Introduction au Diveloppement d'Application Android

OBJECTIF DU COURS

- Comprendre l'architecture du système Android
- Comprendre l'organisation d'une application Android
- Développer et déployer des applications natives Android

Plan du cours

1. Introduction
2. Android SDK
3. Les composants d'une application mobile
   - Manifest
   - Activity
   - Context
   - Intent
   - Resources
   - Layout
   - Widget
   - Adapter
   - Permission
   - Etc…

TPs

Installation et configuration d'Android Studio

Première application Android

Intents explicites et gestion des événements Intents implicites et gestion des événements Android

Adapter

Menu Android

Navigation drawer

Pré requis

Programmation Orienté

Objet JAVA

XML

Enjeux du développement mobile

CPU entre 500 et 600 MHz

Mémoire allouée à une application est de l'ordre de quelques mégabytes

Gestion du cycle de vie → une seule application peut être visible à la fois

Densité multiples des écrans

- Très faible résolution
- Très haute définition

Plusieurs orientations (Portrait, Paysage, Portrait inverse, …)

Introduction / Définition

Android est un système d'exploitation OPEN SOURCE pour terminaux mobiles (Smartphones, Tablet ,....

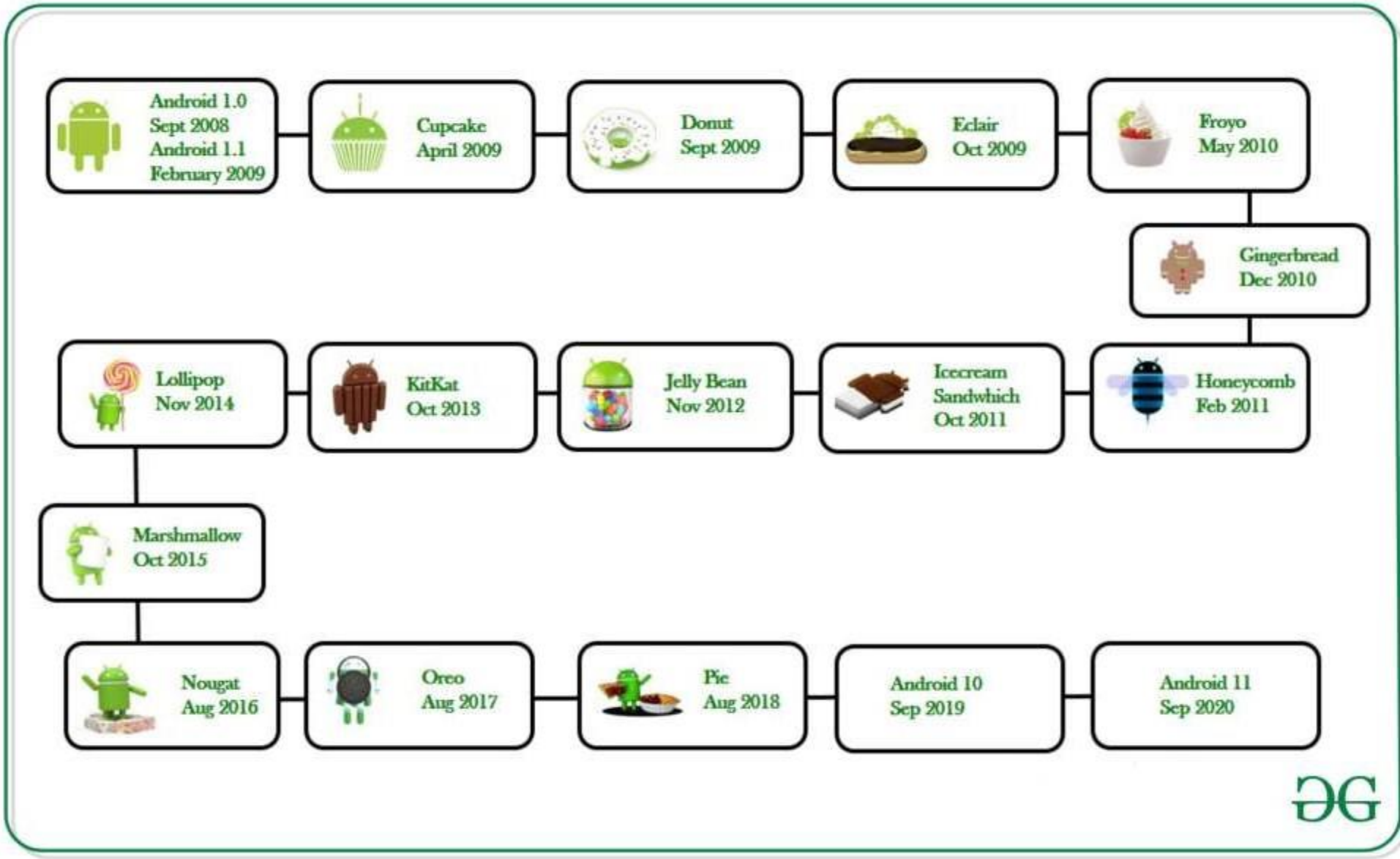
Conçu à la base par une startup (Android) rachetée par Google en 2005

Pour la promotion de ce système Google a fédéré autour de lui une trentaine de partenaires réunis au sein de l'Open Handset Alliance (OHA)

C'est aujourd'hui le système d'exploitation mobile le plus utilisé à travers le monde

Introduction / Définition

Versions de l'OS



Plateforme Android

Le système d'exploitation Android est basé sur Linux. Au plus bas niveau de ce système se trouve un noyau Linux destiné à la gestion du matériel comme :

Drivers de ces terminaux,

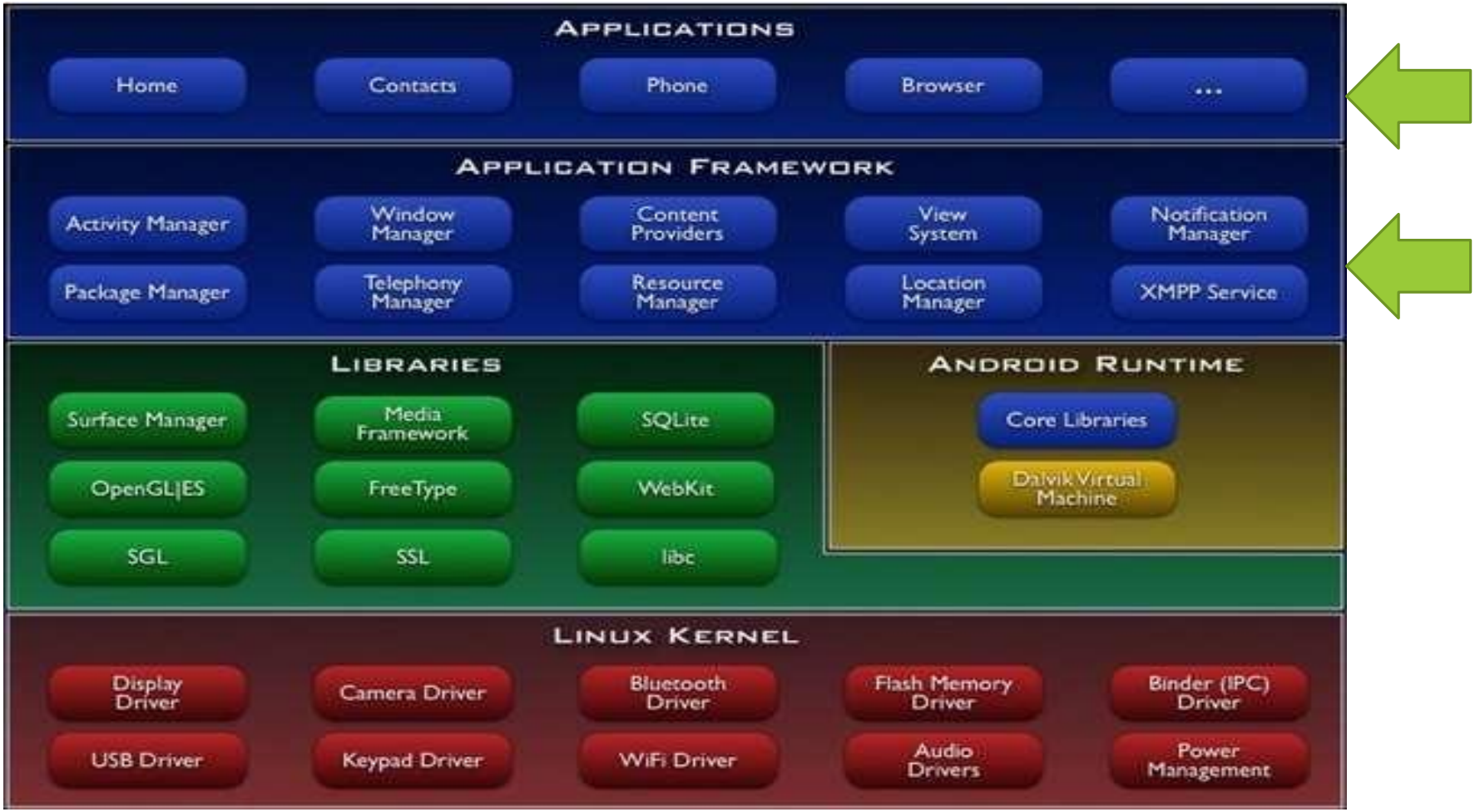
La gestion de la mémoire,

La gestion des processus

L'accès au réseau

…

Plateforme / Architecture



Android pour tous

Développeurs

- Pas besoin de licence
- Simple et intuitifs
- Modulables

Constructeurs

- Tous les constructeurs peuvent utiliser Android
- Un ensemble de services sont déjà disponibles dans le core
- API disponible pour les accès de bas niveau

Android et les langages de programmation

Android n'est pas un langage de programmation

Pour développer sous Android, il existe deux possibilités :

- Développement native (Java, C, C#)
- Développement hybride

Android & Java

Le SDK Android est développé en Java → Permet de développer des applications avec un haut niveau d'abstraction

Android a sa propre machine virtuelle Dalvik Virtual Machine

Ne supporte pas toutes les fonctionnalités de la JRE

Une application Android ne peut pas s'exécuter sur une machine virtuelle Java

Une application Java (native) ne peut pas s'exécuter sous Android

Android & C/C++

Il est possible d'écrire des applications Android en utilisant le langage C/C++ qui seront exécutées directement par le système d'exploitation Linux embarqué

Android fournit le kit de développement NDK pour les développements d'application en C/C++

Utilisé dans le développement de jeux 2D/3D se basant fortement sur la librairie OpenGL

Android vs Développement Hybride

Android supporte le développement hybride

- Titanium
- Phonegap
- Neomad

Pour ce cours….

Framework et langage

- Android SDK
- Java
- XML

Outils

- Android Studio

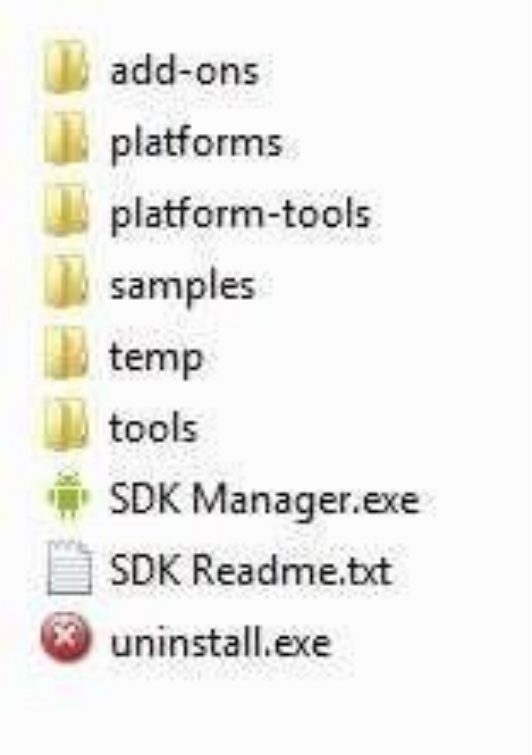
Android SDK

Disponible en téléchargement sur :

developper.android.com/sdk Propose le SDKManager qui :

¨Permet de télécharger les plateformes et outils :

- Android versions xx
- Google API versions xx
- Outils (tools et platform-tools)



Outils du SDK

adb➔ Accessibles à partir d'une ligne de commande (fenêtre DOS)
adb permet la connexion au terminal (smartphone ou simulateur) pour :

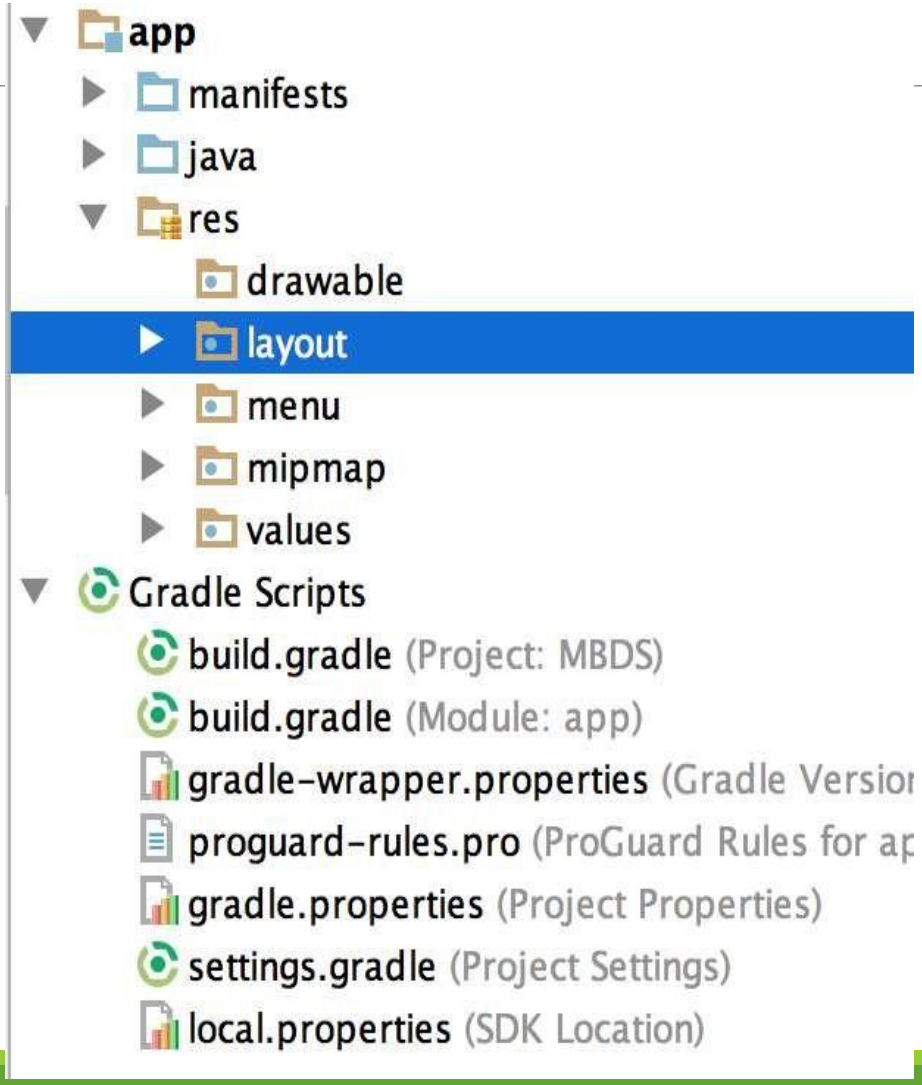
- Transférer des fichiers (push / pull)
- Installer une application (install)
- Paramétrer le réseau (forward)

dx ➔ Transforme le bytecode java en code Dalvik

apkbuilder ➔ Compiler les sources d'une application Android pour constituer une archieve (.apk) directement installable sous un terminal Android

DDMS / Monitor ➔ Monitoring sur les activités du terminale

Structure d'une application



Référence : http://developer.android.com/tools/projects/index.html

Type d'applications Android

Application de premier plan : Application utilisable que lorsqu'elle est visible et n'effectuant aucune tache de fond

Services : Application s'exécutant en tache de fonds et ne présentant pas d'interfaces utilisateurs

Intermittente : Application exécutant à la fois des taches des fonds et permet l'interaction avec l'utilisateur; la communication entre les
taches de fonds et l'utilisateurs se fait par des notifications

Widgets: Application(utilitaires) pouvant être placé directement sur un écran du téléphone

Composantes d'une application

Activité (android.app.Activity) ➔ Programme qui gère une interface graphique

Service (android.app.Service) ➔ Programme qui fonctionne en tâche de fond sans interface

Fournisseur de contenu (android.content.ContentProvider) ➔
Partage d'informations entre applications

Intentions : Communication inter et intra-application

Ecouteur d'intention (android.content.BroadcastReceiver) ➔ Permet à une application de récupérer des informations générales (réception d'un SMS, batterie faible, ...)

Widgets : Bibliothèque de composants visuels

Manifest: Fichier de configuration de l'application

Manifest

Configuration de l'application

- Définit le nom de package de l'application;
- Décrit les composantes de l'application: activités, services, fournisseurs de contenus…., il nomme les classes implémentant chacun de ces composants et propage leur capacité sur le système;
- Détermine quel processus répond à une action donnée;
- Déclare les permissions de l'application;

Manifest : Exemple

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

<manifest
xmlns:android="http://schemas.android.com/apk/res/android"
package="ht.solutions.android.whereami" android:versionCode="1"
android:versionName="1.0">

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

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

<application android:label="@string/app_name"
android:icon="@drawable/ic_launcher">

<activity android:name="Home" android:label="@string/app_name">
<intent-filter> <action android:name="android.intent.action.MAIN"
/>

<category android:name="android.intent.category.LAUNCHER" />
</intent-filter> </activity>

</application>

</manifest>
```

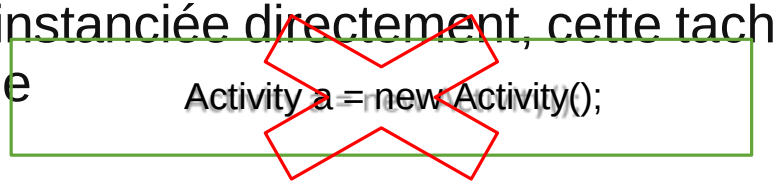
Activité (Activity)

Modélise et gère les interfaces de l'application. Dans un modèle MVC il joue le rôle de contrôleur

Hérite (toujours) de la classe Activity d'Android

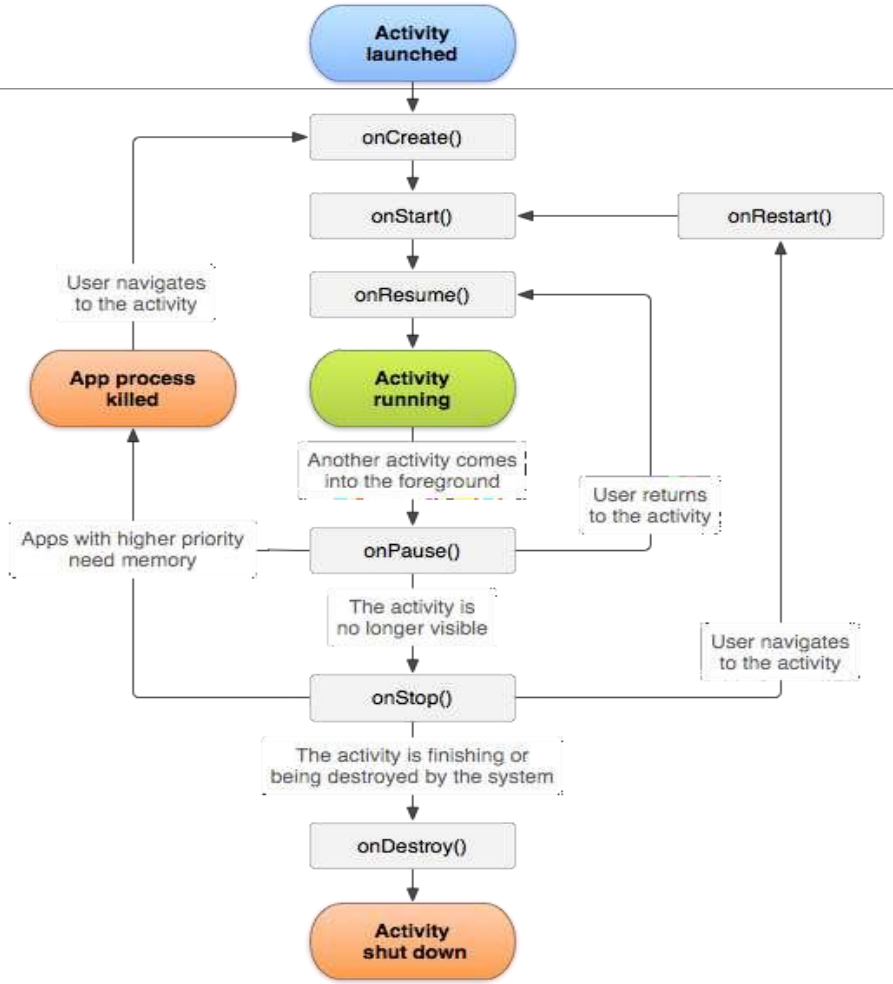
Doit être déclarée dans le Manifest pour être visible par le système

Ne peut être instanciée directement, cette tache est effectuée par le système



Une activité est instanciée en utilisant les intentions

Activité /cycle de vie



Activité / Création

```
public class ExampleActivity extends Activity {
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        // The activity is being created.
    }
    @Override
    protected void onStart() {
        super.onStart();
        // The activity is about to become visible.
    }
    @Override
    protected void onResume() {
        super.onResume();
        // The activity has become visible (it is now "resumed").
    }
    @Override
    protected void onPause() {
        super.onPause();
        // Another activity is taking focus (this activity is about to be
"paused").
    }

    }
```

Activité/Création

Une activité doit etre obligatoirement déclarée dans le Manifest

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

Contexte (Context)

- Interface fournissant des informations sur l'environnement de l'application:
- C'est une classe abstraite implémentée par le système Android
- Il permet d'accéder aux principales ressources de l'application
- Obtenir le contexte courant d'une application
  - Context c = getApplicationContext(): Contexte global de l'application
  - Context c = Activity.this, getContext(): Contexte d'une activité ou un service

Les intentions

Il existe deux principaux types d'intentions :

- **Explicite** : Spécifie les composants qui précisent la classe exacte qui doit être exécutée (setComponant (nom) ou setClass(context, class))
- **Implicite** : Ne spécifie pas le composant mais fournit assez d'informations permettant au système de determiner les composants nécessaires correspondant à cette action.

Les Intentions (Intent)

Description abstraite d'une action à exécuter:

Action : Action à exécuter

Data : Données sur lesquelles l'action va opérer

Exemple : Action_DIAL tel:0641xxxx

Autres attributs :

Category : Précise la catégorie de l'activité demandée

Type: Préciser le type de contenus des données de l'intention

Extras : Informations additionnelles envoyées à l'action (envoie d'un email, par exemple)

Démarrer une activité

Démarrage explicite

```
Intent demarre = new Intent(context,
NomDeLaClasseDeLActiviteALancer.class); startActivity(demarre);
```

Démarrage implicite

- Exemple : lancer un navigateur web:

```
Uri chemin = Uri.parse("http://www.google.fr");
Intent naviguer = new Intent(Intent.ACTION_VIEW,
chemin); startActivity(naviguer);
```

- Exemple : appeler un n° de téléphone :

```
Uri numero = Uri.parse("tel:0123456789");
Intent appeler = new Intent(Intent.ACTION_CALL,
numero); startActivity(appeler);
```

Méthodes de la classe Intent

Construction

- Intent(String) : avec action
- Intent(String, Uri) :avec action et Uri

Ajout de catégorie

- addCategory(String) ajoute une catégorie
- setDataAndType(Uri, String) définit l'Uri et le type mime des données

Paramètres

- putExtra(nom, valeur) ajoute un paramètre associé à un nom
- getxxxExtra(nom) renvoie le paramètres correspondant au nom (xxx dépend du

type de paramètre : Int, String, StringArray ...)

Quelques valeurs prédéfinies

Actions

- android.intent.action.CALL appel téléphonique
- android.intent.action.EDIT affichage de données pour édition par l'utilisateur
- android.intent.action.MAIN activité principale d'une application
- android.intent.action.VIEW affichage de données
- android.intent.action.WEB_SEARCH recherche sur le WEB

Catégories

- android.intent.category.DEFAULT activité pouvant être lancée explicitement
- android.intent.category.BROWSABLE peut afficher une information désignée par un lien
- android.intent.category.LAUNCHER activité proposée au lancement par Android
- android.intent.category.TAB activité associée dans un onglet d'interface (TabHost)

Les filtres d’intention

Permettent de définir les intentions d’une activité

```
<activity
android:name=".ExampleActivity"
android:icon="@drawable/app_icon">
    <intent-filter>
        <action android:name="android.intent.action.MAIN" />
        <category
        android:name="android.intent.category.LAUNCHER" />
    </intent-filter>
</activity>
```

Lancer une activité et attendre un résulat

La méthode startActivityForResult (intent), permet de demarrar une activité et attendre un résultat

La méthode onActivityResult permet de récupérer les données renvoyées par l'activité appelante

startActivityForResult et onActivityResult

```
Intent intent = new
Intent(Intent.ACTION_PICK,
Contacts.CONTENT_URI);
```

startActivityForResult(intent, PICK_CONTACT_REQUEST);

onActivityResult (int requestCode, int resultCode, Intent data)

- **requestCode : Identifie l'action qui a dlancé l'activité**
- **resultCode: Indique le statut du résultat (RESULT_OK, …)**
- **Data : renforme des données renvoyées par l'activité appelée**

Démarrer une activité

La classe Intent permet de passer des paramètres à l'activité appelée et d'en récupérer les valeurs en retour

Ajouter des paramètres (types simples ou tableaux)
intent.putExtra(String, val)
Le 1er paramètre est un nom
(clé) Le second paramètre est la
valeur :

- De type simple (boolean, int, short, long, float, double, char)
- Tableau de types simples

L'activité appelée pourra récupérer ces paramètres
par leur nom

Activités / Paramètres (Bundle)

L'activité lancée récupère l'objet de classe Bundle contenant les paramètres par :
Bundle params = getIntent().getExtras()

- Les paramètres sont récupérés dans ce Bundle par ses méthodes :
  - getBoolean(String)
  - getInt(String)
  - getBooleanArray(String)
  - ...

Exemple :
String myId = getIntent().getStringExtra(« id»);

Ressources (Resource)

➔Une application Android n'est pas seulement fait de codes mais aussi de ressources statiques (images, sons, text statique,....)

➔Tout projet Android a un dossier de ressources (res/) contenant les
ressources du projet (bitmap, xml,...)

- /res/drawable ➔ images (R.drawable.nom_de_la_ressources)
- /res/layout ➔ Design des vues (R.layout.nom_de_la_vue)
- /res/values/strings ➔ Chaînes de caractères, tableaux, valeurs numériques
  ... (R.string.nom_chaine, R.array.nom)
- /res/anim ➔description d'animations (R.anim.nom_animation_)
- /res/menus ➔ Menus pour l'application (R.menu.nom_menu)
- /res/values/color ➔ Code de couleurs (R.color.nom_couleur)
- ...

Ressources & valeurs

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">Mon application</string>
<string name="server_url">http://mon.serveur.com</string>
<integer-array name="codes_postaux">
<item>64100</item>
<item>33000</item>
</integer-array>
<string-array name= "planetes">
<item>Mercure</item>
<item>Venus</item>
</string-array>
<dimen name "taille">55px</dimen>
</resources>
```

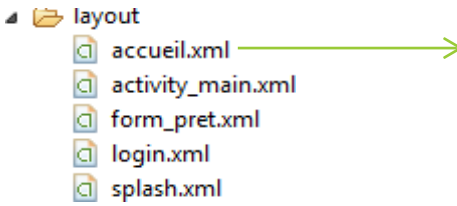
Référencer les ressources

L'ensemble ressources sont modélisés par la classe « R.java » et les

sous dossiers par des classes internes à R

Chaque ressource est un attribut de la classe représentant le sous dossier dans lequel il est déclaré

R.layout.accueil




```
<?xml version="1.0" encoding="utf-8"?>
<resources>
<string name="app_name">Mon application</string>
……
</resources>
```

R.string.app_name

Référencer les ressources

Référencement d'une ressource dans une autre ressource. La forme générale est : "@type/identificateur"

```
<TextView
        android:id="@+id/lblUsername"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_weight="0.00"
        android:text="@string/lblUsername"
        android:textColor="@android:color/ho
        lo_blue_dark"
      android:textAppearance="?android:attr/textA
      ppearanceMedium"
/>
```

Référencer les ressources

Dans le code (java), on obtient une instance de cette classe par getResources()

Principales méthodes de la classe Resources (le paramètre est un identifiant défini dans R de la forme R.type.nom) :

- boolean getBoolean(int)
- int getInteger(int)
- int[] getArray(int)
- String getString(int)
- String[] getStringArray(int)
- int getColor(int)
- float getDimension(int)
- Drawable getDrawable(int)

Exemple : String titre = context.getResources().getString(R.string.ma_chaine);

Mais dans une activité on peut faire plus simplement : String titre = getString(R.string.ma_chaine);

Référencer les ressources

Accéder aux vues :

getResources().getLayout(R.layout.nom_layout); Accéder aux valeurs :

```
String chaine = getResources().getString (R.string.nom_string);
String[] tableau= getResources().
getStringArray(R.string.nom_array);
```

Accéder aux images :

```
Drawable monImage = getResources().getDrawable(R.drawable.nom_image)
```

Interfaces (Layout)

Les interfaces (Layout) permettent de dessiner la vue tel qu'elle doit s'afficher à l'utilisateur.

Il existe deux possibilités de développer des interfaces :

➔Code java

➔Fichier XML

Android recommande l'utilisation des fichiers XML pour définir les interfaces.

➔La méthode setContentView permet d'associer un layout à
une activité

Exemple :

```
setContentView(R.layout.login);
```

Interfaces (Layout) / Vues

Les interfaces sont composées de vues :

Les vues ne peuvent être modifiées que par le thread principal « UIThread » .

Tous les autres threads créés par un utilisateur ne peuvent pas modifier les vues.

Vues Propriétés communes

Identifiant : android:id = "@+id/*mon_ident*"

Dimension:

layout_height, android:layout_width : fill_parent / match_parent / wrap_content

fill_parent : Remplit toute la place

wrap_content : remplit la place que necessite le contenu

match_parent : remplit la place qui reste dans le parent **Fond**

android: background

Visibilité

android:visibility : visible / invisisble/gone

Vues Propriétés communes

Marges internes

android:layout_paddingBottom , android:layout_paddingLeft

,

android:layout_paddingRight , android:layout_paddingTop

Marges externes

android:layout_marginBottom , android:layout_marginLeft

,

android:layout_marginRight , android:layout_marginTop

Vues

Les vues déclarées dans les fichiers xml sont automatiquement instanciées par Android.

Pour récupérer l'instance d'une vue associée à une interface, Android propose la méthode findViewById() prenant en paramètre l'identifiant de la vue dans le fichier xml.

La méthode findViewById utilise le conteneur principal de l'interface de l'activité courante.

```
View v = findViewById(R.id.myView);
```

Connaissant le type de la vue

```
Button b = (Button)findViewById(R.id.btnCancel);
```

Les vues peuvent être créés dynamiquement :

```
Button b = new Button(this); b.setId(1245);
```

Vues / Evènements

Les évènements permettent de gérer les actions utilisateurs sur les vues:

Pour gérer les évènements sur les vues, il suffit d'ajouter un écouteur

```
button.setOnClickListener(new View.OnClickListener()
@Override
public void onClick(DialogInterface dialog, int which) {
// Du code ici
}
});
```

```
button.setOnClickListener(new OnClickListener() {
@Override
public void onClick(View v) {
// TODO Auto-generated method stub
}
});
```

Les conteneurs (ViewGroups)

Permettent de positionner des widgets dans une interface :

- FrameLayout
- AbsoluteLayout
- LinearLayout
- TableLayout
- RelativeLayout
- ScrollView
- HorizontalScrollView

LinearLayout

Définit le positionnement linéaire (horizontall ou vertical) des vues filles

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical" >

      <TextView
            android:id="@+id/textView1"       android:layout_width="wrap_content"
            android:layout_height="wrap_content"  android:text="@string/label_nom"
            android:textAppearance="?android:attr/textAppearanceLarge" />

      <EditText
            android:id="@+id/editText1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName"    >

            <requestFocus />
        </EditText>

    </LinearLayout>
```

LinearLayout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation=« horizontal" >

      <TextView
            android:id="@+id/textView1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/label_nom"

android:textAppearance="?android:attr/textAppearanceLarge" />

      <EditText
            android:id="@+id/editText1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:ems="10"
            android:inputType="textPersonName" >

            <requestFocus />
        </EditText>

    </LinearLayout>
```

RelativeLayout

Positionner les éléments de l'interface les uns par rapport aux autres

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/label_nom"
        android:textAppearance="?android:attr/textAppearanceLarge" />

    <EditText
        android:id="@+id/editText1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:layout_toRightOf="@+id/textView1"
        android:inputType="textPersonName" >

        <requestFocus />
    </EditText>

</RelativeLayout>
```

TableLayout

- Tableau de positionnement des vues en ligne de TableRow (similaire au <table> <tr> <td> de HTML)
- TableRow hérite de LinearLayout avec alignement automatique des colonnes sur chaque ligne
- Propriétés de TableRow.LayoutParams
  - layout_column: indice de départ de la colonne (à partir de 0)
  - layout_span: nombre de colonnes occupées

Widgets

- Les widgets représentent les blocs élémentaires visibles dans l'interface utilisateurs. Une vue occupe une portion rectangulaire et répond aux évènements d'interaction entre l'utilisateur final et l'application
- Leur disposition dans l'interface est défini par son contenant (Viewgroup)
- On distingue :
  - TextView
  - Button
  - EditText
  - CheckBox
  - RadioButton
  - ImageView
  - ...

Widgets

```
<TextView android:id="@+id/my_text"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:layout_weight="0.00"
android:text="@string/lblUsername"
android:textColor="@android:color/holo_blue_dark"
android:textAppearance="?android:attr/textAppearanceMedium"
/>

<Button android:id="@+id/my_button"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="@string/my_button_text"/>

<ImageView android:id="@+id/imageView1"
android:layout_width="fill_parent"
android:layout_height="wrap_content"
android:contentDescription="@string/image_description"
android:src="@drawable/login" />
```

Les groupes

Les groupes permettent d'afficher une collection d'éléments à un utilisateur:

- ListView
- Gridview
- RadioGroupe
- Galery
- Spinner

Les adaptateurs

Pour ajouter des éléments à un groupe (ListView, Spinner…), on utilise des adaptateurs.

Adaptateur: Classe qui définit une structure d'affichage pour les groupes.

- Android propose des adapter génériques que le développeur peut utiliser pour afficher des informations dans un élément de groupe
- Le développeur peut définir son propre adapter pour personnaliser l'affichage dans les groupes

Adaptateurs

```
<string-array name="country_arrays">
<item>Malaysia</item>
<item>United States</item>
<item>Indonesia</item>
<item>France</item>
<item>Italy</item>
<item>Singapore</item>
<item>New Zealand</item>
<item>India</item>

</string-array>

</resources>

<Spinner android:id="@+id/spinner1"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:entries="@array/country_arrays"
android:prompt="@string/country_prompt
"
/>
```

```
Spinner sp = (Spinner)
findViewById(R.id.spinne
r1);

ArrayAdapter<String>
dataAdapter = new
ArrayAdapter<String>(this,
android.R.layout.simple_spin
ner
_item, R.array-string.
country_arrays);

dataAdapter.setDropDownVie
w
Resource(android.R.layout.si
mpl
e_spinner_dropdown_item);
spinnerAdapter(dataAdapter );
```

Adaptateurs

Adapter personnalisé

```
public class MySimpleArrayAdapter extends
  ArrayAdapter<String> { private final Context context;
  private final String[] values;

  public MySimpleArrayAdapter(Context context, String[] values) {
    super(context, R.layout.rowlayout, values);
    this.context =
    context; this.values
    = values;
  }

  @Override
  public View getView(int position, View convertView, ViewGroup
    parent) { LayoutInflater inflater = (LayoutInflater) context
        .getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    View rowView = inflater.inflate(R.layout.rowlayout, parent, false);
    TextView textView = (TextView) rowView.findViewById(R.id.label);
    ImageView imageView = (ImageView)
    rowView.findViewById(R.id.icon);
    textView.setText(values[position]);
    // Change the icon for Windows and iPhone
    String s =
    values[position]; if
    (s.startsWith("iPhone")) {
      imageView.setImageResource(R.drawable.no);
    } else {
      imageView.setImageResource(R.drawable.
      ok);
    }

    return rowView;
  }
}
```

```
MySimpleArrayAdapter adapter = new MySimpleArrayAdapter  (context,
values); mySpinner.setAdapter (adapter);
```

Permission

Moyen de sécurité proposé par Android pour gérer l'accès aux
ressources du terminal

Elles apparaissent dans le fichier Android Manifest et sont visibles par l'utilisateur au moment de l'installation de
l'application

Elles concernent :

- La géolocalisation (GPS)
- Les accès aux contacts et à l'agenda du téléphone
- Les modifications de paramètres (orientation, fond
- Les appels téléphoniques
- L'envoi et réception de SMS/MMS
- L'audio
- Le réseau (dont l'accès à Internet)
- Le matériel (bluetooth, appareil photo, …)

Permission

La déclaration des permissions doit se faire explicitement dans le fichier manifest.

Si une permission a été omise et vous essayez d'utiliser la ressource correspondante, vous aurez une erreur à l'exécution!

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

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

Connectivité

Android fournit des mécanismes permettant aux applications d'utiliser la connectivité pour accomplir certaines taches :

- Internet (WIFI, 3G,...)
- Bluetooth
- NFC

Pour accéder à ces ressources, il est obligatoire de de
- <uses-permission ent la permission associée
  android:name="android.permission.INTERNET" />

Internet

Une application Android peut utiliser la connexion réseau du téléphone pour échanger par HTTP des informations avec une application distante (web service, web application,..)

Pour pouvoir utiliser internet :

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

Android utilise la bibliothèque Apache Components pour les communications par http

```
HttpClient client =new        DefaultHttpClient();

HttpGet request =new      HttpGet(url);

HttpResponse
response = client.execute(request);

…
```

Taches Asynchrones

La UIThread est le thread principal de gestion de l'interface utilisateur. L'exécution des processus longs dans ce thread comme l'accès au réseau doit se faire dans un thread à part.

A partir de la version 4, Android génère une exception pour tout
opération longue s'exécutant dans le thread principal.

L'exécution des taches asynchrones peut se faire de deux manières:

- L'utilisateur crée explicitement un thread auquel il délègue l'exécution des processus longs
- Utiliser la classe AsyncTask du SDK qui se chargera de la création du
thread secondaire

Taches Asynchrones

```
protected class LongTask extends AsyncTask<String, Void, Void> {
        @Override
        protected void onPreExecute() {
            super.onPreExecute();
           //Avant l'execution du processus long (Thread principal)
        }
        @Override
        protected void onPostExecute(Void resp) {
            super.onPostExecute(resp);
            //Apres l'execution du processus long (Thread principal)
        }
            @Override
            protected Void doInBackground(String... arg0) {
                        //Execution du processus long (Thread secondaire)
                        return null;
                        }
}
```

Bien plus encore!

Persistance : SQLite,

Préférences Géolocalisation

Multimédia : Vidéo, Musique,...

Connectivité : Bluetooth, NFC, QR

Code Services

Notifications, BroadCast