50. Android Remote Bound Services – A Worked Example

In this, the final chapter dedicated to Android services, an example application will be developed to demonstrate the use of a messenger and handler configuration to facilitate interaction between a client and remote bound service.

50.1 Client to Remote Service Communication

As outlined in the previous chapter, interaction between a client and a local service can be implemented by returning to the client an IBinder object containing a reference to the service object. In the case of remote services, however, this approach does not work because the remote service is running in a different process and, as such, cannot be reached directly from the client.

In the case of remote services, a Messenger and Handler configuration must be created which allows messages to be passed across process boundaries between client and service.

Specifically, the service creates a Handler instance that will be called when a message is received from the client. In terms of initialization, it is the job of the Handler to create a Messenger object which, in turn, creates an IBinder object to be returned to the client in the *onBind()* method. This IBinder object is used by the client to create an instance of the Messenger object and, subsequently, to send messages to the service handler. Each time a message is sent by the client, the *handleMessage()* method of the handler is called, passing through the message object.

The simple example created in this chapter will consist of an activity and a bound service running in separate processes. The Messenger/Handler mechanism will be used to send a string to the service, which will then display that string in a Toast message.

50.2 Creating the Example Application

Launch Android Studio and follow the steps to create a new project, entering *RemoteBound* into the Application name field and *ebookfrenzy.com* as the Company Domain setting before clicking on the *Next* button.

On the form factors screen, enable the *Phone and Tablet* option and set the minimum SDK setting to API 14: Android 4.0 (IceCreamSandwich). Continue to proceed through the screens, requesting the creation of an Empty Activity named *RemoteBoundActivity* with a corresponding layout resource file named *activity_remote_bound.*

50.3 Designing the User Interface

Locate the *activity_remote_bound.xml* file in the Project tool window and double-click on it to load it into the Layout Editor tool. With the Layout Editor tool in Design mode, delete the default TextView instance and drag and drop a Button widget from the palette so that it is positioned in the center of the layout. Change the text property of the button to read "Send Message" and extract the string to a new resource named *send_message*.

Finally, configure the *onClick* property to call a method named *sendMessage*.

50.4 Implementing the Remote Bound Service

In order to implement the remote bound service for this example, add a new class to the project by right-clicking on the package name (located under *app -> java*) within the Project tool window and select the *New -> Service -> Service* menu option. Specify *RemoteService* as the class name and make sure that both the *Exported* and *Enabled* options are selected before clicking on *Finish* to create the class.

The next step is to implement the handler class for the new service. This is achieved by extending the Handler class and implementing the *handleMessage()* method. This method will be called when a message is received from the client. It will be passed a Message object as an argument containing any data that the client needs to pass to the service. In this instance, this will be a Bundle object containing a string to be displayed to the user. The modified class in the *RemoteService.java* file should read as follows once this has been implemented:

```
package com.ebookfrenzy.remotebound;

import android.app.Service;
import android.content.Intent;
import android.os.IBinder;
import android.os.Bundle;
import android.os.Handler;
```

```
import android.os.Message;
import android.widget.Toast;
import android.os.Messenger;

public class RemoteService extends Service {

    public RemoteService() {
    }

    class IncomingHandler extends Handler {
        @Override
        public void handleMessage(Message msg) {

            Bundle data = msg.getData();
            String dataString = data.getString("MyString");
            Toast.makeText(getApplicationContext(),
                    dataString, Toast.LENGTH_SHORT).show();
        }
    }

    @Override
    public IBinder onBind(Intent intent) {
        // TODO: Return the communication channel to the service.
        throw new UnsupportedOperationException("Not yet
implemented");
    }
}
```

With the handler implemented, the only remaining task in terms of the service code is to modify the *onBind()* method such that it returns an IBinder object containing a Messenger object which, in turn, contains a reference to the handler:

```
final Messenger myMessenger = new Messenger(new IncomingHandler());

@Override
public IBinder onBind(Intent intent) {
      return myMessenger.getBinder();
}
```

The first line of the above code fragment creates a new instance of our handler class and passes it through to the constructor of a new Messenger object. Within the *onBind()* method, the *getBinder()* method of the messenger object is called to return the messenger's IBinder object.

50.5 Configuring a Remote Service in the Manifest File

In order to portray the communication between a client and remote service accurately, it will be necessary to configure the service to run in a separate process from the rest of the application. This is achieved by adding an *android:process* property within the <service> tag for the service in the manifest file. In order to launch a remote service it is also necessary to provide an intent filter for the service. To implement these changes, modify the *AndroidManifest.xml* file to add the required entries:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ebookfrenzy.remotebound" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".RemoteBoundActivity" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

        <service
            android:name=".RemoteService"
            android:enabled="true"
            android:exported="true"
            android:process=":my_process" >
        </service>
    </service>
    </application>

</manifest>
```

50.6 Launching and Binding to the Remote Service

As with a local bound service, the client component needs to implement an

instance of the ServiceConnection class with *onServiceConnected()* and *onServiceDisconnected()* methods. Also, in common with local services, the *onServiceConnected()* method will be passed the IBinder object returned by the *onBind()* method of the remote service which will be used to send messages to the server handler. In the case of this example, the client is *RemoteBoundActivity*, the code for which is located in *RemoteBoundActivity.java*. Load this file and modify it to add the ServiceConnection class and a variable to store a reference to the received Messenger object together with a Boolean flag to indicate whether or not the connection is established:

```
package com.ebookfrenzy.remotebound;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.view.View;

public class RemoteBoundActivity extends AppCompatActivity {

    Messenger myService = null;
    boolean isBound;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_remote_bound);
    }

    private ServiceConnection myConnection =
            new ServiceConnection() {
                public void onServiceConnected(
                            ComponentName className,
                                        IBinder service) {
                    myService = new Messenger(service);
```

```
                isBound = true;
            }

            public void onServiceDisconnected(
                        ComponentName className) {
                myService = null;
                isBound = false;
            }
        };
}
```

Next, some code needs to be added to bind to the remote service. This involves creating an intent that matches the intent filter for the service as declared in the manifest file and then making a call to the *bindService()* method, providing the intent and a reference to the ServiceConnection instance as arguments. For the purposes of this example, this code will be implemented in the activity's *onCreate()* method:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
       super.onCreate(savedInstanceState);
       setContentView(R.layout.activity_remote_bound);

       Intent intent = new Intent(getApplicationContext(),
                           RemoteService.class);

       bindService(intent, myConnection, Context.BIND_AUTO_CREATE);
}
```

50.7 Sending a Message to the Remote Service

All that remains before testing the application is to implement the *sendMessage()* method in the *RemoteBoundActivity* class which is configured to be called when the button in the user interface is touched by the user. This method needs to check that the service is connected, create a bundle object containing the string to be displayed by the server, add it to a Message object and send it to the server:

```
public void sendMessage(View view)
{
        if (!isBound) return;

        Message msg = Message.obtain();
```

```
        Bundle bundle = new Bundle();
        bundle.putString("MyString", "Message Received");

        msg.setData(bundle);

        try {
            myService.send(msg);
        } catch (RemoteException e) {
            e.printStackTrace();
        }
}
```

With the code changes complete, compile and run the application. Once loaded, touch the button in the user interface, at which point a Toast message should appear that reads “Message Received”.

50.8 Summary

In order to implement interaction between a client and remote bound service it is necessary to implement a handler/message communication framework. The basic concepts behind this technique have been covered in this chapter together with the implementation of an example application designed to demonstrate communication between a client and a bound service, each running in a separate process.