

Programmation mobile

Cours 1 : Généralités

Julien Grange <julien.grange@lacl.fr>

Mardi 9 septembre 2025



Informations pratiques

- CM : 6 séances de 1H30
- TP : 7 séances de 3H
- Enseignants : Quentin Le Houérou (TP), Julien Grange (CM et TP)
- Évaluation : 3 CC, moyenne des 2 meilleurs

Ajoutez une photo sur Eprel !

Outils

- IDE : Android Studio
<https://developer.android.com/studio>
- Guide et documentation
<https://developer.android.com/docs>
- Développement en **Java** ou **Kotlin**
- Exécution possible
 - sur vos smartphones et tablettes Android
 - sur le simulateur intégré

AndroidStudio

Cours1 – MainActivity.java [Cours1.app]

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Cours1 app src main java com.example.cours1 MainActivity
strings.xml AndroidManifest.xml activity_main.xml MainActivity.java activity_hello_world.xml HelloWorld.java
17 // onCreate est appellée :
18     - quand l'activité est créée
19     - à chaque fois qu'elle est recréée, que ce soit
20         - parce que l'BS l'a tuée pour libérer des ressources
21         - parce que l'écran a été renversé (besoin de recréer l'UI)
22 }
23
24 @Override
25 protected void onCreate(Bundle savedInstanceState) {
26     super.onCreate(savedInstanceState);
27     // affiche le layout tel que défini dans activity_main.xml
28     setContentView(R.layout.activity_main);
29
30     // si on ne vient pas juste d'être créé, on récupère la valeur du compteur sauvegardée
31     if(savedInstanceState!=null){
32         compteur=savedInstanceState.getInt(COMPTEUR, defaultValue: 0);
33     }
34
35     // on récupère les vues définies dans activity_main.xml
36     myTextView = findViewById(R.id.textView);
37     Button incrButton = findViewById(R.id.button);
38     Button displayButton = findViewById(R.id.button2);
39
40     // on affiche le compteur
41     myTextView.setText(Integer.toString(compteur));
42
43     // on définit l'action à effectuer (callback) quand l'utilisateur appuie sur incrButton
44     // on fait ici appel à une implémentation anonyme de l'interface View.OnClickListener, pour gagner du temps
45     incrButton.setOnClickListener(new View.OnClickListener(){
46         @Override
47         public void onClick(View view) {myTextView.setText(Integer.toString(+compteur)); }
48     });
49
50
51     displayButton.setOnClickListener(new View.OnClickListener() {
52         @Override
53         public void onClick(View view) {
54             // on crée un intent dans le but de passer la main à l'activité helloWorld
55             Intent intent = new Intent(getApplicationContext(), HelloWorld.class);
56             // on ajoute la valeur du compteur à l'intent, avec la clef COMPTEUR
57             intent.putExtra(COMPTEUR,compteur);
58             startActivity(intent);
59         }
60     });
61 }
```

12:40 Cours1

INCREMENT

DISPLAY

Device File Explorer Emulator

Event Log Layout Inspector

17:1 LF UTF-8 4 spaces

Julien Grange

Programmation mobile

Conventions du cours

- Les classes et interfaces seront encadrées : **Classe**
- Les méthodes (et champs) statiques d'une classe sont dénotés `Class.foo()`
- Les méthodes (et champs) non-statiques (i.e. propres aux objets) d'une classe sont dénotés `Class::foo()`
- Si la classe en question est claire, on se contentera de `foo()`

Principes généraux

- Chaque application est composée d'une ou plusieurs **Activity**
- Chaque écran correspond (grossso modo) à une activité

Principes généraux

- Chaque application est composée d'une ou plusieurs **Activity**
- Chaque écran correspond (grossso modo) à une activité
- Le layout d'une activité est constitué d'un arbre de **View**
 - Les feuilles sont des **View**, e.g.
 - **TextView**
 - **Button**
 - Les nœuds internes sont des **ViewGroup**, e.g.
 - **LinearLayout**
 - **ScrollView**

Principes généraux

- Chaque application est composée d'une ou plusieurs **Activity**
- Chaque écran correspond (grossièrement) à une activité
- Le layout d'une activité est constitué d'un arbre de **View**
 - Les feuilles sont des **View**, e.g.
 - **TextView**
 - **Button**
 - Les nœuds internes sont des **ViewGroup**, e.g.
 - **LinearLayout**
 - **ScrollView**
- Une activité peut lancer une autre activité via un **Intent**
 - soit en la nommant (interne à une application)
 - soit à la criée (e.g. pour prendre une photo)

Principes généraux

- Chaque application est composée d'une ou plusieurs **Activity**
- Chaque écran correspond (grossso modo) à une activité
- Le layout d'une activité est constitué d'un arbre de **View**
 - Les feuilles sont des **View**, e.g.
 - **TextView**
 - **Button**
 - Les nœuds internes sont des **ViewGroup**, e.g.
 - **LinearLayout**
 - **ScrollView**
- Une activité peut lancer une autre activité via un **Intent**
 - soit en la nommant (interne à une application)
 - soit à la criée (e.g. pour prendre une photo)
- Programmation événementielle, à base de callbacks

Gestion agressive des ressources par l'OS

- Dès qu'une activité n'est plus visible, elle peut être tuée si l'OS a besoin de ressources
- Elle est alors relancée quand l'utilisateur la repasse en premier plan
- Idem à chaque rotation d'écran (nouveau calcul de l'UI)

Gestion agressive des ressources par l'OS

- Dès qu'une activité n'est plus visible, elle peut être tuée si l'OS a besoin de ressources
- Elle est alors relancée quand l'utilisateur la repasse en premier plan
- Idem à chaque rotation d'écran (nouveau calcul de l'UI)

Il faut sauvegarder les données courantes dès qu'on quitte le premier plan.

Première application : Hello World!

Deux `Activity` :

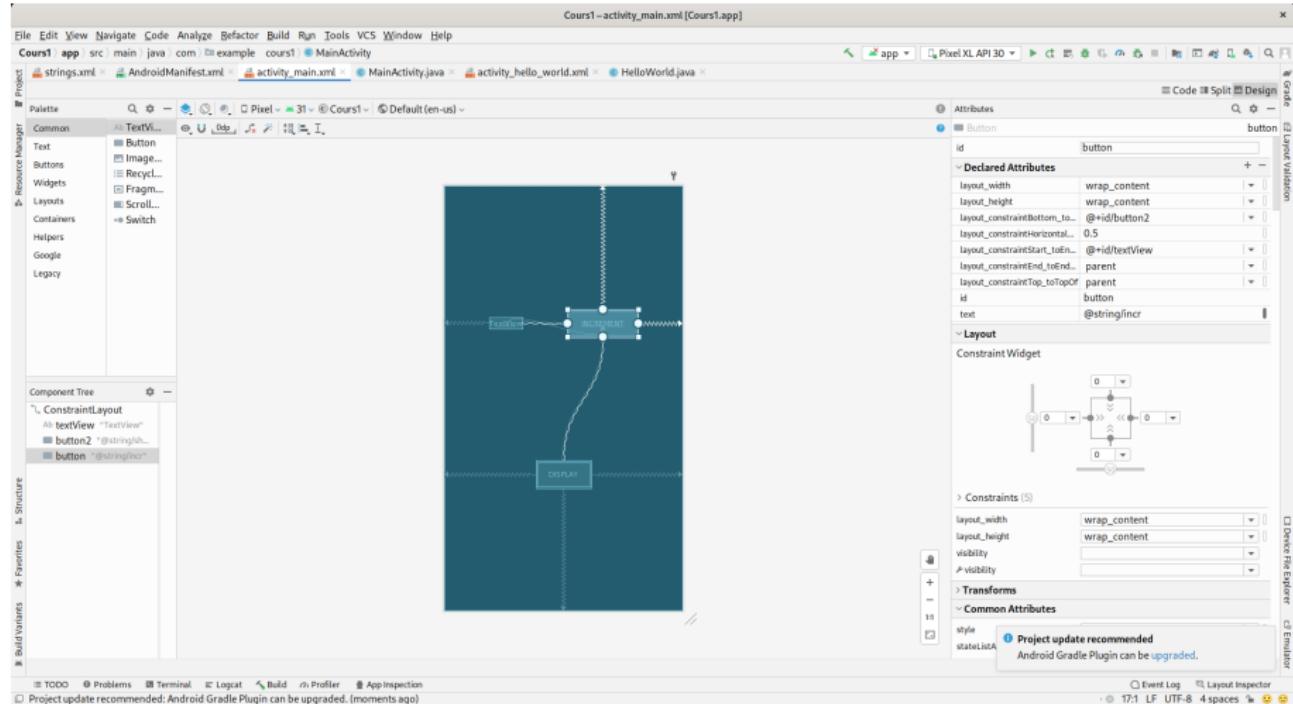
- `MainActivity` : classe principale, accessible depuis le launcher
 - premier bouton → incrémente un compteur
 - deuxième bouton → lance `HelloWorld` via un `Intent`

Première application : Hello World!

Deux **Activity** :

- **MainActivity** : classe principale, accessible depuis le launcher
 - premier bouton → incrémente un compteur
 - deuxième bouton → lance **HelloWorld** via un **Intent**
- **HelloWorld**
 - affiche autant de fois "Hello World !" que la valeur du compteur

MainActivity | layout



MainActivity code

The screenshot shows the Android Studio interface with the code editor open to `MainActivity.java`. The code implements a counter application with a static final string `COMPTEUR` and handles the `onCreate` and `onSaveInstanceState` methods.

```
package com.example.cours1;

import ...

public class MainActivity extends AppCompatActivity {

    private TextView myTextView;
    private int compteur;
    public static final String COMPTEUR = "com.example.cours1.COMPTEUR";

    /* onCreate est appelée :
       - quand l'activité est créée
       - à chaque fois qu'elle est recréée, que ce soit
          - parce que l'OS l'a tuée pour libérer des ressources
          - parce que l'écran a été renversé (besoin de recalculer l'UI)
    */
    @Override
    protected void onCreate(Bundle savedInstanceState) { ... }

    //onSaveInstanceState permet de sauvegarder l'état de l'activité
    //ici, on ne veut pas perdre la valeur de compteur si l'activité est tuée
    @Override
    protected void onSaveInstanceState(Bundle outState) { ... }
}
```

The code editor includes annotations in French explaining the behavior of the `onCreate` method. The AndroidManifest.xml and activity_main.xml files are visible in the Project tab, and the Pixel XL API 30 emulator is selected in the toolbar.

MainActivity::onCreate()

The screenshot shows the Android Studio interface with the following details:

- Project:** Cours1
- File:** MainActivity.java [Cours1.app]
- Code Content:** The code implements the `onCreate` method. It first checks if the activity is being recreated (due to a configuration change or the app being killed and restored). If so, it retrieves the previous value of the counter from `savedInstanceState`. Then, it initializes views from `activity_main.xml`: `myTextView`, `incrButton`, and `displayButton`. It sets the initial counter value to 0. Next, it defines a click listener for the increment button (`incrButton`) that updates the text view. Finally, it sets a click listener for the display button (`displayButton`) that starts a new activity (`HelloWorld`) with the current counter value as an extra.
- Emulator:** An iPhone X-like device is shown running the application. The screen displays "Cours1" at the top. Below it, there are two purple buttons: "INCREMENT" and "DISPLAY".
- Toolbars and Menus:** Standard Android Studio menus like File, Edit, View, Navigate, Code, Analyze, Refactor, Build, Run, Tools, VCS, Window, Help are visible at the top. Bottom toolbars include Build, Run, App Inspection, Event Log, Layout Inspector, and a status bar showing "17:1 LF UTF-8 4 spaces".

Activity::onCreate()

```
public class MainActivity extends AppCompatActivity {  
  
    private TextView myTextView;  
  
    protected void onCreate(Bundle savedInstanceState) {  
  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
  
        myTextView = findViewById(R.id.textView);  
        ...  
    }  
}
```

View::setOnClickListener()

```
incrButton.setOnClickListener(new View.OnClickListener(){

    @Override
    public void onClick(View view){
        myTextView.setText(Integer.toString(++compteur));
    }

});
```

HelloWorld code

The screenshot shows the Android Studio interface with the following details:

- Code Editor:** Displays the `HelloWorld.java` file under the `Cours1/app/src/main/java/com/example/cours1` package. The code implements an `onCreate` method that appends "Hello World!\n" to a `TextView` 20 times.
- Emulator:** Shows a Pixel XL API 30 device running the application. The screen displays the text "Hello World!" repeated 20 times.
- Status Bar:** Shows the time as 12:36 and battery level.
- Bottom Status Bar:** Shows "Project update recommended" and "Android Gradle Plugin can be upgraded".
- Bottom Navigation:** Includes tabs for TODO, Problems, Terminal, Logcat, Build, Profiler, Run, App Inspection, Event Log, Layout Inspector, and a message about a successful launch.

Activity::startActivity()

Dans **MainActivity** :

```
public static final String COMPTEUR = "cours1.COMPTEUR";  
  
//dans le setOnClickListener() du second bouton  
public void onClick(View view) {  
  
    Intent intent = new Intent(getApplicationContext(), HelloWorld.  
        class);  
    Intent.putExtra(COMPTEUR,compteur);  
    startActivity(intent);  
}
```

Dans **HelloWorld** :

```
protected void onCreate(Bundle savedInstanceState) {  
  
    Intent intent = getIntent();  
    int compteur = intent.getIntExtra(MainActivity.COMPTEUR,1);  
    ...  
}
```

Activity::onSaveInstanceState()

Rappel : une **Activity** peut être tuée (malgré elle) en cas de

- passage au second plan (typiquement, en cas d'appel)
- rotation d'écran

On peut sauvegarder des clefs/valeurs dans un **Bundle** dans la méthode `onSaveInstanceState()`, qui permettront de retrouver les valeurs courantes lors du prochain appel à `onCreate()`.

MainActivity::onSaveInstanceState()

The screenshot shows the Android Studio interface with the following details:

- Project Bar:** Cours1 - MainActivity.java [Cours1.app]
- File Structure:** Shows the project structure with files like strings.xml, AndroidManifest.xml, activity_main.xml, MainActivity.java, activity_hello_world.xml, and HelloWorld.java.
- Code Editor:** The MainActivity.java file is open, showing Java code for an AppCompatActivity. A callout box highlights the `onCreate` method with the following notes:
 - quand l'activité est créée
 - à chaque fois qu'elle est recréée, que ce soit
 - parce que l'OS l'a tuée pour libérer des ressources
 - parce que l'écran a été renversé (besoin de recalculer l'UI)
- Emulator:** A Pixel XL API 30 emulator is running, displaying the app's UI with the title "Cours1". The UI includes a counter labeled "4" and two buttons: "INCREMENT" and "DISPLAY".
- Bottom Bar:** Includes tabs for TODO, Problems, Terminal, Logcat, Build, Profiler, Run, and App Inspection. It also shows "Launch succeeded (5 minutes ago)" and "17:1 LF UTF-8 4 spaces".
- Right Sidebar:** Contains icons for Device File Explorer, Emulator, and other developer tools.

MainActivity::onSaveInstanceState()

```
private int compteur;
public static final String COMPTEUR = "cours1.COMPTEUR";

protected void onSaveInstanceState(Bundle outState) {

    outState.putInt(COMPTEUR,compteur);
    super.onSaveInstanceState(outState);

}

protected void onCreate(Bundle savedInstanceState) {

    super.onCreate(savedInstanceState);

    if(savedInstanceState!=null){
        compteur = savedInstanceState.getInt(COMPTEUR,0);
    }
    ...
}
```

Android Manifest

The screenshot shows the Android Studio interface with the project 'Cours1' open. The main window displays the AndroidManifest.xml file, which defines the application structure and its activities.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.cours1">

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Cours1"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Cours1">
        <activity
            android:name=".HelloWorld"
            android:exported="true" />
        <activity
            android:name=".MainActivity"
            android:exported="true">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

The code editor shows the XML structure of the manifest file. The application tag contains two activity tags. The first activity is exported, and both activities have intent filters. The second activity is set as the main launcher activity. The manifest ends with a closing manifest tag.