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.

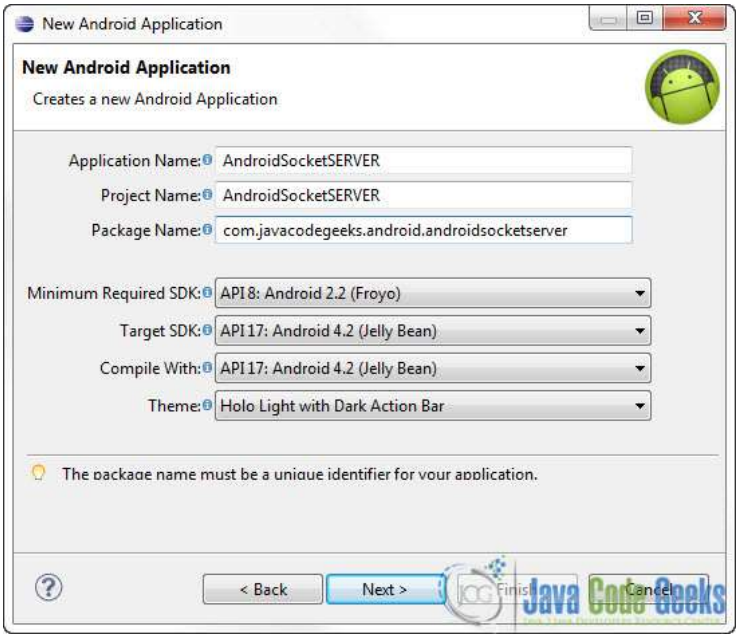
For this tutorial, we will use the following tools in a Windows 64-bit platform:

- JDK 1.7
- Eclipse 4.2 Juno
- Android SKD 4.2

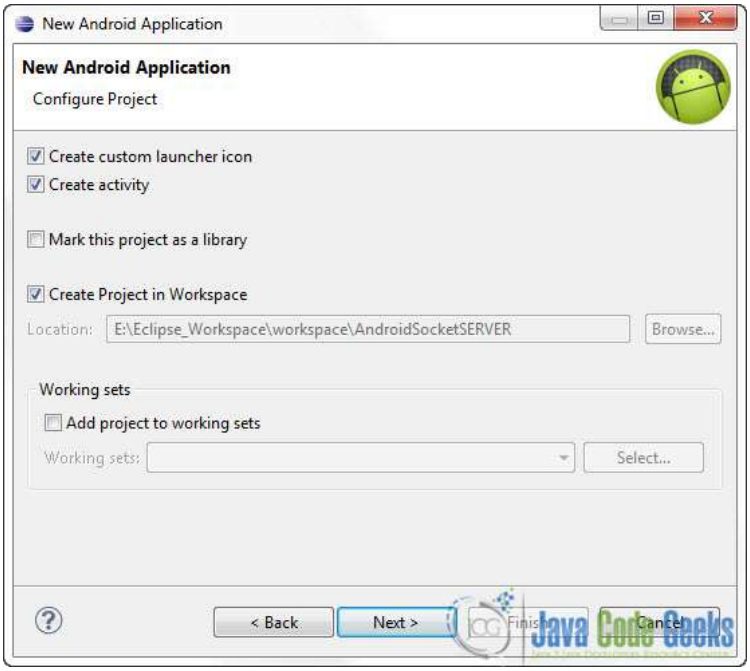
First , we have to create two Android Application Project, one for the Server and one for the Client. I'm going to display in detail, the Project creation of the Server. Of course the same apply to the Client Project creation. Then, for the Client side I'm just going to present the necessary code.

1. Create a new Android Project

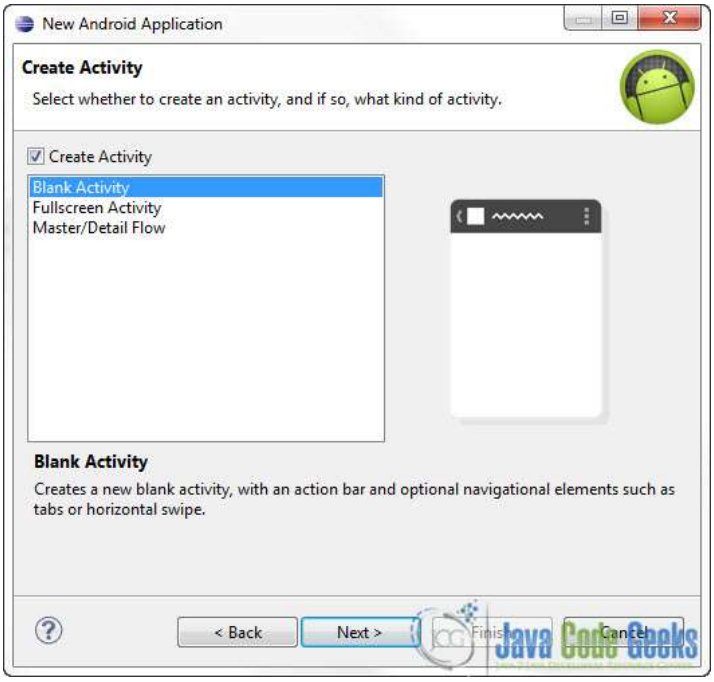
Open Eclipse IDE and go to File -> New -> Project -> Android -> Android Application Project. You have to specify the Application Name, the Project Name and the Package name in the appropriate text fields and then click Next.



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.



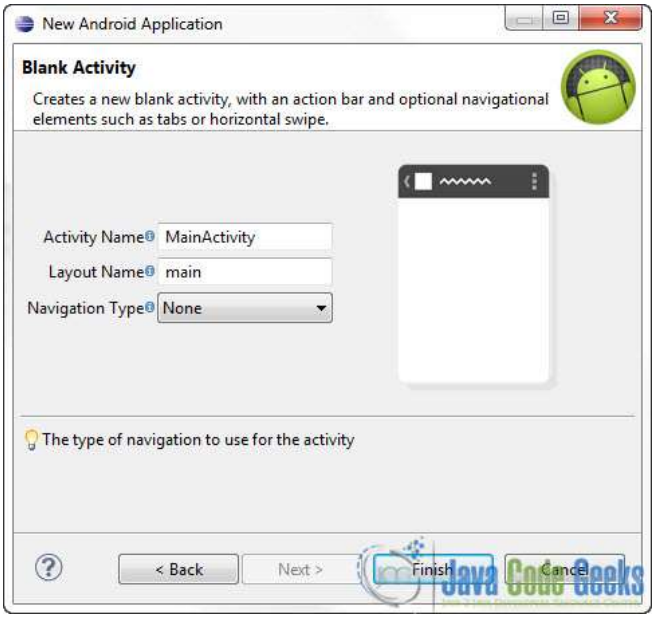
Select "BlankActivity" and click Next.



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.

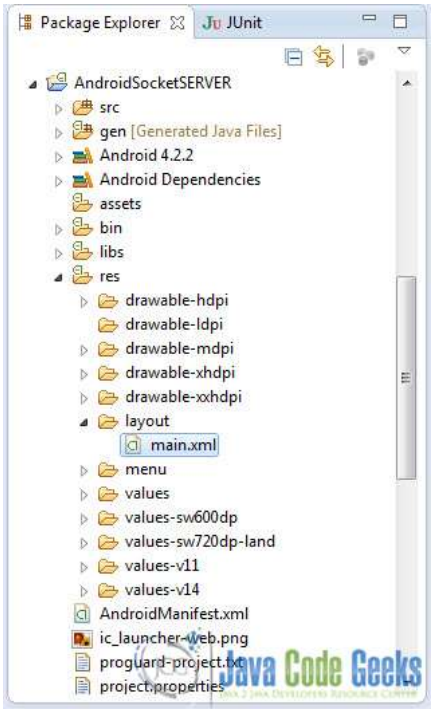


2. Create the main layout of the Server Application

Open

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

file :



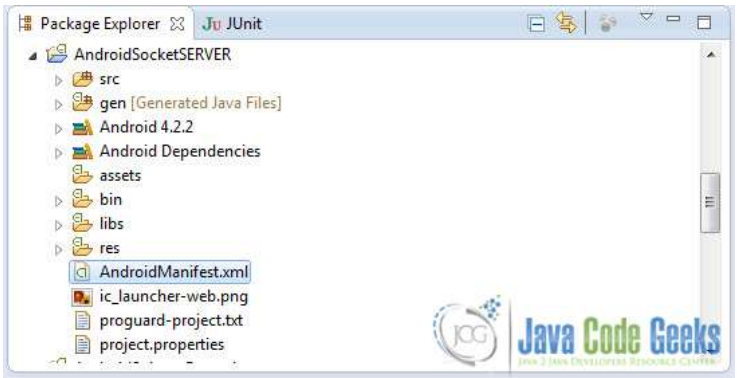
And paste the following code :

main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <TextView
        android:id="@+id/text2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="" >
    </TextView>

</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:

```
<uses-permission android:name="android.permission.INTERNET" >
</uses-permission>

<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" >
</uses-permission>
```

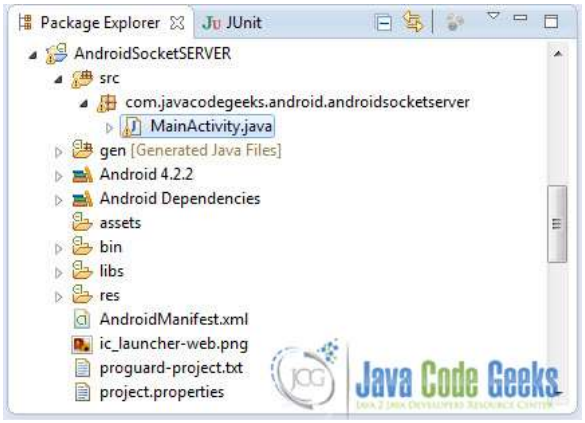
AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.javacodegeeks.android.androidsocketserver"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <uses-permission android:name="android.permission.INTERNET" >
    </uses-permission>

```

```
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" >
    </uses-permission>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.javacodegeeks.android.androidsocketserver.Server"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</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
016     private ServerSocket serverSocket;
017
018     Handler updateConversationHandler;
019
020     Thread serverThread = null;
021
022     private TextView text;
023
024     public static final int SERVERPORT = 6000;
025
026     @Override
027     public void onCreate(Bundle savedInstanceState) {
028
029         super.onCreate(savedInstanceState);
030         setContentView(R.layout.main);
031
032         text = (TextView) findViewById(R.id.text2);
033
034         updateConversationHandler = new Handler();
035
036         this.serverThread = new Thread(new ServerThread());
037         this.serverThread.start();
038
039     }
040
041     @Override
042     protected void onStop() {
043         super.onStop();
044         try {
045             serverSocket.close();
046         } catch (IOException e) {
047             e.printStackTrace();
048         }
049     }
050
051     class ServerThread implements Runnable {
052
053         public void run() {
054             Socket socket = null;
055             try {
056                 serverSocket = new ServerSocket(SERVERPORT);
057             } catch (IOException e) {
058                 e.printStackTrace();
```

```
            }
            while (!Thread.currentThread().isInterrupted()) {

                try {

                    socket = serverSocket.accept();

                    CommunicationThread commThread = new CommunicationThread(socket);
                    new Thread(commThread).start();

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

    class CommunicationThread implements Runnable {

        private Socket clientSocket;

        private BufferedReader input;

        public CommunicationThread(Socket clientSocket) {

            this.clientSocket = clientSocket;

            try {

                this.input = new BufferedReader(new
InputStreamReader(this.clientSocket.getInputStream()));

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

        public void run() {

            while (!Thread.currentThread().isInterrupted()) {

                try {

                    String read = input.readLine();

                    updateConversationHandler.post(new updateUIThread(read));

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

    }

    class updateUIThread implements Runnable {
        private String msg;

        public updateUIThread(String str) {
            this.msg = str;
        }

        @Override
        public void run() {
            text.setText(text.getText().toString()+"Client Says: "+ msg + "\n");
        }
    }
}
```

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:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" >

    <EditText
        android:id="@+id/EditText01"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="JavaCodeGeeks" >
    </EditText>

    <Button
        android:id="@+id/myButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="onClick"
        android:text="Send" >
    </Button>

</LinearLayout>
```

AndroidManifest.xml:

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

```

```
    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <uses-permission android:name="android.permission.INTERNET" >
    </uses-permission>

    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" >
    </uses-permission>

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="com.javacodegeeks.android.androidsocketclient.Client"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Client.java:

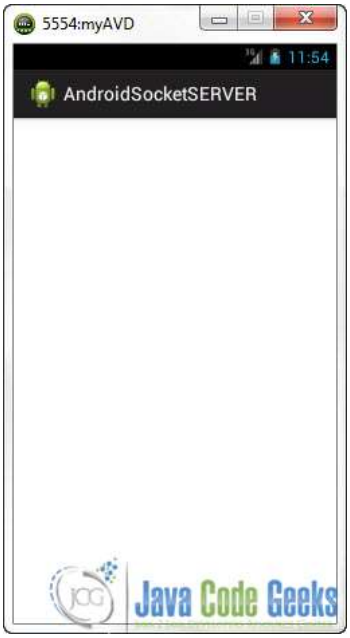
```
package com.javacodegeeks.android.androidsocketclient;

import java.io.BufferedWriter;
import java.io.IOException;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.Socket;
import java.net.UnknownHostException;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.EditText;

public class Client extends Activity {

    private Socket socket;

    private static final int SERVERPORT = 5000;
    private static final String SERVER_IP = "10.0.2.2";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        new Thread(new ClientThread()).start();
    }

    public void onClick(View view) {
        try {
            EditText et = (EditText) findViewById(R.id.EditText01);
            String str = et.getText().toString();
            PrintWriter out = new PrintWriter(new BufferedWriter(
                    new OutputStreamWriter(socket.getOutputStream())),
                    true);
            out.println(str);
        } catch (UnknownHostException e) {
            e.printStackTrace();
        } catch (IOException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    class ClientThread implements Runnable {

        @Override
        public void run() {

            try {
                InetAddress serverAddr = InetAddress.getByName(SERVER_IP);

                socket = new Socket(serverAddr, SERVERPORT);

            } catch (UnknownHostException e1) {
                e1.printStackTrace();
            } catch (IOException e1) {
                e1.printStackTrace();
            }

        }

    }
}
```

6. Port Forwarding

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

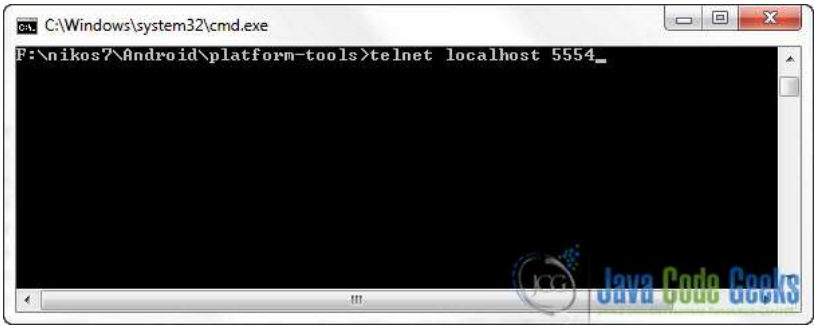
1. The Server program will open the port 6000 on emulator A. That means that porst 6000 is open on the ip of the emulator which is 10.0.2.15.
2. Now, the client in emulator B will connect to the locahost, that is the development machine, which is aliased at 10.0.2.2 at port 5000.
3. The development machine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

So in order to do that we have to do some port forwatding 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 cosnole of this emulator at localhost : 5554. Press Windows Button + R, write cmd on the text box to open a comman line. In order to connect to the emulator you have to do :

```
1 telnet localhost 5554
```



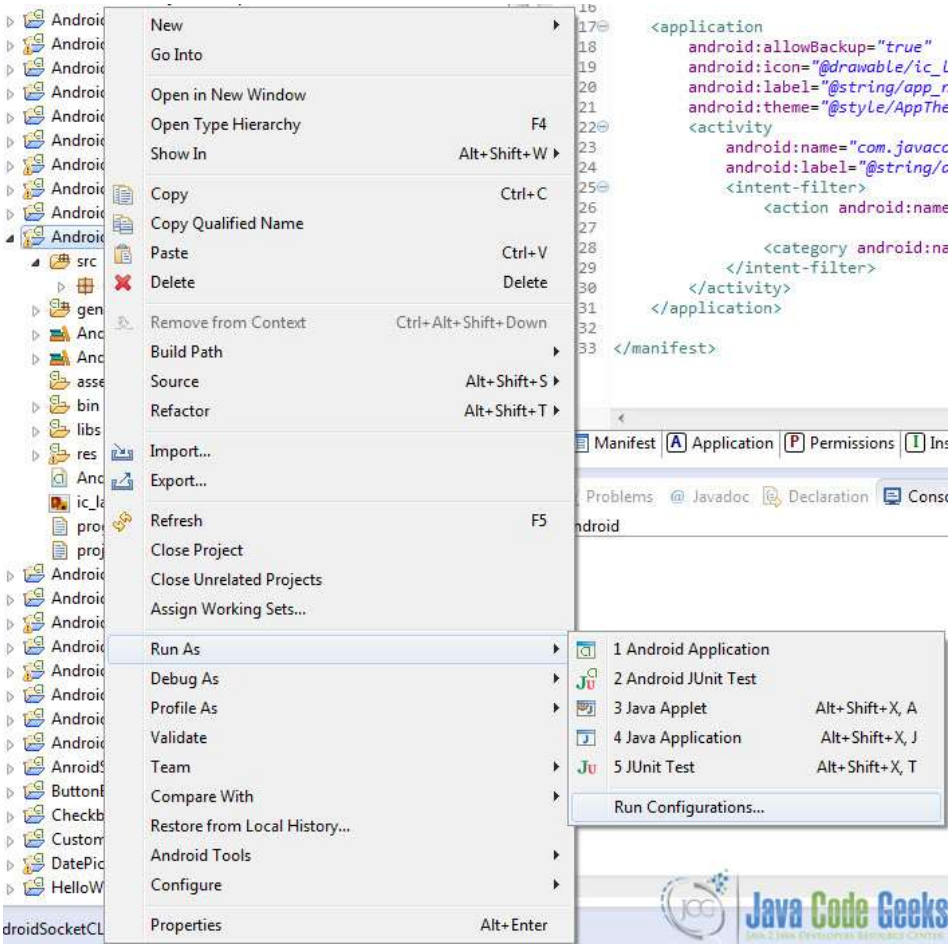
To perform the port forwarding write:

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

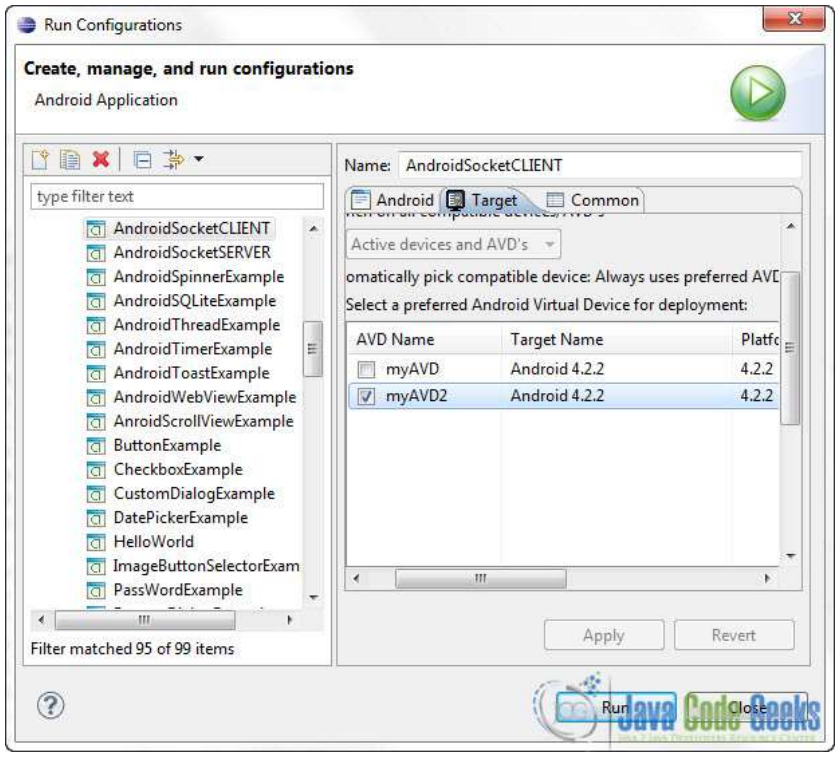
So now the packet will go through this direction : Emulator B -> development machine at 10.0.2.2 : 5000 -> Emulator A at 10.0.2.15 : 6000.

7. Run the client on another emulator.

In oder 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:

Your email address

Sign up

Spring JdbcTemplate Example