Berney Alec & Forestier Quentin & Herzig Melvyn

Poo2 – 05.10.2021

Introduction à Android

Labo 1



Table des matières

[Introduction 2](#_Toc83295850)

[Langue de l’interface 2](#_Toc83295851)

[Champs textuels de saisie 3](#_Toc83295852)

[Mode paysage 4](#_Toc83295853)

[Sources 4](#_Toc83295854)

# Introduction

Ce laboratoire est une initiation au développement mobile qui nous amène à effectuer plusieurs manipulations afin de nous familiariser avec le développement Android.

Ce document répond aux questions posées dans la donnée.

# Langue de l’interface

Cette étape consiste à mettre en place la prise en charge du français. Actuellement l’application est en anglais.

Les applications Android peuvent demander un certain nombre de ressources. Ces dernières se trouvent dans le dossier **res/**. Les ressources, en fonction de leur type, sont stockées dans des sous-dossiers différents. Les chaînes de caractères sont généralement stockées dans le sous-dossier **values/**.

Il est possible d’ajouter des suffixes aux sous-dossiers de **res/** afin d’utiliser différentes ressources en fonctions des caractéristiques de l’appareil ou de sa localisation.

Ainsi nous pouvons créer un dossier ressources Android pour notre traduction française **values-fr/**.

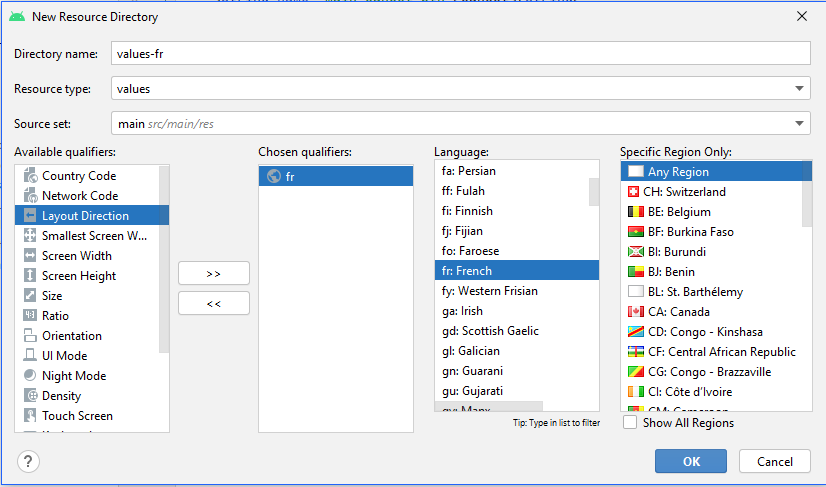


Figure Création du dossier de ressources pour la traduction française

A ce point, nous comprenons que si nous souhaitons créer une applications multi-langues, nous aurons un dossier *values* par langue supportée.

De plus, il est possible de créer des spécifications par région en fonction des langues. Si par exemple, nous souhaitons distinguer le français suisse et le français canadien, nous aurions eu l’arborescence suivante :

|  |  |
| --- | --- |
| res/ | |
| values-fr-CH/ | values-fr-CA/ |
| strings.xml | strings.xml |

Dans notre cas, nous avons créé un unique dossier *values* et un fichier *strings* pour ne pas distinguer le français en fonction des régions.

Si une langue demandée est non traduite, la langue affichée sera la langue par défaut, c’est-à-dire celle qui se trouve dans le dossier *values* sans spécification de langue **values/**. Ainsi en mettant l’anglais dans **res/values/string.xml** nous faisons en sorte que la langue par défaut soit l’anglais.

**Pourquoi regrouper les chaînes de caractères dans un fichier XML indépendamment des layouts ?**

Paracerque c’est une bonne pratique.

Cette méthode permet de maintenir indépendamment les ressources et le code. Pour une application plus conséquente avec la prise en charge de toutes les langues, cela permettrait aux traducteurs et aux développeur de travailler parallèlement sans empiéter les uns sur les autres.

De plus, cette pratique est native à l’environnement de développement. Il a suffi de nommer les répertoires et les fichiers pour que la compilation du programme gère la logique du choix des langues. Si nous avions voulu mélanger code et traduction, nous aurions du nous charger de la détection de la langue de l’appareil, de choisir le bon tableau de string et remplir manuellement les layouts.

**Que se passe-t-il si une traduction est manquante dans la langue par défaut ou dans une langue additionnelle ?**

Quelle que soit la langue sélectionnée, si la traduction manquante est dans la langue additionnelle, la traduction est remplacée par la traduction de la langue par défaut. Si la traduction manquante est dans la langue par défaut, le programme de compile pas, même si celle de la langue additionnelle (demandée ou non) est présente.

# Champs textuels de saisie

Les champ textuel de l’email et du mot de passe sont de type *EditText* sans spécification du type. Il est possible de spécialiser ces deux champs en utilisant l’attribut *InputType.*

|  |  |
| --- | --- |
| Champ | InputType |
| Email | textEmailAddress |
| Mot de passe | textPassword |

Il est possible de le faire directement dans le code



Sinon la même action peut être réalisée dans l’éditeur de layout



# Mode paysage

Comme les fichiers de traduction, les layouts sont des ressources. Ainsi, pour changer de layout en fonction de l’orientation du téléphone, nous allons spécialiser le dossier layout pour l’orientation paysage.

Le dossier **res/layout-land/** doit être créé.

Une fois encore, comme pour la traduction, nous y plaçons une copie du fichier par défaut dans **res/layout/**.

Finalement nous éditons ce nouveau fichier pour obtenir l’affichage désiré.

Arborescence résultante :

|  |  |
| --- | --- |
| res/ | |
| layout/ | layout-land/ |
| activity\_main.xml | activity\_main.xml |

Le layout a été réorganisé comme suit :

|  |  |
| --- | --- |
| Vertical | Horizontal |
| Une image contenant texte  Description générée automatiquement |  |

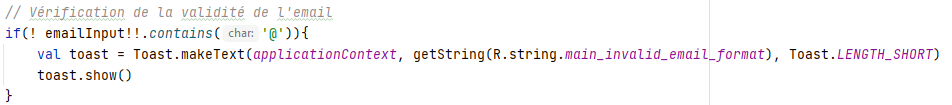
Le passage du *ConstraintLayout* en *LinearLayout* nous permet d’afficher l’image à côté du formulaire. Pour une meilleure lisibilité nous centrons le *main\_logo* et le *relativeLayout* avec l’attribut *layout\_gravity* -> *center*.

Nous agrandissons l’image avec les attribut *layout\_width* et *layout\_height* à 200.

Finalement, nous plaçons le lien *main\_new\_account* dans le *relativeLayout* et nous lui ordonnons de se placer sous le l*inearLayout*, identifié par *main\_buttons*, grâce à la clause *layout\_below*.

# Vérification du format de l’email

Dans cette partie, nous vérifions si l’email est correct. Pour des raison de simplicité, un email est correct s’il contient un ‘@’.



À la suite du *ClickListener* du bouton valider, si le contenu du champ de l’email n’est pas null, nous vérifions s’il contient un ‘@’.

Si ce n’est pas le cas, nous affichons le toast avec le message d’erreur *main\_invalid\_email\_format* que nous avons pris soin de créer pour l’anglais et le français.

# Vérification du couple email / mot de passe

Nous créons une paire *emailInput* et *passwordInput* puis, nous vérifions son existence dans la liste *credentials*.



Si le couple n’est pas trouvé, nous affichons une fenêtre de dialogue qui indique l’erreur et vide les champs.

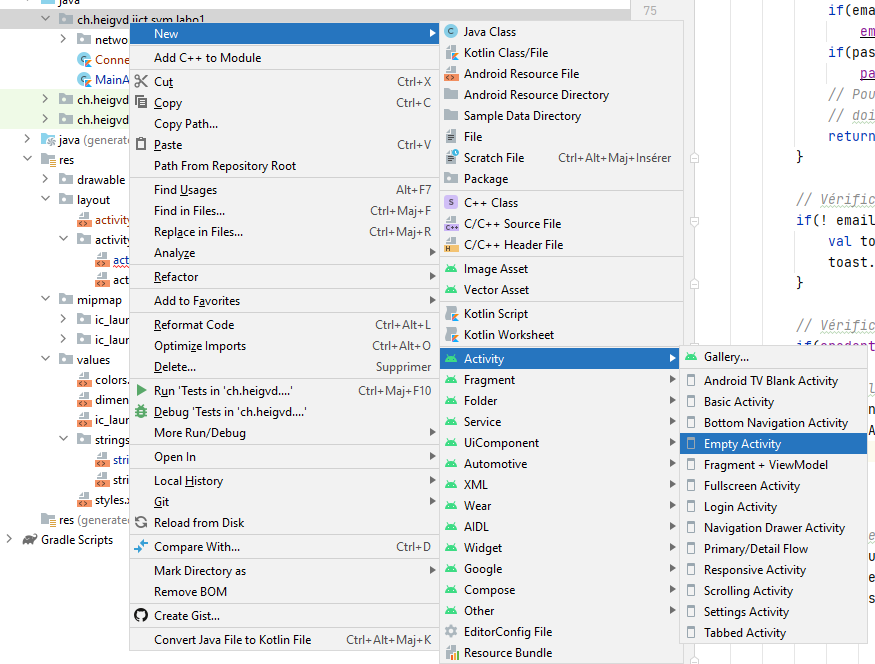
Une image contenant texte

Description générée automatiquement

Une fois encore l’erreur *main\_error\_email\_pass* a été traduite dans les deux langues. Toutefois, la chaîne *main\_ok\_button* ( de valeur « ok ») a été traduite uniquement en anglais.

# Création et lancement de la nouvelle activité

Nous avons créé une nouvelle activité *connectedActivity* de type *Empty Activity* avec l’utilitaire AndroidStudio.



Finalement, pour démarrer notre nouvelle activité, nous avons créé une intention que nous démarrons aussitôt.



# Passage de paramètres à la nouvelle activité

Pour passer des paramètres à la nouvelle activité, nous avons ajouté un extra à l’intention précédemment créée.



Désormais, nous transmettons la valeur de l’email avec la clé *EXTRA\_EMAIL* ( email.MESSAGE ) qui est une constante globale.

Concernant le layout de la nouvelle activité, nous avons ajouté deux *TextView*:

* *connected\_greetings\_title* : qui affiche « Bonjour » ou « Good morning »
* *connected\_email\_greetings* : qui affiche l’adresse de l’utilisateur.

Finalement, le code de l’activité *ConnectedActivity* se présente comme suit :

Une image contenant texte

Description générée automatiquement

Nous récupérons une référence sur la *TextView* qui affichera l’email. Ensuite, nous récupérons l’extra de l’intention que nous affichons dans la *TextView* précédemment récupérée.

# Permissions simples

Nous avons autorisé l’accès à internet en plaçant ces deux lignes, en dehors des balises application, dans notre manifeste.

<uses-permission android:name="android.permission.INTERNET" />  
<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE" />

Ensuite, nous avons modifié notre activité *ConnectedActivity* en y ajoutant une *ImageView* du nom de *ConnectedLogo*.

Le code de l’*ImageView* est le suivant :

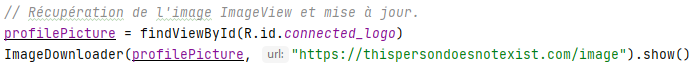
<ImageView  
 android:id="@+id/connected\_logo"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 app:layout\_constraintBottom\_toTopOf="@+id/relativeLayout"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent"  
 android:contentDescription="@string/connected\_profile\_picture\_desc" />

Nous ajoutons une image vide à laquelle nous spécifions un id, le comportement pour sa taille, les contraintes de position ainsi qu’une description.

Le nouveau design de l’activité se présente comme suit :

|  |  |
| --- | --- |
| Arborescence des composants | Aperçu |
|  |  |

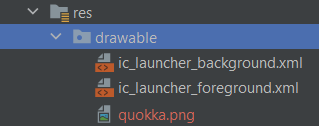
Finalement, dans le code de l’activité nous récupérons une référence sur l’élément et nous appliquons l’image à l’aide du code fourni :



# Affichage d’une activité

**Dans quel(s) dossier(s) devons-nous ajouter cette image ?**

Il suffit d’ajouter à la main l’image dans le dossier « drawable », comme ceci :



Et on retrouve ces dossiers à l’aide du chemin suivant dans l’explorateur Windows du projet :



Voici le contenu du dossier :

Une image contenant texte

Description générée automatiquement

**Veuillez décrire brièvement la logique derrière la gestion des ressources de type « image matricielle » sur Android.**

Toutes les images de type « matricielle » (png ou jpeg) ne subissent aucun traitement et sont directement stockés dans le dossier « drawable ».

**Quel intérêt voyez-vous donc à utiliser une image vectorielle ?**

L’image vectorielle obtient différentes formes que l’on peut utiliser comme on le souhaite au sein de l’application. Une image vectorielle ne perd aucune qualité lorsque cette dernière est redimensionnée et est donc plus responsive.

**Est-ce possible pour tout type d’images (logos, icônes, photos, etc.) ?**

Les images vectorielles sont uniquement utilisables pour les images pouvant être dimensionnées à l’aide de « formules » mathématiques. On parle donc ici de logos et d’icônes. Des photos prises à l’aide d’un appareil ne pourront pas être vectorielle car elles sont basées sur des pixels.

# Cycle de vie

**Décrivez brièvement à quelles occasions ces méthodes sont invoquées. Vous expliquerez aussi l’enchainement de ces appels lorsque l’on passe d’une activité à une autre.**

Voici un schéma résumant bien les actions réalisées sur les activités et plus particulièrement leurs changements de contexte :



Figure 2 Diagramme d’états d’une activité

Voici un tableau récapitulant quelle méthode est invoquée et à quel moment.

|  |  |
| --- | --- |
| Méthode | Invocation |
| onCreate | Lors de la création de l’activité, quand elle est lancée |
| onStart | Lors du démarrage de l’activité, quand elle est lancée |
| onResume | Lors du lancement de l’activité, quand elle est lancée |
| onPause | Lors de la mise en pause de l’activité, quand elle est en cours d’exécution |
| onStop | Lors de l’arrêt de l’activité, quand elle est en cours d’exécution |
| onDestroy |  |
| onRestart |  |

**Enchainement des appels lorsque que l’on passe d’une activité à une autre :**

La première activité s’arrête et rentre en dans l’état Paused ou Stopped. Pendant ce temps l’autre activité est créée. Si ces activités partagent des données, la première activité n’est pas complétement stoppée avant que la seconde soit créée afin d’assurer la bonne transmission des données.

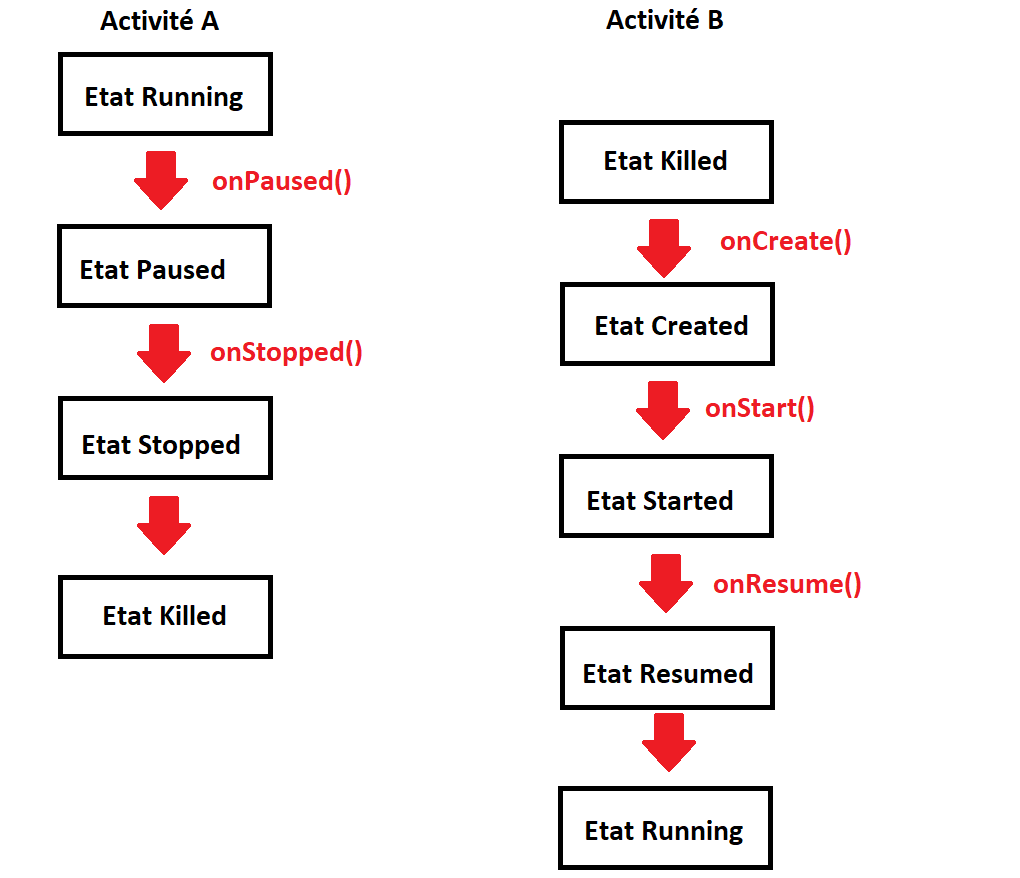
Voici le cycle de vie de 2 activités (A et B) s’exécutant dans la même application :

Activity A : la méthode onPause() est exécutée.

Activity B : la méthode onCreate(), puis la méthode onStart(), et finalement la méthode onResume() s’exécutent à la suite.

Ensuite, si l’activité A n’est plus utilisée au niveau visuel, sa méthode onStop() est exécutée.

Voici un schéma avec les états dans lequel les activités passent :



# Sources

Android Developers, *App resources Overview – 14/07/2021,* [*https://developer.android.com/guide/topics/resources/providing-resources#ResourceTypes*](https://developer.android.com/guide/topics/resources/providing-resources#ResourceTypes)*, 22/09/2021*

Android Developers, *Support different languages and cultures – 17/09/2021*, <https://developer.android.com/training/basics/supporting-devices/languages>, 22/09/2021

Android Developers, *InputType – 11/08/2021*, <https://developer.android.com/reference/android/widget/TextView#attr_android:inputType>, 23/09/2021

Android Developers, *Dialogs – 30/06/2021*, <https://developer.android.com/guide/topics/ui/dialogs>, 24/09/2021

Android Developers, *Start another activity – 13/07/2021*, <https://developer.android.com/training/basics/firstapp/starting-activity>, 24/09/2021

Android Developers, *Connect to the network – 25/09/2021,* <https://developer.android.com/training/basics/network-ops/connecting>, 25/09/2021

Android Developers, *The Activity Lifecycle– 10/04/2021,*

<https://developer.android.com/guide/components/activities/activity-lifecycle>, *10/04/2021*