Android Messaging

Babylon University, IT College, SW Dep., Android Assist. Lecturer: Wadhah R. Baiee (2014)

Once your basic Android application is up and running, the next interesting thing you can add to it is the capability to communicate with the outside world.

You may want your application to send an SMS message to another phone when an event happens.

You learn how to send and receive SMS messages programmatically from within your Android application.
Any mobile phone you buy today should have at least SMS messaging capabilities, and nearly all users of any age know how to send and receive such messages.

Android comes with a built-in SMS application that enables you to send and receive SMS messages.

However, in some cases you might want to integrate SMS capabilities into your own Android application.
SMS Messaging

- For example, this would be useful if you wanted to track the location of your kids — simply give them an Android device that sends out an SMS message containing its geographical location every 30 minutes.

- Now you know if they really went to the library after school! (Of course, such a capability also means you would have to pay the fees incurred from sending all those SMS messages...)
Sending SMS Programmatically

- Using this approach, your application can automatically send an SMS message to a recipient without user intervention.

```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" >

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

</LinearLayout>
```
Sending SMS Programmatically

- In the AndroidManifest.xml file, add the following statements in bold:

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

- Because sending SMS messages incurs additional costs on the user’s end, indicating the SMS permissions in the AndroidManifest.xml file enables users to decide whether to allow the application to install or not.

- Add the following statements in bold to the SMSActivity.java file:
package net.learn2develop.SMS;

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

import android.telephony.SmsManager;
import android.view.View;

public class SMSActivity extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }

    public void onClick(View v) {
        sendSMS("5556", "Hello my friends!");
    }

    //---sends an SMS message to another device---
    private void sendSMS(String phoneNumber, String message)
    {
        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message, null, null)
    }
}
Sending SMS Programmatically

- On the first Android emulator (5554), click the Send SMS button to send an SMS message to the second emulator (5556).
- Figure 8-1 shows the SMS message received by the second emulator (note the notification bar at the top).
Sending SMS Programmatically

- To send an SMS message programmatically, you use the SmsManager class.
- Unlike other classes, you do not directly instantiate this class; instead, you call the `getDefault()` static method to obtain an SmsManager object.
- You then send the SMS message using the `sendTextMessage()` method:
Following are the five arguments to the `sendTextMessage()` method:

- `destinationAddress` — Phone number of the recipient
- `scAddress` — Service center address; use null for default SMSC
- `text` — Content of the SMS message
- `sentIntent` — Pending intent to invoke when the message is sent
- `deliveryIntent` — Pending intent to invoke when the message has been delivered

**NOTE** If you send an SMS message programmatically using the `SmsManager` class, the message sent will not appear in the built-in Messaging application of the sender.
Besides sending SMS messages from your Android applications, you can also receive incoming SMS messages from within your applications by using a `BroadcastReceiver` object.

This is useful when you want your application to perform an action when a certain SMS message is received.
Receiving SMS Programmatically

- For example, you might want to track the location of your phone in case it is lost or stolen. In this case, you can write an application that automatically listens for SMS messages containing some secret code.

- Once that message is received, you can then send an SMS message containing the location’s coordinates back to the sender.
SMS (Short Message Service) is specified by the ETSI (standards GSM 03.401 and 03.382).

- It can contain up to 160 characters, when each character is written according the 7-bits GSM default alphabet.

Next to a message the SMS contain also some meta-data, for example:
- Info about the senders (Service center number, sender number)
- Protocol information (Protocol identifier, Data coding scheme)
- Time stamp
SMS PDU format

- There are two ways to receive and send SMS messages: PDU (Protocol Description Unit) and Text mode. In this document, we focus on PDU mode.
- PDU format can be used on any encoding.
- To explain you the SMS PDU encoding, we use an example:

  07911326040000F0040B911346610089F60000208062917314080CC8F71D14969741F977FD07
The above PDU string contains the message "How are you?" and was read from a Siemens C35i mobile phone.

The string is build from hexadecimal-octets and semi decimal-octets.

As mentioned before, the SMS contains some meta-data about him self.

We explain it using the example above:
## SMS PDU format

<table>
<thead>
<tr>
<th>Octet(s)</th>
<th>Description</th>
<th>Format</th>
<th>In this example</th>
</tr>
</thead>
<tbody>
<tr>
<td>07</td>
<td>Length of the SMSC information</td>
<td>hex-octet</td>
<td>7 octets</td>
</tr>
<tr>
<td>91</td>
<td>Type of address of SMSC</td>
<td>hex-octet</td>
<td>internation format</td>
</tr>
<tr>
<td>13 26 04 00 00 F0</td>
<td>SMSC number</td>
<td>decimal semi-octets</td>
<td>316240000000</td>
</tr>
<tr>
<td>04</td>
<td>First octet of this SMS-DELIVER message.</td>
<td>hex-octet</td>
<td>TP-MMS</td>
</tr>
<tr>
<td>0B</td>
<td>Length of the sender address</td>
<td>hex-octet</td>
<td>11 (decimal)</td>
</tr>
<tr>
<td>91</td>
<td>Type of address of the sender number</td>
<td>hex-octet</td>
<td>...</td>
</tr>
<tr>
<td>13 46 61 00 89 F6</td>
<td>Sender number</td>
<td>decimal semi-octets</td>
<td>31641600986</td>
</tr>
<tr>
<td>00</td>
<td>Protocol identifier</td>
<td>hex-octets</td>
<td>...</td>
</tr>
<tr>
<td>00</td>
<td>Data encoding scheme</td>
<td>hex-octets</td>
<td>...</td>
</tr>
<tr>
<td>20 80 62 91 73 14 08</td>
<td>Time stamp</td>
<td>decimal semi-octets</td>
<td>06-08-02 29:17:31</td>
</tr>
<tr>
<td>0C</td>
<td>Length of User data (SMS message)</td>
<td>hex-octets</td>
<td>12 (decimal)</td>
</tr>
<tr>
<td>C8 F7 1D 14 96 97 41 F9 77 FD 07</td>
<td>User data</td>
<td>8-bit octets</td>
<td>How are you?</td>
</tr>
<tr>
<td></td>
<td></td>
<td>representing 7-bit data</td>
<td></td>
</tr>
</tbody>
</table>
Using the same project created in the previous section, add the following statements in bold to the AndroidManifest.xml file:

```xml
<uses-sdk android:minSdkVersion="10" />
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>

<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name">
    <activity
        android:label="@string/app_name"
        android:name=".SMSActivity">
        <intent-filter>
            <action android:name="android.intent.action.MAIN"/>
            <category android:name="android.intent.category.LAUNCHER"/>
        </intent-filter>
    </activity>
    <receiver android:name=".SMSReceiver">
        <intent-filter>
            <action android:name="android.provider.Telephony.SMS_RECEIVED"/>
        </intent-filter>
    </receiver>
</application>
```
Receiving SMS Programmatically
Receiving SMS Programmatically

Code the `SMSReceiver.java` file as follows:

```java
package net.learn2develop.SMS;

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

public class SMSReceiver extends BroadcastReceiver
{
    @Override
    public void onReceive(Context context, Intent intent)
    {
        //---get the SMS message passed in---
        Bundle bundle = intent.getExtras();
        SmsMessage[] msgs = null;
        String str = "SMS from ";
        if (bundle != null)
```
Receiving SMS Programmatically

```java
{  
    //---retrieve the SMS message received---
    Object[] pdus = (Object[]) bundle.get("pdus");  
    msgs = new SmsMessage[pdus.length];  
    for (int i=0; i<msgs.length; i++) {  
        msgs[i] = SmsMessage.createFromPdu((byte[])pdus[i]);  
        if (i==0) {  
            //---get the sender address/phone number---
            str += msgs[i].getOriginatingAddress();  
            str += ": ";
        }  
        //---get the message body---
        str += msgs[i].getMessageBody().toString();
    }  
    //---display the new SMS message---
    Toast.makeText(context, str, Toast.LENGTH_SHORT).show();
    Log.d("SMSReceiver", str);
}
```
Receiving SMS Programmatically

- Using the DDMS, send a message to the emulator (use the Emulator Control).
- Your application should be able to receive the message and display it using the Toast class.
Receiving SMS Programmatically

- To listen for incoming SMS messages, you create a `BroadcastReceiver` class.
- It enables your application to handle events raised by other applications.
- When an intent is received, the `onReceive()` method is called; hence, you need to override this.
- The SMS message is contained in the `Intent` object (`intent`; the second parameter in the `onReceive()` method) via a `Bundle` object.
Receiving SMS Programatically

- Each SMS message is stored in an Object array in the PDU format.
- If the SMS message is fewer than 160 characters, then the array will have one element. If an SMS message contains more than 160 characters, then the message will be split into multiple smaller messages and stored as multiple elements in the array.
Receiving SMS Programmatically

- To extract the content of each message, you use the static `createFromPdu()` method from the `SmsMessage` class.

- The phone number of the sender is obtained via the `getOriginatingAddress()` method; therefore, if you need to send an autoreply to the sender, this is the method to obtain the sender’s phone number.

- To extract the body of the message, you use the `getMessageBody()` method.
Receiving SMS Programmatically

- One interesting characteristic of the BroadcastReceiver is that your application will continue to listen for incoming SMS messages even if it is not running;

- As long as the application is installed on the device, any incoming SMS messages will be received by the application.