ByteBuffer

1. ByteBuffer的字节序

如果**最低有效位**在**最高有效位**的前面，则称**小端序**；反之则称**大端序**

- `ByteOrder` 类
  - ByteOrder. `BIG_ENDIAN` ：大端序
  - ByteOrder. `LITTLE_ENDIAN` ：小端序
  - ByteOrder. `nativeOrder()` ：系统默认

2. 其他类型的buffer

- `ByteBuffer` 另一个特别的地方是可以在它的基础上得到其他类型的 `buffer` 。eg：
  - *CharBuffer asCharBuffer()*
    - 在 `ByteBuffer` 上创建一个 `CharBuffer` 视图（从 `position` 位置到 `limit` 位置），在该视图上读写操作会按照 `ByteBuffer` 的字节序去操作数；
    - 视图buffer的 `readOnly` 属性和 `direct` 属性与 `ByteBuffer` 的一致；
    - 只有通过这种方式得到其他类型的 `direct buffer` ；

3. ByteBuffer的重要子类

3.1 HeapByteBuffer

基于Java堆的实现

3.2 DirectByteBuffer

使用了 `unsafe` 的API进行了堆外的实现

3.3 MappedByteBuffer

直接缓冲区。物理内存中创建缓冲区，而不在堆中创建。

3.4 子类性能

- direct缓冲区最适合IO，但是可能创建更加耗时

  direct缓冲区使用的内存，绕过了JVM堆，通过本地代码调用分配。创建和销毁都要比驻留在JVM堆里的缓冲区更加耗时(依赖于操作系统和JVM实现)。direct缓冲区使用的内存不受垃圾收集的控制，因为它们在JVM堆的外部。

4. 核心API

ByteBuffer的读写模式是分开的，正常的应用场景是：往ByteBuffer里写一些数据，然后flip()，然后再读出来。

4.1 put(byte)

写数据，position加1

4.2 get()

读数据，position加1

4.3 flip()

读取数据准备动作，limit=position；position=0；mark = -1;

4.4 clear()

写入数据准备动作，limit=capacity； position=0；mark = -1;

4.5 mark()

设置标记，mark = position;

4.6 reset()

重新读取，position=mark；

4.7 rewind()

倒回， position=0；mark = -1;

4.8 allocate

创建堆缓存区，实际上就是在堆中创建一个数组；创建的是HeapByteBuffer实例；

4.9 allocateDirect

创建堆外（物理内存）缓存区，创建的是allocateDirect实例；

4.10 wrap

```
public static ByteBuffer wrap(byte[] array)
public static ByteBuffer wrap(byte[] array, int offset, int length);
```

通过wrap类把字节数组包装成缓冲区ByteBuffer实例。
注意：把array的引用赋值给ByteBuffer对象中字节数组。如果array数组中的值更改，则ByteBuffer中的数据也会更改的。

5. Channel

介绍两个Channel方面的对象，以便更好的理解Buffer。

Channel隐含的流程：

> 创建临时的direct ByteBuffer
>
> 复制non-direct buffer中的内容到临时buffer
>
> 使用临时buffer执行IO操作
>
> 临时buffer不被引用，最终被垃圾收集

5.1 ReadableByteChannel

从Channel中读取数据，并保存到ByteBuffer的接口：

```
public int read(ByteBuffer dst) throws IOException;
```

5.2 WritableByteChannel

从ByteBuffer中读取数据，并输出到Channel的接口：

```
public int write(ByteBuffer src) throws IOException;
```

5.3 使用示例

```
byteBuffer = ByteBuffer.allocate(N);
//读取数据，写入byteBuffer
readableByteChannel.read(byteBuffer);
//变读为写
byteBuffer.flip();
//读取byteBuffer，写入数据
writableByteChannel.write(byteBuffer);
```

参考博客：

- Java NIO学习笔记之二-图解ByteBuffer
- ByteBuffer实现解析Direct vs Heapped性能比较
- DirectByteBuffer vs. HeapByteBuffer选择问题 - 还没好好研究
- DirectByteBuffer更快吗？