LẬP TRÌNH TRÊN THIẾT BỊ DI ĐỘNG

CHƯƠNG 2:

CÁC THÀNH PHẦN CƠ BẢN CỦA ỨNG DỤNG ANDROID

- Giảng viên: Ths. Nguyễn Trung Hiếu
- Email: hieunt.tg@ptithcm.edu.vn
- Mobie: 0983051825

1

PHẦN I:
CÁC THÀNH PHẦN CƠ BẢN

Mục tiêu

- Kết thúc bài học này bạn có khả năng
  - Hiểu về các thành phần cơ bản của ứng dụng Android
  - Hiểu về Intent, Context
  - Hiểu về Activity và vòng đời của Activity

NỘI DUNG

- Phần I: Các thành phần cơ bản của Android
  - Activity
  - Service
  - ContentProvider
  - BroadcastReceiver
  - Intent, Context
- Phần II: Activity và vòng đời của Activity
  - Activity - Task, stack
  - Vòng đời của Activity

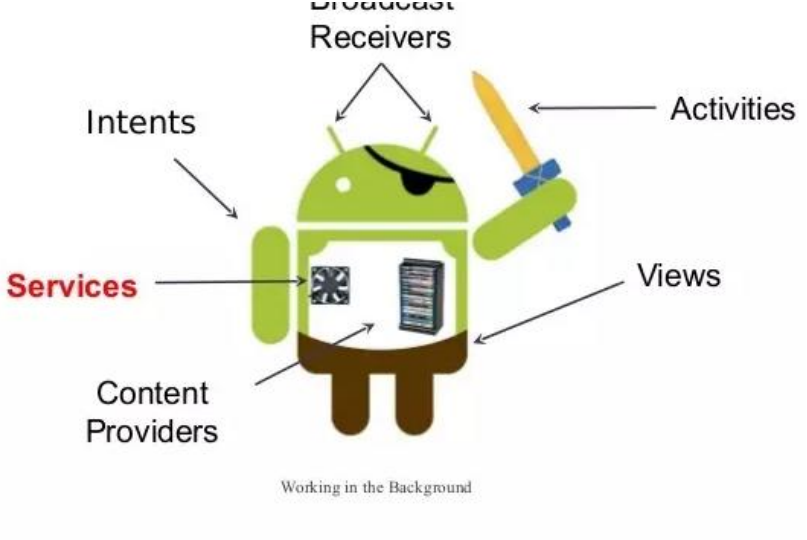
- Các thành phần:
  - ❖ Activity
  - ❖ Service
  - ❖ ContentProvider
  - ❖ BroadcastReceiver
  - ❖ Intent

Activity

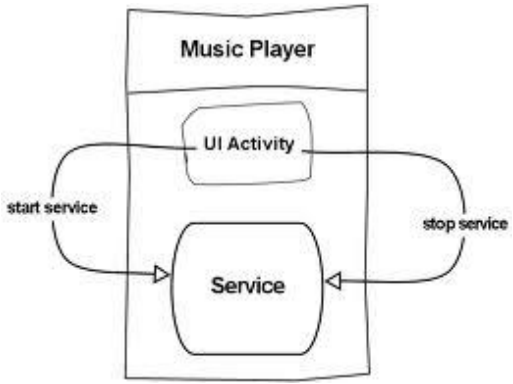
- Activity là tầng trình diễn của ứng dụng. Giao diện người dùng của ứng dụng được xây dựng dựa trên một hoặc nhiều Activity. Mỗi Activity đều là một lớp kế thừa từ lớp Activity. Activity sử dụng Fragment và View để bố trí và hiển thị thông tin và tương tác với hoạt động của người dùng
- Activity bao gồm 1 Class java và 1 file .xml dùng để thiết kế giao diện người dùng.

Service

- Service là một thành phần chạy ngầm trên hệ điều hành để thực hiện các hoạt động dài hạn mà không cần phải tương tác với người sử dụng và nó hoạt động ngay cả khi ứng dụng bị phá hủy. Một khi được gọi, dịch vụ này có thể chạy ở chế độ nền vô thời hạn, thậm chí cả khi thành phần đã khởi động nó bị phá hủy



- Có các loại Service:
  - **Foreground Service**: Một Foreground Service thực hiện một số thao tác mà người dùng chú ý, có thể thấy rõ ràng.
  - ➢ Ví dụ một ứng dụng nghe nhạc có thể chơi một bản nhạc và control nó bằng Foreground Service.

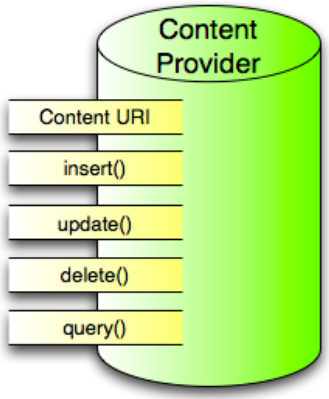


Service

- Có các loại Service:
  - **Background Service:** Một Background Service sẽ thực hiện các hoạt động mà không được người dùng chú ý trực tiếp
  - ➢ Ví dụ một ứng dụng sử dụng một service để thu gom bộ nhớ chẳng hạn thì service là một Background Service.
  - ***Bound Service:*** Một service được gọi là Bound khi một thành phần của ứng dụng ràng buộc với nó bởi lời gọi bindService(). Một Bound Service cung cấp một giao diện Client - Server cho phép các thành phần tương tác với nó: gửi yêu cầu, nhận kết quả.

ContentProvider

- Content provider là một thành phần để quản lý truy cập dữ liệu, nó cung cấp các phương thức khác nhau để các ứng dụng có thể truy cập dữ liệu từ một ứng dụng khác bằng cách sử dụng ContentResolver.
- Content Provider có thể giúp cho một ứng dụng quản lý quyền truy cập đến dữ liệu được lưu bởi ứng dụng đó, hoặc các ứng dụng khác, và đó là một cách để ta có thể chia sẻ dữ liệu cho các ứng dụng khác nhau.



ContentProvider

Truy cập thông qua truy vấn content://URI

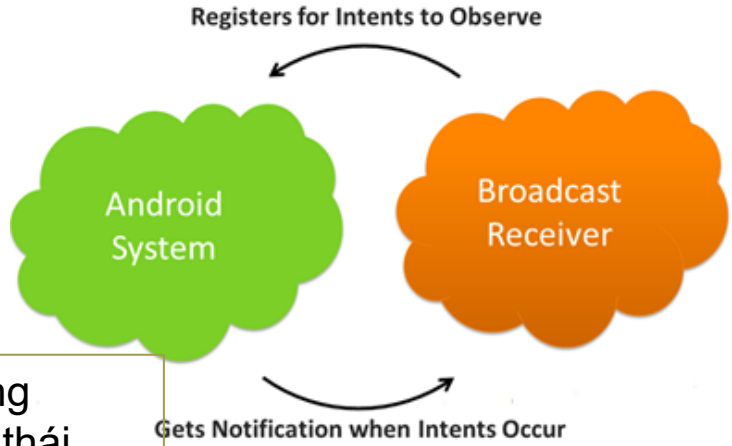
Các phương thức cần được Override trong lớp Content Provider: onCreate(), query(), insert(), delete(), update(), getType():

Tìm hiểu sâu hơn các bài sau

```
ContentResolver cr= Context.getContentResolver();
cr.query(content://android.provider.
Contacts.Phones.CONTACT_URI,…)
```

BroadcastReceiver

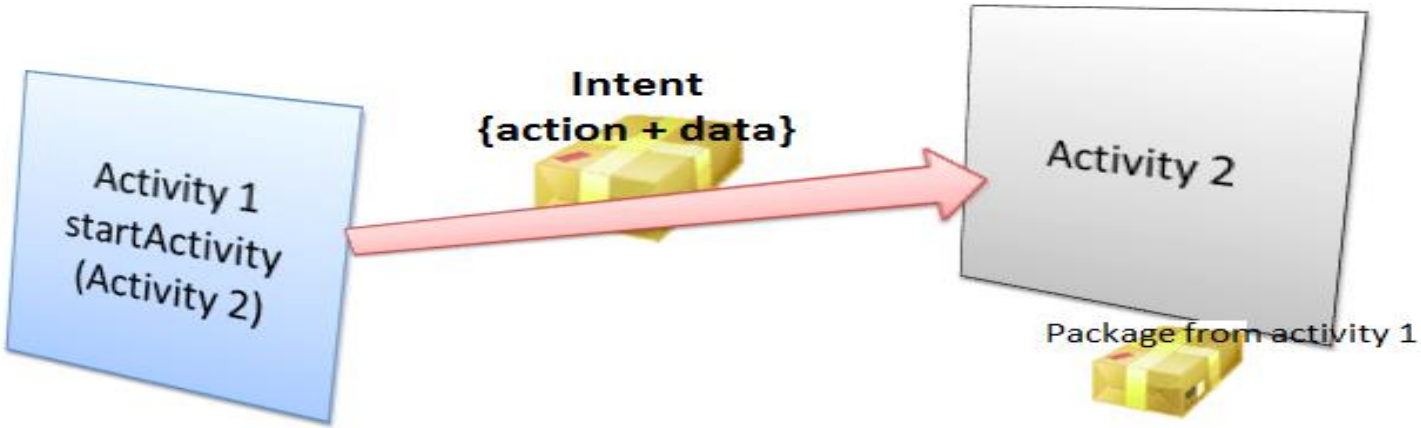
- Đánh thức bởi broadcast hệ thống
- Rất đơn giản – chỉ là onReceive handler
  - ➢ Nhận context và Intent miêu tả broadcast



- ➢ Broadcast Receiver là một trong 4 component lớn trong Android, với mục đích là lắng nghe các sự kiện, trạng thái của hệ thống phát ra thông qua Intent nhờ đó mà các lập trình viên có thể xử lí được các sự kiện hệ thống ở bên trong ứng dụng của mình.
- ➢ Broadcast Receiver có thể hoạt động được cả khi ứng dụng bị tắt đi, nghĩa là ở background chính vì vậy nó thường được sử dụng với service.

Có 2 loại Intent

- Action, Data = Implicit
- Action, Data, Component = Explicit



Có 2 loại Intent

Explicit Intent (intent tường minh): Hiểu đơn giản explicit intents là intent xác định rõ và cụ thể các thành phần tham gia hành động.

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

Implicit Intent (Intent không tường minh): chỉ ra hành động cần được thực hiện (action) và dữ liệu cho hành động đó (data). Khi bạn sử dụng implicit intent, hệ thống Android sẽ tìm kiếm tất cả thành phần thích hợp để start bằng cách cách so sánh nội dung của Intent đc gửi với các Intent filter đc khai báo trong ứng dụng khác. Nếu intent đc gửi đó khớp với intent filter trong một component hoặc một ứng dụng nào đó, thì ngay lập tức hệ thống sẽ khởi động thành phần đó và cung cấp cho nó intent ban đầu đc gửi . Nếu nhiều intent filter tương thích thì hệ thống sẽ hiển thị hộp thoại để người dùng có thể chọn ứng dụng nào sẽ sử dụng.

Intent filter (bộ lọc intent) là một thẻ trong minifest nhằm xác định loại intent mà thành phần chứa intent filter đó muốn nhận.

```
Intent sendIntent = new Intent(Intent.ACTION_VIEW);
sendIntent.setData(Uri.parse(Constant.TYPE_MESSAGE));
startActivity(sendIntent);
```

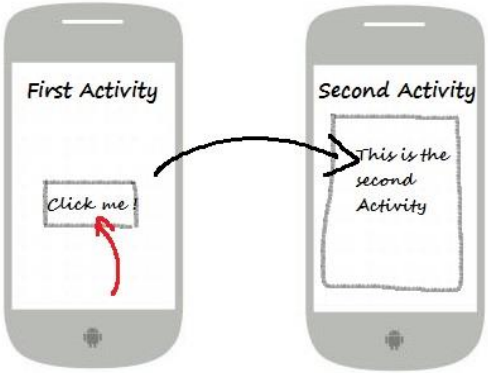
CONTEXT

- Lớp Context cung cấp truy cập tới chức năng và dịch vụ của hệ thống
- Activity và Service kế thừa Context, do đó có thể gọi các phương thức trong Context trực tiếp
- BroadcastReceiver có chứa tham số Context trong tham số đầu vào ở các hàm quản lý sự kiện
- ContentProvider gọi hàm getContext để lấy đối tượng Context

2

PHẦN II:
Activity
và Vòng Đời Của Activity

- Khởi tạo Activity bằng cách gọi startActivity(Intent)
- Subactivity: Là activity được gọi bởi activity khác.
- Gọi Subactivity sử dụng phương thức startActivityForResult
  - Truyền Intent và integer code trong tham số đầu vào
  - Khi subactivity kết thúc, trả lại mã code
  - startActivityForResult là phương thức không đồng bộ

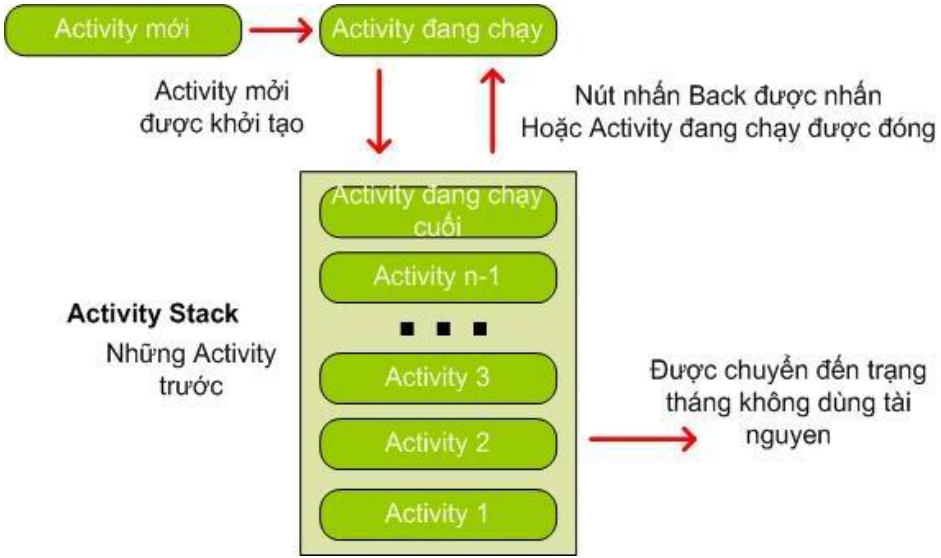


Task

- Android nhóm các activity trong một chương trình vào một công việc chung (hàng đợi các activity liên quan đến nhau)
- Người dùng nhấn nút HOME và khởi tạo một chương trình mới
  - Chuyển task hiện tại sang chế độ nền
  - Bắt đầu task mới, đặt activity mặc định của ứng dụng mới ở đầu Stack
- Nếu ứng dụng được quay lại, task cũ (stack cũ) sẽ được khôi phục

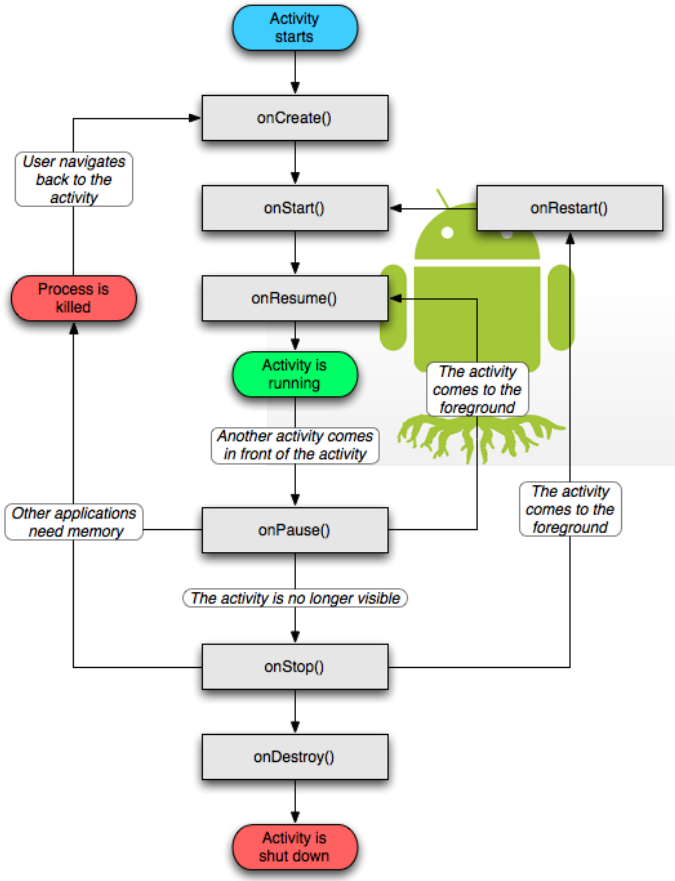
Stack

- Các activity cấu tạo nên Stack
  - ❖ Activity mới sẽ xuất hiện ở đầu Stack
  - ❖ Thông thường, khi nhấn nút back sẽ quay lại activity trước đó

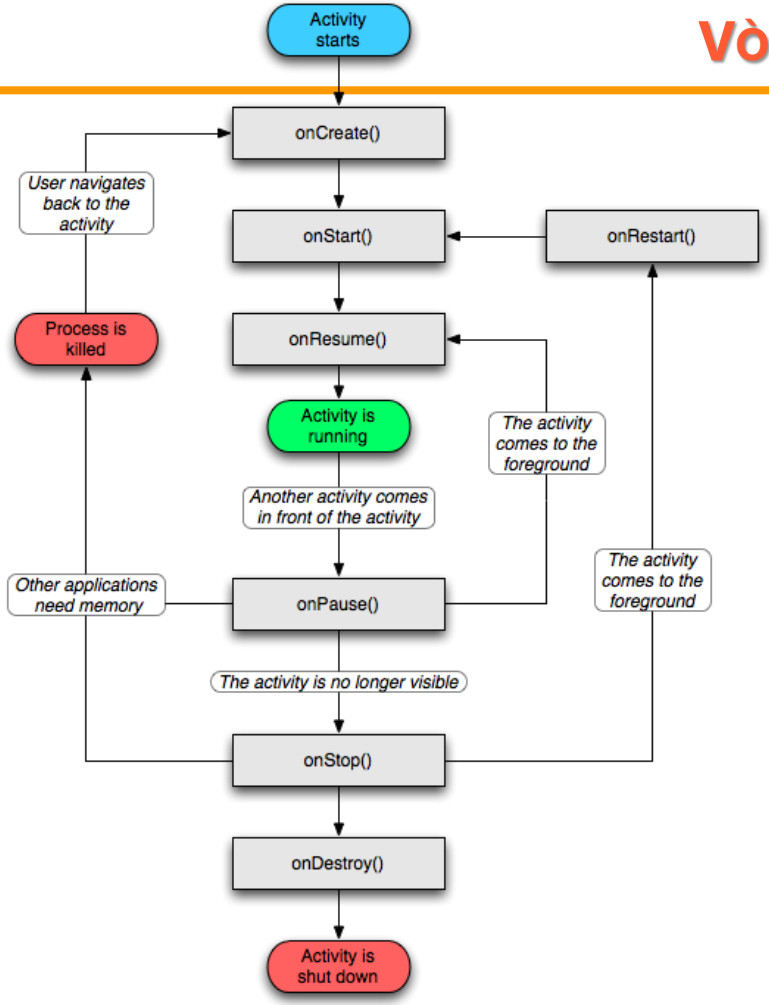


Vòng đời của Activity

- startActiviy đảm bảo Activity được khởi tạo
  - Nếu Activity được khởi tạo, sẽ được đưa lên đầu
  - Activity được quản lý như thế nào?
- Mô hình hướng sự kiện
  - Activity có một số hàm để điều khiển các sự kiện
    - onCreate, onResume, onPause,…
  - Tất cả Activity phải nạp chồng hàm onCreate để thực hiện một việc gì đó
  - Các hàm nạp chồng phải gọi phương thức của superclass

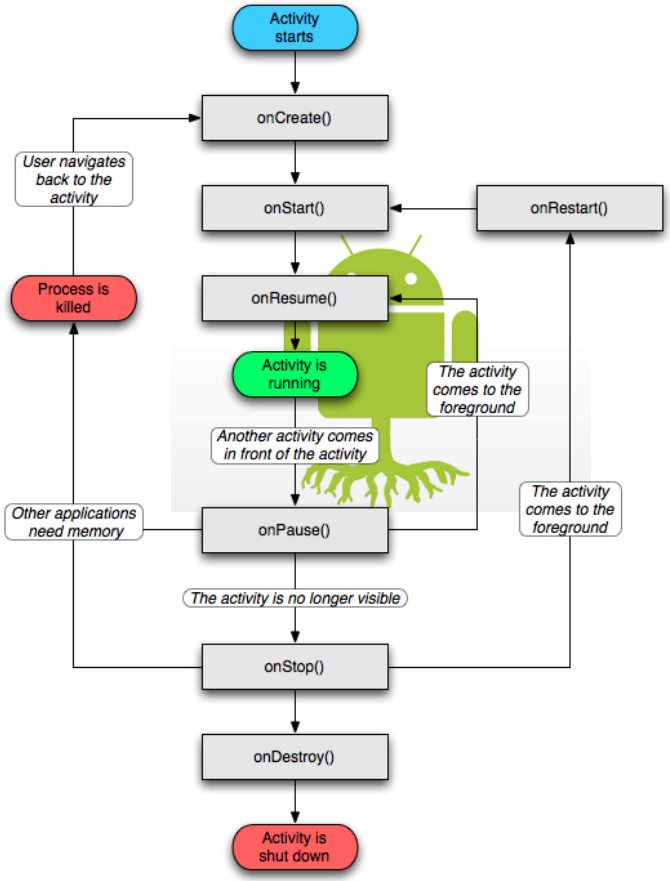


Vòng đời của Activity



Vòng đời của Activity

- Ba trạng thái
  - Kích hoạt (active): ở chế độ nền, đang chạy
  - Tạm dừng (pause): vẫn hiển thị nhưng bị che khuất bởi Activity khác
    - Giống active, nhưng có thể bị hủy nếu dung lượng bộ nhớ thấp
  - Dừng (stop): không hiển thị trên màn hình



VÒNG ĐỜI CỦA ACTIVITY

- onCreate()
  - ❖ Gọi khi Activity đầu tiên được tạo
  - ❖ Chuẩn bị GUI và các bước khởi tạo kh
- onResume()
  - ❖ Gọi khi Activity ở trên đầu Stack
  - ❖ Cập nhật giá trị GUI
    Chú ý: được gọi khi Activity đầu tiên được khởi tạo
- onPause()
  - ❖ Activity chuẩn bị biến mất
  - ❖ Cập nhật các dữ liệu quan trọng, dừng các công việc tốn nhiều tài nguyên

Vòng đời của Activity

START ACTIVITY

- Để gọi activity khác chạy bạn dùng phương thức startActivity

```
Intent i = new
Intent(getBaseContext(),ActivityB.class);
startActivity(i)
```

Put và get data qua intent:

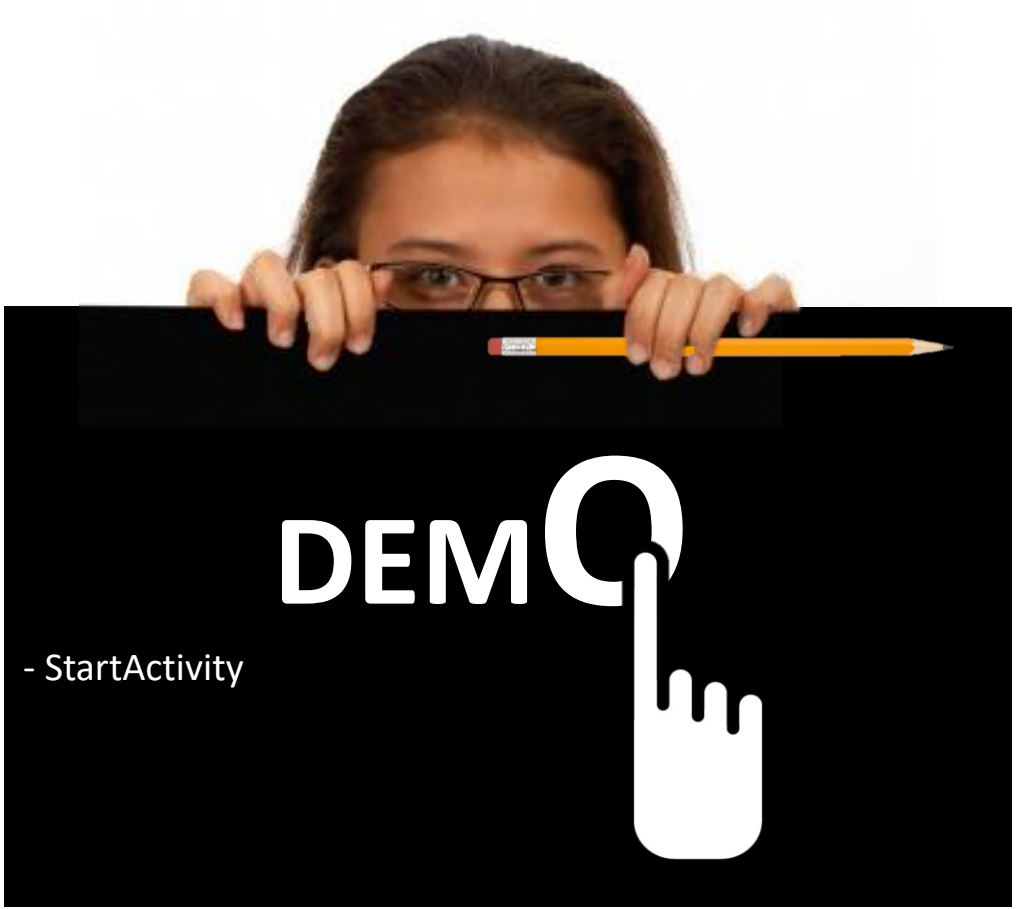
- Gửi:

```
Intent i = new Intent(getBaseContext(),ActivityB.class);

//Set the Data to pass
EditText txtInput = (EditText)findViewById(R.id.txtInput);
String txtData = txtInput.getText().toString();
i.putExtra("txtData", txtData);

startActivity(i);
```

- Nhận

```
Intent i = getIntent();
//The second parameter below is the default string returned
String txtData = i.getExtras().getString("txtData","");
EditText txtInput2 = (EditText)findViewById(R.id.txtInput2);
txtInput2.setText(txtData);
```

Start Activity



PARCELABLE

```
public class SinhVien implements Parcelable
```

```
public SinhVien(Parcel in) {

    this.Fulname = in.readString();
    this.Address = in.readString();
    this.Major = in.readString();
}
```

```
@Override
public int describeContents() {
    return 0;
}
@Override
public void writeToParcel(Parcel parcel, int i) {
    parcel.writeString(Fulname);
    parcel.writeString(Address);
    parcel.writeString(Major);
}
```

```
public static final Creator<SinhVien> CREATOR = new Creator<SinhVien>() {
    @Override
    public SinhVien createFromParcel(Parcel in) {
        return new SinhVien(in);
    }

    @Override
    public SinhVien[] newArray(int size) {
        return new SinhVien[size];
    }
};
```

PARCELABLE

```
SinhVien sv = new SinhVien("Nguyễn Văn A", “PTIT","IT");
Intent  intent = new Intent(this, RegisterActivity.class);

Bundle bundle = new Bundle();
bundle.putParcelable("sv", sv);
bundle.putString("name","Lập trình Android");
intent.putExtra("data",bundle);

startActivity(intent);
```

```
Intent intent = getIntent();

Bundle bundle = intent.getBundleExtra("data");
String name = bundle.getString("name");
SinhVien sv =  bundle.getParcelable("sv");
Student st= (Student) bundle.getSerializable("st");

Toast.makeText(this,sv.getFulname().toString(), Toast.LENGTH_LONG).show();
```

SERIALIZABLE

```
public class Student implements Serializable
```

```
Student st = new Student("Nguyễn Thi B", “PTIT","IT");
Intent  intent = new Intent(this, RegisterActivity.class);

Bundle bundle = new Bundle();

bundle.putSerializable("st", st);
bundle.putString("name","Lập trình Android");
intent.putExtra("data",bundle);
startActivity(intent);
```

```
Intent intent = getIntent();
Bundle bundle = intent.getBundleExtra("data");
String name = bundle.getString("name");

Student st= (Student) bundle.getSerializable("st");
Toast.makeText(this,st.getFulname().toString(), Toast.LENGTH_LONG).show();
```


PTIT
ptit.edu.vn