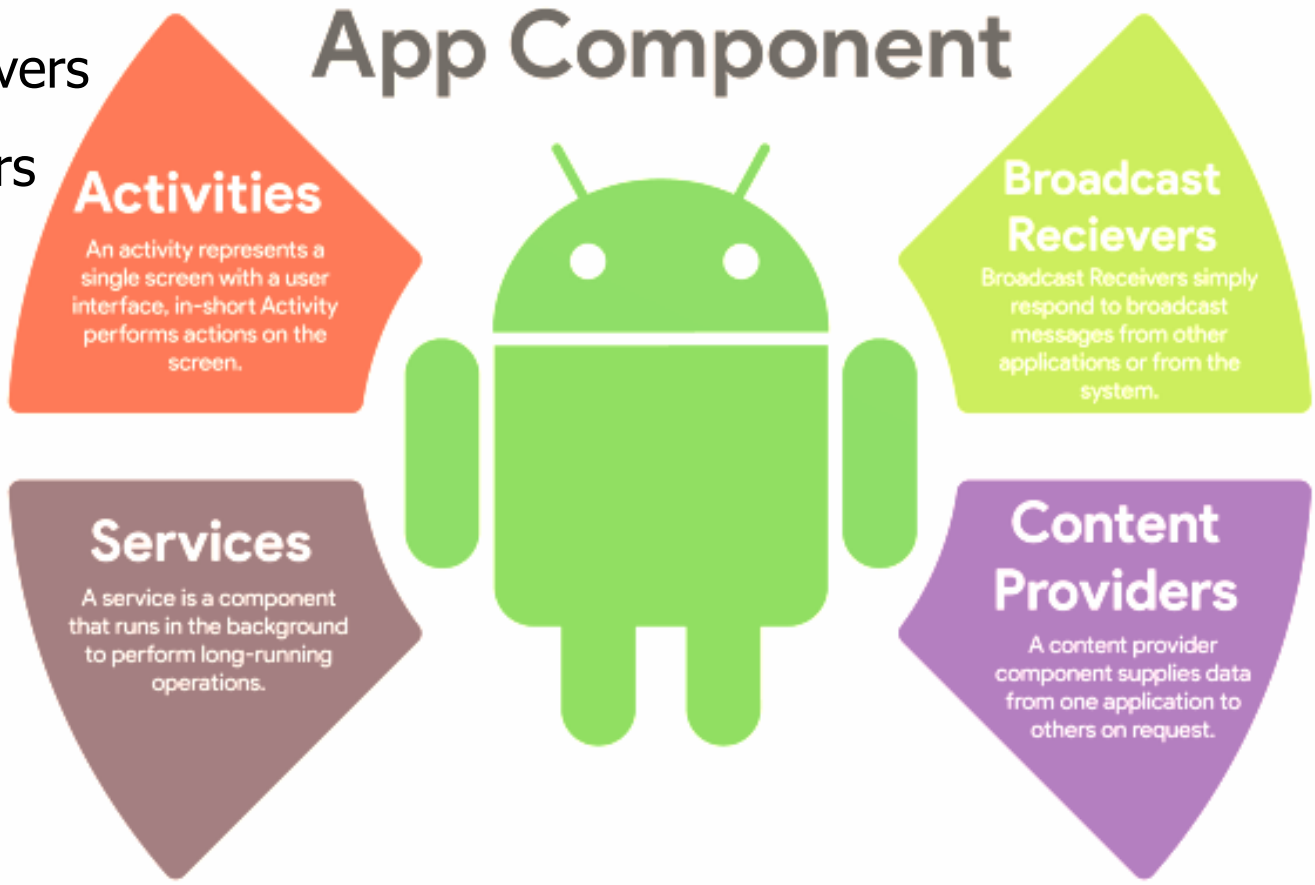
Activity

Android

Các thành phần cơ bản trong APP ANDROID

- APP Android - Ứng dụng Android có 4 thành phần chính:
  - Activities
  - Services
  - Broadcast Recievers
  - Content Providers



Các thành phần cơ bản trong APP ANDROID

❑ Activities

- Trong ứng dụng Android, Activity đóng vai trò đặc biệt quan trọng, là nơi giúp người dùng tương tác trực tiếp với ứng dụng, ví dụ như gọi điện thoại, chụp ảnh, gửi e-mail hoặc xem bản đồ.
- Activity được coi là xương sống của một ứng dụng Android, một ứng dụng có thể có một hoặc nhiều Activity (bất kì ứng dụng nào cũng cần có ít nhất 1 Activity).
- Activity có thể hiển thị ở chế độ toàn màn hình, dạng cửa sổ hoặc với một kích thước nhất định
- Một Activity có thể gọi đến một Activity khác, Activity được gọi đến sẽ tương tác với người dùng tại thời điểm được gọi tới.
- Một ứng dụng bên ngoài có thể gọi tới bất kỳ Activity nào trong ứng dụng (nếu được cấp quyền).
- **Ví dụ:** Một ứng dụng chụp ảnh sau khi chụp ảnh xong, sẽ gửi yêu cầu để start một activity có chức năng soạn e-mail trong ứng dụng email nhằm mục đích gửi ảnh vừa chụp đi.

Các thành phần cơ bản trong APP ANDROID

❑ Services

- Ví dụ: Chúng ta có thể vừa nghe nhạc, vừa lướt facebook là do ứng dụng nghe nhạc có một **service** chạy ngầm trong background để phát nhạc trong khi người dùng đang tương tác với ứng dụng facebook.
- Service trong Android được chia thành 3 loại đó là: **Foreground Service**, **Background Servie** và **Bound Service**.
- Service là một thành phần ứng dụng chạy ngầm trên hệ điều hành ví dụ như nghe nhạc, hoặc tương tác với một content provider. Service không tương tác trực tiếp với người dùng, khi service chạy thì người dùng vẫn có thể tương tác với một thành phần khác trong ứng dụng hoặc có thể tương tác với một ứng dụng khác trong hệ thống.

Các thành phần cơ bản trong APP ANDROID

❑ Broadcast Receiver

- Broadcast Receiver là một thành phần của ứng dụng giúp lắng nghe các sự kiện mà hệ thống phát ra thông qua Intent, hệ thống có thể truyền phát ngay cả khi app không chạy.
- Broadcast Receiver không có giao diện cụ thể nhưng nó có thể thực hiện thông báo thông qua thanh Notification.
- Có rất nhiều broadcast được phát ra từ hệ thống, chúng ta có thể lấy ví dụ như một broadcast thông báo rằng màn hình điện thoại đã tắt, hay điện thoại đang ở trạng thái "Battery Low", "Power Connected", "Power Disconnected" hoặc một bức ảnh đã được chụp.
- Cũng có những broadcast được phát ra từ ứng dụng như sau khi download một tệp, ví dụ: Sau khi hoàn thành download một tệp tin, ứng dụng A phát ra thông báo là dữ liệu đã download xong, tệp đã sẵn sàng cho các ứng dụng khác có thể sử dụng.

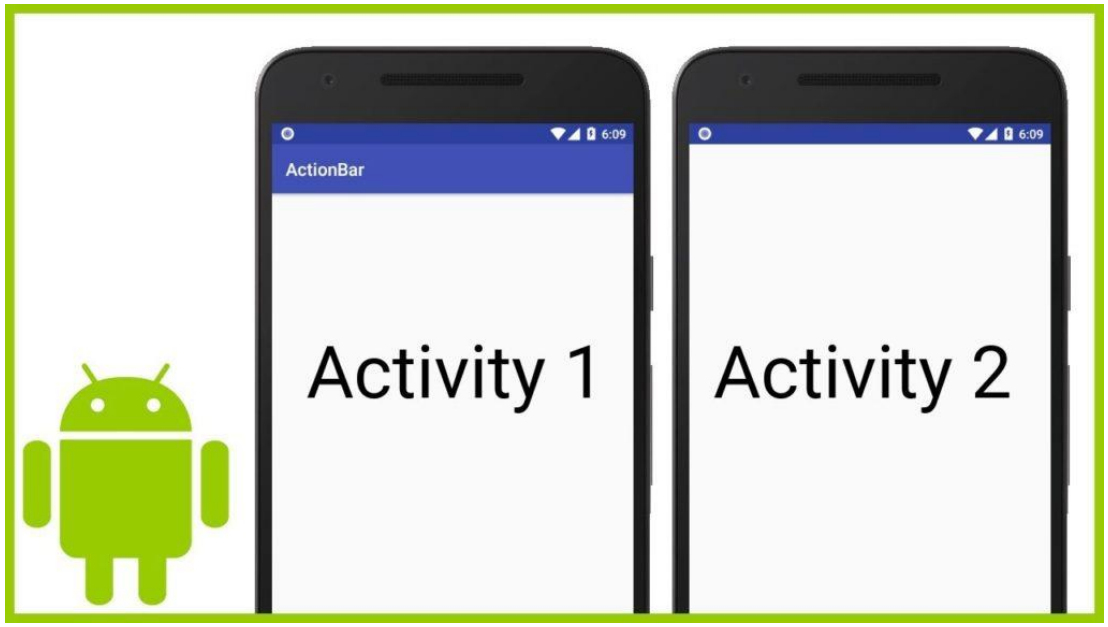
Các thành phần cơ bản trong APP ANDROID

Content Provider

- Content Provider là một thành phần giúp các một ứng dụng có thể đọc và ghi dữ liệu từ một file hoặc từ SQLite của một ứng dụng khác trong cùng một hệ thống. Bất kỳ ứng dụng nào có quyền (permisson) đều có thể truy xuất, chỉnh sửa dữ liệu của một ứng dụng khác.
- Content Provider được chia thành 2 loại:
  - Native Content Provider: Là những Content Provider có sẵn, được tạo ra bởi hệ thống, ví dụ như Contacts, Message, …
  - Custom Content Provider: Bao gồm các Content Provider được tạo ra bởi các developer phụ thuộc vào đặc điểm của từng ứng dụng.

ACTIVITY trong APP ANDROID

- Activity là gì: Activity đại diện cho một chức năng của app, là một giao diện màn hình, nơi user tương tác trực tiếp với app. Một ứng dụng có thể có một hoặc nhiều Activity. Một Activity từ khi được gọi đến khi kết thúc sẽ có những trạng thái (state) khác nhau.



ACTIVITY trong APP ANDROID

❑ Vòng đời Activity (Activity Lifecycle)

- ❖ Trong quá trình sử dụng app, user có thể di chuyển sự tương tác giữa các Activity hoặc thoát app. Trong Android, Activity class đã cung cấp các hàm callback để chúng ta có thể nắm bắt được trạng thái của Activity mỗi khi có thay đổi.
- ❖ Với các callback này, bạn có thể biết được Activity đang ở trạng thái nào mỗi khi user thoát ra và vào lại Activity đó. Ví dụ như user đang sử dụng app thì nhận được một cuộc gọi điện thoại, lúc này hệ thống sẽ chuyển sang app khác để phục vụ cho cuộc gọi đến, sau khi kết thúc cuộc gọi hệ thống sẽ điều hướng quay lại app đang sử dụng trước đó

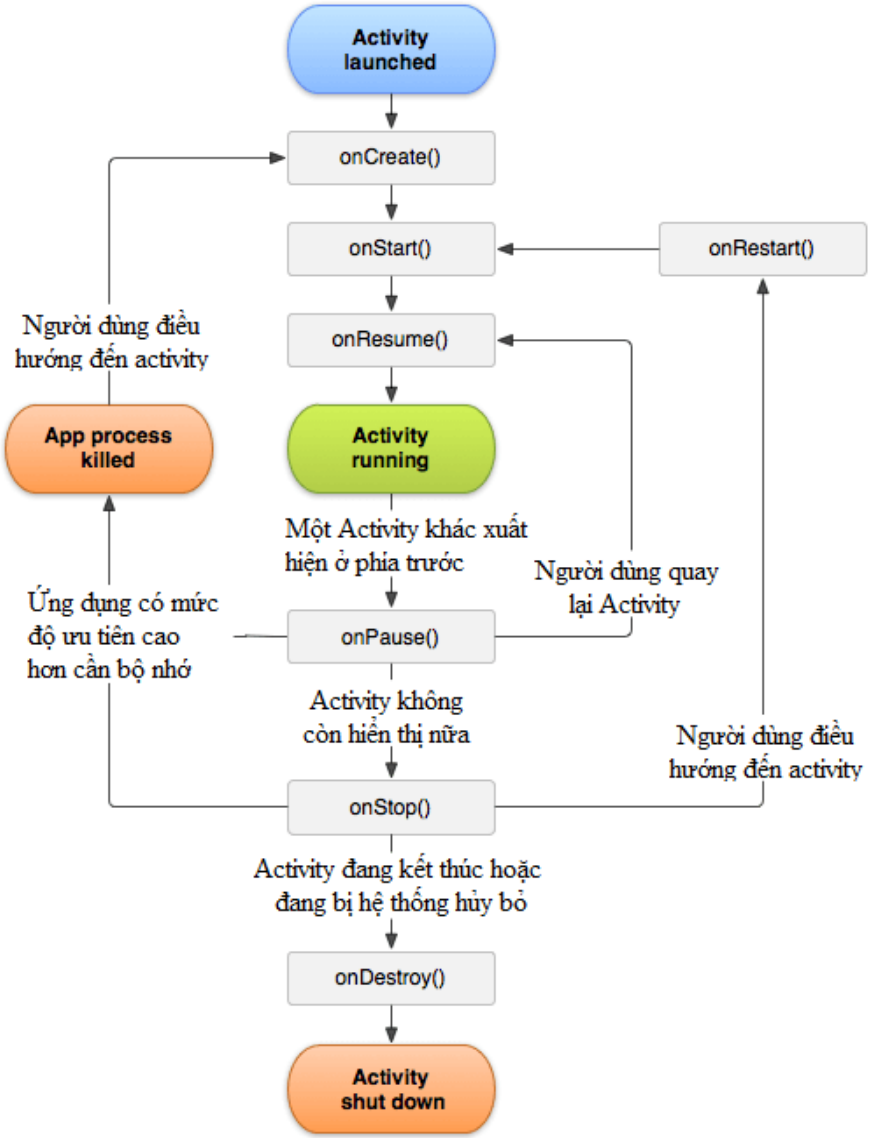
ACTIVITY trong APP ANDROID

Vòng đời Activity (Activity Lifecycle)

- Việc hiểu đúng và kiểm soát tốt Activity Lifecycle sẽ giúp bạn tránh được các lỗi không đáng có như:
  - Crash app khi hệ thống ưu tiên một app khác khi app này đang sử dụng (ví dụ cuộc gọi đến).
  - Tiêu tốn tài nguyên của hệ thống khi mà user đang không thực sự sử dụng app.
  - Bị mất data hoặc không lưu vết các thao tác của user khi luân chuyển giữa các app.
  - Crash app hoặc mất dữ liệu, thao tác của user khi luân chuyển giữa 2 chế độ Landscape và Portrait (chế độ ngang và dọc) của điện thoại.

ACTIVITY trong APP ANDROID

- Sơ đồ vòng đời Activity (Activity Lifecycle)



ACTIVITY trong APP ANDROID

- Vòng đời Activity (Activity Lifecycle) - Có 3 vòng lặp chính:
  - Entire lifetime: Xảy ra giữa onCreate() và onDestroy().
    - 1 activity sẽ cài đặt các trạng thái trong onCreate() và giải phóng toàn bộ tài nguyên trong onDestroy().
    - Ví dụ có 1 luồng chạy ngầm tải dữ liệu từ trên mạng, ta có thể tạo luồng tại onCreate() và kết thúc luồng tại onDestroy().
  - Visible lifetime: Xảy ra giữa onStart() và onStop().
    - Trong giai đoạn này, người dùng có thể nhìn thấy activity trên màn hình, tuy nhiên nó không ở trên đầu ngăn xếp và không thể tương tác với người dùng. Giữa 2 phương thức này người dùng có thể lưu trữ được tài nguyên cần thiết hiển thị lên activity cho người dùng.
    - Ví dụ ta cần quan sát thay đổi ảnh hưởng tới giao diện, thì ta có thể đặt Broadcast Receiver tại onStart() và loại bỏ tại onStop().
  - Foreground lifetime: Xảy ra giữa onResume() và onPause().
    - Trong giai đoạn này activity ở trên đầu ngăn xếp và có thể tương tác với người dùng.

ACTIVITY trong APP ANDROID

- Vòng đời Activity (Activity Lifecycle) – Các hàm trong vòng đời:
  - onCreate() - Đây là hàm bắt buộc phải được implement, được gọi đầu tiên một lần duy nhất khi hệ thống khởi tạo Activity, callback này chỉ được gọi lại khi hệ thống xóa Activity đi hoặc khi chuyển trạng thái giữa 2 chế độ màn hình (Landscape và Portrait). Vì được gọi đầu tiên và một lần duy nhất nên phương thức setContentView() sẽ được gọi ở hàm này, tùy vào logic của Activity, một số logic cũng sẽ được cho vào hàm này.
  - onStart() - Hàm này được gọi khi giao diện được hiển thị nhưng chưa tương tác được với user.
  - onResume() - Khi hệ thống gọi đến callback này, có nghĩa là các View, Component,... của Activity đã khởi tạo xong, đây là trạng thái mà app có thể tương tác với user. Activity sẽ ở trạng thái này cho đến khi có một Activity hoặc app khác được gọi chẳng hạn như có cuộc gọi đến, hoặc tắt màn hình điện thoại.

ACTIVITY trong APP ANDROID

- Vòng đời Activity (Activity Lifecycle) – Các hàm trong vòng đời:
  - onPause() - Callback này sẽ được gọi tới đầu tiên và ngay lập tức khi user thoát khỏi Activity (Activity chưa bị destroy). Tức là Activty không còn nằm ở foreground nữa nhưng vẫn có thể nhìn thấy đc Activity trên màn hình. onPause() thường được sử dụng khi Activity đang không được focus bởi user, một số tính năng trong đó sẽ được tạm dừng hoặc duy trì ở một mức độ nào đó để chờ cho user quay lại.
  - Một số lý do mà bạn nên sử dụng onPause():
    - Một số sự kiện làm cho Activity bị gián đoạn. Đây là trường hợp xảy ra phổ biến nhất.
    - Từ Android 7.0 (API 24) trở lên, Android có chức năng đa nhiệm (Multi Window). Khi chạy trong chế độ này ở bất kì thời điểm nào cũng chỉ có 1 app được focus bởi user, hệ thống sẽ tạm dừng tất cả các app khác.
    - Khi bị một Activity khác đè lên (ví dụ như một dialog). Activity hiện tại sẽ không tương tác với user, tuy nhiên vẫn được nhìn thấy trên màn hình.

ACTIVITY trong APP ANDROID

- Vòng đời Activity (Activity Lifecycle) – Các hàm trong vòng đời:
  - onStop() - Khi Activity không còn được nhìn thấy trên màn hình nữa, Activity sẽ rơi vào trạn thái onStop(), lúc này bất kỳ chức năng nào trong Activity cũng có thể bị dừng lại, qua đó app sẽ giải phóng được các tài nguyên không cần thiết khi mà nó không còn hữu dụng với user. Tuy nhiên, trong một số trường hợp onStop() cũng được dùng để lưu trữ dữ liệu.
  - onRestart() - Phương thức callback này gọi khi activity đã stoped, gọi trước khi bắt đầu start lại Activity.
  - onDestroy() - Callback này được gọi khi user thoát hoàn toàn khỏi Activity (nhấn nút back hoặc gọi tới hàm finish() của Activity).

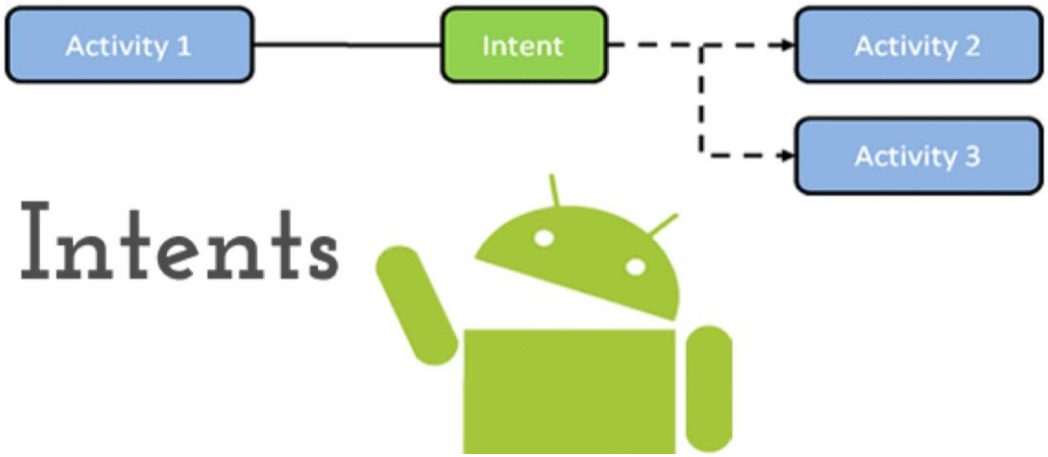
ACTIVITY trong APP ANDROID

- Activity sẽ có 4 trạng thái chính như sau:
  - Active/Running: Khi activity ở foreground và đang tương tác trực tiếp với user.
  - Visible/Pause: Activity không thể tương tác nhưng vẫn được nhìn thấy (bị che khuất không toàn toàn bởi một Activity khác).
  - Hidden/Stop: Activity bị che khuất hoàn toàn bởi một Activity khác, vẫn có thể lưu trữ thông tin những sẽ bị ưu tiên xóa bỏ nếu hệ thống thiếu bộ nhớ.
  - Destroy: Khi Activity gọi tới hàm finish() hoặc bị hệ thống xóa bỏ. Khi acitivty đó hiển thị lại với user, nó được khởi tạo lại và khôi phục lại trạng thái trước đó.

INTENT

Intent

- Intent là một thành phần quan trọng trong Android. Nó cho phép các thành phần ứng dụng có thể yêu cầu các hàm từ các thành phần ứng dụng Android khác.
- Intent là một object của class android.content.Intent. Intent sẽ được gửi đến hệ thống Android để xác định hành động bạn muốn thực hiện, đối tượng bạn muốn xử lý.

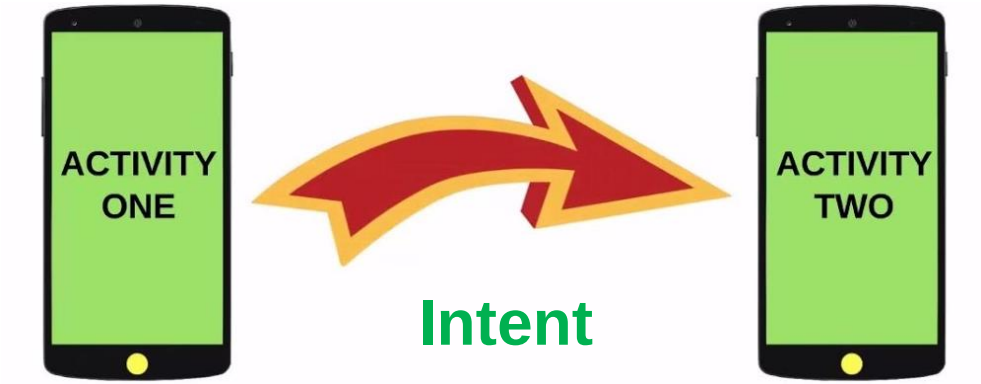


INTENT

Intent

- Ví dụ: Thông qua startActivity() bạn có thể xác định một Intent sử dụng để gọi chạy một Activity khác. Tại Activity mục tiêu, với startActivity() bạn có thể xác định được Intent của người gửi đến để khởi động Activity này.

```
Intent intent = new Intent(MainActivity.this, LoginActivity.class);
startActivity(intent);
```



INTENT

- Android intents thường được sử dụng chính:
  - Start dịch vụ
  - Gọi một activity
  - Hiển thị một trang web
  - Hiển thị danh sách liên hệ
  - Gửi một tin nhắn
  - Gọi điện thoại
  - …

INTENT

- Intent trong Android có 2 loại:
  - Intent tường minh (explicit):
    - ✓ Intent tường minh là loại intent mà bạn sử dụng khi bạn biết rõ tên của thành phần (component) mà bạn muốn khởi chạy (activity, service, broadcast receiver).
    - ✓ Khi bạn sử dụng intent tường minh, bạn cung cấp tên của lớp (class) cụ thể bạn muốn gọi.
  - Intent không tường minh (implicit):
    - ✓ Intent không tường minh là loại intent mà bạn sử dụng khi bạn muốn hệ thống Android tìm kiếm một thành phần phù hợp để thực hiện một hành động cụ thể dựa trên các tiêu chí nhất định (ví dụ: mở một trình duyệt web, gửi một email).
    - ✓ Bạn chỉ định hành động cần thực hiện và dữ liệu liên quan, và hệ thống sẽ tìm kiếm các thành phần có khả năng thực hiện hành động đó.

INTENT

- Ví dụ dạng tường minh: Gửi và nhận dữ liệu từ Activity này sang Activity khác

***Tại Activity gửi dữ liệu**

```
Intent intent = new Intent(LoginActivity.this, HomeActivity.class);
intent.putExtra("username", “tmdat");
intent.putExtra("old", 18);
intent.putExtra("phone", "09137213824");
startActivity(intent);
```

***Tại Activity nhận dữ liệu**

```
Intent intent = this.getIntent();
String username= intent.getStringExtra("username");
int old = intent.getIntExtra("old", -1);
String phone = intent.getStringExtra("phone");
```

INTENT

- Ví dụ dạng không tường minh: Trường hợp này không cần phải chính xác biết tên lớp Activity đích, mà chỉ cần biết rằng hệ thống sẽ tìm kiếm các ứng dụng có thể xử lý hành động "SEND".

```
// Dữ liệu cần gửi
String message    "Hello, this is a message from SourceActivity.";
// Tạo Intent không tường minh với hành động (action) và dữ liệu
Intent intent    new Intent(Intent.ACTION_SEND);
intent.putExtra  Intent.EXTRA_TEXT, message);
intent.setType("text/plain");
// Khởi chạy các ứng dụng có thể xử lý hành động (action) SEND
startActivity(intent);
```

Bundle

- Ngoài việc sử dụng Extra trong Intent để truyền dữ liệu chúng ta có thể sử dụng Bundle.
- Nếu như Extra truyền từng dòng dữ liệu qua Actitvity thì Bundle sẽ ”đóng gói” dữ liệu lại và gửi nguyên kiện.
- Bundle sẽ tiện hơn trong trường hợp bạn muốn gửi cùng một bộ dữ liệu đến nhiều Activity khác nhau.
- Ngoài nhiệm vụ đóng gói dữ liệu để truyền qua lại giữa các Activity, Bundle còn dùng trong một số mục đích khác, ví dụ như truyền dữ liệu qua Fragment

Bundle

- Ví dụ: Gửi & nhận dữ liệu từ Activity này sang Activity khác sử dụng Bundle.

***Tại Activity gửi dữ liệu**

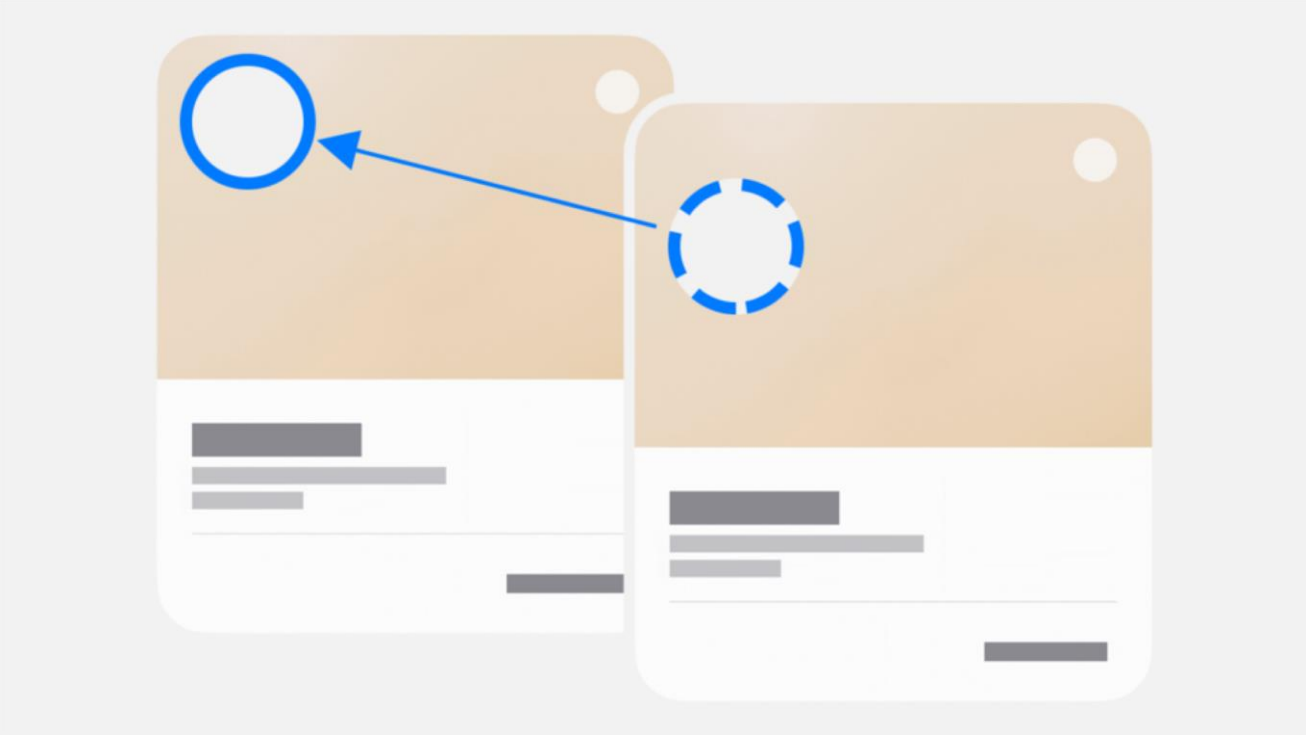
```
Intent intent = new Intent(LoginActivity.this, HomeActivity.class);
Bundle bundle = new Bundle();
bundle.putString("username", “tmdat");
bundle.putInt("old", 18);
intent.putExtras(bundle);
startActivity(intent);
```

***Tại Activity nhận dữ liệu**

```
Intent intent = getIntent();
Bundle bundle = intent.getExtras();
String username= bundle.getString("username", "");
int old = bundle.getInt("old", -1);
```

registerForActivityResult

- registerForActivityResult được sử dụng khi Activity A start Activity B và muốn nhận dữ liệu trả về từ Activity B đó.



registerForActivityResult

- Tại Activity nhận dữ liệu tạo ActivityResultLauncher

```
ActivityResultLauncher<Intent> getData = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
        new ActivityResultCallback<ActivityResult>() {
            @Override
            public void onActivityResult(ActivityResult result) {
                if (result.getResultCode() == 2) {
                    Intent intent = result.getData();
                    Bundle bundle = intent.getExtras();
                    String data = bundle.getString("data");
                }
            }
        });
```

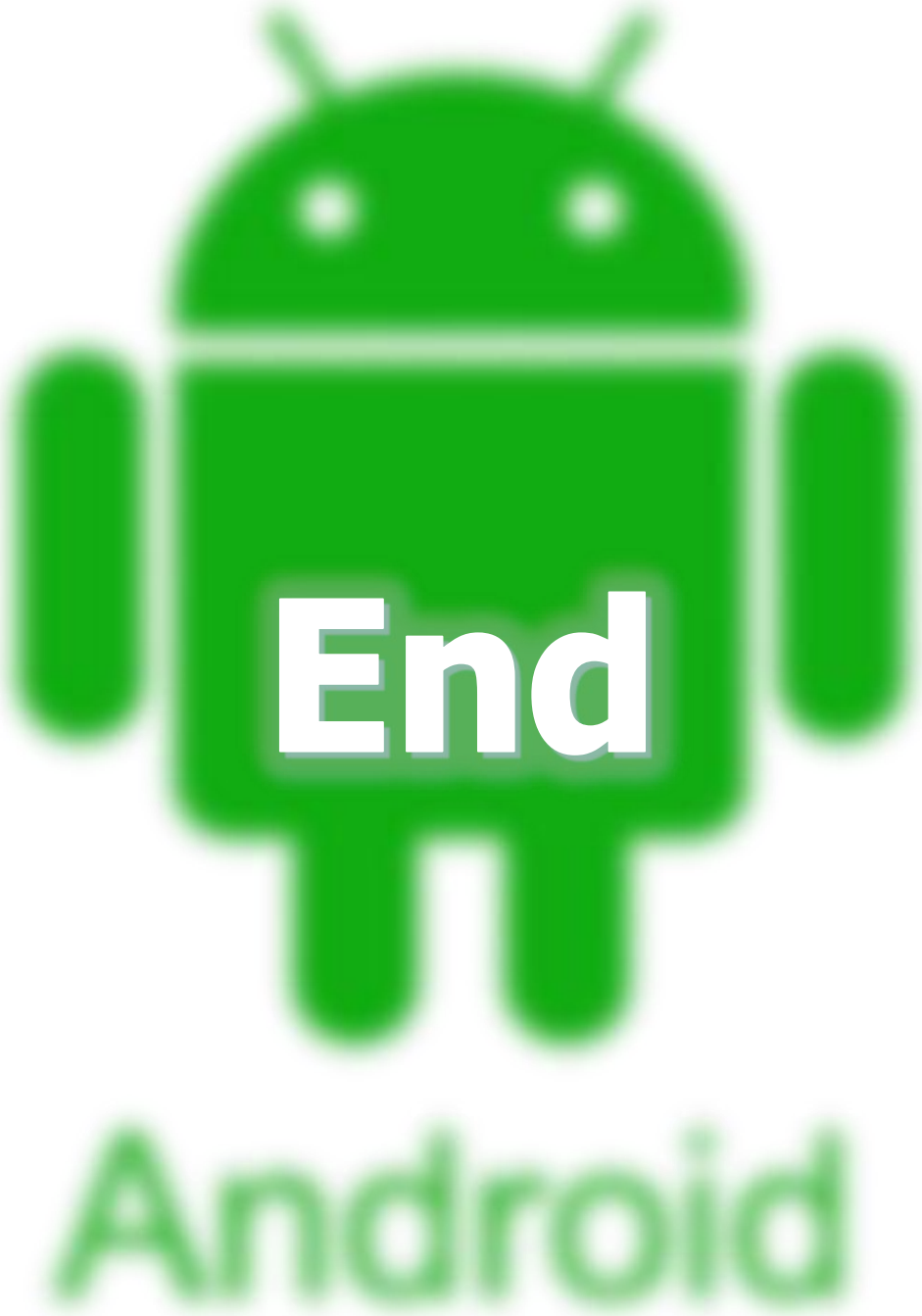
registerForActivityResult

- Gọi intent và launch hàm nhận dữ liệu

```
Intent intent = new Intent(MainActivity.this, LoginActivity.class);
getData.launch(intent);
```

- Tại Activity gửi dữ liệu

```
Intent intent = new Intent();
Bundle bundle = new Bundle();
bundle.putString("data", "Data được gửi về");
intent.putExtras(bundle);
setResult(2, intent);
finish();
```


End
Android