

Programmation mobile

Cours 2 : Intent

Julien Grange <julien.grange@acl.fr>

Mardi 23 septembre 2025



Rappel : lancer une **Activity** sans attendre de résultat

Dans l'**Activity** appelante :

```
int value1 = ...;
String value2 = ...;

// Intent explicite
Intent intent = new Intent(context, NewActivity.class);
intent.putExtra("Key1",value1);
intent.putExtra("Key2",value2);

startActivity(intent);
```

Dans **NewActivity** :

```
Intent intent = getIntent();

int myInt = intent.getIntExtra("Key1",42);
String myString = intent.getStringExtra("Key2");
```

Quatre façons de lancer une `Activity`

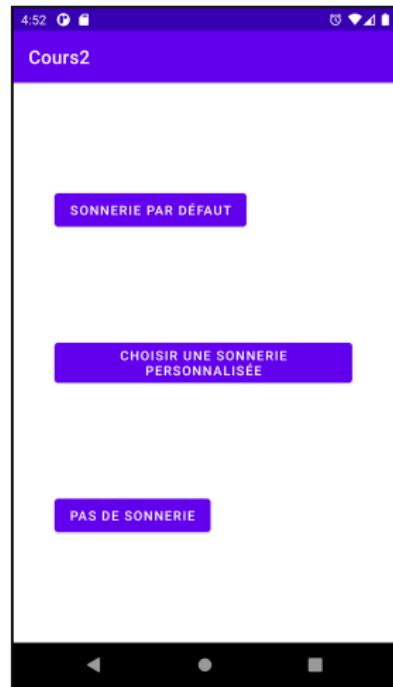
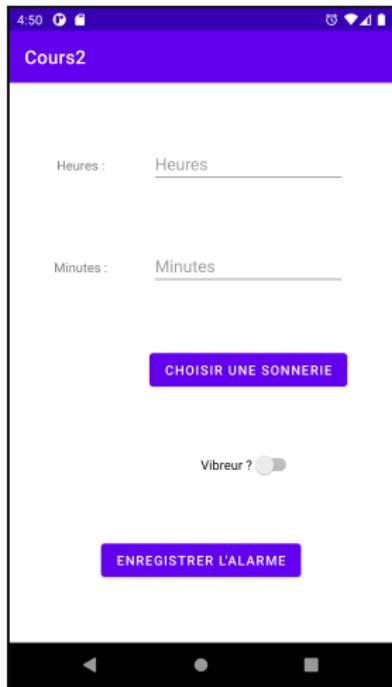
Deux types d'`Intent` :

- explicite (en nommant l'`Activity`)
- implicite (en spécifiant le type d'action)

Deux manières de lancer une `Activity` :

- sans attendre de résultat : `startActivity()`
- dans l'attente d'un résultat : `registerForActivityResult()`

Application du jour : Alarme



Application du jour : Alarme

Trois appels à une autre **Activity** :

- ① “Choisir une sonnerie” : explicite, avec résultat
- ② “Choisir une sonnerie personnalisée” : implicite, avec résultat
- ③ “Enregistrer l’alarme” : implicite, sans résultat

On a déjà vu les appels explicites sans résultat.

MainActivity layout

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help

Cours2 app src main res layout activity_main.xml

strings.xml activity_main.xml MainActivity.java activity_ringtone.xml Ringtone.java AndroidManifest.xml build.gradle (Cours2) build.gradle (app)

Project Resource Manager Favorites Build Variants

Code Split Editor

Attributes

id: hours

Declared Attributes

layout_width: wrap_content
layout_height: wrap_content
layout_constraintBottom_toBottomOf: @+id/minutes
layout_constraintHorizontal_bias: 0.5
layout_constraintStart_toStartOf: parent
layout_constraintTop_toBottomOf: @+id/vibreur />

layout_constraintBottom_toTopOf: parent
layout_constraintEnd_toEndOf: parent
layout_constraintHorizontal_bias: 0.5
layout_constraintStart_toEndOf: @+id/hours
layout_constraintTop_toTopOf: parent

ems: 10
hint: @string/eHeures
id: hours
inputType: numberDecimal

Layout

Constraint Widget

Constraints (5)

layout_width: wrap_content
layout_height: wrap_content
visibility: visible
A visibility

Transforms

Common Attributes

Event Log Layout Inspector

61.5 LF UTF-8 4 spaces

Julien Grange

Programmation mobile

MainActivity code

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

- Project Bar:** Cours2 app src main java com.example.cours2 MainActivity.java
- Main Activity Code (MainActivity.java):**

```
package com.example.cours2;

import ...

public class MainActivity extends AppCompatActivity {

    private EditText editTextHours, editTextMinutes;
    private Button buttonRingtone, buttonAdd;
    private Switch switchVibration;

    public static final String
        NOTYET = "com.example.cours2.NOTYET",
        RINGTONE = "com.example.cours2.RINGTONE",
        DEFAULT = "com.example.cours2.DEFAULT",
        CHOSEN = "com.example.cours2.CHOSEN",
        NO = "com.example.cours2.NO";

    private String typeRingtone=NOTYET;
    private Uri ringtone;

    //...
    private ActivityResultCallback<Intent> callback = new ActivityResultCallback<Intent>() {...};

    private ActivityResultContract<Void, Intent> contract = new ActivityResultContract<Void, Intent>() {...};

    // à lancer pour débuter l'activité Ringtone
    ActivityResultLauncher<Void> selectRingtone = registerForActivityResult(contract, callback);

    @Override
    protected void onCreate(Bundle savedInstanceState) {...}

    //on attache les views déclarées dans le XML
    private void catchViews() {...};

    private void ringtoneIssue() {...}
    private void formatIssue() {...}
}
```

- Emulator View:** An iPhone X-style emulator displays a ringtone selection screen titled "Cours2". The screen includes fields for "Heures" and "Minutes", a "CHOISIR UNE SONNERIE" button, a "Vibration ?" toggle switch, and an "ENREGISTRER L'ALARME" button.
- Bottom Navigation Bar:** Event Log, Layout Inspector, Logcat, Build, Profiler, Run, App Inspection.
- Status Bar:** Launch succeeded (3 minutes ago), 74:48 LF, UTF-8, 4 spaces.

`registerForActivityResult()`

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

- Project Bar:** Cours2 app, src, main, java, com.example.cours2, MainActivity, selectingRingtone
- Main Editor:** Content of MainActivity.java. The code handles ringtone selection via startActivityForResult. It defines a callback to get the selected ringtone type (CHOSEN or DEFAULT) and creates an intent contract for Ringtones.
- Preview Area:** Shows a mobile phone screen titled "Cours2". The screen has fields for "Heures" (Hours) and "Minutes" (Minutes), a "CHOISIR UNE SONNERIE" (Select a ringtone) button, and a "Vibrer ?" (Vibrate?) switch. Below these are "ENREGISTRER L'ALARME" (Save alarm) and a back/forward navigation bar.
- Bottom Status Bar:** Event Log, Layout Inspector, Device File Explorer, and Emulator tabs.
- Bottom Navigation:** TODO, Problems, Terminal, Logcat, Profiler, Run, App Inspection.
- Bottom Status:** Launch succeeded (8 minutes ago), 7448 LF, UTF-8, 4 spaces, and a smiley face icon.

registerForActivityResult()

Paramétré par deux classes : **I** (input) et **O** (output)

```
public ActivityResultLauncher<I> registerForActivityResult  
(ActivityResultContract<I,O> contract,  
ActivityResultCallback<O> callback)
```

registerForActivityResult()

Paramétré par deux classes : **I** (input) et **O** (output)

```
public ActivityResultLauncher<I> registerForActivityResult  
(ActivityResultContract<I,O> contract,  
ActivityResultCallback<O> callback)
```

- callback : implémente l'interface **ActivityResultCallback<O>**

```
public void onActivityResult(O result)
```

registerForActivityResult()

Paramétré par deux classes : **I** (input) et **O** (output)

```
public ActivityResultLauncher<I> registerForActivityResult  
(ActivityResultContract<I,O> contract,  
ActivityResultCallback<O> callback)
```

- **callback** : implémente l'interface **ActivityResultCallback<O>**

```
public void onActivityResult(O result)
```

- **contract** : implémente l'interface **ActivityResultContract<I,O>**

```
public Intent createIntent(Context context, I input)  
public O parseResult(int resultCode, Intent intent)
```

registerForActivityResult()

Paramétré par deux classes : **I** (input) et **O** (output)

```
public ActivityResultLauncher<I> registerForActivityResult  
(ActivityResultContract<I,O> contract,  
ActivityResultCallback<O> callback)
```

- **callback** : implémente l'interface **ActivityResultCallback<O>**

```
public void onActivityResult(O result)
```

- **contract** : implémente l'interface **ActivityResultContract<I,O>**

```
public Intent createIntent(Context context, I input)  
public O parseResult(int resultCode, Intent intent)
```

- l'appel peut ensuite être fait via la méthode `launch(I input)` du résultat de `registerForActivityResult()`

registerForActivityResult()

Paramétré par deux classes : **I** (input) et **O** (output)

```
public ActivityResultLauncher<I> registerForActivityResult  
(ActivityResultContract<I,O> contract,  
ActivityResultCallback<O> callback)
```

- **callback** : implémente l'interface **ActivityResultCallback<O>**

```
public void onActivityResult(O result)
```

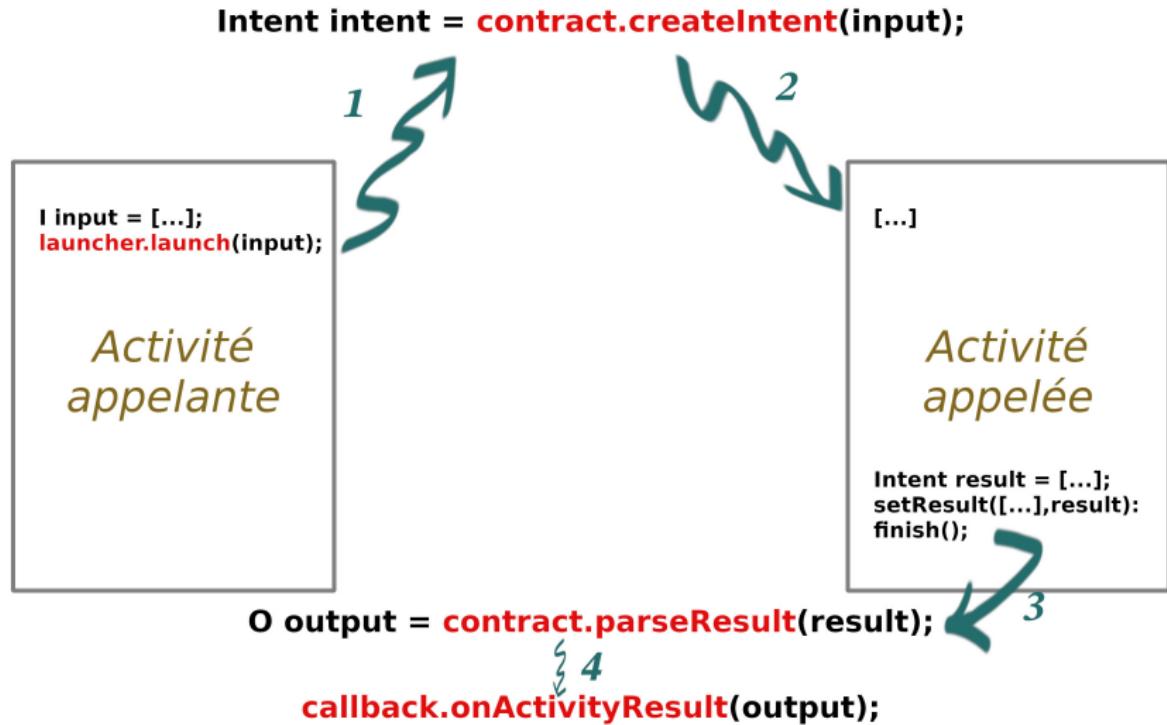
- **contract** : implémente l'interface **ActivityResultContract<I,O>**

```
public Intent createIntent(Context context, I input)  
public O parseResult(int resultCode, Intent intent)
```

- l'appel peut ensuite être fait via la méthode `launch(I input)` du résultat de `registerForActivityResult()`

Le callback doit être défini dès la création de l'**Activity** appelante.

registerForActivityResult() : derrière la scène



```
registerForActivityResult()
```

Ici, **I** = **Void** (aucun input) et **O** = **Intent**.

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Cours2 app src main java com.example.cours2 MainActivity selectRingtone
strings.xml activity_main.xml MainActivity.java activity_ringtone.xml Ringtone.java AndroidManifest.xml build.gradle(Cours2) build.gradle(:app)
1224 Cours2
Heures: Minutes: CHOSIR UNE SONNERIE
Vibrer? ENREGISTRER L'ALARME
1
/*
 * ! IMPORTANT !
Le callback doit être déclaré dès la création de l'activité !
En effet, il est possible que l'activité appellante ait été détruite avant le retour l'activité appelée ;
il faut malgré tout que la nouvelle instance de l'activité appellante sache comment traiter le résultat !
*/
private ActivityResultCallback<Intent> callback = new ActivityResultCallback<Intent>() {
    @Override
    public void onActivityResult(Intent result) {
        typeRingtone = result.getStringExtra(RINGTONES);
        if(typeRingtone.equals(CHOOSEN)){
            ringtone = result.getData();
            if(ringtone == null){
                //verrou durant la sélection
                typeRingtone = DEFAULT;
            }
        }
    }
};

private ActivityResultContract<Void, Intent> contract = new ActivityResultContract<Void, Intent>() {
    @Override
    public Intent createIntent(Context context, Void input) {
        return new Intent(context, Ringtone.class);
    }

    @Override
    public Intent parseResult(int resultCode, Intent intent) { return intent; }
};

//à lancer pour débuter l'activité Ringtone
ActivityResultLauncher<Void> selectRingtone = registerForActivityResult(contract, callback);
```

registerForActivityResult()

Cours2 – MainActivity.java [Cours2.app]

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Cours2 app : src : main : java : com : example : cours2 : MainActivity : selectRingtone
strings.xml activity_main.xml MainActivity.java activity_ringtone.xml Ringtone.java AndroidManifest.xml build.gradle (Cours2) build.gradle (app)
Project Resource Manager
73
74 // à lancer pour débuter l'activité Ringtone
75 ActivityResultLauncher<Void> selectRingtone = registerForActivityResult(contract, callback);
76
77 @Override
78 protected void onCreate(Bundle savedInstanceState) {
79     super.onCreate(savedInstanceState);
80     setContentView(R.layout.activity_main);
81     catchViews();
82
83     buttonRingtone.setOnClickListener(new View.OnClickListener() {
84         @Override
85         public void onClick(View view) { selectRingtone.launch( input: null); }
86     });
87
88     buttonAdd.setOnClickListener(new View.OnClickListener() {...});
89
90     buttonAdd.setonClickListener(new View.OnClickListener() {...});
91
92     // on attrape les views déclarées dans le XML
93     private void catchViews(){
94         editHours = findViewById(R.id.hours);
95         editMinutes = findViewById(R.id.minutes);
96         buttonRingtone = findViewById(R.id.ringtone);
97         switchVibreur = findViewById(R.id.vibreur);
98         buttonAdd= findViewById(R.id.add);
99     }
100 }
```

12:22 Courses

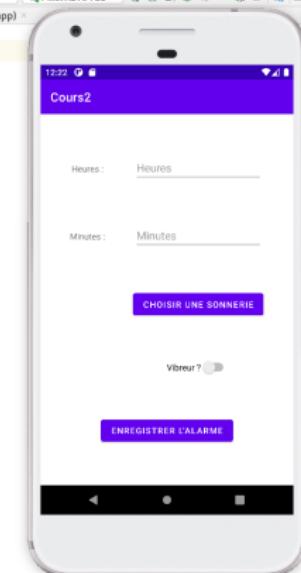
Heures :

Minutes :

Vibreur ?

CHOISIR UNE SONNERIE

ENREGISTRER L'ALARME



Device File Explorer Emulator

Event Log Layout Inspector

74:48 LF UTF-8 4 spaces

TODO Problems Terminal Logcat Build Profiler Run App Inspection

Launch succeeded (6 minutes ago)

Ringtone

Ici, **I** = **String** (MIME type) et **O** = **Uri** (fichier audio).



The screenshot shows an Android application interface. At the top, there is a navigation bar with icons for power, volume, and other controls. Below the navigation bar, the screen displays three large purple buttons with white text. From top to bottom, the buttons are labeled: "SONNERIE PAR DÉFAUT", "CHOISIR UNE SONNERIE PERSONNALISÉE", and "PAS DE SONNERIE".

The application is titled "Cours2" and the time is 12:27. The status bar also shows signal strength, battery level, and connectivity icons.

The code editor on the left shows the Java file `MainActivity.java` with the following content:

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Cours2 app src main java com.example.cours2 R.java onCreate anonymous OnClickListener onClick
strings.xml activity_main.xml MainActivity.java activity_ringtone.xml Ringtone.java AndroidManifest.xml build.gradle (Cours2) build.gradle (app)
1 package com.example.cours2;
2
3 import ...
4
5 public class MainActivity extends AppCompatActivity {
6
7     private Button buttonDefault, buttonChoose, buttonNone;
8
9     private Intent result;
10    private Uri ringtone;
11
12    ActivityResultCallback<Uri> callback = new ActivityResultCallback<Uri>() {
13        @Override
14        public void onActivityResult(Uri result) { ringtone = result; }
15    };
16
17    //ActivityResultContracts.GetContent() est le contrat par défaut String (MIME type) -> Uri
18    //on aurait aussi pu déclarer notre contrat à la main
19    private ActivityResultLauncher<String> select = registerForActivityResult(new ActivityResultContracts.GetContent(),callback);
20
21    @Override
22    protected void onCreate(Bundle savedInstanceState) {...}
23
24    private void sendRingtone(){...}
25
26}
```

The code uses `ActivityResultContracts.GetContent()` as the default contract for getting a Uri from a content provider. It also registers a launcher for this contract.

Ringtone



The screenshot shows an Android application interface. At the top, there is a navigation bar with icons for power, volume, and camera. Below the navigation bar, the title "Cours2" is displayed. The main screen contains three large purple buttons with white text:

- SONNERIE PAR DÉFAUT
- CHOISIR UNE SONNERIE PERSONNALISÉE
- PAS DE SONNERIE

The application's code is visible in the code editor on the left side of the interface. The code is written in Java and defines an activity named Ringtone. It includes methods for handling button clicks to set default ringtones or choose custom ones.

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Cours2 app src main java com.example.cours2 Ringtone onCreate anonymous OnClickListener onClick
strings.xml MainActivity.java activity_main.xml activity_ringtone.xml Ringtone.java AndroidManifest.xml build.gradle(Cours2) build.gradle(app)
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_ringtone);

    buttonDefault = findViewById(R.id.default_ringtone);
    buttonChoose = findViewById(R.id.choose_ringtone);
    buttonNo = findViewById(R.id.no_ringtone);

    result = new Intent();

    buttonDefault.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            result.putExtra(MainActivity.RINGTONE,MainActivity.DEFAULT);
            sendRingtone();
        }
    });

    buttonChoose.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            selectLauncher(Intent.ACTION_GET_CONTENT);
            result.putExtra(MainActivity.RINGTONE,MainActivity.CHOSSEN);
            result.setData(Uri.parse("ringtone"));
            sendRingtone();
        }
    });

    buttonNo.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            result.putExtra(MainActivity.RINGTONE,MainActivity.NO);
            sendRingtone();
        }
    });
}

private void sendRingtone(){
    setResult(Activity.RESULT_OK,result);
    finish();
}
```

Project Resource Manager Favorites Build Variants

Event Log Layout Inspector 35:56 LF UTF-8 4 spaces

Julien Grange Programmation mobile

setResult()

Pour passer un résultat à l'**Activity** appelante :

```
Intent result = new Intent();
result.putExtra(...);
result.setData(data);

setResult(code, result);
finish();
```

où code est `Activity.RESULT_OK` ou `Activity.RESULT_CANCELED`.

MainActivity Enregistrement de l'alarme

Cours2 – MainActivity.java [Cours2.app]

```
File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Cours2 app src main java com.example.cours2 MainActivity selectRingtone
strings.xml activity_main.xml MainActivity.java activity_ringtone.xml Ringtone.java AndroidManifest.xml build.gradle (Cours2) build.gradle (app)
Project Resource Manager
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
buttonAdd.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        int hours, minutes;
        try {
            hours = Integer.valueOf(String.valueOf(editHours.getText()));
            minutes = Integer.valueOf(String.valueOf(editMinutes.getText()));

            if(0 <= hours &<= 24 &<= minutes &< 60){
                // L'heure est valide , on lance l'alarme
                Intent setAlarm = new Intent(AlarmClock.ACTION_SET_ALARM);
                setAlarm.putExtra(AlarmClock.EXTRA_HOUR, Integer.valueOf(hours));
                setAlarm.putExtra(AlarmClock.EXTRA_MINUTES, Integer.valueOf(minutes));
                setAlarm.putExtra(AlarmClock.EXTRA_VIBRATE, switchVibrate.isChecked());
                switch (typeRingtone){
                    case DEFAULT:
                        //Si pas de EXTRA_RINGTONE, la sonnerie par défaut est utilisée
                        startActivity(setAlarm);
                        break;
                    case CHOSEN:
                        setAlarm.putExtra(AlarmClock.EXTRA_RINGTONE,ringtone);
                        startActivity(setAlarm);
                        break;
                    case NOT:
                        setAlarm.putExtra(AlarmClock.EXTRA_RINGTONE,AlarmClock.VALUE_RINGTONE_SILENT);
                        startActivity(setAlarm);
                        break;
                    default:
                        ringtoneIssue();
                }
            } else {
                formatIssue();
            }
        } catch(NumberFormatException e) {
            formatIssue();
        }
    }
});
```

12:17 Cours2

Heures : Minutes :

CHOISIR UNE SONNERIE

Vibrer ? ENREGISTRER L'ALARME

Julien Grange

Programmation mobile

Event Log Layout Inspector

74:48 LF UTF-8 4 spaces

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
Cours2 app src main java com.example.cours2 MainActivity selectRingtone
strings.xml activity_main.xml MainActivity.java activity_ringtone.xml Ringtone.java AndroidManifest.xml build.gradle (Cours2) build.gradle (app)
Project Resource Manager
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
buttonAdd.setOnClickListener(new View.OnClickListener() {
 @Override
 public void onClick(View view) {
 int hours, minutes;
 try {
 hours = Integer.valueOf(String.valueOf(editHours.getText()));
 minutes = Integer.valueOf(String.valueOf(editMinutes.getText()));

 if(0 <= hours &<= 24 &<= minutes &< 60){
 // L'heure est valide , on lance l'alarme
 Intent setAlarm = new Intent(AlarmClock.ACTION_SET_ALARM);
 setAlarm.putExtra(AlarmClock.EXTRA_HOUR, Integer.valueOf(hours));
 setAlarm.putExtra(AlarmClock.EXTRA_MINUTES, Integer.valueOf(minutes));
 setAlarm.putExtra(AlarmClock.EXTRA_VIBRATE, switchVibrate.isChecked());
 switch (typeRingtone){
 case DEFAULT:
 //Si pas de EXTRA_RINGTONE, la sonnerie par défaut est utilisée
 startActivity(setAlarm);
 break;
 case CHOSEN:
 setAlarm.putExtra(AlarmClock.EXTRA_RINGTONE,ringtone);
 startActivity(setAlarm);
 break;
 case NOT:
 setAlarm.putExtra(AlarmClock.EXTRA_RINGTONE,AlarmClock.VALUE_RINGTONE_SILENT);
 startActivity(setAlarm);
 break;
 default:
 ringtoneIssue();
 }
 } else {
 formatIssue();
 }
 } catch(NumberFormatException e) {
 formatIssue();
 }
 }
});

12:17 Cours2

Heures : Minutes :

CHOISIR UNE SONNERIE

Vibrer ? ENREGISTRER L'ALARME

Julien Grange

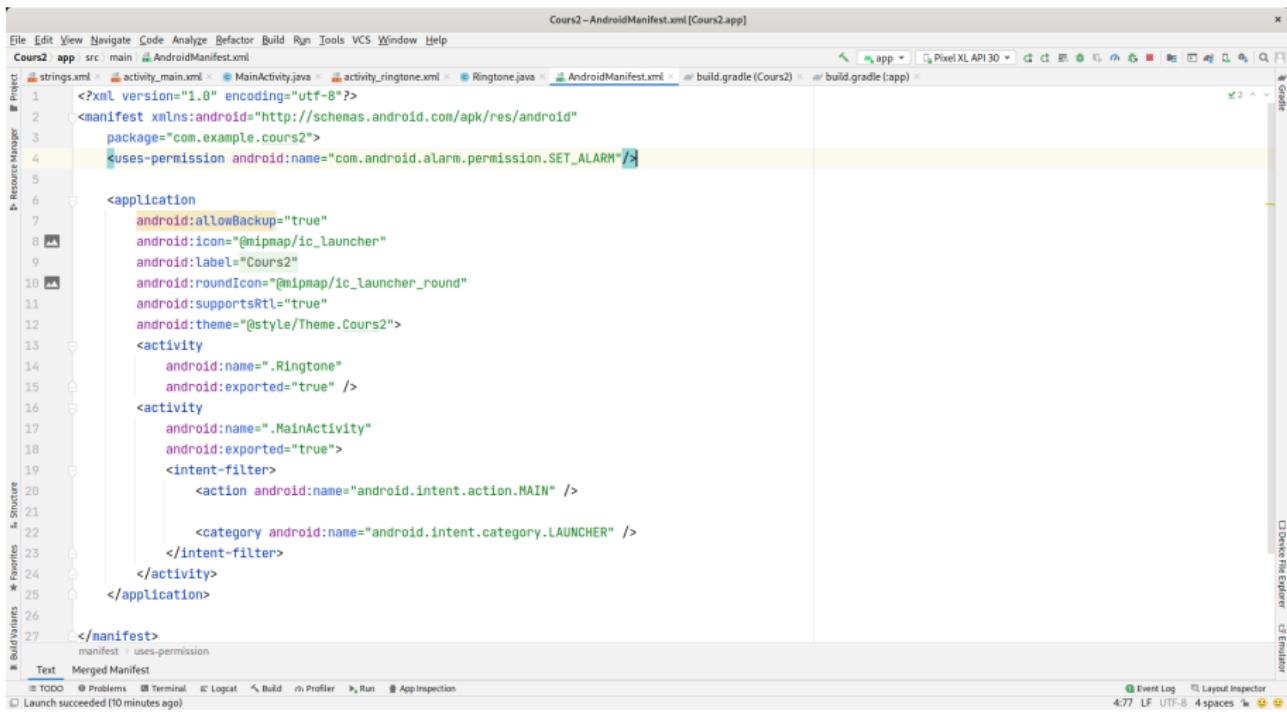
Programmation mobile

Event Log Layout Inspector

74:48 LF UTF-8 4 spaces

Permissions

On demande la permission “`com.android.alarm.permission.SET_ALARM`”.



The screenshot shows the Android Studio interface with the project "Cours2" open. The "AndroidManifest.xml" file is selected in the Project Manager. The manifest file contains the following XML code:

```
<manifest version="1.0" encoding="utf-8">
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.example.cours2">
        <uses-permission android:name="com.android.alarm.permission.SET_ALARM"/>

        <application
            android:allowBackup="true"
            android:icon="@mipmap/ic_launcher"
            android:label="Cours2"
            android:roundIcon="@mipmap/ic_launcher_round"
            android:supportsRtl="true"
            android:theme="@style/Theme.Cours2">
            <activity
                android:name=".Ringtone"
                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 line `<uses-permission android:name="com.android.alarm.permission.SET_ALARM"/>` is highlighted in yellow, indicating it has been selected or is being edited. The bottom status bar shows "Launch succeeded (10 minutes ago)".

Signaler qu'on sait traiter un Intent implicite

Pour déclarer qu'on sait envoyer des images par mail, ajouter dans le Android Manifest :

```
<activity android:name="MyMailer">
    <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="image/*"/>
    </intent-filter>
</activity>
```