

Applications sur Android

Objets connectés



Applications Android

- Introduction à cette formation
 - Votre formateur ... Et Vous
 - Le matériel
 - Le support de cours
 - Android Studio
 - Téléphones MOTO
 - L'organisation – horaires
 - Formation de 3,5 jours
 - La forme :
 - Un mélange de concepts avec application directe par un exemple simple
 - Des exercices, une évaluation



Applications Android

- Des liens utiles
 - <http://developer.android.com> site officiel
 - <https://developer.android.com/guide>
 - <https://openclassrooms.com/courses/creez-des-applications-pour-android/votre-premiere-application-1>

- Un peu d'histoire sur Android
- Quelques rappel JAVA
- Mise en route
- Les bases
- Les ressources
- Les Widgets
- Les événements
- Layouts
-

- Créé en 2003 par Android Incorporated
- Racheté en 2005 par Google
- Existe avant l'iPhone d'Apple
- Le but :
 - Créer un OS intelligent, pour ne pas se limiter à la téléphonie, SMS, MMS
 - Interagir avec l'environnement : géo localisation, accélérateur/ capteur de mouvements ...
- En 2007 sortie de l'iPhone, une révolution, avec iOS, l'iPhone OS
- L'OHA – l'Open Handset Alliance, à été créée en 11/2007 pour définir une solution concurrente. Cette alliance comptait 35 entreprises à sa création, dont Google. 80 entreprises actuellement.



- En 2010 devient l'OS mobile le plus utilisé.
- Associé aux smart phones, tablettes, consoles de jeu, objets connectés, systèmes embarqués, appareil photo ...
- Open Source
- Gratuit
- Existence du Play Store pour déposer vos applications (25\$ pour un nombre illimité d'applications)
- Bibliothèques puissantes : SQLite, OpenGL ...
- Des outils de développement gratuits
- Java est LE langage de développement sur Android jusqu'en 2019
- Le langage Kotlin, disponible en 2017, devient LE langage préconisé par Google pour le développement Android.

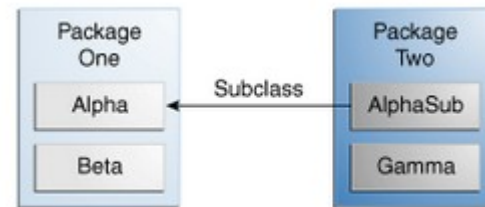


- Il existe des Framework et outils pour développement en « cross - platform » : un seul développement et cible Android et iOS
 - Xamarin (Microsoft 2016, langage C#)
 - React Native (Facebook 2015, langage Javascript)
 - Flutter (Google 2017, langage Dart)
 - Phone Gap
 - Titanium
 - Apache Cordova
 - Angular JS / Ionic
 -
- Pourquoi développer avec Android Studio uniquement pour une cible Android ?
 - Une formation Java et Java EE a eu lieu
 - Android Studio moins lourd que d'autres (ex Visual Studio 2019 et son simulateur)
 - Nous n'avons que 3,5 jours !

- Un module externe se nomme un « package ». Il contient des définitions de classes traitant d'un sujet donné.
L'utilisation d'un package ou d'une partie se fait par
`import <nom du package>`
ex : `import android.content.Context;`
- Toute donnée membre de classe ou variable locale doit être déclarée
ex : `boolean isSent = false;`
- Un fichier source .java ne contient la définition que d'une seule classe déclarée public. Le nom du fichier correspond exactement au nom de cette classe.
Pour autant une classe peut définir une autre classe ou interface imbriquée.
- La déclaration d'une classe se fait par le mot `class`

- Les « modifieurs »

Visibility				
Modifieur	Alpha	Beta	Alphasub	Gamma
public	Y	Y	Y	Y
protected	Y	Y	Y	N
no modifier	Y	Y	N	N
private	Y	N	N	N



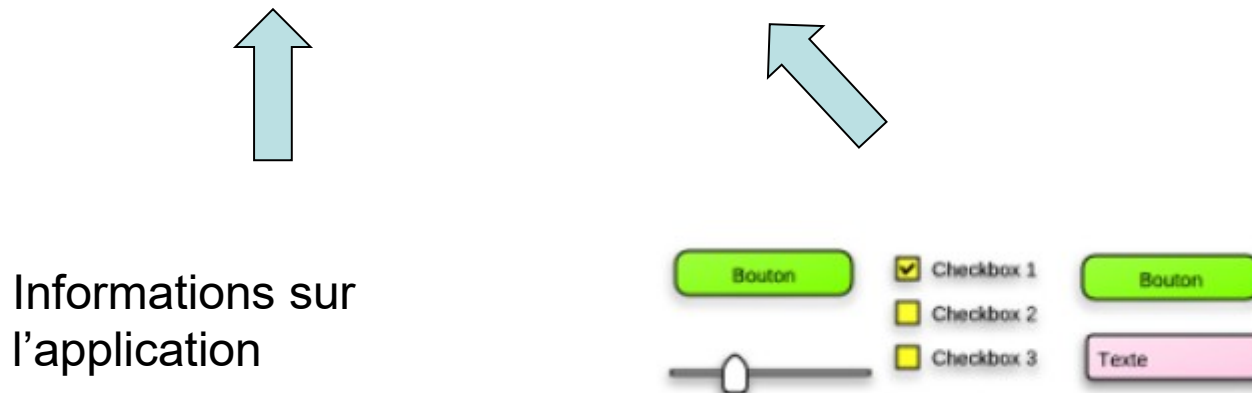
Ex `public class` MaClasse { }

- L'héritage
`class` MaClasseDerivee `extends` MaClasse { }
- L'interface
Sorte de classe ne contenant que des méthodes abstraites
`class` MaClasse `implements` ImonInterface { }
- Le polymorphisme s'applique à des Objets de **même lignée** (héritage) ou implémentant la **même interface**

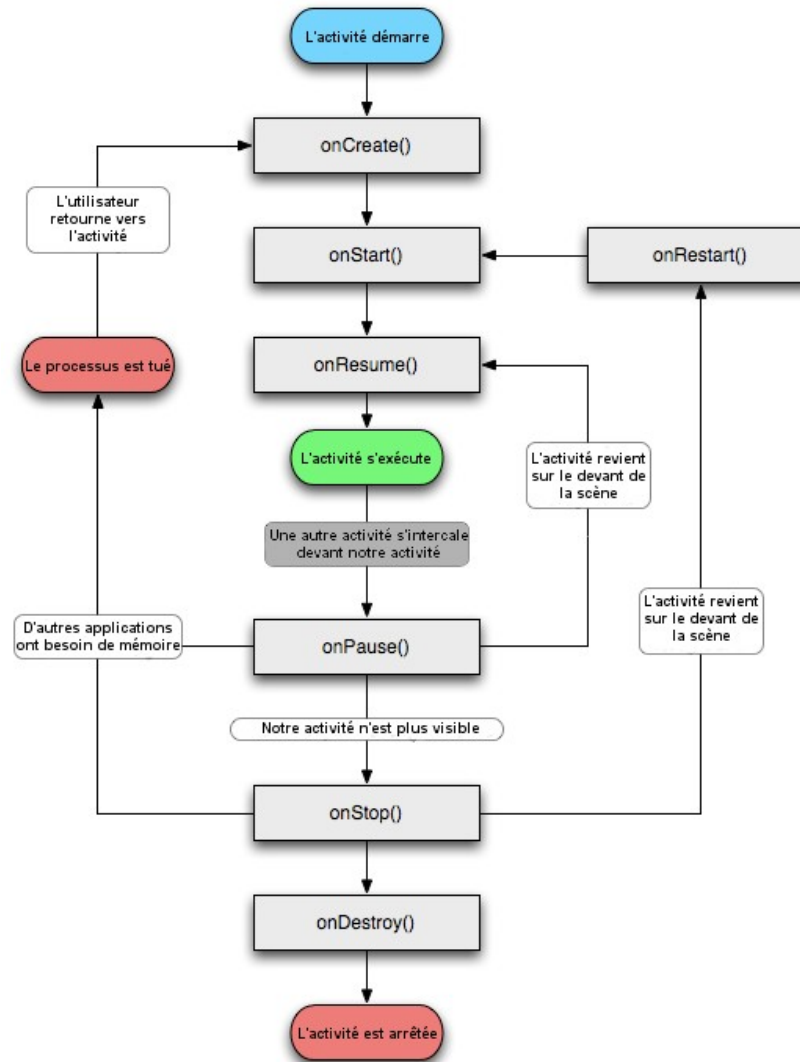
- Ressources de la machine de développement
 - 4 Go de mémoire juste suffisant, 8 Go préférable
 - Environ 2 Go d'espace disque nécessaire
 - Processeur récent, le « support HAXM » est nécessaire pour l'émulation sur le poste de développement.
- Utiliser le fichier d'exercices pour démarrer ...



- Une application Android est un ensemble de fenêtres appelables les unes par les autres.
- Chacune de ces fenêtres s'appelle une « Activité »
- Une Activité remplit graphiquement tout l'écran
- Une Activité = Context + interface graphique



- L'état d'une activité
L'état varie au fil du temps.
- Les Activités sont dans une pile d'Activités. Seule l'Activité en haut de la pile est visible
- Une application peut laisser place à une autre application. Les activités de celle-ci seront en haut de pile. Ex appel téléphonique
- Si une application utilise trop de ressource, Android peut décider de la supprimer



- Côté code, une Activité hérite de la classe Activity, qui implémente l'interface Context

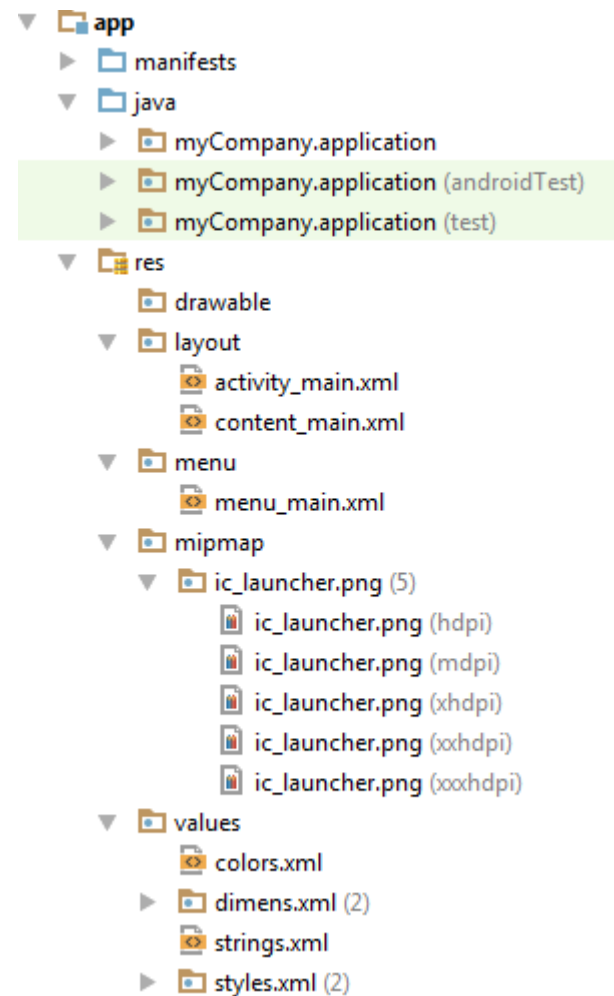
```
public class Activity extends ApplicationContext {  
    protected void onCreate(Bundle savedInstanceState);  
  
    protected void onStart();  
  
    protected void onRestart();  
  
    protected void onResume();  
  
    protected void onPause();  
  
    protected void onStop();  
  
    protected void onDestroy();  
}
```

<https://developer.android.com/reference/android/app/Activity.html>

- Les applications Android sont conçues pour être affichées :
 - sur des supports différents, avec des tailles différentes
 - Peut être dans plusieurs langues
- Les « ressources » sont des fichiers non Java qui contiennent des informations statiques. Il n'est alors pas nécessaire de redévelopper du code pour changer de taille, de langue ...
- Les fichiers ressources sont au format XML
-

Lien <https://developer.android.com/guide/topics/resources/providing-resources>

- f



- Les différentes ressources sont organisées en répertoires

Répertoire	Usage	Types de fichiers
res/drawable	images matricielles. images vectorielles en xml	png, jpeg, gif xml
res/layout	disposition des objets dans les vues	xml exclusivement
res/menu	items de menus	xml exclusivement
res/values	de nombreux fichiers de	xml exclusivement
res/mipmap	icônes de l'application, selon résolution	png

- Les répertoires du tableau précédent sont des répertoires par défaut.
- On peut créer des sous-répertoires pour qualifier des cas plus précis : une langue précise, une taille précise ...
- Ces précisions sont représentés par des « quantificateurs » selon la syntaxe

`res/<type_de_ressource>-<quantificateur1>-<quantificateur2>-...<quantificateurN>`

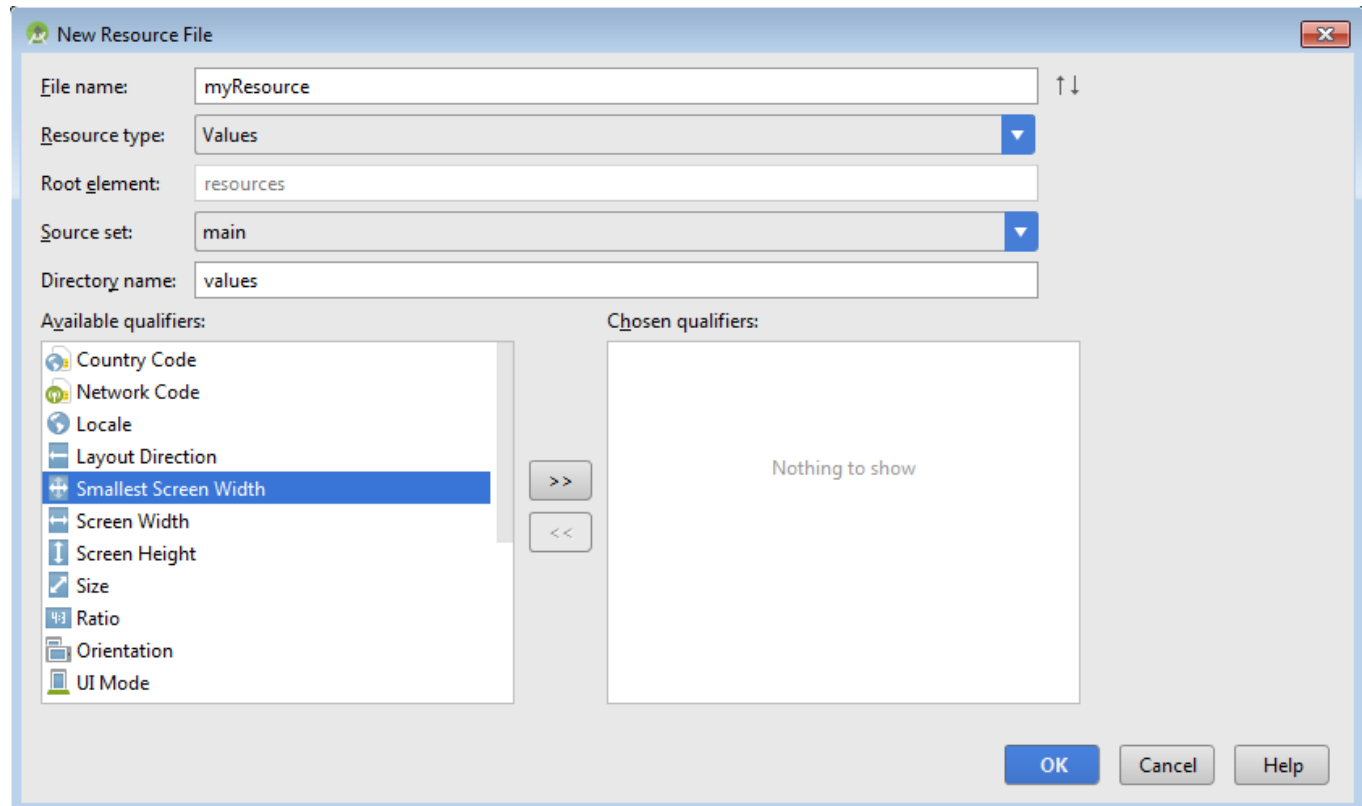
ex : `res/layout-fr`

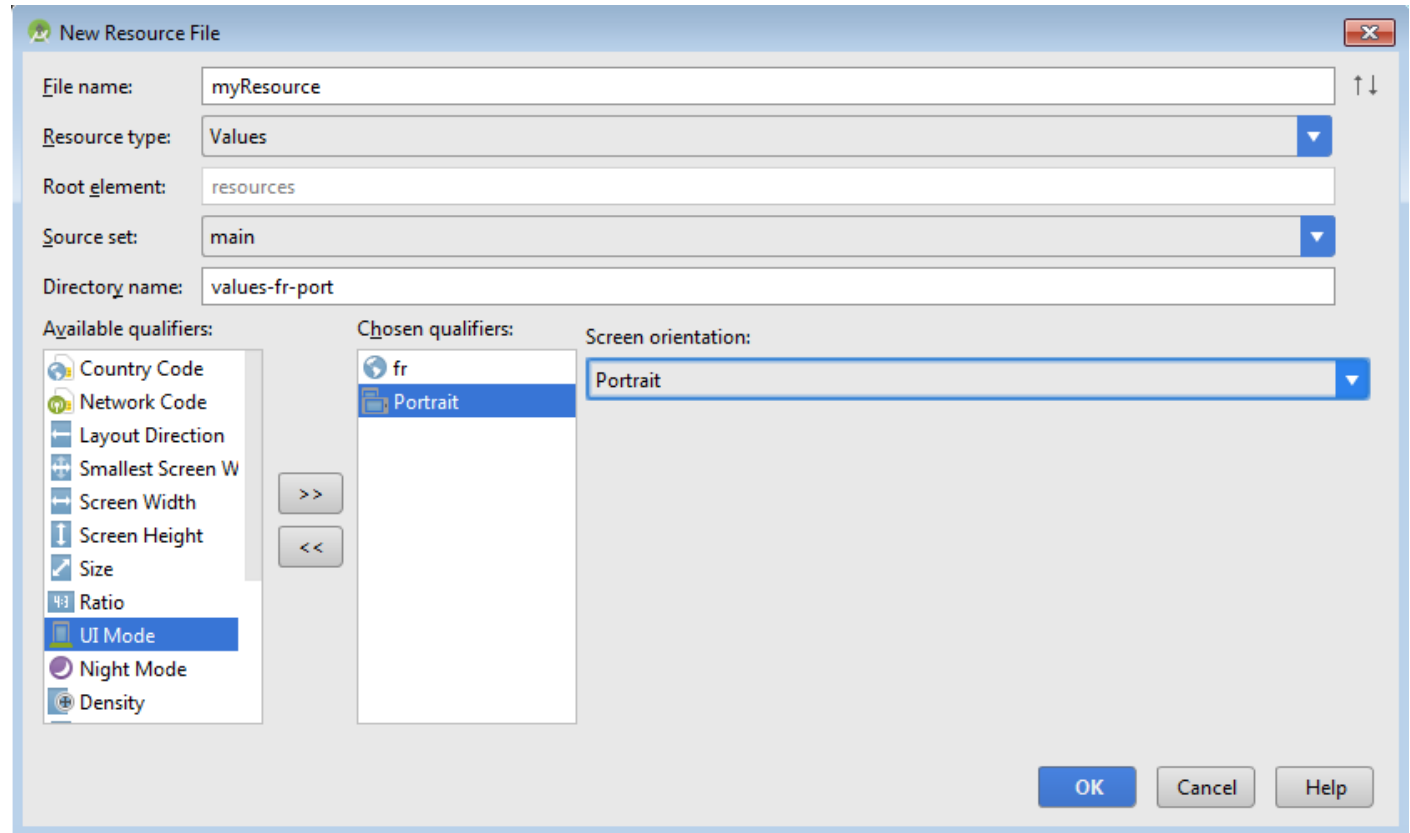
Pour une mise en page spécifique si OS en français

- Principaux quantificateurs
 - Priorité 2 : langue de l'OS . Langue et éventuellement une région
 - en pour anglais
 - fr pour français
 - fr-rFR
 - fr-rCA pour le français utilisé au Québec
 - Priorité 3 : taille de la diagonale de l'écran
 - small pour les écrans de petite taille ;
 - normal pour les écrans standards ;
 - large pour les grands écrans, comme dans les tablettes tactiles ;
 - xlarge pour les très grands écrans, genre téléviseurs
 - Priorité 5 : Orientation
 - port , mode portrait
 - land , mode paysage (landscape)

- Priorité 8 : résolution écran
 - ldpi environ 120 dpi
 - mdpi environ 160 dpi
 - hdpi environ 240 dpi
 - xhdpi environ 320 dpi
 - nodpi pour ne pas redimensionner les images jpeg, png, gif
- Priorité 14 : version Android
- Exemples
 - res/drawable-hdpi , res/drawable-mdpi , res/drawable-ldpi
Android ira chercher, en fonction de la taille réelle de l'écran du terminal, les ressources dans le bon répertoire.

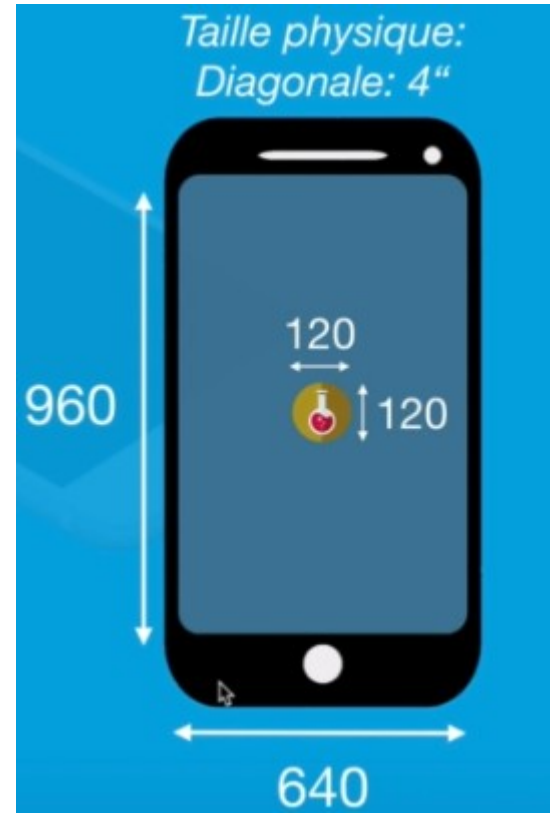
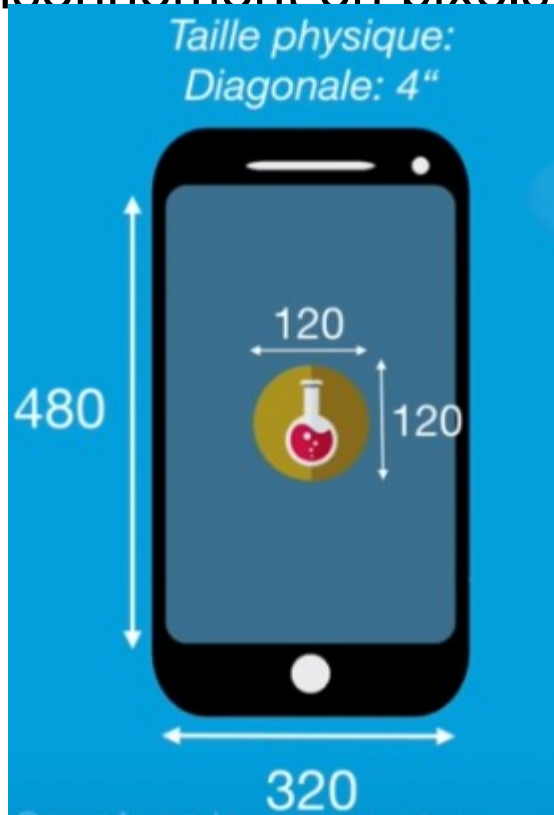
- Ajouter une ressource avec Android Studio :
 - Clic droit sur le nœud res
 - New-> Android resource file







- Récupérer dans le code Java les ressources : utiliser la classe R
- Le code de la classe R est généré automatiquement par Android Studio et contient des identifiants sur les ressources
- Utiliser dans votre code Java source l'aide à la saisie (IntelliSense) à partir de « R »
ex `R.menu.main_menu` `R.string.app_name`
- Ces identifiants sont des int
Pour récupérer le contenu d'une ressource, utiliser `getResources()`
`String applicationName = getResources().getString(R.string.app_name);`

- <https://developer.android.com/guide/topics/graphics>
- Densité d'image
- Raisonnement en pixels



Il faut raisonner en dp : density independant pixel.

Est le reflet de ce que l'on voit, quelle que soit la résolution de l'écran.

	mdpi	hdpi	xhdpi	xxhdpi	xxxhdpi
Rapport px/dp	1x	1.5x	2x	3x	4x
Résolution téléphone	320x480	480x720	640x960	960x1440	1280x1920
 Image logo	120x120	180x180	240x240	360x360	480x480
 iOS	lab_logo.png		lab_logo@2x.png	lab_logo@3x.png	lab_logo@4x.png

- La fonction Asset Studio de l'outil Android Studio génère automatiquement la série d'images aux bonnes densités pour les différents formats connus
- Le format 9 patch permet de définir la partie utile d'une image

- TextView
 - Xml

```
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/textView"
    android:textSize="8sp"
    android:textColor="#112233" />
```

- Fichier de ressources strings.xml

```
<resources>
    <string name="app_name">Application1</string>
    <string name="action_settings">Settings</string>
    <string name="textView">Hello My Word !</string>
</resources>
```

- Code Java

```
TextView textView = new TextView(this);
textView.setText(R.string.textView);
textView.setTextSize(8);
textView.setTextColor(0x112233);
```

- EditText : saisie de texte sous de nombreux formats
 - Xml

```
<EditText
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:hint="@string/editText"
    android:inputType="textMultiLine"
    android:lines="5" />
```

- Code Java

```
EditText editText = new EditText(this);
editText.setHint(R.string.editText);
editText.setInputType(InputType.TYPE_TEXT_FLAG_MULTI_LINE);
editText.setLines(5);
```

- Button
 - Xml

```
<Button  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/button" />
```

- Code Java

```
Button button = new Button(this);  
editText.setText(R.string.button);
```

- Checkbox
 - Xml

```
<CheckBox  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/checkbox"  
    android:checked="true" />
```

- Code Java

```
CheckBox checkBox = new CheckBox(this);  
checkBox.setText(R.string.checkbox);  
checkBox.setChecked(true)  
if(checkBox.isChecked())  
    // Faire quelque chose si le bouton est coché
```

- Radiogroup et RadioButton

- Xml

```
<RadioGroup
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal" >
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true" />
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
    <RadioButton
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />
</RadioGroup>
```

- Code Java

```
RadioGroup radioGroup = new RadioGroup(this);
RadioButton radioButton1 = new RadioButton(this);
RadioButton radioButton2 = new RadioButton(this);
RadioButton radioButton3 = new RadioButton(this);

// On ajoute les boutons au RadioGroup
radioGroup.addView(radioButton1, 0);
radioGroup.addView(radioButton2, 1);
radioGroup.addView(radioButton3, 2);

// On sélectionne le premier bouton
radioGroup.check(0);

// On récupère l'identifiant du bouton qui est coché
int id = radioGroup.getCheckedRadioButtonId();
```

- Événement : action de l'utilisateur sur un widget
- Pour que le code intercepte un événement, il faut ajouter un ou plusieurs « Listener » au widget.
- Un « Listener » permet d'associer la bonne Méthode (au sens objet = fonction) au bon type d'événement. Cette méthode est appelée un « Callback »

- Plusieurs méthodes existent pour déclarer un Listener :
 - Par codage

```
protected void onCreate(Bundle savedInstanceState) {  
  
    Button b = (Button) findViewById(R.id.button);  
    b.setOnClickListener( buttonClickListener);  
  
}  
  
private ClickListener buttonClickListener = new ClickListener();  
  
class ClickListener implements View.OnClickListener  
{  
    public void onClick(View v) {  
        switch (v.getId()) {  
            case R.id.button:  
                // TODO Auto-generated method stub  
                TextView textView = (TextView) findViewById(R.id.textView);  
                textView.setText("Cliqué");  
                break;  
            case View.NO_ID:  
            default:  
                // TODO Auto-generated method stub  
                break;  
        }  
    }  
}
```

- Par codage et classe anonyme

```
protected void onCreate(Bundle savedInstanceState) {  
  
    Button b = (Button) findViewById(R.id.button);  
    b.setOnClickListener( buttonClickListener );  
  
}  
  
private View.OnClickListener buttonClickListener = new View.OnClickListener()  
{  
    @Override  
    public void onClick(View v) {  
        switch (v.getId()) {  
            case R.id.button:  
                // TODO Auto-generated method stub  
                TextView textView = (TextView) findViewById(R.id.textView);  
                textView.setText("Cliqué");  
                break;  
            case View.NO_ID:  
            default:  
                // TODO Auto-generated method stub  
                break;  
        }  
    }  
};
```

- Par implémentation d'Interface

```
public class MainActivity extends AppCompatActivity implements
View.OnTouchListener, View.OnClickListener {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        ....

        Button b = (Button) findViewById(R.id.button);
        b.setOnClickListener(this);
        b.setOnTouchListener(this);

        ...

    }

    @Override
    public boolean onTouch(View v, MotionEvent event) {
        /* Réagir au toucher */
        return true;
    }

    @Override
    public void onClick(View v) {
        /* Réagir au clic */
    }
}
```

- Par classe anonyme, en direct

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {

        ....

        Button b = (Button) findViewById(R.id.button);
        b.setOnClickListener(new View.OnClickListener() {
            @Override // classe anonyme en direct
            public void onClick(View v) {
                /* Réagir au clic */
                TextView textView = (TextView) findViewById(R.id.textView);
                String texte = textView.getText() + "D_";
                textView.setText(texte);
            }
        });

        ...

    }
}
```

- Organiser l'interface : les Layouts
- <https://developer.android.com/guide/topics/ui/declaring-layout>

Linear Layout



Relative Layout



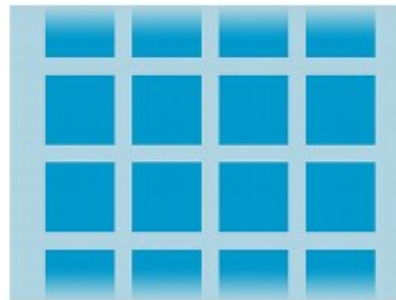
Web View



List View



Grid View



- Android sur un téléphone permet ... de téléphoner

<https://developer.android.com/guide/topics/connectivity/telecom/selfManaged>

- Permet aussi la géo localisation

<https://developer.android.com/training/location>