



LẬP TRÌNH DI ĐỘNG

Bài 8: Intent & Broadcast Receiver



Nhắc lại bài trước

- Các điều khiển hữu ích: ProgressBar, ProgressDialog, AutoComplete TextView, TimePicker, TimePicker Dialog, DatePicker, DatePicker Dialog, ListView, Spinner, WebView
- Quá trình xây dựng giao diện:
 1. Thiết lập giao diện trong XML
 2. Cái này không dùng XML được thì viết trong onCreate
 3. Thiết lập dữ liệu cho điều khiển
 4. Viết các hàm xử lý sự kiện cho điều khiển
- Vài kiểu viết mã xử lý sự kiện



Nội dung

1. Intent

1. Intent filter
2. Intent tường minh vs ngầm định
3. Các thành phần của intent
4. Sử dụng intent để trao đổi dữ liệu

2. Broadcast Receiver

1. Vòng đời của boardcast receiver
2. Tự tạo một tín hiệu broadcast
3. Viết receiver xử lý tín hiệu broadcast

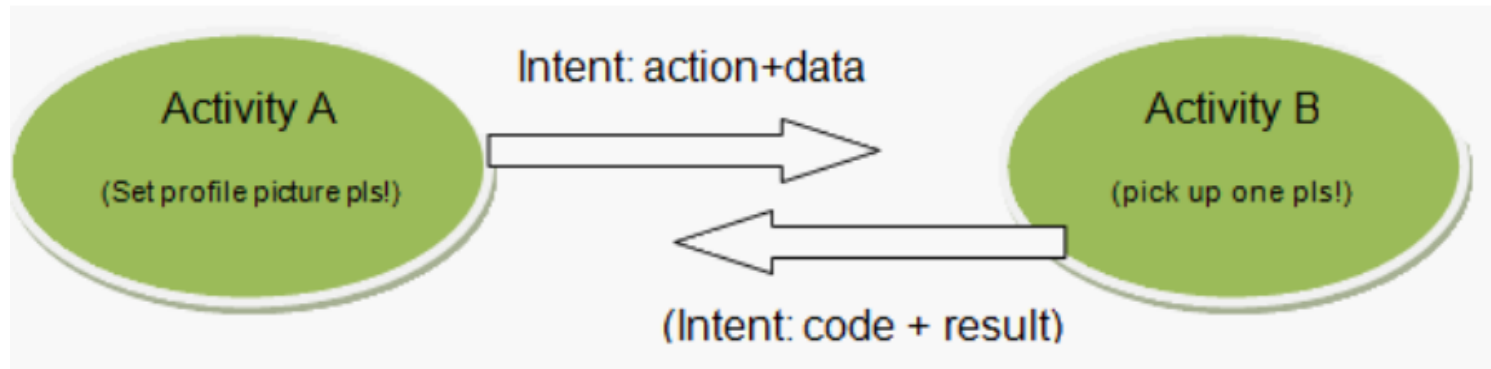


Phần 1

Intent

Intent & Intent Service

- **Intent** là chuẩn giao tiếp giữa các thành phần trong Android OS (activity, service, provider, receiver)
- **Intent giống một lá thư**: các thông tin cần thiết được đóng gói bên trong một intent (địa chỉ + nội dung)
- **Intent service** là dịch vụ hệ thống, vai trò như người đưa thư: chuyển intent tới thành phần nhận phù hợp nhất (chiều theo địa chỉ ghi trong intent)





Ví dụ sử dụng Intent

```
Intent x = new Intent(this, Login.class);  
x.putExtra("loginname", "abcxyz");  
startActivity(x);
```

**mở activity
Login với dữ
liệu gửi kèm
loginname là
abcxyz**

```
Intent i = new Intent(Intent.ACTION_VIEW);  
i.setData(Uri.parse("http://txnam.net"));  
startActivity(i);
```

mở trang web txnam.net

**mở activity quay số với
số 0912102165 điền sẵn**

```
Intent dialIntent = new  
Intent(Intent.ACTION_DIAL, Uri.parse("tel:0912102165"));  
startActivity(dialIntent);
```

Dùng Intent và nhận dữ liệu trả về

```
private static int TAKE_PICTURE = 1;
private Uri outputFileUri;
private void TakePhoto() {
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
    File file = new File(Environment.getExternalStorageDirectory(), "test.jpg");

    outputFileUri = Uri.fromFile(file);
    intent.putExtra(MediaStore.EXTRA_OUTPUT, outputFileUri);
    startActivityForResult(intent, TAKE_PICTURE);
}
```

bật ứng dụng camera chụp ảnh và trả về ảnh chụp

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

    if (requestCode == TAKE_PICTURE){
        ShowMessage(outputFileUri.toString());
    }
}
```



Phần 1.1

Intent filter



Intent filter (bộ lọc intent)

- Activity, Service và Broadcast receiver sử dụng intent filter để thông báo cho hệ thống biết các dạng intent mà nó có thể xử lý
- Phân giải intent (intent resolution): khi nhận được một intent, hệ thống tiến hành chọn activity phù hợp nhất với intent đó theo ưu tiên sau
 - Action trong intent
 - Chuỗi tham số (URI trong phần data)
 - Category của intent

Intent filter (bộ lọc intent)

```
<activity android:name=".Dialer"
          android:label="@string/app_name">
  <intent-filter android:priority="100" >
    <action android:name="android.intent.action.DIAL" />
    <category android:name="android.intent.category.DEFAULT" />
    <data android:scheme="tel"/>
  </intent-filter>
</activity>
```

Đoạn mã XML trên đăng kí với hệ thống một activity “lắng nghe” việc người dùng muốn thực hiện cuộc gọi bằng cách chỉ định thuộc tính **action** trong intent-filter





Phần 1.2

Intent tường minh vs ngầm định



Intent tường minh (explicit)

- Sử dụng thành phần component để chỉ định rõ đối tượng sẽ thực thi
- Sử dụng phương thức:
 - `setComponent(ComponentName)`
 - `setClass(Context, Class)`
 - `setClassName(Context, String)`
 - `setClassName(string, string)`
- Chỉ được dùng để gọi các activity trong cùng một ứng dụng
- Dữ liệu trao đổi nên chuyển vào phần extras



Intent tường minh (explicit)

```
Intent intent = new Intent();  
  
intent.setClassName("ten_package", "ten_class");  
  
// chuẩn bị dữ liệu trước khi gửi  
  
...  
  
startActivity(intent);
```

Hoặc

```
Intent i = new Intent(this, Activity2.class);  
  
// chuẩn bị dữ liệu trước khi gửi  
  
...  
  
startActivity(intent);
```



Intent ngầm định (implicit)

- Dùng các thành phần action, category,...
- Hệ thống tự động xác định đối tượng phù hợp nhất để đáp ứng với Intent đó (theo nguyên tắc “phân giải intent” đã trình bày ở slide 9)
- Dùng để giao tiếp với các dịch vụ hệ thống hoặc dịch vụ do bên thứ ba cung cấp:
 - Gọi activity: `startActivity` / `startActivityForResult`
 - Gọi service: `startService` / `bindService`
 - Gửi broadcast: `sendBroadcast` / `sendOrderedBroadcast` / `setStickyBroadcast`

Intent ngầm định (implicit)

Một số trường hợp sử dụng implicit intent

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

Các action được định nghĩa sẵn

Built-in Standard Actions	
<u>ACTION MAIN</u> <u>ACTION VIEW</u> <u>ACTION ATTACH DATA</u> <u>ACTION EDIT</u> <u>ACTION PICK</u> <u>ACTION CHOOSER</u> <u>ACTION GET CONTENT</u> <u>ACTION DIAL</u> <u>ACTION CALL</u> <u>ACTION SEND</u>	<u>ACTION ANSWER</u> <u>ACTION INSERT</u> <u>ACTION DELETE</u> <u>ACTION RUN</u> <u>ACTION SYNC</u> <u>ACTION PICK ACTIVITY</u> <u>ACTION SEARCH</u> <u>ACTION WEB SEARCH</u> <u>ACTION FACTORY TEST</u> <u>ACTION SENDTO</u>
Built-in Standard Broadcast Actions	
<u>ACTION TIME TICK</u> <u>ACTION TIME CHANGED</u> <u>ACTION TIMEZONE CHANGED</u> <u>ACTION BOOT COMPLETED</u> <u>ACTION PACKAGE ADDED</u> <u>ACTION PACKAGE CHANGED</u> <u>ACTION PACKAGE REMOVED</u>	<u>ACTION PACKAGE RESTARTED</u> <u>ACTION PACKAGE DATA CLEARED</u> <u>ACTION UID REMOVED</u> <u>ACTION BATTERY CHANGED</u> <u>ACTION POWER CONNECTED</u> <u>ACTION POWER DISCONNECTED</u> <u>ACTION SHUTDOWN</u>



Phần 1.3

Các thành phần của intent

Các thành phần của Intent

Thuộc tính chính	Thuộc tính phụ
action -tên (string) của action mà Intent sẽ yêu cầu thực hiện -có thể là action được Android định nghĩa sẵn (built-in standard action) hoặc do người lập trình tự định nghĩa	category -thông tin về nhóm của action type -định dạng kiểu dữ liệu (chuẩn MIME) -thường được tự động xác định
data -dữ liệu mà Activity được gọi sẽ xử lý -định dạng Uri (thông qua hàm Uri.parse(data))	component -chỉ định cụ thể lớp sẽ thực thi Activity -khi được xác định, các thuộc tính khác trở thành không bắt buộc (optional)
	extras -chứa tất cả các cặp (key,value) do ứng dụng thêm vào để truyền qua Intent (cấu trúc Bundle)
http://developer.android.com/reference/android/content/Intent.html	



Các thành phần của Intent

- **Component name:** tên class xử lý intent (ví dụ: “com.example.project.app.MyActivity1”)
- **Action:** tên các hành động mà intent yêu cầu thực hiện (ví dụ: action_view, action_call,...)
- **Data:** dữ liệu yêu cầu được xử lý, dữ liệu này thường được biểu diễn dưới dạng URI (ví dụ: "tel:216-555-1234", "http://txnam.net",...)
 - Trường hợp dữ liệu phức tạp hoặc không cố định, người ta thường đẩy vào phần extras



Các thành phần của Intent

- **Type**: định dạng kiểu dữ liệu của data (dùng chuẩn MIME), thường được tự xác định bởi hệ thống
- **Category**: bổ sung thông tin cho các action của intent (ví dụ: nếu một activity có thuộc tính category là `CATEGORY_LAUNCHER` nghĩa là activity đó có thể khởi chạy cấp ứng dụng)
- **Extras**: dữ liệu bổ sung nếu vùng Data là chưa đủ, extras sử dụng cấu trúc bundle gồm các cặp (key, value)



Phần 1.4

Sử dụng intent để trao đổi dữ liệu



Trao đổi intent giữa các activity

- Dữ liệu được đóng gói vào intent
 - Lựa chọn dữ liệu đưa vào intent phù hợp
 - Có chuẩn trao đổi dữ liệu chung để dễ xử lý
- Sau đó gọi activity phù hợp để xử lý
 - `startActivity(myIntent);`
- Nếu muốn nhận kết quả trả về thì dùng
 - `startActivityForResult(myIntent, CODE);`
 - Trong đó CODE là một số nguyên, dùng để phân biệt kết quả trả về giữa các activity khác nhau
 - Xử lý bằng cách viết lại hàm `onActivityResult`



Trao đổi intent giữa các activity

- Về phía bên activity nhận, lấy intent gửi cho mình bằng `getIntent()`
- Nếu cần trả về kết quả nào đó cho activity trước
 - Thông báo thất bại: `setResult(RESULT_CANCELED);`
 - Thông báo thành công: `setResult(RESULT_OK, x);`
 - Với x là intent mà sẽ trả ngược lại cho activity gọi (tất nhiên là cần đưa dữ liệu vào x trước khi setResult, việc đưa dữ liệu sử dụng các hàm putExtra)
 - Chú ý trường hợp dữ liệu phức tạp: putSerializable

```

public void btnNhap(View v) {
    Intent i = new Intent(this, NhapSoActivity.class);
    startActivityForResult(i, 1001);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == 1001) {
        if (resultCode == RESULT_OK) {
            TextView t1 = (TextView) findViewById(R.id.textView1);
            TextView t2 = (TextView) findViewById(R.id.textView2);
            TextView t3 = (TextView) findViewById(R.id.textView3);
            String a = data.getStringExtra("SoA");
            String b = data.getStringExtra("SoB");
            t1.setText("A = " + a);
            t2.setText("B = " + b);
            t3.setText("Tổng = " + (Integer.parseInt(a) + Integer.parseInt(b)));
            Toast.makeText(this, "Trả về thành công", Toast.LENGTH_SHORT).show();
        }
        else
            Toast.makeText(this, "Trả về thất bại", Toast.LENGTH_SHORT).show();
    }
    else
        super.onActivityResult(requestCode, resultCode, data);
}
}

```

Đoạn mã minh họa việc gọi activity nhập liệu và xử lý kết quả trả về

`onActivityResult`: được tự động gọi khi activity nhập liệu kết thúc


```

public class NhapSoActivity extends Activity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_nhap_so);
    }

    public void btnCancel(View v) {
        setResult(RESULT_CANCELED);
        finish();
    }

    public void btnOK(View v) {
        Intent i = new Intent();
        EditText t1 = (EditText) findViewById(R.id.editText1);
        EditText t2 = (EditText) findViewById(R.id.editText2);
        i.putExtra("SoA", t1.getText().toString());
        i.putExtra("SoB", t2.getText().toString());
        setResult(RESULT_OK, i);
        finish();
    }
}

```

Xử lý ở phía activity nhập liệu

Dùng `setResult` để thiết lập dữ liệu trả về cho activity cha



Phần 2

Broadcast Receivers



Broadcast Receiver

- Broadcast receiver (gọi tắt là receiver): là một trong bốn loại thành phần cơ bản của ứng dụng android
- Receiver là một class java nhận và xử lý các sự kiện mà hệ thống (hoặc ứng dụng nào đó) phát ra
 - VD: tín hiệu báo mất wifi, tín hiệu báo cuộc gọi đến,...
- Khi hệ thống phát đi sự kiện, có 2 cơ chế phát:
 - Không thứ tự: mọi receiver đủ điều kiện đều nhận được
 - Có thứ tự: receiver nào ưu tiên hơn thì nhận trước và có thể điều chỉnh thông tin tín hiệu đến các receiver sau
 - Cơ chế này khá giống xử lý ngắt (interrupt) trong HĐH



Một số broadcast thông dụng

- Báo hệ thống khởi động xong
- Báo có package mới cài vào hoặc xóa đi
- Báo tắt máy
- Báo cắm sạc
- Báo rút sạc
- Thông báo cắm thẻ nhớ
- Thông báo rút thẻ nhớ
- Thông báo tin nhắn tới
- Thông báo có cuộc gọi đi/đến



Phần 2.1

Vòng đời của broadcast receiver



Vòng đời của broadcast receiver

1. Khi ứng dụng được cài lên thiết bị:
 - Hệ thống đọc file `AndroidManifest.xml` để xem receiver đăng ký xử lý những sự kiện nào
 - Cho receiver vào danh sách các receiver cài trên thiết bị
2. Khi xảy ra sự kiện:
 - Hệ thống tạo một intent chứa thông tin về sự kiện
 - Gửi intent đó đến các receiver đăng ký xử lý sự kiện
 - Nếu sự kiện không thứ tự: gửi đồng loạt đến tất cả receiver
 - Nếu sự kiện có thứ tự: gửi lần lượt đến từng receiver
 - Khi intent được gửi đến receiver: hệ thống gọi hàm `onReceive` để xử lý intent đó



Vòng đời của broadcast receiver

- Broadcast receiver hoạt động vô cùng đơn giản
 - Chỉ cần viết duy nhất phương thức **onReceive**
 - Khi có sự kiện mà broadcast receiver đã đăng ký nhận được phát đi, thì phương thức **onReceive** của broadcast receiver đó sẽ được gọi
 - Sau khi thực thi xong phương thức này, lifecycle của receiver kết thúc
- Trường hợp broadcast có thứ tự, ta có thể gọi hàm **abortBroadcast** trong khi **onReceive** đang chạy để ngăn không cho các receiver sau nhận broadcast



Vòng đời của broadcast receiver

- Ngay khi **onReceive** kết thúc, hệ thống coi như receiver đã không còn hoạt động và có thể hủy tiến trình chứa receiver này bất cứ lúc nào, vì thế:
 - Tránh xử lý các code quá lâu trong onReceive
 - Không có xử lý bất đồng bộ, chờ callback... trong receiver (cụ thể như hiển thị Dialog, kết nối service...)
 - Nếu cứ cố dùng, hệ thống sẽ thông báo ứng dụng bị waiting quá lâu, có force close hay không, đây là lỗi nên tránh vì đa số người dùng sẽ chọn YES
- Độ ưu tiên của receiver nên từ -1000 đến 1000
`<intent-filter android:priority="1000">`



Phần 2.2

Tự tạo một tín hiệu broadcast

Tự tạo một tín hiệu broadcast

- Thông thường các tín hiệu broadcast sẽ do hệ thống sinh ra, nhưng LTV có thể tự tạo một tín hiệu riêng
 1. Tạo một intent chứa thông tin về tín hiệu
 2. Yêu cầu hệ thống gửi broadcast phù hợp
- Android có nhiều cơ chế phát tín hiệu khác nhau
 - **Normal broadcast**: gửi tín hiệu broadcast không đồng bộ
 - **Ordered broadcast**: gửi lần lượt các broadcast, vào mỗi thời điểm chỉ gửi đến một receiver
 - **Sticky broadcast**: intent được giữ lại sau khi receiver xử lý xong (sử dụng trong một số tình huống ít gặp hơn, ví dụ như cảnh báo pin yếu)



Ví dụ: gửi tín hiệu báo động đất

```
// chuẩn bị một intent
```

```
Intent intent = new Intent(NEW_EARTHQUAKE_FOUND);
```

```
// nạp dữ liệu về broadcast vào intent
```

```
intent.putExtra("date", "03/07/2016 07:00:00");
```

```
intent.putExtra("details", "London");
```

```
intent.putExtra("longitude", "-0.129098");
```

```
intent.putExtra("latitude", "51.535142");
```

```
intent.putExtra("magnitude", "3.0");
```

```
// chọn 1 trong 3 cách phát tín hiệu phù hợp
```

```
sendBroadcast(intent);
```

```
sendOrderedBroadcast(intent);
```

```
sendStickyBroadcast(intent);
```



Phần 2.3

Viết receiver xử lý tín hiệu broadcast



Sử dụng BroadcastReceiver

- Để đăng ký đối tượng receiver, có 2 cách:
 - Sử dụng `Context.registerReceiver()` để đăng ký và `Context.unregisterReceiver()` để hủy
 - Đăng ký trong file `AndroidManifest.xml` thông qua thẻ `<receiver />`
- Dùng cách thứ nhất nếu ta chỉ muốn chặn tín hiệu broadcast trong một khoảng thời gian nào đó
 - Ví dụ: khi chơi game, ta có thể chặn broadcast để biết có cuộc gọi đến hay không (để xử lý cho phù hợp)
- Cách thứ hai sử dụng nếu muốn chặn tín hiệu broadcast bất kỳ khi nào nó được phát ra



Ví dụ: viết ứng dụng auto start

- Chọn sự kiện `RECEIVE_BOOT_COMPLETED`
- Viết receiver tương ứng, trong `onReceive` chạy service ngầm hoặc khởi động activity
- Chú ý: Nếu ứng dụng của bạn ở SD card, lúc này SD card chưa sẵn sàng, vì thế ứng dụng sẽ không chạy được, trường hợp này ta chọn sang sự kiện `ACTION_EXTERNAL_APPLICATIONS_AVAILABLE`
 - Một số điện thoại (HTC) không nhận sự kiện này
 - Từ Android 3.0, phải chạy ứng dụng ít nhất một lần thì ứng dụng mới nhận sự kiện này



Ví dụ: viết ứng dụng auto start

```
<uses-permission
```

```
    android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

```
<receiver
```

```
    android:enabled="true"
```

```
    android:name=".MyApp"
```

```
    android:permission="android.permission.RECEIVE_BOOT_COMPLETED">
```

```
    <intent-filter>
```

```
        <action
```

```
            android:name="android.intent.action.BOOT_COMPLETED" />
```

```
            <category android:name="android.intent.category.DEFAULT" />
```

```
        </intent-filter>
```

```
</receiver>
```



Ví dụ: viết ứng dụng auto start

```
public class MyApp extends BroadcastReceiver {
    @Override
    public void onReceive(Context context, Intent intent) {
        // Trường hợp chạy activity
        Intent i = new Intent(context, MyActivity.class);
        i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(i);
        // Trường hợp chạy service
        Intent service = new Intent(context, MyService.class);
        context.startService(service);
    }
}
```