Berney Alec & Forestier Quentin & Herzig Melvyn

Poo2 – 23.09.2021

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/**.

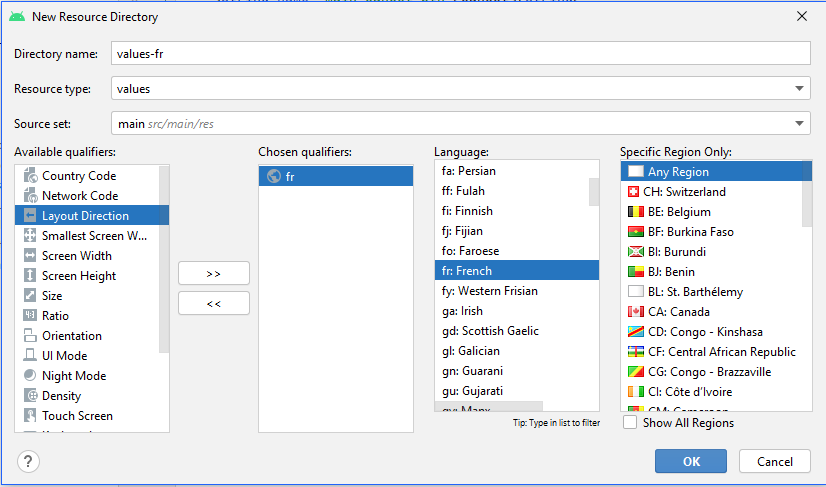


Figure 1 Création du dossier de ressources pour la traduction française

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*.

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

# 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