Notre objectif : paramétrer l'application pour qu'elle fonctionne sur n'importe quel support

1. Les différents types de fichiers « *ressources* »

Une application Android va être utilisée sur une grande variété de supports. Il faut donc une solution :

- pour permettre à une application de s'afficher de la même manière sur un téléphone Smartphone ou sur une tablette avec un écran 10",
- pour faire en sorte que les textes s'adaptent à la langue de l'utilisateur,
- pour que les images soient adaptées au format du support d'exécution.

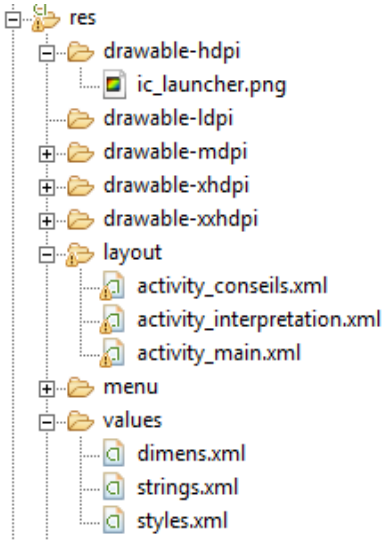
Les ***ressources*** sont des éléments capitaux dans une application Android. On y trouve entre autre :

- des déclaratives de substitution de chaînes de caractères, permettant d'adapter l'application à la langue déclarée sur le support mobile,
- des images dans différents formats permettant de s'adapter à la taille du support.

→ Les différents éléments sont organisés de manière très précise, de façon à ce qu'Android sache lesquels utiliser pour quels types de terminaux.

Les ressources sont stockées selon une hiérarchie organisée en plusieurs types.

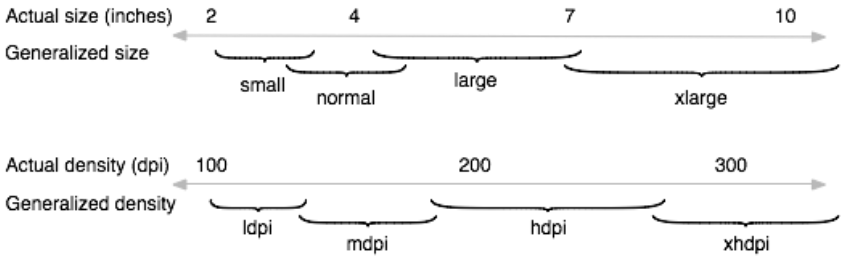
Pour permettre à Android de les retrouver facilement, chaque type de ressources est associé à un répertoire particulier :



res	C'est le dossier qui contient toutes les ressources de l'application (images, chaines, vidéos, styles)
drawable-xxx	Contient les images dans les différentes résolutions (basse, moyenne, haute et très haute). On y trouve les images matricielles (les images de type PNG, JPEG ou encore GIF) ainsi que des fichiers XML qui permettent de décrire des dessins simples (par exemple des cercles ou des carrés)
ic_launcher.png	L'icône de l'application
layout-xxx	C'est dans ce dossier que l'on crée l'ensemble des fichiers XML décrivant les interfaces de l'application. On peut/doit créer autant de dossiers layout, typés, que de types de supports sur lesquels s'exécutera l'application.
menu	Les fichiers XML pour pouvoir constituer des menus
values	Ce dossier contient un ensemble de fichiers décrivant les valeurs (pseudo variables) utilisées par l'application. On peut, par exemple, y mettre des chaines de caractères (strings.xml), des tableaux (arrays.xml), des entiers, des couleurs, … Vous peut/doit créer différents dossiers en fonction de l'orientation, de la taille de l'écran (values-large) et de la version d'Android.
strings.xml	Fichier qui contient les déclarations de chaines de caractères
dimens.xml	Contient les dimensions utilisées dans L'application. Ces dimensions seront surchargées pour les écrans larges (values-large) afin de prendre en compte tout l'espace disponible
styles.xml	Représente le thème par défaut utilisé par votre application.

Dans ce support on s'intéressera à : **drawable-xxx, layout-xxx, strings.xml**

2. Taille du support mobile et résolution d'écran



- ldpi : 120 dpi ;
- mdpi : 160 dpi ;
- hdpi : 240 dpi ;
- xhdpi : 320 dpi (disponible à partir de l'API 8 uniquement) ;
- nodpi : pour ne pas redimensionner les images matricielles (vous savez, JPEG, PNG et GIF !).

Résolution

dpi *: dots per inch (points par pouce) : http://fr.wikipedia.org/wiki/Point_par_pouce*

Si l'écran du terminal de l'utilisateur a une **grande résolution** ? Android ira chercher dans **res/drawable-hdpi** !
Si L'écran du terminal de l'utilisateur a une **petite résolution** ? Il ira chercher dans **res/drawable-ldpi/** !
Si l'écran du terminal de l'utilisateur a une **très grande résolution** ? ... dans **res/drawable-hdpi**

Taille des images conseillées

- *mdpi* (48×48), *hdpi* (72×72),
- *xhdpi* (96×96),
- *xxhdpi* (144×144),
- xxxhdpi(192x192).

Exemples et règles à suivre

- res/drawable-hdpi pour les petits écrans
- res/drawable-ldpi
- res/drawable-mdpi avec des images spécifiquement pour les grands écrans.
- res/layout-fr pour avoir une mise en page spécifique destinée aux mobiles " français ".
-

Une mise en page pour chaque orientation et des images adaptées pour chaque résolution

Résolution, tablettes & smartphones

Avec une troisième génération d'iPad, Apple a semé le trouble dans les résolutions d'écrans. En effet l'iPad 3 présente un nombre plus élevé de pixels que la majorité des PC avec une taille d'écran divisé par deux.

Je propose donc un petit tableau pour essayer de faire le point sur le parc des résolutions d'écrans.

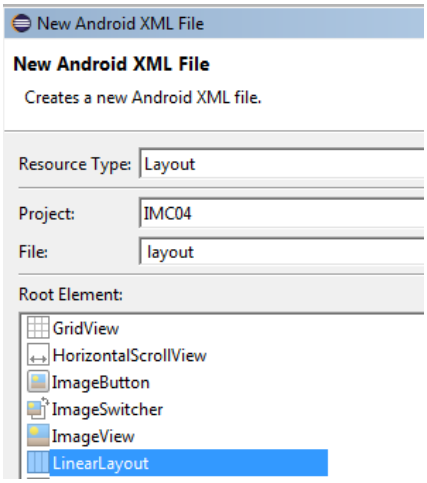
Modèle	Définition (pixels)	Taille de l'écran (pouce)	Résolution (ppp)
iPhone 3GS	320×480	3,5	164
HTC Hero	320×480	3,2	180
Nokia N900	800×480	3,5	266
iPhone 4s	960×640	3,5	326
Galaxy Notes	1280×800	5,3	285
iPad 2	1024×768	9,7	132
Galaxy Tab 10,1	1280×800	10,1	150
Playbook	1024×600	7	170
Galaxy Tab 8,9	1280×800	8,9	170
iPad 3	2048×1536	9,7	264
Macbook Air	1440×900	13,3	128
iMac	2560×1440	27	109
Macbook pro Retina	2880×1800	15,4	220

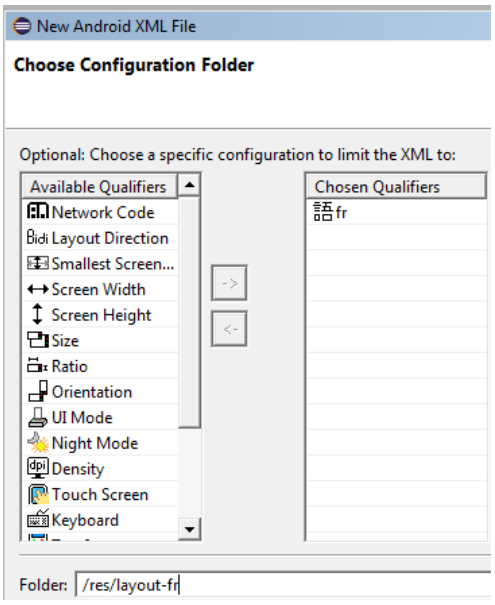
3. Créer des layout adaptés à la langue paramétrée sur le mobile d'exécution

Exemple d'application : On veut que notre programme IMC s'exécute en français ou en anglais selon la langue déclarée par défaut dans le paramétrage du support mobile.

On doit créer deux layout typés, l'un pour mobile en français, l'autre pour mobile en anglais

- **dossier** **res**
- **clic droit** **new/XML Android file**
- **nommer le fichier** ***activity_main***
- **sélectionner** **LinearLayout**
- **sélectionner** **Language / fr**
- **valider**





Un nouveau fichier XML a été créé dans le dossier res : ***layout-fr***

Procéder de même en créant un layout pour support en langue anglaise : **layout-en**

Lors de l'exécution de l'application, Android détecte la langue du mobile et va chercher le layout correspondant -en pour anglais, ou - fr pour français.

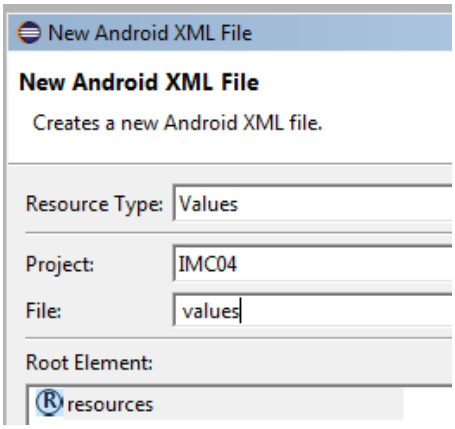
Tester
Copier/coller le code de *activity_main.xml*, saisi précédemment, vers les deux nouveaux layouts. Modifier une partie du code pour l'exécution du *layout-en* en langue anglaise. Modifier la langue par défaut du mobile dans les paramétrages, déclarez-la en anglais. Tester !

4. La gestion des chaînes de caractères adaptées à la langue déclarée sur le support mobile

Ça serait beaucoup mieux si les textes s'affichaient dans la langue de l'utilisateur ! C'est là qu'intervient toute la puissance du paramétrage générique des chaînes de caractères.

On sait maintenant que l'application Android est capable de s'adapter à la langue paramétrée sur le mobile. Il suffit de modifier les chaînes de caractère de chaque fichier XML pour que tout cela fonctionne sans problème.

On peut de la même manière que précédemment typer le dossier **values** en ***values-en*** et ***values-fr***.



Les **strings** servent à référencer des chaines de caractères utilisées dans l'application. On utilisera donc les chaines de caractères génériques côté application Java, laquelle demeure tout à fait indifférente au format d'affichage, les chaînes de caractères spécifiques de la langue côté interface. Android ira chercher la bonne translation conformément aux paramètres du mobile d'exécution en activant les chaines de caractères à partir du dossier values-typé.

Comment ça marche ?

Dans le fichier XML qui s'affiche on déclare les chaînes de manière générique. Exemple :

- **@string/ *app_name***

  app_name est un nom générique utilisé dans toute l'application : programmes java ou interfaces

Au moment de l'exécution du programme sur le support mobile, une conversion est effectuée avec le nom déclaré dans le fichier des strings. Exemple :

```
<string name = "app_name"> Application IMC  </string>
```

Dans cet exemple, ***app_name*** déclaré dans le fichier XML est converti en **Application IMC** lors de l'exécution.

Si on place **Application IMC** dans les déclaratives du layout-fr, c'est ce nom là qui sera affiché sur le mobile paramétré en français.

Si on place **IMC Application** dans les déclaratives du layout-en, c'est ce nom là qui sera affiché sur le mobile paramétré en anglais.

CQFO !

Tester

Copier/coller le code de *activity_main.xml*, déjà saisi, vers les deux nouveaux layouts.

Modifier une partie du code pour l'exécution du *layout-en* en langue anglaise.

Modifier la langue par défaut du mobile dans les paramétrages, déclarez-la en anglais. Tester !

Il faut maintenant spécifier chacun des layouts pour que l'interface de l'IMC soit totalement adaptée aux besoins de l'utilisateur final.

Procédez à l'amélioration de l'interface pour deux formats de sortie : Smartphone et tablette