


[ANDROID](#)
[CORE JAVA](#)
[DESKTOP JAVA](#)
[ENTERPRISE JAVA](#)
[JAVA BASICS](#)
[JVM LANGUAGES](#)
[SOFTWARE DEVELOPMENT](#)
[DEVOPS](#)
[Home](#) » [Android](#) » [core](#) » [socket](#) » Android Socket Example

ABOUT NIKOS MARAVITSAS



Nikos has graduated from the Department of Informatics and Telecommunications of The National and Kapodistrian University of Athens. During his studies he discovered his interests about software development and he has successfully completed numerous assignments in a variety of fields. Currently, his main interests are system's security, parallel systems, artificial intelligence, operating systems, system programming, telecommunications, web applications, human – machine interaction and mobile development.



Android Socket Example

Posted by: Nikos Maravitsas in socket May 26th, 2013

In this tutorial we are going to see how to use Sockets in Android Applications. In Android, sockets work exactly as they do in Java SE. In this example we are going to see how to run an Server and a Client android Application in two different emulators. This requires some special configuration regarding port forwarding, but we are going to discuss this later on.

For this tutorial, we will use the following tools in a Windows 64-bit platform:

- JDK 1.7
- Eclipse 4.2 Juno
- Android SDK 4.2

First , we have to create two Android Application Project, one for the Server and one for the Client. I'm going to display in detail, the Project creation of the Server. Of course the same apply to the Client Project creation. Then, for the Client side I'm just going to present the necessary code.

Want to create a kick-ass Android App?

Subscribe to our newsletter and download the Android UI Design mini-book [right now!](#)

With this book, you will delve into the fundamentals of Android UI design. You will understand user input, views and layouts, as well as adapters and fragments. Furthermore, you will learn how to add multimedia to an app and also leverage themes and styles!

Email address:

[Sign up](#)

1. Create a new Android Project

Open Eclipse IDE and go to File -> New -> Project -> Android -> Android Application Project. You have to specify the Application Name, the Project Name and the Package name in the appropriate text fields and then click Next.

NEWSLETTER

153724 insiders are already weekly updates and compliance whitepapers!

Join them now to gain **ACCESS** to the latest news in as well as insights about Android Groovy and other related tech

Email address:

[Sign up](#)

JOIN US



With **1,** unique v
500 at
placed a
related s
Constan
lookout
encoura
So If you

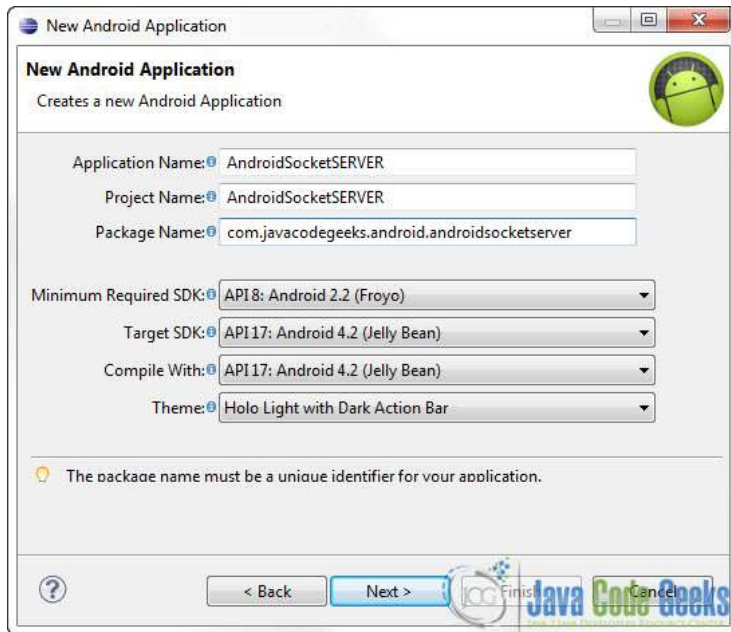
unique and interesting content then check out our **JCG** partners program be a **guest writer** for Java Code Geeks and showcase your writing skills!

CAREER OPPORTUNITIES

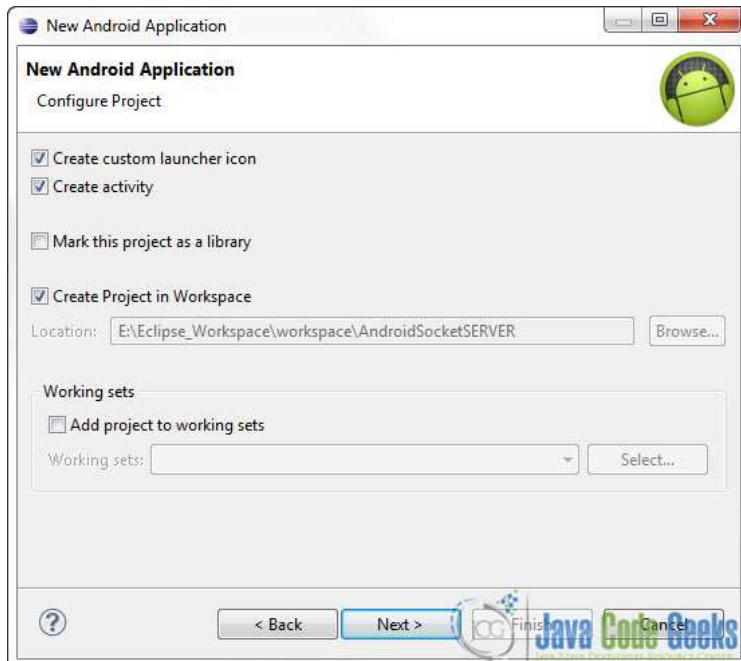
what:

where:

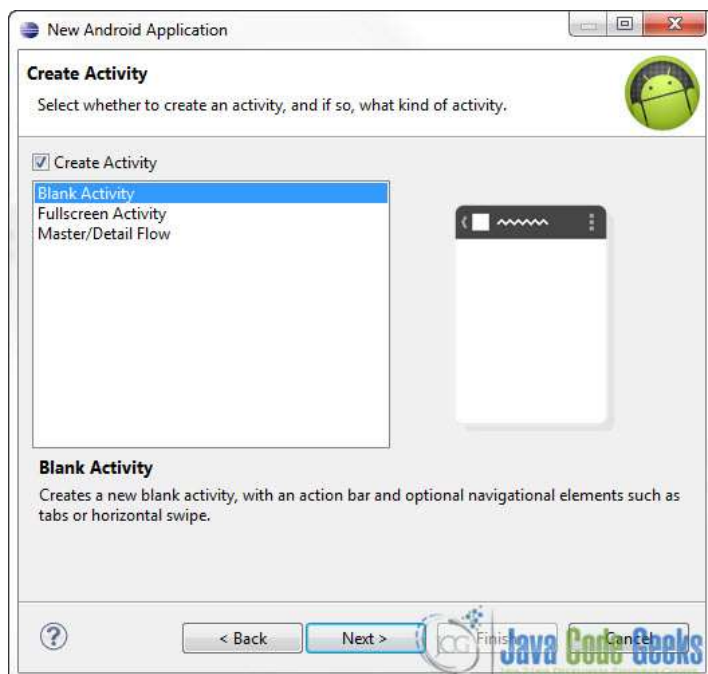
[Find Jobs](#)



In the next window make sure the "Create activity" option is selected in order to create a new activity for your project, and click Next. This is optional as you can create a new activity after creating the project, but you can do it all in one step.



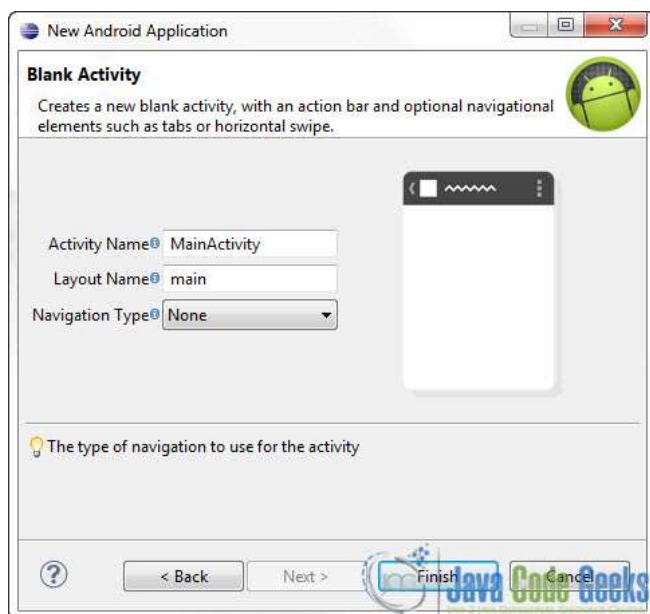
Select "BlankActivity" and click Next.



You will be asked to specify some information about the new activity. In the Layout Name text field you have to specify the name of the file that will contain the layout description of your app. In our case the file

```
res/layout/main.xml
```

will be created. Then, click Finish.

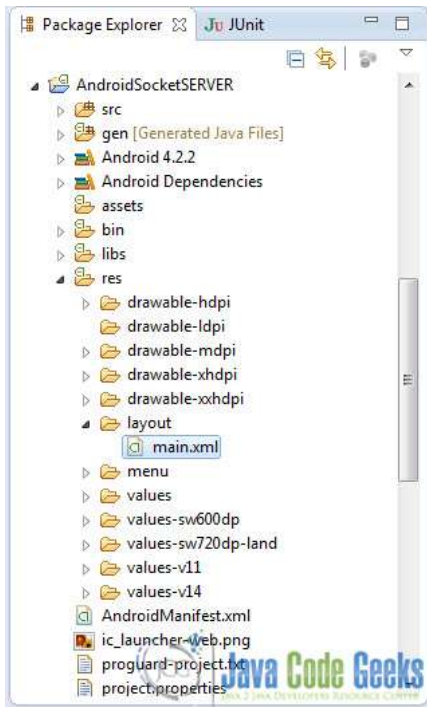


2. Create the main layout of the Server Application

Open

```
res/layout/main.xml
```

file :



And paste the following code :

main.xml:

```

01 <?xml version="1.0" encoding="utf-8"?>
02 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
03     android:layout_width="fill_parent"
04     android:layout_height="fill_parent"
05     android:orientation="vertical" >
06
07     <TextView
08         android:id="@+id/text2"
09         android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:text="" >
12     </TextView>
13
14 </LinearLayout>

```

3. Set up the Appropriate permission on AndroidManifest.xml

In order develop networking applications you have to set up the appropriate permissions in AndroidManifest.xml file :



These are the permissions:

```

1 <uses-permission android:name="android.permission.INTERNET" >
2 </uses-permission>
3
4 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" >
5 </uses-permission>

```

AndroidManifest.xml:

```

01 <?xml version="1.0" encoding="utf-8"?>
02 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
03     package="com.javacodegeeks.android.androidsocketserver"
04     android:versionCode="1"
05     android:versionName="1.0" >
06
07     <uses-sdk
08         android:minSdkVersion="8"
09         android:targetSdkVersion="17" />
10
11     <uses-permission android:name="android.permission.INTERNET" >
12     </uses-permission>
13

```

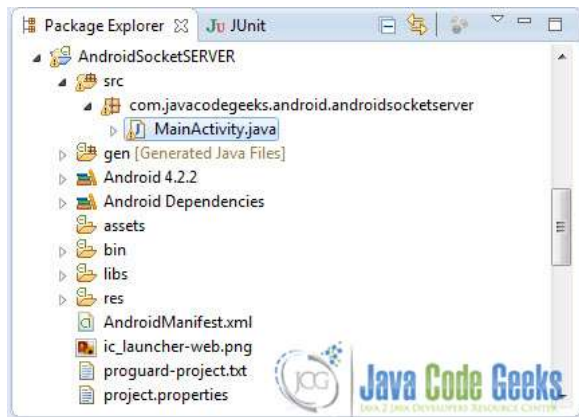
```

14 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" >
15 </uses-permission>
16
17 <application
18     android:allowBackup="true"
19     android:icon="@drawable/ic_launcher"
20     android:label="@string/app_name"
21     android:theme="@style/AppTheme" >
22     <activity
23         android:name="com.javacodegeeks.android.androidsocketserver.Server"
24         android:label="@string/app_name" >
25         <intent-filter>
26             <action android:name="android.intent.action.MAIN" />
27
28             <category android:name="android.intent.category.LAUNCHER" />
29         </intent-filter>
30     </activity>
31 </application>
32
33 </manifest>

```

4. Main Server Activity

Open the source file of the main Activity and paste the following code:



Server.java:

```

001 package com.javacodegeeks.android.androidsocketserver;
002
003 import java.io.BufferedReader;
004 import java.io.IOException;
005 import java.io.InputStreamReader;
006 import java.net.ServerSocket;
007 import java.net.Socket;
008
009 import android.app.Activity;
010 import android.os.Bundle;
011 import android.os.Handler;
012 import android.widget.TextView;
013
014 public class Server extends Activity {
015     private ServerSocket serverSocket;
016
017     Handler updateConversationHandler;
018
019     Thread serverThread = null;
020
021     private TextView text;
022
023     public static final int SERVERPORT = 6000;
024
025     @Override
026     public void onCreate(Bundle savedInstanceState) {
027         super.onCreate(savedInstanceState);
028         setContentView(R.layout.main);
029
030         text = (TextView) findViewById(R.id.text2);
031
032         updateConversationHandler = new Handler();
033
034         this.serverThread = new Thread(new ServerThread());
035         this.serverThread.start();
036
037     }
038
039     @Override
040     protected void onStop() {
041         super.onStop();
042         try {
043             serverSocket.close();
044         } catch (IOException e) {
045             e.printStackTrace();
046         }
047     }
048
049     class ServerThread implements Runnable {
050
051         public void run() {
052             Socket socket = null;
053             try {
054                 serverSocket = new ServerSocket(SERVERPORT);
055             } catch (IOException e) {
056                 e.printStackTrace();
057             }
058

```

```

059         }
060         while (!Thread.currentThread().isInterrupted()) {
061             try {
062                 socket = serverSocket.accept();
063                 CommunicationThread commThread = new CommunicationThread(socket);
064                 new Thread(commThread).start();
065             } catch (IOException e) {
066                 e.printStackTrace();
067             }
068         }
069     }
070 }
071
072 class CommunicationThread implements Runnable {
073     private Socket clientSocket;
074     private BufferedReader input;
075     public CommunicationThread(Socket clientSocket) {
076         this.clientSocket = clientSocket;
077         try {
078             this.input = new BufferedReader(new
079             InputStreamReader(this.clientSocket.getInputStream()));
080         } catch (IOException e) {
081             e.printStackTrace();
082         }
083     }
084     public void run() {
085         while (!Thread.currentThread().isInterrupted()) {
086             try {
087                 String read = input.readLine();
088                 updateConversationHandler.post(new updateUIThread(read));
089             } catch (IOException e) {
090                 e.printStackTrace();
091             }
092         }
093     }
094 }
095
096 class updateUIThread implements Runnable {
097     private String msg;
098     public updateUIThread(String str) {
099         this.msg = str;
100     }
101     @Override
102     public void run() {
103         text.setText(text.getText().toString()+"Client Says: "+ msg + "\n");
104     }
105 }
106 }
107
108 }
109
110 }
111
112 class updateUIThread implements Runnable {
113     private String msg;
114     public updateUIThread(String str) {
115         this.msg = str;
116     }
117     @Override
118     public void run() {
119         text.setText(text.getText().toString()+"Client Says: "+ msg + "\n");
120     }
121 }
122 }
123
124 }
125 }

```

5. Code for the Client project

Go ahead and create a new Android Application project, as you did with the Server Application. And paste the following code snippets in the respective files:

main.xml:

```

01 <?xml version="1.0" encoding="utf-8"?>
02 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
03     android:layout_width="fill_parent"
04     android:layout_height="fill_parent"
05     android:orientation="vertical" >
06
07     <EditText
08         android:id="@+id/EditText01"
09         android:layout_width="fill_parent"
10         android:layout_height="wrap_content"
11         android:text="JavaCodeGeeks" >
12     </EditText>
13
14     <Button
15         android:id="@+id/myButton"
16         android:layout_width="wrap_content"
17         android:layout_height="wrap_content"
18         android:onClick="onClick"
19         android:text="Send" >
20     </Button>
21
22 </LinearLayout>

```

AndroidManifest.xml:

```

01 <?xml version="1.0" encoding="utf-8"?>
02 <manifest xmlns:android="http://schemas.android.com/apk/res/android"
03     package="com.javacodegeeks.android.androidsocketclient"
04     android:versionCode="1"
05     android:versionName="1.0" >
06

```

```

07 <uses-sdk
08     android:minSdkVersion="8"
09     android:targetSdkVersion="17" />
10
11 <uses-permission android:name="android.permission.INTERNET" >
12 </uses-permission>
13
14 <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" >
15 </uses-permission>
16
17 <application
18     android:allowBackup="true"
19     android:icon="@drawable/ic_launcher"
20     android:label="@string/app_name"
21     android:theme="@style/AppTheme" >
22     <activity
23         android:name="com.javacodegeeks.android.androidsocketclient.Client"
24         android:label="@string/app_name" >
25         <intent-filter>
26             <action android:name="android.intent.action.MAIN" />
27
28             <category android:name="android.intent.category.LAUNCHER" />
29         </intent-filter>
30     </activity>
31 </application>
32
33 </manifest>

```

Client.java:

```

01 package com.javacodegeeks.android.androidsocketclient;
02
03 import java.io.BufferedWriter;
04 import java.io.IOException;
05 import java.io.OutputStreamWriter;
06 import java.io.PrintWriter;
07 import java.net.InetAddress;
08 import java.net.Socket;
09 import java.net.UnknownHostException;
10
11 import android.app.Activity;
12 import android.os.Bundle;
13 import android.view.View;
14 import android.widget.EditText;
15
16 public class Client extends Activity {
17
18     private Socket socket;
19
20     private static final int SERVERPORT = 5000;
21     private static final String SERVER_IP = "10.0.2.2";
22
23     @Override
24     public void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.main);
27
28         new Thread(new ClientThread()).start();
29     }
30
31     public void onClick(View view) {
32         try {
33             EditText et = (EditText) findViewById(R.id.EditText01);
34             String str = et.getText().toString();
35             PrintWriter out = new PrintWriter(new BufferedWriter(
36                 new OutputStreamWriter(socket.getOutputStream())),
37                 true);
38             out.println(str);
39         } catch (UnknownHostException e) {
40             e.printStackTrace();
41         } catch (IOException e) {
42             e.printStackTrace();
43         } catch (Exception e) {
44             e.printStackTrace();
45         }
46     }
47
48     class ClientThread implements Runnable {
49
50         @Override
51         public void run() {
52
53             try {
54                 InetAddress serverAddr = InetAddress.getByName(SERVER_IP);
55
56                 socket = new Socket(serverAddr, SERVERPORT);
57
58             } catch (UnknownHostException e1) {
59                 e1.printStackTrace();
60             } catch (IOException e1) {
61                 e1.printStackTrace();
62             }
63
64         }
65     }
66 }
67

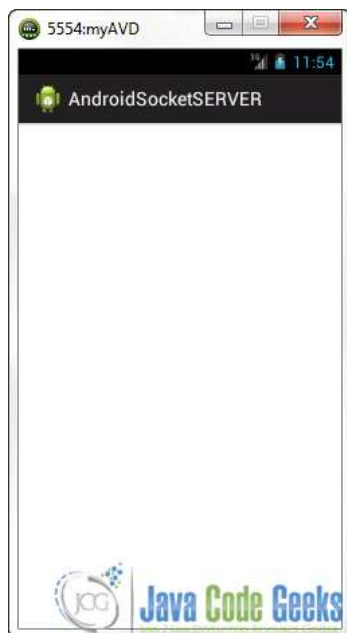
```

6. Port Forwarding

In order to interconnect the programs in the two different emulators this is what happens:

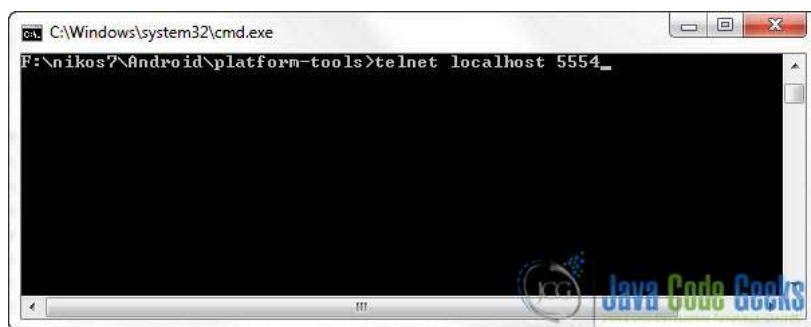
1. The Server program will open the port 6000 on emulator A. That means that port 6000 is open on the ip of the emulator which is 10.0.2.15.
2. Now, the client in emulator B will connect to the localhost, that is the development machine, which is aliased at 10.0.2.2 at port 5000.
3. The development machine (localhost) will forward the packets to 10.0.2.15 : 6000

So in order to do that we have to do some port forwarding on the emulator. To do that, run the Server Programm in order to open the first emulator:



Now, as you can see in the Window bar, we can access the console of this emulator at localhost : 5554. Press Windows Button + R, write cmd on the text box to open a command line. In order to connect to the emulator you have to do :

```
1 | telnet localhost 5554
```



To perform the port forwarding write:

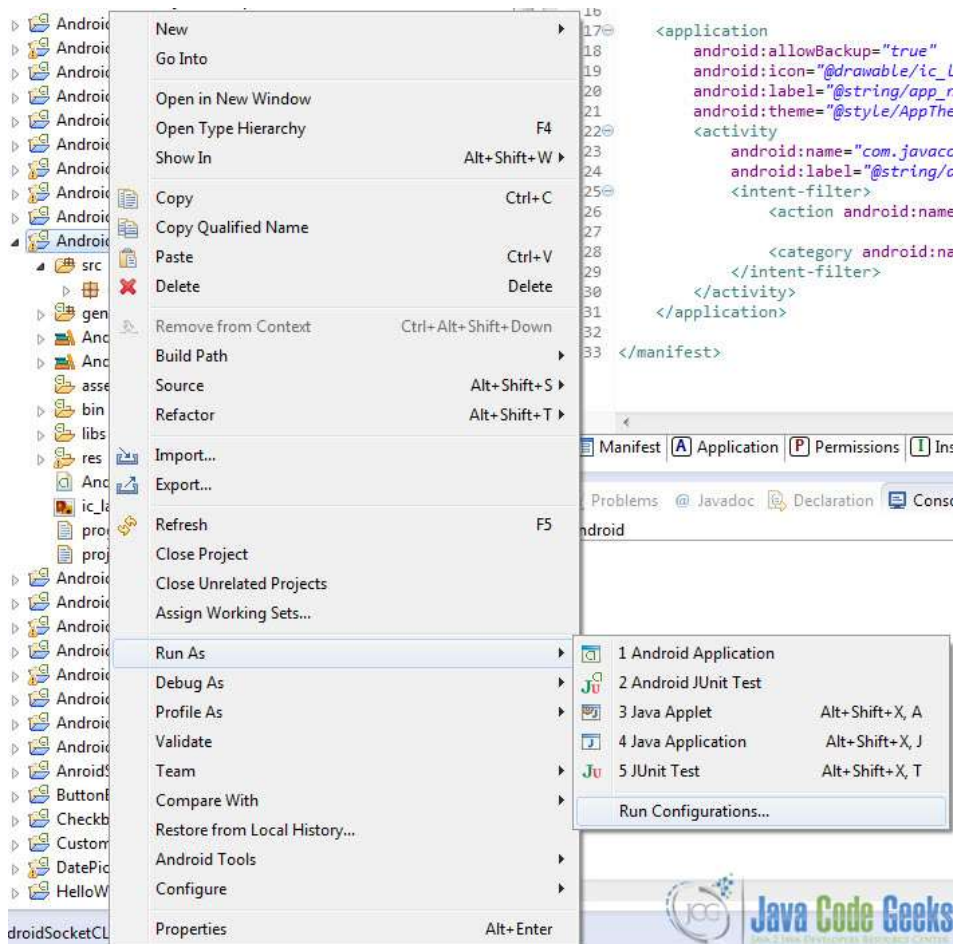
```
1 | redir add tcp:5000:6000
```



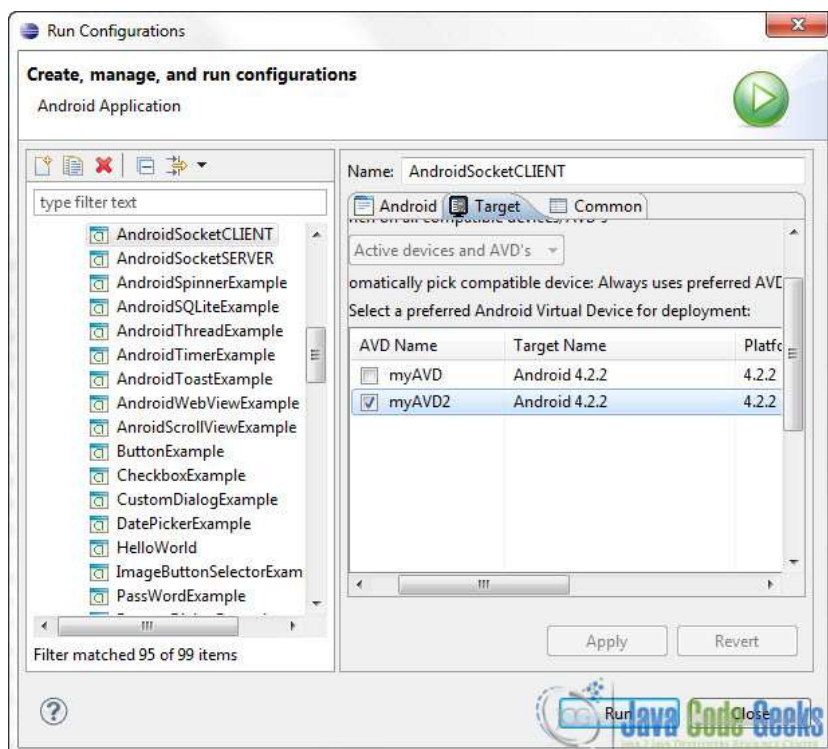
So now the packet will go through this direction : Emulator B -> development machine at 10.0.2.2 : 5000 -> Emulator A at 10.0.2.15 : 6000.

7. Run the client on another emulator.

In order to run the client on another emulator, go to the Package explorer and Right Click on the Client Project. Go to Run As -> Run Configuration:

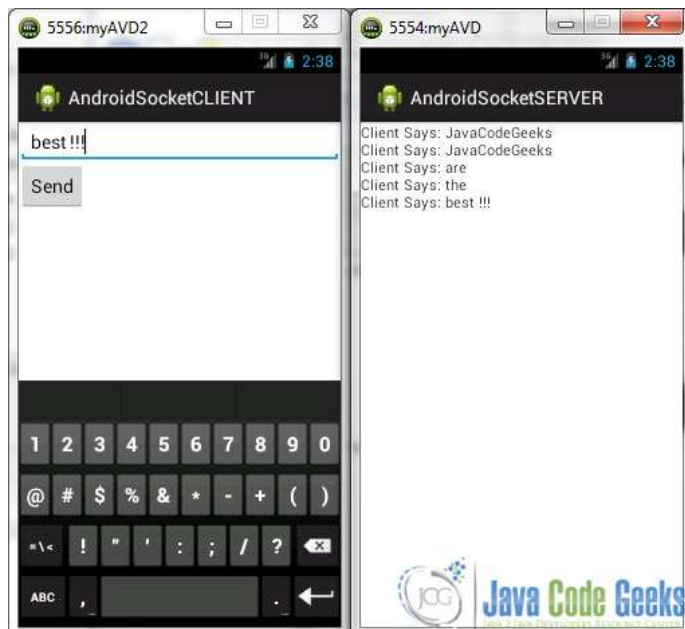


The select the Client Project for the list on the left and Click on the Target Tab. Select the second AVD and click Run:



8. Run the Application

Now that the client program is running you can send messages to the server:



Download Eclipse Project

This was an Android Socket Example. Download the Eclipse Project of this tutorial: [AndroidSocketExample.zip](#)

Do you want to know how to develop your skillset to become a **Java Rockstar**?

Subscribe to our newsletter to start Rocking right now!

To get you started we give you our best selling eBooks for **FREE!**

1. JPA Mini Book
2. JVM Troubleshooting Guide
3. JUnit Tutorial for Unit Testing
4. Java Annotations Tutorial
5. Java Interview Questions
6. Spring Interview Questions
7. Android UI Design

and many more

Email address:

Sign up

KNOWLEDGE BASE

Courses

News

Resources

Tutorials

Whitepapers

THE CODE GEEKS NETWORK

.NET Code Geeks

Java Code Geeks

System Code Geeks

Web Code Geeks

HALL OF FAME

Android Alert Dialog Example

Android OnClickListener Example

How to convert Character to String and a String to Character Array in Java

Java Inheritance example

Java write to File Example

java.io.FileNotFoundException – How to solve File Not Found Exception

java.lang.arrayindexoutofboundsexception – How to handle Array Index Out Of Bounds Exception

java.lang.NoClassDefFoundError – How to solve No Class Def Found Error

JSON Example With Jersey + Jackson

ABOUT JAVA CODE GEEKS

JCGs (Java Code Geeks) is an independent online community focused on ultimate Java to Java developers resource center; targeted at the technical team lead (senior developer), project manager and junior dev JCGs serve the Java, SOA, Agile and Telecom communities with daily new domain experts, articles, tutorials, reviews, announcements, code snippet source projects.

DISCLAIMER

All trademarks and registered trademarks appearing on Java Code Geeks are the property of their respective owners. Java is a trademark or registered trademark of Oracle Corporation in the United States and other countries. Examples J is not connected to Oracle Corporation and is not sponsored by Oracle Corporation.

