回到首页

Q

我是一段不羁的公告!

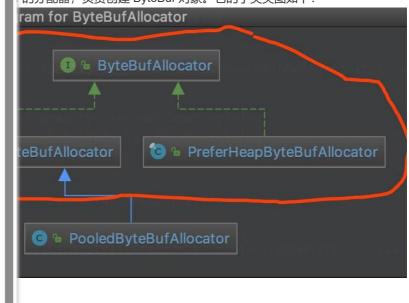
记得给艿艿这 3 个项目加油,添加一个 STAR 噢。 https://github.com/YunaiV/SpringBoot-Labs https://github.com/YunaiV/onemall https://github.com/YunaiV/ruoyi-vue-pro

文章目录

- 1. 概述
- 2. ByteBufAllocator
 - 2.1 DEFAULT
 - 2.2 buffer
 - 2.2.1 ioBuffer
 - 2.2.2 heapBuffer
 - 2.2.3 directBuffer
 - 2.3 compositeBuffer
 - 2.3.1 compositeHeapBuffer
 - 2.3.2 compositeDirectBuffer
 - 2.4 isDirectBufferPooled
 - 2.5 calculateNewCapacity
- 3. AbstractByteBufAllocator
 - 3.1 构造方法
 - 3.2 buffer
 - 3.2.1 ioBuffer
 - 3.2.2 heapBuffer
 - 3.2.3 directBuffer
 - 3.3 compositeBuffer
 - 3.3.1 compositeHeapBuffer
 - 3.3.2 compositeDirectBuffer
 - 3.4 toLeakAwareBuffer
 - 3.5 calculateNewCapacity
- 4. PreferHeapByteBufAllocator 666. 彩蛋

Buffer 之 ByteBufAllocator (一) 简

的分配器,负责创建 ByteBuf 对象。它的子类类图如下:



- PreferHeapByteButAllocator, 倾问创建 Heap ByteBuf 的分配器。
- PooledByteBufAllocator,基于内存池的 ByteBuf 的分配器。
- UnpooledByteBufAllocator, 普通的 ByteBuf 的分配器。

本文分享上面类图红框部分,后面两篇文章再分别分享 UnpooledByteBufAllocator 和 PooledByteBufAllocator。

2. ByteBufAllocator

io.netty.buffer.ByteBufAllocator , ByteBuf分配器接口。

还是老样子,我们逐个来看看每个方法。

2.1 DEFAULT

ByteBufAllocator DEFAULT = ByteBufUtil.DEFAULT_ALLOCATOR;

• 默认 ByteBufAllocator 对象,通过 ByteBufUtil.DEFAULT ALLOCATOR 中获得。代码如下:

```
static final ByteBufAllocator DEFAULT_ALLOCATOR;
   static {
       // 读取 ByteBufAllocator 配置
       String allocType = SystemPropertyUtil.get("io.netty.allocator.type", PlatformDependent.isAndro
       allocType = allocType.toLowerCase(Locale.US).trim();
       // 读取 ByteBufAllocator 对象
ByteBufAllocator alloc;
文章目录。
       if ("unpooled".eauals(allocTvpe)
                                      oc: or. DEFAULT;
  1. 概述
                                      itor.type: {}", allocType);
  2. ByteBufAllocator
                                      oc (e)) {
     2.1 DEFAULT
                                     ite
                                          DEFAULT;
    2.2 buffer
                                      itor.type: {}", allocType);
       2.2.1 ioBuffer
       2.2.2 heapBuffer
       2.2.3 directBuffer
                                      atc DEFAULT;
    2.3 compositeBuffer
                                      itor.type: pooled (unknown: {})", allocType);
       2.3.1 compositeHeapBuffer
       2.3.2 compositeDirectBuffer
    2.4 isDirectBufferPooled
     2.5 calculateNewCapacity
  3. AbstractByteBufAllocator
    3.1 构造方法
    3.2 buffer
       3.2.1 ioBuffer
                                      3.2.2 heapBuffer
                                      Buf
                                          ocator 作为默认 ByteBufAllocator 对象。
       3.2.3 directBuffer
                                          ocator 作为默认 ByteBufAllocator 对象。因为 Android 客户端的内
     3.3 compositeBuffer
       3.3.1 compositeHeapBuffer
       3.3.2 compositeDirectBuffer
     3.4 toLeakAwareBuffer
     3.5 calculateNewCapacity
                                          体创建的是 Heap ByteBuf 还是 Direct ByteBuf,由实现类决定。
  4. PreferHeapByteBufAllocator
  666. 彩蛋
 /**
  * Allocate a {@link ByteBuf}. If it is a direct or heap buffer
  * depends on the actual implementation.
 */
ByteBuf buffer();
ByteBuf buffer(int initialCapacity);
ByteBuf buffer(int initialCapacity, int maxCapacity);
```

2.2.1 ioBuffer

#ioBuffer(...) 方法,创建一个用于 IO 操作的 ByteBuf 对象。倾向于 Direct ByteBuf ,因为对于 IO 操作来说,性能更优。

```
/**
 * Allocate a {@link ByteBuf}, preferably a direct buffer which is suitable for I/O.
 */
ByteBuf ioBuffer();
ByteBuf ioBuffer(int initialCapacity);
ByteBuf ioBuffer(int initialCapacity, int maxCapacity);
```

```
2.2.2 heapBuffer
文章目录
                                      3u
                                           对象。代码如下:
  1. 概述
  2. ByteBufAllocator
    2.1 DEFAULT
    2.2 buffer
       2.2.1 ioBuffer
       2.2.2 heapBuffer
I
                                      y)
      2.2.3 directBuffer
                                          nt maxCapacity);
I
    2.3 compositeBuffer
      2.3.1 compositeHeapBuffer
       2.3.2 compositeDirectBuffer
    2.4 isDirectBufferPooled
    2.5 calculateNewCapacity
                                          ffer 对象。代码如下:
                                      ct
  3. AbstractByteBufAllocator
    3.1 构造方法
    3.2 buffer
                                      wi
                                           the given initial capacity.
       3.2.1 ioBuffer
      3.2.2 heapBuffer
                                      it
      3.2.3 directBuffer
                                      it
                                           int maxCapacity);
I
   3.3 compositeBuffer
       3.3.1 compositeHeapBuffer
       3.3.2 compositeDirectBuffer
    3.4 toLeakAwareBuffer
    3.5 calculateNewCapacity
  4. PreferHeapByteBufAllocator
                                          osite ByteBuf 对象。具体创建的是 Heap ByteBuf 还是 Direct ByteBuf,
  666. 彩蛋
由实现类决定。
```

```
/**
 * Allocate a {@link CompositeByteBuf}.
 * If it is a direct or heap buffer depends on the actual implementation.
 */
CompositeByteBuf compositeBuffer();
CompositeByteBuf compositeBuffer(int maxNumComponents);
```

2.3.1 compositeHeapBuffer

#compositeHeapBuffer(...) 方法,创建一个 Composite Heap ByteBuf 对象。代码如下:

```
/**
  * Allocate a heap {@link CompositeByteBuf}.
  */
CompositeByteBuf compositeHeapBuffer();
CompositeByteBuf compositeHeapBuffer(int maxNumComponents);
```

2023/10/27 17:50

2.3.2 compositeDirectBuffer

#compositeDirectBuffer(...) 方法,创建一个 Composite Direct ByteBuf 对象。代码如下:

```
* Allocate a direct {@link CompositeByteBuf}.
 */
CompositeByteBuf compositeDirectBuffer();
CompositeByteBuf compositeDirectBuffer(int maxNumComponents);
文章目录
4 1. 概述
  2. ByteBufAllocator
                                     FI ect ByteBuf 对象池。代码如下:
    2.1 DEFAULT
    2.2 buffer
       2.2.1 ioBuffer
      2.2.2 heapBuffer
                                     ir ByteBuf}'s are pooled
      2.2.3 directBuffer
   2.3 compositeBuffer
      2.3.1 compositeHeapBuffer
       2.3.2 compositeDirectBuffer
    2.4 isDirectBufferPooled
    2.5 calculateNewCapacity
  3. AbstractByteBufAllocator
                                     ci
                                         , int maxCapacity) 方法, 在 ByteBuf 扩容时, 计算新的容量, 该容
    3.1 构造方法
Ī
                                         围内。代码如下:
                                     ]
    3.2 buffer
      3.2.1 ioBuffer
      3.2.2 heapBuffer
                                     li ByteBuf} that is used when a {@link ByteBuf} needs to expand
       3.2.3 directBuffer
                                         Capacity} as upper-bound.
    3.3 compositeBuffer
                                     : n
       3.3.1 compositeHeapBuffer
       3.3.2 compositeDirectBuffer
                                     pa ty, int maxCapacity);
    3.4 toLeakAwareBuffer
                                     3.5 calculateNewCapacity
  4. PreferHeapByteBufAllocator
4 666. 彩蛋
           ..ac.by.cban.mc32+Or
```

无

io.netty.buffer.AbstractByteBufAllocator ,实现 ByteBufAllocator 接口,ByteBufAllocator 抽象实现类,为 PooledByteBufAllocator 和 UnpooledByteBufAllocator 提供公共的方法。

3.1 构造方法

```
/**

* 是否倾向创建 Direct ByteBuf

*/
private final boolean directByDefault;
/**

* 空 ByteBuf 缓存

*/
private final ByteBuf emptyBuf;

/**

* Instance use heap buffers by default

*/
```

```
无
protected AbstractByteBufAllocator() {
    this(false);
}
/**
 * Create new instance
 st @param preferDirect {@code true} if {@link #buffer(int)} should try to allocate a direct buffer rat
                  a heap buffer
文章目录
                                   " o] n preferDirect) {
  1. 概述
                                        tformDependent.hasUnsafe(); // 支持 Unsafe
                                      : F
  2. ByteBufAllocator
                                      ;
    2.1 DEFAULT
    2.2 buffer
       2.2.1 ioBuffer
      2.2.2 heapBuffer
                                      Dir
                                           ByteBuf。有一个前提是需要支持 Unsafe 操作。
      2.2.3 directBuffer
                                      制
                                          #buffer() 等方法, 创建空 ByteBuf 对象时。
    2.3 compositeBuffer
      2.3.1 compositeHeapBuffer
      2.3.2 compositeDirectBuffer
    2.4 isDirectBufferPooled
    2.5 calculateNewCapacity
( 3. AbstractByteBufAllocator
   3.1 构造方法
    3.2 buffer
      3.2.1 ioBuffer
      3.2.2 heapBuffer
      3.2.3 directBuffer
    3.3 compositeBuffer
       3.3.1 compositeHeapBuffer
(
       3.3.2 compositeDirectBuffer
                                      ci
                                          ) {
1
    3.4 toLeakAwareBuffer
    3.5 calculateNewCapacity
                                        ty);
  4. PreferHeapByteBufAllocator
  666. 彩蛋
    return heapBufter(initialCapacity);
@Override
public ByteBuf buffer(int initialCapacity, int maxCapacity) {
    if (directByDefault) {
        return directBuffer(initialCapacity, maxCapacity);
    return heapBuffer(initialCapacity, maxCapacity);
}
```

• 根据 directByDefault 的值,调用 #directBuffer(...) 方法,还是调用 #heapBuffer(...) 方法。

3.2.1 ioBuffer

```
* 默认容量大小
*/
static final int DEFAULT_INITIAL_CAPACITY = 256;
```

```
@Override
public ByteBuf ioBuffer() {
    if (PlatformDependent.hasUnsafe()) {
        return directBuffer(DEFAULT_INITIAL_CAPACITY);
    }
    return heapBuffer(DEFAULT INITIAL CAPACITY);
}
@Override
文章目录ByteBuf ioBuffer(int initialCapa ty) {
     1. 概述
                                     pa ty);
  2. ByteBufAllocator
    2.1 DEFAULT
                                     );
    2.2 buffer
       2.2.1 ioBuffer
      2.2.2 heapBuffer
(
      2.2.3 directBuffer
                                     pa ty, int maxCapacity) {
    2.3 compositeBuffer
                                     )
      2.3.1 compositeHeapBuffer
                                         ty, maxCapacity);
                                     рā
       2.3.2 compositeDirectBuffer
    2.4 isDirectBufferPooled
                                         xCapacity);
    2.5 calculateNewCapacity
  3. AbstractByteBufAllocator
    3.1 构造方法
    3.2 buffer
                                         ctBuffer(...) 方法, 还是调用 #heapBuffer(...) 方法。
                                     di
      3.2.1 ioBuffer
      3.2.2 heapBuffer
       3.2.3 directBuffer
    3.3 compositeBuffer
       3.3.1 compositeHeapBuffer
       3.3.2 compositeDirectBuffer
    3.4 toLeakAwareBuffer
     3.5 calculateNewCapacity
  4. PreferHeapByteBufAllocator
  666. 彩蛋
```

```
* 默认最大容量大小,无限。
static final int DEFAULT MAX CAPACITY = Integer.MAX VALUE;
@Override
public ByteBuf heapBuffer() {
    return heapBuffer(DEFAULT_INITIAL_CAPACITY, DEFAULT_MAX_CAPACITY);
文章目录
<sup>(</sup> 1. 概述
                                    Ca city) {
1 2. ByteBufAllocator
                                        FAULT MAX CAPACITY);
    2.1 DEFAULT
    2.2 buffer
      2.2.1 ioBuffer
      2.2.2 heapBuffer
(
      2.2.3 directBuffer
                                    Ca city, int maxCapacity) {
    2.3 compositeBuffer
      2.3.1 compositeHeapBuffer
                                    pa ty == 0) {
       2.3.2 compositeDirectBuffer
    2.4 isDirectBufferPooled
    2.5 calculateNewCapacity
                                    ci ); // 校验容量的参数
  3. AbstractByteBufAllocator
    3.1 构造方法
                                    it
                                         maxCapacity);
    3.2 buffer
      3.2.1 ioBuffer
      3.2.2 heapBuffer
      3.2.3 directBuffer
                                        city, int maxCapacity) 抽象方法, 创建 Heap ByteBuf 对象。代
    3.3 compositeBuffer
       3.3.1 compositeHeapBuffer
       3.3.2 compositeDirectBuffer
    3.4 toLeakAwareBuffer
                                    3.5 calculateNewCapacity
  4. PreferHeapByteBufAllocator
                                    3u r(int initialCapacity, int maxCapacity);
  666. 彩蛋
```

• 因为是否基于对象池的方式,创建 Heap ByteBuf 对象的实现会不同,所以需要抽象。

3.2.3 directBuffer

```
@Override
public ByteBuf directBuffer() {
    return directBuffer(DEFAULT_INITIAL_CAPACITY, DEFAULT_MAX_CAPACITY);
}

@Override
public ByteBuf directBuffer(int initialCapacity) {
    return directBuffer(initialCapacity, DEFAULT_MAX_CAPACITY);
}

@Override
public ByteBuf directBuffer(int initialCapacity, int maxCapacity) {
    // 空 ByteBuf 对象
    if (initialCapacity == 0 && maxCapacity == 0) {
        return emptyBuf;
```

```
}
validate(initialCapacity, maxCapacity); // 校验容量的参数
// 创建 Direct ByteBuf 对象
return newDirectBuffer(initialCapacity, maxCapacity);
}
```

无

• 最终调用 #newDirectBuffer(int initialCapacity, int maxCapacity) **抽象**方法,创建 Direct ByteBuf 对象。 代码如下:

```
文章目录
                                            the given initialCapacity and maxCapacity.
  1. 概述
  2. ByteBufAllocator
    2.1 DEFAULT
                                      :tl
                                           fer(int initialCapacity, int maxCapacity);
    2.2 buffer
                                       В
                                           Buf 对象的实现会不同,所以需要抽象。
       2.2.1 ioBuffer
       2.2.2 heapBuffer
      2.2.3 directBuffer
    2.3 compositeBuffer
      2.3.1 compositeHeapBuffer
(
       2.3.2 compositeDirectBuffer
    2.4 isDirectBufferPooled
                                       er
    2.5 calculateNewCapacity
  3. AbstractByteBufAllocator
                                       );
    3.1 构造方法
    3.2 buffer
       3.2.1 ioBuffer
       3.2.2 heapBuffer
       3.2.3 directBuffer
    3.3 compositeBuffer
                                      er  nt maxNumComponents) {
       3.3.1 compositeHeapBuffer
       3.3.2 compositeDirectBuffer
                                           umComponents);
                                      ma
    3.4 toLeakAwareBuffer
    3.5 calculateNewCapacity
                                      Cc
                                           onents);
  4. PreferHeapByteBufAllocator
666. 彩蛋
```

• 根据 directByDefault 的值,调用 #compositeDirectBuffer(...) 方法,还是调用 #compositeHeapBuffer(...) 方法。

3.3.1 compositeHeapBuffer

```
/**

* Composite ByteBuf 可包含的 ByteBuf 的最大数量

*/
static final int DEFAULT_MAX_COMPONENTS = 16;

@Override
public CompositeByteBuf compositeHeapBuffer() {
    return compositeHeapBuffer(DEFAULT_MAX_COMPONENTS);
}

@Override
public CompositeByteBuf compositeHeapBuffer(int maxNumComponents) {
```

2023/10/27 17:50

```
return toLeakAwareBuffer(new CompositeByteBuf(this, false, maxNumComponents));
}
```

- 创建 CompositeByteBuf 对象,并且方法参数 direct 为 false ,表示 Heap 类型。
- 调用 #toLeakAwareBuffer(CompositeByteBuf) 方法, 装饰成 LeakAware 的 ByteBuf 对象。

3.3.2 compositeDirectBuffer

```
文章目录
                                      ct
                                          ffer() {
I 1. 概述
  2. ByteBufAllocator
                                      .UL MAX COMPONENTS);
    2.1 DEFAULT
    2.2 buffer
       2.2.1 ioBuffer
      2.2.2 heapBuffer
                                      ct ffer(int maxNumComponents) {
1
      2.2.3 directBuffer
                                          eByteBuf(this, true, maxNumComponents));
    2.3 compositeBuffer
       2.3.1 compositeHeapBuffer
       2.3.2 compositeDirectBuffer
    2.4 isDirectBufferPooled
                                      数 ■ irect 为 true ,表示 Direct 类型。
    2.5 calculateNewCapacity
                                      te f) 方法,装饰成 LeakAware 的 ByteBuf 对象。
  3. AbstractByteBufAllocator
    3.1 构造方法
    3.2 buffer
      3.2.1 ioBuffer
7
                                     3ut
                                          三)内存泄露检测》中的「3.1 创建 LeakAware ByteBuf 对象」小节,
      3.2.2 heapBuffer
ŧ
       3.2.3 directBuffer
    3.3 compositeBuffer
       3.3.1 compositeHeapBuffer
       3.3.2 compositeDirectBuffer
    3.4 toLeakAwareBuffer
    3.5 calculateNewCapacity
  4. PreferHeapByteBufAllocator
  666. 彩蛋
static final int CALCULATE_THRESHOLD = 48576 * 4; // 4 MiB page
  1: @Override
  2: public int calculateNewCapacity(int minNewCapacity, int maxCapacity) {
         if (minNewCapacity < 0) {
             throw new IllegalArgumentException("minNewCapacity: " + minNewCapacity + " (expected: 0+)
  4:
  5:
         if (minNewCapacity > maxCapacity) {
  7:
             throw new IllegalArgumentException(String.format(
                      "minNewCapacity: %d (expected: not greater than maxCapacity(%d)",
  8:
  9:
                     minNewCapacity, maxCapacity));
 10:
         }
         final int threshold = CALCULATE_THRESHOLD; // 4 MiB page
 11:
 12:
         // <1> 等于 threshold , 直接返回 threshold 。
 13:
         if (minNewCapacity == threshold) {
             return threshold;
 15:
 16:
         }
 17:
         // <2> 超过 threshold , 增加 threshold , 不超过 maxCapacity 大小。
 18:
```

```
19:
         // If over threshold, do not double but just increase by threshold.
         if (minNewCapacity > threshold) {
 20:
             int newCapacity = minNewCapacity / threshold * threshold;
 21:
 22:
             if (newCapacity > maxCapacity - threshold) { // 不超过 maxCapacity
                 newCapacity = maxCapacity;
 23:
             } else {
 25:
                 newCapacity += threshold;
 26:
 27:
             return newCapacity;
文章目录 }
  1. 概述
                                         干始两倍计算, 不超过 4M 大小。
                                      64
  2. ByteBufAllocator
                                          to 4 MiB, starting from 64.
    2.1 DEFAULT
    2.2 buffer
                                     ar ity) {
       2.2.1 ioBuffer
       2.2.2 heapBuffer
       2.2.3 directBuffer
                                        Capacity);
    2.3 compositeBuffer
       2.3.1 compositeHeapBuffer
       2.3.2 compositeDirectBuffer
    2.4 isDirectBufferPooled
    2.5 calculateNewCapacity
                                          龙 3 种情况: <1> / <2> / <3> 。代码比较简单,胖友自己看注释。
  3. AbstractByteBufAllocator
    3.1 构造方法
                                          cator
    3.2 buffer
       3.2.1 ioBuffer
       3.2.2 heapBuffer
                                     lo
                                          tor ,实现 ByteBufAllocator 接口,倾向创建 Heap ByteBuf 的分配器。
       3.2.3 directBuffer
t
                                          和 #compositeBuffer(...) 方法,创建的都是 Heap ByteBuf 对象。
    3.3 compositeBuffer
1
       3.3.1 compositeHeapBuffer
       3.3.2 compositeDirectBuffer
    3.4 toLeakAwareBuffer
    3.5 calculateNewCapacity
  4. PreferHeapByteBufAllocator
1 666. 彩蛋
public PreferHeapByteBufAllocator(ByteBufAllocator allocator) {
    this.allocator = ObjectUtil.checkNotNull(allocator, "allocator");
}
@Override
public ByteBuf buffer() {
    return allocator.heapBuffer();
}
@Override
public ByteBuf ioBuffer() {
    return allocator.heapBuffer();
}
@Override
public CompositeByteBuf compositeBuffer() {
    return allocator.compositeHeapBuffer();
}
```

其它方法,就是调用 allocator 的对应的方法。

666. 彩蛋

₩ 小水文一篇。铺垫铺垫,你懂的。

© 後音段 ^{芋道源码} | 总访客数次 & & 总访问量次 🤦

- 1. 概述
- 2. ByteBufAllocator
 - 2.1 DEFAULT
 - 2.2 buffer
 - 2.2.1 ioBuffer
 - 2.2.2 heapBuffer
 - 2.2.3 directBuffer
 - 2.3 compositeBuffer
 - 2.3.1 compositeHeapBuffer
 - 2.3.2 compositeDirectBuffer
 - 2.4 isDirectBufferPooled
 - 2.5 calculateNewCapacity
- 3. AbstractByteBufAllocator
 - 3.1 构造方法
 - 3.2 buffer
 - 3.2.1 ioBuffer
 - 3.2.2 heapBuffer
 - 3.2.3 directBuffer
 - 3.3 compositeBuffer
 - 3.3.1 compositeHeapBuffer
 - 3.3.2 compositeDirectBuffer
 - 3.4 toLeakAwareBuffer
 - 3.5 calculateNewCapacity
- 4. PreferHeapByteBufAllocator

666. 彩蛋