Programmation mobile pour des systèmes Android

Chap2: Création d'applications et découverte des activités

Azyat Abdelilah
azyat- abdelilah.blogspot.com
geossc.ma

Sommaire

- **Composition d'une application**
- **Composants applicatif**
- **Éléments d'interaction**
- **Cycle de vie d'une application**
  - une activité Android
  - Cycle de vie d'une activité
  - Vues
- **Ressources**
  - Utilisation des ressources
  - Créer des ressources
- **Fichier de configuration Android**
- **Personnaliser une application Android**
- **Une première application**
- **Création d'interfaces graphiques**

sommaire

Composition d'une application

Une application → une structure et un cycle de vie

- Activités *(éléments de l'interface graphique);*
- Vues et contrôles;
- Ressources;
- Fichiers de configuration (manifest);

sommaire

Composition d'une application

activité

- Elle peut être assimilée à un **écran** structuré par un ensemble de ***vues*** et de ***contrôles*** composant son interface;
- Elle correspond à la partie ***présentation*** de l'application;
- Elle fonctionne par le biais de vues qui affichent des interfaces graphiques et répondent aux actions utilisateur.

sommaire

Composition d'une application

Vues (Views)

- Ce sont les éléments de l'interface graphique que l'utilisateur voit et sur lesquels il pourra agir.

- Elles contiennent des composants, organisés selon diverses mises en page (les uns à la suite des autres, en grille…).

sommaire

Composition d'une application

Contrôles

- Boutons,
- Champs de saisie,
- Case à cocher, etc.

Ce sont eux-mêmes un sous-ensembles des vues,

- Ils ont besoin d'accéder aux textes et aux images qu'ils affichent,
  - **par exemple :** un bouton représentant un téléphone aura besoin de l'image du téléphone correspondante.

sommaire

Composition d'une application

Ressources

- Ce sont des textes et des images qui sont puisés dans les fichiers de l'application

MainActivity.java × AndroidManifest.xml × activity_main.xml ×

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.application01.admin.myfistrap" >

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="My FistrAp"
        android:theme="@style/AppTheme" >
        <activity
            android:name=".MainActivity"
            android:label="My FistrAp" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

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

</manifest>
```

sommaire

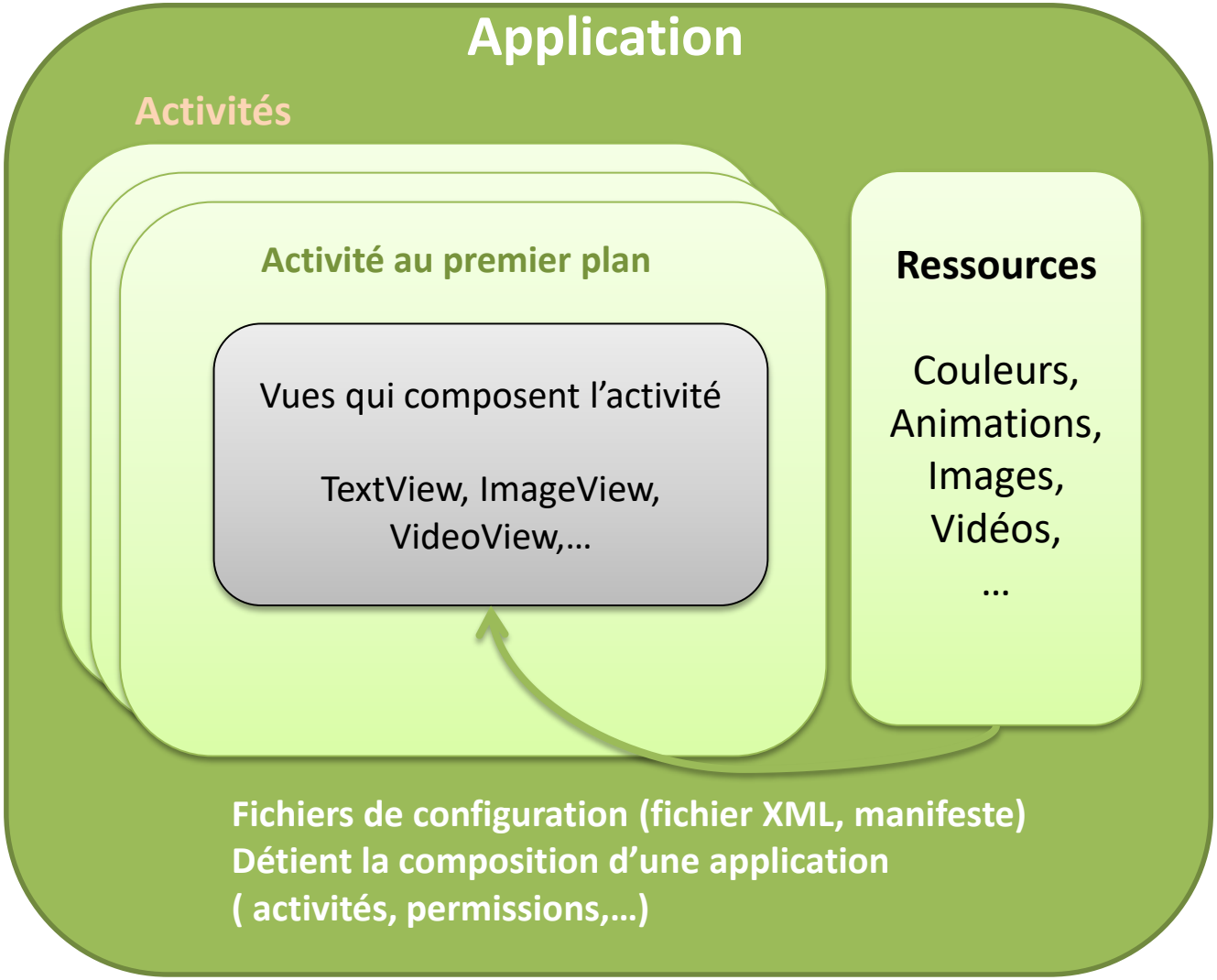
Composition d'une application

Fichier de configuration

- **manifest**: fichier indispensable à chaque application qui décrit entre autres :
    - ➢ point d'entrée de votre application;
    - ➢ composants constituent ce programme ;
    - ➢ permissions nécessaires à l'exécution du programme

sommaire

Composition d'une application



Composants applicatifs

- ***Activity:***
  représente le bloc de base d'une application

- ***Service:***
  un composant qui fonctionne en tâche de fond;
  - ✓ 1 Service pour télécharger l'application
  - ✓ 1 Service pour installer l'application

- ***Fournisseur de contenu:***

  Permet l'**échange de données entre applications**
  Echange contrôlé par un ensemble de **permissions**

- ***Gadgets:***
  Widget : Programmes « visuels » (ie gadget windows)
  un composant graphique qui s'installe sur le bureau Android

Composants applicatif

différents composants applicatifs Android et les classes associées

Nom	Classe ou paquetage concerné(e)
Activité	android.app.Activity
Service	android.app.Service
Fournisseurs de contenu (content provider)	android.content.ContentProvider
Gadgets (app widget)	android.appwidget.*

sommaire

Éléments d'interaction

- **Intents,**
- **Récepteurs (Broadcast Receiver),**
- **Notifications,**

sommaire

Éléments d'interaction

- **Intents**
  - ils permettent de diffuser des messages en demandant la réalisation d'une action,
  - L'accès aux autres applications et au système étant restreinte par le modèle de sécurité Android,

sommaire

Éléments d'interaction

- **Récepteur d'Intents**

il permet à une application d'être à l'écoute des autres,

afin de répondre aux objets *Intent* qui lui sont destinés et qui sont envoyés par d'autres composants applicatifs.

sommaire

Éléments d'interaction

- **Notification**:
  - Signale une information à l'utilisateur sans interrompre ses actions en cours.
  - gérer et partager des informations; gestion des annonces (batterie, fin de téléchargement ...) générées par l'OS,
  - Détecter la fin d'un téléchargement d'application
  - Différentes types de notifications :
    - **Toast Notification**
      - Message immédiat (bulle)
      - Simple d'utilisation
    - **Status Bar Notification**
      - Message + icône (barre des notifications)
      - Utiliser dans les tâches de fond (Service)
    - **Dialog Notification**
      - Boîte de dialogue ...
      - AlertDialog, ProgressBar

sommaire

Éléments d'interaction

Nom	Classe concernée
Intent	**android.content.Intent**
Récepteur d'Intents (**Broadcast Receiver**)	**android.content.BroadcastReceiver**
Notification	**android.app.Notification**

sommaire

Cycle de vie d'une application

gestion des processus

- Les applications Android ont un fonctionnement particulier : elles réagissent à des *changements d'états* imposés par le système (démarrage, pause, reprise, arrêt...).
- Chaque application fonctionne dans son propre processus.
- Le système Android est responsable de la création et de la destruction des processus et gère ses ressources
  - avec comme objectif la sécurité mais aussi la disponibilité et la réactivité de l'appareil.

sommaire

Cycle de vie d'une application

gestion des processus

- L'ordre d'arrêt des processus pour libérer des ressources est déterminé par la priorité du processus donnée par le système.

- La priorité de l'application est elle-même déterminée par son composant le plus prioritaire

  (une application possédant un fournisseur de contenu a par exemple plus de priorité qu'une application qui en est dépourvue).

sommaire

Cycle de vie d'une application

Priorités établies par le système en fonction de l'état de ses composants

Processus	Priorité	Description
1- Processus actif	Maximale	Processus au premier plan qui héberge des applications dont des composants interagissent avec l'utilisateur.
2- Processus visible	Élevée	Processus visible mais inactif, il contient des activités au moins en partie visibles mais pas au premier plan.
3- Processus en tâche de fond	Faible	Processus hébergeant des activités non visibles ou des services non démarrés.
4- Processus en fin de vie	Minimale	Android dispose d'un cache d'applications qui ont terminé leur cycle de vie, à des fins d'optimisation lors de leur redémarrage.

sommaire

Cycle de vie d'une application

Une activité Android

- Une activité peut être assimilée à un écran qu'une application propose à son utilisateur.

chaque écran

doit contenir une activité.

sommaire

Une activité Android

- Une activité est composée de deux volets :
  - la **logique de l'activité** et la gestion du cycle de vie de l'activité qui sont implémentés en Java dans une classe héritant de Activity;
  - **l'interface utilisateur**, qui pourra être définie soit dans le code de l'activité soit de façon plus générale dans un fichier XML placé dans les ressources de l'application.

sommaire

Une activité Android

- **Création d'un écran**

Chaque écran est géré par une instance de Activity. Sa méthode **onCreate** définit ce qui doit être affiché sur l'écran :

```
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
```

sommaire

Cycle de vie d'une activité

diagramme de principaux étapes

onResume: active

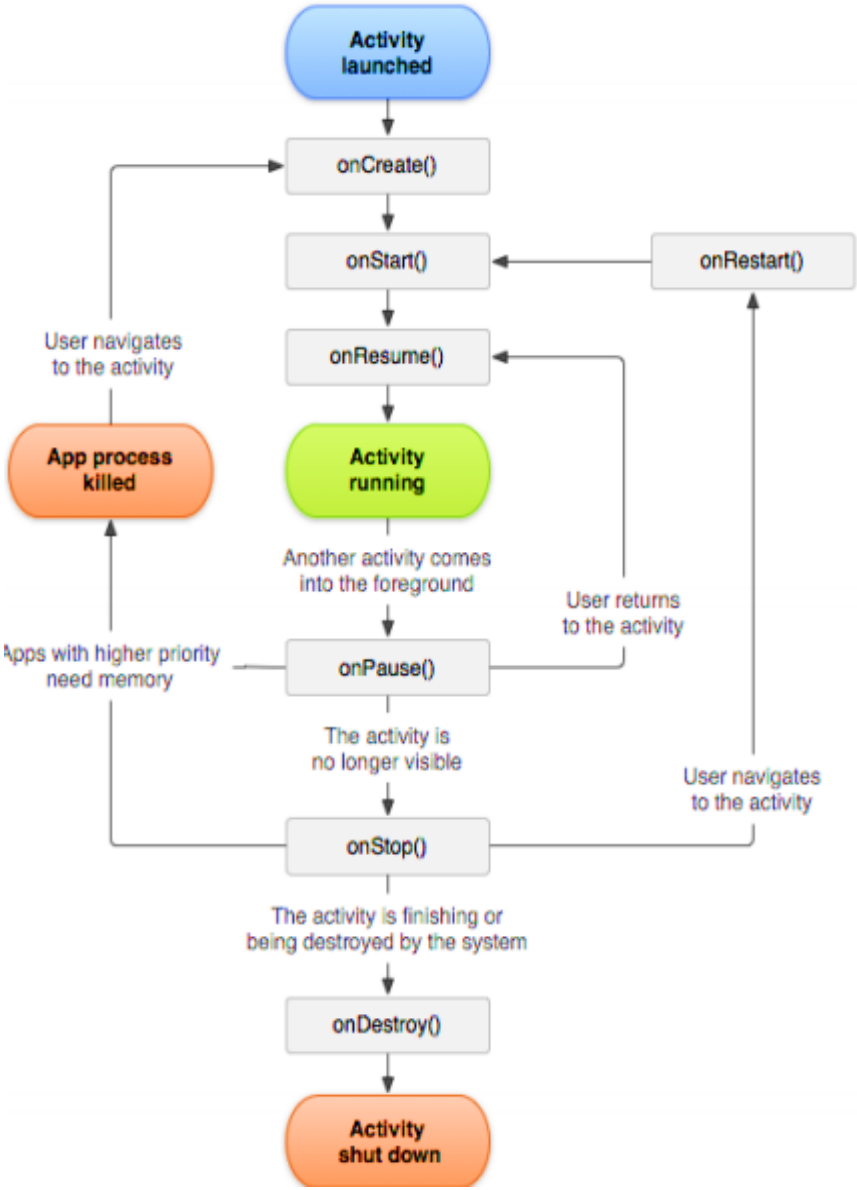
onPause: mettre en pause quand une autre activité devient active;

suspendue (onStop) :

activité est totalement cachée,

Pas de code exécuté,

Ses informations sont conservées,



sommaire

Cycle de vie d'une activité

- Android exécute le code d'une activité en appelant des *callbacks,*
- Ces callbacks correspondent aux phrases de la vie d'une activité,
- Il n'est pas nécessaire d'implémenter toutes les callbacks

sommaire

Cycle de vie d'une activité

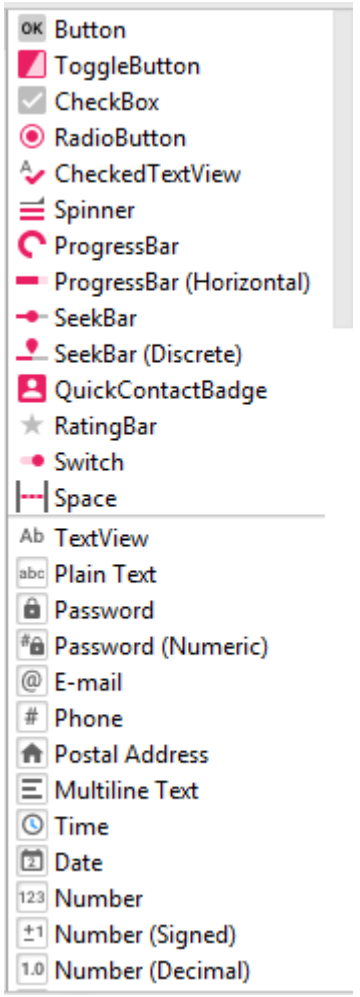
Les callbacks

```
public class ExampleActivity extends Activity {

    @Override
    public void onCreate(Bundle savedInstanceState) { /* ... */ }

    @Override
    protected void onStart() { /* ... */ }

    @Override
    protected void onResume() { /* ... */ }

    @Override
    protected void onPause() { /* .... */ }

    @Override
    protected void onStop() { /* ... */ }

    @Override
    protected void onDestroy() { /* ... */ }

    }
```

sommaire

Vues (Views)

- Elles sont les briques de construction de l'interface graphique d'une activité Android.
- Les objets View représentent des éléments à l'écran qui permettent d'interagir avec l'utilisateur via un mécanisme d'événements.
- Elles peuvent être disposées dans une activité et donc à l'écran soit par
  - une description XML,
  - un morceau de code Java.



sommaire

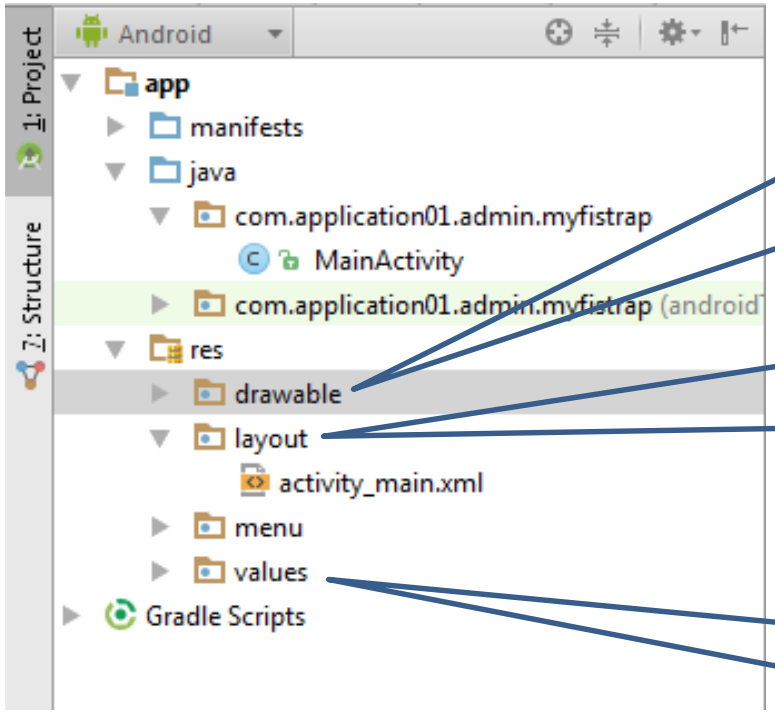
Structure d'un fichier XML

- Espaces de nommage dans un fichier XML
- Dans le cas d'Android, il y a un grand nombre d'éléments et d'attributs normalisés.
- Les attributs ont été regroupés dans un namespace (**xmlns**). Leur nom est android:nomattribut
- **xmlns:android="http://schemas.android.com/apk/res/android"**

sommaire

Ressources

- Les ressources sont des fichiers externes – ne contenant pas d'instruction – qui sont utilisés par le code et liés à une application au moment de sa construction.



drawable: Fichiers .png, .jpeg qui sont convertis en bitmap ou .9.png qui sont convertis en "9-patches" c'est-à-dire en images ajustables.

layout: Fichiers XML convertis en mises en page d'écrans (ou de parties d'écrans), que l'on appelle aussi gabarits.

values: Fichiers XML convertis en différents types de ressources (array, string,…).

sommaire

Utilisation des ressources

- Toutes Les ressources sont placées, converties ou non, dans un fichier de type **APK** qui constituera le programme distribuable de votre application.
- De plus, Android crée une classe nommée **R** qui sera utilisée pour se référer aux ressources dans le code.
- Les ressources qu'on créez ou qu'Android propose peuvent être utilisées directement dans une application ou être référencées dans d'autres ressources.

sommaire

Utilisation des ressources

La classe R

- Elle est générée automatiquement par Android,
- Elle permet de désigner les ressources dans le code Java,

  Exemple: **activity_main.xml = R.layout.activity_main**

- Générée à la compilation du projet,
- R.java contient tous les éléments GUI qui ont une ***id***,

- Permet le lien entre XML et Objet JAVA
  – **findViewById(int id):** récupère l'objet java décrit par un XML

sommaire

Utilisation des ressources
La classe R

- **R.id. ressource_type**
  - ressource_type: string, anim, layout, …
  - Retrouver un élément par son Id: R.id.xxxxxx,
  - Id fonctionne sur n'importe quelle type de ressource nommée;

sommaire

Utilisation des ressources

- Pour utiliser une ressource, il suffit de connaître son type et son identifiant.

- La syntaxe est alors la suivante :

  android.R.type_de_ressource.nom_ressource

- **Exemple:**

```
// Fixe la mise en page d'une activité
setContentView(R.layout.ecran_de_demarrage);
```

Utilisation des ressources

Programme et ressources

- Il est souhaitable de stocker l'interface dans un fichier res/layout/main.xml :

```
<RelativeLayout ...>
<TextView android:text="Bonjour !" ... />
.....
</RelativeLayout>
```

qui est référencé par son identifiant **R.layout.nom_du_fichier** dans le programme Java :

```
public class MainActivity extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }
}
```

sommaire

Ressources référencées par d'autres ressources

- On peut utiliser notre ressources comme valeurs d'attributs dans d'autres ressources sous forme XML.
- Cette possibilité est très utilisée dans les mises en page par exemple.
- La notation pour faire référence à une autre ressource est la suivante:

attribute="@resource_type/resource_identifier "

Exemple d'une disposition d'écran *: mise en page sous forme de table (d'une ligne et de deux colonnes),*

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

les cellules sont remplies par des chaînes de caractères

sommaire

Créer des ressources

Valeurs simples

- Il est possible de déclarer dans des fichier XML,
    - ➢ des chaînes de caractères,
    - ➢ des dimensions,
    - ➢ des couleurs,
    - ➢ des tableaux (de chaînes de caractères ou d'entiers) comme valeurs simples.

sommaire

Créer des ressources

Définition d'un fichier de ressources contenant plusieurs types

```
<?xml version="1.0" encoding="utf-8"?> la déclaration XML
<resources>
    <string name="nom_application">Suivi des SMS</string>
    <color name="couleur_des_boutons">#FFAABBCC</color>
    <array name="tableau_entiers">
        <item>1765</item>
        <item>3</item>
    </array>
</resources>
```

sommaire

Créer des ressources

Couleurs

- Une couleur définit une valeur RVB (rouge, vert et bleu) et une transparence.

- Il existe différents formats dont la syntaxe globale est la suivante :

  <color name=nom_couleur>valeur_de_la_couleur</color>

Définition d'un fichier de ressources de couleurs:

```
<resources>
<color name="bleu_transparent">#50ff00FF</color>
</resources>
```

Utilisation des couleurs (exemple avec la déclaration de la couleur bleu_transparent):

- **Java : R.color.bleu_transparent**
- **XML : @color/bleu_transparent**

sommaire

Créer des ressources

Chaînes de caractères et texte formaté

- Les chaînes de caractères avec un format optionnel peuvent être utilisées comme ressources.

syntaxe : **<string name=nom_chaine>valeur_de_la_chaine</string>**

- utilisation de trois balises HTML standard (**gras**), <i> (*italique*) et <u> (souligné) :

  <string name="ex1">Un texte mis en forme </string>

- si vous voulez utiliser des guillemets ou des apostrophes, vous devez les échapper en les faisant précéder du caractère slash ('\'):

  <string name="ex2">Voici l\'application de ce conseil</string>

- exemples de déclaration de chaînes de caractères dans un fichier de ressources de chaînes de caractères

```
<resources>
    <string name="app_name">Exemple Android Eyrolles</string>
    <string name="menu_principal">Menu Principal</string>
</resources>
```

sommaire

Créer des ressources

Unités de mesure (« dimensions »)

- Les dimensions sont la plupart du temps référencées dans les styles et les mises en page.
- Elles peuvent servir de constantes entre différents éléments d'écrans.

- unités prises en charge par Android :
  - **px** (pixels);: correspond à un pixel de l'écran,
  - **in** (pouces): unité de longueur, correspondant à 2,54 cm. Basé sur la taille physique de l'écran,
  - ***dp*** (density-independant pixel): se basant sur une taille physique de l'écran de 160 dpi. Avec cette unité, 1 dp est égal à 1 pixel sur un écran de 160 pixels ,
  - **sp** (scale-independant pixel): fonctionne de la même manière que les pixels à densité indépendante à l'exception qu'ils sont aussi fonction de la taille de polices spécifiée par l'utilisateur.

- Définition d'un fichier de ressources de dimensions:

```
<resources>
    <dimen name="taille_texte">5sp</dimen>

</resources>
```

sommaire

Le fichier de configuration Android

- Chaque application Android nécessite un fichier de configuration :

  AndroidManifest.xml

- Ce fichier est placé dans le répertoire de base du projet, à sa racine.
- Il décrit le contexte de l'application,
  - les activités,
  - les services,
  - les récepteurs d'Intents (Broadcast receivers),
  - les fournisseurs de contenu
  - les permissions.

sommaire

Le fichier de configuration Android

Structure du fichier de configuration

- Un fichier de configuration est composé d'une racine (le tag **manifest**) et d'une suite de nœuds enfants qui définissent l'application.

- **Exemple**: Structure vide d'un fichier de configuration d'une application

```
<manifest
xmlns:android=http://schemas.android.com/apk/res/android
package="fr.domaine.application">
</manifest>
```

sommaire

Démonstration: Une premier application

sommaire

Dispositions de Layouts et Views

Structure d'une interface Android

- Les éléments graphiques héritent de la classe **View**.
- On peut regrouper des éléments graphiques dans une **ViewGroup**.
- Des **ViewGroup** particuliers sont prédéfinis: ce sont des gabarits (***Layouts***) qui proposent une prédispositions des objets graphiques:

  - **LinearLayout**
  - ***RelativeLayout***
  - **TableLayout**
  - **FrameLayout**
  - **GridLayout**
  - **ConstraintLayout**

sommaire

Dispositions de Layouts et Views

Structure d'une interface Android

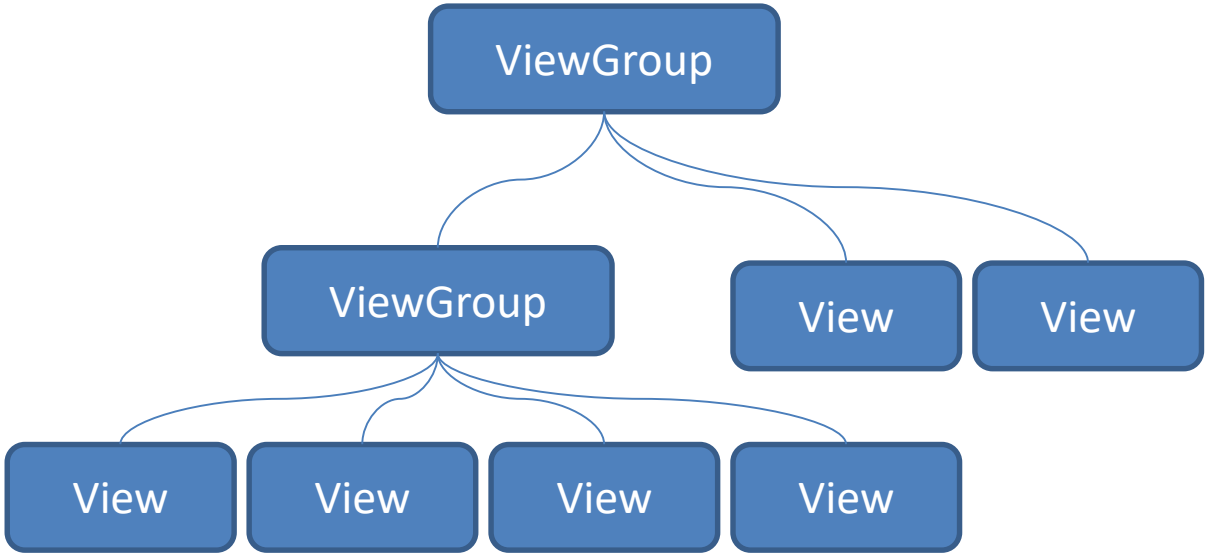
- **LinearLayout (vertical/horizontal):** positionne ses vues en ligne ou colonne,
- ***RelativeLayout:*** positionne ses vues l'une par rapport à l'autre, et par rapport aux bords du RelativeLayout
- **TableLayout:** positionne ses vues sous forme d'un tableau,
- **FrameLayout:** disposition en haut à gauche en empilant les éléments,
- **GridLayout:** disposition matricielle avec N colonnes et un nombre infini de lignes,
- **ConstraintLayout** dispose de toutes les fonctionnalités d'un RelativeLayout, et peut donc être utilisé de la même manière.

sommaire

Dispositions de Layouts et Views

Organisation hiérarchique des vues

- la classe ViewGroup (hérite de View) permet de regrouper des vues

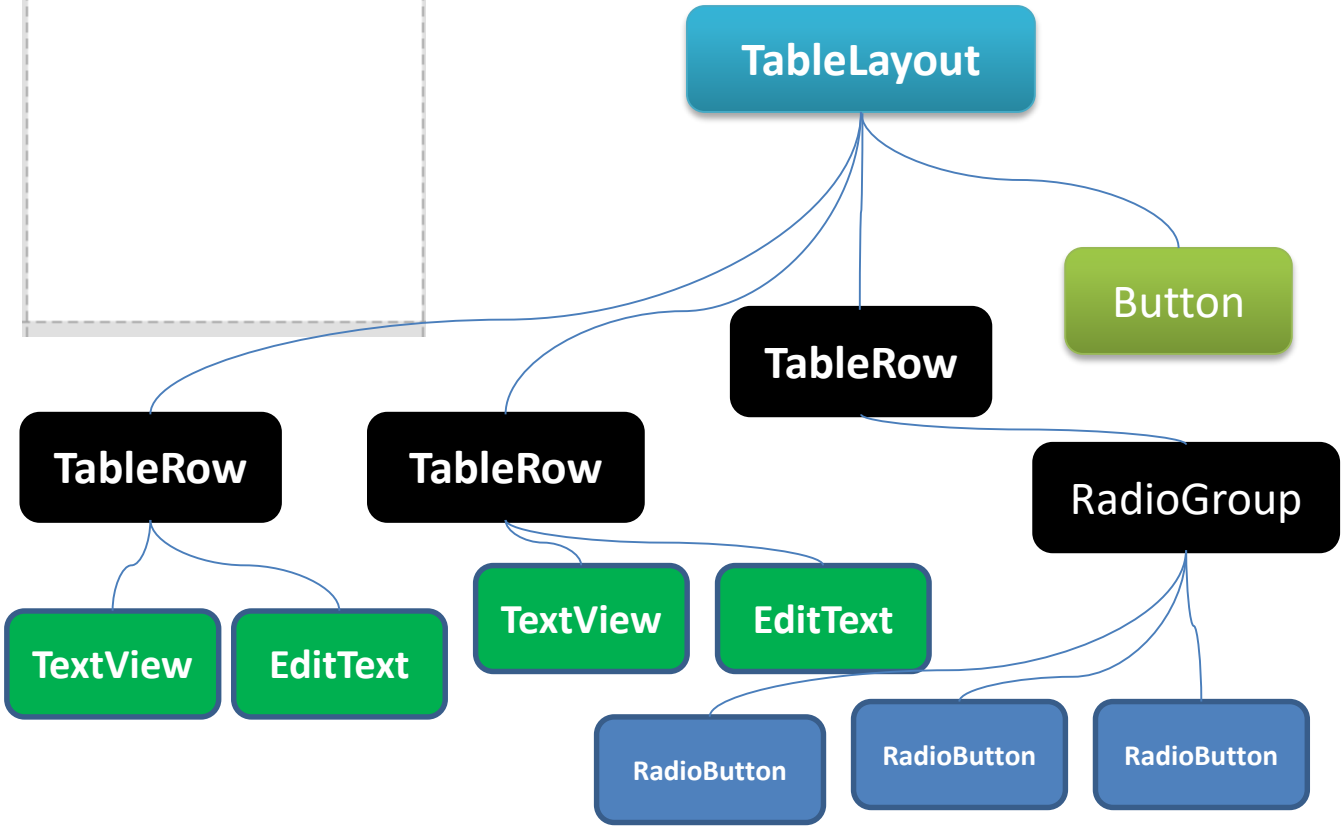


sommaire

Dispositions de Layouts et Views

Structure d'une interface Android

Les groupes et vues forment un arbre



sommaire

Dispositions de Layouts et Views

Représentation en XML

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/activity_etabli" android:layout_width="match_parent"
    android:layout_height="match_parent" android:stretchColumns="1"
    android:background="#ffffffff"
    tools:context="com.example.admin.etablissements.Etablissement_Activity">

    <TableRow>
        <TextView android:text="Name:" android:textSize="20dp" />
        <EditText android:id="@+id/name" android:background="#ffffff77"/>
    </TableRow>

    <TableRow>
        <TextView android:text="Address:" android:textSize="20dp"/>
        <EditText android:id="@+id/addr" android:background="#ffffff88" />
    </TableRow>
    <TableRow>
        <TextView android:text="Type:" android:textSize="20dp" />
        <RadioGroup android:id="@+id/types" >
            <RadioButton android:id="@+id/ingenieurie" android:text="Ingenieurie" />
            <RadioButton android:id="@+id/science_technique"
                android:text="Science_Technique" />
            <RadioButton android:id="@+id/droit_economie" android:text="Droit_Economie" />
        </RadioGroup>
    </TableRow>

    <Button android:id="@+id/save" android:layout_width="fill_parent"
        android:layout_height="wrap_content" android:text="Save"/>
</TableLayout>
```

Dispositions de Layouts et Views

Paramètres de disposition

- La plupart des groupes utilisent des paramètres de placement sous forme d'attributs XML.
- Par exemple, telle vue à droite de telle autre, telle vue la plus grande possible, telle autre la plus petite.
- Ces paramètres sont de deux sortes :
  - ceux qui sont demandés pour toutes les vues,

    android:layout_width,
    android:layout_height
    android:layout_weight

  - ceux qui sont demandés par le groupe englobant et qui en sont spécifiques, comme:

    android:layout_alignParentBottom,
    android:layout_centerInParent. . .

sommaire

Dispositions de Layouts et Views

Paramètres généraux

- Toutes les vues doivent spécifier les deux attributs suivants:

android:layout_width = largeur de la vue
android:layout_height = hauteur de la vue

android:orientation: définit l'orientation d'empilement
android:gravity: définit l'alignement des éléments

- Ils peuvent valoir :

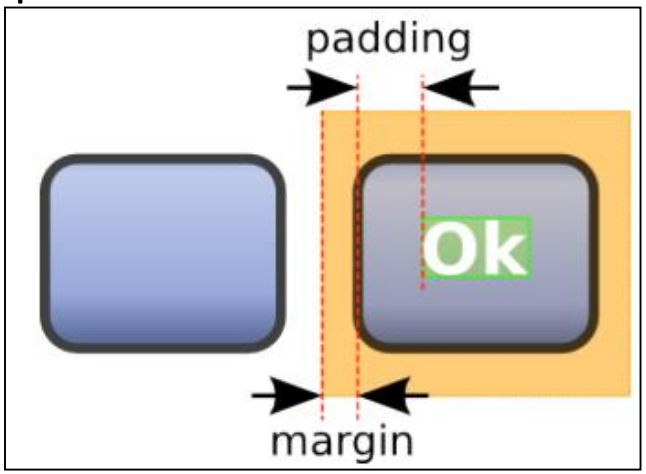
="**wrap_content**" : la vue est la plus petite possible
="**match_parent**" : la vue est la plus grande possible
="**fill_parent**": comme **match_parent**

"valeurdp" : une taille fixe, ex : "100dp" mais c'est peu recommandé

Dispositions de Layouts et Views

Autres paramètres géométriques

- Il est possible de modifier l'espacement des vues :
  - **Padding:** espace entre le texte et les bords, géré par chaque vue,
  - **Margin:** espace autour des bords, géré par les groupes,
  - On peut définir les marges et les remplissages séparément sur chaque bord (Top, Bottom, Left, Right), ou identiquement sur tous :



sommaire

Dispositions de Layouts et Views

paramètres géométriques: exemple

-

```
<Button
    android:text="Add Data"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="0.08"
    android:layout_margin="20dp"
    android:paddingLeft="20dp"
    android:background="#ff11ffff"
    android:id="@+id/button3"
    android:textSize="19sp" />
```

Views

Les évènements

- Chaque élément (ViewGroup et View) peut (ou non) implémenter des interfaces évènementielles – par ex. ***View.OnClickListener***

- On ajoute des écouteurs aux vues pour écouter les évènements :

```
 Button saveBut= (Button) findViewById(R.id.save); //récuperation du
bouton saveBut
        saveBut.setOnClickListener(onSave);//positionnons un listner sur
ce bouton
    }
//méthode déclanchée par appui sur onSave
    View.OnClickListener onSave= new View.OnClickListener(){
        @Override
        public void onClick(View view) {
                EditText name = (EditText)findViewById(R.id.name);
                EditText adress = (EditText) findViewById(R.id.addresse);

                etabUniv.setName(name.getText().toString());
                etabUniv.setAdress(adress.getText().toString());
        }
    };
```

sommaire

Views

Les évènements - Les écouteurs

- La méthode setOnClickListener prend en paramètre un objet qui implémente l'interface View.OnClickListener :

```
public interface OnClickListener {
    void onClick(View view);
}
```

- Une interface est associée à chaque type d'évènements
- On observe les évènements en connectant des listeners Pour cela, on utilise les méthodes **setOn*Listener**
- Ensuite, un évènement entraîne l'appel d'une méthode de l'interface sur l'objet que vous avez connecté à la vue

sommaire

Views

Implémentation avec une classe classique

```
class MyOnClickListener implements View.OnClickListener
{
        private MainActivity mainActivity;

        public MyOnClickListener(MainActivity mainActivity) {
            this.mainActivity = mainActivity;
         }
@Override
        public void onClick(View v) {

        mainActivity.count++;
        }
}
```

Views

Implémentation avec une classe interne

```
public class MainActivity extends Activity {
        public int count = 0;
        @Override
        protected void onCreate(Bundle savedInstanceState)
        {
        /* ... */
        button.setOnClickListener(new MyOnClickListener());
        /* ... */
        }
        class MyOnClickListener implements View.OnClickListener
        {
        @Override
        public void onClick(View v)
                { count++; }

        }
}
```