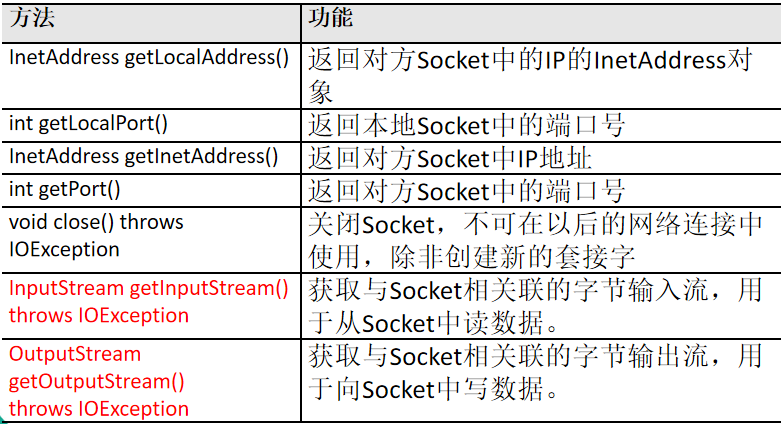
* **Socket**
* 利用**套接字(Socket)——（IP地址+端口号）**开发网络应用程序早已被广泛的采用，以至于成为事实上的标准。
* 通信的两端都要有Socket，是两台机器间通信的端点。
* 网络通信其实就是Socket间的通信。
* Socket允许程序把网络连接当成一个流，**数据在两个Socket间通过IO传输**。
* **一般主动发起通信的应用程序属客户端，等待通信请求的为服务端**。
* **Java语言的基于套接字编程分为服务端编程和客户端编程，其通信模型如图所示：**

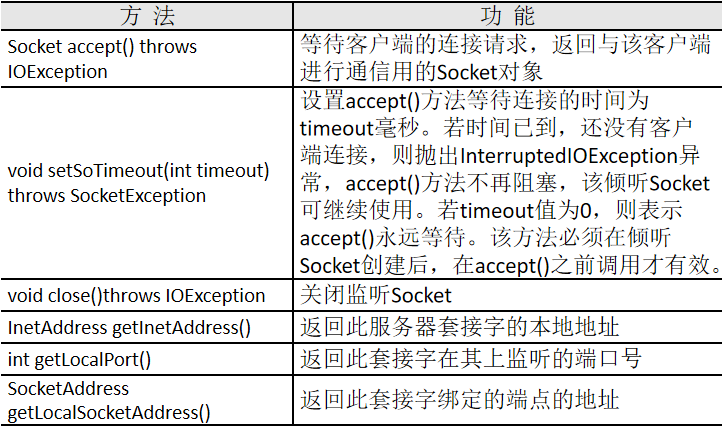


基于TCP的Socket通信

* **Socket类的常用方法**



* **ServerSocket类的常用方法**



* **客户端Socket的工作过程包含以下四个基本的步骤：**
* **1）创建Socket**：根据指定服务端的IP地址或端口号构造Socket类对象。若服务器端响应，则建立客户端到服务器的通信线路。若连接失败，会出现异常。
* **2）打开连接到Socket的输入/出流**：使用**getInputStream()**方法获得输入流，使用**getOutputStream()**方法获得输出流，进行数据传输
* **3）按照一定的协议对Socket进行读/写操作**：通过输入流读取服务器放入线路的信息（但不能读取自己放入线路的信息），通过输出流将信息写入线程。
* **4）关闭Socket**：断开客户端到服务器的连接，释放线路。
* 客户端程序可以使用Socket类创建对象，**创建的同时会自动向服务器放发起连接**。Socket的构造方法是：
  + Socket(String host,int port)

throws UnknownHostException,IOException：**向服务器(域名是host。端 口号为 port)发起TCP连接**，若成功，则创建Socket对象，否则抛出异常。

* + Socket(InetAddress address,int port)

throws IOException：**根据InetAddress对象所表示的IP地址以及端口号 port发起连接**。

* **服务器程序的工作过程包含以下四个基本的步骤：**
* **1）调用ServerSocket(int port)**：创建一个服务器端套接字，并绑定到指定端口上。用于监听客户端的请求。
* **2）调用accept()**：监听连接请求，如果客户端请求连接，则接受连接，返回通信套接字对象。
* **3）调用该Socket类对象的getOutputStream()和getInputStream()**：获取输出流和输入流，开始网络数据的发送和接收。
* 4）**关闭ServerSocket和Socket对象**：客户端访问结束，关闭通信套接字。
* ServerSocket对象负责等待客户端请求建立套接字连接，类似邮局某个窗口中的业务员。也就是说，**服务器必须事先建立一个等待客户请求建立套接字连接的ServerSocket对象**。
* 所谓“接收”客户的套接字请求，就是accept()方法会返回一个Socket对象。
* **客户端——服务端**
* 客户端：
  + 自定义
  + 浏览器
* 服务端：
  + 自定义
  + Tomcat服务器

*/\*\*  
 \* 客户端  
 \*/*@Test  
**public void** client() {  
  
 Socket socket = **null**;  
 OutputStream os = **null**;  
 **try** {  
 *//1）创建一个Socket的对象，通过构造器指明服务端的IP地址，以及其接受程序的端口号* socket = **new** Socket(InetAddress.*getByName*(**"127.0.0.1"**), 9090);  
  
 *//2）getOutputStream()，发送数据，方法返回OutputStream的对象* os = socket.getOutputStream();  
  
 *//3）具体的输出过程* os.write(**"我是客户端，请多关照"**.getBytes());  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **finally** {  
 *//4）关闭相应的流和Socket* **if** (os != **null**) {  
 **try** {  
 os.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 **if** (socket != **null**) {  
 **try** {  
 socket.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
  
}  
  
*/\*\*  
 \* 服务端  
 \*/*@Test  
**public void** server() {  
  
 ServerSocket ss = **null**;  
 Socket s = **null**;  
 InputStream is = **null**;  
 **try** {  
 *//1）创建一个ServerSocket的对象，通过构造器指明自身的端口号* ss = **new** ServerSocket(9090);  
 *//2）调用其accept()方法，返回一个Socket的对象* s = ss.accept();  
 *//3）调用Socket对象的getInputStream()获取一个从客户端发送过来的输入流* is = s.getInputStream();  
 *//4）对获取的输入流进行的操作* **byte**[] b = **new byte**[20];  
 **int** len;  
 **while** ((len = is.read(b)) != -1) {  
 String str = **new** String(b, 0, len);  
 System.***out***.println(str);  
 }  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 } **finally** {  
 *//5）关闭相应的流以及Socket、ServerSocket对象* **if** (is != **null**) {  
 **try** {  
 is.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 **if** (s != **null**) {  
 **try** {  
 s.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 **if** (ss != **null**) {  
 **try** {  
 ss.close();  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
}

*/\*\*  
 \* TCP编程例三：从客户端发送文件给服务端，服务端保存到本地。并返回“发送成功”给客户端。并关闭相应的连接。  
 \* 如下的程序，处理异常时，要使用try-catch-finally!!本例仅为了书写方便  
 \*/***public class** TestTCP3 {  
  
 */\*\*  
 \* 客户端  
 \*/* @Test  
 **public void** client() **throws** Exception {  
 *//1.创建Socket的对象* Socket socket = **new** Socket(InetAddress.*getByName*(**"127.0.0.1"**), 9898);  
 *//2.从本地获取一个文件发送给服务端* OutputStream os = socket.getOutputStream();  
 FileInputStream fis = **new** FileInputStream(**new** File(**"1.jpg"**));  
 **byte**[] b = **new byte**[1024];  
 **int** len;  
 **while** ((len = fis.read(b)) != -1) {  
 os.write(b, 0, len);  
 }  
  
 socket.shutdownOutput();  
  
 *//3.接收来自于服务端的信息* InputStream is = socket.getInputStream();  
 **byte**[] b1 = **new byte**[1024];  
 **int** len1;  
 **while** ((len1 = is.read(b1)) != -1) {  
 String str = **new** String(b1, 0, len1);  
 System.***out***.print(str);  
 }  
 *//4.关闭相应的流和Socket对象* is.close();  
 os.close();  
 fis.close();  
 socket.close();  
 }  
  
 */\*\*  
 \* 服务端  
 \*/* @Test  
 **public void** server() **throws** Exception {  
 *//1.创建一个ServerSocket的对象* ServerSocket ss = **new** ServerSocket(9898);  
 *//2.调用其accept()方法，返回一个Socket的对象* Socket s = ss.accept();  
 *//3.将从客户端发送来的信息保存到本地* InputStream is = s.getInputStream();  
 FileOutputStream fos = **new** FileOutputStream(**new** File(**"2.jpg"**));  
 **byte**[] b = **new byte**[1024];  
 **int** len;  
 **while** ((len = is.read(b)) != -1) {  
 fos.write(b, 0, len);  
 }  
 System.***out***.println(**"收到来自于"** + s.getInetAddress().getHostAddress() + **"的文件"**);  
 *//4.发送"接收成功"的信息反馈给客户端* OutputStream os = s.getOutputStream();  
 os.write(**"你发送的图片我已接收成功！"**.getBytes());  
 *//5.关闭相应的流和Socket及ServerSocket的对象* os.close();  
 fos.close();  
 is.close();  
 s.close();  
 ss.close();  
 }  
}