.out.println("&quot;position:&quot; + buffer.position());
        buffer.rewind();
        System.out.println("&quot;执行rewind, 重新读取数据&quot;");
        System.out.println("&quot;limit:&quot; + buffer.limit());
        System.out.println("&quot;position:&quot; + buffer.position());
        byte[] dt = new byte[10];
        buffer.get(dt, 0, dst.length);
        System.out.println(String.format(
            "&quot;byte[:%s,%s,%s,%s,%s,%s,%s,%s,%s,%s&quot;, dt[0], dt[1], dt[2],
            dt[3], dt[4], dt[5], dt[6], dt[7], dt[8], dt[9]));
        System.out.println("&quot;读取完10个字节的数据后:&quot;");
        System.out.println("&quot;limit:&quot; + buffer.limit());
        System.out.println("&quot;position:&quot; + buffer.position());
        System.out.println("&quot;在当前位置做标记mark&quot;");
        buffer.mark();
    }
}

```

```

buffer.get();
buffer.get();
buffer.get();
System.out.println("读取3个字节后position:" + buffer.position());
// buffer.rewind();
buffer.reset();
System.out.println("执行reset后position的位置:" + buffer.position());
// buffer.clear();
// System.out.println(buffer.get(3));
buffer.compact();
System.out.println("取出10个字节后, 执行完compact后ByteBuffer第一个字节:" + buffer.get(0));
}
// capacity:作为一个内存块, Buffer有一个固定的大小值, 也叫“capacity”.
// 你只能往里写capacity个byte、long, char等类型。一旦Buffer满了, 需要将其清空(通过数据或者清除数据)才能继续写数据往里写数据。

// position
// 当你写数据到Buffer中时, position表示当前的位置。初始的position值为0.当一个byte、long等数据写到Buffer后,
// position会向前移动到下一个可插入数据的Buffer单元。position最大可为capacity - 1.
// 当读取数据时, 也是从某个特定位置读。当将Buffer从写模式切换到读模式, position会被重为0.
// 当从Buffer的position处读取数据时, position向前移动到下一个可读的位置。

// limit
// 在写模式下, Buffer的limit表示你最多能往Buffer里写多少数据。写模式下, limit等于Buffer的capacity.
// 当切换Buffer到读模式时,
// limit表示你最多能读到多少数据。因此, 当切换Buffer到读模式时, limit会被设置成写模式的position值。
// 换句话说, 你能读到之前写入的所有数据 (limit被设置成已写数据的数量, 这个值在写模式就是position)

// flip
// flip方法将Buffer从写模式切换到读模式。调用flip()方法会将position设回0, 并将limit设置之前position的值。
// 换句话说, position现在用于标记读的位置, limit表示之前写进了多少个byte、char等——现在能读取多少个byte、char等。

// rewind
// 将position设回0, 所以你可以重读Buffer中的所有数据。limit保持不变, 仍然表示能从Buffer中读取多少个元素 (byte、char等)。

// 一旦读完Buffer中的数据, 需要让Buffer准备好再次被写入。可以通过clear()或compact()来完成。
// clear
// 如果调用的是clear()方法, position将被设回0, limit被设置成 capacity的值。换句话说, Buffer
// 被清空了。Buffer中的数据并未清除, 只是这些标记告诉我们可以从哪里开始往Buffer里写数据。
// 如果Buffer中有一些未读的数据, 调用clear()方法, 数据将“被遗忘”, 意味着不再有任何标会告诉你哪些数据被读过, 哪些还没有。
// compact

```

// 如果Buffer中仍有未读的数据，且后续还需要这些数据，但是此时想要先写些数据，那么用compact()方法。

// compact()方法将所有未读的数据拷贝到Buffer起始处。然后将position设到最后一个未读元正后面。limit属性依然像clear()方法一样，设置成capacity。现在Buffer准备好写数据了，但是不会盖未读的数据。

```
}</pre>
```