



Introduction a la programmation native Android

Partie 1

OBJECTIF DU COURS

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

Plan sommaire du cours

1. Introduction
2. La platform Android et la SDK
3. Les types d'application
3. Outil de développement
4. Les composants d'une application mobile
 - ❖ Manifest
 - ❖ Activity
 - ❖ Context
 - ❖ Intent
 - ❖ Resources
 - ❖ Layout
 - ❖ Widget
 - ❖ Adapter
 - ❖ Permission
 - ❖ Etc...
- 5- Développement via Jetpack Compose

TPs

Installation et configuration d'Android Studio

Création d'un projet application Android et structure d'un projet Android

Les vues groups et mise en pages (Layouts)

Les composants d'interface graphique

Intents explicites et gestion des événements

Intents implicites et gestion des événements

Android Adapter et utilisation dans les collections lists/group

Les Bar d'action, dialogues et Menu Android

Navigation drawer

Introduction sur les Jetpack Compose UI

Projects de cours

Pré requis

Programmation Orienté Objet

JAVA / Kotlin

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

Major Devices that runs on Android Operating System



Smartphones



Tablets



Smart watches



Smart TVs



Versions de l'OS

List of Android Versions and Initial Stable Release Dates



Android 1.0
September 23, 2008



1.5 - Cupcake
April 27, 2009



1.6 - Donut
September 15, 2009



2.0/2.1 - Éclair
October 26, 2009



2.2 - Froyo
May 20, 2010



2.3 - Gingerbread
December 6, 2010



3.0 - Honeycomb
February 22, 2011



4.0 - Ice Cream Sandwich
October 18, 2011



4.1/4.3 - Jelly Bean
July 9, 2012



4.4 - KitKat
October 31, 2013



5.0 - Lollipop
November 12, 2014



6.0 - Marshmallow
October 5, 2015



7.0 - Nougat
August 22, 2016



8.0 - Oreo
August 21, 2017



9.0 - Pie
August 6, 2018



Android 10
September 3, 2019



Android 11
September 8, 2020



Android 12
October 17, 2021



Versions de l'OS

Android 13

Protégez la confidentialité des utilisateurs grâce au sélecteur de photos et aux autorisations de notification. Améliorez la productivité avec des icônes d'applications à thème, des langues propres à l'application et un aperçu du presse-papiers. Adaptez-vous aux normes les plus récentes, comme Bluetooth LE Audio et MIDI 2.0 via USB. Offrez une meilleure expérience sur les tablettes et les grands écrans.

En savoir plus

Android 12

Une nouvelle interface utilisateur système expressive, dynamique et personnelle avec Material You. Enrichissez vos applis avec de nouveaux widgets, AppSearch, le mode Jeu et de nouveaux codecs. Intégrez de nouvelles protections, comme le tableau de bord Confidentialité et la position approximative. Gagnez en productivité grâce à l'insertion de contenu enrichi, au floutage simplifié, au débogage natif amélioré et plus encore.

En savoir plus

Android 11

Une approche expressive et centrée sur l'utilisateur, avec un nouvel espace de contrôle et davantage de fonctionnalités de confidentialité. Ajoutez des notifications de conversation et sous forme de bulles dans vos applications, explorez les autorisations ponctuelles ou affichez des appareils et des contenus multimédias dans les commandes. Travaillez plus rapidement grâce à des outils tels que les boutons d'activation/de désactivation de la compatibilité, les installations incrémentielles ADB et plus encore.

En savoir plus

Versions de l'OS

Android 15

Android 15 poursuit notre mission de créer une plate-forme privée et sécurisée qui vous aide à améliorer votre productivité. Elle vous donne également accès à de nouvelles fonctionnalités pour créer des applications attrayantes, des expériences multimédias et des fonctionnalités photo avancées, ainsi qu'une expérience utilisateur intuitive, en particulier sur les tablettes et les appareils pliables.

Accueil

Android 14

Créez des expériences photo et multimédias plus riches avec Ultra HDR, plus d'extensions de l'appareil photo et l'audio USB sans perte. Simplifiez la connexion à l'aide du Gestionnaire d'identifiants et fournissez des insights sur la santé et la remise en forme de manière sécurisée à l'aide de Santé Connect. Rendez votre application plus expressive et personnelle avec la prévisualisation du Retour, les langues par application, l'inflexion grammaticale et plus encore. Proposez des expériences engageantes et différenciées sur les grands écrans.

Accueil

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,

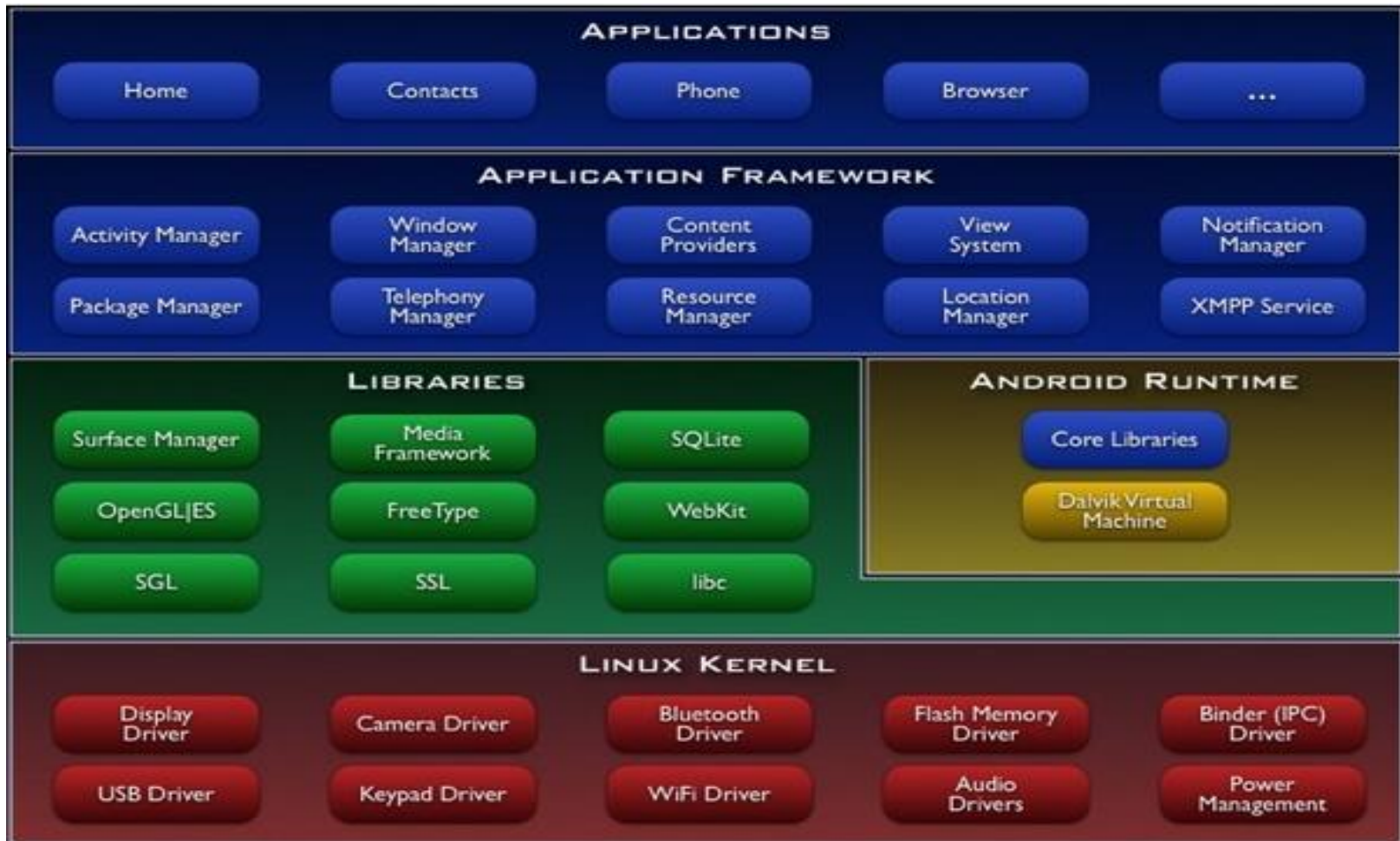
La gestion de la mémoire,

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 / Kotlin
- XML

Outils

- Android Studio

TP sur ordinateur

Installation et configuration d'Android Studio

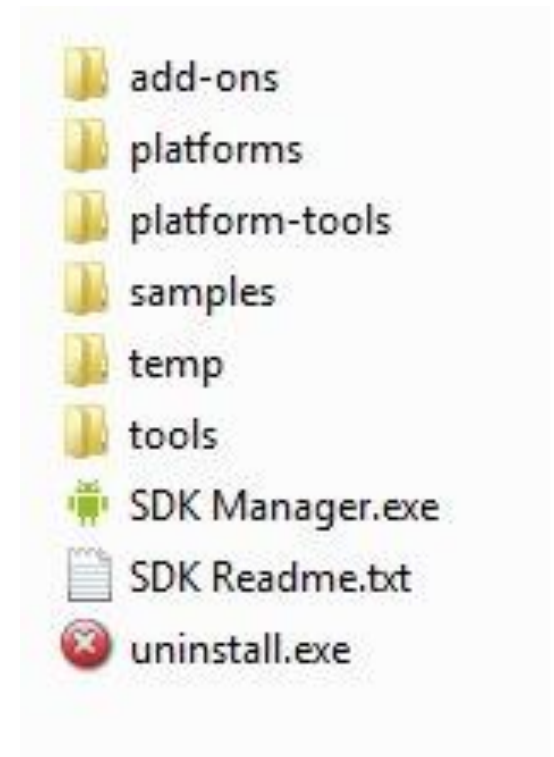
Android SDK

Disponible en téléchargement sur : developer.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 archive (.apk) directement installable sous un terminal Android


















DDMS / Monitor → Monitoring sur les activités du terminale

TP sur ordinateur

Création d'un projet application Android et structure d'un projet Android

Choix des projets a développer

Structure d'une application

- ▼  **app**
 - ▶  manifests
 - ▶  java
 - ▼  **res**
 -  drawable
 -  **layout**
 - ▶  menu
 - ▶  mipmap
 - ▶  values
- ▼  **Gradle Scripts**
 -  build.gradle (Project: MBDS)
 -  build.gradle (Module: app)
 -  gradle-wrapper.properties (Gradle Version)
 -  proguard-rules.pro (ProGuard Rules for app)
 -  gradle.properties (Project Properties)
 -  settings.gradle (Project Settings)
 -  local.properties (SDK Location)

Type d'applications Android

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

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

Intermittente : Application exécutant à la fois des tâches des fonds et permet l'interaction avec l'utilisateur; la communication entre les tâches de fonds et l'utilisateur 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 : Example

```
<?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

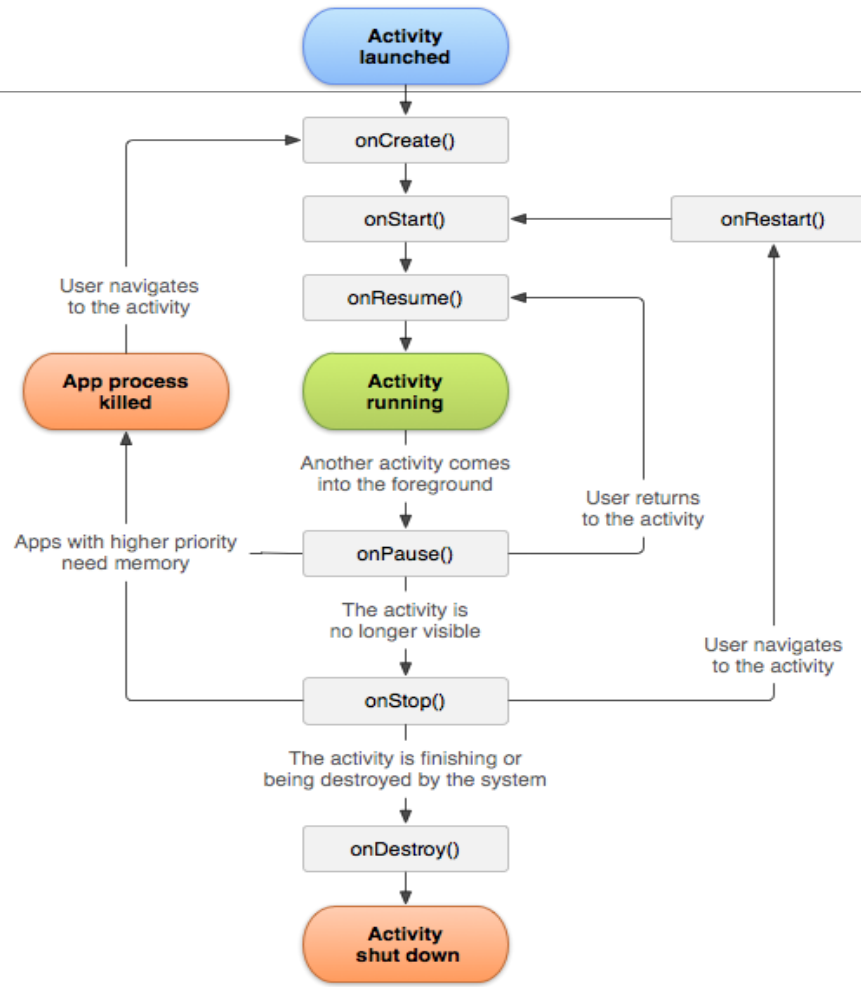
Ne peut être instanciée directement, cette tâche est effectuée par le système



```
Activity a = new Activity();
```

Une activité est instanciée en utilisant les intentions

Activité /cycle de vie



Par défaut, la méthode `onCreate()` est automatiquement surchargée lorsque vous créez une nouvelle activité. Mais comme vous pouvez le voir sur le diagramme ci-dessus, il en existe d'autres, que nous allons découvrir maintenant.

La méthode `onCreate()`

Cette méthode est appelée lorsque votre activité est créée par le système, et qu'elle entre dans l'état `Created`.

Généralement, les opérations effectuées dans cette méthode servent à mettre en place l'interface graphique et à initialiser les données qui seront utilisées par l'interface.

À ce stade, l'activité est créée, mais l'utilisateur ne la voit pas encore et ne peut pas interagir avec.

La méthode `onStart()`

Cette méthode est appelée par le système lorsque l'activité entre dans l'état `Started`.

L'interface graphique devient visible à l'utilisateur, mais il ne peut pas encore interagir avec les différents éléments.

La méthode `onResume()`

Cette méthode est appelée lorsque l'activité entre dans l'état **Resumed**. L'activité devient entièrement opérationnelle. L'utilisateur peut utiliser l'application et cliquer sur les différents éléments graphiques.

L'application reste dans cet état tant qu'il n'y a pas d'interruption, comme par exemple la réception d'un appel téléphonique, le démarrage d'une nouvelle activité ou l'affichage d'une boîte de dialogue.

La méthode `onPause()`

Cette méthode est appelée lorsque l'activité entre dans l'état **Paused**. Une Activity passe par exemple dans l'état **Paused** lorsqu'une **AlertDialog** est affichée : l'Activity sous-jacente est toujours visible, mais elle n'est pas au premier plan. Cette méthode est "l'inverse" de la méthode `onResume()` : tout ce qui est initié dans `onResume()` doit être mis en pause dans cette méthode. Par exemple, une animation présentée à l'utilisateur est démarrée dans `onResume()` puis stoppée dans `onPause()`.

Les traitements effectués dans cette méthode doivent être les plus courts possible.

La méthode `onStop()`

Cette méthode est appelée lorsque l'activité entre dans l'état **Stopped**. Par exemple, lorsqu'une nouvelle activité est démarrée, l'activité appelante va se retrouver dans cet état. Elle n'est donc plus du tout visible à l'utilisateur. Les traitements liés à la mise à jour de l'interface graphique peuvent être arrêtés.

La méthode `onDestroy()`

Cette méthode est appelée lorsque l'activité est arrêtée.

Séchez vos larmes, c'est normal 😊 La première fois, cela effraie toujours un peu. Nous allons mettre tout cela en pratique et vous irez tout de suite mieux.

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 être obligatoirement déclarée dans le Manifest

```
<manifest ... >
```

```
  <application ... >
```

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

```
    ...
```

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

```
  ...
```

```
</manifest >
```

TP sur ordinateur

Suite des tps

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.

Exemple :

```
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 / Évènements

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

TP sur ordinateur

Suite des tps

LinearLayout

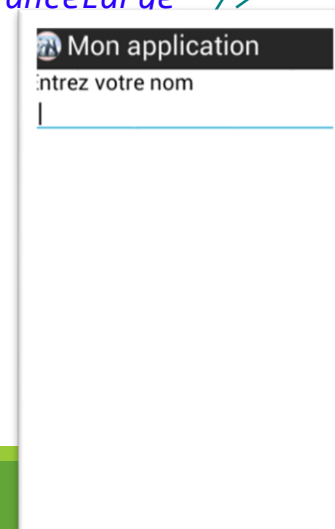
Définit le positionnement linéaire (horizontal ou vertical) des vues filles

```
<?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

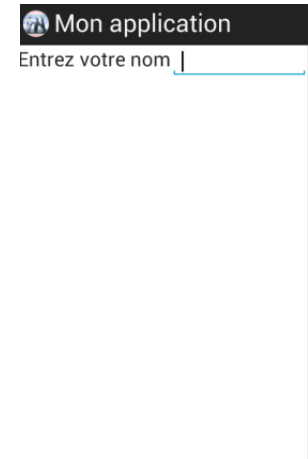
```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal" >

    <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>
```



TP sur ordinateur

Suite des tps

RelativeLayout

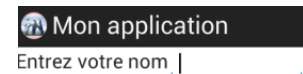
Positionner les éléments de l'interface les uns par rapport aux autres

```
<?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

TP sur ordinateur

Suite des tps

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" />
```

TP sur ordinateur

Suite des tps

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 (Intent)

Il existe deux principaux types d'intentions :

- ➔ **Explicite** : Spécifie les composants qui précisent la classe exacte qui doit être exécutée (`setComponent (nom)` ou `setClass(context, class)`)
- ➔ **Implicite** : Ne spécifie pas le composant mais fournit assez d'informations permettant au système de déterminer 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 (envoi 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);
```

TP sur ordinateur

Suite des tps

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 paramètre 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>
```

TP sur ordinateur

Suite des tps

Lancer une activité et attendre un résultat

La méthode `startActivityForResult(intent)`, permet de demarrer 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 lancé l'activité
- **resultCode**: Indique le statut du résultat (RESULT_OK, ...)
- **Data** : renferme 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 »);`

TP sur ordinateur

Suite des tps

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>

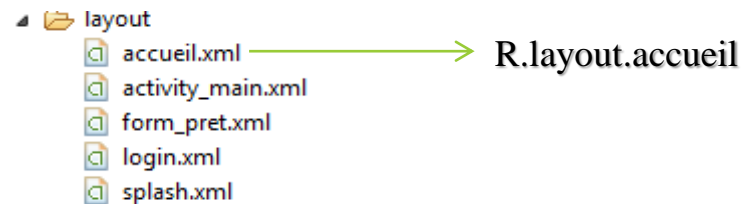
<dimen name="taille">55px</dimen>

</resources>
```

Référencer les ressources

L'ensemble des 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



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

R.string.app_name

TP sur ordinateur

Suite des tps

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/holo_blue_dark"
    android:textAppearance="?android:attr/textAppearanceMedium"
/>
```


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)`

TP sur ordinateur

Suite des tps

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.spinner1);
```

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

```
dataAdapter.setDropDownView
Resource(android.R.layout.simpl
e_spinner_dropdown_item);
```

```
sp.setAdapter(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 d'écran ...)
- 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 " />
```

TP sur ordinateur

Suite des tps

Connectivité

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

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

Pour accéder à ces ressources, il est obligatoire de demander explicitement la permission associée

- `<uses-permission
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,..)

Permission pour 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 toute opération longue s'exécutant dans le thread principal.

L'exécution des tâches 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);  
  
        //Après l'execution du processus long (Thread principal)  
  
    }  
  
    @Override  
  
    protected Void doInBackground(String... arg0) {  
  
        //Execution du processus long (Thread secondaire)  
  
        return null;  
  
    }  
  
}
```

Bien plus encore!

Persistence : SQLite, Préférences

Capteur et Géolocalisation, Camera

Multimédia : Vidéo, Musique,...

Connectivité : Bluetooth, NFC, QR Code

Services

Notifications, BroadCast

Dessin 2D

TP sur ordinateur

Suite des tps

Développement via Jetpack Compose

Voir suite du cours Partie 2