李晨玮

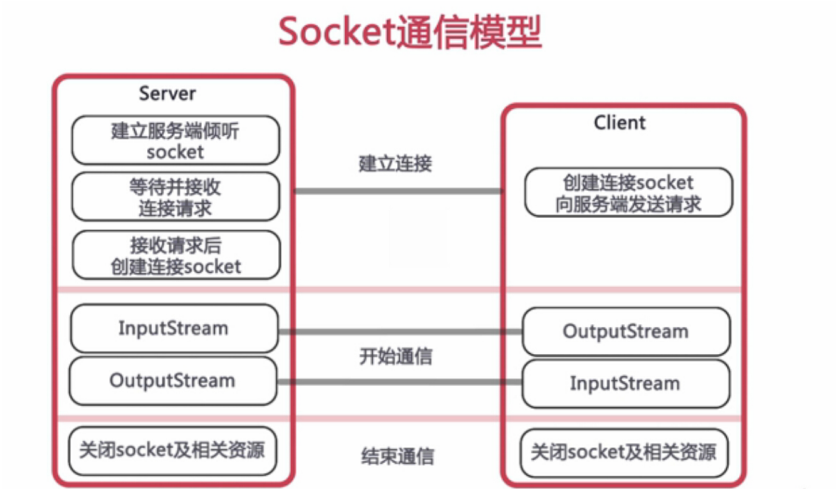
当你的才华还撑不起你的野心时，你就应该静下心来好好学习。

基于Tcp协议的简单Socket通信实例（JAVA）

好久没写博客了，前段时间忙于做项目，耽误了些时间，今天开始继续写起~

今天来讲下关于Socket通信的简单应用，关于什么是Socket以及一些网络编程的基础，这里就不提了，只记录最简单易懂实用的东

1、首先先来看下基于TCP协议Socket服务端和客户端的通信模型：



Socket通信步骤：（简单分为4步）

1.建立服务端ServerSocket和客户端Socket

2.打开连接到Socket的输出输入流

3.按照协议进行读写操作

4.关闭相对应的资源

2、相关联的API：

1.首先先来看下ServerSocket

类 ServerSocket

此类实现服务器套接字。服务器套接字等待请求通过网络传入。它基于该请求执行某些操作，然后可能向请求者返回结果。

服务器套接字的实际工作由 `SocketImpl` 类的实例执行。应用程序可以更改创建套接字实现的套接字工厂来配置它自身，从而创建墙的套接字。

一些重要的方法：（具体大家查看官方api吧）

`ServerSocket(int port, int backlog)`
利用指定的 backlog 创建服务器套接字并将其绑定到指定的本地端口号。

`bind(SocketAddress endpoint, int backlog)`
将 `ServerSocket` 绑定到特定地址（IP 地址和端口号）。

`accept()`
侦听并接受到此套接字的连接

`getInetAddress()`
返回此服务器套接字的本地地址。

`close()`
关闭此套接字。

2.再来看下Socket

类 Socket

此类实现客户端套接字（也可以就叫“套接字”）。套接字是两台机器间通信的端点。

套接字的实际工作由 `SocketImpl` 类的实例执行。应用程序通过更改创建套接字实现的套接字工厂可以配置它自身，以创建适合本地防火墙的套接字。

一些重要的方法：（具体大家查看官方api吧）

`Socket(InetAddress address, int port)`
创建一个流套接字并将其连接到指定 IP 地址的指定端口号。

`getInetAddress()`
返回套接字连接的地址。

`shutdownInput()`
此套接字的输入流置于“流的末尾”。

`shutdownOutput()`
禁用此套接字的输出流。

`close()`
关闭此套接字。

3、代码实现：（注释很全，这里就不详细多说了）

服务端Server.java

1.创建ServerSocket对象，绑定并监听端口

2.通过accept监听客户端的请求

3.建立连接后，通过输出输入流进行读写操作

4.关闭相关资源

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;


public class Server {

    /**
     * Socket服务端
     */
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket=new ServerSocket(8888);
            System.out.println("服务端已启动，等待客户端连接..");
            Socket socket=serverSocket.accept();//侦听并接受到此套接字的连接,返回一个Socket对象


            //根据输入输出流和客户端连接
            InputStream inputStream=socket.getInputStream();//得到一个输入流，接收客户端传递的信息
            InputStreamReader inputStreamReader=new InputStreamReader(inputStream);//提高效率，将自己字节流转为字符流
            BufferedReader bufferedReader=new BufferedReader(inputStreamReader);//加入缓冲区
            String temp=null;
            String info="";
            while((temp=bufferedReader.readLine())!=null){
                info+=temp;
                System.out.println("已接收到客户端连接");
                System.out.println("服务端接收到客户端信息："+info+",当前客户端ip为："+socket.getInetAddress().getHostAd
            }

            OutputStream outputStream=socket.getOutputStream();//获取一个输出流，向服务端发送信息
            PrintWriter printWriter=new PrintWriter(outputStream);//将输出流包装成打印流
            printWriter.print("你好，服务端已接收到您的信息");
            printWriter.flush();
            socket.shutdownOutput();//关闭输出流
```

```
40
41
42
43              //关闭相对应的资源
44              printWriter.close();
45              outputStream.close();
46              bufferedReader.close();
47              inputStream.close();
48              socket.close();
49
50          } catch (IOException e) {
51              e.printStackTrace();
52          }
53      }
54
55  }
```

客户端Client.java

1.创建Socket对象，指定服务端的地址和端口号

2.建立连接后，通过输出输入流进行读写操作

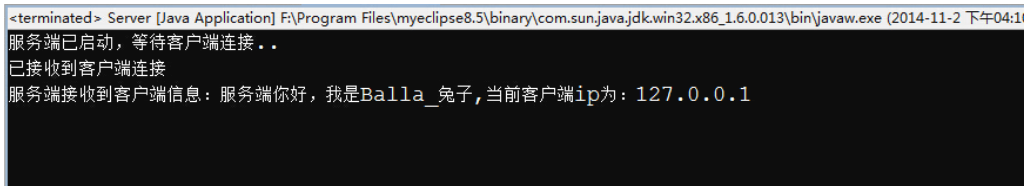
3.通过输出输入流获取服务器返回信息

4.关闭相关资源

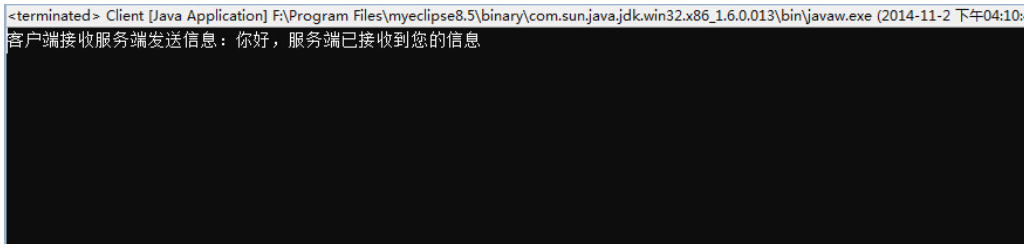
```
1  import java.io.BufferedReader;
2  import java.io.IOException;
3  import java.io.InputStream;
4  import java.io.InputStreamReader;
5  import java.io.OutputStream;
6  import java.io.PrintWriter;
7  import java.net.Socket;
8  import java.net.UnknownHostException;
9
10
11 public class Client {
12
13     /**
14      * Socket客户端
15      */
16     public static void main(String[] args) {
17         try {
18             //创建Socket对象
19             Socket socket=new Socket("localhost",8888);
20
21             //根据输入输出流和服务端连接
22             OutputStream outputStream=socket.getOutputStream();//获取一个输出流，向服务端发送信息
23             PrintWriter printWriter=new PrintWriter(outputStream);//将输出流包装成打印流
24             printWriter.print("服务端你好，我是Balla_兔子");
25             printWriter.flush();
26             socket.shutdownOutput();//关闭输出流
27
28             InputStream inputStream=socket.getInputStream();//获取一个输入流，接收服务端的信息
29             InputStreamReader inputStreamReader=new InputStreamReader(inputStream);//包装成字符流，提高效率
30             BufferedReader bufferedReader=new BufferedReader(inputStreamReader);//缓冲区
31             String info="";
32             String temp=null;//临时变量
33             while((temp=bufferedReader.readLine())!=null){
34                 info+=temp;
35                 System.out.println("客户端接收服务端发送信息：“+info);
36             }
37
38             //关闭相对应的资源
39             bufferedReader.close();
40             inputStream.close();
41             printWriter.close();
42             outputStream.close();
43             socket.close();
44         } catch (UnknownHostException e) {
45             e.printStackTrace();
46         } catch (IOException e) {
47             e.printStackTrace();
48         }
49
50     }
51
52 }
```

4、效果截图：

服务端：



客户端：



以上代码实现了单客户端和服务端的连接，若要实现多客户端操作，需要涉及到多线程，只要你把每个接收到的Socket对象单独开作，然后用一个死循环while(true)去监听端口就行，这边直接给代码了

线程操作类：SocketThread.java

```
1 import java.io.BufferedReader;
2 import java.io.IOException;
3 import java.io.InputStream;
4 import java.io.InputStreamReader;
5 import java.io.OutputStream;
6 import java.io.PrintWriter;
7 import java.net.Socket;
8
9 /**
10  * Socket多线程处理类 用来处理服务端接收到的客户端请求（处理Socket对象）
11  */
12 public class SocketThread extends Thread {
13     private Socket socket;
14
15     public SocketThread(Socket socket) {
16         this.socket = socket;
17     }
18
19     public void run() {
20         // 根据输入输出流和客户端连接
21         try {
22             InputStream inputStream = socket.getInputStream();
23             // 得到一个输入流，接收客户端传递的信息
24             InputStreamReader inputStreamReader = new InputStreamReader(
25                     inputStream);// 提高效率，将自己字节流转为字符流
26             BufferedReader bufferedReader = new BufferedReader(
27                     inputStreamReader);// 加入缓冲区
28             String temp = null;
29             String info = "";
30             while ((temp = bufferedReader.readLine()) != null) {
31                 info += temp;
32                 System.out.println("已接收到客户端连接");
33                 System.out.println("服务端接收到客户端信息：" + info + ",当前客户端ip为："
34                         + socket.getInetAddress().getHostAddress());
35             }
36
37             OutputStream outputStream = socket.getOutputStream();// 获取一个输出流，向服务端发送信息
38             PrintWriter printWriter = new PrintWriter(outputStream);// 将输出流包装成打印流
39             printWriter.print("你好，服务端已接收到您的信息");
40             printWriter.flush();
41             socket.shutdownOutput();// 关闭输出流
42
43             // 关闭相对应的资源
44             bufferedReader.close();
45             inputStream.close();
46             printWriter.close();
47             outputStream.close();
48         } catch (IOException e) {
```

```
            e.printStackTrace();
        }

    }

}
```

服务端类：Server.java

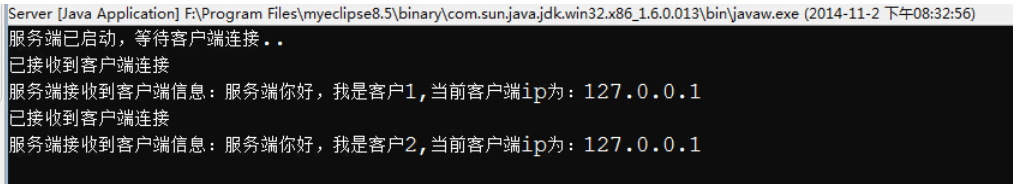
```
import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;

public class Server {

    /**
     * Socket服务端
     */
    public static void main(String[] args) {
        try {
            ServerSocket serverSocket = new ServerSocket(8888);
            System.out.println("服务端已启动，等待客户端连接..");

            while (true) {
                Socket socket = serverSocket.accept();// 侦听并接受到此套接字的连接,返回一个Socket对象
                SocketThread socketThread = new SocketThread(socket);
                socketThread.start();
            }

        } catch (IOException e) {
            e.printStackTrace();
        }
    }

}
```

客户端类:Client.java

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;


public class Client {

    /**
     * Socket客户端
     */
    public static void main(String[] args) {
        try {
            //创建Socket对象
            Socket socket=new Socket("localhost",8888);

            //根据输入输出流和服务端连接
            OutputStream outputStream=socket.getOutputStream();//获取一个输出流，向服务端发送信息
            PrintWriter printWriter=new PrintWriter(outputStream);//将输出流包装成打印流
            printWriter.print("服务端你好，我是客户1");
            printWriter.flush();
            socket.shutdownOutput();//关闭输出流

            InputStream inputStream=socket.getInputStream();//获取一个输入流，接收服务端的信息
            InputStreamReader inputStreamReader=new InputStreamReader(inputStream);//包装成字符流，提高效率
            BufferedReader bufferedReader=new BufferedReader(inputStreamReader);//缓冲区
            String info="";
            String temp=null;//临时变量
            while((temp=bufferedReader.readLine())!=null){
                info+=temp;
                System.out.println("客户端接收服务端发送信息：“+info);
            }

            //关闭相对应的资源
            bufferedReader.close();
            inputStream.close();
```

```
            printWriter.close();
            outputStream.close();
            socket.close();
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

}
```

看下效果实现图：



这里只是抛砖引玉，在实际开发中，基于Socket编程，一般传递的并非字符串，很多情况下是对象，我们可以使用ObjectOutputS流对象序列化。

例如：

```
            OutputStream outputStream = socket.getOutputStream();
            ObjectOutputStream objectOutputStream=new ObjectOutputStream(outputStream);
            User user=new User("admin","123456");
            objectOutputStream.writeObject(user);
```

作者：Balla_兔子
出处：http://www.cnblogs.com/lichenwei/

分类: JAVA

标签: JAVA, socket, 通信, 服务端, 客户端

« 上一篇： Gson简要使用笔记
» 下一篇： 安卓智能聊天机器人开发（一）

posted @ 2014-11-02 16:26 李晨玮 阅读(45563) 评论

评论列表

#1楼 [楼主] 2014-11-03 16:12 李晨玮
@ 奥巴马说你代码写的好
- -。。啥意思?

支持

#2楼 2017-06-15 15:44 tong__

推荐一个socket编程的异步socket类库，https://github.com/typ0520/bizsocket，对一些业务场景做了支持
断线重连
一对一请求
通知、粘性通知
串行请求合并
包分片处理
缓存
拦截器
支持rxjava，提供类似于retrofit的支持

支持

刷新评论 刷新

注册用户登录后才能发表评论，请 登录 或 注册， 访问 网站首页。

相关博文：
· 基于Qt的TCPSocket通信编程研究（八）
· Java实现基于TCP协议的Socket通信
· 多线程TCP的socket通信
· 通过Socket实现TCP编程
· 基于TCP和UDP的Socket实现（JAVA）