

## Développement Android

### Labo 2 - Gérer plusieurs activités

<b>1</b>	<b>Layout paysage</b>	<b>2</b>
<b>2</b>	<b>Cycle de vie d'une application</b>	<b>4</b>
<b>3</b>	<b>Conserver les données avec une rotation de l'appareil</b>	<b>5</b>
3.1	Enregistrer l'état . . . . .	6
3.2	Récupérer l'état . . . . .	6
<b>4</b>	<b>Nouvelle activité</b>	<b>7</b>
4.1	Ajout d'un layout . . . . .	7
4.2	Lancer la nouvelle activité . . . . .	8
4.3	Communication de l'activité parent vers l'activité enfant . . . . .	9
4.4	Communication de l'activité enfant vers l'activité parent . . . . .	11
<b>5</b>	<b>Exercices</b>	<b>12</b>

# 1 Layout paysage

Reprenez le code du labo 1 (Quiz).

## Aide

Suivez le texte ci-dessous. Si vous êtes perdu ou si vous voulez plus d'informations, vous pouvez suivre la vidéo d'explications sur ce labo disponible ici :

<https://youtu.be/60WtxYkIigY>

Le code résultat de ce labo est disponible ici :

<https://github.com/fpluquet/ue28-android-labo2>

Si vous faites passer votre émulateur/device du mode portrait au mode paysage, vous verrez que votre application s'adapte automatiquement au mode utilisé. Mais le même layout ne convient pas forcément dans les deux modes (cela peut ne pas être joli visuellement). Nous allons donc modifier notre projet pour ajouter la possibilité d'avoir deux layouts différents en fonction du mode utilisé.

Nous allons créer un layout pour le mode portrait et un layout pour le mode paysage. Pour cela :

- Ouvrez le menu contextuel sur `res/layout`.
- Allez dans *New* → *Layout Resource File*.
- Indiquez `activity_main.xml` comme nom du fichier (le même nom que le layout existant).
- Choisissez *Orientation* et ensuite *Landscape*, et finalement *Paysage*. Remarquez que le directory de destination est alors `layout-land`.
- Cliquez sur *OK*.

Vous verrez alors que vous avez désormais deux fichiers de layout, dont un est suffixé par (`land`). Ces deux fichiers sont indépendants : vous pouvez designer deux vues très différentes en mode paysage ou portrait. Par contre, le contrôleur reste le même. Les ids doivent donc rester les mêmes pour que cela continue de fonctionner. Pour illustrer cela, complétez le nouveau fichier `activity_quiz.xml` comme suit :

```

<?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"
    android:gravity="center">

    <TextView
        android:id="@+id/question_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:padding="24dp"/>

    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical|center_horizontal"
        android:orientation="horizontal">

        <Button
            android:id="@+id/true_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/true_button"
            android:layout_marginRight="30dp"/>

        <Button
            android:id="@+id/false_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/false_button" />

    </LinearLayout>
    <Button
        android:id="@+id/next_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|right"
        android:text="@string/next_button"
        android:drawablePadding="4dp"/>

</LinearLayout>

```

Testez alors l'application pour voir si le jeu fonctionne bien en mode portrait et paysage avec 2 layouts différents.

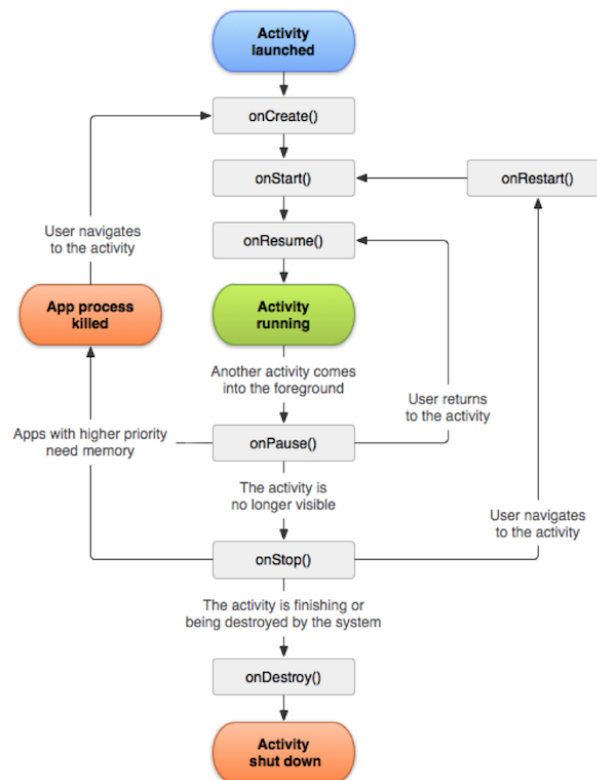
Vous pouvez remarquer que si vous êtes après la question 1 et que vous changez d'orientation, le jeu recommence au début. Pour comprendre pourquoi cela fonctionne comme cela, il faut bien comprendre le cycle de vie d'une application.

## 2 Cycle de vie d'une application

Une application Android suit un cycle de vie bien défini. En d'autres mots, l'application va passer d'état en état en fonction de différentes interactions :

- l'application est lancée
- l'utilisateur quitte l'application
- l'utilisateur change (*switch*) d'application
- l'utilisateur revient à l'application
- ...

Les états sont repris dans la figure suivante :



La classe `Activity` fournit des *callbacks* pour chacune des transitions entre les états de l'application. Les méthodes sont : `onCreate`, `onDestroy`, `onStart`, `onStop`, `onResume`, `onPause`, ...

Dans `MainActivity`, rajoutez des logs pour chacune des 6 méthodes de callback. Par exemple, redéfinissez la méthode `onStart` comme ceci :

```
@Override
protected void onStart() {
    super.onStart();
    Log.d(TAG, "onStart method");
}

@Override
```

```

protected void onStop() {
    super.onStop();
    Log.d(TAG, "onStop method");
}

@Override
protected void onDestroy() {
    super.onDestroy();
    Log.d(TAG, "onDestroy method");
}

@Override
protected void onPause() {
    super.onPause();
    Log.d(TAG, "onPause method");
}

@Override
protected void onResume() {
    super.onResume();
    Log.d(TAG, "onResume method");
}

@Override
protected void onRestart() {
    super.onRestart();
    Log.d(TAG, "onRestart method");
}

```

...en important correctement la classe `Log` et en définissant `TAG` comme une chaîne de caractères constante, par exemple `"QUIZ_APP"`. C'est alors plus facile pour filtrer les messages importants dans la masse produite par notre application dans le log !

### Question 1

Quelles méthodes de callback sont appelées lorsque :

1. vous lancez l'application ?
2. vous quittez l'application (bouton *Back* après l'avoir lancée) ?
3. vous changez d'application (avec le task manager) ?
4. vous revenez à l'application (avec le task manager) ?
5. vous effectuez une rotation avec votre appareil ?

## 3 Conserver les données avec une rotation de l'appareil

Si vous avez répondu aux questions précédentes, vous avez remarqué que la méthode `onCreate` est appelée lorsqu'une rotation est effectuée. En fait, l'activité en cours avant la rotation est tuée et remplacée par une nouvelle dans le nouveau mode. Les variables sont donc oubliées et l'état est réinitialisé.

Pour régler ce problème, nous allons utiliser les méthodes de callback. Nous allons enregistrer l'état lorsque l'activité est détruite et charger l'état enregistré lorsque l'activité est relancée.

### 3.1 Enregistrer l'état

Pour enregistrer l'état une méthode spéciale est prévue : `onSaveInstanceState`. Cette méthode est appelée automatiquement (avant la méthode `onStop`) lorsqu'il est nécessaire d'enregistrer l'état (lors d'un changement d'activité par exemple). Grâce à cette méthode nous pouvons enregistrer l'état dans un *Bundle* (un ensemble de clefs/valeurs).

Par exemple, pour enregistrer l'index de la question en cours, nous pouvons faire :

```
@Override
protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    Log.i(TAG, "onSaveInstanceState");
    outState.putInt(KEY_INDEX, mCurrentIndex);
}
```

...où `KEY_INDEX` est une constante de type `String`. Par exemple, `KEY_INDEX="index"` est définie dans la classe et servira à retrouver l'index.

### 3.2 Récupérer l'état

Le *Bundle* utilisé lors du `onSaveInstanceState` est passé à la méthode `onCreate` et peut donc servir à réinitialiser l'activité avec l'état ainsi sauvegardé.

Par exemple, pour récupérer la question en cours, nous pourrions ajouter à la méthode `onCreate` le code :

```
protected void onCreate(Bundle savedInstanceState) {
    // (...)
    if(savedInstanceState != null) { // null si on vient de lancer
        l'application
        mCurrentIndex = savedInstanceState.getInt(KEY_INDEX);
    }
    // (...)
}
```

Implémentez cette gestion de l'état dans votre application afin de gérer les rotations.

#### Question 2

Quels sont les types des éléments que l'on peut enregistrer dans un *Bundle*?  
Les types primitifs? Les objets? Tous?

## 4 Nouvelle activité

Dans cette section, vous allez ajouter une seconde activité à votre application. Cette activité permettra de connaître la réponse à la question (via une nouvelle fenêtre). Cette fenêtre sera composée de :

- un texte : "Êtes-vous sûr de vouloir tricher ?"
- un bouton : "Oui :D"
- un texte pour accueillir la réponse, invisible au départ.



### 4.1 Ajout d'un layout

Ajoutez une nouvelle activité : *New* → *Activity* → *Empty Activity*. Appelez cette activité `CheatActivity`. Comme auparavant, cela crée le contrôleur `CheatActivity.java` et son layout `activity_cheat.xml`.

Si ce n'est pas déjà fait, rangez le contrôleur de cette activité (`CheatActivity.java`) dans le package des contrôleurs.

Modifiez ensuite le layout comme ceci (et ajoutez les strings nécessaires dans le fichier `strings.xml`) :

```

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

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"
        android:text="@string/warning_text"/>

    <TextView
        android:id="@+id/answer_text_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="24dp"
        tools:text="Answer"/>

    <Button
        android:id="@+id/show_answer_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/show_answer_button"/>

</LinearLayout>

```

Créez les `@strings`<sup>1</sup> comme il se doit pour que cela corresponde au screenshot précédent.

## 4.2 Lancer la nouvelle activité

Ajoutez un nouveau bouton "Tricher..." dans votre `MainActivity` (pensez à l'ajouter aux deux layouts de la `MainActivity`).

Nous allons désormais lancer la nouvelle activité lorsque ce bouton est pressé. Pour cela, modifiez le contrôleur du bouton pour y ajouter un listener qui lance l'activité comme suit :

```

Intent intent = new Intent(MainActivity.this, CheatActivity.class);
startActivityForResult(intent, REQUEST_CODE);

```

...où `REQUEST_CODE` est une constante entière à définir (0, 1, 1234 ou ce que vous voulez).

La méthode `startActivityForResult` permet d'appeler l'`ActivityManager` de votre appareil, qui se chargera de lancer la nouvelle activité. Le premier paramètre (`MainActivity.this`) permet d'indiquer à partir de quel contexte on démarre la nouvelle activité. Le second paramètre (`CheatActivity.class`) permet de donner la classe à instancier pour la nouvelle activité. Si l'on avait indiqué `MainActivity.class` comme second paramètre, une nouvelle instance de la fenêtre de Quiz s'ouvrirait (vous pouvez le tester ;-)).

1. Voir laboratoire 1 si vous ne vous en souvenez pas.



Vérifiez que votre bouton est fonctionnel et démarre la nouvelle activité. Si vous cliquez sur le bouton *Back* de l'appareil, la première activité réapparaît.

Il y a deux méthodes qui permettent de lancer une autre activité :

- `startActivityForResult` qui lance une activité dans l'espoir d'une réponse (voir plus loin).
- `startActivity` qui lance une activité et qui n'a pas besoin de réponse.

Les activités lancées sont mises sur une pile. Si bien que, si une activité est arrêtée (via le bouton *Back* ou via la méthode `finish()`), l'activité est *poppée* de la pile et la précédente est affichée. Si c'était la dernière sur la pile, l'application est alors arrêtée.

### 4.3 Communication de l'activité parent vers l'activité enfant

Nous devons donner la bonne réponse à notre `CheatActivity`. De même, lorsque la `CheatActivity` se termine, elle doit informer la `MainActivity` si le joueur a triché ou non.

Tout d'abord, ajoutez un *extra* à l'`Intent` permettant de démarrer l'activité (dans le listener sur le bouton "Tricher...") :

```
Intent intent = new Intent(MainActivity.this, CheatActivity.class);
intent.putExtra(CheatActivity.ANSWER_EXTRA,
mQuestionBank[mCurrentIndex].isAnswerTrue());
startActivityForResult(intent, REQUEST_CODE);
```

Demandez à Android Studio de créer la constante `ANSWER_EXTRA` dans la classe `CheatActivity` (la valeur peut être le string "ANSWER\_EXTRA" ou un autre identifiant unique).

La méthode `putExtra` permet d'ajouter des paires clef/valeur qui sont passées à la nouvelle activité créée.

Il suffit maintenant de récupérer cet *extra* lors du démarrage de l'activité. Pour cela, dans la méthode `onCreate` de la classe `CheatActivity`, on ajoute :

```
mAnswerIsTrue = getIntent().getBooleanExtra(ANSWER_EXTRA, false);
```

...où `mAnswerIsTrue` est un attribut booléen de la classe qui stocke la bonne réponse. La méthode `getIntent` de la classe `Activity` retourne toujours l'`intent` qui a créé l'activité.

Le code de votre classe devrait être celui-ci :

```
public class CheatActivity extends AppCompatActivity {

    public static final String ANSWER_EXTRA = "ANSWER_EXTRA";
    private boolean mAnswerIsTrue;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cheat);
        mAnswerIsTrue =
            getIntent().getBooleanExtra(ANSWER_EXTRA, false);
    }
}
```

Vous pouvez maintenant coder l'activation du bouton montrant la bonne réponse : par défaut, la bonne réponse n'est pas affichée, il faut cliquer sur le bouton "Oui :)" pour que la réponse s'affiche et la triche soit effective.

Essayez de coder cela correctement puis passez à la page suivante contenant la solution.

Votre code devrait alors ressembler à cela :

```
import androidx.appcompat.app.AppCompatActivity;

import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class CheatActivity extends AppCompatActivity {

    public static final String ANSWER_EXTRA = "ANSWER_EXTRA";
    private boolean mAnswerIsTrue;

    private Button mCheatButton;
    private TextView mAnswerTextView;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cheat);
        mAnswerIsTrue = getIntent().getBooleanExtra(ANSWER_EXTRA, false);

        mAnswerTextView = findViewById(R.id.answer_text_view);
        mCheatButton = findViewById(R.id.show_answer_button);

        mCheatButton.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (mAnswerIsTrue)
                    mAnswerTextView.setText(R.string.true_button);
                else
                    mAnswerTextView.setText(R.string.false_button);
            }
        });
    }
}
```

Testez votre application pour vérifier que cela fonctionne bien et que vous pouvez tricher :)

#### 4.4 Communication de l'activité enfant vers l'activité parent

Il se pourrait que le joueur change d'idée, ou pas. Nous allons signaler à l'activité `MainActivity` si le joueur a réellement triché (il a cliqué sur le bouton "Oui :)") ou non.

Il faut donc communiquer dans l'autre sens : de la `CheatActivity` à la `MainActivity`. Pour cela, un autre mécanisme est mis à disposition toujours à l'aide de *callbacks*.

Tout d'abord lorsque le bouton `show_answer` est cliqué, et donc que le joueur a réellement triché, nous complétons le résultat (de l'activité) avec cette information :

```
Intent data = new Intent();
data.putExtra(EXTRA_ANSWER_SHOWN, true);
this.setResult(RESULT_OK, data);
```

- On crée un nouvel intent qui sera communiqué à l'activité parent (**MainActivity**, qui a créé cette activité).
- On y met l'information booléenne : le joueur a triché. La constante **EXTRA\_ANSWER\_SHOWN** est une constante de type **String** que vous créez dans la classe **CheatActivity** et qui est la clef de la pair clef/valeur. Cette clef sera utilisée par la **MainActivity** pour récupérer cette information.
- On appelle la méthode **setResult** de la classe **CheatActivity** qui sert à transmettre les résultats vers l'activité appelante. La constante **RESULT\_OK** est une constante prédéfinie de la classe **Activity**, dont hérite la classe **CheatActivity**.

Lorsque l'activité **CheatActivity** sera terminée, c'est-à-dire lorsque le bouton *Back* est cliqué, l'appareil retournera à l'activité précédente (**MainActivity**). Pour récupérer la réponse, il suffit d'implémenter (redéfinir) la méthode de callback **onActivityResult** dans **MainActivity.java** :

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (resultCode == CheatActivity.RESULT_OK) {
        if (data.getBooleanExtra(CheatActivity.EXTRA_ANSWER_SHOWN, false)) {
            Toast.makeText(
                this, getString(R.string.answer_has_been_shown),
                Toast.LENGTH_SHORT).show();
        }
    }
}
```

Ce retour à l'activité affiche un message (Toast) si le joueur a demandé de tricher.

Remarquez que vous devez définir **R.string.answer\_has\_been\_shown** dans la ressource des strings avec la valeur : "La réponse a été montrée. Tricheur !". Comme **R.string.answer\_has\_been\_shown** n'est qu'un identifiant entier, si on l'affiche tel quel, nous verrons s'afficher un entier. En utilisant la méthode **getString(int)**, on récupère le string des ressources lié à l'identifiant passé en paramètre.

### Question 3

Où est définie la méthode **getString(int)** ?

## 5 Exercices

Terminez votre application avec les fonctionnalités suivantes :

- Gérez la rotation de l'activité **CheatActivity**.
- Ajoutez au score (exercice Labo 1) le nombre de tricheries (gérez cela aussi lors de rotation).
- Faites en sorte que le joueur ne puisse donner qu'une seule réponse (désactivez le bouton). Attention aux rotations !
- Faites en sorte qu'on ne puisse passer à la question suivante qu'après avoir répondu à la question.
- Le Quiz se termine lorsqu'on arrive à la dernière question. Le score final est affiché et on peut décider de recommencer, le score est alors réinitialisé.
- Ajoutez une activité permettant de choisir parmi un ensemble de quiz : 'géographie', 'histoire', 'sport', etc. Une fois le choix effectué, le quiz sélectionné démarre.