Programmation sous Android

Gauthier Picard

Ecole Nationale Supérieure des Mines

2012

Cette présentation a été conçue par
Jean-Paul Jamont
(Université Pierre Mendès France, IUT de Valence)

Plan

Plan

Qu'est ce qu'Android ?

Un système d'exploitation open source

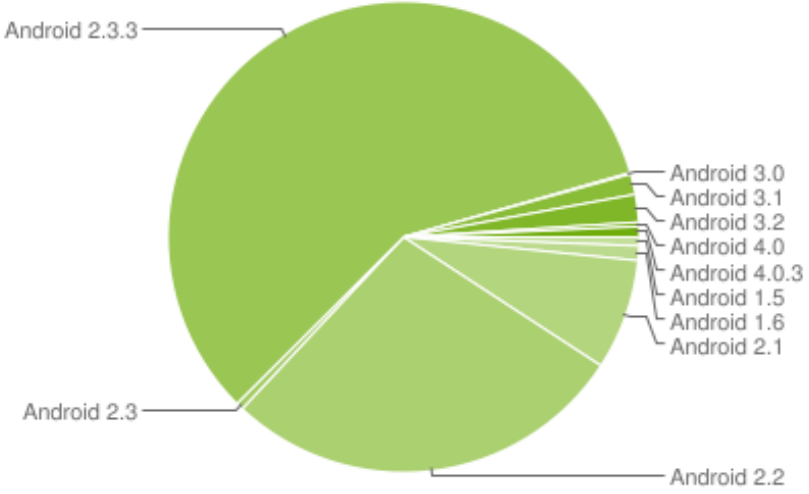
- Un système d'exploitation orienté dispositif mobiles
  - il s'agit donc d'un ensemble de logiciels qui sert d'interface entre le matériel (les composants du téléphone, d'une tablette...) et les logiciels applicatifs (ceux que vous allez développer).
- Un système d'exploitation open source
  - disponibilité du code,
  - importante communauté d'utilisateurs.
- Un système d'exploitation basé sur le noyau Linux
- Environnement de développement gratuit
  - Programmation en Java ou en langage C,
  - Kit de développement (SDK Android) disponible au lien `http://developer.android.com/sdk/index.html`

Les versions d'Android

De nombreuses versions...

Les différentes versions d'Android ont toutes des noms de desserts depuis la sortie de la version 1.5 et suivent un ordre alphabétique :

- 1.0 -- 2007 : Version peu connue (version du SDK distribuée avant la sortie du premier téléphone Android)
- 1.1 -- 2008 : Version incluse dans le premier téléphone, le HTC Dream
- 1.5 - Cupcake - 04/2009 : Dernière révision officielle en mai 2010
- 1.6 - Donut - 09/2009 : Dernière révision officielle en mai 2010
- 2.0 - Eclair - 2009 : Vite remplacée à cause de nombreux bugs par la 2.0.1 puis par la 2.1
- 2.1 - Eclair - 01/2010 : Dernière révision officielle en mai 2010
- 2.2 (2.2.3) - FroYo - 05/2010 : Dernière révision officielle en 2011
- 2.3 (2.3.7) - Gingerbread - 12/2010 : Version actuelle pour smartphones et petites tablettes
- 3.0 (3.2) - Honeycomb - 01/2011 : Version actuelle pour grandes tablettes et TV connectés
- 4.0 (4.0.3) - Ice Cream Sandwich - 10/2011 : Version unifiée pour Smartphone, Tablette et GoogleTV, fortement inspirée d'Honeycomb
- 5.0 - Jelly Bean : Version à venir



Les plateformes Android

De plus en plus d'équipements

Téléphones et autres PDA

Tablettes

Google TV

Android et ses concurrents 1/2

L'Open Handset Alliance

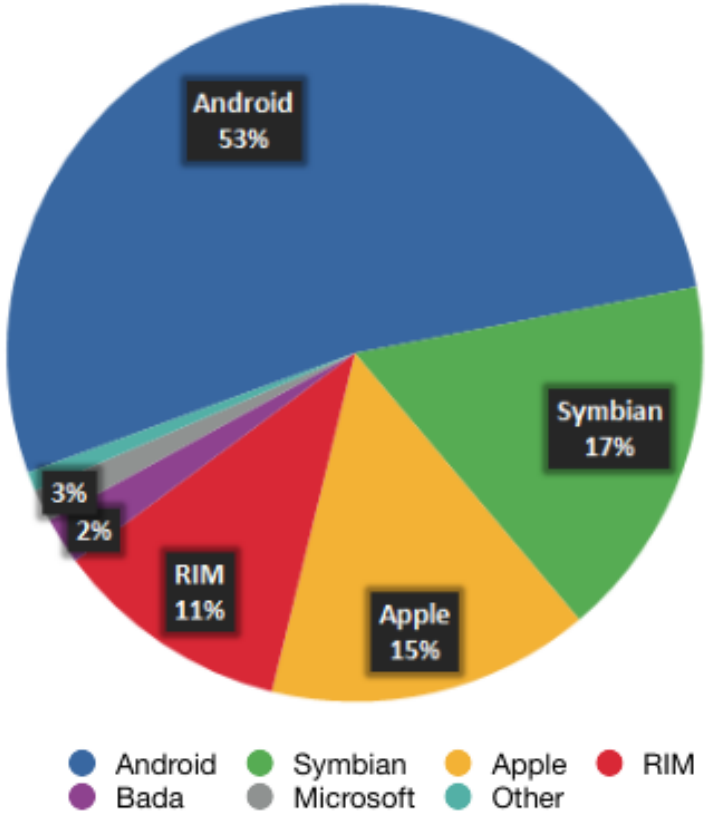
- **Date de création :** Le 5 novembre 2007 à l'initiative de Google
- **Objectifs :** Développer des normes ouvertes pour les appareils de téléphonie mobile
- **Membres :** 34 grands acteurs opérateurs de téléphonie mobile, fabricants de semi-conducteurs, d'appareils mobiles, de logiciels...

Concurrents d'Android

- Apple avec iOS
- Research In Motion (RIM) avec BlackBerry OS
- Microsoft avec Windows Phone
- Samsung avec Bada (même si Samsung utilise aussi Android)
- HP avec Palm webOS devenu webOS (Arrêté en 2011)
- Nokia avec Symbian OS (Arrêté en 2011, Nokia utilisera désormais Windows Phone).
- ...

Android et ses concurrents 2/2

Les smartphones vendus selon leur OS



Share of worldwide 2011 Q3 smartphone sales to end users by operating system, according to Gartner.

Les challenges

Fonctionnalités d'Android

- Framework d'application
- Machine virtuelle Dalvik
- Navigateur web intégré
- API graphique 2D et 3D
- SQLite
- Codecs audio et vidéo
- WiFi, EDGE, 3G, Bluetooth...
- Camera, GPS, Accéléromètre, compass...

Contraintes

- CPU cadencés 500-600 Mhz
- Faible mémoire RAM disponible
- Important temps d'accès (en écriture) sur disque flash
- Cycle de vie spécifique des applications (autonomie énergétique, ressources limitées)
- Faible débit et intermittence des réseaux
- Conception particulière des IHM :
  - Écran type : HVGA (320x480)
  - Utilisation en mode *portrait* ou *paysage*
  - Texte de petites polices peuvent être non lisible (DPI)
  - Faible résolution de touché de la dalle tactile (environ 25 pixels)

Télécharger/Distribuer des applications

Google Play Store (ancien Android Market)

Système standard de téléchargement/distribution d'applications.

- Pas de vérification des applications
- Navigation plus laborieuse que AppMarket (recherche par catégories, mots clés, prix)
- Nécessite un terminal certifié (camera, 3G, compass...)
- Gestion des autorisations avant l'installation
- Des centaines de milliers d'applications disponibles (57% gratuites)

Si vous souhaitez vendre vos applications :

- 25$ pour s'inscrire en tant que développeur
- 70% du prix revient au développeur, 30% à Google
- Revenus perçus via Google CheckOut

Télécharger/Distribuer des applications

Autres plateformes

Il existe d'autres plateformes de distributions légales (i.e. autorisées par Google) d'applications. Elles permettent notamment l'accès aux terminaux non certifiés.

- AppsLib (Archos) : http ://appslib.com
- AndroLib : http ://www.androlib.com
- Market Samsung
- ...

Plan

Termes importants

Activité (Activity)

- Brique de base d'une interface utilisateur
- Équivalent d'une fenêtre (Windows, Linux) ou d'une boîte de dialogue
- Une activité peut éventuellement ne pas avoir d'interface utilisateur (cas des services par exemple)

Fournisseur de contenu (Content provider)

- Niveau d'abstraction pour toutes données stockées sur le terminal
- Android encourage la mise à disposition de ses propres données aux autres programmes

⇒ Le *content provider* le permet en proposant un contrôle sur la façon dont on accédera aux données

Termes importants

Intention (Intent)

- Une intention est un message système qu'on peut qualifier d'événement
- Émis par le terminal pour prévenir les applications de la survenue d'événements (cas des événements systèmes) ou par tout autre application (cas des événements applicatifs).
  - Système :
    - Insertion d'une carte SD
    - Réception d'un SMS
    - ...
  - Applicatif : (on peut imaginer)
    - Un *Intent* "Le logiciel NetSpyR&T démarre"
    - Un *Intent* "L'utilisateur arrive à Paris" en utilisant les informations de géolocalisation du terminal"

Termes importants

Service (Service)

- Logiciel autonome prévu pour durer (contrairement aux activités, fournisseurs de contenus, récepteur d'intentions).
- Ne nécessite pas d'interface utilisateur.
- Exemples :
  - Service vérifiant périodiquement des mises-à-jour de flux RSS
  - Service permettant d'écouter une playlist (indépendamment de toute activité)

Manifeste (Manifeste)

- Point de départ de toute application Android
- Permet de déclarer ce que l'application contient (activités, services...)
- Précise comment ces composants sont reliés à Android (que fait-on apparaitre dans le menu ? ...)
- Précise les permissions de l'application (contrôle de la webcam, accès au réseaux, accès au service de localisation...)

Termes importants

Gadget graphique (Widget)

- Terme résultant de la contraction des termes window et gadget
- Concrètement c'est un composant d'interface graphique (libellés, champs de saisie, boutons...)

XML (XML)

- Extensible Markup Language (langage de balisage extensible)
- Langage de balisage extensible pour structurer des données

Positionnement XML (XML Layout)

- Permet de concevoir des interfaces plus simplement qu'en langage Java
- Permet concrètement d' instancier les widgets
- Ce fichier est souvent généré par des outils qui permettent de construire graphiquement les interfaces

Termes importants

Identifiant uniforme de ressource (Uniform Resource Identifier - URI)

- courte chaîne de caractères identifiant une ressource sur un réseau réel ou abstrait
- respecte une norme d'Internet mise en place pour le Web (voir RFC 3986).
- Sont des URI :
  - les Uniform Resource Locator (URL) : identifie une ressource sur un réseau et fournit les moyens d'agir sur la ressource ou d'obtenir une représentation de la ressource en décrivant son mode d'accès primaire.
    Exemple : `http://www.wikipedia.org/` identifie une ressource (page d'accueil Wikipédia) et implique qu'une représentation de cette ressource (une page HTML en caractères encodés) peut être obtenue via le protocole HTTP depuis un réseau hôte appelé www.wikipedia.org.
  - les Uniform Resource Name (URN) : identifie une ressource par son nom dans un espace de noms.
    Exemple : `urn:isbn:0-395-36341-1` identifie une ressource par un numéro de l'International Standard Book Number (ISBN), permet de faire référence à un livre, mais il ne suggère ni où, ni comment en obtenir une copie réelle.

Termes importants

Conteneur (Container)

- Permet de disposer un ensemble de widgets pour obtenir la présentation désirée
- La plupart des outils de construction d'interfaces graphique fournissent des gestionnaires de disposition (layout manager) qui sont organisés le plus fréquemment en conteneurs.

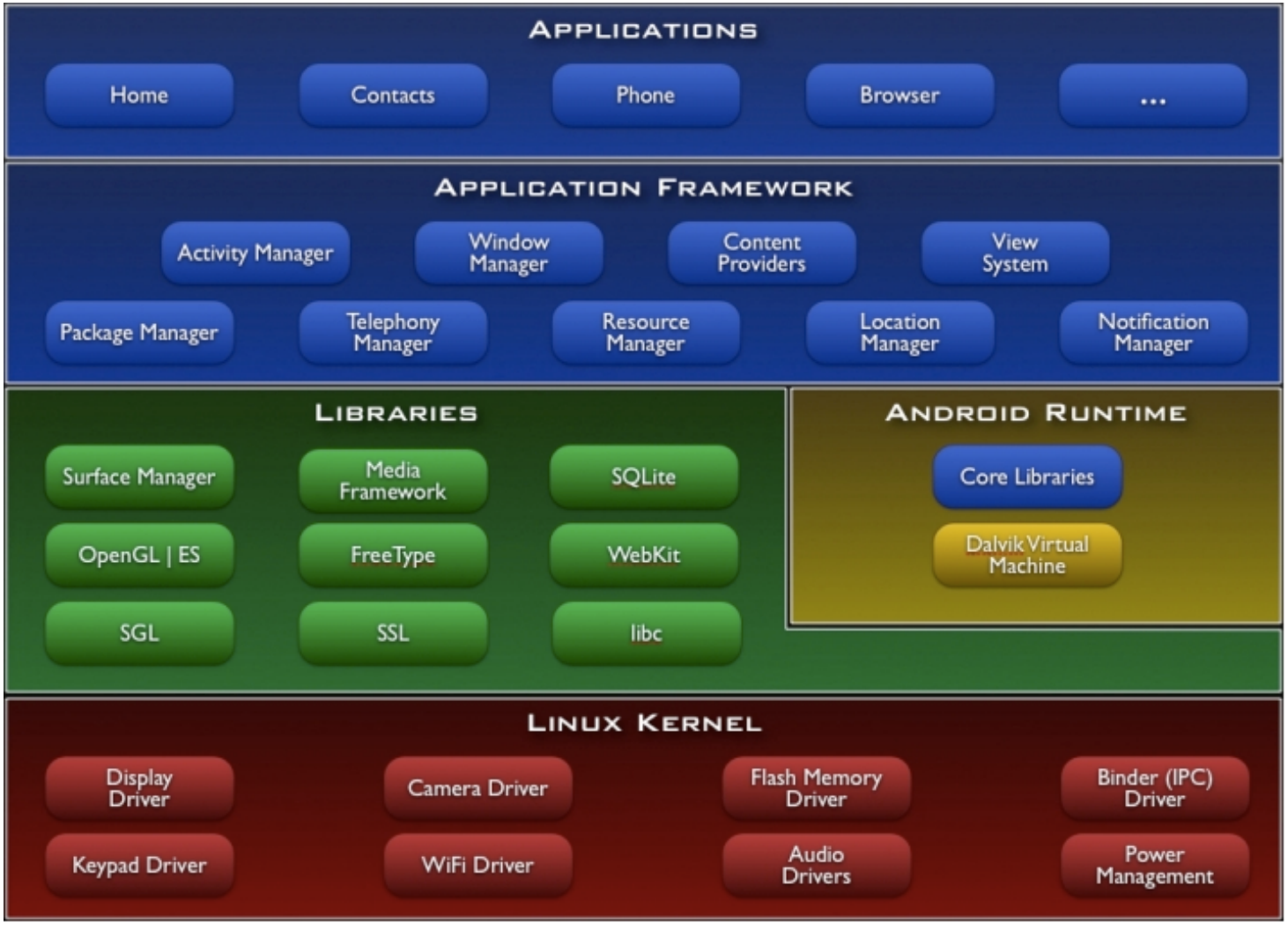
Équipement Android virtuel (Android Virtual Devices)

- Les AVD permettent de simuler l'exécution d'un terminal Android sur un ordinateur
- Ces terminaux sont personnalisables (version d'Android, type de processeur, espace de stockage...).

⇒ Simplifie le développement et la mise au point des applications

Architecture logicielle d'Android

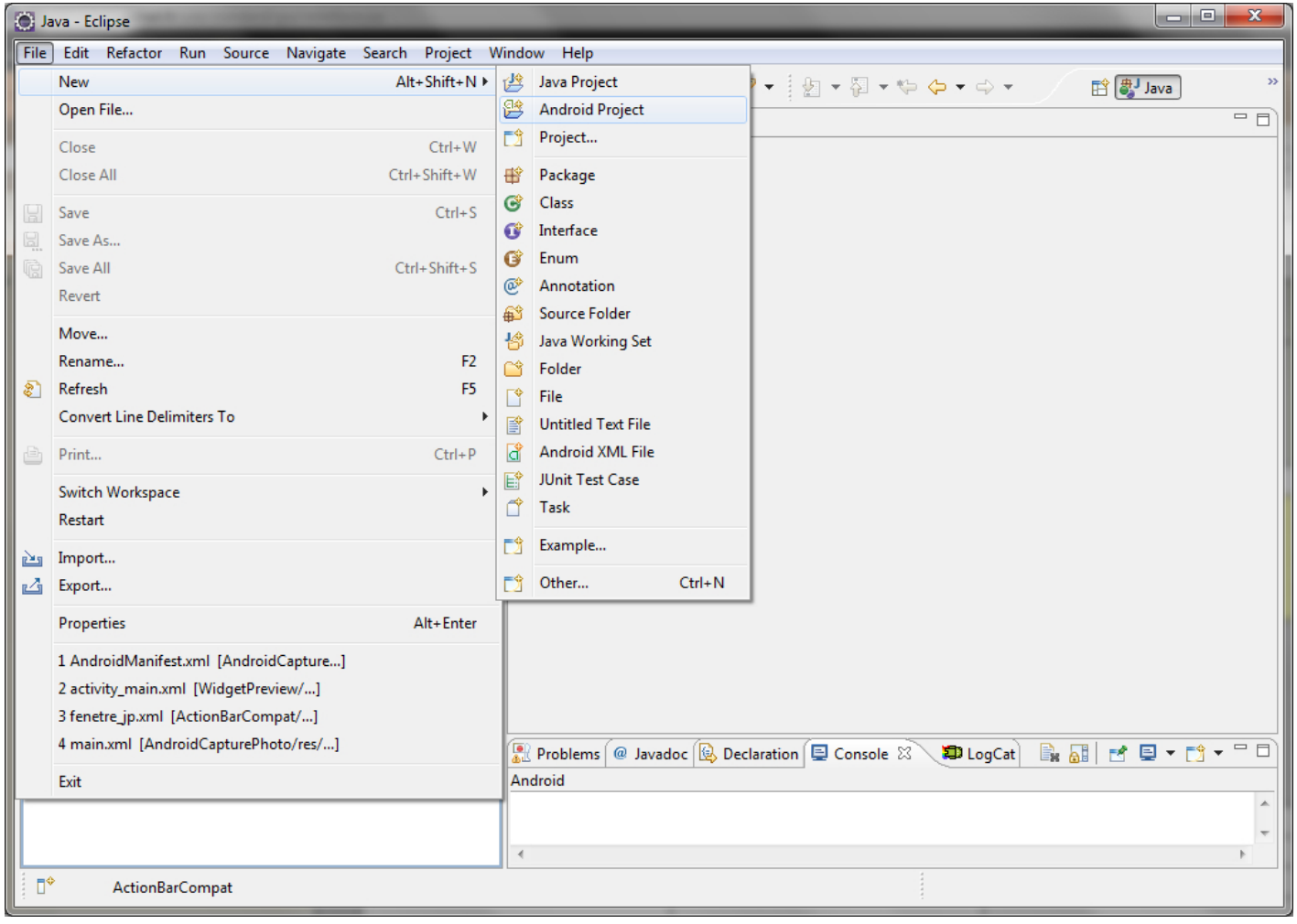
Architecture



Plan

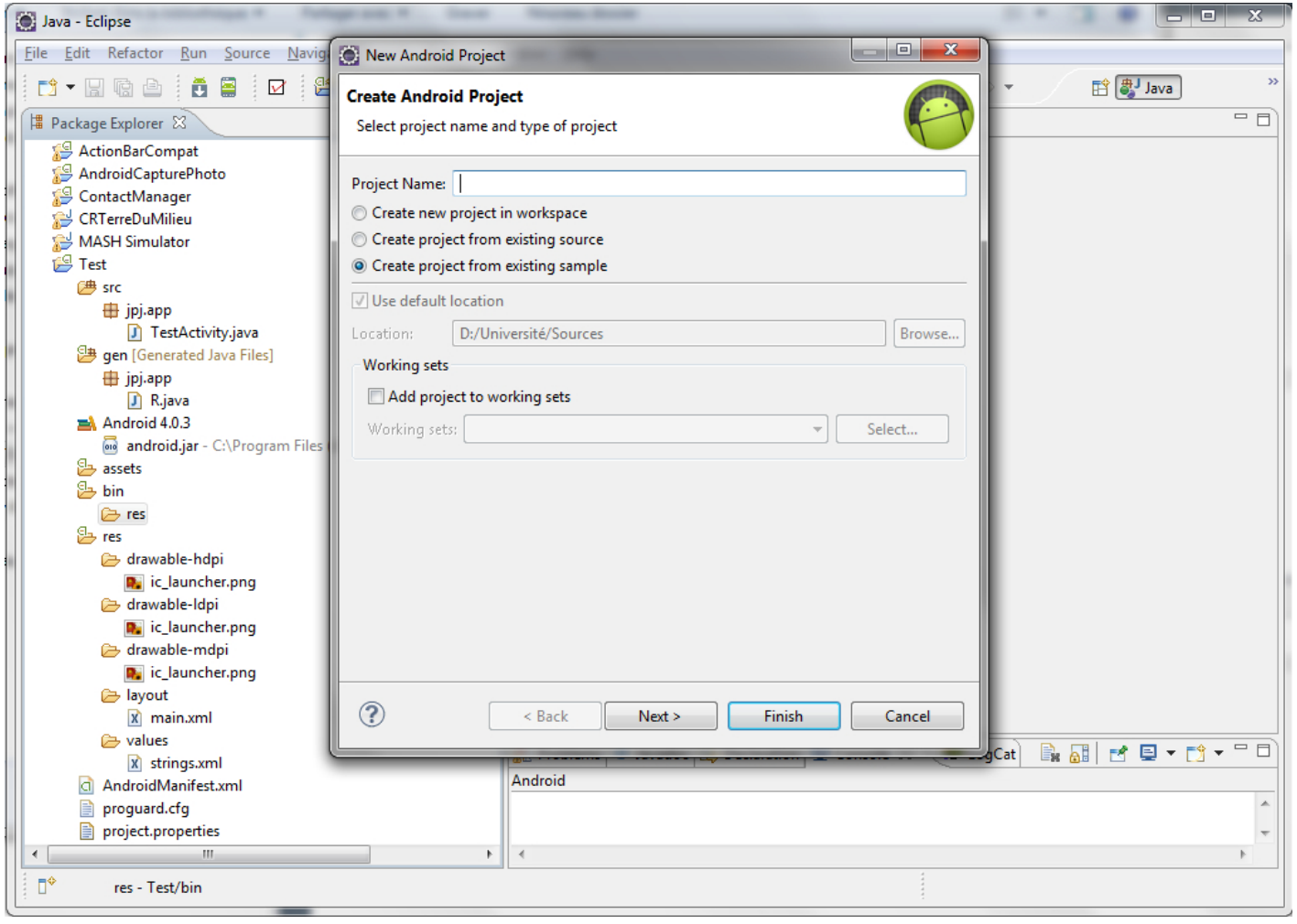
Structure d'un projet Android

Structure du projet vue sous Eclipse



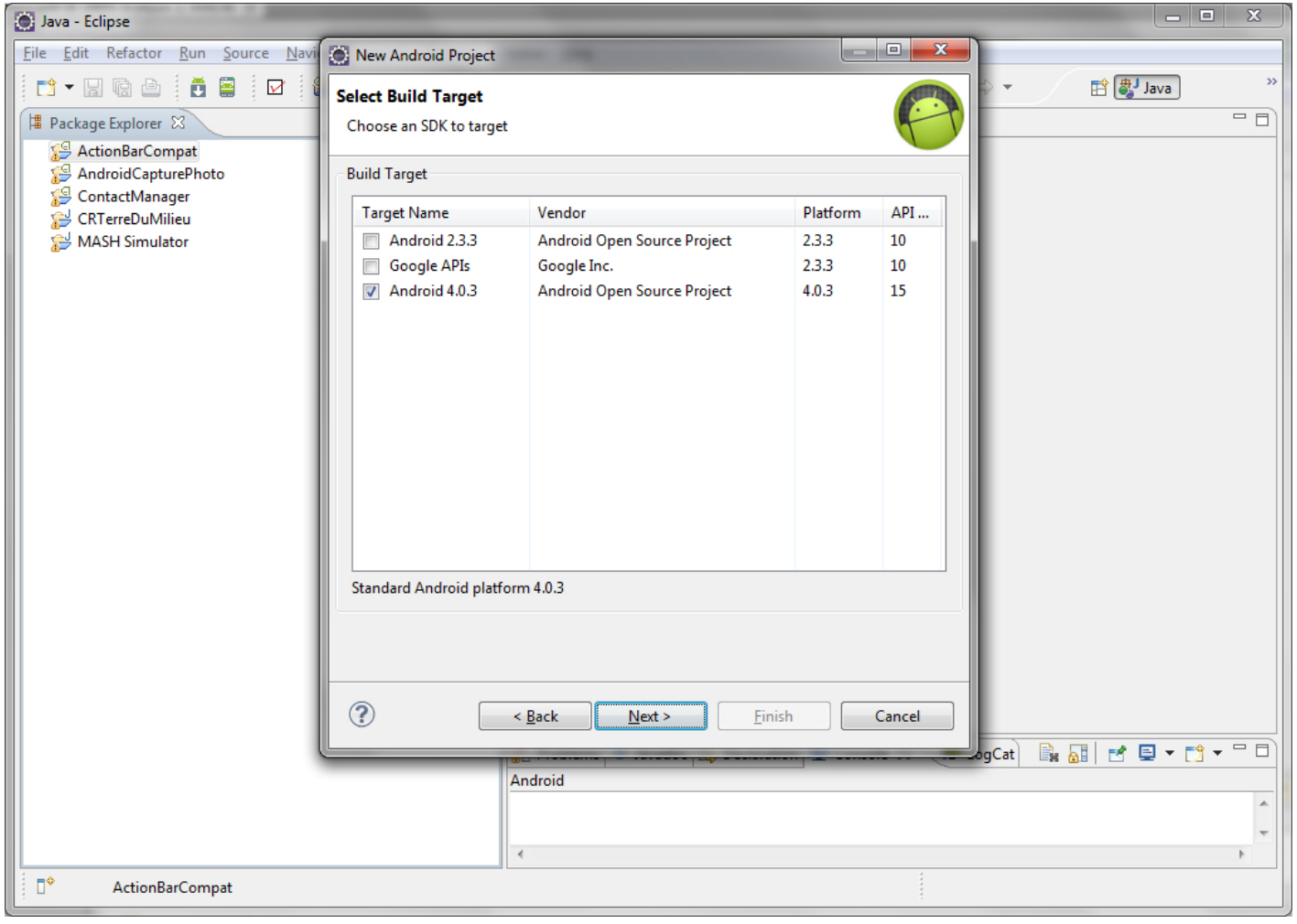
Structure d'un projet Android

Structure du projet vue sous Eclipse



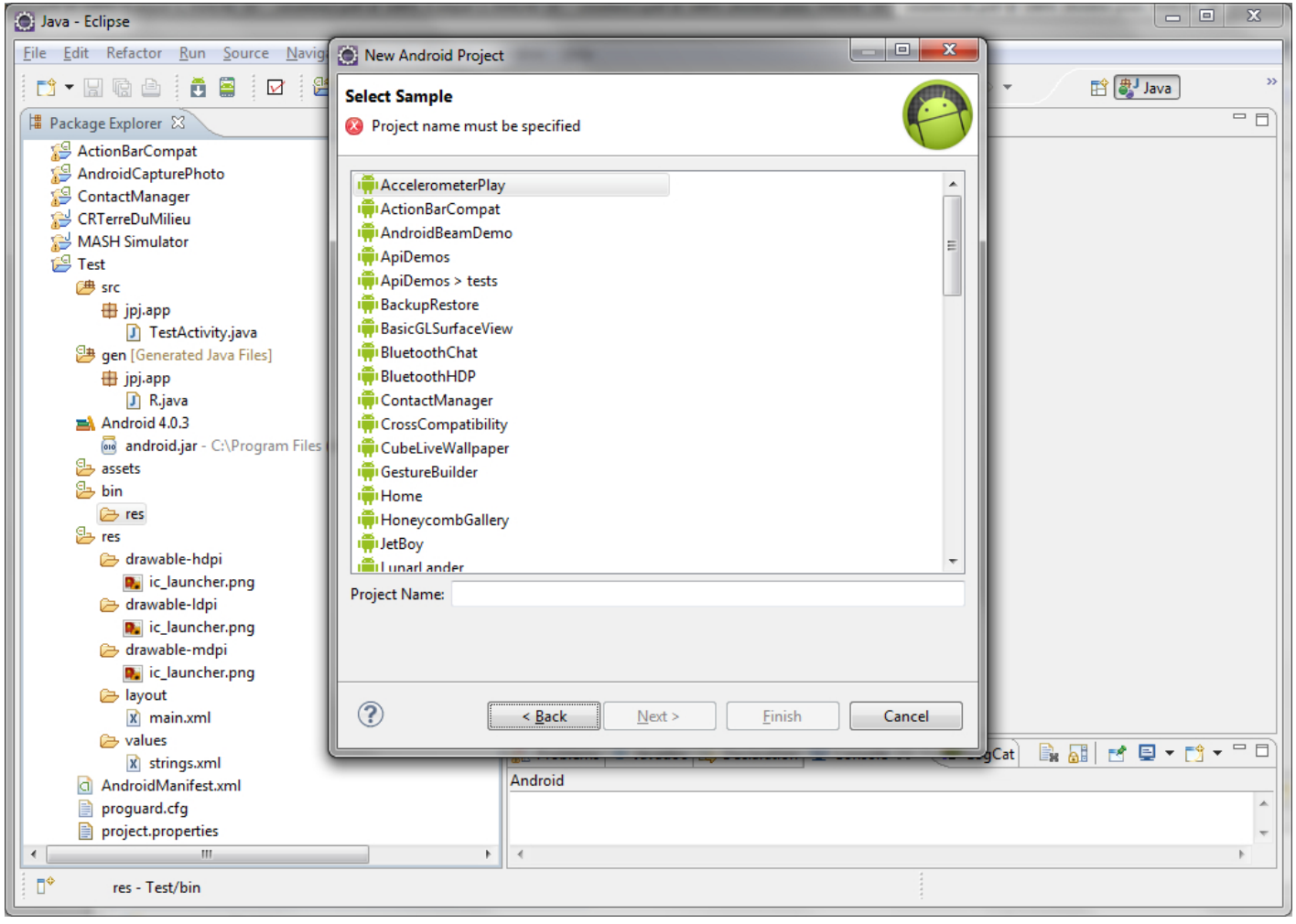
Structure d'un projet Android

Structure du projet vue sous Eclipse



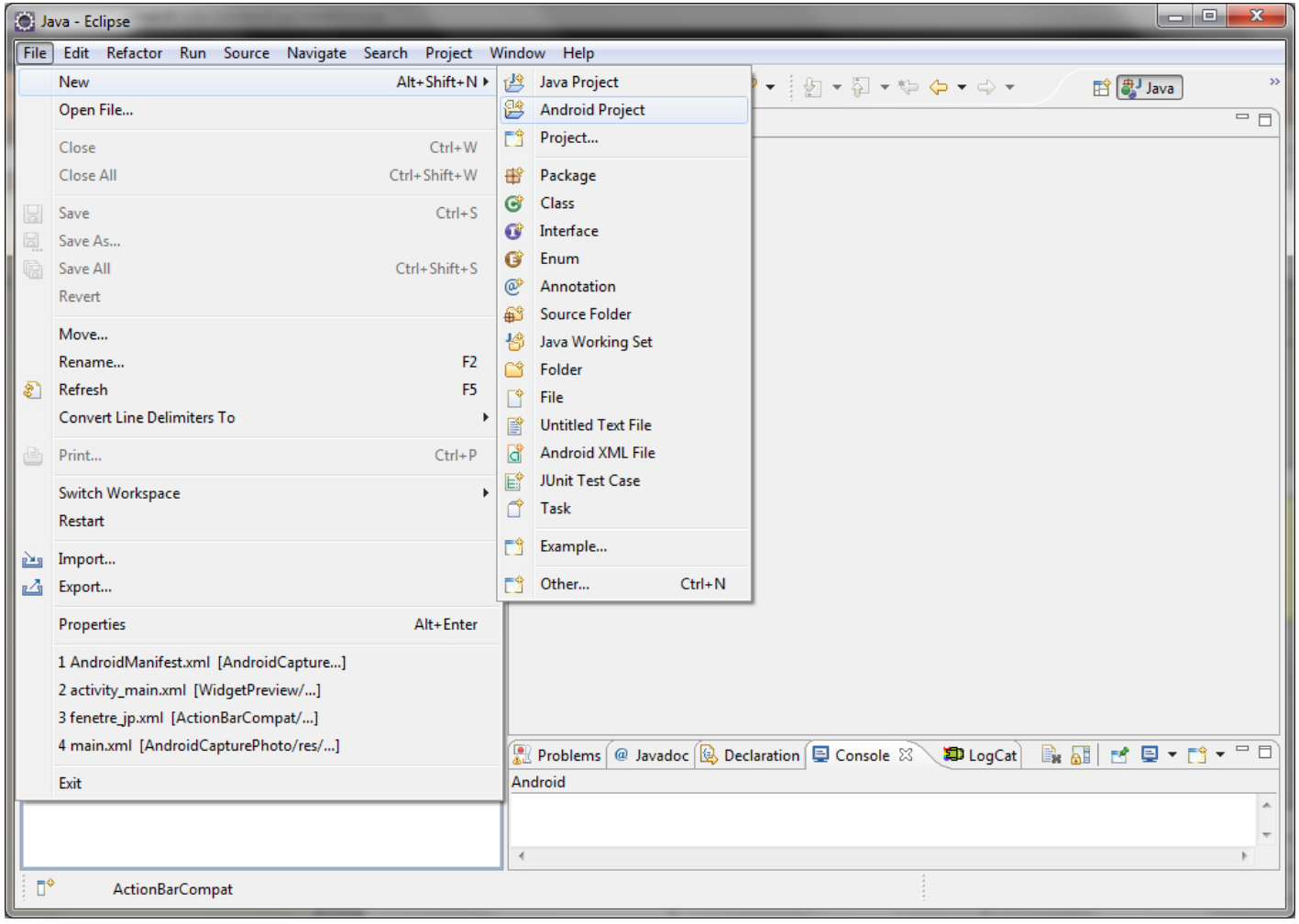
Structure d'un projet Android

Structure du projet vue sous Eclipse



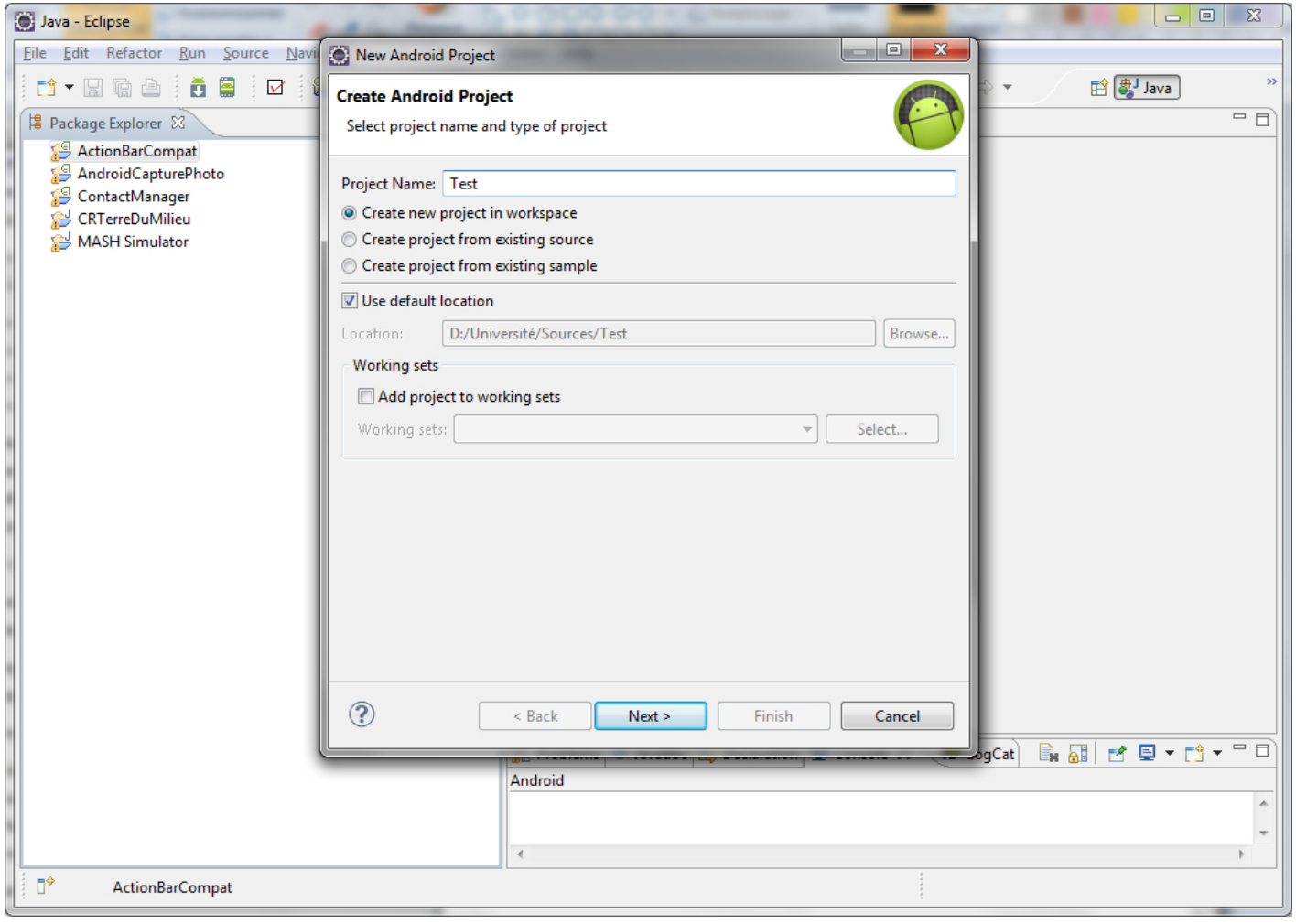
Structure d'un projet Android

Structure du projet vue sous Eclipse



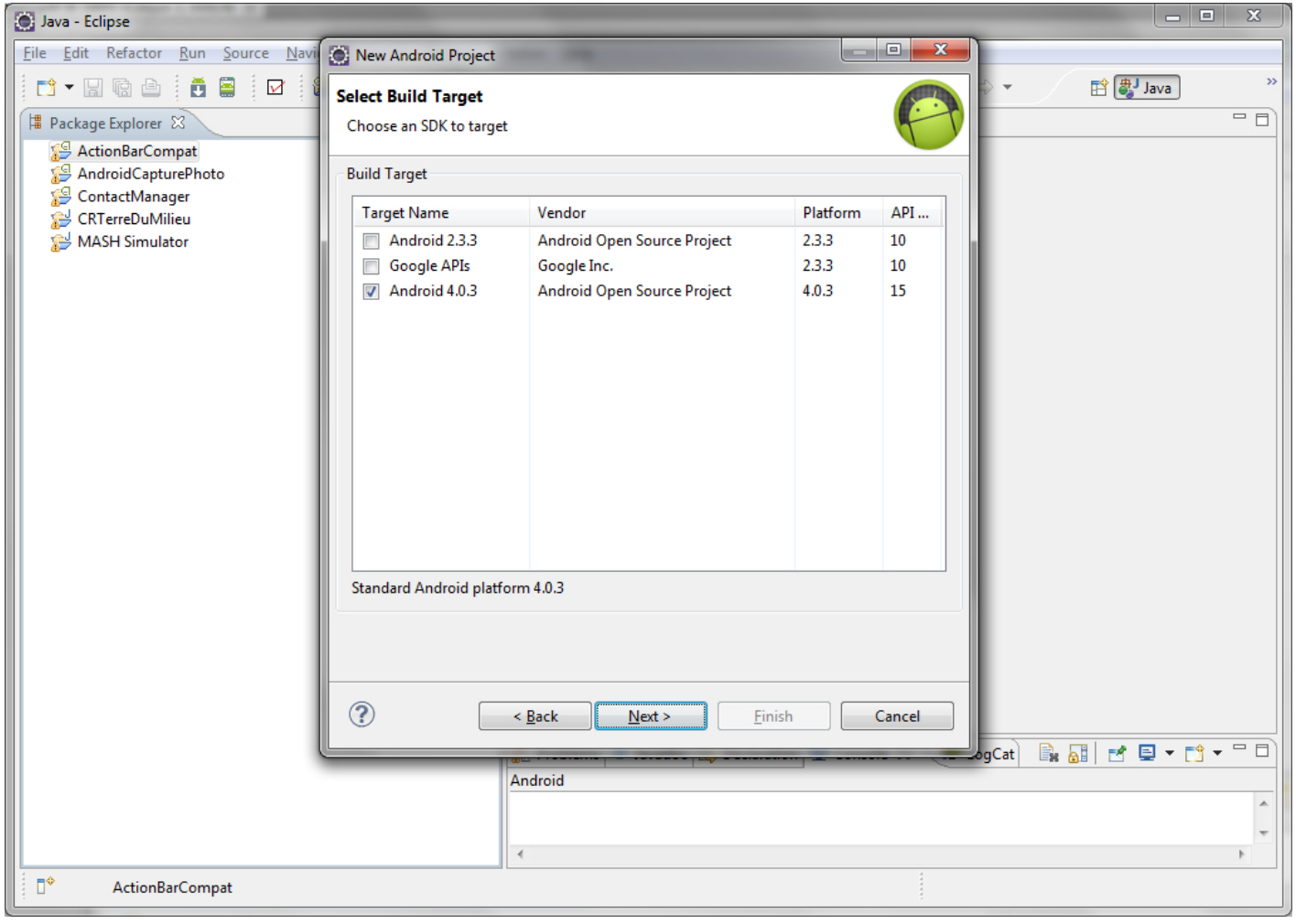
Structure d'un projet Android

Structure du projet vue sous Eclipse



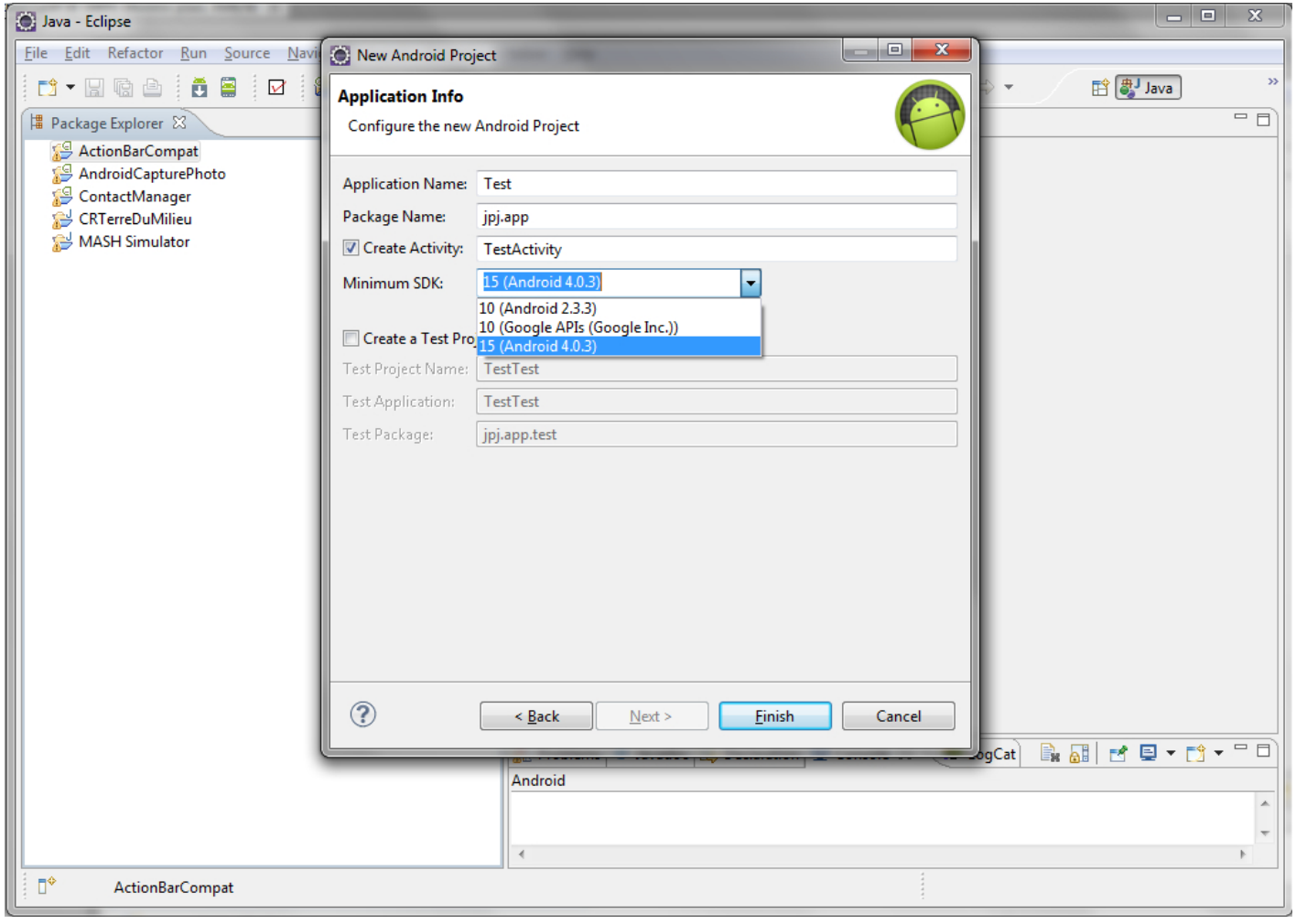
Structure d'un projet Android

Structure du projet vue sous Eclipse



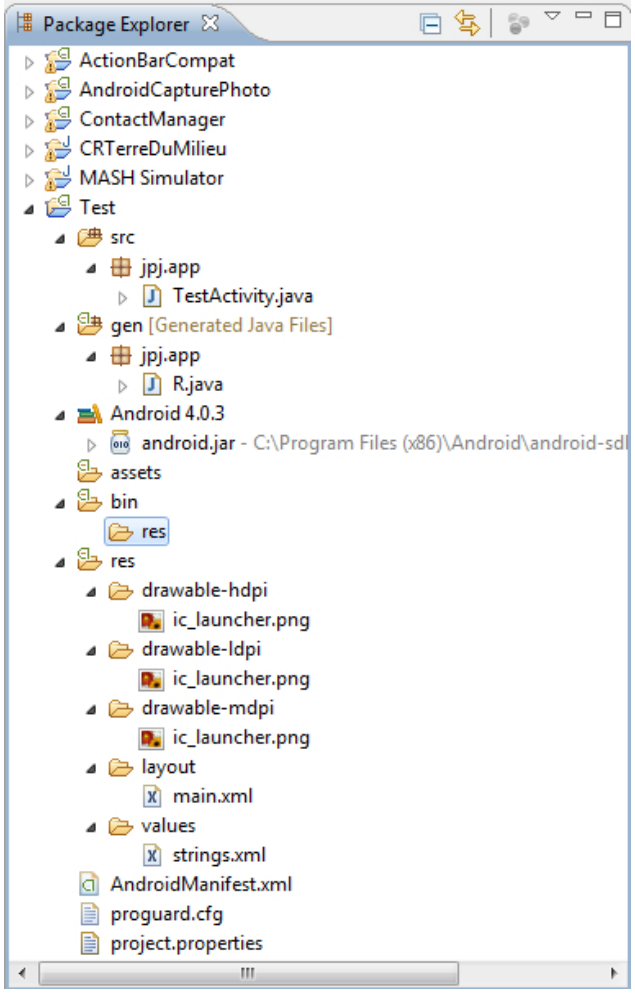
Structure d'un projet Android

Structure du projet vue sous Eclipse



Structure d'un projet Android

Structure du projet vue sous Eclipse



Structure d'un projet Android : la racine

Répertoire /

- `manifest.xml` : Le fichier manifest qui décrit l'application
- `build.xml` : Le script Ant qui permet de compiler l'application et de l'installer sur le terminal
- `default.properties` et `local.properties` : Deux fichiers de propriétés utilisés par le script Ant
- `bin/` : Répertoire qui contient l'application compilée
- `gen/` : Répertoire qui contient le code source produit par les outils de compilation Android
- `libs/` : Répertoire qui contient les fichiers JAR extérieurs à l'application
- `src/` : Répertoire qui contient code source Java de l'application
- `res/` : Répertoire qui contient les ressources (icônes, layouts...)
- `tests/` : Répertoire qui contient un projet Android complètement distinct qui permet de tester celui qui est créé
- `assets/` : Répertoire qui contient les autres fichiers statiques fournis avec l'application pour son déploiement sur le terminal

Structure d'un projet Android : les ressources

Contenu du répertoire `res/`

- `res/drawable/` : Répertoire qui contient les images (JPG, PNG...)
- `res/layout/` : Répertoire qui contient les descriptions XML de la composition de l'IHM (les layouts)
- `res/menu/` : Répertoire qui contient les descriptions XML des menus
- `res/raw/` : Répertoire qui contient les fichiers généraux (un fichier CSV contenant les informations de compte par exemple)
- `res/values/` : Répertoire qui contient les messages, les dimensions...
- `res/xml/` : Répertoire qui contient les autres fichiers XML que vous souhaitez fournir

Structure d'un projet Android : les exécutables

Contenu du répertoire `bin/`

- `bin/classes/` : Répertoire qui contient les classes java compilées
- `bin/classes.dex` : Répertoire qui contient l'exécutable créé à partir des classes compilées
- `bin/votreApp.ap_` : Répertoire qui contient les ressources de l'application (fichier zip)
- `bin/votreApp-debug.apk` : Répertoire qui contient la véritable application Android

Plan

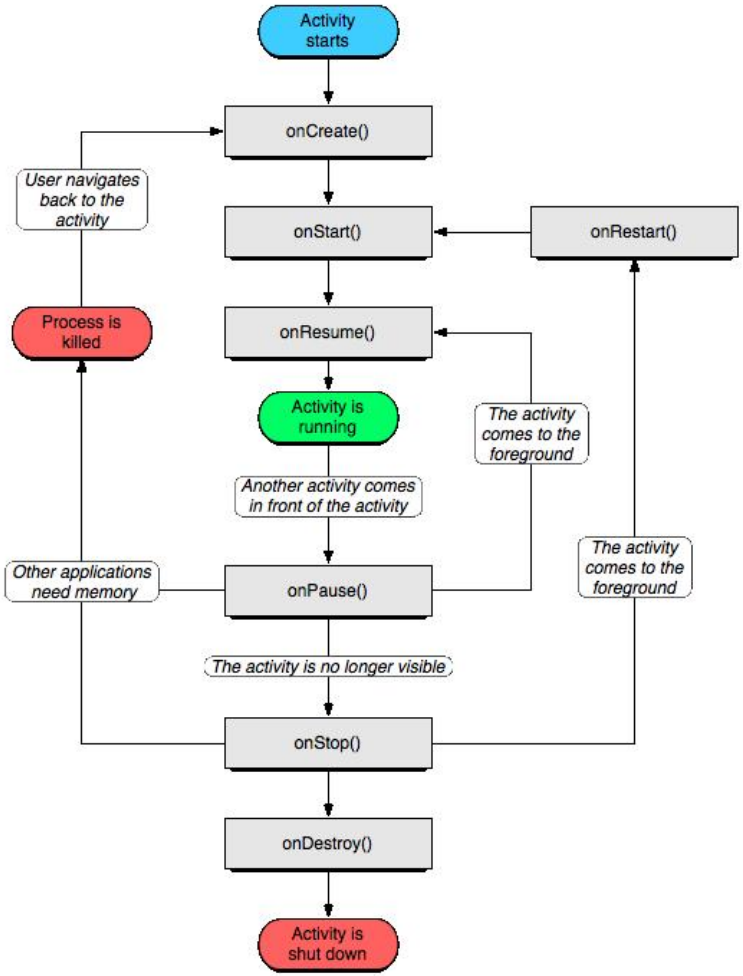
Qu'est ce qu'une activité ?

Définition

- Une activité (`Activity`) = une IHM pour un cas d'utilisation (cf cours UML) : Exemple : Soit une application d'envoi de SMS
  - Une activité pour choisir un contact dans le répertoire
  - Une activité pour la saisie du message
  - Une activité pour afficher le journal des envois
- D'un point de vue opérationnel :
  - Une activité doit hériter de la classe `android.app.Activity`
  - Une activité est indépendante des autres activités MAIS :
    - Il faut désigner une activité de départ (celle qui sera utilisée en 1er)
    - Il faut chainer les activités (une activité doit activer la suivante)
  - Nous considérons dans notre cours qu'une activité est liée à un `layout XML`.

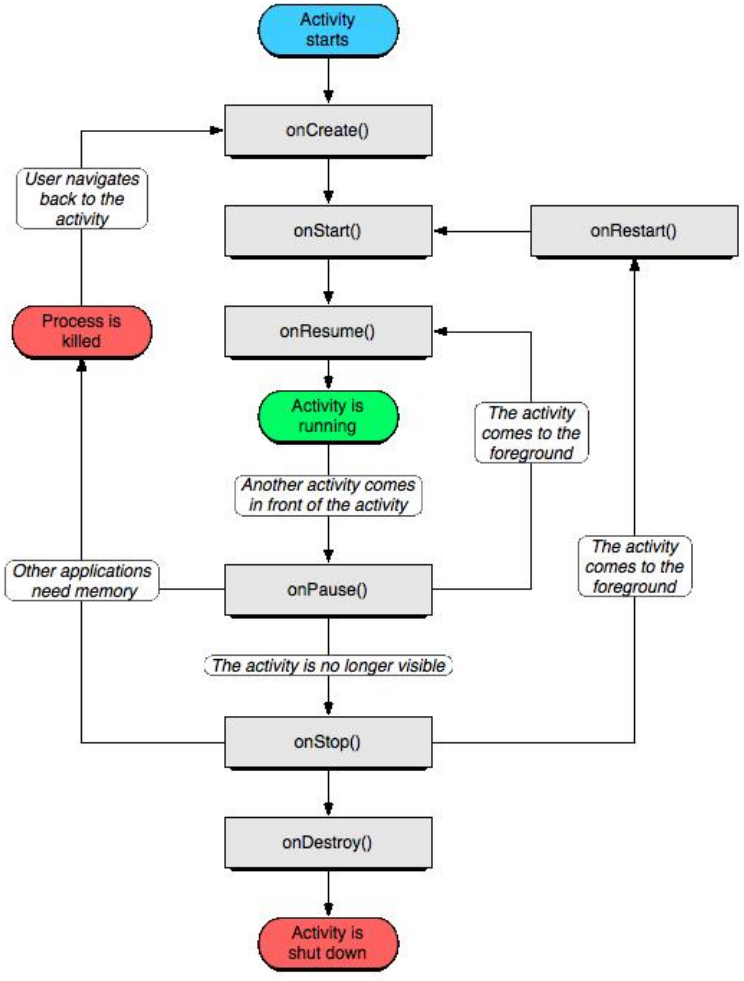
Cycle de vie d'une activité

Le cycle de vie



Cycle de vie d'une activité

Le cycle de vie

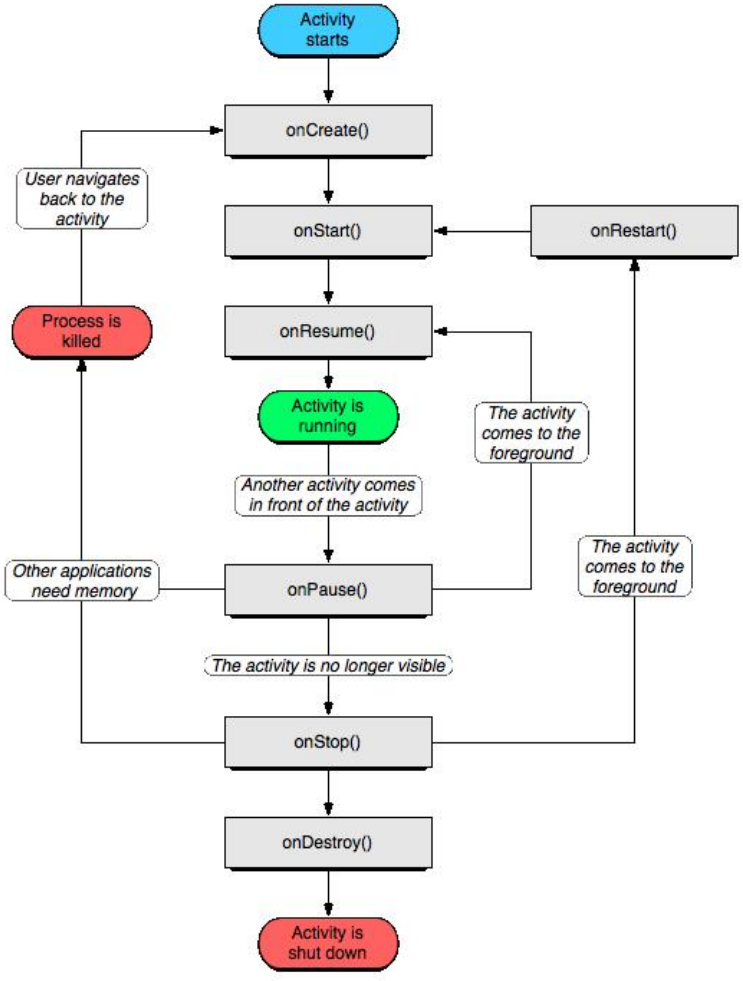


`onCreate()` :

- Méthode exécutée quand l'activité est créée
- Si c'est la l'activité est la première d'une application, cette méthode est donc appelée quand l'utilisateur exécute l'application
- Méthode utilisée pour initialiser :
  - la vue XML
  - si nécessaire, les fichiers/données temporaires

Cycle de vie d'une activité

Le cycle de vie

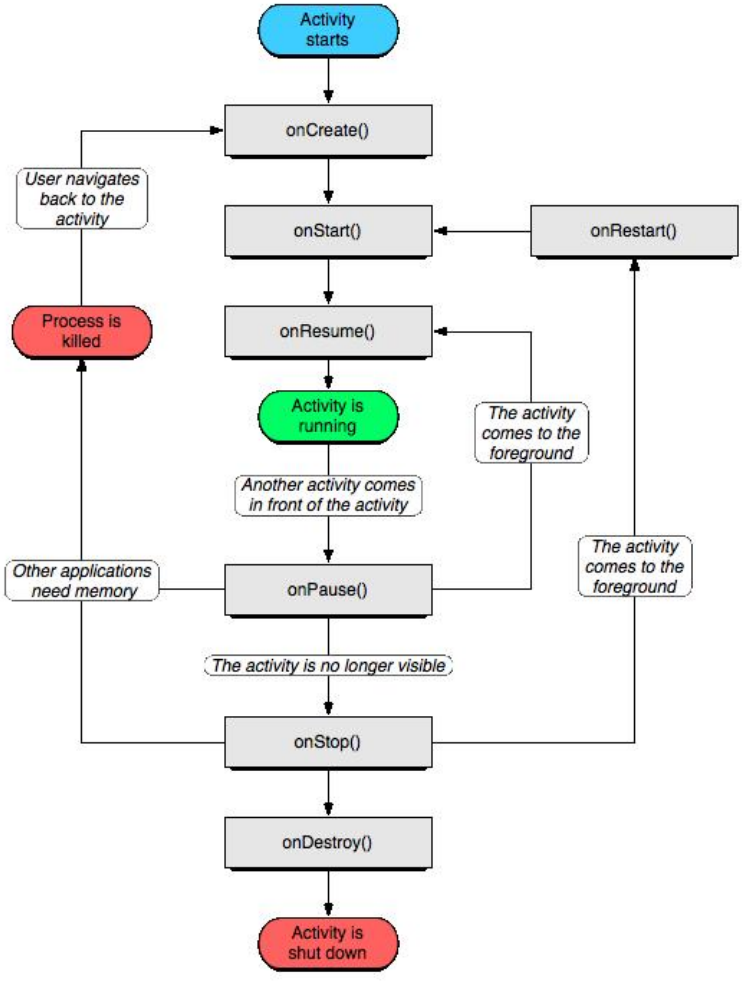


`onRestart()` :

- Méthode exécutée lorsque on redémarre l'activité après un arrêt (provoqué par un appel de la méthode `stop()`)
- Cette méthode est donc appelée quand l'application repasse en premier plan après un arrêt prolongé

Cycle de vie d'une activité

Le cycle de vie

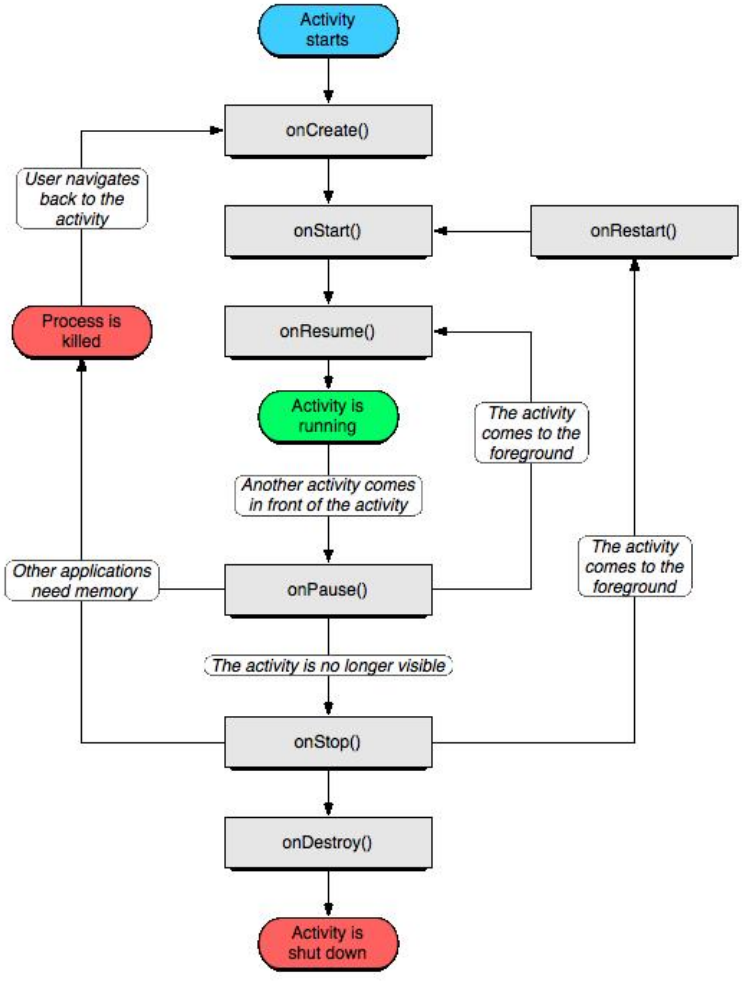


`onStart()` :

- Méthode exécutée après chaque `onCreate()` ou `onRestart()`
- Si nécessaire, recharger les données sauvegardées durant le dernier arrêt

Cycle de vie d'une activité

Le cycle de vie

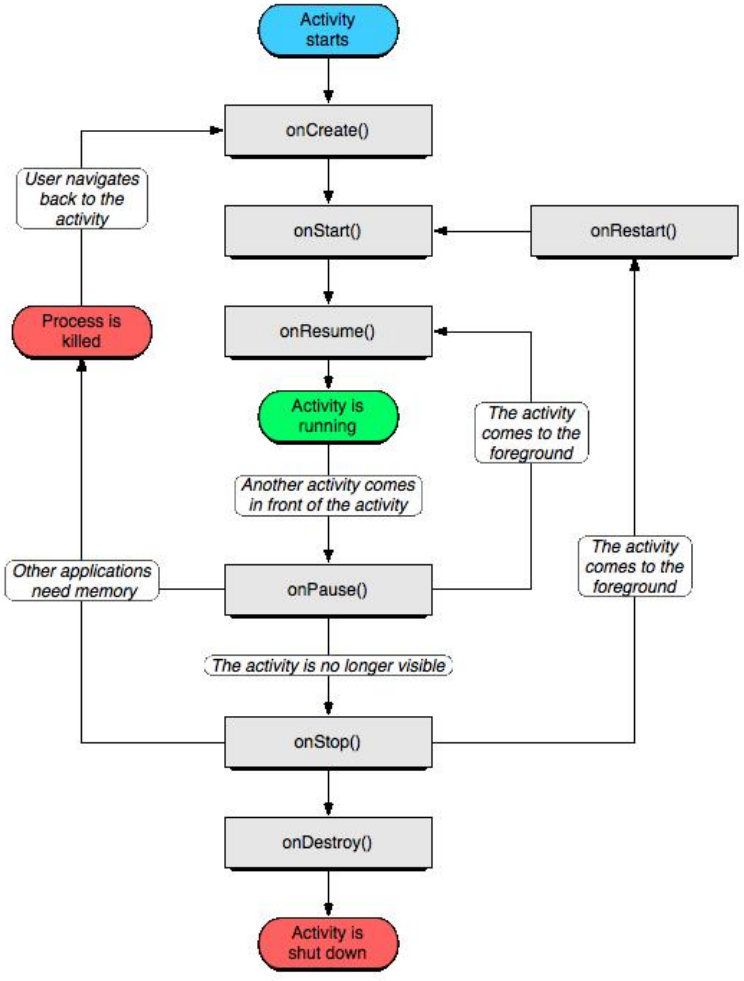


`onResume()` :

- Méthode exécutée après chaque appel de la méthode `onStart()`
- Méthode exécutée a chaque passage en premier plan de l'activité (si pas de `stop()`)
- Si nécessaire :
  - gérer la connexion à la base de données
  - mise à jour des données qui auraient pu être modifiées entre temps (avant le `onResume()`)

Cycle de vie d'une activité

Le cycle de vie

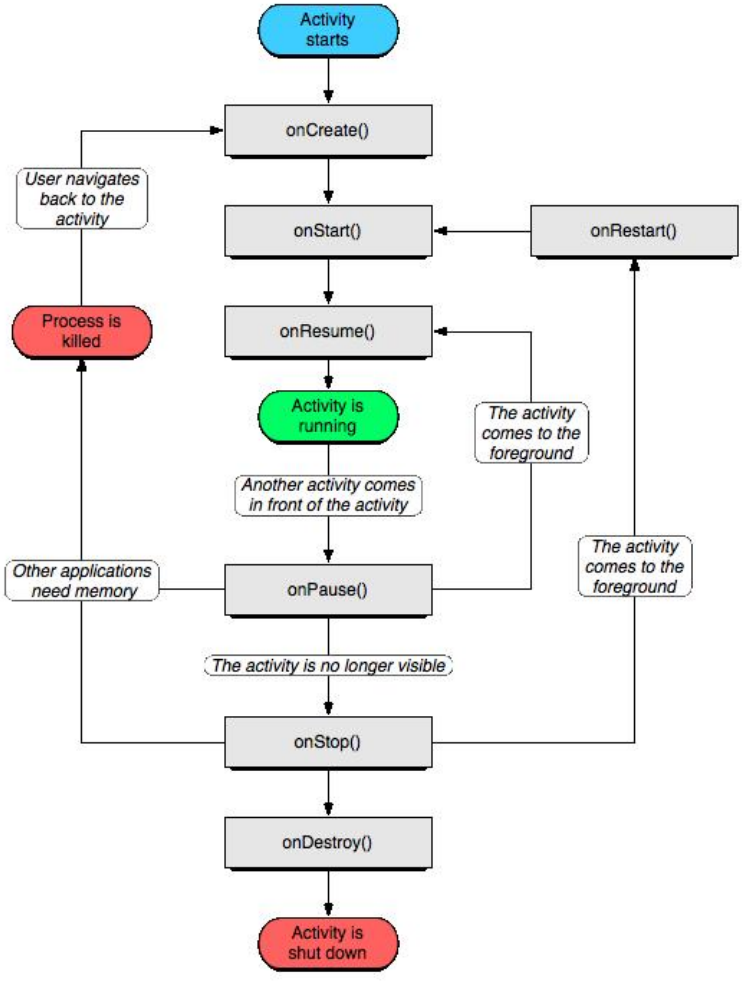


`onPause()` :

- Méthode exécutée à chaque fois que :
  - l'utilisateur passe à une autre activité,
  - l'utilisateur demande un `finish()` sur cette activité
  - le système a besoin de libérer de la mémoire
- Remarquer que la méthode est exécutée systématiquement avant chaque `onStop()`
- Si nécessaire, il faut :
  - sauvegarder les données qui seront perdues après l'arrêt si elles ne sont pas sauvegardées
  - gérer la déconnexion à la base de données
- Attention : l'exécution de cette fonction doit être rapide car la prochaine activité ne démarrera pas tant que l'exécution de cette fonction n'est pas terminée

Cycle de vie d'une activité

Le cycle de vie

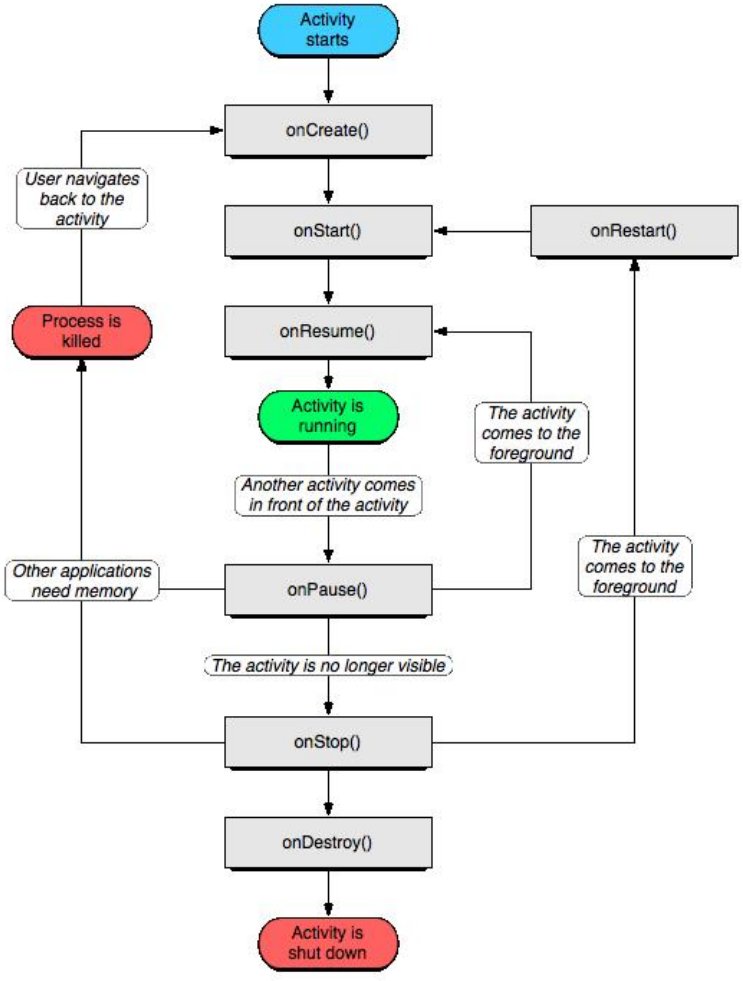


`onStop()` :

- Méthode exécutée avant chaque mise en sommeil
- Méthode exécutée avant chaque `onDestroy()`
- Libération des ressources

Cycle de vie d'une activité

Le cycle de vie



`onDestroy()` :

- Méthode exécutée lors du kill/arrêt de l'activité
- `onCreate()` devra à nouveau être exécuté pour obtenir à nouveau l'activité
- Si nécessaire, libérer les fichiers temporaires

Une illustration

Hello world !!!

```
1  package com.developpez.android;
2  import android.app.Activity;
3  import android.os.Bundle;
4  import android.widget.TextView;
5  public class HelloWorld extends Activity
6  {
7          /** Called with the activity is first created. */
8          @Override
9          public void onCreate(Bundle icicle)
10         {
11                 super.onCreate(icicle);
12                 /** remarquez qu'ici on utilise pas de layout XML */
13                 TextView textView = new TextView(this);
14                 textView.setText("Hello world !");
15                 setContentView(textView);
16         }
17
18          protected void onRestart()
19         {
20                 super.onRestart();
21                 Toast.makeText(this, "appel de la méthode onRestart()", 1).show();
22         }
23
24         protected void onPause()
25         {
26                 super.onPause();
27                 Toast.makeText(this, "appel de la méthode onRestart()", 1).show();
28         }
29
30         ...
31
32 }
```

Plan

XML, format textuel, structuré, et extensible

Définition

- eXtensible Markup Language
- Langage de balisage extensible pour structurer des données
- Deux versions : 1.0 et 1.1

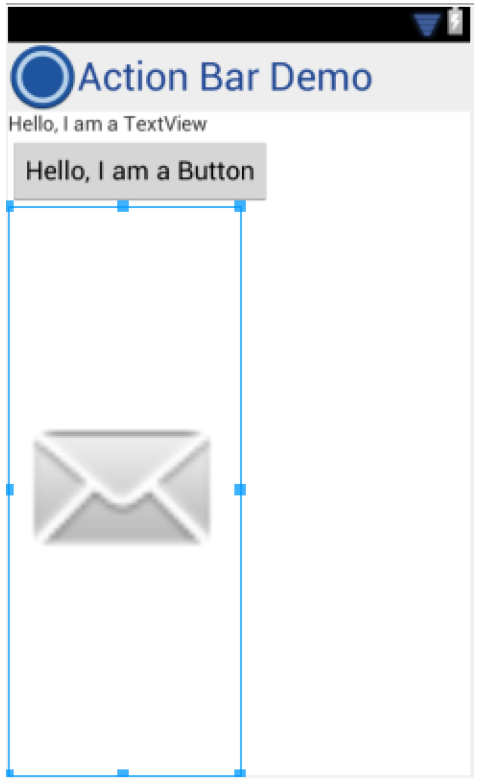
Exemple simple d'utilisation de XML

```
1  <?xml version="1.0" encoding="ISO-8859-1"?>            <!-- En-tête du fichier XML (i.e. prologue) -->
2  <BIBLIOTHEQUE>                                        <!-- Balise d'ouverture de BIBLIOTHEQUE -->
3
4       <MAGAZINE>                                       <!-- On déclare un MAGAZINE -->
5          <TITRE>Science et Vie</TITRE>
6          <DATEPARUTION>01-02-2012</DATEPARUTION>
7          <PRIX devise="Euro">4.20</PRIX>               <!-- "PRIX" est une balise, "devise" une clé, "32.30" une valeur -->
8       </MAGAZINE>
9
10      <LIVRE type="education" >                        <!-- On déclare un LIVRE de type "education" -->
11         <TITRE>L'art du développement Android</TITRE>
12         <AUTEUR>Mark Murphy</AUTEUR>
13         <PRIX devise="Euro">32.30 </PRIX>
14      </LIVRE>
15
16      <LIVRE type="roman">                             <!-- On déclare un LIVRE de type "roman" -->
17         <TITRE>Le livre d'Android</TITRE>
18         <AUTEUR>Patrick Beuzit</AUTEUR>
19         <PRIX devise="Dollar">22.25</PRIX>
20      </LIVRE>
21
22      ...
23
24 </BIBLIOTHEQUE>                                       <!-- Balise de fermeture de BIBLIOTHEQUE -->
```

XML, format textuel, structuré, et extensible

Exemple de layout XML dans Android

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="fill_parent"
4      android:layout_height="fill_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/text"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:text="Hello, I am a TextView" />
12
13     <Button
14         android:id="@+id/button"
15         android:layout_width="wrap_content"
16         android:layout_height="wrap_content"
17         android:text="Hello, I am a Button" />
18
19     <ImageView
20         android:id="@+id/imageView1"
21         android:layout_width="161dp"
22         android:layout_height="wrap_content"
23         android:layout_weight="0.36"
24         android:src="@android:drawable/sym_action_email" />
25
26 </LinearLayout>
```



Prologue (ligne 1)

(L1) `<?xml ... ?>` : Entête XML

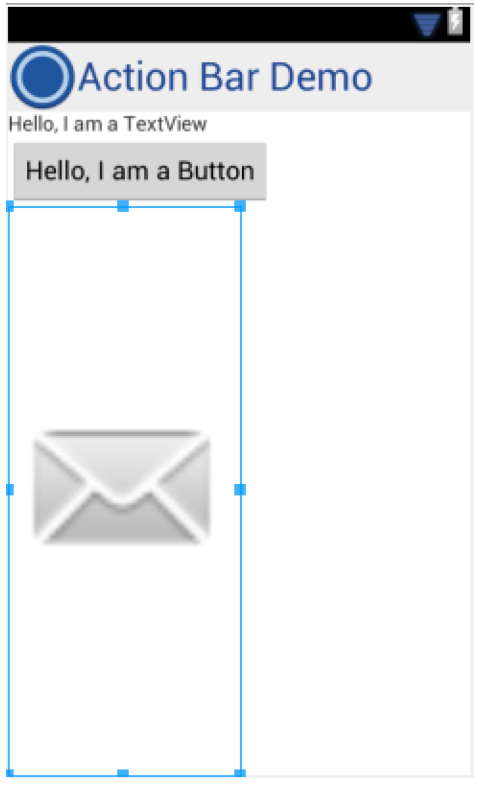
(L1) `version="1.0"` : XML version 1.0

(L1) `encoding="utf-8"` : format de codage de caractères ISO/CEI 10646 (caractères codés sur 8bits)

XML, format textuel, structuré, et extensible

Exemple de layout XML dans Android

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="fill_parent"
4      android:layout_height="fill_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/text"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:text="Hello, I am a TextView" />
12
13     <Button
14         android:id="@+id/button"
15         android:layout_width="wrap_content"
16         android:layout_height="wrap_content"
17         android:text="Hello, I am a Button" />
18
19     <ImageView
20         android:id="@+id/imageView1"
21         android:layout_width="161dp"
22         android:layout_height="wrap_content"
23         android:layout_weight="0.36"
24         android:src="@android:drawable/sym_action_email" />
25
26 </LinearLayout>
```



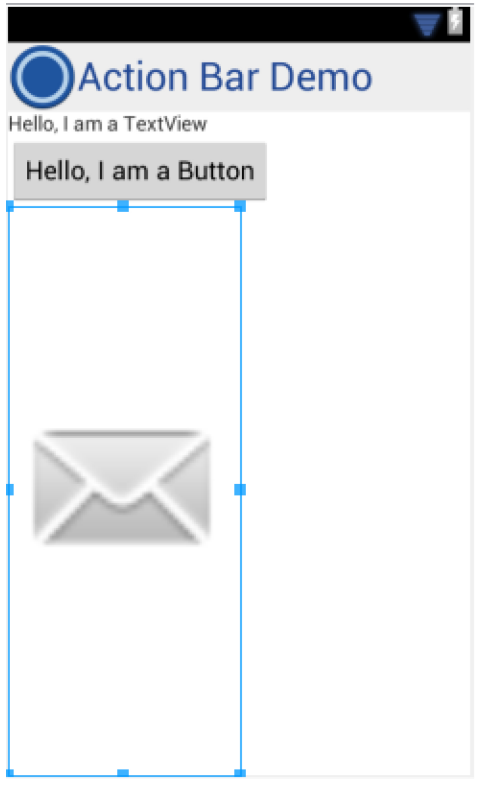
LinearLayout (lignes 2 à 5)

(L2 et 5) `<LinearLayout ... />` : Spécifie le conteneur (ici : disposition des widgets les un derrière les autres)
(L2) `xmlns:android="..."` : Déclaration de l'espace de noms XML d'Android
(L3) `android:layout_width="..."` : la largeur du layout "remplit" le parent (ici l'écran)
(L4) `android:layout_height="..."` : la hauteur du layout "remplit" le parent (ici l'écran)
(L5) `android:orientation="..."` : les widgets se suivront dans une direction ici verticale

XML, format textuel, structuré, et extensible

Exemple de layout XML dans Android

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="fill_parent"
4      android:layout_height="fill_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/text"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:text="Hello, I am a TextView" />
12
13     <Button
14         android:id="@+id/button"
15         android:layout_width="wrap_content"
16         android:layout_height="wrap_content"
17         android:text="Hello, I am a Button" />
18
19     <ImageView
20         android:id="@+id/imageView1"
21         android:layout_width="161dp"
22         android:layout_height="wrap_content"
23         android:layout_weight="0.36"
24         android:src="@android:drawable/sym_action_email" />
25
26 </LinearLayout>
```



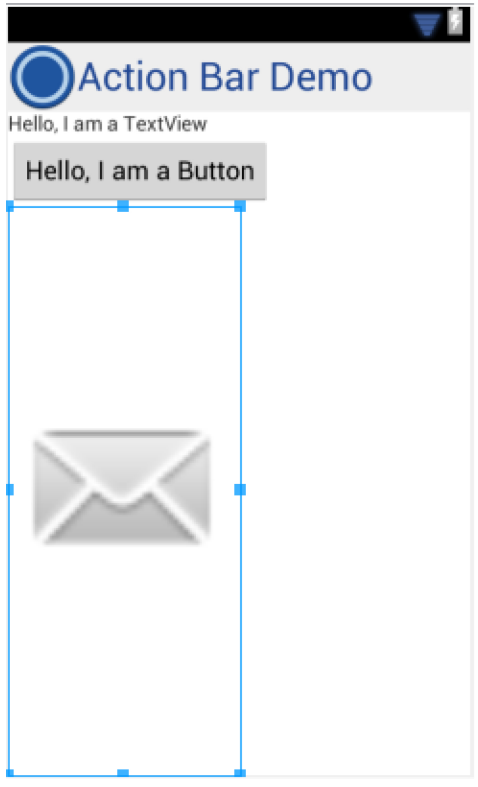
TextView (lignes 7 à 11)

(L7 et L11) `<TextView ... />` : le widget défini ici est un label (un texte)
(L8) `android:id="@+id/text"` : identifiant associé à ce widget (cf. code java)
(L9) `android:layout_width="wrap_content"` : largeur du widget adapté au texte qu'il contient
(L10) `android:layout_height="wrap_content"` : hauteur du widget adapté au texte qu'il contient
(L11) `android:text="Hello, I am a TextView"` : texte affiché par le widget

XML, format textuel, structuré, et extensible

Exemple de layout XML dans Android

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="fill_parent"
4      android:layout_height="fill_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/text"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:text="Hello, I am a TextView" />
12
13     <Button
14         android:id="@+id/button"
15         android:layout_width="wrap_content"
16         android:layout_height="wrap_content"
17         android:text="Hello, I am a Button" />
18
19     <ImageView
20         android:id="@+id/imageView1"
21         android:layout_width="161dp"
22         android:layout_height="wrap_content"
23         android:layout_weight="0.36"
24         android:src="@android:drawable/sym_action_email" />
25
26 </LinearLayout>
```



Button (lignes 13 à 17)
(L13 et L17) <Button . . . / > : le widget défini ici est un label (un texte)
android:id="@+id/**button**" : identifiant associé à ce widget

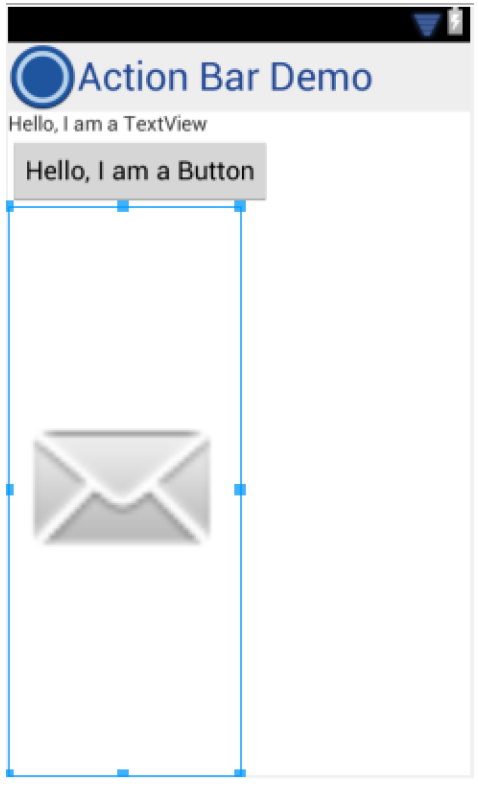
En java, on va créer une instance de la classe Button pour manipuler ce widget :

```
Button myButton = (Button) findViewById(R.id.button);
```

XML, format textuel, structuré, et extensible

Exemple de layout XML dans Android

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="fill_parent"
4      android:layout_height="fill_parent"
5      android:orientation="vertical" >
6
7      <TextView
8          android:id="@+id/text"
9          android:layout_width="wrap_content"
10         android:layout_height="wrap_content"
11         android:text="Hello, I am a TextView" />
12
13     <Button
14         android:id="@+id/button"
15         android:layout_width="wrap_content"
16         android:layout_height="wrap_content"
17         android:text="Hello, I am a Button" />
18
19     <ImageView
20         android:id="@+id/imageView1"
21         android:layout_width="161dp"
22         android:layout_height="wrap_content"
23         android:layout_weight="0.36"
24         android:src="@android:drawable/sym_action_email" />
25
26 </LinearLayout>
```



ImageView (lignes 19 à 24)
(L19 et L24) `<ImageView ... />` : (L20) `android:id="@+id/imageView1"` : identifiant de l'image
(L21) `android:layout_width="161dp"` : la largeur de l'image est de 161 *dot point*
(L22) `android:layout_height="wrap_content"` : prend la hauteur adaptée à l'image
(L23) `android:layout_weight="0.36"` : voir la propriété `android:layout_gravity`
(L24) `android:src="@android:drawable/sym_action_email"` : URI de l'image

Intégrer des layouts XML dans mon application

Les étapes :

Avec un éditeur texte ou un outil le générant :

1. Créer un layout XML (porter une attention particulière aux identifiants des widgets)

Dans le programme Java :

2. Charger le layout
3. Instancier les widgets sur lesquels on veut agir

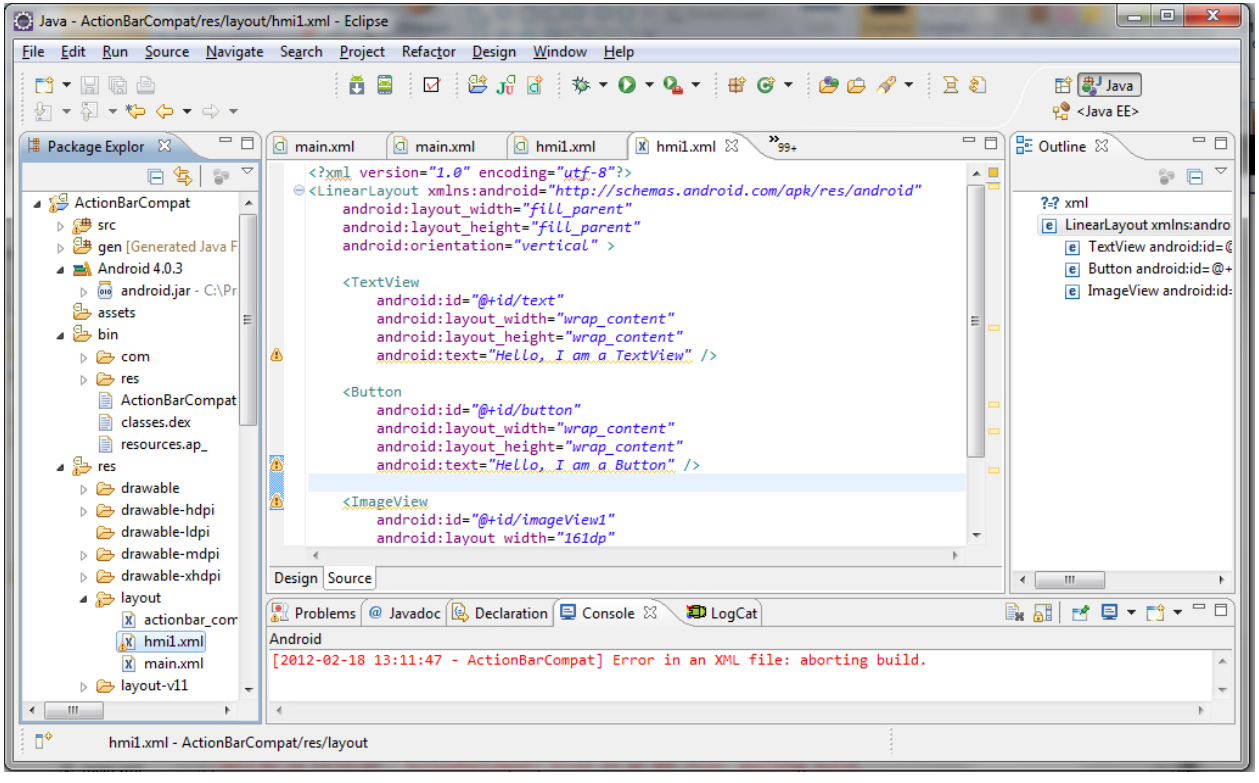
1. Créer un layout XML

Création manuelle

Documentation disponible sur
http ://developer.android.com/guide/topics/ui/declaring-layout.html

Éditer le layout avec un éditeur XML.

Exemple sous eclipse :



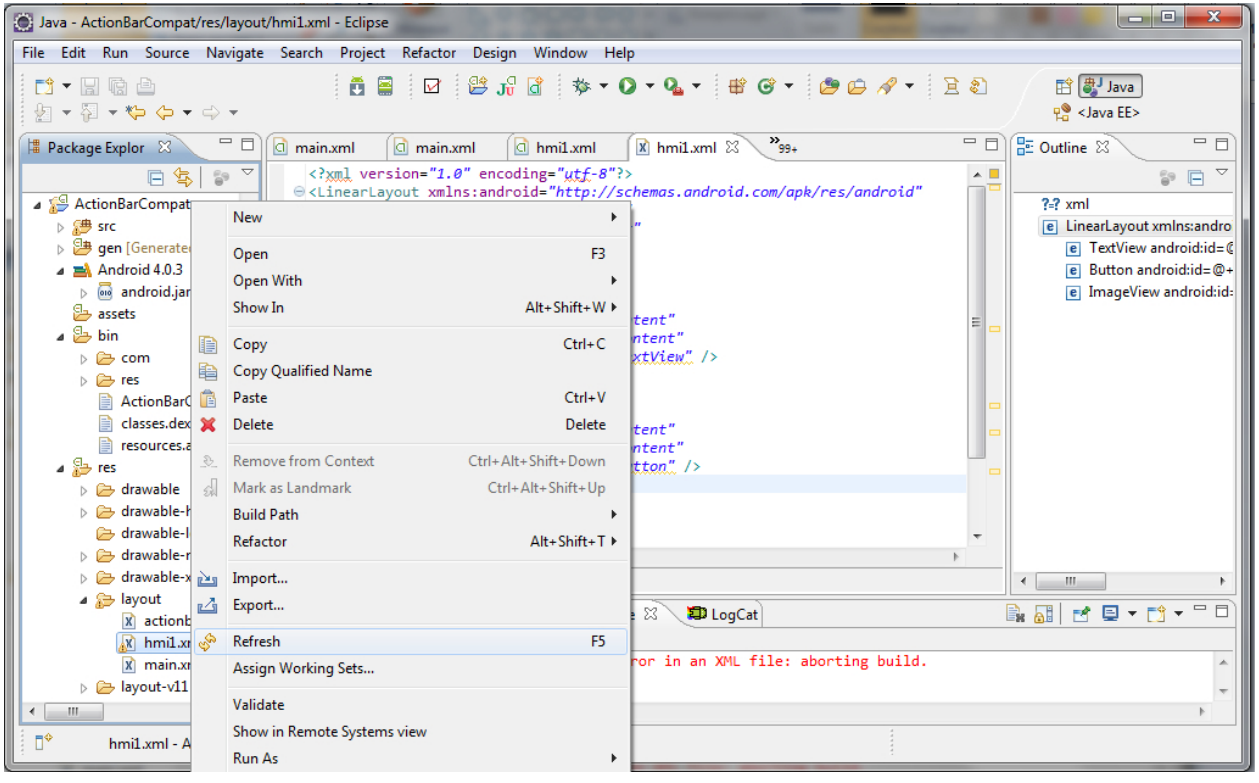
1. Créer un layout XML

Création manuelle

Documentation disponible sur
http ://developer.android.com/guide/topics/ui/declaring-layout.html

Éditer le layout avec un éditeur XML.

Exemple sous eclipse :



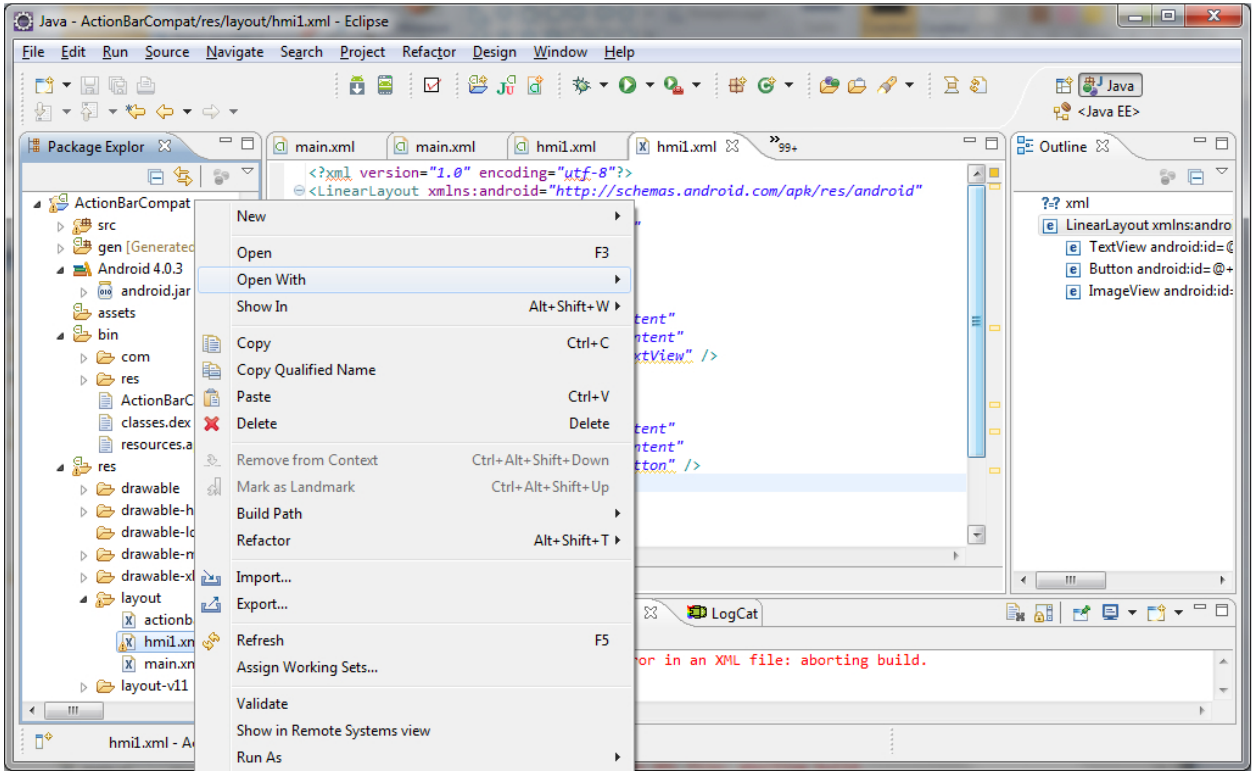
1. Créer un layout XML

Création manuelle

Documentation disponible sur
http ://developer.android.com/guide/topics/ui/declaring-layout.html

Éditer le layout avec un éditeur XML.

Exemple sous eclipse :



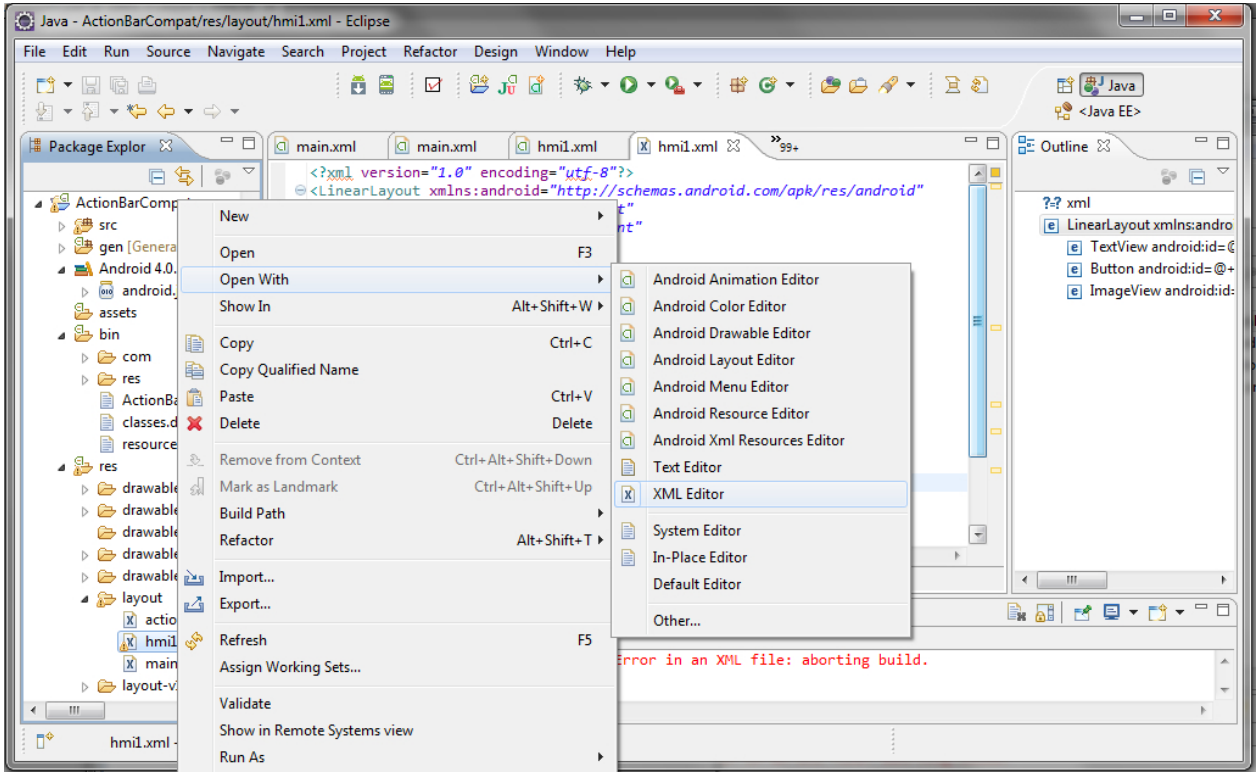
1. Créer un layout XML

Création manuelle

Documentation disponible sur
http ://developer.android.com/guide/topics/ui/declaring-layout.html

Éditer le layout avec un éditeur XML.

Exemple sous eclipse :



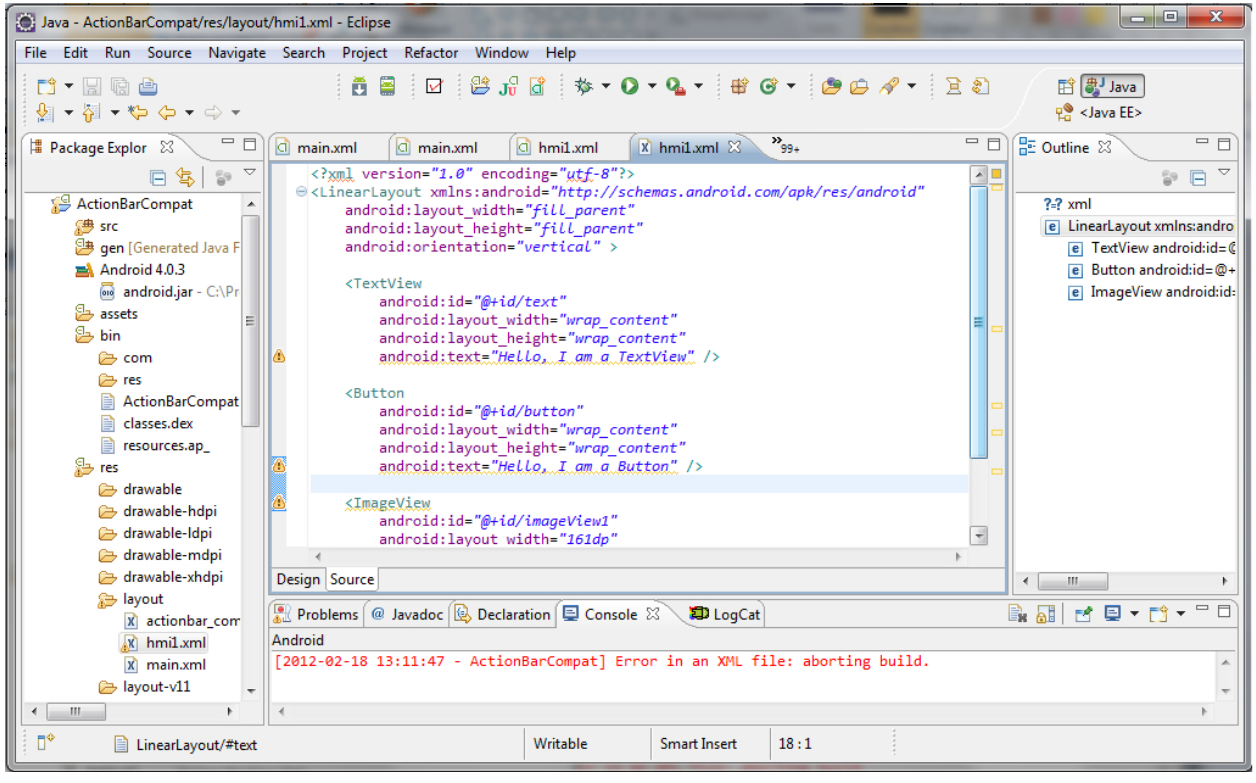
1. Créer un layout XML

Création manuelle

Documentation disponible sur
http ://developer.android.com/guide/topics/ui/declaring-layout.html

Éditer le layout avec un éditeur XML.

Exemple sous eclipse :



1. Créer un layout XML

Choix du layout

LinearLayout : Il organise les différents éléments de votre interface sur une ligne ou sur une colonne.

AbsoluteLayout : Cette mise en page vous laisse définir les coordonnées exactes des éléments qui le composent.

RelativeLayout : Il permet de définir la position des éléments en fonction de la position de leurs éléments parents. On ne peut pas avoir de dépendance circulaire dans la taille dans le RelativeLayout.

TableLayout : Cette mise en page peut se faire en colonne et en ligne.

Placer les widgets

EditText : Un champ de texte éditable

Toast : Un pop up message qui s'affiche sur l'écran

ImageView : Une image

CheckBox : Une case à cocher

Button : Un bouton cliquable

RadioButton : sélecteurs/interrupteurs.

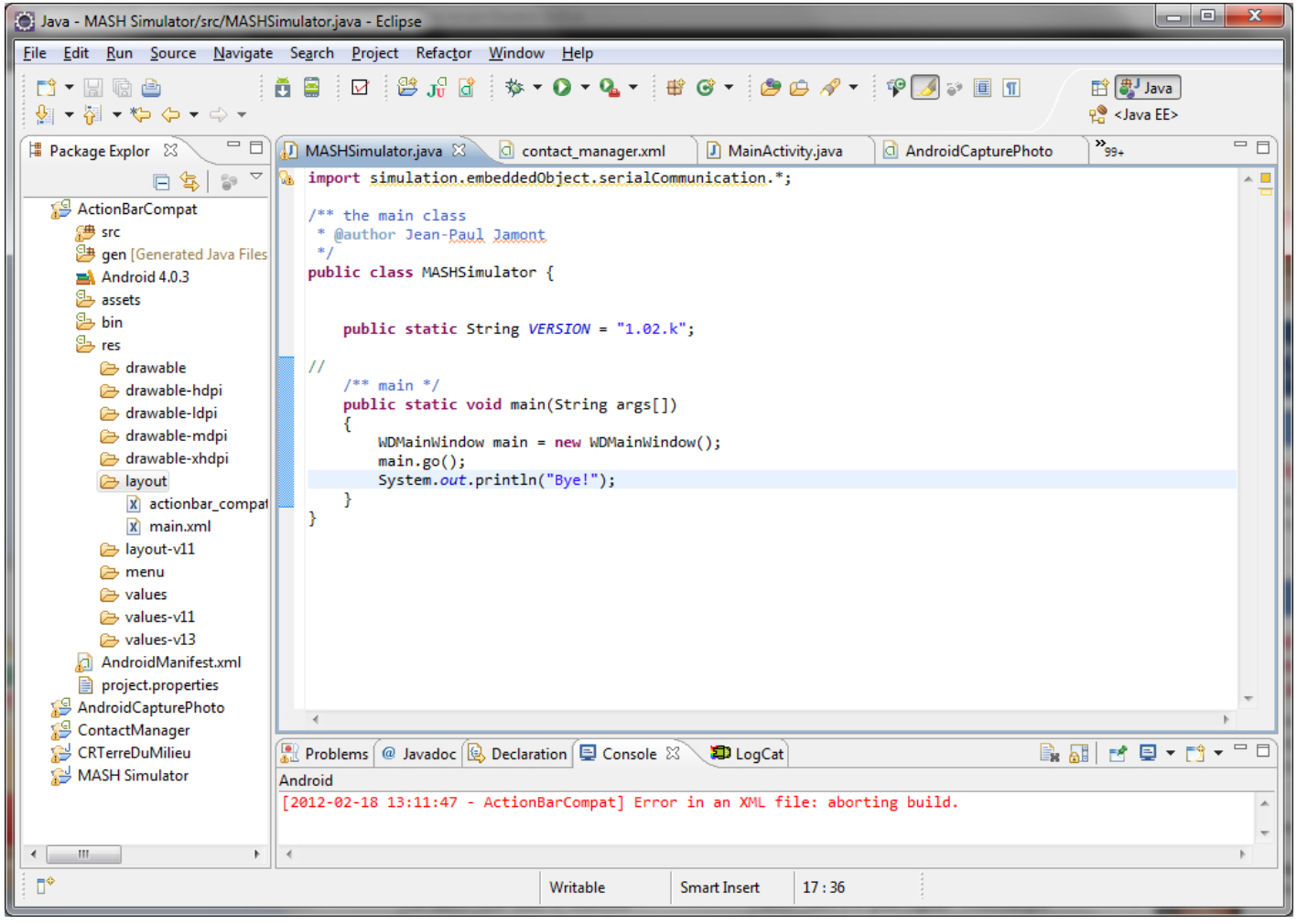
ImageButton : Une image qui se comporte comme un bouton

DatePicker : Un sélecteur de dates

SlidingDrawer : Un élément qui se présente sous forme d'un tiroir qu'on ouvre et ferme

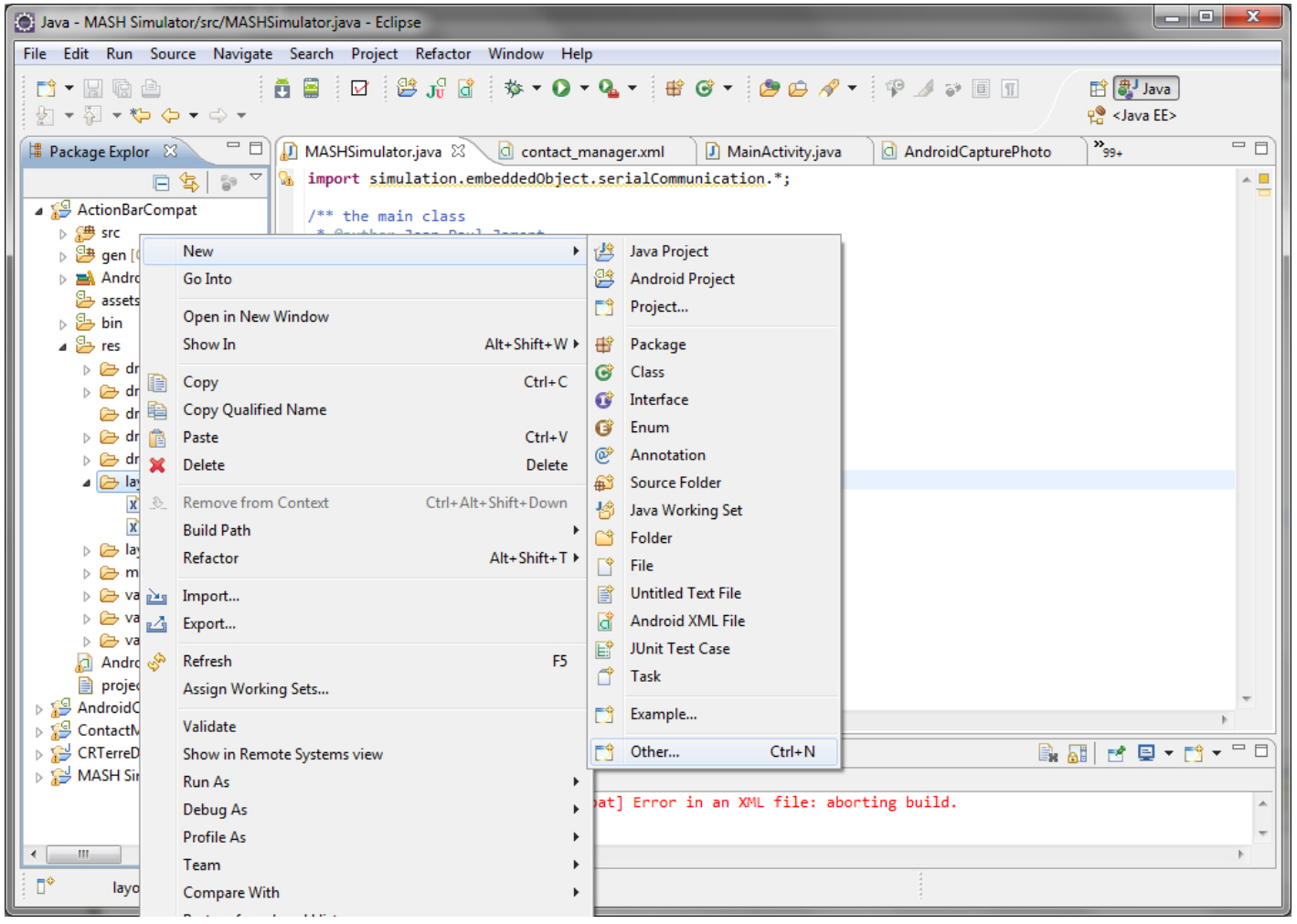
1. Créer un layout XML

Création avec plugin (ou autre outil externe)



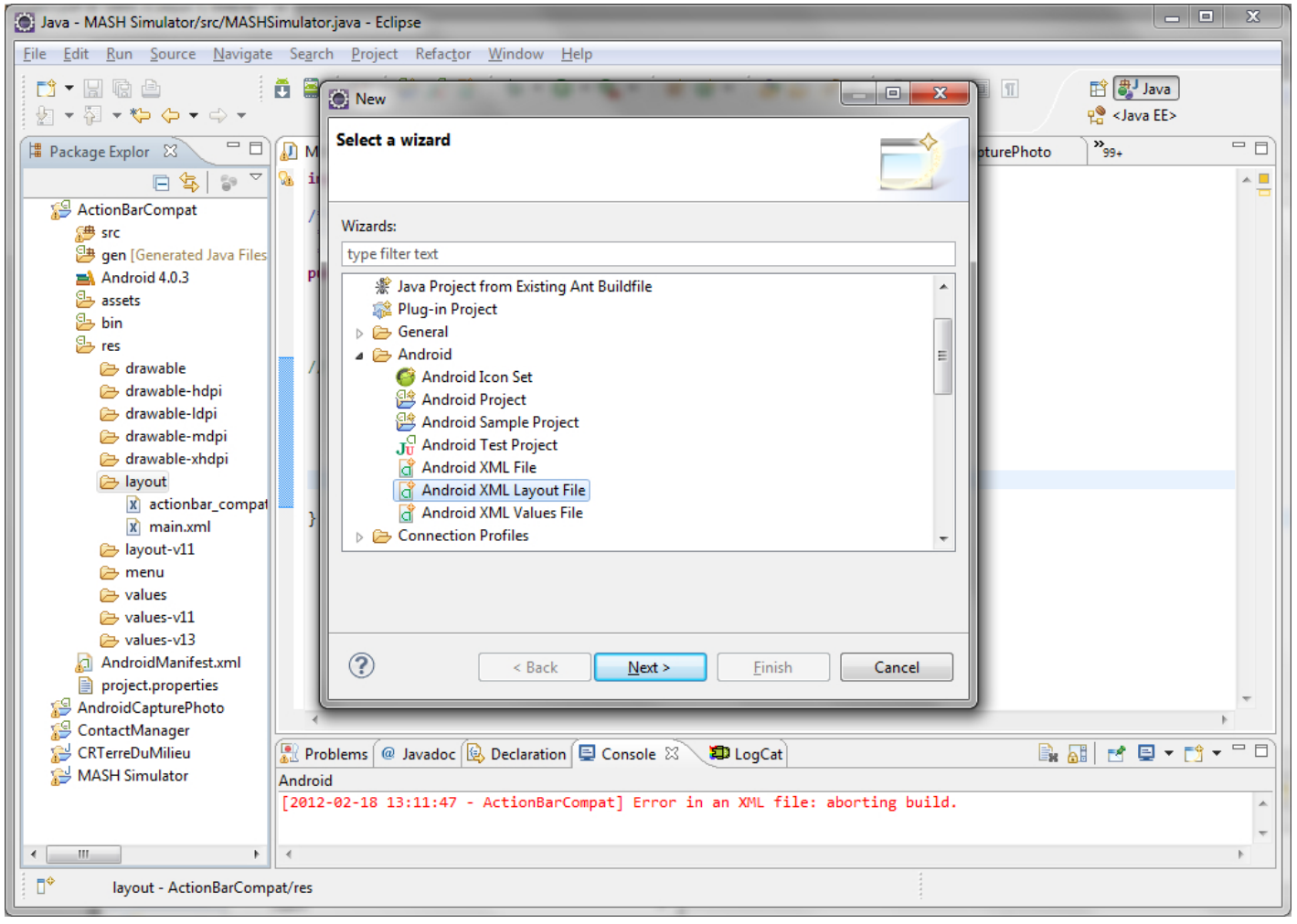
1. Créer un layout XML

Création avec plugin (ou autre outil externe)



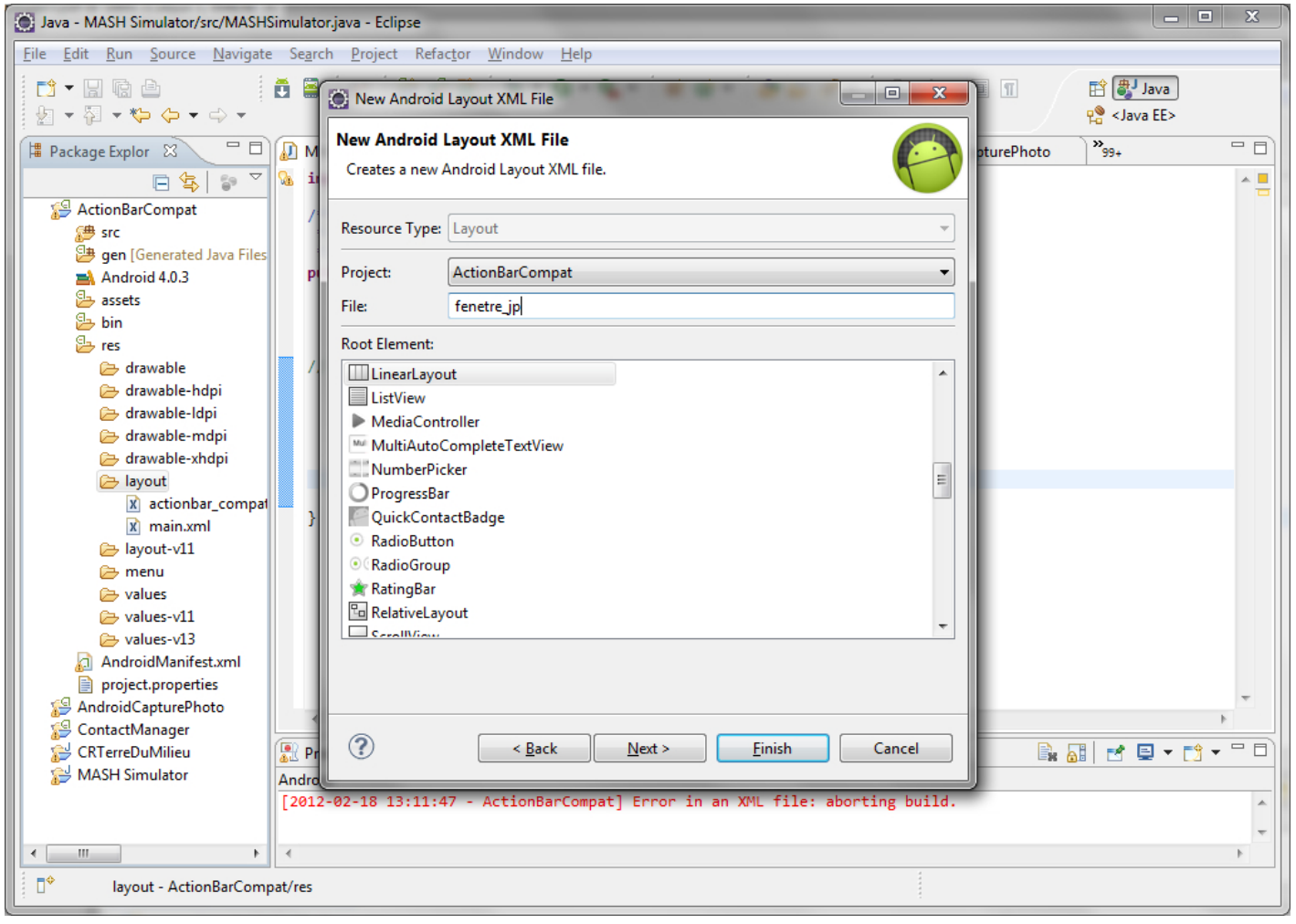
1. Créer un layout XML

Création avec plugin (ou autre outil externe)



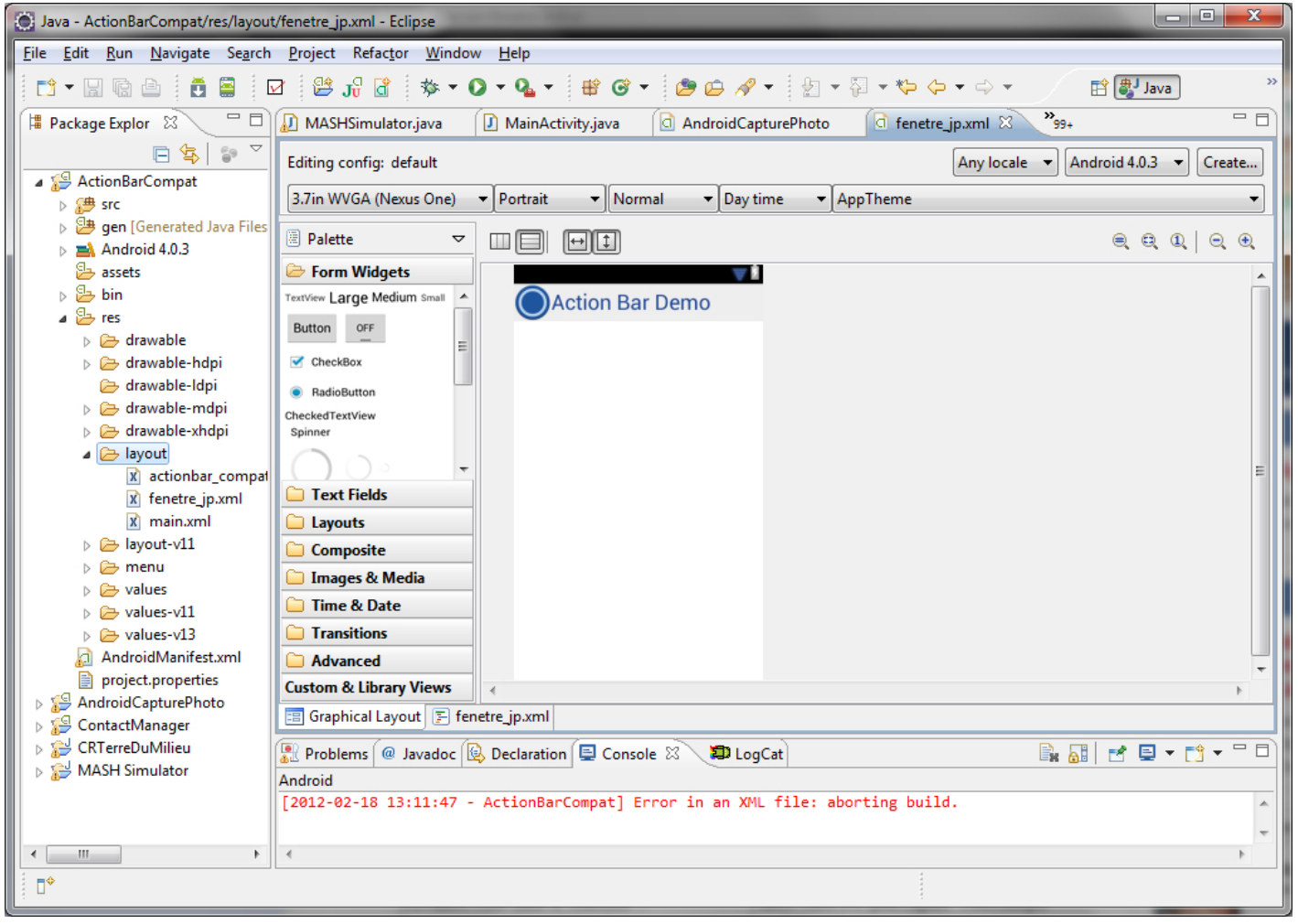
1. Créer un layout XML

Création avec plugin (ou autre outil externe)



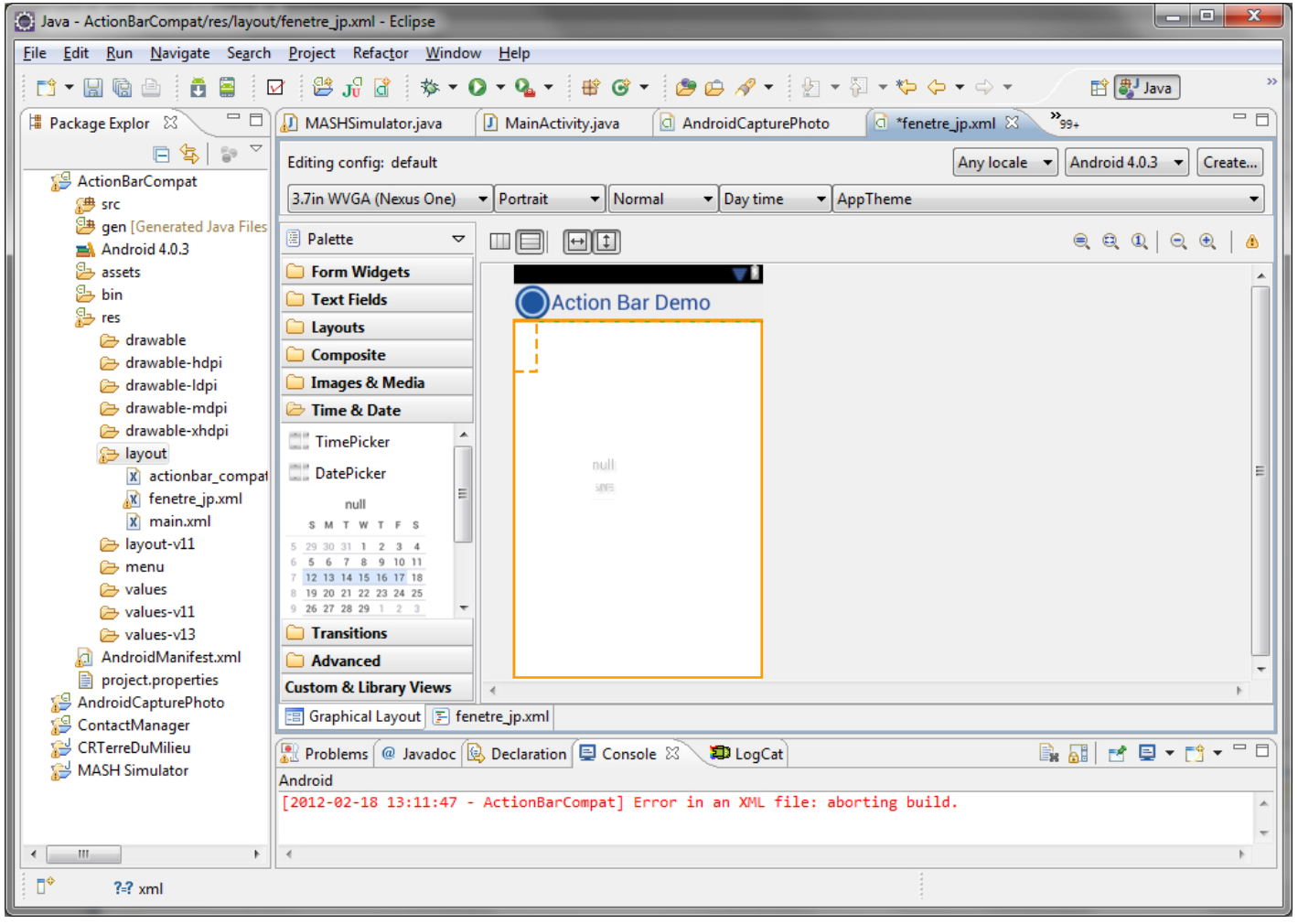
1. Créer un layout XML

Création avec plugin (ou autre outil externe)



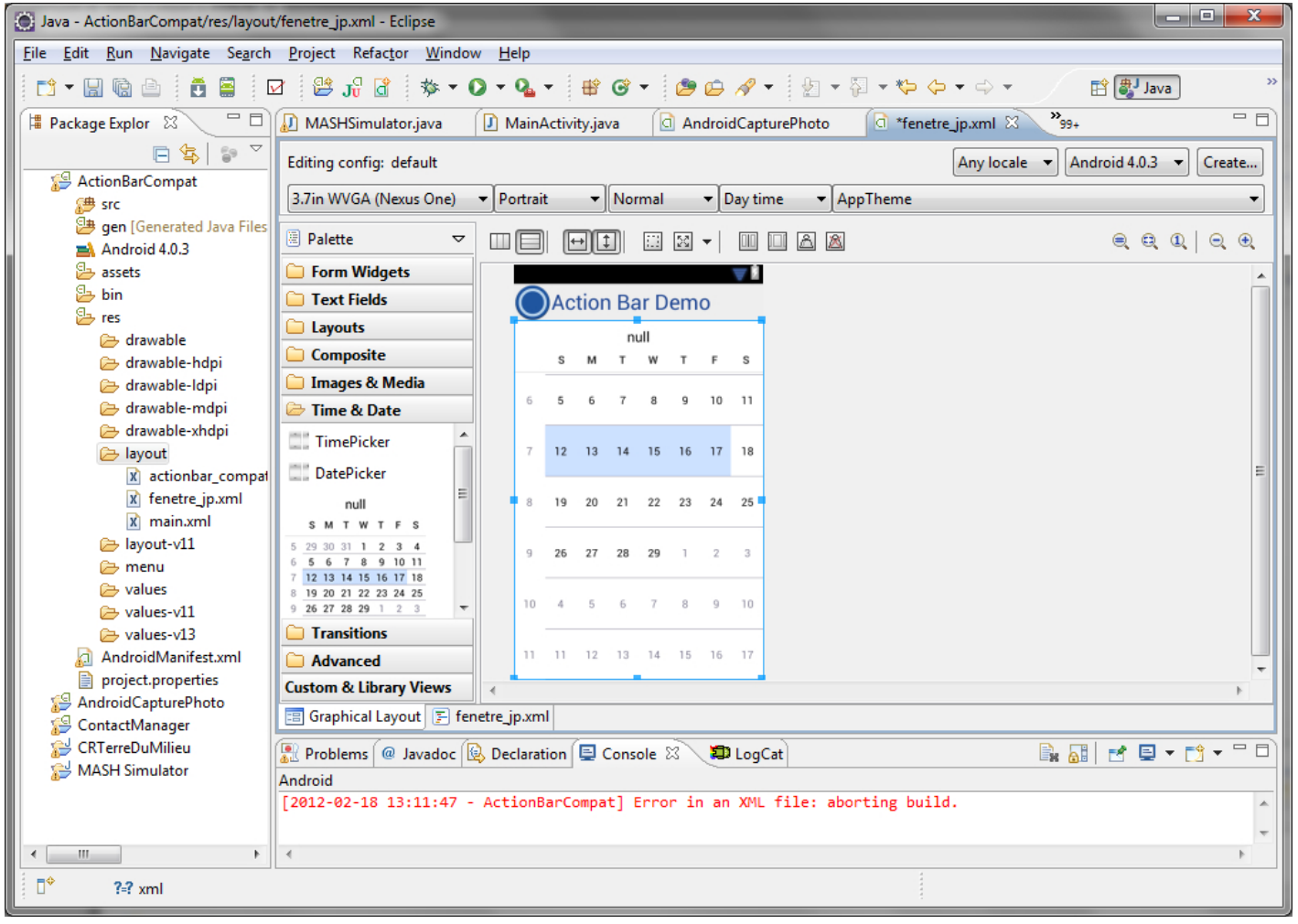
1. Créer un layout XML

Création avec plugin (ou autre outil externe)



1. Créer un layout XML

Création avec plugin (ou autre outil externe)



Intégrer des layouts XML dans mon application

2. Charger le layout

- Quand l'application est compilée, chaque layout XML est compilée en une ressource `View`.
- Le code applicatif doit alors charger le layout dans la méthode `onCreate()` de l'activité via un appel de la méthode `setContentView()`
- La méthode `setContentView()` prend en paramètre la référence vers le layout. Cette référence est de la forme `R.layout.nom_du_layout`.
  Exemple : Soit le layout `main_layout.xml`. *Dans l'activité, la méthode* `onCreate()` *doit contenir au minimum* :

```
1   public void onCreate(Bundle savedInstanceState) {
2       super.onCreate(savedInstanceState);
3       setContentView(R.layout.main_layout);
4   }
```

Intégrer des layouts XML dans mon application

3. Instancier les widgets sur lesquels on veut agir

Il va falloir maintenant dans la méthode `onCreate()` de votre activité créer une instance de chaque objet de votre layout XML que vous souhaitez manipuler.

Exemple :

```
Button myButton = (Button) findViewById(R.id.my_button);
myButton.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
      // Perform action on click
    }
});
```

Plan

Manifeste

Avant-propos

- Fichier XML
- Précise l'architecture de l'application
- Chaque application doit en avoir un
- `AndroidManifest.xml` est dans la racine du projet

Manifeste

Contenu

- Précise le nom du package java utilisant l'application. Cela sert d'identifiant unique !
- Décrit les composants de l'application
  - Liste des activités, services, broadcast receivers
  - Précise les classes qui les implémentent
  - Précise leurs capacités (à quels intents ils réagissent)

  ⇒ Ceci permet au système de savoir comment lancer chaque partie de l'application afin de satisfaire au principe de réutilisabilité.
- Définit les permissions de l'application
  - Droit de passer des appels
  - Droit d'accéder à Internet
  - Droit d'accéder au GPS
  - ...
- Précise la version d'Android minimum nécessaire
- Déclare les librairies utilisées
- Déclare des outils d'Instrumentation (uniquement pour le développement)

Manifeste

Conventions

- Seuls deux éléments sont obligatoires :
  - `<manifest>` : contient le package, la version... Englobe tout le fichier
  - `<application>` :décrit l'application et contiendra la liste de ses composants
- Les données sont passées en tant qu'attribut et non en tant que contenu
- Tous les attributs commencent par `android:` (sauf quelques un dans `<manifest>`)

Manifeste

Les ressources

- Au lieu de contenir les données en tant que tel le fichier manifest peut faire appel à des ressources
- `<activityandroid:icon="@drawable=smallPic":::>`
- Ces ressources sont définies dans le répertoire `res` de l'application.

Manifeste

Permissions

- Une application ne peux pas utiliser certaines fonctionnalités sauf s'il le précise dans le fichier manifest
- Il faut donc préciser les permissions nécessaires grâce à : `<uses-permission>`
- Il existe des permissions standards :
  - `android.permission.CALL_EMERGENCY_NUMBERS`
  - `android.permission.READ_OWNER_DATA`
  - `android.permission.SET_WALLPAPER`
  - `android.permission.DEVICE_POWER`
- Il est possible de définir ses propres permissions

Manifeste

Intent Filter

- Ils informent le système à quelles `intent` les composants peuvent réagir
- Un composant peut avoir plusieurs filtres
- Exemple : Cas d'un éditeur de texte
  - Filtre pour éditer un document existant
  - Filtre pour initier un nouveau document
- Un filtre doit posséder une "action" qui définit à quoi il correspond

Manifeste

Exemple

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3        package="com.eyrolles.android.multimedia"
4        android:versionCode="1"
5        android:versionName="1.0">
6      <application android:icon="@drawable/icon" android:label="@string/app_name">
7          <activity android:name=".CapturePhoto"
8                    android:label="@string/app_name">
9              <intent-filter>
10                 <action android:name="android.intent.action.MAIN" />
11                 <category android:name="android.intent.category.LAUNCHER" />
12             </intent-filter>
13         </activity>
14     </application>
15     <uses-sdk android:minSdkVersion="3" />
16     <uses-permission android:name="android.permission.CAMERA"/>
17 </manifest>
```

Plan

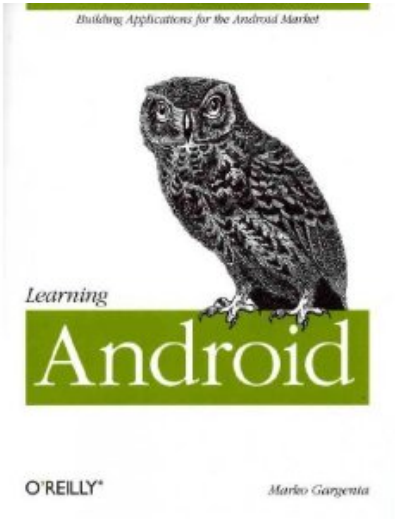
Travaillons à partir de la documentation en ligne

- android.widget.Toast
- android.os.CountDownTimer
- android.os.BatteryManager
- android.speech.tts.TextToSpeech
- android.hardware.Camera
- android.hardware.Sensor

Plan

Ressources

Livres



Autres documents (cours...)

- Romain Raveaux, *Cours Android --- Développement et API*, Laboratoire L3I, IUT de La Rochelle.
- Nazim Benbourahla, Diverses ressources fournies via `Developpez.com`.
- Philippe Lacomme, Raksmey Phan, *Créer des applications Android*