Chapitre 04
Les ressources

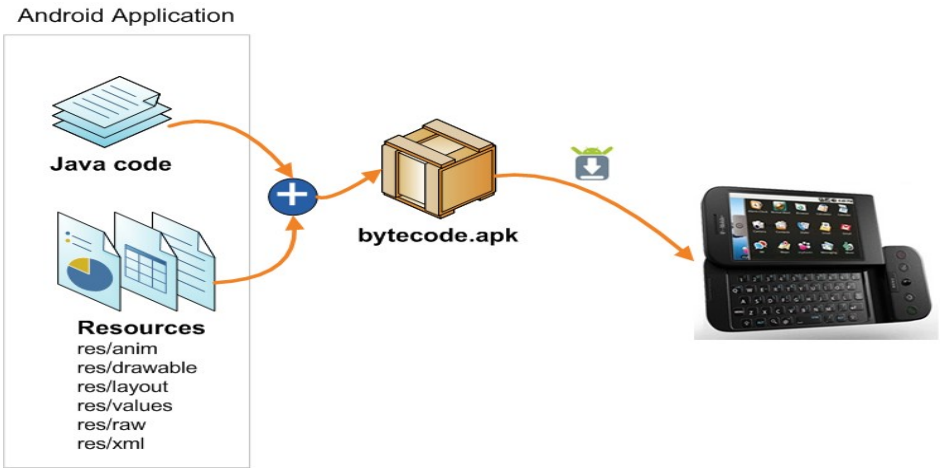
1 Introduction

- Le système Android est utilisé sur une variété de supports (taille de l'appareil, la résolution de son écran, la langue d'utilisation, l'orientation de l'écran, etc.).
- Android a donc besoin de s'adapter à ces supports.
- Ceci est réalisé à l'aide des ressources.
- Ces ressources sont des informations statiques stockées en dehors du code Java.
- Il est donc important de les externaliser.
- Les externaliser permet donc de fournir des ressources alternatives pour tenir compte de la variabilité des supports.
- Ainsi donc la réalisation d'une interface graphique devient simplifiée.
- Or, par exemple, une description pour une langue risque de ne plus être bonne pour une utilisation dans une autre langue.

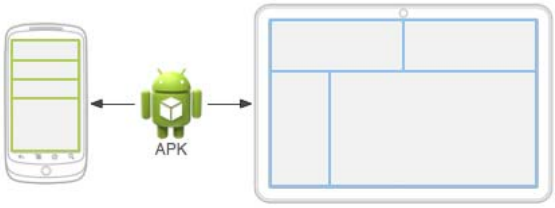
- Ainsi donc, il nous est possible de spécifier des ressources par défaut comme nous pouvons spécifier des ressources alternatives.

- Parmi ces ressources, on peut citer : images, vidéo, audio, chaînes de caractères, etc.

- Ces ressources sont regroupées dans le fichier « apk »lors du processus de construction de l'application.



- Deux appareils différents, chacun utilisant la configuration par défaut. L'application ne fournit pas une mise en page alternative pour un appareil ayant un écran plus grand.

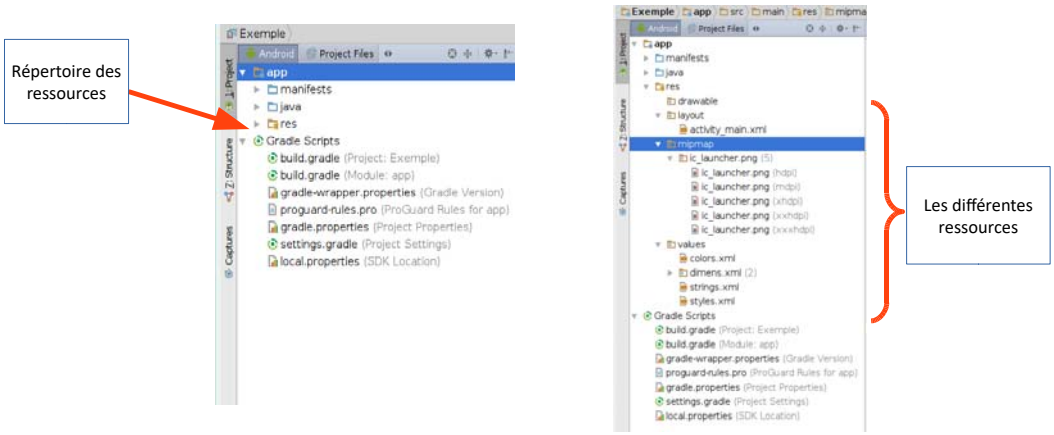


- Deux appareils différents, chacun utilisant une mise en page différente associée à la taille de l'écran.

2 Le répertoire « res »

- Ce répertoire stocke les différentes ressources associées à notre application.

- Ces ressources sont organisées dans une hiérarchie particulière.



- « drawable » : ce répertoire contient les ressources qui peuvent être dessinées. Parmi ces ressources, l'on peut citer les images (jpeg, gif, etc.), des formes génériques (cercles, carrés, etc.) définies dans des fichiers XML, etc.

  Chaque résolution de la ressource est associée à son propre dossier.

ldpi	mdpi	hdpi	xhdpi	xxhdpi
120 dpi	160 dpi	240 dpi	320 dpi	480 dpi

- « layout » : c'est l'interface graphique de votre application ou bien les différentes vues nécessaires pour la mise en page.

- « menu » : tout ce qui est en rapport avec les menus de l'application ou les barres d'action.

- « values » : des fichiers XML qui contiennent des valeurs simples, tels que chaînes, des entiers et des couleurs.

Chaque ressource est définie avec son propre élément XML, vous pouvez nommer le fichier que vous voulez et placer différents types de ressources dans un seul fichier. Cependant, pour plus de clarté, nous séparons les choses comme suit :

- « arrays.xml » pour les tableaux des ressources.
- « colors.xml » pour les couleurs.
- « dimens.xml » pour les dimensions.
- « strings.xml » pour les chaînes de caractères.
- « styles.xml » pour les styles.

- « raw » : des données brutes dans leur format d'origine. Ce répertoire peut contenir des fichiers audio, vidéo, html, etc.

- « color » : des listes d'états des couleurs. Bouton appuyé, couleur rouge, bouton désactivé, couleur grise.

- « anim » : fichiers d'animation.

- « mipmap » : les ressources de différentes densités pour les icônes de lancement.

- « xml » : des fichiers xml quelconques lus au lancement de l'application.

3 Accéder aux ressources

- Les ressources sont rangées dans des répertoires particuliers.

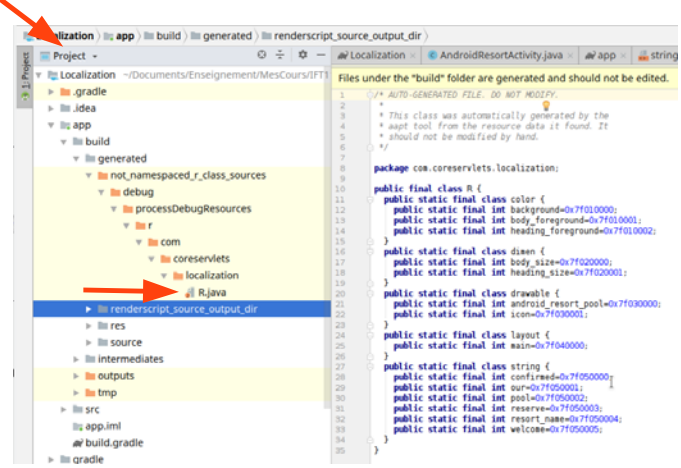
- Android va se charger automatiquement d'associer un identificateur à chaque ressource.

- L'utilisateur peut associer son propre identifiant. Dans ce cas, Android va de facto prendre en compte cette information, au lieu de générer lui-même un identifiant pour cette ressource.

- Elles sont enregistrées par la suite dans l'application.

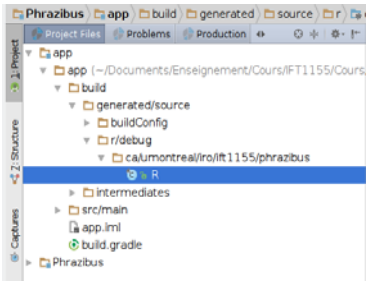
- Cet enregistrement va se faire à travers la classe « R ».

- Après avoir choisi l'onglet « Project », examiner la classe « R » de votre exemple dans : app/build/generated/……../R.java



- Chaque ressource est une classe imbriquée dans la classe « R ».

- Chaque élément de la ressource a un identificateur du type entier.



- Cette hiérarchie facilite l'accès à un élément quelconque de la ressource.

  L'étiquette « adore » associée au bouton dont l'identifiant est « button8 » a été définie dans le fichier «activity_main.xml » comme suit :

```
<Button
  android:id="@+id/button8"
  style="@android:style/Widget.DeviceDefault.Button.Small"
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_weight="1"
  android:onClick="Verbe"
  android:text="@string/adore"
  android:textColor="@android:color/black" />
```

Ce bouton a été enregistré par la suite dans la classe « R.java » comme suit :

```
public final class R {
    // [...]
    public static final class id {
        // [...]
        public static final int button8=0x7f060015;
    }
}
```

Il sera accessible comme suit :

Java	R.id.button8
XML	android:id= "@id/button8"

L'étiquette « adore » a été associée au bouton numéro 8 dans le fichier « activity_main.xml », lors de la création du bouton.

Cette étiquette a été définie dans le fichier « strings.xml » comme suit :

```
<string name="adore">adore</string>
```

Cette étiquette a été enregistrée par la suite dans la classe « R.java » comme suit :

```
public final class R {
    // […]
      public static final class string {
        //[…]
        public static final int adore=0x7f05000a;
        //[…]
    }
}
```

Elle sera accessible comme suit :

Java	R.string.adore
XML	android:text= "@string/adore"

- La syntaxe générale en Java :

```
[<package_name>.]R.<resource_type>.<resource_name>
```

- <*package_name*> est le nom du paquetage où est située la ressource. Il n'est pas obligatoire quand la ressource est localisée dans le même paquetage.
- <*resource_type*> est le nom de la sous-classe de R associée au type de la ressource.
- <*resource_name*> est soit le nom du fichier associé à la ressource sans son extension, ou bien une valeur android:name définie dans le fichier XML (une simple valeur).

- La syntaxe générale en XML :

```
@[<package_name>:]<resource_type>/<resource_name>
```

- <*package_name*> est le nom du paquetage où est située la ressource. Il n'est pas obligatoire quand la ressource est localisée dans le même paquetage.
- <*resource_type*> est le nom de la sous-classe de R associée au type de la ressource.
- <*resource_name*> est soit le nom du fichier associé à la ressource sans son extension, ou bien une valeur android:name définie dans le fichier XML (une simple valeur).

- Pour récupérer une ressource à partir d'un identificateur :

  getResources().getString(id);

- L'exemple « DeuxActivIntent » du chapitre 03 contient les deux vues « main.xml » et « second.xml ». Pour sélectionner :

  - la vue « main » : setContentView(R.layout.main);
  - la vue « second » : setContentView(R.layout.second);

4 « Strings »

- Elles sont localisées dans le fichier « strings.xml » situé dans « res/values ».

```
<string name="string_nom">text_string</string>
```

- Attention aux caractères spéciaux:

```
<string name="string_nom">L\'apostrophe</string>

<string name="string_nom">"L'apostrophe"</string>

<string name="string_nom">
            "L'apostrophe dans \"un double quotte\" "</string>
```

- Avec un peu de html :

```
<string name="string_nom"><b>L\'apostrophe</b></string>

<string name="string_nom"><u>L\'apostrophe</u></string>

<string name="string_nom"><i>L\'apostrophe</i></string>
```

- Formatage « HTML » et valeurs dynamiques :

```
<string name="welcome_messages">Bonjour, %1$s! Vous avez
                &lt;b>%2$d nouveaux messages&lt;/b>.</string>
```

Si %1 vaut Michel et %2 vaut 10, la chaîne affichée est :

Bonjour, Michel! Vous avez **10 nouveaux messages**.

Examinons la chaîne « %1$s » :

- %n : « n » un entier non nul. Il sert à indiquer le rang de l'argument.
- $x : la nature de l'information. Le caractère « s » est associé à une chaîne de caractères, « d » à un entier.

```
String escapedUsername = TextUtil.htmlEncode(username);

Resources res = getResources();

String text = String.format(res.getString(R.string.welcome_messages),
                                                escapedUsername, mailCount);

CharSequence styledText = Html.fromHtml(text);
```

L'exemple associé est le projet « **C04 : StringsDemo** ».

- « String Array » : un tableau de chaînes de caractères :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="planets_array">
        <item>Mercury</item>
        <item>Venus</item>
        <item>Earth</item>
        <item>Mars</item>
    </string-array>
</resources>
```

```
Resources res = getResources();
String[] planets = res.getStringArray(R.array.planets_array);
```

L'exemple associé est le projet « **C04 : ResourceApp1** ».

- La force de « Plurals » :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <plurals name="numberOfSongsAvailable">
        <item quantity="one">One song found.</item>
        <item quantity="other">%d songs found.</item>
    </plurals>
</resources>
```

```
int count = getNumberOfsongsAvailable();
Resources res = getResources();

String songsFound =
         res.getQuantityString(R.plurals.numberOfSongsAvailable,
                                          count, count);
```

L'exemple associé est le projet « **C04 : PluralDemoActivity** ».

5 Localisation et ressources

- Trouver les paramètres qui affectent votre application :

Langage, orientation, densité de l'écran, station d'accueil, etc.

- Trouver le qualificateur adéquat qui correspond à chaque paramètre :

http://developer.android.com/guide/topics/resources/providing-resources.html#table2

- Langage : en, en-rUS, es, es-rMX, fr, fr-rCA

http://www.loc.gov/standards/iso639-2/php/code_list.php

http://www.iso.org/iso/prods-services/iso3166ma/02iso-3166-code-lists/country_names_and_code_elements

- Déterminer l'orientation de l'unité : port, land
- Déterminer la densité de l'écran : ldpi, mdpi, hdpi, xhdpi, etc.
- Déterminer le mode d'accueil : car, desk, etc.

- Ajouter le qualificateur au nom du répertoire

  - res/values/strings.xml,
    res/values-es/strings.xml,
    res/values-es-rMX/strings.xml

  - res/layout/main.xml,
    res/layout-land/main.xml

- Accéder aux ressources normalement

  - R.string.title, R.layout.main, etc.
  - Android va sélectionner la vue adéquate.
  - Il va se charger aussi de sélectionner la ressource la plus appropriée en fonction de la configuration de votre équipement, si cette ressource a été définie. Dans le cas contraire, il va sélectionner la ressource par défaut.

- L'idée est d'ajouter les ressources interchangeables en fonction de la langue par exemple. Il n'est pas conseillé d'adapter la vue à la langue pour éviter un méchant casse-tête conceptuel!

- Pour changer l'orientation de l'écran dans l'émulateur : F11 (paysage), F12 (portrait).

- Modifier la langue et tester votre application. Un autre test à faire est de prendre un indicatif régional qui n'a pas été prévu et d'examiner le comportement de votre application.

  Pour modifier la langue et l'indicatif régional sur votre appareil, référez-vous au guide d'utilisation de votre appareil.

  Pour modifier la langue sur l'émulateur, il faudra cliquer sur « Paramètres », puis « Langue et saisie », puis « Langue » et choisir la langue désirée.

  Pour modifier l'indicatif régional, il faudra cliquer sur l'application « Locales personnalisées » et choisir le code dans la liste.

  Vous pouvez aussi modifier la langue et l'indicatif régional dans le code Java, même s'il est déconseillé de le faire. Il suffit que la personne change de langue et vous êtes enfermés sur vous-mêmes!

Vous pouvez le faire aussi à travers « adb », en étant « root » comme suit :

Se connecter à l'émulateur, par exemple (s'il n'y a qu'un en service) :

```
adb shell
su
```

Écrire l'une des commandes en fonction de la langue et l'indicatif régional désirés.

Canadien Français : `setprop persist.sys.locale fr-CA;stop;sleep 5;start`

Espagnol Mexicain : `setprop persist.sys.locale es-MX;stop;sleep 5;start`

Espagnol d'Espagne : `setprop persist.sys.locale es-ES;stop;sleep 5;start`

Anglais USA : `setprop persist.sys.locale en-US;stop;sleep 5;start`

Pour obtenir la configuration courante

```
getprop|grep -E "language|country|locale"
```


6. Projet « Android Resort »

- L'application est disponible dans 3 langues (en, es, fr) et 4 codes régionaux (US, ES, MX, FR).

- Nous avons ajouté pour cela 3 répertoires supplémentaires dans l'application : « values-es », « values-es-rMX » et « values-fr ». Pour l'anglais, il est dans le répertoire par défaut associé à l'application « values ». Ce répertoire va être utilisé pour toutes les langues sauf les 2 autres langues définies avec votre application (français et espagnole).

- Dans ces 4 répertoires, nous avons ajouté le fichier « strings.xml ». Ce dernier définit les différentes chaînes de caractères utilisées par notre application dans la langue et l'indicatif régional appropriés.

- Le répertoire contenant le fichier « strings.xml » associé à un indicatif régional d'une langue donnée doit-être nommé comme suit : « values-langue-rCODERÉGIONAL ». Pour le mexicain, « langue=es », et « CODERÉGIONAL=MX ». L'indicatif régional est précédé par la lettre « r ».

- Ainsi donc, dans le fichier « strings.xml » situé dans le répertoire « values-es-rMX », nous n'avons ajouté qu'un seul mot spécifique à l'espagnol utilisé au Mexique. En effet, le mot « piscine » en espagnol est « piscina » et en mexicain « alberca ». Android va utiliser toutes les chaînes de caractères définies dans « values-es ». Mais dès qu'il arrive à la chaîne « pool », il va utiliser celle redéfinie dans le fichier « values-es-rMX ».

- Avec cette approche, vous constatez qu'il est plus simple de développer des applications multilingues avec Android. Il suffit pour cela de regrouper les choses qui changent ensemble en dehors de l'interface graphique.

- L'orientation de l'écran a été définie dans deux répertoires : « layout » se charge par défaut de l'orientation portrait; « layout-land » se charge de l'orientation « paysage ».

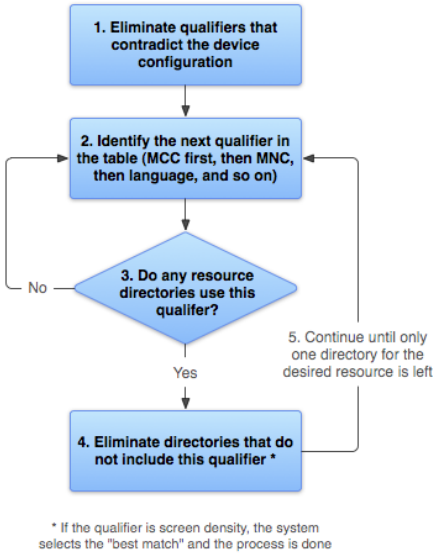
- Les dimensions sont aussi ajustées en fonction de l'orientation. Dans le répertoire « values » nous avons les dimensions par défaut et dans le répertoire « values-land » celles associées à une orientation paysage.

- Pour l'image affichée, inclure une image de la taille de la résolution de l'écran. Comme nous l'avons déjà expliqué au début de ce chapitre, en général vous allez avoir affaire à 5 résolutions de l'écran. Il faudra ajuster les images en conséquence. Vous pouvez aussi ajuster l'image en modifiant ses propriétés dans le fichier XML associé.

- Assurez-vous de faire le même traitement pour les icônes.

L'exemple associé est le projet « **C04 : Localization** ».

7. Comment Android va sélectionner la ressource la plus appropriée



- Nous disposons de ces répertoires :

```
drawable/
drawable-en/
drawable-fr-rCA/
drawable-en-port/
drawable-en-notouch-12key/
drawable-port-ldpi/
drawable-port-notouch-12key/
```

- Nous allons supposer que l'appareil à la configuration suivante :

Locale = `en-GB`
Screen orientation = `port`
Screen pixel density = `hdpi`
Touchscreen type = `notouch`
Primary text input method = `12key`

- Android va sélectionner le répertoire « drawable-en-port/ ».

- Comment Android a-t-il procédé pour arriver à ce résultat?

- 1) Éliminer les ressources qui contredisent la configuration de l'appareil.

```
drawable/
drawable-en/
~~drawable-fr-rCA/~~
drawable-en-port/
drawable-en-notouch-12key/
drawable-port-ldpi/
drawable-port-notouch-12key/
```

- 2) Choisir le prochain qualificateur de la table -2- disponible ici :

http://developer.android.com/guide/topics/resources/providing-resources.html#table2

- 3) Est-ce qu'il y a des répertoires ressources associés à ce qualificateur, si oui étape 4, sinon retourner à l'étape 2 et prendre le prochain qualificateur de la liste.
- 4) Refaire les étapes 2, 3 et 4 jusqu'à arriver à un seul répertoire.

```
~~drawable-en/~~
drawable-en-port/
~~drawable-en-notouch-12key/~~
```

- Ainsi donc il ne va rester que le répertoire « `drawable-en-port` ».

8 Bibliographies

Resources Overview

http://developer.android.com/guide/topics/resources/overview.html

Localization and Resources

http://courses.coreservlets.com/Course-Materials/pdf/android/Android-Localization.pdf