

Introduction à android

Sources :

Site : <https://developer.android.com/>

Livre « Android Programming the big nerd ranch guide » de B. Philips et B. Hardy

Cours de Jean-Michel Douin (CNAM)

Cours de Michael Laguerre

historique

- <http://www.openhandsetalliance.com/>
- <http://www.android.com/>
- Diverses versions (1.x à 4.4 KitKat, 7 Nougat)
 - Créé en 2007
 - Nom de sucrerie
 - 3.X version pour tablette
 - Fusion en 4.x

Généralité sur android

- OS « smartphone » en tête
 - http://en.wikipedia.org/wiki/Mobile_operating_system
 - En résumé : android gagne des parts...
 - Sur d'autres devices
- Evolution rapide des API/SDK/nom de sucrerie
 - Quelle(s) cible(s) pour votre application ?
 - Hétérogénéité des versions d'OS

Android, vue générale

- Les plus
 - Système open-source
 - Accès plus en profondeur au matériel en comparaison avec l'iPhone
 - Disponible sur une grande variété de devices
 - Peut être modifié pour s'adapter à un nouveau matériel
- Les moins
 - Nature open-source
 - Tous les constructeurs et intégrateurs ajoutent leurs touches personnelles
 - Très grande inertie dans les mises à jour ([des téléphones \(0,1% des androids\) encore en 2.2 fin 2016 selon android.com](#))
 - Hétérogénéité des plateformes

D'après le cours de Michael Laguerre

Ce qu'on ne verra pas dans cette présentation

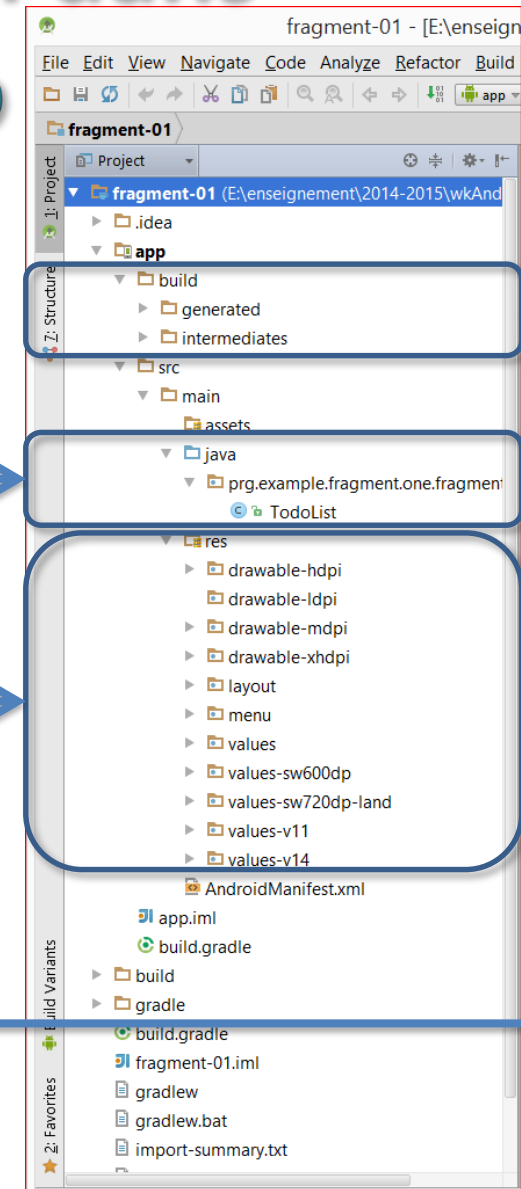
- Pas de gestion des différentes versions, des différents dispositifs
- Pas de transition
- Pas de programmation « système » ou « native »
- Pas de programmation pour smartphone
- Pas de « web apps »
- Pas d'open gl / unity

Installation et outils

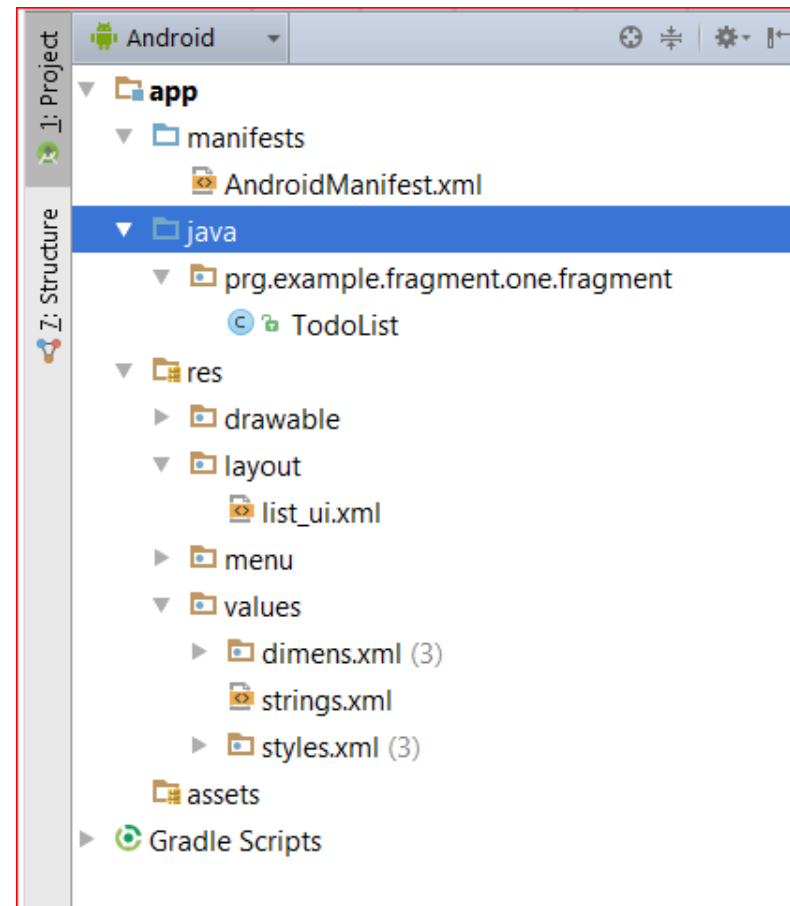
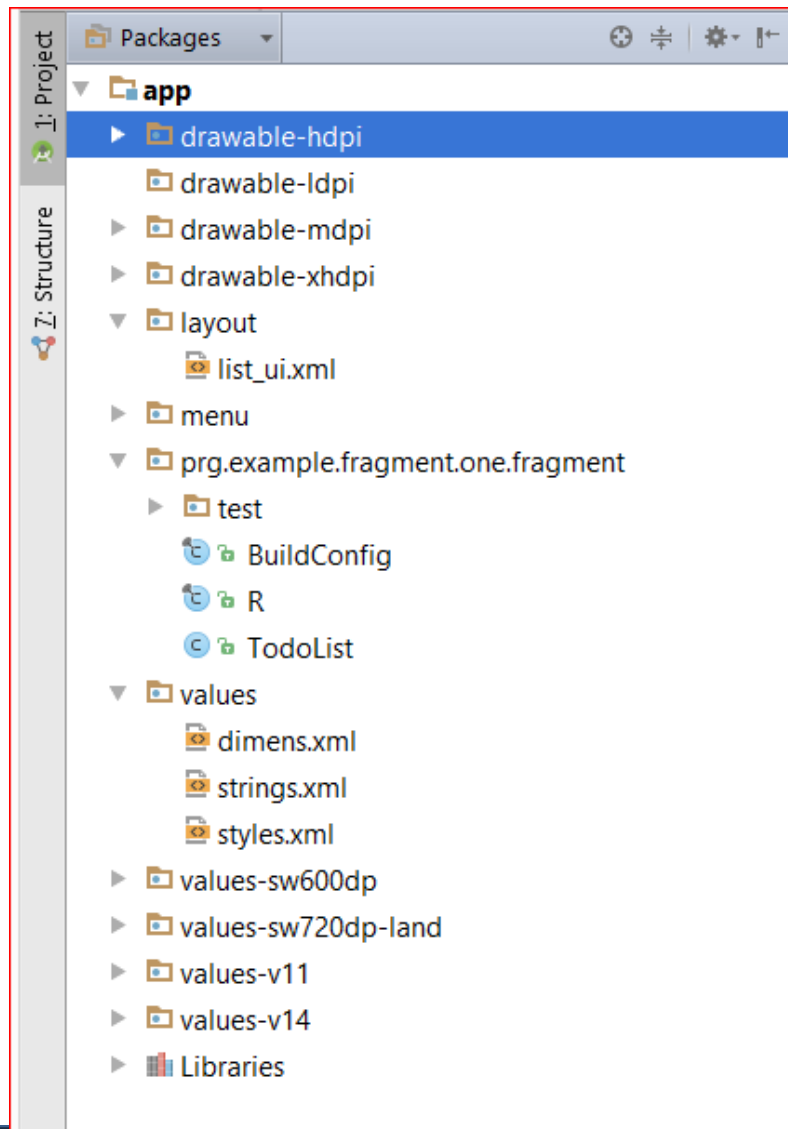
- Il faut un sdk : <https://developer.android.com/sdk/>
- Il vaut mieux un Ide
 - Sinon en ligne de commande...
 - Android Studio
 - 0.4.6 début mars 2014
 - 1.03 fin janvier 2015
 - 1.4 beta fin septembre 2015
 - 2.0 mars 2016
 - 2.2 fin septembre 2016
 - Ou Eclipse (plus ancien)
- un émulateur
 - Celui fournit dans le sdk / android studio
 - <http://www.genymotion.com/>
- Ou un dispositif
 - Éventuellement besoin de driver... (sur Windows)
 - Fichier avec les constructeurs sous linux : `rules.d/51-android.rules`

Exemple d'utilisation dans Android Studio

- Création d'une projet / une application / d'une activité
- Contenu d'un projet
 - Dans app/src/main/java : les classes java
 - Dans app/src/res, les ressources
 - Dans layout, la description de l'interface graphique
 - Dans values, les constantes
 - Des images...
 - Les lib (app/libs)
 - Dans build, ce qui est généré par le sdk (dont R)
 - manifest, etc.



Autres vues dans Android Studio



Interface définie en XML

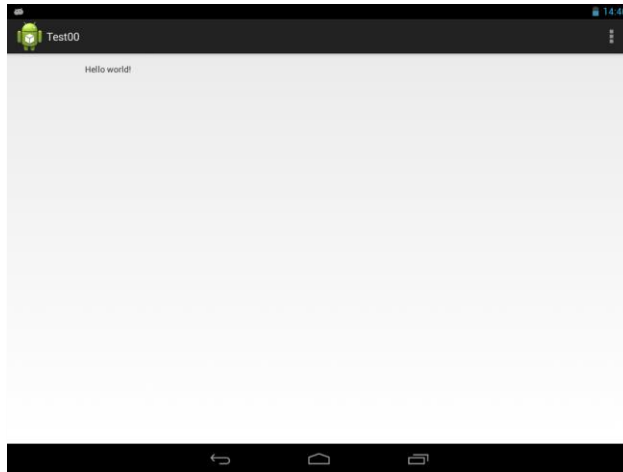
```
public class MainActivity extends Activity {
```

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState) {
```

```
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    View v = findViewById(R.id.relative);
    // ...
}
```

```
}
```



```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/relative"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:paddingTop="@dimen/activity_vertical_margin"
tools:context=".MainActivity" >
```

```
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/hello_world" />
```

```
</RelativeLayout>
```

Mais où est mon sysout ?

- Dans la console appelée LogCat
- Privilégier `Log.println(niveau, tag, message);`
 - Filtrage par niveau
 - Filtrage par tag
 - Lieux de tous les messages textuels

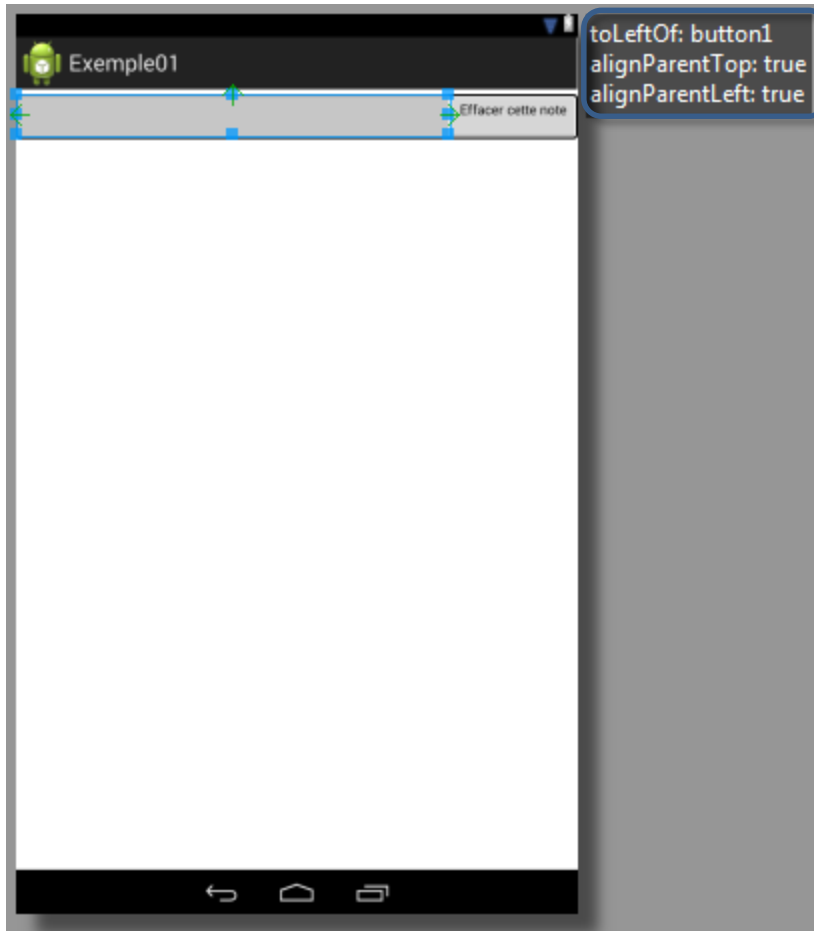
Autres outils

- Autres vues pour android :
 - Device (capture, stop)
- Autre outils : monitor (dans sdk)

Interface graphique avec android

Étape 1 : une seule activité,
exécution dans une même unité de
temps, pas de rotation...

Les éléments de l'interface graphique



toLeftOf: button1
alignParentTop: true
alignParentLeft: true

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:background="@drawable/bords"
android:layout_marginTop="5dp"
android:layout_marginBottom="5dp" >
  <Button
    android:id="@+id/button1"
    android:tag="action"
    style="?android:attr/buttonStyleSmall"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
    android:layout_alignParentTop="true"
    android:textAlignment="center"
    android:gravity="right"
    android:text="@string/remove" />

  <TextView
    android:id="@+id/textView1"
    android:tag="note"
    android:layout_width="wrap_content"
    android:layout_height="45dp"
    android:layout_alignParentLeft="true"
    android:layout_alignParentTop="true"
    android:layout_toLeftOf="@+id/button1"
    android:gravity="center_vertical|left"
    android:paddingLeft="5dp"
    android:background="#AAAAAAA" />
</RelativeLayout>
```

Ressources : string, dimension

- xml avec la racine `<resources>` et des enfants `<string>` ou `<dimen>`
 - Attribut name pour la désignation dans `"@string/un_name"` ou `"@dimen/un_autre_name"`

```
<string name="remove">Effacer cette note</string>
```

Ou

```
<dimen name="padding_liste_hor">10dp</dimen>
```

- dp (ou dip) : density-independent pixel
 - 1 dp = 1/160^{ième} d'un pouce (inch) sur l'écran
 - s'adapte à la résolution

Qualifiers

- <http://developer.android.com/guide/topics/resources/providing-resources.html>
- Définissent les noms des ressources (dossiers) et les variations possibles.
D'abord pour la lang (optionnel)
 - en / -fr / -fr-rFr / -fr-rCa / etc.puis, pour l'orientation
 - port / -landpuis, pour la résolution (screen pixel density)
 - ldpi / -mdpi / -hdpi / -xhdpi / etc.ou
 - w720dp (largueur minimal) / etc.ou...

Ressources : drawable

- Image
 - Ex : `android:icon="@drawable/ic_launcher"` pour le manifeste

- shape

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle"
    android:id="@+id/bords"
    >
    <corners
        android:topRightRadius="4dp"
        android:topLeftRadius="4dp"
        android:bottomRightRadius="4dp"
        android:bottomLeftRadius="4dp" />
    <stroke
        android:width="2dp"
        android:color="@android:color/black" />
</shape>
```


View : classe de base

- Brique de base de l'interface graphique
- [Pakage android.view](http://developer.android.com/reference/android/view/View.html)
- <https://developer.android.com/reference/android/view/View.html>
- Occupe un rectangle à l'écran (*draw* et *event*)
- Création (et ajout) par xml ou par code
- Appartenance à un arbre
 - Liste de propriétés
 - Parfois compliqué de retrouver l'équivalent du code par rapport aux attributs xml
 - Liées au type de View et au layout...
 - *Listeners* (chaque vue peut s'auto écouter)
 - Surcharge de `onTouchEvent` ou `setOnTouchListener`

View : classe de base

- id dans le xml (création, d'où le +)
 - Attribut de la balise
 - `android:id="@+id/my_button"`
 - Utilisation dans une activité :
`findViewById(R.id.my_button); // retourne une View, potentiellement à "caster"`
- Tag dans le xml (même principe que id)
 - `android:tag="action"`
 - Méthode `findWithTag("nom_de_tag");` de `View` (retourne une `View`)
- Taille (en pixel)
 - [`getMeasuredWidth\(\)`](#) and [`getMeasuredHeight\(\)`](#) pour la taille désirée
 - [`getWidth\(\)`](#) and [`getHeight\(\)`](#) pour la taille actuelle (peut être nulle...)
 - Il existe aussi une taille min et max
- Marge interne (padding) (en pixel)
 - [`setPadding\(int, int, int, int\)`](#)
 - [`getPaddingLeft\(\)`](#), [`getPaddingTop\(\)`](#), [`getPaddingRight\(\)`](#), [`getPaddingBottom\(\)`](#)
 - Différence entre le set et les get s'il y a des scrollbars
- Etc.
- Marge externe (margin) : dans `ViewGroup` (container-layout)

ViewGroup / Les layouts

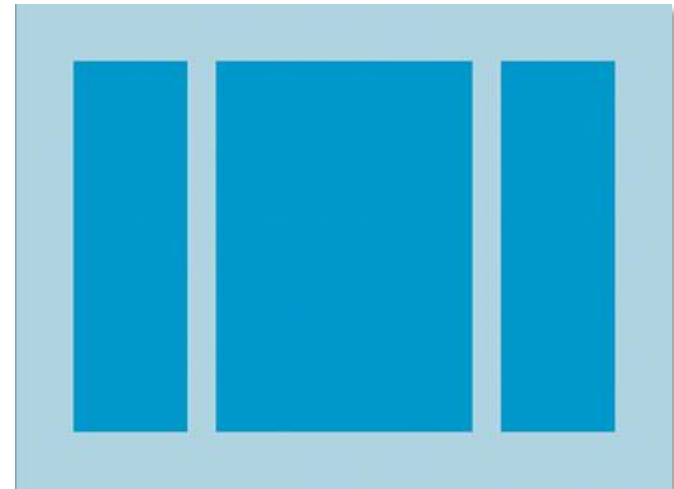
- Héritage de View
- Nœuds de l'arbre « graphique »
- Désigne à la fois un container et l'organisation à l'intérieur de celui-ci
- Création par code (new ...)
 - Méthode(s) `addView` dont :
 - [addView\(View child\)](#)
 - [addView\(View child, int index\)](#)
 - [addView\(View child, ViewGroup.LayoutParams params\)](#)
- Création dans le xml
 - Balise englobante
 - Méthode public void **setContentView** (int layoutResID) de la classe Activity
 - C.f. fragment

ViewGroup / Les layouts

- Chaque Layout à sa classe interne ViewGroup.LayoutParams
 - Instanciation à faire (c.f. plus loin)
 - Ou déduit du xml (et des attributs des balises des Views incluses)
 - De base : la dimension width et height
 - MATCH_PARENT (anciennement FILL_PARENT) : aussi grand que le parent, moins le padding
 - WRAP_CONTENT : sa taille désirée (+ padding)
 - Une dimension (généralement en dp)
 - + des propriétés en fonction du layout
- Position (location) d'une View (en pixel) : getTop(), getLeft(), getRight() et getBottom()
- Margin
 - <https://developer.android.com/reference/android/view/ViewGroup.MarginLayoutParams.html>

Linear Layout

- Alignement vertical ou horizontal
- Ajout dans l'ordre (indexation)



- `android:gravity`
- `android:orientation`
- `android:weightSum`
- `setGravity(int)`
- `setOrientation(int)`
- Φ
- Constante de [Gravity](#)
- Constante HORIZONTAL ou VERTICAL
- Le maxium de la somme des poids
- [android.widget.LinearLayout.LayoutParams](#)
 - [android:layout_gravity](#) : positionnement (centrage) du contenu de la view (enfant), plusieurs valeurs possibles (avec des |)
 - [android:layout_weight](#) : poids (un entier) donnés au vue pour se répartir l'espace exédant
 - si un seul à un poids, c'est le seul à grandir,
 - Sinon, répartition de l'espace au prorata des poids

Relative Layout



- Définition des positions les Views par rapport aux autres
 - Gravity
- <https://developer.android.com/reference/android/widget/RelativeLayout.LayoutParams.html>
 - Propriété « prédicat », par exemple : [android:layout_alignParentTop](#) (si vrai, le haut de la View est aligné sur le haut de relative layout) ;
ou
[android:layout_centerVertical](#) (si vrai, la View est centrée dans le relative layout)
 - Propriété qui fait référence à (id), par exemple, [android:layout_below](#) (pour positionner en dessous) ;
ou
[android:layout_toRightOf](#) (pour positionner à droite de)

Les autres layout

Layouts

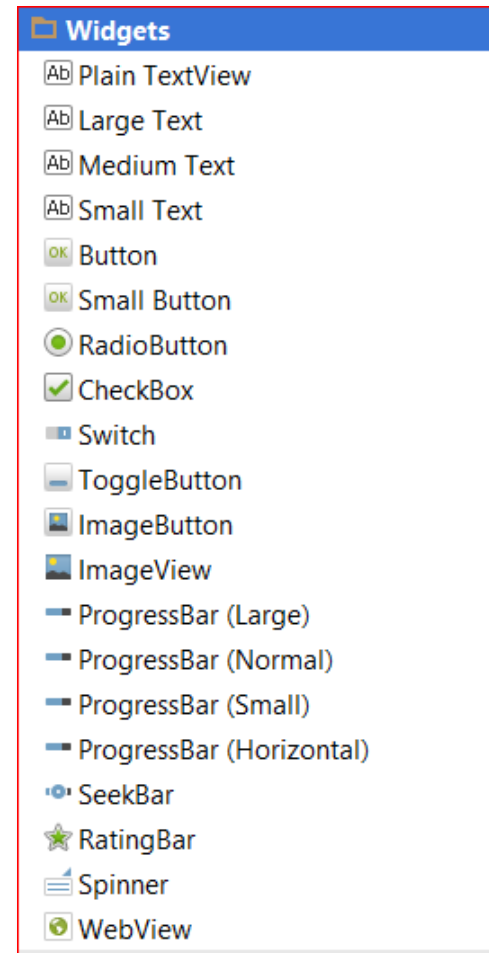
- FrameLayout
- LinearLayout (Horizontal)
- LinearLayout (Vertical)
- TableLayout
- TableRow
- GridLayout
- RelativeLayout

Containers

- RadioGroup
- ListView
- GridView
- ExpandableListView
- ScrollView
- HorizontalScrollView
- SearchView
- TabHost
- SlidingDrawer
- Gallery
- VideoView
- TwoLineListItem
- DialerFilter

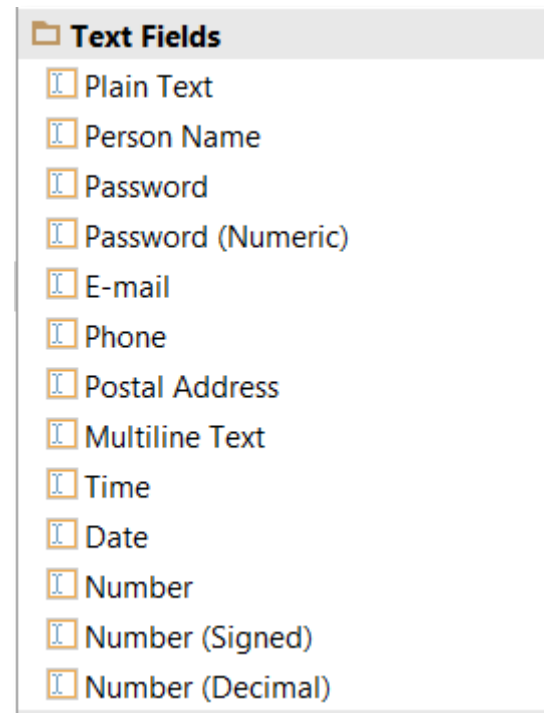
Les différents widgets : form

- Toutes une collections de boutons
- Button
 - Avec texte ou ImageButton ou les 2 avec un drawable par exemple à gauche du texte (android:drawableLeft="@drawable/image")
 - android:onClick permet de donner le nom d'une méthode m de signature m(View v) qui sera appelée lorsqu'on appuiera sur le bouton
- RadioButton
 - un RadioGroup (LinearLayout vertical par défaut)
 - Méthode avec une view en paramètre définie par android:onClick ou par setOnClickListener
 - boolean checked = ((RadioButton) view).isChecked();
 - switch(view.getId())
- SeekBar (slider) ...



Les différents widgets : text


- Différente variété avec différents claviers... sans autres vérifications
- Pour autocomplétion : <https://developer.android.com/reference/android/widget/AutoCompleteTextView.html>



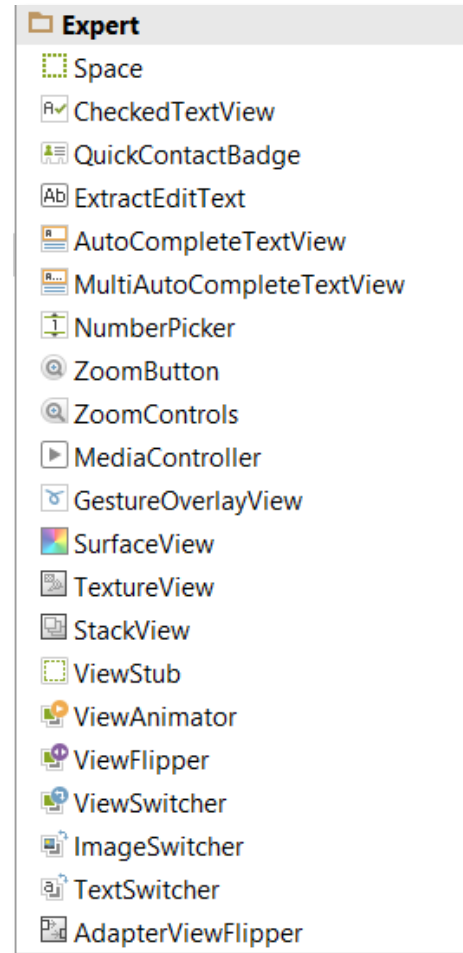
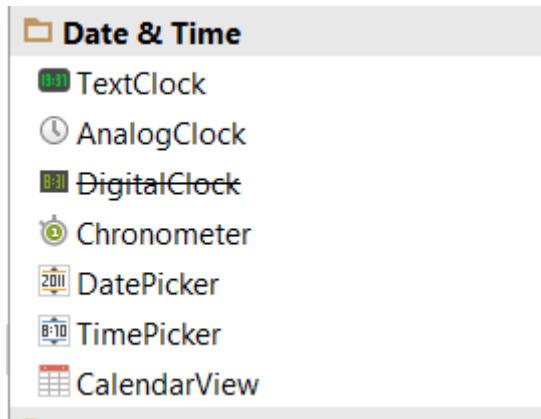
AutoComplétion

```
public class MainActivity extends Activity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_main);  
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,  
            android.R.layout.simple_dropdown_item_1line, COUNTRIES);  
        AutoCompleteTextView textView = (AutoCompleteTextView)  
            findViewById(R.id.autoCompleteTextView1);  
        textView.setAdapter(adapter);  
    }  
  
    private static final String[] COUNTRIES = new String[] {  
        "Belgium", "France", "Italy", "Germany", "Spain"  
    };  
}
```

Adapter avec une constante de R pour désigner le rendu de l'autocomplétion et un tableau de String



Les différents widgets...



Quelques événements

- Une seule méthode en callback
- Défini dans View : set_XXX_Listener
- onTouch() From [View.OnTouchListener](#).
 - This is called when the user performs an action qualified as a touch event, including a press, a release, or any movement gesture on the screen (within the bounds of the item).
- onClick() From [View.OnClickListener](#).
 - This is called when the user either touches the item (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses the suitable "enter" key or presses down on the trackball.
- onLongClick() From [View.OnLongClickListener](#).
 - This is called when the user either touches and holds the item (when in touch mode), or focuses upon the item with the navigation-keys or trackball and presses and holds the suitable "enter" key or presses and holds down on the trackball (for one second).
- onFocusChange() From [View.OnFocusChangeListener](#).
 - This is called when the user navigates onto or away from the item, using the navigation-keys or trackball.
- onKey() From [View.OnKeyListener](#).
 - This is called when the user is focused on the item and presses or releases a hardware key on the device. This is only useful for hardware keyboards; a software input method has no obligation to trigger this listener.
- onCreateContextMenu() From [View.OnCreateContextMenuListener](#).
 - This is called when a Context Menu is being built (as the result of a sustained "long click"). See the discussion on context menus in the [Menus](#) developer guide.

Event Handlers

- Méthodes de callback par défaut dans View
 - [onKeyDown\(int, KeyEvent\)](#) - Called when a new key event occurs.
 - [onKeyUp\(int, KeyEvent\)](#) - Called when a key up event occurs.
 - [onTrackballEvent\(MotionEvent\)](#) - Called when a trackball motion event occurs.
 - [onTouchEvent\(MotionEvent\)](#) - Called when a touch screen motion event occurs.
 - [onFocusChanged\(boolean, int, Rect\)](#) - Called when the view gains or loses focus.

Interception des événements

- Par surcharge
 - Appel à la méthode « super. »
- [Activity.dispatchTouchEvent\(MotionEvent\)](#) - This allows your [Activity](#) to intercept all touch events before they are dispatched to the window.
- [ViewGroup.onInterceptTouchEvent\(MotionEvent\)](#) - This allows a [ViewGroup](#) to watch events as they are dispatched to child Views.
- Refus de l'interception :
[ViewParent.requestDisallowInterceptTouchEvent\(boolean\)](#) - Call this upon a parent View to indicate that it should not intercept touch events with [onInterceptTouchEvent\(MotionEvent\)](#).

Dialogue : Toast

- Simple petit texte qui apparait et disparaît

```
Context context = getApplicationContext();
```

```
CharSequence text = "le message";
```

```
int duration = Toast.LENGTH_SHORT;
```

```
Toast toast = Toast.makeText(context, text, duration);
```

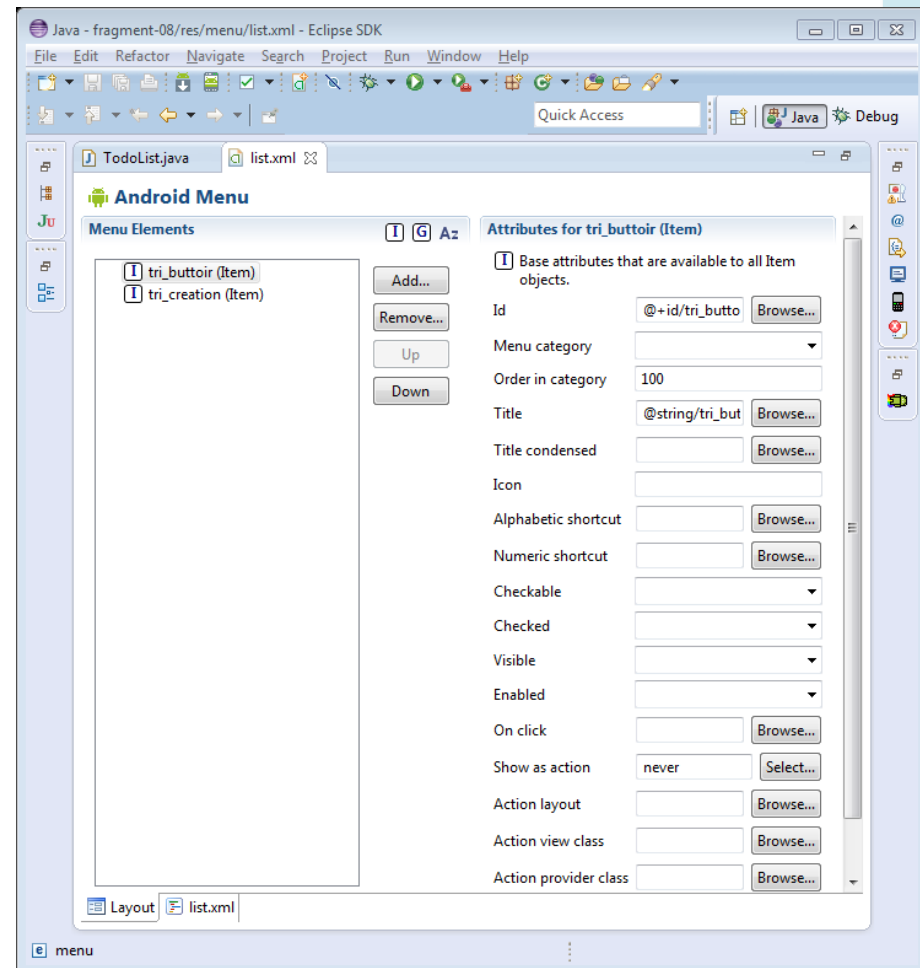
```
toast.show();
```

- Personnalisable

Menu et xml

- Dossier menu dans res

```
<menu
xmlns:android="http://schemas.android.com/apk/res/android" >
  <item
    android:id="@+id/tri_buttoir"
    android:orderInCategory="100"
    android:showAsAction="never"
    android:title="@string/tri_buttoir"/
  >
  <item
    android:id="@+id/tri_creation"
    android:orderInCategory="100"
    android:showAsAction="never"
    android:title="@string/tri_creation"
  />
</menu>
```



Menu en xml

- **<menu>**
Defines a [Menu](#), which is a container for menu items. A <menu> element must be the root node for the file and can hold one or more <item> and <group> elements.
- **<item>**
Creates a [MenuItem](#), which represents a single item in a menu. This element may contain a nested<menu> element in order to create a submenu.
- **<group>**
An optional, invisible container for <item> elements. It allows you to categorize menu items so they share properties such as active state and visibility. For more information, see the section about [Creating Menu Groups](#).
- Pour la balise item :
- **android:id**
A resource ID that's unique to the item, which allows the application can recognize the item when the user selects it.
- **android:icon**
A reference to a drawable to use as the item's icon.
- **android:title**
A reference to a string to use as the item's title.
- **android:showAsAction**
Specifies when and how this item should appear as an action item in the [action bar](#).

Menu et chargement du xml

- Méthode pour la création dans une Activity

`@Override`

```
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = getMenuInflater();  
    inflater.inflate(R.menu.nom_du_fichier_xml,  
menu);  
    return true;  
}
```

écouter le menu

- Toujours dans une Activity

`@Override`

```
public boolean onOptionsItemSelected(Menuitem item) {  
    // Handle item selection  
    switch (item.getItemId()) {  
        case R.id.un_id:  
            // des actions...  
            return true;  
        case R.id.un_autre_id:  
            // des actions en réponse à cet item là  
            return true;  
        // et ainsi de suite  
        default:  
            return super.onOptionsItemSelected(item);  
    }  
}
```

Evolution des vues dans une même activité

Introduction aux fragments et à la personnalisation des Views

Un exemple : faire une liste

- En rotation bloquée
- Un EditText, un Button, un LinearLayout
- Ajout d'un écouteur d'événement

- Pour effacer ou ajouter un item interactif ?
 - Une sous zone : View personnalisée (extends)
 - Une sous zone : avec fragment

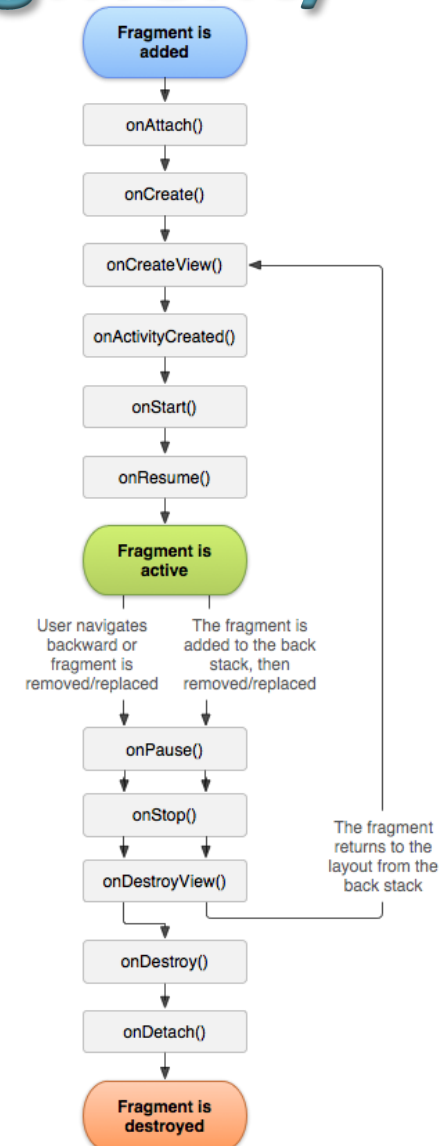
View personnalisée (extends)

- problème pour gérer les paramètres graphiques (trouver l'équivalent xml)
- Problème pour les tailles
 - Pas encore de taille, donc `getMeasuredWidth`
 - Certaines tailles ne sont pas déterminables (facilement)
- Problème pour encadrer
 - Création d'une Shape
 - Code différent selon cible

```
int sdk = android.os.Build.VERSION.SDK_INT;
if(sdk < android.os.Build.VERSION_CODES.JELLY_BEAN) {
    setBackgroundDrawable(bords);
} else {
    setBackground(bords);
}
```

View personnalisée (fragment)

- Cycle de vie des fragments
- Pas fait uniquement pour le graphique (connexion avec web service...)
- Ici montrer avec intégration côté programmation, mais intégration possible en XML



Faire un Fragment

Partie java

```

public class Item extends Fragment {

    public void onCreate(Bundle savedInstanceState) { /* [...] */ }
    public View onCreateView(LayoutInflater inflater,
        ViewGroup container, Bundle savedInstanceState)
        { /* [...] */ }

    public void onStart() {
        super.onStart();
        /* [...] */
    }

    public static Item newInstance(...) { /* [...] */ }
    public void onSaveInstanceState(Bundle outState) { }

}

```

Partie XML

```

<RelativeLayout      [...]
    android:background="@drawable/bords">
    <TextView
        android:id="@+id/textView1"
        [...]
        android:tag="note" />
    <Button
        android:id="@+id/action"
        android:tag="action"
        [...] />
    <ImageButton
        android:id="@+id/up"
        [...]
        android:tag="up" />
    <ImageButton
        android:id="@+id/down"
        [...]
        android:tag="down" />
    <TextView
        android:id="@+id/dates"
        [...]
        android:tag="dates" />
</RelativeLayout>

```


Exemple de Fragment

```
public View onCreateView(LayoutInflater inflater,  
                        ViewGroup container, Bundle savedInstanceState)
```

```
Bundle args = getArguments();  
this.note = args.getString("txt");  
this.chargee = inflater.inflate(R.layout.item,  
container, false);
```

```
/* [...] */  
chargee.setTag("item_"+myNumb);
```

```
View noteView =  
chargee.findViewById("note");  
/* [...] */
```

```
return chargee;
```

```
public void onCreate(Bundle  
saved)
```

```
super.onCreate(saved);  
if (saved == null) {  
    myNumb =  
calendrier.getTime().getTime() ;  
}  
else {  
    myNumb =  
saved.getLong("creation");  
/* [...] */  
}
```

Exemple de Fragment

```
public static Item newInstance(String txt, Date dateButtoir)
```

```
Item fragment = new Item();  
Bundle args = new Bundle();
```

```
args.putString("txt", txt);
```

```
args.putSerializable("dateButtoir", dateButtoir);
```

```
fragment.setArguments(args);  
return fragment;
```

```
public void onSaveInstanceState(Bundle outState)
```

```
super.onSaveInstanceState(outState);
```

```
outState.putLong("creation", myNumb);
```

Chargement fragment par programmation

- FragmentTransaction pour déterminer les modifications
- Un commit qui met fin à la transaction
- (en faire une nouvelle pour ajouter à nouveau...)

- Échange de paramètre avec un Bundle (une map)
- Chargement d'un layout / xml avec la méthode inflate

Gestion des fragments

- Généralement pas besoin
- Sinon, il faut conserver des listes et faire l'association n° de fragment avec la view correspondante
 - utilisation de tag
 - interface logicielle pour la liste
 - À refaire à chaque rotation, à chaque « retour », etc.

Faire sa propre view (dont le dessin...)

onDraw, Canvas, Paint

Les méthodes à surcharger (en partie)

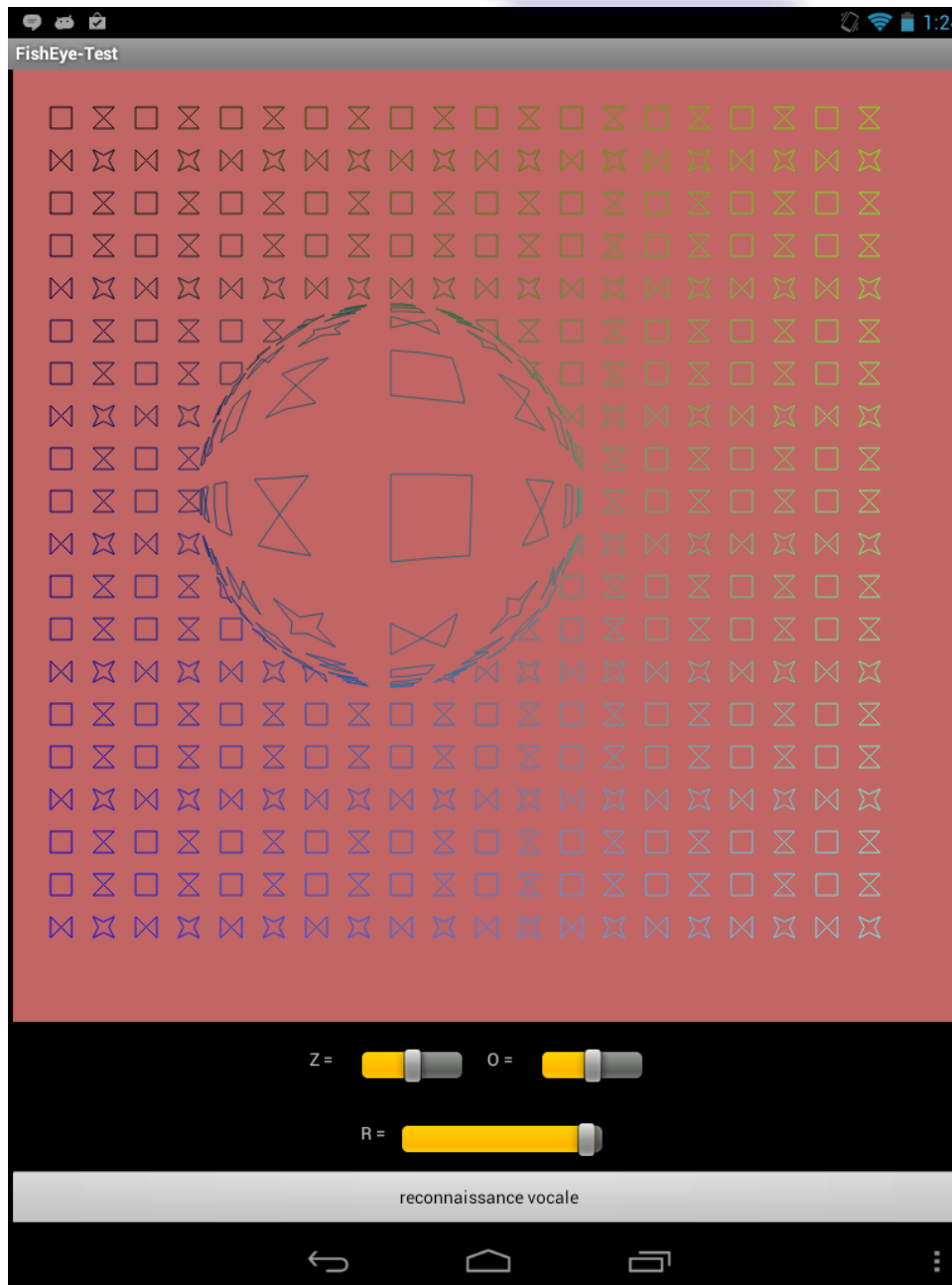
Creation	Constructors View(Context context) View(Context context, AttributeSet attrs)	There is a form of the constructor that are called when the view is created from code and a form that is called when the view is inflated from a layout file. The second form should parse and apply any attributes defined in the layout file.
	onFinishInflate()	Called after a view and all of its children has been inflated from XML.
Layout	onMeasure(int, int)	Called to determine the size requirements for this view and all of its children.
	onLayout(boolean, int, int, int, int)	Called when this view should assign a size and position to all of its children.
	onSizeChanged(int, int, int, int)	Called when the size of this view has changed.
Drawing	onDraw(android.graphics.Canvas)	Called when the view should render its content.
Event processing	onKeyDown(int, KeyEvent)	Called when a new hardware key event occurs.
	onKeyUp(int, KeyEvent)	Called when a hardware key up event occurs.
	onTrackballEvent(MotionEvent)	Called when a trackball motion event occurs.
	onTouchEvent(MotionEvent)	Called when a touch screen motion event occurs.
Focus	onFocusChanged(boolean, int, android.graphics.Rect)	Called when the view gains or loses focus.
	onWindowFocusChanged(boolean)	Called when the window containing the view gains or loses focus.
Attaching	onAttachedToWindow()	Called when the view is attached to a window.
	onDetachedFromWindow()	Called when the view is detached from its window.
	onWindowVisibilityChanged(int)	Called when the visibility of the window containing the view has changed.

Méthode onDraw

```
Paint paint = new Paint(Paint.ANTI_ALIAS_FLAG);
```

```
public void onDraw(Canvas g) {
    // ...
    paint.setColor(0x99990000);
    g.drawRect(0, 0, mySize[0], mySize[1], paint); // on remplit un rectangle couleur "saumon"

    // ...
    // elts est une liste de Polygon, soit une liste de points et une couleurs
    // on trace un polygone... en reliant tous les sommets et en rebouclant
    for(MyPolygon p : elts) {
        paint.setColor(p.color);
        float [] pts = p.getPoints();
        for(int i = 0; i<pts.length-3; i=i+2) {
            g.drawLine(pts[i]+marges, pts[i+1]+marges, pts[i+2]+marges,
pts[i+3]+marges, paint);
        }
        g.drawLine(pts[pts.length-2]+marges, pts[pts.Length-1]+marges, pts[0]+marges,
pts[1]+marges, paint);
    }
}
```



Le dessin : un pinceau et un canvas

- <https://developer.android.com/reference/android/graphics/Paint.html>
 - Création une fois pour toute
 - Avec des paramètres comme [ANTI_ALIAS_FLAG](#)
- <https://developer.android.com/reference/android/graphics/Canvas.html>
 - Pour dessiner...

Intégration au xml

- Constructeur (pour aperçu) :
 - (Context context, AttributeSet attrs)
- Balise <nom.qualifié>
- Attributs
 - values/attrs.xml

```
<declare-styleable name="DeformableImage">  
    <attr name="image" format="reference" />  
    <attr name="correction" format="boolean" />  
</declare-styleable>
```

Exemple de XML avec une View

« custom »

```
<fr.unice.reneviergonin.fisheye01.images.DeformableImage
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/deformed"
    android:layout_alignParentTop="true"

    style="@style/fishview"

    app:image="@drawable/charlie"
    app:correction="true"
/>
```

- Nom qualifié (complet) de la View
- Attributs standards
- Attributs définis dans une ressources (paramétrables)

- Attributs définis dans l'application avec un usage :

```
public DeformableImage(Context context, AttributeSet attrs) {
    super(context, attrs);

    TypedArray attributes = context.obtainStyledAttributes(attrs, R.styleable.DeformableImage);
    int img_id = attributes.getResourceId(R.styleable.DeformableImage_image, R.drawable.tab);
    corrected = attributes.getBoolean(R.styleable.DeformableImage_correction, true);
    // etc...
}
```

Activity (1)

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

Cycle de vie

```
public class Activity extends
ApplicationContext {
    protected void onCreate(Bundle
savedInstanceState);
```

```
    protected void onStart();
```

```
    protected void onRestart();
```

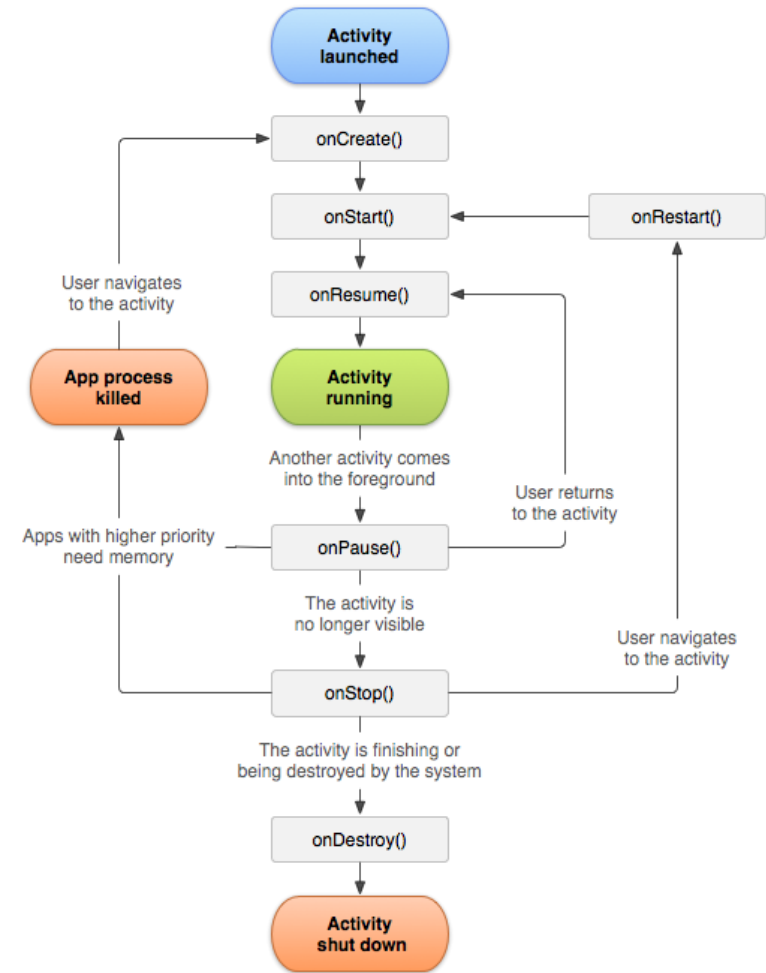
```
    protected void onResume();
```

```
    protected void onPause();
```

```
    protected void onStop();
```

```
    protected void onDestroy();
```

```
}
```



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

Rotation et bundle

- **Caution:** "Your activity will be destroyed and recreated each time the user rotates the screen. When the screen changes orientation, the system destroys and recreates the foreground activity because the screen configuration has changed and your activity might need to load alternative resources (such as the layout)." <http://developer.android.com/training/basics/activity-lifecycle/recreating.html>
- Re-cr  er => refaire des objets...
- Pour les fragments, on s'en sort car il y a le param  tres (et recrer avec ce m  me param  tre)
- Des fonctions peuvent   tre appel  es avant destruction
 - onSaveInstanceState pour une Activit  
 - onSaveInstanceState pour un fragment
 - on peut y remplir le bundle
 - Ne pas oublier l'appel    super.xxx
- M  thode pour utiliser ce qui est sauv  
 - onCreate(Bundle saved) pour un fragment
 - onRestoreInstanceState(Bundle inState) pour une activit  
- A faire   voluer... en m  me temps que le code...

Principe de la Sauvegarde/Lecture Locale

- Ouverture de flux :
 - Méthode de la classe Context
 - FileInputStream avec `openFileInput(Nom)`
 - FileOutputStream avec `openFileOutput(Nom, Mode)`
 - `Mode = Context.MODE_PRIVATE` (écrasement)
 - `Mode = Context.MODE_APPEND` (concaténation)
 - `/data/data/<app>/files`
- *Flux usuels de java*

Tip: If you want to save a static file in your application at compile time, save the file in your project `res/raw/` directory. You can open it with `openRawResource()`, passing the `R.raw.<filename>` resource ID. This method returns an `InputStream` that you can use to read the file (but you cannot write to the original file). [http://developer.android.com/]

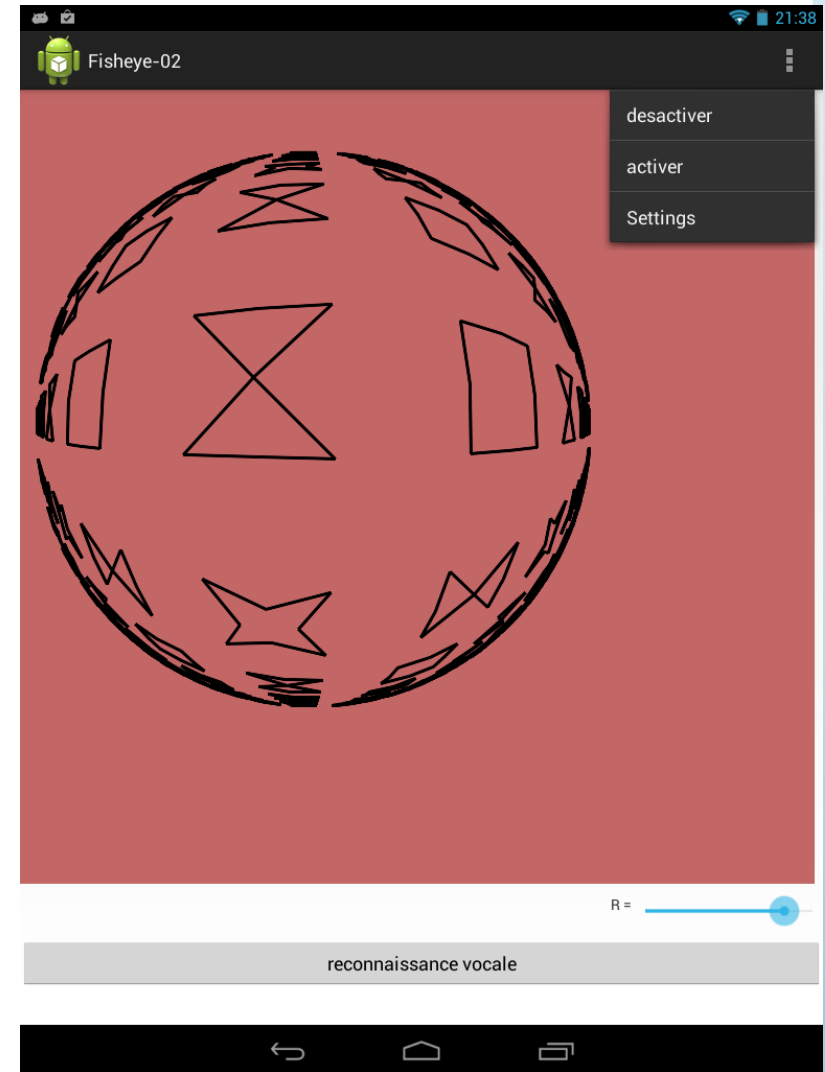
Sauvegarde/Lecture « publique »

- Permission :
 - `<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />`
 - `READ_EXTERNAL_STORAGE`
- `context.getExternalFilesDir(null)`
 - [http://developer.android.com/reference/android/content/Context.html#getExternalFilesDir\(java.lang.String\)](http://developer.android.com/reference/android/content/Context.html#getExternalFilesDir(java.lang.String))
 - *Starting in KITKAT, no permissions are required to read or write to the returned path; it's always accessible to the calling app. This only applies to paths generated for package name of the calling application. To access paths belonging to other packages, WRITE_EXTERNAL_STORAGE and/or READ_EXTERNAL_STORAGE are required.*
- `new File(ctx.getExternalFilesDir(null), "nomdufichier")`
 - `/sdcard/Android/data/pakage.de.application/files/nomdufichier`

Présentation d'un exemple

- Fisheye, exemple de personnalisation d'une View
 - Intégration dans le xml

```
<fisheye.four.image.DeformablePolygons [...] />
```
 - Surcharge de onDraw, utilisation de Paint
- Mise en place d'un menu « simple »
 - inflate du menu dans onCreateOptionsMenu
 - onOptionsItemSelected pour réagir aux actions
 - Utilisation de invalidate() sur la vue pour la mettre à jour



Un autre écran pour les paramètres

- Autre écran = autre activité
- Attention au cycle de vie (détruit, pas détruit ?)
- Création d'une autre activité
- Déclaration dans le manifest (après l'activité principale)

```
<activity
  android:name="fisheye.four.SettingsActivity"
  android:label="@string/title_activity_settings"
  android:parentActivityName="fisheye.four.Fisheye">
  <meta-data
    android:name="android.support.PARENT_ACTIVITY"
    android:value="fisheye.four.Fisheye" />
</activity>
```

Lancer une activité depuis une autre

- Méthode simple : `startActivity(Intent i)` dans Activity
- Intent
 - Objet de communication avec l'OS
 - Dans ce cas, constructeur `new Intent(Context, Class)`
 - Context = activité qui lance
 - Class = le type de l'activité à lancer
 - `putExtra` pour ajouter des données
- Dans le `onCreate` de la nouvelle activité
 - `getIntent()`
 - `getIntent().getIntExtra("clef", val_defaut);`

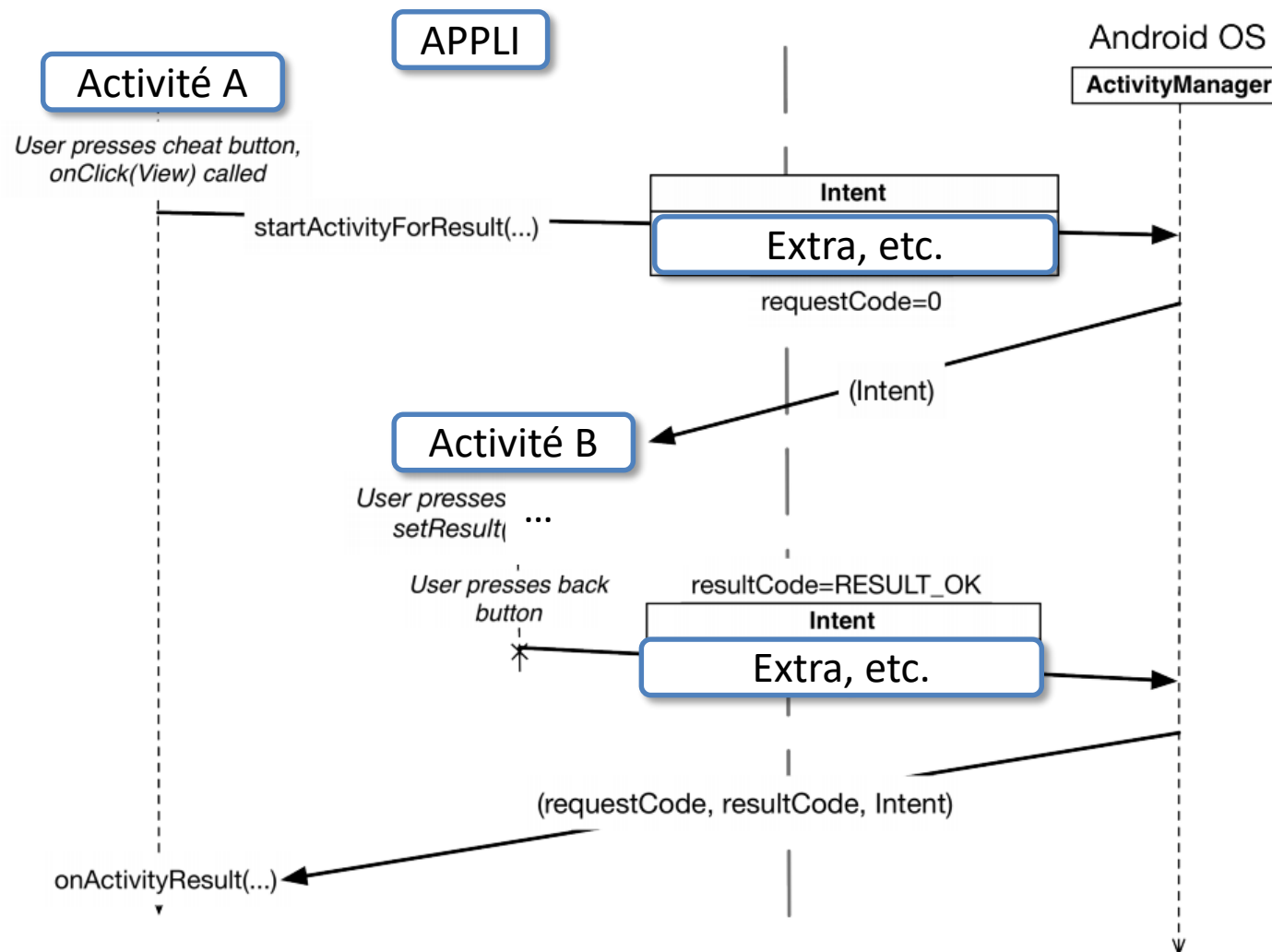
Retour de données

- Lancement avec `startActivityForResult`
- Méthode `setResult(code, intent)`
 - Avec un code de retour
 - Avec un Intent (et des extra)
- Méthode de retour dans l'activité qui a lancé

```
protected void onActivityResult(int requestCode, int  
resultcode, Intent data) {  
    if (data == null) return ;  
    Date d = (Date)  
data.getSerializableExtra("date");  
    // etc.  
}
```

android programming, the big nerd ranch guide

Cycle de message



Bouton « home » depuis 3.x

Affichage du bouton (dans la barre)

```
if (Build.VERSION.SDK_INT >=
    Build.VERSION_CODES.HONEYCOMB) {
    getActionBar().setDisplayHomeAsUpEnabled(true);
}
```

Récupérer l'événement

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home: // etc. }
    return true; }
}
```

Deux fins possibles

- Appel à `setResults`
 - Faire l'Intent de retour (`putExtra` pour les valeurs de retour)
 - Quitter la vue avec `finish`
 - Pas dans l'esprit
 - Mais l'avantage de conserver l'instance de l'activité...
- Utilisation du « up »
 - L'activité de départ repart quasiment de zéro
 - Perte du **`onActivityResult`**

Navigation « up » (on remonte à la tâche parent)

Voir : <https://speakerdeck.com/jgilfelt/this-way-up-implementing-effective-navigation-on-android>

```
// pour « relancer » l'activité parent
```

```
Intent up = NavUtils.getParentActivityIntent(this);
```

```
// éventuellement des valeurs de retours comme extra
```

```
up.putExtra(...);
```

```
// retour à l'activité, qui repasse par onCreate, mais  
qui
```

```
// n'a pas été tuée (pas de onRestoreInstanceState)
```

```
NavUtils.navigateUpTo(this, up);
```

```
// s'il n'y a pas de paramètre
```

```
// NavUtils.navigateUpFromSameTask(this);
```


Tester s'il faut relancer l'activité

- <http://developer.android.com/training/implementing-navigation/ancestral.html>

```
if (NavUtils.shouldUpRecreateTask(this, up)) {  
    // This activity is NOT part of this app's task, so create a new  
task  
    // when navigating up, with a synthesized back stack.  
    TaskStackBuilder.create(this)  
        // Add all of this activity's parents to the back stack  
        .addNextIntentWithParentStack(up)  
        // Navigate up to the closest parent  
        .startActivities();  
} else {  
    // This activity is part of this app's task, so simply  
    // navigate up to the logical parent activity.  
    NavUtils.navigateUpTo(this, up);  
}
```

Tuer une activité ? Vider la pile ?

- Pour « tuer »
 - « Normalement réserver à l'OS »
 - `finish()` (+`System.exit(0)`)
 - Etc.
- Pour relancer l'application au début à partir de là où on est :
 - Dans le manifest, pour l'activité ciblée
`android:launchMode="singleTop"`
 - Depuis l'endroit où on veut revenir « au début »

```
Intent intent = new Intent(this, LActivitéPrincipale.class);
intent.addFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
startActivity(intent);
```

Retour à l'exemple : utilisation d'un service

- Reconnaissance de parole de google
- Intégrée à Android

```
private static final int VOICE_RECOGNITION_REQUEST_CODE = 1234;
/**
 * Fire an intent to start the speech recognition activity.
 */
@Override
public void onClick(View v) {
    // lancement de l'intent avec le nom de la classe... (constante)
    Intent intent = new Intent(RecognizerIntent.ACTION_RECOGNIZE_SPEECH);
    // paramètre requis pour interprétation du résultat...
    intent.putExtra(RecognizerIntent.EXTRA_LANGUAGE_MODEL,
        RecognizerIntent.LANGUAGE_MODEL_FREE_FORM);
    // titre
    intent.putExtra(RecognizerIntent.EXTRA_PROMPT, "Fisheye - Speech recognition");
    // VOICE_RECOGNITION_REQUEST_CODE pour retrouver l'appel pour le résultat
    // nombre choisi par le développeur
    startActivityForResult(intent, VOICE_RECOGNITION_REQUEST_CODE);
}
```

Retour de plusieurs activités...

```
protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
```

```
// test si le retour est ok et si le retours vient bien de le reco vocale
```

```
    if (requestCode == VOICE_RECOGNITION_REQUEST_CODE &&
resultCode == RESULT_OK) {
```

```
        ArrayList<String> matches =
data.getStringArrayListExtra(RecognizerIntent.EXTRA_RESULTS);
```

```
        // [...]
```

```
    }
```

```
    else {
```

```
        // un autre cas...
```

```
    }
```

```
}
```

Android & Thread, des actions qui prennent du temps

Voir les thread en java indépendant d'Android

<http://docs.oracle.com/javase/tutorial/essential/concurrency/>

<http://developer.android.com/guide/components/processes-and-threads.html>

Certaines actions prennent du temps, elles figeraient l'application (UI).

Certaines actions sont interdites (car longues ou bloquantes) dans l'application

AsyncTask

Handler & HandlerThread, CountdownTimer

AsyncTask par l'exemple

- Utilisation de « worker » (AsyncTask)
- Exemple : adresse vers coordonnées (nominatim) puis image (openstreetmap)
- V1 : l'intention, qui ne fonctionne pas
- V2 : une AsyncTask pour l'adresse
- V3 : évolution pour charger aussi l'image

Faire une requête sur le web

- Classes fournis dans java.net

```
URL siteUrl = new URL(url);
```

```
URLConnection conn = siteUrl.openConnection();
```

```
// on peut vouloir suivre les redirections...
```

```
// HttpURLConnection.setFollowRedirects(true);
```

```
// ensuite on a l'InputStream... conn.getInputStream()
```

```
// à encapsuler selon ce qu'on reçoit
```

```
// par exemple du JSON / text / etc.
```

```
BufferedReader in = new BufferedReader(new InputStreamReader( conn.getInputStream(), "UTF-8"));
```

```
// lecture tant qu'il y a à lire
```

```
while ((line = in.readLine()) != null) { file += line; }
```

```
// à la fin, file (une String) contient le fichier distant
```

```
in.close();
```

- Alternative : package d'apache intégré dans android
 - <http://developer.android.com/reference/org/apache/http/package-summary.html>
 - package org.apache.http

Mais il faut demander l'autorisation d'utiliser le réseau

```
<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
    package="adresseverscoordonnees.three"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="18" />

    <uses-permission
        android:name="android.permission.INTERNET" />

    <application ...
```


Parser JSON

<http://developer.android.com/reference/org/json/package-summary.html>

```
// on reçoit un json qui contient un tableau d'objet
```

```
JSONArray fileJson= new JSONArray(file);
```

```
// si le tableau n'est pas vide
```

```
if ((fileJson != null) && (fileJson.length() > 0)) {
```

```
    // on prend le premier élément, qui est supposé être un  
    objet
```

```
    JSONObject adresseRetournee = (JSONObject) fileJson.get(0);  
    result = new Object[2];
```

```
    // on retrouve la propriété « lat » de cette objet
```

```
    result[0] = (String) adresseRetournee.get("lat") ;
```

```
    // etc.
```

```
}
```

Mais la connexion, elle prend combien de temps ?

- Impossible à savoir...
- Risque de figer l'interface graphique (et l'application)...
- ... mais de toutes façons, depuis 3.x, il est interdit de faire certaines choses dans le thread graphique...
- Donc, il faut faire autrement
 - Il faut un thread,
 - qui puisse faire des choses dans le graphique, plus facilement que la méthode `runOnUiThread(Runnable)` de la classe `Activity`
 - bref qui ressemble aux (Swing)Worker

AsyncTask

- <http://developer.android.com/reference/android/os/AsyncTask.html>
- AsyncTask enables proper and easy use of the UI thread. This class allows to perform background operations and publish results on the UI thread without having to manipulate threads and/or handlers.
- AsyncTask is designed to be a helper class around [Thread](#) and [Handler](#) and does not constitute a generic threading framework. AsyncTasks should ideally be used for short operations (a few seconds at the most.)
- If you need to keep threads running for long periods of time, it is highly recommended you use the various APIs provided by the `java.util.concurrent` package

AsyncTask<Param, Progress, Result>

- An asynchronous task is defined by a computation that runs on a background thread and whose result is published on the UI thread.
- An asynchronous task is defined by 3 generic types, called Params, Progress and Result, and 4 steps

Exemple

```

public class Tache extends AsyncTask<String, String, Object[]> {
// [...] // attributs de la classe
public Tache(TextView lat, TextView lon, ImageView img) {
// constructeur pour obtenir des éléments
// par exemple graphique, pour mise à jour..
}

@Override
protected Object[] doInBackground(String... params) { // lancer par execute()
// tester les paramètres
if ((params == null) || (params.length != 2)) return null;

Object [] result = null;
String url = params[0];
// [...] // utilise publishProgress("<string>");
return result;
}

@Override
protected void onProgressUpdate(String... result){
// [...] // pour montrer la progression // appelé par publishProgress
}

@Override
protected void onPostExecute(Object [] res){
// traitement des résultats... // [...] // appelé à la fin de doInBackground
}
}

```

Vie d'une AsyncTask<Pa,Pr,R>, des garanties sur l'ordre

- [Création \(new\)](#)
- [Appel à execute\(\)](#)
 1. [onPreExecute\(\)](#), invoked on the UI thread before the task is executed. Setup / Graphical initialization (progress bar)
 2. [doInBackground\(Params...\)](#), invoked on the background thread immediately after [onPreExecute\(\)](#). This step is used to perform background computation that can take a long time. The result of the computation must be returned by this step and will be passed back to the last step.
 3. [onProgressUpdate\(Progress...\)](#), invoked on the UI thread after a call to [publishProgress\(Progress...\)](#) from [doInBackground](#). The timing of the execution is undefined. For user feedback.
 4. [onPostExecute\(Result\)](#), invoked on the UI thread after the background computation finishes. The result of the background computation is passed to this step as a parameter.

Utilisation d'une ASYNCTASK

- Cancel : [cancel\(boolean\)](#).
 - [isCancelled\(\)](#) return true.
 - [onCancelled\(Object\)](#) instead of [onPostExecute](#)
 - checking [isCancelled\(\)](#) periodically in [doInBackground\(Object\[\]\)](#)
- The AsyncTask class must be loaded on the UI thread.
- The task instance must be created on the UI thread.
- [execute\(Params...\)](#) must be invoked on the UI thread.
- Do not call [onPreExecute](#), [onPostExecute](#), [doInBackground](#), [onProgressUpdate](#) manually.
- The task can be executed only once

AsyncTask : un seul thread en fond

- Starting with [HONEYCOMB](#), tasks are executed on a single thread to avoid common application errors caused by parallel execution.
- If you truly want parallel execution, you can invoke [executeOnExecutor\(java.util.concurrent.Executor, Object\[\]\)](#) with [THREAD POOL EXECUTOR](#).

Android et les threads

- Similaire à java
 - <http://developer.android.com/guide/components/processes-and-threads.html>
 - Pensez à onResume / onPause et onStart / onStop, etc.
- Ou lié à des services
 - <http://developer.android.com/guide/components/services.html>

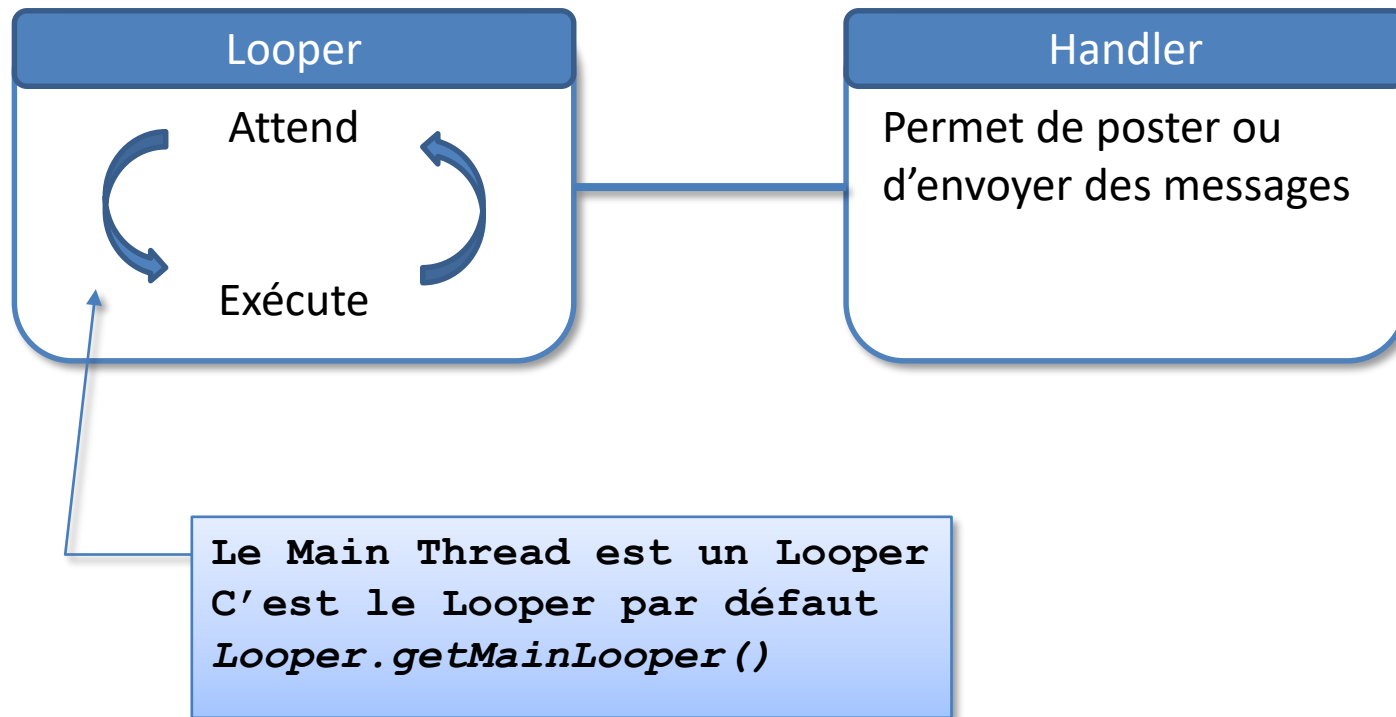
Retour sur l'exemple

- Tache a besoin
 - D'éléments graphiques pour les mettre à jour
 - D'une url, d'un message par défaut
- Tache retourne
 - Des coordonnées (latitude, longitude)
 - Une image (enfin, si c'est ok)
- Publication de l'avancement : du texte

Dans le DoInBackground et ensuite...

- On recherche un json
 - C.f; exemple ci-dessus
- On recherche une image
 - Création d'une Bitmap à partir d'un InputStream
- Pour les résultats
 - Appels à des setters
 - Mise à jour avec invalidate()
 - Masquer / montrer l'image

Handler & Looper



Handler, post

- Handler handler = new Handler()
 - En paramètre, un Looper est possible, c.f. HandlerThread
 - Par défaut Looper = Main Thread
- Utilisable « de partout »
 - Exécution sur le Looper Thread
 - Par défaut sur le Main Thread
- handler.post(*Runnable*)
 - postAtFrontOfQueue
 - postDelayed avec un second paramètre un délai (long, en ms)
 - postAtTime avec un second paramètre un temps (long, en ms)
 - Date, System.uptimeMillis()+ ...
- handler.removeCallbacks(r)
 - Garder une référence vers le Runnable r
 - Si pas encore exécuté, pas lancé ; sinon, c'est trop tard

Handler, send (1)

- Plutôt que poster des Runnable, envoyer des messages
 - Tâches connues à l'avance
 - Economie en mémoire (moins d'objet)
- Message
 - <http://developer.android.com/reference/android/os/Message.html>
 - Des champs publics (facile d'accès)
 - Avec un « id » : message.what
 - Avec des données
 - get/setData => un [Bundle](#)
 - ou int .arg1 / .arg2 et Object obj
- Création par fabrique
 - Message.obtain
 - avec différents paramètres : handler (pour le champ replyTo), what, arg1, arg2, obj
 - Ex: Message.obtain(handler, 42, "réponse à la grande question sur la vie, l'univers et le reste");

Handler, send (2)

- Pour envoyer `sendMessage`
 - `sendMessage`
 - `sendMessageAtFrontOfQueue`
 - `sendMessageDelayed` avec un second paramètre un délai (long, en ms)
 - `sendMessageAtTime` avec un second paramètre un temps (long, en ms)
- Variante avec `sendEmptyMessage(int what)`
- Pour annuler (si pas encore traité) : `removeMessage(int what)`
 - Car message recyclé dans la fabrique

Handler, send (3)

traiter les messages

- Délégation à un Handler.Callback
 - [handleMessage\(Message msg\)](#)
 - En paramètre du constructeur du Handler (seul ou après le looper)
- Surcharge de la méthode handleMessage(Message msg)
 - Appelé par défaut par le Looper
 - Appel par défaut (si pas surchargée) le Handler.Callback
 - Déclaration en classe à part entière (ou en classe static) pour éviter un nettoyage mémoire intempestif
- Possible de combiner les deux
 - dans la méthode surchargée handleMessage, appel à super.handleMessage

Handlers Intégrés

- Méthode post de View
 - `view.post(new Runnable() { /* ... */ });`
- Méthode `runOnUiThread` de Activity

Avoir son propre Looper :

HandlerThread

- HandlerThread fournit un Looper (caché)

```
// création du HandlerThread, avec un priorité basse pour ne
// pas surcharger le système
HandlerThread encapsuleLooper =
new HandlerThread("un nom", Process.THREAD_PRIORITY_BACKGROUND);
encapsuleLooper.start(); // obligatoire !

// création du handler
Handler handler = new Handler( encapsuleLooper.getLooper() );
// les posts et messages seront traité sur encapsuleLooper
```

CountDownTimer

- Classe Abstraite à compléter
 - onTick
 - onFinish
- Constructeur CountdownTimer(**long** millisInFuture, **long** countdownInterval)
 - millisInFuture : délai en ms pour la fin
 - countdownInterval : délai en ms pour évolution intermédiaire
- Utilise un handler
 - Sur le thread « principal »
 - Utilise sendMessageDelayed
 - Appel selon le temps restant
 - onTick (**long** millisUntilFinished) s'il reste du temps
 - onFinish() à la fin (dépassement possible)
 - Accès à tout ce qui est graphique