Android

Intents, Adapters

2010.04.02

Database Laboratory
Hanyang Univ.

Select a Contact Example

- You'll create a new sub-Activity that services the *PICK_ACTION* for contact data
- It displays each of the contacts in the contact database
- Lets the user select one, before closing and returning its URI to the calling Activity

1. Create a new ContactPicker project that includes a *ContactPicker* Activity

```
package com.paad.contactpicker;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.Contacts.People;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.AdapterView.OnItemClickListener


public class ContactPicker extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

Select a Contact Example

2. Modify the main.xml layout resource to include a single *ListView* control

- This control will be used to display the contacts

```
<?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"
    >
    <ListView
    android:layout_height="wrap_content"
    android:id="@+id/contactListView"
    android:layout_width="fill_parent">
    </ListView>
</LinearLayout>
```

Select a Contact Example

3. Create a new listitemlayout.xml layout resource that includes a single Text View

- This will be used to display each contact in the List View

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_height="fill_parent"
  android:layout_width="fill_parent"
  android:orientation="vertical">
    <TextView
    android:id="@+id/itemTextView"
    android:textColor="#FFF"
    android:textSize="16px"
    android:padding="10px"
    android:layout_height="wrap_content"
    android:layout_width="fill_parent">
    </TextView>
</LinearLayout>
```

Select a Contact Example

4. Return to the ContactPicker Activity. Override the onCreate method

- extract the data path from the calling Intent

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

        Intent intent = getIntent();
        String dataPath = intent.getData().toString();
    }
}
```

Select a Contact Example

① Create a new data URI for the people stored in the contact list
- bind it to the List View using a *SimpleCursorArrayAdapter*
- *simpleCursorArrayAdapter :* lets you assign Cursor data, used by Content Providers, to Views

```
public class ContactPicker extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        Intent intent = getIntent();
        String dataPath = intent.getData().toString();

        final Uri data = Uri.parse(dataPath + "people/");
        final Cursor c = managedQuery(data, null, null, null, null);

        String[] from = new String[] {People.NAME};
        int[] to = new int[] { R.id.itemTextView };

        SimpleCursorAdapter adapter = new SimpleCursorAdapter(this,
                R.layout.listitemlayout,
                c,
                from,
                to);
        ListView lv = (ListView)findViewById(R.id.contactListView);
        lv.setAdapter(adapter);
    }
```

Select a Contact Example

② Add an *ItemClickListener* to the List View

- Selecting a contact from the list should return a path to the item to the calling Activity

```
    lv.setOnItemClickListener(new OnItemClickListener() {
        public void onItemClick(AdapterView<?> parent, View view, int pos,
        long id) {
        // Move the cursor to the selected item
        c.moveToPosition(pos);
        // Extract the row id.
        int rowId = c.getInt(c.getColumnIndexOrThrow("_id"));
        // Construct the result URI.
        Uri outURI = Uri.parse(data.toString() + rowId);
        Intent outData = new Intent();
        outData.setData(outURI);
        setResult(Activity.RESULT_OK, outData);
        finish();
        }
        });
}
```

Select a Contact Example

5. Modify the application manifest and replace the *intent-filter* tag of the Activity to add support for the pick action on contact data

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.paad.contactpicker"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name
        <activity android:name=".ContactPicker"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.PICK" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:path="contacts"
                    android:scheme="content">
                    </data>
            </intent-filter>
        </activity>

    </application>
    <uses-sdk android:minSdkVersion="7" />
</manifest>
```

Select a Contact Example

6. To test it, create a new test harness *ContentPickerTester* Activity

- Create a new layout resource — contentpickertester — that includes a TextView to display the selected contact and a Button to start the sub-activity

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_height="fill_parent"
  android:layout_width="fill_parent"
  android:orientation="vertical">
    <TextView
    android:id="@+id/selected_contact_textview"
    android:padding="10px"
    android:textSize="20px"
    android:textColor="#FFF"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    </TextView>
    <Button
    android:id="@+id/pick_contact_button"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Pick Contact">
    </Button>
</LinearLayout>
```

Select a Contact Example

7. Override the *onCreate* method of the *ContentPickerTester* to add a Click Listener to the button so that it implicitly starts a new sub-Activity by specifying the *PICK_ACTION* and the contact database URI *(content://contacts/)*.

```
package com.paad.contactpicker;

import android.app.Activity;
import android.content.Intent;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.Contacts.People;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class ContentPickerTester extends Activity {
    public static final int PICK_CONTACT = 1;
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.contentpickertester);
        Button button = (Button)findViewById(R.id.pick_contact_button);
        button.setOnClickListener(new OnClickListener() {
            public void onClick(View _view) {
                Intent intent = new Intent(Intent.ACTION_PICK,
                        Uri.parse("content://contacts/"));
                            startActivityForResult(intent, PICK_CONTACT);
            }
        });
    }
}
```

Select a Contact Example

8. When the sub-Activity returns, use the result to populate the Text View with the selected contact's

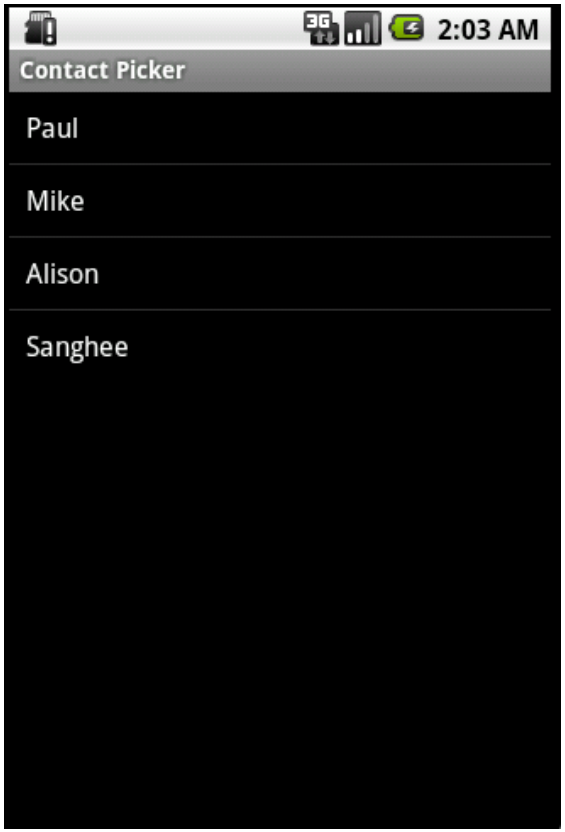
```
    @Override
    public void onActivityResult(int reqCode, int resCode, Intent data) {
        super.onActivityResult(reqCode, resCode, data);
        switch(reqCode) {
        case (PICK_CONTACT) : {
            if (resCode == Activity.RESULT_OK) {
                Uri contactData = data.getData();
                Cursor c = managedQuery(contactData, null, null, null, null);
                c.moveToFirst();
                String name;
                name = c.getString(c.getColumnIndexOrThrow(People.NAME));
                TextView tv;
                tv = (TextView)findViewById(R.id.selected_contact_textview);
                tv.setText(name);
            }
            break;
        }
    }
}
```

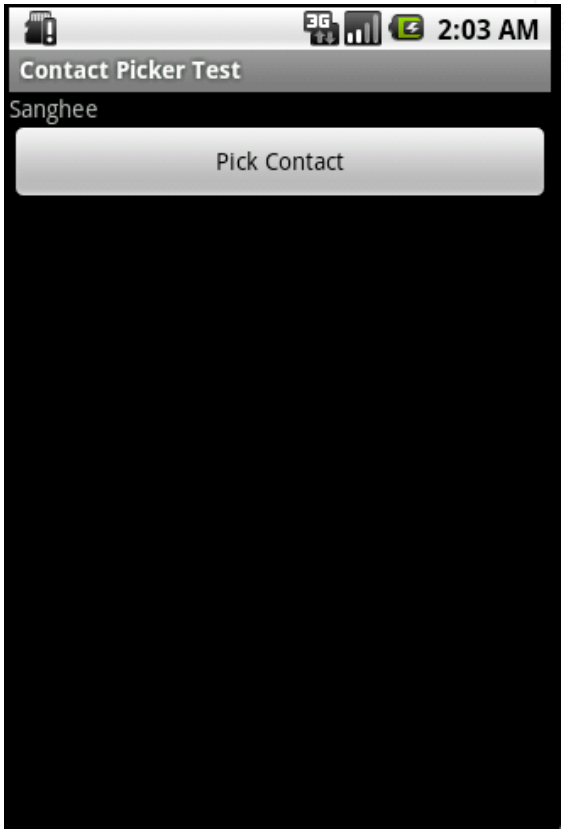
Select a Contact Example

9. **With your test harness complete, simply add it to your application manifest**
   - You'll also need to add a READ_CONTACTS permission within a uses-permission tag, to allow the application to access the contacts database

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.paad.contactpicker"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".ContactPicker"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.PICK" />
                <category android:name="android.intent.category.DEFAULT" />
                <data android:path="contacts"
                    android:scheme="content">
                    </data>
            </intent-filter>
        </activity>
        <activity android:name=".ContentPickerTester"
                  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>
    <uses-permission android:name="android.permission.READ_CONTACTS"/>
    <uses-sdk android:minSdkVersion="7" />
</manifest>
```

Select a Contact Example





Call Contact