Service

# Giới thiệu

Một Service là một application component, nó thực hiện các lệnh dưới background (chạy ngầm) và không cần phải có giao diện người sử dụng. Một chương trình khác có thể kêu một service chạy và service sẽ vẫn chạy dù ta tắt chương trình gọi nó.

# Service có 2 loại:

### Started:

Service loại này sẽ chạy khi gọi startService(). Sau khi chạy, nó sẽ chạy ngầm và không tắt khi chương trình gọi nó bị hủy. Service dạng này thường không trả về giá trị cho chương trình gọi nó. Vd: việc download hay upload một file trên mạng, khi thao tác đã xong nó sẽ tự dùng nó.

Bound:

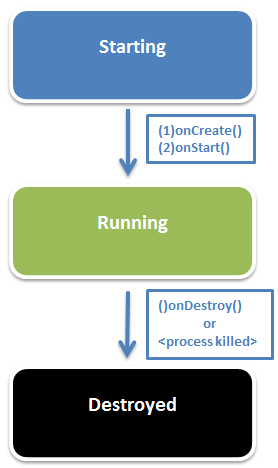
Service dạng “bound” (ràng buộc) khi một chương trình bind (liên kết) nó bằng cách gọi bindService(). Một Bound service là dạng giao tiếp client-server với chương trình gọi nó để gởi yêu cầu, lấy kết quả… Một bound service còn chạy khi còn một chương trình liên kết đến nó. Nhiều chương trình có thể có liên kết đến nó cùng một thời điểm. Khi tất cả đều không còn liên kết thì nó sẽ bị hủy.

# Cơ bản về service

Để tạo service ta cần tạo ra một lớp kế thừa từ Service. Và cần override lên một số phương thức để điều khiển service cũng như ràng buộc nó. Các phương thức quan trọng là:

* **onStartCommand():** Hệ thống gọi phương thức này khi một thành phần nào đó vd: activity, gọi hàm startService(). Khi phương thức này thực thi thì service bắt đầu chạy và chạy ngầm. Khi công việc đã xong ta có thể gọi stopSelf() hoặc stopService() để kết thúc service.
* **onBind():** Hệ thống gọi phương thức này khi một thành phần muốn kết nối đến service bằng cách gọi bindService(). Khi dùng nó ta cần cung cấp một interface để client sử dụng nó để giao tiếp với service bằng cách trả về một Ibinder. Nếu không muốn cho kết nối ta sẽ trả về null.
* **onCreate():** hệ thống gọi phương thức này khi service lần đầu tiên được tạo, nó giúp thực hiện các thao tác chạy một lần (trước khi chạy onStartCommand() hoặc onBind()). Nếu service đang chạy, phương thức này sẽ không thực hiện.
* **onDestroy():** Hệ thống gọi phương thức này khi service không còn sử dụng nữa và được hủy. Phương thức này giúp giải phóng các tài nguyên như thread, các listener, các receiver…. Đây là phương thức cuối cùng của service.

Hệ thống android chỉ force-stop một service chỉ khi hết bộ nhớ.

# Khai báo trong manifest

Cũng giống như Activity, ta phải khai báo service trong manifest.

Ta khai báo 1 cặp thẻ <service> trong cặp thẻ <application> với thuộc tính android:name bằng tên của file java.

<application ……….>

……………………………………………….

<service android:enabled=*"true"* android:name=*".MyService"*>

</service>

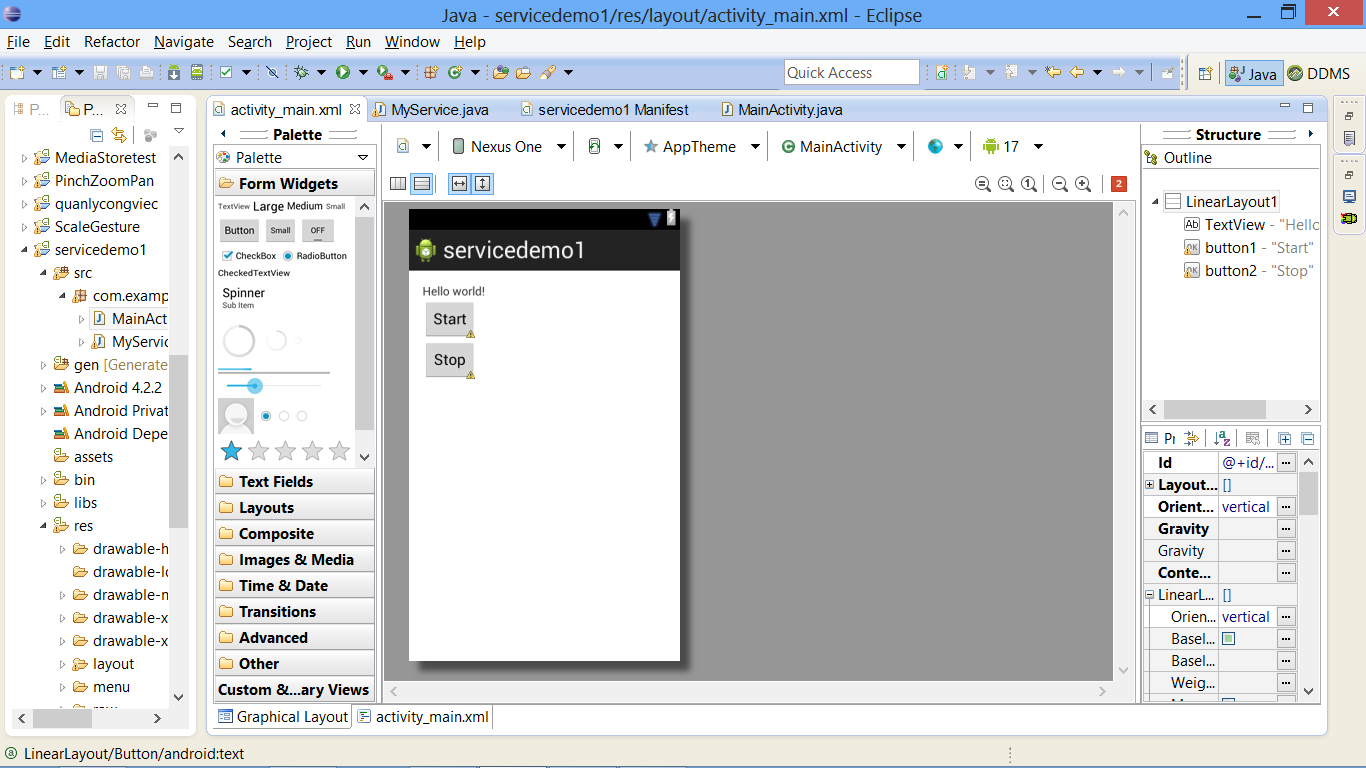
</application>

Muốn service là riêng tư (private) của chương trình thì ta thêm thuộc tính android:exported=”false”

Thực hành

# Bài 1: Viết chương trình chơi nhạc bằng service (không tắt khi thoát khỏi chương trình).

1. Tạo project, thiết kế giao diện gồm 2 nút, start (để chơi nhạc) và stop (để tắt nhạc), vào java ánh xạ và bắt sự kiện cho 2 nút.



1. Tạo một class mới tên MyService kế thừa từ lớp Service và override lên các hàm như sau:

**public** **class** MyService **extends** Service{

@Override

**public** IBinder onBind(Intent arg0) {

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

**return** **null**;

}

@Override

**public** **void** onCreate() {

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

**super**.onCreate();

}

@Override

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

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

**super**.onDestroy();

}

@Override

**public** **int** onStartCommand(Intent intent, **int** flags, **int** startId) {

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

**return** **super**.onStartCommand(intent, flags, startId);

}

}

1. Trong class khai báo một biến toàn cục

MediaPlayer player;

1. Trong hàm onCreate viết như bên dưới để toast và khởi tạo đối tượng player:

Toast.*makeText*(**this**, "service duoc tao",500).show();

player=MediaPlayer.*create*(**this**,R.raw.*a*);

player.setLooping(**true**);

1. Trong onStartCommand() thêm như sau để play nhạc.

Toast.*makeText*(**this**, "service chay",500).show();

player.start();

1. Trong onDestroy() thêm như sau:

player.stop();

Toast.*makeText*(**this**, "service huy",500).show();

1. Mở Manifest thêm vào trong thẻ <application> như sau:

<service android:enabled=*"true"* android:name=*".MyService"*></service>

1. Quay về chương trình chính trong sự kiện nút start ta gọi service như sau:

startService(**new** Intent(getApplicationContext(),MyService.**class**));

1. Sự kiện nút stop ta viết để tắt service như sau:

stopService(**new** Intent(getApplicationContext(),MyService.**class**));

1. Chạy chương trình để thấy kết quả.

Tiếp theo ta phát triển để có 1 thread đếm số hiện liên tục ra Toast để thấy rằng service chạy ngầm

1. Mở lại file MyService.java. Khai báo các biến toàn cục

**public** Handler handler=**new** Handler();

**boolean** isRunning=**false**;

Runnable chay;

**int** tang=0;

1. Xây một phương thức mới như sau:

**public** **void** tangthoigian()

{

**if**(isRunning==**true**)

{

chay=**new** Runnable() {

@Override

**public** **void** run() {

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

tang++;

Toast.*makeText*(getApplicationContext(), "tang"+tang,500).show();

tangthoigian();

};

};

handler.postDelayed(chay,5000);

}

}

1. Trong onStartCommand() viết thêm đoạn sau:

isRunning=**true**;

tangthoigian();

1. Trong onDestroy() viết thêm như sau:

handler.removeCallbacks(chay);

1. Chạy chương trình và xem kết quả:

|  |  |  |
| --- | --- | --- |
| Nhấn Start, service bắt đầu chạy | Thoát khỏi chương trình service vẫn chạy ngầm | Chạy lại chương trình và nhấn stop để hủy service |