

# **CHAPITRE 04**

## **ANDROID**

<https://bit.ly/2KrvnjO>

# Composantes

- Composantes d'une application Android :
  - activity
  - service
  - broadcast receiver
  - content provider
- Autres :
- Views, Manifest, Intent, Resources, Layouts, Fragments

# Applications

- Ecrites en Java
- Séparation :
  - Chaque application s'exécute dans son propre processus.
  - Chaque processus possède sa propre machine virtuelle.
  - Chaque application est affectée à un seul user ID

# Composantes Majeures

- **Activities** – interface visuelle dédiée à une seule activité que l'utilisateur peut faire. Un écran.
- **Services** – pas d'interface visuelle. Tâche qui s'exécute en arrière-plan.
- **Broadcast Receivers** – reçoit et réagit à des annonces.
- **Content Providers** – échange de données entre applications.

# Activities

- Composantes de base de toute application
- La plupart des applications ont plusieurs activités. Une activité peut déclencher une autre activité.
- Une activité est implémentée en héritant de la classe Activity.

```
public class MainActivity extends Activity {  
  
}
```

# Activities – The View

- Chaque activité dispose d'une fenêtre par défaut pour dessiner (peut également déclencher des boîtes de dialogues ou des notifications).
- La fenêtre est constituée de vues ou de groupes de vues (dérivées de View ou de ViewGroup)
- Exemple de vues : buttons, text fields, scroll bars, menu items, check boxes, etc.
- View(Group) visibles à travers `Activity setContentView()`.

# Services

- Pas d'interface visuelle
- S'exécutent en arrière-plan.
- Exemples
  - Téléchargement (réseau)
  - Bandes sonores
  - Serveurs TCP/UDP
- Hérite de la classe Service

```
public class MyService extends Service {  
  
}
```

# Broadcast Receivers

- Reçoit et réagit aux broadcasts.
- Hérite de la class BroadcastReceiver.
- Exemples de broadcasts :
  - Batterie faible, branchement, préparation pour un shutdown, changement de zone horaire, etc.
  - D'autres applications peuvent initier un broadcast.

```
public class MyReceiver extends BroadcastReceiver {  
  
}
```

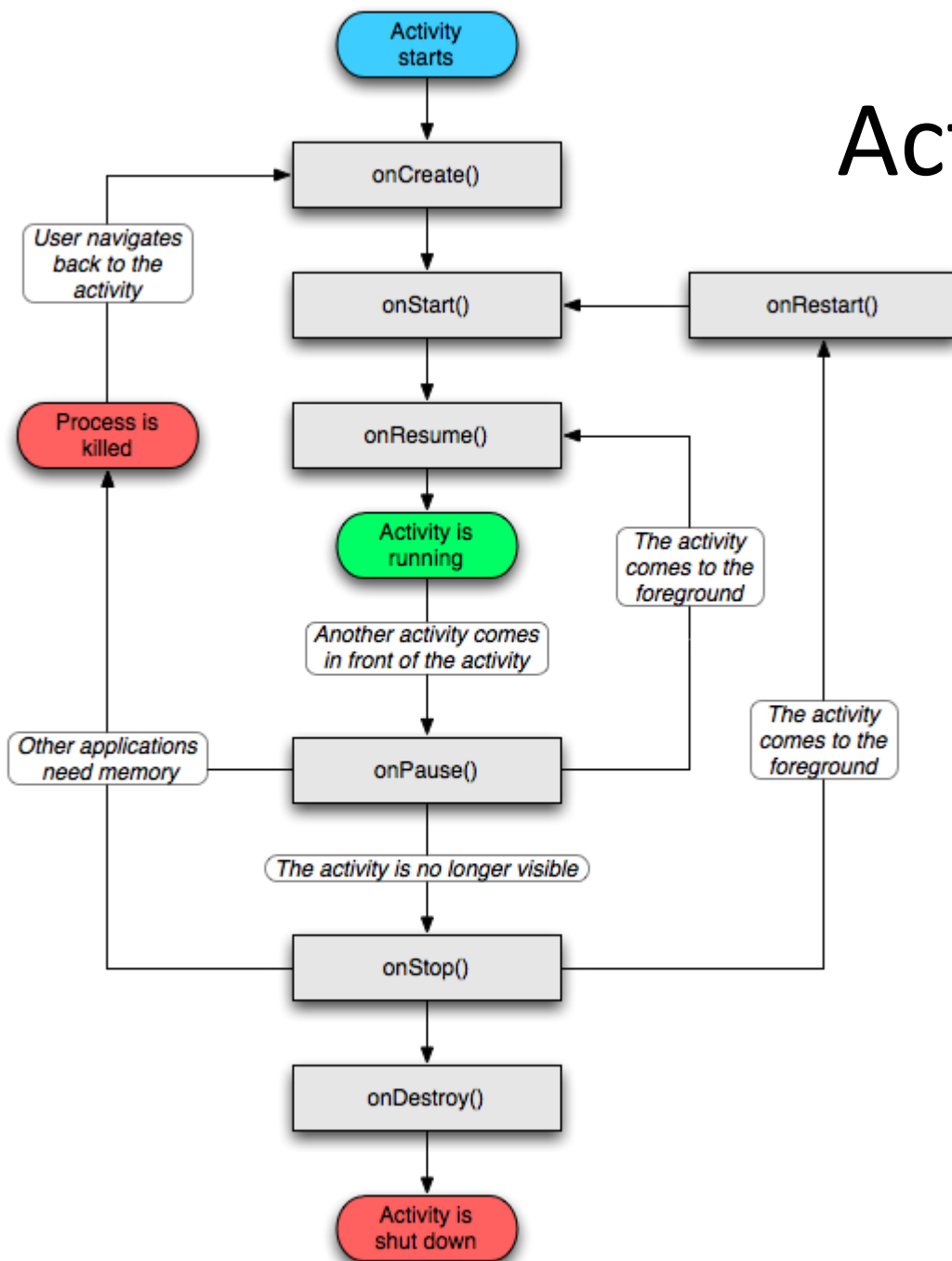


# Content Providers

- Met des données à disposition d'une application
- Permet d'accéder aux données de certaines applications.
- Le seul moyen d'échanger des données entre applications.
- Héritite de la classe `ContentProvider`;

```
public class MyContentProvider extends ContentProvider {  
  
}
```

# Activity life-cycle



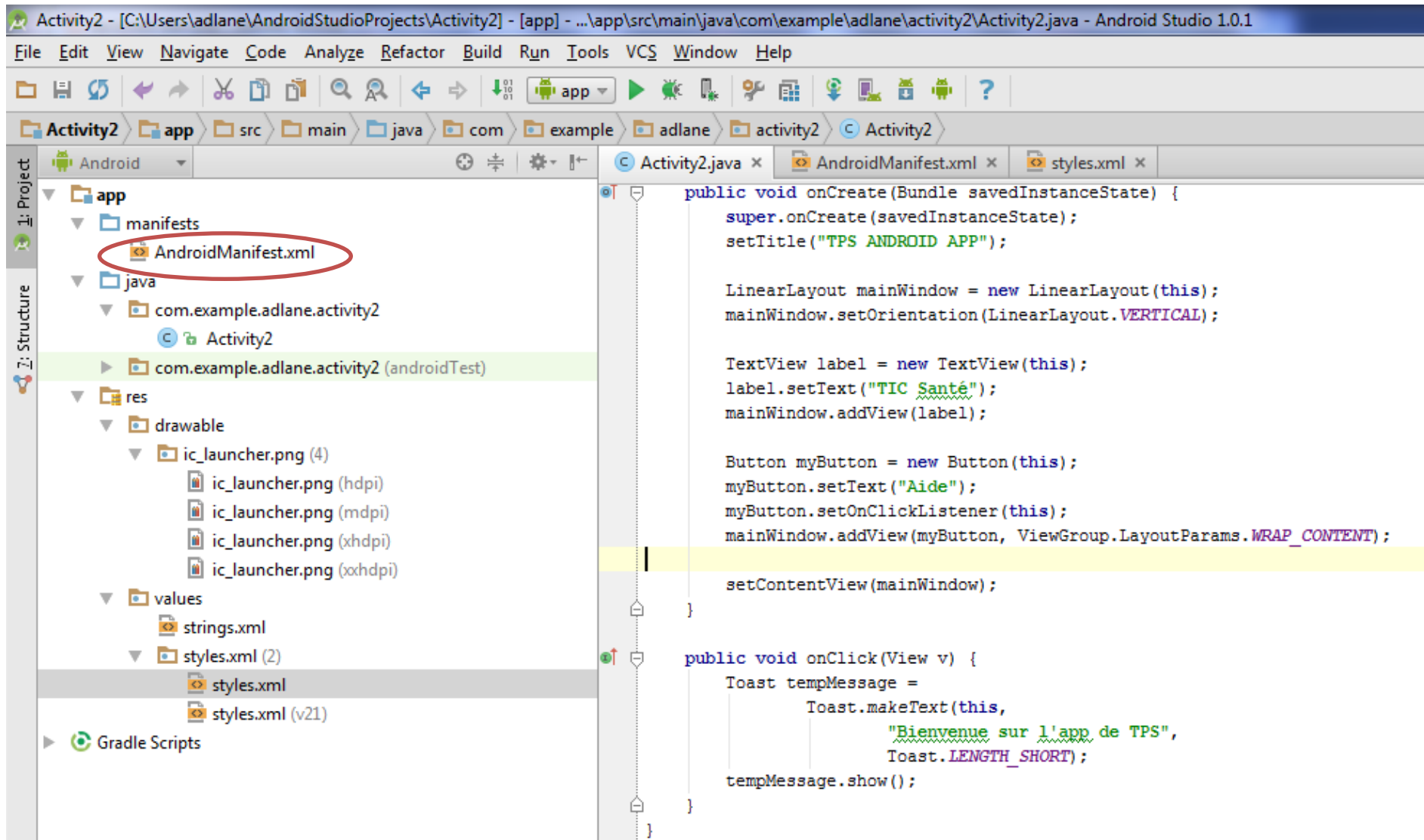
# Activity

```
public class Activity extends ApplicationContext {  
    protected void onCreate(Bundle  
savedInstanceState);  
  
    protected void onStart();  
  
    protected void onRestart();  
  
    protected void onResume();  
  
    protected void onPause();  
  
    protected void onStop();  
  
    protected void onDestroy();  
}
```

# Exemple 1

```
public class Activity2 extends Activity {  
  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setTitle("TPS ANDROID APP");  
  
        LinearLayout mainWindow = new LinearLayout(this);  
        mainWindow.setOrientation(LinearLayout.VERTICAL);  
  
        TextView label = new TextView(this);  
        label.setText("TIC Santé");  
        mainWindow.addView(label);  
  
        Button myButton = new Button(this);  
        myButton.setText("Aide");  
  
        mainWindow.addView(myButton);  
  
        setContentView(mainWindow);  
    }  
}
```





# AndroidManifest.xml

```
<manifest ...>
```

```
<application ...>
```

```
<activity
```

```
  android:name="com.example.adlane.activity2.Activity2"
```

```
<intent-filter>
```

```
  <action android:name="android.intent.action.MAIN" />
```

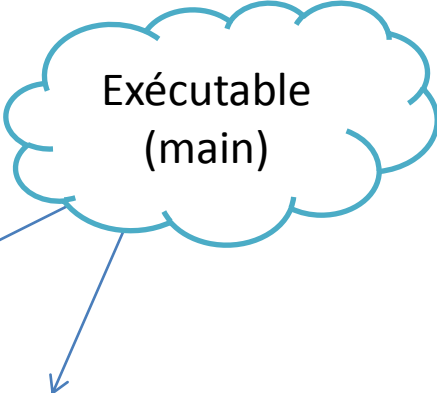
```
    <category android:name="android.intent.category.LAUNCHER" />
```

```
</intent-filter>
```

```
</activity>
```

```
</application>
```

```
</manifest>
```



Exécutable  
(main)

# View.OnClickListener

public static interface

## View.OnClickListener

android.view.View.OnClickListener

### ► Known Indirect Subclasses

[CharacterPickerDialog](#), [KeyboardView](#), [QuickContactBadge](#), [SearchOrbView](#), [SpeechOrbView](#)

## Class Overview

Interface definition for a callback to be invoked when a view is clicked.

## Summary

### Public Methods

abstract void	<a href="#">onClick</a> ( <a href="#">View v</a> ) Called when a view has been clicked.
---------------	--

# Example 2...

```
public class Activity2 extends Activity implements View.OnClickListener {
```

```
@Override
```

```
public void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
```

```
    setTitle("TPS ANDROID APP");
```

```
    LinearLayout mainWindow = new LinearLayout(this);
```

```
    mainWindow.setOrientation(LinearLayout.VERTICAL);
```

```
    TextView label = new TextView(this);
```

```
    label.setText("TIC Santé");
```

```
    mainWindow.addView(label);
```

```
    Button myButton = new Button(this);
```

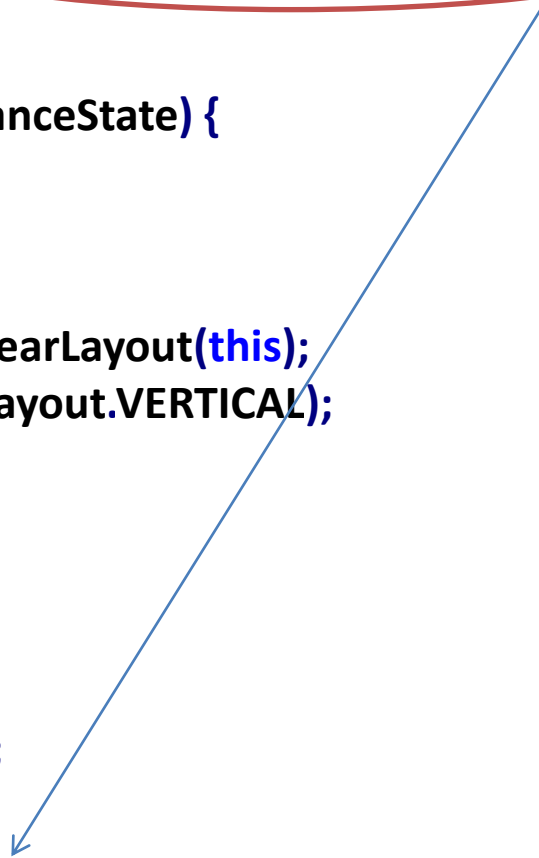
```
    myButton.setText("Aide");
```

```
    myButton.setOnClickListener(this);
```

```
    mainWindow.addView(myButton, ViewGroup.LayoutParams.WRAP_CONTENT);
```

```
    setContentView(mainWindow);
```

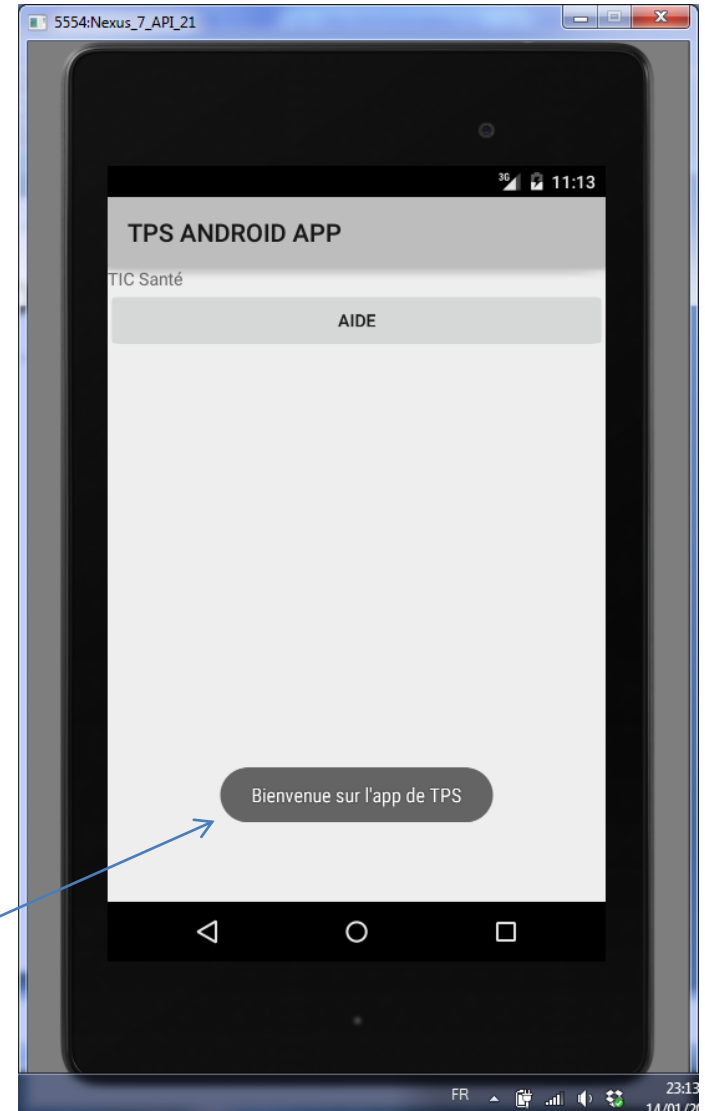
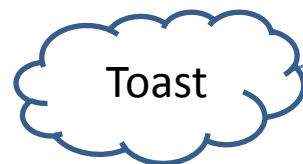
```
}
```

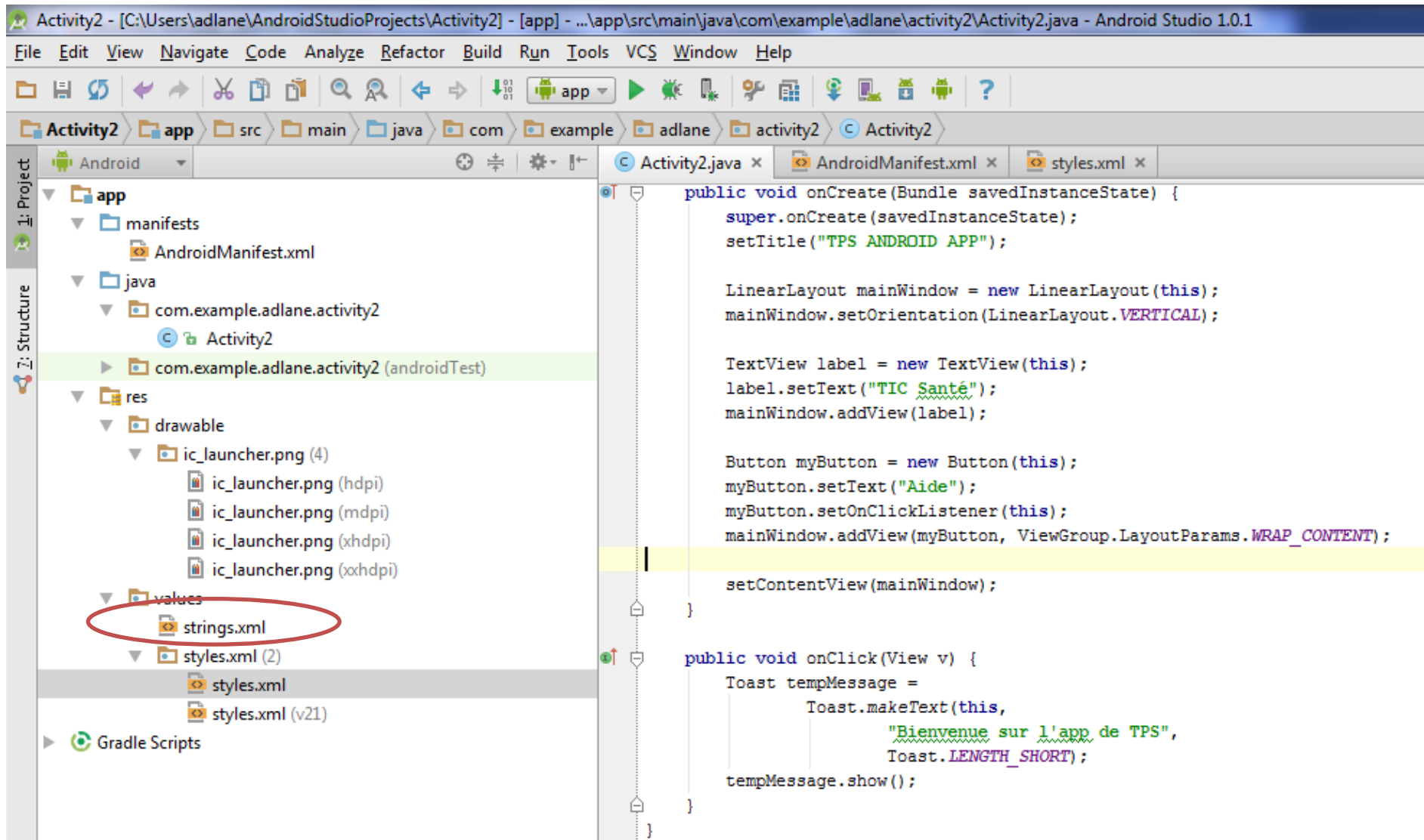




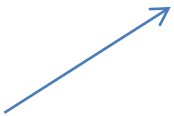
# ...suite exemple 2

```
public void onClick(View v) {  
    Toast tempMessage =  
        Toast.makeText(this,  
            "Bienvenue sur l'app de TPS",  
            Toast.LENGTH_SHORT);  
    tempMessage.show();  
}  
} //end class Activity2
```





# strings.xml



```
<resources>
  <string name="app_name">TPS ANDROID APP</string>
  <string name="tics">TIC Santé</string>
  <string name="aide">Aide</string>
</resources>
```

```
public class Activity2 extends Activity {
```

```
    @Override
```

```
    public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setTitle(R.string.app_name);
```

```
        LinearLayout mainWindow = new LinearLayout(this);
```

```
        mainWindow.setOrientation(LinearLayout.VERTICAL);
```

```
        TextView label = new TextView(this);
```

```
        label.setText(R.string.tics);
```

```
        mainWindow.addView(label);
```

```
        Button myButton = new Button(this);
```

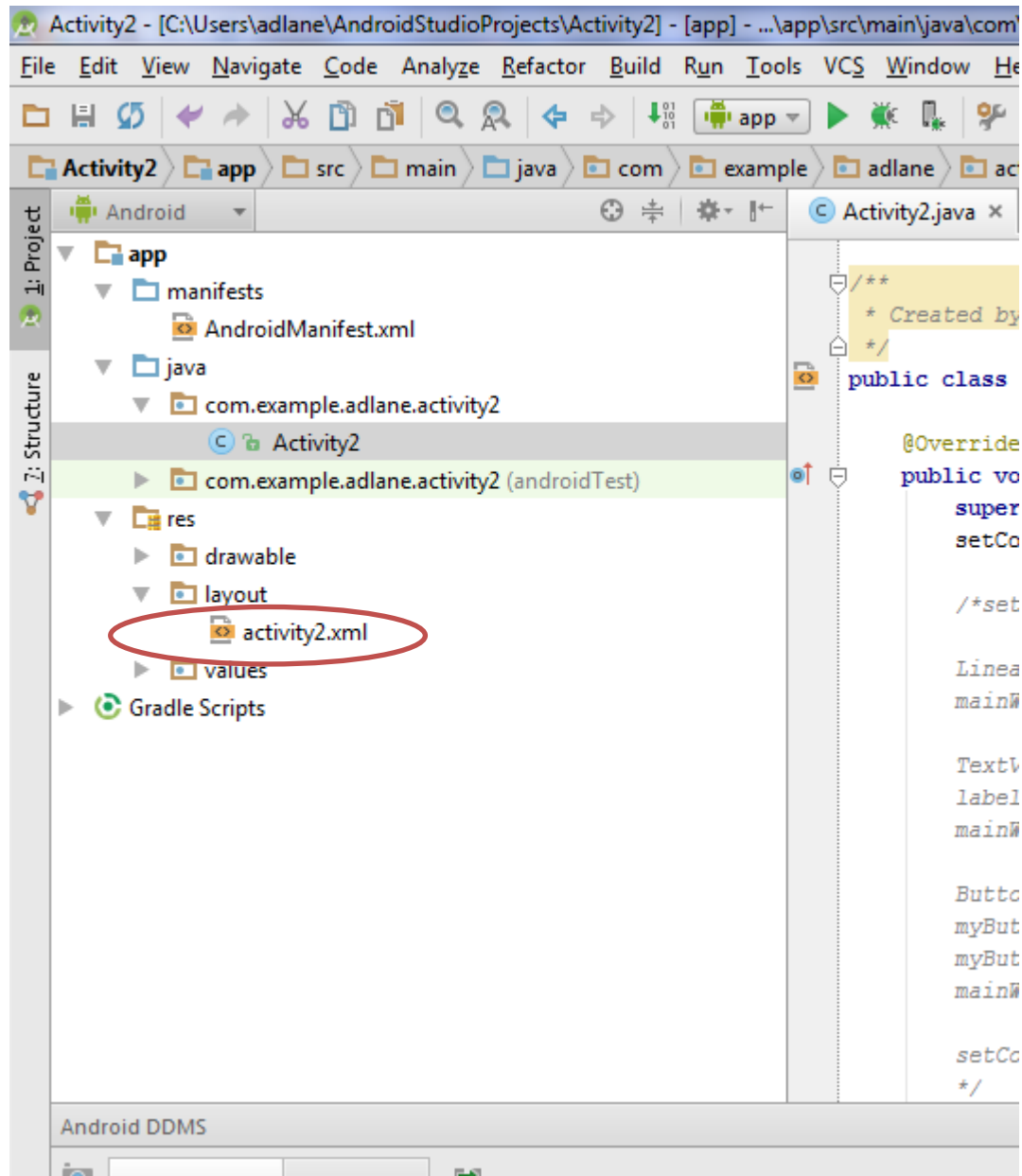
```
        myButton.setText(R.string.aide);
```

```
        mainWindow.addView(myButton);
```

```
        setContentView(mainWindow);
```

```
    }
```

```
...
```



# activity2.xml

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

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:text="@string/tics"/>

    <Button
        android:text="@string/aide"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:onClick="activer"/>

</LinearLayout>
```

# Simplification de Activity2

```
public class Activity2 extends Activity {  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity2);  
    }  
  
    public void activer(View v) {  
        Toast tempMessage =  
            Toast.makeText(this,  
                "Bienvenue sur l'app de TPS",  
                Toast.LENGTH_SHORT);  
        tempMessage.show();  
    }  
}
```

# XML vs.Java

- XML statique
- Java dynamique

Combiner les deux ?

# Ajouter des identifiants

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

**<TextView**

```
  android:layout_width="match_parent"  
  android:layout_height="wrap_content"  
  android:text="@string/tics"/>
```

**<Button**

```
  android:id="@+id/my_button"  
  android:text="@string/aide"  
  android:layout_width="match_parent"  
  android:layout_height="wrap_content"  
  android:onClick="activer"/>
```

```
</LinearLayout>
```



```
public class Activity2 extends Activity {  
    Button b;
```

```
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity2);
```

```
        b = (Button) findViewById(R.id.my_button);
```

```
    }
```

```
    public void activer(View v) {  
        b.setBackgroundColor(Color.BLUE);  
        Toast tempMessage =  
            Toast.makeText(this,  
                "Bienvenue sur l'app de TPS",  
                Toast.LENGTH_SHORT);  
        tempMessage.show();
```

```
    }
```

```
}
```



# EditText

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"  
    android:layout_width="match_parent" android:layout_height="match_parent"  
    android:orientation="vertical">
```

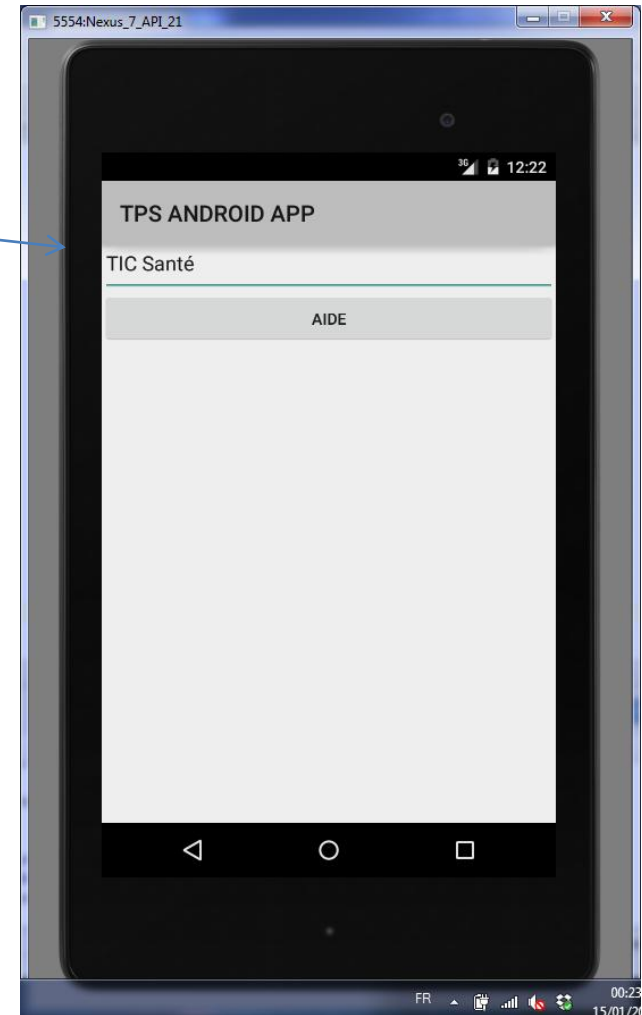
```
<EditText
```

```
    android:id="@+id/my_editor"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/tics"/>
```

```
<Button
```

```
    android:id="@+id/my_button"  
    android:text="@string/aide"  
    android:layout_width="match_parent"  
    android:layout_height="wrap_content"  
    android:onClick="activer"/>
```

```
</LinearLayout>
```



```

public class Activity2 extends Activity {
    Button b;
    EditText et;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity2);
        b = (Button) findViewById(R.id.my_button);
        et = (EditText) findViewById(R.id.my_editor);
    }

```

```

    public void activer(View v) {
        if (v==b)b.setBackgroundColor(Color.BLUE);
        else {
            Toast tempMessage =
                Toast.makeText(this,
                    et.getText(),
                    Toast.LENGTH_SHORT);
            tempMessage.show();
        }
    }
}

```



# Passage d'une activité à une autre


## Intents

# Possibilités

- Utilisation du nom de la classe d'activité qu'on souhaite lancer :  
L'activité à lancer doit être dans le même projet.
- Lancer une activité à travers un URI (Uniforme Ressource Identifier) :  
L'activité à lancer peut être à l'extérieur du projet.

# Par nom de classe

```
public class NewActivity extends Activity {  
    ...  
}
```



- Dans l'activité de départ :

```
Intent activityIntent = new Intent(this, NewActivity.class);  
startActivity(activityIntent);
```

- Dans AndroidManifest.xml

```
<activity android:name=".NewActivity"  
    android:label="@string/some_app_name">  
    <intent-filter>  
        <action android:name="android.intent.action.VIEW" />  
        <category android:name="android.intent.category.DEFAULT" />  
    </intent-filter>  
</activity>
```

# A travers un URI

Chaîne de caractères

- Dans l'activité de départ :

```
Intent activityIntent = new Intent(Intent.ACTION_VIEW, Uri.parse(s));  
startActivity(activityIntent);
```

# Dessiner

```
public class MyView extends View implements View.OnTouchListener {
```

```
    float x, y;
```

```
    public MyView(Context context) {  
        super(context);  
        x = 500; y = 500;  
        setLayoutParams(new ActionBar.LayoutParams(1000, 1000));  
        invalidate(); // équivalent de repaint()  
        setOnTouchListener(this);  
    }
```

```
    protected void onDraw(Canvas canvas){  
        super.onDraw(canvas);  
        Paint paint = new Paint();//Contrôle le style de dessin  
        paint.setColor(Color.RED);  
        canvas.drawRect(x,y,x+100,y+100,paint);  
    }
```

```
@Override
```

```
public boolean onTouch(View view, MotionEvent event) {  
    //Traitement approprié ici, puis  
    invalidate();  
    return true;  
}  
}
```