Android

Amaury Gauthier
David Pequegnot

http://marketingbutcher.com/2011/05/25/android-cat-apps/

Rappels

Les OS pour téléphones mobiles (1)

Ce qui a facilité leur déploiement :

- une bonne connectivité ;
- du matériel fournissant un potentiel important
- des systèmes d'exploitations qui sont capables de profiter de ce potentiel

Premiers systèmes d'exploitation :

- des systèmes d'exploitation pour professionnels
- pas très "*user friendly*" (Windows Mobile)
- pas forcément très intuitif...

Les OS pour téléphones mobiles (2)

Puis...

- est arrivé l'iPhone !
  - fluide, élégant,
  - utilisation d'un accéléromètre,
  - etc.
- avec une interface utilisateur très intuitive

Et ensuite

- Android
- Windows Phone
- ...

http://www.topito.com/top-chats-relax

Android

Introduction

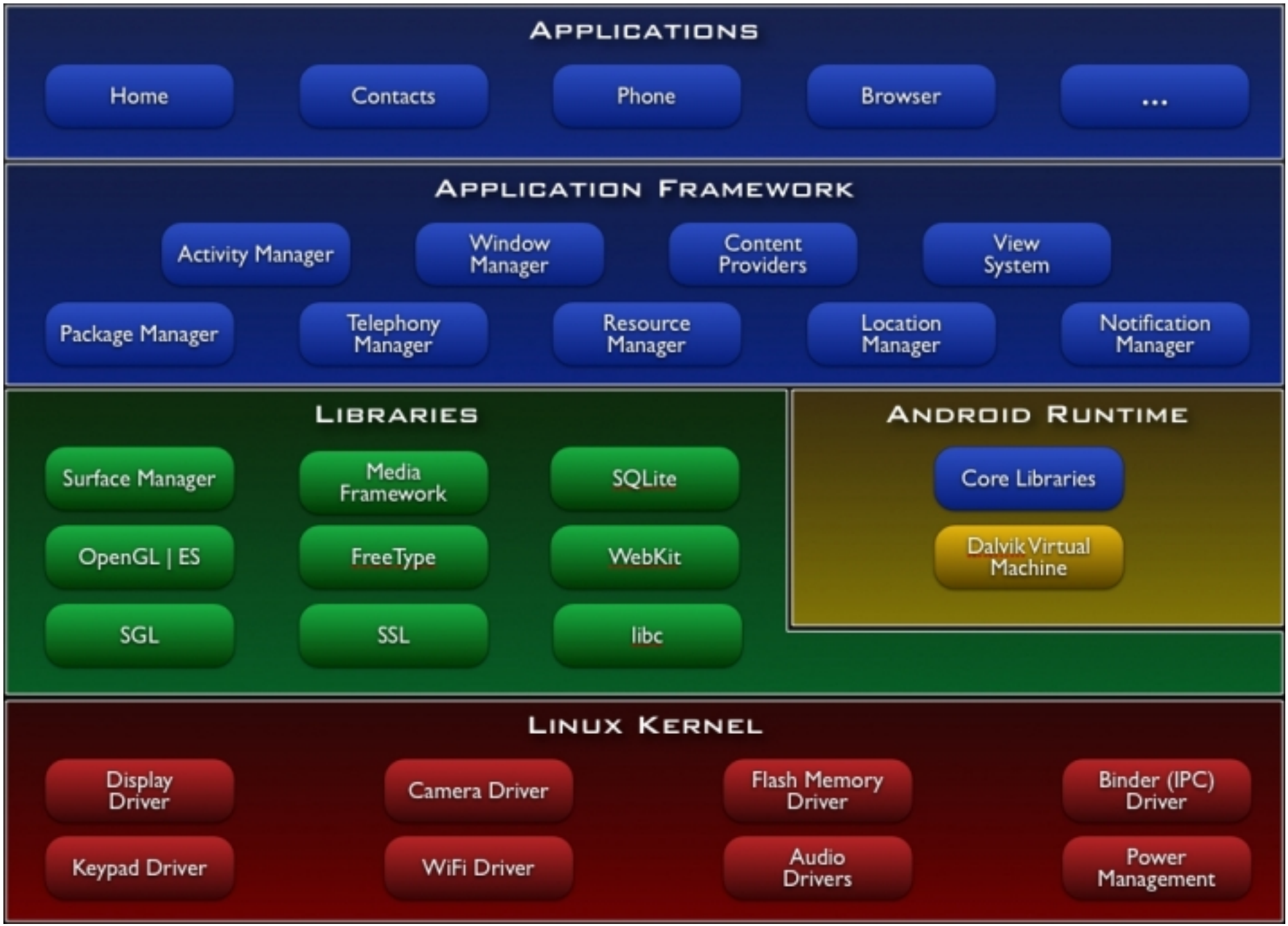
Qu'est-ce ?

- Un système d'exploitation pour appareils connectés
  - Smartphones
  - Tablettes
  - PDA
  - Télévisions
  - Autoradios
- Une plateforme complète permettant le développement d'applications tierces

Que propose la plateforme Android ?

- Une machine virtuelle "Java" : Dalvik
- Un framework applicatif pour le développement d'application tierces
- Un ensemble d'APIs afin d'interagir avec le matériel
  - GPS, accéléromètre, évidemment GSM, ...
- Un SDK complet :
  - des outils pour la création et la compilation de projets
  - un plugin Eclipse
  - un émulateur
  - des outils de débogage, profiling, etc.

Architecture



Profiles

- API levels
- minSDKVersion
- targetSdkVersion
- Compatibilité ascendente
- Compatibilité descendente

Platform Version	API Level	VERSION_CODE
Android 3.2	13	HONEYCOMB_MR2
Android 3.1.x	12	HONEYCOMB_MR1
Android 3.0.x	11	HONEYCOMB
Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1
Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD
Android 2.2.x	8	FROYO
Android 2.1.x	7	ECLAIR_MR1
Android 2.0.1	6	ECLAIR_0_1
Android 2.0	5	ECLAIR
Android 1.6	4	DONUT
Android 1.5	3	CUPCAKE
Android 1.1	2	BASE_1_1
Android 1.0	1	BASE

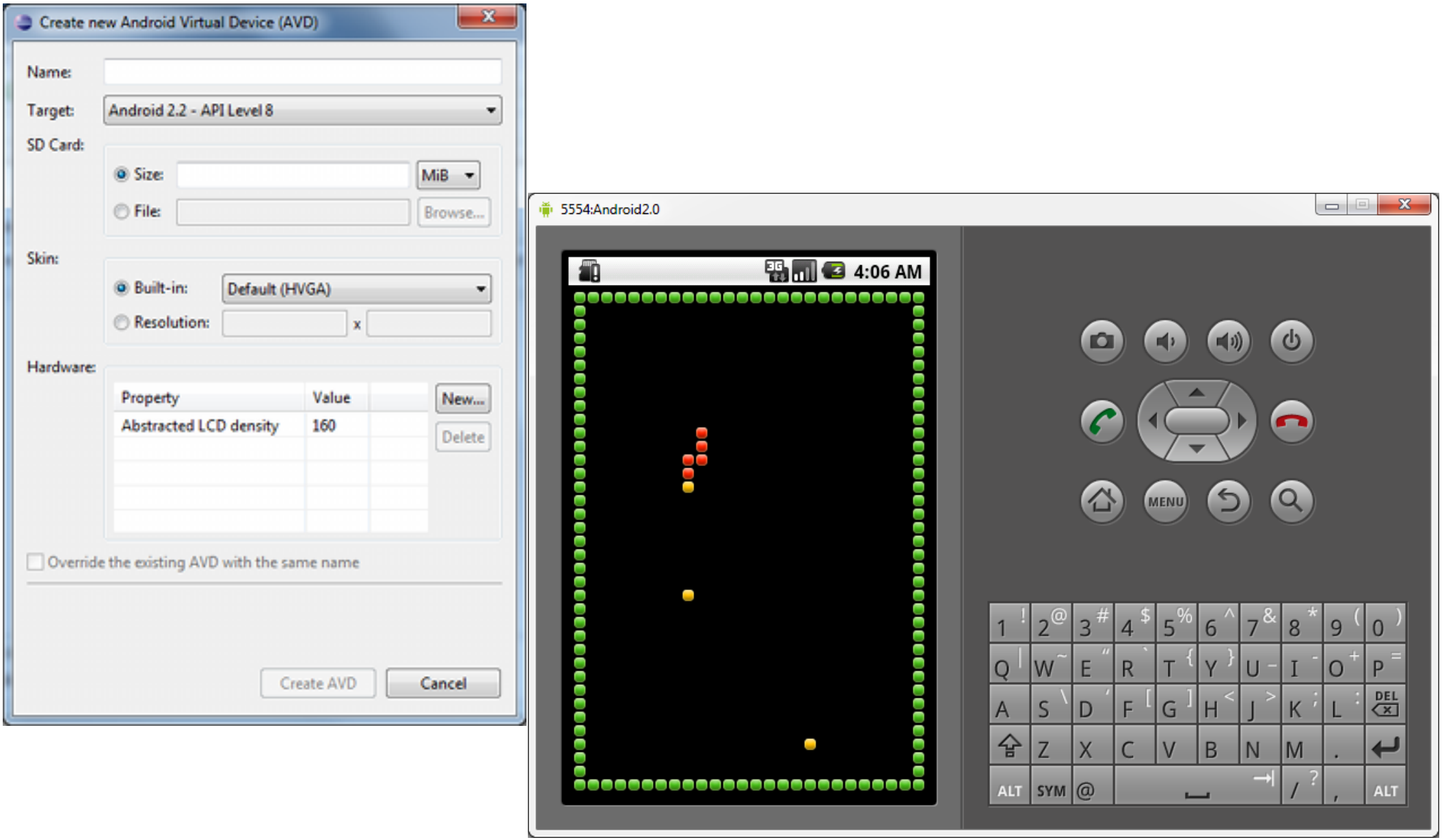
Android Virtual Device

- Comment tester une application sur différent profiles ?

- AVD :
  - profile matériel
  - API level
  - skin (taille de l'écran, apparence, etc.)
  - périphérique de stockage (émulation de carte SD)

- Gestion des AVD depuis AVD Manager ou depuis la commande android

Android Virtual Device



http://www.caffeinedi.com/wp-content/uploads/2010/07/Snake-in-Android-Virtual-Device.png

Android

Créer un projet

http://www.topito.com/top-chats-relax

Structure des projets Android

root

- AndroidManifest.xml
- Fichiers Ant :
  - build.xml, default.properties, local.properties
- /bin : résultat de compilations
- /gen : code source généré par les outils de compilation Android
- /libs : jar externes
- /src : code source Java
- /res : ressources
- /tests : tests pour votre application Android
- /assets : fichiers statiques pour le déploiement sur le terminal

Structure des projets Android

res

- Dossier contenant les ressources
- Ressources identifiées automatiquement par le système Android
- Contenu :
  - /res/anim/ : animations courtes
  - /res/color/ : couleurs
  - /res/drawable/ : images
  - /res/layout/ : layout pour les interfaces
  - /res/menu/ : contenu des menus
  - /res/strings/ : chaînes de caractères
  - /res/values/ : constantes
  - /res/xml/ : stocke vos propres données
  - etc.

Structure des projets Android

bin

- Résultat de la compilation
- Contenu :
  - /bin/classes/ : classiques fichiers Java compilés
  - /bin/classes.dex
  - /bin/<application name>.ap_ : ressources zippées
  - /bin/<application name>-debug.apk ou
  - /bin/<application name>-unsigned.apk

- Création des clés pour signature avec keytool

```
keytool -genkey -v -keystore nyancat.keystore \
    -alias nyan -keyalg DSA -validity 10000
```

- Modifier le fichier build.properties en conséquence
  - key.store=/chemin/vers/nyancat.keystore
  - key.alias=nyan

Structure des projets Android

La première fois...

- Possibilité d'utiliser la ligne de commande ou un IDE (Eclipse, Netbeans, Intellij Idea, ...)

```
$ android create project --target <cible> \
                --path /chemin/vers/le/projet/ \
                --activity <Activité Principale> \
                --package <package>
```

- Exemple :

```
$ android create project --target 2 \
        --path /home/nyan/workspace/NyanRoxxTheWorld \
        --activity NyanActivity \
        --package nyan.roxx.the.world
```

  - *Cible : Android 1.1*
  - *Activité principal : NyanActivity*
  - *Package : nyan.roxx.the.world*

- L'arborescence est automatiquement créée

Composants d'une application

Quelques définitions...

- Activité (Activity)
  - Une vue de l'application
  - Possibilité de partager une activité avec d'autres applications
  - Étend la classe Activity
- Service
  - Un composant logiciel qui va fonctionner en arrière-plan
  - Ne fournit pas d'interface utilisateur
  - Idéal pour certaines opérations longues ou d'accès à des fichiers distants
  - Étend la classe Service

Composants d'une application

Quelques définitions...

- Fournisseur de contenu (*Content Provider*)
  - Permet de gérer des données via une abstraction
  - SQLite, fichiers, serveurs, etc.
  - Partage des données entre applications :
    - l'application peut fournir
    - ou obtenir des données à partir d'une autre
  - Étend la classe ContentProvider
- *Broadcast receivers*
  - Réceptionner des évènements système via des intentions
  - Pas d'UI
  - Minimal : possibilité de lancer une notification dans la barre de status
  - Étend la classe BroadcastReceiver

Composants d'une application

Quelques définitions...

- Intentions (*Intents*)
  - Messages asynchrones
  - Une application peut répondre à une intention...
  - ... ou en créer
  - Possibilité d'activer une activité, un service ou un `broadcast receiver` grâce aux intentions

Manifest

Aperçu

- `AndroidManifest.xml`
- Peut contenir beaucoup d'informations
- Décrit la structure et certaines propriétés de l'application
  - l'API level à utiliser
  - déclaration des activités (dont la principale), services, fournisseurs de contenu ...
  - les réactions aux intents
  - les pré-requis du terminal
  - etc.

http://www.topito.com/top-chats-relax

Manifest

Structure

- Fichier xml
- Base :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="
nyan.roxx.the.world">
  ...
</manifest>
```

- L'attribut package permet de définir le package de base pour les classes Java
  - .NyanNyan => nyan.roxx.the.world.NyanNyan

Manifest

Ma première activité

- Déclaration de l'activité principale :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="nyan.roxx.the.
world">
  <application android:icon="@drawable/app_icon.png" ...>
    <activity android:name="nyan.roxx.the.world.NyanActivity
            android:label="@string/app_label" ...>
      ...
      <action android:name="android.intent.action.MAIN" />
      <category android:name="android.intent.category.LAUNCHER"/>
    </activity>
  </application>
</manifest>
```

Manifest

Principaux attributs

- Déclaration de l'activité principale :

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="nyan.roxx.the.
world">
  <uses-permission /> <!-- Demandes de permission -->
  <permission />      <!-- Protéger l'application -->
  <uses-sdk />        <!-- Informations sur le SDK -->
  <uses-configuration /> <!-- Configuration matérielle -->
  <uses-feature /> <!-- Dépendances hardware -->
  <support-screens /> <!-- Types d'écran supportés -->
  <compatible-screens /> <!-- Types d'écran supportés -->
  <support-gl-texture /> <!--Types de compression supportées-->

  <application>
    ...
  </application>
</manifest>
```

Classe R

- Classe auto-générée par les outils Android
  - Ne pas essayer de modifier cette classe !
- Package android
- Disponible dans le dossier /gen/
- Permet d'accéder aux ressources dans le code Java
  - R.drawable.nyan_picture (image dans le dossier /res/drawable/)
  - R.string.nyanvoice (string dans le fichier /res/values/filename.xml)
  - R.color.nyancolor (couleur dans le fichier /res/values/filename.xml)
  - R.id.nyanTextView (identifiant d'une "vue" dans un layout)

Classe R

Stocker des Strings, tableaux de Strings et des pluriels

- Fichier xml dans le dossier `/res/values/`
- Nom des fichiers au choix du développeur
- Exemple :

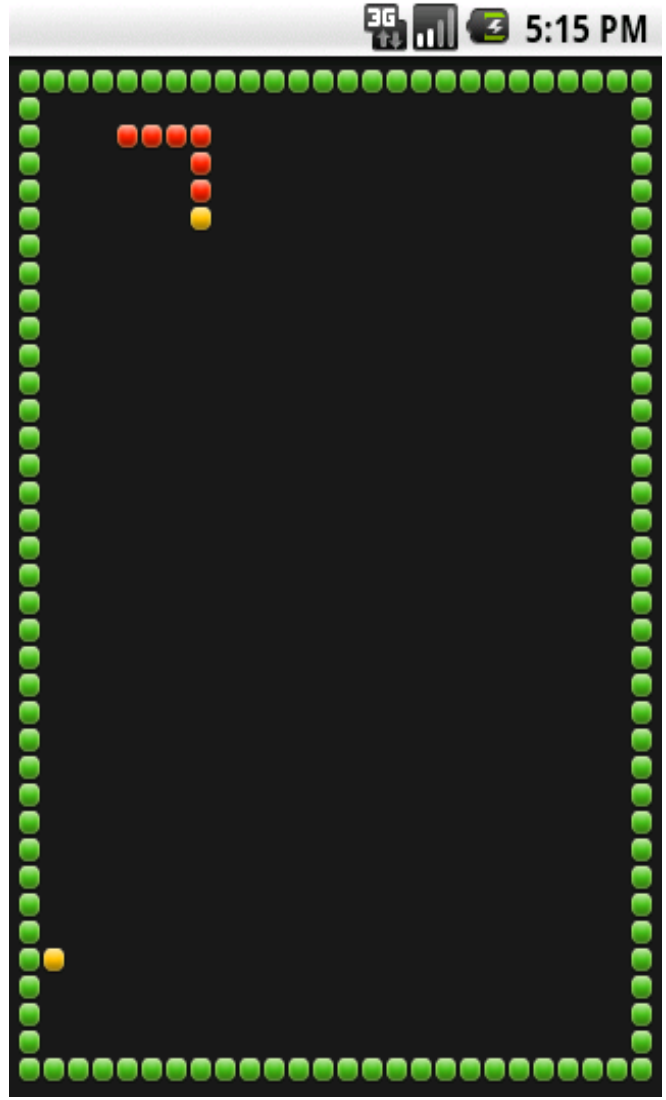
```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello_nyan">Nyan nyan!</string>

  <string-array name="hello_nyan_array">
    <item>nyan!</item>
    <item>miahou!</item>
  </string-array>

  <plural name="hello_nyan_plural">
    <item quantity="one">Nyan cat</item>
    <item quantity="other">Nyan cats</item>
  </plural>
</resources>
```

Android

Activities

Activity

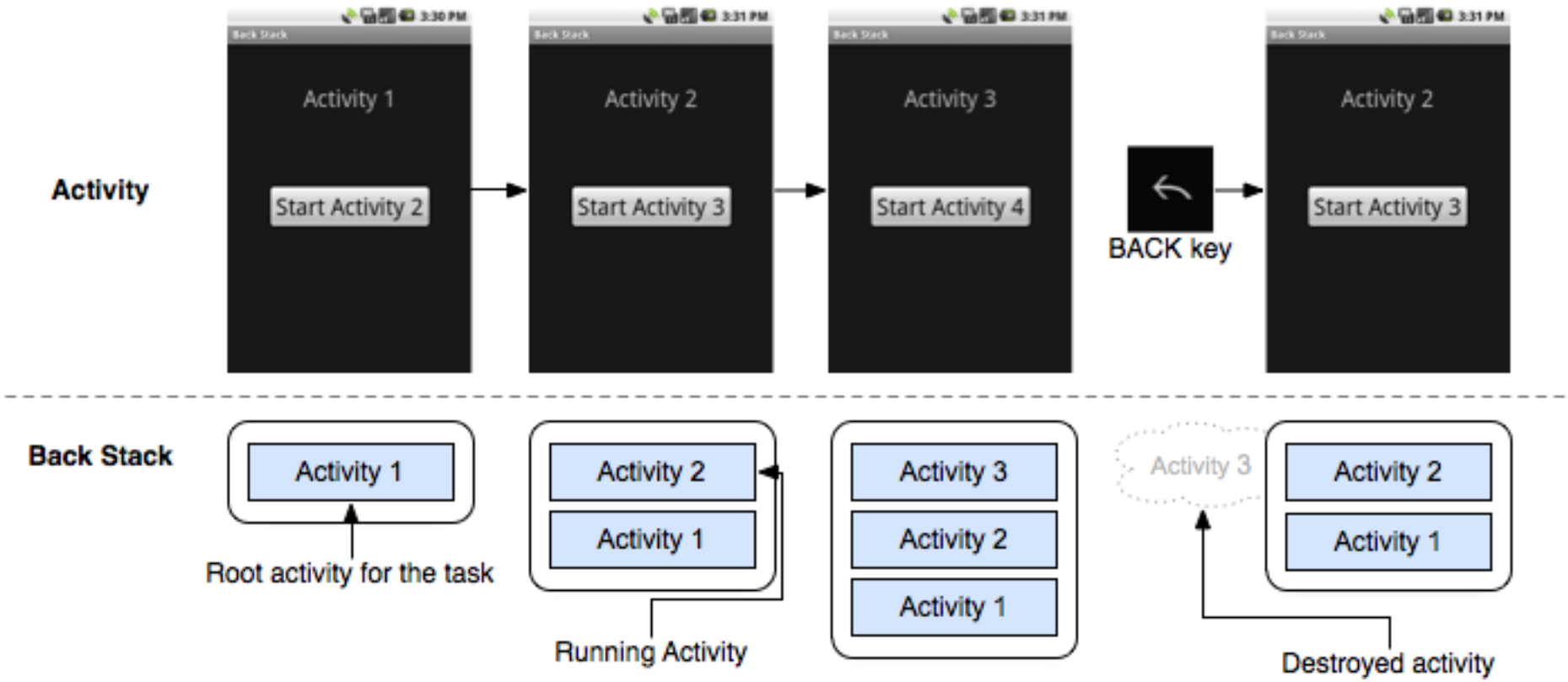
Introduction



- Élément de base : fourni un écran contenant des objets graphiques avec lesquels l'utilisateur peut interagir
- Chaque activity possède une fenêtre sur laquelle les éléments graphiques sont dessinés
- Un application est composé d'un ensemble d'*activities* formant une IHM cohérante
- Chaque application possède une *activity* principale démarrée automatiquement
- Il existe un plugin Eclipse pour concevoir la vue d'une Activity (WYSIWYG).

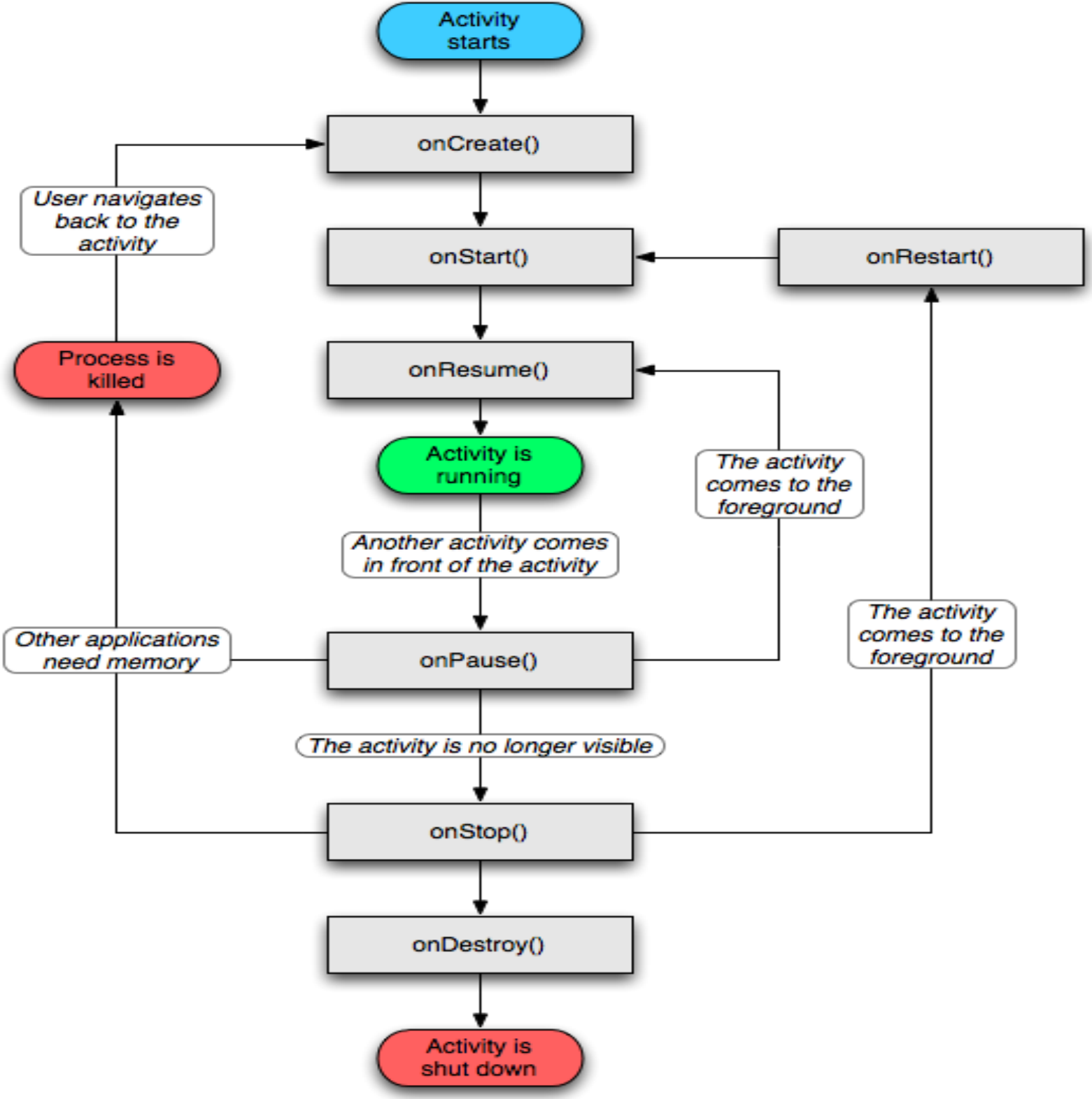
Activity

Stack



Activity

Cycle de vie



Squelette d'une activity

- Créer une classe qui hérite de Activity (ou d'une de ses sous classes)

- Surcharger les méthodes suivantes :
  - onCreate(Bundle bundle)
  - onPause()
  - onSaveInstanceState(Bundle outState)

- Pour chaque méthode surchargée il faut appeler la méthode de la classe mère

- La vue affichée par l'activity est une hiérarchie d'objet de type View

- Chaque objet de type View gère son affichage et ses évènements

Squelette d'une activity

AndroidManifest.xml

- android:allowTaskReparenting
- android:alwaysRetainTaskState
- android:clearTaskOnLaunch
- android:configChanges
- android:enabled
- android:excludeFromRecents
- android:exported
- android:finishOnTaskLaunch
- android:hardwareAccelerated
- android:icon
- android:label
- android:launchMode
- android:name
- android:noHistory
- android:permission
- android:process
- android:screenOrientation
- android:stateNotNeeded
- android:taskAffinity
- android:windowSoftInputMode

```
<manifest ... >
  <application ... >
      <activity android:name=".
ExampleActivity" />
      ...
  </application ... >
  ...
</manifest >
```

Squelette d'une activity

```
public class CalendarActivity extends Activity {

    static final int DAY_VIEW_MODE = 0;
    static final int WEEK_VIEW_MODE = 1;

    private SharedPreferences mPrefs;
    private int mCurViewMode;

    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        SharedPreferences mPrefs = getSharedPreferences();
        mCurViewMode = mPrefs.getInt("view_mode" DAY_VIEW_MODE);
    }

    protected void onPause() {
        super.onPause();

        SharedPreferences.Editor ed = mPrefs.edit();
        ed.putInt("view_mode", mCurViewMode);
        ed.commit();
    }
}
```

Squelette d'une activity

```
private void pickContact() {
    // Create an intent to "pick" a contact, as defined by the content provider URI
    Intent intent = new Intent(Intent.ACTION_PICK, Contacts.CONTENT_URI);
    startActivityForResult(intent, PICK_CONTACT_REQUEST);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    // If the request went well (OK) and the request was PICK_CONTACT_REQUEST
    if (resultCode == Activity.RESULT_OK && requestCode == PICK_CONTACT_REQUEST) {
        // Perform a query to the contact's content provider for the contact's name
        Cursor cursor = getContentResolver().query(data.getData(),
        new String[] {Contacts.DISPLAY_NAME}, null, null, null);
        if (cursor.moveToFirst()) { // True if the cursor is not empty
            int columnIndex = cursor.getColumnIndex(Contacts.DISPLAY_NAME);
            String name = cursor.getString(columnIndex);
            // Do something with the selected contact's name...
        }
    }
}
```

Layout XML

- Permet de ne pas utiliser de code Java pour la composition de l'interface utilisateur
- Association des composants (widgets) avec les conteneurs
  - organisation et placement des widgets
- Fichier sauvegardé dans `/res/layout/`
  - souvent nommé `/res/layout/main.xml` pour l'activité principale
- À chaque modification des layouts, `R.java` va être modifié *via* l'outil Android Asset Packaging Tool (AAPT)
  - Ainsi, les identifiants des éléments pourront être accessible *via* findViewById(R.id.element)

Layout XML

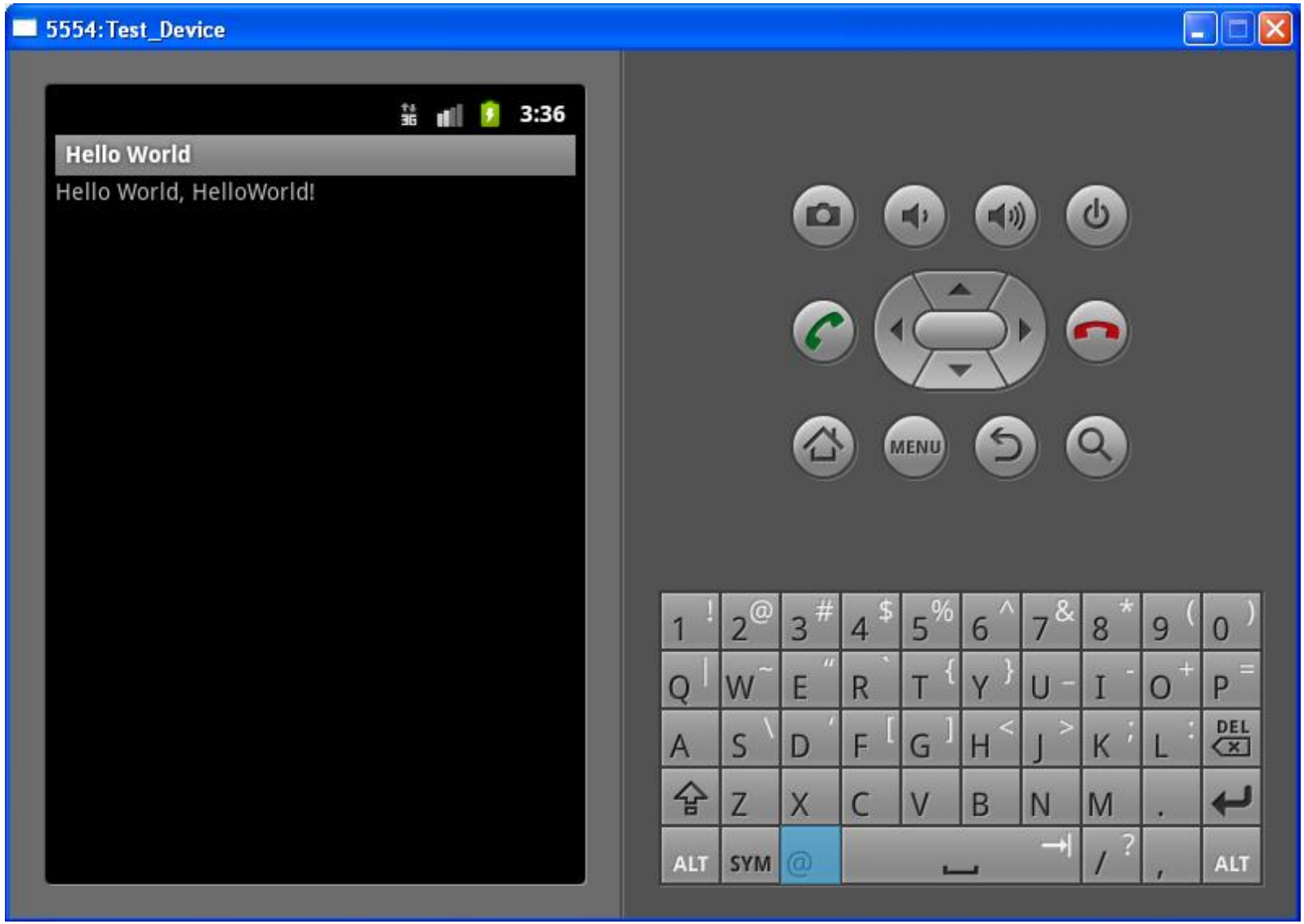
Exemple

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/label"
  android:text="Hello World, HelloWorld!"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"/>
```

- Ici, un exemple de label qui va afficher le texte Hello World, HelloWorld!
- TextView tv = (TextView) findViewById(R.id.label)
- Si le texte n'est pas sensé être modifié, ne pas déclarer d'id
- fill_parent = remplit le conteneur parent (ici en largeur)
- wrap_content = n'occupe que le nécessaire (ici en hauteur)

Layout XML

Exemple



Layout XML

Utiliser les ressources

- Fichier /res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:id="@+id/label"
  android:text="@string/hello_world"
  android:text="Hello World, HelloWorld!"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"/>
```

- Fichier /res/values/hello.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string name="hello_world">
    Hello World, HelloWorld!
  </string>
</resources>
```

Layout XML

Déclaration dans l'activité

```
...
public class NyanActivity extends Activity {
  ...
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    TextView tv = (TextView) findViewById(R.id.label);
    tv.setText("Hello, Nyan!");
  }
  ...
}
```

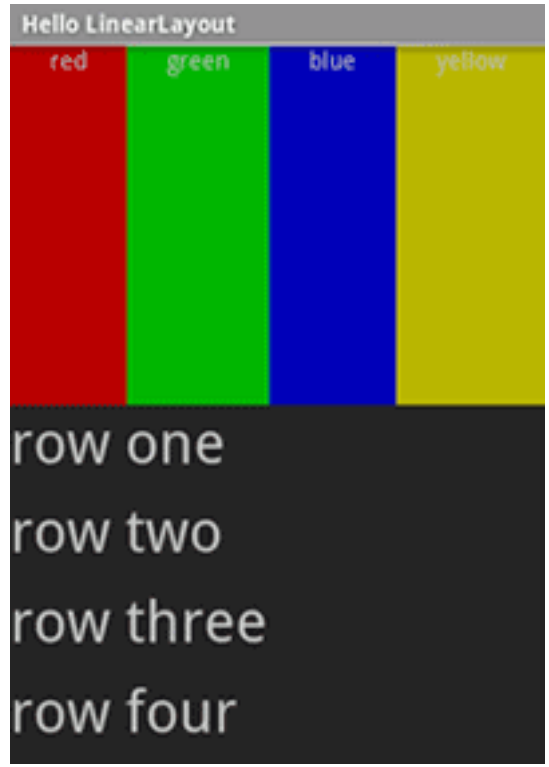
Conteneurs

- Placement des Widgets (composants graphiques) sur l'écran
- Notions de :
  - Orientation (horizontal, vertical)
    - android:orientation="vertical"
  - Remplissage
    - android:layout_width, android:layout_height
    - taille en px, wrap_content, fill_parent
  - Poids (android:layout_weight)
  - Gravité
    - android:gravity
    - left, center_horizontal, right, etc.
  - Padding (android:padding, ou android:paddingRight, android:paddingTop, android:paddingBottom, android:paddingLeft)

Conteneurs

LinearLayout (1)

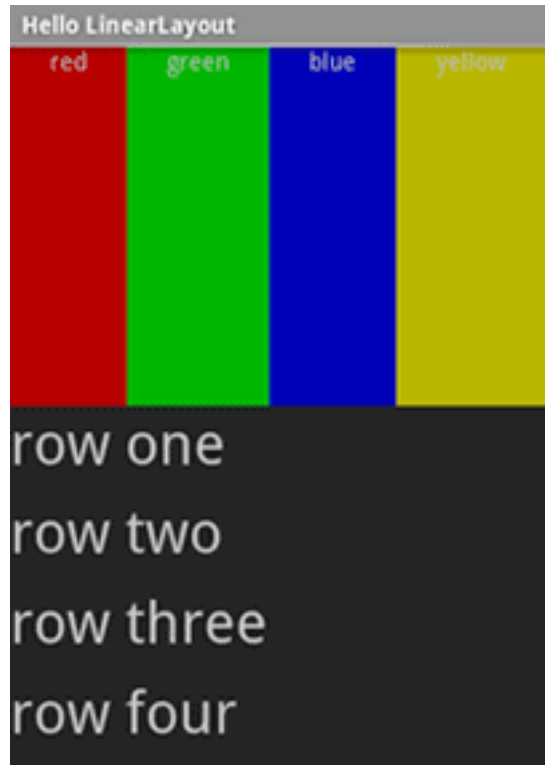
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<LinearLayout
android:orientation="horizontal"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:layout_weight="1">
<TextView
android:text="red"
android:gravity="center_horizontal"
android:background="#aa0000"
android:layout_width="wrap_content"
android:layout_height="fill_parent"
android:layout_weight="1"/>
...
</LinearLayout>
```



Conteneurs

LinearLayout (2)

```
<LinearLayout
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:layout_weight="1">
<TextView
android:text="row one"
android:textSize="15pt"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:layout_weight="1"/>
...
</LinearLayout>
</LinearLayout>
```



Conteneurs

RelativeLayout (1)

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<TextView
android:id="@+id/label"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:text="Type here:"/>
<EditText
android:id="@+id/entry"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:background="@android:drawable/editbox_background"
android:layout_below="@id/label"/>
```

Conteneurs

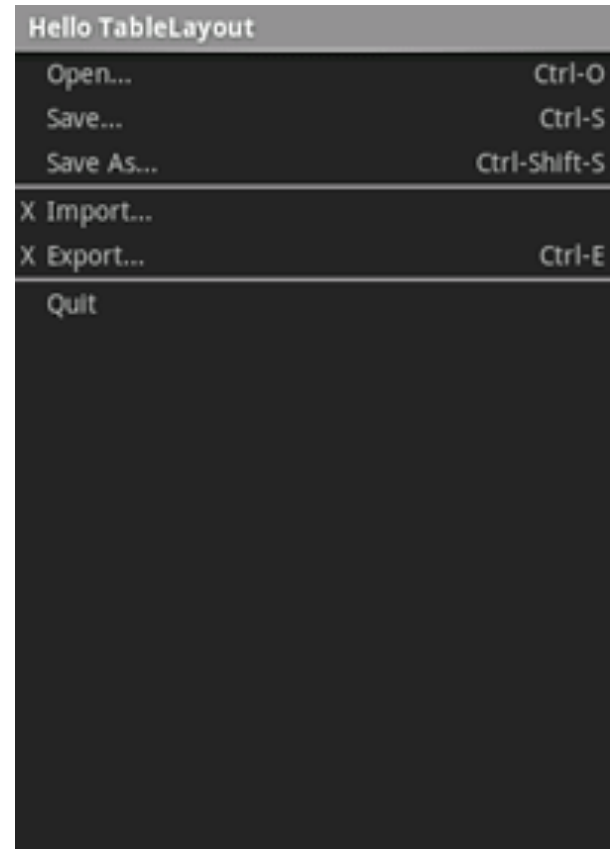
RelativeLayout (2)

```
<Button
android:id="@+id/ok"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_below="@id/entry"
android:layout_alignParentRight="true"
android:layout_marginLeft="10dip"
android:text="OK" />
<Button
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_toLeftOf="@id/ok"
android:layout_alignTop="@id/ok"
android:text="Cancel" />
</RelativeLayout>
```

Conteneurs

TableLayout (1)

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:stretchColumns="1">

<TableRow>
<TextView
android:layout_column="1"
android:text="Open..."
android:padding="3dip" />
<TextView
android:text="Ctrl-O"
android:gravity="right"
android:padding="3dip" />
</TableRow>
...
<View
android:layout_height="2dip"
android:background="#FF909090" />
...
</TableLayout>
```



Conteneurs

ScrollView

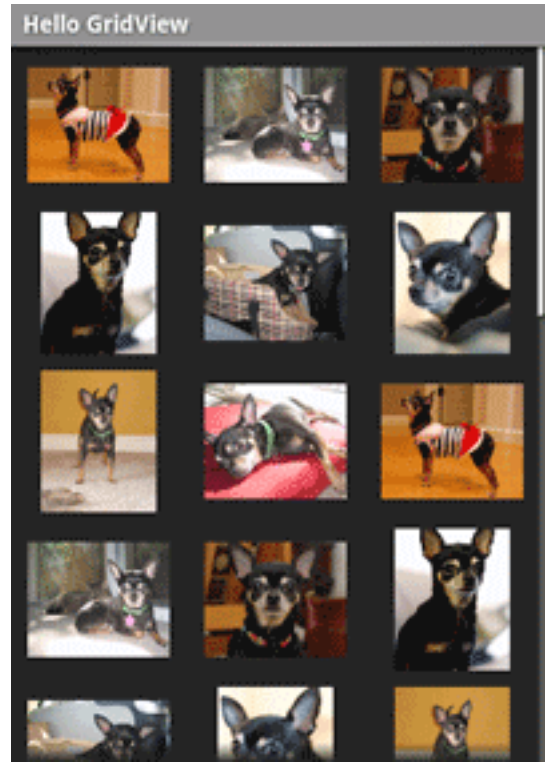
- Rend possible le scrolling de la vue lorsque le contenu dépasse la taille de l'écran
- En général conteneur de plus haut niveau

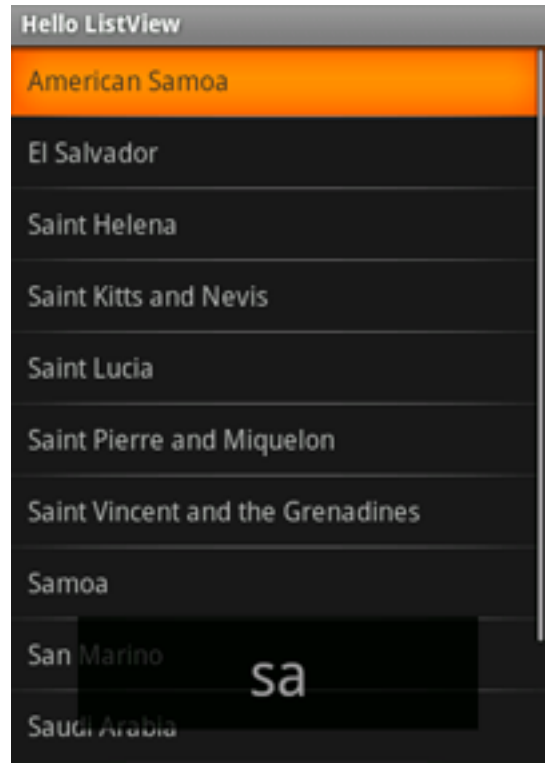
```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content">
  <TableLayout ...>
    ...
  </TableLayout>
  ...
</ScrollView>
```

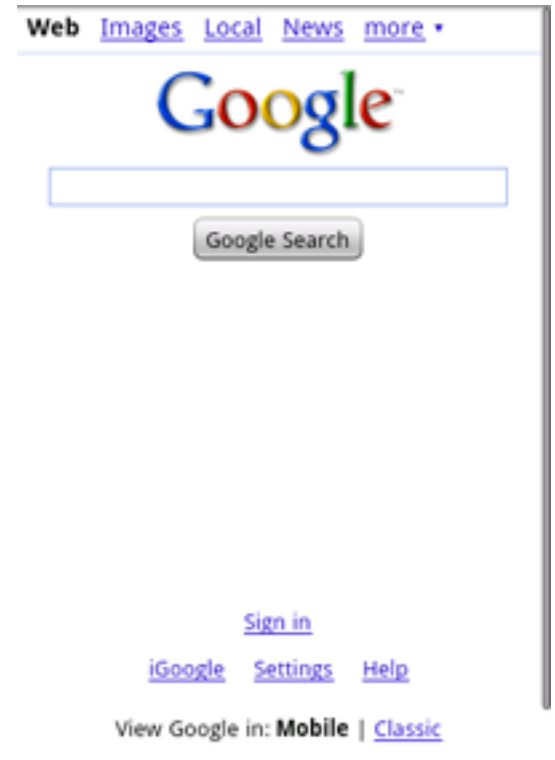
Conteneurs

Autres

- Grid View
- Tab Layout
- List View
- Web View
- etc.







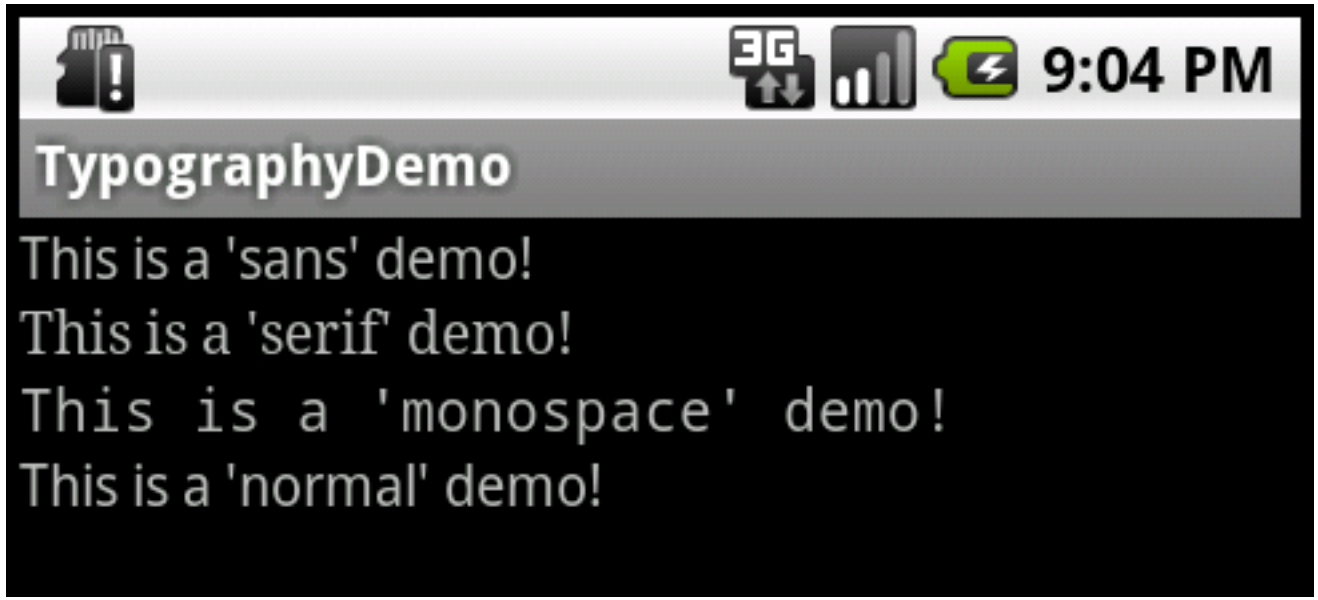
Widget

- Comme avec Swing (ou AWT), il existe un nombre important de widgets
- Ce sont des objets permettant d'interagir avec l'utilisateur en affichant des informations, demandant des entrées, etc.
- En général, déclarés dans le fichier XML de layout et accessibles *via* findViewById(R.id.element)
- JavaDoc complète !

- Package android.widget
- Héritent de View

Widget

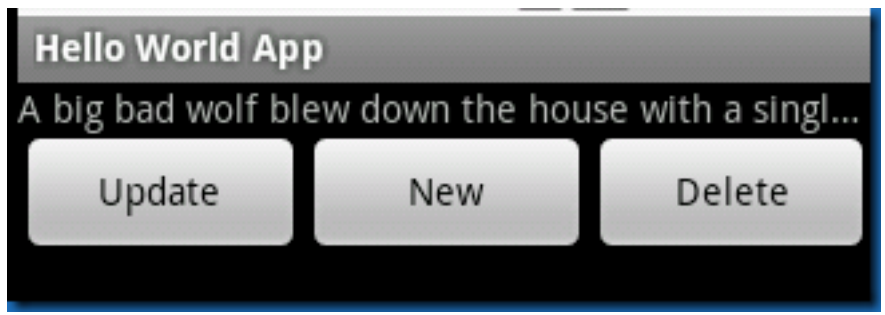
Exemple TextView



http://mobile.tutsplus.com/tutorials/android/customize-android-fonts/

Widget

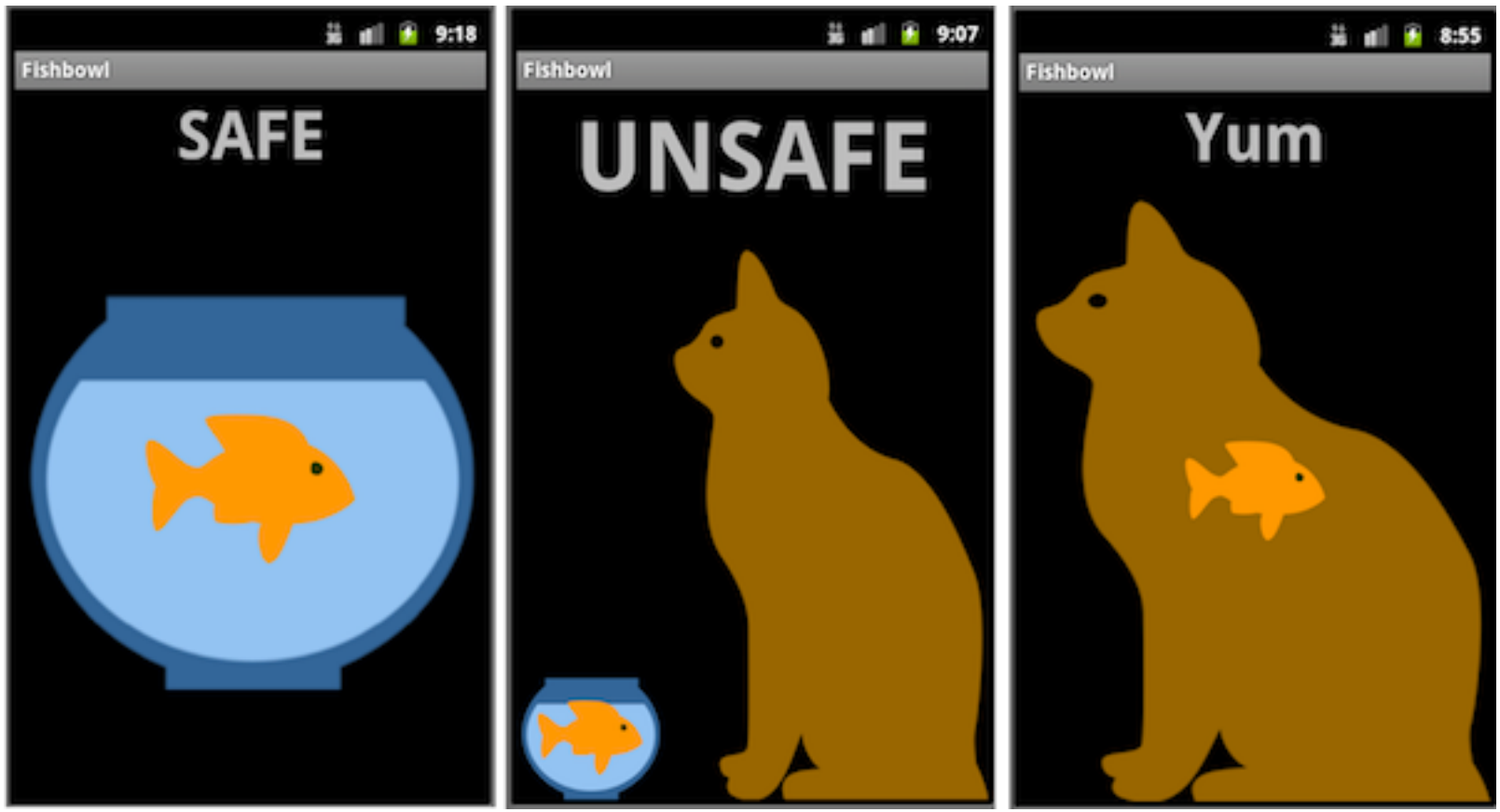
Exemple Button



http://blog.webagesolutions.com/archives/154

Widget

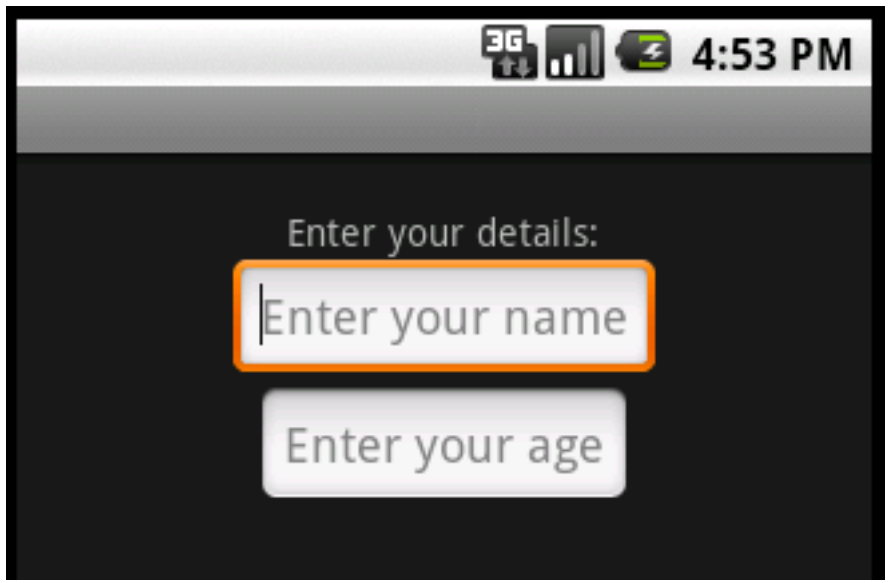
Exemple ImageView



http://www.onesecretlife.com/archives/3762

Widget

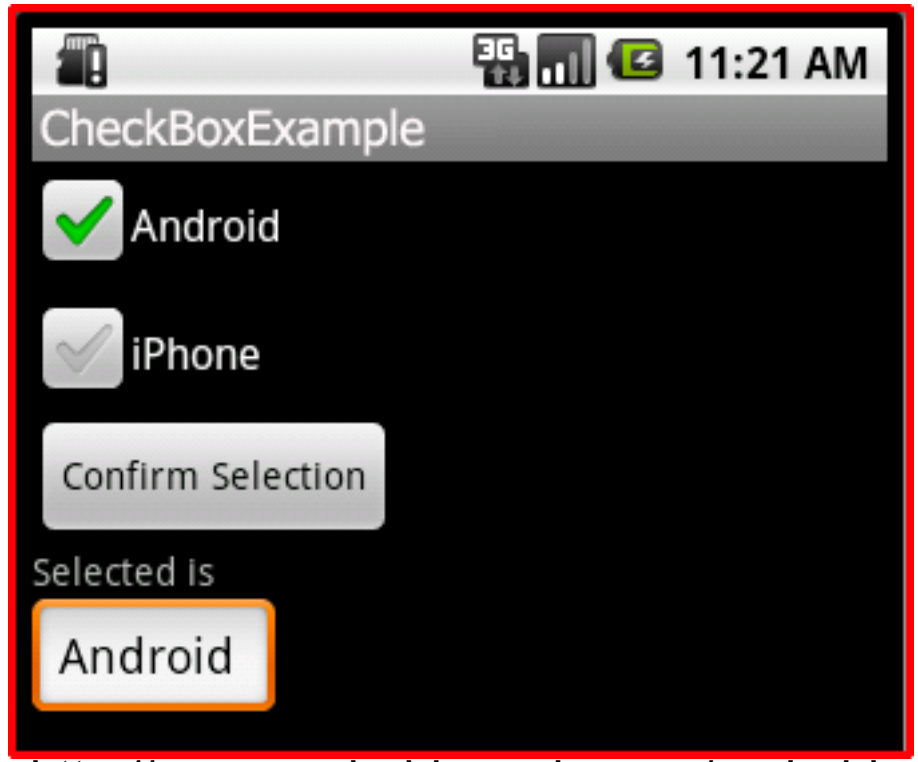
Exemple EditText



http://androidforbeginners.blogspot.com/2010/03/default-text-for-edittext-heres-hint.html

Widget

Exemple CheckBox



http://www.androidpeople.com/android-checkbox-example

Gestion des événements

Introduction

- Chaque objet de type View peut gérer ses évenements

- La classe View possède un ensemble d'interfaces pour chaque type d'événement :
  - View.OnClickListener
  - View.OnLongClickListener
  - View.OnKeyListener
  - View.OnTouchListener

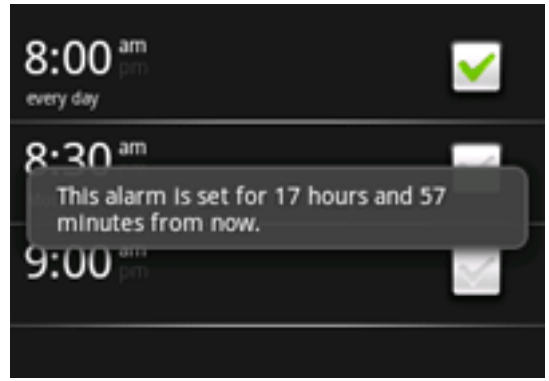
- L'ensemble des objets View fonctionne dans un même thread (même problème que pour Swing!)

Gestion des événements

Exemple

```
final EditText edittext = (EditText) findViewById(R.id.edittext);
edittext.setOnKeyListener(new OnKeyListener() {
    public boolean onKey(View v, int keyCode, KeyEvent event) {
        // If the event is a key-down event on the "enter" button
        if ((event.getAction() == KeyEvent.ACTION_DOWN) &&
            (keyCode == KeyEvent.KEYCODE_ENTER)) {
          // Perform action on key press
          Toast.makeText(HelloFormStuff.this, edittext.getText(), Toast.LENGTH_SHORT).show();
          return true;
        }
        return false;
    }
});

final CheckBox checkbox = (CheckBox) findViewById(R.id.checkbox);
checkbox.setOnClickListener(new OnClickListener() {
    public void onClick(View v) {
        // Perform action on clicks, depending on whether it's now checked
        if (((CheckBox) v).isChecked()) {
            Toast.makeText(HelloFormStuff.this, "Selected", Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(HelloFormStuff.this, "Not selected", Toast.LENGTH_SHORT).show();
        }
    }
});
```

Notifications

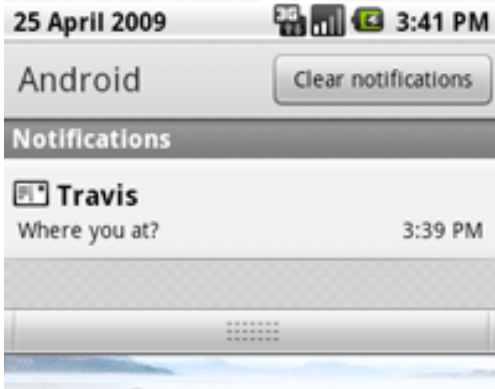


Toast

- Message de type « pop-up »
- Apparaît au-dessus de l'activité, temporairement
- Peut être affiché par une activité ou un service

```
Context context = getApplicationContext();
CharSequence text = "GuiiiiGuiiiiiiiii!";
int duration = Toast.LENGTH_SHORT;

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

http://developer.android.com/guide/topics/ui/notifiers/toasts.html



Notifications

Barre de status (1)

- Notification apparaissant dans la barre de status
- Nécessite
  - Icône
  - Titre
  - Message
  - PendingIntent
- Possibilités d'ajouter
  - Ticker
  - Son
  - Vibration
  - Notifications visuelles (LED)

http://developer.android.com/guide/topics/ui/notifiers/notifications.html

Notifications

Barre de status (2)

```
String ns = Context.NOTIFICATION_SERVICE;
NotificationManager mNotificationManager =
                    (NotificationManager) getSystemService(ns);
int icon = R.drawable.notification_icon;
CharSequence tickerText = "Hello";
long when = System.currentTimeMillis();
Notification notification =
                        new Notification(icon, tickerText, when);
Context context = getApplicationContext();
CharSequence contentTitle = "My notification";
CharSequence contentText = "Hello World!";
Intent notificationIntent = new Intent(this, MyClass.class);
PendingIntent contentIntent = PendingIntent.getActivity(this, 0, notificationIntent, 0);
notification.setLatestEventInfo(context, contentTitle,
                            contentText, contentIntent);
```

Android

Content Provider

Qu'est-ce que c'est ?

- Fournisseur de contenu
- Les ressources sont accessibles *via* des Uris
  - content://
  - content://identifiant/ressource
  - Exemple :

    content://angry.birds/green.pig.king

- Possibilités de créer, lire, mettre à jour et supprimer des ressources
- Abstraction sur la manière dont sont stockées les ressources
  - Dépend du fournisseur de contenu

Interroger un fournisseur de contenu

- Constantes dans le package android.provider
  - Contact.People.CONTENT_URI
  - Contact.Phones.CONTENT_URI
  - etc.
- Utilisation de la méthode managedQuery(...)
  - Prend en paramètres :
    - L'Uri du fournisseur de contenu ou de l'instance de l'objet
    - Les propriétés (« colonnes » dans un SGBD)
    - Une contrainte de sélection (WHERE)
    - Paramètres de la contrainte de sélection
    - Paramètre de tri
  - Retourne une instance de Cursor
    - *Design pattern Iterator* !

Interroger un fournisseur de contenu

Exemple (1)

```
private static final String[] PROPERTIES = new String[] {
  Contact.People._ID, Contact.People.DISPLAY_NAME,
  Contact.People.NUMBER, Contact.People.TYPE } ;

...

Cursor cursor = managedQuery(Contact.People.CONTENT_URI, PROPERTIES, null, null, null);
```

Interroger un fournisseur de contenu

Exemple (2)

```
if (cursor.moveToFirst()) {
  int peopleID;
  String peopleDisplayName;
  String peopleNumber;
  String peopleNumberType;
  int idIdx = cursor.getColumnIndex(Contact.People._ID);
  int displayNameIdx = cursor.getColumnIndex(Contact.People.DISPLAY_NAME);
  int numberIdx = cursor.getColumnIndex(Contact.People.NUMBER);
  int peopleNumberType = cursor.getColumnIndex(Contact.People.TYPE);

  do {
    peopleId = cursor.getInteger(idIdx);
    peopleDisplayName = cursor.getString(displayNameIdx);
    ...
  } while (cursor.moveToNext());
  ...
}
...
```

Ajouter, supprimer, mettre à jour

- ContentResolver
- Méthodes :
  - final Uri insert(Uri url, ContentValues values)
  - final int delete(Uri url, String where, String[] args)
  - final int update(Uri url, ContentValues values, String where, String[] args)
- ContentValues
  - Contient un ensemble de clés/valeurs
  - Ici, identifiant de la colonne et valeur
  - Ajout avec la méthode put(String key, value)

Créer un fournisseur de contenus (1)

- Étendre la classe ContentProvider
  - Redéfinition des méthodes :
    - boolean onCreate();
    - Cursor query(Uri url, String[] properties, String where, String[] args, String sort)
    - Uri insert(Uri url, ContentValues values)
    - int update(Uri url, ContentValues values, String where, String[] args)
    - int delete(Uri url, String where, String[] args)
    - String getType(Uri url)
- Type MIME :
  - vnd.<custom>.cursor.dir/type
  - vnd.<custom>.cursor.item/type

Créer un fournisseur de contenus (2)

- Fournir une Uri

```
public final static Uri CONTENT_URI = Uri.parse("content://fr.unilim.java.avance.Miahou/cats");
```

- Déclarer les noms de colonnes (propriétés)
  - Classe public static final implémentant BaseColumns

```
public static final class MiahouColumns implements BaseColumns {
  public static final Uri CONTENT_URI = Uri.parse("content://fr.unilim.java.avance.
Miahou/MiahouColumns");
  public static final String NAME = "name";
  public static final String BELL = "bell";
  ...
}
```

- Déclarer le fournisseur dans le manifest

```
<provider android.name=".Miahou"
        android:authorities="fr.unilim.java.avance.Miahou" />
```

À voir...

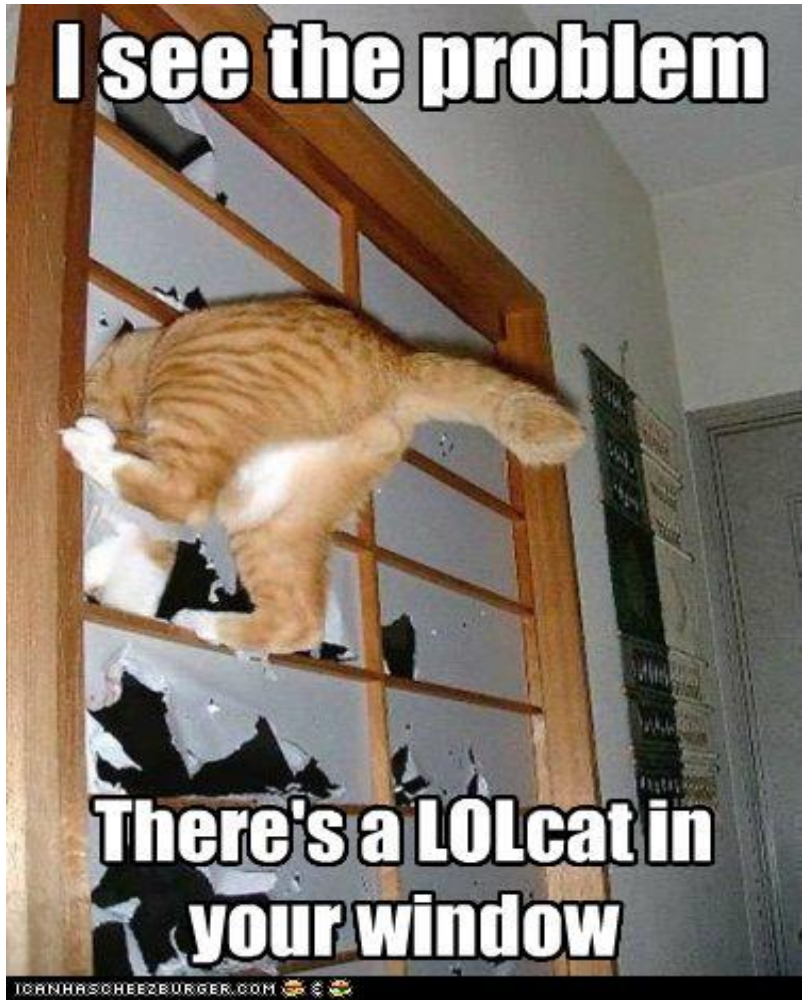
- Nécessité d'avoir certaines permissions afin d'accéder aux fournisseurs de contenu (READ_CONTACTS par exemple)

- Depuis Android 2.0
  - Utilisation de ContactsContract
  - ContactsContract.* (Contacts, RawContacts, PhoneLookup ...

- managedQuery(...) *deprecated*
  - Utilisation de CursorLoader
  - Éviter les blocages de l'UI (AsyncTaskLoader)

Android

Graphisme



Différentes stratégies

- 2D :
  - dessiner directement dans un objet View
  - dessiner dans un Canvas
- 3D :
  - OpenGL ES
    http://developer.android.com/guide/topics/graphics/opengl.html
  - renderscript
    http://developer.android.com/guide/topics/renderscript/index.html

View

```
public class CustomDrawableView extends View {
    private ShapeDrawable mDrawable;

    public CustomDrawableView(Context context) {
        super(context);

        int x = 10;
        int y = 10;
        int width = 300;
        int height = 50;

        mDrawable = new ShapeDrawable(new OvalShape());
        mDrawable.getPaint().setColor(0xff74AC23);
        mDrawable.setBounds(x, y, x + width, y + height);
    }

    protected void onDraw(Canvas canvas) {
        mDrawable.draw(canvas);
    }
}
```

Canvas

- implémentation de la méthode onDraw (Canvas canvas) d'un objet View
- appel de la méthode invalidate ()

- étendre la classe SurfaceView
- implémenter les classes SurfaceHolder.Callback et Thread
- http://developer.android.com/resources/samples/LunarLander/src/com/example/android/lunarlander/LunarView.html

Android

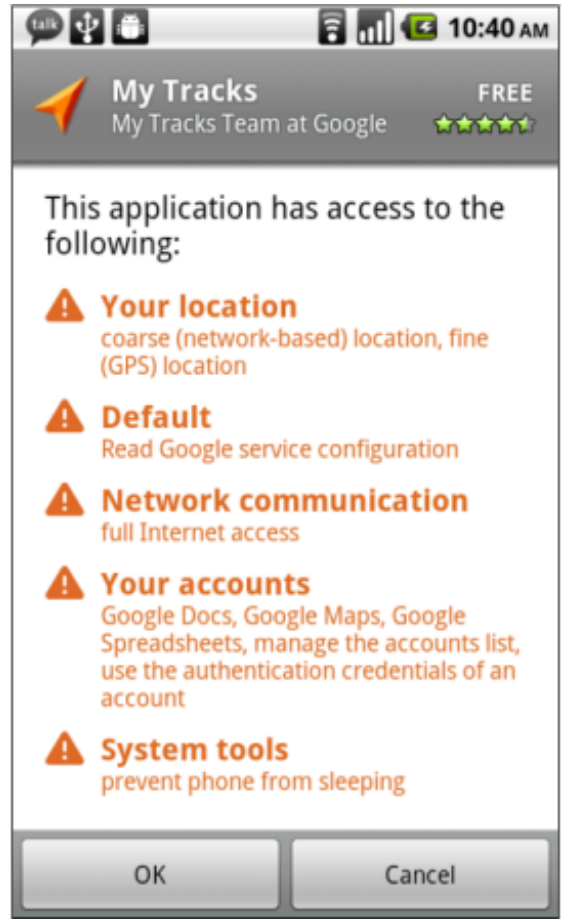
Sécurité

Sécurité dans Android

- Plusieurs mécanismes de sécurité
- Système de permissions
- Signature de l'application
- Mécanisme de SandBox
  - Chaque application dans une instance de Dalvik
  - Chaque application dans un processus Linux

Obtenir des permissions

- Utilisation de uses-permission dans le manifest
  - voir partie relative au manifest
- « Package » android.Manifest.permission
- Exemples :
  - BRICK : peut désactiver l'appareil (dangereux ;-))
  - INTERNET : autorise l'ouverture de sockets
  - READ_SMS : permet la lecture de SMS
- <uses-permission android:name="android.permission.INTERNET" />



http://news.cnet.com/8301-27080_3-20008518-245.html

Demander des permissions (1)

- Déclarer les permissions possibles

```
<permission
  android:name="nyan.cat.FLY"
  android:label="Nyan Cat can fly!"
  android:description="Allow the application to make Nyan Cat
                       flying" />
```

- Indiquer l'endroit où s'applique la permission
  - Activité, services, broadcast receivers
  - Exemple :

```
<activity
  ...
  android:permission="nyan.cat.FLY">
  ...
</activity>
```

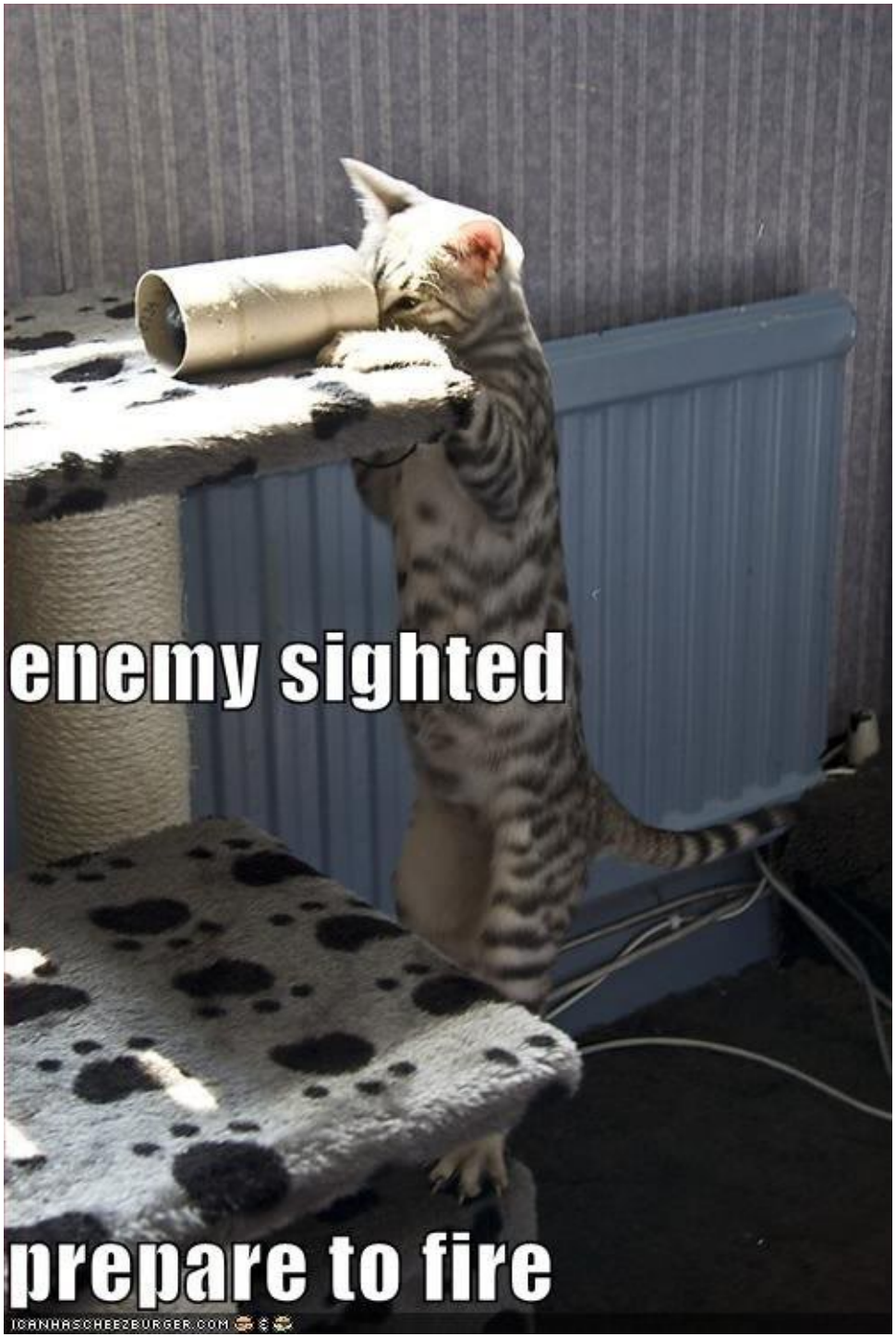
Demander des permissions (2)

- Fournisseurs de contenu bénéficient de deux attributs supplémentaires :
  - android:readPermission
  - android:writePermission

- Services : possibilités de vérifier les permissions grâce à la méthode

  Service#checkCallingPermission(String permission)
  - il existe d'autres méthodes pour tester les permissions (voir classe Service)

- Possibilité d'inclure une permission à la méthode sendBroadcast (Intent intent, String receiverPermission)



Intents

Les Intents

- Système de messages abstrait contenant un ensemble de données permettant la réalisation d'actions
- Une intention peut être fournie à tous les types de composants : activités, services, broadcast receivers
- Selon l'intention, Android va choisir la meilleurs action à exécuter
- Intent peut prendre en paramètre de constructeur :
  - une action
  - une action et une Uri
  - un contexte et une classe
  - une action, une Uri, un contexte et une classe

Les Intents

Envoyer une intention

- Démarrer une activité :
  - Context#startActivity, Activity#startActivityForResult (et résultat avec Activity#setResult)
- Démarrer un service :
  - Context#startService
- Démarrer un broadcast receiver
  - Context#sendBroadcast

Les Intents

Paramètres

- Action : le type d'action à exécuter (ACTION_CALL, ACTION_EDIT...)
- Data : des données sous la forme d'une Uri (par exemple tel://<numéro de téléphone>)
- Category : le type de composant qui doit gérer l'intention (CATEGORY_LAUNCHER ...)
- Extra : données additionnelles qui peuvent être passées à une intension sous la forme de clés/valeurs
- Flags : options supplémentaires sur la gestion de l'intention (FLAG_ACTIVITY_NO_HISTORY ...)

Les Intents

Filters

- Déclaration des types d'intention auxquels le composant peut répondre

Exemple d'une application de gestion de notes

- Pour l'activité principale :

```
<intent-filter>
  <action android:name="android.intent.action.MAIN" />
  <category android:name="android.intent.category.LAUNCHER" />
</intent-filter>
```

- Autres filtres :

```
<intent-filter>
  <action android:name="android.intent.action.VIEW" />
  <action android:name="android.intent.action.EDIT" />
  <action android:name="android.intent.action.PICK" />
  <category android:name="android.intent.category.DEFAULT" />
  <data
     android:mimeType="vnd.android.cursor.dir/vnd.google.note" />
</intent-filter>
```

Android

Misc

Threads

- Toute opération longue peut être bloquante pour l'application (freeze)
  - Utilisation des threads (tâches) !
- Possibilité d'utilisation des threads Java (Thread et Runnable)
- Possibilité d'utilisation d'un Handler
  - Envoi de messages
    - sendMessage pour l'envoi
    - obtainMessage pour en recevoir
  - ou de runnable
    - post pour attacher un thread au handler

Threads

Exemple Handler

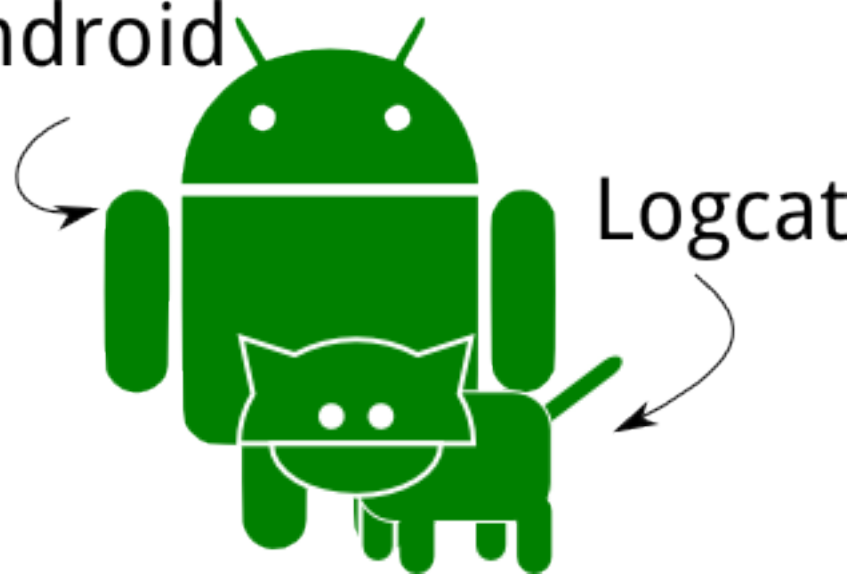
```
Handler handler = new Handler() { // Handler declaration
  @Override
  public void handleMessage(Message msg) {
    // Do something (set a progress bar for instance)
  }
}
...
// Add the onStart method in the Activity
public void onStart() {
  super.onStart();
  Thread thread = new Thread(new Runnable() {
    public void run() {
      // Do something
      Message message = handler.obtainMessage();
      // Do something on message
      handler.sendMessage(message)
    }
  }
  thread.start();
}
```

Threads

AsyncTask<Parameter, Progression, Result>

- AsyncTask a été créée pour gérer les petites tâches
- Ressemble à SwingWorker
- Trois paramètres de généricité :
  - Paramètre
  - Progression
  - Résultat
- Méthodes pouvant être redéfinies :
  - doInBackground(Paramètre ... params)
  - onCancelled(Résultat res)
  - onProgressUpdate(Progression ... progs)
  - onPostExecute(Résultat res)
- Possibilité de publier un résultat via la méthode publishProgress(Progression prog)

Logcat



- Framework de journalisation léger
  - android.util.Log

- Plusieurs niveaux :
  - Log.v(String tag, String message[, String Throwable])
  - Log.d(String tag, String message[, String Throwable])
  - Log.i(String tag, String message[, String Throwable])
  - Log.w(String tag, String message[, String Throwable])
  - Log.e(String tag, String message[, String Throwable])

- Récupérer les logs :
  - $ adb logcat (shell)
  - Window > Show View > Logcat (Eclipse)

Android Debug Bridge

- Outil pour communiquer avec un AVD ou un périphérique Android
- Commande adb ou outils dans Eclipse
- Permet :
  - l'installation d'application : adb install une_app.apk
  - d'avoir un shell sur le périphérique : adb shell
  - récupérer les logs : adb logcat

Internationalisation

- Internationalisation permet d'avoir une application dans plusieurs langues
- Android facilite la mise en place de l'internationalisation
- Utiliser les ressources pour stocker les chaînes de caractères
  - /res/values/
- Le système de dossier permet l'internationalisation
  - /res/values/
  - /res/values-en/
  - /res/values-fr/
  - /res/values-fr-rCA/
- /res/values/ agit comme ressource par défaut (important d'en spécifier)

Gestion des ressources

- Appliquer le principe d'internationalisation à d'autres ressources
- /res/<type>-<qualifier>/
  - /res/drawable-xlarge/
- Les *qualifier* peuvent s'appliquer à tous les types de ressources (même celles concernant le langage)
- Exemples de *qualifier* :
  - port ou land pour l'orientation
  - ldpi, mdpi, ... pour la densité
  - notouch, stylus, finger
  - nonav, dpad, trackball, wheel
  - v3, v7, v10, ...
- http://developer.android.com/guide/topics/resources/providing-resources.html

Capteurs

Accéléromètre

- Rotation par défaut ! :-D
- Possibilité de définir un sens par défaut et invariable pour une activité :
  - attribut : android:screenOrientation
  - valeurs : portrait ou landscape
- Possibilité de gérer manuellement le changement d'orientation
  - utiliser l'attribut: android:configChanges="orientation"
  - implémenter la méthode onConfigurationChanged(Configuration newConfig)
- Si changement de layout selon orientation :
  - res/layout-port/main.xml (portrait)
  - res/layout-land/main.xml (paysage)
  - même extensions de dossier pour les autres ressources

Capteurs

Localisation (1)

- Obtenir la localisation nécessite des permissions :
  - android.permission.ACCESS_COARSE_LOCATION (NETWORK_PROVIDER)
  - android.permission.ACCESS_FINE_LOCATION (GPS_PROVIDER)
- Étapes :
  - Obtenir une instance de LocationManager

```
LocationManager locationManager =
        this.getSystemService(Context.LOCATION_SERVICE)
```

  - Créer une instance de LocationListener pour gérer les évènements liés au service de localisation (nouvelle localisation, activation ou désactivation d'un provider...)

```
LocationListener locationListener = new LocationListener() {
  public void onLocationChanged(Location location) {...}
  public void onStatusChanged(String provider, int status, Bundle extras) {...}
  public void onProviderEnabled(String provider) {...}
  public void onProviderDisabled(String provider) {...}
};
```

Capteurs

Localisation (2)

- Déclarer le listener :

locationManager.requestLocationUpdates(String provider, long minTime, float minDistance, PendingIntent intent)

  - provider peut être :
    - LocationManager.GPS_PROVIDER
    - LocationManager.NETWORK_PROVIDER
    - LocationManager.PASSIVE_PROVIDER
  - exemple :

```
locationManager.requestLocationUpdates(
    LocationManager.NETWORK_PROVIDER, 0, 0, locationListener)
```

Tests unitaires

- Permettent de tester une application
- Définition des comportements attendus
  - bref, des tests unitaires...
  - développement orienté tests
- Possibilité de tester une application Android
  - tester la couche métier (classique)
  - tester l'interface utilisateur
- Pas vraiment trivial avec Android...

Tests unitaires

Tester une activité

- Étendre la classe ActivityInstrumentationTestCase2
- Définitions
  - dans le constructeur :

```
super("<package>", <Activity>.class);
```

  - setUp(), tearDown()
  - définition des tests
- assert* pour faire les assertions
  - assertTrue, assertEquals, etc.
- Obtenir l'activité avec la méthode getActivity()
- Possibilité d'utiliser la classe R
- Possibilité d'exécuter une tâche dans le thread de l'interface utilisateur avec Activity#runOnUiThread

Tests unitaires

- Possibilité de tester tous les composants Android :
  - ProviderTestCase2 (fournisseurs de contenus)
  - ServiceTestCase (services)
  - ...
  - Package android.test
- Possibilité d'utiliser des « mocks objects »
  - android.test.mock
  - Cursor, Application, ContentProvider, Resources, ContentResolver...
- Autre framework de test unitaire:
  - Robotium (http://code.google.com/p/robotium/)

Native Development Kit

- Permet de programmer une partie d'une application en C ou C++
- À utiliser avec parcimonie...
- Permet d'utiliser des instructions SIMD par exemple

Langages de programmation

- En théorie tous les langages fonctionnant au dessus d'une JVM peuvent servir à créer des applications Android

- .java ⇒ .class ⇒ .dex
- .foo ⇒ .class ⇒ .dex

- JRuby

- Scala

- Clojure


The idiot in the jeep...he didn't really say
"Here, kitty, kitty, kitty"...did he?
ICANHASCHEEZBURGER.COM