主要内容

# Socket通道

新的socket通道类可以非阻塞模式并且是可选择的。这两个性能可以激活大程序（如网络服务区和中间件组件）巨大的可伸缩性和灵活性。本节中我们会看到，再也没有为两个socket连接使用一个线程的必要了，也避免了管理大量线程所需的上下文交换总开销。借助新的NIO类，一个或几个线程就可以管理成百上千的活动。socket连接了并且只有很少甚至可能没有性能损失。

全部socket通道类（DatagramChannel、SocketChannel和ServerSocketChannel）在被实例化时都会创建一个对等socket对象。这些事我们所熟悉的来自java.net的类（Socket、ServerSocket和DatagramSocket），它们已经被更新以识别通道。对等socket可以通过调用socket()方法从一个通道上获取。此外，这三个java.net类现在都有getChannel()方法。

虽然每个socket通道（在java.nio.channels包中）都有一个关联的java.net socket对象，却并非所有的socket都有一个关联的通道。如果您用传统方式（直接实例化）创建了一个Socket对象，它就不会有关联的SocketChannel并且它的getChannel()方法将总是返回null。

**非阻塞模式**

Socket通道可以在非阻塞模式下运行。这个说法虽然简单却有着深远的含义。传统Java Socket的阻塞性质曾经是Java程序可伸缩性的重要制约之一。非阻塞I/O是许多复杂的、高性能的程序构建的基础。





































