IUT de Paris - Rives de Seine
Université Paris Cité

R4.Real.11 – Développement pour applications mobiles

– Cours 2 –

Cycle de vie d'une activité et adaptation graphique

Pr Chaouche A.-C.
ac.chaouche@gmail.com

Résumé

Prérequis

- Programmation Java
- Cycle de développement d'Android

Objectifs

- Comprendre le cycle de vie d'une activité
- Adapter les ressources en fonction du matériel
- Internationaliser l'application

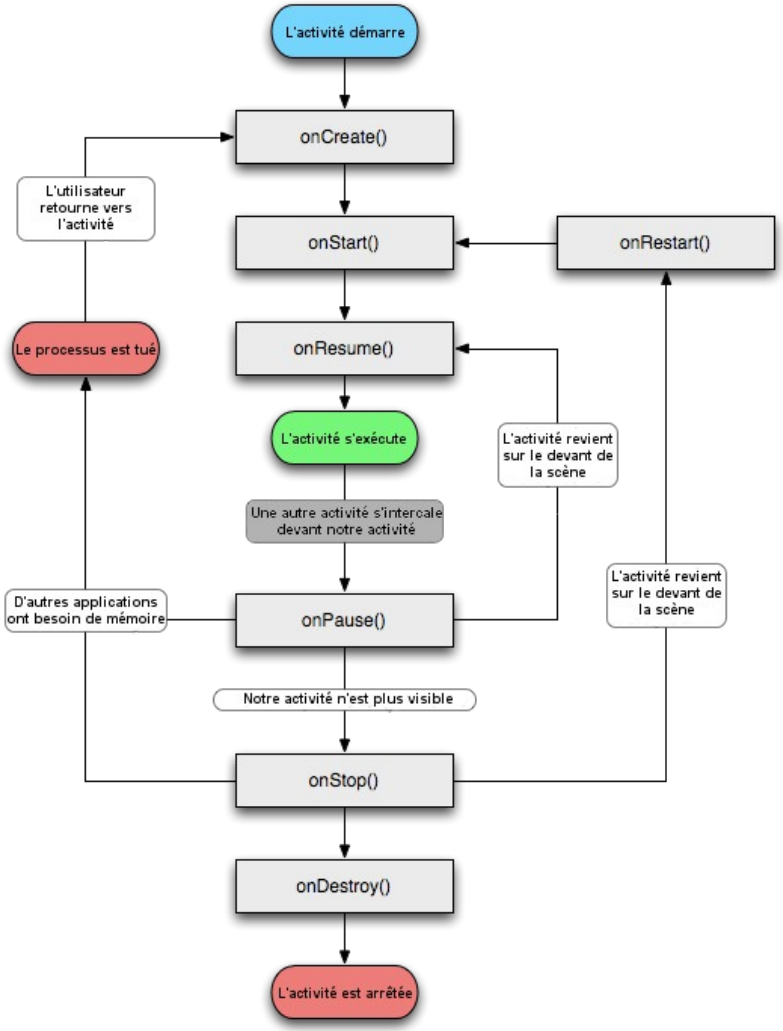
Activité

Cycle de vie (1/2)

Le **changement d'état d'une activité** provoque le déclenchement de la **méthode callback** correspondante.

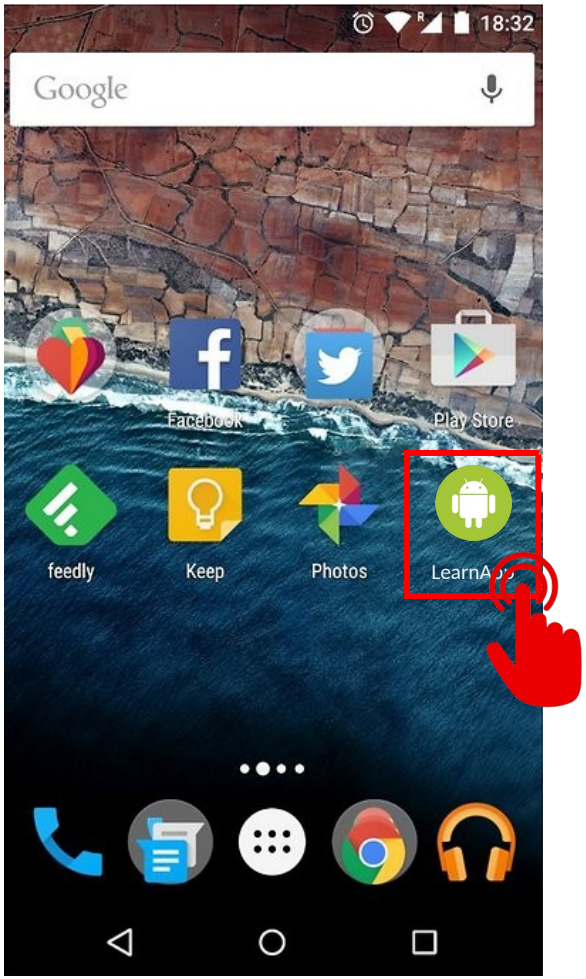
Méthodes callback :

```
void onCreate(...)
void onStart()
void onRestart()
void onResume()
void onPause()
void onStop()
void onDestroy()
```

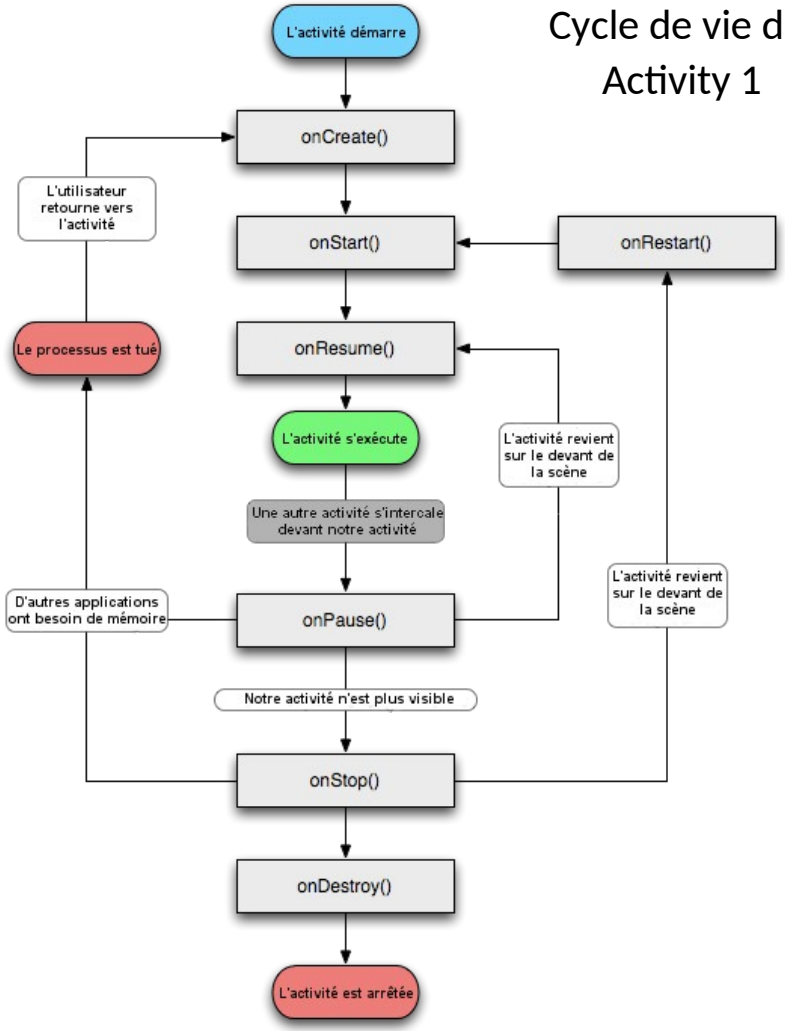


Activité

Cycle de vie (2/2) – Explication

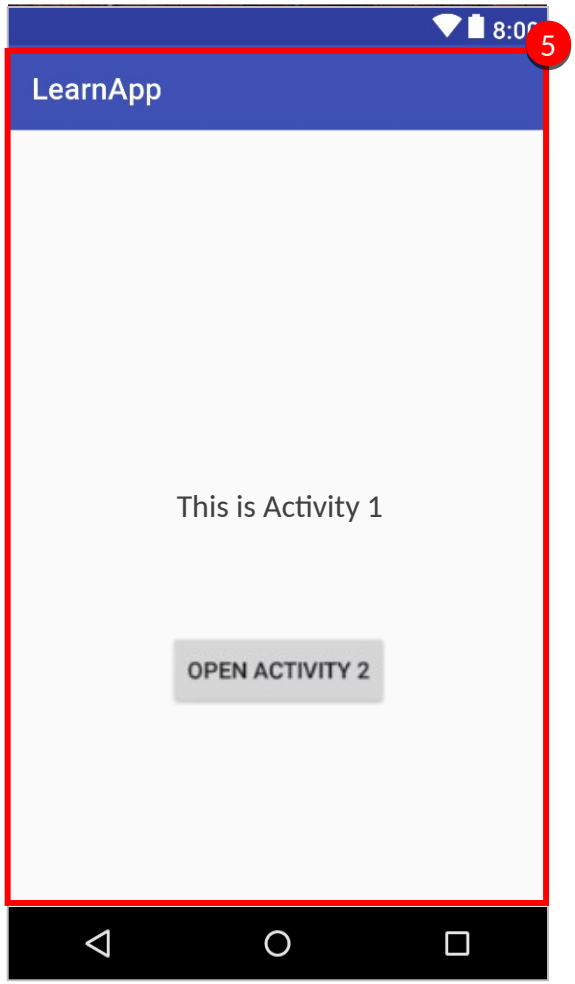


Cycle de vie de Activity 1

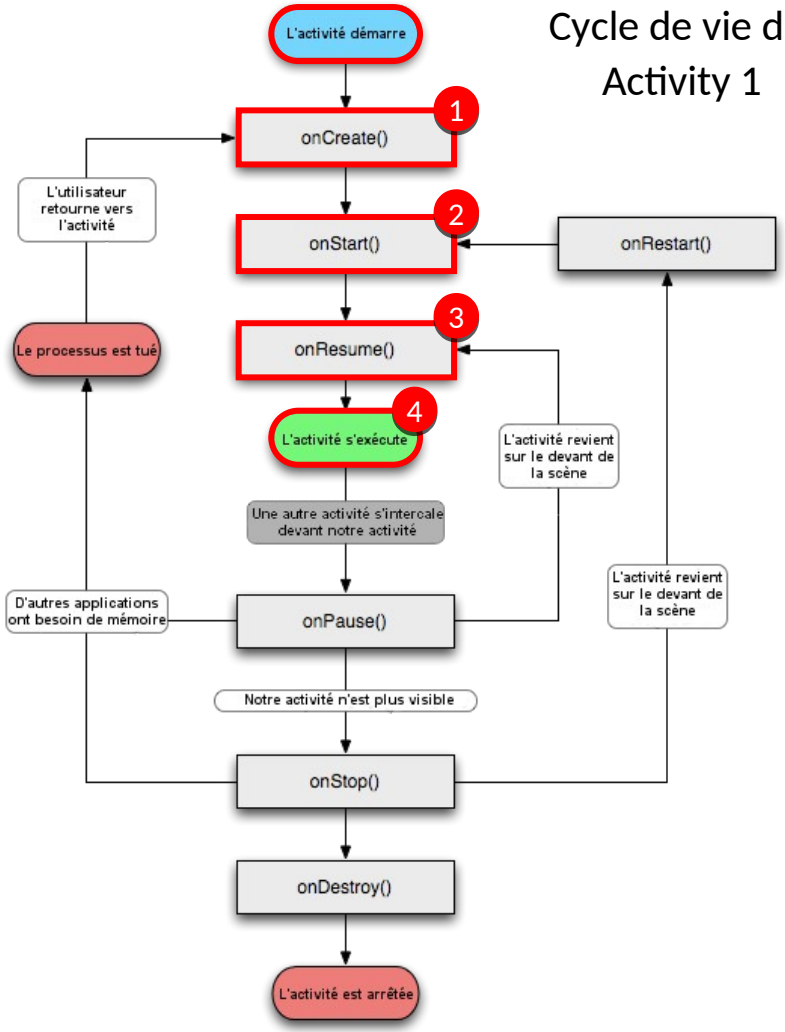


Activité

Cycle de vie (2/2) – Explication

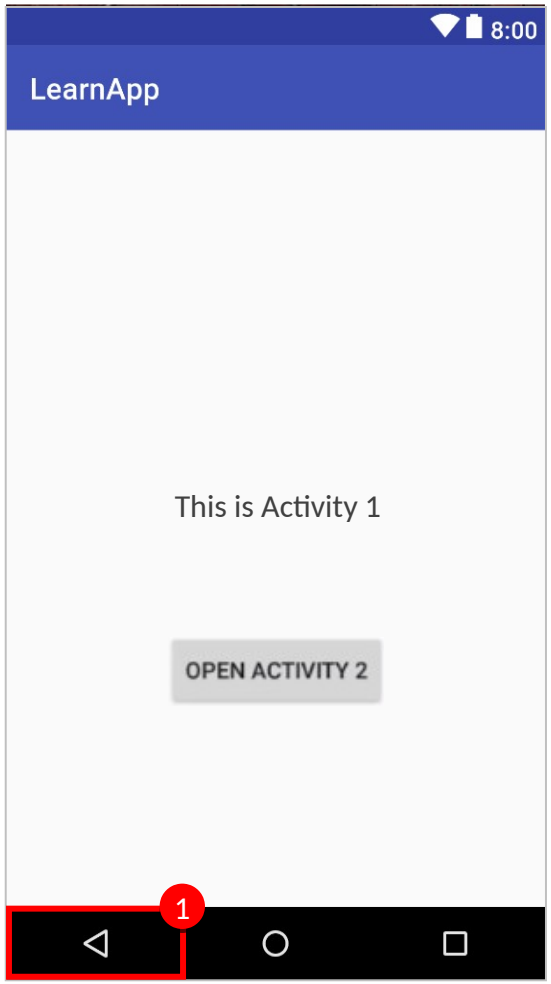


Cycle de vie de Activity 1

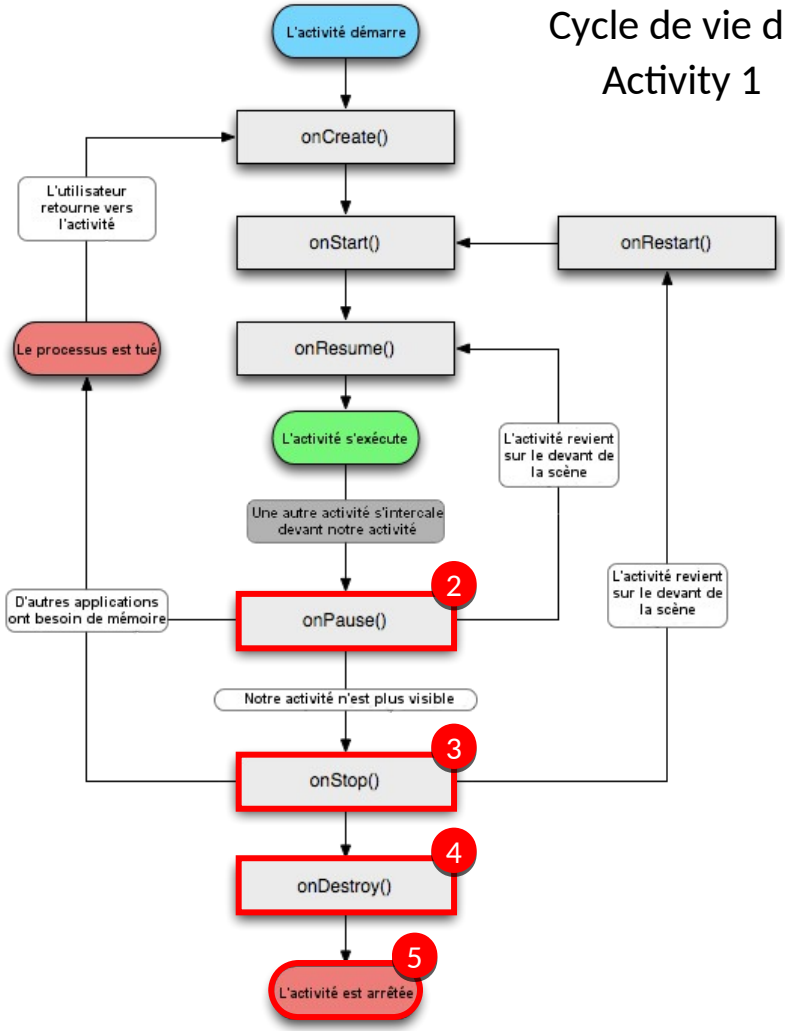


Activité

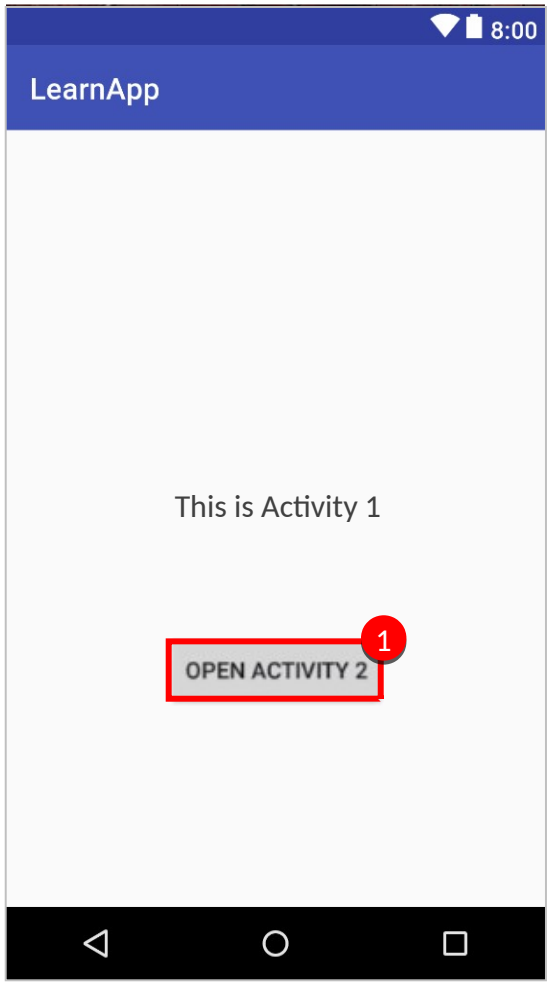
Cycle de vie (2/2) – Explication



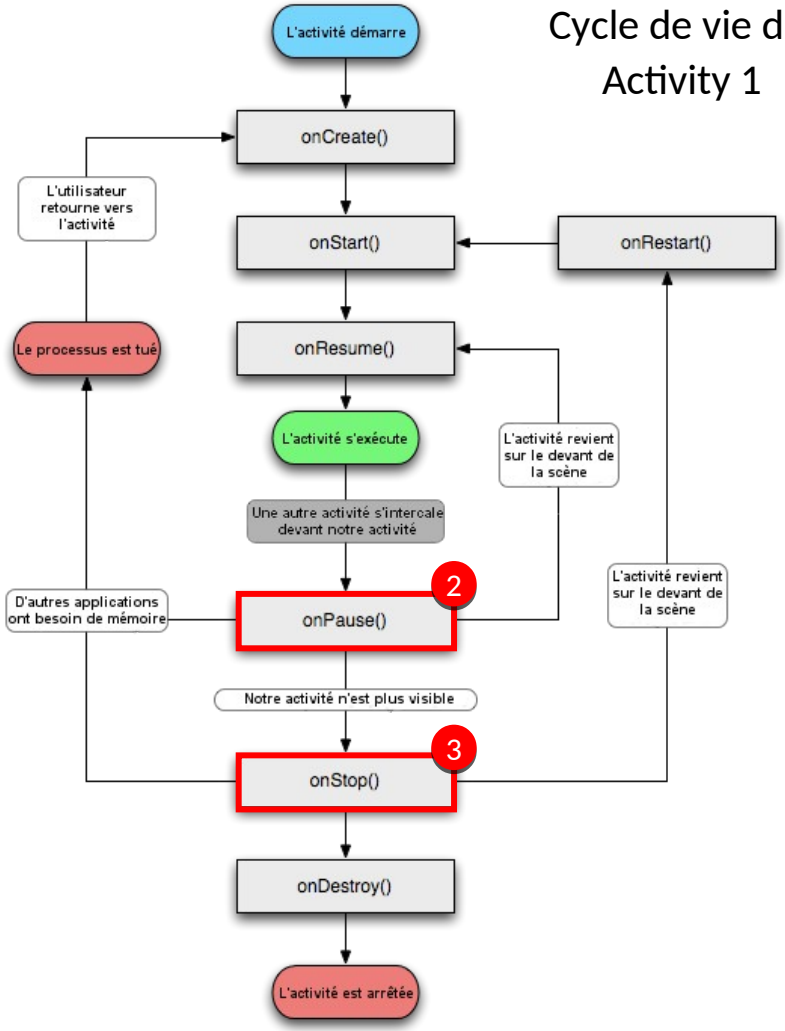
Cycle de vie de Activity 1



Cycle de vie (2/2) – Explication

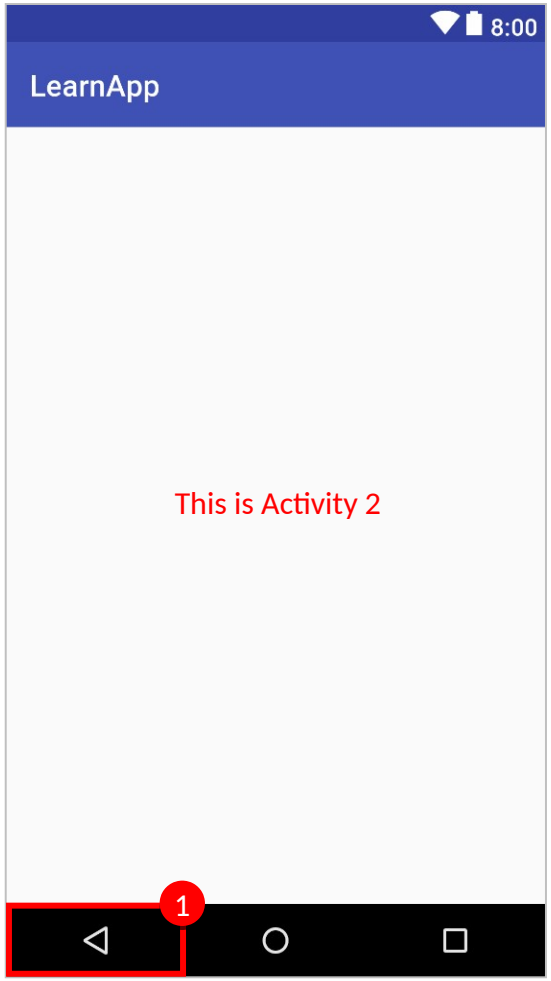


Cycle de vie de Activity 1

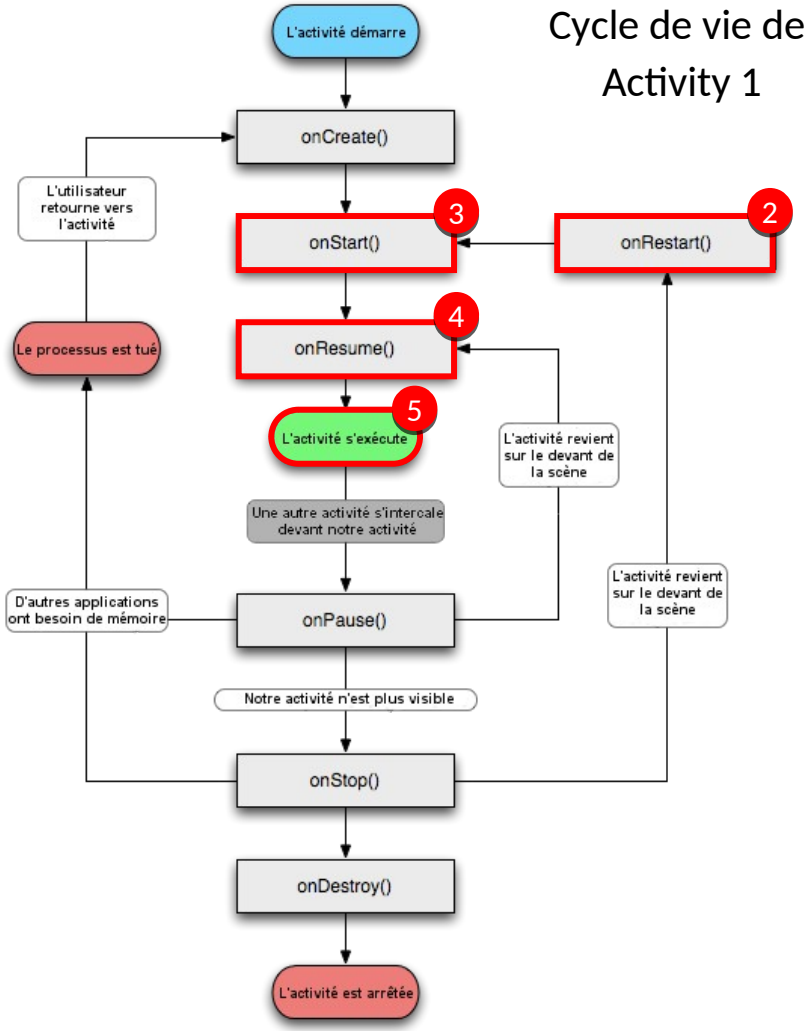


Activité

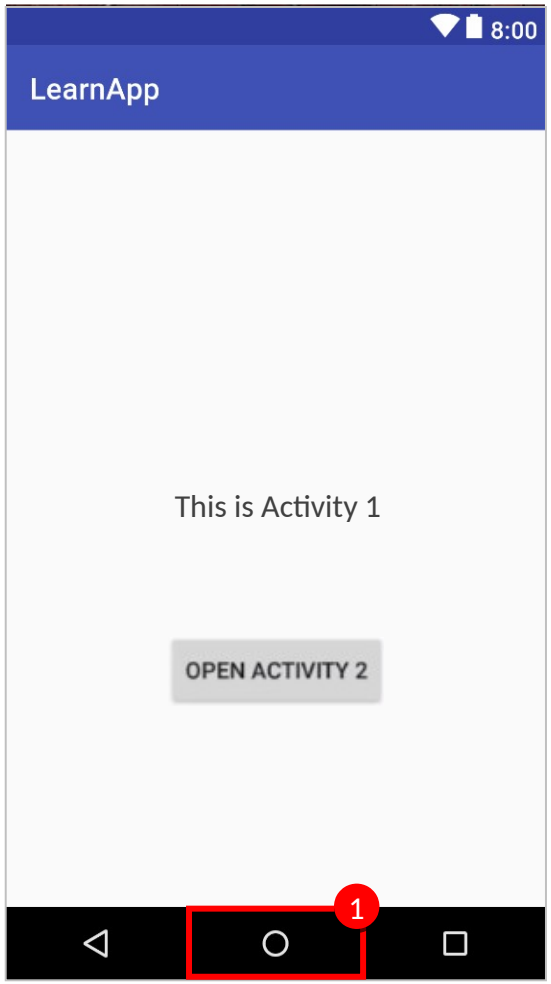
Cycle de vie (2/2) – Explication



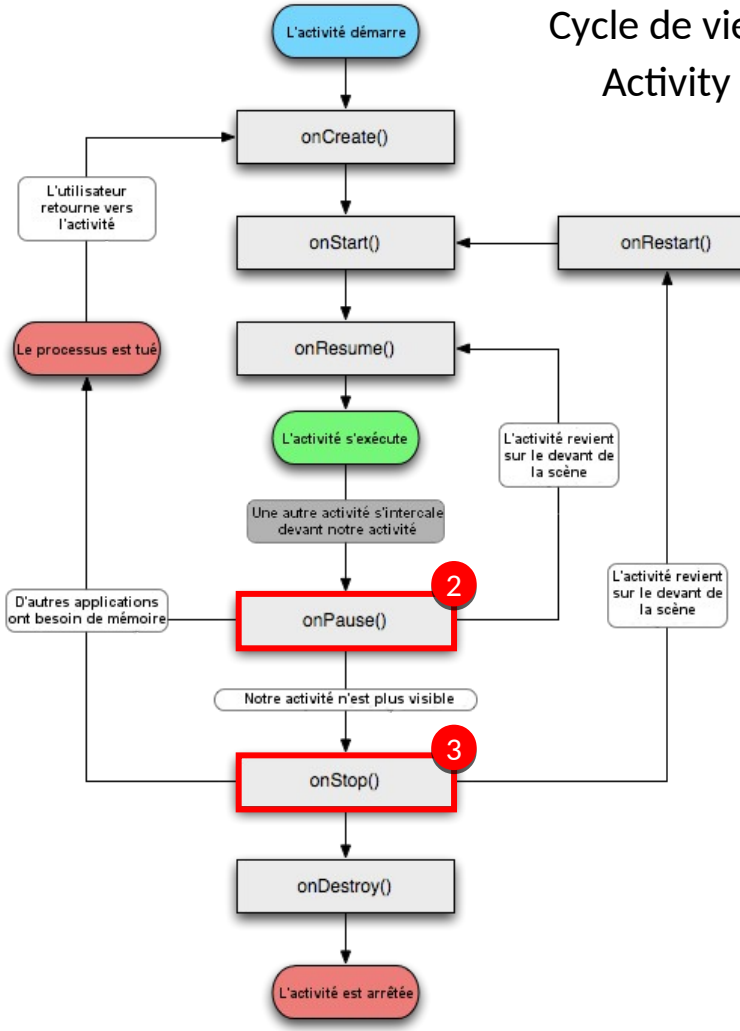
Cycle de vie de Activity 1



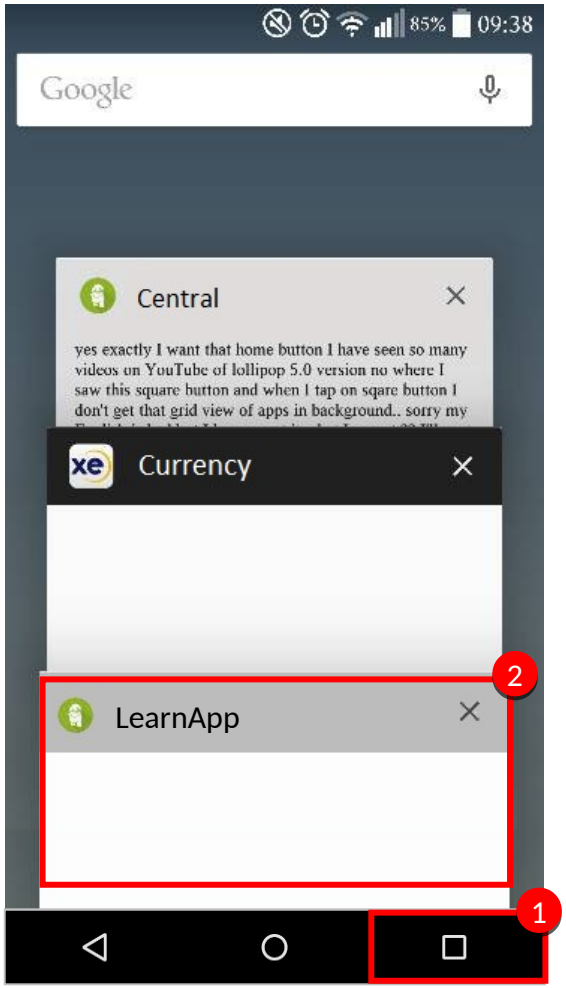
Cycle de vie (2/2) – Explication



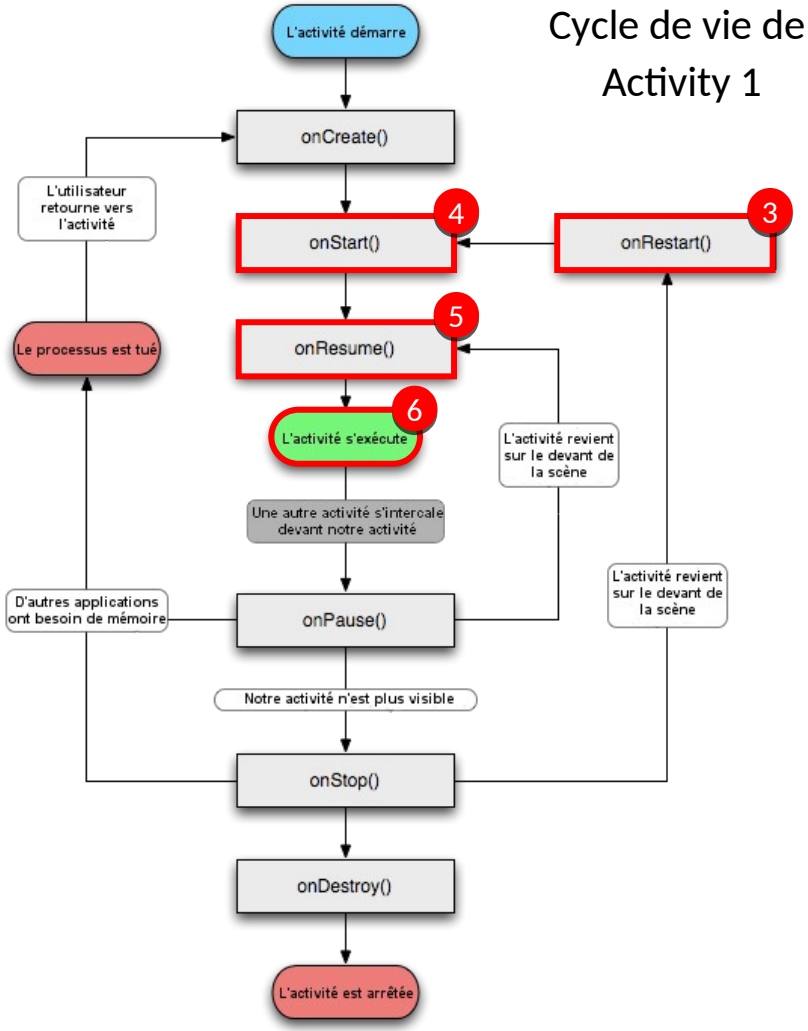
Cycle de vie de Activity 1



Cycle de vie (2/2) – Explication

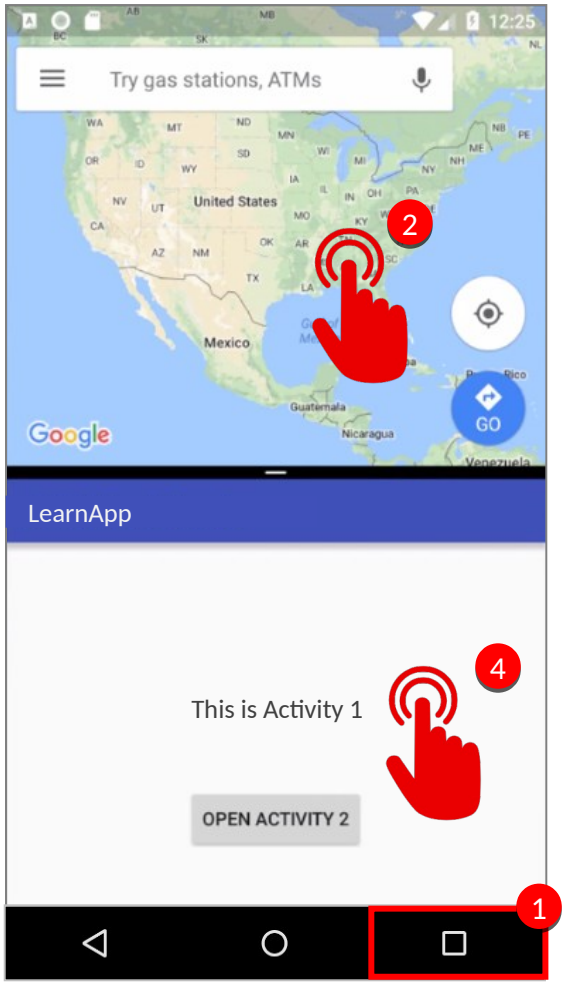


Cycle de vie de Activity 1



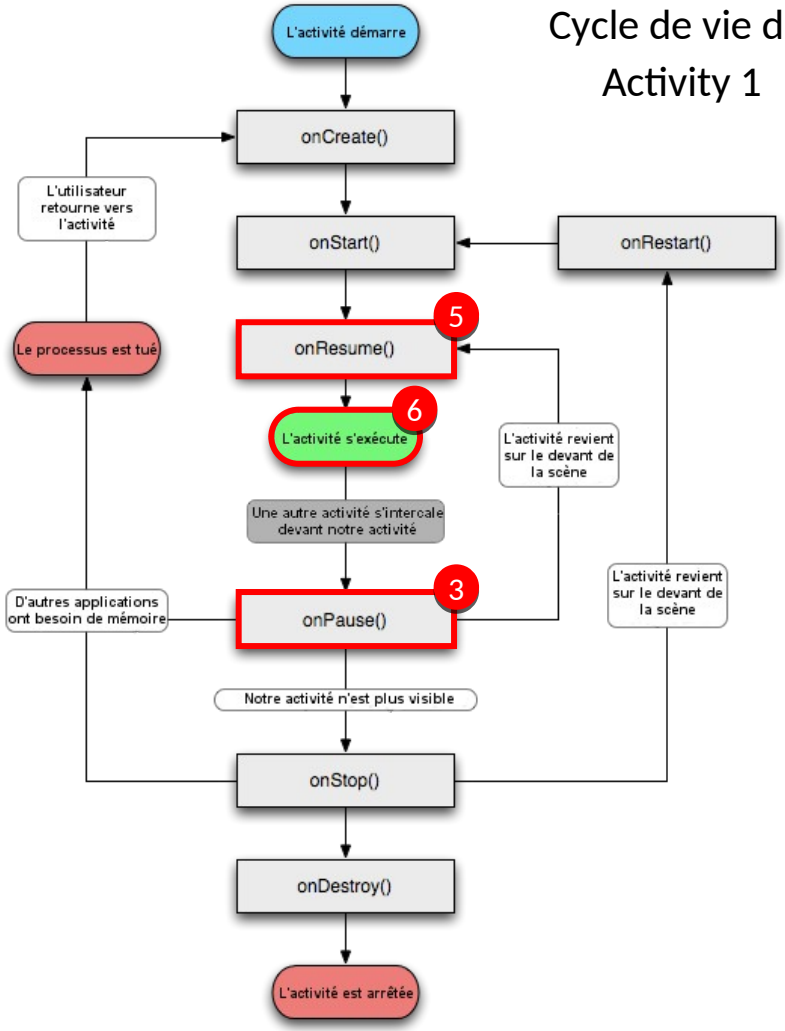
Activité

Cycle de vie (2/2) – Explication

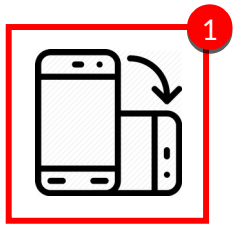


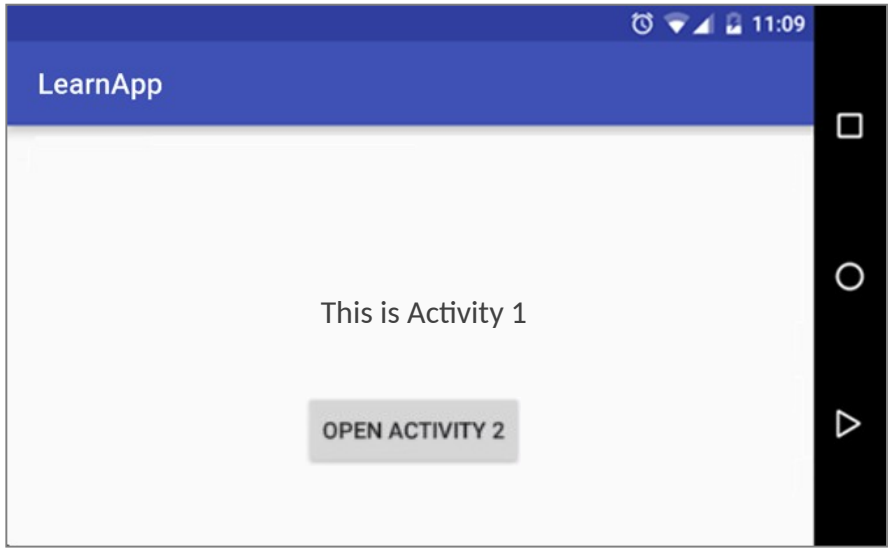
Partage d'écran (supporté depuis Android 7.0)

Cycle de vie de Activity 1

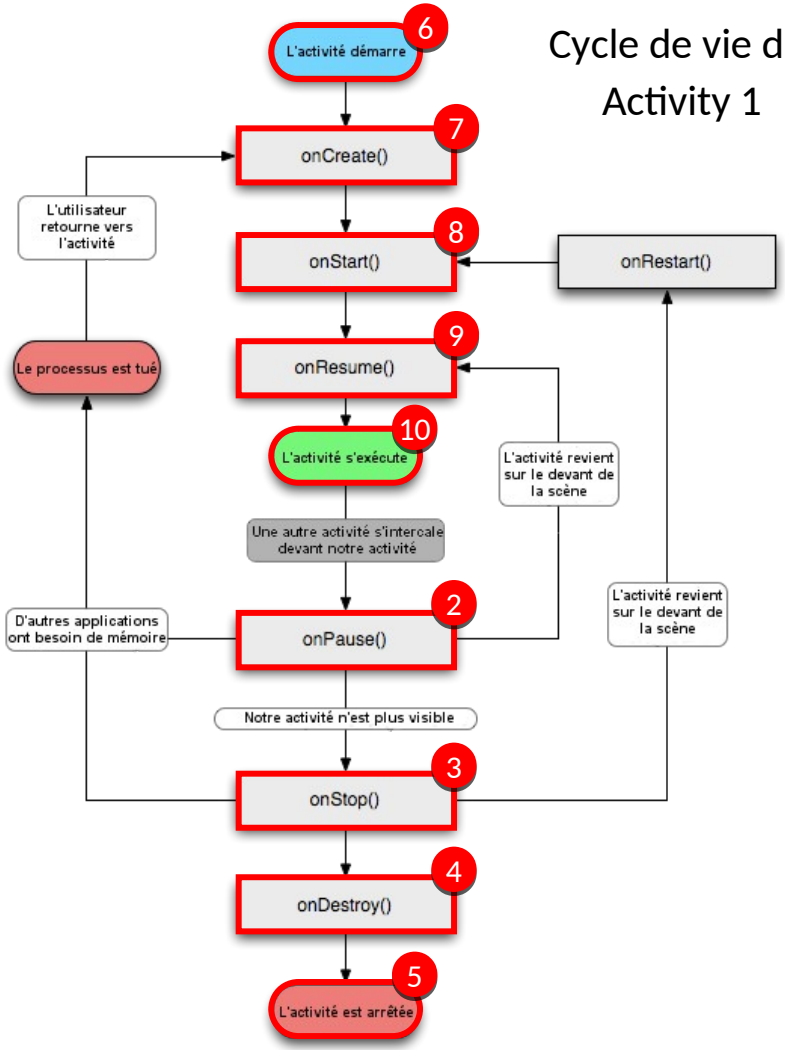


Cycle de vie (2/2) – Explication

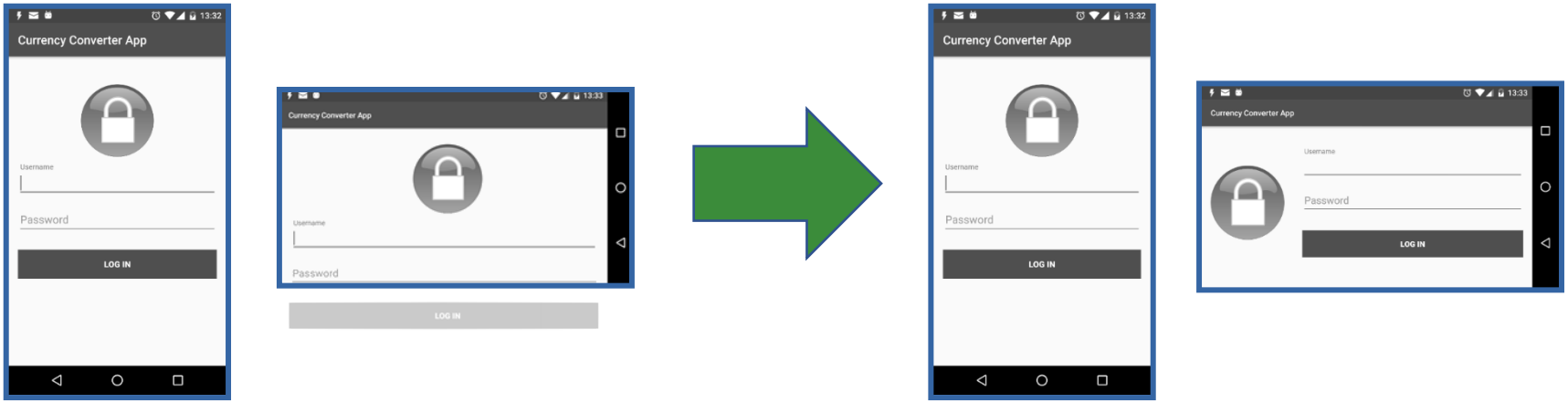




Cycle de vie de Activity 1

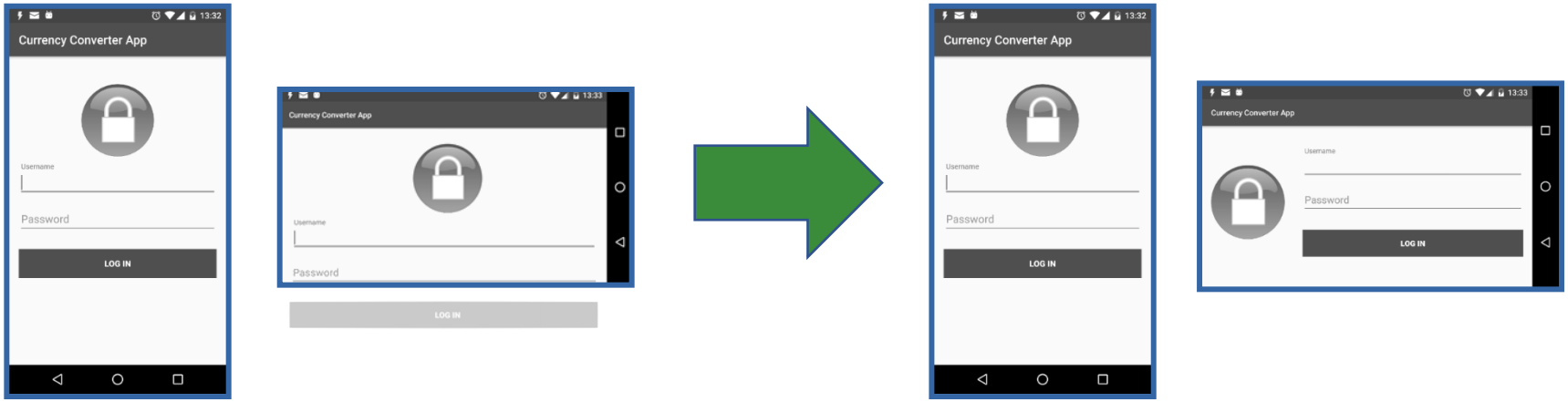


Vue portrait vs. vue paysage



Vue portrait vs. vue paysage

Utilisation des **`LinearLayout`** (LL)



`layout.xml`

```
<LL o="vertical">
   ...
</LL>
```

`portrait.xml`

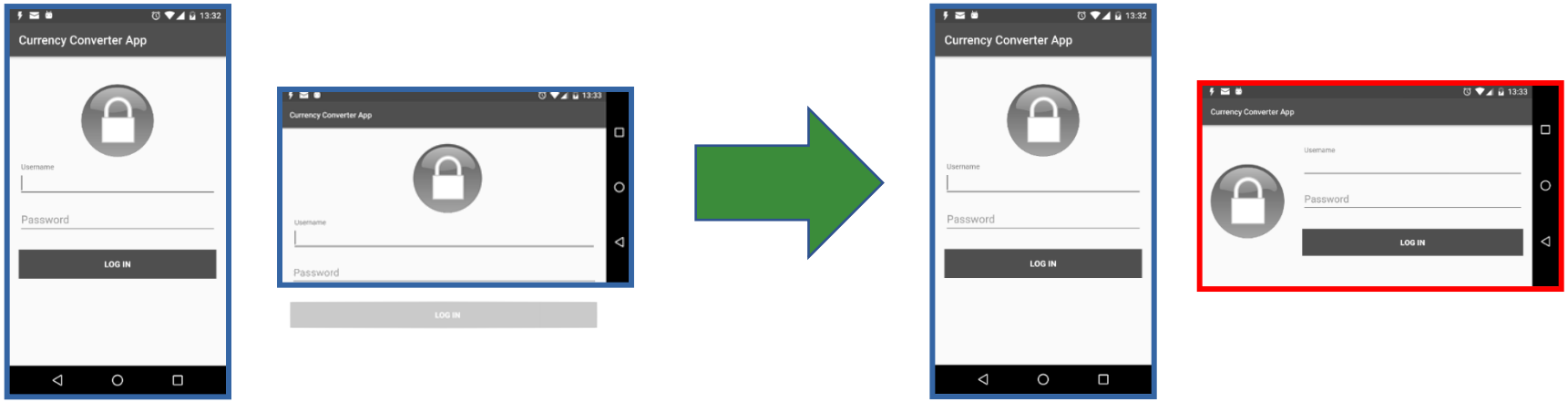
```
<LL o="vertical">
   ...
</LL>
```

`landscape.xml`

```
<LL o="horizontal">
   ...
   <LL o="vertical">
      ...
   </LL>
</LL>
```

Vue portrait vs. vue paysage

Utilisation des **RelativeLayout** (RL)



layout.xml

```
<LL o="vertical">
   ...
</LL>
```

portrait.xml

```
<LL o="vertical">
   ...
</LL>
```

landscape.xml

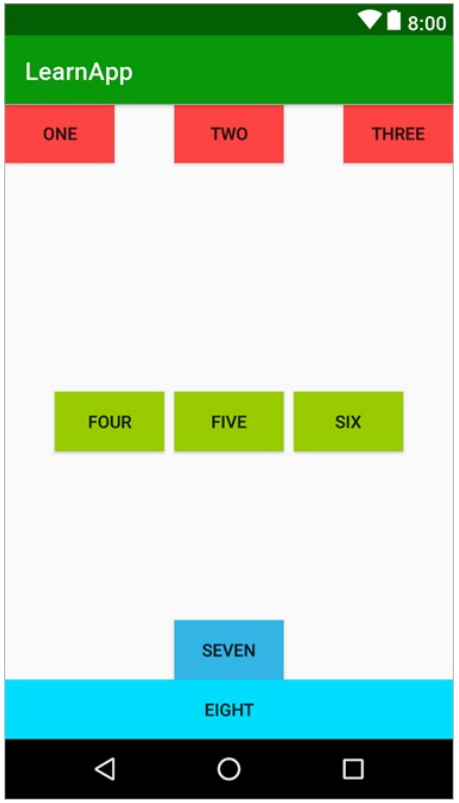
```
<RL>
   ...
</RL>
```

Layouts : RelativeLayout

Principe

Positionnement des vues :

- **Aligner par rapport une autre vue**
- **Aligner par rapport au parent**
- **Centrer par rapport au parent**
- **Positionner par rapport à une autre vue**



Layouts : RelativeLayout

Attributs de **RelativeLayout** (1/2)

Aligner par rapport une autre vue (**"@[+][package:]type:name"**)

- android:layout_above, android:layout_below
- android:layout_alignStart, android:layout_alignEnd
- android:layout_alignTop, android:layout_alignBottom
- android:layout_alignLeft, android:layout_alignRight

Aligner par rapport au parent (**"true"** ou **"false"**)

- android:layout_alignParentStart, android:layout_alignParentEnd
- android:layout_alignParentTop, android:layout_alignParentBottom
- android:layout_alignParentLeft, android:layout_alignParentRight
- android:layout_alignWithParentIfMissing

Layouts : RelativeLayout

Attributs de **RelativeLayout** (2/2)

Centrer par rapport au parent ("true" ou "false")

- android:layout_centerHorizontal
- android:layout_centerVertical
- android:layout_centerInParent

Positionner par rapport à une autre vue ("@[+][package:]type:name")

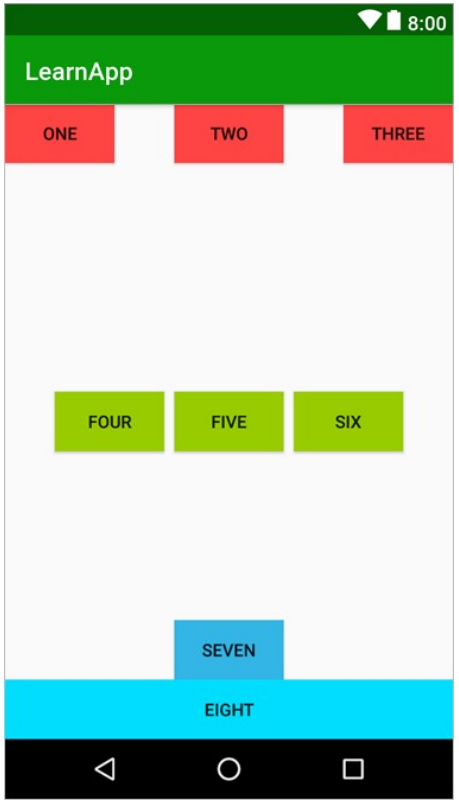
- android:layout_toStartOf, android:layout_toEndOf
- android:layout_toLeftOf, android:layout_toRightOf

Layouts : RelativeLayout

Exercice

Créer une activité `ComplexActivity`

- **Créer 8 Boutons**
- **Aligner les boutons dans un `RelativeLayout`**



Layouts : RelativeLayout

Solution

```
<RelativeLayout
android:layout_width="match_parent"
android:layout_height="match_parent">

<Button android:text="ONE"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/Button01" />

<Button android:text="TWO"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/Button02"
android:layout_centerHorizontal="true" />

<Button android:text="THREE"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/Button03"
android:layout_alignParentRight="true" />

<Button android:text="FOUR"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/Button04"
android:layout_toLeftOf="@+id/Button05"
android:layout_centerVertical="true" />

<Button android:text="FIVE"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/Button05"
android:layout_centerInParent="true" />

<Button android:text="SIX"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/Button06"
android:layout_toRightOf="@+id/Button05"
android:layout_centerVertical="true" />

<Button android:text="SEVEN"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:id="@+id/Button07"
android:layout_above="@+id/Button08"
android:layout_centerHorizontal="true" />

<Button android:text="EIGHT"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:id="@+id/Button08"
android:layout_alignParentBottom="true" />

</RelativeLayout>
```

Layouts : `RelativeLayout`

Evaluation

RelativeLayout : positionnement basé sur des règles relatives simples.

Avantages :

- Simple à comprendre
- Adapté aux interfaces peu complexes

Limites :

- Hiérarchies de vues souvent profondes
- Performances dégradées pour des interfaces complexes
- Difficulté de maintenance à grande échelle

`RelativeLayout` et **`ConstraintLayout`** permettent tous deux de positionner des vues les unes par rapport aux autres.

Layouts : ConstraintLayout

Principe

ConstraintLayout : positionnement basé sur un système de contraintes bidimensionnelles.

- Chaque vue est définie par au moins une **contrainte horizontale** et une **contrainte verticale**

Fonctionnalités avancées :

- Bias (positionnement proportionnel)
- Chains (alignement en série)
- Guidelines (repères invisibles)
- Barriers (contraintes dynamiques)

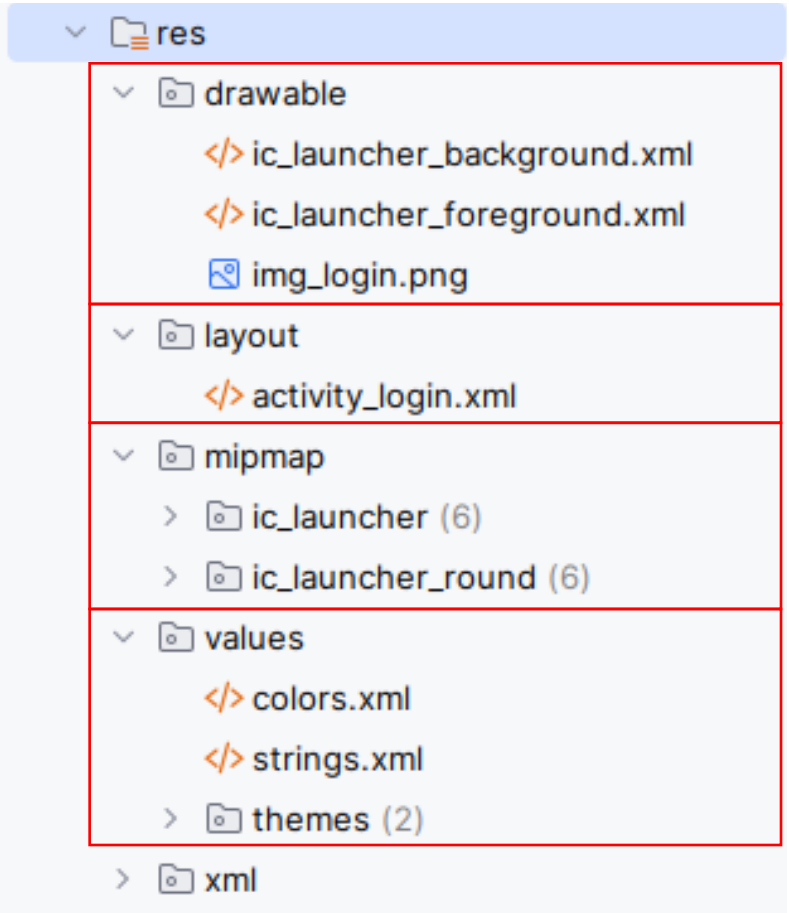
Avantages :

- Réduction de la profondeur de la hiérarchie
- Interfaces plus flexibles et adaptatives

Ressources "./res/"

Types de ressources :

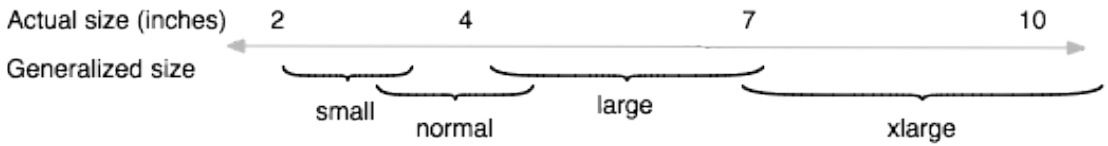
- **Layouts et menus**
- **Icones et images**
- **Valeurs (values) :**
  - strings, attrs,
  - styles, colors,
  - dimens,
  - …

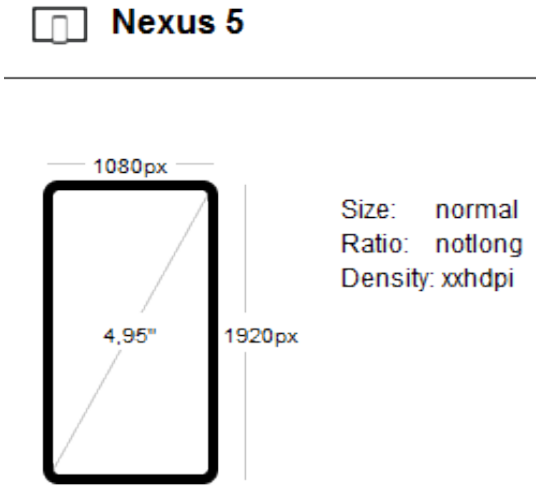


Taille et densité des écrans

Taille des écrans : 1 inch = 2,54 cm

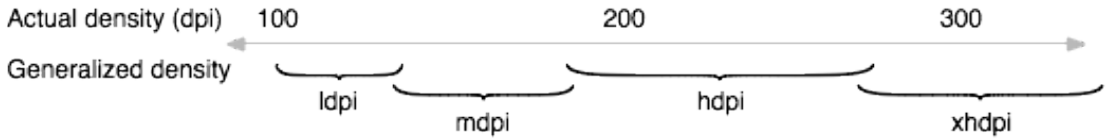
- **`small, normal, large, xlarge`**





Densité : DPI (Dot Per Inch)

- **`ldpi, mdpi, hdpi, xhdpi, xxhdpi, xxxhdpi`**



Taille et densité des écrans

Distribution

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small					0.5 %		0.5 %
Normal			0.2 %	2.8 %	45.2 %	23.7 %	**71.9 %**
Large		0.9 %	4.8 %	0.9 %	10.4 %	0.9 %	17.9 %
Xlarge		4.4 %	0.1 %	4.4 %	0.8 %		9.7 %
Total	0.0 %	5.3 %	5.1 %	8.1 %	**56.9 %**	**24.6 %**	

Source : Google Play Store, Novembre 2025

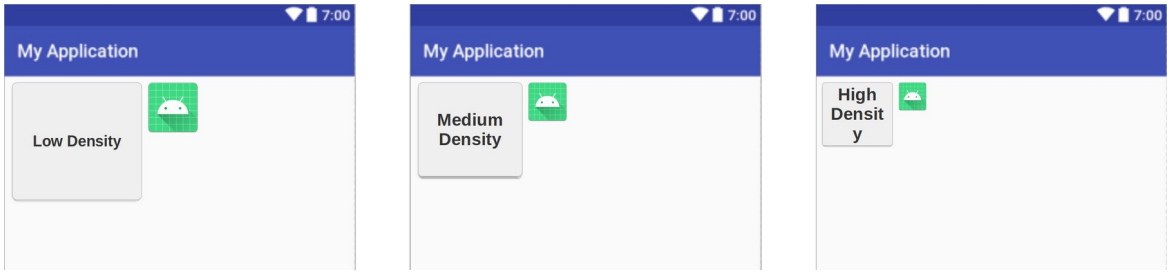
Taille et densité des écrans

Indépendance de la densité

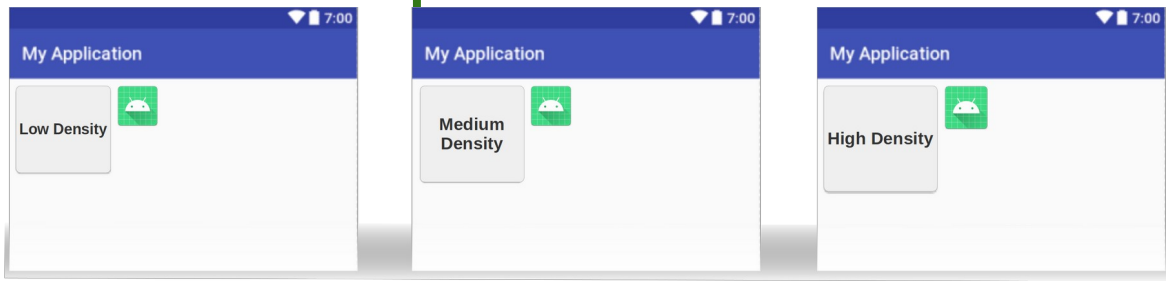
Density independant Pixel (dp)

- `dp = px / (dpi / 160)`

Dépendance de densité (px)

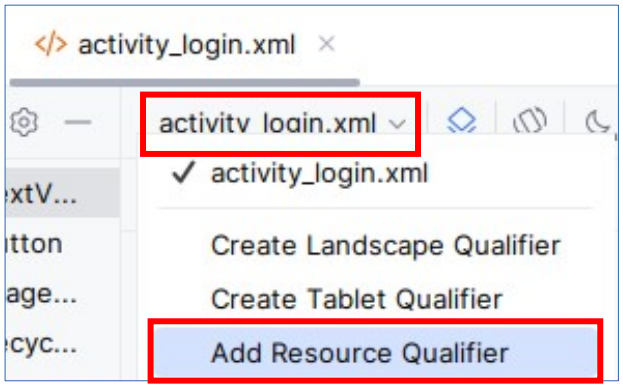


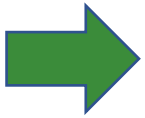
Indépendance de densité

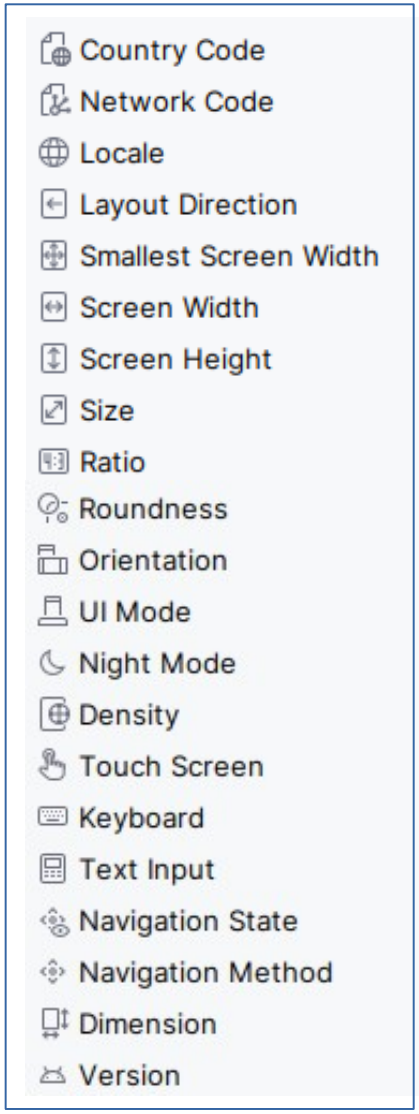


Prise en charge des différentes configurations

Il est possible de personnaliser les ressources, en fonction de :







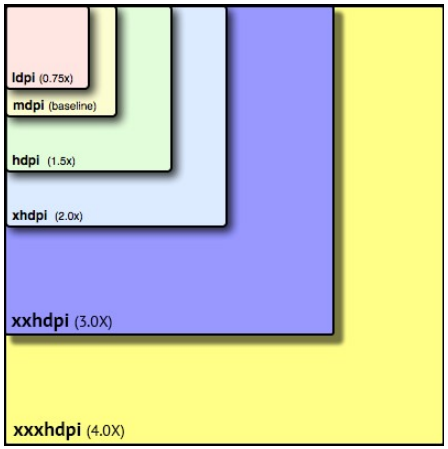
En utilisant des qualificateurs : par exemple :

- **Par défaut :** `layout` **:** layouts par défaut
- **Orientation :** `layout-land` **:** versions **paysage** des layouts
- **Densité :** `mipmap-xxhdpi` **:** icônes pour des écrans de densité **xxhdpi**
- **Taille :** `layout-large` **:** layouts pour des écrans de taille **large**
- **Langue :** `values-fr/strings.xml` **:** chaînes de caractères en **français**
- …

Drawables et densité de l'écran

Images matricielles :

~~drawable-ldpi/graphic.png~~	36 x 36 (0.75x)
drawable-mdpi/graphic.png	**48 x 48 (1.0x baseline)**
drawable-tvdpi/graphic.png	64 x 64 (1.33x)
drawable-hdpi/graphic.png	72 x 72 (1.5x)
drawable-xhdpi/graphic.png	96 x 96 (2.0x)
drawable-xxhdpi/graphic.png	144 x 144 (3.0x)
drawable-xxxhdpi/graphic.png	192 x 192 (4.0x)
drawable-nodpi/graphic.png	



→ **Mako Android Drawable Importer (Plugin) :**

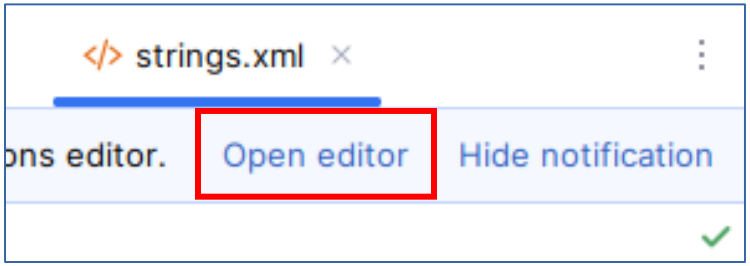
- Installation : **Settings... > Plugins > Browse repositories... >**
- Utilisation : **res/drawable > Mako Android Drawable Importer**

Icônes vectorielles : (Material ou SVG)

- **drawable-nodpi/graphic.xml**

→ **Utilisation : res/drawable > New > Vector Asset > Local file (SVG)**

Internationalisation (1/2)



/res/values/strings.xml

```
<resources>
     <string name="hello"> Hello </string>
     ...
</resources>
```

/res/values-fr/strings.xml

```
<resources>
     <string name="hello"> Salut </string>
     ...
</resources>
```

Internationalisation (2/2)

- Pour changer de langue **dynamiquement (par programmation)**

/java/LoginActivity.java

```
...
Locale myLocale = new Locale("fr");
DisplayMetrics dm = getResources().getDisplayMetrics();
Configuration conf = getResources().getConfiguration();
conf.locale = myLocale;
getResources().updateConfiguration(conf, dm);
...
Intent refresh = new Intent(this, LoginActivity.class);
startActivity(refresh);
finish();
...
```