

[首页](#) [ANDROID](#) [互联网](#) [杂乱无章](#) [科技资讯](#) [程序员人生](#) [程序员笑话](#) [编程技术](#) [网址导航](#)

## 7.6.2 基于TCP协议的Socket通信(1)

分类 **Android 基础入门教程**

**Android 基础入门教程(Q群号 : 153836263)**

### 本节引言：

上一节的概念课枯燥无味是吧，不过总有点收获是吧，本节开始我们来研究基于TCP协议的Socket 通信，先来了解下Socket的概念，以及Socket通信的模型，实现Socket的步骤，以及作为Socket服务端与客户端的两位各做要做什么事情！好的，我们由浅入深来扣这个Socket吧！

- 1.0 Android基础入门教程
  - 1.0.1 2015年最新Android基...
  - 1.1 背景相关与系统架构分析
  - 1.2 开发环境搭建
    - 1.2.1 使用Eclipse + ADT + S...
    - 1.2.2 使用Android Studio开...
  - 1.3 SDK更新不了问题解决
  - 1.4 Genymotion模拟器安装
  - 1.5.1 Git使用教程之本地仓...
  - 1.5.2 Git之使用GitHub搭建...
  - 1.6 .9(九妹)图片怎么玩
  - 1.7 界面原型设计
  - 1.8 工程相关解析(各种文件...
  - 1.9 Android程序签名打包
  - 1.11 反编译APK获取代码&...
- 2.1 View与ViewGroup的概念
  - 2.2.1 LinearLayout(线性布局)
  - 2.2.2 RelativeLayout(相对布...
  - 2.2.3 TableLayout(表格布局)
  - 2.2.4 FrameLayout(帧布局)
  - 2.2.5 GridLayout(网格布局)
  - 2.2.6 AbsoluteLayout(绝对...
  - 2.3.1 TextView(文本框)详解
  - 2.3.2 EditText(输入框)详解
  - 2.3.3 Button(按钮)与ImageB...
  - 2.3.4 ImageView(图像视图)
  - 2.3.5.RadioButton(单选按钮...
  - 2.3.6 开关按钮ToggleButton...
  - 2.3.7 ProgressBar(进度条)
  - 2.3.8 SeekBar(拖动条)
  - 2.3.9 RatingBar(星级评分条)
  - 2.4.1 ScrollView(滚动条)

### 1.什么是Socket？



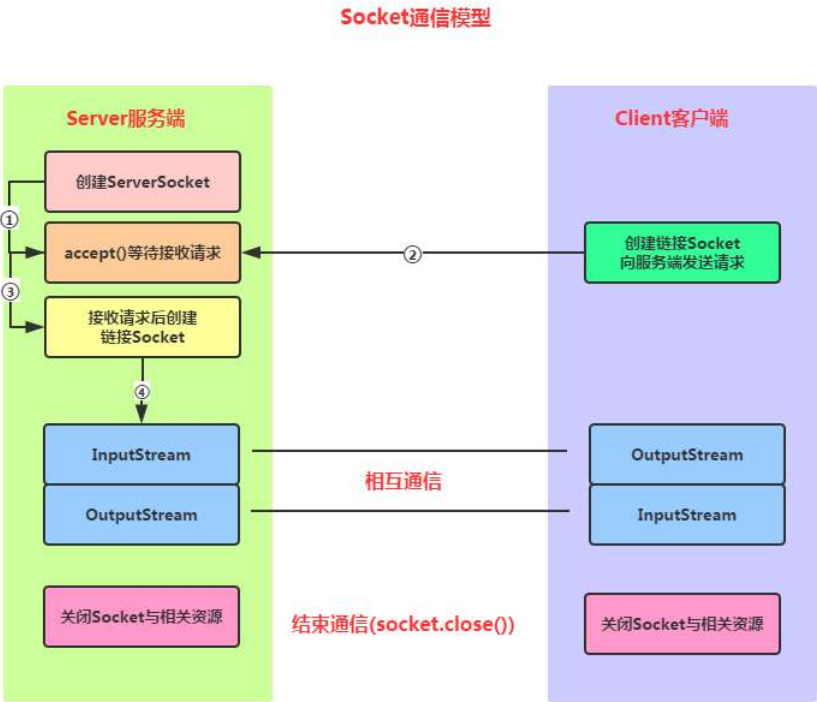
什么是Socket?

Socket(套接字),用来描述IP地址和端口,是通信链的句柄,应用程序可以通过Socket向网络发送请求或者应答网络请求!Socket是支持TCP/IP协议的网络通信的基本操作单元,是对网络通信过程中端点的抽象表示,包含了进行网络通信所必须的四种信息:连接所使用的协议;本地主机的IP地址;本地进程的协议端口;远地主机的IP地址以及远地进程的协议端口

### 2.Socket通信模型：



反馈



**Socket通信实现步骤解析：**

- Step 1：**创建ServerSocket和Socket
- Step 2：**打开连接到的Socket的输入/输出流
- Step 3：**按照协议对Socket进行读/写操作
- Step 4：**关闭输入输出流，以及Socket

好的，我们接下来写一个简单的例子，开启服务端后，客户端点击按钮然后链接服务端，并向服务端发送一串字符串，表示通过Socket链接上服务器~

**3.Socket服务端的编写：**

**服务端要做的事有这些：**

- Step 1：**创建ServerSocket对象，绑定监听的端口
- Step 2：**调用accept()方法监听客户端的请求
- Step 3：**连接建立后，通过输入流读取客户端发送的请求信息
- Step 4：**通过输出流向客户端发送响应信息
- Step 5：**关闭相关资源

**代码实现：**

直接在Eclipse下创建一个Java项目，然后把Java代码贴进去即可！

```
public class SocketServer {  
    public static void main(String[] args) throws IOExcepti  
on {  
        //1.创建一个服务器端Socket，即ServerSocket，指定绑
```

- 2.4.2 Date & Time组件(上)
- 2.4.3 Date & Time组件(下)
- 2.4.4 Adapter基础讲解
- 2.4.5 ListView简单实用
- 2.4.6 BaseAdapter优化
- 2.4.7 ListView的焦点问题
- 2.4.8 ListView之checkbox错...
- 2.4.9 ListView的数据更新问题
- 2.5.0 构建一个可复用的自定...
- 2.5.1 ListView Item多布局的...
- 2.5.2 GridView(网格视图)的...
- 2.5.3 Spinner(列表选项框)...
- 2.5.4 AutoCompleteTextVie...
- 2.5.5 ExpandableListView(...
- 2.5.6 ViewPager(翻转视图)...
- 2.5.7 Toast(吐司)的基本使用
- 2.5.8 Notification(状态栏通...
- 2.5.9 AlertDialog(对话框)详解
- 2.6.0 其他几种常用对话框基...
- 2.6.1 PopupWindow(悬浮框...
- 2.6.2 菜单(Menu)
- 2.6.3 ViewPager的简单使用
- 2.6.4 DrawerLayout(官方侧...
- 3.1.1 基于监听的事件处理机制
- 3.2 基于回调的事件处理机制
- 3.3 Handler消息传递机制浅析
- 3.4 TouchListener PK OnTo...
- 3.5 监听EditText的内容变化
- 3.6 响应系统设置的事件(Co...
- 3.7 AsyncTask异步任务
- 3.8 Gestures(手势)
- 4.1.1 Activity初学乍练
- 4.1.2 Activity初窥门径
- 4.1.3 Activity登堂入室
- 4.2.1 Service初涉
- 4.2.2 Service进阶
- 4.2.3 Service精通
- 4.3.1 BroadcastReceiver牛...
- 4.3.2 BroadcastReceiver庖...
- 4.4.1 ContentProvider初探

定的端口，并监听此端口

```
ServerSocket serverSocket = new ServerSocket(12345);

InetAddress address = InetAddress.getLocalHost();

String ip = address.getHostAddress();
Socket socket = null;
//2.调用accept()等待客户端连接
System.out.println("~~~服务端已就绪，等待客户端接入~，服务端ip地址：" + ip);
socket = serverSocket.accept();
//3.连接后获取输入流，读取客户端信息
InputStream is=null;
InputStreamReader isr=null;
BufferedReader br=null;
OutputStream os=null;
PrintWriter pw=null;
is = socket.getInputStream(); //获取输入流
isr = new InputStreamReader(is,"UTF-8");
br = new BufferedReader(isr);
String info = null;
while((info=br.readLine())!=null){//循环读取客户端的信息

        System.out.println("客户端发送过来的信息"
+ info);
    }
    socket.shutdownInput();//关闭输入流
    socket.close();
}
```

然后我们把代码run起来，控制台会打印：

```
SocketServer [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (2015年9月16日)
~~~服务端已就绪，等待客户端接入~，服务端ip地址： 172.16.2.54
```

好的，接下来到Android客户端了！

## 4.Socket客户端的编写：

客户端要做的事有这些：

**Step 1：**创建Socket对象，指明需要链接的服务器的地址和端号

**Step 2：**链接建立后，通过输出流向服务器发送请求信息

**Step 3：**通过输出流获取服务器响应的信息

**Step 4：**关闭相关资源

代码实现：

MainActivity.java：

```
public class MainActivity extends AppCompatActivity implements
View.OnClickListener {
```

4.4.2 ContentProvider再探...

4.5.1 Intent的基本使用

4.5.2 Intent之复杂数据的传递

5.1 Fragment基本概述

5.2.1 Fragment实例精讲—...

5.2.2 Fragment实例精讲—...

5.2.3 Fragment实例精讲—...

5.2.4 Fragment实例精讲—...

5.2.5 Fragment实例精讲—...

6.1 数据存储与访问之——文...

6.2 数据存储与访问之——S...

6.3.1 数据存储与访问之——...

6.3.2 数据存储与访问之——...

7.1.1 Android网络编程要学...

7.1.2 Android Http请求头与...

7.1.3 Android HTTP请求方...

7.1.4 Android HTTP请求方...

7.2.1 Android XML数据解析

7.2.2 Android JSON数据解析

7.3.1 Android 文件上传

7.3.2 Android 文件下载（1）

7.3.3 Android 文件下载（2）

7.4 Android 调用WebService

7.5.1 WebView(网页视图)基...

7.5.2 WebView和JavaScip...

7.5.3 Android 4.4后WebVie...

7.5.4 WebView文件下载

7.5.5 WebView缓存问题

7.5.6 WebView处理网页返...

7.6.1 Socket学习网络基础准备

7.6.2 基于TCP协议的Socket...

7.6.3 基于TCP协议的Socket...

7.6.4 基于UDP协议的Socke...

8.1.1 Android中的13种Draw...

8.1.2 Android中的13种Draw...

8.1.3 Android中的13种Draw...

8.2.1 Bitmap(位图)全解析 P...

8.2.2 Bitmap引起的OOM问题

8.3.1 三个绘图工具类详解

8.3.2 绘图类实战示例

```

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Button btn_accept = (Button) findViewById(R.id.btn_acce
pt);
    btn_accept.setOnClickListener(this);
}

@Override
public void onClick(View v) {
    new Thread() {
        @Override
        public void run() {
            try {
                acceptServer();
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }.start();
}

private void acceptServer() throws IOException {
    //1.创建客户端Socket, 指定服务器地址和端口
    Socket socket = new Socket("172.16.2.54", 12345);
    //2.获取输出流, 向服务器端发送信息
    OutputStream os = socket.getOutputStream();//字节输出流
    PrintWriter pw = new PrintWriter(os);//将输出流包装为打印
流

    //获取客户端的IP地址
    InetAddress address = InetAddress.getLocalHost();
    String ip = address.getHostAddress();
    pw.write("客户端: ~" + ip + "~ 接入服务器!!");
    pw.flush();
    socket.shutdownOutput();//关闭输出流
    socket.close();
}
}

```

因为Android不允许在主线程(UI线程)中做网络操作, 所以这里需要我们自己另开一个线程来连接Socket!

### 运行结果:

点击按钮后, 服务端控制台打印:

```

<terminated> SocketServer [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.e
~~~服务端已就绪, 等待客户端接入~, 服务端ip地址: 172.16.2.54
客户端发送过来的信息客户端: ~127.0.0.1~ 接入服务器!!

```

## 5.增强版案例: 小猪简易聊天室

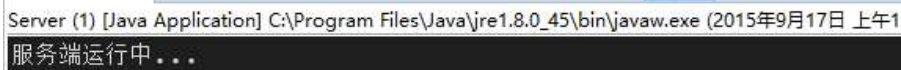
只是点击个按钮, 然后服务器返回一串信息, 肯定是很无趣的是吧,

- 8.3.3 Paint API之—— Mask...
- 8.3.4 Paint API之—— Xferm...
- 8.3.5 Paint API之—— Xferm...
- 8.3.6 Paint API之—— Xferm...
- 8.3.7 Paint API之—— Xferm...
- 8.3.8 Paint API之—— Xferm...
- 8.3.9 Paint API之—— Color...
- 8.3.10 Paint API之—— Colo...
- 8.3.11 Paint API之—— Colo...
- 8.3.12 Paint API之—— Path...
- 8.3.13 Paint API之—— Sha...
- 8.3.14 Paint几个枚举/常量值...
- 8.3.15 Paint API之——Type...
- 8.3.16 Canvas API详解(Part 1)
- 8.3.17 Canvas API详解(Part...
- 8.3.18 Canvas API详解(Part...
- 8.4.1 Android动画合集之帧...
- 8.4.2 Android动画合集之补...
- 8.4.3 Android动画合集之属...
- 8.4.4 Android动画合集之属...
- 9.1 使用SoundPool播放音...
- 9.2 MediaPlayer播放音频与...
- 9.3 使用Camera拍照
- 9.4 使用MediaRecord录音
- 10.1 TelephonyManager(电...
- 10.2 SmsManager(短信管理...
- 10.3 AudioManager(音频管...
- 10.4 Vibrator(振动器)
- 10.5 AlarmManager(闹钟服务)
- 10.6 PowerManager(电源服...
- 10.7 WindowManager(窗口...
- 10.8 LayoutInflater(布局服务)
- 10.9 WallpaperManager(壁...
- 10.10 传感器专题(1)——相...
- 10.11 传感器专题(2)——方...
- 10.12 传感器专题(3)——加...
- 10.12 传感器专题(4)——其...
- 10.14 Android GPS初涉
- 11.0 《2015最新Android基...

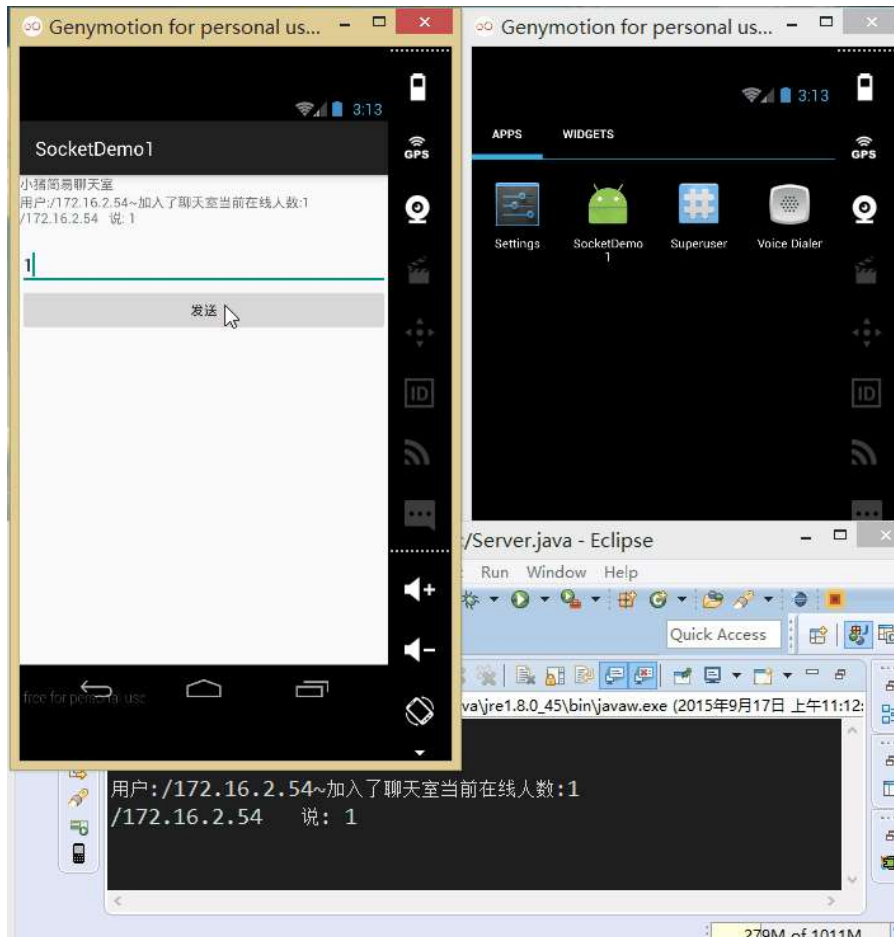
接下来我们来 搭建一个超简单的聊天室，我们需要用到线程池，存储Socket链接的集合，我们还需要 字节写一个线程，具体的我们在代码中来体会！

### 实现的效果图：

先把我们的服务端跑起来：



接着把我们的程序分别跑到两台模拟器上：



接下来我们来写代码：

首先是服务端，就是将读写socket的操作放到自定义线程当中，创建ServerSocket后，循环 调用accept方法，当有新客户端接入，将socket加入集合当中，同时在线程池新建一个线程！

另外，在读取信息的方法中，对输入字符串进行判断，如果为bye字符串，将socket从集合中 移除，然后close掉！

### Server.java：

```
public class Server {  
    //定义相关的参数,端口,存储Socket连接的集合,ServerSocket对象  
    //以及线程池  
    private static final int PORT = 12345;  
    private List<Socket> mList = new ArrayList<Socket>();  
    private ServerSocket server = null;  
    private ExecutorService myExecutorService = null;
```

```

    public static void main(String[] args) {
        new Server();
    }

    public Server()
    {
        try
        {
            server = new ServerSocket(PORT);
            //创建线程池
            myExecutorService = Executors.newCached
ThreadPool();

            System.out.println("服务端运行中...\n");
            Socket client = null;
            while(true)
            {
                client = server.accept();
                mList.add(client);
                myExecutorService.execute(new S
ervice(client));
            }

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

    class Service implements Runnable
    {
        private Socket socket;
        private BufferedReader in = null;
        private String msg = "";

        public Service(Socket socket) {
            this.socket = socket;
            try
            {
                in = new BufferedReader(new Inp
utStreamReader(socket.getInputStream()));
                msg = "用户:" +this.socket.get
InetAddress() + "~加入了聊天室"
                +"当前在线人数:" +mList.size();

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

        @Override
        public void run() {
            try{
                while(true)
                {
                    if((msg = in.readLine())

```

```

    ) != null)

        {

            if (msg.equals ("
bye"))

                {

                    System.

out.println("~~~~~");

                    mList.r

emove(socket);

                    in.close();
                    msg = "用户:" + socket.getInetAddre

ss()

                    + "退出:" + "当前在线人数:"+mL

ist.size();

                    socket.close();
                    this.sendmsg();
                    break;

                }else{

                    msg = s

ocket.getInetAddress() + " 说: " + msg;
                    this.sendmsg();

                }

            }

        }catch(Exception e){e.printStackTrace()

;}

    }

    //为连接上服务端的每个客户端发送信息
    public void sendmsg()
    {

        System.out.println(msg);
        int num = mList.size();
        for(int index = 0;index < num;index++)
        {

            Socket mSocket = mList.get(inde

x);

            PrintWriter pout = null;
            try {
                pout = new PrintWriter(new BufferedWriter(

                    new OutputStreamWriter(mSocket.getO

utputStream(),"UTF-8")),true);
                pout.println(msg);
            }catch (IOException e) {e.printStackTrace();}

        }

    }

}

```

接着到客户端，客户端的难点在于要另外开辟线程的问题，因为Android不允许直接在主线程中做网络操作，而且不允许在主线程外的线程操作UI，这里的做法是自己新建一个线程，以及通过Handler来更新UI，实际开发不建议

直接这样做！！

布局文件:activity\_main.xml :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res
/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="小猪简易聊天室" />

    <TextView
        android:id="@+id/txtshow"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />

    <EditText
        android:id="@+id/editsend"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        />

    <Button
        android:id="@+id/btnsend"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="发送"
        />

</LinearLayout>
```

MainActivity.java :

```
public class MainActivity extends AppCompatActivity implements
Runnable {

    //定义相关变量,完成初始化
    private TextView txtshow;
    private EditText editsend;
    private Button btnsend;
    private static final String HOST = "172.16.2.54";
    private static final int PORT = 12345;
    private Socket socket = null;
    private BufferedReader in = null;
    private PrintWriter out = null;
    private String content = "";
    private StringBuilder sb = null;

    //定义一个handler对象,用来刷新界面
    public Handler handler = new Handler() {
        public void handleMessage(Message msg) {
            if (msg.what == 0x123) {
                sb.append(content);
            }
        }
    };
}
```







