**Handling Activity Lifecycle Events**

Một Activity là thành phần quan trọng cấu thành nên ứng dụng Android. Nó cung cấp một màn hình cho phép người dùng có thể tương tác ví dụ như gọi điện, gửi sms, xem bản đồ… Một ứng dụng thường có nhiều activities.

Website hay được dùng để so sánh với activities, giống như website có nhiều pages, android app có nhiều activities, website có home pages, android app có main acitivity, chính là màn hình đầu tiên khi launch ứng dụng. Website có thể chuyển qua lại giữa nhiều pages, android cũng tương tự, bạn có thể chuyển từ 1 page từ trang web này sang page của trang web khác, android cũng có thể start 1 activity của ứng dụng khác. Ví dụ bạn đang open Contacts app, bạn có thể chuyển sang màn hình gửi tin nhắn của ứng dụng Messaging app.

## Creating an Activity

Để tạo một activity, bạn phải tạo một lớp con của lớp Activity. Trong lớp con này bạn phải implement một số phương thức của lớp cha. Hai phương thức quan trọng nhất cần implement:

* onCreate() : Bạn phải implement phương thức này, nó sẽ được gọi khi khởi chạy activity, dùng để khởi tạo những thành phần cần thiết của activity. Quan trọng nhất nó là nơi gọi phương thức setContentView() để định nghĩa layout.
* onPause() : Phương thức này sẽ được gọi khi bạn chuyển sang màn hình khác (not invisible). Được dùng để lưu trữ lại một số trạng thái bị thay đổi bởi user.

## Declaring the activity in the manifest

Bạn phải khai báo activity vào file manifest để hệ thống có thể truy cập được.

<manifest ... >

<application ... >

<activity android:name=*".ExampleActivity"* />

...

</application ... >

...

</manifest >

## Using intent filters

Sử dụng intent filter cho activity để khai báo ứng dụng có thể start activity như thế nào. Ví dụ nếu bạn muốn activity của bạn sẽ được launcher khi start app, bạn phải khai báo intent filter với action main và category launcher cho nó.

<activity android:name=*".ExampleActivity"* android:icon=*"@drawable/app\_icon"*>

<intent-filter>

<action android:name=*"android.intent.action.MAIN"* />

<category android:name=*"android.intent.category.LAUNCHER"* />

</intent-filter>

</activity>

## Starting an Activity

Bạn có thẻ start một activity bằng cách gọi phương thức startActivity(), truyền vào một intent. Intent dùng để chỉ rõ activity nào mà bạn muốn start.

Intent intent = **new** Intent(**this**, ExampleActivity.**class**);

startActivity(intent);

Trong một số trường hợp bạn muốn truyền dữ liệu giữa các activity, bạn cũng có thể thông qua intent

Intent intent = **new** Intent(**this**, ExampleActivity.**class**);

intent.putExtra(“number”, 1);

startActivity(intent);

Trong ExampleActivity:

Bundle extras = getIntent().getExtras();

**int** number = extras.getInt("number");

## 

## Starting an activity for a result

Trong một số trường hợp activity muốn nhận kết quả trả về từ một activity khác mà nó start, ví dụ activity A hiển thị thông tin tài khoản, nó start activity B để update thông tin, khi activity B update thông tin xong nó quay lại activity A và activity A sẽ nhận được thông tin trả về từ activity B.

Start activity B từ activity A:

**int** requestCode = 1;

Intent intent = **new** Intent(getApplicationContext(), CActivity.**class**);

startActivityForResult(intent, requestCoe);

Trong activity B:

Intent intent = **new** Intent();

intent.putExtra("status", "Code for fun");

setResult(RESULT\_OK, intent);

finish();

Activity C nhận kết quả từ activity B:

@Override

**protected** **void** onActivityResult(**int** requestCode, **int** resultCode, Intent data) {

// **TODO** Auto-generated method stub

**super**.onActivityResult(requestCode, resultCode, data);

**if**(resultCode == Activity.RESULT\_OK && requestCode == 1) {

TextView textView = (TextView) findViewById(R.id.textView1);

textView.setText(data.getString("status"));

}

}

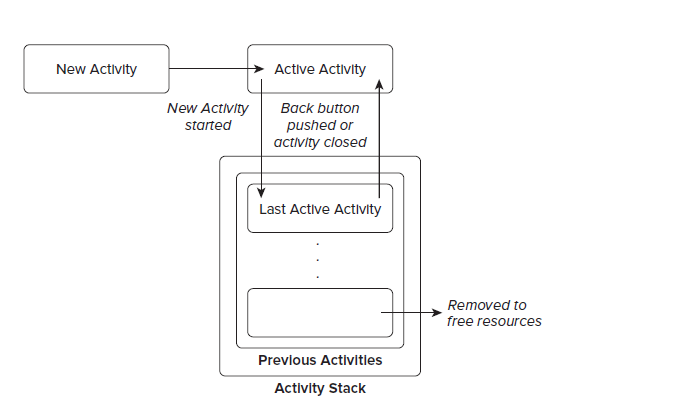
## Activity Life Cycle

Launching một activity khá là tốn bộ nhớ, nó phải phân bổ bộ nhớ cho tất cả các UI objects, đọc tất cả các object từ file xml và setting lên màn hình. Để giảm sự tiêu tốn bộ nhớ, activity life cycle sẽ được quản lý thông qua Activity Manager.

Activity Manager có trách nhiệm khởi tạo, hủy và quản lý các activities. Ví dụ khi user start ứng dụng lần đầu, activity manager sẽ khởi tạo activity, sau đó nếu user chuyển màn hình, activity manager sẽ giữ lại activity trước đó ( không xóa ) để nếu có trường hợp user muốn back lại màn hình trước, thì nó có thể start nhanh chóng ( không phải khởi tạo lại ). Trong trường hợp activity không được sử dụng trong một thời gian thì sẽ bị hủy để giải phóng bộ nhớ.

## Activity Stacks

Trạng thái của activity xác định nó nằm ở vị trí nào trong stack, last-in-first-out collection của tất cả các activity đang running. Khi có một activity mới được start, màn hình hiện thời ở foreground được chuyển đến vị trí đầu tiên của stack, nếu user quay lại màn hình trước bằng cách click Back button, hoặc foreground activity bị close thì activity phía dưới trong stack sẽ được chuyển lên vị trí đầu tiên.



## Managing the Activity Lifecycle

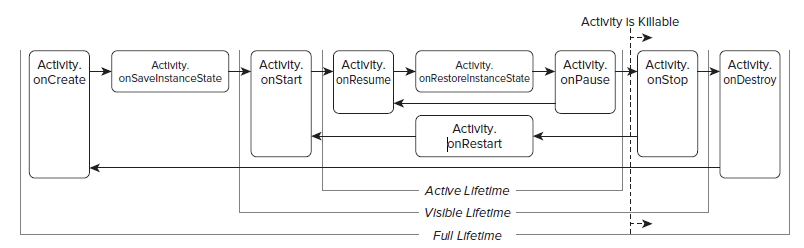
Về cơ bản activity có 3 trạng thái:

* Running (active) : Là trạng thái khi activity ở vị trí trên cùng của stack, hiển thị trên màn hình và user có thể tương tác được ( như là typing, touching , click button …) Ở trạng thái running, activity được sự ưu tiên trong việc sử dụng bộ nhớ để nó có thể phản hồi một cách nhanh chóng.
* Paused: Khi activity không được focus ( không có sự tương tác với user) nhưng vẫn hiển thị trên màn hình, chúng ta gọi nó là trạng thái paused. Có một activity khác ở foreground của màn hình, nhưng activity này vẫn có thể trông thấy được. User có thể vẫn nhìn thấy nhưng không tương tác được (ví dụ khi có dialog) .Một activity ở trong trạng thái paused chưa bị xóa, vẫn được giữ trong bộ nhớ, nhưng có thể sẽ bị kill bởi system trong trường hợp đặc biệt cần bộ nhớ.
* Stopped: Activity ở background không hiển thị trên màn hình, hoàn toàn bị che bởi activity khác. Activity ở trạng thái stopped vẫn được giữ lại trong bộ nhớ, tuy nhiên trong trường hợp cần bộ nhớ nó sẽ dễ bị kill bởi system hơn.

Nếu một activity ở trong trạng thái paused, stopped, hệ thống có thể xóa nó ra khỏi bộ nhớ bằng cách gọi đến phương thức finish(), hoặc đơn giản là kill process.

## Monitoring State Changes

Để đảm bảo Activity có thể tác động khi state change, Android cung cấp một số phương thức callback cái mà sẽ được gọi khi activity chuyển trạng thái.



**package** com.itplus.mobile.android.chapter17;

**import** android.app.Activity;

**import** android.os.Bundle;

**public** **class** MyActivity **extends** Activity {

// Called at the start of the full lifetime.

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

// Initialize activity.

}

// Called after onCreate has finished, use to restore UI state

@Override

**public** **void** onRestoreInstanceState(Bundle savedInstanceState) {

**super**.onRestoreInstanceState(savedInstanceState);

// Restore UI state from the savedInstanceState.

// This bundle has also been passed to onCreate.

}

// Called before subsequent visible lifetimes

// for an activity process.

@Override

**public** **void** onRestart() {

**super**.onRestart();

// Load changes knowing that the activity has already

// been visible within this process.

}

// Called at the start of the active lifetime.

@Override

**public** **void** onResume() {

**super**.onResume();

// Resume any paused UI updates, threads, or processes required

// by the activity but suspended when it was inactive.

}

// Called to save UI state changes at the

// end of the active lifecycle.

@Override

**public** **void** onSaveInstanceState(Bundle savedInstanceState) {

// Save UI state changes to the savedInstanceState.

// This bundle will be passed to onCreate if the process is

// killed and restarted.

**super**.onSaveInstanceState(savedInstanceState);

}

// Called at the end of the active lifetime.

@Override

**public** **void** onPause() {

// Suspend UI updates, threads, or CPU intensive processes

// that don’t need to be updated when the Activity isn’t

// the active foreground activity.

**super**.onPause();

}

// Called at the end of the visible lifetime.

@Override

**public** **void** onStop() {

// Suspend remaining UI updates, threads, or processing

// that aren’t required when the Activity isn’t visible.

// Persist all edits or state changes

// as after this call the process is likely to be killed.

**super**.onStop();

}

// Called at the end of the full lifetime.

@Override

**public** **void** onDestroy() {

// Clean up any resources including ending threads,

// closing database connections etc.

**super**.onDestroy();

}

}

### The Full Lifetime

Full lifetime của ứng dụng xảy ra ở giữa lần đầu gọi phương thức onCreate và lần cuối gọi phương thức onDestroy(). Trong một số trường hợp activity process bị kết thúc mà phương thức ondestroy() không được gọi.

Sử dụng phương thức onCreate() để khởi tạo activity, inflate UI, phân bổ references cho các variables, tạo service và thread. Method onCreate() được truyền vào Bundle object chứa trạng thái của UI được save trong lần gọi method onSaveInstanceState. Nên sử dụng Bundle object để khổi phục lại UI của trạng thái trước, có thể trong phương thức onCreate() hoặc onRestoreInstanceState().

Sử dụng onDestroy() method để xóa toàn bộ resources được khởi tạo trong onCreate và đảm bảo tất cả các kết nối với bên ngoài như là network, database được đóng.

Trong lập trình Android, để viết mã code hiệu quả nên tránh việc tạo các short-term object (vòng đời ngắn), việc object được khởi tạo rồi bị xóa trong khoảng thời gian ngắn dẫn đến sử dụng garbage collection, do đó nên cân nhắc việc khởi tạo chúng trong method onCreate(), vì nó sẽ chỉ được gọi 1 lần trong activity lifetime.

### The Visible Lifetime

Activity visible lifetime nằm giữa onStart và onStop method. Giữ 2 method này, activity sẽ visible với user mặc dù nó có thể không đuoc focus hoặc bị che mờ. Activity có thể lặp lại visible lifetime nhiều lần trong cả full lifetime như là việc nó chuyển qua lại giữa foreground và background. Trong trường hợp đặc biệt, android sẽ kill activity trong visible lifetime mà không gọi đên method onStop.

Method onStop nên được sử dụng để dừng việc xử lý animations, thread, sensor lisners, GPS lookups, timers, Services hoặc một số xử lý dùng để update UI ( các xử lý gây tiêu tốn resources để update UI trong khi nó không visible).

Sử dụng method onStart (hoặc onRestart) để khôi phục lại các xử lý trên khi UI quay lại trạng thái visible.

Method onRestart được sử dụng để implement một sô xử lý đặc biệt chỉ trong trường hợp activity khởi động lại trong full lifetime.

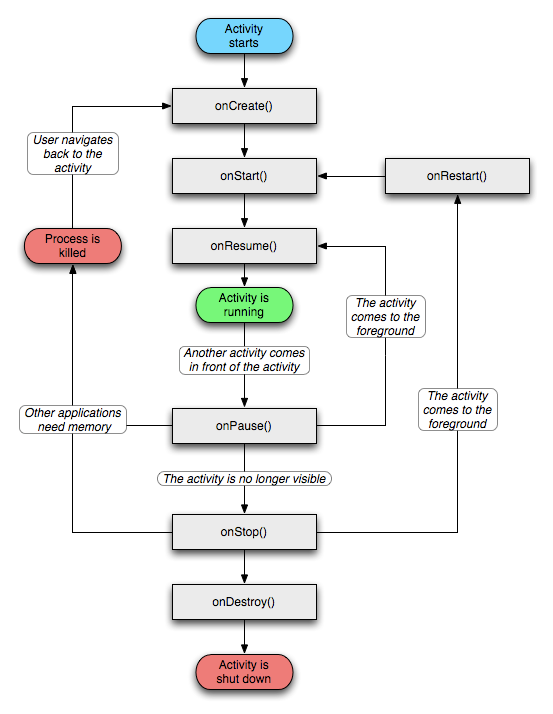
Method onStart/ onStop được sử dụng để register, unregister Broadcast Receivers (được dùng để update UI).

### The Active Lifetime

Active lifetime nằm giữa onResume và onPause method, ở trạng thái foreground. Activity lặp lại trạng thái active lifetime nhiều lần nên cố gắng xử lý code trong 2 method onResume và onPause nhanh.

Trước khi method onPause(), android gọi method onSaveInstanceState, method này dùng để save lại trạng thái UI vào Bundle object, cái mà sẽ được truyền vào onCreate() hay onResotoreInstanceState() method. Sử dụng onSaveInstanceState để giữ lại trạng thái của UI ( ví dụ như checkbox, edit text…) để đảm bảo activity có thể hiển thị cùng một UI như trước khi nó bị invisible.

Hầu hết các application sử dụng onPause() method để save lại trạng thái của activity, vì ở trạng thái này activity có thể bị kill bởi system. Tùy thuộc vào kiến trúc của application mà có thể xử lý cả việc dừng thread, process, BroadCast Receivers trong khi activity không ở background.



## Saving activity state

Trong phần activity life cycle ở trên đã để cập ngắn gọn đến việc khi activity ở trạng thái paused hay stopped, trạng thái của activity được giữ lại. Vì vậy bất cứ sự thay đổi nào của user sẽ được giữ lại trong bộ nhớ, và sau khi activity quay lại foreground, thông tin được cập nhật đúng.

Tuy nhiên trong một số trường hợp system destroy activity để giải phóng bộ nhớ, nó sẽ không thể được phục hồi với trạng thái cũ. Trong trường hợp như vậy để đảm bảo những thông tin quan trọng của activity được duy trì, phải implement thêm một callback method : onSaveInstanceState(). Method này sẽ được gọi trước khi activity bị destroy. Data sẽ được lưu trữ trong object Bundle, là nơi có thể lưu trữ thông tin dưới dạng name-value. Nếu system kill activity, và user vào lại activity đấy, hệ thông sẽ truyền object Bundle đến method onCreate().

@Override

**protected** **void** onSaveInstanceState(Bundle outState) {

// **TODO** Auto-generated method stub

**super**.onSaveInstanceState(outState);

outState.putString(BUTTON\_TEXT, buttonText);

outState.putBoolean(ORIENTATION, mScreenState);

}

Bundle object sẽ được truyền đến method onCreate().

@Override

**public** **void** onCreate(Bundle savedInstanceState) {

**super**.onCreate(savedInstanceState);

setContentView(R.layout.b);

mScreenState = savedInstanceState != **null** ? savedInstanceState.getBoolean(ORIENTATION) : **false**;

buttonText = savedInstanceState != **null** ? savedInstanceState.getString(BUTTON\_TEXT) : ORIENT\_LANDSCAPE;

}

Android mặc định đã implement method onSaveInstanceState để giữ lại trạng thái của view, ví dụ như edit text giữ được giá trị mà user input, checkbox giữ được trạng thái checked hay là uncheked. Chỉ có một yêu cầu là view phải có id, nếu không có id nó sẽ không thể giữ được trạng thái trước.