HelHa
Tournai
Bachelor en Informatique

2019 - 2020
Android

Développement Android

Gérer plusieurs activités

Version du document : 17 septembre 2019

1 Layout paysage

Afin de gérer les vues lors de la rotation de votre application de Quiz, nous allons créer un layout pour le mode portrait et un layout pour le mode paysage.

- 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".

Vous verrez alors que deux fichiers de layout ont été créés, 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
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">


        <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" />

            <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"/>

</FrameLayout>
```

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.

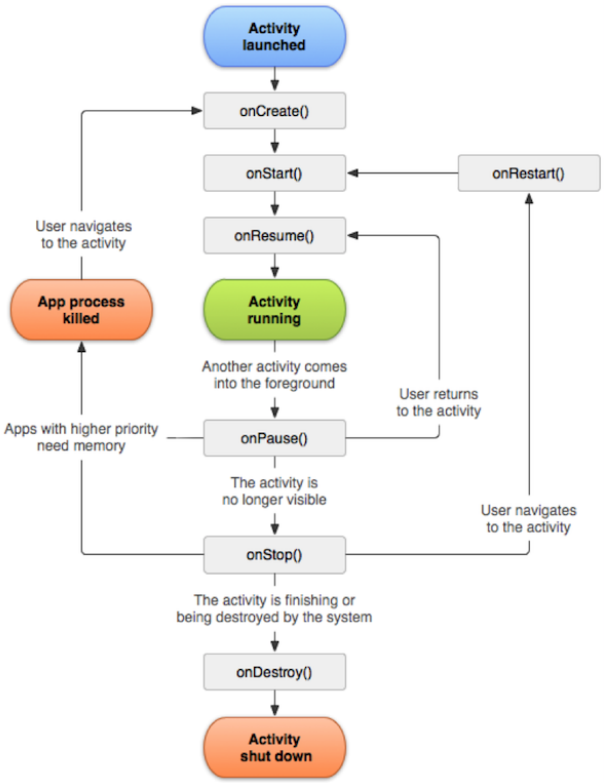
2 Cycle de vie d'une application

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
— etc

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");
}
```

...en important correctement la classe `Log` et en définissant `TAG` comme une chaîne de caractères constante par exemple.

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

Nous venons de voir que les activités Android sont relancées lorsqu'on effectue une rotation de l'appareil.

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

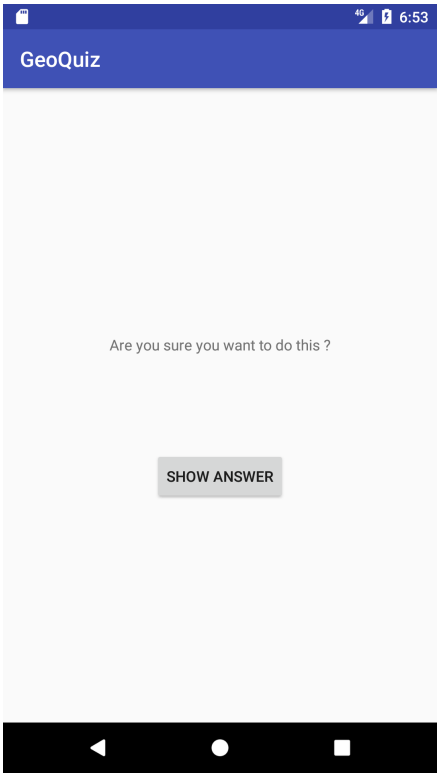
4 Nouvelle activité

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 est composée de :

— un texte : "Are you sure you wanna do this ?"

— un bouton : "Show answer"

— 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`.

Modifiez 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: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:orientation="vertical"
    android:gravity="center"
    tools:context="votrePackage.CheatActivity">

    <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>
```

4.2 Lancer la nouvelle activité

Vous ajoutez un nouveau bouton "cheat" dans votre `MainActivity`. Nous allons lancer la nouvelle activité lorsque ce bouton est pressé. Pour cela, modifiez le contrôleur du bouton et lancez 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.

La méthode `startActivityForResult` permet d'appeler l'ActivityManager de votre appareil, qui se chargera de lancer la nouvelle activité.

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.

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 :

```
1 intent.putExtra(CheatActivity.ANSWER_EXTRA,
2                 mQuestionBank[mCurrentIndex].isAnswerTrue());
```

Demandez à Android Studio de créer la constante `ANSWER_EXTRA` dans la classe `CheatActivity`.

La méthode `putExtra` permet d'ajouter des pairs 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 :

```
1 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.

Vous pouvez maintenant activer le bouton montrant la bonne réponse.

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 triché 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 :

```
1     Intent data = new Intent();
2     data.putExtra(EXTRA_ANSWER_SHOWN, true);
3     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 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 `Activity` qui sert à transmettre les résultats vers l'activité appelante. La constante `RESULT_OK` est une constante prédéfinie de la classe `Activity`.

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,
                R.string.answer_is + " " + getCurrentQuestion().isAnswerTrue(),
                Toast.LENGTH_SHORT
            ).show();
        }
    }
}
```

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

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 (gérer 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 réiniatilisé.

BONUS : Ajoutez une activité permettant de choisir parmi un ensemble de Quiz : 'géographie', 'histoire', 'sport', etc. Une fois le choix effectué, le Quiz sélectionné démarre.