LẬP TRÌNH ANDROID

Giảng Viên: Phùng Mạnh Dương

Trường ĐH Công nghệ, ĐHQGHN

CHƯƠNG 5
NOTIFICATION VÀ BROADCAST RECEIVER

- Notification:
  - Toast
  - Dialog
  - Status bar
- BroadcastReceiver
  - Đăng ký
  - Các loại sự kiện
  - Quảng bá sự kiện
  - Xử lý sự kiện

Notification

- Notification là các thông điệp gửi tới người dùng bên ngoài giao diện người dùng thông thường.
- VD: Download book
- Android hỗ trợ 2 kiểu notification:
  - Phản hồi người dùng:
    - Toast: popup message (xuất hiện rồi tự động ẩn)
    - Dialog: Cửa sổ popup
  - Thông báo sự kiện: Thanh notification bar

Toast

- Các thông điệp ngắn xuất hiện trên cửa sổ giao diện hiện tại. Ví dụ, để thông báo cho người dùng về một hành động đã hoàn thành.
- Các thông điệp xuất hiện và biến mất tự động.
- Toast không cho phép người dùng tương tác hoặc trả lời.
- Khai báo:
  - Tạo đối tượng Toast:

    Toast.makeText(context, text, duration)
  - Hiển thị Toast:

    Toast.show()
- VD: NotificationToast

Tùy biến giao diện Toast

- Tạo một layout tùy biến bằng XML
- Inflate layout và đưa nó vào Toast thông qua phương thức Toast.setView().
- VD: NotificationToastWithCustomView

```
@Override
public void onClick(View v) {

    Toast toast = new Toast(getApplicationContext());

    toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
    toast.setDuration(Toast.LENGTH_LONG);

    toast.setView(getLayoutInflater().inflate(R.layout.custom_toast,null));

    toast.show();
}
```

Dialog

- Dialog là cửa sổ phụ được sử dụng bởi Activity để tương tác với người dùng.
- Một số kiểu dialog:
  - AlertDialog
  - ProgressDialog
  - DatePickerDialog
  - TimePickerDialog
- AlertDialog bao gồm 3 phần:
  - Tiêu đề
  - Nội dung
  - Nút Hành động

Tạo và Hiển thị Dialog

- Tạo Dialog:
  1. Tạo lớp kế thừa DialogFragment
  2. Thiết lập Dialog trong phương thức onCreateDialog()
- Hiển thị Dialog:
  - Khởi tạo đối tượng dialog: MyDialog d = new MyDialog
  - Hiển thị Dialog: d.show(getFragmentManager(),"TAG")

- VD: UIAlertDialog

Thanh trạng thái (Status bar)

- Android cung cấp 1 thanh trạng thái để thông báo tới người về các sự kiện.
- Android cũng cung cấp một vùng thông báo mở rộng (notification drawer) mà người dùng có thể kéo xuống để xem thông tin chi tiết về các thông báo.
- VD trực tiếp trên điện thoại: Vừa gọi điện vừa lướt web để lấy thông tin

Cấu trúc của một thông báo

- Một thông báo cần có các thành phần sau:
  - Tiêu đề
  - Nội dung chi tiết
  - Biểu tượng
- Khi một thông báo được gửi đi, nó sẽ xuất hiện trên thanh trạng thái bao gồm:
  - Một biểu tượng nhỏ xuất hiện
  - Một đoạn text có thể xuất hiện kèm theo
- Nếu người dùng kéo mở vùng thông báo mở rộng, chi tiết thông báo sẽ xuất hiện bao gồm:
  - Tiêu đề, Nội dung, Biểu tượng, Thời gian
  - Cần định nghĩa hành động khi người dùng click vào thông báo.

Quản lý thông báo

- Thông báo được quản lý bởi lớp NotificationManager
- Qua đó, thông báo có thể được gửi, sửa, hoặc xóa...
- VD: NotificationStatusBar

BroadcastReceiver

- BroadcastReceiver là lớp cơ sở cho các thành phần cần nhận và xử lý các sự kiện.
- Để thực hiện, các BroadcastReceiver đăng ký nhận sự kiện chúng quan tâm với hệ thống.
  - VD: đăng ký lắng nghe sự kiện nhận tin nhắn.
- Khi sự kiện xuất hiện, chúng được biểu diễn như các Intent và được gửi quảng bá toàn hệ thống.
- Android sẽ chuyển các Intent trên tới BroadcastReceiver đã đăng ký nhận chúng.
- BroadcastReceiver nhận Intent thông qua lời gọi tới phương thức onReceive()

Quy trình hoạt động của BroadcastReceiver

1. BroadcastReceiver đăng ký nhận sự kiện

2. Sự kiện xuất hiện dẫn tới Intent tương ứng được tạo và quảng bá trong hệ thống

3. Android chuyển các Intent tới các nơi nhận đã đăng ký bằng cách gọi phương thức onReceive() của chúng

4. Sự kiện được xử lý trong phương thức onReceive()

Đăng ký nhận Intent

- BroadcastReceiver có thể đăng ký theo 2 cách:
  - Tĩnh trong file AndroidManifest.xml
  - Động thông qua phương thức registerReceiver()
- Đăng ký tĩnh: Đặt các thẻ <receiver> và <intent-filter> trong file AndroidManifest.xml

```
<receiver
    android:enabled=["true" | "false"]
    android:exported=["true" | "false"]
    android:icon="drawable resource"
    android:label="string resource"
    android:name="string"
    android:permission="string">
    <intent-filter>...</intent-filter>
. . .
</receiver>
```

Đăng ký nhận Intent

- Đăng ký tĩnh sẽ được đăng ký khi hệ thống khởi động hoặc khi ứng dụng được cài đặt lúc hệ thống đang hoạt động.
- VD: SingleBroadcastStaticRegistration

SV tự viết 1 receiver không thuộc app, không có activity nhận intent vừa quảng bá

Receiver.java | SimpleBroadcast.java

```
1  package course.examples.BroadcastReceiver.singleBroadcastStaticRegistration;
2
3  import android.content.BroadcastReceiver;
9
10 public class Receiver extends BroadcastReceiver {
11
12     private final String TAG = "Receiver";
13
14     @Override
15     public void onReceive(Context context, Intent intent) {
16
17         Log.i(TAG, "INTENT RECEIVED");
18
19         Vibrator v = (Vibrator) context
20                 .getSystemService(Context.VIBRATOR_SERVICE);
21         v.vibrate(500);
22
23         Toast.makeText(context, "INTENT RECEIVED by Receiver", Toast.LENGTH_LONG).show();
24
25     }
26
```

Đăng ký nhận Intent

- Đăng ký động có thể dùng trong các trường hợp yêu cầu tương tác từ ứng dụng.
  - VD: chỉ đăng ký BroadcastReceiver khi chương trình đang chạy + hiển thị với người dùng bằng cách thiết lập trong phương thức onResumed() và onPaused().
- Các bước thực hiện:
  1. Tạo một IntentFilter trong đó chỉ định Intent đăng ký nhận.
  2. Tạo một BroadcastReceiver.
  3. Đăng ký BroadcastReceiver sử dụng registerReceiver()
     - LocalBroadcastManager: cho các Intent chỉ quảng bá trong ứng dụng.
     - Context: cho các Intent quảng bá trên toàn hệ thống.
  4. Gọi unRegisterReceiver() để hủy đăng ký BroadcastReceiver

VD: SingleBroadcastDynRegistration

Quy trình hoạt động của BroadcastReceiver

1. BroadcastReceiver đăng ký nhận sự kiện

2. Sự kiện xuất hiện dẫn tới Intent tương ứng được tạo và quảng bá trong hệ thống

3. Android chuyển các Intent tới các nơi nhận đã đăng ký bằng cách gọi phương thức onReceive() của chúng

4. Sự kiện được xử lý trong phương thức onReceive()

Sự kiện quảng bá

- Android cung cấp nhiều kiểu Intent quảng bá khác nhau:
  - Thông thường hoặc Có thứ tự
    - Thông thường: Intent được gửi đồng thời tới các receiver
    - Có thứ tự: Intent được gửi được gửi lần lượt tới các receiver theo thứ tự ưu tiên
  - Lưu hoặc không lưu
    - Lưu (sticky): Intent vẫn được lưu sau khi đã được quảng bá (VD: battery level ← chỉ quan tâm tới trạng thái hiện tại)
    - Không lưu (non-sticky): Intent bị hủy sau khi đã quảng bá (quan tâm tới thời điểm xảy ra sự kiện).
  - Có hoặc không có yêu cầu permission đối với receiver.

Quy trình hoạt động của BroadcastReceiver

1. BroadcastReceiver đăng ký nhận sự kiện

2. Sự kiện xuất hiện dẫn tới Intent tương ứng được tạo và quảng bá trong hệ thống

3. Android chuyển các Intent tới các nơi nhận đã đăng ký bằng cách gọi phương thức onReceive() của chúng

4. Sự kiện được xử lý trong phương thức onReceive()

Truyền sự kiện

- Intent được truyền đi bằng cách gọi phương thức onReceive() với 2 đối số:
  - Ngữ cảnh mà receiver đang hoạt động.
  - Intent được truyền

Quy trình hoạt động của BroadcastReceiver

1. BroadcastReceiver đăng ký nhận sự kiện

2. Sự kiện xuất hiện dẫn tới Intent tương ứng được tạo và quảng bá trong hệ thống

3. Android chuyển các Intent tới các nơi nhận đã đăng ký bằng cách gọi phương thức onReceive() của chúng

4. Sự kiện được xử lý trong phương thức onReceive()

Xử lý sự kiện

- Một số lưu ý khi xử lý sự kiện:
  - onReceive() có mức ưu tiên cao nên cần chú ý hiệu năng do 1 sự kiện có thể dẫn đến nhiều onReceive() thực thi đồng thời.
  - onReceive() chạy ở luồn chính của ứng dụng nên cần kết thúc trong thời gian ngắn để tránh gián đoán ứng dụng chính. Trong trường hợp cần thời gian dài, nên khởi phát 1 service để thực hiện (VD: thông điệp MMS).
- VD: BcastRecCompBcast

Quảng bá có thứ tự

- Sử dụng khi muốn BroadcastReceiver nhận sự kiện theo 1 thứ tự nhất định hoặc khi muốn mỗi BroadcastReceiver có 1 mức truy cập nhất định vào Intent

```
// Gửi Intent tới BroadcastReceivers theo thư tự ưu tiên
void sendOrderedBroadcast (Intent intent,
                           String receiverPermission)

// Gửi Intent tới BroadcastReceivers theo thư tự ưu tiên
// bổ sung thêm các tham số tùy chỉnh
void sendOrderedBroadcast (Intent intent,
                           String receiverPermission,
                           BroadcastReceiver resultReceiver,
                           Handler scheduler,
                           int initialCode,
                           String initialData,
                           Bundle initialExtras)
```

VD: BcastRecCompOrdBcast & BcastRecCompOrdBcastWithResRec

Quảng bá được lưu (Sticky Broadcast)

- Có những sự kiện phản ánh sự thay đổi trạng thái diễn ra theo thời gian.
- BroadcastReceiver cần biết trạng thái hiện tại của thiết bị. VD: Chúng ta vẫn cần thông tin đó thậm chí nếu chúng không được đăng ký để được thông báo về sự thay đổi trạng thái đó.
- VD: Nếu mức pin xuống rất thấp, android sẽ thông báo sự kiện này với hệ thống. Các ứng dụng tốt sẽ thực hiện các tác vụ tốn nhiều pin trong trường hợp này. Bây giờ, khi một chương trình được thực thi và nó kiểm tra trạng thái pin.
- Sticky Intent được lưu bởi Android
- Intent mới tự động chèn lên Intent cũ.

VD:

- Ứng dụng phải có BROADCAST_STICKY permission mới được gửi sticky Intent.

```
//public abstract class Context ...
// send sticky Intent to interested BroadcastReceivers
void sendStickyBroadcast (Intent intent)

// send sticky Intent to interested BroadcastReceivers in priority order
// sender can provide various parameters for greater control
void sendStickyOrderedBroadcast (Intent intent,
                                 BroadcastReceiver resultReceiver,
                                 Handler scheduler,
                                 int initialCode,
                                 String initialData,
                                 Bundle initialExtras)
```

VD: BcastRecStickyInt