SMS Applications with Android

SMS is one of the most used functions on mobile phones. It is great for short person to person communications. It can also be used in a variety of ways within applications beyond it's normal use. For instance, it can be used to exchange small amounts of data between phones running the same application.

We'll start with normal person to person SMS messaging and move towards developing applications that utilize SMS for other purposes.

Sending SMS with an Intent

As with most built-in functionality in Android, an Intent can be used to send an SMS message utilizing the normal SMS Activity:

```
        Intent smsIntent = new Intent(Intent.ACTION_VIEW);
        smsIntent.setType("vnd.android-dir/mms-sms");
        smsIntent.putExtra("address", "12125551212");
        smsIntent.putExtra("sms_body","Body of Message");
        startActivity(smsIntent);
```

The Intent specified above is a generic "ACTION_VIEW". Android knows based upon the specified "type" that it is meant for the built-in SMS Activity. The "address" and "sms_body" extras specify the phone number and the body of the message that will be filled in when the Activity launches.

The following example Activity uses the above code launch the built-in SMS Activity when a Button is pressed. (This example requires that you have a Button with the ID of "Button01" included in the "main.xml" layout file as part of the project.)

```
package com.mobvcasting.sendsmsintent;

import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;

public class SendSMS extends Activity implements OnClickListener {

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

        Button sendButton = (Button) this.findViewById(R.id.Button01);
        sendButton.setOnClickListener(this);
    }

     public void onClick(View v) {
          Intent smsIntent = new Intent(Intent.ACTION_VIEW);
          smsIntent.setType("vnd.android-dir/mms-sms");
          smsIntent.putExtra("address", "12125551212");
          smsIntent.putExtra("sms_body","Body of Message");
          startActivity(smsIntent);
     }
}
```

Programmatically sending an SMS

We don't have to rely upon the built-in applications to handle SMS. We can utilize the Android API to programmatically send SMS messages from our own applications.

First, in order to send SMS, we need to add the following permission to our AndroidManifest.xml file:

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

Here is a basic snippet of code for sending an SMS:

```
     SmsManager sms = SmsManager.getDefault();
     sms.sendTextMessage("1-212-555-1212", null, "Hi there", null, null);
```

The first argument in the "sendTextMessage" method is the phone number to send to and the third argument is the body of the message. We'll explore the other arguments which are set to null later.

When this code is run, it will just send an SMS message. No questions asked.

One problem with the above is that users may not like applications which just send out SMS messages without a prompt or at

least the knowledge that it has happened. To indicate to a user that a message has been sent, we can pass in something called a "PendingIntent". This is an Intent that the receiving Activity will use to call our Activity back with once the message has been sent. It is passed as the fourth argument in the "sendTextMessage" method of the SmsManager object.

To receive the PendingIntent message we need to create a "BroadcastReceiver" and register it.

Here is a full example:

```
package com.mobvcasting.smsfun;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class SMSFun extends Activity {

     // Button to trigger sending the SMS
     Button aButton;
     // PendingIntent to tell the SMS app to notify us
     PendingIntent sentPI;
     // The intent action we are using
    String SENT = "SMS_SENT";
    // The BroadcastReceiver that we use to listen for the notification back
    BroadcastReceiver br;

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

        // Create the Pending Intent
        sentPI = PendingIntent.getBroadcast(this, 0,
                new Intent(SENT), 0);

        aButton = (Button) this.findViewById(R.id.Button01);
        aButton.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {

                     SmsManager sms = SmsManager.getDefault();
                     // send the message, passing in the pending intent, sentPI
                   sms.sendTextMessage("1-212-555-1212", null, "Hi there", sentPI, null);

                   registerReceiver(br, new IntentFilter(SENT));

                }});

        // In order to receive the results via the pending intent we need
        // to register a new BroadcastReceiver and pay attention to the various
        // values we could get back
        br = new BroadcastReceiver(){
                @Override
                public void onReceive(Context ctx, Intent intent) {
                    switch (getResultCode())
                    {
                        case Activity.RESULT_OK:
                            Toast.makeText(getBaseContext(), "SMS sent",
                                    Toast.LENGTH_SHORT).show();
                            break;
                        case SmsManager.RESULT_ERROR_GENERIC_FAILURE:
                            Toast.makeText(getBaseContext(), "SMS: Generic failure",
                                    Toast.LENGTH_SHORT).show();
                            break;
                        case SmsManager.RESULT_ERROR_NO_SERVICE:
                            Toast.makeText(getBaseContext(), "SMS: No service",
                                    Toast.LENGTH_SHORT).show();
                            break;
                        case SmsManager.RESULT_ERROR_NULL_PDU:
                            Toast.makeText(getBaseContext(), "SMS: Null PDU",
                                    Toast.LENGTH_SHORT).show();
                            break;
                        case SmsManager.RESULT_ERROR_RADIO_OFF:
                            Toast.makeText(getBaseContext(), "SMS: Radio off",
                                    Toast.LENGTH_SHORT).show();
                            break;
                    }

                    unregisterReceiver(br);
                }

            };
```

```
    }
}
```

Receiving an SMS

We can also make an application which receives SMS messages. In order to do so, we need to add the following permission to our AndroidManifest.xml file:

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

We also need to tell Android that we want to handle incoming SMS messages. To do this, we need to have an "Intent Filter" which tells Android to launch a specific class when an SMS comes in. This is done in the AndroidManifest.xml file as well:

```
          <receiver android:name=".SMSFunReceive">
            <intent-filter>
                <action android:name=
                   "android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
          </receiver>
```

Here is the full XML for the AndroidManifest.xml file in an application which both sends and receives SMS messages:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.mobvcasting.smsfun"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".SMSFun"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

          <receiver android:name=".SMSFunReceive">
            <intent-filter>
                <action android:name=
                   "android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
          </receiver>
     </application>
    <uses-sdk android:minSdkVersion="4" />
     <uses-permission android:name="android.permission.SEND_SMS"></uses-permission>
     <uses-permission android:name="android.permission.RECEIVE_SMS"></uses-permission>
</manifest>
```

This XML specifies that Android should hand SMS messages to a class called SMSFunReceive.

SMSFunReceive itself isn't an Activity, rather it is a BroadcastReceiver.

```
package com.mobvcasting.smsfun;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsMessage;
import android.util.Log;

// This will run when an SMS message comes in.
// We can see if we want to do something based upon the message
// Perhaps launch an activity

public class SMSFunReceive extends BroadcastReceiver
{
     @Override
     public void onReceive(Context ctx, Intent intent)
     {
          Bundle bundle = intent.getExtras();
          Object[] pdus = (Object[]) bundle.get("pdus");
          SmsMessage[] messages = new SmsMessage[pdus.length];

          for (int i = 0; i < messages.length; i++)
          {
               messages[i] = SmsMessage.createFromPdu((byte[])pdus[i]);

               Log.v("SMSFun","Body: " + messages[i].getDisplayMessageBody());
               Log.v("SMSFun","Address: " + messages[i].getDisplayOriginatingAddress());

               //If say we wanted to do something based on who sent it
               if (messages[i].getDisplayOriginatingAddress().contains("2125551212"))
               {
                    // we could launch an activity and pass the data
                    Intent newintent = new Intent(ctx, SecretMessage.class);
                    newintent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

```
                    // Pass in data
                    newintent.putExtra("address", messages[i].getDisplayOriginatingAddress());
                    newintent.putExtra("message", messages[i].getDisplayMessageBody());
                    ctx.startActivity(newintent);
                }
            }
        }
}
```

This code launches an Activity called "SecretMessage" if the messages is from the phone number 212-555-1212. Alternatively, it could check the contents of the message for keywords or a phrase and launch an Activity based upon that.

The SecretMessage Activity needs to be specified in the AndroidManifest.xml file:

```
        <activity android:name=".SecretMessage"></activity>
```

Here is it's code:

```
package com.mobvcasting.smsfun;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;


public class SecretMessage extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.secret);

        // Get the extra data
        Bundle extras = getIntent().getExtras();
        String address = extras.getString("address");
        String message = extras.getString("message");

        TextView addresstv = (TextView) findViewById(R.id.addresstv);
        TextView messagetv = (TextView) findViewById(R.id.messagetv);

        messagetv.setText(message);
        addresstv.setText(address);
    }
}
```

You see that it has two TextView components which get populated with data that is passed in when the activity is launched. These two TextView component need to be created and created with the correct IDs (addresstv and messagetv). In this example they are created in a layout XML file called secret.xml.

To create that XML file in Eclipse, choose File: New: Android XML File. Make sure it is saved in the correct project, and give it the file name "secret.xml". Make sure it is a "Layout" resource and it is put into the "/res/layout" folder.

Here is the full contents of the secret.xml file referenced above that includes the two TextView components:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content">
<TextView android:layout_height="wrap_content" android:id="@+id/addresstv" android:text="address" android:layout_width="fill_parent"
android:layout_gravity="fill_horizontal"></TextView>
<TextView android:layout_height="wrap_content" android:id="@+id/messagetv" android:text="message" android:layout_width="fill_parent"
android:layout_gravity="fill_horizontal"></TextView>
</LinearLayout>
```