Phát triển ứng dụng Smartphone

Tài liệu lưu hành nội bộ

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

Phát triển ứng dụng Smartphone

Phần 01: Làm quen với Android

Lê Đức Huy

Email: leduchuy89vn@gmail.com

Mục lục

1 Tổng quan Android

Ứng dụng Android được viết bằng ngôn ngữ Java. Bộ công cụ Android SDK biên dịch mã lệnh cùng với bất kì dữ liệu cũng như tài nguyên(hình ảnh, âm thanh) đi kèm với ứng dụng tạo thành gói có phần mở rộng .apk (Viết tắt của từ Android package). Tất cả phần mã thực thi đi kèm với tài nguyên (resources) được nén trong một tệp đơn có đuôi .apk được xem như là một ứng dụng chạy trên các thiết bị chạy Android. File nén này có thể dùng để cài đặt và thực thi.

Khi được cài đặt vào thiết bị, mỗi ứng dụng Android "sống" trong một "hộp cát bảo mật" của chính nó. Trong đó:

- Hệ điều hành Android được xem như một hệ thống Linux đa người dùng. Nơi mà mỗi ứng dụng được xem như một người dùng.
- Mặc định, hệ thống sẽ cấp phát cho mỗi ứng dụng một định danh (User ID-Unique Linux user ID), định danh này chỉ được sử dụng bởi hệ điều hành và ứng dụng không hề hay biết về định danh này. Hệ thống sẽ gán quyền cho mọi tệp tin trong một ứng dụng để chỉ những User ID được gán cho ứng dụng đó mới có thể truy cập chúng.
- Mỗi tiến trình trên nền tảng Android sẽ có một máy ảo (Virtual machine) của riêng nó, bằng cách này các câu mã lệnh của một ứng dụng sẽ được thực thi một cách độc lập với các ứng dụng khác.
- Mặc định, mỗi ứng dụng sẽ chạy trên tiến trình của riêng nó. Android sẽ khởi động tiến trình khi mà bất kì một thành phần nào của ứng dụng cần thực thi và sau đó sẽ tắt mọi tiến trình khi mà nó không cần dùng thêm nữa hoặc khi hệ thống cần bộ nhớ cho ứng dụng khác.

Bằng cách này Android đã hiện thực hiện "nguyên tắc đặc quyền tối thiểu". Đó là, theo mặc định, mỗi ứng dụng sẽ chỉ được truy cập đến những thành phần mà nó cần phải sử dụng để làm việc và không thêm gì khác. Điều này tạo ra một môi trường cực kì bảo mật, nơi luôn đảm bảo mọi ứng dụng không thể truy xuất các phần khác của hệ thống nếu nó không được quyền. Điều này không có nghĩa là mọi ứng dụng hoàn toàn độc lập với các ứng dụng khác cũng như với hệ điều hành. Có nhiều cách để một ứng dụng có thể chia sẻ dữ liệu với ứng dụng khác hoặc để truy xuất các dịch vụ của hệ điều hành(contact, phone, sms, email…) như:

- Nếu hai ứng dụng có thể truy xuất nguồn tài nguyên của nhau thì có thể cho hai ứng dụng nhận cùng một User ID. Để tiết kiệm nguồn tài nguyên thì các ứng dụng có cùng User ID sẽ thực thi trên cùng một tiến trình trên cùng một máy ảo.
- Một ứng dụng có thể yêu cầu quyền được truy xuất dữ liệu của thiết bị như contact, sms, SD card, camera, Bluetooth… Mọi quyền được sử dụng trên một ứng dụng cụ thể phải được gán khi ứng dụng đó được cài đặt lên thiết bị (Android sẽ thông báo những quyền(Truy cập Internet, contact, sms…) mà ứng dụng cần sử dụng để hỏi người dùng có đồng ý cho phép cài đặt hay không).

Phần dưới đây sẽ cung cấp cho người đọc một số khái niệm về các thành phần của một ứng dụng Android.

1.1 Các thành phần của một ứng dụng Application

Cách thành phần của một ứng dụng Android là các khối thiết yếu dùng để ghép thành một ứng dụng Android. Mỗi thành phần là một góc nhìn khác nhau tạo thành một ứng dụng Android đầy đủ. Không phải bất kì ứng dụng nào cũng có đầy đủ các thành phần dưới đây, việc có hay không có một thành phần nào tùy thuộc vào mỗi ứng dụng. Một số thành phần sẽ có quan hệ phụ thuộc lẫn nhau. Mỗi thành phần có một vai trò khác nhau giúp định nghĩa một ứng dụng Android đầy đủ.

Dưới đây có bốn loại thành phần. Mỗi loại có một mục đích khác nhau và có một vòng đời khác nhau giúp định nghĩa cách mà một thành phần được tạo ra và được hủy khác nhau:

1.1.1 Activity

Một Activity đại diện một màn hình đơn với giao diện người dùng. VD: Chương trình Email tích hợp sẵn của hệ điều hành Android có một activity để hiển thị một danh sách các email, một activity thứ hai dùng để hiển thị nội dung chi tiết của một email. Mặc dù cách activity làm việc với nhau để tạo nên một trải nghiệm thống nhất trên ứng dụng, tuy nhiên bản thân chúng hoàn toàn độc lập với nhau. VD: Khi soạn thảo email với ứng dụng Email tích hợp sẵn người dùng có thể mở một activity của ứng dụng Camera nếu người dùng cần chia sẽ một tấm hình chụp bằng thiết bị hiện tại.

1.1.2 Service

Service là một thành phần dùng để thực thi một công việc dưới dưới nền hệ điều hành (Công việc được thực thi ẩn với người dùng). Thành phần này được dùng cho các công việc cần nhiều thời gian để thực thi hoặc dùng để điều khiển thực thi các công việc từ xa bởi một tiến trình khác. VD: Ứng dụng A chạy trên tiến trình pA kết nối đến ứng dụng B hiện đang chạy trên tiến trình pB để yêu cầu B thực hiện một công việc nào đó.

1.1.3 Content provider

Một content provider dùng quản lý việc chia sẻ một tập dữ liệu ứng dụng nào đó. Bạn có thể lưu trữ dữ liệu trên file, trên SQLite database, trên web hoặc trên bất kì nơi lưu trữ nào mà ứng dụng có thể truy xuất. Thông qua việc sử dụng content provider một ứng dụng khác có thể truy vấn, chỉnh sửa tập dữ liệu đó. VD: Ứng dụng Contact tích hợp sẵn trong Android cung cấp một content provider dùng để quản lý danh sách contact trên ứng dụng. Bằng cách này, bất kì chương trình nào cũng có thể truy xuất danh sách contact này mặc cho "nguyên tắc đặc quyền tối thiểu" đã đề cập ở trên. (Lưu ý: Nếu không cung cấp một content provider thì dữ liệu contact chỉ có thể được sử dụng bởi ứng dụng Contact và không thể được truy xuất bởi ứng dụng nào khác).

1.1.4 Broadcast receiver

Broadcast receiver là một thành phần hồi đáp những tín hiệu được phát ra trên toàn hệ thống. Có rất nhiều broadcast receiver được xây dựng sẵn trên hệ điều hành Android. VD: Có broadcast receiver dùng để bắt tín hiệu tắt màn hình, tín hiệu pin yếu… Người phát triển có thể tự xây dựng một broadcast receiver để bắt các tín hiện cần thiết cho ứng dụng. VD: Bắt tín hiệu để biết một dữ liệu nào đó đã được tải về thiết bị và sẵn sang để sử dụng. Mặc dù một broadcast receiver không hiển thị giao diện cho người dùng quan sát nhưng nó có thể tạo các thông báo trên thanh trạng thái. Trong một số trường hợp broadcast receiver có thể khởi động một service để thực thi một số công việc nào đó.

Một khía cạnh độc đáo của hệ điều hành Android là nó được thiết kế để bất kì ứng dụng nào cũng có thể khởi động một activity của ứng dụng khác. VD: Nếu bạn muốn người dùng ứng dụng có có thể chụp hình từ camera của thiết bị. Hiển nhiên công việc này đã được một ứng dụng khác làm rồi, bạn không cần tạo mới một activity để làm công việc này mà đơn giản chỉ cần khởi động một activity chụp ảnh của ứng dụng Camera để chụp hình. Khi chụp hình hoàn tất, bức ảnh sẽ được trả về cho ứng dụng của bạn. Bằng cách này ta có thể tiết kiệm rất nhiều thời gian bằng cách sử dụng lại cách thành phần có sẵn của hệ điều hành.

Khi hệ thống khởi động một thành phần, nó cũng khởi động tiến trình cho ứng dụng đó(Nếu nó chưa được khởi động). VD: Nếu ứng dụng của bạn sử dụng một activity của ứng dụng Camera để chụp hình, hệ thống sẽ khởi động tiến trình của ứng dụng Camera. Chính vì điểm này mà một ứng dụng Android sẽ hoàn toàn khác so với các ứng dụng trên các nền tảng khác. Một ứng dụng Android không có một điểm bắt đầu duy nhất (Không có hàm main).

Do hệ thống thực thi mỗi ứng dụng trên một tiến trình riêng biệt cùng với quyền truy xuất hạn chế nên bất kì một ứng dụng nào cũng không được quyền khởi động một thành phần của ứng dụng khác. Để khởi động một thành phần của ứng dụng khác, bạn phải chuyển một đối tượng kiểu Intent đến hệ điều hành, đối tượng kiểu Intent này sẽ chứa các thông tin về thành phần mà bạn muốn khởi động. Nhờ những thông tin này mà hệ điều hành sẽ khởi động thành phần phù hợp.

Khởi động thành phần

Ba trong số bốn loại thành phần chính của một ứng dụng Android (Activity, Service, Broadcast receiver) có thể được khởi động bởi một đối tượng Intent. Trong đó đối tượng Intent có ý nghĩ gần giống như một đường dẫn đến một địa chỉ web (URLs) mà hằng ngày mọi người vẫn sử dụng.

Định dạng URLs được Tim Berners phát minh để sử dụng trong giao thức Hypertext Transfer Protocol (HTTP). Định dạng này là một hệ thống các động từ đi kèm các địa chỉ. Địa chỉ sẽ xác định nguồn tài nguyên như Web page, hình ảnh... Động từ sẽ xác định cần phải làm cái gì với nguồn tài nguyên đó: GET để lấy dữ liệu về, POST để đưa dữ liệu lên để thực thi một công việc nào đó. Khái niệm Intent cũng tương tự, Intent là một mô tả trừu tượng của một hành động được thực thi. Trong đó đối tượng Intent sẽ xác định thành phần nào (Activity, Service, Broadcast receiver) sẽ được khởi động, dữ liệu cũng như hành động cần thực thi với dữ liệu đi kèm.

VD: Để hiển thị thông tin của một contact. Ta cần tạo một đối tượng kiểu Intent xác định hành động là "xem thông tin contact" thành phần Android sẽ được khởi động để thực thi hành động là activity "Contact details" của ứng dụng Contact (Ứng dụng tích hợp sẵn trong hệ điều hành Android) và dữ liệu đi kèm sẽ là id của contact cần xem thông tin.

Có thể sử dụng Intent để:

- Khởi động một Activity.

- Khởi động một Service.

- Kết nối đến một Remote service.

- Khởi động một Broadcast receiver.

- Thực thi một câu truy vấn dữ liệu trên Content Provider

1.2 AndroidManifest.xml

Trước khi hệ điều hành Android có thể khởi động một thành phần (Activity, Services, Broadcast receiver) thì hệ thống phải biết thành phần này có tồn tại hay không bằng cách đọc file AndroidManifest.xml của ứng dụng. Mỗi ứng dụng phải khai báo mọi thành phần (Activity, Service, Broadcast receiver) trong file này và phải đặt ở thư mục gốc của ứng dụng.

Mỗi file manifest có những khai báo:

- Định danh xác định các quyền của người sử dụng như truy xuất internet, truy xuất danh sách contact…
- Xác định phiên bản API tối thiểu mà ứng dụng có thể thực thi. Phiên bản API này tương ứng với các phiên bản của hệ điều hành Android.
- Các tính năng phần cứng cần thiết cho ứng dụng như GPS, camera, Bluetooth…
- Các bộ API liên kết sử dụng trong ứng dụng (VD: Google map…).
- Và …
- Một file AndroidManifest.xml có cấu trúc như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest ... >
   <application android:icon="@drawable/app_icon.png" ... >
      <activity android:name="com.example.project.ExampleActivity"
            android:label="@string/example_label" ... >
      </activity>
      ...
   </application>
</manifest>
```

Trong đó có thẻ gốc là thẻ manifest. Thẻ gốc này sẽ có ít nhất một thể con là application với các thuộc tính xác định icon của ứng dụng, tên ứng dụng… . Trong thẻ application sẽ đặt các khai báo các thành phần tồn tại trong ứng dụng bằng cách sử dụng các thẻ:

- <activity>: Khai báo một Activity.
- <service>: Khai báo một service.
- <receiver>: Khai báo một Broadcast reciever.
- <provider>: Khai báo một Content Provider.

Ngoài việc khai bao các thành phần tồn tại trong ứng dụng còn có các khai báo:

- <supports-screens>: Khai báo loại màn hình mà ứng dụng hổ trợ.
- <uses-configuration>: Khai báo phần cứng nhập liệu cần thiết cho ứng dụng (VD: Bàn phím, trackball, phiếm bấm năm chiều…).
- <uses-feature>: Các tính năng phần cứng cần thiết cho ứng dụng.
- <uses-sdk>: Phiên bản API tối thiểu.

2 Hello Android

Ứng dụng kinh điển nhất mà mọi lập trình viên khi làm việc với một ngôn ngữ lập trình là xuất là câu “Xin chào thế giới”. Để bắt đầu làm quen với Android ta sẽ bắt đầu với ứng dụng “Xin chào Android”.

Tạo mới một project Android với các thông số sau:

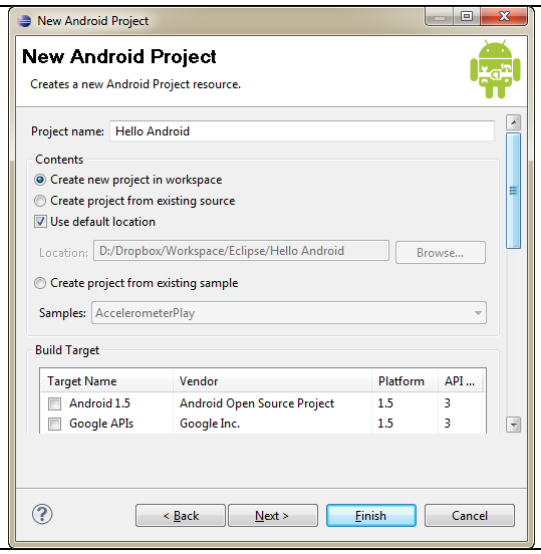
Project name: Hello Android

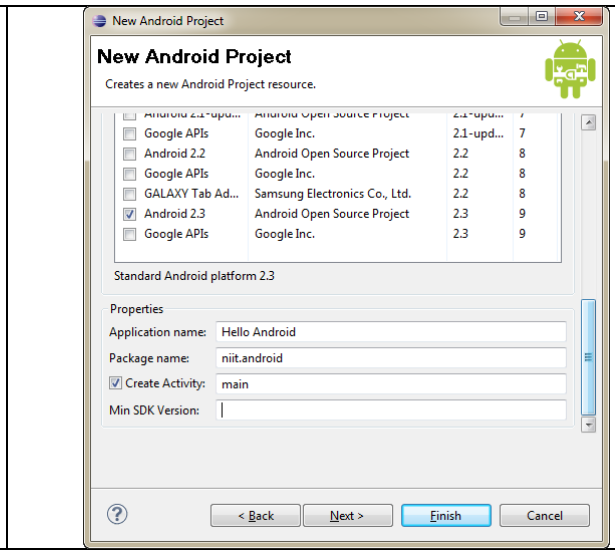
Build target: Android 2.3.3.

Application name: Hello Android

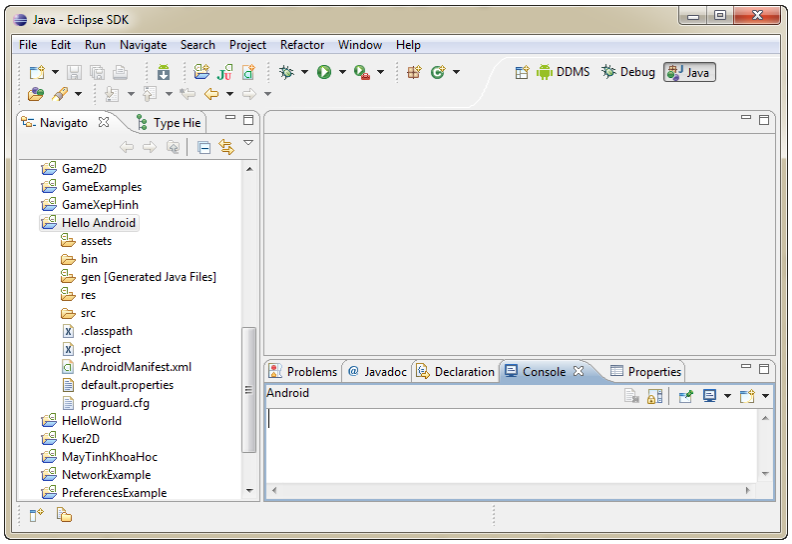
Package name: niit.android

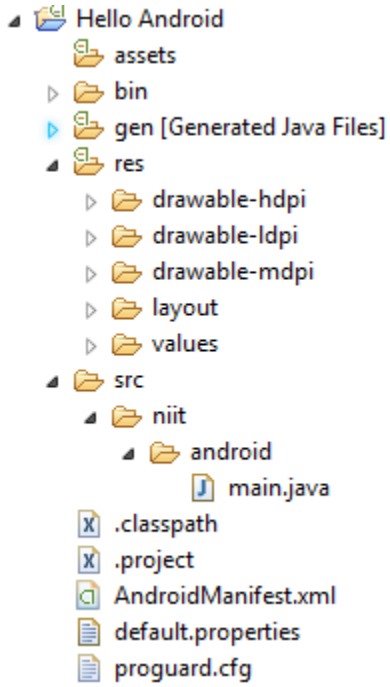
Create Activity: main





Sau khi tạo project ta sẽ có được một cấu trúc như sau:





Trong đó có các tệp tin và thư mục chính sau đây:

AndroidManifest.xml: file XML khai báo các thành phần có trong ứng dụng (activities, services,...).

build.xml: Một tệp chứa mã script Ant (ant.apache.com) để biên dịch và cài đặt ứng dụng lên máy.

default.properties: Tệp property tạo bởi script Ant trên.

bin/ : Thư mục chứa ứng dụng sau khi được biên dịch.

bin/classes/ : Thư mục chứa tệp tin code java đã thông dịch.

bin/classes.dex : Thư mục chứa các tệp tin có khả năng thực thi tạo bởi các lớp Java.

bin/yourapp.apk : Tệp tin cài đặt và thực thi.

bin/yourapp-debug.apk hay **bin/yourapp-unsigned.apk** : Thư mục chứa các tệp tin thực thi dùng để debug.

libs/ : Thư mục chứa các tệp JAR sử dụng trong ứng dụng(thư viện của hãng thứ ba).

src/ : Thư mục chứa mã nguồn Java của ứng dụng.

res/ : Thư mục chứa các tài nguyên của ứng dụng, như các icons, tệp tin layout,...

res/drawable/ : Thư mục chứa file hình ảnh (PNG, JPEG,...).

res/layout/ : Thư mục chứa các tệp tin xml đánh giấu giao diện.

res/menu/ : Thư mục chứa các tệp tin xml đánh giấu menu của ứng dụng.

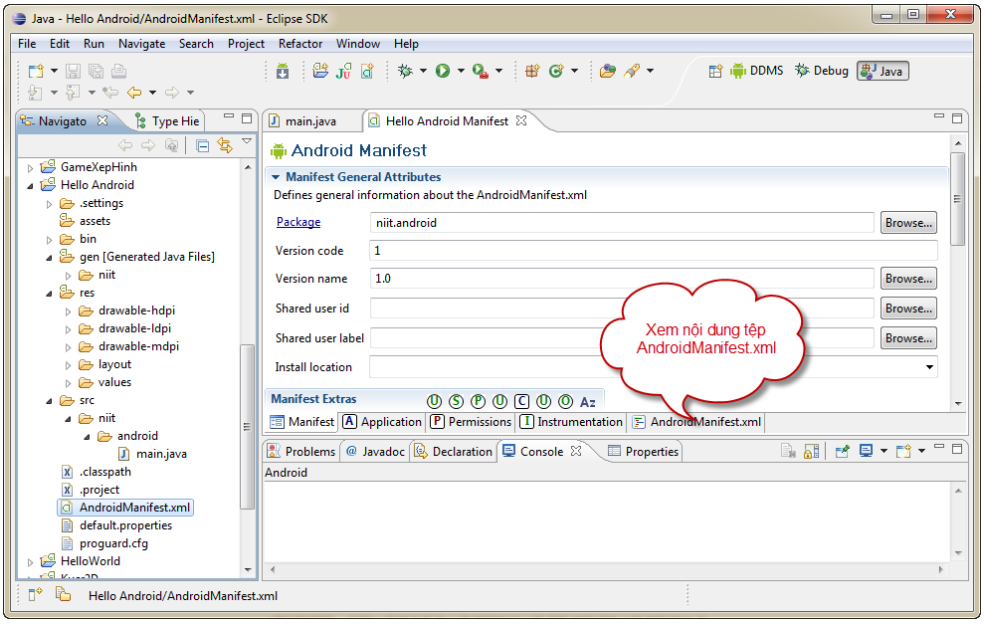
res/raw/ : chứa các file khác (CSV chứa thông tin account,...).

res/values/ : chứa các strings, dimensions,...

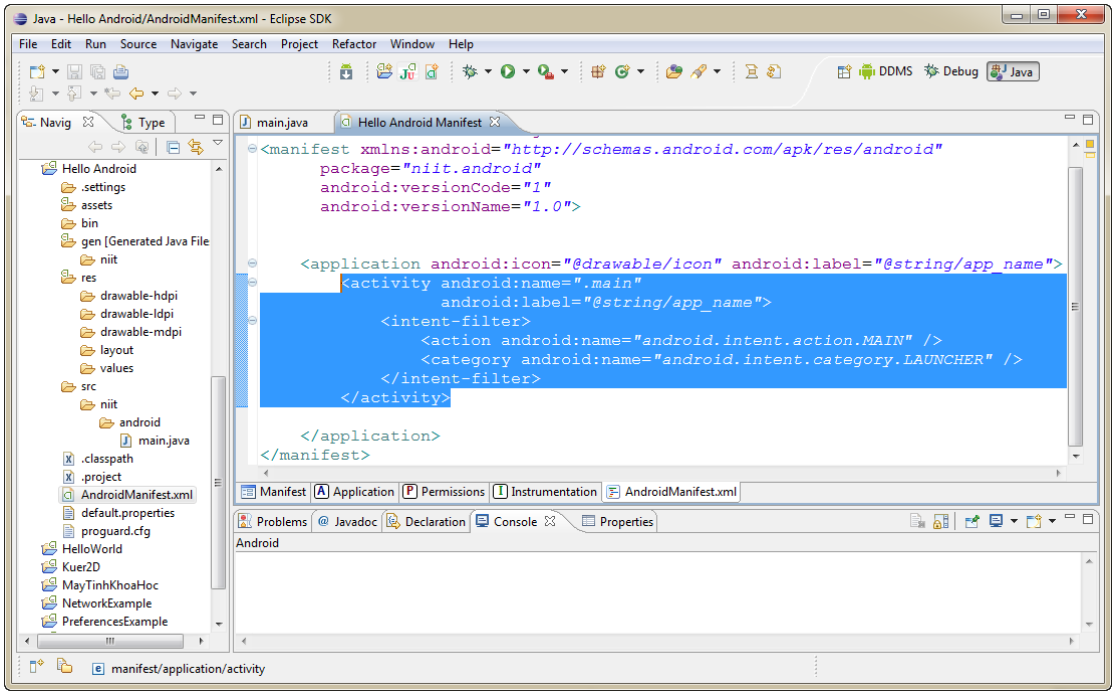
res/xml/ : chứa các file XML khác cần cho ứng dụng.

assets/ : nơi chứa các files tĩnh (static) được yêu cầu đi kèm với ứng dụng.

Mở tệp tin AndroidManifest.xml và chọn thẻ AndroidManifest.xml như hình dưới:



Sau khi chọn thẻ AndroidManifest.xml Eclipse sẽ cho phép ta xem nội dung của tệp dưới dạng một trình soạn thảo như hình sau :



Tệp AndroidManifest.xml có nội dung như sau:

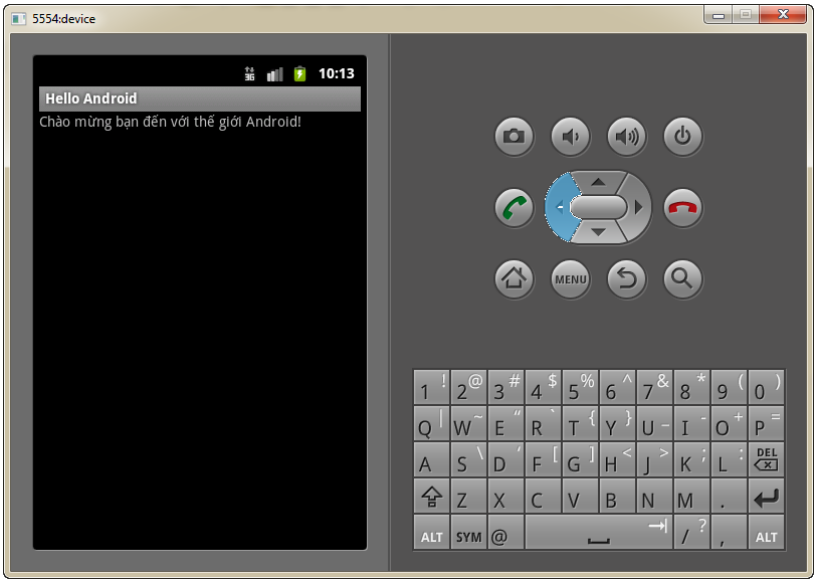
```
<?xml version="1.0" encoding="utf-8"?>
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
      package="niit.android"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon"
android:label="@string/app_name">
        <activity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

Trong đó khai báo một Activity tên là main sẽ là Activity đầu tiên được mở lên khi ứng dụng được khởi động bởi người sử dụng.

Mở tệp tin src/niit/android/main.java và sao chép phần code dưới đây vào tệp tin:

```
package niit.android;

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

public class main extends Activity {

      TextView tv;

      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);

            tv = new TextView(this);
            tv.setText("Chào mừng bạn đến với thế giới Android!");

            setContentView(tv);

      }
}
```

Thực thi chương trình bằng cách Menu Run>Run (hoặc nhấn Ctrl + F11) ta sẽ được kết quả như sau:



2.1 Phân tích ứng dụng "Hello Android"

Đầu tiên, qua file AndroidManifest.xml ta thấy có một Activity tên là main là Activity chính hay là Activity đầu tiên sẽ được mở lên khi ứng dụng được khởi động bởi người dùng. Tệp tin src/niit/android/main.java dùng để định nghĩa một cửa sổ Activity sử dụng trong ứng dụng "Hello Android". Ta sẽ tiến hành làm rỏ tệp tin này để có một cái nhìn tổng quan về cách định nghĩa một Activity trên Android. Trong file main.java kể trên ta có thể thấy được một số dòng lệnh dùng để import các thư viện cần thiết:

```
// Khai báo package chứa tệp tin.
package niit.android;

// Khai báo import các thư viện cần thiết.
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;
```

Ba lệnh trên dùng để import ba lớp được cung cấp bởi Android là Activity, Bundle và TextView. Ba lớp này sẽ được dùng ở bên dưới.

Trong tệp tin kể trên có một lớp tên là main mở rộng lại lớp Activity của Android. Bằng cách này ta đang định nghĩa một Activity mới cho ứng dụng. Trong lớp này ta bổ xung thêm một thuộc tính là tv kiểu TextView sẽ được dùng để hiển thị một đoạn văn bản.

```
public class main extends Activity {
     TextView tv;
```

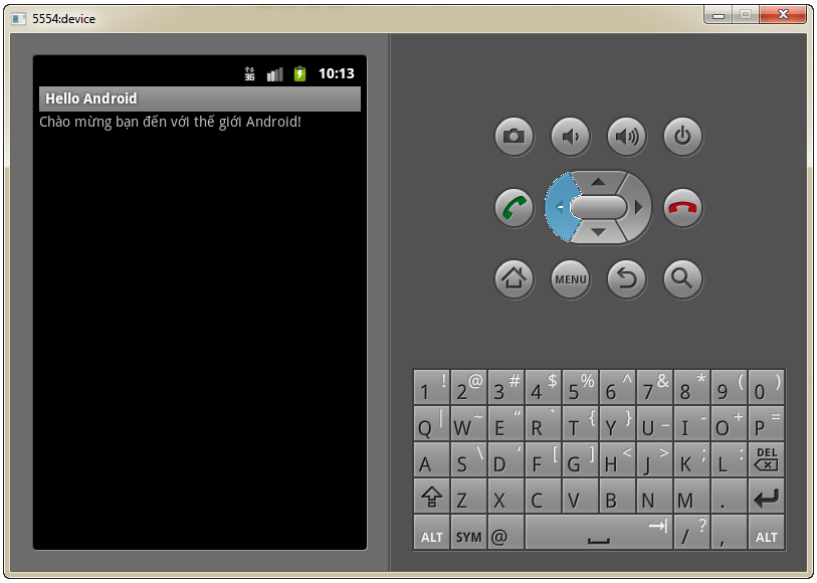
```
}
```

Trên lớp main ta override lại một phương thức tên là onCreate với nội dung:

```
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);

      tv = new TextView(this);
      tv.setText("Chào mừng bạn đến với thế giới Android!");

      setContentView(tv);

}
```

Phương thức này chính là phương thức được gọi khi Activity được tạo ra. Trên phương thức này ta tạo mới một đối tượng TextView và gán cho biến tv. Gọi phương thức tv.setText(…) để gán một đoạn văn bản cho đối tượng tv. Đối tượng tv là một đối tượng dùng để hiển thị một đoạn văn bản đến người dùng.

Phương thức setContentView() của lớp Activity có thể được dùng để gán giao diện người dùng cho cửa sổ Activity. Ở đây ta gọi phương thức setContentView(tv) để thiết lập giao diện của cửa sổ Activity hiện tại bằng đối tượng tv. Sau khi thực thi ta sẽ được một ứng dụng có giao diện như sau:

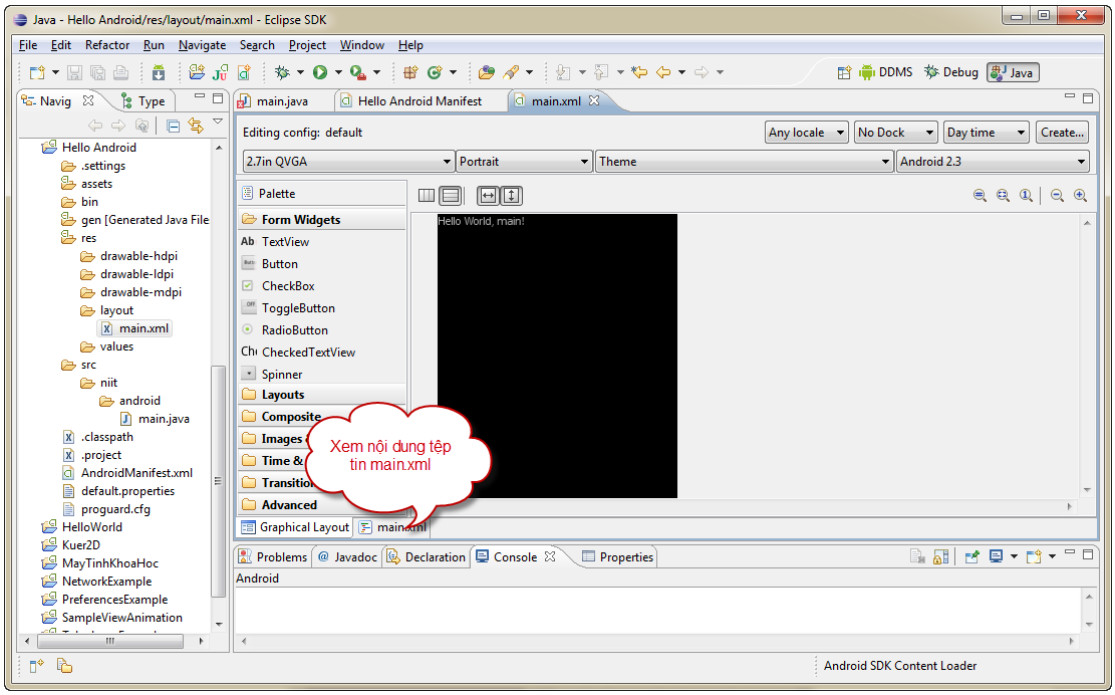


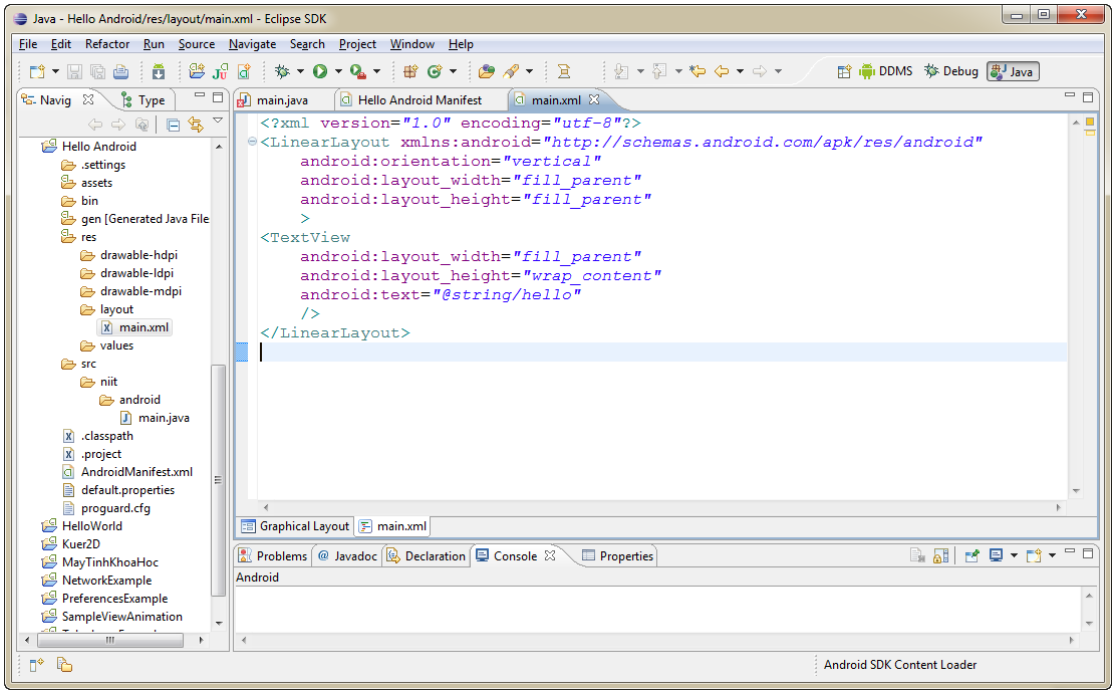
Toàn bộ màn hình ứng dụng chỉ có một đoạn văn bản duy nhất là "**Chào mừng bạn đến với thế giới Android**".

Với cách trên ta đã tạo mới một TextView, gán đoạn văn bản cho đối tượng mới tạo ra và đưa đối tượng TextView này lên màn hình ứng dụng điện thoại. Tất cả các bước đều thực hiện bằng các lệnh java. Cách xây dựng giao diện ứng dụng như trên rất linh hoạt, cần đối tượng đồ họa nào(Button,

TextView, EditText) thì bỏ lên. Tuy nhiên nó sẽ làm phần code java của chúng ta trở nên rất dài dòng và dễ bị rối trong một đống các câu lệnh tạo giao diện ứng dụng.

Ngoài cách ta vừa tiến hành ở trên thì còn một cách khác để tạo nên giao diện cho một ứng dụng. Cách thực hiện thứ hai này sẽ đơn giản và dễ thực hiện hơn nhưng không linh hoạt được như ở cách đầu tiên. Để minh họa cách sử dụng thứ hai ta tiến hành thay đổi project trên đôi chút. Mở tệp tin res/layout/main.xml trên Eclipse và nhấn thẻ main.xml như trong hình để xem nội dung của file main.xml.





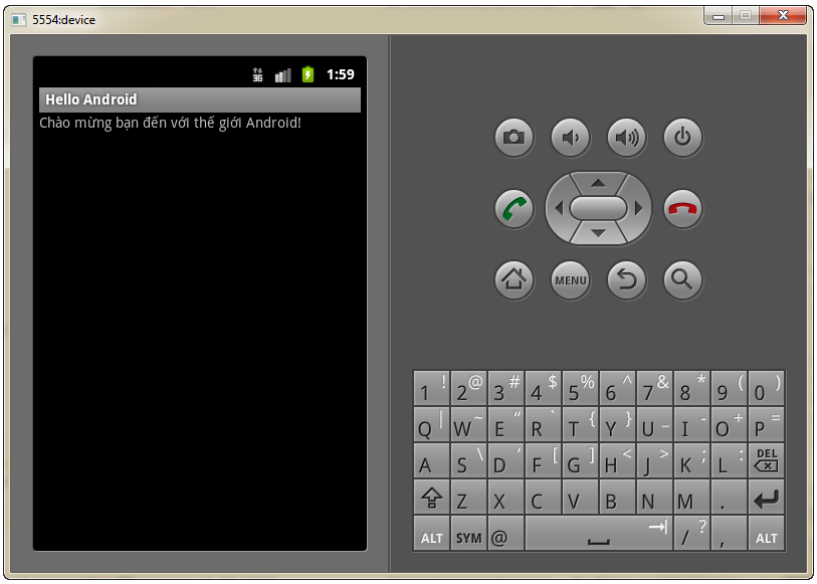
Thay đổi nội dung của tệp main.xml với nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
     <TextView
          android:id="@+id/tvHello"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="@string/hello"
     />
</LinearLayout>
```

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

```
package niit.android;

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

public class main extends Activity {
     TextView tv;
     @Override
     public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);

          setContentView(R.layout.main);

          tv = (TextView) findViewById(R.id.tvHello);
          tv.setText("Chào mừng bạn đến với thế giới Android!");
     }
}
```

Sau khi thực thi ta cũng nhận được một kết quả như ở cách thứ nhất:



Ở cách thứ hai này, ta không tiến hành tạo mới một đối tượng tv kiểu TextView mà thay vào đó ta tiến hành khai báo tệp tin main.xml. Tệp tin này sẽ chứa các thẻ xml đánh giấu các đối tượng đồ họa ta muốn dùng trong cửa sổ Activity cũng như các thuộc tính của các đối tượng đồ họa này. Các thuộc tính đó (Độ cao, độ rộng, màu sắc…) sẽ quyết định cách thức hiển thị nội dung.

Ta xem xét tệp main.xml ở trên:

```
<?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:id="@+id/tvHello"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="@string/hello"
     />
</LinearLayout>
```

Trong tệp tin này có một thẻ TextView có một thuộc tính android:id="@+id/tvHello". Đây là định danh của TextView kể trên. Nó sẽ được dùng để phân biệt các đối tượng đồ họa với nhau trong trường hợp giao diện ứng dụng có nhiều hơn một đối tượng.

Trên phương thức onCreate(…):

```
public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);

     setContentView(R.layout.main);
```

```
        tv = (TextView) findViewById(R.id.tvHello);
        tv.setText("Chào mừng bạn đến với thế giới Android!");
    }
```

ta sẽ dùng phương thức setContentView(R.layout.main) để gán giao diện cửa sổ Activity hiện tại bằng một giao diện đã khai báo trong tệp res/layout/main.xml bằng từ khóa R.layout.main. Từ khóa này tương đương với việc bạn chỉ định tệp tin đánh giấu giao diện đặt trong thư mục res/layout/ có tên là main.xml. Bằng cách này hệ thống sẽ tự tạo các đối tượng đã khai báo trong tệp main.xml và bỏ lên giao diện Activity. Do hệ thống đã tạo cho bạn các đối tượng đồ họa (TextView) dùng để hiển thị một nội dung nào đó nên ta không cần tạo mới một đối tượng TextView và gán cho tv mà ta chỉ đến đối tượng mà hệ thống đã tự động tạo ra bằng phương thức tv = (TextView) findViewById(R.id.tvHello);. Phương thức findViewById(R.id.tvHello) sẽ tìm trong các đối tượng trên Activity hiện tại xem đối tượng nào có id là tvHello, lấy đối tượng đó về và gán cho biến tv. Khi bạn khai báo trong thẻ TextView có thuộc tính android:id="@+id/tvHello" thì trên code java bạn có thể dùng biến R.id.tvHello Android sẽ tự hiểu giá trị của nó là "@+id/tvHello".

Lưu ý đọc thêm: Khi bạn khai báo một thẻ TextView có thuộc tính là android:id="@+id/tvHello" thì Eclipse tự phát sinh ra một file R.java trong đó có một lớp tĩnh là id có một thuộc tính int tvHello có một giá trị nào đó.

Trên đây là hai cách tạo một giao diện ứng dụng Android:

- Cách 01: Tạo từng đối tượng đồ họa và đưa lên cửa sổ Activity.
- Cách 02: Khai báo các đối tượng đồ họa trên tệp tin xml và nạp lên cửa sổ Activity.

Cách một thì linh hoạt nhưng sẽ tốn nhiều công sức để tự tạo từng đối tượng đồ họa và đưa lên màn hình ứng dụng. Cách thứ hai thì đơn giản và dễ thực hiện hơn nhưng lại không thể linh hoạt thay đổi theo tình huống sử dụng cụ thể. Trên thực tế ta sẽ chọn lựa các phù hợp với tình huống sử dụng. Nếu giao diện cố định, ít thay đổi thì ta có thể dùng cách 02. Nếu giao diện cần linh hoạt thì ta có thể sử dụng cách 01. Tuy nhiên ta vẫn có thể sử dụng cách 02 kết hợp với cách 01 để thêm, xóa một số đối tượng đồ họa khi cần thiết.

3 Activity

Là thành phần tối quan trọng của bất kỳ một ứng dụng Android nào. Thuật ngữ Activity chỉ một việc mà người dùng có thể thực hiện trong một ứng dụng Android. Do gần như mọi activity đều tương tác với người dùng, lớp Activity đảm nhận việc tạo ra một cửa sổ (window) để người lập trình đặt lên đó một giao diện UI với setContentView(View). Một activity có thể mang nhiều dạng khác nhau: Một cửa sổ toàn màn hình (full screen window), một cửa sổ floating (với windowsIsFloating) hay nằm lồng bên trong 1 activity khác (với ActivityGroup).

Để có thể sử dụng trong ứng dụng, mọi activity đều phải được khai báo trong tệp AndroidManifest.xml với một thẻ <activity> như ví dụ sau:

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

Vòng đời của một Activity

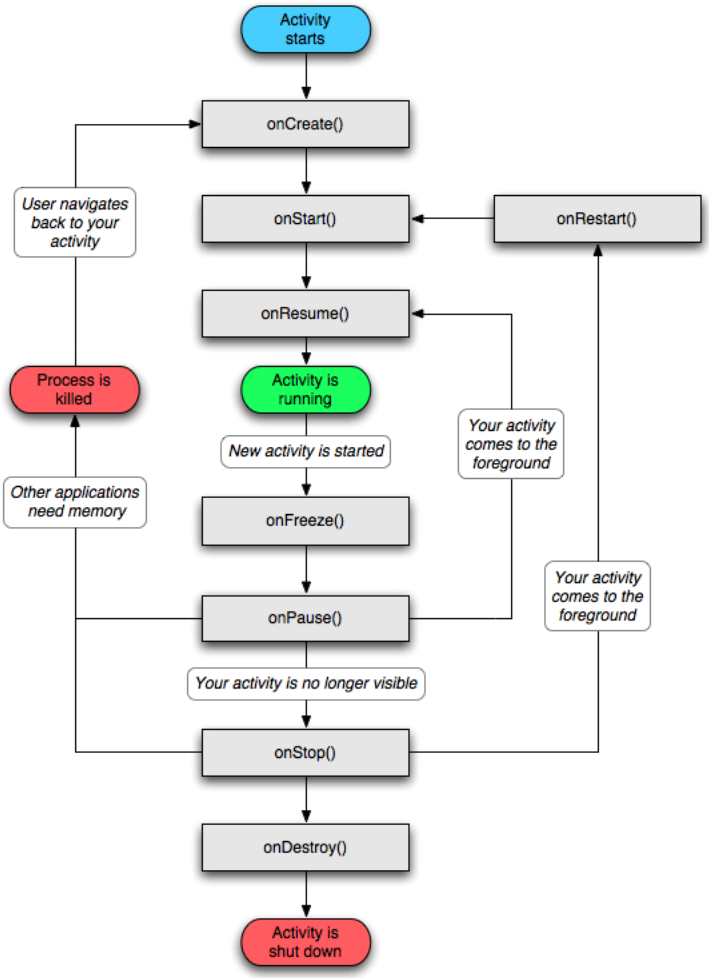
Các Activity được quản lí trong một stack chứa activity– (Cơ chế vào trước ra sau):

- Khi ứng dụng được mở lên thi activity chính sẽ được tạo ra, nó sẽ được thêm vào stack.
- Khi một activity mới được khởi tạo, nó sẽ được đặt lên trên cùng của stack. Lúc này chỉ có duy nhất Activity trên cùng là hiển thị nội dung đến người dùng. Tất cả các Activity còn lại đều chuyển về trạng thái dừng hoạt động.
- Khi một Activity bị đóng nó sẽ bị loại khỏi stack. Lúc này Activity nằm dưới đó sẽ chuyển từ trạng thái tạm dừng sang trạng thái hoạt động.

Một Activity có bốn trạng thái:

- **Active** hay **Running**: Activity đang chạy trên màn hình.
- **Paused**: Khi một Activity mất focus nhưng vẫn đang chạy trên màn hình (Activity bị một activity trong suốt(transparent) hoặc một Activity không chiếm toàn bộ màn hình thiết bị (non-full-sized) đè lên). Tuy vẫn còn tồn tại, nhưng các "paused activity" này sẽ bị hệ thống bắt chấm dứt khi thiếu bộ nhớ trầm trọng.
- **Stopped**: Khi một activity bị che khuất hoàn toàn bởi một activity khác. Tuy vẫn tồn tại, nhưng các "stopped activity" này sẽ thường xuyên bị hệ thống bắt chấm dứt để dành bộ nhớ cho các công việc khác.
- **Killed** hay **Shutdown**: Khi một activity đang **Paused** hay **Stopped**, hệ thống sẽ xóa activity ấy ra khỏi bộ nhớ.

Toàn bộ trạng thái của một Activity được thể hiện qua sơ đồ sau:



Dựa vào lược đồ trên, thấy được có ba vòng lặp quan trọng sau:

- **Vòng đời toàn diện (Entire Lifetime)**: Diễn ra từ lần gọi **onCreate(Bundle)** đầu tiên và kéo dài tới lần gọi **onDestroy()** cuối cùng.
- **Vòng đời thấy được (Visible Lifetime)**: Diễn ra từ khi gọi **onStart()** và kéo dài tới khi gọi **onStop()**. Ở vòng đời này, activity được hiển thị trên màn hình mặc dù có thể nó không thể tương tác với người dùng (Activity có thể bị đè bởi một Activity trong suốt hoặc một Activity không chiếm toàn bộ màn hình thiết bị(non-full-sized)). Giữa hai phương thức này cần giữ lại các tài nguyên dùng để hiển thị nội dung lên màn hình Activity. VD: Bạn có thể đăng kí (register) một BroadcastReceiver để theo dõi "những thay đổi" có ảnh hưởng đến phần giao diện của bạn trong phương thức **onStart()** và hủy đăng kí (unregister) BroadcastReceiver trong phương thức **onStop()**. Các phương thức **onStart()** và **onStop()** có thể được gọi nhiều lần.

- **Vòng đời trên nền (Foreground Lifetime)**: Diễn ra từ khi gọi **onResume()** và kéo dài tới khi gọi **onPause()**. Ở vòng đời này, activity nằm trên mọi activity khác và tương tác được với người dùng. Một activity có thể liên tục thay đổi giữa hai trạng thái **Paused** và **Resumed** (VD: Chẳng hạn khi thiết bị sleep).

Để xây dựng một Activity ta định nghĩa một lớp với dạng sau:

```
package niit.android;

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

public class ActivityExample extends Activity{

	@Override
	protected void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
	}
	@Override
	protected void onStart() {
		super.onStart();
	}
	@Override
	protected void onRestart() {
		super.onRestart();
	}
	@Override
	protected void onResume() {
		super.onResume();
	}
	@Override
	protected void onPause() {
		super.onPause();
	}
	@Override
	protected void onStop() {
		super.onStop();
	}
	@Override
	protected void onDestroy() {
		super.onDestroy();
	}
}
```

Có 2 phương thức mà gần như mọi lớp con của Activity đều phải hiện thực:

- **onCreate(Bundle)** - Nơi khởi tạo activity. Quan trọng hơn, đây chính người lập trình gọi setContentView(int) để gán giao diện cho Activity. Ngoài ra ở đây ta còn sử dụng

phương thức findViewById(int) giúp gọi về các đối tượng đồ họa (Button, TextView…) đã được hệ thống tạo tự động theo khai báo trong tệp tin xml để có thể sử dụng sau này. (Xen lại ví dụ: "Hello Android").

- **onPause()** - Nơi giải quyết sự kiện người dùng rời khỏi activity. Mọi dữ liệu được người dùng tạo ra tới thời điểm này cần phải được lưu trữ lại.

Lưu ý: Bổ xung khai báo trong AndroidManifest.xml bằng cách bổ xung thẻ <activity> như sau:

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

Thẻ Activity trên có một thuộc tính duy nhất là android:name=".ActivityExample". Trong đó giá trị của nó sẽ là ".TênActivityCầnKhaiBáo".

4 Cơ chế sử lý sự kiện trên Android

Tạo Project với các thông số:

- Project name: Action listenner Example.
- Build target: Android 2.3.3.
- Application name: Demo Action listenner
- Package name: niit.android
- Create Activity: main

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

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      >
      <TextView
            android:id="@+id/textView"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="Hello"
            />
      <EditText
            android:id="@+id/editText"
            android:layout_height="wrap_content"
            android:layout_width="match_parent">
      </EditText>
      <Button
            android:id="@+id/button"
            android:text="Button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
      </Button>
</LinearLayout>
```

Tiếp tục hay đổi nội dung tệp tin main.java với nội dung:

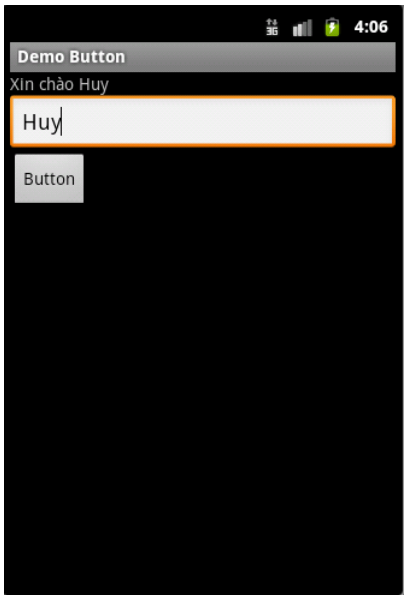
```
package niit.android;

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

public class main extends Activity implements OnClickListener {

      TextView textView;
      EditText  editText;
      Button    button;

      @Override
      public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        textView = (TextView)findViewById(R.id.textView);
        editText = (EditText)findViewById(R.id.editText);
        button = (Button)findViewById(R.id.button);

        button.setOnClickListener(this);
    }
    @Override
    public void onClick(View v) {
        String chuoiTen =   editText.getText().toString();
        textView.setText("Xin chào " + chuoiTen);
    }
}
```

Activity main có ba thuộc tính:

- TextView textView;
- EditText editText;
- Button button;

Trước tiên ta cũng gán textView, editText và button bằng đối tượng lấy về từ phương thức findViewById(…). Sau khi gán giá trị cho đối tượng button ta gọi phương thức setOnClickListener(…) để chỉ định đối tượng sẽ sử lý sự kiện người dùng nhấn lên button. Bằng cách sử dụng từ khóa this, ta đã chỉ định đối tượng sẽ lắng nghe sự kiện người dùng nhấn lên button là đối tượng main hiện tại. Một đối tượng muốn được giao nhiệm vụ sử lý sự kiện người dùng nhấn lên một đối tượng đồ họa (Button, TextView…) nào đó thì đối tượng đó phải hiện thực interface OnClickListener và cài đặt phương thức onClick như trong phần mã lệnh phía trên. Khi người dùng nhấn lên button thì hệ thống sẽ báo cho đối tượng main biết bằng cách gọi phương thức onClick(). Ở phương thức này, ta sẽ lấy phần văn bản người dùng nhập vào textView để in ra editText câu: “Xin chào …”. Thực thi chương trình, nhập một chuỗi vô EditText và nhấn Button ta sẽ được kết quả như sau:



Ngoài cách khai báo activity main cài đặt giao diện lập trình OnClickListener và chỉ định đối tượng sẽ được gọi khi người dùng nhấn vào đối tượng button ở trên ta còn có một cách thứ hai là:

```
btn.setOnClickListener(new OnClickListener() {
      @Override
      public void onClick(View arg0) {
            String chuoiTen =   editText.getText().toString();
            textView.setText("Xin chào " + chuoiTen);
      }
});
```

Đây là cách khai báo tắt một lớp nội. Ngoài ra còn có một các gán sự kiện khác nữa là gán thuộc tính trên tệp xml bằng cách sau:

```
<Button
      android:id="@+id/button"
      android:text="Button"
      android:onClick="tenPhuongThuc"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content">
</Button>
```

Trên thẻ Button có một thuộc tính:

```
android:onClick="tenPhuongThuc"
```

Thuộc tính này cho biết tên phương thức được khai báo trong activity (activity mà sử dụng tệp xml giao diện kể trên) sẽ được gọi khi người dùng nhấn lên đối tượng Button. Trên activity main bạn có thể định nghĩa một phương thức như sau:

```
public void tenPhuongThuc(View v) {
```

```
        String chuoiTen =   editText.getText().toString();
        textView.setText("Xin chào " + chuoiTen);
    }
```

Phương thức này phải có dạng: public void tenPhuongThuc(View v). Trong đó tham số truyền vào View v chính là đối tượng mà mình nhấn vào. Bằng cách này ta có thể gán thuộc tính android:onClick cho nhiều đối tượng Button với cùng một phương thức. Tham số View v sẽ được dùng để xác định người dùng nhấn lên đối tượng Button nào. Ta có thể ép kiểu đối tượng v thành kiểu Button để sử dụng các phương thức thuộc lớp Button. Cách này không những có tác dụng với Button mà còn có tác dụng với mọi loại View(Button, TextView, EditText…) và ViewGroup(LinearLayout, TableLayout…).

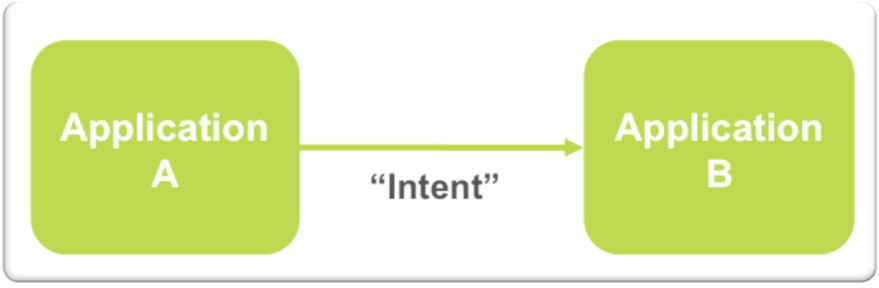
Ở phần này các bạn có thể tham khảo thêm bài tập thực hành đi kèm với giáo trình này để thấy rõ cách thức sử lý sự kiện của một đối tượng trên Android.

5 Intent

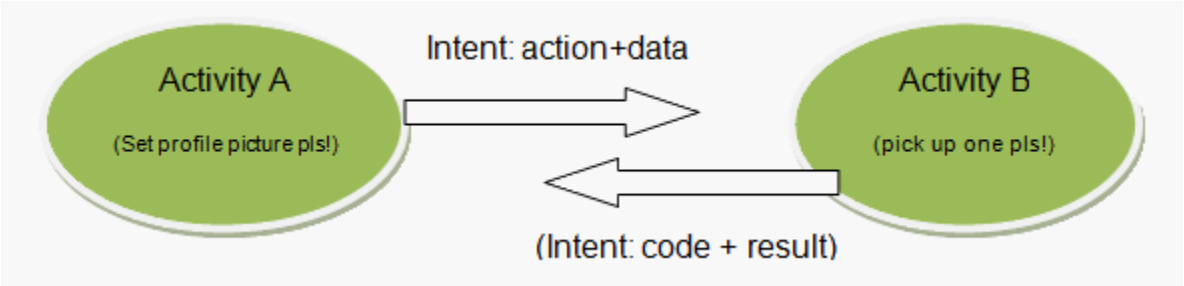
Intent là gì?

Khi Tim Berners phát minh ra giao thức Hypertext Transfer Protocol (HTTP), ông cũng đã phát minh ra một định dạng URLs chuẩn. Định dạng này là một hệ thống các động từ đi kèm các địa chỉ. Địa chỉ sẽ xác định nguồn tài nguyên như Web page, hình ảnh hay các server-side program. Động từ sẽ xác định cần phải làm cái gì với nguồn tài nguyên đó: GET để nhận dữ liệu về, POST để đưa dữ liệu cho nó để thực thi một công việc nào đó. Khái niệm Intent cũng tương tự, Intent là một mô tả trừu tượng của một hành động được thực thi. Nó đại diện cho một hành động đi kèm với một ngữ cảnh xác định. Với Intent thì có nhiều hành động và nhiều thành phần Android (Activity, Service, Broadcast receiver, Content Provider) dành cho Intent của Android hơn là so với HTTP verbs (POST, GET) và nguồn tài nguyên (hình ảnh, web page) của giao thức HTTP, tuy nhiên khái niệm vẫn tương tự nhau.

Intent được sử dụng với phương thức startActivity() để mở một Activity, dùng với broadcastIntent để gởi nó đến một BroadcastReceiver, và dùng với startService(Intent), bindService(Intent, ServiceConnection, int) để giao tiếp với các Service chạy dưới nền. Intent còn cung cấp một chức năng cho phép kết nối hai chương trình khác nhau trong lúc thực thi (Cung cấp khả năng cho phép hai chương trình khác nhau giao tiếp với nhau).



Trong đó chức năng quan trọng và được sử dụng nhiều nhất của một Intent là mở một Activity, nơi mà nó có thể được dùng như một vật kết nối các Activity lại với nhau giúp truyền thông tin giữa hai Activity khác nhau.



Trong hình vẽ trên Activity B chỉ trả về kết quả khi cần thiết.

VD : Giả sử Activity A nhắc người dùng chọn ảnh profile ; Activity B liệt kê các ảnh trong sdcard và cho phép người dùng chọn ảnh. Khi đó cặp "**code+result**" là cần thiết và có thể là "**CANCEL+null**" tức hành động chọn ảnh đã bị người dùng hủy bỏ hoặc "**OK+ảnh có mã là 20**" tức chọn ảnh 20.

Có thể nói Intent là một khái niệm then chốt và đặc trưng của nền tảng Android và lập trình ứng dụng trên nền tảng Android là lập trình dựa vào intetnt.

5.1 Intent chứa những dữ liệu gì ?

Intent về cơ bản là một cấu trúc dữ liệu, được mô tả trong lớp android.content.Intent. Trong đó có các thuộc tính:

Thuộc tính chính	Thuộc tính phụ
action: Tên của hành động cần thực hiện. Các hành động này có thể là hành động được Android định nghĩa (Mở Contact, gọi điện thoại, gởi email) sẵn hoặc hành động do người dùng định nghĩa.	**category**: Thông tin về nhóm của hành động cần thực thi.
data: Dữ liệu mà thành phần được gọi (Activity, Service, Broadcast receiver, Content provider). Dữ liệu được lưu trữ dưới định dạng Uri.	**type**: Định dạng kiểu dữ liệu gởi kèm.
	component: Thành phần (Activity, Service, Broadcast receiver, Content provider) nhận đối tượng Intent. Khi thuộc tính này được chỉ định thì mọi thuộc tính khác trở thành không bắt buộc.
	extras: Chứa các cặp (key,value) được gắn vào Intent.

Các action định nghĩa sẵn:

Built-in Standard Actions	
ACTION_MAIN	ACTION_ANSWER
ACTION_VIEW	ACTION_INSERT
ACTION_ATTACH_DATA	ACTION_DELETE
ACTION_EDIT	ACTION_RUN
ACTION_PICK	ACTION_SYNC
ACTION_CHOOSER	ACTION_PICK_ACTIVITY
ACTION_GET_CONTENT	ACTION_SEARCH
ACTION_DIAL	ACTION_WEB_SEARCH
ACTION_CALL	ACTION_FACTORY_TEST
ACTION_SEND	ACTION_SENDTO
Built-in Standard Broadcast Actions	
ACTION_TIME_TICK	ACTION_PACKAGE_RESTARTED
ACTION_TIME_CHANGED	ACTION_PACKAGE_DATA_CLEARED
ACTION_TIMEZONE_CHANGED	ACTION_UID_REMOVED
ACTION_BOOT_COMPLETED	ACTION_BATTERY_CHANGED
ACTION_PACKAGE_ADDED	ACTION_POWER_CONNECTED
ACTION_PACKAGE_CHANGED	ACTION_POWER_DISCONNECTED
ACTION_PACKAGE_REMOVED	ACTION_SHUTDOWN

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 một 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 Phone Dialer (mặc định là Phone Dialer của Android) và điền thông tin lấy từ địa chỉ chứa trong data URI.

ACTION_EDIT: Mở một Activity cho phép chỉnh sửa dữ liệu mà địa chỉ lấy từ data URI.

ACTION_SEND: Mở một Activity cho phép gửi dữ liệu lấy từ data URI, kiểu của dữ liệu xác định trong thuộc tính type.

ACTION_SENDTO: Mở một Activity cho phép gửi thông điệp tới địa chỉ lấy từ data URI.

ACTION_VIEW: Đây là action thông dụng nhất, khởi chạy activity thích hợp để hiển thị dữ liệu trong data URI.

ACTION_MAIN: Sử dụng để khởi chạy một Activity.

Đây là những hằng String đã được định nghĩa sẵn trong lớp Intent. Đi kèm với nó là các thành phần(Activity, Broadcast receiver, Service) hoặc ứng dụng được xây dựng sẵn sẽ được triệu gọi mỗi khi Intent tương ứng được gửi (tất nhiên khi được cung cấp đúng data).

VD:

Quay số điện thoại:

```
Intent dialIntent = new Intent(Intent.ACTION_DIAL,
Uri.parse("tel:123456"));
startActivity(dialIntent);
```

Mở danh sách contact:

```
Intent listContacts = new
Intent(Intent.ACTION_VIEW,Uri.parse("content://contacts/people/"
));
startActivity(listContacts);
```

Đến đây chắc bạn sẽ tự hỏi những chuỗi data trong hàm Uri.parse(data) có nghĩa là gì? Đó là định dạng dữ liệu tương ứng với mỗi action (chuẩn RFC 3986). Một khi bạn sử dụng hành động đã được dựng sẵn thì bạn phải cung cấp data cho nó theo định dạng này. Bảng dưới đây liệt kê một số định dạng và action tương ứng đã được định nghĩa sẵn:

Định dạng	Action	Mô tả
tel:phone_number	ACTION_VIEW	Mở Dial form (chưa gọi)
tel:phone_number	ACTION_CALL	Thực hiện gọi tới số phone
http://web_address https://web_address	ACTION_VIEW	Mở trình duyệt web với địa chỉ được cấp
"some_words" (string) http://web_address https://web_address	ACTION_WEB_SEARCH	Thực hiện search
sms://	ACTION_SENDTO	Gửi tin nhắn
geo:latitude,longitude geo:latitude,longitude?z=zoom geo:0,0?q=my+street+address geo:0,0?q=business+near+city	ACTION_VIEW	Mở ứng dụng Maps và chỉ tới vị trí được xác định

Tự định nghĩa action

Về nguyên tắc bạn có thể đặt tên action của một intent là bất cứ thứ gì theo chuẩn đặt tên thông thường, hay thậm chí dùng luôn hằng action đã định nghĩa sẵn như ACTION_VIEW (hay "android.intent.action.VIEW"). Cái tên VIEW thực chất chỉ là một tên gợi tả, bạn có thể dùng nó với mục đích thực hiện một activity để … gửi mail! Tuy nhiên điều đó rõ ràng là rất "ngớ ngẩn". Thay vào đó ta hãy dùng ACTION_SEND hay ACTION_SENDTO. Việc đặt tên action cho intent đúng tên gợi tả còn có một ý nghĩa khác đó là app của bạn có thể được triệu gọi từ một app khác. Ví dụ bạn viết một app có activity đáp ứng intent ACTION_SEND và để chia sẻ một bức ảnh lên trang web của bạn (giống

như ta làm với Facebook, Flickr etc.) Khi đó có thể ứng dụng của bạn sẽ là một lựa chọn chia sẻ ảnh của người dùng điện thoại.

Intent có 02 dạng chính

Intent tường minh - Explicit Intent: Xác định rỏ một thành phần(Activity, Broadcast Receiver, Service) (thông qua phương thức setComponent(ComponentName) hoặc setClass(Context, Class)) sẽ thực thi các hành động được đặc tả trong Intent. Thông thường thì những Intent này không chứa bất kỳ thông tin nào khác (như category, type) mà đơn giản chỉ là cách để ứng dụng mở các Activity khác bên trong một Activity.

Intent không tường minh - Implicit Intent: Không chỉ định một thành phần nào cả, thay vào đó, chúng sẽ chứa đủ thông tin để hệ thống có thể xác định component có sẵn nào là tốt nhất để thực thi hiệu quả cho Intent đó.

Khi sử dụng **Implicit intent**, do tính chất chuyên quyền của loại Intent này, ta cần phải biết phải làm gì với nó. Công việc này được đảm nhiệm bởi "**Tiến trình phân giải Intent**". Tiến trình này giúp chỉ định Intent đến một Actvity, BroadcastReceiver, hoặc Service (hoặc thỉnh thoảng có thể là 2 hay nhiều hơn một activity/receiver) để có thể xử lý các hành động được đặc tả trong Intent.

Bất cứ thành phần nào (Activity, Broadcast Receiver, Service) khi muốn sử dụng trong ứng dụng đều phải được đăng kí trong file AndroidManifest.xml. Trong đó cần định nghĩa một thẻ <intent-filter> cung cấp các thông tin để hệ thống có thể xác định được cái mà các thành phần này có thể xử lý được (những action mà thành phần này có thể thực hiện được).

Intent Filter

Một thành phần(Activity, Service, Broadcast receiver) cung cấp thông tin cho hệ điều hành biết loại đối tượng Intent mà nó có thể "sử lý" bằng cách khai báo một hoặc nhiều **Intent Filter**. Mỗi Intent Filter cung cấp các thông tin về "cái" mà thành phần này có thể xử lý(Hiển thị một contact, gọi điện thoại…). Các thành phần khai báo Intent Filter bằng cách sử dụng thẻ <intent-filter> đặt trong thẻ khai báo thành phần(<activity>, <service>, <receiver>) như ví dụ sau:

Ví dụ XX:

```
<activity android:name=".main"
          android:label="@string/app_name">
     <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category
     android:name="android.intent.category.LAUNCHER" />
     </intent-filter>
</activity>
```

Ví dụ trên khai báo một Activity với thẻ <intent-filter> chứa các thông tin dùng để chọn lựa Intent.

Với Intent tường minh thì thành phần nhận Intent đó đã được xác định cụ thể, tuy nhiên với các Intent không tường minh thì hệ thống sẽ tiến hành so sánh các thông tin phụ được khai báo trong Intent với các thành phần được khai báo trong tệp AndroidManifest.xml của tất cả các ứng dụng cài đặt trên thiết bị để tìm ra thành phần phù hợp.

Một Intent Filter có các thành phần chính sau:

- **action**: Tên hành động mà thành phần có thể thực thi.
- **type**:Kiểu dữ liệu thành phần có thể thực thi.
- **category**: Phân nhóm các thành phần.

Đối với những dữ liệu không phải là nội dung cụ thể (VD: URI) thì việc xem xét lựa chọn Intent phù hợp sẽ dựa vào lược đồ (Scheme) của dữ liệu được cung cấp (VD: http:// mailto: …)

Luật xác định thành phần phù hợp Intent

Để xác định một thành phần là phù hợp với một Intent hay không hệ thống sẽ tiến hành xem xét các nguyên tắc dưới đây:

- Trước tiên khi một Intent được gởi đi, Android sẽ tìm kiếm những thành phần (Activity, BroadcastReceiver, Service) có action-name phù hợp với Intent. Nếu có thành phần phù hợp thì Android sẽ mở thành phần đó lên để thực thi các hành động theo yêu cầu. Ngược lại nếu có nhiều hơn một thành phần có action-name phù hợp thì Android sẽ yêu cầu người dùng chọn một thành phần.
- Ngược lại nếu không có thành phần nào phù hợp thì Android sẽ tiến hành xem xét kiểu dự liệu của Intent cung cấp xem có thành phần nào có đủ năng lực để sử lý kiểu dữ liệu đó không. Nếu không được Android sẽ tiến hành xem xét scheme của dữ liệu đó để tìm kiếm thành phần phù hợp. Nếu vẫn không tìm được thành phù hợp Android sẽ tiến hành xem xét các thành phần có cùng category với category được xác định trong đối tượng Intent để chọn thành phần.

Ví dụ XX: Khai báo Activity chính sẽ được mở khi một ứng dụng được khởi chạy:

```
<activity android:name=".main"
          android:label="@string/app_name">
     <intent-filter>
          <action android:name="android.intent.action.MAIN" />
          <category
     android:name="android.intent.category.LAUNCHER" />
     </intent-filter>
</activity>
```

VD trên khai báo một Activity tên là main. Intent-Filter của Activity này có các thuộc tính:

- <action android:name="android.intent.action.MAIN"/>: Khai báo Activity main của thể thực thi được một hành động là mở một Activity.
- <category android:name="android.intent.category.LAUNCHER"/>: Khai báo Activity main thuộc nhóm các Activity được mở ra khi ứng dụng được chạy bởi người dùng.

Bằng cách khai báo như trên Activity tên main sẽ là Activity đầu tiên (Hay còn gọi là Activity chính) sẽ được mở ra khi ứng dụng được chạy bởi người dùng.

5.2 Sử dụng Intent như thế nào?

Các phương thức để khởi động các thành phần bằng cách sử dụng Intent:

Tên hàm	Mô tả
startActivity(intent) ←android.app.Activity	Thực thi activity như mô tả trong intent(không lấy kết quả trả về)
startActivityForResult(intent) ←android.app.Activity	Thực thi activity như mô tả trong intent(có lấy kết quả trả về)
sendBroadcast(intent) ←android.content.ContextWrapper	Phát tán intent tới bất kỳ thành phần BroadcastReceiver nào
startService(intent) ←android.content.ContextWrapper	Chạy một service
bindService(intent,ServiceConnection,int) ←android.content.ContextWrapper	Bind service

5.2.1 Intent tường minh thực thi Activity

Như đã trình bày ở phần trên, intent có thể dùng thuộc tính phụ để chỉ định đích danh tên Activity sẽ được mở. Để thực hiện điều này, lớp Intent cung cấp các phương thức setComponent(ComponentName) và setClass(Context, Class) và setClassName(Context, String) setClassName(String, String). Cách gọi này chỉ có thể được dùng để gọi các Activities trong cùng một ứng dụng.

Ví dụ:

```
     Intent intent = new Intent();
     intent.setClassName("ten_package",
"ten_lop_activity_ben_trong_package");
     startActivity(intent);
```

5.2.2 Intent không tường minh thực thi Activity

Trong trường hợp này intent không chỉ định một lớp cụ thể mà thay vào đó hệ thống dùng các dữ liệu khác (action, data, type, etc.) và quyết định xem thành phần nào (Activity, Service, Broadcast receiver) của ứng dụng nào sẽ thích hợp để đáp ứng intent đó.

Như đã nhắc đến ở phần trên, mọi thành phần của một ứng dụng Android phải được khai báo trong tệp AndroidManifest.xml. Trong đó thông tin action và category của bất kì thành phần nào(Activity, Service, Broadcast receiver, Content Provider) được khai báo trong thẻ intent-filter (Tất nhiên nếu chúng ta muốn gọi một hành động được thực thi bởi một thành phần định sẵn thì ta không cần quan tâm đến việc khai báo này).

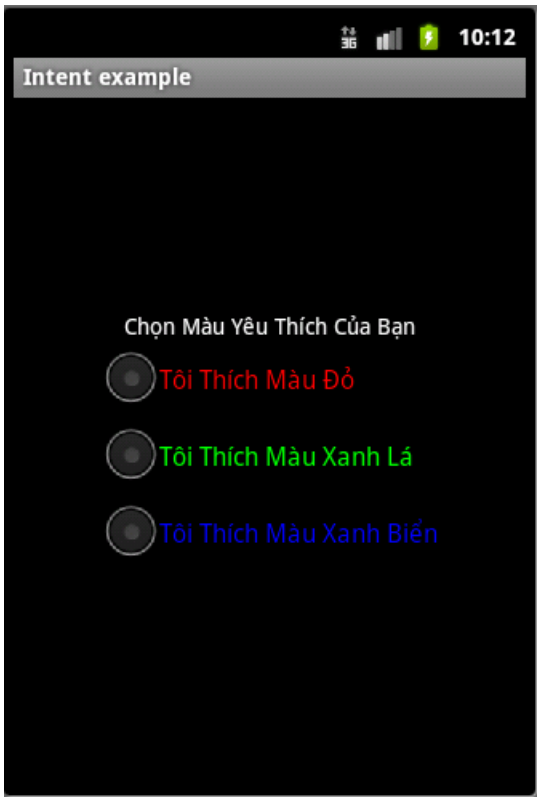
Ví dụ XX:

```
<activity android:name=".ActivityExample">
     <intent-filter>
          <action android:name="action_name"/>
          <category android:name="category_name"/>
          <data
               android:mimeType="video/*"
               android:scheme="http"/>
     </intent-filter>
</activity>
```

Ví dụ trên khai báo một Activity tên là ActivityExample. Intent-Filter của Activity này có các thuộc tính: <data android:mimeType="video/*" android:scheme="http"/> xác định ActivityExample có thể sử lý được một đối tượng Intent có đính kèm dữ liệu là tệp video với đường dẫn kiểu đường dẫn http.

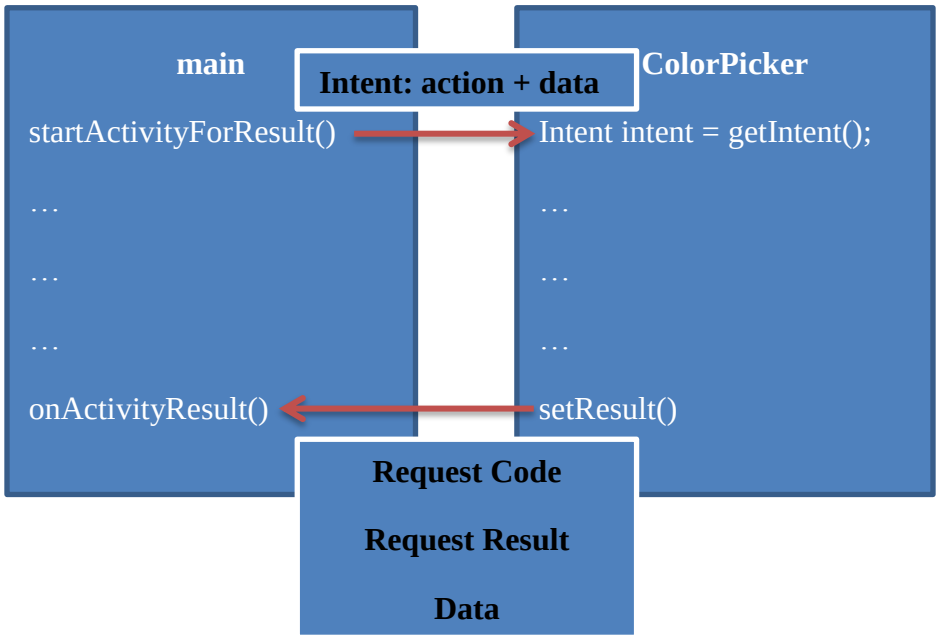
5.2.3 Truyền nhận thông tin giữa các Activity sử dụng đối tượng intent

Đầu tiên ta có hai Activity tên là main và ColorPicker lần lượt có giao diện như sau:



Activity main sẽ là Activity chính của chương trình, trên đó hiển thị ba màu dòng RED, GREEN và BLUE. Khi người dùng nhấp và nút "Lựa chọn màu yêu thích" thì chương trình mở lên cửa sổ Activity ColorPicker cho phép người dùng check chọn một trong ba màu. Sau khi check chọn một màu thì ứng dụng tự tắt Activity ColorPicker và to màu một trong ba dòng RED, GREEN hoặc BLUE như hình sau:

Trong đó việc truyền nhận thông tin giữa hai Activity được thể hiện qua mô hình sau:



Trên phương thức main ta tạo một đối tượng Intent chứa thông tin dùng để mở Activity ColorPicker và sử dụng phương thức starActivityForResult() để mở Activity ColorPicker. Khi Activity ColorPicker được mở lên, nếu main có gởi dữ liệu cho ColorPicker thì trên phương thức onCreate() của ColorPicker có thể dùng phương thức getIntent() để lấy về đối tượng Intent dùng để mở nó lên và lấy

các dữ liệu đính kèm trong đó. Sau khi thao tác xong, ColorPicker sẽ tạo một đối tượng Intent và trả về cho main. Activity main sẽ được thông báo nhận đối tượng Intent trả về thông qua phương thức onActivityResult(). Khi main mở Activity ColorPicker cần nhận về kết nên ta phải cung cấp một thông số là "request code". Khi ColorPicker được tắt đi thì nó sẽ trả về một đối tượng Intent chứa 03 thành phần sau:

1. "Request code" đã dùng để mở nó lên.
2. "Request result": Thuộc tính này có hai giá trị là RESULT_OK và RESULT_CANCEL tương ứng với việc người dùng đã hoàn tất công việc hoặc đã hủy công việc thực hiện ở ColorPicker.
3. Data: Dữ liệu trả về.

Tiến hành tạo mới một project Android với các thông số sau:

Project name: Intent example

Build target: Android 2.3

Application name: Intent example

Package name: niit.android.intentexample

Create Activity: main

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

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center"
    android:padding="8dp">
     <TextView
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:gravity="center"
          android:textColor="#ffffff"
          android:text="Màu Sắc Yêu Thích Của Bạn Là:"/>
     <TextView
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:gravity="center"
          android:textColor="#666666"
          android:textSize="72sp"
          android:textStyle="bold"
```

```
            android:text="RED"
            android:id="@+id/labelRed" />
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:textColor="#666666"
            android:textSize="72sp"
            android:textStyle="bold"
            android:text="GREEN"
            android:id="@+id/labelGreen" />
        <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:textColor="#666666"
            android:textSize="72sp"
            android:textStyle="bold"
            android:text="BLUE"
            android:id="@+id/labelBlue" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:id="@+id/pickbutton"
            android:onClick="buttonClickHandler"
            android:text="Lựa Chọn Màu Yêu Thích"/>
    </LinearLayout>
```

Phần mã xml ở trên tạo thành giao diện Activity main như hình ở trên. Trên giao diện kể trên có một Button có thuộc tính android:onClick="*buttonClickHandler*" cho biết khi người dùng nhấn lên nút nhấn đó thì hệ thống sẽ gọi phương thức buttonClickHanler() đặt trong Activity chứa giao diện đó.

Chỉnh sửa tệp main.java thành nội dung như sau:

```
package niit.android.intentexample;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.widget.TextView;

public class main extends Activity {
    private int color = Color.RED;

    public static final int COLOR_PICKER_CODE = 100;
```

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

	public void resetAllTextView() {
		TextView labelred, labelgreen, labelblue;
		labelred = (TextView)findViewById(R.id.labelRed);
		labelred.setTextColor(Color.parseColor("#666666"));
		labelgreen = (TextView)findViewById(R.id.labelGreen);
		labelgreen.setTextColor(Color.parseColor("#666666"));
		labelblue = (TextView)findViewById(R.id.labelBlue);
		labelblue.setTextColor(Color.parseColor("#666666"));
	}

	public void buttonClickHandler(View v) {
		Intent request = new Intent(this,
ColorPickerActivity.class);
		Bundle bundle = new Bundle();
		bundle.putInt("color", color);
		request.putExtras(bundle);
		startActivityForResult(request, COLOR_PICKER_CODE);
	}

	@Override
	protected void onActivityResult(int requestCode, int
resultCode, Intent data) {
		switch (requestCode) {
			case COLOR_PICKER_CODE:
				if (resultCode == RESULT_OK) {
					Bundle bundle = data.getExtras();
					color = bundle.getInt("color");
					setTextView();
				}
		}
	}

	private void setTextView() {
		int tvId = 0;
		resetAllTextView();
		switch (color) {
			case Color.RED:
				tvId = R.id.labelRed;
				break;
```

```
                case Color.GREEN:
                    tvId = R.id.labelGreen;
                    break;
                case Color.BLUE:
                    tvId = R.id.labelBlue;
                    break;
                default:
                    break;
            }
            TextView mycolor = (TextView)findViewById(tvId);
            mycolor.setTextColor(color);
        }
    }
```

Trong Activity main mới chỉnh sửa trên ta sẽ nạp lên nó phần giao diện đã được định nghĩa trong tệp main.xml:

```
setContentView(R.layout.main);
```

Sau khi thiết lập giao diện cho activity main ta gọi phương thức setTextView() để gián màu cho các TextView. Ở phương thức này ta sẽ tiến hành thay đổi màu của cả ba TextView RED, GREEN và BLUE trên activity main của chương trình thành màu #666666(Màu xám) bằng cách gọi phương thức resetAllTextView().

Ngoài ra ta còn định nghĩa thêm một phương thức buttonClickHanler() là phương thức sẽ được gọi khi người dùng nhấn vào nút "Lựa chọn màu sắc yêu thích". Trong phương thức này ta khai báo một đối tượng Intent:

```
Intent request = new Intent(this,ColorPickerActivity.class);
```

Bằng cách này ta đã tạo ra một đối tượng Intent tường minh chỉ định rỏ thành phần cần mở lên chính là activity tên là ColorPickerActivity. Sau đó ta tiến hành khởi tạo một đối tượng Bundle bundle và đặt vào trong đối tượng Bundle một biến kiểu int (private in color = Color.RED) với giá trị mặc định là Color.RED (Đây là một số int đại diện cho màu đỏ) cho từ khóa "color" và đặt đối tượng Bundle vào request:

```
Bundle bundle = new Bundle();
bundle.putInt("color", color);
request.putExtras(bundle);
```

Sau khi khởi tạo thành công một đối tượng Intent ta tiến hành gởi đối tượng đó đi bằng lệnh:

```
startActivityForResult(request, COLOR_PICKER_CODE);
```

Lệnh này sẽ gởi đi một đối tượng Intent dùng để mở một Activity và request code là **COLOR_PICKER_CODE**. Đây là một hằng tĩnh được khai báo bên trên:

```
public static final int COLOR_PICKER_CODE = 100;
```

Đây là một hằng kiểu int có giá trị là 100. Hằng tĩnh này sẽ được dùng để xác định kết quả được trả về bởi Activity ColorPickerActivity sau này.

Thêm một tệp picker.xml với nội dung như sau:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:gravity="center"
    android:padding="8dp">
     <TextView
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:gravity="center"
          android:textColor="#ffffff"
          android:text="Chọn Màu Yêu Thích Của Bạn"/>
    <RadioGroup
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:id="@+id/radiogroup">
     <RadioButton
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:textColor="#ff0000"
          android:id="@+id/radioRed"
          android:onClick="radioClickHandler"
          android:text="Tôi Thích Màu Đỏ"/>
     <RadioButton
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:textColor="#00ff00"
          android:id="@+id/radioGreen"
          android:onClick="radioClickHandler"
          android:text="Tôi Thích Màu Xanh Lá"/>
     <RadioButton
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"
          android:textColor="#0000ff"
          android:id="@+id/radioBlue"
          android:onClick="radioClickHandler"
          android:text="Tôi Thích Màu Xanh Biển"/>
    </RadioGroup>
</LinearLayout>
```

Phần mã xml trên đây sẽ tạo thành giao diện ColorPickerActivity như hình ở trên. Trong giao diện đó sẽ có 03 RadioButton cho phép chọn một trong ba màu đỏ, xanh lá và xanh biển. Ta đặt cả 03 RadioButton này trong một RadioGroup để đảm bảo tại một thời điểm chỉ có thể chọn một trong 03 màu sắc kể trên. Cả 03 RadioButton để có một thuộc tính là:

```
android:onClick="radioClickHandler"
```

Thuộc tính này cho biết tên phương thức sẽ được gọi khi người dùng click vào một trong các RadioButton kể trên thì phương thức radioClickHandler() trên Activity chứa giao diện trên sẽ được gọi thực thi.

Thêm tệp ColorPickerActivity.java với nội dung như sau:

```
package niit.android.intentexample;

import android.app.Activity;
import android.content.Intent;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;

public class ColorPickerActivity extends Activity {

     @Override
     protected void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.picker);
          Intent request = getIntent();
          Bundle bundle = request.getExtras();
          int color = bundle.getInt("color");
          int radioButtonId = 0;
          switch (color) {
               case Color.RED:
                    radioButtonId = R.id.radioRed;
               break;
               case Color.GREEN:
                    radioButtonId = R.id.radioGreen;
               break;
               case Color.BLUE:
                    radioButtonId = R.id.radioBlue;
               break;
               default:
               break;
          }
          RadioButton rBtn =
(RadioButton)findViewById(radioButtonId);
          rBtn.setChecked(true);
     }
```

```
    public void radioClickHandler(View v)
    {
        Intent answer = new Intent();
        Bundle bundle = new Bundle();
        if(v.getId()==R.id.radioBlue)
        {
            bundle.putInt("id", R.id.labelBlue);
            bundle.putInt("color", Color.BLUE);
        }
        else if(v.getId()==R.id.radioRed)
        {
            bundle.putInt("id", R.id.labelRed);
            bundle.putInt("color", Color.RED);
        }
        else
        {
            bundle.putInt("id", R.id.labelGreen);
            bundle.putInt("color", Color.GREEN);
        }
        answer.putExtras(bundle);
        setResult(RESULT_OK, answer);
        finish();
    }

}
```

Trong Activity ColorPickerActivity mới tạo ra ở trên ta nạp lên phần giao diện đã được định nghĩa trong tệp picker.xml:

```
setContentView(R.layout.picker);
```

Sau khi nạp giao diện cho ColorPickerActivity xong ta sẽ tiến hành lấy các dữ liệu được gởi đi bởi activity main. Đầu tiên ta gọi phương thức getIntent() để lấy về đối tượng Intent đã dùng để mở activity hiện tại lên:

```
Intent request = getIntent();
```

Sau đó lấy về đối tượng Bundle được đính kèm trong đối tượng Intent đã được gởi đi bởi main:

```
Bundle bundle = request.getExtras();
```

Và lấy ra biến "color":

```
int color = bundle.getInt("color");
int radioButtonId = 0;
```

Tùy thuộc biến color có giá trị là Color.RED, Color.GREEN hay Color.BLUE mà ta sẽ gán giá trị chi biến radioButtonId là R.id.radioRed, R.id.radioGreen hay R.id.radioBlue:

```
switch (color) {
    case Color.RED:
```

```
            radioButtonId = R.id.radioRed;
        break;
        case Color.GREEN:
            radioButtonId = R.id.radioGreen;
        break;
        case Color.BLUE:
            radioButtonId = R.id.radioBlue;
        break;
        default:
        break;
}
```

Cuối cùng tiến hành lấy về RadioButton có id là radioButtonId và chuyển nó thành trạng thái "Checked".

```
RadioButton rBtn = (RadioButton)findViewById(radioButtonId);
rBtn.setChecked(true);
```

Qua phần mã lệnh kể trên ta đã hiện thực phần chức năng cho phép activity main gởi đi một biến chứa mã màu (Mặc định là Color.RED), khi ColorPickerActivity nhận được mã màu do main gởi đi. Nó sẽ cập nhập RadioButton tương ứng.

Ngoài ra ColorPickerActivity có một phương thức:

```
public void radioClickHandler(View v)
```

Đây là phương thức sẽ được gọi khi người dùng nhấn lên một trong 03 RadioButton được định nghĩa trong tệp xml chứa giao diện picker.xml. Phương thức này nhận vào một tham số là View v. Khi người dùng nhấn chọn một trong 03 RadioButton thì phương thức radioClickHandler() sẽ được khởi chạy và truyền theo chính đối tượng RadioButton mà người dùng nhấn vào. Trên phương thức này ta sẽ có các khai báo:

```
Intent answer = new Intent();
Bundle bundle = new Bundle();
```

Ta khởi tạo hai đối tượng Intent answer và Bundle bundle. Đối tượng Intent answer là đối tượng sẽ được trả về cho activity main khi activity ColorPickerActivity được đóng lại. Đối tượng Bundle bundle là đối tượng chứa các dữ liệu cần trả về cho main.

Tùy theo người dùng nhấn vào RadioButton nào thì ta thêm vào đối tượng bundle hai số int có mã là "id" chứa khóa của một trong ba nhãn (RED, GREEN hoặc BLUE) và một số int có mã là "color" là giá trị của màu sắc cần cập nhật cho nhãn.

```
bundle.putInt("color", Color.GREEN);
```

Cuối cùng ta đưa đặt đối tượng bundle vào answer:

```
answer.putExtras(bundle);
```

Tiến hành gọi phương thức setResult() và truyền vào 02 thông số là:

- "Request result" là RESULT_OK để báo công việc thực hiện trên ColorPickerActivity đã hoàn tất, ngoài ra có thể gán "request result" là REQUEST_CANCEL.

- Đối tượng Intent answer chứa dữ liệu trả về.

```
setResult(RESULT_OK, answer);
```

Cuối cùng gọi phương thức finish() để đóng ColorPickerActivity:

```
finish();
```

Sau khi ColorPickerActivity đóng thì kết quả sẽ được trả về cho activity main qua việc gọi phương thức:

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data)
```

Phương thức này sẽ nhận vào 03 tham số:

- **int** requestCode: Request code đã dùng để mở ColorPickerActivity lên.
- **int** resultCode: RESULT_OK hoặc RESULT_CANCEL cho biết công việc đã hoàn tất hay bị hủy bởi người dùng.
- **Intent** data: Đối tượng Intent chứa dữ liệu được ColorPickerActivity trả về.

Trong phương thức này ta sẽ tùy theo dữ liệu trả về mà tiến hành cập nhật màu sắc của các nhãn bằng cách gọi lại phương thức setTextView() như đã trình bày ở trên.