eXpertise @ndroid

Développement d'applications Android

Table des matières

Introduction à Android

Interface graphique

Les activity et leur intéractions

Données persistantes

Table des matières

Services et Alarmes

Accès distants et multithreading

Autres composants

Liens utiles

eXpertise @ndroid

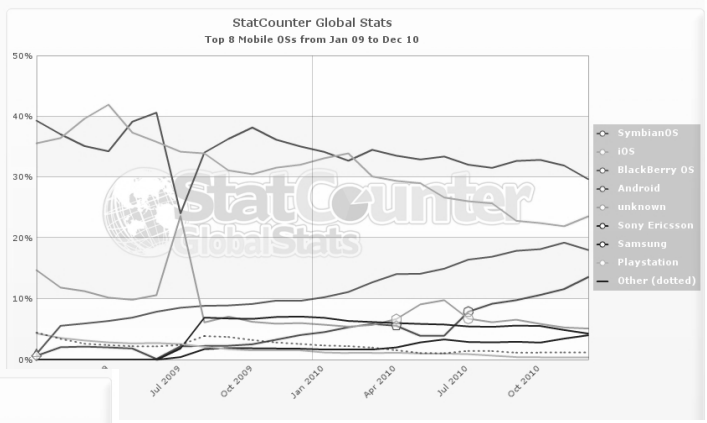
Présentation Android

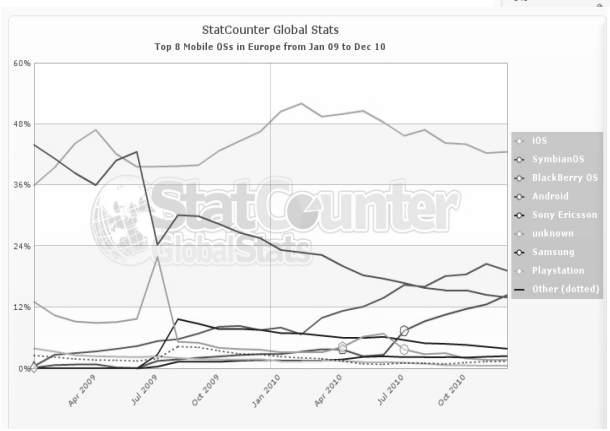
Présentation Android

- Google
  - Acteur majeur d'internet
  - 1er moteur de recherche
  - 1er publicité en ligne
  - Solutions d'entreprises: Google Apps, Google Enterprise Appliance...
  - Innovation et services gratuits: gmail, apps, photos, vidéos...
- 2005: Rachat d'Android Inc.
  - Développement d'une solution mobile basée sur Linux
  - Richard Miner (jusqu'à fin 2008) + Andy Rubin
- Objectif du rachat
  - Convergence Web et Mobile
  - Créer une grande communauté Open Source Mobile (constructeurs, intégrateurs, développeurs...)
  - Système le plus flexible possible
- Janvier 2010 : Vendre du matériel Android (Nexus One, Nexus Two...)

Présentation Android

- Acteurs des OS mobiles
  - Symbian
  - Windows Mobile
  - Blackberry / RIM
  - Palm Source / Access
  - Apple
  - Linux





eXpertise @android

Les ventes de smartphone dans le monde au premier trimestre 2009 en millions d'unités

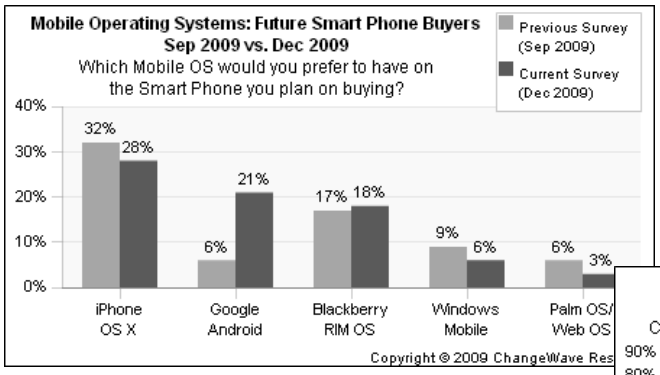
Société	Ventes T1/09	Part de marché T1/09 (en %)	Ventes T1/08	Part de marché T1/08 (en%)	Croissance (en %)
Nokia	14,99	41,2	14,59	45,1	2,8
Research In Motion	7,23	19,9	4,31	13,3	67,8
Apple	**3,94**	**10,8**	**1,73**	**5,3**	**128,3**
HTC	1,96	5,4	1,28	4,0	53,3
Fujitsu	1,39	3,8	1,32	4,1	5,3
Autres	6,90	18,8	9,09	28,1	-24,2
Total	**36,40**	**100,0**	**32,31**	**100,0**	12,7

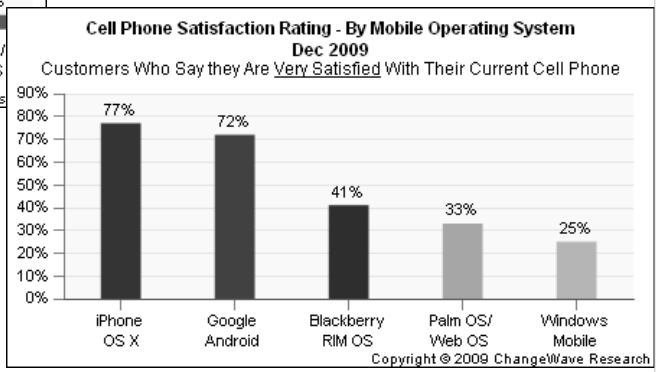
Source: Gartner Estimates Mai 2009

Gartner lors d'une étude parue en novembre 2009, a annoncé que des prédictions montraient qu'Android serait le deuxième système d'exploitation le plus utilisé d'ici 2014, derrière Symbian !
Eric Schmidt à annoncé au WMC2010 que 60000 matériels se vendaient chaque jour

Présentation Android

- Satisfaction utilisateur





Alors qu'Android n'était considéré que par 1% des utilisateurs de smartphone en septembre 2008, à la fin de l'année 2009 21% des utilisateurs de smartphones indiquaient vouloir acquérir un smartphone fonctionnant avec Android !

Présentation Android

- Open Handset Alliance (OHA)
  - 30 partenaires en 2007, 47 en 2009, 60 fin 2010, 83 en 2011

Présentation Android

Avantages et inconvénients

Constructeurs
+ Linux – Open Source
+ Coût de licence nul
+ Adaptabilité
-Accord Google (services)
- Évolution rapide

Développeurs
+ Langage Java
+ Modularité, partage
+ Kit de développement gratuit
+ SDK complet
+ Android Market
- Android Market
- Évolution du kit
- Diversité des matériels

Utilisateurs
+ Fonctionnel, intuitif et évolutif
+ Google, services de messagerie
+ Multitâches
+ Applications nouvelles
+ Nombreuses applications par défaut
+ Un seul OS pour tous les appareils ?

eXpertise @ndroid

Le constructeur doit disposer d'un accord avec Google pour pouvoir proposer les applications Google sur son matériel, notamment l'Android Market, l'une des forces de la plateforme. Or, pour disposer de cet accord, le matériel doit répondre à des spécifications bien particulières : touchscreen, GSM, appareil photo. Sans accord avec Google, les utilisateurs devront trouver des solutions alternatives pour installer des applications sur leur matériel, ou le constructeur devra proposer une plateforme de distribution logicielle. Ce manque peut être rattrapé par des plateformes opérateurs.

L'évolution très rapide de la plateforme (5 versions majeures en un an), provoque des difficultés chez les constructeurs à faire évoluer leurs matériels pour fournir à leurs utilisateurs la dernière version du système.

L'Android Market est un point fort pour les développeurs mais également une faiblesse, notamment liée à la règle des 24 heures, qui permet à un utilisateur de refuser l'achat d'un logiciel. Fonctionnement qui permet de pirater des applications : avec un téléphoné hacké, l'utilisateur va récupérer le binaire d'installation de l'application sur sa carte mémoire par exemple, refuser l'achat, et réinstaller l'application directement depuis sa carte mémoire. Estimation 20% à 30% de ventes annulées.

Présentation Android: matériel

- Matériels disponibles
  - Smartphones
  - Lecteurs multimédia
  - GPS
  - Netbook
  - eReader
  - Machine à laver
  - ...
- Constructeurs présents
  - HTC, Samsung, LG, Motorola, Huawei, Acer, Archos, Philips, Sony, Asus/Garmin...
- Nouveaux constructeurs
  - ZTE, oPhone...
  - Ceux qu'on ne connaît pas encore

La présence d'Android par Fujitsu

Lors du lancement du Google Nexus One, Google a annoncé
que le nombre de matériels devraient plus que doubler sur l'année 2010 !
Tous les constructeurs se sont aujourd'hui penchés sur la plateforme Android
et une grande partie ont fait le pas de commercialiser des produits.
La stratégie Open Source de la plateforme a également poussé d'autres
constructeurs à libérer leur code source : dernier exemple la société française Archos,
spécialisée dans la création de lecteur vidéo portable.

Présentation Android: anatomie

- Android: 2 parties
  - Système d'exploitation Linux
  - Environnement d'exécution Dalvik (Java)
- Kit de développement Java disponible depuis 2007
  - Concours Android Developer Challenge (Google) – 4 versions
  - SDK beta : octobre 2007
  - SDK public : octobre 2008
  - Mai 2009 : NDK
  - Novembre 2009 : SDK Updater
  - ADK 2011 : Android Open Accessory API
- Sources (OS + SDK) disponibles depuis novembre 2008
- Applications payantes depuis février 2009
- Développement : SDK et/ou Sources

eXpertise @ndroid

Tous les matériels peuvent disposer de la dernière mise à jour du système d'exploitation.

Les mises à jour se présentent aujourd'hui sous deux formes :

-OTA (Over The Air) : le smartphone reçoit automatiquement une mise du système. L'utilisateur est averti qu'une mise à jour est disponible. Il peut refuser son installation, mais sera avisé 3 heures plus tard.

-Manuellement : souvent proposé par le constructeur, ce dernier met à disposition un fichier d'archive à disposition contenant la dernière mise à jour ainsi que la documentation d'installation à réaliser via le cable USB et une carte mémoire.

Il est bien sûr beaucoup plus pratique pour l'utilisateur lambda de se voir notifier des mises à jour sans qu'il n'interviennent sur quoique ce soit. Les mises à jour du système tout en conservant les données de l'utilisateur.

Présentation Android: fonctionnalités

- Fonctionnalités
  - Navigation web, réseaux sociaux
  - Connexion Wifi, Bluetooth, GPS, NFC (2.3)
  - Maps, Logiciel de navigation
  - Multimédia : audio, vidéo, photo
  - Accéléromètre, gyroscope
  - Affichage 2D, 3D, Gaming
  - Réalité augmentée
  - Chaque application est égale : remplacement possible
  - Tout composant est modifiable
  - Agir sur des applications existantes
  - Appel d'un écran sans en connaître le nom ou chemin

eXpertise @ndroid

Tous les matériels peuvent disposer de la dernière mise à jour du système d'exploitation.

Les mises à jour se présentent aujourd'hui sous deux formes :

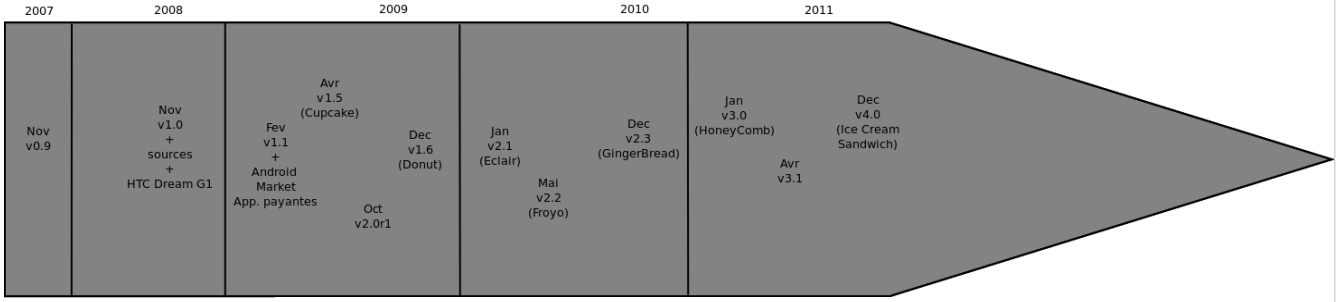
-OTA (Over The Air) : le smartphone reçoit automatiquement une mise du système. L'utilisateur est averti qu'une mise à jour est disponible. Il peut refuser son installation, mais sera avisé 3 heures plus tard.

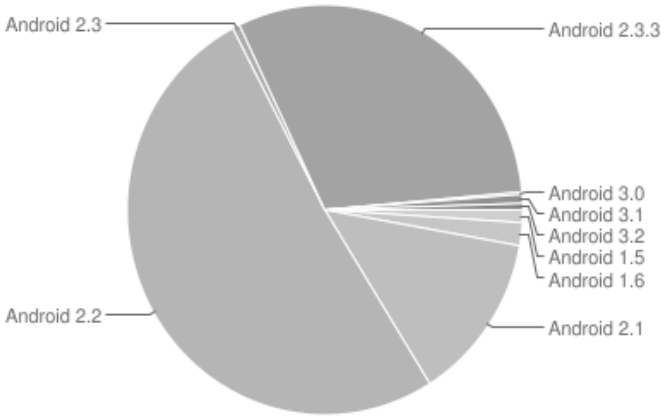
-Manuellement : souvent proposé par le constructeur, ce dernier met à disposition un fichier d'archive à disposition contenant la dernière mise à jour ainsi que la documentation d'installation à réaliser via le cable USB et une carte mémoire.

Il est bien sûr beaucoup plus pratique pour l'utilisateur lambda de se voir notifier des mises à jour sans qu'il n'interviennent sur quoique ce soit. Les mises à jour du système tout en conservant les données de l'utilisateur.

Présentation Android : anatomie

Click to add an outline





eXpertise @ndroid

D'usage nous aurions tendance à développer avec le dernier SDK disponible. Or, ceci engendrerait une perte d'utilisateur importante, ici 80% si le développement était réalisé sur la dernière version du SDK.

Deux solutions pour éviter cette perte d'utilisateurs potentiels :

-Développer avec un SDK 1.5 sans disposer de l'ensemble des fonctionnalités et correctifs. Réaliser un minimum de tests sur les SDK supérieurs.

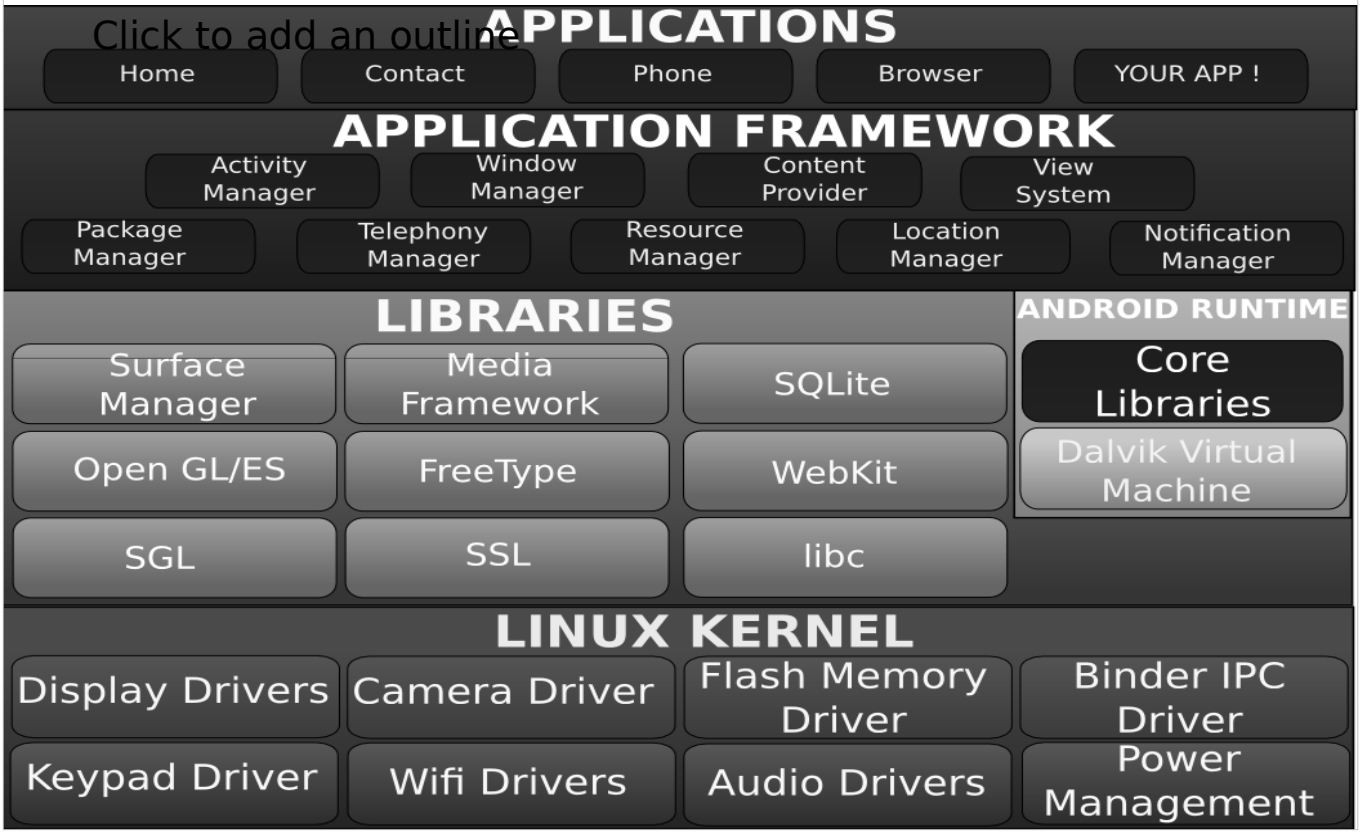
-Développer avec le SDK le plus récent en tenant compte :

- De ne pas utiliser les dernières APIs
- Réaliser de nombreux de tests de validation sur l'ensemble des plateformes précédentes
- Indiquer que l'application cible a été compilée sur l'environnement du dernier SDK disponible mais est capable de fonctionner sur un SDK minimum de 1.5

Pour réaliser cela, on le fera dans le descripteur de l'application minSDK = 3 (pour Android 1.5) targetSDK=6 (Android 2.0.1)

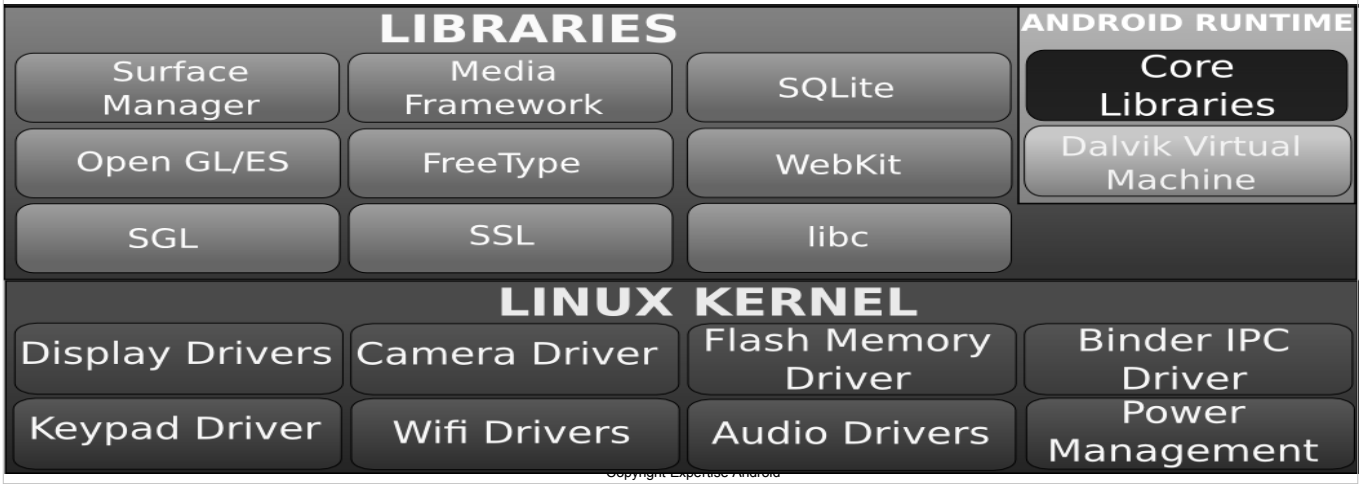
Présentation Android: Anatomie

Click to add an outline



Présentation Android: anatomie

- OS Android
  - Linux kernel 2.6 ARM
  - Pas de système natif X
  - Pas de support Glibc
  - Optimisation mémoire, processus et alimentation
  - Gestion utilisateurs
- Dalvik
  - VM Android
  - Optimisée embarqué
  - Multi Instance
  - Optimisation : bytecode, sécurité, PM
  - Fichier dex



Le kernel linux évolue avec les versions du SDK : 2.6.24 avec le SDK 1.5, 2.6.29 pour les SDK suivants

Linux a été un choix facile pour Google:

- Open Source
- drivers C facilement portables
- librairies partagées
- sécurité et gestion de la mémoire

Linux Kernel: OS de base ainsi que l'ensemble des drivers travaillant avec le matériel. Un opérateur ou un constructeur travaillera directement sur cette couche

Android utilise une librairie nommée Bionic Libc. Bionic est sous licence BSD. Elle ne pèse que 200ko soit environ la moitié de la libc standard. Cette librairie étant chargée pour chaque processus, sa taille devait être restreinte et ses performances accrues. Cette librairie gère également une grande partie du système d'authentification Google. Ne supporte pas certaines fonctionnalités de POSIX ou de C++. La création d'une librairie (ou d'un binaire) doit être réalisée en utilisant la librairie bionic et glibc.

Librairies: ensemble de librairies C apportant les fonctionnalités attendues par l'OS (gestion de l'affichage, codec, sécurité, 3D, gestion réseau...)

Core libraries: Ensemble des API de bases Java et accès ressources

Pour chaque application lancée par l'utilisateur, une machine Dalvik est instanciée. Il peut donc y avoir plusieurs machines virtuelles simultanément et totalement indépendant les unes des autres.

Présentation Android: anatomie

- Base de l'API
  - Point d'entrée pour les applications
  - Accès à toutes les ressources inférieures via librairie
  - Accès possible aux ressources C via JNI (bypass de cette couche)

eXpertise @ndroid

L'application framework est le point d'entrée pour accéder aux librairies fournies par le système et les ressources matérielles. Il contient tous les services auquel il sera possible de faire appel pour récupérer des informations des couches inférieures

Activity Manager : concerne l'application en cours

Windows Manager : toutes les informations sur l'écran

Content Provider : donne accès à des données partagées (base de données, fichiers…)

View System : système interne permettant la construction des vues de l'interface graphique

Package Manager : gestionnaire de packaging (pas uniquement de l'application)

Telephony Manager : informations et services liés à la téléphonie

Resource Manager : accès toutes les ressources (image, son, chaîne de caractères, layout…)

Location Managier : gestionnaire lié à la géolocalisation (Wifi, GSM, GPS)

Notification Manager : le système d'information utilisateur, nommé notification (réception d'un SMS par exemple)

Présentation Android : anatomie

- Open Source, différentes licences
  - Linux LGPL v2
  - Android Apache v2
  - Librairie Bionic BSD
  - Webkit LGPL v2
  - Sqlite : public
  - Freetype : GPL / BSD
  - ...
- Vos programmes et le code privé (attention web!)
  - com.android.* = privé = modification possible par les constructeurs

Le framework source fourni par Android contient du code privé et du code public.

Le code dit « privé » fait parti d'un développement spécifique par Google. Ce code peut ne pas être présent sur un appareil dont le constructeur aurait décidé de modifier.

Android.* : toujours disponible sur les OS Android

Com.android.* : risque de pas trouver la librairie !

Présentation Android: développement

- Environnement de développement
  - Windows
  - Mac OS > 10
  - Linux
- J2SE
  - 1.5
  - 1.6 *
- SDK Android + API
- Eclipse Ganymede / Galileo / Helios
  - Plugin ADT (Android Developer Toolkit)
- Autre IDE supporté : Netbeans, IntelliJ

eXpertise @ndroid

Concernant le système d'exploitation Linux, Google a basé toutes ses aides aux développeurs sur une distribution Linux Ubuntu 8.04 (compatibilité jusqu'à Ubuntu 10.04)

Un OS Linux permet de pouvoir travailler également avec les sources Android (Linux et framework) permettant un deboggage poussé.

Le développement peut également se faire avec un simple éditeur de texte et l'outil Ant. Pour des raisons de simplicité nous utiliserons l'IDE Eclipse, utilisé par Google pour ses démonstrations et connu de tout développeur Java

Présentation Android: développement

- Une application est une succession d'écrans
- Elle inclut un ensemble de descripteurs pour chaque écran
- Un écran peut ouvrir un autre écran d'une même application ou d'une autre application
- 5 composantes majeures
  - Intent
  - Activity
  - Broadcast Receiver
  - Content Provider
  - Service
- L'ensemble de ces composantes est décrit dans le fichier AndroidManifest.xml
- Les applications ont pour extension APK (Android PacKage)

Présentation Android: développement

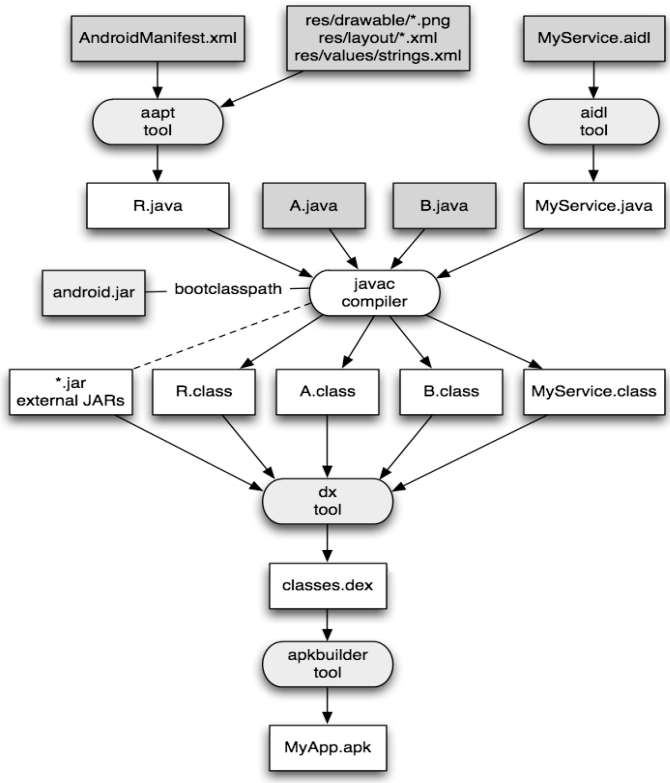
- Construction d'une application

Code source

Byte Code Java

Byte code optimisé
Génération fichier.dex

Construction du package
destiné à être installé



Le fichier APK (Android PacKage) contient toutes les ressources nécessaires à l'application dont les fichiers classes – compressés et optimisés au format DEX – ainsi que le descripteur de l'application AndroidManifest.xml.

Un apk peut être installé via l'Android Market, qui appelle directement l'Android Package Manager ou alors en utilisant la ligne de commande adb (install, push, pull...).

L'application est toujours installée sur /data/data/monpackage. L'espace disque pris par votre application dispose de son utilisateur et ne peut accéder aux autres espaces d'applications, excepté en utilisant les Content Providers pour les applications qui les ont autorisés.

Présentation Android: TD0 – Hello Android

- Les grandes étapes (avec connexion Internet !)
  - Télécharger et installer le Sun JDK 1.5 ou 1.6
  - Télécharger et installer l'IDE Eclipse
  - Télécharger le SDK Android
  - Télécharger l'Android Developer Tools (ADT)
  - Télécharger les dépendances Android (SDK) (API, Tools, Doc., API maps...)
- Installer et configurer ADT pour référencer le Android SDK
  - Ouvrir Eclipse
  - Cliquer sur Help/Software updates
  - Cliquer sur le bouton Add site
  - Saisir l'URL: https://dl-ssl.google.com/android/eclipse/
  - Cocher la case « Developer tools » incluant deux packages
  - Valider la licence et installer
  - Redémarrer Eclipse
  - Dans Eclipse, cliquer sur Windows/Preferences puis Android sur le panneau de gauche
  - A droite renseigner le chemin au SDK précédemment décompressé
  - Redémarrer Eclipse

eXpertise @ndroid

Les postes disposent d'un environnement Windows ou Linux

URLs de téléchargement:

J2SE JDK 1.5 (ou JDK 1.6): http://java.sun.com/javase/downloads/index_jdk5.jsp

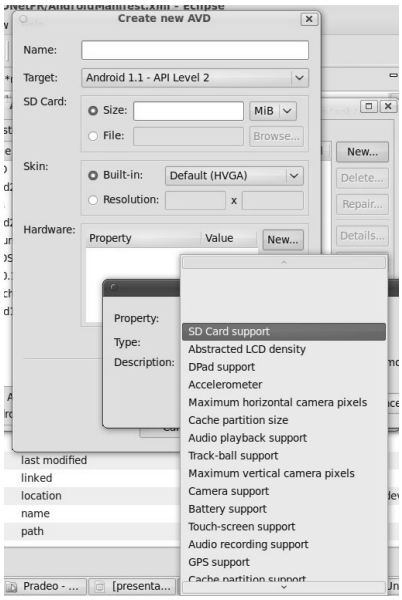
Eclipse IDE Galileo for Java Developer: http://www.eclipse.org/downloads/

Android SDK: http://developer.android.com/intl/de/sdk/index.html et décompresser le fichier à l'endroit de votre choix

Pour le plugin adt, s'il y a une erreur au téléchargement utiliser le protocole non sécurisé (http à la place de http)

Présentation Android: TD0 – Hello Android

- AVD : depuis Eclipse ☺



- Création manuelle si besoin
  - Ouvrir une console cmd
  - Se placer dans [PATH_SDK]/tools
  - Saisir la commande suivante

```
android create avd –target 3 –name avdimage
```

  - Relancer Eclipse

L'ensemble des fonctionnalités disponibles depuis l'interface Eclipse correspondent à des utilitaires propres au SDK Android (disponible dans votre répertoire SDK) ou Java (JDK).

Exemple : http://developer.android.com/guide/developing/tools/avd.html

Par défaut,à la création d'un AVD, toutes options possibles pour un smartphone sont disponibles sur la machine virtuelle.

Sous Windows les fichiers AVD seront situés sous

C:\Documents and Settings\Utilisateur\.android\avd

Sous Linux /home/utilisateur/.android/avd

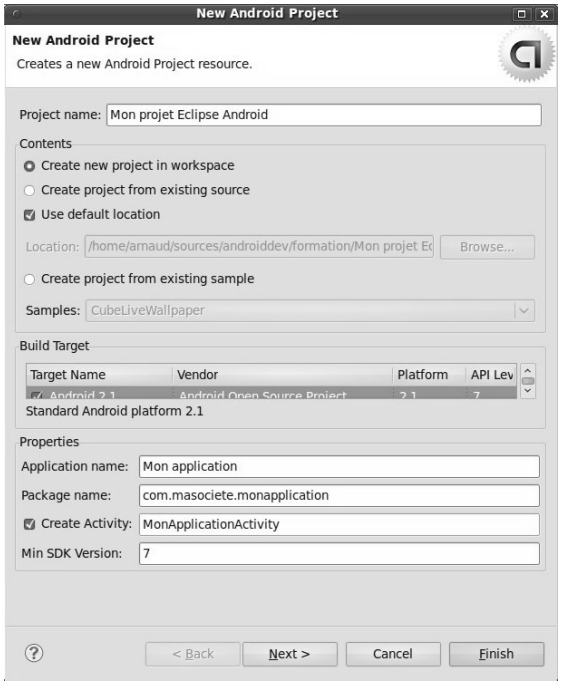
Exemple de commande:

android create avd -n cupcake_camera -t 3 hw.camera

android create avd -n cupcake_camera -t 2

Présentation Android: TD0 – Hello Android

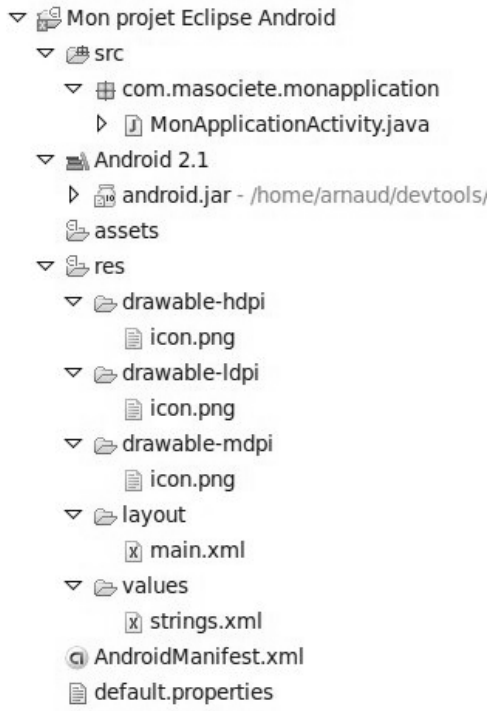
- Créer un nouveau projet Android selon le modèle ci-dessous



Présentation Android: TD0 – Hello Android

- Arborescence du fichier créé

- Construire le projet via le menu Project/Build



eXpertise @ndroid

Src : les sources Java du projet

Gen : généré par le framework, ne pas modifier manuellement

Assets : peut contenir des binaires, fichiers, qui seront ensuite exploitable au sein de l'application. N'y mettre que ce qui ne peut entrer dans le répertoire res

Res : toutes les ressources requises pour le fonctionnement de l'application : images, sons mais également toutes les valeurs statiques (texte, couleur, menu, vues…)

AndroidManifest : descripteur de l'application

Default.properties : fichier non inclus dans le SDK final, utilisé par ADT

Présentation Android: TD0 – Hello Android

- Exécuter votre premier programme
  - Menu Run puis Run Configuration
  - Cliquez sur Android puis sur l'icône « New »

  - Saisir le nom souhaité pour votre configuration d'exécution
  - Cliquer sur le bouton « Browse » pour sélectionner votre projet
  - L'onglet « Target » devrait afficher le fichier AVD précédemment créé
  - Cliquer sur « Apply » pour sauvegarder puis « Run » pour exécuter

- Construire le projet via le menu Project/Build

Si vous disposez de plusieurs activity au sein de votre projet vous devrez spécifier, dans le premier onglet Android, l'activity à lancer dans la zone « Launch »

L'émulateur peut être lancé avec différentes options à son lancement (la définition d'un proxy par exemple). Ceci est à réaliser sur l'onglet Target dans la zone « Additionnal Emulator Command Line options ».

La première exécution est toujours plus longue car il s'agit d'un émulateur qui charge un système d'exploitation Linux. Les autres exécutions s'opéreront plus rapidement. Une fois l'émulateur démarré, ne l'arrêtez pas : le plugin adt est capable de détecter qu'un émulateur est déjà lancé et lancera votre application (run/history) directement dans le bon émulateur.

Plus de renseignements sur les options de l'émulateur disponibles sur

http://developer.android.com/guide/developing/tools/emulator.html

Gérer des logs

- Utiliser des Logs
  - Niveau d’erreur : e(rror), w(arning), d(ebug), i(nfo), v(erbose)
  - Import de la classe android.util.Log
  - Deux paramètres : TAG et message
  - Log.e(‘’Mon TAG’’, ‘’Un message d’erreur’’);
- Visualiser les Logs
  - Eclipse : Window / Open Perspective / DDMS
  - Onglet Log
  - Visualisation de tous les messages émulateurs ou matériels
  - Filtrage possible sur le TAG, le niveau d’erreur
  - Export au format texte

eXpertise @ndroid

Interfaces graphiques

IHM: présentation

- Construction différente de J2ME, AWT ou Swing
- Deux méthodes de création
  - XML (recommandée)
  - Code Java
- Avantages XML
  - Allègement et lisibilité du code Java
  - Facilité d'accès (langage XML)
  - Rapidité de développement (Similaire à un page HTML)
  - Faible risque d'erreur dans la construction
- Nombreux composants natifs
- Personnalisation possibles
- Fenêtre > Conteneurs > Vues

IHM: les layouts, les vues

- Layout = contenant mais aussi contenu (view spécifique)
  - Présent dans /res/layout
    - Comme tout autre composant dans /res peut être sous classé en fonction de la langue, la résolution...
  - Représente tout ou partie d'un écran
- Chaque composant
  - Dispose d'une référence personnalisable
  - Peut être accédé, ajouté, modifié et supprimé depuis le code Java
- Création d'une référence et accès à un composant existant
  - android:id=''@+id/myid'' pour créer la référence
  - ''@id/myid'' pour accéder à la référence
  - ''@android:id/empty'': accède à la référence d'un composant Android
- Chaque référence et ressource est géré dans R.java
- Compiler sous forme de vue embarquée dans le fichier dex

eXpertise @ndroid

• Un identifiant pourra être utilisé dans plusieurs fichiers de layout dès que ce dernier n'est pas présent plusieurs fois au sein d'un même écran

• Aapt génère un fichier R.java à chaque compilation contenant tous les identifiants des ressources utilisées par le projet. Ce fichier servira à retrouver fichiers ressources au moment où elles sont appelées

• Les noms des fichiers de ressources (string, layout, images, son...) ne doivent contenir ni majuscules, ni chiffres, ni caractères spéciaux. Leur nom doit respecter le codage UTF-8.

•Possibilité de créer des layouts qui seront inclus dans d'autres layouts utilisation de la balise include.

IHM: les layouts, les vues

- Différents contenus « View » - dit Widget
  - Contenu dans un Layout
  - Button, TextView, EditText, ImageView, Lists, CheckBox...
  - Peut être étendus par extension
- Composant de type « Layout »
  - FrameLayout: Basiquement utilisé pour placer un seul élément (coin haut gauche)
  - LinearLayout: succession verticale ou horizontale d'éléments
  - RelativeLayout: positionnement relatif à un parent ou un autre composant
  - TableLayout: comme un tableau en HTML (colonne, ligne)

IHM: les layouts, les vues

- Ecrire son fichier XML
  - Contient obligatoirement un élément racine (Root)
  - Peut contenir autant de layout ou vue que désiré (attention à la profondeur - layoutopt)
  - Sauvegardé sous /res/layout/monlayout.xml
  - Ne pas contenir de majuscule

```
<?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">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello"/>
</LinearLayout>
```


• Ceci est le contenu d'un fichier généré lors de la création d'un projet Android dans Eclipse (main.xml) disponible dans /res/layout

• Il définit trois éléments

- Un root élément (LinearLayout) – OBLIGATOIRE – qui doit être un View ou ViewGroup
- Un LinearLayout contenant ...
- Une zone de texte (TextView) dans laquelle la valeur @string/hello est affichée

• @string/hello fait référence à une valeur définie dans le fichier string.xml, qui contient l'ensemble des valeurs à afficher au sein des IHM

IHM: les layouts, les vues

- Charger le layout désiré
  - Référencé sur le nom de fichier (sans extension) dans R.java
  - R.layout.monlayout
  - setContentView(layout)

```
public void onCreate(Bundle savedInstanceState){
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
}
```

- Un layout peut être chargé au sein d'un autre composant en utilisant l'objet LayoutInflater
- Exemple de ViewGroup: GridView, Gallery, Spinner, ListView, ...
- Chaque composant dispose d'un certain nombre de paramètres définissant l'aspect
  - android:layout_width et android:layout_height (fill_parent, wrap_content...)
  - android:id, android:text, android:background...
- Chaque composant dispose de ses arguments propres:
  - android:orientation pour LinearLayout (vertical, horizontal)

• La méthode onCreate est appelée lors de la création d'une Activity

• Gallery: Affiche une liste d'images défilant horizontalement

• GridView: Affiche des éléments sur m colonnes et n lignes avec défilement

• ListView: affiche une simple liste en gérant le défilement

• ScrollView: Affichage d'élément affiché en colonne avec défilement vertical

• Spinner: Affiche un objet contenant une ligne de texte, à un instant parmi une liste. Le défilement peut s'opérer horizontalement ou verticalement.

• SurfaceView: fournit un accès direct à la couche de dessin. Sera utilisé pour un affichage devant dessiner les pixels ou pour les widgets.

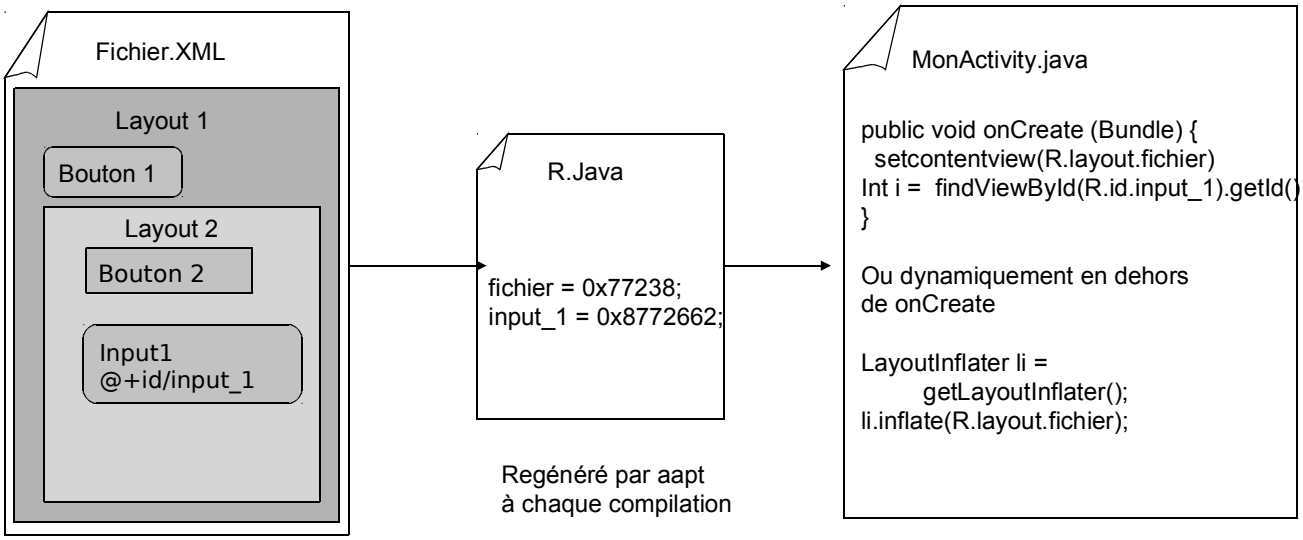
• TabHost: fournit une liste d'onglets qui gère les évènements utilisateurs afin de changer le contenu de l'écran.

• ViewFlipper: Fait défiler à l'écran les éléments d'une liste. Il peut contenir un timer tel qu'un diaporama

•ViewSwitcher: identique au ViewFlipper.

IHM : les layouts et les vues

Click to add an outline



TD1: créer une interface

- Objectif : créer une interface graphique
- Créer un projet TP1 (Android 2.3)
- Créer la même interface avec des layouts différents
  - Fichier main_linear.xml
  - Fichier main_relative.xml

Créer un projet TP1 basé sur Android 2.3

Vous définirez le nom du package com.[mon_organisation].tp1

Créer un fichier main_linear.xml contenant comme élément de plus haut niveau un LinearLayout

Jouer avec l'éditeur graphique et surtout le mode XML afin de reproduire l'interface défini ci-dessus .

Les noms et numéros de téléphones seront contenus dans deux TextView différents.

Au sein de votre projet, vous créer une classe Person contenant les informations suivantes

- id (int), name (string), et phone (string)

Votre classe principale activity contiendra une ArrayList<Person> que vous alimenterez avec les données de votre choix.

Vous créerez deux méthodes initList pour initier l'ArrayList et les données et une méthode setDataView() qui affichera chaque élément de la liste dans votre interface graphique.

Vous créerez ensuite un fichier main_relative.xml reprenant le contenu de main_linear.xml mais vous modifierez l'élément de plus haut niveau par un RelativeLayout.

IHM: string et internationalisation

- Les chaines de caractères et l'IHM
  - Fichier XML
  - /res/values
  - Par défaut: strings.xml
  - <string name=''cle''>Valeur à afficher</string>

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">Hello World, HelloActivity!
</string>
    <string name="app_name">Mon Hello Android</string>
</resources>
```

- Internationalisation
  - Basée sur la locale du système
  - /res/values-fr, /res/values-en contenant un fichier strings.xml
  - Utiliser les mêmes clés pour afficher la valeur dans la langue correspondante

eXpertise @ndroid

La notion de multilingues n'est pas encore complètement implémentée dans Android. Mais cette méthode est testée et approuvée.

Si l'application contient des images contenant du texte qu'il est nécessaire de localiser, il est nécessaire de créer le même type d'arborescence sur /res/drawable, res/drawable-fr-FR, res/drawable-fr-CA....qui contiendront chacun le fichier image personnalisé.

IHM: les menus

Menu Option

Sous menu

Menu contextuel

IHM: évènements

- Gérés comme en Java / AWT / Swing / J2ME
- Mise en place de Listener (écouteurs)
- Peuvent être appliqués sur chaque composant View
- Évènements supplémentaires en fonction du type de la vue
- Processus
  - Récupérer l'objet vue concerné, le caster au besoin
  - Ajout du listener
  - Implémentation des actions à réaliser par le listener

```
Button monBoutton = (Button)findViewById(R.id.monBouton);
monBoutton.setOnClickListener(evtClickSurMonBoutton);
…
…
OnClickListener evtClickSurMonBoutton = new
OnClickListener(){
    public void onClick(View v){
        // Implémentation
    }
};
```


• On retrouve un élément présent dans un layout en utilisant la méthode findViewById, prenant en paramètre l'identifiant de la vue (il est donc requis d'utiliser la balise android:id ou setId(valeur) dès que la conception montre qu'une interaction sera réalisée avec le composant)

• Bien qu'il soit possible de réaliser l'ajout d'un listener tel que

findViewById(monId).setOnClickListener(evt)

Il sera nécessaire de caster dans le type de vue retrouvé, afin de bénéficier d'évènements (et donc de listener) spécifiques au type de vue.

IHM: évènements

Evènement	Description
onClick	Touche et relâche un objet / Appuie sur la trackball alors que le focus est sur l'objet
onLongClick	Idem que onClick mais relâché au delà d'une seconde
onFocusChange	Prise de focus ou sortie de focus d'un objet
onKey	Appuie sur une touche alors que l'objet possède le focus
onTouch	Capture tous les événements de toucher sur l'objet

Nous avons vu sur la page précédente que la gestion de l'évènement se faisait par l'implémentation de la méthode – classe interne - au sein de votre activity de l'évènement à gérer. Si vous avez de nombreux évènements à gérer, utilisez de préférence l'implémentation du listener le plus utilisé afin d'éviter la création d'objet:

```
public class ExampleActivity extends Activity implements
OnClickListener {

protected void onCreate(Bundle savedValues) {
        ...
        Button button = (Button)findViewById(R.id.corky);
        button.setOnClickListener(this);
    }

    public void onClick(View v) {
      // le traitement à réaliser
    }
}
```

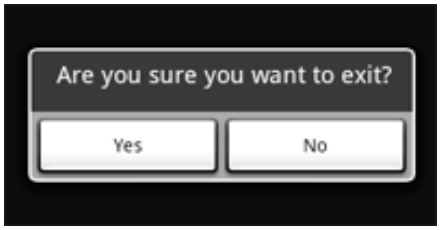
IHM: Alertes

- Toast: Un simple message


```
Toast.makeText(HelloActivity.this, "Hello
Android",
Toast.LENGTH_LONG)
.show();
```

IHM: Boite de dialogue

- AlertDialog
  - AlertDialogBuilder attaché à un contexte
  - Retourne un objet AlertDialog
  - Interface créée dynamiquement



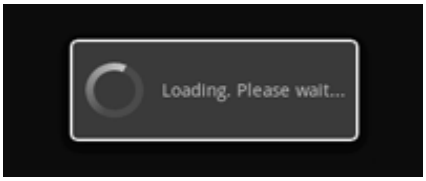

```
AlertDialog.Builder builder = new AlertDialog.Builder(this);
builder.setMessage("Are you sure you want to exit?")
       .setCancelable(false)// ne tient pas compte de BACK
       .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int id) {
                MyActivity.this.finish();
           }
       })
       .setNegativeButton("No", new DialogInterface.OnClickListener() {
           public void onClick(DialogInterface dialog, int id) {
                dialog.cancel();
           }
       });
AlertDialog alert = builder.create();
alert.show()
```

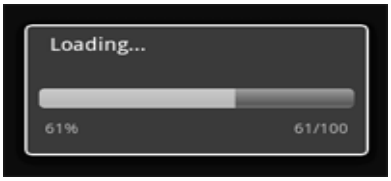

• Les labels – message, boutons - utilisés, pour plus de clarté et de compréhension du code, devront se trouver dans les fichiers de ressources string.xml.

•

IHM: Boite de dialogue

- ProgessDialog
  - Affiche la progression d'une tâche
  - Étend de AlertDialog
  - Cercle animé
  - Barre de progression
  - Fin de tâches connues ou non
- Utilisé couramment avec des Threads
- Possibilité d'ajouter un indicateur de progression dans la barre de titre Window.requestFeature





IHM: Boite de dialogue

- Personnaliser une boite de dialogue
  - Créer un layout en XML
  - Initié un objet de type Dialog ou AlertDialog
  - setContentView sur le dialog / setView sur une AlertDialog
  - On peut travailler sur la vue comme habituelleme

```
// Initie l'objet dialog
Dialog d = new Dialog(context);
View v =
getLayoutInflater().inflate(R.layout.monlayout) ;
d.setContentView(v);

// défini le titre
d.setTitle(getString(R.id.montitre));

// Pour chaque élément du layout
// on met une valeur
(TextView) texte = v.findViewById(R.id.text);
texte.setText(getString(R.id.mon_message));
(ImageView) img = v.findViewById(R.id.image);
img.setImageRessource(R.drawable.monimage);

d.show();
```

• La création du contenu peut également être réalisée en utilisant l'objet LayoutInflater. Dans notre exemple, le code serait le suivant (ceci peut être utile pour créer des alertes demandant l'entrée d'une valeur par l'utilisateur):

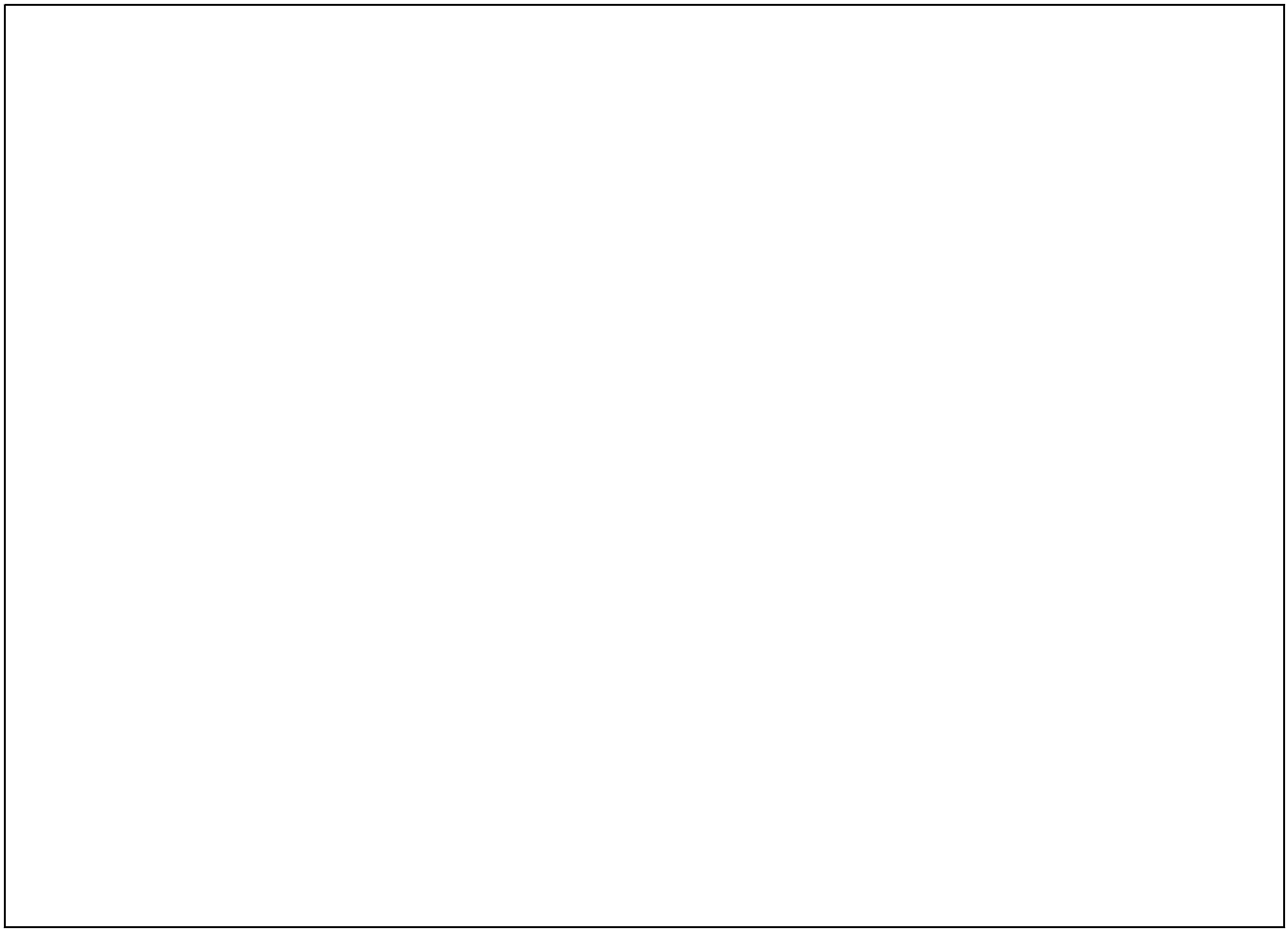
```
AlertDialog.Builder builder;
AlertDialog alertDialog;

LayoutInflater inflater = (LayoutInflater)
context.getSystemService(LAYOUT_INFLATER_SERVICE);
// utiliser getLayoutInflater dans une activity
View layout = inflater.inflate(R.layout.monlayout, null);
TextView text = (TextView) layout.findViewById(R.id.text);
text.setText(getString(R.string.custom_dialog_text));
ImageView image = (ImageView) layout.findViewById(R.id.image);
image.setImageResource(R.drawable.monimage);
builder = new AlertDialog.Builder(HelloActivity.this);
builder.setView(layout);
alertDialog = builder.create();
alertDialog.setTitle(R.string.mon_titre);
alertDialog.show();
```

eXpertise @ndroid

Les activity et leurs intéractions

Activity

- Une classe étendant de Activity
- Point d'entrée d'une application
- Généralement un écran d'une application (plein écran ou non)
- Comportement défini dans le fichier AndroidManifest.xml
- Une Activity peut
  - S’afficher en plein écran
  - S’afficher sous forme de popup
  - Retourner des valeurs
  - Lancer d'autres Activity
- Possède un cycle de vie



Activity

- Une Activity est capable de lancer une autre Activity
  - Interne à l'application
  - Externe à application
  - Récupérer le résultat d'une SubActivity
  - Notion importante d'Intent

eXpertise @ndroid

ActivityGroup: un écran qui contient une ou plusieurs instances d'objet Activity

ExpandableListActivity: Une Activity qui affiche une liste extensible d'objets dont les données pointent sur une source de données en implémentant un adapter de type ExpandableListAdapter.

ListActivity: Une activity qui affiche une liste d'objets pointant sur une source de données tel qu'un curseur ou un tableau

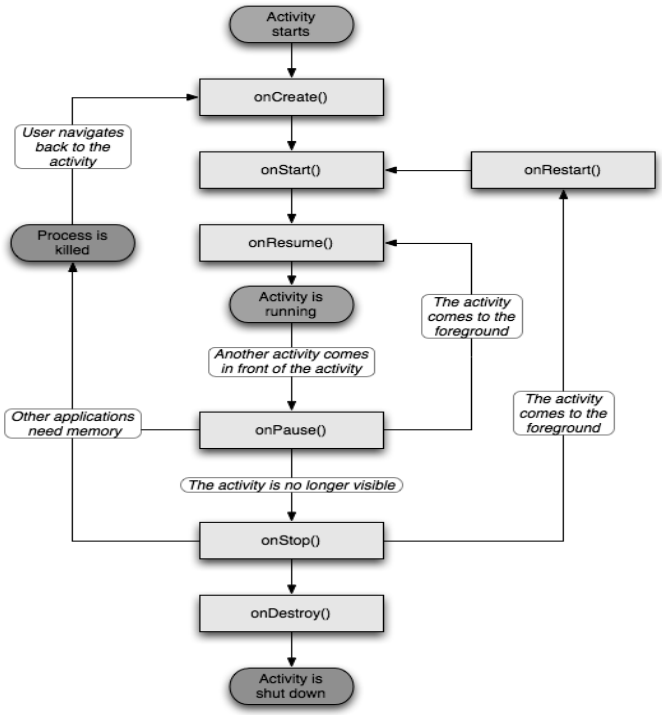
LauncherActivity: affiche une liste de tous les objets Activity pouvant répondre à un intent particulier

PreferenceActivity: Affiche une hiérarchie des objets de type Preference sous forme de liste.

TabActivity: Une Activity qui affiche et exécute de multiples activity ou vues embarquées

Activity

- Cycle de vie d'une Activity



onCreate: allocations des ressources

onStart: affichage écran

onResume: passage au premier plan

onPause: passage en arrière plan

onStop: arrêt de l'exécution

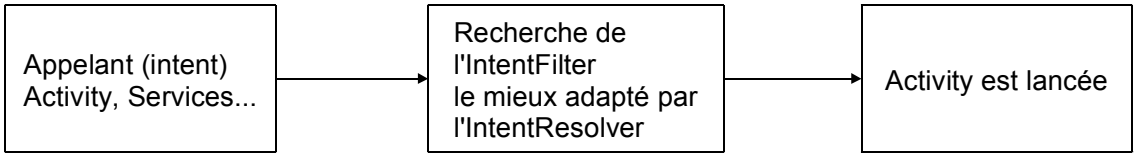
onDestroy: libération des ressources

Android conserve l'état des 6 dernières applications lancées (6 activity). Ainsi il est possible d'avoir 6 applications graphiques accessibles rapidement aux utilisateurs. Lorsqu'une 7 application est lancée, l'application la moins utilisée dans la file est alors supprimée supposant que l'utilisateur n'utilise plus cette application. Le fait d'utiliser la méthode finish() de l'activity, conserve quand même le processus dans un état endormi.

Afin de conserver l'état des éléments présents dans l'interface courant on implémentera la méthode onSaveInstanceState(Bundle b) ce qui permettra de retrouver les données lorsque la méthode onCreate sera appelée.

Intents

- Intent
  - Intention de faire quelque chose
  - Définir une action à effectuer pour certaines données
  - Décrit par une action (VIEW, EDIT, MAIN,...)
  - URI pour renseigner les données à traiter
  - Le système proposera les bonnes activity
- IntentFilter
  - Décrit les capacités d'une application
  - Lié à une Activity
  - Écoute les demandes d'Intent (handler) en provenance de la plateforme
  - Enregistré au niveau système



Activity

- Lancement d'une autre Activity
  - Définition du type d'Intent
  - Définition du chemin de l'Activity
  - startActivity appel explicite

```
startActivity(new Intent(CONTEXT, CLASSE ACTIVITY));
```

  - startActivity appel implicite

```
startActivity(new Intent(ACTION,URI));
```

  - StartActivityForResult
    - Lancer l'activity

```
startActivityForResult(INTENT, REQUEST_CODE);
// si 0 ne passera pas par onActivityResult
```

    - Récupérer le résultat

```
protected void onActivityResult(int requestCode, int resultCode,
            Intent intent) {
super.onActivityResult(requestCode, resultCode, intent);
}
```


La définition du chemin de l'Activity est à éviter lorsque vous souhaitez lancer une activity n'appartenant pas à votre application. Vous n'êtes pas au courant des évolutions des autres applications, qui pourrait rendre votre application non fonctionnelle.

L'action d'un Intent de type PICK_UP permet de lancer une Activity en mode « selection », notamment lorsque l'on donne la possibilité à l'utilisateur de pouvoir sélectionner un élément d'une liste. Ci-dessous, exemple d'un code ouvrant la liste des contacts pour que l'utilisateur puisse en sélectionner un:

```
Intent i=new Intent(Intent.ACTION_PICK,
      Uri.parse("content://contacts/people"));
startActivityForResult(i, ID_REQUEST);
```

Id_request est une valeur arbitraire définie par le développeur afin d'identifier l'Activity appelante par l'Activity appelée.

Ainsi il est possible d'appeler plusieurs sous activités, en passant des paramètres aux enfants mais en récupérant également le résultat de ces derniers.

Le passage de paramètres à un enfant se réalisera par l'ajout de données dans l'Intent (méthode putExtra(clé, valeur) qui permettra d'appeler l'enfant. Ce dernier récupérera les valeurs soumises en utilisant get[TYPE]Extra(clé):

```
public void onCreate(Bundle b){
....
Intent i = getIntent();
        if (i!=null){
      i.getIntExtra("idenfiant_cle_entier");
      i.getStringExtra("identifiant_cle_string");
        }
}
```

Descripteur d'application

- 1 fichier de description de l'application: AndroidManifest.xml
- Descripteur de l'application
  - Renseignements sur l'application (nom, version de code, SDK supporté...)
  - Définit le comportement des Activity et leurs actions
  - Déclare auprès du système les Intent réalisables par cette appliation
  - Définit les services, Broadcast receiver et content provider
- Déclaration des ressources nécessaires (applications, Internet, matériels, données...)
  - L'utilisateur est toujours prévenu à l'installation des ressources requises pour le bon fonctionnement de l'application. Il peut accepter ou refuser.

L'ensemble des paramètres utilisables sont décrit sur

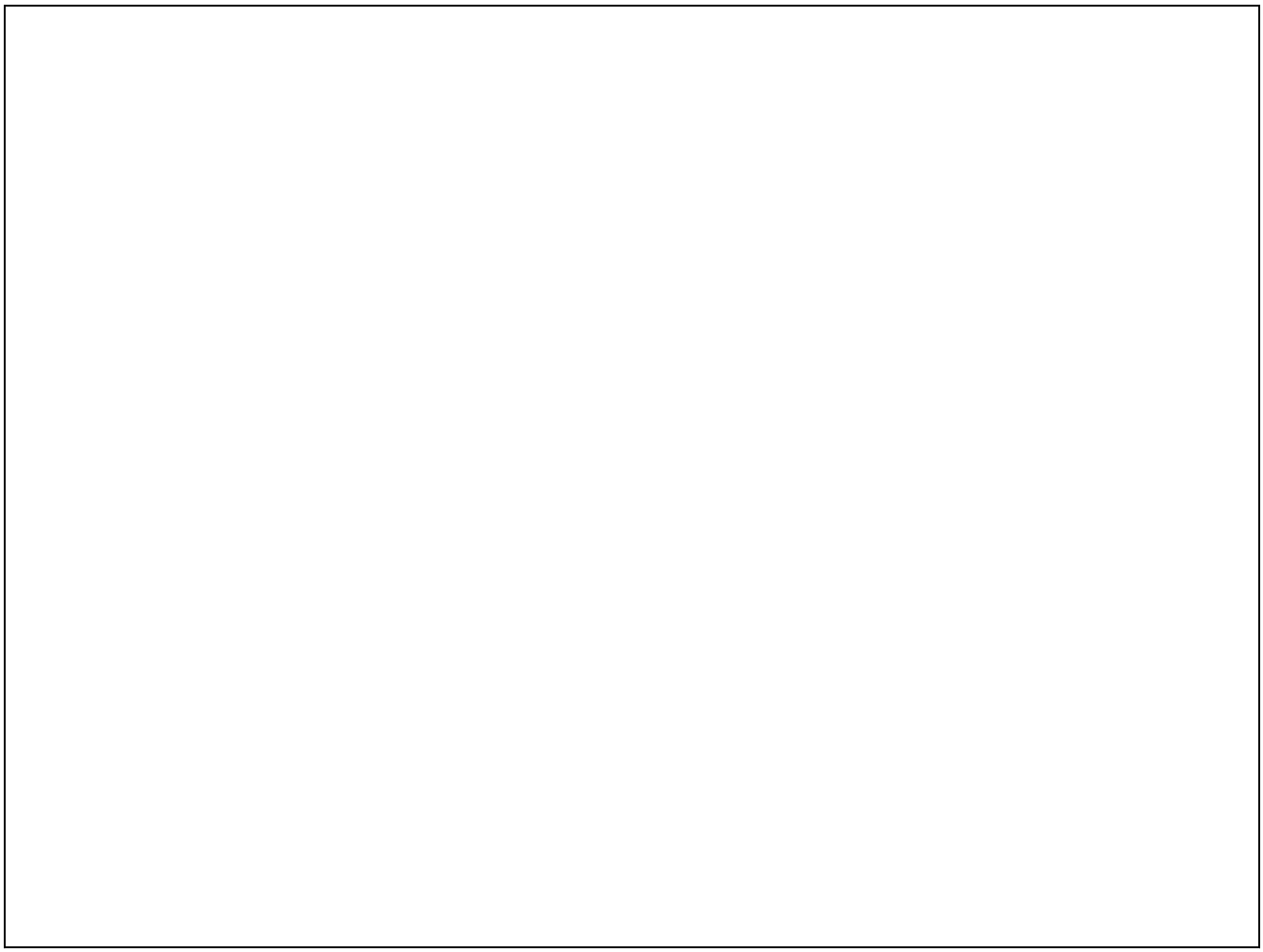
http://developer.android.com/guide/topics/manifest/manifest-intro.html

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
   <uses-permission />
   <permission />
   <permission-tree />
   <permission-group />
   <uses-sdk />
   <application>
      <activity>
         <intent-filter>
            <action />
            <category />
            <data />
         </intent-filter>
         <meta-data />
      </activity>
      <activity-alias>
         <intent-filter> . . . </intent-filter>
         <meta-data />
      </activity-alias>
.....
</manifest>
```

```
<service>
        <intent-filter> . . . </intent-filter>
        <meta-data/>
</service>
 <receiver>
        <intent-filter> . . . </intent-filter>
        <meta-data />
 </receiver>
 <provider>
        <grant-uri-permission />
        <meta-data />
 </provider>
 <uses-library />
 <uses-configuration />
 </application>
```

Accès aux fichiers

- Par défaut 3 accès potentiels
  - Répertoire /res
    - Classe R
  - Répertoire /assets
    - Classe AssetManager via getAssets()
  - /sdcard
- Une application ne peut accéder qu'à ses propres ressources ressources
- L'accès aux fichiers d'une autre application
  - Utilisation des Content Provider
  - Accès via URI
- L'écriture de fichier respecte les mêmes restrictions
  - /data/data/package_de_mon_application/files
  - Ne se fera que dans le répertoire files ou SD

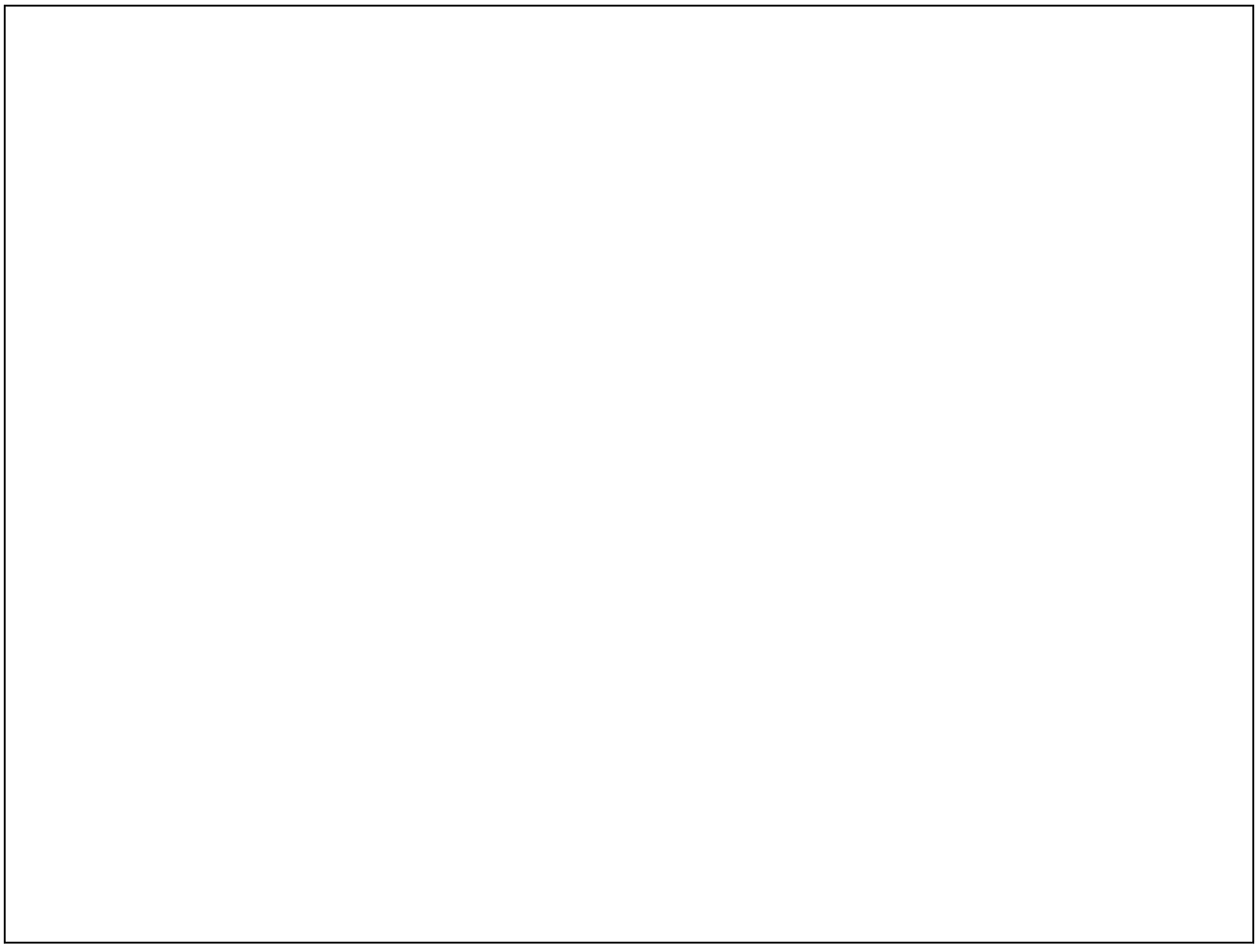


eXpertise @ndroid

Les données persistantes

Les données persistantes

- 3 types de données persistantes:
  - Préférences utilisateurs
  - Base de données (Sqlite3)
  - Content Provider
- Gestion de vos données dans des fichiers



Les préférences utilisateurs

- Composants fournis
  - IHM
  - Données
- Mise en place facile : un fichier XML + une classe PreferenceActivity
- Fichier de préférence clés(String)/valeurs(primitive)
- SharedPreferences
  - Setters
  - remove() pour supprimer un couple clé/valeur
  - clear() pour supprimer toutes les préférences définies pour l'activity ou l'application
  - commit() valide les modifications sur les préférences
- Déclarations dans AndroidManifest
- Données sauvegardées dans /data/data/monpackage/shared_prefs/monpackage_preferences.xml

Le fichier xml devra être créé dans /res/xml.

L'élément root sera de type PreferenceScreen. Le fichier peut contenir plusieurs PreferenceScreen qui ouvriront des Activity différentes.

Type des éléments disponibles dans un PreferenceScreen

- CheckBoxPreference: sauvegarde un boolean

- EditTextPreference: sauvegarde une chaine de caractère libre

- ListPreference: sauvegarde une chaine issue d'une tableau de chaine. Vous avez la possibilité de distinguer les valeurs affichées à l'écran de celles qui seront sauvegardées dans le fichier de préférence par les attributs entries et entryValues

- PreferenceCategory: permet de mettre un titre aux préférences qui se trouveront en dessous

- RingtonePreference: permet de (dé)sélectionner une sonnerie parmi celles disponibles

En utilisant le code Java, il est possible d'activer ou non certaines parties de l'écran de préférences.

Chaque objet préférence devra disposer d'un clé qui sera utilisée dans le fichier pour associer la valeur (android:key)

Par défaut les préférences sont sauvegardées sous forme de fichier xml situé dans /data/data/monpackage/shared_refs/monpackage_preferences.xml.

```
SharedPreferences sp = getSharedPreferences("com.mycompany.hello_preferences", MODE_WORLD_READABLE);
```

L'objet SharedPreferences vous permettra de retrouver l'ensemble des valeurs sauvegardées sous forme d'une MAP ou obtenir une valeur spécifique en fonction du nom de la clé utilisée lors de la sauvegarde.

Les préférences utilisateurs

- Fichier XML
  - PreferenceScreen
  - Composant EditTextPreference, RingtonePreference, CheckboxPreferences,ListPreferences, PreferenceCategory
- Attribut obligatoire sur les composants
  - key
  - title
- PreferenceActivity
  - onCreate
  - super.onCreate(bundle)
  - addPreferenceFromResource (R.xml.fichier_ressource)

L'accès à un fichier de préférence peut être fait depuis n'importe quelle classe pouvant accéder au contexte de l'application par l'objet SharedPreference.
On accédera aux différentes valeurs par l'appel aux méthodes de sharedPreference.get[TYPE](CLE_ACCES, VALEUR_PAR_DEFAUT).

Exemple fichier de préférences

```
<?xml version="1.0" encoding="utf-8"?>
<PreferenceScreen
        xmlns:android="http://schemas.android.com/apk/res/android">
        <PreferenceCategory android:title="@string/prefsFirstGroup">
        <CheckBoxPreference
        android:key="@string/prefsKeyCheck"
        android:title="@string/prefsTitleCheck"
        android:summary="@string/prefsSummaryCheck" />
        <ListPreference
        android:key="@string/prefsKeyList"
        android:title="@string/prefsTitleList"
        android:summary="@string/prefsSummaryList"
        android:entries="@array/prefuserlabel"
        android:entryValues="@array/prefuservalue" />
        </PreferenceCategory>
        <PreferenceCategory android:title="@string/prefsSecondGroup">
        <RingtonePreference
        android:key="@string/prefsKeyRing"
        android:title="@string/prefsTitleRing"
        android:summary="@string/prefsSummaryRing"
        android:ringtoneType="notification"
        android:showDefault="true"
        android:showSilent="true" />
        <EditTextPreference
        android:key="@string/prefsKeyEditText"
        android:title="@string/prefsTitleEditText"
        android:hint="@string/prefsSummaryEditText" />
        </PreferenceCategory>
</PreferenceScreen>
```

Base de données

- SQLITE
  - Projet Open Source
  - Base de données transactionnelles
  - Pas de partie serveur
  - Toutes les tables sont contenues dans un seul fichier
  - Crossplatform
  - Données typées
  - Langage SQL
  - Clé primaire
  - Non disponible: FOREIGN_KEY, RIGHT/LEFT_OUTER_JOIN, certaines options ALTER_TABLE
- SQLITE3
  - Librairie de manipulation
  - Accessible par adb shell
- Type de données supportées: NULL, INTEGER, REAL, TEXT, BLOB


```
ALTER TABLE
ANALYZE
ATTACH DATABASE
BEGIN TRANSACTION
comment
COMMIT TRANSACTION
core functions
CREATE INDEX
CREATE TABLE
CREATE TRIGGER
CREATE VIEW
CREATE VIRTUAL TABLE
date and time functions
DELETE
DETACH DATABASE
DROP INDEX
DROP TABLE
DROP TRIGGER
DROP VIEW
END TRANSACTION
EXPLAIN
expression
INDEXED BY
INSERT
keywords
ON CONFLICT clause
PRAGMA
REINDEX
RELEASE SAVEPOINT
REPLACE
ROLLBACK TRANSACTION
SAVEPOINT
SELECT
UPDATE
VACUUM
```

Base de données

- SQLITE & Android
  - Aucune base initiée lors de la création d'une application
  - Possibilité de créer une à x bases de données pour la même application
  - Manipulation de l'API SQLiteDatabase
  - SQLiteOpenHelper
    - onCreate
    - onUpgrade
  - La base de données est fermée lorsqu'on quitte l'application (appel à close() est quand même préférable)
- Le fichier est sauvegardé par défaut dans /data/data/monpackage/databases
  - L'enregistrement de fichier de base de données sur une carte mémoire ne pourra être réalisé par SQLiteOpenHelper, recours à l'objet SQLiteDatabase
- Problème potentiel d'espace « disque »
  - Aucune création sur la carte mémoire possible (même version 2.2)

Base de données

- La bonne méthode de création
  - Créer une classe définissant les variables suivantes
  - Nom de la base et sa version
  - Le nom des tables
  - Le nom des colonnes de la table
    - Prévoir une colonne _id
  - Requêtes de création des tables
  - Créer une classe interne
  - Etend de SQLiteOpenHelper
    - Constructeur, onCreate, onUpgrade
  - Ajouter dans la classe principale
  - Une variable conservant le contexte
  - Une variable pour conserver l'instance du helper
  - Une variable de type SQLiteDatabase, pour gérer les données
- Possibilité de création de la base et de l'alimenter sur PC

ETAPE 1

```
public class DBAdapter {
    public static final String
      KEY_ROWID = "_id";
    public static final String
      KEY_ISBN = "isbn";
    public static final String
      KEY_TITLE = "title";
    private static final String
      DATABASE_NAME = "books";
    private static final String
      DATABASE_TABLE = "titles";
    private static final int
      DATABASE_VERSION = 1;

    private static final String
DATABASE_CREATE =
        "create table titles (_id
integer primary key
autoincrement, "
            + "isbn text not null,
title text not null);";
    private final Context context;

    private SQLiteDatabase db;
}
```

ETAPE 2

```
private static class DatabaseHelper extends
       SQLiteOpenHelper   {
        DatabaseHelper(Context context) {
            super(context, DATABASE_NAME,
      null, DATABASE_VERSION);
        }
        @Override
        public void onCreate(
      SQLiteDatabase db) {
            db.execSQL(DATABASE_CREATE);
        }
        @Override
        public void onUpgrade(
      SQLiteDatabase db,
       int oldVersion,
      int newVersion) {
            db.execSQL("DROP TABLE "
      + "IF EXISTS titles");
            onCreate(db);
        }
    }
```

ETAPE 3

Dans DBAdapter on ajoute une variable faisant référence à un objet DatabaseHelper. On complète le constructeur de l'adapter pour y inclure la création d'un objet DatabaseHelper. Ajouter deux méthodes dans l'adapter qui permettra d'ouvrir et de fermer une connexion en utilisant l'objet DatabaseHelper initialisé et en y appliquant la méthode getWritableDatabase() ou getReadableDatabase() selon les besoins de l'accès.

- Manipulation des données
  - execSQL: exécution de commande SQL passée en paramètre sous forme de chaine de caractères (dans la classe de l'adapter)

```
db.execSQL("insert into " + DATABASE_TABLE_USER
+ " (" + COL_TAB_HELLO_USER_NOM
+ ") values('" + name + "');");
```

  - Insert(), delete() et update() de SQLiteDatabase
    - Nom de la table concernée
    - ContentValues().put(nom de colonne, valuer)

```
ContentValues cv = new ContentValues();
cv.put(COL_TAB_HELLO_USER_NOM, name);
db.insert(DATABASE_TABLE_USER, COL_TAB_HELLO_USER_ID, cv);
```

    - Permet de retourner des valeurs (ex.: identifiant d'un nouveau élément ajout)

Le deuxième argument de la méthode insert doit contenir le nom d'une colonne de la table qui devra être mise à Null (et non insérer) si le ContentValue est vide.

Syntaxe insertion d'un enregistrement

insert(nom de la table, nom de la colonne null, collection de valeurs)

update(nom de la table, collection de valeurs, chaine de caractère représentant les conditions)

delete(nom de la table, chaine de caractère représentant les conditions)

Base de données

- Requête de sélection
  - rawQuery() traite directement une commande SELECT
  - query() construit une requête SELECT à partir d'arguments
  - Retourne un Cursor
- rawQuery
  - Création de la commande SQL « select * from table »
  - Les arguments de recherche utiliseront des « ? »
    - where nom=? And prenom=?
  - La valeur des arguments sera utilisée par un tableau de chaines (ordonné!)

```
String rawQueryStr = "Select * from " + DATABASE_TABLE_USER
    + " where " +
    COL_TAB_HELLO_USER_NOM + "=?";
String[] strArgs = new String[]{nomRecherche};
Cursor cursor = db.rawQuery(rawQueryStr, strArgs);
```

Base de données

- query

```
public Cursor query (String table,// nom de la table
String[] Columns, // nom des colonnes a retourner ou null pour toute
String Selection, // clause where, sans le mot 'where' ex.: nom=?
String[] SelectionArgs, // tableau ordonné des valeurs
                // utilisées dans la clause where
String GroupBy, // argument groupBy, ex: nom, prenom - null sinon
String Having,  // argument clause having - null sinon
String OrderBy) // argument d'ordonnancement ex: nom - null sinon
```

```
public void getInfo(String nomRecherche){
    String[] columns=new String[]{"nom", "prenom"};
    String[] params={nomRecherche};
    Cursor result=db.query(DATABASE_TABLE_USER,
    columns, "nom=?",
             params,
    null, null, null);
…
}
```


Utilisation d'un objet Cursor

```
if (cursor != null) {
        cursor.moveToFirst();
        while (!cursor.isAfterLast()) {
                // traitement
                cursor.moveToNext();
        }
        cursor.close();
}
```

ContentProvider

- Accès uniquement par URI
- Encapsulation de la structure des données
- Base de données, fichiers plats, accès distant
- A partir d'une URI
  - Create
  - Read
  - Update
  - Delete
- Utilisation de ContentProvider existant – d'autres applications
- Créer, utiliser et partager vos données
- Définition d'une URI
- getContentResolver fournit l'accès au ContentProvider

PREFIXE://IDENTIFIANT DU TYPE DE DONNEES/DEFINITION DE LA DONNEES/ENREGISTREMENT

```
content://contact/people/123
content://com.mycompany.hello/user/53
```

eXpertise @android

Dernière brique majeure de la plateforme Android.

L'accès à certains ContentProvider, notamment ceux fournis en standard sur la plateforme Android, demanderont l'ajout dans le fichier AndroidManifest.xml les droits d'accès en lecture et ou écriture.

Exemple

```
<uses-permission android:name="android.permission.READ_CONTACTS">
<uses-permission android:name="android.permission.WRITE_CONTACTS">
```

ContentProvider

- Réaliser une requête
  - managedQuery() depuis un objet Activity ou getContentResolver().query
    - URI
    - Un tableau des propriétés du ContentProvider à obtenir dans le résultat (nommé projection)
    - Les contraintes (clause Where)
    - Ensemble des paramètres permettant de compléter les contraintes (? dans la clause where)
    - Clause d'ordonnancement
  - Retourne un Cursor
- Propriété d'un ContentProvider = Colonnes d'une base de données

```
private static final String[] PROJECTION = new String[] {
    Provider.Constants._ID, Provider.Constants.TITLE,
    Provider.Constants.VALUE};
....
constantsCursor=managedQuery(Provider.Constants.CONTENT_URI,
                             PROJECTION, null, null, null)
```


Le nom des propriétés d'un ContentProvider doit être fourni par le ContentProvider et présent dans la Javadoc. Ceci rend très facile la construction d'une requête sur des informations que le développeur ne maitrise pas forcément, notamment lors de l'accès à des données externes à son application.

Exemple d'URI lié à la base de données de contacts (accessible dans la Javadoc Android)

`People.CONTENT_URI`

`Contacts.ContactMethods.CONTENT_EMAIL_URI`

`Contacts.Phones.CONTENT_URI`

La dernière version Android propose de synchroniser ses données avec N système de synchronization. Le fonctionnement des providers reste présent, cependant de nouvelles méthodes ont été ajoutées pour créer des contacts à synchroniser ou non, et ce manière manuel ou individuel.

Exemplen de définition de colonne depuis un ContentProvider

`Contacts.Phones.NUMBER`

`Contacts.Phones.TYPE`

`People._ID`

`People.NAME`

Propriété par défaut: _ID, _COUNT

ContentProvider

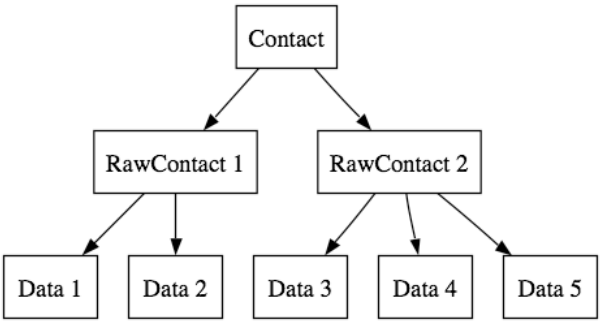
- Insérer des enregistrements
  - insert()
    - Uri + contentValues
    - Retourne l'identifiant de l'élément inséré
  - bulkInsert()
    - Tableau d'Uri et tableau de ContentValues pour ajouter plusieurs enregistrements en une seule fois
    - Retourne le nombre d'éléments insérés
- Suppression d'un enregistrement
  - Delete
    - Uri, clause where, arguments complétant la clause where
    - Les informations dépendantes à la table visée seront également supprimées (cascading)
    - Retourne le nombre d'éléments supprimés
- Mise à jour
  - Update
    - Uri, valeurs à insérer, clause where, arguments complétant la clause where
    - Retourne le nombre d'éléments mis à jour

eXpertise @ndroid

Après l'ajout d'un enregistrement nous recommanderons d'utiliser la méthode requery() de l'objet Cursor afin que ce dernier soit mis à jour pour afficher des informations correctes dans l'IHM.

ContentProvider Contacts

http://developer.android.com/reference/android/provider/ContactsContract.Contacts.html



Récupérer l'identifiant de plus haut niveau
Parcours les différents content providers en cherchant dans la colonne CONTACT_ID

Les adapters

- Présentation dynamique
- Mapping entre des données et un affichage
- Un ensemble de données pour une partie de l'affichage
  - ListView: liste simple de texte peut être étendu pour afficher une image
  - TabView: défini des onglets mais le contenu dépendra de l'onglet choisi
  - Spinner: afficher une liste déroulante
  - ....
- Implémentation d'un ViewGroup en fonction d'un Adapter
  - SimpleAdapter: utilisé pour les données contenues dans les fichiers de ressources
  - CursorAdapter: généralement utilisé pour traiter les données retournées par un Content Provider
  - ArrayAdapter: mapping vers une liste

eXpertise @ndroid

La création d'un adapter prend 3 paramètres: le contexte de l'application, un layout qui correspondra à celui devant être utilisé pour chaque élément contenu dans la liste des données contenu en troisième paramètre

```
String[] items={"ceci", "est", "un", "exemple", "d'adapter"};
new ArrayAdapter<String>(this,
  android.R.layout.simple_list_item_1, items);
```

Par défaut, l'objet ArrayAdapter ne traite que des chaines de caractères. Le layout fournit par Android pour afficher une simple ligne de texte est android.R.layout.simple_list_item.

Pour afficher chaque élément, la méthode getView de la classe adapter est exploitée. Vous avez la possibilité (et la nécessité) de surcharger cette classe si vous souhaitez créer votre propre adapter

```
public View getView(int position, View convertView,ViewGroup parent) {
  if (convertView==null) {
    convertView=new TextView(this);
  }
  convertView.setText(items(position));
  return(convertView);
}
```

Cursor > Adapter > IHM

- Composant dédié à l'affichage de liste
  - Gestion du Cursor
  - Mise à jour automatique lors d'un ajout, modification, suppression
  - Personnalisation de l'affichage possible
- Processus
  - Utilisation d'un composant gérant un adapter (spinner, listview…)
  - Récupération d'un Cursor
  - Création de l'adapter
  - Affichage
- En cas de personnalisation de l'affichage, nécessité de créer son propre Adapter

Cursor > Adapter > IHM

- Exemple simple: ListActivity
  - Une Activity devient une ListActivity
  - Obligation de présence d'un objet ListView dans le layout
    - Id spécifique: `android:id="@android:id/list"`
  - SimpleAdapterCursor
  - Application de l'adapter sur la vue courante

```
Cursor c = managedQuery(uri, PROJECTION,
        null, null, null);
startManagingCursor(c);
ListAdapter adapter = new SimpleCursorAdapter(
        this,
         android.R.layout.two_line_list_item,
         c,
         PROJECTION,
         new int[]{android.R.id.empty,
            android.R.id.text1,
            android.R.id.text2}
        );
 setListAdapter(adapter);
```


La méthode onListItemClick de ListActivity permet de savoir quel élément a été sélectionné.

Il sera alors possible d'ouvrir une Activity pour afficher le contenu global de l'objet sélectionné (ou de le modifier).

Le contenu du Cursor sera mis à jour automatiquement (startManagingCursor).

eXpertise @ndroid

Services et Alarme

Introduction

Services

- Services
  - Process actif sur le long terme
  - Lancer au redémarrage
  - Implémentation simple
    - Étend de Services
    - OnCreate
    - OnStart
    - OnDestroy
    - onBind
    - Déclarations dans l'Android Manifest
  - Préférer une Alarme plutôt qu'un service pour des actions espacées dans le temps

Les services doivent être créer uniquement si une tâche doit tout le temps être disponible.

```
@Override
public void onCreate() {
  super.onCreate();
  client=new DefaultHttpClient();
  format=getString(R.string.url);

myLocationManager=(LocationManager)getSystemService(Context.LOCATIO
N_SERVICE);
myLocationManager.requestLocationUpdates("gps", 10000,
10000.0f,onLocationChange);
}

@Override
public void onDestroy() {
  super.onDestroy();
  myLocationManager.removeUpdates(onLocationChange);
}
```

Alarmes et notifications

- Alarmes
  - Différents des services
  - Rappel à un instant T
  - Appel d'une action, un évènement à réaliser
- AlarmManager
  - PendingIntent
  - Enregistrée au niveau du système
  - Annulée si le système est redémarré


```
// Définition de la classe qui sera appelée lorsque l'alarme sera déclenchée
Intent intent = new Intent(this, MonAlarmReceiver.class);
// Definition du PendingIntent
PendingIntent pending = PendingIntent.getBroadcast(context, REQUEST_CODE, intent, 0);
// Définition de l'alarme
AlarmManager alarmMgr = (AlarmManager)getSystemService(ALARM_SERVICE);
alarmMgr.set(AlarmManager.RTC_WAKEUP, System.currentTimeMillis() + (5*1000),
pending);
```

Alarmes et notifications

- Notifications
  - Toutes les alertes émanant du système ou d'autres applications
  - Affichage d'une icône et/ou texte dans la barre de notifications + action à réaliser
  - Coloration de la led du téléphone, vibration
- NotificationManager
  - Hardware (son, led, vibreur)
  - Icons / Texte
- Obtenir une instance du NotificationManager
- Création de l'objet Notification (icon, message...)
- Création d'un PendingIntent pour la finalité de la notification (ouverture d'une activity)
- Attribuer le pendingIntent à la notification
- Envoyer la notification

eXpertise @ndroid

Les icônes et textes de la barre de notifications ne sont pas visibles lorsqu'une application s'exécute en plein écran.

Il est également possible d'ajouter un compteur de notification. Pour cela on affectera la valeur du compteur à la notification avant de passer la notification.

La mise en place d'évènement led ou vibreur n'aura aucun effet en mode émulateur. Il est également très dépendant du hardware. L'implémentation d'un modification de la couleur de led n'est donc pas recommandée pour une application qui devra être installé sur différents matériels.

Broadcast Receiver

- Un écouteur sur les événements systèmes
  - Logiciel
  - Matériel
  - Vos propres Intent
- Peut être défini dans
  - AndroidManifest.xml
  - Code Java
- Le plus de la déclaration dans AndroidManifest
  - IHM non nécessaire, démarré dès l'installation
- Création d'une classe
  - étend BroadcastReceiver
  - Surcharge onReceive pour définir les actions
    - Action simple (affichage d'un message, d'une notification…)
    - Lancement d'Activity

Déclaration dans le fichier XML

```
<receiver android:name=".ObjetBroadcastReceiver" android:enabled="true">
    <intent-filter>
        <action android:name="Type.evenement.ecouté" />
    </intent-filter>
</receiver>
```

Déclaration dans le code

```
ObjetBroadcastReceiver intentReceiver = new ObjetBroadcastReceiver();
IntentFilter intentFilter = new IntentFilter("type.evenement.ecouté");
registerReceiver(intentReceiver, intentFilter);
```

Les types d'évènements correspondent à des objets de type Broadcast.

Afin de lancer une activity depuis un BroadcastReceiver il sera nécessaire d'utiliser une instruction supplémentaire sur l'Intent à exécuter:

```
Intent monIntent = new Intent(context, MonActivity.class);
monIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

eXpertise @ndroid

Accès distants et Multithreading

Accès distants

- Librairies java.net et org.apache.http
- HTTP/HTTPS
- Ajout de librairies externes au besoin : FTP, SOAP, SMTP...
- Permission android.permission.INTERNET
- Fonctions standards Java
- HTTP POST/GET
  - HttpClient
  - Renseigner les informations sur la requête
  - execute()
- Requête GET

```
DefaultHttpClient httpclient = new DefaultHttpClient();
HttpGet httpget = new HttpGet(url);
try{
   HttpResponse response = httpclient.execute(httpget);
}
catch (ClientProtocolException cpe) {
    Log.e(TAG, "ClientProtocolException retrieveInfo:" + cpe);
}
```


Dans le cadre des développements mobiles on préférera l'utilisation de librairies JSON : plus légère dans la transmission d'information et plus facile à manipuler grâce aux librairies fournies.

De nombreuses librairies sont fournies, notamment par la fondation Apache, pour les accès distants, encodages...Bien qu'il soit possible des jars dans les projets Android, les classes appelées par ces jars ne sont pas forcément disponible dans les classes Java de base embarqué dans Android. Vous serez dans l'obligation de tester votre application afin de savoir si la librairie se charge correctement. Il est parfois possible qu'une librairie fonctionne dans une version et pas dans un autre. Vous devrez donc descendre ou monter en version sur la librairie souhaitée.

Utilité des Threads

- Blocage de l'application lors des accès distants (Wake Lock)
  - Exemple précédent, ouverture de la boite de dialogue
- Solution (recommandation!)
  - Utilisation de Thread pour les traitements bloquant: accès distant, calculs...
  - Utilisation de AsyncTask
- Création d'une classe de type AsyncTask qui implémentera :
  - Méthode doInBackground (params) : réalise le traitement et retourne (ou non) une valeur
  - Méthode onPostExecute(params_result) : appeler lorsque la méthode doInBackground est terminée
  - Méthode onProgress : permet de gérer l'état d'avancement du traitement

eXpertise @ndroid

Nous avons aussi la possibilité la classe Thread Java. Or, il est a noté qu'un thread ne peut interagir avec un autre thread, en l'occurrence celui qui permet d'afficher votre application.

Vous devrez alors mettre en place un système de Message et de Handler. Lorsque la méthode run() de l'objet runnable aura terminé son traitement, vous passerez un message à un Handler, qui aura connaissance du contexte (Activity) et qui aura donc accès à l'ensemble de l'interface graphique (pour rafraîchir des données par exemple).

- Classe implémentant l'interface Runnable
  - Constructeur prenant en paramètre l'Activity qui appellera le Thread
  - Surcharge de la méthode run() dans laquelle
    - Les traitements sont exécutés
    - Les messages sont envoyés au Handler
  - Création d'un objet Handler
    - Gestion des codes reçus
    - Mise à jour de l'IHM de l'Activity appelante
- Une Thread ne peut être tuée, le système s'en charge
  - Gestion d'un état permettant ou non d'appeler ou pas le contenu de run()

eXpertise @ndroid

Autres composants API

Autres composants

- Gestion des appels entrants et sortants
- Envoi/réception SMS
- Connexion GSM/Bluetooth/Wifi/GPS
- SearchManager (moteur de recherche intégré à vos contenus)
- Multimedia
  - Son
  - Caméra
  - 2D/3D
- Géolocalisation
  - Localisation
  - Map/MapView
  - Accéléromètre

Gestion des Appels

- Gestion des appels entrants et sortants
  - Manipulation de téléphone comme tout autre composant
  - TelephonyManager
    - Ensemble d'information sur l'état des connexions, des appels..
    - getSystemService(TELEPHONY_SERVICE)
- Appeler
  - Uri tel://xxxxxxx
  - StartActivity via un Intent ACTION_CALL
  - Permission: *android.permission.CALL_PHONE*
- Recevoir un appel
  - Récupération du TelephonyManager
  - Événement PhoneStateListener
  - Permission: *android.permission.READ_PHONE_STATE*

TelephonyManager est la partie de l'API à utiliser pour récupérer toutes les données (Sim, opérateur, device...) de la partie téléphonie (getPhoneState, getNetworkType, getOperateur, getCountry...) mais également interagir avec cette dernière (émission, réception d'appel).

Elle permet également de numéroter ou d'appeler.

L'historique des appels (CallLogs) est disponible dans les ContentProviders des Contacts.

D'une manière générale on appellera getSystemService présent dans le Context afin de pouvoir accéder à toutes les informations systèmes et matériels: AudioManager, NotificationManager, TelephonyManager, Connectivitymanager, PowerManager, SensorManager, WifiManager, Vibrator...

SMS

- Envoyer un SMS
  - Permission:
    <uses-permission android:name="android.permission.SEND_SMS">
  - SMSManager
  - PendingIntent

```
PendingIntent pi = PendingIntent.getActivity(this, 0,
            new Intent(this, SMS.class), 0);
SmsManager sms = SmsManager.getDefault();
sms.sendTextMessage(numeroTel, null, message, pi, null);
```

- Recevoir un SMS
  - Permission:
    <uses-permission android:name="android.permission.RECEIVE_SMS">
  - Receiver
  - Broadcast

Le PendingIntent est un Intent permettant d'invoquer un service à un certain moment. Il est également possible de gérer les accusés réception:

```
String SENT = "SMS_SENT";
String DELIVERED = "SMS_DELIVERED";
PendingIntent sentPI = PendingIntent.getBroadcast(this, 0,new Intent(SENT), 0);
PendingIntent deliveredPI = PendingIntent.getBroadcast(this, 0,new Intent(DELIVERED), 0);
registerReceiver(new BroadcastReceiver(){  // Gère l'envoi du SMS
   @Override
   public void onReceive(Context arg0, Intent arg1) {
      switch (getResultCode()){
         case Activity.RESULT_OK:.....
         case SmsManager.RESULT_ERROR_GENERIC_FAILURE:.....
         case SmsManager.RESULT_ERROR_NO_SERVICE:.....
    .............
 }, new IntentFilter(SENT));
registerReceiver(new BroadcastReceiver(){ //---when the SMS has been delivered---
   @Override
   public void onReceive(Context arg0, Intent arg1) {
      switch (getResultCode())  {
         case Activity.RESULT_OK:
            Toast.makeText(getBaseContext(), "SMS delivered",
                  Toast.LENGTH_SHORT).show();.....
}, new IntentFilter(DELIVERED));
```

Connexion réseau

- Service de connexion réseau
  - ConnectivityManager
  - getSystemService(CONNECTIVITY_SERVICE)
  - NetworkInfo
    - Type de connexion réseau active
  - NetworkInfo.State
    - Enum: Les états possibles des connexions
  - NetworkInfo.DetailedState
    - Enum: activités courantes des connexions
  - Permission: *android.permission.ACCESS_NETWORK_STATE*
- Wifi
  - WifiManager: gestion des accès Wifi
  - Permission : *android.permission.ACCESS_WIFI_STATE*
- Bluetooth (Level 7)
  - android.blutooth (recherche matériel, connexion, échange de données – sockets)
  - Permission : *android.permission.ACCESS_BLUETOOTH*
  - Permission : *android.permission.ACCESS_BLUETOOTH_ADMIN*

eXpertise @ndroid

La méthode getAllnetworkInfo du ConnectvityManager permettra de retourner toutes les connexions possibles par l'appareil (connecté ou nom) pour les connexions réseaux.

Médias

- Image
  - Support PNG, BMP, JPG, GIF
  - Enregistrement de fichiers image
  - Traitements
- 2D/3D
  - Librairie OpenGL ES / Khronos Group
  - www.kronos.org/opengles/spec.html
  - SurfaceManager / SurfaceHolder
  - Optimisation avec Android 2.0 et stratégie de Google

Les manipulations d'images (simples ou complexes) seront traitées dans des Thread afin d'éviter tout blocage de l'application.

Attention: la sauvegarde de fichier ne peut être réalisée qu'à deux endroits: le répertoire de l'application ou la carte mémoire.

OmiGSoft : Snow Rally Canada

Polarbit AB : Racing Thunder 1 et 2, Toonz Warz, Armagedon Squadron

Médias

- Animations
  - Animation sur tout objet graphique: Image, View, Widget, Activity...
  - Actif sur une un layout et/ou tous ses composants
  - Animations simples
    - AlphaAnimation: change la transparence
    - TranslateAnimation: modifie la position
    - RotateAnimation: effectue une rotation
    - ScaleAnimation: modifie la taille
    - AccelerateInterpolator: accélère le déplacement
  - Effet d'animations enchainées
    - AnimationSet
  - L'enchainement successif d'AnimationSet se fera via des Threads

Les animations ont été intégrées dans le Android SDK 1.1.

Deux types:

- *tweened animation*: réaliser une série d'animation simple

- *frame-by-frame*: réalise l'affichage successif de ressources de type Drawable (AnimationDrawable)

Les animations peuvent être appliquées sur n'importe quelle vue et de ce fait il est alors possible par exemple d'animer un liste affichée à l'écran, bouger un EditView lorsque ce dernier prend le focus...

Certaines animations sont désormais natives à Android 1.5. Des effets d'animations sont réalisés à l'ouverture/fermeture d'une Activity (si l'utilisateur les actives).

Création animation dans le code Java

```
AnimationSet set = new AnimationSet(true);
Animation animation = new AlphaAnimation(0.0f, 1.0f);
animation.setDuration(100);
set.addAnimation(animation);
animation = new TranslateAnimation(Animation.RELATIVE_TO_SELF, 0.0f,
Animation.RELATIVE_TO_SELF, 0.0f, Animation.RELATIVE_TO_SELF, -1.0f,
Animation.RELATIVE_TO_SELF, 0.0f);
animation.setDuration(500);
set.addAnimation(animation);
LayoutAnimationController controller = new LayoutAnimationController(set, 0.25f);
```

- Animations
  - Gestion d'évènement sur l'animation AnimationListener
    - OnAnimationStart
    - OnAnimationEnd
    - onAnimationRepeat
  - Les animations définies par
    - Code Java
    - Code XML (à placer dans /res/anim)
  - Peuvent être répétées
    - Fixe
    - Infinie

Exécution de l'animation:

```
 Animation animation = AnimationUtils.loadAnimation(ctx,
R.anim.monanim);
  target.startAnimation(animation);
  return animation;
```

Ici l'animation est chargée depuis un fichier XML nommé monanim.xml présent dans /res/anim.

```
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
       android:interpolator="@android:anim/accelerate_interpolator"
       android:fromAlpha="0.0" android:toAlpha="1.0"
     android:duration="100" />
```

Elle est ensuite exécutée sur l'objet target.

Une fois que l'animation est lancée, les éléments visualisés ne sont plus atteignables. Toute modification sur le contenu de la vue sera visible après la fin de l'animation.

Médias

- Audio
  - Décodeur
    - Service MediaPlayer
    - Fichiers distants ou locaux (pas de streaming)
    - MediaPlayer
    - /res/raw

```
MediaPlayer mp = new MediaPlayer();
mp.setDataSource(PATH_TO_FILE);
mp.prepare(); // non nécessaire pour une ressource !
mp.start();

Ou mp = new MediaPlayer() ;
mp.create(context, rawId) ;
mp.start();
```

  - Encodeur
    - Service MediaRecorder
    - Enregistrement format 3GP
  - Moteur JetAudio (jeu)
- Vidéo
  - MediaPlayer ou VideoView

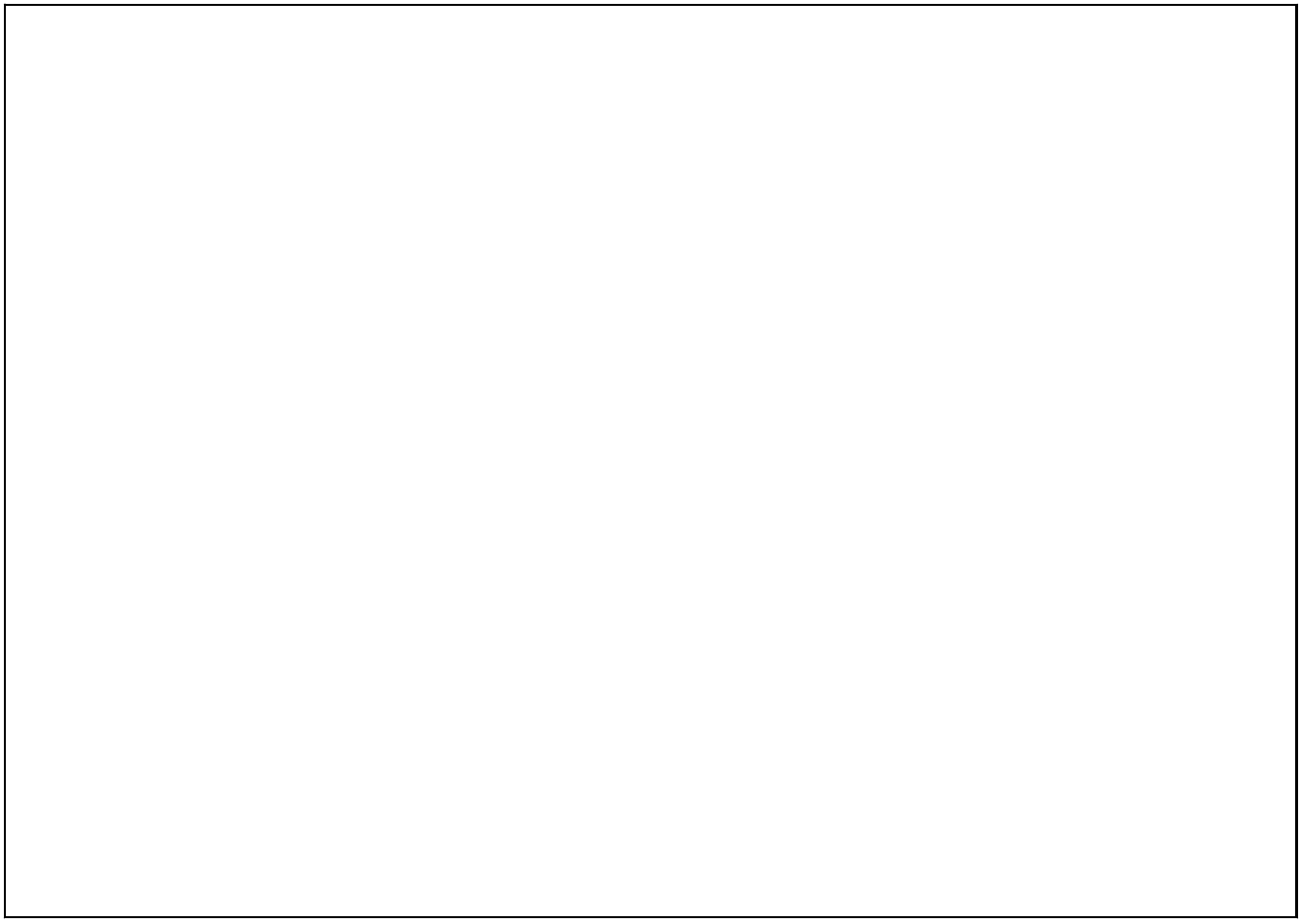
La lecture de son de petite taille (notamment pour des sons d'effet) pourra être réalisée par un objet de type SoundPool qui ne requiert pas l'accès au media player.

Il est à noter que la lecture d'un média via le MediaPlayer ne s'arrête pas lorsque l'Activity cesse il sera donc nécessaire d'implémenter l'arrêt sur la méthode onPause ou onDestroy.

Type	Format	Encoder	Decoder	Details	File Type(s) Supported
Audio	AAC LC/LTP		X	Mono/Stereo content in any combination of standard bit rates up to 160 kbps and sampling rates from 8 to 48kHz	3GPP (.3gp) and MPEG-4 (.mp4, .m4a). No support for raw AAC (.aac)
	HE-AACv1 (AAC+)		X		
	HE-AACv2 (enhanced AAC+)		X		
	AMR-NB	X	X	4.75 to 12.2 kbps sampled @ 8kHz	3GPP (.3gp)
	AMR-WB		X	9 rates from 6.60 kbit/s to 23.85 kbit/s sampled @ 16kHz	3GPP (.3gp)
	MP3		X	Mono/Stereo 8-320Kbps constant (CBR) or variable bit-rate (VBR)	MP3 (.mp3)
	MIDI		X	MIDI Type 0 and 1. DLS Version 1 and 2. XMF and Mobile XMF. Support for ringtone formats RTTTL/RTX, OTA, and iMelody	Type 0 and 1 (.mid, .xmf, .mxmf). Also RTTTL/RTX (.rtttl, .rtx), OTA (.ota), and iMelody (.imy)
	Ogg Vorbis		X		Ogg (.ogg)
	PCM/WAVE		X	8- and 16-bit linear PCM (rates up to limit of hardware)	WAVE (.wav)
Image	JPEG	X	X	Base+progressive	JPEG (.jpg)
	GIF		X		GIF (.gif)
	PNG		X		PNG (.png)
	BMP		X		BMP (.bmp)
Video	H.263	X	X		3GPP (.3gp)
	H.264 AVC		X		3GPP (.3gp) and MPEG-4 (.mp4)
	MPEG-4 SP		X		3GPP (.3gp)

Médias

- Photo / Vidéo
  - Api Camera
  - android.hardware.Camera/Video
    - Récupérer le flux de la caméra Camera.open()
  - Utiliser les Intents pour prendre une photo/vidéo et l'intégrer dans votre application
    - Abstraction de la gestion du hardware et des actions utilisateurs
    - Récupération du flux
    - Utilise le système de ContentProvider
  - Visualiser le contenu de la caméra directement dans l'application via SurfaceHolder
    - Contrôler totalement l'action sur la caméra
    - Les traitements graphiques devront être réalisés dans un Thread séparé



TD2

- Objectif : manipuler les médias
- Jouer et arrêter un son (embedded)
- Sélectionner une photo dans la galerie
- Prendre une photo et récupérer le contenu

eXpertise @ndroid

- Dans la classe MédiaTraining qui sera appelée depuis un nouvel OptionsMenu de FormationActivity :

  - récupérer tous les boutons présents dans le layout afin d'y affecter un onClickListener evtBtnMedia que vous implémenterez

Le bouton play, jouera un son « sound » provenant du répertoire /res/raw ; Appliquer une boucle automatique sur ce son pour le jouer indéfiniment.

Le bouton stop arrêtera le flux audio courant

Le bouton choose_gallery permettra d'accéder à la librairie d'image du téléphone depuis laquelle l'utilisateur sélectionnera l'image qui viendra en fond de l'activity courante. Vous utiliserez un Intent d'action ACTION_PICK sur l'URI android.provider.MediaStore .Images.Media.EXTERNAL_CONTENT_URI

Le bouton take_pic permettra à l'utilisateur de prendre une photo qui s'affichera en dessous des boutons dans un ImageView. Vous utiliserez l'action MediaStore.ACTION_IMAGE_CAPTURE. Il sera nécessaire de passer un EXTRA_OUTPUT Intent avec le chemin d'un fichier temporaire.

Les deux appels aux Intents attendent un résulat, prévoir les méthodes et valeurs requises.

Géolocalisation

- Intégration de GoogleMap dans vos applications
  - MapActivity
  - MapView
  - Plein écran?
- Possibilité
  - Afficher des cartes
  - Basculer sur plusieurs modes (StreeView, Satellite, ...)
  - Ajouter des informations (Image, texte...)
  - Géolocalisation des points
- Google API Key (clé différente debug / publication !)
  - Utilisation des données de Google
  - Prouve et certifie l'utilisation pour vos applications
  - http://code.google.com/android/add-ons/google-apis/mapkey.html

```
keytool.exe -list -alias androiddebugkey -keystore "C:\android\debug.keystore"
-storepass android -keypass android
```

eXpertise @ndroid

La génération du certificat est basée sur le code MD5 d'un certificat de debug utilisé dans le cadre du développement de vos applications. Ce certificat de debug est généré automatiquement par ADT lorsque vous exécutez pour la première fois une application sur votre environnement de développement.

Pour la diffusion de vos applications, il est nécessaire de générer un certificat valide et ce dernier sera utilisé pour générer une nouvelle clé d'accès à l'API Google Map.

Par défaut, le certificat, debug.keystore, de debug est situé sur Windows sous:

C:\Documents and Settings\utilisateur\.android

Sur Linux il sera placé sur /home/utilisateur/.android

Dans Eclipse, avec le plugin ADT, allez dans Windows/Preferences/Android/build où le chemin complet est affiché.

L'utilitaire keytool est fourni avec le JDK et non avec Android SDK.

Après avoir obtenu le code MD5 du certificat, rendez-vous sur l'URL donné en référence et remplissez le formulaire. Il vous sera alors fourni une clé à placer dans le fichier Manifest de votre application.

Géolocalisation

- Prérequis
  - <uses-library android:name="com.google.android.maps" />
    - A placer dans la balise Application
  - <uses-permission android:name="android.permission.INTERNET" />
    - A placer dans la balise Manifest
- Classe étend de MapActivity
- Layout contenant un objet MapView
  - Utilisation de la clé

C'est lors de l'enregistrement de votre MapView dans votre fichier de Layout que la clé fournie par Google sera utilisée comme suit

```
<com.google.android.maps.MapView
      android:id="@+id/mapContent"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:apiKey="0K253o_NK2cnG7l2KIQ2MMzQbnWIRMbt_NfJ9tA"/>
```

Géolocalisation

- Affichage plein écran
- Cliquable
  - Ajout de paramètres sur MapView

```
android:enabled="true"
android:clickable="true"
```

- Contrôleur zoom

```
setBuiltInZoomControls(true);
```

- Retourner les coordonnées pointées

```
GeoPoint p = mapView.getProjection().fromPixels(
     (int) event.getX(),(int) event.getY());
latitude = p.getLatitudeE6() / 1E6;
longitude = p.getLongitudeE6() / 1E6;
```


L'appel à StreetView ne pourra se faire au sein de votre application. Il sera nécessaire d'appeler l'intent StreetView avec un code semblable à celui ci-dessous:

```
GeoPoint point = this.getProjection().fromPixels((int) x, (int) y);
uri = "google.streetview:cbll=" + point.getLatitudeE6() / 1E6 + ","
                        + point.getLongitudeE6() / 1E6
                        + "&cbp=1,45,,45,1.0&mz=5.0";
Intent intent = new Intent();
intent.setData(Uri.parse(uri));
intent.setAction("android.intent.action.VIEW");
getContext().startActivity(intent);
```

Géolocalisation

- Récupérer une adresse postale
  - GéoCoding: à partir du couple latitude/longitude
  - android.location.GeoCoder

```
Geocoder geoCoder = new Geocoder(getBaseContext(), Locale.getDefault());
try {
    List<Address> addresses = geoCoder.getFromLocation(latitude,
                                                       longitude, 1);
    if (addresses.size() > 0) {
    for (int i = 0; i < addresses.get(0).getMaxAddressLineIndex(); i++)
       adresse += addresses.get(0).getAddressLine(i) + "\n";
    }
} catch (IOException e) {...};
```

- Positionner en fonction d'une adresse
  - Reverse Geocoding
  - android.location.GeoCoder
  - getFromLocationName(String adresse, int maxResult)

Se positionner sur la carte à partie d'une adresse

```
List<Address> adresses = geoCoder.getFromLocationName(
            "Avenue Mozart, Paris", 5);
if (addresses.size() > 0) {
    p = new GeoPoint(
                (int) (adresses.get(0).getLatitude() * 1E6),
                (int) (adresses.get(0).getLongitude() * 1E6));
}
```

Géolocalisation

- Prévoir un container (overlay) par type de contenu
  - MapView contient une liste d'Overlay
  - Les Overlay contiennent des OverlayItem
- Récupérer la liste d'overlay depuis la MapView
- Créer votre classe étendant de itemizedOverlay<OverlayItem>
- Gestion du touch et des éléments OverlayItem sélectionnés
- Populate de ItemizedOverlay rafraîchit les vues affichées
- Possibilité d'utiliser la méthode onDraw

```
Point screenPts = new Point();
mapView.getProjection().toPixels(point, screenPts);
Bitmap bmp = BitmapFactory.decodeResource(
    LocationTrainingActivity.this.getResources(),
    R.drawable.pic);
canvas.drawBitmap(bmp, screenPts.x, screenPts.y, null);
```


Dans l'exemple donné ici, l'image est positionnée à x, y...Mais attention, il est nécessaire d'ajuster ces coordonnées en fonction de la taille de votre image (coin bas gauche).

Le position en fonction du GPS ou l'utilisation d'un compass sera réalisée en utilisant les API de com.google.maps.* en plus particulièrement la classe MyPositionOverlay:

```
myLocationOverlay = new MyLocationOverlay(this, mapView);
mapView.getOverlays().add(myLocationOverlay);
myLocationOverlay.enableCompass();
myLocationOverlay.enableMyLocation();
```

Les méthodes enableCompass et enableMyLocation doivent être enlevées (disableCompass, disableMyLocation) lors de la sortie de l'application onDestroy.

L'utilisation de certaines fonctionnalités, telle que celle définie ci-dessus, n'est pas testable dans un environnement de type émulateur.

Géolocalisation

Gérer le déplacement

Récupérer un LocationManager

Utiliser un LocationListener pour obtenir l'accès au méthode

- OnLocationChanged
- OnProviderDisabled
- OnProviderEnable
- OnProviderChanged

Demander au service de localisation un rafraichissement de la position

- RequestLocationUpdate (provider, LocationListener)


```
private void initLocationManager() {
		locManager = (LocationManager)
getSystemService(Context.LOCATION_SERVICE);
	locListener = new LocationListener() {
		public void onLocationChanged(Location newLocation) {
				// afficher votre OverlayItem dans un Overlay
		}

			public void onProviderDisabled(String arg0) {
			}

			public void onProviderEnabled(String arg0) {
			}

			public void onStatusChanged(String arg0, int arg1, Bundle arg2) {
			}
		};
		locManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0,
				locListener);

	}
```

LIENS UTILES

Livres
Professionnal Android 2 Application development – Reto Meier – Wrox Press (mars 2010)
Paru en français aux éditions Pearson

Busy coder's guide to Android Development – Mark. L. Murphy – Edition CommonsWare – 444 pages + 2 livres tutoriaux et Advanced Development
http://www.commonsware.com
Disponible en français aux editions Pearson Education “L’art du développement Android” - 3ème édition octobre 2011

Programmation Android : De la conception au déploiement (mai 2010) – Edition Eyrolles

Ressources en lignes: site web
http://developer.android.com (documentations officielle)
http://www.anddev.org (tutoriaux, forums)
http://www.androidcommunity.com (actualités)
http://www.devx.com/wireless (actualités, tutoriaux)
http://www.helloandroid.com (actualités)

Ressources en lignes: forums
http://groups.google.fr/group/android-developers
http://groups.google.fr/group/android-beginners
Http://www.stackoverflow.com

Ressources francophones: actualités et développement
http://www.frandroid.com (actualités, forums, wiki, tutoriaux)
http://www.pointgphone.com (actualités, forums)
http://www.android-france.fr

Contact

eXpertise @ndroid
10, rue du Cherche Midi
06 250 MOUGINS
Tél: 06 30 05 46 49
email: contacts@expertiseandroid.com