



## Bài 1: Lập trình Adroi cơ bản

### Yêu cầu kiến thức cho lập trình Android:

Để lập trình android, mình nghĩ mọi người chỉ cần kiến thức java căn bản là hoàn toàn ok. Căn bản ở đây có nghĩa là hiểu được thế nào là class, package, biết ý nghĩa của các từ khóa như public, private, protected,... thành thạo các lệnh cơ bản như if, for(), switch(), while(), ... biết sd các lệnh như Integer.parseInt() hay String.valueOf()... Nên có thêm kiến thức về gói java.util vì đây là gói hỗ trợ nhiều lớp rất mạnh được sử dụng trên mọi nền, ngoài ra các gói như java.io, java.net... cũng được recommended

Các kiến thức về các gói lập trình cho desktop như java.awt, java.swing hoàn toàn không cần thiết (bản thân mình cũng chưa sd cái này bao giờ, nhảy vào học java là học J2ME luôn), hay các gói của J2ME cũng vậy Lập trình Android tuy cũng là lập trình di động, nhưng các điện thoại sử dụng hđh Android có cấu hình rất mạnh (Nexus One có VXL lên tới 1Ghz), vì vậy 2 nền tảng Android và J2ME cũng rất khác nhau. Android có những gói riêng hỗ trợ lập trình cho nó và không yêu cầu khắt khe về việc tối ưu code như J2ME. Thật đáng tiếc vì J2ME mình học ko ứng dụng được mấy vào lập trình Android (tuy nhiên 1 số kỹ thuật cơ bản cho lập trình game 2D như Sprite, double buffering, Tile... thì vẫn ko hề phí phạm chút nào )

### Cài đặt Android để lập trình:

Để lập trình Android thì mỗi bộ SDK của Google là không đủ, bạn còn cần tích hợp nó vào một IDE như Eclipse. Anh Giáp đã có 2 bài hướng dẫn rất chi tiết về cài đặt Android trong Eclipse cũng như Netbeans, nhưng theo mình mọi người nên sử dụng Eclipse hơn vì nó có nhiều tính năng hỗ trợ lập trình Google, còn Netbeans thì plugin cho Android vẫn chưa hoàn thiện

Tiện thể mình nói luôn, mình học Android theo 2 cuốn *Professional Android Application Development* và *Unlocking Android*. Cả 2 cuốn đều dành cho beginner nhưng cuốn đầu code nhiều, giải thích ít, cuốn thứ 2 giải thích rõ ràng hơn. Nếu có ai có ý định tham khảo thì nên đọc cuốn UA trước để hiểu rõ hơn Android, sử dụng cuốn PAAD trong việc tham khảo các đoạn code cho lập trình.

### • Understanding Android Application:

Việc hiểu được các thành phần (component) tạo nên một ứng dụng Android là rất cần thiết cho việc lập trình. Các thành phần này được chia làm 6 loại bao gồm:

- 1.Activity: hiểu một cách đơn giản thì Activity là nền của 1 ứng dụng. Khi khởi động 1 ứng dụng Android nào đó thì bao giờ cũng có 1 main Activity được gọi, hiển thị màn hình giao diện của ứng dụng cho phép người dùng tương tác.
- 2.Service: thành phần chạy ẩn trong Android. Service sử dụng để update dữ liệu, đưa ra các cảnh báo (Notification) và không bao giờ hiển thị cho người dùng thấy.
- 3.Content Provider: kho dữ liệu chia sẻ. Content Provider được sử dụng để quản lý và chia sẻ dữ liệu giữa các ứng dụng.
- 4.Intent: nền tảng để truyền tải các thông báo. Intent được sử dụng để gửi các thông báo đi nhằm khởi tạo 1 Activity hay Service để thực hiện công việc bạn mong muốn. VD: khi mở 1 trang web, bạn gửi 1 intent đi để tạo 1 activity mới hiển thị trang web đó.

5.Broadcast Receiver: thành phần thu nhận các Intent bên ngoài gửi tới. VD: bạn viết 1 chương trình thay thế cho phần gọi điện mặc định của Android, khi đó bạn cần 1 BR để nhận biết các Intent là các cuộc gọi tới.

6.Notification: đưa ra các cảnh báo mà không làm cho các Activity phải ngừng hoạt động.

Activity, Service, Broadcast Receiver và Content Provider mới là những thành phần chính cấu thành nên ứng dụng Android, bắt buộc phải khai báo trong AndroidManifest (tham khảo bài 2 có giới thiệu đầy đủ về file này).

Understanding Android Application Life Cycle:

Android có cơ chế quản lý các process theo chế độ ưu tiên. Các process có priority thấp sẽ bị Android giải phóng mà không hề cảnh báo nhằm đảm bảo tài nguyên.

1.Foreground process: là process của ứng dụng hiện thời đang được người dùng tương tác.

2.Visible process: là process của ứng dụng mà activity đang hiển thị đối với người dùng (onPaused() của activity được gọi).

3.Service process: là Service đang running.

4.Background process: là process của ứng dụng mà các activity của nó ko hiển thị với người dùng (onStoped() của activity được gọi).

5.Empty process: process không có bất cứ 1 thành phần nào active.

Theo chế độ ưu tiên thì khi cần tài nguyên, Android sẽ tự động kill process, trước tiên là các empty process.

Android Activity Life Cycle:

Như mình đã giới thiệu ở trên , Activity là thành phần quan trọng nhất và đóng vai trò chính trong xây dựng ứng dụng Android. Hệ điều hành Android quản lý Activity theo dạng stack: khi một Activity mới được khởi tạo, nó sẽ được xếp lên đầu của stack và trở thànhrunning activity, các Activity trước đó sẽ bị tạm dừng và chỉ hoạt động trở lại khi Activity mới được giải phóng.

Activity bao gồm 4 state:

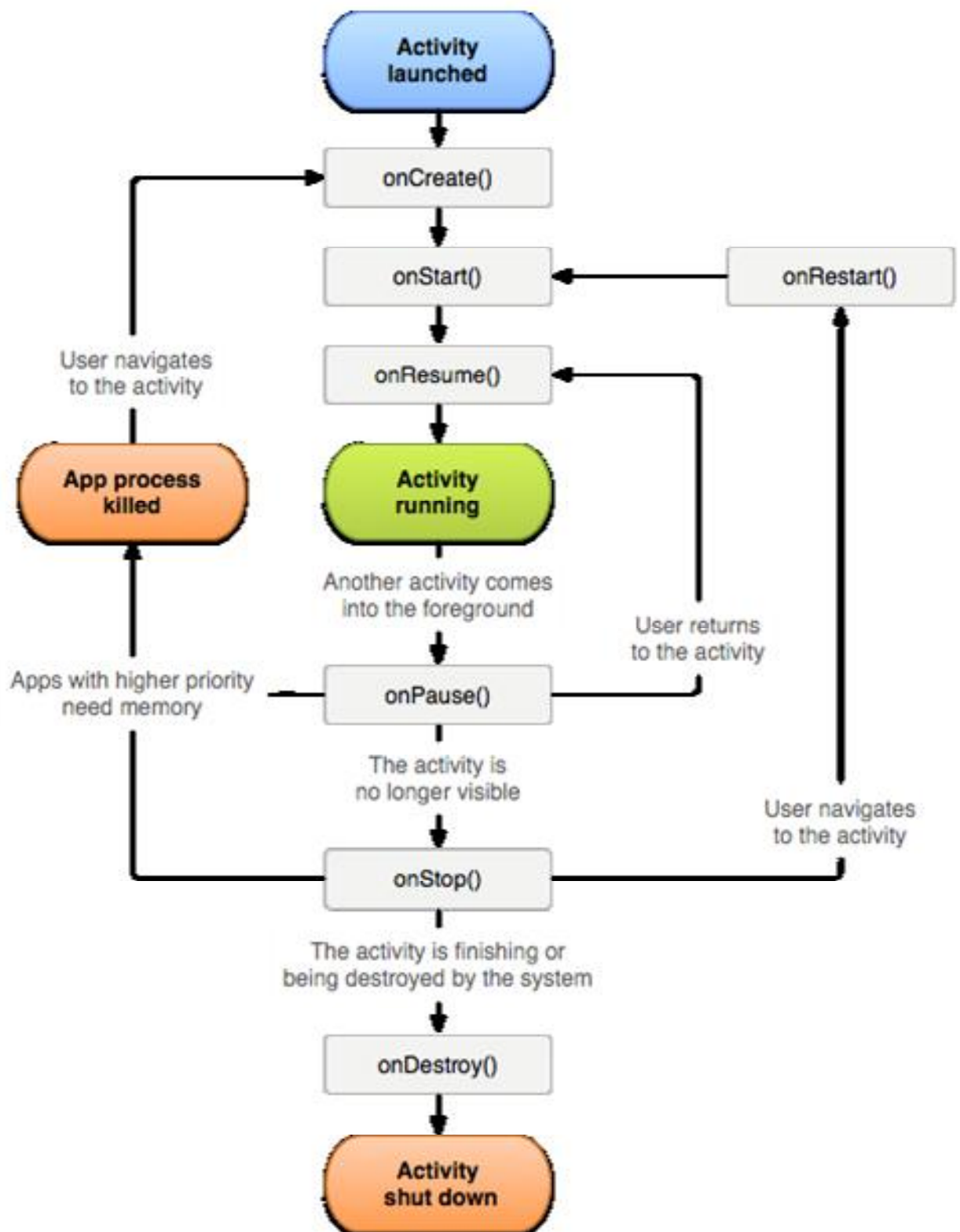
- active (running): Activity đang hiển thị trên màn hình (foreground).

- paused: Activity vẫn hiển thị (visible) nhưng không thể tương tác (lost focus). VD: một activity mới xuất hiện hiển thị giao diện đè lên trên activity cũ, nhưng giao diện này nhỏ hơn giao diện của activity cũ, do đó ta vẫn thấy được 1 phần giao diện của activity cũ nhưng lại không thể tương tác với nó.

- stop: Activity bị thay thế hoàn toàn bởi Activity mới sẽ tiến đến trạng thái stop

- killed: Khi hệ thống bị thiếu bộ nhớ, nó sẽ giải phóng các tiến trình theo nguyên tắc ưu tiên. Các Activity ở trạng thái stop hoặc paused cũng có thể bị giải phóng và khi nó được hiển thị lại thì các Activity này phải khởi động lại hoàn toàn và phục hồi lại trạng thái trước đó.

Biểu đồ miêu tả Activity state



Vòng đời của Activity:

- Entire lifetime: Từ phương thức `onCreate()` cho tới `onDestroy()`
- Visible lifetime: Từ phương thức `onStart()` cho tới `onStop()`

- Foreground lifetime: Từ phương thức onResume( ) cho tới onPause( )

Khi xây dựng Activity cho ứng dụng cần phải viết lại phương thức onCreate( ) để thực hiện quá trình khởi tạo. Các phương thức khác có cần viết lại hay không tùy vào yêu cầu lập trình.

XML trong Android:

Không giống như lập trình java thông thường, lập trình android ngoài các lớp được viết trong \*.java còn sử dụng XML để thiết kế giao diện cho ứng dụng. Tất nhiên bạn hoàn toàn có thể thiết kế 1 giao diện như ý muốn mà không cần tới bất cứ 1 dòng XML nào, nhưng sd XML sẽ đơn giản công việc đi rất nhiều. Đồng thời sd XML sẽ giúp việc chỉnh sửa ứng dụng sau này trở nên dễ dàng.

Về nguyên tắc, khi lập trình ứng dụng ta thiết kế giao diện bằng XML và cài đặt các xử lý khi tương tác với giao diện trong code.

1 số thành phần cơ bản trong Android:

1.Các layout:

Layout được dùng để quản lý các thành phần giao diện khác theo 1 trật tự nhất định.

- FrameLayout: Layout đơn giản nhất, thêm các thành phần con vào góc trên bên trái của màn hình.

- LinearLayout: thêm các thành phần con theo 1 chiều nhất định (ngang hoặc dọc). Đây là layout được sử dụng nhiều nhất.

- RelativeLayout: thêm các thành phần con dựa trên mối quan hệ với các thành phần khác hoặc với biên của layout.

- TableLayout: thêm các thành phần con dựa trên 1 lưới các ô ngang và dọc.

- AbsoluteLayout: thêm các thành phần con dựa theo tọa độ x, y.

Layout được sử dụng nhằm mục đích thiết kế giao diện cho nhiều độ phân giải. Thường khi lập trình nên kết hợp nhiều layout với nhau để tạo ra giao diện bạn mong muốn.

2.XML unit:

Để hiểu được các thành phần cơ bản của XML cũng như việc sử dụng XML kết hợp với code, ta sẽ đi xây dựng thử một chương trình đơn giản.

Yêu cầu: Xây dựng 1 ứng dụng cho phép gõ 1 nội dung vào rồi hiển thị ra nội dung đó ở bên dưới.

B1: Khởi tạo 1 project (ở đây sử dụng Eclipse để minh họa).

Vào thẻ File -> New -> Android Project. Nếu bạn mới lập trình Android lần đầu thì có lẽ dòng Android Project sẽ không hiện ra, khi đó xuống phía cuối chọn Other rồi vào Android -> Android Project.

B2: Điền thông tin cho project

Project name: Example 1

Build Target: Chọn Android 1.5 (mới nhất là 2.1 nhưng hiện tại bạn chưa cần quan tâm )

Application name: Example 1

Package name: at.exam

Create Activity: Example

=> Kích nút Finish.

B3: Bên khung Package Explore bên trái đi tới thư mục res, bạn sẽ thấy có 3 thư mục con:

- drawable: thư mục chứa các hình ảnh để làm icon hoặc tài nguyên cho giao diện...
- layout: chứa các file xml để thiết kế giao diện.
- values: chứa các giá trị sử dụng trong ứng dụng được bạn định nghĩa, như các dòng ký tự (string), các màu (color), các themes...

B4: Vào thư mục layout, chọn file main.xml và gõ đoạn code sau vào thay cho toàn bộ nội dung có sẵn (Eclipse hỗ trợ kéo thả cho xml nhưng theo mình không nên sử dụng):

Mã:

```
<?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"
>
<EditText
android:id="@+id/edit_text"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:hint="@string/edit_hint"
/>
<TextView
android:id="@+id/text_view"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:textColor="@color/text_color"
android:textSize="28px"
android:typeface="monospace"
/>
</LinearLayout>
```

Trong đoạn XML này chúng ta khai báo một Linear Layout với 2 thành phần con của nó là 1 Edit Text (dùng để gõ xâu ký tự) với 1 Text View (hiển thị xâu ký tự). Linear Layout được khai báo với từ khóa orientation nhằm chỉ ra chiều sắp xếp của 2 thành phần con là chiều dọc. Còn với layout\_width, layout\_height các bạn có thể cho giá trị bằng "fill\_parent" hoặc "wrap\_content" để thông báo thành phần này sẽ có chiều rộng (dài) phủ đầy thành phần cha hoặc chỉ vừa bao đủ nội dung. Trong Edit Text và Text View các bạn có thể thấy có từ khóa id, từ khóa này cho phép khai báo id của các thành phần để lấy về trong code (sẽ đề cập sau). Ngoài ra từ khóa hint trong Edit Text cho phép hiện ra phần nội dung mờ khi Edit Text vẫn chưa có ký tự nào. "@string/edit\_hint" thông báo lấy trong file strings.xml xâu có tên là edit\_hint. Còn textColor của Text View thì thông báo đoạn ký tự sẽ được hiển thị với màu lấy trong file colors.xml, textSize chỉ ra cỡ chữ bằng 28 pixel và typeface chỉ ra kiểu chữ là monospace

B5: Vẫn trong thư mục res, vào values và chọn file strings.xml. Bổ sung thêm dòng định nghĩa cho edit\_hint như sau:

Mã:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="hello">Hello World, Example!</string>
<string name="app_name">Example 1</string>
<string name="edit_hint">Enter the work here</string>
</resources>
```

B6: Trong thư mục values, tạo file colors.xml (chuột phải vào thư mục, chọn New -> Android XML File, và lưu ý chữ s, không phải là color.xml). Gõ nội dung cho file như sau:

Mã:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<color name="text_color">#ff3300</color>
</resources>
```

OK, vậy là bạn đã tạo một màu mới cho dòng chữ sẽ được hiển thị trong Text View (ff3300 là mã hexa của màu đỏ). Thực chất bạn hoàn toàn có thể gõ thẳng

Mã:

```
android:textColor="#ff3300"
```

trong file main.xml mà không cần tạo mới file colors.xml, nhưng mục đích của XML trong Android chính là để hỗ trợ nâng cấp chỉnh sửa dễ dàng. Nếu sau này bạn muốn sửa màu của dòng text thì chỉ cần vào colors.xml thay đổi thay vì mò mẫm trong main.xml (có thể rất dài nếu giao diện phức tạp).

Các thành phần trên mới chỉ là các phần cơ bản của XML. Ngoài ra các bạn có thể khai báo thêm về Animation, Style và Theme (phức tạp hơn nhiều nên mình không giới thiệu trong phần cơ bản này).

**BZ:** Vậy là chúng ta đã hoàn thiện phần giao diện với XML, giờ đến viết code để xử lý các sự kiện cho các thành phần:

⇒ vào thư mục src (source code của project) => at.exam => Example.java, gõ nội dung code sau vào:

⇒ Mã:

```
package at.exam;
import android.app.Activity;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.EditText;
import android.widget.TextView;
public class Example extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);

//Thiết lập giao diện lấy từ file main.xml
```

```

setContentView(R.layout.main);

//Lấy về các thành phần trong main.xml thông qua id
final EditText edit = (EditText) findViewById(R.id.edit_text);
final TextView text = (TextView) findViewById(R.id.text_view);
//Thiết lập xử lý cho sự kiện nhấn nút giữa của điện thoại
edit.setOnKeyListener(new OnKeyListener() {
    @Override
    public boolean onKey(View v, int keyCode, KeyEvent event) {
        if (event.getAction() == KeyEvent.ACTION_DOWN
            && keyCode == KeyEvent.KEYCODE_DPAD_CENTER) {
            text.setText(edit.getText().toString());
            edit.setText("");
            return true;
        }
        else {
            return false;
        }
    }
});
}
}
}

```

Đạo qua một chút kiến thức cơ bản: Trong Android, các lớp sử dụng để tạo giao diện (Edit Text, Text View...) đều là lớp con của lớp View. Một số lớp thường xuyên được sử dụng để tạo giao diện:

- TextView
- EditText
- ListView
- Spinner
- CheckBox
- Button
- RadioButton

Ngoài ra bạn còn có thể tạo 1 View riêng của mình bằng cách kế thừa View có sẵn.

Các Listener được sử dụng để bắt 1 sự kiện nào đó. Ở đây mình sử dụng OnKeyListener dùng để bắt sự kiện khi nhấn 1 phím của điện thoại. Ngoài ra thường sử dụng OnClickListener để bắt sự kiện chạm vào 1 View đang hiển thị trên màn hình. Mỗi View đều phải set Listener riêng để xử lý cho sự kiện tương tác với nó, và mỗi loại View cũng lại có những Listener dành riêng cho nó (VD: CheckBox có OnCheckedChangeListener)

Ở đây mình sử dụng hàm dạng inner để định nghĩa xử lý cho OnKeyListener nên có thể mọi người không quen lắm, nhưng nó cũng nằm trong phần cơ bản của Java đấy nhé.

Đề nghị lưu ý thêm phần R.id.edit\_text. Để lấy hoặc truy nhập các thành phần ta đã định nghĩa trong XML ta phải sử dụng R.\* như R.layout.main, R.id.edit\_text. Lệnh **findViewById** sẽ trả về 1 View có Id thiết lập trong phần XML. Do View là lớp cha của EditText với TextView nên ở đây ta phải ép kiểu.

Ngoài ra các string hay color cũng có thể lấy về bằng lệnh getResources() . Vd:



getResource().getColor(R.color.text\_color)

**BS:** Chạy chương trình. Chọn Run => Android Application và chờ cho emulator khởi động nhé. Ai có 1 Android thật có thể kết nối qua USB và thử nghiệm luôn. Tự chỉnh sửa trong code và trong XML để hiểu thêm về lập trình Android.

VD:

Mã:

```
edit.setOnClickListener(new OnClickListener() {  
    @Override  
    public void onClick(View v) {  
        // TODO Auto-generated method stub  
    }  
});
```

hoặc trong XML thêm vào phần Text View

Mã:

```
android:textSize="50px"
```

để xem chương trình thay đổi như thế nào nhé ^\_^

## Bài 2: Lập trình Android cơ bản.

Trong bài 1 mình đã giới thiệu sơ lược về các thành phần cơ bản của Android cũng như việc sử dụng XML để lập trình ứng dụng Android. Trong bài này mình sẽ giới thiệu thêm về Android Manifest và đi sâu hơn về vấn đề làm việc với View.

### Android Manifest

Trong khung Package Explorer, ở phía dưới thư mục res, bạn sẽ thấy 1 file có tên là AndroidManifest.xml. Mỗi ứng dụng đều cần có AndroidManifest.xml để mô tả những thông tin quan trọng của nó cho hệ thống Android biết. Let's look closer:

Mã:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="at.exam"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".Example"
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-sdk android:minSdkVersion="3" />
</manifest>
```

Cụ thể những công việc mà AndroidManifest.xml thực hiện:

- Đặt tên cho Java package của ứng dụng.
- Mô tả các thành phần (component) của ứng dụng: activity, service, broadcast receiver hoặc content provider.
- Thông báo những permission mà ứng dụng cần có để truy nhập các protected API và tương tác với các ứng dụng khác.
- Thông báo những permission mà các ứng dụng khác cần có để tương tác với ứng dụng hiện thời.

- Thông báo level thấp nhất của Android API mà ứng dụng cần để chạy. (Android 1.0 là level 1, 1.1 là level 2, 1.5 level 3, 1.6 level 4 và 2.0 là level 5).

...

Hãy xem thử file AndroidManifest.xml của chương trình ToDo mình đang xây dựng:

Mã:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="android.at"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity
android:name=".ToDo"
android:screenOrientation="landscape"
```

```

android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".WorkEnter">
</activity>
<receiver android:name=".AlarmReceiver">
</receiver>
</application>
<uses-sdk android:minSdkVersion="3" />
<uses-permission android:name="android.permission.VIBRATE"/>
</manifest>

```

Main Activity của chương trình Too Do này là activity TooDo. Ngoài ra mình còn có 1 Activity khác có tên là WorkEnter để cho phép nhập vào thời gian và nội dung công việc. 1 Broadcast Receiver có tên là AlarmReceiver để nhận alarm gửi tới trong intent. Khi alarm được nhận sẽ có âm thanh và rung (vibration). Tất cả công việc sẽ được viết trong code, nhưng bắt buộc bạn phải khai báo các thành phần có trong ứng dụng vào AndroidManifest nếu muốn chương trình hoạt động. Tương tự, set permission để truy cập camera, internet, đọc contact... cũng đều phải khai báo trong AM. Từ khóa **screenOrientation** cho phép thiết lập giao diện khi vào ứng dụng theo chiều dọc (portrait - mặc định) hay ngang (landscape), **theme** cho phép sử dụng style có sẵn của android là full-screen (ko có thanh status bar nữa). Intent filter là bộ lọc dùng để giới hạn các intent được sử dụng trong activity hay receiver...

Mã:

```

<intent-filter>
<action android:name="android.intent.action.VIEW"/>
<category android:name="android.intent.category.DEFAULT"/>
<category android:name="android.intent.category.BROWSABLE"/>
<data android:scheme="http" android:host="www.google.com"
android:path="/m/products/scan"/>
</intent-filter>

```

Bộ lọc trên chỉ cho phép intent mở internet với đường dẫn định nghĩa sẵn (<http://www.google.com/m/products/scan>)

Ok, hi vọng mọi người đã nắm được chức năng cơ bản cũng như cách sử dụng Android Manifest

### Working with View

Trong bài 1 mình đã giới thiệu qua cách sử dụng Edit Text và Text View. Thực chất các View còn lại cũng có cách sử dụng tương tự, bạn sẽ kết hợp nhiều View khác nhau để cho ra giao diện mình mong muốn. Ở đây mình sẽ đề cập nhiều tới List View (theo ý kiến mình là View khó sử dụng nhất).

Yêu cầu: Xây dựng một chương trình cho phép nhập nội dung công việc và thời gian rồi list ra

**B1:** Vẫn bắt đầu bằng cách khởi tạo một Project mới: File -> New -> Android Project.

Project name: Example 2

Build Target: Chọn Android 1.5  
Application name: Example 2  
Package name: at.exam  
Create Activity: Example  
=> Kích nút Finish.

**B2:** Đi tới res/main.xml để xây dựng giao diện cho chương trình:

Mã:

```
<?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"
    >
    <EditText
        android:id="@+id/work_enter"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:hint="@string/work_hint"
        android:lines="1"
        android:textSize="24px"
    />
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
    >
        <TextView
            android:layout_width="50px"
            android:layout_height="wrap_content"
            android:text="@string/hour_edit"
            android:typeface="normal"
            android:textSize="15px"
            android:textStyle="bold"
            android:padding="5px"
        />
        <EditText
            android:id="@+id/hour_edit"
            android:layout_width="45px"
            android:layout_height="wrap_content"
            android:hint="12"
            android:textColorHint="@color/hint_color"
            android:textSize="20px"
            android:gravity="center"
            android:padding="5px"
            android:numeric="integer"
            android:maxLength="2"
        />
    <TextView
        android:layout_width="65px"
        android:layout_height="wrap_content"
```

```

android:text="@string/minute_edit"
android:typeface="normal"
android:textSize="15px"
android:textStyle="bold"
android:padding="5px"
/>
<EditText
android:id="@+id/minute_edit"
android:layout_width="45px"
android:layout_height="wrap_content"
android:hint="00"
android:textColorHint="@color/hint_color"
android:textSize="20px"
android:gravity="center"
android:padding="5px"
android:numeric="integer"
android:maxLength="2"
/>
</LinearLayout>
<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:gravity="center"
android:text="@string/button_content"
/>
<ListView
android:id="@+id/list"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
</LinearLayout>

```

Giao diện ta thiết kế ở đây có 1 Linear Layout làm thành phần chính, các thành phần con của nó gồm 1 Edit Text (dùng để nhập nội dung công việc), 1 Linear Layout (lại gồm các thành phần con để nhập giờ và phút thực hiện công việc), 1 Button (để thêm nội dung công việc vào List View) và 1 List View dùng để list các công việc bạn đã nhập.

Từ khóa **lines** được dùng để cố định số dòng và nên sử dụng với Edit Text thay vì dùng mỗi wrap\_content vì nếu sd wrap\_content thì Edit Text sẽ tự giãn ra nếu dòng nhập vào vượt giới hạn đường bao (làm hỏng giao diện bạn thiết kế).

Từ khóa **gravity** thông báo các thành phần con sẽ được sắp xếp ntn ở thành phần cha. Ở đây mình dùng "center" nghĩa là thành phần con nằm ở trung tâm. Hãy thử thêm vào 1 Edit Text:

Mã:

```
android:gravity="center"
```

Bạn sẽ thấy dòng chữ nhập vào sẽ bắt đầu từ giữa của Edit Text chứ không bắt đầu từ bên trái như trước nữa.

Từ khóa **padding** dùng để cách 1 khoảng cách cho thành phần. Nếu không có padding thì 2 thành phần con thuộc cùng 1 LinearLayout sẽ được xếp sát nhau, nhưng nếu 1 thành phần con sử dụng padding thì sẽ tạo được khoảng cách với thành phần còn lại theo mong muốn. Ngoài ra còn

có `paddingLeft`, `paddingRight`, `paddingTop`, `paddingBottom`.

Từ khóa `numeric` dùng để giới hạn dạng ký tự nhập vào. Ở đây mình muốn chỉ nhập vào chữ số nên dùng `"integer"`

Từ khóa `maxLength` dùng để giới hạn số ký tự nhập vào. Do Edit Text này dùng để nhập giờ nên `maxLength="2"`.

Ok, giờ đến 1 chút kiến thức về các đơn vị của dimension:

- px (pixel): điểm chấm trên màn hình.
- in (inch)
- mm (milimet)
- pt (point) = 1/72 m
- dp (density - independent pixel): cái này hơi khó giải thích. Nói chung dp được sử dụng cho nhiều độ phân giải, và với độ phân giải 160 px/inch thì 1 dp = 1 px.
- sp: gần giống dp, nên sử dụng cho text size.

Nói chung nên sử dụng dp và sp để định nghĩa size cho các thành phần, vì nó có tỉ lệ cố định với độ phân giải của màn hình. Còn nếu bạn chủ tâm xây dựng cho 1 độ phân giải nhất định thì dùng px cho chính xác và chắc chắn.

**B3:** Tới values/strings.xml chỉnh sửa như sau:

Mã:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">Example 2</string>
<string name="work_hint">Enter the work here</string>
<string name="hour_edit">Hour</string>
<string name="minute_edit">Minute</string>
<string name="button_content">Add work</string>
</resources>
```

**B4:** Tạo mới colors.xml trong values với nội dung:

Mã:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<color name="hint_color">#cccccc</color>
</resources>
```

OK, vậy là đã hoàn thiện phần giao diện. Các bạn có thể cho chạy thử ngay để kiểm tra xem giao diện đã như ý muốn chưa chứ không cần đợi hoàn thành cả code (Run as -> Android Application).

**B5:** Time to coding. Tới thư mục src/Example.java và thay đổi nội dung file như sau:

Mã:

```
package at.exam;

import java.util.ArrayList;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
```

```

import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;

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

        //Tạo mảng để chứa String nội dung công việc và giờ
        final ArrayList<String> arrayWork = new ArrayList<String>();
        //Adapter dùng để kết nối mảng với List View
        final ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(this,
            android.R.layout.simple_list_item_1, arrayWork);

        //Các EditText để vào nội dung công việc được lấy về từ XML
        final EditText workEnter = (EditText) findViewById(R.id.work_enter);
        final EditText hourEdit = (EditText) findViewById(R.id.hour_edit);
        final EditText minuteEdit = (EditText) findViewById(R.id.minute_edit);

        //Button khi nhấn sẽ thêm công việc vào ListView
        final Button button = (Button) findViewById(R.id.button);

        //ListView chứa danh sách công việc
        final ListView list = (ListView) findViewById(R.id.list);
        //Cần set Adapter cho list để biết sẽ lấy nội dung từ mảng arrayWork
        list.setAdapter(arrayAdapter);

        //Định nghĩa Listener xử lý sự kiện nhấn vào button
        OnClickListener add = new OnClickListener() {
            @Override
            public void onClick(View v) {
                //Nếu 1 trong 3 Edit Text không có nội dung thì hiện lên thông báo
                if (workEnter.getText().toString().equals("") ||
                    hourEdit.getText().toString().equals("") ||
                    minuteEdit.getText().toString().equals("")) {
                    AlertDialog.Builder builder = new AlertDialog.Builder(Example.this);
                    builder.setTitle("Info missing");
                    builder.setMessage("Please enter all information of the work");
                    builder.setPositiveButton("Continue", new DialogInterface.OnClickListener() {
                        public void onClick(DialogInterface dialog, int which) {
                            // TODO Auto-generated method stub
                        }
                    });
                    builder.show();
                }
                //Lấy nội dung công việc và thời gian ra từ Edit Text và đưa vào list
                else {
                    String str = workEnter.getText().toString() + " - "

```



```

+ hourEdit.getText().toString() + ":"
+ minuteEdit.getText().toString();
arrayWork.add(0,str);
arrayAdapter.notifyDataSetChanged();
workEnter.setText("");
hourEdit.setText("");
minuteEdit.setText("");
}
}

};

//set Listener cho button
button.setOnClickListener(add);
}
}

```

Mình đã chú thích đầy đủ và đoạn code cũng khá dễ hiểu. Tuy nhiên cần lưu ý 2 vấn đề ở đây.

- Khởi tạo đối tượng ArrayAdapter: Các bạn thấy đối số truyền vào là **(this, android.R.layout.simple\_list\_item\_1, arrayWork)**. This là đối số của lớp Context (ở đây chính là activity Example). Bạn sẽ gặp Context trong rất nhiều khởi tạo các lớp và nên hiểu Context có ý nghĩa gì. Mình xin đưa ra giải thích của anh Giáp (thank mr giaplv):

Quote:

**Context thuộc android.content (android.content.Context).**

Là một Interface (lớp giao tiếp) chứa hầu hết thông tin về môi trường ứng dụng của android, có nghĩa là mọi thao tác, tương tác với hệ điều hành đều phải qua lớp này. Nó là một lớp abstract (trừu tượng) cung cấp cho những lớp khác các phương thức để tương tác với hệ thống Android.

Nó cho phép truy cập tới các nguồn tài nguyên (resources) đã được định nghĩa và các lớp khác. Ví dụ như nó có thể khởi tạo và chạy các activities, các broadcast và các intents,... Chúng ta coi như Context là một lớp ở mức ứng dụng (Application level- liên quan tới hệ thống).

Tóm lại context giúp chúng ta dễ dàng truy cập và tương tác tới các tài nguyên của hệ thống, các thông tin, các dịch vụ (services), các thông số cấu hình, database, wallpaper, danh bạ, cuộc gọi, kết nối, chế độ rung (vibrator),...

\*\*\*sở dĩ hầu hết các lớp có liên quan tới UI (layout, button, textview, imageview, listview,...) đều phải super tới Context vì bản thân nó đảm nhiệm việc truy cập resource (R.id, R.layout,...). Nếu chúng ta không tham chiếu tới Context class thì đương nhiên không thể dùng tới các resources mà chúng ta đã tạo ra. Tiếp theo là android.R.layout.simple\_list\_item\_1, đối này định nghĩa cách thể hiện item (ở đây là String) trong List View. Các bạn hãy ghi nhớ android.R.\* là các tài nguyên (resource) có sẵn của Android cho phép bạn truy cập và sử dụng. Sau này khi hướng dẫn tạo custom View cho List View mình sẽ đề cập lại vấn đề này. Cuối cùng arrayWork chính là mảng cần được bind của adapter.

- AlertDialog là lớp cho phép đưa ra 1 hộp thoại, thường dùng để đưa ra thông tin hoặc cảnh báo đơn giản. Trong code mình tạo 1 builder, tạo tiêu đề (title) cho nó, đưa ra thông báo (message) và cuối cùng là tạo 1 positive button (nhưng không



định nghĩa xử lý khi nhấn nút này, vì vậy nếu bạn nhấn nút thì dialog sẽ chỉ đơn giản thực hiện việc đóng lại).

**B6:** Tiến hành chạy thử chương trình. Run as -> Android Application. Enjoy yourself.

### Bài 3 - Lập trình Android cơ bản: ViewGroup và Custom Adapter

Trong bài này mình sẽ hướng dẫn cách tạo 1 custom ViewGroup, sử dụng ViewGroup này vào ListView, và cuối cùng là tạo 1 Option Menu. Đây cũng sẽ là bài cuối cùng mình viết về làm việc với View, các bài sau sẽ chuyển qua Intent và BroadCast Receiver.

#### Custom ViewGroup

ViewGroup thông thường chúng ta hay gặp là LinearLayout, Relative Layout. Xây dựng custom ViewGroup cho phép chúng ta tạo 1 tập các widget được sắp xếp theo ý muốn rồi đưa vào sử dụng.

Yêu cầu: Xây dựng ứng dụng dạng To Do List: Cho phép nhập vào nội dung công việc và thời gian thực hiện công việc rồi đưa vào list công việc. Cho phép xóa các công việc khỏi list.

**B1:** Khởi tạo project: File -> New -> Android Project

Project name: Example 3

Build Target: Chọn Android 1.5

Application name: Example 3

Package name: at.exam

Create Activity: Example

=> Kích nút Finish.

**B2:** Xây dựng custom view group trong XML. Đi tới res/layout tạo 1 file XML mới là list.xml. Gõ nội dung sau vào:

Mã:

```
<?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"
android:orientation="horizontal">
<CheckBox
android:id="@+id/check_work"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text=""
android:paddingTop="45px"
android:paddingRight="10px"
/>
<LinearLayout
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="vertical"
>
<TextView
android:id="@+id/work_content"
android:textSize="24px"
android:layout_width="wrap_content"
```

```

android:layout_height="wrap_content"
android:lines="1"
android:textColor="@color/work_color"
/>
<TextView
android:id="@+id/time_content"
android:textSize="16px"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:lines="1"
android:textColor="@color/time_color"
/>
</LinearLayout>
</LinearLayout>

```

Custom ViewGroup của chúng ta ở đây khá đơn giản, đó là 1 LinearLayout chứa 2 thành phần: 1 CheckBox và 1 LinearLayout khác gồm 2 TextView để hiển thị nội dung công việc và thời gian.

**B3:** Đã xong giao diện cho custom ViewGroup, chúng ta sẽ thiết kế giao diện cho chương trình trong main.xml. Ở đây mình dùng lại giao diện của Example 2 trong bài 2.

Mã:

```

<?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"
>
<EditText
android:id="@+id/work_enter"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:hint="@string/work_hint"
android:lines="1"
android:textSize="24px"
/>
<LinearLayout
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:orientation="horizontal"
>
<TextView
android:layout_width="50px"
android:layout_height="wrap_content"
android:text="@string/hour_edit"
android:typeface="normal"
android:textSize="15px"
android:textStyle="bold"
android:padding="5px"
/>

```

```
<EditText
android:id="@+id/hour_edit"
android:layout_width="45px"
android:layout_height="wrap_content"
android:hint="12"
android:textColorHint="@color/hint_color"
android:textSize="20px"
android:gravity="center"
android:padding="5px"
android:numeric="integer"
android:maxLength="2"
/>
<TextView
android:layout_width="65px"
android:layout_height="wrap_content"
android:text="@string/minute_edit"
android:typeface="normal"
android:textSize="15px"
android:textStyle="bold"
android:padding="5px"
/>
<EditText
android:id="@+id/minute_edit"
android:layout_width="45px"
android:layout_height="wrap_content"
android:hint="00"
android:textColorHint="@color/hint_color"
android:textSize="20px"
android:gravity="center"
android:padding="5px"
android:numeric="integer"
android:maxLength="2"
/>
</LinearLayout>
<Button
android:id="@+id/button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:gravity="center"
android:text="@string/button_content"
/>
<ListView
android:id="@+id/list"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
/>
</LinearLayout>
```

**B4:** Tạo file colors.xml trong resvalue:

Mã:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<color name="work_color">#ffffff</color>
<color name="time_color">#cccccc</color>
<color name="hint_color">#cccccc</color>
</resources>
```

**work\_color** là màu của nội dung công việc trong list. **time\_color** màu của thời gian công việc. **hint\_color** màu của text hint (dòng hướng dẫn) các EditText.

**B5:** Chỉnh sửa file strings.xml trong resvalue:

Mã:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">Example 3</string>
<string name="work_hint">Enter the work here</string>
<string name="hour_edit">Hour</string>
<string name="minute_edit">Minute</string>
<string name="button_content">Add work</string>
</resources>
```

**B6:** Time to coding. Đi tới srcat.exam tạo một class mới là CustomViewGroup với nội dung sau:

Mã:

```
package at.exam;

import android.content.Context;
import android.view.LayoutInflater;
import android.widget.CheckBox;
import android.widget.LinearLayout;
import android.widget.TextView;

public class CustomViewGroup extends LinearLayout {
    public CheckBox cb;
    public TextView workContent;
    public TextView timeContent;
    public CustomViewGroup(Context context) {
        super(context);

        //Sử dụng LayoutInflater để gán giao diện trong list.xml cho class này
        LayoutInflater li = (LayoutInflater) this.getContext()
            .getSystemService(Context.LAYOUT_INFLATER_SERVICE) ;
        li.inflate(R.layout.list, this, true);

        //Lấy về các View qua Id
        cb = (CheckBox) findViewById(R.id.check_work);
        workContent = (TextView) findViewById(R.id.work_content);
        timeContent = (TextView) findViewById(R.id.time_content);
    }
}
```

Đoạn code trên giúp ta định nghĩa giao diện của custom ViewGroup mới dựa trên file list.xml. Mọi người cũng có thể tạo giao diện bằng code, ko cần sử dụng XML nhưng sẽ phức tạp hơn và mình cũng ko giới thiệu ở đây.

**BZ:** Tạo 1 class Work cũng trong at.exam để thể hiện công việc:

Mã:

```
package at.exam;

public class Work {
    private String workContent;
    private String timeContent;
    private boolean isChecked;

    public Work(String workContent, String timeContent) {
        this.workContent = workContent;
        this.timeContent = timeContent;
        isChecked = false;
    }

    public String getContent() {
        return workContent;
    }

    public String getTime() {
        return timeContent;
    }

    public void setChecked(boolean isChecked) {
        this.isChecked = isChecked;
    }

    public boolean isChecked() {
        return isChecked;
    }
}
```

Code rất đơn giản nên mình sẽ không chú thích nữa.

**B8:** Chúng ta đã tạo xong custom ViewGroup, bây giờ chính là lúc sử dụng. Tạo 1 class mới tên là ListWorkAdapter trong at.exam:

Mã:

```
package at.exam;

import java.util.ArrayList;
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.CheckBox;
import android.widget.CompoundButton;
```

```

import android.widget.TextView;
import android.widget.CompoundButton.OnCheckedChangeListener;

public class ListWorkAdapter extends ArrayAdapter<Work>{
    ArrayList<Work> array;
    int resource;
    Context context;

    public ListWorkAdapter(Context context, int textViewResourceId,
        ArrayList<Work> objects) {
        super(context, textViewResourceId, objects);
        this.context = context;
        resource = textViewResourceId;
        array = objects;
    }

    //Phương thức xác định View mà Adapter hiển thị, ở đây chính là CustomViewGroup
    //Bắt buộc phải Override khi kế thừa từ ArrayAdapter
    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View workView = convertView;

        if (workView == null) {
            workView = new CustomViewGroup(getContext());
        }

        //Lấy về đối tượng Work hiện tại
        final Work work = array.get(position);

        if (work != null) {
            TextView workContent = ((CustomViewGroup) workView).workContent;
            TextView timeContent = ((CustomViewGroup) workView).timeContent;
            CheckBox checkWork = ((CustomViewGroup) workView).cb;

            //Set sự kiện khi đánh dấu vào checkbox trên list
            checkWork.setOnCheckedChangeListener(new OnCheckedChangeListener() {

                @Override
                public void onCheckedChanged(CompoundButton buttonView,
                    boolean isChecked) {
                    work.setChecked(isChecked);
                }
            });

            //Lấy về nội dung cho TextView và CheckBox dựa vào đối tượng Work hiện tại
            workContent.setText(work.getContent());
            timeContent.setText(work.getTime());
            checkWork.setChecked(work.isChecked());
        }
        return workView;
    }

```

```
}
}
```

ListWorkAdapter sẽ được sử dụng thay thế cho ArrayAdapter được bind với ListView. Thông thường ArrayAdapter chỉ cho hiển thị String bằng TextView, nhưng với việc kế thừa và override phương thức getView, ta có thể định nghĩa lại hiển thị cho các thành phần của ListView.

**B9:** Việc cuối cùng cần làm là viết lại Activity. Tới Example.java và chỉnh sửa theo nội dung sau:

Mã:

```
package at.exam;

import java.util.ArrayList;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.AdapterView;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ListView;

public class Example extends Activity {
    //Các hằng dùng cho tạo Option Menu
    private static final int DELETE_WORK = Menu.FIRST;
    private static final int ABOUT = Menu.FIRST + 2;

    ArrayList<Work> array;
    ListWorkAdapter arrayAdapter;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        array = new ArrayList<Work>();
        arrayAdapter = new ListWorkAdapter(this,
            R.layout.list, array);

        final EditText workEnter = (EditText) findViewById(R.id.work_enter);
        final EditText hourEdit = (EditText) findViewById(R.id.hour_edit);
        final EditText minuteEdit = (EditText) findViewById(R.id.minute_edit);

        final Button button = (Button) findViewById(R.id.button);

        //Tạo list view cho danh sách công việc
```



```

final ListView list = (ListView) findViewById(R.id.list);
list.setAdapter(arrayAdapter);

OnClickListener add = new OnClickListener() {
    @Override
    public void onClick(View v) {
        if (workEnter.getText().toString().equals("") ||
            hourEdit.getText().toString().equals("") ||
            minuteEdit.getText().toString().equals("")) {
            AlertDialog.Builder builder = new AlertDialog.Builder(Example.this);
            builder.setTitle("Info missing");
            builder.setMessage("Please enter all information of the work");
            builder.setPositiveButton("Continue", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                    // TODO Auto-generated method stub
                }
            });
            builder.show();
        }
        else {
            String workContent = workEnter.getText().toString();
            String timeContent = hourEdit.getText().toString() + ":"
                + minuteEdit.getText().toString();
            Work work = new Work(workContent, timeContent);
            array.add(0, work);
            arrayAdapter.notifyDataSetChanged();
            workEnter.setText("");
            hourEdit.setText("");
            minuteEdit.setText("");
        }
    }
};

button.setOnClickListener(add);
}

//Tao Option Menu
public boolean onCreateOptionsMenu(Menu menu) {
    super.onCreateOptionsMenu(menu);
    menu.add(0, DELETE_WORK, 0, "Delete" ).setIcon(android.R.drawable.ic_delete);
    menu.add(0, ABOUT, 0, "About" ).setIcon(android.R.drawable.ic_menu_info_details)
    ;
    return true;
}

//Xử lý sự kiện khi các option trong Option Menu được lựa chọn
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case DELETE_WORK: {

```



trong list, rồi nhấn Menu và chọn delete thì thành phần sẽ bị gỡ bỏ khỏi danh sách.

## Bài 4: Intent và Broadcast Receiver

### Khái niệm về Intent:

Theo định nghĩa của Google, Intent là một miêu tả về một hoạt động cần được thực hiện. Còn nói một cách đơn giản và dễ hiểu hơn, Intent là một cơ cấu cho phép truyền thông điệp giữa các thành phần của 1 ứng dụng và giữa các ứng dụng với nhau.

### Các thuộc tính của Intent:

- **action**: là hành động được thực hiện, vd : ACTION\_VIEW, ACTION\_MAIN
- **data**: là dữ liệu sẽ được xử lý trong action, thường được diễn tả là một Uri (Uniform Resource Identifier, tham khảo [http://en.wikipedia.org/wiki/Uniform...rce\\_Identifier](http://en.wikipedia.org/wiki/Uniform_Resource_Identifier) để hiểu rõ thêm chi tiết).

VD:

**ACTION\_VIEW content://contacts/people/1** - Hiển thị thông tin về người với mã danh 1

**ACTION\_DIAL content://contacts/people/1** - Hiển thị màn hình gọi đến người với mã danh 1

**ACTION\_DIAL tel:123** - Hiển thị màn hình gọi với số gọi là 123

Ngoài ra còn có 1 số thuộc tính mà ta có thể bổ sung vào Intent:

- **category**: bổ sung thêm thông tin cho action của Intent. VD: CATEGORY\_LAUNCHER thông báo sẽ thêm vào Launcher như là một ứng dụng top-level

- **type**: chỉ rõ kiểu của data

- **component**: chỉ rõ thành phần sẽ nhận và xử lý intent. Khi thuộc tính này được xác định thì các thuộc tính khác sẽ trở thành thuộc tính phụ.

- **extras**: mang theo đối tượng Bundle chứa các giá trị bổ sung.

VD:

**ACTION\_MAIN** và **CATEGORY\_HOME**: trở về màn hình Home của Android (khi bấm nút Home của di động)

### Phân loại Intent:

Intent được chia làm 2 loại:

- **Explicit Intents**: intent đã được xác định thuộc tính component, nghĩa là đã chỉ rõ thành phần sẽ nhận và xử lý intent. Thông thường intent dạng này sẽ không bổ sung thêm các thuộc tính khác như action, data. Explicit Intent thường được sử dụng để khởi chạy các activity trong cùng 1 ứng dụng.

- **Implicit Intents**: Intent không chỉ rõ component xử lý, thay vào đó nó bổ sung thông tin trong các thuộc tính. Khi intent được gửi đi, hệ thống sẽ dựa vào những thông tin này để quyết định component nào thích hợp nhất để xử lý nó.

VD:

**ACTION\_DIAL tel:123** thông thường sẽ được hệ thống giao cho activity Phone Dialer mặc định của Android xử lý.

### Một số action thường sử dụng trong Intent:

**ACTION\_ANSWER** - mở Activity để xử lý cuộc gọi tới, thường là Phone Dialer của Android

**ACTION\_CALL** - mở 1 Phone Dialer (mặc định là PD của Android) và ngay lập tức thực hiện cuộc gọi dựa vào thông tin trong data URI

**ACTION\_DELETE** - mở Activity cho phép xóa dữ liệu mà địa chỉ của nó chứa trong data URI

**ACTION\_DIAL** - mở 1 Phone Dialer (mặc định là PD của Android) và điền thông tin lấy từ địa chỉ chứa trong data URI

**ACTION\_EDIT** - mở 1 Activity cho phép chỉnh sửa dữ liệu mà địa chỉ lấy từ data URI

**ACTION\_SEND** - mở 1 Activity cho phép gửi dữ liệu lấy từ data URI, kiểu của dữ liệu xác định trong thuộc tính type

**ACTION\_SENDTO** - mở 1 Activity cho phép gửi thông điệp tới địa chỉ lấy từ data URI

**ACTION\_VIEW** - action thông dụng nhất, khởi chạy activity thích hợp để hiển thị dữ liệu trong data URI

**ACTION\_MAIN** - sử dụng để khởi chạy 1 Activity

OK, lý thuyết như thế là đã tạm ổn. Giờ chúng ta sẽ chuyển qua phần thực hành để hiểu rõ cách sử dụng Intent. Như đã nêu ở trên, Intent chia làm 2 loại: explicit intent và implicit intent. Mỗi loại Intent sẽ có cách cài đặt và sử dụng khác nhau.

## Using Explicit Intents

**Yêu cầu:** Xây dựng chương trình gồm 2 Activity. Activity1 là Activity chạy ban đầu lúc khởi động ứng dụng, cho phép nhập vào 1 giá trị, cho phép khởi chạy Activity2 và gửi giá trị này tới Activity2. Activity2 sẽ nhận và hiển thị giá trị, rồi lại gửi giá trị này tới 1 BroadcastReceiver. Cơ chế gửi và khởi chạy Activity sử dụng thông qua Intent.

<b>B1:</b>	Khởi tạo project:	File -> New -> Android Project
Project name:	Explicit Intent	Example
Build Target:	Chọn Android	1.5
Application name:	Explicit Intent	Example
Package name:		at.exam
Create Activity:		Activity1
=>	Kích nút	Finish.

**B2:** Tạo giao diện cho Activity1 -> res/layout/main.xml chuyển tên thành activity1\_layout.xml

Mã:

```
<?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="Activity 1 - Send value"
  android:typeface="normal"
  android:textSize="14px"
  android:textStyle="bold"
  android:textColor="#cccccc"
  android:background="#333333"
/>
<EditText
  android:id="@+id/value_edit"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:textSize="20px"
  android:gravity="center"
  android:lines="1"
  android:numeric="integer"
/>
<RelativeLayout
  android:layout_width="fill_parent"
  android:layout_height="fill_parent">
  <Button
    android:id="@+id/send_button"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="Send to Activity 2"
    android:layout_alignParentBottom="true"
  />
</RelativeLayout>
</LinearLayout>

```

Layout cho Activity1 bao gồm 1 LinearLayout chứa 1 TextView, 1 EditText để nhập giá trị (đã giới hạn kiểu nhập là number), và 1 RelativeLayout có 1 Button để khởi chạy Activity2. Mình sử dụng RelativeLayout để có thể xếp Button này xuống phía cuối của giao diện.

**B3:** Tạo giao diện cho Activity2 -> Chuột phải vào folder reslayout -> New -> Android XML File -> Gõ tên là activity2\_layout.xml  
Mã:

```

<?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="Activity 2 - Receive value"

```

```

    android:typeface="normal"
    android:textSize="14px"
    android:textStyle="bold"
    android:textColor="#cccccc"
    android:background="#333333"
  />
  <EditText
    android:id="@+id/value_receive"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textSize="20px"
    android:gravity="center"
    android:lines="1"
    android:numeric="integer"
    android:enabled="false"
  />
  <RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <Button
      android:id="@+id/call_button"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:text="Call Broadcast Receiver"
      android:layout_alignParentBottom="true"
    />
  </RelativeLayout>
</LinearLayout>

```

Layout của Activity2 tương tự như Activity1, nhưng Button bây giờ là để gọi Broadcast Receiver. Ngoài ra mình dùng EditText để hiển thị value nhận được (do nó có cái đường bao ngoài đẹp hơn TextView ^^) nên không cho phép nhập giá trị vào EditText này

Mã:

```

    android:enabled="false"

```

**B4:** Sửa lại nội dung của Activity1.java như sau:

Mã:

```

package at.exam;

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

public class Activity1 extends Activity {
  /** Called when the activity is first created. */

```

```

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity1_layout);

    final EditText editValue = (EditText) findViewById(R.id.value_edit);
    final Button sendButton = (Button) findViewById(R.id.send_button);

    sendButton.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
            String valueString = editValue.getText().toString();
            long value;
            if (valueString != null) {
                value = Long.parseLong(valueString);
            }
            else {
                value = 0;
            }

            //Tạo 1 đối tượng Bundle để gửi đi cùng Intent
            Bundle sendBundle = new Bundle();
            sendBundle.putLong("value", value);

            //Tạo Intent để khởi chạy Activity2 và gắn sendBundble vào Intent
            Intent i = new Intent(Activity1.this, Activity2.class);
            i.putExtras(sendBundle);
            startActivity(i);

            //Giải phóng Activity1 khỏi Activity Stack vì ta sẽ ko quay lại nó nữa
            finish();
        }
    });
}
}
}
}

```

**B5:** Tạo mới 1 Class Activity2.java trong package at.exam -> chỉnh sửa nội dung:

Mã:

```

package at.exam;

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

public class Activity2 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {

```



```

super.onCreate(savedInstanceState);
setContentView(R.layout.activity2_layout);

final EditText receiveValueEdit = (EditText) findViewById(R.id.value_receive);
final Button callReceiverButton = (Button) findViewById(R.id.call_button);

//Lấy về Bundle được gửi kèm Intent rồi lấy ra giá trị
Bundle receiveBundle = this.getIntent().getExtras();
final long receiveValue = receiveBundle.getLong("value");

receiveValueEdit.setText(String.valueOf(receiveValue));

callReceiverButton.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
//Khởi tạo 1 Intent để gửi tới BroadCast Receiver
//Gắn giá trị vào Intent, lần này ko cần Bundle nữa
Intent i = new Intent(Activity2.this, Receiver.class);
i.putExtra("new value", receiveValue - 10);
sendBroadcast(i);
}
});
}
}

```

**B6:** Tạo BroadCast Receiver để nhận Intent mà Activity2 gửi tới -> Tạo 1 file Receiver.java trong at.exam -> Nội dung:  
Mã:

```

package at.exam;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class Receiver extends BroadcastReceiver{

@Override
public void onReceive(Context context, Intent intent) {
long value = intent.getLongExtra("new value", -10) + 10;
Toast toast = Toast.makeText(context, "Broadcast Receiver catch an Intent"
+ "
" + "The value is stored in the Intent is "
+ String.valueOf(value), Toast.LENGTH_LONG);
toast.show();
}

}
}

```

Code không hề khó hiểu, và mình cũng đã add comment. Chỉ cần lưu ý ở đây là Toast là lớp để hiển thị một thông báo đơn giản trong 1 khoảng thời gian cố định, và ko thể thay đổi thời gian này T\_T (why???) chỉ có thể chọn giữa LENGTH\_SHORT với LENGTH\_LONG

**B7:** Bổ sung thêm thông tin về component mới vào AndroidManifest.xml:

Mã:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="at.exam"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".Activity1"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
</activity>
<activity android:name=".Activity2"></activity>
<receiver android:name=".Receiver"></receiver>
</application>
<uses-sdk android:minSdkVersion="3" />
</manifest>
```

**B4:** Sửa lại nội dung của Activity1.java như sau:

Mã:

```
package at.exam;

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

public class Activity1 extends Activity {
/** Called when the activity is first created. */
@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);
setContentView(R.layout.activity1_layout);

final EditText editValue = (EditText) findViewById(R.id.value_edit);
final Button sendButton = (Button) findViewById(R.id.send_button);

sendButton.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
String valueString = editValue.getText().toString();
long value;
if (valueString != null) {
value = Long.parseLong(valueString);
}
}
```

```

else {
    value = 0;
}

//Tạo 1 đối tượng Bundle để gửi đi cùng Intent
Bundle sendBundle = new Bundle();
sendBundle.putLong("value", value);

//Tạo Intent để khởi chạy Activity2 và gắn sendBundle vào Intent
Intent i = new Intent(Activity1.this, Activity2.class);
i.putExtras(sendBundle);
startActivity(i);

//Giải phóng Activity1 khỏi Activity Stack vì ta sẽ ko quay lại nó nữa
finish();
}
});
}
}
}

```

**B5:** Tạo mới 1 Class Activity2.java trong package at.exam -> chỉnh sửa nội dung:  
Mã:

```

package at.exam;

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

public class Activity2 extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity2_layout);

        final EditText receiveValueEdit = (EditText) findViewById(R.id.value_receive);
        final Button callReceiverButton = (Button) findViewById(R.id.call_button);

        //Lấy về Bundle được gửi kèm Intent rồi lấy ra giá trị
        Bundle receiveBundle = this getIntent().getExtras();
        final long receiveValue = receiveBundle.getLong("value");

        receiveValueEdit.setText(String.valueOf(receiveValue));

        callReceiverButton.setOnClickListener(new OnClickListener() {
            public void onClick(View v) {
                //Khởi tạo 1 Intent để gửi tới BroadCast Receiver
            }
        });
    }
}

```

```
//Gắn giá trị vào Intent, lần này ko cần Bundle nữa
Intent i = new Intent(Activity2.this, Receiver.class);
i.putExtra("new value", receiveValue - 10);
sendBroadcast(i);
}
});
}
}
```

**B6:** Tạo Broadcast Receiver để nhận Intent mà Activity2 gửi tới -> Tạo 1 file Receiver.java trong at.exam -> Nội dung:

Mã:

```
package at.exam;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.widget.Toast;

public class Receiver extends BroadcastReceiver{

    @Override
    public void onReceive(Context context, Intent intent) {
        long value = intent.getLongExtra("new value", -10) + 10;
        Toast toast = Toast.makeText(context, "Broadcast Receiver catch an Intent"
        + "
        " + "The value is stored in the Intent is "
        + String.valueOf(value), Toast.LENGTH_LONG);
        toast.show();
    }
}
}
```

Code không hề khó hiểu, và mình cũng đã add comment. Chỉ cần lưu ý ở đây là Toast là lớp để hiển thị một thông báo đơn giản trong 1 khoảng thời gian cố định, và ko thể thay đổi thời gian này T\_T (why???) chỉ có thể chọn giữa LENGTH\_SHORT với LENGTH\_LONG

**B7:** Bổ sung thêm thông tin về component mới vào AndroidManifest.xml:

Mã:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="at.exam"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".Activity1"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

```

</activity>
<activity android:name=".Activity2"></activity>
<receiver android:name=".Receiver"></receiver>
</application>
<uses-sdk android:minSdkVersion="3" />
</manifest>

```

## Using Implicit Intent:

**Yêu cầu:** Xây dựng chương trình nhập số và gọi. Lưu ý chương trình của mình ở đây chỉ xây dựng đến mức khi nhấn nút Call của di động thì sẽ chạy ứng dụng và hiển thị giao diện cho phép nhập số. Phần gọi dành cho ai yêu thích tìm hiểu thêm ^\_^ Phần này không hề khó nhưng ở đây mình chỉ muốn minh họa Implicit Intent nên sẽ không đưa vào.

**B1:** Khởi tạo project: File -> New -> Android Project  
 Project name: Implicit Intent Example  
 Build Target: Chọn Android 1.5  
 Application name: Implicit Intent Example  
 Package name: at.exam  
 Create Activity: Example  
 => Kích nút Finish.

**B2:** Đây là bước quan trọng nhất và cũng là bước có ý nghĩa duy nhất trong cả project này, các bước còn lại chỉ là bước râu ria mà mình thêm vào cho cái project nó ra hồn 1 chút. Bước này sẽ thêm 1 bộ lọc Intent Filter vào cho activity Example của chúng ta để bắt sự kiện nhấn nút Call của di động  
 -> Vào AndroidManifest.xml chỉnh sửa như sau:

Mã:

```

<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
package="at.exam"
android:versionCode="1"
android:versionName="1.0">
<application android:icon="@drawable/icon" android:label="@string/app_name">
<activity android:name=".Example"
android:label="@string/app_name">
<intent-filter>
<action android:name="android.intent.action.MAIN" />
<category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
<intent-filter>
<action android:name="android.intent.action.CALL_BUTTON" />
<category android:name="android.intent.category.DEFAULT" />
</intent-filter>
</activity>
</application>
<uses-sdk android:minSdkVersion="3" />

```

```
</manifest>
```

Thực chất chỉ là bổ sung thêm dòng chữ đỏ mình đánh dấu thôi ^\_^

**B3:** Xây dựng giao diện trong main.xml, bước này ko quan trọng, chỉ là râu ria cho activity có cái giao diện:

Mã:

```
<?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"
    >
    <TextView
        android:paddingTop="10px"
        android:id="@+id/number_display"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="30px"
        android:gravity="center"
        android:lines="2"
        android:background="#ffffff"
        android:textColor="#000000"
    />
    <TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
    >
    <TableRow
        android:gravity="center"
        android:paddingTop="30px"
    >
    <Button
        android:id="@+id/button1"
        android:layout_width="80px"
        android:layout_height="80px"
        android:gravity="center"
        android:text="1"
        android:textSize="25px"
    />
    <Button
        android:id="@+id/button2"
        android:layout_width="80px"
        android:layout_height="80px"
        android:gravity="center"
        android:text="2"
        android:textSize="25px"
    />
    <Button
```

```
android:id="@+id/button3"
android:layout_width="80px"
android:layout_height="80px"
android:gravity="center"
android:text="3"
android:textSize="25px"
/>
</TableRow>
<TableRow
android:gravity="center"
>
<Button
android:id="@+id/button4"
android:layout_width="80px"
android:layout_height="80px"
android:gravity="center"
android:text="4"
android:textSize="25px"

/>
<Button
android:id="@+id/button5"
android:layout_width="80px"
android:layout_height="80px"
android:gravity="center"
android:text="5"
android:textSize="25px"
/>
<Button
android:id="@+id/button6"
android:layout_width="80px"
android:layout_height="80px"
android:gravity="center"
android:text="6"
android:textSize="25px"
/>
</TableRow>
<TableRow
android:gravity="center"
>
<Button
android:id="@+id/button7"
android:layout_width="80px"
android:layout_height="80px"
android:gravity="center"
android:text="7"
android:textSize="25px"

/>
<Button
```

```
android:id="@+id/button8"
android:layout_width="80px"
android:layout_height="80px"
android:gravity="center"
android:text="8"
android:textSize="25px"
/>
<Button
android:id="@+id/button9"
android:layout_width="80px"
android:layout_height="80px"
android:gravity="center"
android:text="9"
android:textSize="25px"
/>
</TableRow>
<TableRow
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:gravity="center"
>
<Button
android:id="@+id/button_star"
android:layout_width="80px"
android:layout_height="80px"
android:gravity="center"
android:text="*"
android:textSize="25px"
/>
<Button
android:id="@+id/button0"
android:layout_width="80px"
android:layout_height="80px"
android:gravity="center"
android:text="0"
android:textSize="25px"
/>
<Button
android:id="@+id/button_clear"
android:layout_width="80px"
android:layout_height="80px"
android:gravity="center"
android:text="Clear"
android:textSize="25px"
/>
</TableRow>
</TableLayout>
</LinearLayout>
```

LinearLayout chứa 1 TextView để hiển thị số nhân, 1 TableLayout có các Button tương ứng với các số và 1 Button để clear cho TextView.



**B4:** Code code code... So tired... Tutorial is really take time. Chinh Example.java:

Mã:

```
package at.exam;

import android.app.Activity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;

public class Example extends Activity {
    Button button1, button2, button3;
    Button button4, button5, button6;
    Button button7, button8, button9;
    Button button0, buttonStar, buttonClear;

    TextView numberView;

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

        numberView = (TextView) findViewById(R.id.number_display);

        button1 = (Button) findViewById(R.id.button1);
        button2 = (Button) findViewById(R.id.button2);
        button3 = (Button) findViewById(R.id.button3);

        button4 = (Button) findViewById(R.id.button4);
        button5 = (Button) findViewById(R.id.button5);
        button6 = (Button) findViewById(R.id.button6);

        button7 = (Button) findViewById(R.id.button7);
        button8 = (Button) findViewById(R.id.button8);
        button9 = (Button) findViewById(R.id.button9);

        button0 = (Button) findViewById(R.id.button0);
        buttonStar = (Button) findViewById(R.id.button_star);
        buttonClear = (Button) findViewById(R.id.button_clear);

        button1.setOnClickListener(this.appendString("1"));
        button2.setOnClickListener(this.appendString("2"));
        button3.setOnClickListener(this.appendString("3"));
        button4.setOnClickListener(this.appendString("4"));
    }
}
```

```

button5.setOnClickListener(this.appendString("5")) ;
button6.setOnClickListener(this.appendString("6")) ;
button7.setOnClickListener(this.appendString("7")) ;
button8.setOnClickListener(this.appendString("8")) ;
button9.setOnClickListener(this.appendString("9")) ;
button0.setOnClickListener(this.appendString("0")) ;
buttonStar.setOnClickListener(this.appendString("* "));

buttonClear = (Button) findViewById(R.id.button_clear);
buttonClear.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
numberView.setText("");
}
});
}

public OnClickListener appendString(final String number) {
return new OnClickListener() {
public void onClick(View arg0) {
numberView.append(number);
}
};
}

public boolean onCreateOptionsMenu(Menu menu) {
super.onCreateOptionsMenu(menu);
menu.add(0, Menu.FIRST, 0, "Exit" ).setIcon(android.R.drawable.ic_delete);
return true;
}

public boolean onOptionsItemSelected(MenuItem item) {
switch (item.getItemId()) {
case Menu.FIRST: {
finish();
break;
}
}
return false;
}
}
}

```

Code quá đơn giản, mình còn ko thèm comment nữa. Lưu ý có 1 Option Menu để đóng Activity và cũng là đóng luôn ứng dụng.

**B5:** Time to test... Khởi chạy project, rồi sử dụng Option Menu của mình (bấm nút Menu của Emulator hoặc di động Android) để thoát khỏi chương trình. Ok, sau khi chọn Exit ta có thể chắc chắn là ứng dụng đã được đóng hoàn toàn, activity ko còn tồn tại trong stack của Emulator/di động nữa. Giờ nhấn nút Call của Emulator/di động, Tadaaaaaaaaa

## Bài 5 - Lập trình Android cơ bản - Android Service

Nguồn bài viết : DroidViet.Com [Lập trình Android cơ bản: Bài 5 Android Service](#)  
Service là 1 trong 4 thành phần chính trong 1 ứng dụng Android ( Activity, Service, BroadcastReceiver, ContentProvider) thành phần này chạy trong hậu trường và làm những công việc không cần tới giao diện như chơi nhạc, download, xử lý tính toán...

*Một Service có thể được sử dụng theo 2 cách:*

- Nó có thể được bắt đầu và được cho phép hoạt động cho đến khi một người nào đó dừng nó lại hoặc nó tự ngắt. Ở chế độ này, nó được bắt đầu bằng cách gọi Context.startService() và dừng bằng lệnh Context.stopService(). Nó có thể tự ngắt bằng lệnh Service.stopSelf() hoặc Service.stopSelfResult(). Chỉ cần một lệnh stopService() để ngừng Service lại cho dù lệnh startService() được gọi ra bao nhiêu lần

- Service có thể được vận hành theo như đã được lập trình việc sử dụng một Interface mà nó định nghĩa. Các người dùng thiết lập một đường truyền tới đối tượng Service và sử dụng đường kết nối đó để thêm nhập vào Service. Kết nối này được thiết lập bằng cách gọi lệnh Context.bindService() và được đóng lại bằng cách gọi lệnh Context.unbindService(). Nhiều người dùng có thể kết nối tới cùng một thiết bị. Nếu Service vẫn chưa được khởi chạy, lệnh bindService() có thể tùy ý khởi chạy nó. Hai chế độ này thì không tách biệt toàn bộ. Bạn có thể kết nối với một Service mà nó đã được bắt đầu với lệnh startService(). Ví dụ, một Service nghe nhạc ở chế độ nền có thể được bắt đầu bằng cách gọi lệnh startService() cùng với một đối tượng Intent mà định dạng được âm nhạc để chơi. Chỉ sau đó, có thể là khi người sử dụng muốn kiểm soát trình chơi nhạc hoặc biết thêm thông tin về bài hát hiện tại đang chơi, thì sẽ có một Activity tạo lập một đường truyền tới Service bằng cách gọi bindService(). Trong trường hợp như thế này, stopService() sẽ không thực sự ngừng Service cho đến khi liên kết cuối cùng được đóng lại.

Giống như một Activity, một Service cũng có các phương thức chu kỳ thời gian mà bạn có thể cài đặt để kiểm soát những sự thay đổi trong trạng thái của nó. Những những phương thức của Service thì ít hơn là của Activity – chỉ có 3- và chúng thì được sử dụng rộng rãi, không được bảo vệ.

**void onCreate()**  
**void onStart(Intent intent)**  
**void onDestroy()**

Bằng việc thực hiện những phương thức này, bạn có thể giám sát 2 vòng lặp của chu kỳ thời gian của mỗi Service

Entire lifetime của một Service diễn ra giữa thời gian onCreate() được gọi ra và thời gian mà onDestroy() trả lại. Giống như một Activity, một Service lại tiết hành cài đặt ban đầu ở onCreate(), và giải phóng tất cả các tài nguyên còn lại ở onDestroy() Ví dụ, một Service phát lại nhạc có thể tạo ra một luồng và bắt đầu chơi nhạc onCreate(), và sau đó luồng chơi nhạc sẽ dừng lại ở onCreate(),

Active lifetime của một Service bắt đầu bằng một lệnh tới onStart(). Đây là phương thức được chuyển giao đối tượng Intent mà đã được thông qua để tới startService()

Service âm nhạc sẽ mở đối tượng Intent để quyết định xem sẽ chơi loại nhạc nào và bắt đầu phát nhạc.

Không có callback tương đương nào cho thời điểm Service ngừng lại – không có phương thức onStop()

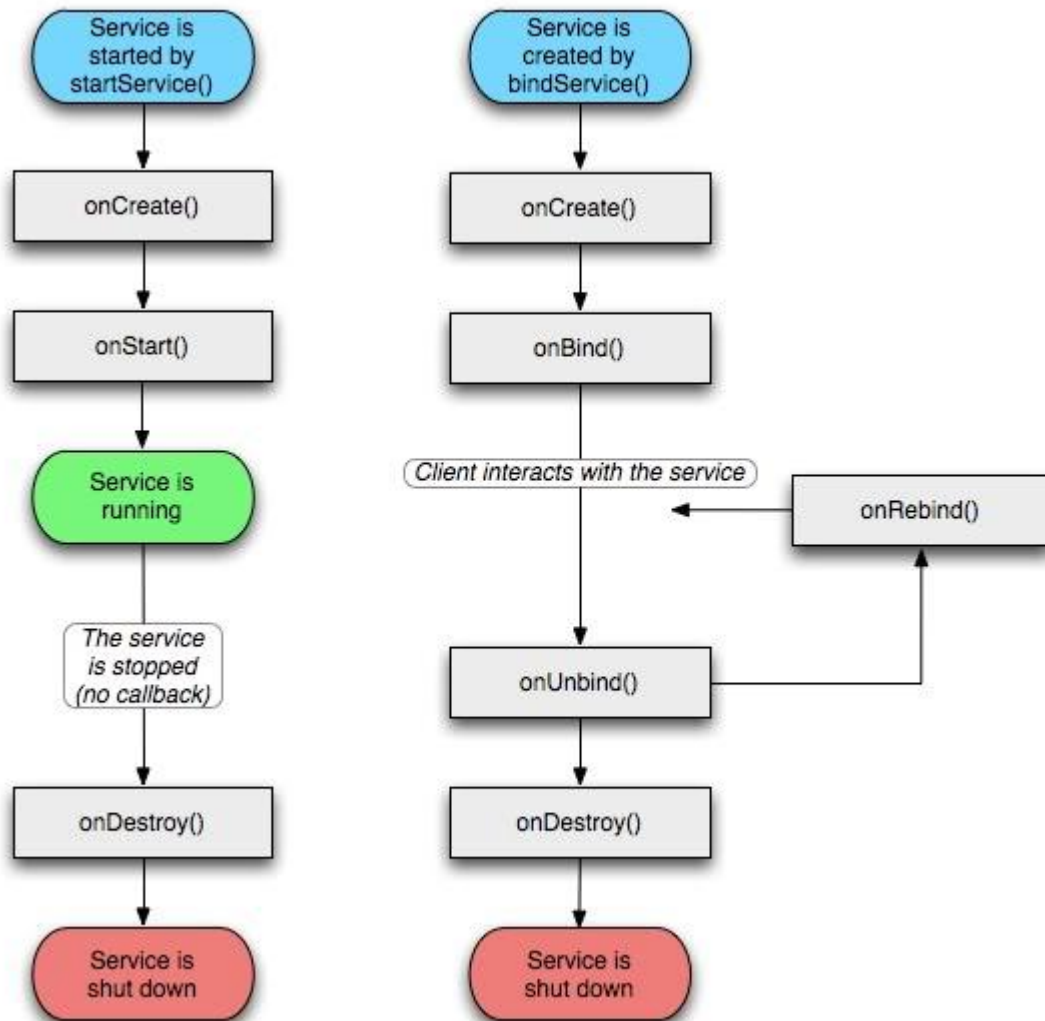
Các phương thức onCreate() và onDestroy() được gọi cho tất cả các Service dù chúng có được bắt đầu bằng Context.startService() hoặc Context.bindService() hay không. Tuy nhiên thì, onStart() chỉ được gọi ra đối với các Service bắt đầu bằng startService().

Nếu một Service cho phép những Service khác kết nối với nó thì sẽ có thêm các phương thức callback dành cho Service đó để thực hiện

**IBinder onBind(Intent intent)**  
**boolean onUnbind(Intent intent)**  
**void onRebind(Intent intent)**

Hàm callback onBind() thông qua đối tượng Intent đã được truyền đến bindService và onUnbind() được chuyển giao đối tượng mà đã được chuyển đến. Nếu Service đang được chỉ định (binding), onBind() quay trở lại kênh thông tin mà người dùng sử dụng để tương tác với Service. Phương thức onUnbind() có thể yêu cầu onRebind() được gọi nếu một người dùng kết nối với Service

Biểu đồ dưới đây minh họa cho các phương thức callback giành cho một Service. Mặc dù, nó phân tách các Service được tạo ra thông qua startService với các Service mà được tạo ra bằng bindService(). Hãy nhớ rằng bất kì Service nào, cho dù nó được khởi tạo như thế nào thì nó vẫn có thể cho phép các người dùng kết nối tới nó một cách hiệu quả nhất, cho nên bất kì Service nào cũng có thể được chỉ định thông qua các các phương thức onBind() và onUnbind()



Service LifeCycle

## Bài 6: Tự học lập trình Android - Android SQLite Database

### 0. Giới thiệu SQLite Database

Mỗi ứng dụng đều sử dụng dữ liệu, dữ liệu có thể đơn giản hay đôi khi là cả 1 cấu trúc. Trong Android thì hệ cơ sở dữ liệu được sử dụng là SQLite Database, đây là hệ thống mã nguồn mở được sử dụng rộng rãi trong các ứng dụng ( Mozilla Firefox sử dụng SQLite để lưu trữ các dữ liệu về cấu hình, iPhone cũng sử dụng cơ sở dữ liệu là SQLite).

Trong Android , cơ sở dữ liệu mà bạn tạo cho 1 ứng dụng thì chỉ ứng dụng đó có quyền truy cập và sử dụng, các ứng dụng khác thì không. Khi đã được tạo, cơ sở dữ liệu SQLite được chứa trong thư mục `/data/data/<package_name>/databases`.

*Tiếp theo mình sẽ giới thiệu với các bạn cách tạo và sử dụng cơ sở dữ liệu SQLite . Cách tốt nhất để hiểu lý thuyết là thực hành, các bạn có thể tạo ngay 1 project trong workspace đặt tên là : SQLiteDemo. Ứng dụng này đơn giản chỉ tạo 1 cơ sở dữ liệu USER bao gồm 2 trường ( id , name ) , ngoài ra có các thao tác thêm , đọc , sửa , xóa.*

#### Các bước thực hiện với CSDL:

1. Tạo 1 CSDL ( thông thường chỉ cần làm 1 lần )
2. Mở CSDL đó
3. Thêm giá trị vào trong table
4. Truy vấn.
5. Đóng CSDL

#### 1. Tạo cơ sở dữ liệu.

Đầu tiên các bạn add 1 class DBAdapter để xử lý tất cả các thao tác liên quan đến CSDL.

Mã:

```
public static final String KEY_ID = "_id";
public static final String KEY_NAME = "name";

private DatabaseHelper mDbHelper;
private SQLiteDatabase mDB;

private static final String DATABASE_CREATE = "create table users (_id integer
primary key autoincrement, "
+ "name text not null);";
private static final String DATABASE_NAME = "Database_Demo";
private static final String DATABASE_TABLE = "users";
private static final int DATABASE_VERSION = 2;

private final Context mContext;
Tạo 1 lớp bên trong DBAdapter được extend từ lớp SQLiteOpenHelper, override 2
phương thức onCreate() và onUpgrade() để quản lý việc tạo CSDL và version của
```

CSDL đó.

Mã:

```
private static class DatabaseHelper extends SQLiteOpenHelper{

public DatabaseHelper(Context context, String name,
CursorFactory factory, int version) {
super(context, name, factory, version);
// TODO Auto-generated constructor stub
}

@Override
public void onCreate(SQLiteDatabase db) {
// TODO Auto-generated method stub
db.execSQL(DATABASE_CREATE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
// TODO Auto-generated method stub
Log.i(TAG, "Upgrading DB");
db.execSQL("DROP TABLE IF EXISTS users");
onCreate(db);
}
}
```

## 2. Mở CSDL :

Mã:

```
public DBAdapter open()
{
mDbHelper = new DatabaseHelper(mContext, DATABASE_NAME, null,
DATABASE_VERSION);
mDB = mDbHelper.getWritableDatabase();
return this;
}
```

## 3. Thêm giá trị vào CSDL

Mã:

```
public long createUser(String name){
ContentValues inititalValues = new ContentValues();
inititalValues.put(KEY_NAME, name);
return mDB.insert(DATABASE_TABLE, null, inititalValues);
}
```

## 4. Truy vấn

Bạn có thể get toàn bộ data hoặc có thể get data theo ID ( tiện cho việc chỉnh sửa hay cập nhật thông tin của từng bản ghi).

Mã:

```
public Cursor getAllUsers(){
return mDB.query(DATABASE_TABLE, new String[] {KEY_ID, KEY_NAME}, null, null,
null, null, null);
}
```

Còn rất nhiều các thao tác như sửa, xóa, update.... bản ghi, các bạn có thể tự phát triển.

Tất cả các chức năng đó đều được cung cấp bởi lớp SQLiteDatabase, các bạn chỉ cần cụ thể hóa bằng các câu truy vấn là được.

## 5. Đóng CSDL

Mã:

```
public void close(){
    mDbHelper.close();
}
```

## 6. Sử dụng CSDL

Để test CSDL mà bạn vừa tạo, các bạn có thể thêm 1 vài dòng code để thêm 1 user và hiển thị CSDL lên màn hình thông qua lớp Activity ban đầu:

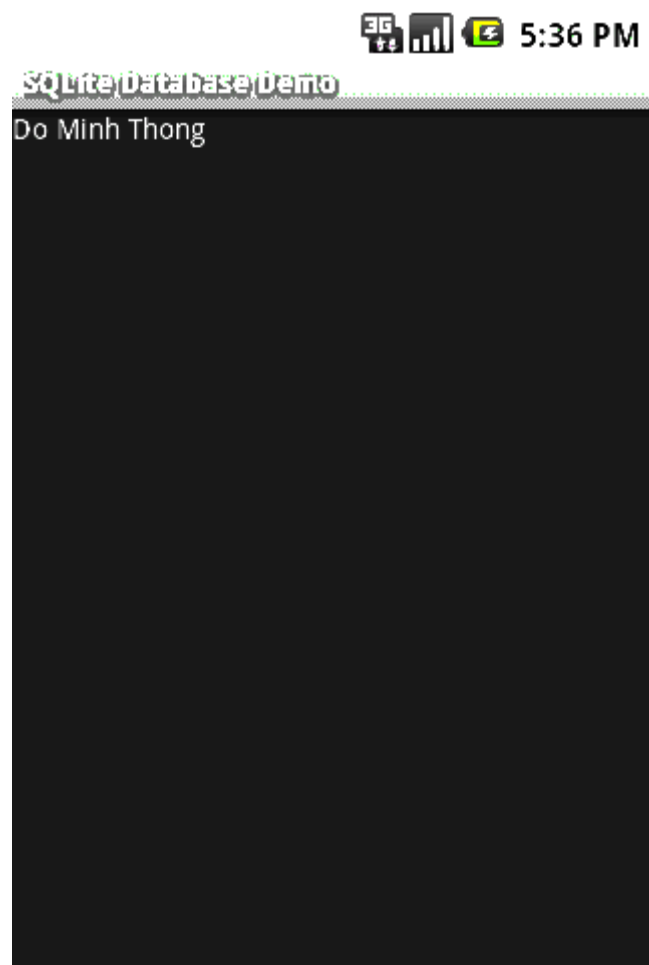
Ở đây mình create 1 user thông qua câu lệnh `mDB.createUser("Username");` sau đó Mã:

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    mDB = new DBAdapter(this);
    mDB.open();
    mDB.createUser("Do Minh Thong");
    getData();
}
```

```
private void getData(){
    mCursor = mDB.getAllUsers();
    startManagingCursor(mCursor);
    String[] from = new String[]{DBAdapter.KEY_NAME};
    int[] to = new int[] {R.id.text1};
    SimpleCursorAdapter users = new SimpleCursorAdapter(this, R.layout.users_row,
    mCursor, from, to);
    setListAdapter(users);
}
```

**Kết quả :**





**Sourcecode đầy đủ** : <http://www.mediafire.com/?yzw2d1ijymo>

## Bài 7 - Lập trình Android cơ bản - Android Content Provider

**Trước khi vào bài, các bạn down Sourcecode của bài học tại đây hoặc**

[http://www.mediafire.com/download/wo2gyjnng2j/VietAndroid.com\\_ContentProvider\\_Tutorial.rar#!](http://www.mediafire.com/download/wo2gyjnng2j/VietAndroid.com_ContentProvider_Tutorial.rar#!)

Sourcecode của bài học bao gồm 3 Project demo :

*ContentProviderDemo*

*ContentProviderDemo1*

*ContentProviderDemo2*

Mỗi Demo sẽ giải quyết từng vấn đề của Tutorial. Các bạn cứ add hết vào workspace để tiện theo dõi.

### 1. Giới thiệu Content Provider

Content Provider là 1 trong 4 thành phần cơ bản của 1 ứng dụng Android thường có bao gồm:

- 1. Activity**
- 2. Service**
- 3. Broadcast Receiver**
- 4. Content Provider**

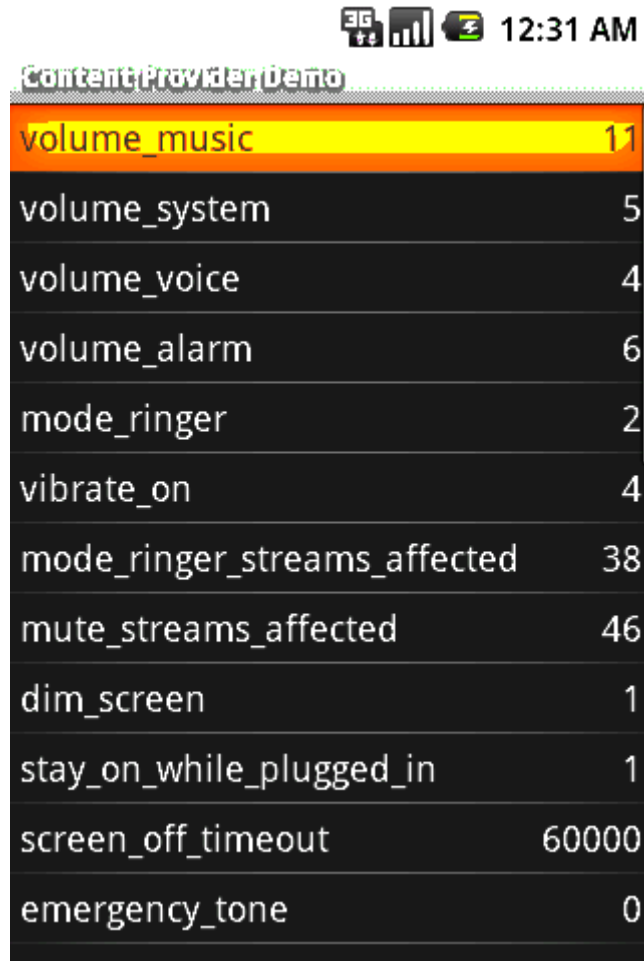
Một Content Provider cung cấp một tập chi tiết dữ liệu ứng dụng đến các ứng dụng khác. Thường được sử dụng khi chúng ta muốn tạo cơ sở dữ liệu dưới dạng public (các ứng dụng khác có thể truy xuất).

Dữ liệu thường được lưu trữ ở file hệ thống, hoặc trong một SQLite database. Đơn giản để các bạn có thể hình dung như : Danh bạ, Call log, cấu hình cài đặt...trên điện thoại là dữ liệu dưới dạng Content Provider.

Content Provider hiện thực một tập phương thức chuẩn mà các ứng dụng khác có thể truy xuất và lưu trữ dữ liệu của loại nó điều khiển.

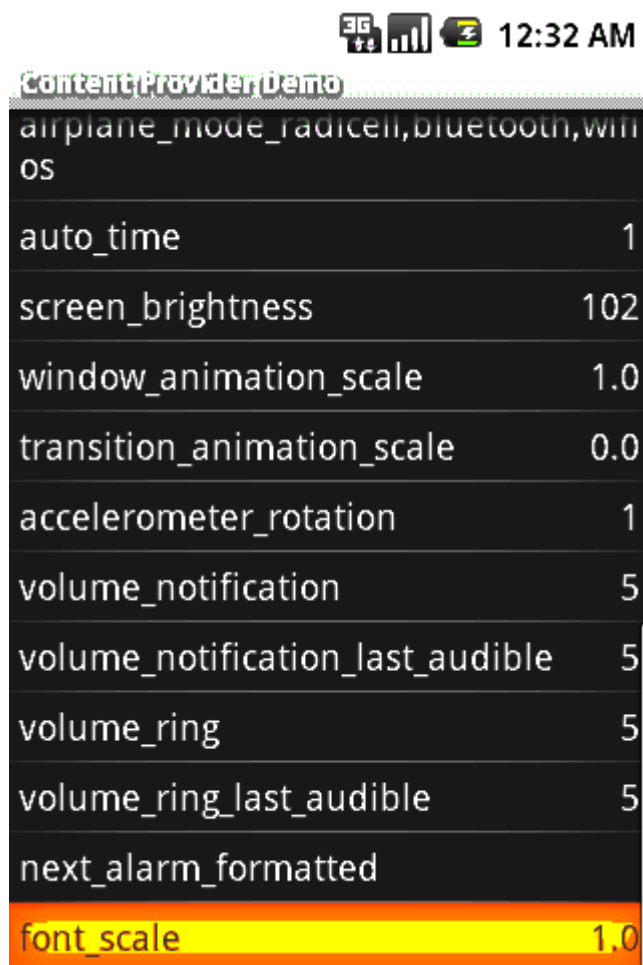
Tuy nhiên, những ứng dụng không thể gọi các phương thức trực tiếp. Hơn thế chúng dùng lớp Content Resolver và gọi những phương thức đó. Một Content Resolver có thể giao tiếp đến nhiều content provider; nó cộng tác với các provider để quản lý bất kỳ giao tiếp bên trong liên quan.

Đơn giản hơn, chúng ta có thể làm 1 ứng dụng nhỏ để lấy tất cả các thông tin cấu hình trong máy load lên listview. Các bạn có thể chạy Project ContentProviderDemo1 trong SourcecodeDemo.



The image shows a screenshot of an Android phone's notification shade. At the top, there are status bar icons for 3G, signal strength, and battery, along with the time 12:31 AM. Below the status bar, the notification shade is pulled down, revealing a list of system settings. The first item, 'volume\_music', is highlighted in orange and has a value of 11. The other items and their values are: volume\_system (5), volume\_voice (4), volume\_alarm (6), mode\_ringer (2), vibrate\_on (4), mode\_ringer\_streams\_affected (38), mute\_streams\_affected (46), dim\_screen (1), stay\_on\_while\_plugged\_in (1), screen\_off\_timeout (60000), and emergency\_tone (0).

Setting	Value
volume_music	11
volume_system	5
volume_voice	4
volume_alarm	6
mode_ringer	2
vibrate_on	4
mode_ringer_streams_affected	38
mute_streams_affected	46
dim_screen	1
stay_on_while_plugged_in	1
screen_off_timeout	60000
emergency_tone	0



Chúng ta có thể tìm hiểu sơ qua về code của demo này, rất ngắn gọn

Mã:

```
ContentResolver cr = getContentResolver();
Cursor cursor = cr.query(Settings.System.CONTENT_URI, null, null, null, null);
startManagingCursor(cursor);

ListView listView = (ListView) findViewById(R.id.listView);
String[] from = { Settings.System.NAME, Settings.System.VALUE };
int[] to = { R.id.textName, R.id.textValue };
SimpleCursorAdapter adapter = new SimpleCursorAdapter(this, R.layout.row, cursor,
from, to);
listView.setAdapter(adapter);
```

Như các bạn thấy, chỉ cần 2 dòng code đơn giản để lấy được con trỏ thao tác trên tập dữ liệu cần lấy:

Mã:

```
ContentResolver cr = getContentResolver();
Cursor cursor = cr.query(Settings.System.CONTENT_URI, null, null, null, null);
```

Lớp Content Resolver cung cấp các phương thức xử lý dữ liệu thông qua các Uri, mỗi Content Provider có 1 Uri cụ thể, ở đây Uri `Settings.System.CONTENT_URI` sẽ trả lại

tập dữ liệu là thông tin cấu hình của thiết bị.

Sau khi lấy được con trỏ tới tập dữ liệu, việc còn lại đơn giản là bind data lên listview để hiển thị:

Mã:

```
startManagingCursor(cursor);
ListView listView = (ListView) findViewById(R.id.listView);
String[] from = { Settings.System.NAME, Settings.System.VALUE };
int[] to = { R.id.textName, R.id.textValue };
SimpleCursorAdapter adapter = new SimpleCursorAdapter(this, R.layout.row, cursor,
from, to);
listView.setAdapter(adapter)
```

## 2. Tạo và sử dụng 1 Content Provider do người dùng tự định nghĩa

Để dễ hiểu hơn các bạn mở Project ContentProviderDemo trong Sourcecode đã down về. Trong Project đó mình tạo 1 Content Provider Books, mỗi bản ghi Book bao gồm 2 trường : ID và Title.

Sau đây là các bước để tạo 1 Content Provider cơ bản ( cụ thể là tạo ContentProvider Book)

### 1. Tạo 1 class thừa kế lớp ContentProvider

Mã:

```
public class BookProvider extends ContentProvider
```

**2.** Định nghĩa 1 biến Uri (public static final ) được gọi CONTENT\_URI. Các xâu này luôn được bắt đầu bằng "content://" tiếp theo đó là nội dung của mà ContentProvider xử lý. Xâu này phải có đặc tính là duy nhất.

Mã:

```
public static final String PROVIDER_NAME = "com.vietandroid.provider.Books";
public static final Uri CONTENT_URI = Uri.parse("content://" + PROVIDER_NAME +
"/books");
```

**3.** Khai báo các xâu để định nghĩa cho từng thuộc tính tương ứng với các cột giá trị từ Cursor.

Mã:

```
public static final String _ID = "_id";
public static final String TITLE = "title";
```

**4.** Chúng ta cần tạo hệ thống chứa dữ liệu cho ContentProvider, có thể chứa dưới nhiều hình thức : sử dụng XML, thông qua CSDL SQLite, hay thậm chí là Webservice. Trong Demo này chúng ta sử dụng cách phổ biến nhất đó là SQLite:

Mã:

```
private SQLiteDatabase bookDB;
private static final String DATABASE_NAME = "Books";
private static final String DATABASE_TABLE = "titles";
private static final int DATABASE_VERSION = 1;
```

**5.** Định nghĩa tên của các cột mà chúng ta sẽ trả lại giá trị cho các clients. Nếu chúng ta đang sử dụng Database ContentProvider hay các lớp SQLiteOpenHelper, tên các cột này chính là id của các cột trong cơ sở dữ liệu SQL. Trong trường hợp này, chúng ta phải gộp cả cột có giá trị là số nguyên được gọi "\_id" để định nghĩa id của mỗi

bản ghi. Nếu đang sử dụng cơ sở dữ liệu SQLite, nó sẽ là INTEGER PRIMARY KEY AUTOINCREMENT. Tùy chọn AUTOINCREMENT không bắt buộc, có tác dụng tự động tăng ID của mỗi bản ghi lên nếu người dùng không nhập. Android cung cấp SQLiteOpenHelper giúp tạo và quản lý các phiên bản của cơ sở dữ liệu.

Mã:

```
private static final String DATABASE_CREATE =
"create table " + DATABASE_TABLE +
" (_id integer primary key autoincrement, "
+ "title text not null);";

private static class DatabaseHelper extends SQLiteOpenHelper
{
public DatabaseHelper(Context context) {
super(context, DATABASE_NAME, null, DATABASE_VERSION);
}

@Override
public void onCreate(SQLiteDatabase db) {
db.execSQL(DATABASE_CREATE);
}

@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
db.execSQL("DROP TABLE IF EXISTS titles");
onCreate(db);
}
}
```

**6.** Nếu chúng ta muốn public các dữ liệu kiểu byte như bitmap thì các trường mà chứa dữ liệu này nên là một chuỗi với 1 content://URI cho file đó. Đây chính là liên kết để các ứng dụng khác có thể truy cập và sử dụng dữ liệu bitmap này.

**7.** Sử dụng Cursor để thao tác trên tập dữ liệu : query (), update(), insert(), delete()..... Có thể gọi phương thức ContentResolver.notifyChange() để biết khi nào dữ liệu được cập nhật.

Add Book

Mã:

```
@Override
public Uri insert(Uri uri, ContentValues values) {
long rowID = bookDB.insert(DATABASE_TABLE, "", values);
if(rowID > 0)
{
Uri mUri = ContentUris.withAppendedId(CONTENT_URI, rowID);
getContext().getContentResolver().notifyChange(mUri, null);
return mUri;
}
}
```

```
throw new SQLException("Failed to insert new row into " + uri);
}
```

Get All Books

Mã:

```
@Override
public Cursor query(Uri uri, String[] projection, String selection,
String[] selectionArgs, String sortOrder) {
    SQLiteQueryBuilder sqlBuilder = new SQLiteQueryBuilder();
    sqlBuilder.setTables(DATABASE_TABLE);
    if(uriMatcher.match(uri) == BOOK_ID)
    sqlBuilder.appendWhere(_ID + "=" + uri.getPathSegments().get(1));
    if(sortOrder == null || sortOrder == "")
    sortOrder = TITLE;
    Cursor c = sqlBuilder.query(bookDB, projection, selection, selectionArgs, null, null,
    sortOrder);
    c.setNotificationUri(getContext().getContentResolver(), uri);
    return c;
}
}
```

Mình chỉ demo 2 chức năng là thêm sách và lấy toàn bộ bản ghi trong CSDL , ngoài ra các phương thức edit, sửa , update, xóa... các bạn có thể tự làm .

## 8. Khai báo Content Provider trong file AndroidManifest.xml

Mã:

```
<provider android:name = "BookProvider"
android:authorities="com.vietandroid.provider.Books" />
```

Như vậy chúng ta đã tạo xong ContentProvider Book tự định nghĩa.

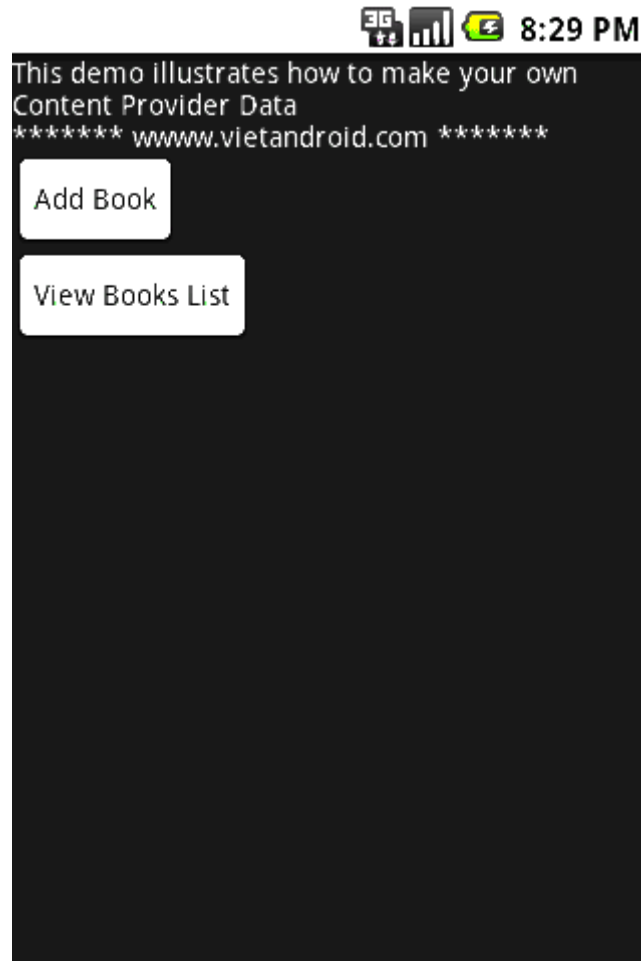
## 9. Test thử thành quả :

Mỗi content Provider gắn với 1 Uri cụ thể, như trên thì ContentProvider Book có Uri là:

Mã:

```
com.vietandroid.provider.Books/books
```

Để test thử , vẫn trong Project Demo ContentProviderDemo , các bạn có thể thêm 2 Button Add Book và View All Books vào . Giao diện như sau:



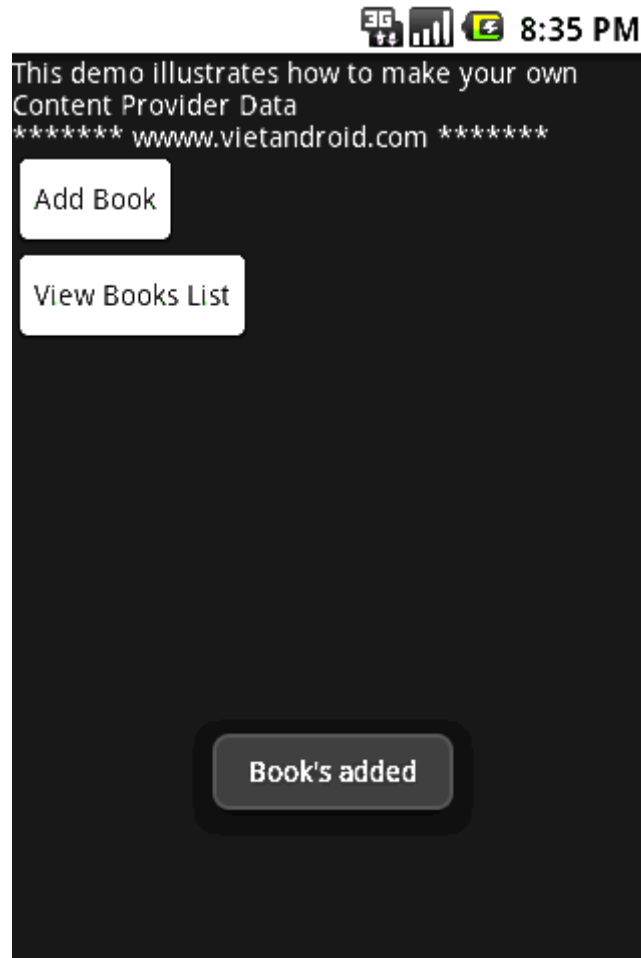
Chúng ta sẽ thêm 1 Book vào CSDL thông qua URI này:

Mã:

```
public void addBook(String title)
{
    ContentValues values = new ContentValues();
    values.put(BookProvider.TITLE, title);
    Uri uriInsert = getContentResolver().insert(BookProvider.CONTENT_URI, values);
    if(uriInsert != null)
    {
        Toast.makeText(this, "Book's added", Toast.LENGTH_SHORT).show();
    }
    Log.d(getClass().getSimpleName(), uriInsert.toString());
}
```

Kết quả :



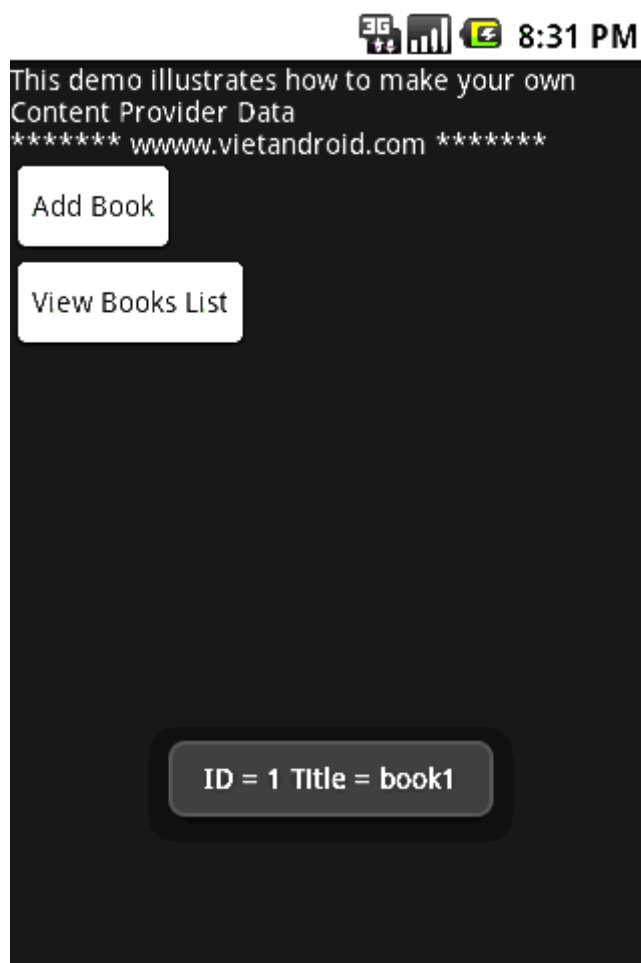


Truy vấn toàn bộ dữ liệu Books có trong CSDL

Mã:

```
public void getAllBooks()
{
    Uri uriGetListTitles = Uri.parse("content://com.vietandroid.provider.Books/books");
    Cursor c = managedQuery(uriGetListTitles, null, null, null, "title desc");
    if(c.moveToFirst()){
        do{
            String bookRecord = "ID = " + c.getString(c.getColumnIndex(BookProvider._ID)) +
            " Title = " +
            c.getString(c.getColumnIndex(BookProvider.TITLE));
            Toast.makeText(this, bookRecord, Toast.LENGTH_LONG).show();
        }while(c.moveToNext());
    }
}
```

Kết quả :



### 3. Sử dụng dữ liệu Content Provider từ 1 ứng dụng bất kỳ

Ở Bài 6 mình đã đề cập về cơ sở dữ liệu SQLite Database, dạng dữ liệu này không public cho các ứng dụng khác sử dụng, dữ liệu của ứng dụng nào thì ứng dụng đó sử dụng.

1 lợi thế của dữ liệu dưới dạng Content Provider là public, tất cả các ứng dụng đều có thể truy cập và sử dụng.

Phần này các bạn sử dụng ProjectDemo là ContentProviderDemo2 trong sourcecode đi kèm ban đầu.

Demo này chỉ đơn giản là đọc lại toàn bộ dữ liệu trong CSDL Books được tạo trong phần 2.

Như mình đã nói ở trên, chỉ cần lấy được Uri của ContentProvider cần lấy và các tên của các trường dữ liệu thì chúng ta có thể truy vấn được hết.

Trong hàm onCreate() các bạn thêm vào:

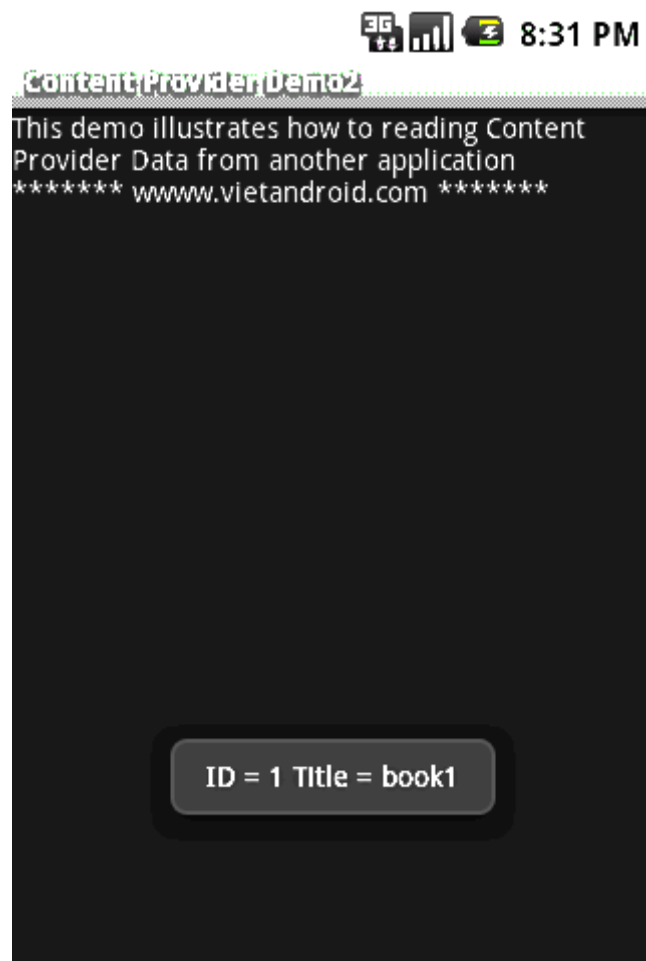
Mã:

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    Uri uriGetListTitles = Uri.parse("content://com.vietandroid.provider.Books/books");
    Cursor c = managedQuery(uriGetListTitles, null, null, null, "title desc");
```

```
if(c != null)
{
if(c.moveToFirst()){
do{
String bookRecord = "ID = " + c.getString(c.getColumnIndex("_id")) + " Title = " +
c.getString(c.getColumnIndex("title"));
Toast.makeText(this, bookRecord , Toast.LENGTH_LONG).show();
}while(c.moveToNext());
}
}
else {
Toast.makeText(this, "Database is empty", Toast.LENGTH_SHORT).show();
}
}
```

2 trường dữ liệu ở đây được định nghĩa ở trên là "\_id" và "title". Phần truy vấn vẫn như vậy.

Kết quả :



Nguồn bài viết : DroidViet.Com [Lập trình Android cơ bản: Bài 7 Android Content Provider](#)