

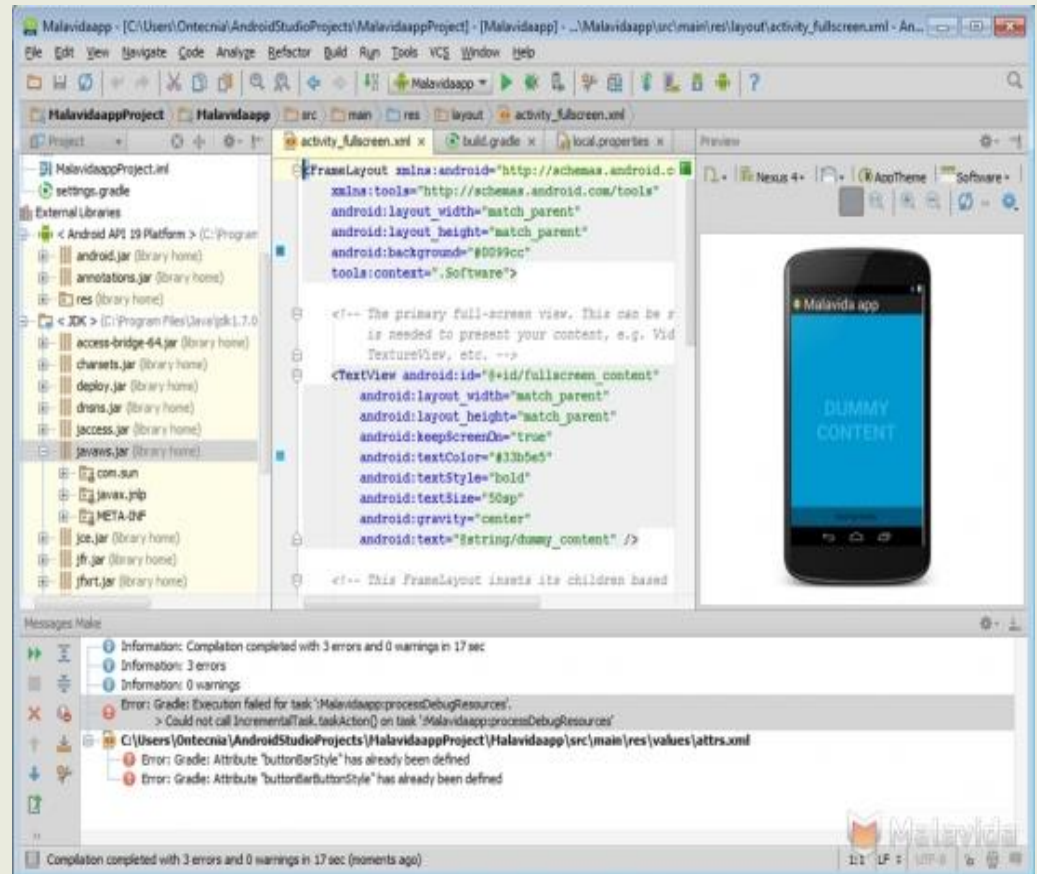
CREER UNE APPLICATION POUR ANDROID



ENVIRONNEMENT DE TRAVAIL

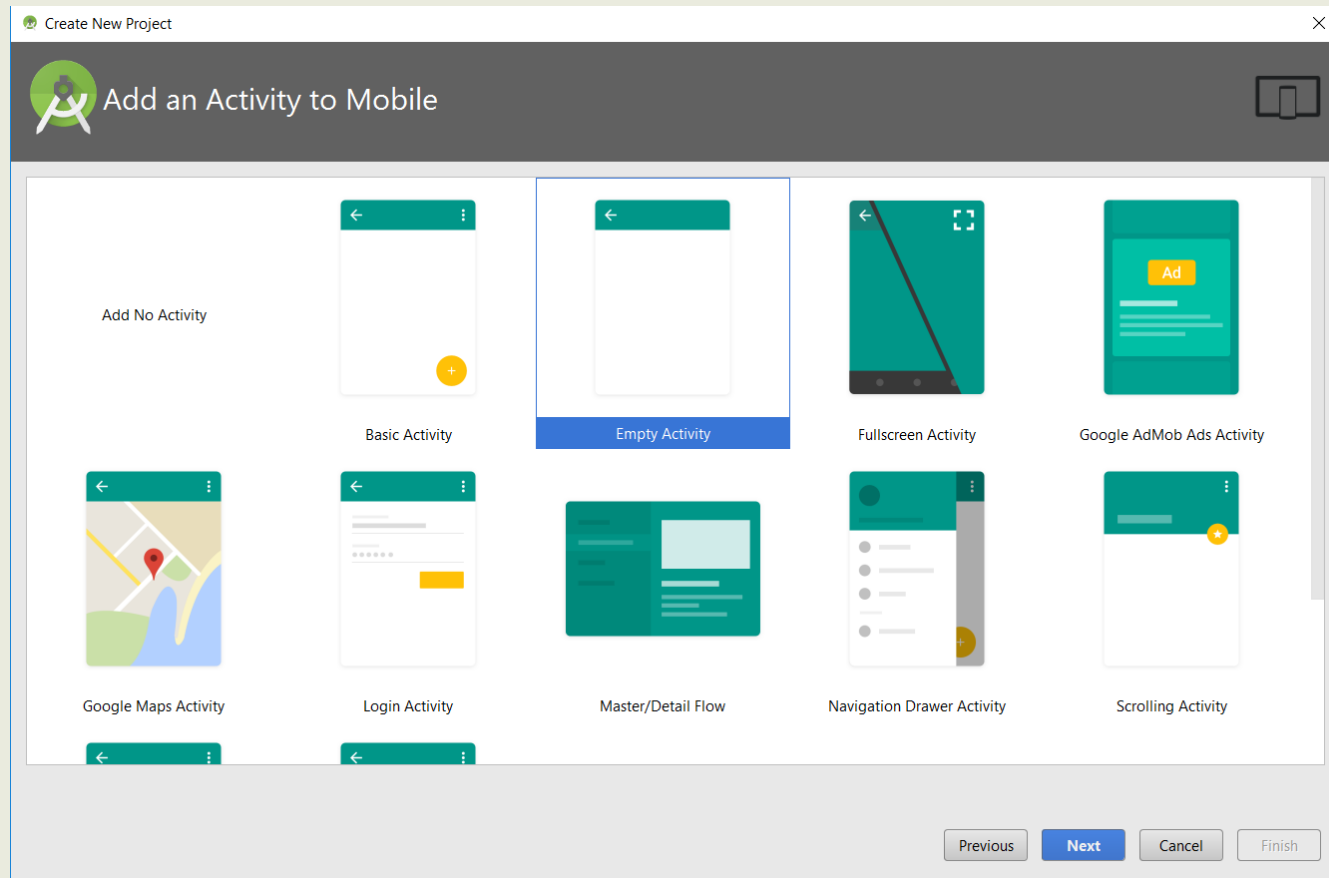
Android Studio

- ◆ Ensemble d'outils indispensables pour développer des applications Android.
- ◆ Un environnement de développement spécialisé dans les applications Android,
- ◆ un outil pour gérer l'installation du SDK Android.

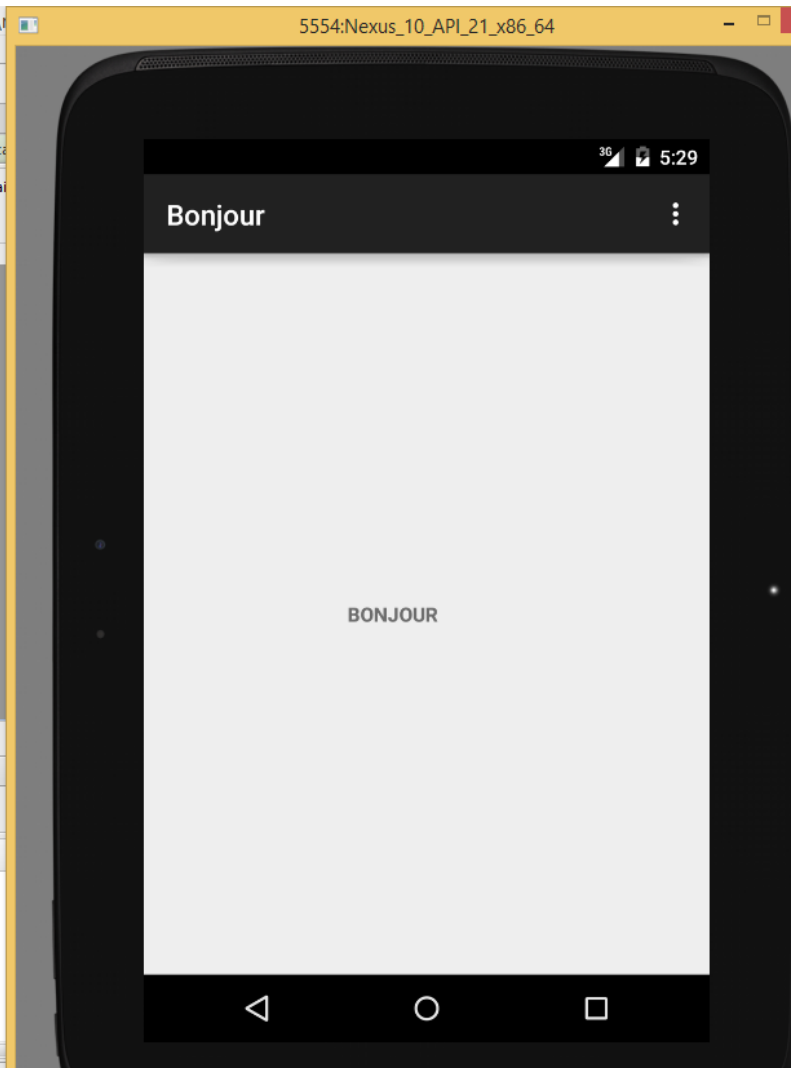
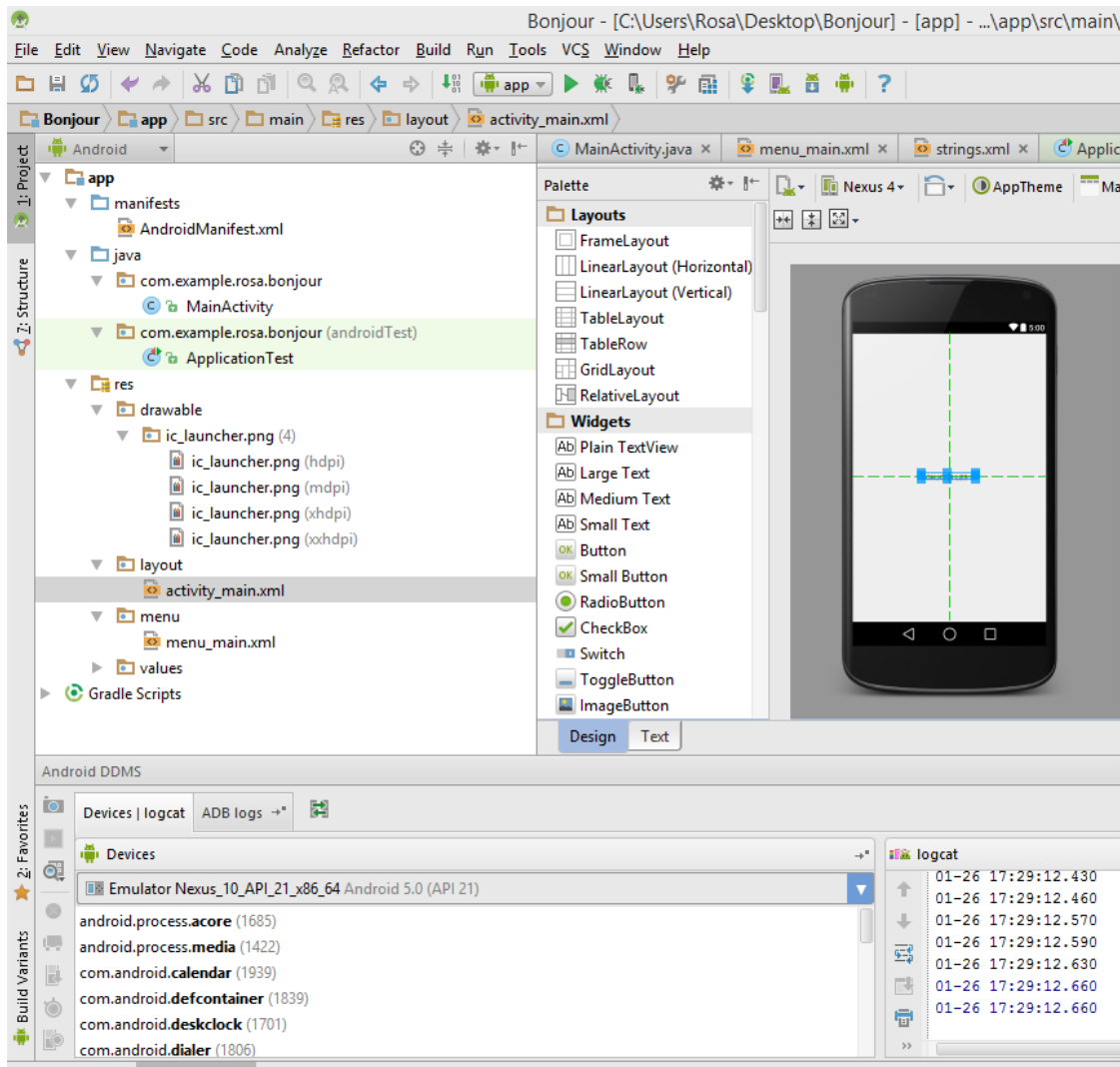


ANDROID STUDIO

Création d'un projet

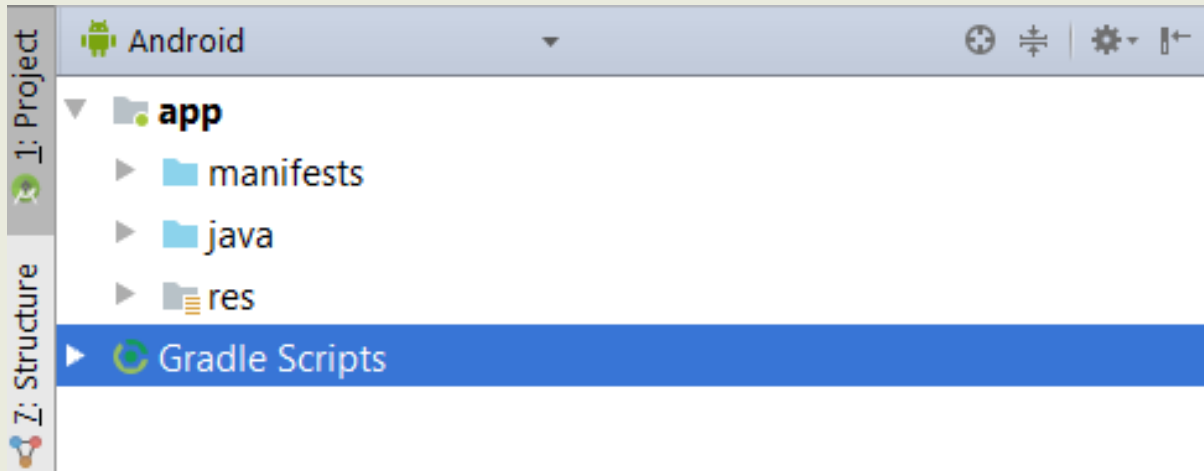


PREMIÈRE APPLICATION

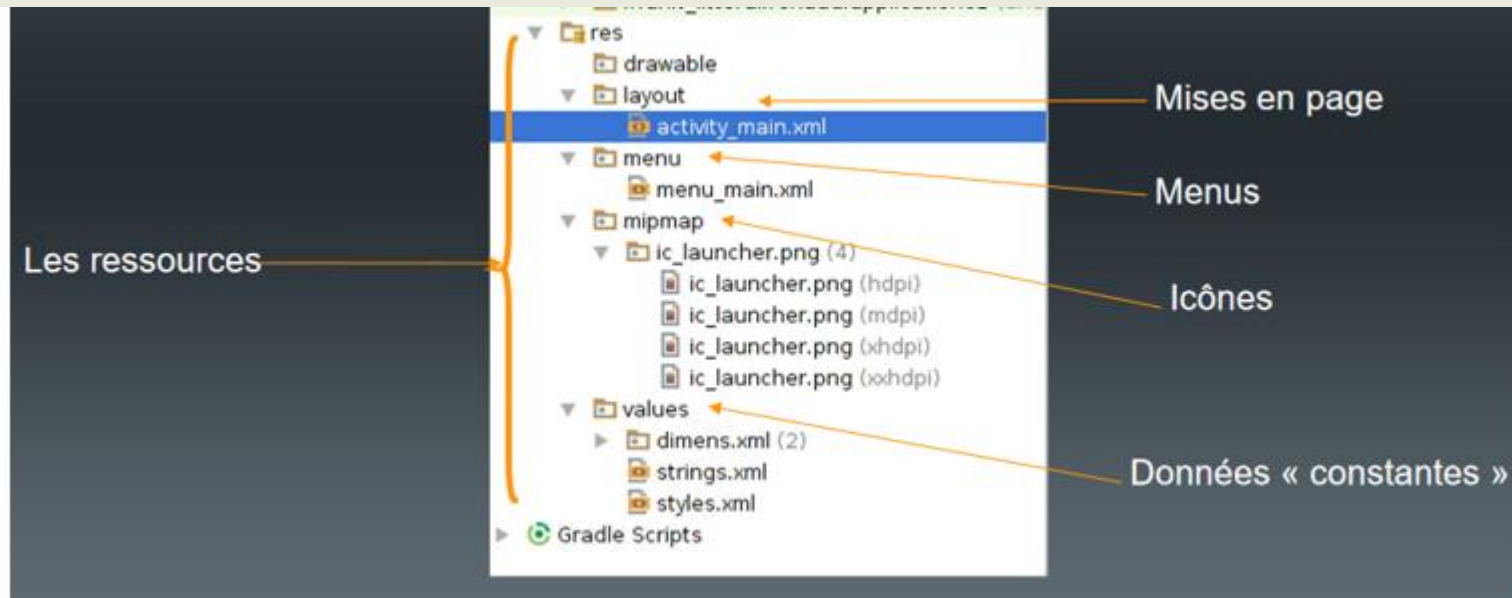


ORGANISATION D'UN PROJET

Un projet type contient les dossiers suivants



LES RESSOURCES



LES RESSOURCES

❑ **res/layout**

- ◆ contient des fichiers de description des interfaces en XML .

❑ **res/values**

- ◆ contient des fichiers de définitions de valeurs en XML décrivant des chaînes, tableaux, valeurs numériques.
- ◆ les layouts en font référence.
- ◆ Cette technique est très utile pour prévoir des versions multilingue.

❑ **res/anim**

- ◆ contient des fichiers de description d'animations en XML.

❑ **.....res/xxxx**

CRÉATION DES VUES

❑ Création de Layout

- ◆ Peut contenir des widgets
- ◆ Hérite de View
- ◆ Peut contenir un autre Layout



Koding Indonesia

❑ Unités

- ◆ dp
- ◆ sp

❑ Mots clés

- ◆ context
- ◆ wrap-content
- ◆ match-parent

Les Layout sont sous la forme d'un fichier XML

ATTRIBUTS DES LAYOUTS

Les attributs des layouts (gabarits) permettent de spécifier des attributs supplémentaires. Les plus importants sont :

- ◆ `android:layout_width` et `android:layout_height`:
= "**fill_parent**" ou "**match_parent**": l'élément remplit tout l'élément parent
= "**wrap_content**": prend la place minimum nécessaire à l'affichage
- ◆ `android:orientation`: définit l'orientation d'empilement
- ◆ `android:gravity`: définit l'alignement des éléments

LINEAR LAYOUT

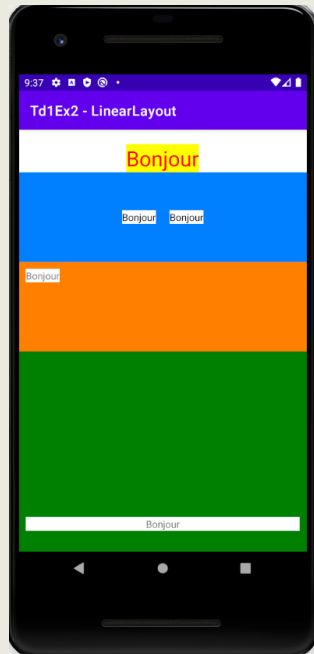
les vues enfants sont organisées les unes à la suite des autres

❑ Orientation

- ◆ Vertical
- ◆ Horizontal

❑ Alignement

- ◆ Gauche
- ◆ Droite
- ◆ Centre
- ◆ Haut
- ◆ Bas



❑ Gravity (alignement du contenu de la vue enfant)

- ◆ Centrer un texte dans un TextView

❑ Layout-Gravity (alignement de la vue enfant dans le parent)

- ◆ Centrer un bouton
- ◆ Centrer un TextView

ATTRIBUTS DES LAYOUTS

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:orientation="vertical"

android:layout_width="match_parent"

android:layout_height="match_parent"

android:gravity="center"

android:id="@+id/accueil"

>

.... Ici on place des widgets

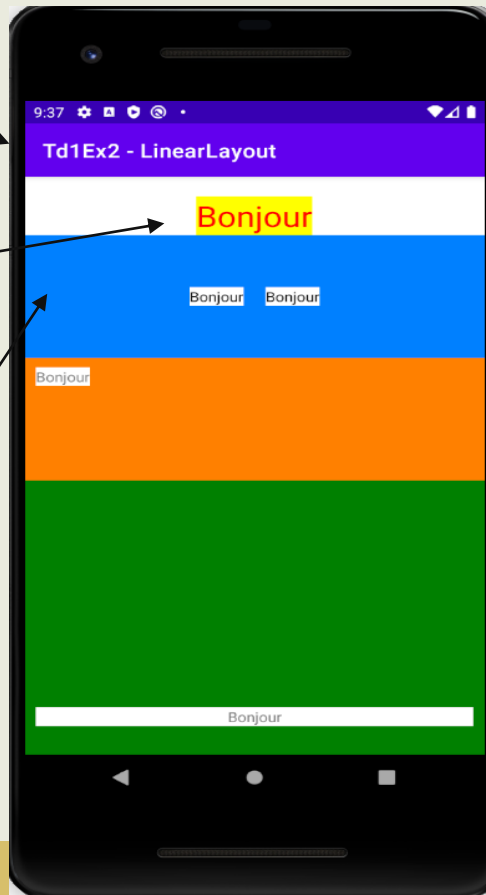
</LinearLayout>

LINEAR LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:gravity="top"
  android:orientation="vertical">
```

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_gravity="center"
  android:layout_marginTop="20dp"
  android:text="@string/contentText"
  android:background="@color/yellow"
  android:textSize="30sp"
  android:textColor="#FF0000" />
```

```
<LinearLayout
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:layout_weight="3"
  android:gravity="center"
  android:background="@color/blue"
  android:orientation="horizontal">
```



Extrait du code XML

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:background="@color/white"
  android:layout_margin="10dp"
  android:text="@string/contentText"
  android:textColor="#000000" />
```

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:background="@color/white"
  android:layout_margin="10dp"
  android:text="@string/contentText"
  android:textColor="#000000" />
```

```
</LinearLayout>
```

CONSTRAINT LAYOUT

les vues enfants sont tous contraint horizontalement ET verticalement

❑ Contrainte (au moins une de chaque)

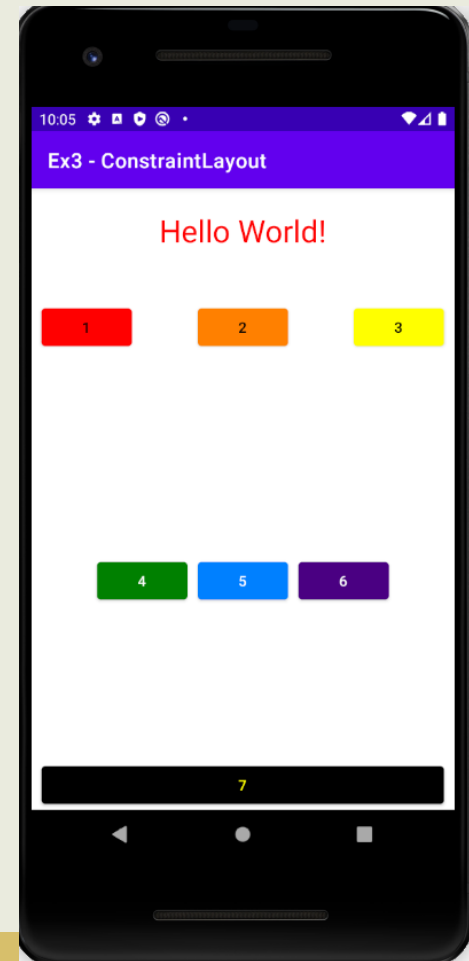
- ◆ Verticale
- ◆ Horizontale

❑ Alignement

- ◆ Par rapport au parent
- ◆ Par rapport à un autre widget

❑ Taille 0dp

- ◆ Détermine dynamiquement en fonction des contraintes du parent
- ◆ On positionne le widget dessus et dessous et on souhaite prendre la hauteur disponible restant entre les 2

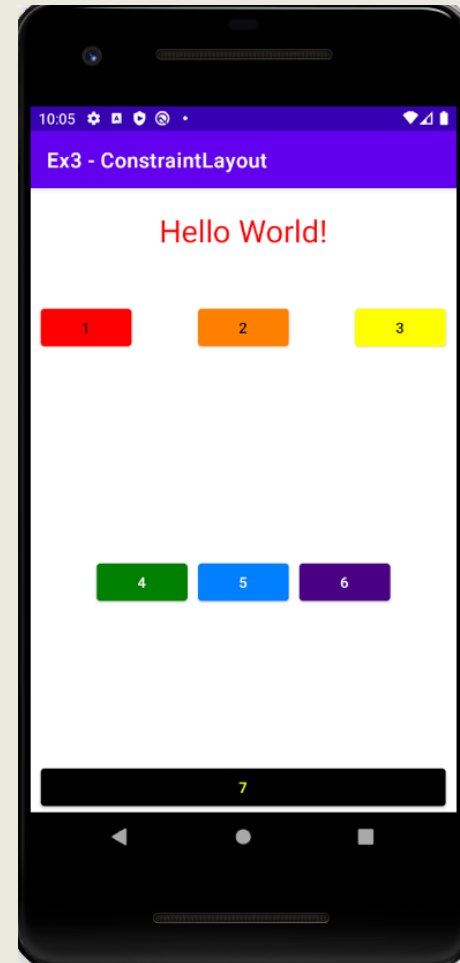


CONSTRAINT LAYOUT

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
```

```
<TextView
    android:id="@+id/label"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:textSize="30sp"
    android:textColor="#FF0000"
    android:layout_marginTop="20dp"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

```
<Button
    android:id="@+id/button1"
    android:text="1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:backgroundTint="@color/red"
    android:textColor="@color/black"
    android:layout_marginTop="50dp"
    android:layout_marginLeft="10dp"
    app:layout_constraintTop_toBottomOf="@id/label"
    app:layout_constraintLeft_toLeftOf="parent"/>
```



LIEN XML / JAVA

❑ La classe abstraite R

- ◆ Créée automatiquement à chaque build
- ◆ Une erreur dans R == une erreur en XML



LA CLASSE STATIQUE R

❑ Classe générée automatiquement par l'IDE

- ◆ Créée à partir de l'arborescence présente dans le dossier res
- ◆ Un identifiant est créé pour chacun des éléments trouvés
- ◆ Permet l'accès aux ressources
 - ✦ dans le code en java, pour désigner une ressource sous la forme : **R.type.nom.**

LA CLASSE STATIQUE R

En résumé

- ❑ Les ressources de l'applications sont utilisées dans le code au travers de la classe statique **R**.
- ❑ La classe statique **R** est régénérée automatiquement à chaque changement dans le projet.
- ❑ Toutes les ressources sont accessibles au travers de **R**, dès qu'elles sont déclarées dans le fichier XML
`android.R.type_ressource.nom_ressource`

LES INTERFACES GRAPHIQUES

- ❑ Ils sont référencés par `R.layout.nom_du_fichierXML`.
- ❑ Les activités peuvent utiliser la méthode **`setContentView(R.layout.nom_du_fichierXML)`** pour mettre en place l'interface décrite par un tel fichier.

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```

QU'EST CE QU'UNE ACTIVITÉ ?



LES ACTIVITÉS

❑ Une classe java par activité ;

- des ressources associées (layout, menu, etc.)
- hérite de la classe Activity (AppCompatActivity avec une ToolBar);



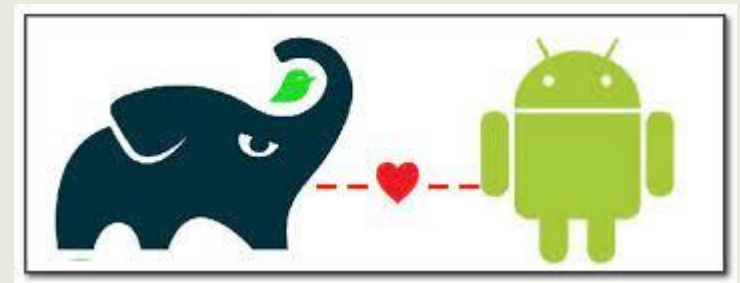
❑ Elle contient généralement les éléments suivants :

- gestion de l'interface utilisateur, telle que l'inflation (inflate) des layout, l
- gestion des événements de clic, etc.
- communique avec les autres composants de l'application, tels que les fragments, les services, les fournisseurs de contenu,
- Le code qui gère le cycle de vie de l'activité, c'est-à-dire les étapes de création, de démarrage, de reprise, de pause, de redémarrage et de destruction de l'activité.

LES ACTIVITÉS

❑ En terme IHM :

- Considérée comme le « contrôleur » dans MVC
- Interagit avec la Vue (Layout)
- Interagit avec le Modèle
-



❑ Design Pattern :

- Difficile de mettre en œuvre le pattern MVC mais plus proche de MVVC
- Utilise régulièrement le pattern Adapter

❑ Événementiel

- Mise en place d'écouteurs (listeners)

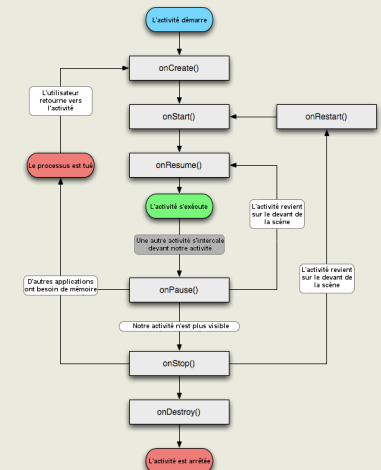
LES ACTIVITÉS

- ◆ Lorsqu'on crée une « empty activity » dans Android Studio, ce dernier crée le code suivant :

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
    }  
}
```



Tiens !... Pourquoi ce schéma ?



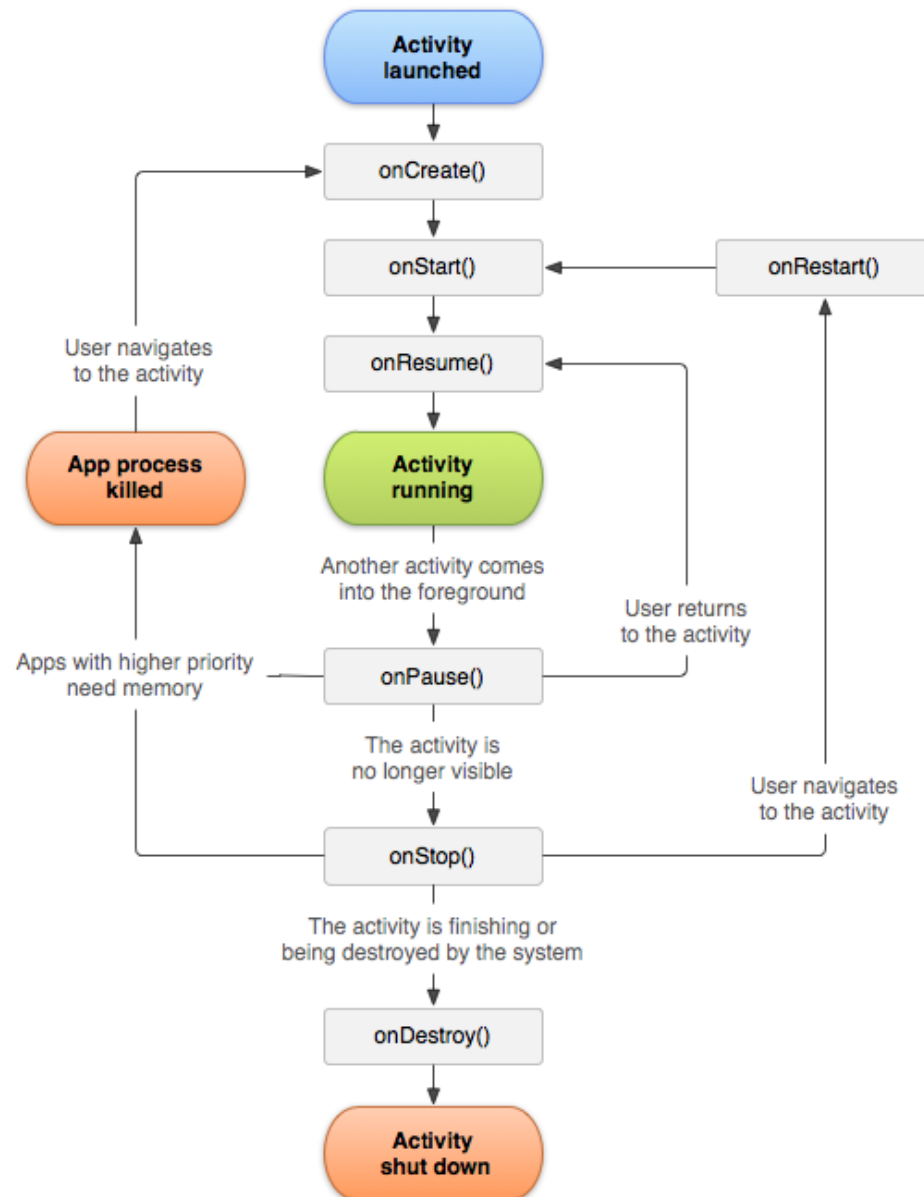
CYCLE DE VIE DES ACTIVITÉS



Android

@.fR Frederic.rallo@unice.fr

UTILISATION DE VOTRE APPLICATION



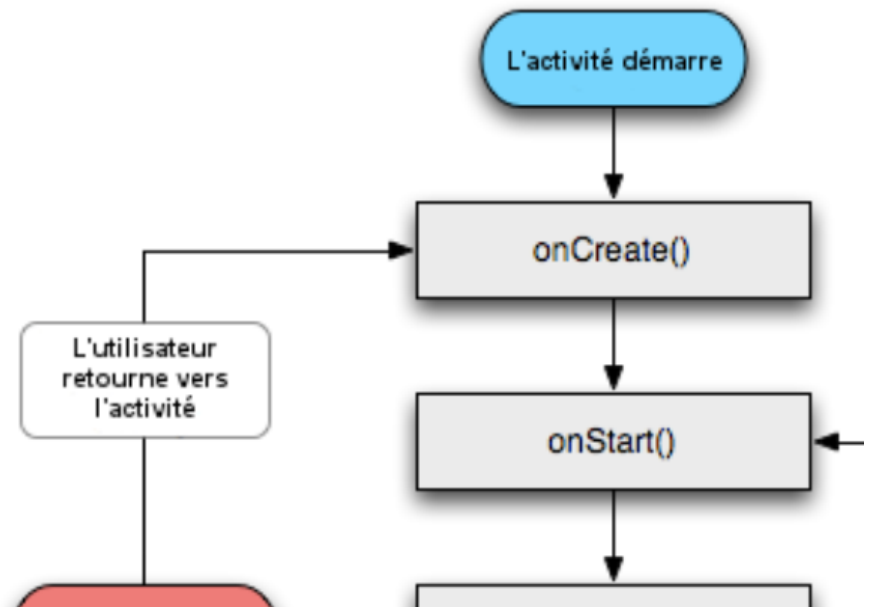
Quand un utilisateur manipule votre application, la quitte pour une autre ou y revient, elle traverse plusieurs états symbolisés par le cycle de vie des activités, schématisé à la figure suivante.

PRINCIPE

Entre chaque étape votre application appelle une méthode.

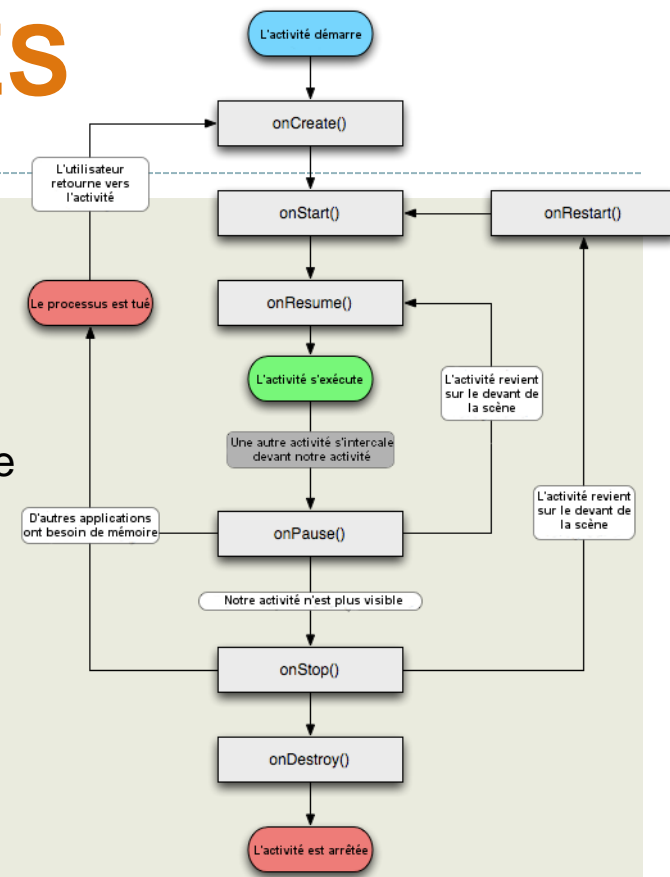
Cette méthode porte le même nom que l'état traversé.

Par exemple à la création est appelée la méthode `onCreate`.

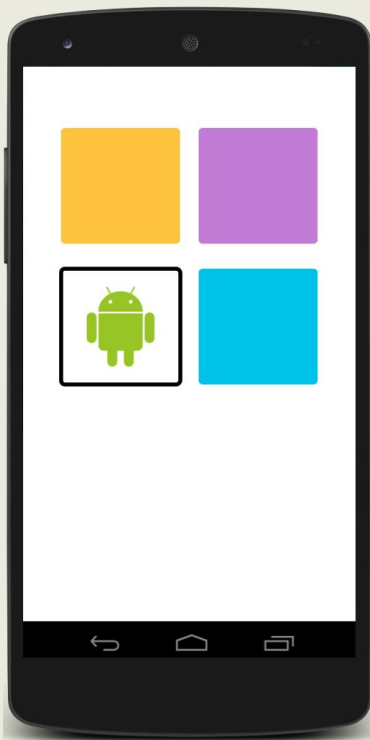


LES ACTIVITÉS

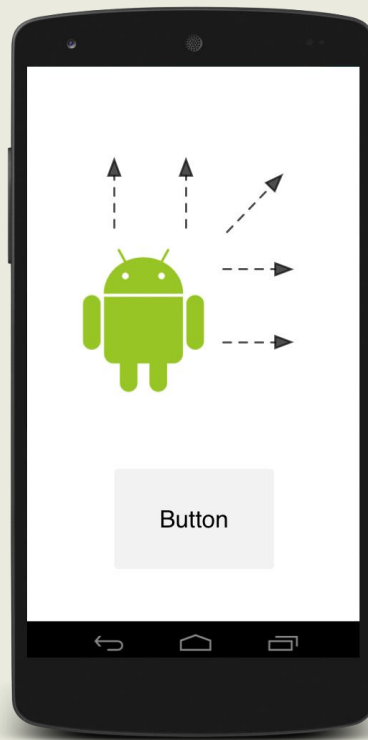
- ❑ **onCreate(Bundle)** → l'activité est prête à être initialisée.
 - ✦ Le paramètre permet de récupérer un état sauvegardé lors de l'arrêt de l'activité (si on a fait une sauvegarde)
- ❑ **onStart()** → la vue associée à l'activité devient visible aux yeux de l'utilisateur
- ❑ **onResume()** → l'utilisateur peut interagir avec la vue
- ❑ **En fonctionnement (run)**
- ❑ **onPause()** → l'utilisateur ne peut plus interagir avec la vue
- ❑ **onStop()** → la vue associée à l'activité devient invisible aux yeux de l'utilisateur
- ❑ **onDestroy()** → l'activité est prête à être perdue



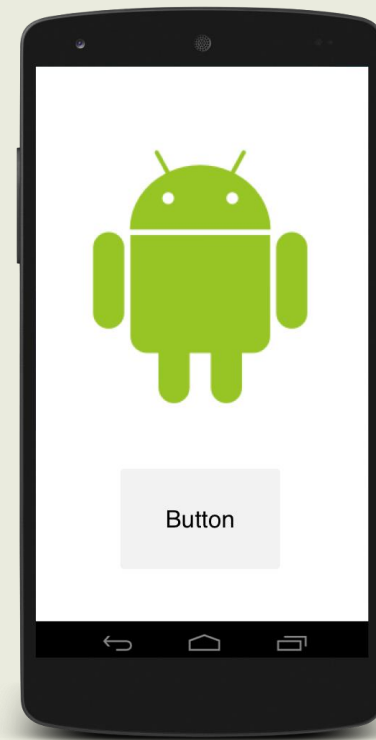
PLUSIEURS ACTIVITÉS



Activity 1



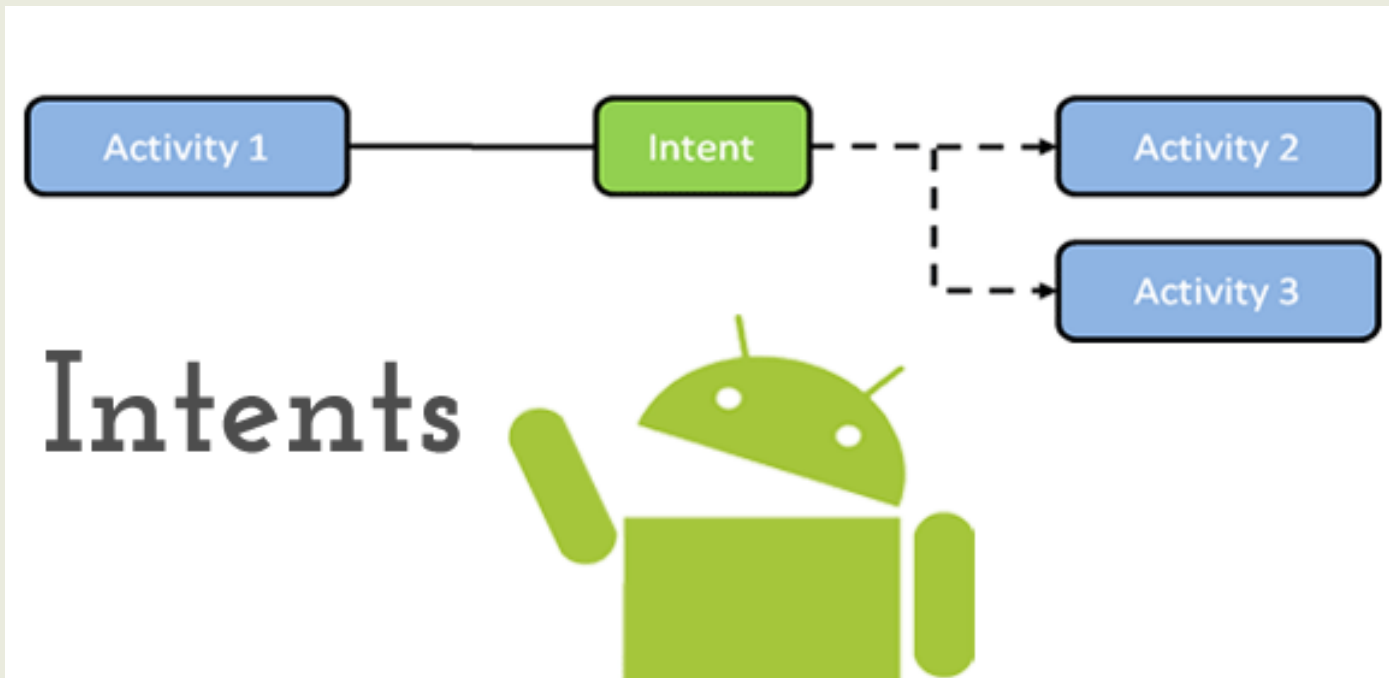
Scene Transition Animation
(common element)



Activity 2

GÉRER PLUSIEURS ACTIVITÉS

- ❑ Les Intents
- ❑ Déclarer et lancer une nouvelle activité
- ❑ Echange de données entre activités
- ❑ Echange de données parcelables



LES INTENTS

- ❑ Les *Intents* permettent de gérer l'envoi et la réception de **messages** afin de faire coopérer les applications.
- ❑ Le but des *Intents* est de déléguer une action à un autre **composant**, une autre **application** ou **une autre activité** de l'application courante.



LES INTENTS

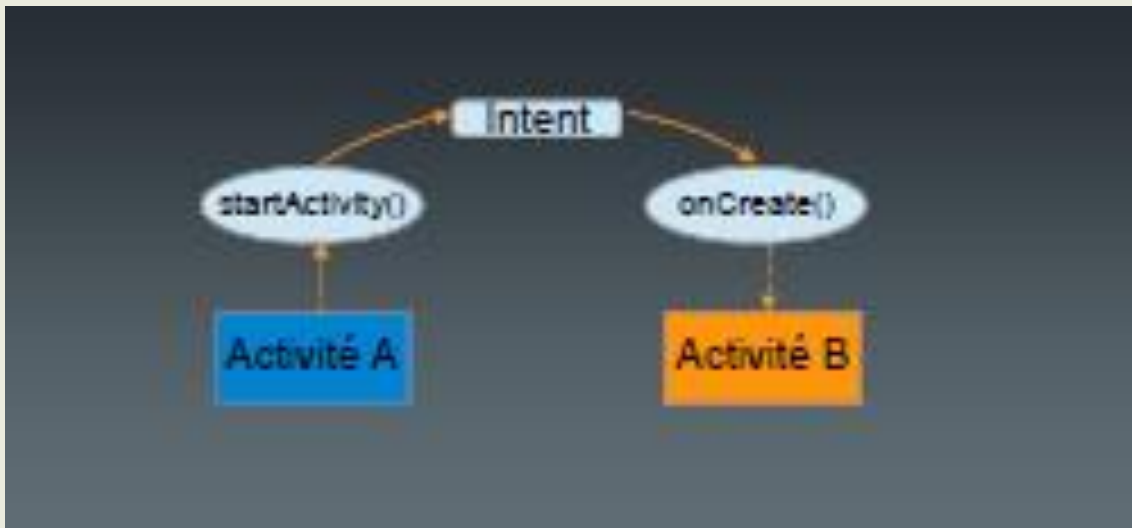
❑ Deux types de messages

- ◆ Explicite : on nomme le composant à démarrer
- ◆ Implicite : on demande au système de trouver un composant adéquat en fonction d'une action à effectuer.

LES INTENTS EXPLICITES

❑ Message adressé à un composant connu

- ◆ On donne le nom de la classe correspondante
- ◆ Réservé aux composants d'une même application



INTENTS POUR NOUVELLE ACTIVITÉ

- ❑ **Déclaration dans le manifest :**

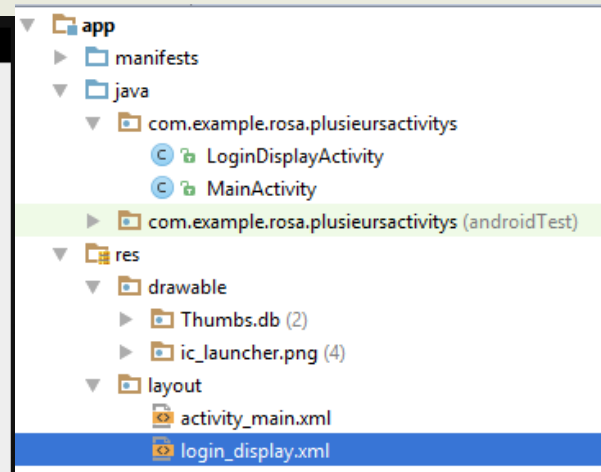
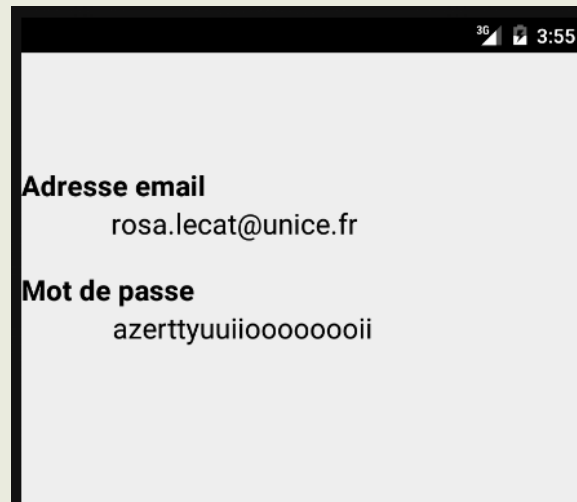
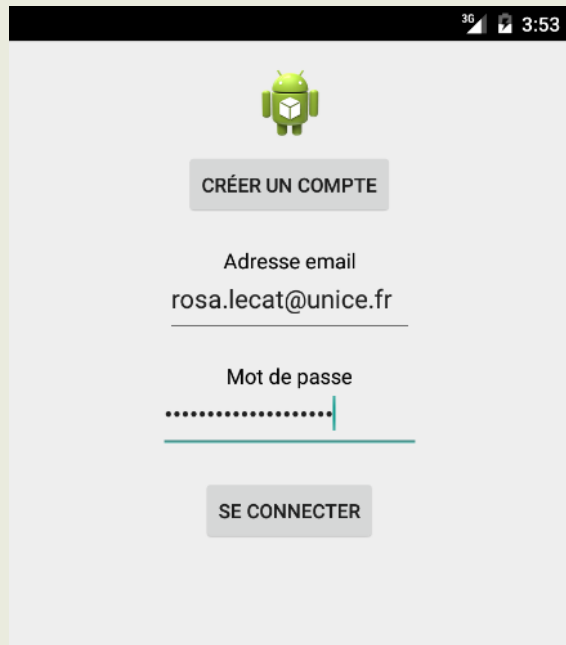
`<activity android:name="package.name.SecondActivity" />`

- ❑ **Pour passer la main à une activité interne à l'application, on peut créer l'Intent avec la classe de l'activité ciblée en paramètre :**

```
Intent monIntent = new Intent(MainActivity.this, SecondActivity.class);  
startActivity (monIntent);
```


ECHANGE DE DONNÉES ENTRE ACTIVITÉS

❑ Echange chaînes de caractères



ECHANGE DE DONNÉES ENTRE ACTIVITÉS

❑ Passage de chaînes de caractères entre activités

◆ //DANS MAINACTIVITY

```
Intent intent = new Intent(MainActivity.this,LoginDisplayActivity.class);  
intent.putExtra(LOGIN, loginTxt);  
intent.putExtra(PASSWORD, passTxt);  
startActivity(intent);
```

LOGIN et PASSWORD sont des chaines de caractère contenant le nom de « la variable » échangée

ECHANGE DE DONNEES ENTRE ACTIVITÉS

❑ Récupération des chaînes de caractères dans l'activités secondaires

```
Intent intent = getIntent();
```

```
TextView loginDisplay = (TextView) findViewById(R.id.email_display);
```

```
TextView passwordDisplay = (TextView) findViewById(R.id.password_display);
```

```
if (intent != null) {
```

```
    loginDisplay.setText(intent.getStringExtra(LOGIN));
```

```
    passwordDisplay.setText(intent.getStringExtra(PASSWORD));
```

ECHANGE DE DONNÉES PARCELABLES

❑ Passage d'objets entre activités

```
import android.os.Parcel;  
import android.os.Parcelable;
```

```
public class User implements Parcelable  
{
```

```
    private String nom;  
    private String prenom;
```

```
    //Constructeurs
```

```
    //Accesseurs
```

```
    public User(Parcel in) {  
        this.nom = in.readString();  
        this.prenom = in.readString();  
    }
```

```
    @Override
```

```
    public int describeContents() {  
return 0;  
    }
```

Objet de type User à passer
La classe doit implémenter l'interface Parcelable

ECHANGE DE DONNÉES PARCELABLES

❑ Passage d'objets entre activités

@Override

```
public void writeToParcel(Parcel dest, int flags) {  
    dest.writeString(nom);  
    dest.writeString(prenom);  
}
```

```
public static final Parcelable.Creator<User> CREATOR = new Parcelable.Creator<User>() {
```

@Override

```
    public User createFromParcel(Parcel source) {  
return new User(source);  
    }
```

@Override

```
public User[] newArray(int size)  
{  
    return new User[size];  
}
```

```
};
```

```
public static Parcelable.Creator<User> getCreator() {  
    return CREATOR;  
}  
} // fin de classe
```

ECHANGE DE DONNÉES PARCELABLES

❑ **Activité principale**

```
User user = new User("LECAT", "Rosa");  
Intent intent = new Intent(MainActivity.this, ResultActivity.class);  
intent.putExtra("user", user);  
startActivity(intent);
```

❑ **Activité secondaire**

```
User user = getIntent().getExtras().getParcelable("user");  
TextView displayUser = (TextView) findViewById(R.id.displayUser);  
displayUser.setText("User : " + "\n" + " NOM : " + user.getNom());
```