https://www.cnblogs.com/wanggangjia/p/5222759.html

Socket又称套接字，是连接运行在网络上两个程序间的双向通讯的端点。

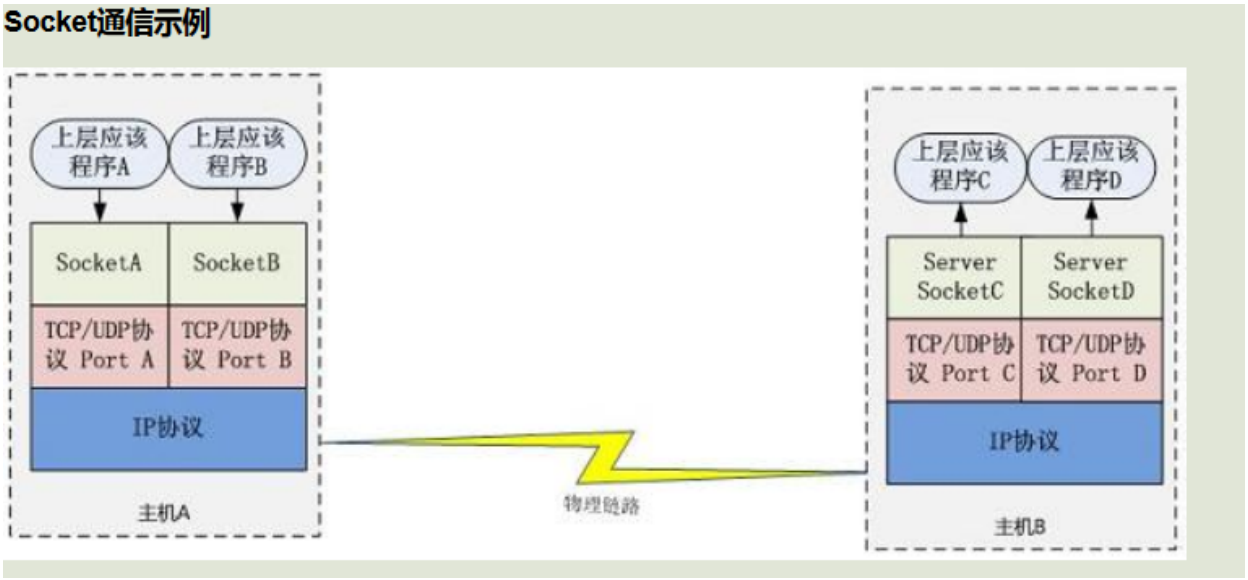
我们知道Java中的socket编程，对于面向连接的编程来说（包括我们每次在网页上向服务器请求资源时），它的第一步就是建立双方之间的通讯链路，而这个过程其实就是TCP的连接，这时就可以联想计算机网络中的TCP连接的三次握手，而在断开时就可以联想TCP断开连接的四次挥手等等

一、使用Socket进行网络通信的过程

服务端：服务器程序将一个套接字绑定到一个特定的端口，并通过此套接字等待和监听客户端的连接请求。

客户端：客户端程序根据你服务器所在的主机名和端口号发出连接请求。

两者之间的通信是通过Socket完成的，我们可以认为Socket是两个城市之间的交通工具，有了它，就可以在两个城市之间穿梭了。



主机A的应用程序和主机B的应用程序通信，必须通过Socket建立连接，而建立Socket必须由底层的TCP/IP协议来建立TCP连接。建立TCP连接需要底层IP协议来寻址网络中的主机。IP地址只能帮助我们找到目标主机，但是一个主机上面有多个应用程序，如何才能找到我们需要的应用程序，这个时候就可以通过端口号来指定了。

服务器端：

```
1     public static void main(String[] args) throws IOException
2     {
3         //创建一个ServerSocket，用于监听客户端Socket连接请求
4         ServerSocket ss = new ServerSocket(8888);
5         System.out.println("server start");
6         //采用循环方式监听客户端的请求
7         while(true)
8         {
9             //侦听并接受到此套接字的连接。此方法在连接传入之前一直阻塞。
10            Socket socket = ss.accept();
11            OutputStream os = socket.getOutputStream();
12            PrintStream ps = new PrintStream(os);
13            ps.print("您好，您收到了来自服务端的中秋祝福");
14            ps.close();
15            os.close();
16            socket.close();
17        }
18    }
```

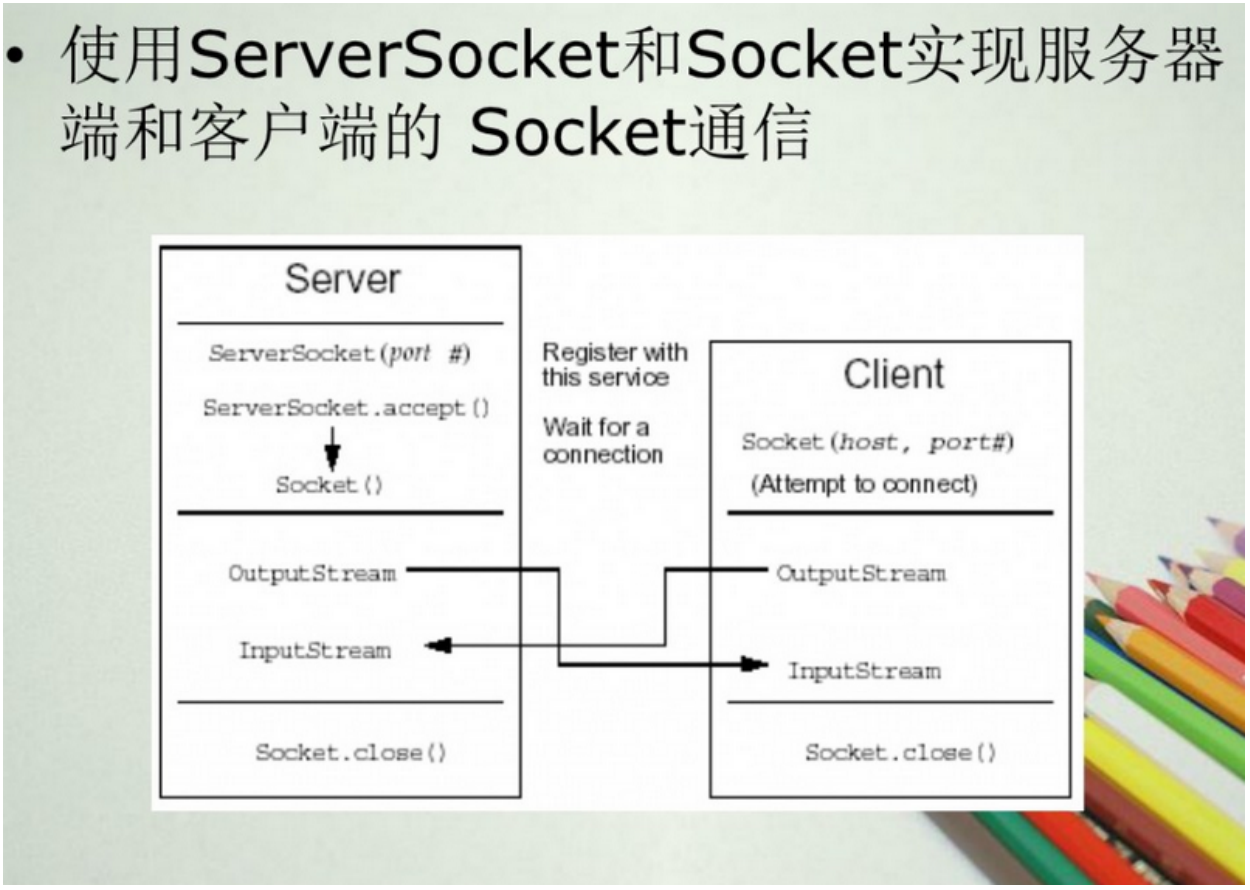
server start

客户端：

```
1     public static void main(String[] args) throws IOException, Exception
2     {
3         Socket socket = new Socket("localhost",8888);
4         InputStream is = socket.getInputStream();
5         BufferedReader br = new BufferedReader(new InputStreamReader(is));
6         String str = br.readLine();
7         System.out.println(str);
8         br.close();
9         is.close();
10        socket.close();
11    }
```

执行结果：

```
您好，您收到了来自服务端的中秋祝福
```



3、首先在server端，指定端口号创建serverSocket对象，通过serverSocket的accpet方法获取套接字，这个方法的特点是：**侦听并接受到此套接字的连接，此方法在连接传入之前一直阻塞。**这也就意味着，如果没有客户端连接请求过来，服务端会一致阻塞在这里。

4、后面的代码就是通过套接字socket可以得到输入输出流，到此为止，就是I/O的内容了。

5、在客户端这边，通过指定的服务器主机名和服务器监听的端口号，得到套接字Socket，这个时候就表示服务端和客户端的连接已经建立了，然后通过输入输出流来进行通信了。

三、半关闭的socket

在上面的Demo中，我们是以行作为通信的最小数据单位，服务器端也是逐行进行处理的。但是我们在大多数场景下，通信的数据单位是多行的，这时候Socket的输出流如何表达输出的数据已经结束？

在IO学习过程中提到过，如何要表示输出已经结束，则通过关闭输出流来实现，但是在socket中是行不通的，因为关闭socket，会导致无法再从该socket中读取数据了。为了解决这种问题，java提供了两个半关闭的方法：

1、shutdownInput():关闭该Socket的输入流，程序还可以通过该Socket的输出流输出数据。

2、shutdownOutput():关闭该Socket的输出流，程序还可以通过该Socket的输入流读取数据。

如果我们对同一个Socket实例先后调用shutdownInput和shutdownOutput方法，该Socket实例依然没有被关闭，只是该Socket既不能输出数据，也不能读取数据。