Trường ĐH Khoa Học Tự Nhiên Tp. Hồ Chí Minh
TRUNG TÂM TIN HỌC

Lập trình Android

Bài 21: *Services*

Phòng LT & Mạng

http://csc.edu.vn/lap-trinh-va-csdl

2019

Nội dung

1. Services
2. IntentService
3. ResultReceiver
4. BroadcastReceiver

Services

- **Service** là một thành phần chạy ở **background** và không tương tác trực tiếp với người dùng.
- **Service** không có giao diện đồ hoạ người dùng, không gắn với lifecycle của một activity
- **Service** có thể được sử dụng để thực hiện các tác vụ tốn nhiều thời gian, như là kiểm tra dữ liệu mới, xử lý dữ liệu…
- **IntentService** là cách thực hiện một tác vụ trên một tiểu trình đơn (single background thread)
- Phân biệt với **AsyncTask**, luôn gắn với một activity. Nếu activity hoặc destroyed hoặc configuration thay đổi thì AsyncTask không thể cập nhật UI khi hoàn tất. Nếu thực hiện một background task trong thời gian ngắn sau đó cập nhật giao diện, nên dùng AsyncTask.
- **IntentService** được sử dụng trong trường hợp thực hiện một tác vụ trong thời gian dài, không phụ thuộc vào activity đang được xem. Có thể chuyển sang activity khác hoặc app có thể bị tạm dừng (paused) và IntentService vẫn tiếp tục chạy ở background.

Tạo IntentService

- Định nghĩa class trong ứng dụng kế thừa từ **IntentService** và override phương thức **onHandleIntent**

```
public class MyIntentService extends IntentService {
    public MyIntentService() {
        super("MyIntentService"); // tên của service, có ý nghĩa trong việc debug
    }

    @Override
    public void onCreate() {
        super.onCreate(); // lưu ý gọi super.onCreate()
        // nếu cần đến context, gọi getApplicationContext ở đây
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        // các xử lý khi service này được gọi
    }
}
```

- Đăng ký service trong **AndroidManifest.xml**

export=false nghĩa là app khác không thể sử dụng service này

```
<application …>
    <service
        android:name=".MyIntentService"
        android:exported="false" />
</application>
```

Thực thi IntentService

- Có thể thực thi **IntentService** thông qua **Intent**:

```
public class MainActivity extends AppCompatActivity {

    public void launchService() {
        Intent i = new Intent(this, MyIntentService.class);
        i.putExtra("foo", "bar");
        i.putExtra("receiver", resultReceiver);
        startService(i);
    }
}
```

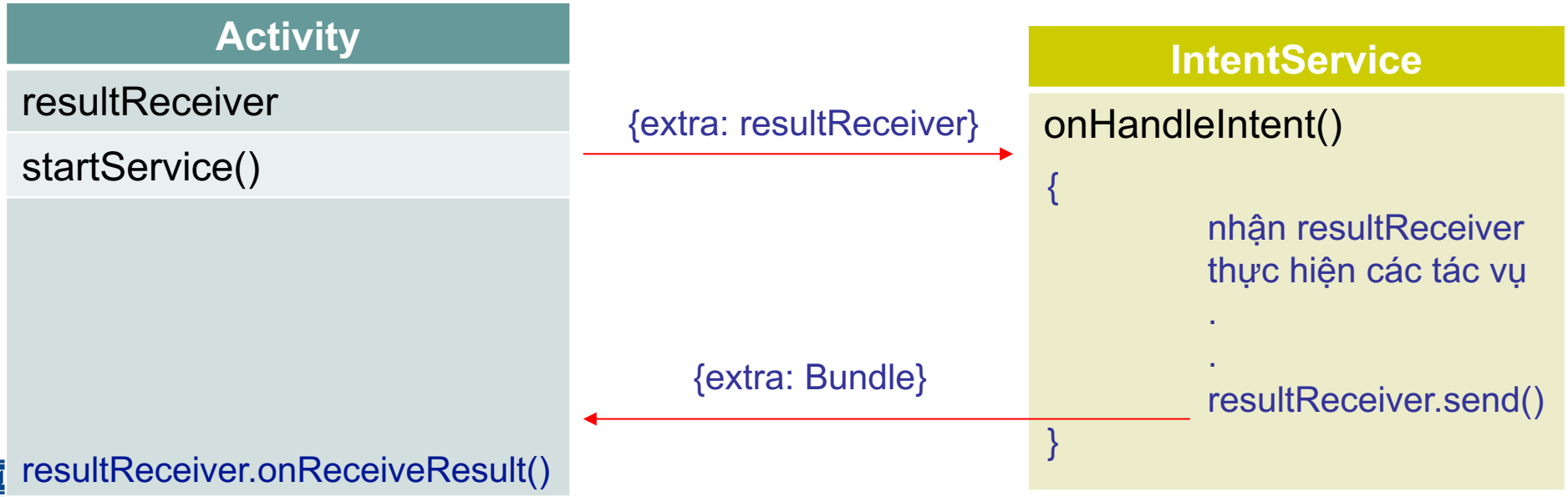
- Có thể bắt đầu một **IntentService** từ Activity hay Fragment tại bất kỳ thời điểm nào. Khi phương thức **startService**() được gọi, IntentService sẽ thực hiện các công việc được định nghĩa trong phương thức **onHandleIntent**() và sau đó kết thúc service.

Liên lạc giữa Service và Application

- Sau khi thực thi một service, cần cơ chế để service trả dữ liệu ngược lại application. Application sẽ có hành động tương ứng với kết quả do service trả về. Có 2 cách tiếp cận:
  - **ResultReceiver**: dùng để gửi dữ liệu tới activity, dùng khi service chỉ kết nối tới application đã tạo ra nó
  - **BroadcastReceiver**: dùng để broadcast event và có thể được bắt lấy bởi bất kỳ ứng dụng nào, dùng khi service cần liên lạc với nhiều thành phần/ứng dụng muốn lắng nghe kết quả từ service

Sử dụng ResultReceiver

- Cách thức hoạt động của ResultReceiver có thể được mô tả thông qua sơ đồ sau:
  - **Activity** giữ một đối tượng **ResultReceiver**
  - Khi gọi hàm **startService()** để bắt đầu một **IntentService**, gắn thêm **extra** là đối tượng **ResultReceiver**
  - Trong hàm **onHandleIntent()**, **IntentService** nhận được đối tượng **resultReceiver** từ **Activity**.
  - Sau khi thực hiện xong service, **resultReceiver** gọi hàm **send**() kèm với dữ liệu kết quả để trigger hàm **onReceiveResult**



Sử dụng ResultReceiver (2)

- Tạo class kế thừa từ **ResultReceiver,** override phương thức **onReceiveResult**. Khai báo thêm interface **Receiver** đóng vai trò listener callback xử lý kết quả trả về.

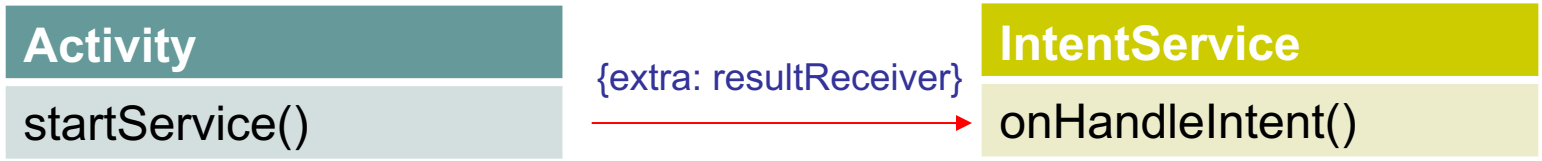
```
public class MyResultReceiver extends ResultReceiver {
    private Receiver receiver;

    public MyResultReceiver(Handler handler) {
        super(handler);
    }

    public void setReceiver(Receiver receiver) {
        this.receiver = receiver;
    }

    public interface Receiver {
        void onReceiveResult(int resultCode, Bundle resultData);
    }

    @Override
    protected void onReceiveResult(int resultCode, Bundle resultData) {
        if (receiver != null) {
            receiver.onReceiveResult(resultCode, resultData);
        }
    }
}
```

X

Sử dụng ResultReceiver (3)

- Khai báo **receiver** trong **activity** và *truyền* vào **IntentService** như một **extra**

```
public class MainActivity extends AppCompatActivity implements MyResultReceiver.Receiver {
    public MyResultReceiver resultReceiver;

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

    public void launchService(View view) {
        Intent i = new Intent(this, MyIntentService.class);
        i.putExtra("foo", "bar");
        i.putExtra("receiver", resultReceiver);
        startService(i);
    }

    …
```



Sử dụng ResultReceiver (4)

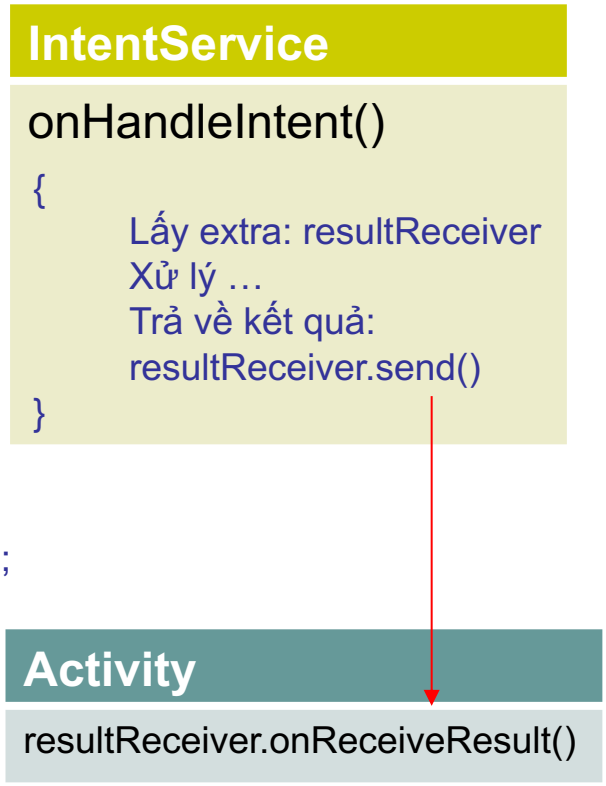
- Thiết lập **callback** để xử lý dữ liệu nhận được từ **service** bằng cách cho **Activity** implements từ interface **MyResultReceiver.Receiver** và override phương thức **onReceiveResult**:

```
…

public void setupServiceReceiver() {
        resultReceiver = new MyResultReceiver(new Handler());
        resultReceiver.setReceiver(this);
    }

    @Override
X   public void onReceiveResult(int resultCode, Bundle resultData) {
        if (resultCode == RESULT_OK) {
            String resultValue = resultData.getString("resultValue");
            Toast.makeText(MainActivity.this, resultValue, Toast.LENGTH_SHORT).show();
        }
    }
}
```

Sử dụng ResultReceiver (5)

- Cập nhật phương thức **onHandleIntent** của class **IntentService**:

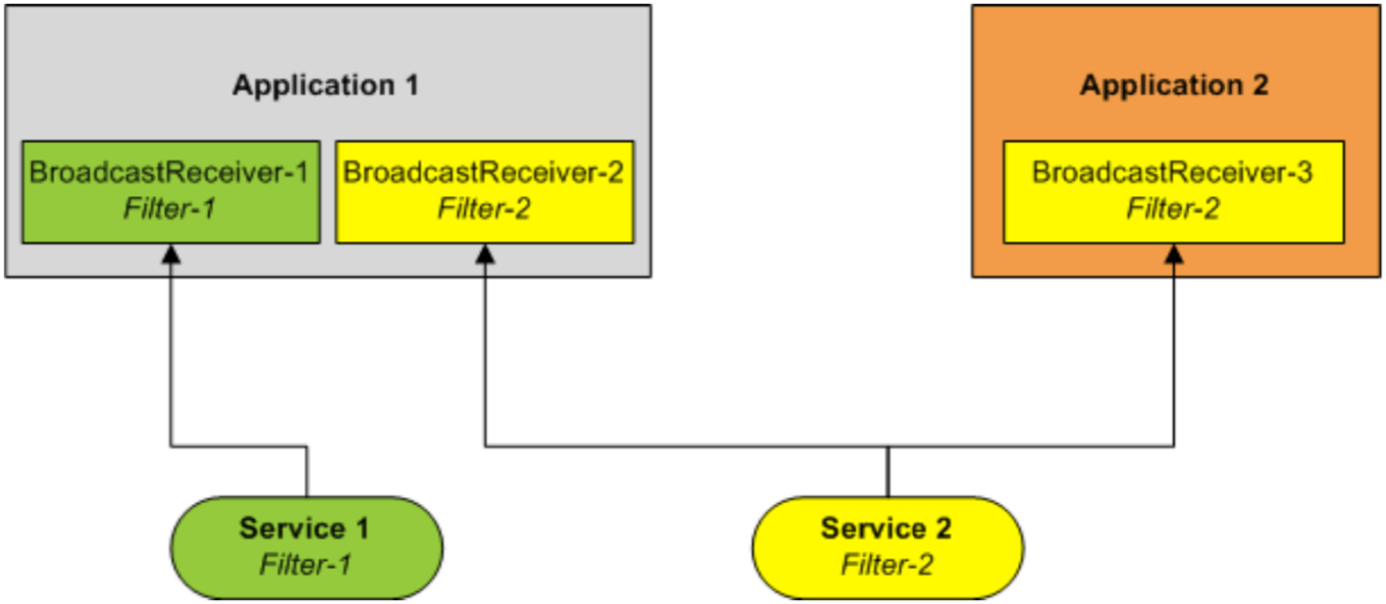
```
public class MyIntentService extends IntentService {
    public MyIntentService() {
        super("MyIntentService");
    }

    @Override
    public void onCreate() {
        super.onCreate();
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        ResultReceiver receiver = intent.getParcelableExtra("receiver");
        String value = intent.getStringExtra("foo");
        Bundle bundle = new Bundle();
        bundle.putString("resultValue", "My result value: " + value);
        receiver.send(Activity.RESULT_OK, bundle);
    }
}
```



- **Receiver** gọi phương thức **send** sẽ trigger phương thức callback **onReceiveResult** bên trong activity

Sử dụng BroadcastReceiver

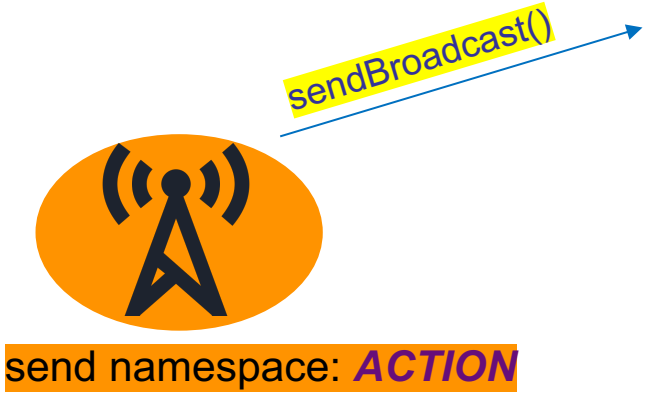
- Sử dụng **ResultReceiver** chưa hẳn là một cách tiếp cận đúng. Có vài trường hợp như thoát app, thì result receiver sẽ không hoạt động khi app được mở lại. Mỗi activity muốn nhận dữ liệu phải tham chiếu đến đối tượng receiver và truyền nó vào trong service.
- Để khắc phục vấn đề trên, có thể sử dụng **BroadcastReceiver** để nhận dữ liệu từ service khi app được mở lại, mặt khác có thể nhiều apps có thể nhận dữ liệu từ cùng một service.
- Minh hoạ sau sử dụng **LocalBroadcastManager** cho phép liên lạc nội bộ giữa service và activity trong cùng một application.



Sử dụng BroadcastReceiver (2)

- Xây dựng **IntentService**, service này sẽ gửi **broadcast** đến tất cả các application muốn lắng nghe dữ liệu dựa trên namespace **ACTION**:

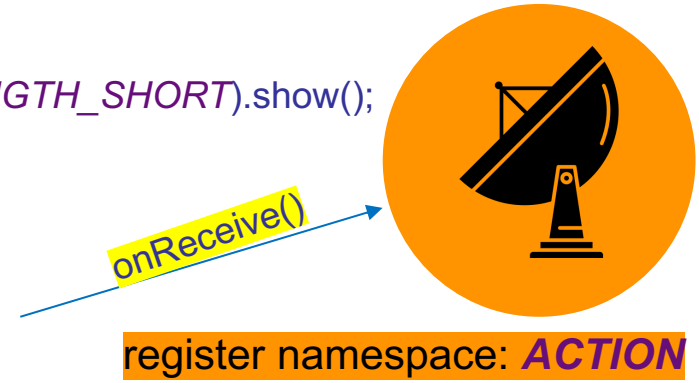
```
public class MyIntentService extends IntentService {
    public static final String ACTION = "edu.csc.intentservice.MyIntentService";

    public MyIntentService() {
        super("MyIntentService");
    }

    @Override
    public void onCreate() {
        super.onCreate();
    }

    @Override
    protected void onHandleIntent(Intent intent) {
        String value = intent.getStringExtra("foo");
        Intent i = new Intent(ACTION);
        i.putExtra("resultCode", Activity.RESULT_OK);
        i.putExtra("resultValue", "My result value: " + value);
        LocalBroadcastManager.getInstance(this).sendBroadcast(i);
    }
}
```



Sử dụng BroadcastReceiver (2)

- Khai báo đối tượng **BroadcastReceiver** trong activity và đăng ký lắng nghe với namespace **ACTION**

```
private BroadcastReceiver receiver = new BroadcastReceiver() {
    @Override // callback để thực hiện xử lý sau khi nhận được dữ liệu từ service
    public void onReceive(Context context, Intent intent) {
        int resultCode = intent.getIntExtra("resultCode", RESULT_CANCELED);
        if (resultCode == RESULT_OK) {
            String resultValue = intent.getStringExtra("resultValue");
            Toast.makeText(MainActivity.this, resultValue, Toast.LENGTH_SHORT).show();
        }
    }
};

@Override
protected void onResume() {
    super.onResume(); // đăng ký lắng nghe thông điệp broadcast với namespace ACTION
    IntentFilter filter = new IntentFilter(MyIntentService.ACTION);
    LocalBroadcastManager.getInstance(this).registerReceiver(receiver, filter);
}

@Override
protected void onPause() {
    super.onPause(); // huỷ đăng ký lắng nghe thông điệp broadcast khi activity không còn visible với user
    LocalBroadcastManager.getInstance(this).unregisterReceiver(receiver);
}
```



Q&A