Phát triển ứng dụng Smartphone

Tài liệu lưu hành nội bộ

Đây là tài liệu tham khảo sử dụng trong môn học Lập trình ứng dụng Smartphone – Android được tổng hợp, biên soạn từ nhiều nguồn bởi các thành viên của Nhóm nghiên cứu và ứng dụng công nghệ A106-Đại học Hoa Sen.

Phát triển ứng dụng Smartphone

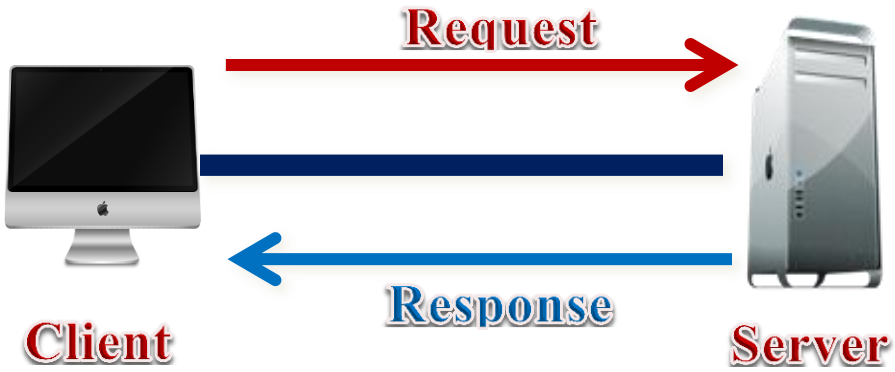
Phần 05: Network & Telephony

Lê Đức Huy

Email: leduchuy89vn@gmail.com

1 HTTP

Chức năng của HTTP như là một giao thức kiểu gởi yêu cầu-nhận hồi đáp trong mô hình điện toán máy chủ-máy khách. Trên HTTP, một trình duyệt web (Firefox, Internet Explorer, Chrome…) sẽ đóng vai trò như một người khách, trong khi ứng dụng chạy trên máy chủ (máy trung tâm) sẽ đóng vai trò như là một máy chủ. Người khash ở đây sẽ gởi một yêu cầu HTTP (HTTP request) lên máy chủ. Máy chủ, nơi lưu trữ một nguồn tài nguyên nào đó (Hình ảnh, âm thanh…) sẽ thay mặt máy khách thực thi một chức năng nào đó là trả về một hồi đáp cho máy khách. Hồi đáp này sẽ bao gồm thông tin trạng thái về yêu cầu được gởi đi cùng với nội dung được yêu cầu bởi máy khách trong nó. Giao thức HTTP được thể hiện qua hình sau:



HTTP Protocol

Trong mô hình trên tồn tại hai đối tượng chính: Client và Server. Trong đó đường dẫn kết nối hai đối tượng chính của mô hình là giao thức HTTP. Đối tượng Client sẽ gởi một Request đến máy Server để yêu cầu thực thi một tác vụ hoặc lấy một dữ liệu nào đó. Đối tượng Request chứa các tham số xác định địa chỉ nó được gởi đến cùng với các tham số khác để xác định tác vụ cần thực thi cũng như loại dữ liệu cần lấy. Các tham số này có thể thuộc hai loại: POST và GET. Tham số thuộc loại GET sẽ nằm trong đường dẫn địa chỉ mà đối tượng Request đang nắm giữ, tham số loại POST sẽ không nằm trong đường dẫn địa chỉ.

VD: Gởi đi request có địa chỉ sau: *http://www.google.com.vn/search?q=leduchuy*. Đây là một request dạng GET. Ta có thể thấy trên địa chỉ trên có một tham số q=leduchuy xác định từ khóa yêu cầu máy chủ Google tìm kiếm từ khóa: leduchuy.

Máy Server sẽ phân tích đối tượng Request để tìm ra yêu cầu của máy Client sử lý và trả về dữ liệu trong đối tượng Response. Máy Client sẽ phân tích đối tượng Response được trả về để lấy các thông tin cần thiết. Đây là cách mà giao thức HTTP hoạt động.

Việc sử dụng giao thức HTTP trên Android hiển nhiên cũng tuân theo cách trên với các bước như sau:

1.1 Sử dụng HTTP với GET Request

1. Khởi tạo một đối tượng HttpClient.
2. Khởi tạo một phương thức HTTP: PostMethod hoặc GetMethod.
3. Thiết lập tham số cho phương thức HTTP.
4. Thực thi HTTP request bằng cách sử dụng HttpClient.
5. Thao tao trên đối tượng Response được trả về.

Định nghĩa một class java như sau:

```
package niit.android;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;

public class NetworkManager {

     public String executeHttpGet(String uri) throws Exception
     {
          BufferedReader in = null;
          String result = "";
        try {
            HttpClient client = new DefaultHttpClient();
            HttpGet request = new HttpGet();
            request.setURI(new URI(uri));
            HttpResponse response = client.execute(request);
            in = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));

            StringBuffer sb = new StringBuffer("");
            String line = "";
            String NL = System.getProperty("line.separator");
            while ((line = in.readLine())!=null) {
               sb.append(line + NL);
            }

            in.close();
```

```
                result = sb.toString();
                } finally {
                    if (in != null) {
                        try {
                            in.close();
                        } catch (IOException e) {
                            e.printStackTrace();
                        }
                    }
                }
            return result;
        }
}
```

Trong đó:

HttpClient đại diện cho máy Client trong mô hình trên. Một đối tượng HttpClient đóng gói những tất cả các đối tượng cần thiết để thực thi một HTTP request (Địa chỉ gởi đi, các tham số đi kèm). Đây cũng là đối tượng sử lý cookie, chứng thực tài khoản, quản lý kết nối cũng như các tính năng khác của giao thức HTTP.

Sử dụng lệnh: **request.setURI(new URI("http://code.google.com/android/"));** để gán địa chỉ sẽ gởi đống tượng request đi.

Lệnh **HttpResponse response = client.execute(request);** sẽ thực thi một HTTP request và trả về một đối tượng Response.

Lệnh **response.getEntity()** sẽ trả về một đối tượng HttpEntity. Đây là đối tượng chứa các thẻ HTML (HTML entity) được trả về từ server.

Lệnh **getContent()** sẽ trả về một đối tượng kiểu InputStream. Đây là một đối tượng đại diện cho một luồng (stream) nối đến khối dữ liệu được trả về chứa trong response. Nó sẽ được dùng để đọc dữ liệu từ khối dữ liệu này.

Tiến hành khởi tạo một đối tượng InputStreamReader bằng lệnh **new InputStreamReader(…)** để tạo thành một bộ đọc từ luồng mới tạo ở trên.

Tiến hành tạo một BufferedReader bằng lệnh new BufferedReader() để tạo thành một bộ đọc có hổ trợ bộ đệm (buffer) để tăng tốc việc đọc dữ liệu từ luồn kể trên.

Tiến hành dựng một đối tượng String từ dữ liệu nhận về sau khi thực thi một HTTP request. Thực tế thì đối tượng String này chính là chuỗi chứa nội dung HTML được trả về. Việc dựng đối tượng String dựa vào StringBuffer (**sb.append(line + NL)**).

Lưu ý: Để cho phép ứng dụng kết nối Internet, ta cần bổ xung dòng sau vào file AndroidManifest.xml:

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

1.2 Sử dụng HTTP với POST Request

Các bước làm việc với giao thức HTTP cùng với một request dạng POST cũng tương tự như với request dạng GET. Ta tiến hành định nghĩa một class java có nội dung như sau:

```
package niit.android;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.URI;
import java.util.ArrayList;
import java.util.List;

import org.apache.http.HttpResponse;
import org.apache.http.NameValuePair;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;

public class NetworkManager {

     public String executeHttpPost(String
uri,List<NameValuePair>  postParameters) throws Exception {
              BufferedReader in = null;
                String result = "";
              try {
                  HttpClient client = new DefaultHttpClient();
                  HttpPost request = new HttpPost(uri);

                  UrlEncodedFormEntity formEntity = new
UrlEncodedFormEntity(
                           postParameters);

                  request.setEntity(formEntity);
                  HttpResponse response =
client.execute(request);
                  in = new BufferedReader(new
InputStreamReader(response.getEntity()
                           .getContent()));

                  StringBuffer sb = new StringBuffer("");
```

```
                String line = "";
                String NL =
System.getProperty("line.separator");
                while ((line = in.readLine()) != null) {
                    sb.append(line + NL);
                }
                in.close();
                result = sb.toString();

            } finally {
                if (in != null) {
                    try {
                        in.close();
                    } catch (IOException e) {
                        e.printStackTrace();
                    }
                }
            }
            return result;
        }

}
```

Trong đó:

Ta định nghĩa List<NameValuePair> postParameters = new ArrayList<NameValuePair>(); để chứa các tham số gởi đi với POST Request. Các tham số này là dữ liệu dùng để thực thi các hành động trên máy chủ hoặc các tham số dùng để xác định dữ liệu trả về.

Dùng lệnh postParameters.add(new BasicNameValuePair("one", "valueGoesHere")); để thêm các tham số vào danh sách tham số đã định nghĩa ở trên.

Tiến hành mã hóa số tham số trên để gởi kèm request: UrlEncodedFormEntity formEntity = new UrlEncodedFormEntity(postParameters);

Sử dụng class **NetworkManager** trên như sau:

```
String resultGet="";
String resultPost="";

NetworkManager networkManager = new NetworkManager();

try {
     resultGet =
networkManager.executeHttpGet("http://www.google.com.vn/search?q
=hoasen+university");
     System.out.print(result);
     } catch (Exception e) {
```

```
                e.printStackTrace();
            }

        }

        List<NameValuePair> postParameters = new
ArrayList<NameValuePair>();
        postParameters.add(new BasicNameValuePair("q",
"hoasen+university"));

        try {
            resultPost = networkManager.executeHttpPost("www.abc.com",
        postParameters);
            System.out.print(result);
        } catch (Exception e) {
            e.printStackTrace();
        }
```

2 Telephony

2.1 SMS

2.1.1 Gởi tin nhắn

Tạo Project với các thông số:

Project name: TelephonyExample

Build target: Android 2.3.3.

Application name: Telephony Example

Package name: niit.android

Create Activity: main

Để ứng dụng có thể gởi/nhận tin nhắn trên ứng dụng, ta cần bổ xung đăng kí trên file AndroidManifest.xml bằng cách bổ xung hai dòng sau:

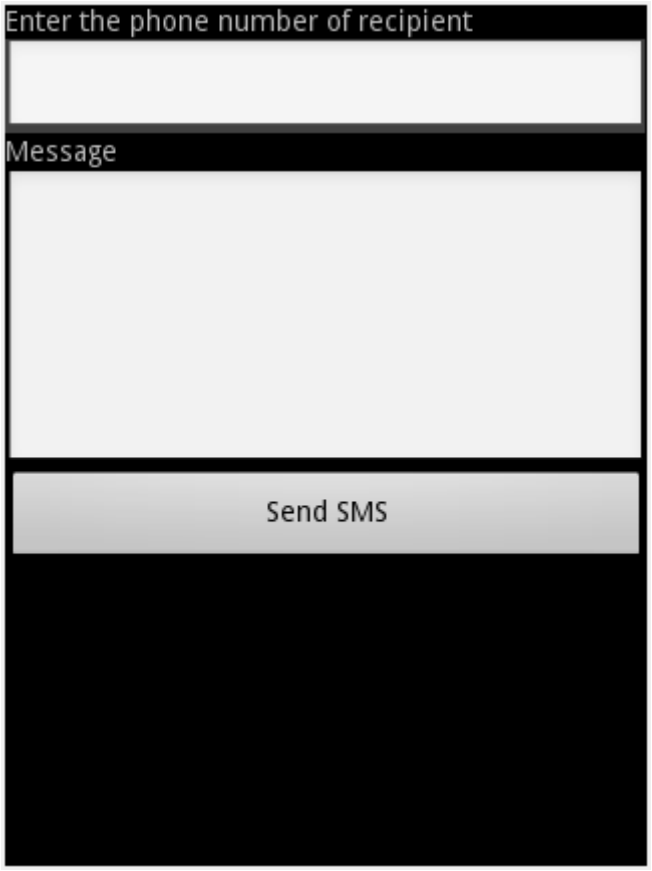
```
<uses-permission android:name="android.permission.SEND_SMS">
</uses-permission>
<uses-permission android:name="android.permission.RECEIVE_SMS">
</uses-permission>
```

Chỉnh sửa file main.xml với nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Enter the phone number of recipient"
        />
    <EditText
        android:id="@+id/txtPhoneNo"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Message"
        />
    <EditText
        android:id="@+id/txtMessage"
        android:layout_width="fill_parent"
        android:layout_height="150px"
        android:gravity="top"
        />
    <Button
        android:id="@+id/btnSendSMS"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Send SMS"
        />
</LinearLayout>
```

để có được giao diện như sau:



Thay đổi file main.java với nội dung như sau:

```
package niit.android;

import android.app.Activity;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.SmsManager;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

public class main extends Activity {
      Button btnSendSMS;
      EditText txtPhoneNo;
      EditText txtMessage;

      /** Called when the activity is first created. */
      @Override
      public void onCreate(Bundle savedInstanceState)
      {
```

```
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);

            btnSendSMS = (Button) findViewById(R.id.btnSendSMS);
            txtPhoneNo = (EditText) findViewById(R.id.txtPhoneNo);
            txtMessage = (EditText) findViewById(R.id.txtMessage);

            btnSendSMS.setOnClickListener(new
view.OnClickListener()
            {
                  public void onClick(View v)
                  {
                        String phoneNo =
txtPhoneNo.getText().toString();
                        String message =
txtMessage.getText().toString();
                        if (phoneNo.length()>0 &&
message.length()>0)
                              sendSMS(phoneNo, message);
                        else
                        Toast.makeText(getBaseContext(),
                            "Please enter both phone number and
message.",
                        Toast.LENGTH_SHORT).show();
                  }
            });
      }
      //---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);
      }
}
```

Để gởi một tin nhắn SMS, ta sử dụng một đối tượng kiểu SmsManager. Tuy nhiên, không giống như các class khác, ta không tiến hành khởi tạo một đối tượng kiểu này; thay vào đó ta sẽ gọi phương thức static tên getDefault() để trả về đối tượng SmsManager mặc định của thiết bị. Trong đó phương thức phương thức sendTextMessage() sẽ gởi tin nhắn SMS với một đối tượng kiểu PendingIntent. Đối tượng PendingIntent này dùng để xác định đối tượng sẽ gọi sau này. Ví dụ: Sau khi gởi tin nhắn, ta có thể sử dụng đối tượng PendingIntent để hiển thị một Activity. Trong trường hợp trên, đối tượng PendingIntent đơn giản là chỉ đến activity đã gởi tin nhắn đi, bằng cách này sẽ không có việc gì sảy ra sau khi tin nhắn được gởi đi. Ta thay đổi phương thức sendSMS với nội dung như sau:

```
private void sendSMS(String phoneNumber, String message)
{
```

```
        PendingIntent sentIntent =
PendingIntent.getActivity(this, 0, new Intent(this,
SentActivity.class), 0);
        PendingIntent deliveryIntent =
PendingIntent.getActivity(this, 0, new Intent(this,
DeliveryActivity.class), 0);

        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message,
sentIntent, deliveryIntent);
}
```

Trong đó ta tạo ra hai đối tượng PendingIntent bằng cách gọi phương thức getActivity(). Phương thức này sẽ trả về một PendingIntent dùng để khởi động một activity với các thông số chỉ định đi kèm. Ta tạo hai đối tượng kiểu PendingIntent dùng để khởi động hai activity, một activity sẽ được mở ra khi tin nhắn SMS đã được gởi đi (SentActivity) và một activity sẽ được mở ra khi tin nhắn SMS đã được chuyển giao thành công.

Nếu bạn mong muốn theo dõi tiến trình gởi tin nhắn, bạn có thể sử dụng PendingIntent cùng với hai BroadcastReceiver như sau:

```
private void sendSMS(String phoneNumber, String message)
    {
        String SENT = "SMS_SENT";
        String DELIVERED = "SMS_DELIVERED";

        PendingIntent sentPI = PendingIntent.getBroadcast(this,
0,
            new Intent(SENT), 0);

        PendingIntent deliveredPI =
PendingIntent.getBroadcast(this, 0,
            new Intent(DELIVERED), 0);

        //---when the SMS has been sent---
        registerReceiver(new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent arg1) {
                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(),
"Generic failure",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_NO_SERVICE:
                        Toast.makeText(getBaseContext(), "No
service",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_NULL_PDU:
                        Toast.makeText(getBaseContext(), "Null
PDU",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case SmsManager.RESULT_ERROR_RADIO_OFF:
                        Toast.makeText(getBaseContext(), "Radio
off",
                                Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        }, new IntentFilter(SENT));

        //---when the SMS has been delivered---
        registerReceiver(new BroadcastReceiver(){
            @Override
            public void onReceive(Context arg0, Intent arg1) {
                switch (getResultCode())
                {
                    case Activity.RESULT_OK:
                        Toast.makeText(getBaseContext(), "SMS
delivered",
                                Toast.LENGTH_SHORT).show();
                        break;
                    case Activity.RESULT_CANCELED:
                        Toast.makeText(getBaseContext(), "SMS
not delivered",
                                Toast.LENGTH_SHORT).show();
                        break;
                }
            }
        }, new IntentFilter(DELIVERED));

        SmsManager sms = SmsManager.getDefault();
        sms.sendTextMessage(phoneNumber, null, message, sentPI,
deliveredPI);
}
```

Trong phần code trên ta dùng một đối tượng PendingIntent (sendPI) để theo dõi quá trình gởi dữ tin nhắn. Khi một tin nhắn SMS được gởi đi, phương thức onReceive của BroadcastReceiver đầu tiên sẽ được thực thi. Trong phương thức này ta sẽ tiến hành kiểm tra trạng thái của tiến trình gởi tin nhắn. Đối tượng PendingIntent thứ hai dùng để theo dõi quá trình chuyển giao tin nhắn. Phương thức onReceive của BroadcastReceiver thứ hai sẽ được thực thi khi quá trình chuyển giao một tin nhắn kết thúc. Tại đây ta cũng tiến hành kiểm tra trạng thái của việc chuyển giao tin nhắn. Ta sử dụng phương thức getBroadcast() để tạo một đối tượng PendingIntent dùng để khởi động một Broadcast receiver.

Đối tượng Intent được tạo trong trường hợp này có intent filter là String SENT = "SMS_SENT"; và String DELIVERED = "SMS_DELIVERED";. Hai biến này đơn giản là bộ lọc dùng để chọn đúng BroadcastReceiver mà ta sẽ định nghĩa ở dưới.

Phương thức registerReceiver() dùng để đăng kí một BroadcastRecevier sẽ chạy trên luồng của activity hiện tại. Phương thức này sẽ nhận vào hai tham số: tham số thứ nhất chính là đối tượng cài đặt lại interface BroadcastReceiver, tham số thứ hai là đối tượng Intent xác định những Intent cùng loại sẽ được dùng để khởi động BroadcastReceiver này.

Đến đây, để kiểm tra sự hoạt động của chương trình ta cần hai máy ảo Android có khả năng gởi nhận tin nhắn với nhau. Để làm được việc này, trước tiên ta cần mở thêm một máy ảo Android thứ hhai bằng cách mở Window/Android SDK and AVD Manager trên Eclipse. Tạo một máy ảo mới và nhấn " "Start" để khởi động máy ảo mới tạo để được kết quả như sau:

Để gởi tin nhắn giữa hai máy ảo với nhau ta thay số điện thoại người nhận bằng cách thay đổi số điện thoại người nhận bằng Emulator's port của máy ảo. Emulator's port của máy ảo trên là 5556 hiển thị ở góc trái trên cùng của máy ảo.

Từ máy ảo đầu tiên, sử dụng ứng dụng mới xây dựng ở trên để tiến hành gởi tin nhắn đến máy ảo thứ hai để kiểm thử chương trình.

2.1.2 Nhận tin nhắn

Bên cạnh việc gởi tin nhắn ta còn có thể định nghĩa một BroadcastReceiver dùng để nhận tin nhắn được gởi đến thiết bị.

Tiến hành chỉnh sửa file AndroidManifest.xml bằng cách thêm một thẻ receiver để đạt được nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
      package="niit.android"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity android:name=".main"
                  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=".SmsReceiver">
            <intent-filter>
                <action android:name=
                    "android.provider.Telephony.SMS_RECEIVED" />
            </intent-filter>
        </receiver>
    </application>
    <uses-permission android:name="android.permission.SEND_SMS">
    </uses-permission>
    <uses-permission
android:name="android.permission.RECEIVE_SMS">
    </uses-permission>
</manifest>
```

Thêm một class SmsReceiver với nội dung như sau:

```
package niit.android;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.telephony.gsm.SmsMessage;
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 = "";
        if (bundle != null)
        {
            //---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]);
                str += "SMS from " +
msgs[i].getOriginatingAddress();
                str += " :";
                str += msgs[i].getMessageBody().toString();
                str += "\n";
            }
            //---display the new SMS message---
            Toast.makeText(context, str,
Toast.LENGTH_SHORT).show();
        }
    }

}
```

2.1.3 Làm việc với các thư mục chứa tin nhắn

Tạo file sms_folder.xml với nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- This file is /res/layout/sms_inbox.xml -->
```

```
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView android:id="@+id/row"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"/>

</LinearLayout>
```

Bổ xung vào file AndroidManifest.xml nội dung như sau:

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

Tạo file SMSFolderExampleActivity.java với nội dung:

```
package niit.android;

import android.app.ListActivity;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.widget.ListAdapter;
import android.widget.SimpleCursorAdapter;

public class SMSFolderExampleActitvity extends ListActivity {
    private ListAdapter adapter;
    private static final Uri SMS_INBOX =
Uri.parse("content://sms/inbox");

    @Override
    public void onCreate(Bundle bundle) {
        super.onCreate(bundle);
        Cursor c = getContentResolver().query(SMS_INBOX, null,
null, null, null);
        startManagingCursor(c);
        String[] columns = new String[] {"body"};
        int[] names = new int[] { R.id.row };
        adapter = new SimpleCursorAdapter(this,
R.layout.sms_folder, c, columns,
                names);

        setListAdapter(adapter);
    }
}
```

Lệnh Cursor c = getContentResolver().query(SMS_INBOX, null, null, null, null); sẽ tạo ra một đối tượng con trỏ dùng để đọc tin nhắn trong thư mục inbox. Lệnh startManagingCursor sẽ báo cho activity hiện tại quản lý vòng đời của con trỏ vừa mới tạo ra (Khi activity thay đổi trạng thái nó cũng thay đổi trạng thái của con trỏ). Bằng cách này khi Activity bị loại khỏi bộ nhớ nó cũng dọn dẹp con trỏ này. Ngoài cách sử dụng SimpleCursorAdapter ta còn có thể sử dụng con trỏ để đọc dữ liệu theo cách sau:

```
result.moveToFirst();
while (!result.isAfterLast()) {
     String address = result.getString(2);
     String body =result.getString(11);
     result.moveToNext();
}
```

Dưới đây là danh sách các thư mục chứa tin nhắn SMS:

- All: content://sms/all
- Inbox: content://sms/inbox
- Sent: content://sms/sent
- Draft: content://sms/draft
- Outbox: content://sms/outbox
- Failed: content://sms/failed
- Queued: content://sms/queued
- Undelivered: content://sms/undelivered
- Conversations: content://sms/conversations

2.2 Gởi email

Android không hổ trợ gởi email trực tiếp từ ứng dụng nên ta sẽ phải tạo một đối tượng Intent để khởi động cửa sổ Activity dùng gởi mail như sau:

```
Intent emailIntent=new Intent(Intent.ACTION_SEND);

String subject = "Hi!";
String body = "hello from android....";

String[] extra = new String[]{"leduchuy89vn@gmail.com"};
emailIntent.putExtra(Intent.EXTRA_EMAIL, extra);

emailIntent.putExtra(Intent.EXTRA_SUBJECT, subject);
emailIntent.putExtra(Intent.EXTRA_TEXT, body);
emailIntent.setType("message/rfc822");
```

```
startActivity(emailIntent);
```

2.3 Làm việc với Telephony maganager

Giả sử ta viết một ứng dụng chơi nhạc cho hệ điều hành Android, khi đó ta cần bắt được các sự kiện khi có cuộc gọi đến để ngừng chơi nhạc. Hoặc khi cuộc điện thoại kết thúc ta cần tiếp tục chơi nhạc. Để làm được việc này ta có thể sử dụng TelephonyManager được cung cấp sẵn trong Android với cách thức như sau:

```
package niit.android;

import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.telephony.PhoneStateListener;
import android.telephony.TelephonyManager;
import android.util.Log;
import android.widget.Toast;

public class TelephonyServiceExampleActivity extends Activity{
    private static final String TAG="TelephonyServiceDemo";
    @Override
    protected void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        TelephonyManager teleMgr =
(TelephonyManager)getSystemService(Context.TELEPHONY_SERVICE);
        teleMgr.listen(new MyPhoneStateListener(),
PhoneStateListener.LISTEN_CALL_STATE);
    }

    class MyPhoneStateListener extends PhoneStateListener
    {
        @Override
        public void onCallStateChanged(int state, String
incomingNumber) {
            super.onCallStateChanged(state, incomingNumber);
            Toast toast =
Toast.makeText(TelephonyServiceExampleActivity.this, "", 3000);
            switch(state)
            {

                case TelephonyManager.CALL_STATE_IDLE:
                    Log.d(TAG, "call state idle...incoming
number is["+
```

```
incomingNumber+"]");break;
                case TelephonyManager.CALL_STATE_RINGING:
                    Log.d(TAG, "call state ringing...incoming
number is["+
incomingNumber+"]");break;
                case TelephonyManager.CALL_STATE_OFFHOOK:
                    Log.d(TAG, "call state Offhook...incoming
number is["+
incomingNumber+"]");break;
                default:
                    Log.d(TAG, "call state ["+state+"]incoming
number is["+
incomingNumber+"]");break;
            }
        }
    }
}
```