

目录	
第1章 基础	
01 开篇词：为什么学习本专栏	
02 String、Long 源码解析和面试题	
03 Java 常用关键字理解	
04 Arrays、Collections、Objects 常用方法源码解析	
第2章 集合	
05 ArrayList 源码解析和设计思路	
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	
08 HashMap 源码解析	
09 TreeMap 和 LinkedHashMap 核心源码解析	
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	

45 Socket 源码及面试题

更新时间：2019-11-29 09:54:23



“

没有智慧的头脑，就象没有腊烛的灯笼。

——列夫·托尔斯泰

”

引导语

Socket 中文翻译叫套接字，可能很多工作四五年的同学都没有用过这个 API，但只要用到这个 API 时，必然是在重要的工程的核心代码处。

大家平时基本都在用开源的各种 rpc 框架，比如说 Dubbo、gRPC、Spring Cloud 等等，很少需要手写网络调用，以下三小节可以帮助大家补充这块的内容，当你真正需要的时候，可以作为手册示例。

本文和《ServerSocket 源码及面试题》一文主要说 Socket 和 ServerSocket 的源码，《工作实战：Socket 结合线程池的使用》这章主要说两个 API 在实际工作中如何落地。

1 Socket 整体结构

Socket 的结构非常简单，Socket 就像一个壳一样，将套接字初始化、创建连接等各种操作包装了一下，其底层实现都是 SocketImpl 实现的，Socket 本身的业务逻辑非常简单。

Socket 的属性不多，有套接字的状态，SocketImpl，读写状态等等，源码如下图：

<div>← 慕课专栏</div> <div>面试官系统精讲Java源码及大厂真题 / 45 Socket 源码及面试题</div>	
目录	
第1章 基础	
01 开篇词：为什么学习本专栏	
02 String、Long 源码解析和面试题	
03 Java 常用关键字理解	
04 Arrays、Collections、Objects 常用方法源码解析	
第2章 集合	
05 ArrayList 源码解析和设计思路	
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	
08 HashMap 源码解析	
09 TreeMap 和 LinkedHashMap 核心源码解析	
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	

```
private boolean connected = false;// 已连接
private boolean closed = false;// 已关闭
private Object closeLock = new Object();// 关闭锁
private boolean shutIn = false;// 读是否关闭
private boolean shutOut = false;// 写是否关闭

// 套接字的实现
SocketImpl impl;
```

套接字的状态变更都是有对应操作方法的，比如套接字新建（createImpl 方法）后，状态就会更改成 created = true，连接（connect）之后，状态更改成 connected = true 等等。

2 初始化

Socket 的构造器比较多，可以分成两大类：

1. 指定代理类型（Proxy）创建套节点，一共有三种类型为：DIRECT（直连）、HTTP（HTTP、FTP 高级协议的代理）、SOCKS（SOCKS 代理），三种不同的代码方式对应的 SocketImpl 不同，分别是：PlainSocketImpl、HttpConnectSocketImpl、SocksSocketImpl，除了类型之外 Proxy 还指定了地址和端口；
2. 默认 SocksSocketImpl 创建，并且需要在构造器中传入地址和端口，源码如下：

```
// address 代表IP地址，port 表示套接字的端口
// address 我们一般使用 InetSocketAddress，InetSocketAddress 有 ip+port、域名+port、InetA
public Socket(InetAddress address, int port) throws IOException {
    this(address != null ? new InetSocketAddress(address, port) : null,
         (SocketAddress) null, true);
}
```

这里的 address 可以是 ip 地址或者域名，比如说 127.0.0.1 或者 www.wenhe.com。

我们一起看一下这个构造器调用的 this 底层构造器的源码：

```
// stream 为 true 时，表示为stream socket 流套接字，使用 TCP 协议，比较稳定可靠，但占用资源
// stream 为 false 时，表示为datagram socket 数据报套接字，使用 UDP 协议，不稳定，但占用资源
private Socket(SocketAddress address, SocketAddress localAddr,
               boolean stream) throws IOException {
    setImpl();

    // backward compatibility
    if (address == null)
        throw new NullPointerException();

    try {
        // 创建 socket
        createImpl(stream);
        // 如果 ip 地址不为空，绑定地址
        if (localAddr != null)
            // create、bind、connect 也是 native 方法
            bind(localAddr);
        connect(address);
    } catch (IOException | IllegalArgumentException | SecurityException e) {
        try {
            close();
        } catch (IOException ce) {
        }
    }
}
```

← 慕课专栏	面试官系统精讲Java源码及大厂真题 / 45 Socket 源码及面试题
目录	<pre> } }</pre>
第1章 基础	
01 开篇词：为什么学习本专栏	从源码中可以看出：
02 String、Long 源码解析和面试题	
03 Java 常用关键字理解	
04 Arrays、Collections、Objects 常用方法源码解析	<ol style="list-style-type: none">1. 在构造 Socket 的时候，你可以选择 TCP 或 UDP，默认是 TCP；2. 如果构造 Socket 时，传入地址和端口，那么在构造的时候，就会尝试在此地址和端口上创建套接字；3. Socket 的无参构造器只会初始化 SocksSocketImpl，并不会和当前地址端口绑定，需要我们手动的调用 connect 方法，才能使用当前地址和端口；4. Socket 我们可以理解成网络沟通的语言层次的抽象，底层网络创建、连接和关闭，仍然是 TCP 或 UDP 本身网络协议指定的标准，Socket 只是使用 Java 语言做了一层封装，从而让我们更方便地使用。
第2章 集合	3 connect 连接服务端
05 ArrayList 源码解析和设计思路	connect 方法主要用于 Socket 客户端连接上服务端，如果底层是 TCP 层协议的话，就是通过三次握手和服务端建立连接，为客户端和服务端之间的通信做好准备，底层源码如下：
06 LinkedList 源码解析	<pre>public void connect(SocketAddress endpoint, int timeout) throws IOException { }</pre>
07 List 源码会问哪些面试题	connect 方法要求有两个入参，第一个入参是 SocketAddress，表示服务端的地址，我们可以使用 InetSocketAddress 进行初始化，比如：new InetSocketAddress(“www.wenhe.com”，2000)。
08 HashMap 源码解析	第二入参是超时时间的意思（单位毫秒），表示客户端连接服务端的最大等待时间，如果超过当前等待时间，仍然没有成功建立连接，抛 SocketTimeoutException 异常，如果是 0 的话，表示无限等待。
09 TreeMap 和 LinkedHashMap 核心源码解析	4 Socket 常用设置参数
10 Map源码会问哪些面试题	Socket 的常用设置参数在 SocketOptions 类中都可以找到，接下来我们来一一分析下，以下理解大多来自类注释和网络。
11 HashSet、TreeSet 源码解析	4.1 setTcpNoDelay
12 彰显细节：看集合源码对我们实际工作的帮助和应用	此方法是用来设置 TCP_NODELAY 属性的，属性的注释是这样的：此设置仅仅对 TCP 生效，主要为了禁止使用 Nagle 算法，true 表示禁止使用，false 表示使用，默认是 false。
13 差异对比：集合在 Java 7 和 8 有何不同和改进	对于 Nagle 算法，我们引用维基百科上的解释：
14 简化工作：Guava Lists Maps 实际工作运用和源码	<div>纳格算法是以减少数据包发送量来增进 [TCP/IP] 网络的性能，它由约翰·纳格任职于Ford Aerospace时命名。</div> <div>纳格的文件[注 1]描述了他所谓的“小数据包问题” - 某个应用程序不断地提交小单位的数据，且某些常只占1字节大小。因为TCP数据包具有40字节的标头信息（TCP与IPv4各</div>
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	

<div>← 慕课专栏</div> <div>面试官系统精讲Java源码及大厂真题 / 45 Socket 源码及面试题</div>	
目录	更糟的是，在慢速的网络连线下，这类的数据包会大量地在同一时段传输，造成 堵塞碰撞 。
第1章 基础	纳格算法的工作方式是合并（ coalescing ）一定数量的输出数据后一次提交。特别的是，只要有已提交的数据包尚未确认，发送者会持续缓冲数据包，直到累积一定数量的数据才提交。
01 开篇词：为什么学习本专栏	总结算法开启关闭的场景：
02 String、Long 源码解析和面试题	1. 如果 Nagle 算法关闭，对于小数据包，比如一次鼠标移动，点击，客户端都会立马和服务端交互，实时响应度非常高，但频繁的通信却很占用不少网络资源；
03 Java 常用关键字理解	2. 如果 Nagle 算法开启，算法会自动合并小数据包，等到达到一定大小（MSS）后，才会和服务端交互，优点是减少了通信次数，缺点是实时响应度会低一些。
04 Arrays、Collections、Objects 常用方法源码解析	Socket 创建时，默认是开启 Nagle 算法的，可以根据实时性要求来选择是否关闭 Nagle 算法。
第2章 集合	4.2 setSoLinger
05 ArrayList 源码解析和设计思路	setSoLinger 方法主要用来设置 SO_LINGER 属性值的。
06 LinkedList 源码解析	注释上大概是这个意思：在我们调用 close 方法时，默认是直接返回的，但如果给 SO_LINGER 赋值，就会阻塞 close 方法，在 SO_LINGER 时间内，等待通信双方发送数据，如果时间过了，还未结束，将发送 TCP RST 强制关闭 TCP 。
07 List 源码会问哪些面试题	我们看一下 setSoLinger 源码：
08 HashMap 源码解析	<pre>// on 为 false，表示不启用延时关闭，true 的话表示启用延时关闭 // linger 为延时的时间，单位秒 public void setSoLinger(boolean on, int linger) throws SocketException { // 检查是否已经关闭 if (isClosed()) throw new SocketException("Socket is closed"); // 不启用延时关闭 if (!on) { getImpl().setOption(SocketOptions.SO_LINGER, new Boolean(on)); } // 启用延时关闭，如果 linger 为 0，那么会立即关闭 // linger 最大为 65535 秒，约 18 小时 } else { if (linger < 0) { throw new IllegalArgumentException("invalid value for SO_LINGER"); } if (linger > 65535) linger = 65535; getImpl().setOption(SocketOptions.SO_LINGER, new Integer(linger)); } }</pre>
09 TreeMap 和 LinkedHashMap 核心源码解析	4.3 setOOBInline
10 Map源码会问哪些面试题	setOOBInline 方法主要使用设置 SO_OOBINLINE 属性。
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	

← 慕课专栏	面试官系统精讲Java源码及大厂真题 / 45 Socket 源码及面试题
目录	查询了很多资料，都建议尽可能的去避免设置该值，禁止使用 TCP 紧急数据。
第1章 基础	4.4 setSoTimeout
01 开篇词：为什么学习本专栏	setSoTimeout 方法主要是用来设置 SO_TIMEOUT 属性的。
02 String、Long 源码解析和面试题	注释上说：用来设置阻塞操作的超时时间，阻塞操作主要有：
03 Java 常用关键字理解	1. ServerSocket.accept() 服务器等待客户端的连接；
04 Arrays、Collections、Objects 常用方法源码解析	2. SocketInputStream.read() 客户端或服务端读取输入超时；
第2章 集合	3. DatagramSocket.receive()。
05 ArrayList 源码解析和设计思路	我们必须在必须在阻塞操作之前设置该选项， 如果时间到了，操作仍然在阻塞，会抛出 InterruptedIOException 异常（Socket 会抛出 SocketTimeoutException 异常，不同的套接字抛出的异常可能不同）。
06 LinkedList 源码解析	对于 Socket 来说，超时时间如果设置成 0，表示没有超时时间，阻塞时会无限等待。
07 List 源码会问哪些面试题	4.5 setSendBufferSize
08 HashMap 源码解析	setSendBufferSize 方法主要用于设置 SO_SNDBUF 属性的，入参是 int 类型，表示设置发送端（输出端）的缓冲区的大小，单位是字节。
09 TreeMap 和 LinkedHashMap 核心源码解析	入参 size 必须大于 0，否则会抛出 IllegalArgumentException 异常。
10 Map源码会问哪些面试题	一般我们都是采取默认的，如果值设置太小，很有可能导致网络交互过于频繁，如果值设置太大，那么交互变少，实时性就会变低。
11 HashSet、TreeSet 源码解析	4.6 setReceiveBufferSize
12 彰显细节：看集合源码对我们实际工作的帮助和应用	setReceiveBufferSize 方法主要用来设置 SO_RCVBUF 属性的，入参是 int 类型，表示设置接收端的缓冲区的大小，单位是字节。
13 差异对比：集合在 Java 7 和 8 有何不同和改进	入参 size 必须大于 0，否则会抛出 IllegalArgumentException 异常。
14 简化工作：Guava Lists Maps 实际工作运用和源码	一般来说，在套接字建立连接之后，我们可以随意修改窗口大小，但是当窗口大小大于 64k 时，需要注意：
第3章 并发集合类	1. 必须在 Socket 连接客户端之前设置缓冲值；
15 CopyOnWriteArrayList 源码解析和设计思路	2. 必须在 ServerSocket 绑定本地地址之前设置缓冲值。
16 ConcurrentHashMap 源码解析和设计思路	4.7 setKeepAlive
17 并发 List、Map源码面试题	setKeepAlive 方法主要用来设置 SO_KEEPAIVE 属性，主要是用来探测服务端的套接字是否还是存活状态，默认设置是 false，不会触发这个功能。
18 场景集合：并发 List、Map的应用	如果 SO_KEEPAIVE 开启的话，TCP 自动触发功能：如果两小时内，客户端和服务端的套接字之间没有任何通信，TCP 会自动发送 keepalive 探测给对方，对方必须响应这个探测（假设是客户端发送给服务端），预测有三种情况：
	1. 服务端使用预期的 ACK 回复，说明一切正常；

<div><div>← 慕课专栏</div><div>☰ 面试官系统精讲Java源码及大厂真题 / 45 Socket 源码及面试题</div></div>	
目录	4.8 setReuseAddress
第1章 基础	setReuseAddress 方法主要用来设置 SO_REUSEADDR 属性，入参是布尔值，默认是 false。
01 开篇词：为什么学习本专栏	套接字在关闭之后，会等待一段时间之后才会真正的关闭，如果此时有新的套接字前来绑定同样的地址和端口时，如果 setReuseAddress 为 true 的话，就可以绑定成功，否则绑定失败。
02 String、Long 源码解析和面试题	5 总结
03 Java 常用关键字理解	如果平时一直在做业务代码，Socket 可能用到的很少，但面试问到网络协议时，或者以后有机会做做中间件的时候，就会有大概率会接触到 Socket，所以多学学，作为知识储备也蛮好的。
04 Arrays、Collections、Objects 常用方法源码解析	<div><div>←</div><div>44 场景实战：ThreadLocal 在上下文传值场景下的实践</div><div>46 ServerSocket 源码及面试题 →</div></div>
第2章 集合	精选留言 1
05 ArrayList 源码解析和设计思路	欢迎在这里发表留言，作者筛选后可公开显示
06 LinkedList 源码解析	
07 List 源码会问哪些面试题	慕粉1150563265
08 HashMap 源码解析	代码地址是什么？方便贴一下吗？
09 TreeMap 和 LinkedHashMap 核心源码解析	<div><div>👍 0</div><div>回复</div><div>2019-11-29</div></div>
10 Map源码会问哪些面试题	
11 HashSet、TreeSet 源码解析	
12 彰显细节：看集合源码对我们实际工作的帮助和应用	千学不如一看，千看不如一练
13 差异对比：集合在 Java 7 和 8 有何不同和改进	
14 简化工作：Guava Lists Maps 实际工作运用和源码	
第3章 并发集合类	
15 CopyOnWriteArrayList 源码解析和设计思路	
16 ConcurrentHashMap 源码解析和设计思路	
17 并发 List、Map源码面试题	
18 场景集合：并发 List、Map的应用	