Programmation ANDROID

M4206C

Réseaux & Télécoms

ANDROID

IUT de Béziers, dépt. R&T © 2013-2020
http://www.borelly.net/
Christophe.BORELLY@umontpellier.fr

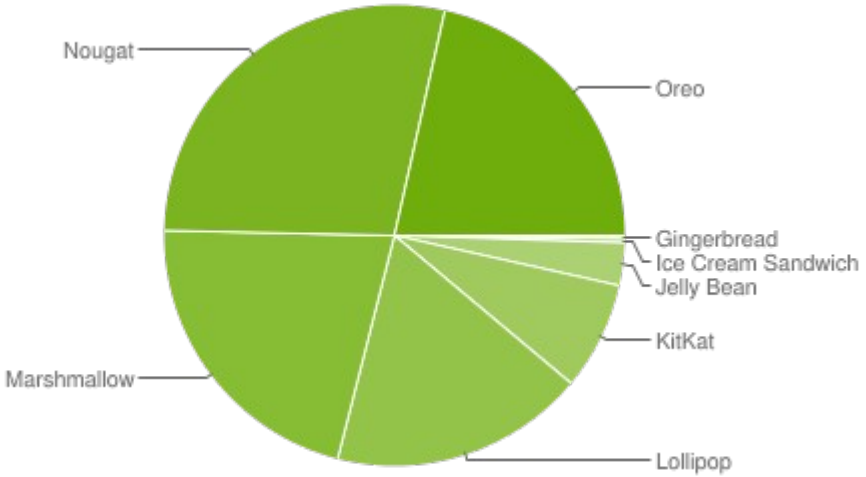
Android

- Android est un système d'exploitation mobile
  - Smartphones, tablettes, PDA…
- Startup rachetée par Google en 2005
- Système Open Source avec noyau LINUX
- Machines virtuelles JAVA
  - **DVM** : Dalvik Virtual Machine (.dex et .apk)
    - JIT : just-in-time compilation (depuis 2010)
  - **ART** : Android RunTime (depuis fin 2013)
    - AOT : ahead-of-time compilation

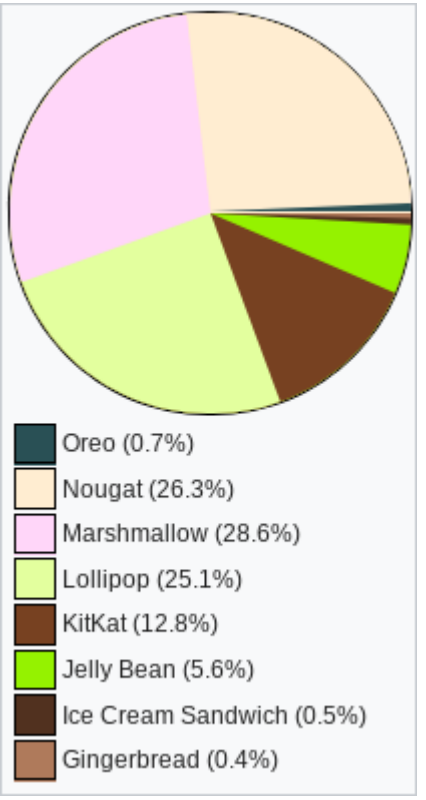
Versions Android

- 1.0 Apple pie 11/11/07
- 1.1 Bananas split 22/10/08
- 1.5 Cupcake 30/04/09
- 1.6 Donut 15/09/09
- 2.0 Eclair 26/10/09
- 2.2 Froyo 20/05/10
- 2.3 Gingerbread 06/12/10
- 3.x Honeycomb 22/02/11
- 4.0 Ice Cream Sandwich 19/10/11 (kernel 3.0.1)
- 4.1 Jelly bean 09/07/12
- 4.2 Jelly bean 13/10/12
- 4.3 Jelly bean 24/07/13
- 4.4 KitKat 31/10/13
- 5.0 Lollipop 15/10/14
- 6.0 Marshmallow 05/10/15
- 7.0 Nougat 22/08/16
- 7.1 Nougat 04/10/16
- 8.0 Oreo 21/08/17
- 8.1 Oreo 05/12/17
- 9.0 Pie 06/08/18
- 10.0 Android 10 03/09/19
- 11.0 ...

Statistiques



10/2018



01/2018

https://en.wikipedia.org/wiki/Android_version_history
https://developer.android.com/about/dashboards/index.html

Taille des écrans

- Taille des écrans (1 Inch = 2,54 cm)
- Densité : DPI (Dot Per Inch)
  - LDPI (Low : 120 dpi)
  - MDPI (Medium : 160 dpi)
  - HDPI (High : 240 dpi)
  - XHDPI (eXtra High : 320 dpi)
  - XXHDPI (eXtra High : 480 dpi)
  - XXXHDPI (eXtra High : 640 dpi)
- **Density independant Pixel** (**dp**)
  - px = dp * (dpi / 160)
- Texte : Scaled Pixels (sp)

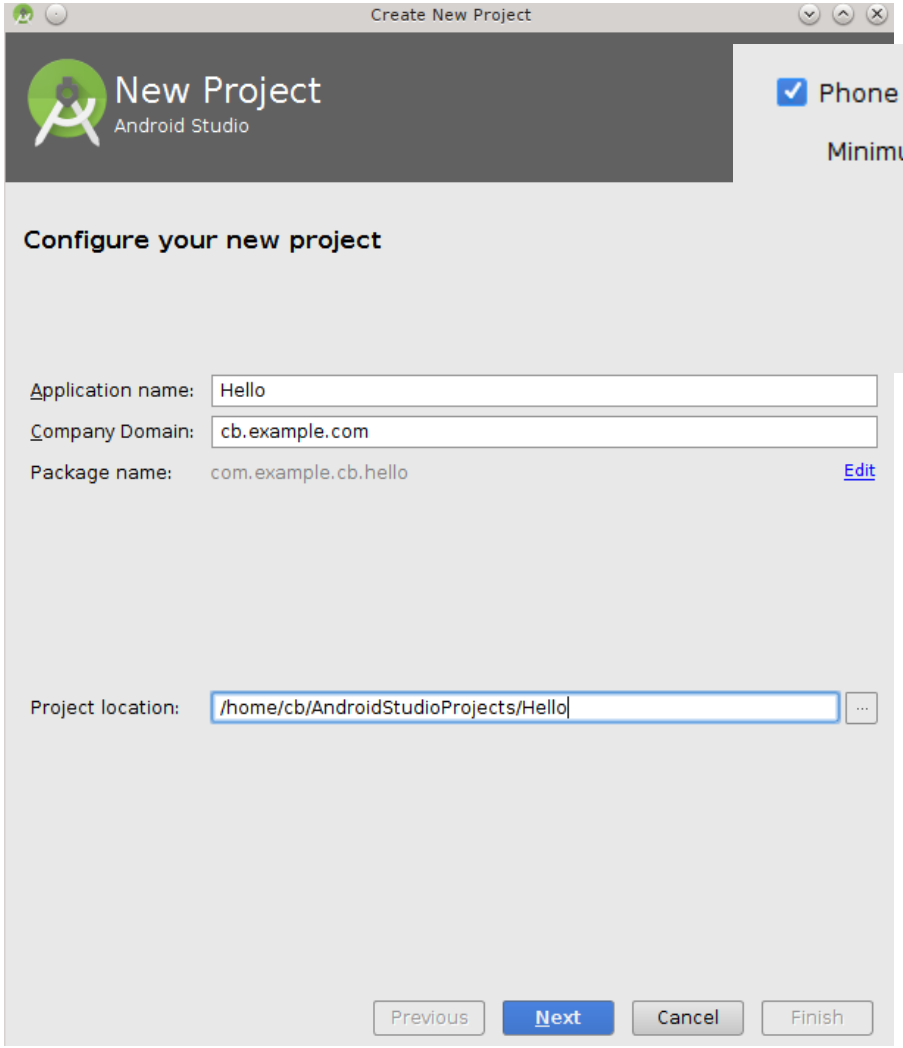
Boite à outils

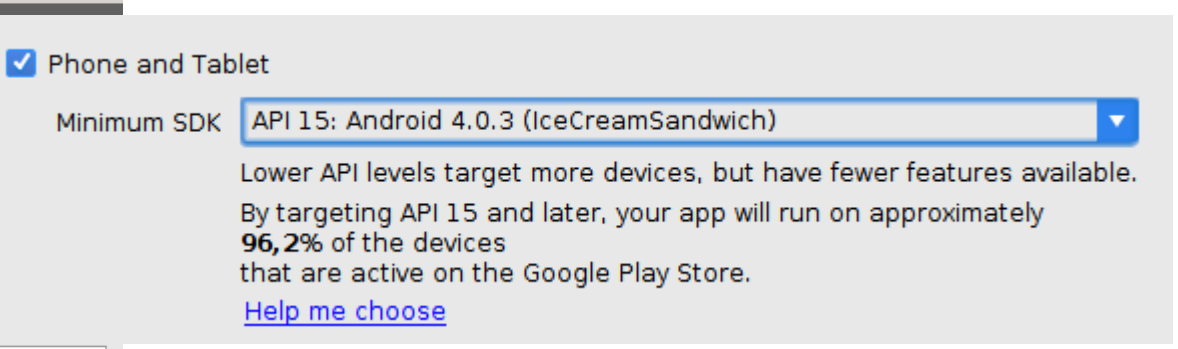
- SDK (Software Development Kit)
  - http://developer.android.com/sdk/
- ADT (Android Developer Tools)
- IDE (Integrated Development Environment) :
  - Eclipse (JDK 1.6 + Ant)
  - Android studio (JDK 1.7 + Gradle + 4 GB RAM)
- Programmation en JAVA

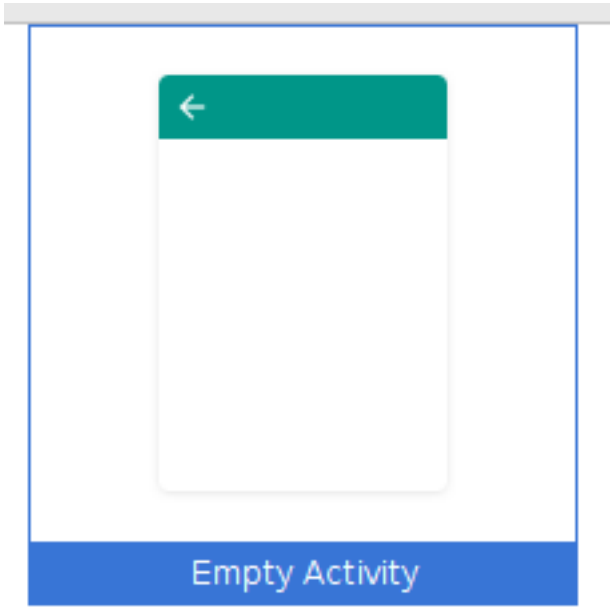
Applications Android

- Pas de « main() », mais des « Activités » !
- Isolées les unes des autres
  - 1 userID par application
  - 1 VM par application
- Les composants d'une application :
  - **Activités** (une « page » de l'interface graphique)
  - **Services** (en tâche de fond)
  - **Fournisseurs de contenu** (ContentProvider)
  - **Récepteurs de diffusion** (BroadcastReceiver)

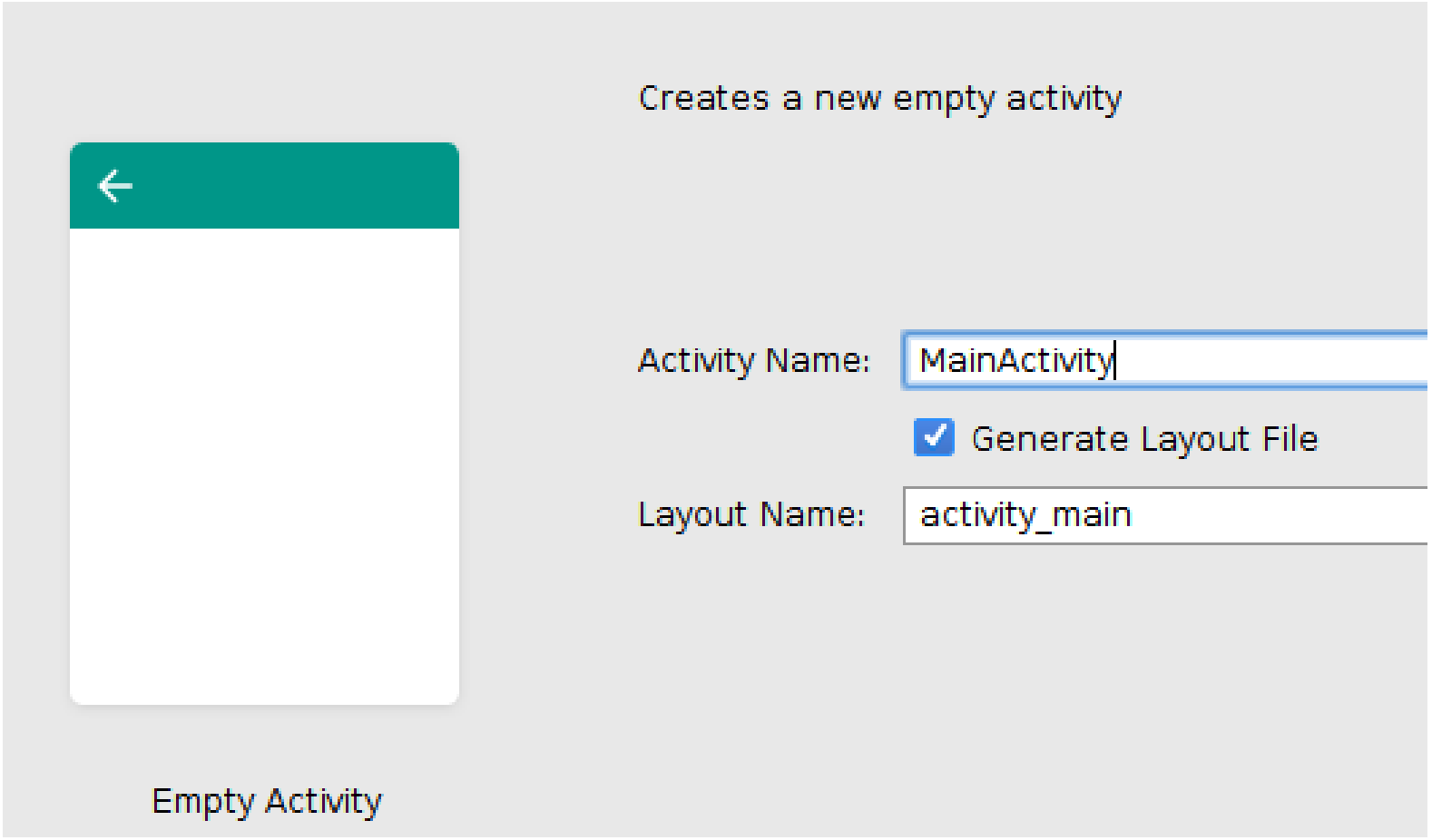
New Project...


Create New Project
New Project
Android Studio
Configure your new project
Application name:
Hello
Company Domain:
cb.example.com
Package name:
com.example.cb.hello
Edit
Project location:
/home/cb/AndroidStudioProjects/Hello
Previous
Next
Cancel
Finish

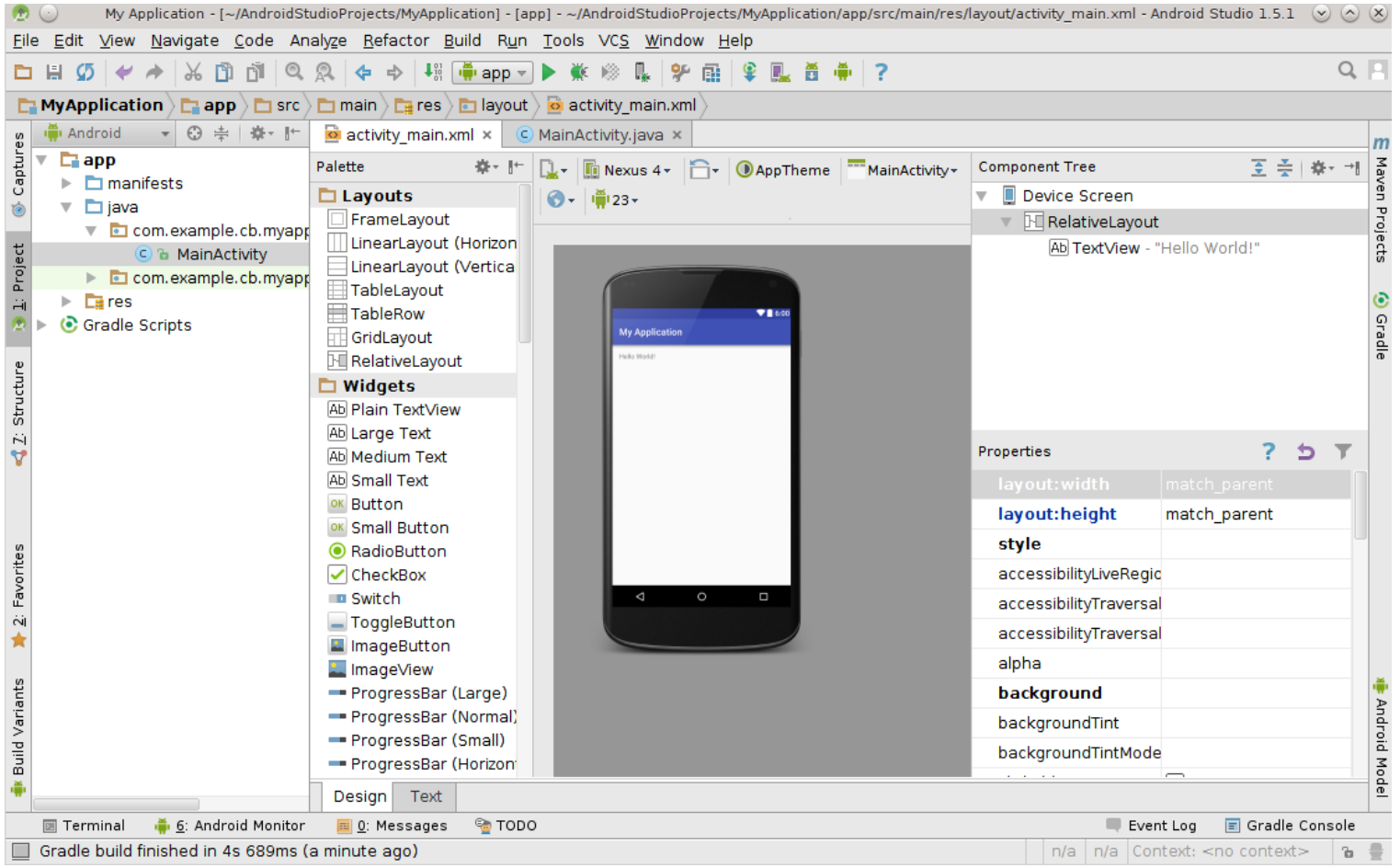

Phone and Tablet
Minimum SDK
API 15: Android 4.0.3 (IceCreamSandwich)
Lower API levels target more devices, but have fewer features available.
By targeting API 15 and later, your app will run on approximately
96,2% of the devices
that are active on the Google Play Store.
Help me choose


Empty Activity

Activité par défaut



Android Studio



Architecture d'un projet

- app/build
- app/libs
- app/src/main
  - AndroidManifest.xml
- app/src/main/java/com/example/cb/hello
  - MainActivity.java
- app/src/main/res/drawable...
- app/src/main/res/layout/activity_main.xml
- app/src/main/res/menu/...
- app/src/main/res/values/...

AndroidManifest.xml

- Décrit le projet
  - Label, Icône et Thème
  - Permissions
  - Librairies utilisées
  - etc...

Exemple de manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.cb.hello" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
              <action android:name="android.intent.action.MAIN" />
              <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Activité

- Classe de base `android.app.Activity` (qui hérite de `android.content.Context`)
- Le contenu visuel est un ensemble de vues (Hérite de `android.view.View`)
  - TextView, EditText, Button, CheckBox, ...
- L'organisation des éléments se fait dans un fichier XML (activity_main.xml).
  - LinearLayout, RelativeLayout, …
  - WebView, ListView, GridView...

MainActivity.java

```
package com.example.cb.hello;
...
public class MainActivity
                extends AppCompatActivity {
  @Override
  protected void onCreate(Bundle sis) {
    super.onCreate(sis);
    setContentView(R.layout.activity_main);
    ...
  }
...
```

Contenu défini dans un fichier XML (`activity_main.xml`)

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<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="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="com.example.cb.hello.MainActivity">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />
</RelativeLayout>
```

Attributs de layout

- **layout_width**, **layout_height**
  - `wrap_content` : place minimum
  - `match_parent` : taille du parent (remplace ~~`fill_parent`~~ depuis API 8)
  - Valeur (en **dp**)
- **orientation** : vertical | horizontal
- **gravity** : top, bottom, left, right, center, …
- ...

Ressources

- app/src/main/res/values/**strings**.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>

    <string name="app_name">Hello</string>
    <string name="action_settings">Settings</string>

</resources>
```

- Notation dans les fichiers XML : @string/xxxx
- Création dynamique de la classe R.java qui contient les identifiants de chaque ressource (ex. : R.**string**.app_name)

Internationalisation

- Il suffit de créer un répertoire avec une extension par langage et d'adapter les valeurs des identifiants :
- app/src/main/res/values-**en**/**strings**.xml

```
<string name="userName">User Name</string>
```

- app/src/main/res/values-**fr**/**strings**.xml

```
<string name="userName">Nom d\'utilisateur</string>
```

- ...
- Mieux : utiliser l’éditeur de traduction !

Récupération des ressources

```
import android.content.res.Resources;
...
Resources res=getResources();
String ap=res.getString(R.string.app_name);
```

- Cherche la valeur nommée app_name dans le fichier de ressources :
  `app/src/main/res/values/strings.xml`

TextView

- Ajout d'un identifiant dans le layout :

```
<TextView android:text="@string/hello_world"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/tv1" />
```

NB : Le + ajoute automatiquement l'identifiant dans la classe R.java

- Dans l'activité :

```
TextView tv=(TextView)findViewById(R.id.tv1);
tv.setText("Nouveau message");
```

Styles

- app/src/main/res/values/**styles**.xml

```
<style name="txtOK">
  <item name="android:textStyle">normal</item>
  <item name="android:textColor">#000000</item>
</style>
<style name="txtErr">
  <item name="android:textStyle">bold</item>
  <item name="android:textColor">#FF0000</item>
  <item name="android:textSize">12sp</item>
</style>
```

Changer le style d'une vue

```
TextView tv=(TextView)findViewById(R.id.tv1);
...
//Context context=view.getContext();
Context context=getApplicationContext();
// Change le style du texte
tv.setTextAppearance(context,R.style.txtOK);
// Requires API 23
//tv.setTextAppearance(R.style.txtOK);
```

- Définir le style dans le layout XML :

```
<TextView android:id="@+id/tv1"
...
    style="@style/txtOK" />
```

android.widget.EditText

- Zone de texte modifiable
- Récupérer la valeur :

```
EditText t=(EditText)
            findViewById(R.id.editText);
String msg=t.getText().toString();
```

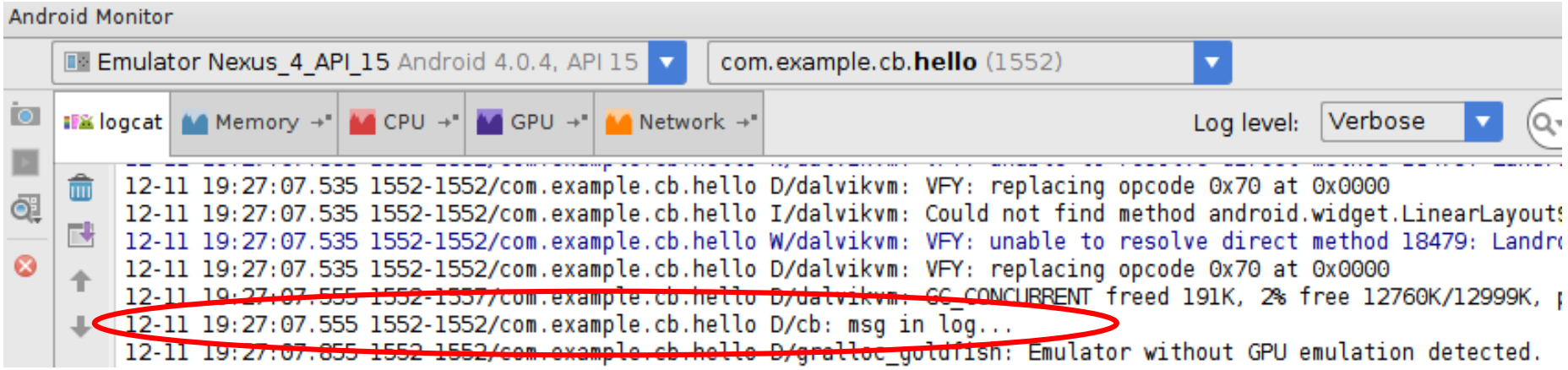
Debug : logcat (1)

- Classe : **android.util.Log**
  - 2 paramètres (tag et msg)
  - Méthodes statiques : debug `d()`, error `e()`, information `i()`, verbose `v()`, warning `w()`, etc...
- Messages affichés dans « logcat »

Debug : logcat (2)

- Exemple :

```
super.onCreate(savedInstanceState);
setContentView(R.layout.activity_main);

Log.d("cb","msg in log...");
```



android.widget.Toast

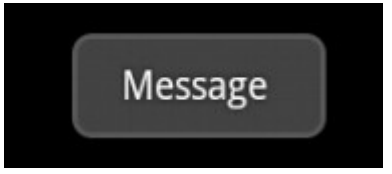
- Exemple détaillé dans une activité :

```
Context context=getApplicationContext();
CharSequence text="Hello world!";
int duration=Toast.LENGTH_SHORT;
Toast t=Toast.makeText(context,text,duration);
t.show();
```

- Possibilité de modifier des attributs :

```
t.setGravity(Gravity.TOP|Gravity.LEFT,0,0);
```

- Exemple condensé :




```
Toast.makeText(view.getContext(),
               "Message",
               Toast.LENGTH_LONG).show();
```

Snackbar

- Nécessite une librairie particulière :
- App/build.gradle


```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:23.0.1'
    compile 'com.android.support:design:23.0.1'
}
```

- android.support.design.widget.Snackbar

```
...
Snackbar.make(view,"Msg 2",
    Snackbar.LENGTH_LONG).show();
```

Les boutons

```
<Button
   android:text="ok"
   android:id="@+id/Bt1" />
```

- Image seulement :

```
<ImageButton
   android:src="@drawable/bticon"
   android:id="@+id/Bt2" />
```

- Texte et image :

```
<Button
   android:text="ok"
   android:id="@+id/Bt3"
   android:drawableLeft="@drawable/bticon" />
```

Gestion du click (1)

- Classe : `android.view.View.OnClickListener`
- Utilisation de classes anonymes :

```
Button b=(Button)findViewById(R.id.Bt1);
b.setOnClickListener(
    new OnClickListener() {
      @Override
      public void onClick(View v) {...}
    });
```

- **Il y a plus simple ! voir diapo suivante...**

Gestion du click (2)

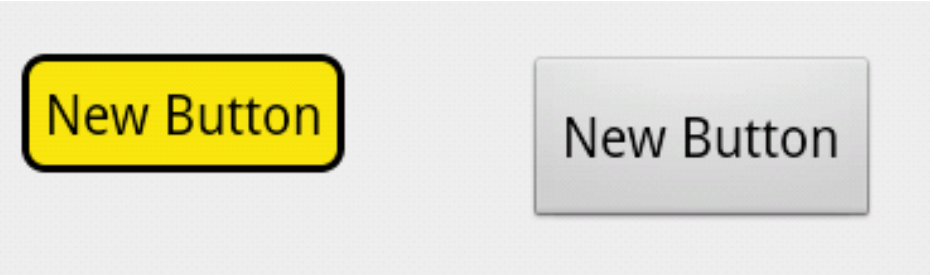
- Ajout d'un attribut dans le layout (depuis API 4 – Android 1.6) :
  - `android:onClick="sendMessage"`
- Ajout de la méthode dans l'activité :

```
...
public void sendMessage(View view){
  ...
}
```

Shape/Drawable (1)

- Permet de dessiner une forme (line, rectangle, oval, ring) dans un fichier XML (ok.xml) : voir
  http://developer.android.com/guide/topics/resources/drawable-resource.html#Shape

```
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid
        android:color="#F9E60E"/>
    <stroke
        android:width="2dp"
        android:color="#000000"/>
    <padding
        android:left="6dp"
        android:top="6dp"
        android:right="6dp"
        android:bottom="6dp"/>
    <corners
        android:radius="5dp"/>
</shape>
```



Shape/Drawable (2)

```
<Button
    android:text="New Button"
    android:id="@+id/button"
    android:background="@drawable/ok"
    ... />

Button ok=(Button)findViewById(R.id.Bt1);
ok.setBackgroundResource(R.drawable.ok);
```

Liste d'états

- Permet de définir plusieurs « drawable » suivant l'état de l'élément (Voir android.graphics.drawable.StateListDrawable):

state_pressed, state_focused, state_selected, state_checkable, state_checked, state_enabled, state_activated, ...

```
<selector
xmlns:android="http://schemas.android.com/apk/res/android">
    <item android:state_pressed="true"
          android:drawable="@drawable/ok_press"/>
    <item android:state_focused="true"
          android:drawable="@drawable/ok_focus"/>
    <item android:drawable="@drawable/ok_base"/>
</selector>
```

Les Intents

- Permet d'interagir avec d'autres Activités.
- Exemple démarrer une autre activité :

```
//Context context=view.getContext();
Context context=getApplicationContext();
Intent i=new Intent(context,
                    Activity2.class);
startActivity(i);
```

Échanges de paramètres

```
Intent i=new Intent(this,
                    Activity2.class);
i.putExtra("txt","Hello world !");
startActivity(i);
```

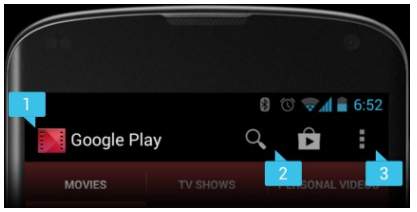
NB : Le contexte utilisé ici est this qui représente l'activité en cours qui hérite naturellement de la classe Context.

- Dans l'activité Activity2 :

```
...
Intent i2=getIntent();
String msg=i2.getStringExtra("txt");
...
```

Les menus

- Jusqu'à Android 2.3 (API 10), limité en affichage à 6 éléments en bas de l'écran (F2 dans le simulateur)
- Depuis Android 3.0 (API 11) plus besoin d'un bouton « physique » sur le téléphone, le menu apparaît en haut dans la barre d'actions
- 3 catégories de menus :
  - OptionMenu et ActionBar
  - ContextMenu (e.g. clic long)
  - PopupMenu



OptionsMenu

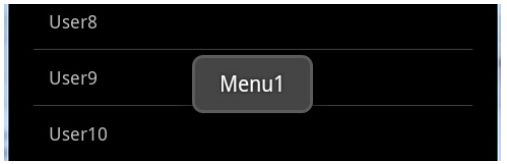
```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
  getMenuInflater().inflate(R.menu.menu_main,menu);
  //return super.onCreateOptionsMenu(menu);
  return true;
}
```

- Fichier : app/src/main/res/**menu**/**menu_main.xml**

```
<menu
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".Main">
  <item android:id="@+id/action_menu1"
        android:title="@string/action_menu1" />
  <item android:id="@+id/action_menu2"
        android:title="@string/action_menu2" />
</menu>
```

onOptionsItemSelected

```
public boolean onOptionsItemSelected(MenuItem item) {
    switch(item.getItemId()) {

      case R.id.action_menu1:
        Toast.makeText(this,"Menu1",Toast.LENGTH_LONG).show();
        return true;

      case R.id.action_menu2:
        Toast.makeText(this,"Menu2",Toast.LENGTH_LONG).show();
        return true;
    }
    return super.onOptionsItemSelected(item);
}
```



Attributs de menus

- android:**icon**
- **app**:**showAsAction**

xmlns:**app**="http://schemas.android.com/apk/res-auto"

  - ifRoom, never, always, withText

ActionBar

- android:**titleCondensed**
- android:checkable
- android:checked
- android:visible
- android:enabled
- android:onClick (depuis API 11) permet de définir directement une méthode à utiliser :

```
public void xxxx(MenuItem item)
```

-

Les listes (ListView)

```
import android.widget.ListView;
import android.widget.ArrayAdapter;
import java.util.ArrayList;
...
ListView lv;                              // Champs
ArrayAdapter<String> listAdapter; // Champs
...
   lv=(ListView)findViewById(R.id.listView);
   ArrayList<String> userList=new ArrayList<String>();
   for (int i=0;i<50;i++) userList.add("User"+i);
   listAdapter=new ArrayAdapter<String>(
                                  getApplicationContext(),
                                  R.layout.ligne,
                                  userList);

   lv.setAdapter(listAdapter);
```

```
ArrayAdapter(Context context,
             int resource,
             List<T> objects)
```

ArrayAdapter

- `void add(T object)`
- `void clear()`
- `Context getContext()`
- `int getCount()`
- `T getItem(int position)`
- `int getPosition(T item)`
- `void insert(T object,int index)`
- `void notifyDataSetChanged()`
- `void remove(T object)`
- `void sort(Comparator<? super T> cmp)`
- `...`

Layouts pour la liste

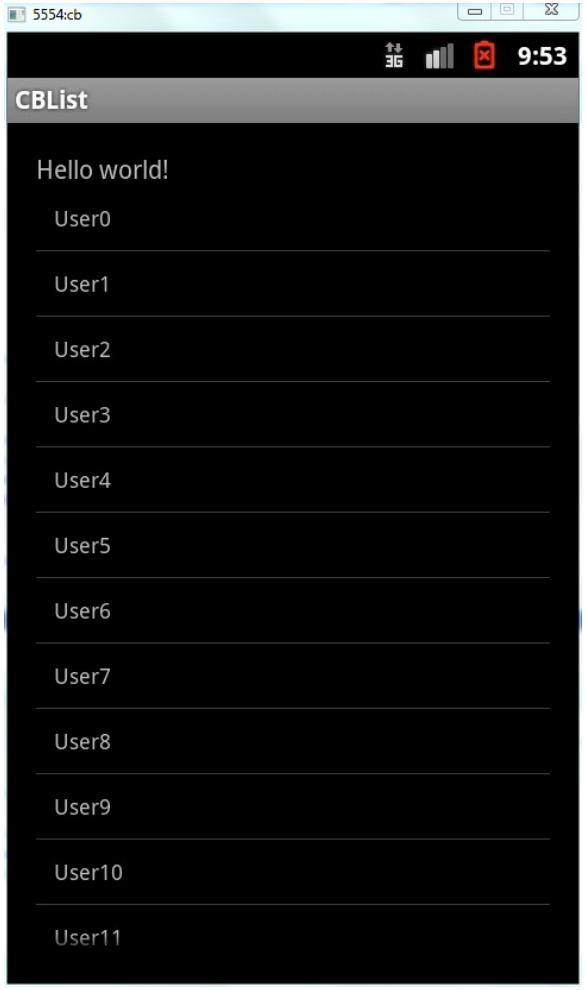
- Layout de l'activité :

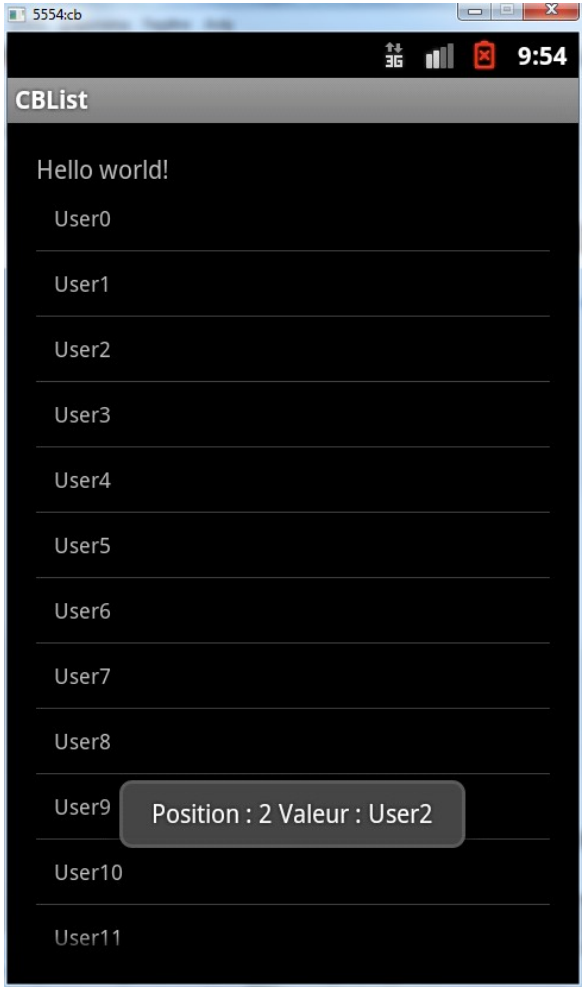
```
...
<ListView
 android:id="@+id/listView"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content" />
```

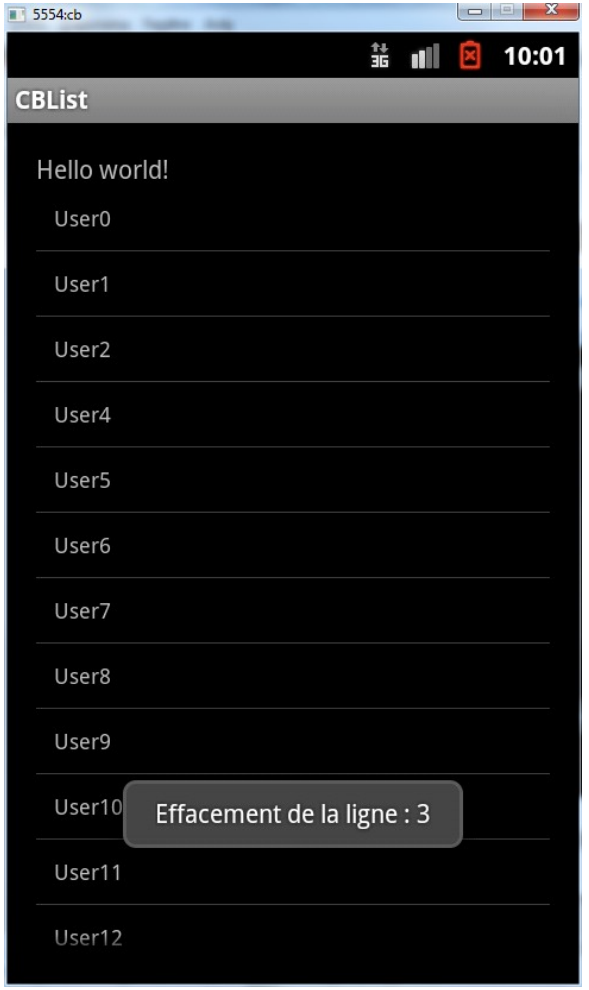
- Layout simple d'une ligne (fichier ligne.xml) :

```
<TextView
xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/ligneTextView"
 android:layout_width="fill_parent"
 android:layout_height="wrap_content"
 ...
 style="@style/txtOK" />
```

Résultats


5554:cb
9:53
CBList
Hello world!
User0
User1
User2
User3
User4
User5
User6
User7
User8
User9
User10
User11


5554:cb
9:54
CBList
Hello world!
User0
User1
User2
User3
User4
User5
User6
User7
User8
User9
Position : 2 Valeur : User2
User10
User11


5554:cb
10:01
CBList
Hello world!
User0
User1
User2
User4
User5
User6
User7
User8
User9
User10
Effacement de la ligne : 3
User11
User12

OnItemClickListener

- Classe : `android.widget.AdapterView.OnItemClickListener`

```
lv.setOnItemClickListener(
    new OnItemClickListener() {
 @Override
 public void onItemClick(AdapterView<?> parent,
                         View view,int pos,long id) {

  String val=(String)lv.getItemAtPosition(pos);
  String msg="Position : "+pos+" Valeur : "+val;
  Toast.makeText(getApplicationContext(),
                 msg,Toast.LENGTH_LONG).show();
 }
});
```

OnItem**Long**ClickListener

- Classe : `android.widget.AdapterView.OnItem`**`Long`**`ClickListener`

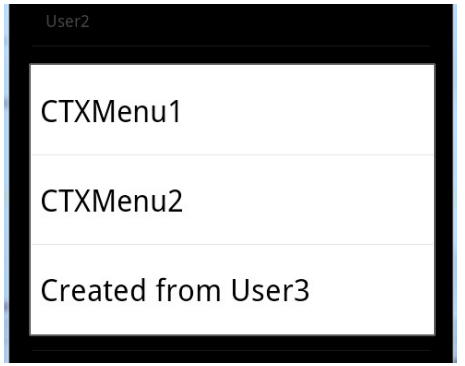
```
lv.setOnItemLongClickListener(
    new OnItemLongClickListener() {
 @Override
 public boolean onItemLongClick(AdapterView<?> parent,
                                View view,int pos,long id) {
  listAdapter.remove(listAdapter.getItem(pos));
  listAdapter.notifyDataSetChanged();
  String msg="Effacement de la ligne : "+pos;
  Toast.makeText(getApplicationContext(),
                 msg,Toast.LENGTH_LONG).show();
  return true; // Permet de ne pas lancer onItemClick()
 }
});
```

ContextMenu

- Incompatible avec onItemLongClick...
- Enregistrement de la vue pour le menu :

```
registerForContextMenu(lv);
```

- Ajout de la méthode de création du menu :




```
@Override
public void onCreateContextMenu(
                    ContextMenu menu,
                    View v,
                    ContextMenuInfo menuInfo) {

  super.onCreateContextMenu(menu,v,menuInfo);
  getMenuInflater().inflate(R.menu.menu_ctx,menu);
  int pos=((AdapterContextMenuInfo)menuInfo).position;
  String itemValue=(String)lv.getItemAtPosition(pos);
  menu.add("Created from "+itemValue);
}
```

onContextItemSelected

```
@Override
public boolean onContextItemSelected(MenuItem item) {
  AdapterContextMenuInfo info=
            (AdapterContextMenuInfo)item.getMenuInfo();

  switch (item.getItemId()) {

    case R.id.ctx_menu1:
        Toast.makeText(getApplicationContext(),
                       "C1 : "+info.position,
                       Toast.LENGTH_LONG).show();
      return true;

    default:
      return super.onContextItemSelected(item);
  }
}
```

Adaptateur personnalisé

- Sert à faire des listes avec un contenu complexe
- On crée un objet spécifique pour les données
- Puis on crée un adaptateur pour cet objet dans lequel la méthode getView() renverra la vue adaptée à la position fournie
- Par exemple pour un layout avec une image et du texte (CBItem.java et CBAdapter.java)

CBItem.java

```
public class CBItem {
   private int imgId;
   private String text;

   public int getImgId() {return imgId;}
   public void setImgId(int i) {imgId=i;}
   public String getText() {return text;}
   public void setText(String t) {text=t;}

   public CBItem(int i,String t) {
      this.imgId=i;
      this.text=t;
   }
}
```

CBAdapter.java (1)

```
public class CBAdapter extends ArrayAdapter<CBItem> {

  CBAdapter(Context ctx,int resource,List<CBItem> items) {
    super(ctx,resource,items);
  }
  @Override
  public View getView(int position,
                      View convertView,
                      ViewGroup parent) {
    View v=convertView;
    LayoutInflater inflater=(LayoutInflater)getContext().
         getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    if(convertView==null) {
       v=inflater.inflate(R.layout.ligne2,null);
    }
    ...
```

CBAdapter.java (2)

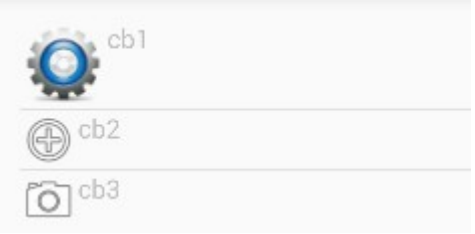
```
...
    // Récupération des éléments visuels
    // Définis dans ligne2.xml
    TextView texte=(TextView)v.findViewById(R.id.txt);
    ImageView image=(ImageView)v.findViewById(R.id.img);
    // Récupération de l'objet dans la liste
    // C'est un ArrayAdapter<CBItem> !
    CBItem cb=this.getItem(position);
    // Modification des éléments visuels
    texte.setText(cb.getText());
    image.setImageResource(cb.getImgId());
    return v;
  }
}
```

res/layout/ligne2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:id="@+id/img" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:id="@+id/txt" />
</LinearLayout>
```

Code d'affichage de la liste

```
...
lv=(ListView)findViewById(R.id.listView);
ArrayList<CBItem> userList=new ArrayList<CBItem>();
userList.add(
 new CBItem(R.drawable.roue,"cb1"));
userList.add(
 new CBItem(android.R.drawable.ic_menu_add,"cb2"));
userList.add(
 new CBItem(android.R.drawable.ic_menu_camera,"cb3"));
CBAdapter cba=new CBAdapter(getApplicationContext(),
                  R.layout.ligne2,
                  userList);
lv.setAdapter(cba);
```



BaseExpandableListAdapter

- Permet d'avoir 2 niveaux de liste
- On crée un adaptateur à partir de android.widget.BaseExpandableListAdapter
  - Méthodes : `getGroup(), getChild(), getGroupCount(), getChildCount(), getGroupId(), getChildId(), getGroupView(), getChildView()`
- Dans le layout, on utilise le tag `<ExpandableListView … />`

Exemple d'implémentation

```
public class CBExpListAdapter
                        extends BaseExpandableListAdapter {

  private LayoutInflater inflater;
  private String[] groups;          // Niveau 1
  private String[][] childs;        // Niveau 2

  public CBExpListAdapter(Context ctx,String[] g,
                          String[][] c) {
    inflater=(LayoutInflater)ctx.getSystemService(
                          Context.LAYOUT_INFLATER_SERVICE);
    this.groups=g;
    this.childs=c;
  }
...
```

getGroup() et getChild()

```
...
  public Object getGroup(int groupPosition) {
    return groups[groupPosition];
  }
  public Object getChild(int groupPosition,
                         int childPosition) {
    return childs[groupPosition][childPosition];
  }
  public int getGroupCount() {
    return groups.length;
  }
  public int getChildrenCount(int groupPosition) {
    return childs[groupPosition].length;
  }
...
```

getGroupView()

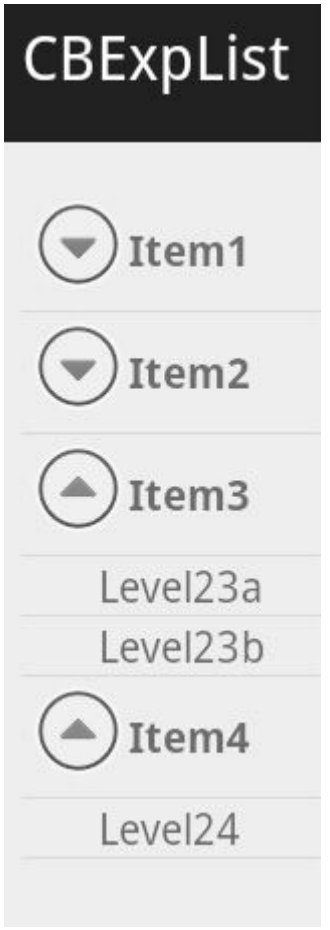
```
...
  public View getGroupView(int groupPosition,
                           boolean isExpanded,
                           View convertView,
                           ViewGroup parent) {
    if (convertView==null) {
      convertView=inflater.inflate(R.layout.level1,null);
    }
    TextView tv=(TextView)convertView.findViewById(R.id.l1);
    tv.setTypeface(null,Typeface.BOLD);
    String msg1=(String)getGroup(groupPosition);
    tv.setText(msg1);
    return convertView;
  }
...
```

getChildView()

```
...
  public View getChildView(int groupPosition,
                           int childPosition,
                           boolean isLastChild,
                           View convertView,
                           ViewGroup parent) {
    if (convertView==null) {
      convertView=inflater.inflate(R.layout.level2,null);
    }
    TextView tv=(TextView)convertView.findViewById(R.id.l2);
    String msg2=(String)getChild(groupPosition,
                                 childPosition);
    tv.setText(msg2);
    return convertView;
  }
...
```

Dans l'activité...

```
...
  String[]
level1={"Item1","Item2","Item3","Item4"};
  String[][] level2={
                {"Level22a","Level22b"},
                {"Level22x","Level22y","Level22z"},
                {"Level23a","Level23b"},
                {"Level24"}
                       };
 ExpandableListView elv=(ExpandableListView)
                    findViewById(R.id.exp_list);
 CBExpListAdapter ela=
    new CBExpListAdapter(this,level1,level2);
 elv.setAdapter(ela);
...
```



Préférences

- Valeurs persistantes pour l'application

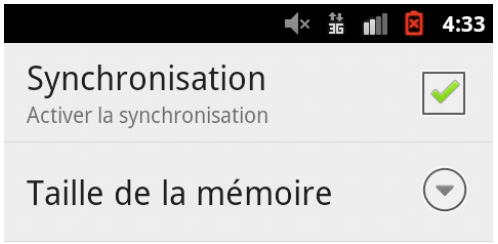
```
SharedPreferences p=
  PreferenceManager.getDefaultSharedPreferences(this);
// getSharedPreferences("infos",MODE_PRIVATE);
// getPreferences(MODE_PRIVATE);
String n=p.getString("nom","Toto"); // Par défaut
int x=p.getInt("age",30);
boolean h=p.getBoolean("estUnHomme",true);
if (p.contains("xxx")) ...

SharedPreferences.Editor e=p.edit();
e.putString("nom","John DOE");
e.putInt("age",25);
e.putBoolean("estUnHomme",true);
e.commit();
```

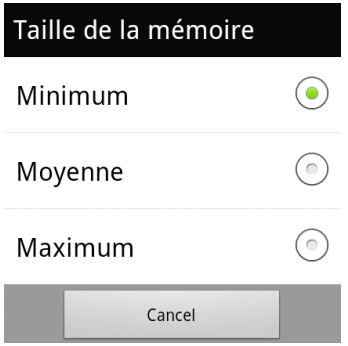
Préférences en XML

- http://developer.android.com/guide/topics/ui/settings.html
- Créer un fichier XML dans res/xml basé sur **PreferenceScreen** avec des éléments EditTextPreference, CheckBoxPreference, ListPreference, ...
- Créer une activité spécifique à base de **PreferenceActivity** et la méthode addPreferencesFromResource() - deprecated avant v3.0 API 11 - ou utilisant **PreferenceFragment**
- Définir l'activité dans le Manifest
- Démarrer l'activité lors du clic sur le menu « Settings »

res/xml/preferences.xml

```
<PreferenceScreen
xmlns:android="http://schemas.android.com/apk/res/android">
    <CheckBoxPreference
        android:key="sync"
        android:title="@string/sync_title"
        android:summary="@string/sync_summary"
        android:defaultValue="true" />
    <ListPreference
        android:key="taille"
        android:title="@string/taille_title"
        android:entries="@array/taille_entries"
        android:entryValues="@array/taille_values"
        android:defaultValue="Moins de 2Go" />
</PreferenceScreen>
```





res/values/arrays.xml

```
<resources>
    <string-array name="taille_entries">
        <item>Minimum</item>
        <item>Moyenne</item>
        <item>Maximum</item>
    </string-array>
    <string-array name="taille_values">
        <item>Moins de 2 Go</item>
        <item>Entre 2 et 5 Go</item>
        <item>Plus de 5 Go</item>
    </string-array>
</resources>
```

Classe SettingsActivity

```
public class SettingsActivity extends PreferenceActivity {
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.HONEYCOMB) {
      addPreferencesFromResource(R.xml.preferences);
    } else {
      getFragmentManager().beginTransaction()
        .replace(android.R.id.content,
                 new PrefFragment()).commit();
    } }
  @TargetApi(Build.VERSION_CODES.HONEYCOMB)
  public static class PrefFragment
                      extends PreferenceFragment {
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      addPreferencesFromResource(R.xml.preferences);
    } } }
```

Affichage de l'activité lors du clic sur le menu

```
@Override
public boolean onOptionsItemSelected(
                                    MenuItem item) {
  int id=item.getItemId();
  if (id==R.id.action_settings) {
    Intent i=new Intent(this,
                        SettingsActivity.class);
    startActivity(i);
  }
  return super.onOptionsItemSelected(item);
}
```

BroadcastReceiver

- Cette classe permet de recevoir des messages envoyés par le système ou bien d'autres applications (Intents ou Events).

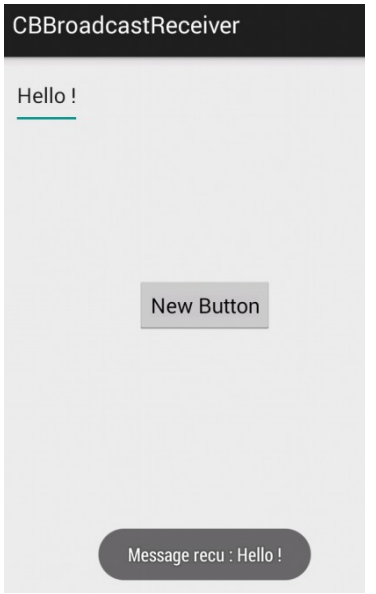
```
public class CBReceiver extends BroadcastReceiver {

   @Override
   public void onReceive(Context context,Intent intent) {
      Toast.makeText(
        context,
        "Message recu : "+intent.getStringExtra("msg"),
        Toast.LENGTH_LONG).show();
   }

}
```

Modification du Manifest

```
...
<application ...>
   <activity ...>
   </activity>
   <receiver android:name=".CBReceiver">
      <intent-filter>
  <action android:name="net.borelly.CB_ACTION"/>
      </intent-filter>
   </receiver>
</application>
```

Modification de l'activité

```
public class MainActivity extends ActionBarActivity {

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
  }

  // Appelé par l'attribut onClick du boutton
  public void cbSendBroadCast(View view) {
    // Log.d("cb","click");
    Intent intent=new Intent();
    EditText et=(EditText)findViewById(R.id.text);
    intent.putExtra("msg",et.getText().toString());
    intent.setAction("net.borelly.CB_ACTION");
    sendBroadcast(intent);
  }
  ...
}
```



Bases de données

- SQLite (Base = un fichier tout simplement)
  - http://www.sqlite.org/
- Package : android.database.sqlite

```
SQLiteDatabase db;
db=SQLiteDatabase.openOrCreateDatabase("cb.sqlite",null);
db.execSQL("DROP TABLE IF EXISTS xxx;");
db.execSQL("CREATE TABLE xxx...");

Cursor query(String table, String[] columns,
             String selection, String[] selectionArgs,
             String groupBy, String having, String orderBy,
             String limit);
Cursor rawQuery(String sql, String[] selectionArgs);
```

Exemple de requêtes

```
String sql="SELECT * FROM users WHERE nom=? AND n>?";
String args[]={"toto","18"};
Cursor c=db.rawQuery(sql,args);
...

String table="users";
String[] columns={"n","nom","prenom","telephone"};
String selection="nom=? AND n>?";
String[] selectionArgs={"toto","18"};
String groupBy=null,having=null,limit=null;
String orderBy="nom,prenom";
Cursor c=db.query(table,columns,
            selection,selectionArgs,
            groupBy,having,orderBy,limit);
...
```

Classe Cursor

```
int nb=cursor.getCount();
if (cursor.moveToFirst()) {
    do {
      int len=cursor.getColumnCount();

      String nom=cursor.getColumnName(0);
      String str=cursor.getString(0);

      int a=cursor.getInt(1);
      double x=cursor.getDouble(2);
    } while (cursor.moveToNext());
}
cursor.close();
```

Classe SQLiteOpenHelper

- Permet de faciliter la gestion des versions d'une BdD.

```
public SQLiteOpenHelper(Context context,String name,
    SQLiteDatabase.CursorFactory factory,int version)
public void onCreate(SQLiteDatabase db)
public void onUpgrade(SQLiteDatabase db,
    int oldVersion,int newVersion)
public void onCreate(SQLiteDatabase db)
public SQLiteDatabase getReadableDatabase()
public SQLiteDatabase getWritableDatabase()
public void close()
```

Service (1)

- Utile pour réaliser des opérations à long terme sans besoin d'interface utilisateur.

```
public class CBService extends IntentService {
  public static final String ACTION_XXX="xxx";
  public CBService() { super("CBService"); }
  protected void onHandleIntent(Intent reqIntent) {
    if (reqIntent!=null) {
      // On récupere les données de l'Intent en paramètre
      String param1=reqIntent.getStringExtra(...);
      // Puis on réalise l'opération...
      Intent repIntent=new Intent();
      repIntent.setAction(ACTION_XXX);
      repIntent.putExtra(...);
      sendBroadcast(repIntent);
    }
  }
}
```

Service (2)

- Déclarer le service dans le manifest :

```
<application>
  ...
  <service android:name=".CBService"
           android:exported="false"/>
<application/>
```

- Dans l'activité, créer un Intent pour paramétrer le service :

```
Intent iSvc=new Intent(this,CBService.class);
iSvc.putExtra(...);
startService(iSvc);
```

Service (3)

- Enregistrer la réception des Intents à l'aide d'un BroadcastReceiver (`CBReceiver`) dans l'activité :

```
registerReceiver(new CBReceiver(),
    new IntentFilter(CBService.ACTION_XXX));
```

- Enfin, adapter la méthode `onReceive()` du `CBReceiver` :

```
public void onReceive(Context context,
                      Intent intent) { ... }
```

ContentProvider

- Permet de fournir un accès central à des données complexes à d'autres applications.
- Doit être déclaré dans le **Manifest**.
- Depuis Android 4.2 (API 17), il faut explicitement exporter le ContentProvider pour qu'il soit accessible aux autres applications.

Références

- http://developer.android.com/sdk/
- http://android.developpez.com/cours/