

RÉSUMÉ THÉORIQUE – FILIÈRE DÉVELOPPEMENT DIGITAL

Option – Applications mobiles

M205 – Développer des interfaces utilisateurs sous Android

18 heures

SOMMAIRE



1. **Créer des interfaces graphiques simples**
   - Implémenter des éléments graphiques
   - Utiliser les Layouts
   - Gérer les évènements
2. **Créer des interfaces graphiques avancées**
   - Coder des custom view
   - Gérer des animations entre les activités
   - Créer des effets d'animations

MODALITÉS PÉDAGOGIQUES



1

LE GUIDE DE SOUTIEN

Il contient le résumé théorique et le manuel des travaux pratiques



2

LA VERSION PDF

Une version PDF est mise en ligne sur l'espace apprenant et formateur de la plateforme WebForce Life



3

DES CONTENUS TÉLÉCHARGEABLES

Les fiches de résumés ou des exercices sont téléchargeables sur WebForce Life



4

DU CONTENU INTERACTIF

Vous disposez de contenus interactifs sous forme d'exercices et de cours à utiliser sur WebForce Life

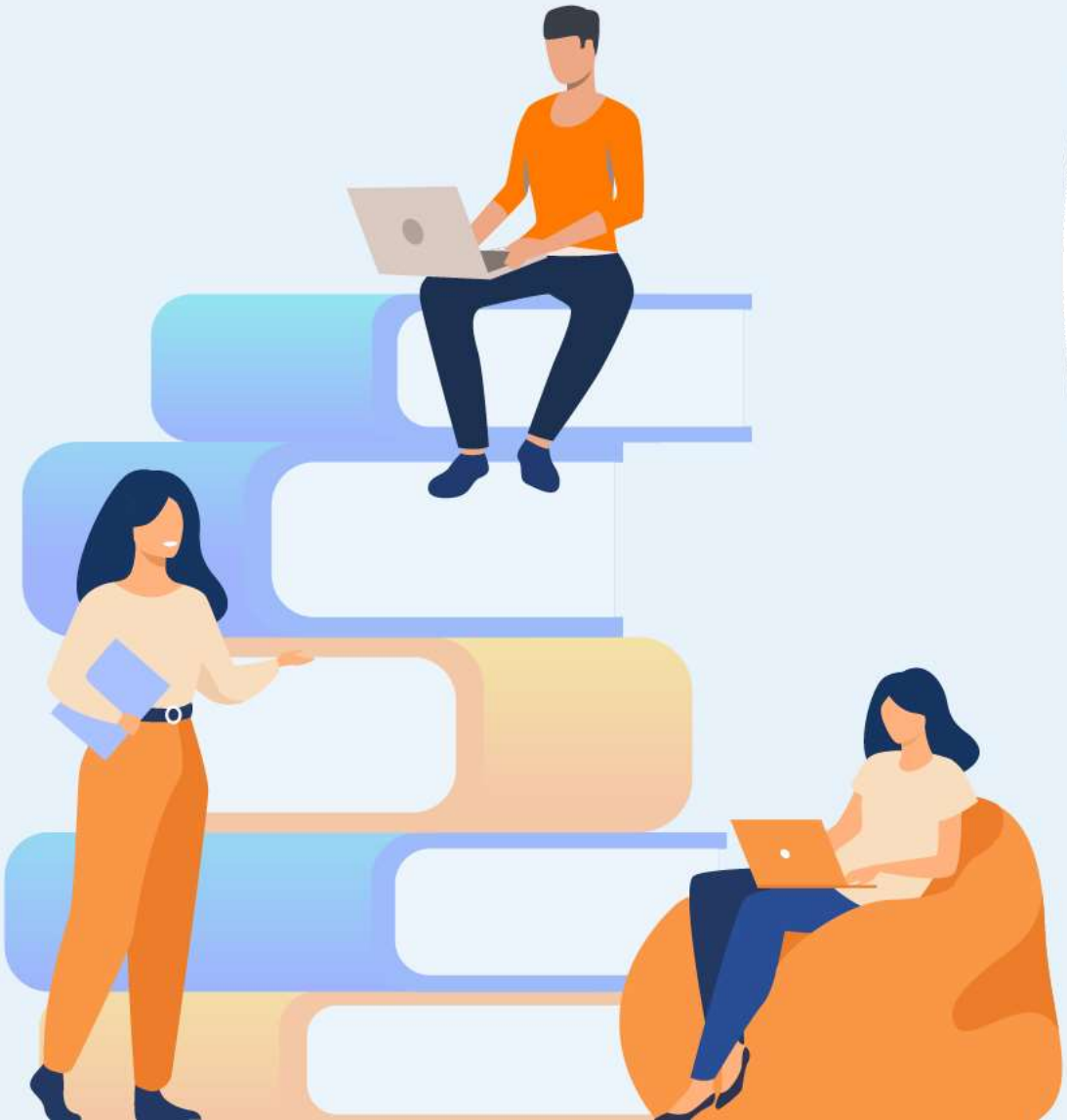


5

DES RESSOURCES EN LIGNES

Les ressources sont consultables en synchrone et en asynchrone pour s'adapter au rythme de l'apprentissage

Copyright - Tout droit réservé - OFPPT



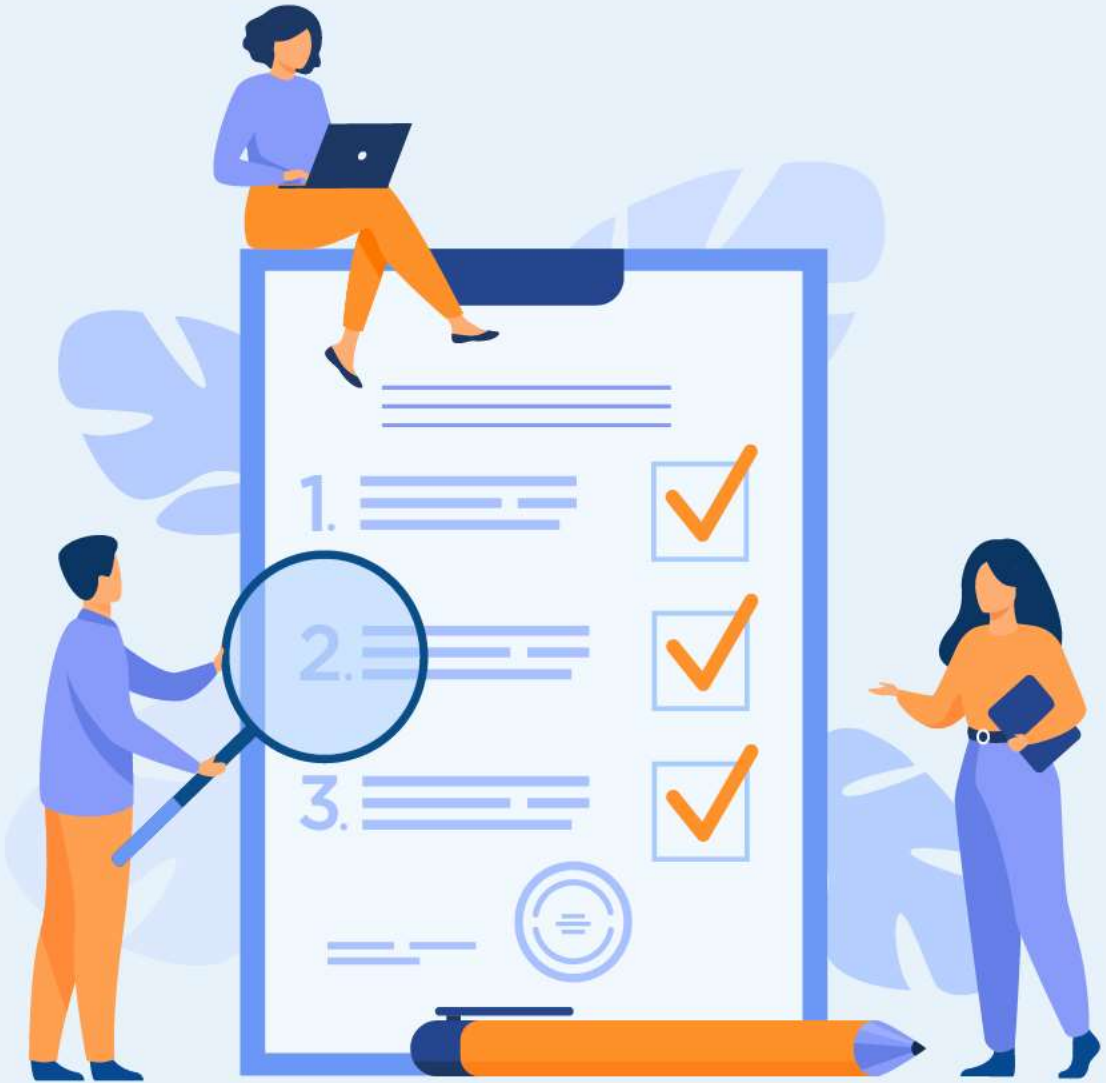
PARTIE 1

Créer des interfaces graphiques simples

Dans ce module, vous allez :

- Création facile de plusieurs interfaces graphiques
- Création correcte d'interfaces utilisant divers layouts

9 heures



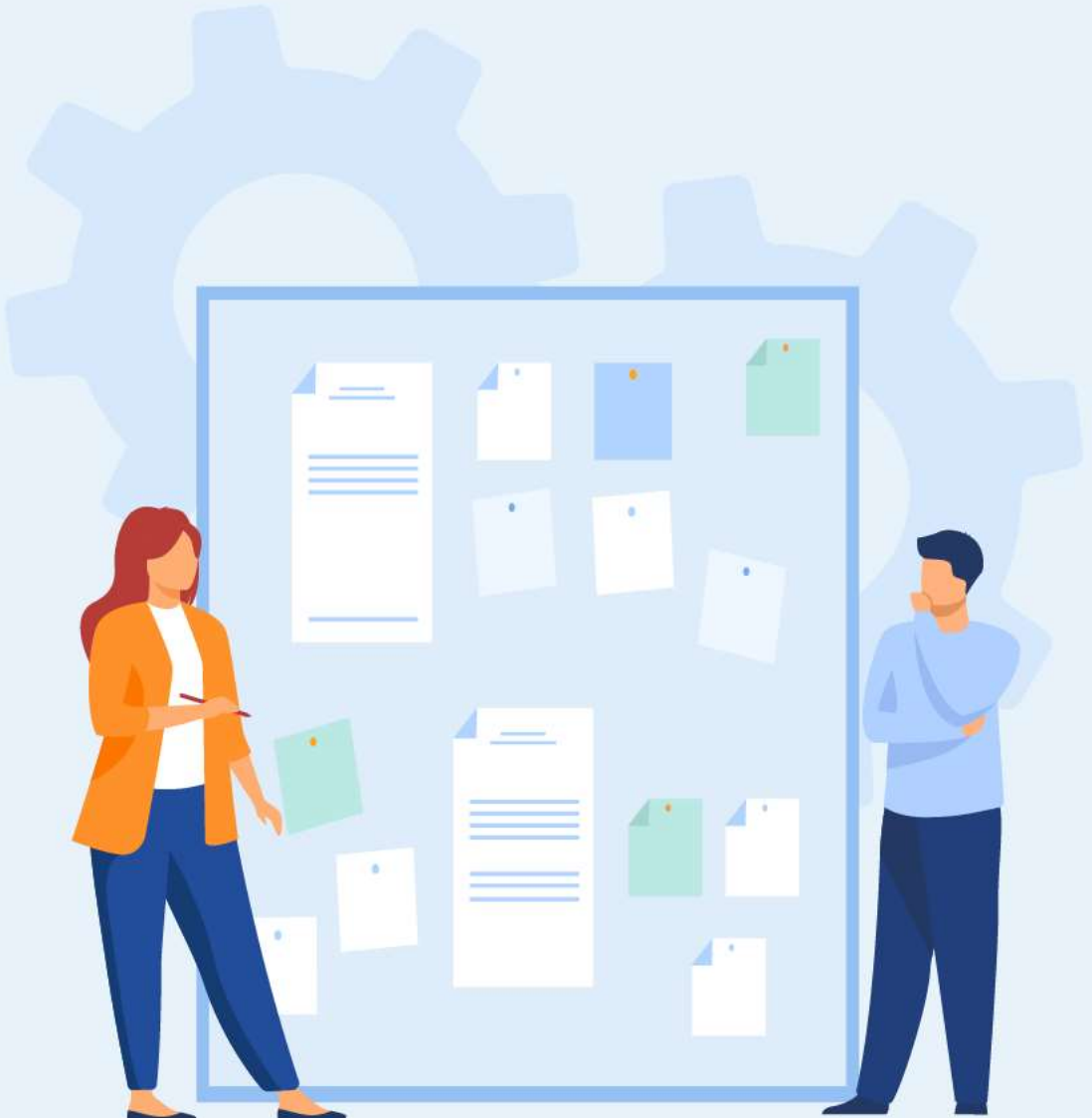
CHAPITRE 1

Implémenter des éléments graphiques

Ce que vous allez apprendre dans ce chapitre :

- Définition des Widgets
- Présentation des Widgets avancés

6 heures



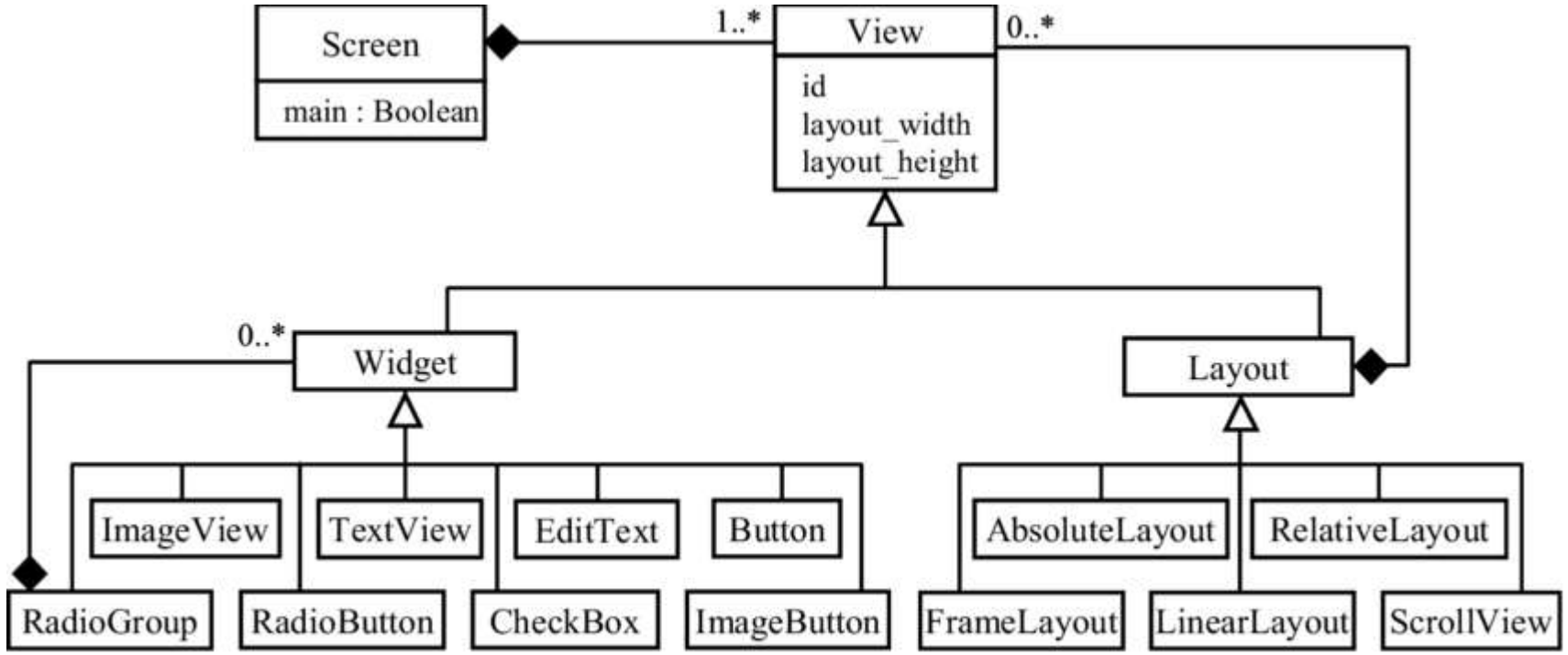
CHAPITRE 1

Implémenter des éléments graphiques

1. **Définition des Widgets**
2. Présentation des Widgets avancés

Méta-modèle d'une application Android

- Une application mobile est composée de plusieurs écrans (**Screen**), chaque écran est composé de plusieurs **View** ;
- Une **View** peut être un **Widght** ou un **Layout**.



Copyright - Tout droit réservé - OFPPT

Définition des widgets

Un widget est un élément d'une interface utilisateur graphique (GUI) qui affiche des informations ou fournit un moyen spécifique pour un utilisateur d'interagir avec le système d'exploitation ou une application.

Les widgets comprennent des icônes, des menus déroulants, des boutons, des boîtes de sélection, des indicateurs de progression, des cases à cocher, des barres de défilement, des fenêtres, des bords de fenêtre (qui vous permettent de redimensionner la fenêtre), des boutons de basculement, des formulaires et de nombreux autres dispositifs permettant d'afficher des informations et d'inviter, d'accepter et de répondre aux actions de l'utilisateur.

En programmation, le terme "widget" désigne également le petit programme qui est écrit afin de décrire l'aspect d'un widget particulier, son comportement et sa réponse aux actions de l'utilisateur. La plupart des systèmes d'exploitation comprennent un ensemble de widgets prêts à l'emploi qu'un programmeur peut incorporer dans une application, en spécifiant comment elle doit se comporter. De nouveaux widgets peuvent être créés.

Copyright - Tout droit réservé - OFPPT

TextView pour le texte

- TextView est une sous-classe de View pour du texte à une ou plusieurs lignes ;
- EditText est une sous-classe de TextView avec du texte modifiable ;
- Contrôlé par des attributs de mise en page (layout) ;
- Définir le texte :
    - statiquement à partir d'une ressource de type chaîne de caractères en XML;
    - dynamiquement à partir du code Java et de toute autre source.

Copyright - Tout droit réservé - OFPPT

Formatage du texte dans une ressource de type chaîne

- Utiliser les balises HTML <b>et <i>pour les caractères gras et italiques ;
- Toutes les autres balises HTML sont ignorées ;
- Ressources en chaînes : une ligne continue = un paragraphe ;
- \n commence une nouvelle ligne ou un nouveau paragraphe ;
- Échapper les apostrophes et les guillemets avec une barre oblique inverse (\", \’);
- Échapper tout caractère non ASCII avec une barre oblique inverse (\).

Copyright - Tout droit réservé - OFPPT

Création d'un TextView en XML

```
<TextView android:id="@+id/textview"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="@string/my_story"/>
```

Attributs communs des TextView

- **android:text** : texte à afficher ;
- **android:textColor** : couleur du texte ;
- **android:textAppearance** : style ou thème prédéfini ;
- **android:textSize** : taille du texte en sp ;
- **android:textStyle** : normal, bold, italic ou bold|italic ;
- **android:Typeface** : normal, sans, serif ou monospace ;
- **android:lineSpacingExtra** : extra espace entre les lignes en sp.

Copyright - Tout droit réservé - OFPPT

Formatage des liens web actifs

```
<string name="article_text">... www.rockument.com ...</string>

<TextView
   android:id="@+id/article"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:autoLink="web"
   android:text="@string/article_text"/>
```



Remarques

N'utilisez pas le langage HTML pour un lien web dans un texte libre.

Les valeurs de autoLink :"web", "email", "phone", "map", "all"

PARTIE 1

Copyright - Tout droit réservé - OFPPT

Création de TextView en code Java

```
TextView myTextview = new TextView(this);

myTextView.setWidth(LayoutParams.MATCH_PARENT);

myTextView.setHeight(LayoutParams.WRAP_CONTENT);

myTextView.setMinLines(3);

myTextView.setText(R.string.my_story);

myTextView.append(userComment);
```


Copyright - Tout droit réservé - OFPPT

Qu'en est-il d'importantes quantités de texte ?

- Actualités, articles, etc...
- Pour faire défiler un TextView, il faut l'intégrer dans un ScrollView ;
- Un seul élément View (généralement TextView) est autorisé dans un ScrollView ;
- Pour faire défiler plusieurs éléments, utilisez un ViewGroup (tel que LinearLayout) dans le ScrollView.

ScrollView pour le défilement du contenu

- ScrollView est une sous-classe de FrameLayout ;
- Garde tout le contenu en mémoire ;
- Ne convient pas aux textes longs et aux mises en page complexes ;
- Ne pas imbriquer plusieurs vues de défilement ;
- Utilisez HorizontalScrollView pour le défilement horizontal ;
- Utilisez un RecyclerView pour les listes

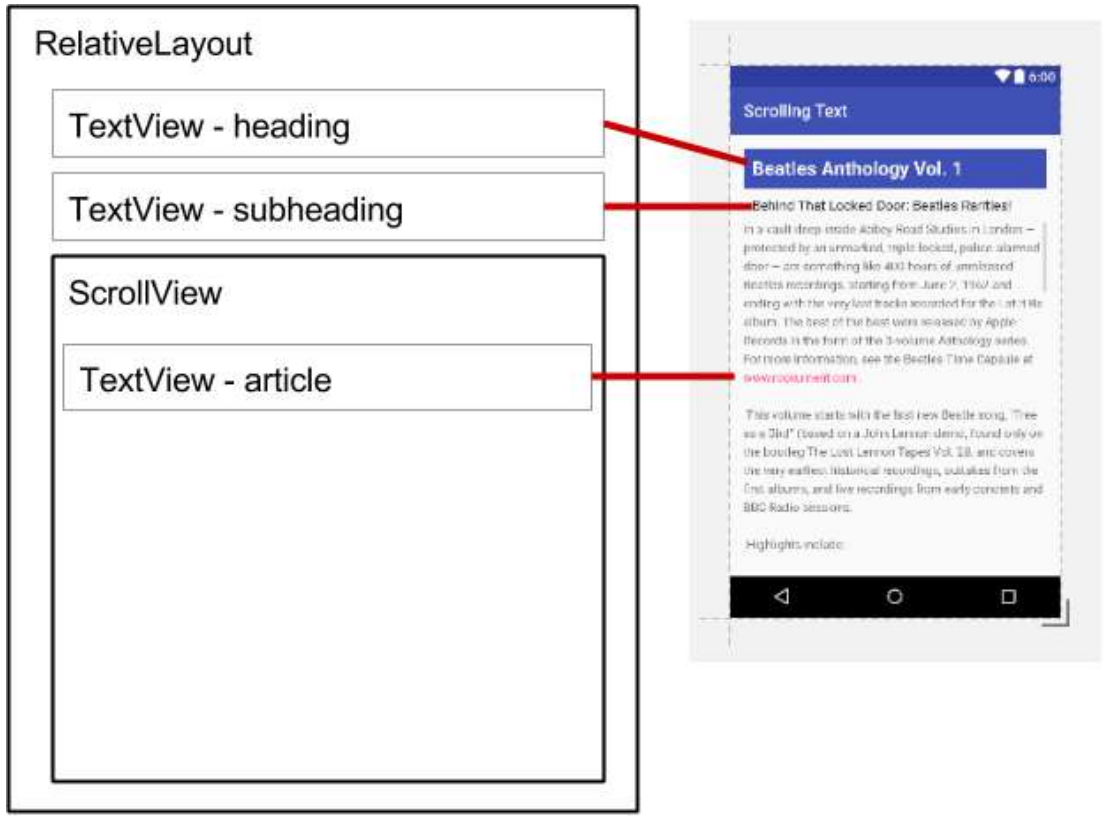
Copyright - Tout droit réservé - OFPPT

Mise en page d'un ScrollView avec un TextView

```
<ScrollView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:layout_below="@id/article_subheading">

  <TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    .../>

</ScrollView>
```



PARTIE 1

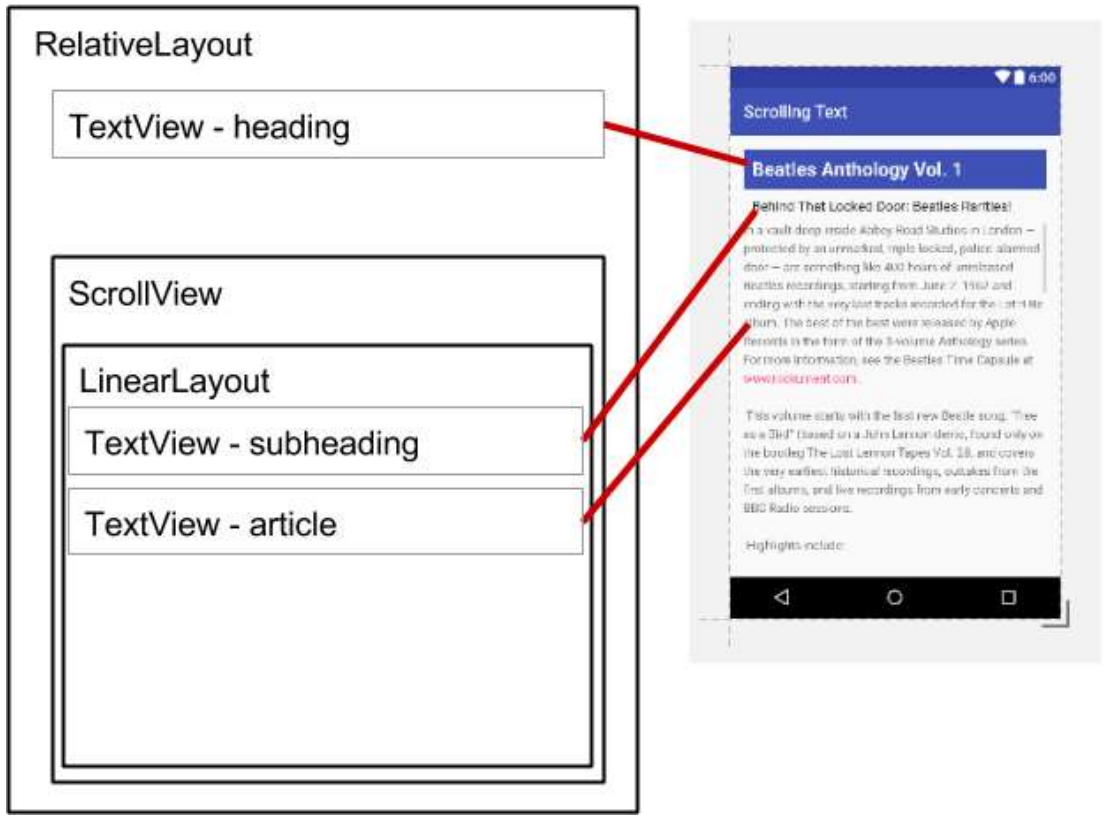
Copyright - Tout droit réservé - OFPPT

ScrollView avec un groupe de vues

```
<ScrollView ...
   <LinearLayout
       android:layout_width="match_parent"
       android:layout_height="wrap_content"
       android:orientation="vertical">

       <TextView
           android:id="@+id/article_subheading"
           .../>

       <TextView
           android:id="@+id/article" ... />
   </LinearLayout>
</ScrollView>
```



Copyright - Tout droit réservé - OFPPT

ScrollView avec image et bouton

```
<ScrollView...>
    <LinearLayout...>
        <ImageView.../>
        <Button.../>
        <TextView.../>
    </LinearLayout>
</ScrollView>
```

Un enfant de ScrollView qui peut être une mise en page (layout)

Enfants de la mise en page

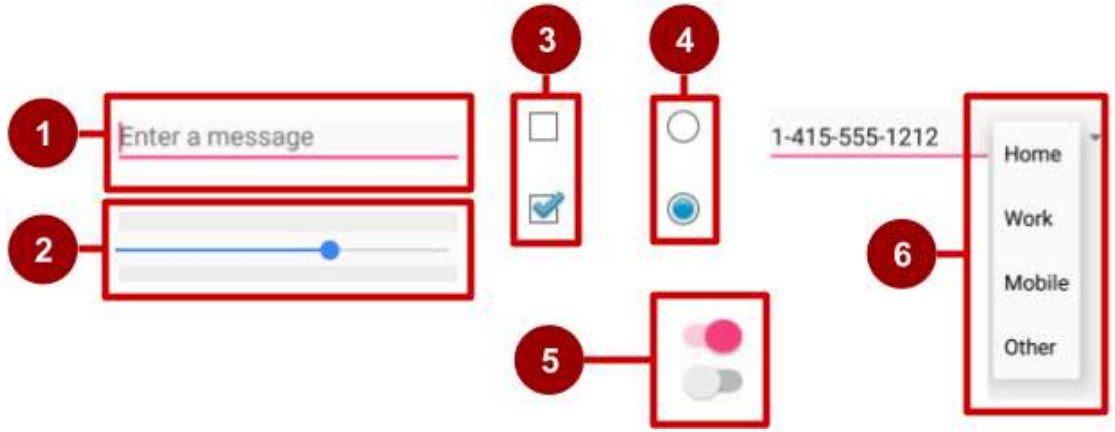
Copyright - Tout droit réservé - OFPPT

Accepter l'entrée de l'utilisateur

- Texte et chiffres de forme libre : EditText (à l'aide du clavier) ;
- Proposer des choix : CheckBox, RadioButton, Spinner (tourniquet) ;
- Activer/désactiver : Toggle, Switch ;
- Choisir une valeur dans une plage de valeurs : SeekBar.

Exemples de contrôles d'entrée

1. EditText
2. SeekBar
3. CheckBox
4. RadioButton
5. Switch
6. Spinner



Copyright - Tout droit réservé - OFPPT

Comment fonctionnent les contrôles d'entrée ?

- Utiliser EditText pour saisir du texte au clavier ;
- Utiliser SeekBar pour glisser vers la gauche ou la droite vers un paramètre ;
- Combiner des éléments CheckBox pour choisir plus d'une option ;
- Combiner les éléments RadioButton en RadioGroup - l'utilisateur ne fait qu'un seul choix ;
- Utiliser un switch pour activer ou désactiver une option ;
- Utiliser le Spinner pour choisir un seul élément dans une liste.

View est une classe de base pour les contrôles d'entrée

- La classe View est le bloc de construction de base de tous les composants de l'interface utilisateur, y compris les contrôles de saisie ;
- View est la classe de base pour les classes qui fournissent des composants interactifs de l'interface utilisateur ;
- La vue fournit une interaction de base via android:onClick

Copyright - Tout droit réservé - OFPPT

Focus

- La vue qui reçoit l'entrée de l'utilisateur a le "Focus" ;
- Une seule vue peut avoir le focus ;
- Le focus permet de déterminer sans ambiguïté la vue qui reçoit les données ;
- Le focus est attribué par :
    - L'utilisateur appuie sur une vue ;
    - L'application guide l'utilisateur d'un contrôle de saisie de texte à l'autre en utilisant les touches Retour, Tab ou les flèches ;
    - Appeler **requestFocus**() sur n'importe quelle vue qui est focusable.

Cliquable versus focusable

- Vue cliquable : La vue peut répondre à un clic ou à une pression ;
- Vue focalisable : La vue peut être focalisée pour accepter des entrées ;
- Les commandes d'entrée telles que les claviers envoient les entrées à la vue qui a le focus.

Copyright - Tout droit réservé - OFPPT

Quelle vue est la suivante ?

- Vue supérieure sous le toucher ;
- Lorsque l'utilisateur soumet une entrée, le focus se déplace vers le voisin le plus proche - la priorité est de gauche à droite, de haut en bas ;
- Le focus peut changer lorsque l'utilisateur interagit avec un contrôle directionnel.

Guider les utilisateurs

- Indiquer visuellement quelle vue a le focus pour que les utilisateurs sachent où vont leurs données ;
- Indiquer visuellement quelles vues peuvent avoir le focus aide les utilisateurs à naviguer dans le flux ;
- Prévisible et logique - pas de surprise !

Copyright - Tout droit réservé - OFPPT

Orientation de focus

- Disposer les contrôles d'entrée dans une mise en page de gauche à droite et de haut en bas dans l'ordre dans lequel ils doivent être mis en évidence ;
- Placer les contrôles de saisie à l'intérieur d'un groupe de vues dans la mise en page ;
- Spécifier l'ordre dans le XML :

```
android:id="@+id/top"
android:focusable="true"
android:nextFocusDown="@+id/bottom"
```

Définir explicitement le focus

Utiliser les méthodes de la classe View pour définir le focus

- setFocusable() définit si une vue peut avoir le focus ;
- requestFocus() donne le focus à une vue spécifique ;
- setOnFocusChangeListener() définit l'écouteur pour quand la vue gagne ou perd le focus ;
- onFocusChanged() est appelé lorsque le focus d'une vue change.

Copyright - Tout droit réservé - OFPPT

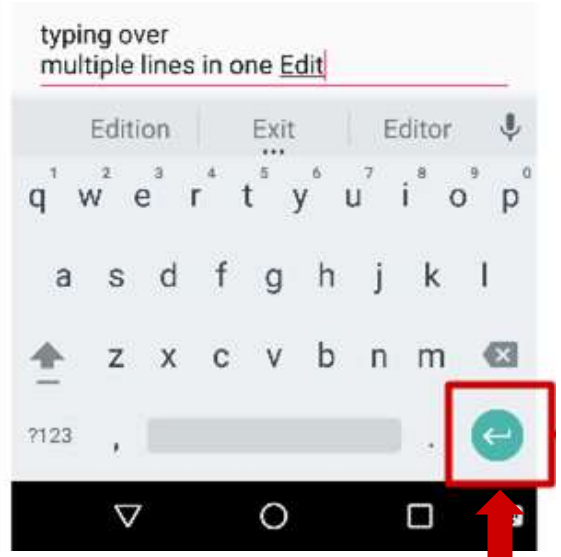
PARTIE 1

Trouver la vue qui a le focus

- Activity.getCurrentFocus() ;
- ViewGroup.getFocusedChild().

EditText pour plusieurs lignes de texte

- EditText par défaut ;
- Clavier alphanumérique ;
- Des suggestions apparaissent ;
- Taper sur la touche Retour (Entrée) pour commencer une nouvelle ligne.

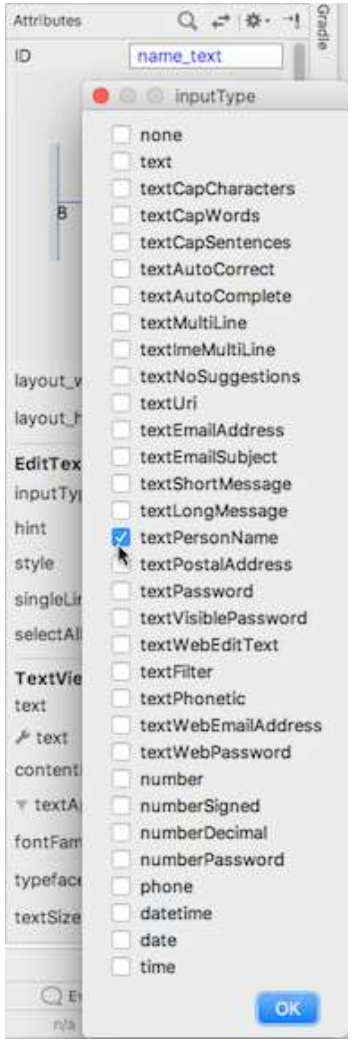


Copyright - Tout droit réservé - OFPPT

Personnaliser avec inputType

- Définir dans le volet Attributs de l'éditeur de mise en page ;
- Code XML pour EditText :

```
<EditText
    android:id="@+id/name_field"
    android:inputType =
                "textPersonName"
    ...
```

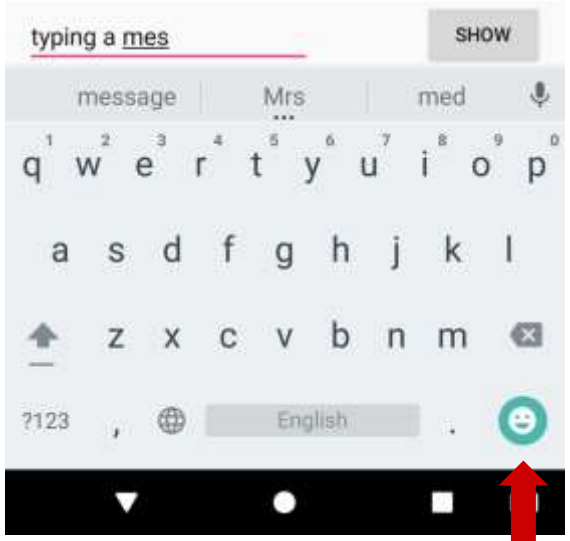


Copyright - Tout droit réservé - OFPPT

EditText pour le message

```
android:inputType
        ="textShortMessage"
```

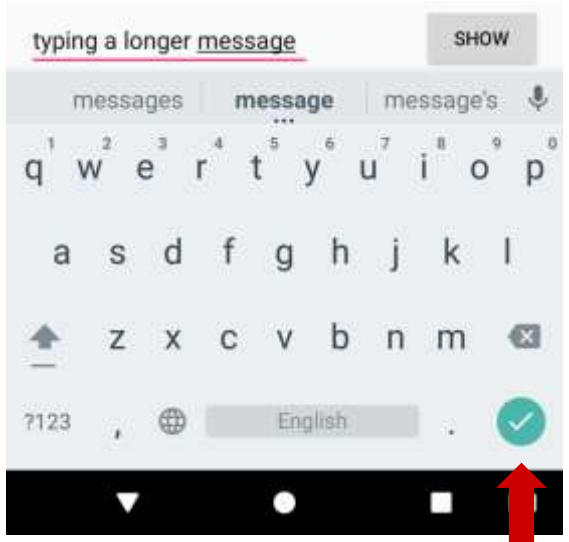
- Ligne de texte unique ;
- En appuyant sur la touche Emoticônes, le clavier se transforme en émoticônes.



Émoticônes

EditText pour une seule ligne

- Les deux fonctionnent :
  - android:inputType
    ="textLongMessage«
  - android:inputType
    ="textPersonName"
- Ligne de texte unique ;
- En appuyant sur la touche Done, on passe à la vue suivante.



Touche Done

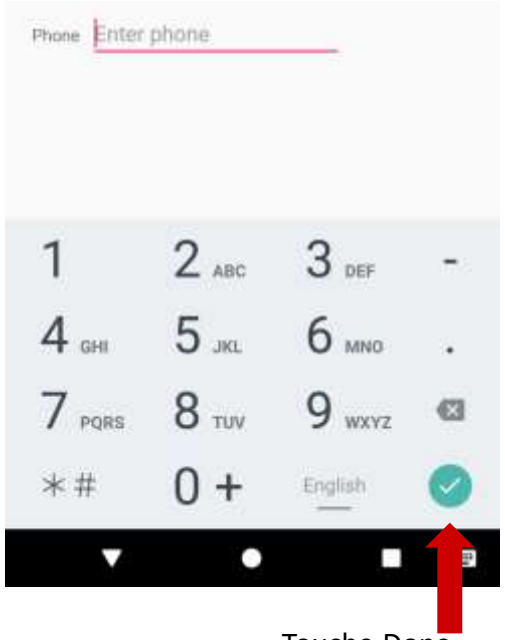
PARTIE 1

Copyright - Tout droit réservé - OFPPT

PARTIE 1

EditText pour la saisie du numéro de téléphone

- `android:inputType ="phone" ;`
- Clavier numérique (chiffres uniquement) ;
- En appuyant sur la touche Done, l'utilisateur passe à la vue suivante.



Touche Done

Obtenir du texte

- Obtenir l'objet EditText pour la vue EditText :

```
EditText simpleEditText =
                    findViewById(R.id.edit_simple);
```

- Récupérer la CharSequence et la convertir en chaîne de caractères :

```
String strValue =
                     simpleEditText.getText().toString();
```

Copyright - Tout droit réservé - OFPPT

Types d'entrées communes

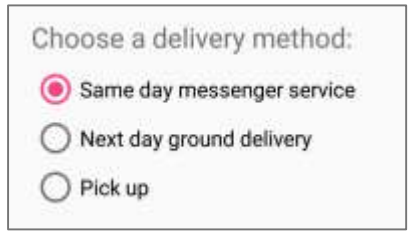
- **textCapCharacters** : Définir toutes les lettres majuscules ;
- **textCapSentences** : Commencer chaque phrase par une majuscule ;
- **textPassword** : dissimuler un mot de passe saisi ;
- **number** (**nombre**) : Restreindre la saisie de texte aux chiffres ;
- **textEmailAddress** : Afficher le clavier avec le symbole @ bien placé ;
- **phone** : Afficher le clavier numérique d'un téléphone ;
- **datetime** : Afficher un clavier numérique avec une barre oblique et deux points pour saisir la date et l'heure.

Copyright - Tout droit réservé - OFPPT

Éléments de l'interface utilisateur permettant de proposer des choix

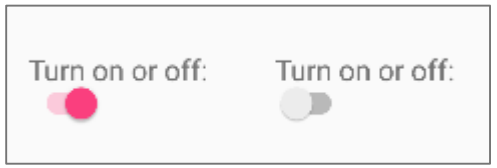
- CheckBox et RadioButton





- ToggleButton et Switch





- Spinner



PARTIE 1

Copyright - Tout droit réservé - OFPPT

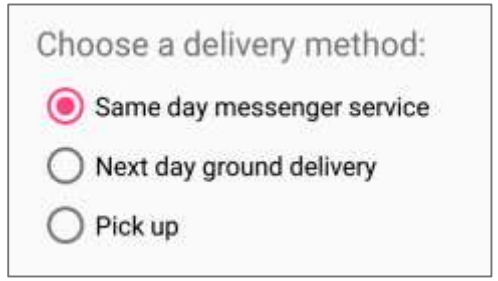
CheckBox

- L'utilisateur peut sélectionner un nombre quelconque de choix ;
- Cocher une case ne décoche pas une autre case ;
- Les utilisateurs s'attendent à voir les cases à cocher dans une liste verticale ;
- Souvent utilisé avec un bouton d'envoi ;
- Chaque case à cocher est une vue et peut avoir un gestionnaire onClick.



RadioButton

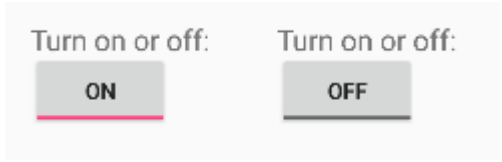
- Placer les éléments RadioButton d'un RadioGroup dans une liste verticale (horizontale si les étiquettes sont courtes) ;
- L'utilisateur ne peut sélectionner qu'un seul des choix ;
- Le fait de cocher l'un d'entre eux désélectionne tous les autres du groupe ;
- Chaque bouton radio peut avoir un gestionnaire onClick ;
- Utilisé couramment avec un bouton d'envoi pour le RadioGroup.



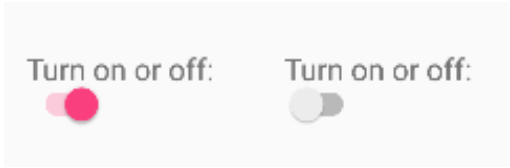
Copyright - Tout droit réservé - OFPPT

Toggle boutons et switches

- L'utilisateur peut switcher entre l'activation et la désactivation ;
- Utiliser android:onClick pour le gestionnaire de clics.



Toggle boutons



Switches

Copyright - Tout droit réservé - OFPPT

Bouton

- Vue qui réagit à un clic ou à une pression ;
- Généralement, un texte ou des éléments visuels indiquent ce qui se passe lorsqu'on appuie dessus ;
- État : normal, focused, disabled, pressed, on/off



Images des boutons

1. Cliquer avec le bouton droit de la souris sur app/res/drawable ;
2. Choisir New > Image Asset ;
3. Choisir Action Bar et Tab Items dans le menu déroulant ;
4. Cliquer sur l'image Clipart : (le logo Android).



2. Essayer :
Choisissez New > Vector Asset

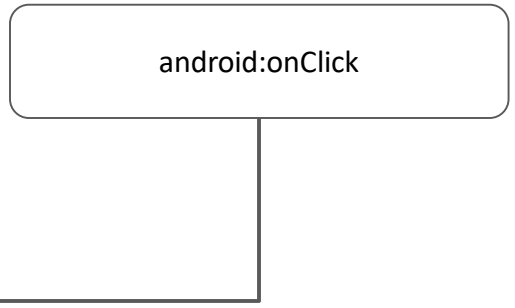
PARTIE 1

Copyright - Tout droit réservé - OFPPT

Réaction aux clics sur les boutons

- Dans le code : Utiliser l'écouteur d'événements OnClickListener ;
- Dans le XML : utilisez l'attribut android:onClick dans le layout XML :

```
<Button
    android:id="@+id/button_send"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/button_send"
    android:onClick="sendMessage" />
```



Configuration de l'écouteur avec le rappel onClick

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

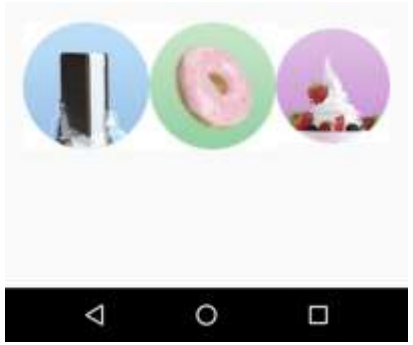
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {
        // Agir en réponse à un clic sur un bouton
    }
});
```

PARTIE 1

Copyright - Tout droit réservé - OFPPT

ImageView

- ImageView avec l'attribut android:onClick ;
- Image pour ImageView dans le dossier app>src>main>res>drawable du projet.



Réaction aux clics sur ImageView

- Dans le code : Utiliser l'écouteur d'événements OnClickListener ;
- Dans le XML : utiliser l'attribut android:onClick dans le layout XML

```
<ImageView
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:src="@drawable/donut_circle"
      android:onClick="orderDonut"/>
```

android:onClick

PARTIE 1

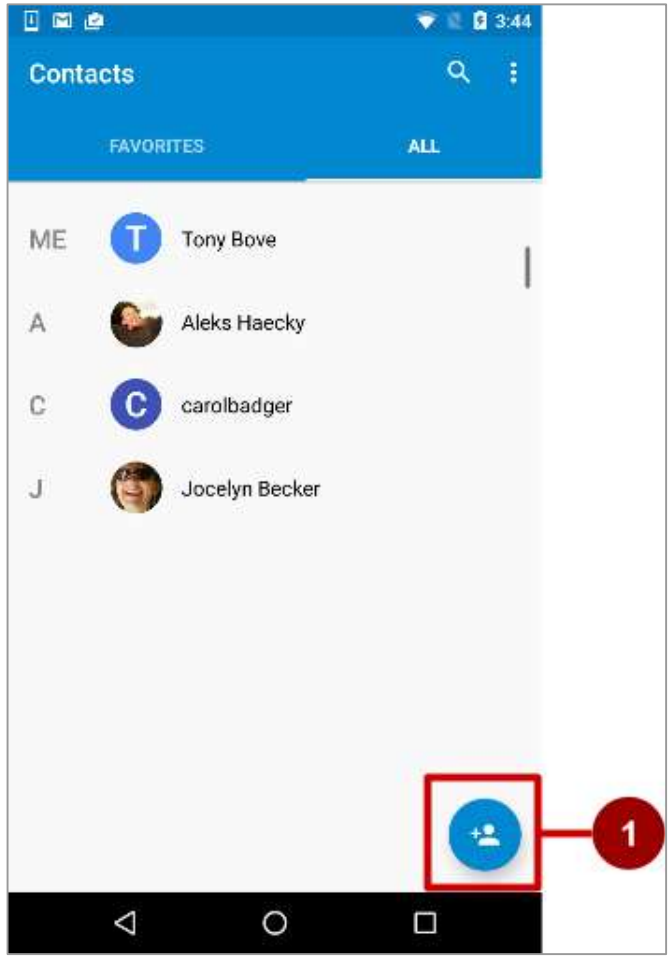
Copyright - Tout droit réservé - OFPPT

Boutons d'action flottants (FAB)

- Surélevé, circulaire, flottant au-dessus de la mise en page
- Action primaire ou "promue" pour un écran
- Une par écran

Par exemple :

Bouton Ajouter un contact dans l'application Contacts



Copyright - Tout droit réservé - OFPPT

Utilisation des FABs

- Commencer par le modèle d'activité de base ;
- Layout:

```
<android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        android:src="@drawable/ic_fab_chat_button_white"

        .../>
```

La taille FAB

- 56 x 56 dp par défaut ;
- Définir la taille mini (30 x 40 dp) avec l'attribut app:fabSize :
    - app:fabSize="mini".
- Définir la taille 56 x 56 dp (par défaut) :
    - app:fabSize="normal".

Copyright - Tout droit réservé - OFPPT

Gestes tactiles

Les gestes tactiles comprennent :

- long touch ;
- double-tap ;
- fling ;
- drag ;
- scroll ;
- Pinch.



Remarques

Ne pas dépendre des gestes tactiles pour le comportement de base de l'application !

Détecter les gestes

Des classes et des méthodes sont disponibles pour faciliter la gestion des gestes.

- Classe GestureDetectorCompat pour les gestes courants ;
- Classe MotionEvent pour les événements de mouvement.

Copyright - Tout droit réservé - OFPPT

Détection de tous les types de gestes

- Recueillir des données sur les événements tactiles ;
- Interpréter les données pour voir si elles répondent aux critères de l'un des gestes pris en charge par l'application.

Pour en savoir plus sur la gestion des gestes, consulter la documentation destinée aux développeurs Android.

Copyright - Tout droit réservé - OFPPT

PARTIE 1

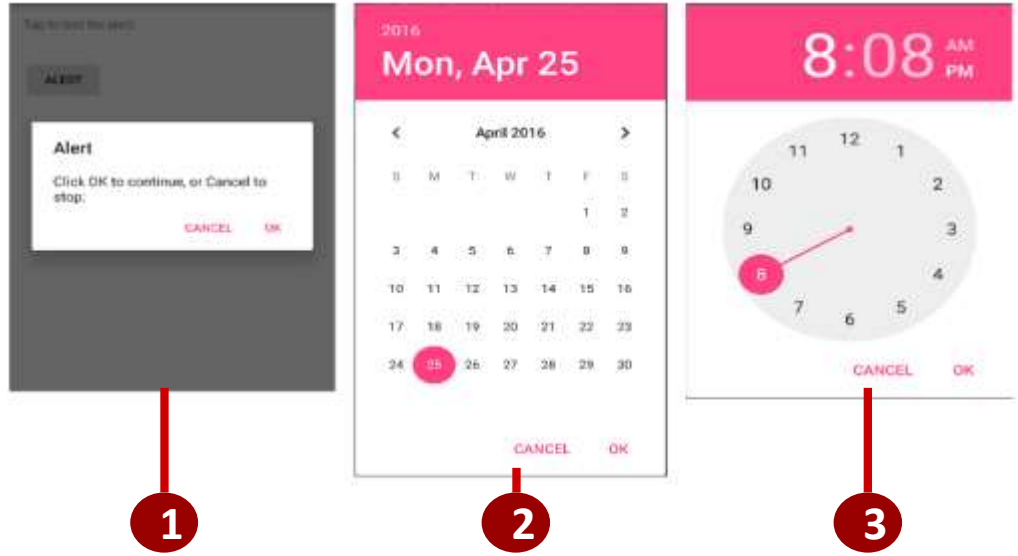
Types des Menus

1. Barre d'applications avec menu d'options ;
2. Menu contextuel ;
3. Barre d'action contextuelle ;
4. Menu contextuel.



Dialogs et pickers

1. Alert dialog ;
2. Date picker ;
3. Time picker.



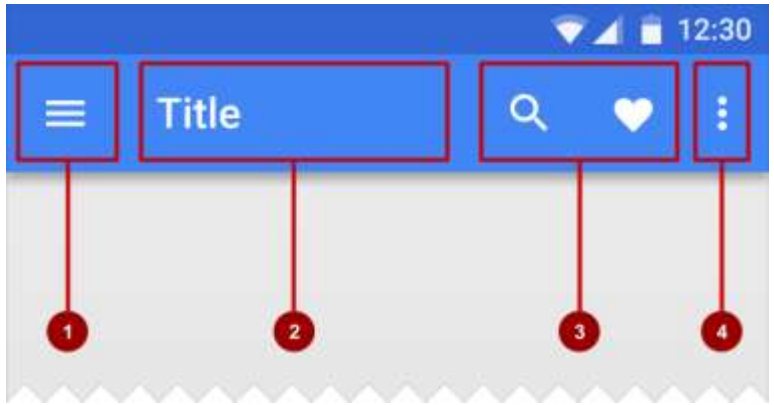
Copyright - Tout droit réservé - OFPPT

PARTIE 1

Qu'est-ce que la barre d'application ?

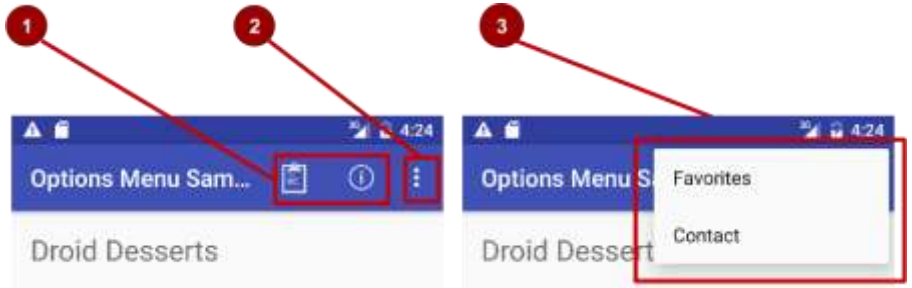
Barre en haut de chaque écran - identique pour tous les appareils (en général)

1. Icône Nav pour ouvrir le volet de navigation ;
2. Titre de l'activité en cours ;
3. Icônes des éléments du menu d'options ;
4. Bouton de débordement d'action pour le reste du menu d'options.



Qu'est-ce que le menu des options ?

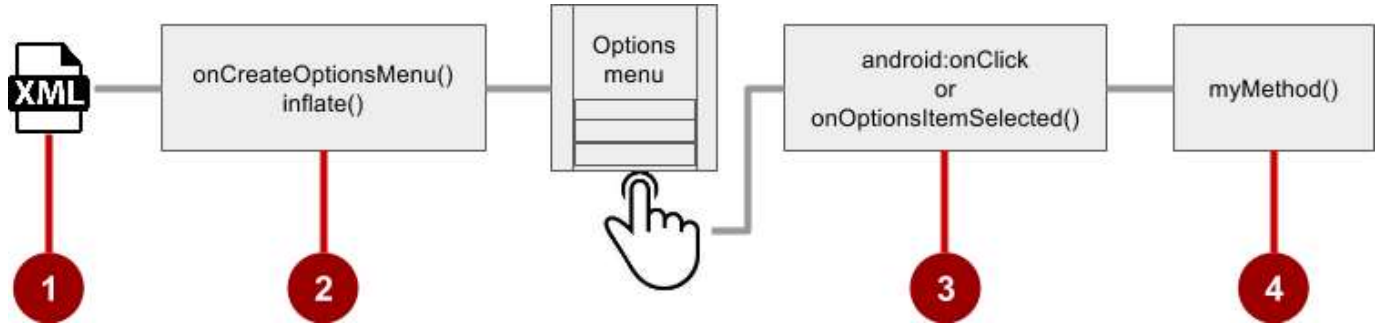
1. Icônes d'action dans la barre d'application pour les éléments importants (1) ;
2. Appuyer sur les trois points, le "bouton de débordement d'action" pour voir le menu d'options (2) ;
3. Apparaît dans le coin droit de la barre d'applications (3) ;
4. Pour naviguer vers d'autres activités et modifier les paramètres de l'application.



Copyright - Tout droit réservé - OFPPT

Étapes de la création d'un menu d'options

1. Ressource de menu XML (menu_main.xml) ;
2. onCreateOptionsMenu() pour inflater le menu ;
3. Attribut onClick ou onOptionsItemSelected() ;
4. Méthode pour gérer le clic sur l'élément.



Copyright - Tout droit réservé - OFPPT

Créer une ressource de menu

1. Créer un répertoire de ressources de menu ;
2. Créer une ressource de menu XML (menu_main.xml) ;
3. Ajouter une entrée pour chaque élément de menu (Paramètres et Favoris) :

```
<item  android:id="@+id/option_settings"
       android:title="Settings" />

 <item  android:id="@+id/option_favorites"
       android:title="Favorites" />
```

Inflater le menu des options

Redéfinir onCreateOptionsMenu() dans l'activité

```
@Override
public boolean onCreateOptionsMenu(Menu menu) {
   getMenuInflater().inflate(R.menu.menu_main, menu);
   return true;
}
```


Copyright - Tout droit réservé - OFPPT

Ajouter des icônes pour les éléments de menu

1. Cliquer avec le bouton droit de la souris sur le dessin ;
2. Choisir New > Image Asset;
3. Choisir Action Bar and Tab Items ;
4. Modifier le nom de l'icône ;
5. Cliquer sur l'image clipart, puis sur l'icône ;
6. Cliquer sur Next, puis sur Finish.



Ajouter des attributs d'éléments de menu

```
<item
        android:id="@+id/action_favorites"
        android:icon="@drawable/ic_favorite"
        android:orderInCategory="30"
        android:title="@string/action_favorites"
        app:showAsAction="ifRoom" />
```

PARTIE 1

Copyright - Tout droit réservé - OFPPT

Redéfinir onOptionsItemSelected()

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case R.id.action_settings:
         showSettings();
         return true;
      case R.id.action_favorites:
         showFavorites();
         return true;
      default:
            return super.onOptionsItemSelected(item);
      }
}
```


Copyright - Tout droit réservé - OFPPT

Que sont les menus contextuels ?

1. Permet aux utilisateurs d'effectuer des actions sur la vue sélectionnée ;
2. Peut être déployé sur n'importe quelle vue ;
3. Utilisé le plus souvent pour les éléments de RecyclerView, GridView, ou autre collection de vues.

Types de menus contextuels

- Menu contextuel flottant en appuyant longuement sur une vue :



  - L'utilisateur peut modifier la vue ou l'utiliser d'une manière ou d'une autre ;
  - L'utilisateur effectue une action sur une seule vue à la fois.

- Mode d'action contextuel : barre d'action temporaire à la place ou en dessous de la barre d'application :

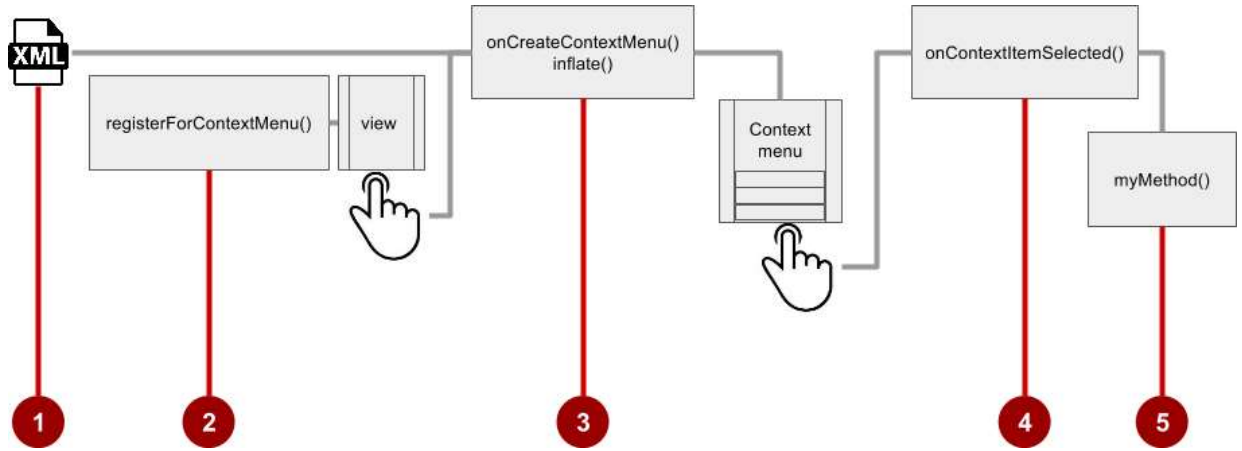


  - Les éléments d'action affectent le ou les éléments de vue sélectionnés ;
  - L'utilisateur peut agir sur plusieurs éléments de vue à la fois.

PARTIE 1

Copyright - Tout droit réservé - OFPPT

Etapes



1. Créer un fichier de ressources de menu XML et attribuer les attributs d'apparence et de position ;
2. Enregistrer la vue en utilisant registerForContextMenu() ;
3. Implémenter onCreateContextMenu() dans l'activité pour gonfler le menu ;
4. Implémenter onContextItemSelected() pour gérer les clics sur les éléments du menu ;
5. Créer une méthode pour exécuter une action pour chaque élément du menu contextuel.

Copyright - Tout droit réservé - OFPPT

Créer une ressource de menu

1. Créer une ressource de menu XML (menu_context.xml) :

```
<item
   android:id="@+id/context_edit"
   android:title="Edit"
   android:orderInCategory="10"/>

<item
   android:id="@+id/context_share"
   android:title="Share"
   android:orderInCategory="20"/>
```

Enregistrer une vue dans un menu contextuel

Dans onCreate() de l'activité :

2. Enregistrer View.OnCreateContextMenuListener à View :

```
TextView article_text = findViewById(R.id.article) ;

registerForContextMenu(article_text) ;
```


Copyright - Tout droit réservé - OFPPT

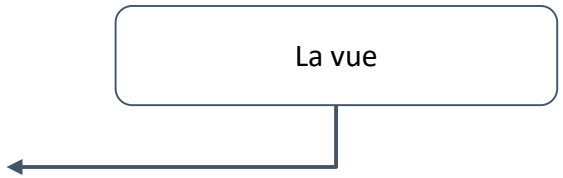
Implémenter la méthode onCreateContextMenu()

3. Spécifier le menu contextuel :

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
                                ContextMenu.ContextMenuInfo menuInfo) {
   super.onCreateContextMenu(menu, v, menuInfo);
   MenuInflater inflater = getMenuInflater();
   inflater.inflate(R.menu.menu_context, menu);
}
```

Implémenter la méthode onContextItemSelected()

```
@Override
public boolean onContextItemSelected(MenuItem item) {
   switch (item.getItemId()) {
      case R.id.context_edit:
         editNote();
         return true;
      case R.id.context_share:
         shareNote();
         return true;
      default:
         return super.onContextItemSelected(item);
     }
}
```



Copyright - Tout droit réservé - OFPPT

Qu'est-ce que le mode action ?

- Mode d'interface utilisateur qui vous permet de remplacer temporairement certaines parties des interactions normales de l'interface utilisateur ;
- Par exemple : La sélection d'une section de texte ou une action longue sur un élément peut déclencher le mode action.

Le mode action a un cycle de vie

- Démarrer avec startActionMode(), par exemple, dans le listener ;
- L'interface ActionMode.Callback fournit des méthodes de cycle de vie que l'on surcharge :
  - onCreateActionMode(ActionMode, Menu) une fois lors de la création initiale ;
  - onPrepareActionMode(ActionMode, Menu) après la création et chaque fois que l'ActionMode est invalidé ;
  - onActionItemClicked(ActionMode, MenuItem) chaque fois qu'un bouton d'action contextuel est cliqué;
  - onDestroyActionMode(ActionMode) lorsque le mode d'action est fermé.

Copyright - Tout droit réservé - OFPPT

Qu'est-ce qu'une barre d'action contextuelle ?

Une pression longue sur View affiche la barre d'action contextuelle.

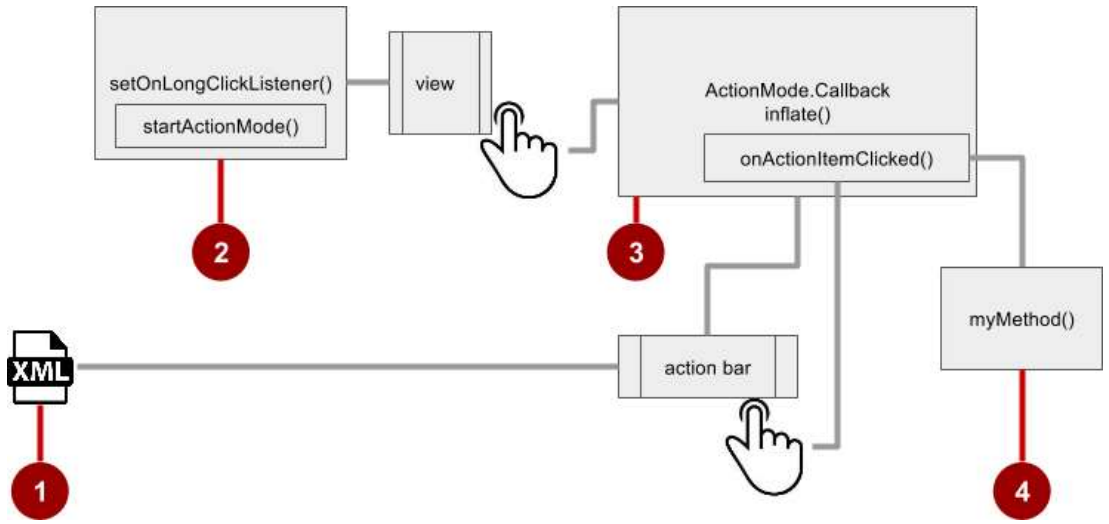
1. Barre d'action contextuelle avec actions :
    - Modifier, Partager, et Supprimer ;
    - Terminé (icône flèche gauche) sur le côté gauche.;
    - La barre d'action est disponible jusqu'à ce que l'utilisateur appuie sur Terminé.
2. Vue sur laquelle un appui long déclenche la barre d'action contextuelle.



Copyright - Tout droit réservé - OFPPT

Étapes pour la barre d'action contextuelle

1. Créer un fichier de ressources de menu XML et attribuer des icônes aux éléments ;
2. **setOnLongClickListener**() sur la vue qui déclenche la barre d'action contextuelle et appeler **startActionMode**() pour gérer le clic ;
3. Implémenter l'interface **ActionMode**.**Callback** pour gérer le cycle de vie de l'ActionMode ; inclure une action pour le clic sur l'élément de menu dans le callback **onActionItemClicked**() ;
4. Créer une méthode pour exécuter une action pour chaque élément du menu contextuel.



Copyright - Tout droit réservé - OFPPT

Utiliser setOnLongClickListener

```
private ActionMode mActionMode;
```

Dans onCreate():

```
View view = findViewById(article);
view.setOnLongClickListener(new View.OnLongClickListener() {
   public boolean onLongClick(View view) {
      if (mActionMode != null) return false;
      mActionMode =
            MainActivity.this.startActionMode(mActionModeCallback);
      view.setSelected(true);
      return true;
   }
});
```


Copyright - Tout droit réservé - OFPPT

Implémenter mActionModeCallback

```
public ActionMode.Callback mActionModeCallback =
   new ActionMode.Callback() {
    // Implémenter les callbacks du mode action ici.
};
```

Implémenter onCreateActionMode

```
@Override
public boolean onCreateActionMode(ActionMode mode, Menu menu) {
   MenuInflater inflater = mode.getMenuInflater();
   inflater.inflate(R.menu.menu_context, menu);
   return true;
}
```


Copyright - Tout droit réservé - OFPPT

Implémenter onPrepareActionMode

- Appelé chaque fois que le mode d'action est affiché ;
- Toujours appelé après onCreateActionMode, mais peut être appelé plusieurs fois si le mode d'action est invalidé.

```
@Override
public boolean onPrepareActionMode(ActionMode mode, Menu menu) {
   return false; // Renvoie false si rien n'est fait.
}
```

Implémenter onActionItemClicked

- Appelé lorsque l'utilisateur sélectionne une action ;
- Gérer les clics dans cette méthode :

```
@Override
public boolean onActionItemClicked(ActionMode mode, MenuItem item) {
   switch (item.getItemId()) {
       case R.id.action_share:
         // Effectuer une action pour l'élément de menu Partager.
         mode.finish(); // Action choisie, donc fermeture de la barre d'action.
         return true;
       default:
         return false;
   }
}
```


Copyright - Tout droit réservé - OFPPT

Implémenter onDestroyActionMode

- Appelé lorsque l'utilisateur quitte le mode action

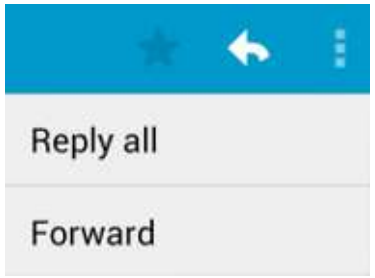
```
@Override
public void onDestroyActionMode(ActionMode mode) {
   mActionMode = null;
}
```


Copyright - Tout droit réservé - OFPPT

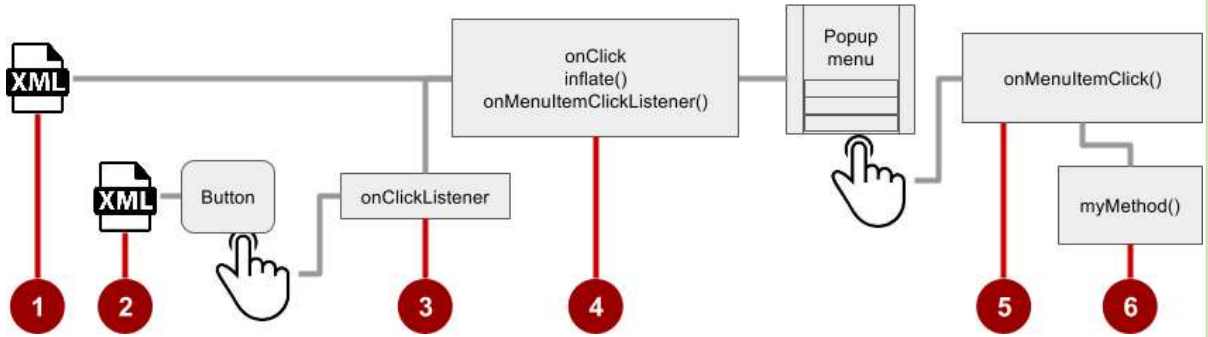
PARTIE 1

Qu'est-ce qu'un menu contextuel ?

- Liste verticale d'éléments ancrés à une vue ;
- Généralement ancrée à une icône visible ;
- Les actions ne doivent pas affecter directement le contenu de la vue
  - Icône de débordement du menu Options qui ouvre le menu Options ;
  - Dans l'application de messagerie, les fonctions Répondre à tous et Transférer se rapportent au message électronique mais n'ont pas d'incidence sur le message.



Etapes



1. Créer un fichier de ressources de menu XML et attribuer les attributs d'apparence et de position ;
2. Ajouter ImageButton pour l'icône du menu contextuel dans le fichier de disposition d'activité XML ;
3. Attribuer un onClickListener à ImageButton ;
4. Remplacer onClick() pour gonfler le popup et l'enregistrer avec onMenuItemClickListener() ;
5. Implémenter onMenuItemClick() ;
6. Créer une méthode pour exécuter une action pour chaque élément du menu contextuel.

Copyright - Tout droit réservé - OFPPT

Ajouter ImageButton



```
<ImageButton
 android:layout_width="wrap_content"
 android:layout_height="wrap_content"
 android:id="@+id/button_popup"
 android:src="@drawable/@drawable/ic_action_popup"/>
```

Attribuer un onClickListener au bouton

```
private ImageButton mButton =
    (ImageButton) findViewById(R.id.button_popup);
```

Dans onCreate():

```
mButton.setOnClickListener(new View.OnClickListener() {
    // définir onClick
});
```


Copyright - Tout droit réservé - OFPPT

Implémenter onClick

```
@Override
public void onClick(View v) {
    PopupMenu popup = new PopupMenu(MainActivity.this, mButton);
    popup.getMenuInflater().inflate(
        R.menu.menu_popup, popup.getMenu());
    popup.setOnMenuItemClickListener(
        new PopupMenu.OnMenuItemClickListener() {
            // implémenter le click listener.
          });
     popup.show();
}
```


Copyright - Tout droit réservé - OFPPT

Implémenter onMenuItemClick

```
public boolean onMenuItemClick(MenuItem item) {
   switch (item.getItemId()) {
       case R.id.option_forward:
           // Implémenter le code pour le bouton Forward.
           return true;
       default:
           return false;
   }
}
```


Copyright - Tout droit réservé - OFPPT

Dialogs

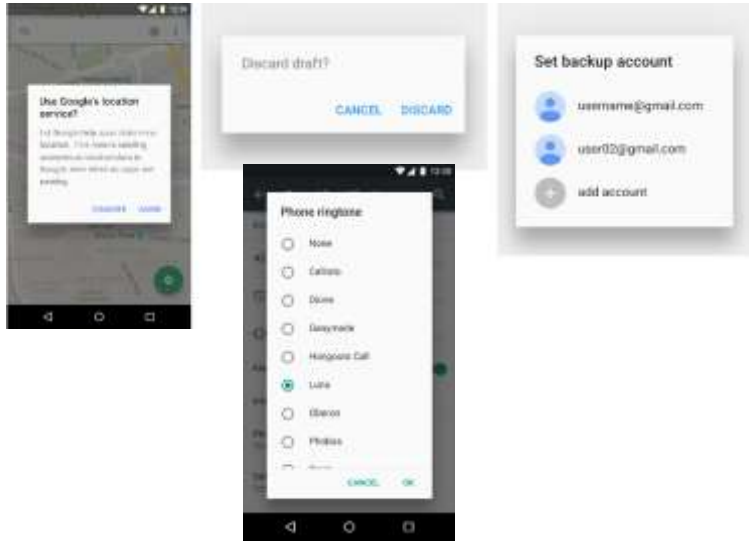
- La boîte de dialogue apparaît en haut de la page, interrompant le flux de l'activité ;
- L'utilisateur doit agir pour le faire disparaître.



TimePickerDialog



DatePickerDialog



AlertDialog

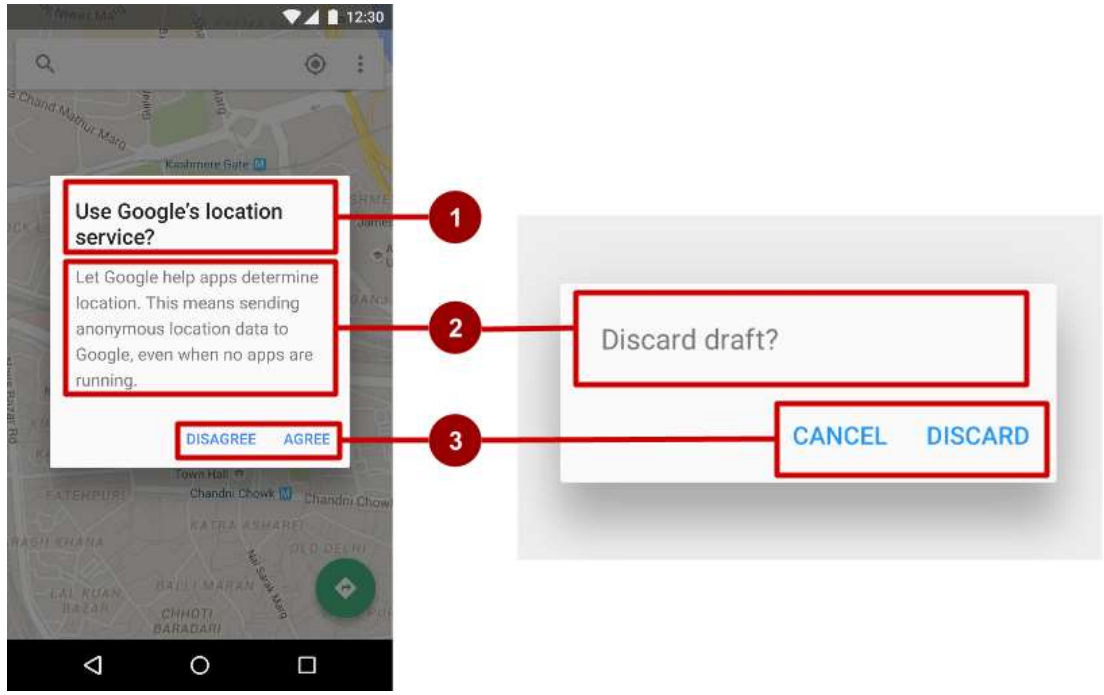
PARTIE 1

Copyright - Tout droit réservé - OFPPT

AlertDialog

AlertDialog peut afficher :

- Titre (facultatif) ;
- Zone de contenu ;
- Boutons d'action.



PARTIE 1

Copyright - Tout droit réservé - OFPPT

Construire l'AlertDialog

Utiliser **AlertDialog.Builder** pour construire une boîte de dialogue d'alerte et définir les attributs :

```
public void onClickShowAlert(View view) {
      AlertDialog.Builder alertDialog = new
                  AlertDialog.Builder(MainActivity.this);
      alertDialog.setTitle("Connect to Provider");
      alertDialog.setMessage(R.string.alert_message);
      // ... Le code pour définir les boutons à mettre ici.
```

Définir les actions des boutons

- alertDialog.setPositiveButton()
- alertDialog.setNeutralButton()
- alertDialog.setNegativeButton()

Copyright - Tout droit réservé - OFPPT

Exemple 1 de code d'alertDialog

```
alertDialog.setPositiveButton(
                    "OK", newDialogInterface.OnClickListener() {
    public void onClick(DialogInterface dialog, int which) {
        // L'utilisateur a cliqué sur le bouton OK.
    }
});
```

Même schéma pour **setNegativeButton()** et **setNeutralButton().**

Copyright - Tout droit réservé - OFPPT

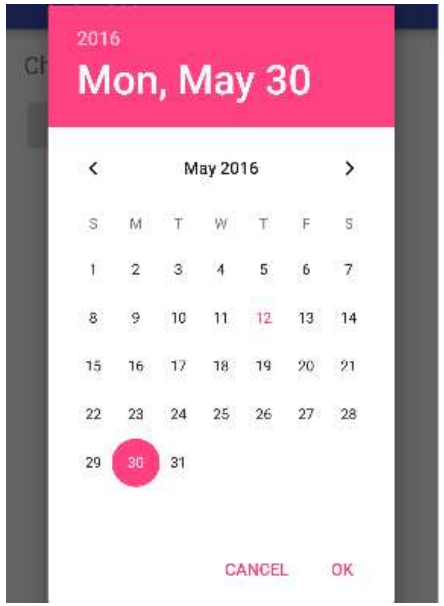
Exemple 2 de code d'alertDialog

```
1. AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(this);
2.     alertDialogBuilder.setMessage("Are you sure,You wanted to make decision");
3.     alertDialogBuilder.setPositiveButton("yes", new DialogInterface.OnClickListener() {
4.       @Override
5.       public void onClick(DialogInterface arg0, int arg1) {
6.         Toast.makeText(MainActivity.this, "You clicked yes button", Toast.LENGTH_LONG).show();
7.       }
8.     });
9.     alertDialogBuilder.setNegativeButton("No",new DialogInterface.OnClickListener() {
10.       @Override
11.       public void onClick(DialogInterface dialog, int which) {
12.         finish();
13.       }
14.     });
15.     AlertDialog alertDialog = alertDialogBuilder.create();
16.     alertDialog.show();
```


Copyright - Tout droit réservé - OFPPT

Pickers

- DatePickerDialog
- TimePickerDialog





Les pickers utilisent des fragments

- Utiliser le **DialogFragment** pour afficher un sélecteur ;
- **DialogFragment** est une fenêtre qui flotte au dessus de la fenêtre Activity.



PARTIE 1

Copyright - Tout droit réservé - OFPPT

Introduction aux fragments

- Un fragment est comme une mini-activité au sein d'une activité ;
    - Gère son propre cycle de vie ;
    - Reçoit ses propres événements d'entrée.
- Peut être ajouté ou supprimé pendant l'exécution de l'activité mère ;
- Plusieurs fragments peuvent être combinés dans une seule activité ;
- Peut être réutilisé dans plus d'une activité.

Création d'un dialogue de sélection de date

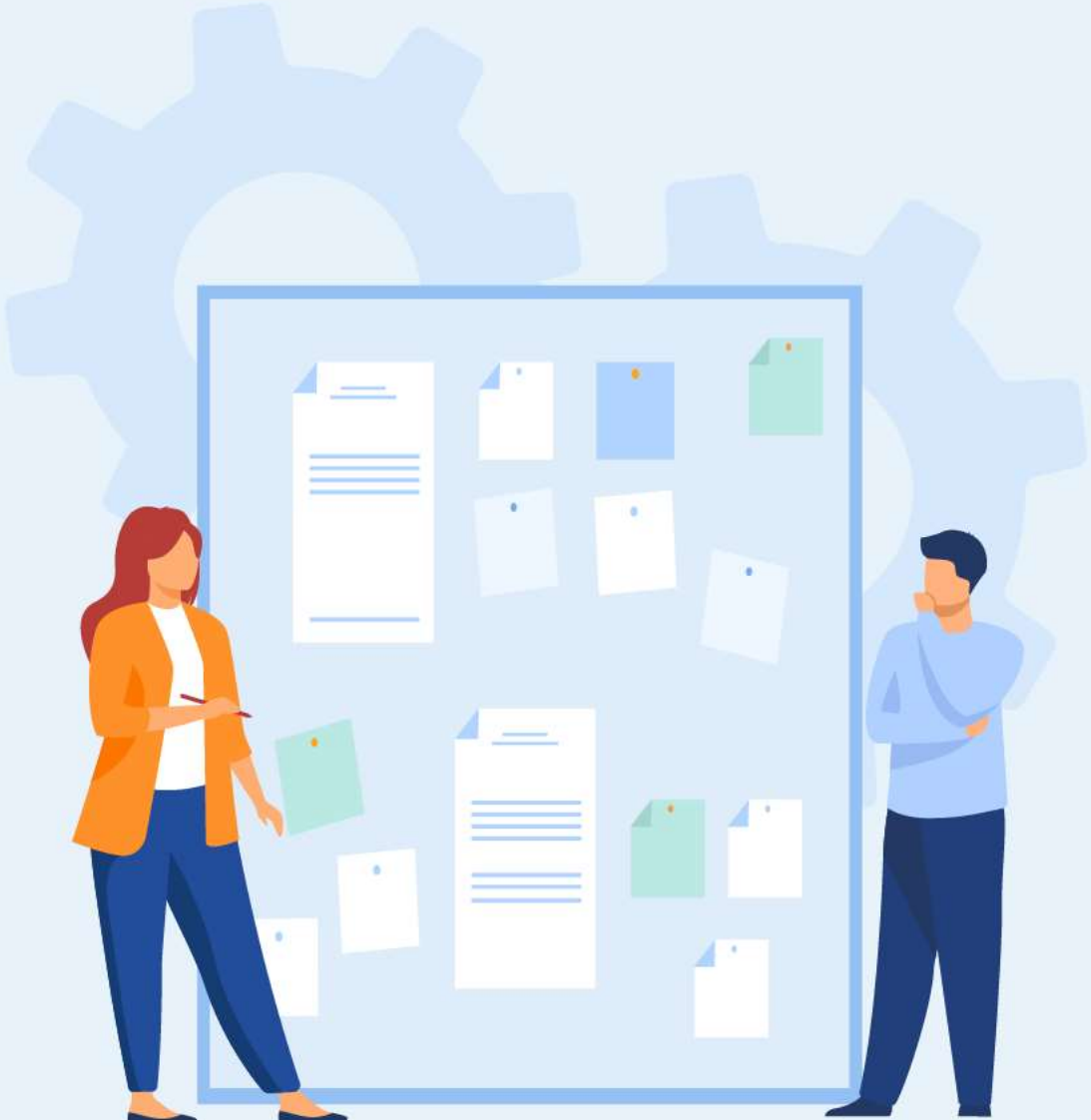
1. Ajouter un fragment vierge qui étend **DialogFragment** et implémente **DatePickerDialog.OnDateSetListener** ;
2. Dans **onCreateDialog()**, initialiser la date et retourner le dialogue ;
3. Dans **onDateSet(),** manipuler la date ;
4. Dans Activity, afficher le sélecteur et ajouter une méthode pour utiliser la date.

Copyright - Tout droit réservé - OFPPT

Création d'une boîte de dialogue de sélection de l'heure

1. Ajouter un fragment vierge qui étend **DialogFragment** et implémente **TimePickerDialog.OnTimeSetListener** ;
2. Dans **onCreateDialog()**, initialiser l'heure et retourner le dialogue ;
3. Dans **onTimeSet(),** gérer l'heure ;
4. Dans l'activité, afficher le sélecteur et ajouter une méthode pour utiliser le temps.

Copyright - Tout droit réservé - OFPPT



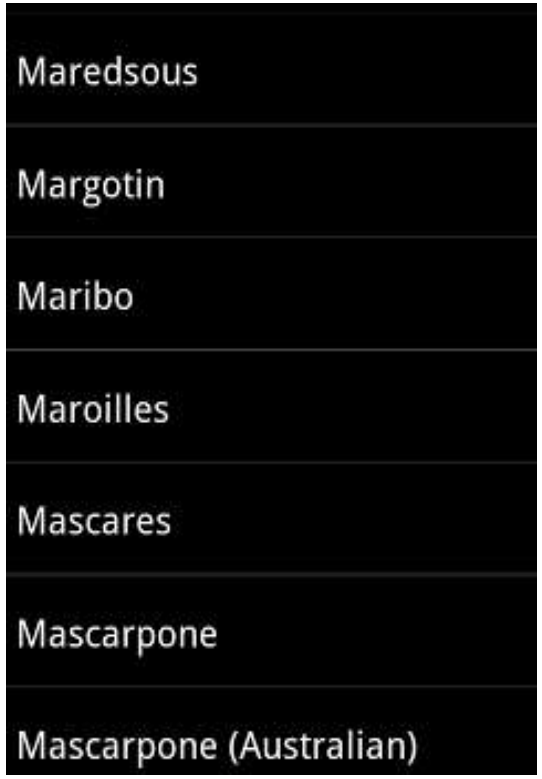
CHAPITRE 1

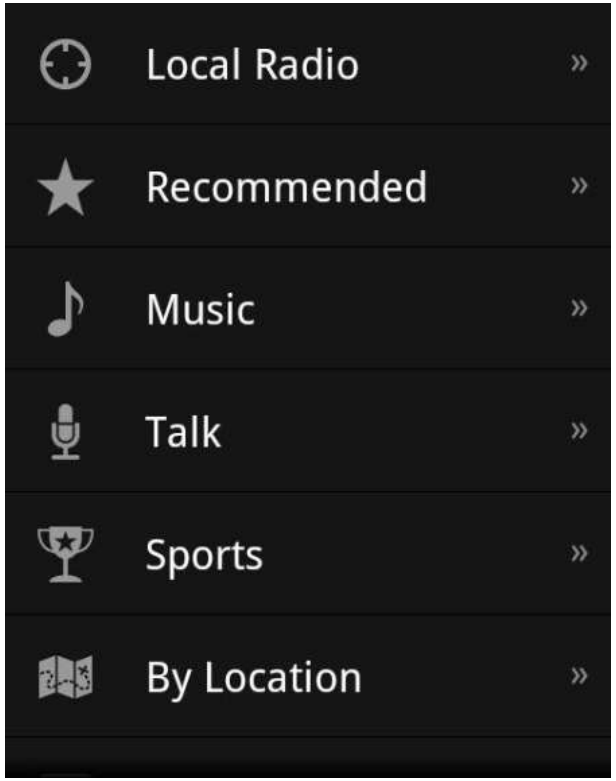
Implémenter des éléments graphiques

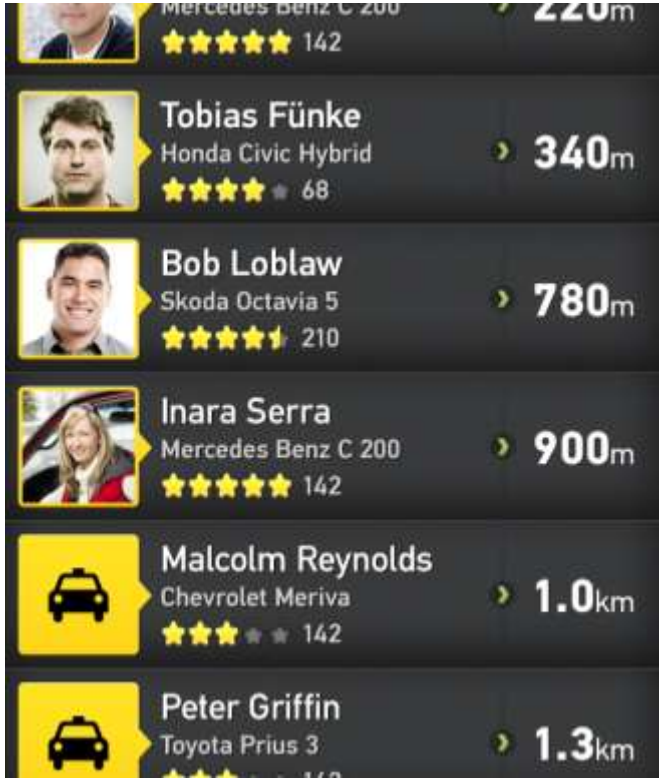
1. Définition des Widgets
2. **Présentation des Widgets avancés**

ListView

Les ListViews sont comme toutes les autres listes où nous avons une série d'éléments alignés de manière verticale. Dans Android, ces listes sont utilisées de nombreuses façons, généralement comme des menus pour accéder à différents écrans.







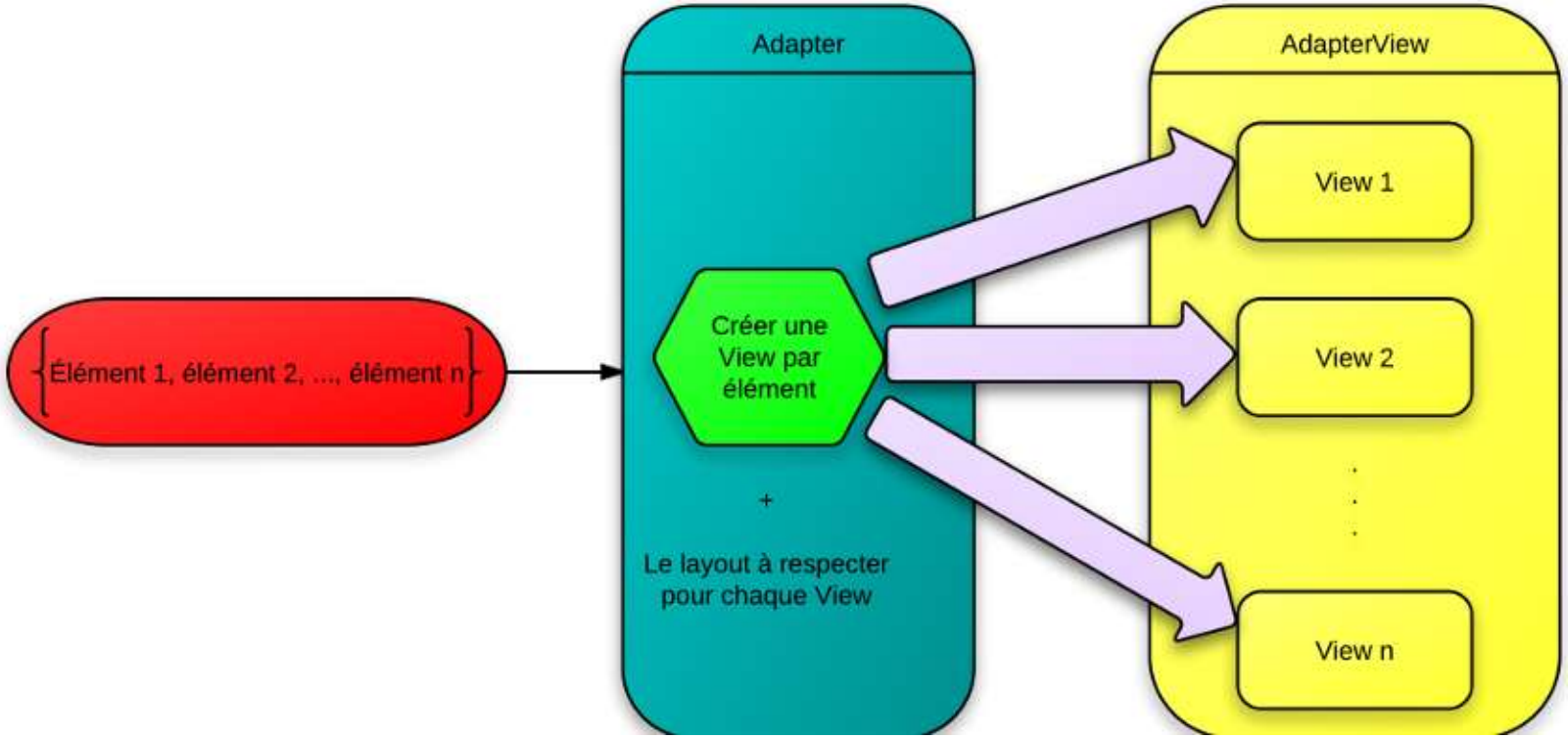
PARTIE 1

Copyright - Tout droit réservé - OFPPT

ListView

La gestion des listes se divise en deux parties distinctes :

- **Adapter** : Les objets qui gèrent les donnée, On peut considérer un adaptateur comme un intermédiaire entre les données et la vue qui représente ces données ;
- **AdapterView :** gère l'affichage des données.

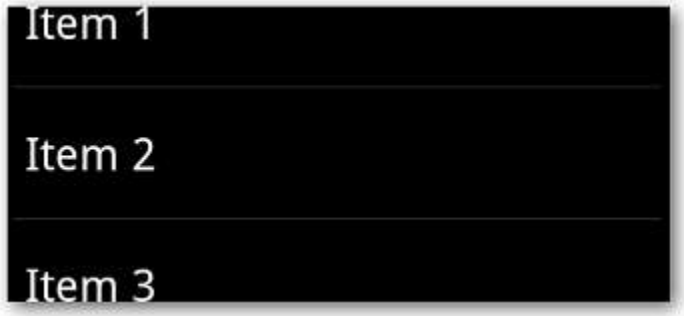


Copyright - Tout droit réservé - OFPPT

ListView

Réalisation d'une liste simple se fait en 5 étapes :

1. Déclaration de la liste dans le fichier XML de l'activity ;
2. Créer la vue type : le layout qui va être utilisé par chaque ligne de la liste ;
3. Déclarer le tableau des données qui va peupler les éléments de la liste ;
4. Instancier le « Adapter » qui va associer les données et les vues ;
5. Attribuer le « Adapter » à votre ListView crée.



Copyright - Tout droit réservé - OFPPT

ListView

1. Déclaration de la liste dans le fichier XML de l'activity

Dans le fichier : /res/layout/main.xml , il faut déclarer la liste comme suit :

Code XML :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical" >

      <ListView
         android:id="@+id/list"
         android:layout_height="wrap_content"
         android:layout_width="match_parent">
      </ListView>

</LinearLayout>
```


Copyright - Tout droit réservé - OFPPT

ListView

2. Créer la vue type : le layout qui va être utilisé par chaque ligne de la liste

Créer le fichier : /res/layout/list_layout.xml , et créer le layout de la liste comme suit :

Code XML :

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@+id/text"
 android:layout_width="match_parent"
 android:layout_height="wrap_content"
 android:padding="10dp"
 android:textSize="16sp" >
</TextView>
```


Copyright - Tout droit réservé - OFPPT

ListView

3. Déclarer le tableau des données qui va peupler les éléments de la liste:

Dans la class Java de votre Activity, créer la liste des données qui vont peupler la liste :

Code Java :

```
public class MainActivity extends ActionBarActivity {

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

          String listItems[] = { "List Item 1", "List Item 2", "List Item 3",
          "List Item 4", "List Item 5", "List Item 6", "List Item 7",
          "List Item 8", "List Item 9", "List Item 10", "List Item 11",
          "List Item 12" };
   }
```


Copyright - Tout droit réservé - OFPPT

ListView

4. Instancier le « Adapter » qui va associer les données et les vues

Dans la class Java de votre Activity, créer le Adapter de la liste :

Code Java :

```
public class MainActivity extends ActionBarActivity {


  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
        String listItems[] = { "List Item 1", "List Item 2", "List Item 3",
        "List Item 4", "List Item 5", "List Item 6", "List Item 7",
        "List Item 8", "List Item 9", "List Item 10", "List Item 11",
        "List Item 12" };
         ArrayAdapter<String> adapter = new
                ArrayAdapter<String>(this,R.layout.list_layout, listItems);
  }
```


Copyright - Tout droit réservé - OFPPT

ListView

5. Attribuer le « Adapter » à votre ListView crée

Code Java :

```
public class MainActivity extends ActionBarActivity {


  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
        String listItems[] = { "List Item 1", "List Item 2", "List Item 3",
        "List Item 4", "List Item 5", "List Item 6", "List Item 7",
        "List Item 8", "List Item 9", "List Item 10", "List Item 11",
        "List Item 12" };
         ArrayAdapter<String> adapter = new                ArrayAdapter<String>(this,R.layout.list_layout, listItems);
         list.setAdapter(adapter);
  }
```


Copyright - Tout droit réservé - OFPPT

ListView

Réalisation d'une liste simple se fait en 5 étapes :

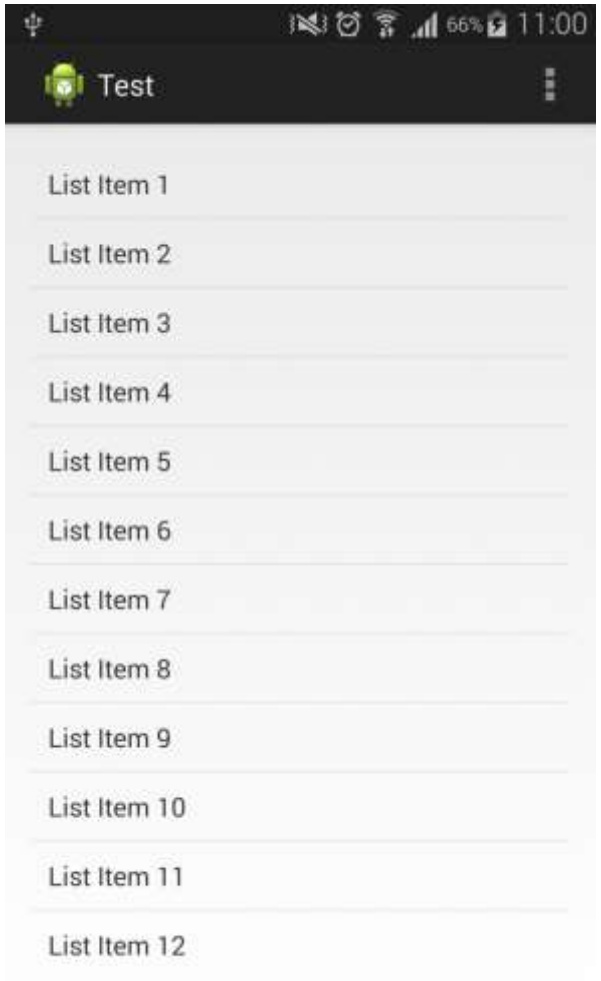
Aperçu du rendu ci après.

Associer une Listener :

- list.**setOnItemClickListener**(this);

```
public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Toast.makeText(getApplicationContext(),
        ((TextView) view).getText(), Toast.LENGTH_SHORT).show();
  }
```

parent	L'AdapterView où le clic s'est produit.
view	La vue de l'AdapterView qui a été cliquée (il s'agit d'une vue fournie par l'adaptateur).
position	La position de la vue dans l'adaptateur.
id	L'identifiant de ligne de l'élément qui a été cliqué.



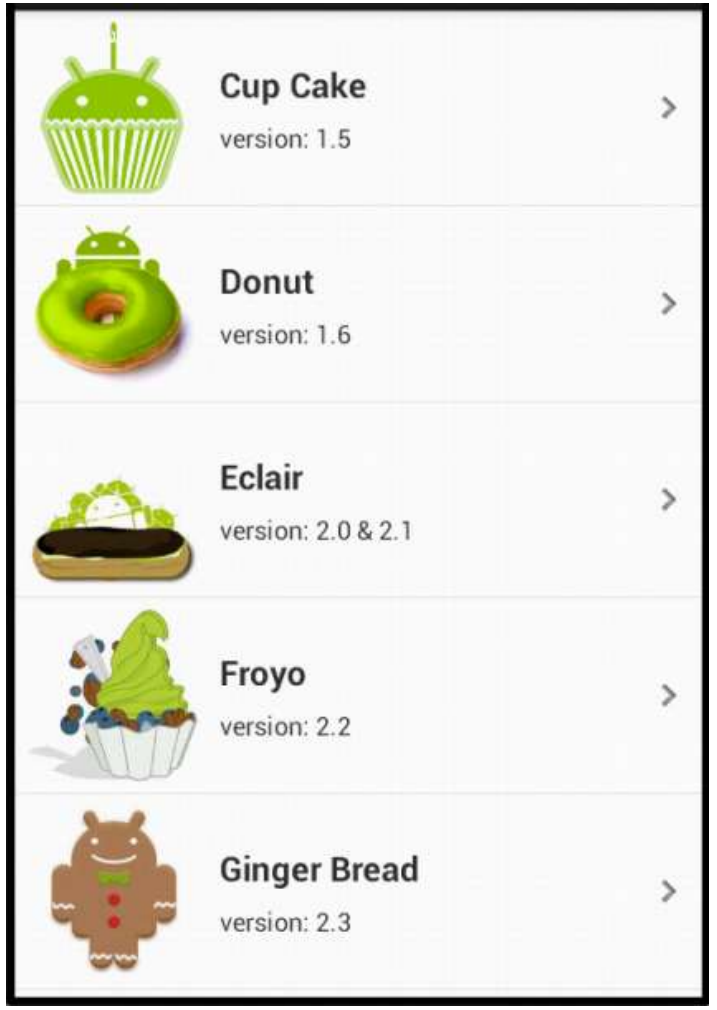
PARTIE 1

Copyright - Tout droit réservé - OFPPT

ListView personnalisée

Réalisation d'une liste personnalisée se fait en 5 étapes :

1. Déclaration de la liste dans le fichier XML de l'activity ;
2. Créer la vue personnalisée : le layout qui va être utilisé par chaque ligne de la liste ;
3. Déclarer les données qui vont peupler les éléments de la liste ;
4. Instancier le « Adapter » qui va associer les données et les vues ;
5. Attribuer le « Adapter » à votre ListView crée.



Copyright - Tout droit réservé - OFPPT

ListView personnalisée

1. Déclaration de la liste dans le fichier XML de l'activity

Dans le fichier : /res/layout/main.xml , il faut déclarer la liste comme suit :

Code XML :

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:orientation="vertical" >

      <ListView
         android:id="@+id/list"
         android:layout_height="wrap_content"
         android:layout_width="match_parent">
      </ListView>

</LinearLayout>
```

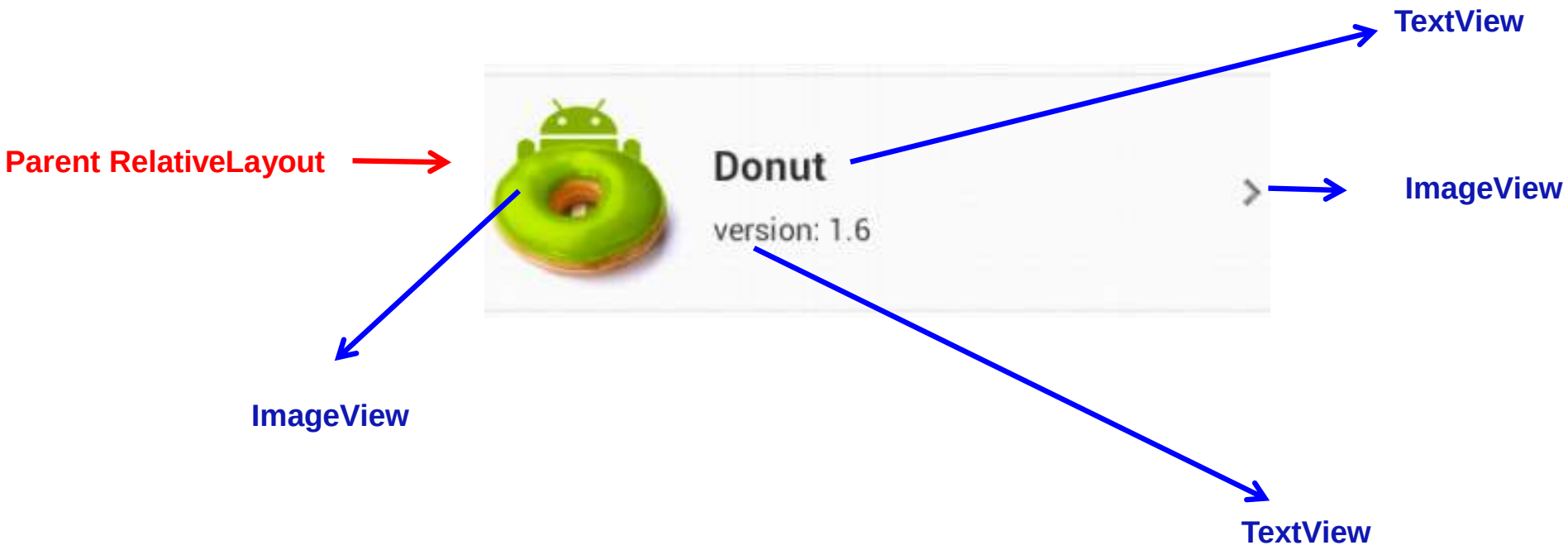

Copyright - Tout droit réservé - OFPPT

ListView personnalisée

2. Déclaration de la liste dans le fichier XML de l'activity

Créer le fichier : /res/layout/list_layout.xml , et créer le layout de la liste .

Il faut créer le design suivant qui va être le même pour tous les éléments de la liste.



Copyright - Tout droit réservé - OFPPT

ListView personnalisée

2. Déclaration de la liste dans le fichier XML de l'activity

Code source fichier : /res/layout/list_layout.xml

```
<RelativeLayout
xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="wrap_content"
android:orientation="vertical"
android:padding="5dp" >

<ImageView
android:id="@+id/thumb"
android:layout_width="80dp"
android:layout_height="80dp"
android:contentDescription="@string/app_name" />
<LinearLayout
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_centerVertical="true"
android:layout_marginLeft="10dp"
android:layout_marginRight="5dp"
android:layout_toLeftOf="@+id/arrow"
android:layout_toRightOf="@+id/thumb"
android:orientation="vertical" >
```

```
<TextView
android:id="@+id/title"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="title"
android:textSize="16dp"
android:textStyle="bold" />
<TextView
android:id="@+id/desc"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_marginTop="5dp"
android:text="desc vers "
android:textSize="12dp" />
</LinearLayout>
<ImageView
android:id="@+id/arrow"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:layout_alignParentRight="true"
android:layout_centerVertical="true"
android:layout_marginRight="5dp"
android:contentDescription="@string/app_name"
android:src="@drawable/arrow_next" />
</RelativeLayout>
```

PARTIE 1

Copyright - Tout droit réservé - OFPPT

ListView personnalisée

3. Déclaration des données de la liste

Code Java :

```
private String title[] = { "Cup Cake", "Donut", "Eclair", "Froyo",
"Ginger Bread", "Honey Comb", "Icecream Sandwich", "Jelly Bean" };

private String desc[] = { "version: 1.5", "version: 1.6",
"version: 2.0 & 2.1", "version: 2.2", "version: 2.3",
"version: 3.0", "version: 4.0", "version: 4.1" };

private int thumb[] = { R.drawable.cupcake, R.drawable.donut,
R.drawable.eclair, R.drawable.froyo, R.drawable.gingerbread,
R.drawable.honeycomb, R.drawable.icecreamsandwich,
R.drawable.jellybean, };
```


Copyright - Tout droit réservé - OFPPT

ListView personnalisée

4. Instancier le « Adapter »

Créer la classe suivante :

```
class VersionAdapter extends BaseAdapter {

private LayoutInflater layoutInflater;

public VersionAdapter(MainActivity activity) {
// TODO Auto-generated constructor stub
layoutInflater = (LayoutInflater) activity
.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
}

@Override
public int getCount() {
// TODO Auto-generated method stub
return title.length;
}

@Override
public Object getItem(int position) {
// TODO Auto-generated method stub
return position;
}
```

```
@Override
public long getItemId(int position) {
// TODO Auto-generated method stub
return position;
}
@Override
public View getView(int position, View convertView, ViewGroup
parent) {
// TODO Auto-generated method stub

View listItem = convertView;
int pos = position;
if (listItem == null) {
listItem = layoutInflater.inflate(R.layout.list_item, null);
}

ImageView iv = (ImageView) listItem.findViewById(R.id.thumb);
TextView tvTitle = (TextView) listItem.findViewById(R.id.title);
TextView tvDesc = (TextView) listItem.findViewById(R.id.desc);

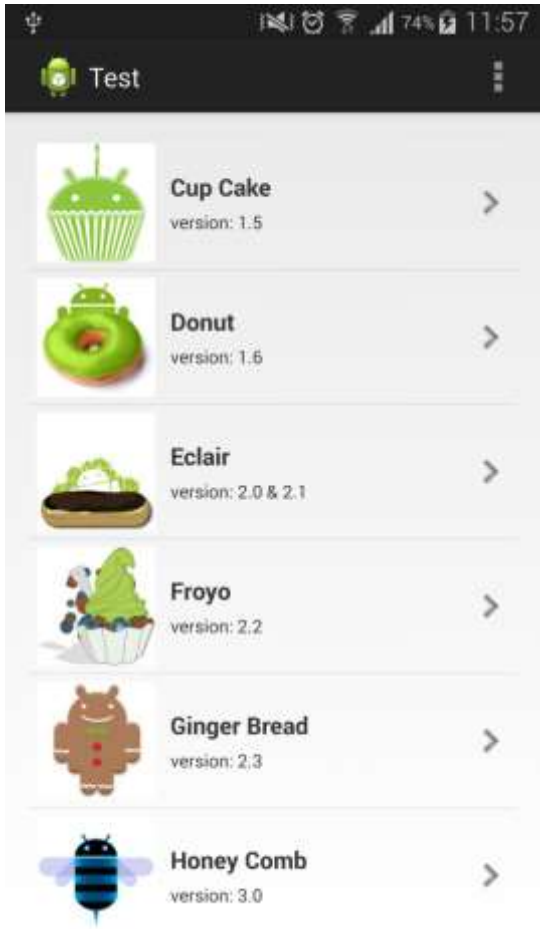
// set the views in the layout
iv.setBackgroundResource(thumb[pos]);
tvTitle.setText(title[pos]);
tvDesc.setText(desc[pos]);
return listItem;
}
```


Copyright - Tout droit réservé - OFPPT

ListView personnalisée

Il faut attribuer le Adapter a la liste pour finalement avoir le rendu ci-après :

```
list.setAdapter(new VersionAdapter(this));
```



Copyright - Tout droit réservé - OFPPT

Qu'est-ce qu'un RecyclerView ?

- RecyclerView est un conteneur défilable pour les grands ensembles de données ;
- Efficace :
  - Utilise et réutilise un nombre limité d'éléments de vue ;
  - Mise à jour rapide des données changeantes.



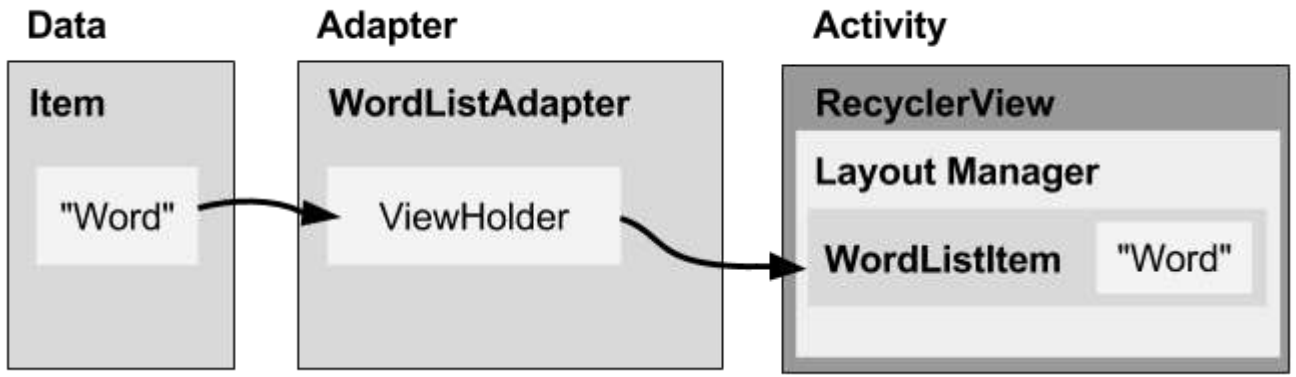
Copyright - Tout droit réservé - OFPPT

Vue d'ensemble

- Données ;
- RecyclerView liste déroulante pour les éléments de la liste : **RecyclerView** ;
- Mise en page pour un élément de données : Fichier XML ;
- Le gestionnaire de mise en page s'occupe de l'organisation des composants de l'interface utilisateur dans une vue : **Recyclerview.LayoutManager** ;
- L'adaptateur relie les données au RecyclerView : **RecyclerView.Adapter ;**
- ViewHolder contient les informations relatives à l'affichage d'un élément : **RecyclerView.ViewHolder**.

Copyright - Tout droit réservé - OFPPT

Aperçu de la façon dont les composants s'assemblent



Copyright - Tout droit réservé - OFPPT

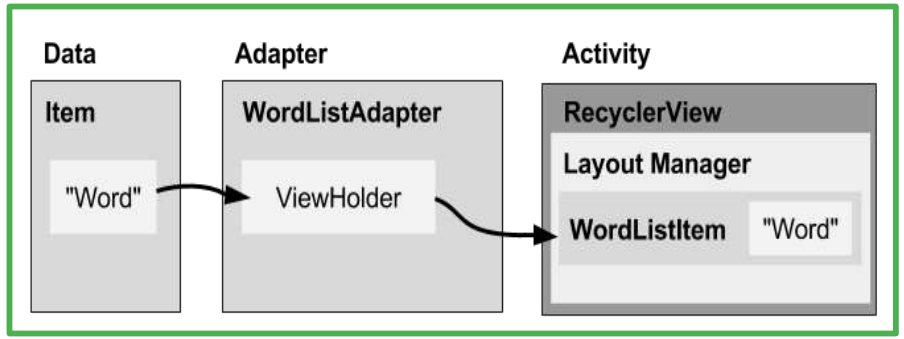
PARTIE 1

Qu'est-ce qu'un layout manager ?

- Chaque ViewGroup possède un layout manager ;
- Permet de positionner les Éléments ;
- Réutilise les éléments d'affichage qui ne sont plus visibles pour l'utilisateur ;
- Gestionnaires de mise en page intégrés :
  - LinearLayoutManager ;
  - GridLayoutManager ;
  - StaggeredGridLayoutManager.
- Extension de RecyclerView.LayoutManager.

Qu'est-ce qu'un adapter ?

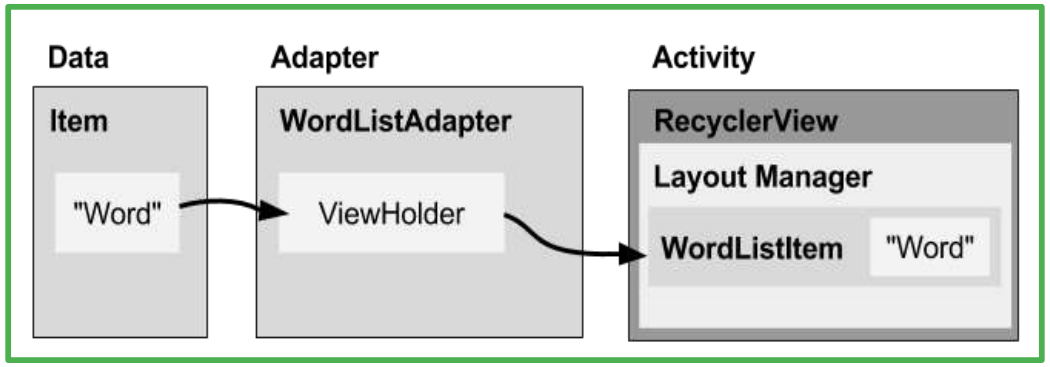
- Aide les interfaces incompatibles à fonctionner ensemble :
  - **Exemple** : Prend les données du curseur de la base de données et prépare les chaînes de caractères à placer dans une vue.
- Intermédiaire entre les données et la vue ;
- Gère la création, la mise à jour, l'ajout et la suppression des éléments de la vue lorsque les données sous-jacentes changent ;
- RecyclerView.Adapter



Copyright - Tout droit réservé - OFPPT

Qu'est-ce qu'un ViewHolder ?

- Utilisé par l'adaptateur pour préparer une vue avec des données pour un élément de liste ;
- Mise en page spécifiée dans un fichier de ressources XML ;
- Peut avoir des éléments cliquables ;
- Est placée par le gestionnaire de mise en page ;
- RecyclerView.ViewHolder.



Copyright - Tout droit réservé - OFPPT

Résumé des étapes

1. Ajouter la dépendance de **RecyclerView** à **build.gradle** si nécessaire ;
2. Ajouter **RecyclerView** à la mise en page ;
3. Créer une mise en page XML pour l'élément ;
4. Extension de **RecyclerView.Adapter** ;
5. Extension de **RecyclerView.ViewHolder** ;
6. Dans l'activité **onCreate()**, créer **RecyclerView** avec l'adaptateur et le gestionnaire de mise en page.

Copyright - Tout droit réservé - OFPPT

Ajouter la dépendance à app/build.gradle

Ajouter la dépendance de RecyclerView à build.gradle si nécessaire :

```
dependencies {
    implementation "androidx.recyclerview:recyclerview:1.2.1"
    // Pour contrôler la sélection des éléments par le toucher et la souris.
    implementation "androidx.recyclerview:recyclerview-selection:1.1.0"
}
```

Ajouter la dépendance à app/build.gradle

Ajouter une vue de recyclage à un modèle XML :

```
<androidx.recyclerview.widget.RecyclerView
    android:id="@+id/recyclerview"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />
```


Copyright - Tout droit réservé - OFPPT

Créer un layout pour 1 élément de liste

```
<LinearLayout …>
  <TextView
    android:id="@+id/word"
    style="@style/word_title" />
</LinearLayout>
```



Copyright - Tout droit réservé - OFPPT

Implémenter l'adaptateur

```
public class WordListAdapter extends RecyclerView.Adapter<WordListAdapter.WordViewHolder> {

    public WordListAdapter(Context context, LinkedList<String> wordList) {
        mInflater = LayoutInflater.from(context);
        this.mWordList = wordList;
    }
}
```

L'adaptateur a 3 méthodes requises

- onCreateViewHolder()
- inBindViewHolder()
- getItemCount()

Jetons-y un coup d'œil !

Copyright - Tout droit réservé - OFPPT

onCreateViewHolder()

```
@Override
public WordViewHolder onCreateViewHolder(
        ViewGroup parent, int viewType) {
        // Create view from layout
        View mItemView = mInflater.inflate(
             R.layout.wordlist_item, parent, false);
        return new WordViewHolder(mItemView, this);
}
```

onBindViewHolder()

```
@Override
public void onBindViewHolder(
        WordViewHolder holder, int position) {
        // Retrieve the data for that position
        String mCurrent = mWordList.get(position);
        // Add the data to the view
        holder.wordItemView.setText(mCurrent);
}
```


Copyright - Tout droit réservé - OFPPT

getItemCount()

```
@Override
public int getItemCount() {
  // Return the number of data items to display
  return mWordList.size();
}
```

Créer le gestionnaire de vue dans la classe d'adaptateur

```
class WordViewHolder extends RecyclerView.ViewHolder { //.. }
```

Si l'on veut gérer les clics de souris :

```
class WordViewHolder extends RecyclerView.ViewHolder
                     implements View.OnClickListener { //.. }
```


Copyright - Tout droit réservé - OFPPT

Constructeur de view holder

```
public WordViewHolder(View itemView, WordListAdapter adapter) {
  super(itemView);
  // Get the layout
  wordItemView = itemView.findViewById(R.id.word);
  // Associate with this adapter
  this.mAdapter = adapter;
  // Add click listener, if desired
  itemView.setOnClickListener(this);
}
// Implement onClick() if desired
```


Copyright - Tout droit réservé - OFPPT

Créer le RecyclerView dans l'activité dans onCreate()

```
mRecyclerView = findViewById(R.id.recyclerview);
mAdapter = new WordListAdapter(this, mWordList);
mRecyclerView.setAdapter(mAdapter);
mRecyclerView.setLayoutManager(new LinearLayoutManager(this));
```


Copyright - Tout droit réservé - OFPPT

RecyclerViewDemo app

Dans cette partie, les étapes pour la mise en place d'une application à base d'un RecyclerView avec le langage Kotlin seront présentées. l'application affiche une liste de nombres, et possède un RecyclerView.Adapter qui utilise une liste d'entiers comme données source pour créer des vues d'éléments pour la liste.

Une mise en page d'élément de liste contient un TextView, et une vue (ViewHolder) est utilisée pour contenir une référence à cette vue. Un ViewHolder facilite la mise à jour du TextView quand les vues sont recyclées, lorsque l'utilisateur fait défiler la liste.



Copyright - Tout droit réservé - OFPPT

Adapter pour RecyclerViewDemo

```
class NumberListAdapter(var data: List<Int>):
        RecyclerView.Adapter<NumberListAdapter.IntViewHolder>() {
    class IntViewHolder(val row: View): RecyclerView.ViewHolder(row) {
        val textView = row.findViewById<TextView>(R.id.number)
    }
}
```

Fonctions pour RecyclerViewDemo

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
            IntViewHolder {
   val layout = LayoutInflater.from(parent.context)
       .inflate(R.layout.item_view, parent, false)
   return IntViewHolder(layout)
}

override fun onBindViewHolder(holder: IntViewHolder, position: Int) {
   holder.textView.text = data.get(position).toString()
}
```


Copyright - Tout droit réservé - OFPPT

Dans MainActivity.tk définir l'adaptateur sur le RecyclerView

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)

    val rv: RecyclerView = findViewById(R.id.rv)
    rv.layoutManager = LinearLayoutManager(this)

    rv.adapter = NumberListAdapter(IntRange(0,100).toList())
}
```

Remarques

Dans l'activité principale, nous configurons le RecyclerView. À noter que nous initialisons NumberListAdapter en créant une liste de chiffres de 0 à 100. Puis nous plaçons l'adaptateur sur le RecyclerView.

Dans NumberListAdapter.kt Rendre les éléments de la liste cliquables

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): IntViewHolder{
    val layout = LayoutInflater.from(parent.context).inflate(R.layout.item_view,
         parent, false)
    val holder = IntViewHolder(layout)
    holder.row.setOnClickListener {
        // Do something on click
    }
    return holder
}
```

Remarques

Comme pour toute autre vue, il est possible d'ajouter des écouteurs de clics à une vue entière d'éléments de liste ou à une vue enfant de celle-ci. Étant donné que les view holders sont recyclés, il suffit de configurer les click listeners une seule fois lorsque le ViewHolder est créé, et non lorsqu'il est lié (ou relié). Lorsqu'un élément est cliqué, il est possible de faire en sorte qu'un message Toast apparaisse avec la valeur du TextView.

Copyright - Tout droit réservé - OFPPT

ListAdapter

- RecyclerView.Adapter :
  - Dispose des données de l'interface utilisateur après chaque mise à jour ;
  - Peut être coûteux et inutile.

- ListAdapter :
  - Calcule la différence entre ce qui est actuellement affiché et ce qui doit réellement l'être ;
  - Les changements sont calculés sur un thread d'arrière-plan.

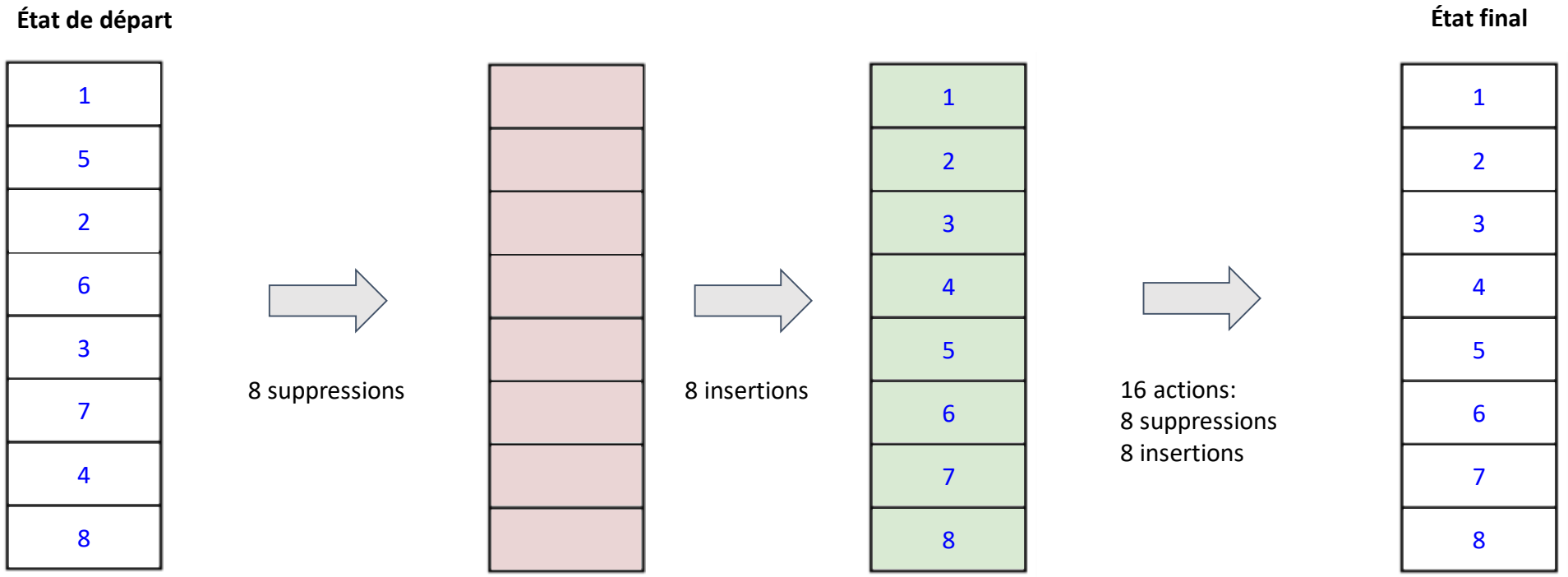


Remarques

Pour un adaptateur personnalisé, il est possible d'étendre RecyclerView.Adapter directement, mais à chaque fois qu'une mise à jour des données source est nécessaire, il faut supprimer toute la liste de données et la reconstruire. Cela est coûteux et inutile. Surtout si la source de données précédente et la source de données mise à jour sont identiques ou presque. Il est préférable d'utiliser ListAdapter.

Copyright - Tout droit réservé - OFPPT

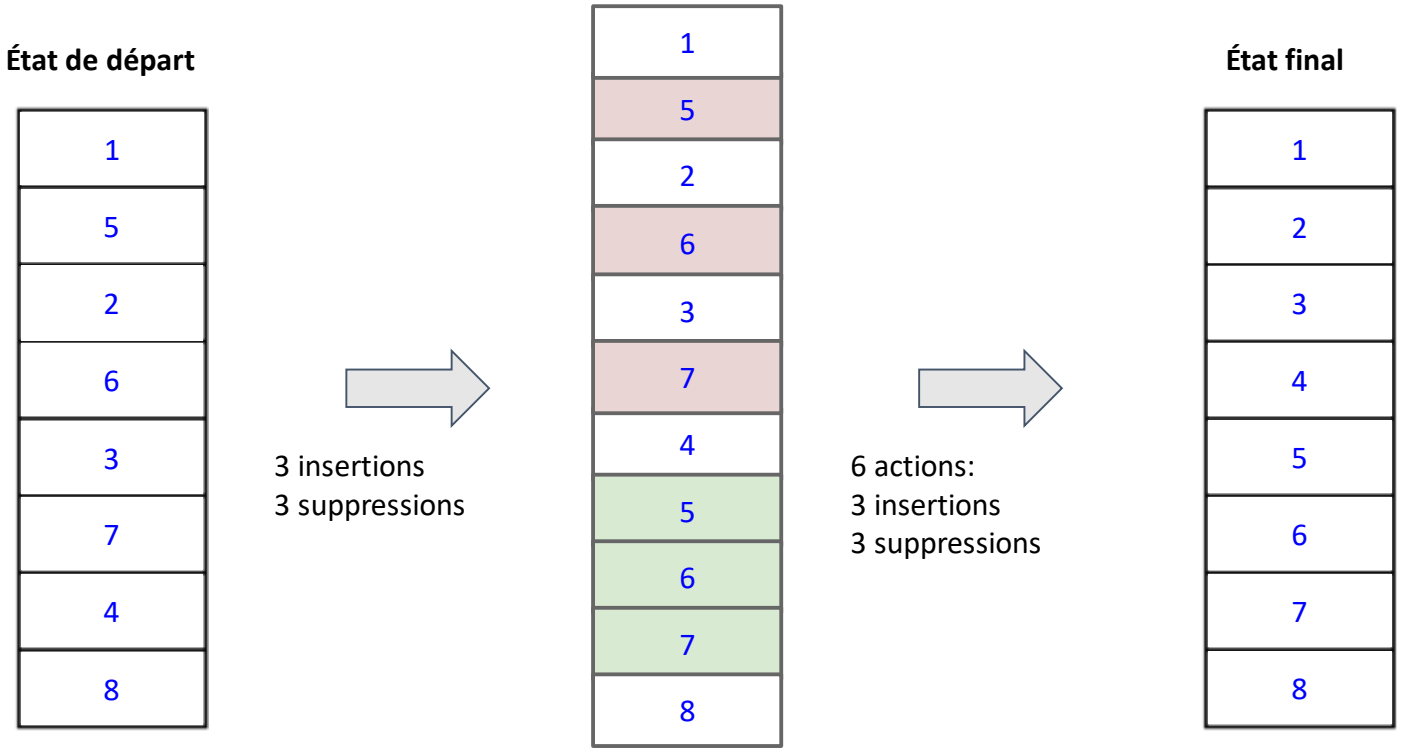
Trier en utilisant RecyclerView.Adapter



PARTIE 1

Copyright - Tout droit réservé - OFPPT

Trier en utilisant ListAdapter



PARTIE 1

Copyright - Tout droit réservé - OFPPT

Exemple de ListAdapter

```
class NumberListAdapter: ListAdapter<Int,
        NumberListAdapter.IntViewHolder>(RowItemDiffCallback()) {


    class IntViewHolder(val row: View):RecyclerView.ViewHolder(row) {
        val textView = row.findViewById<TextView>(R.id.number)
    }

    ...
```

Remarques

Il y a peu de changements entre le code de l'adaptateur précédent et celui de ListAdapter. Le premier changement majeur est l'ajout du type de données de la liste. Au lieu d'étendre RecyclerView.Adapter<ViewHolder>, nous étendons ListAdapter<Int, ViewHolder>. Le deuxième changement est le passage d'un objet DiffUtil.ItemCallback qui détermine comment différencier deux listes.

DiffUtil.ItemCallback

Détermine les transformations nécessaires pour traduire une liste en une autre :

- areContentsTheSame(oldItem: T, newItem: T): Boolean
- areItemsTheSame(oldItem: T, newItem: T): Boolean

Remarques

DiffUtil.ItemCallback spécifie une fonction de rappel pour calculer la différence entre deux éléments non nuls d'une liste. Il possède deux fonctions abstraites qui doivent être surchargées : areContentsTheSame() et areItemsTheSame(). Bien que ces méthodes semblent similaires, le fait d'avoir les deux permet de mieux contrôler la façon dont les données sont traitées. Par exemple, il se peut que dans un ensemble de données, deux vues d'éléments de liste semblent visuellement identiques à l'utilisateur, mais représentent en réalité deux éléments différents dans les données source.

Copyright - Tout droit réservé - OFPPT

Exemple de DiffUtil.ItemCallback

```
class RowItemDiffCallback : DiffUtil.ItemCallback<Int>() {

   override fun areItemsTheSame(oldItem: Int, newItem: Int): Boolean {
       return oldItem == newItem
   }

   override fun areContentsTheSame(oldItem: Int, newItem: Int): Boolean {
       return oldItem == newItem
   }
}
```

Remarques

Plus précisément, areItemsTheSame() est appelée pour vérifier si l'ancien et le nouvel élément représentent le même élément. Par exemple, si les éléments ont des identifiants uniques, cette méthode doit vérifier l'égalité des identifiants.

areContentTheSame()est appelée pour vérifier si oldItem et newItem ont les mêmes données. Lorsque DiffUtil est utilisé avec un RecyclerView.Adapter, il convient de retourner si les représentations visuelles des deux éléments sont les mêmes.

Comme la liste est adossée à une liste d'entiers, les fonctions areItemsTheSame() et areContentsTheSame() ont le même code.

Copyright - Tout droit réservé - OFPPT

ViewHolders et data binding

```
class IntViewHolder private constructor(val binding: ItemViewBinding):
        RecyclerView.ViewHolder(binding.root) {

    companion object {
        fun from(parent: ViewGroup): IntViewHolder {
            val layoutInflater = LayoutInflater.from(parent.context)
            val binding = ItemViewBinding.inflate(layoutInflater,
                parent, false)
            return IntViewHolder(binding)
        }
    }
}
```

Utilisation du ViewHolder dans un ListAdapter

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int):
        IntViewHolder {
    return IntViewHolder.from(parent)
}

override fun onBindViewHolder(holder: NumberListAdapter.IntViewHolder,
        position: Int) {
    holder.binding.num = getItem(position)
}
```


Copyright - Tout droit réservé - OFPPT

Binding adapters (adaptateurs de liaison)

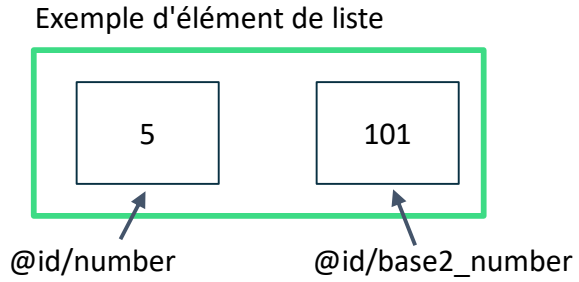
Permet d'associer une fonction à un attribut dans le fichier XML.

- Remplacer le comportement du cadre existant:
  - android:text = "foo" → TextView.setText("foo") est appelé.
- Créer les attributs personnalisés :
  - app:base2Number = "5" → TextView.setBase2Number("5") est appelé.

Attribut personnalisé

Ajouter un autre TextView dans la disposition de l'élément de liste qui utilise un attribut personnalisé :

```
<TextView
    android:id="@+id/base2_number"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:textSize="24sp"
    app:base2Number="@{num}"/>
```



PARTIE 1

Copyright - Tout droit réservé - OFPPT

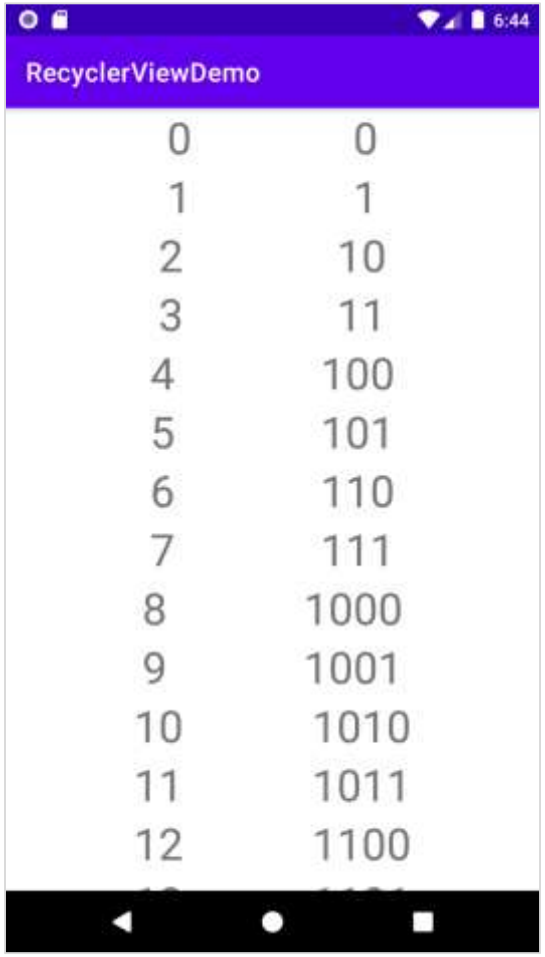
Ajouter un adaptateur de liaison

Déclarer l'adaptateur de liaison :

```
@BindingAdapter("base2Number")
fun TextView.setBase2Number(item: Int) {
    text = Integer.toBinaryString(item)
}
```

Dans NumberListAdapter.kt:

```
override fun onBindViewHolder(holder:
NumberListAdapter.IntViewHolder,
        position: Int) {
    holder.binding.num = getItem(position)
    holder.binding.executePendingBindings()
}
```

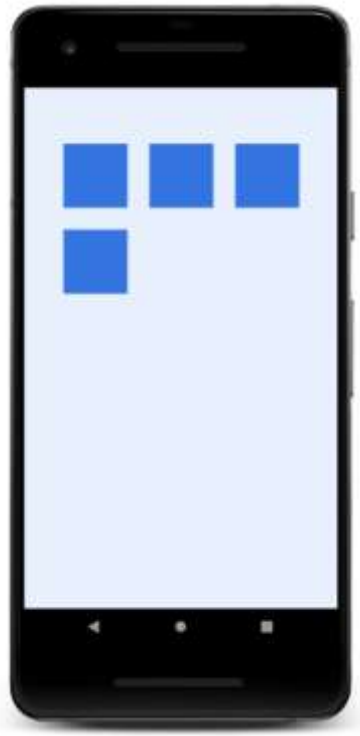


Copyright - Tout droit réservé - OFPPT

List versus grid

Jusqu'à présent, le RecyclerView est utilisé pour afficher une liste verticale de données. Le RecyclerView peut également afficher les données dans d'autres dispositions, comme une grille.





Spécifier un LayoutManager

Dans MainActivity onCreate(), une fois que la référence au RecyclerView a été créée.

- Afficher une liste avec LinearLayoutManager :

  recyclerView.layoutManager = LinearLayoutManager(this).

- Afficher une grille avec GridLayoutManager :

  recyclerView.layoutManager = GridLayoutManager(this, 2).

- Utiliser un gestionnaire de mise en page différent (ou créer son propre gestionnaire).

Copyright - Tout droit réservé - OFPPT

GridLayoutManager

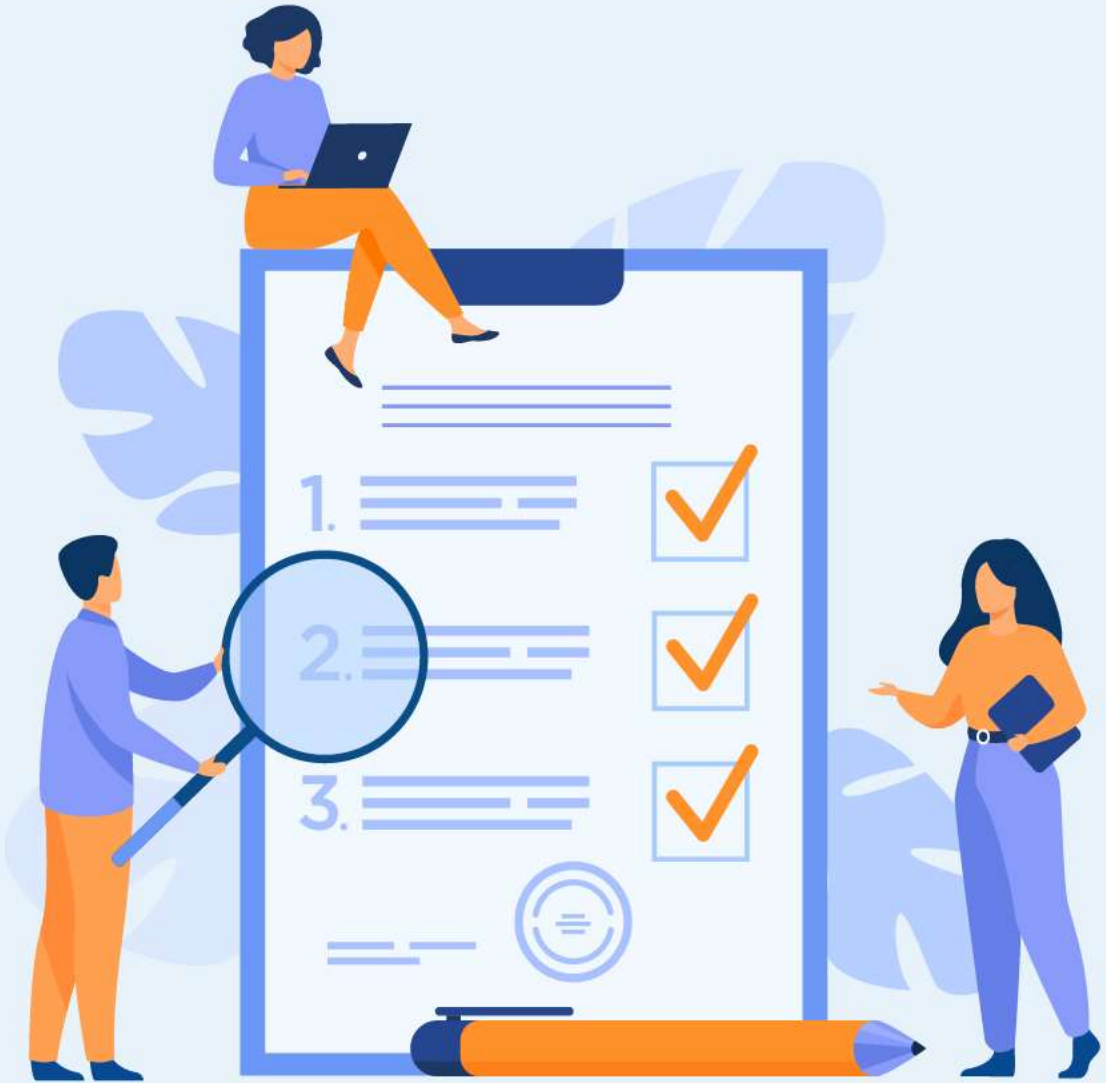
- Organise les éléments dans une grille sous la forme d'un tableau de lignes et de colonnes ;
- L'orientation peut être verticale ou horizontale ;
- Par défaut, chaque élément occupe une plage ;
- La taille de la plage occupée par un élément peut varier (taille de la plage).

Définir la taille de la plage pour un élément

Créer une instance de SpanSizeLookup et remplacer getSpanSize(position) :

```
val manager = GridLayoutManager(this, 2)
manager.spanSizeLookup = object : GridLayoutManager.SpanSizeLookup() {
    override fun getSpanSize(position: Int): Int {
        return when (position) {
            0,1,2 -> 2
            else -> 1
        }
    }
}
```


Copyright - Tout droit réservé - OFPPT



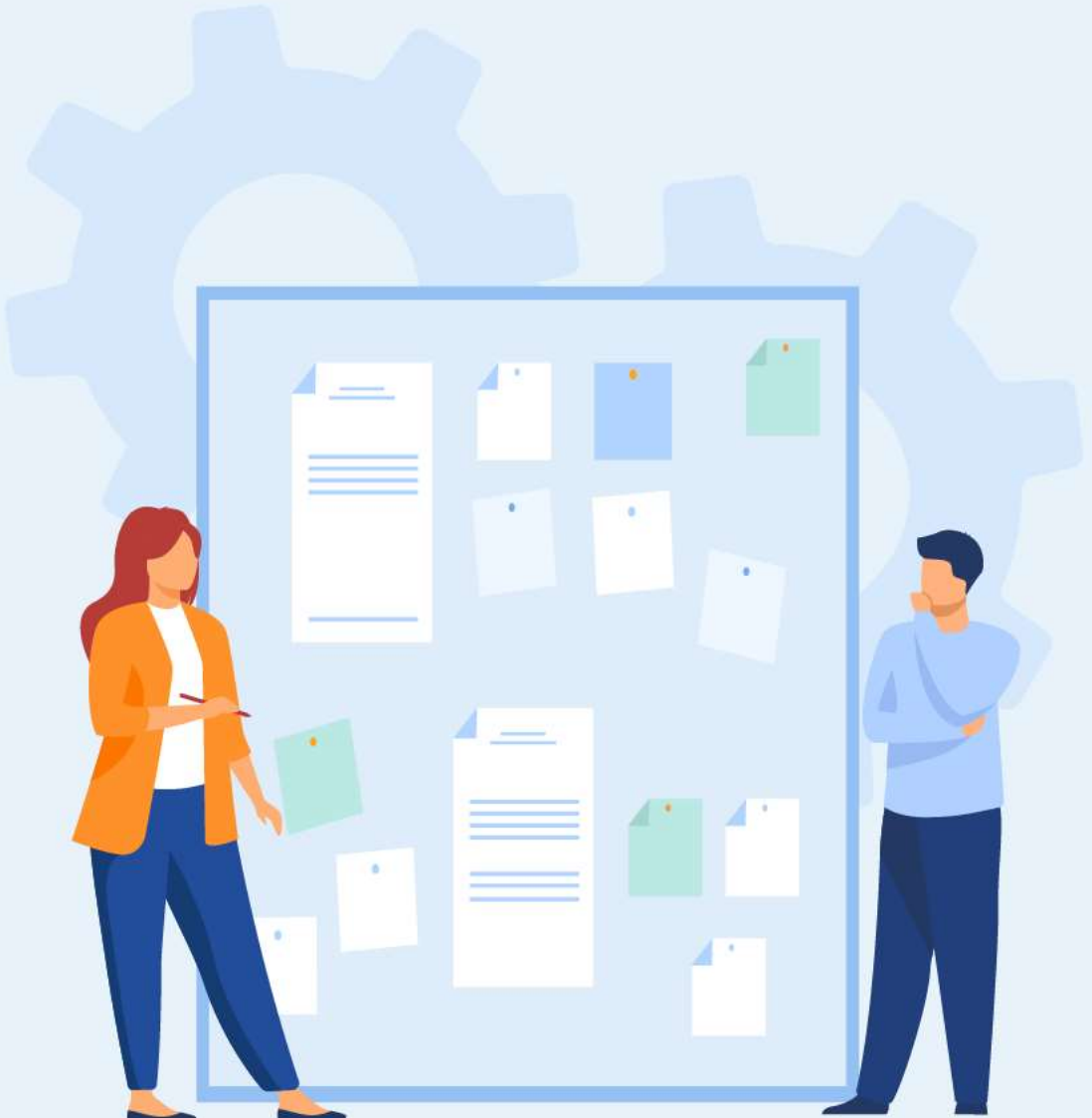
CHAPITRE 2

Utiliser les Layouts

Ce que vous allez apprendre dans ce chapitre :

- Définition d’un Layout
- Présentation des types des layouts

2 heures



CHAPITRE 2

Utiliser les Layouts

1. **Définition des Layouts**
2. Présentation des Layouts

Définition

Un layout définit la structure d'une interface utilisateur dans l'application, par exemple dans une activité. Tous les éléments du layout sont construits à l'aide d'une hiérarchie d'objets View et ViewGroup. Une vue représente généralement un élément que l'utilisateur peut voir et avec lequel il peut interagir. Alors qu'un ViewGroup est un conteneur invisible qui définit la structure du layout pour les objets View et autres ViewGroup, comme le montre la figure.

Exemple

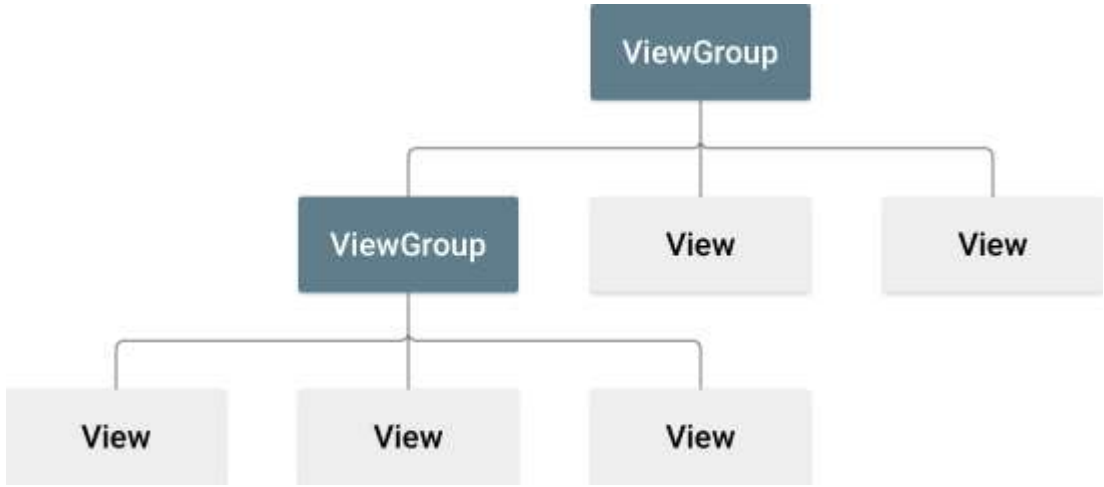


Figure : Illustration d'une hiérarchie de vues, qui définit la disposition de l'interface utilisateur.

Les objets View sont généralement appelés "widgets" et peuvent être l'une des nombreuses sous-classes, telles que Button ou TextView. Les objets ViewGroup sont généralement appelés "layouts" et peuvent être l'un des nombreux types qui fournissent une structure de disposition différente, comme LinearLayout ou ConstraintLayout .

Copyright - Tout droit réservé - OFPPT

Paramètres de layout

Les attributs de layout XML nommés layout_something définissent les paramètres de layout de la vue qui sont appropriés pour le ViewGroup dans lequel elle réside.

Chaque classe ViewGroup implémente une classe imbriquée qui étend ViewGroup.LayoutParams. Cette sous-classe contient des types de propriétés qui définissent la taille et la position de chaque vue enfant, en fonction du groupe de vues. Comme le montre la figure à côté, le groupe de vues parent définit les paramètres de mise en page pour chaque vue enfant (y compris le groupe de vues enfant).

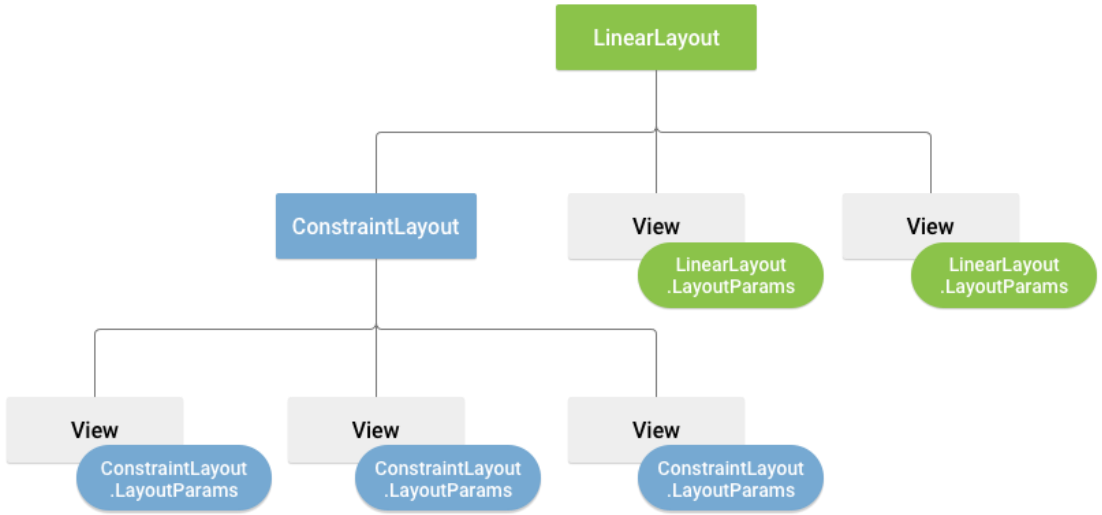


Figure : Visualisation d'une hiérarchie de vues avec des paramètres de mise en page associés à chaque vue.

Copyright - Tout droit réservé - OFPPT

Paramètres de layout

À noter que chaque sous-classe de LayoutParams possède sa propre syntaxe pour définir les valeurs. Chaque élément enfant doit définir les LayoutParams qui conviennent à son parent, bien qu'il puisse également définir des LayoutParams différents pour ses propres enfants.

Tous les groupes de vues comprennent une largeur et une hauteur (layout_width et layout_height), et chaque vue doit les définir. De nombreux LayoutParams comprennent également des marges et des bordures facultatives.

Il est possible de spécifier la largeur et la hauteur avec des mesures exactes, mais cela n'est pas souvent le cas. Le plus souvent, on utilise l'une de ces constantes pour définir la largeur ou la hauteur :

- **wrap_content** indique à la vue de s'adapter aux dimensions requises par son contenu ;
- **match_parent** indique à la vue de devenir aussi large que son groupe de vues parent le permet.

En général, il n'est pas recommandé de spécifier la largeur et la hauteur d'une mise en page en utilisant des unités absolues telles que les pixels. Il est préférable d'utiliser des mesures relatives telles que les unités de pixel indépendantes de la densité (dp), wrap_content ou match_parent, car cela permet de garantir que l'application s'affichera correctement sur des écrans de tailles différentes.

Position de layout

La géométrie d'une vue est celle d'un rectangle. Une vue possède un emplacement, exprimé par une paire de coordonnées gauche et supérieure, et deux dimensions, exprimées par une largeur et une hauteur. L'unité pour l'emplacement et les dimensions est le pixel.

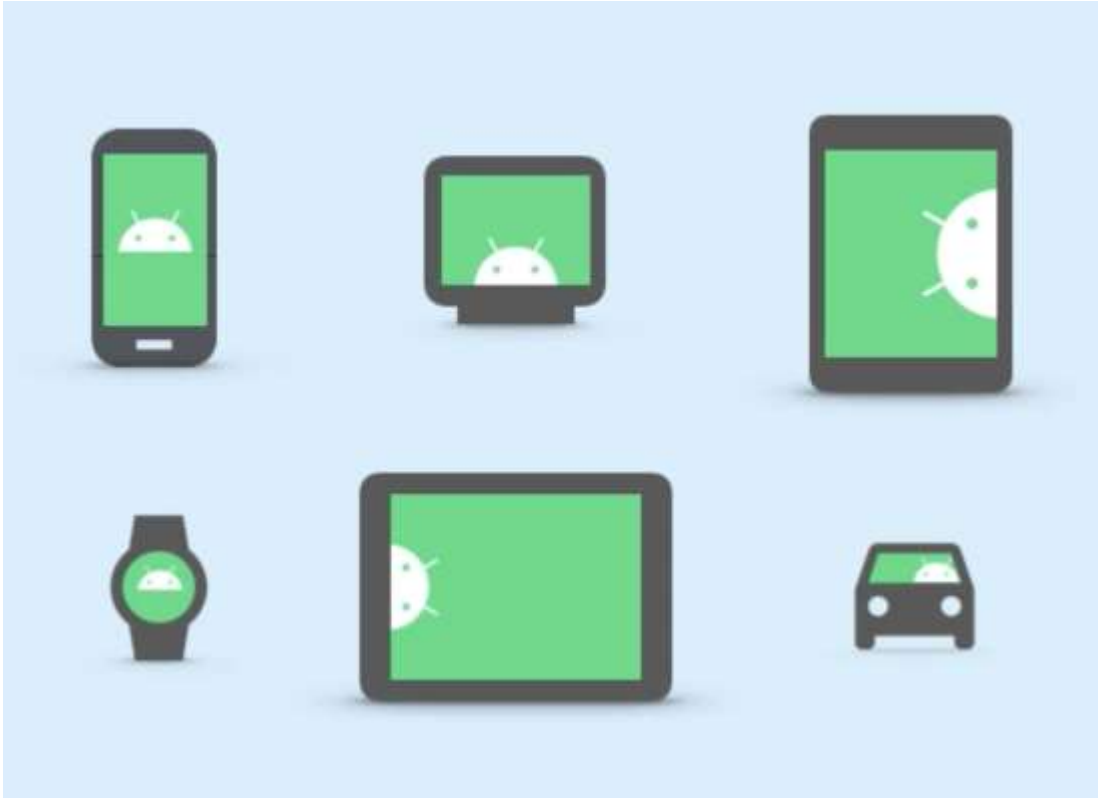
Il est possible de récupérer l'emplacement d'une vue en invoquant les méthodes getLeft() et **getTop**(). La première renvoie la coordonnée gauche, ou X, du rectangle représentant la vue. La seconde renvoie la coordonnée supérieure, ou Y, du rectangle représentant la vue. Ces méthodes renvoient toutes deux l'emplacement de la vue par rapport à son parent. Par exemple, lorsque getLeft() renvoie 20, cela signifie que la vue est située à 20 pixels à droite du bord gauche de son parent direct.

En outre, plusieurs méthodes pratiques sont proposées pour éviter les calculs inutiles, à savoir getRight() et getBottom(). Ces méthodes renvoient les coordonnées des bords droit et inférieur du rectangle représentant la vue. Par exemple, l'appel de getRight() est similaire au calcul suivant : getLeft() + getWidth().

Copyright - Tout droit réservé - OFPPT

Android devices

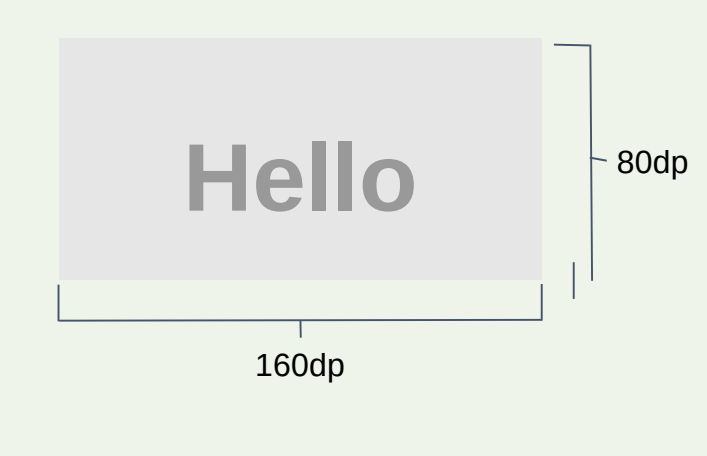
- Les appareils Android se déclinent en de nombreux facteurs de forme différents ;
- Les écrans des appareils contiennent de plus en plus de pixels par pouce ;
- Les développeurs doivent pouvoir spécifier des dimensions de mise en page qui soient cohérentes d'un appareil à l'autre.



Copyright - Tout droit réservé - OFPPT

Density-independent pixels (dp)

Utiliser dp pour spécifier les tailles dans la mise en page, comme la largeur ou la hauteur des vues.



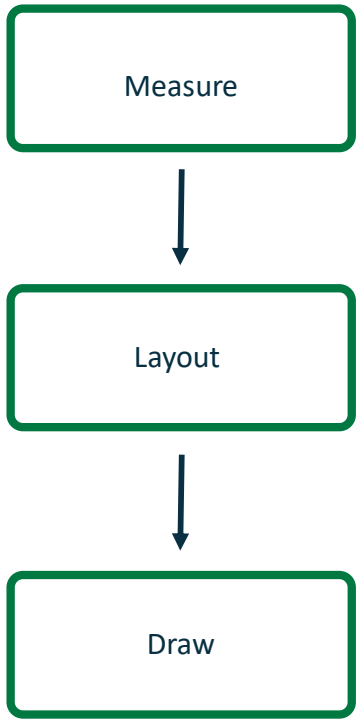
- Les pixels indépendants de la densité (dp) tiennent compte de la densité de l'écran ;
- Les vues Android sont mesurées en pixels indépendants de la densité.
- dp = (largeur en pixels * 160)
  densité de l'écran

Screen-density buckets

Qualificateur de densité	Description	DPI estimate
ldpi (mostly unused)	Low density	~120dpi
mdpi (baseline density)	Medium density	~160dpi
hdpi	High density	~240dpi
xhdpi	Extra-high density	~320dpi
xxhdpi	Extra-extra-high density	~480dpi
xxxhdpi	Extra-extra-extra-high density	~640dpi

Copyright - Tout droit réservé - OFPPT

Cycle de rendu Android View



Zone de dessin

Ce que l'on voit :

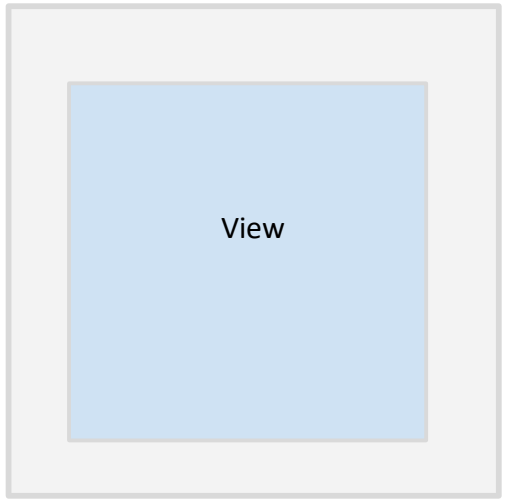
Comment c'est dessiné :



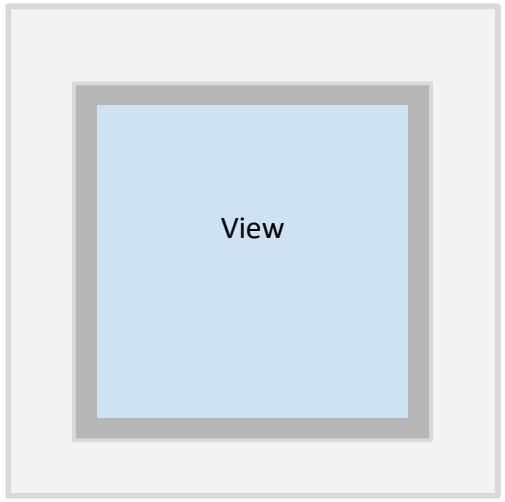
Copyright - Tout droit réservé - OFPPT

View avec margin et padding

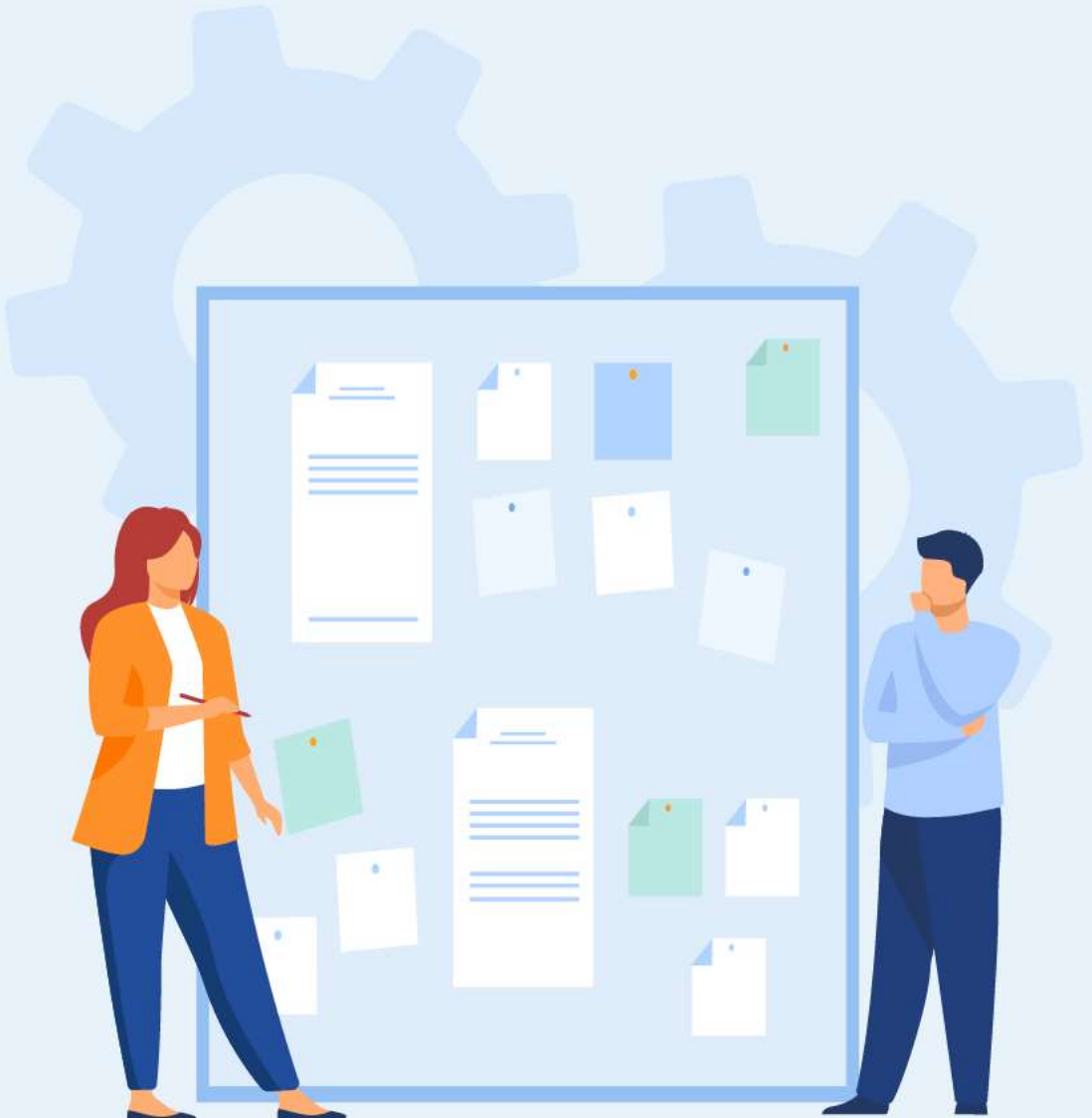
View avec margin



View avec margin et padding



Copyright - Tout droit réservé - OFPPT



CHAPITRE 2

Utiliser les Layouts

1. Définition des Layouts
2. **Présentation des Layouts**

LinearLayout

Ce layout se charge de mettre les vues sur une même ligne, selon une certaine orientation. L'attribut pour préciser cette orientation est « Android: orientation » . On peut lui donner deux valeurs :

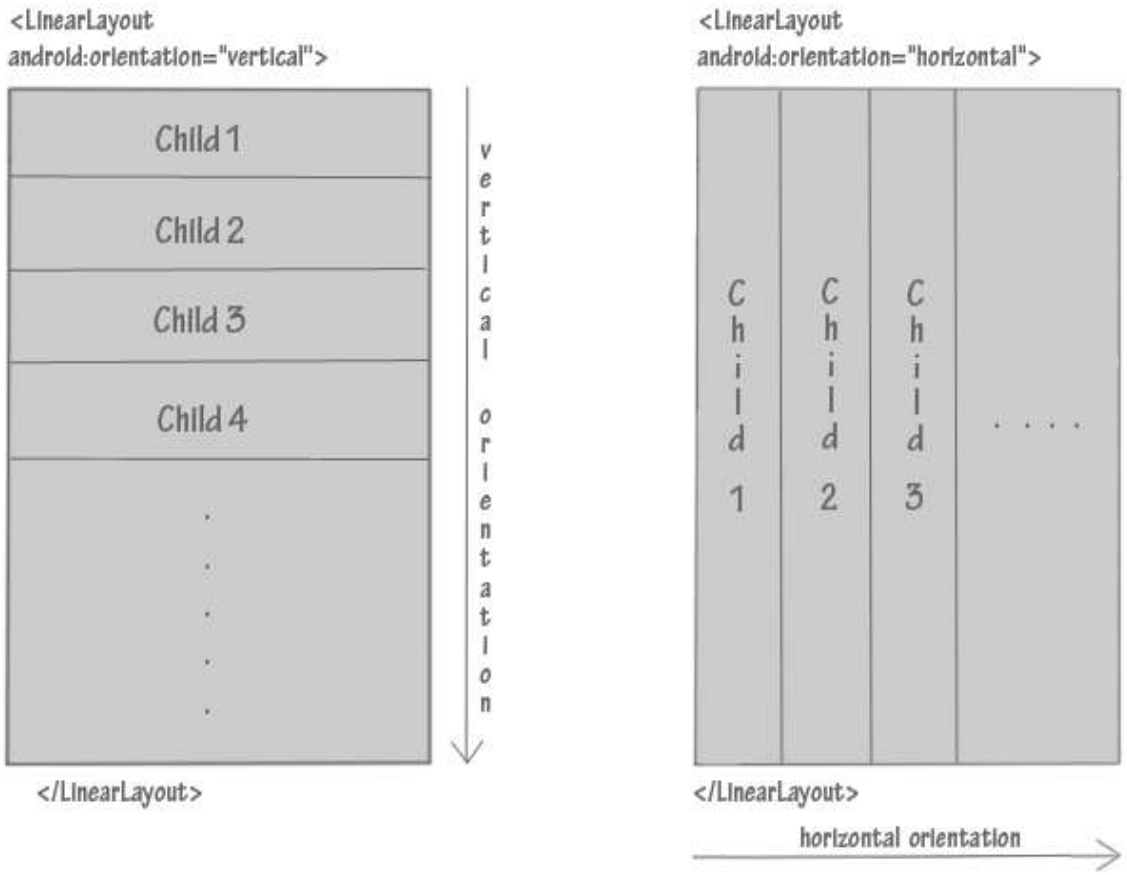
- vertical pour que les composants soient placés de haut en bas (en colonne) ;
- horizontal pour que les composants soient placés de gauche à droite (en ligne).

Code XML

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:orientation="vertical/horizontal" >
...
...
</LinearLayout>
```


Copyright - Tout droit réservé - OFPPT

LinearLayout



Copyright - Tout droit réservé - OFPPT

LinearLayout

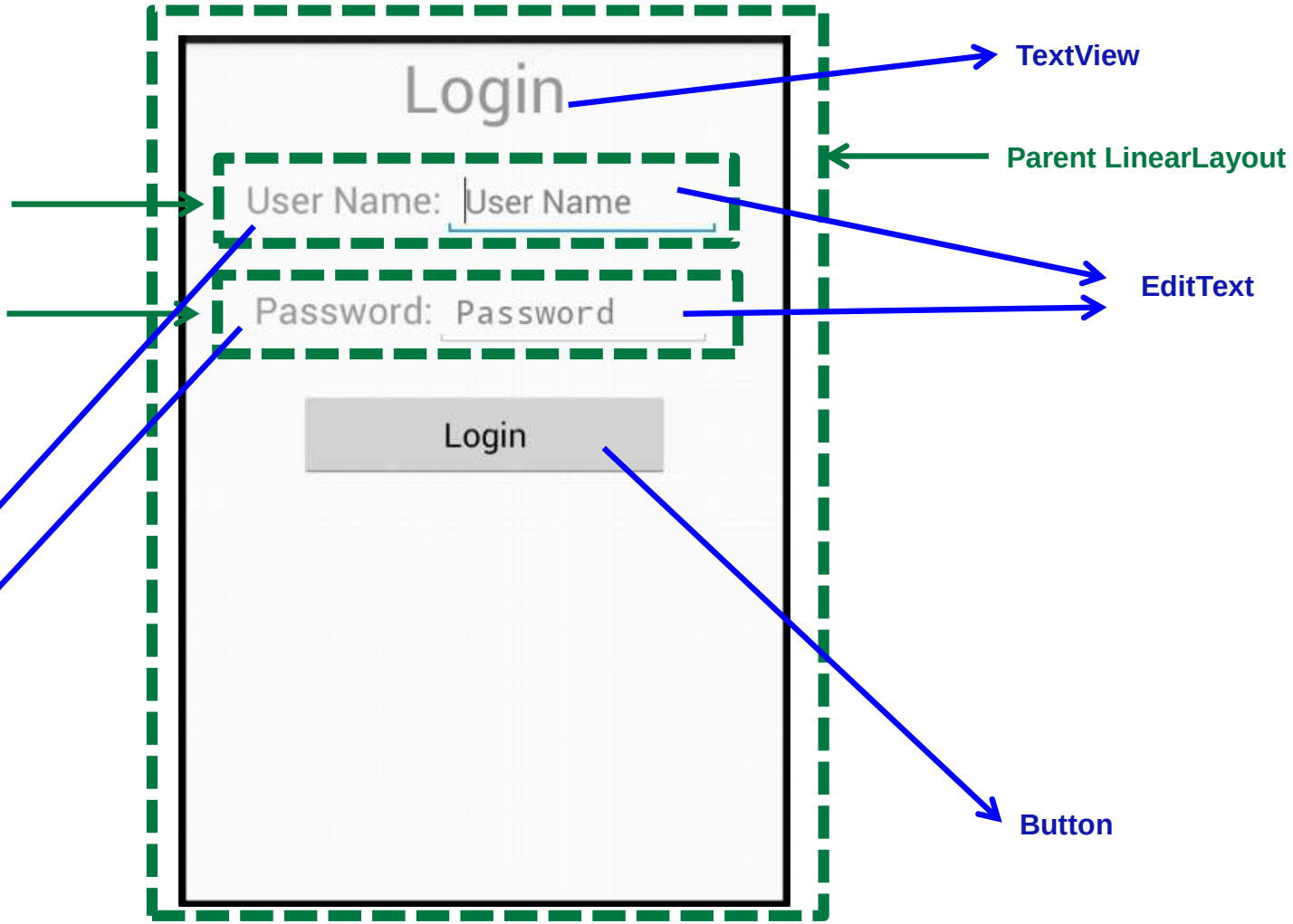
Démo 1 : Réaliser l'interface suivante



Copyright - Tout droit réservé - OFPPT

LinearLayout

Démo 1 : Réaliser l'interface suivante



Copyright - Tout droit réservé - OFPPT

RelativeLayout

De manière totalement différente, ce layout propose plutôt de placer les composants les uns par rapport aux autres. Il est même possible de les placer par rapport au RelativeLayout parent. Il permet en particulier de :

- placer les éléments du layout les uns par rapport aux autres : Below, Above, toLeft, toRight, ..etc ;
- placer les éléments dans une position bien définie au sein du layout: Top , Bottom, Right, Left, center Horizontal, Center Vertical.

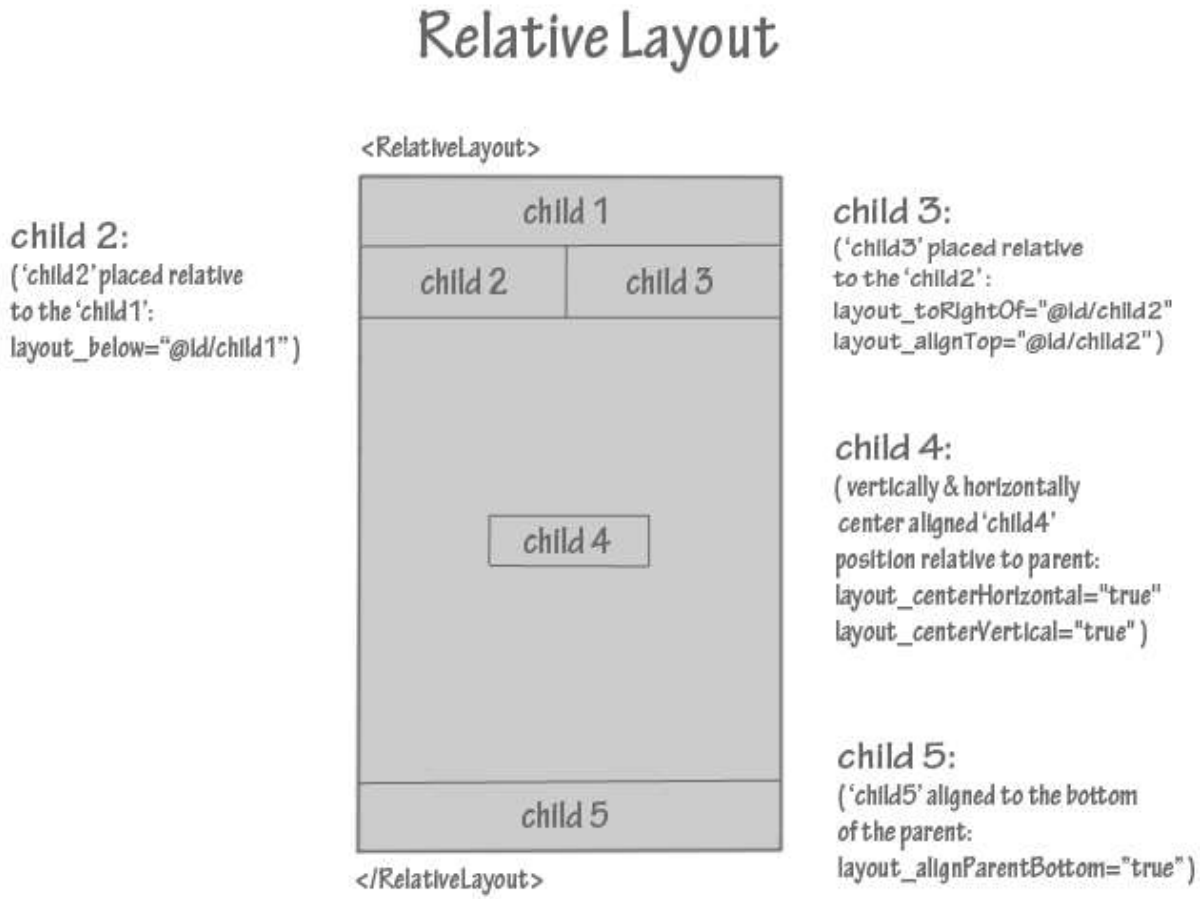
Code XML

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
android:layout_width="match_parent"
android:layout_height="match_parent" >
...
...
</RelativeLayout>
```


Copyright - Tout droit réservé - OFPPT

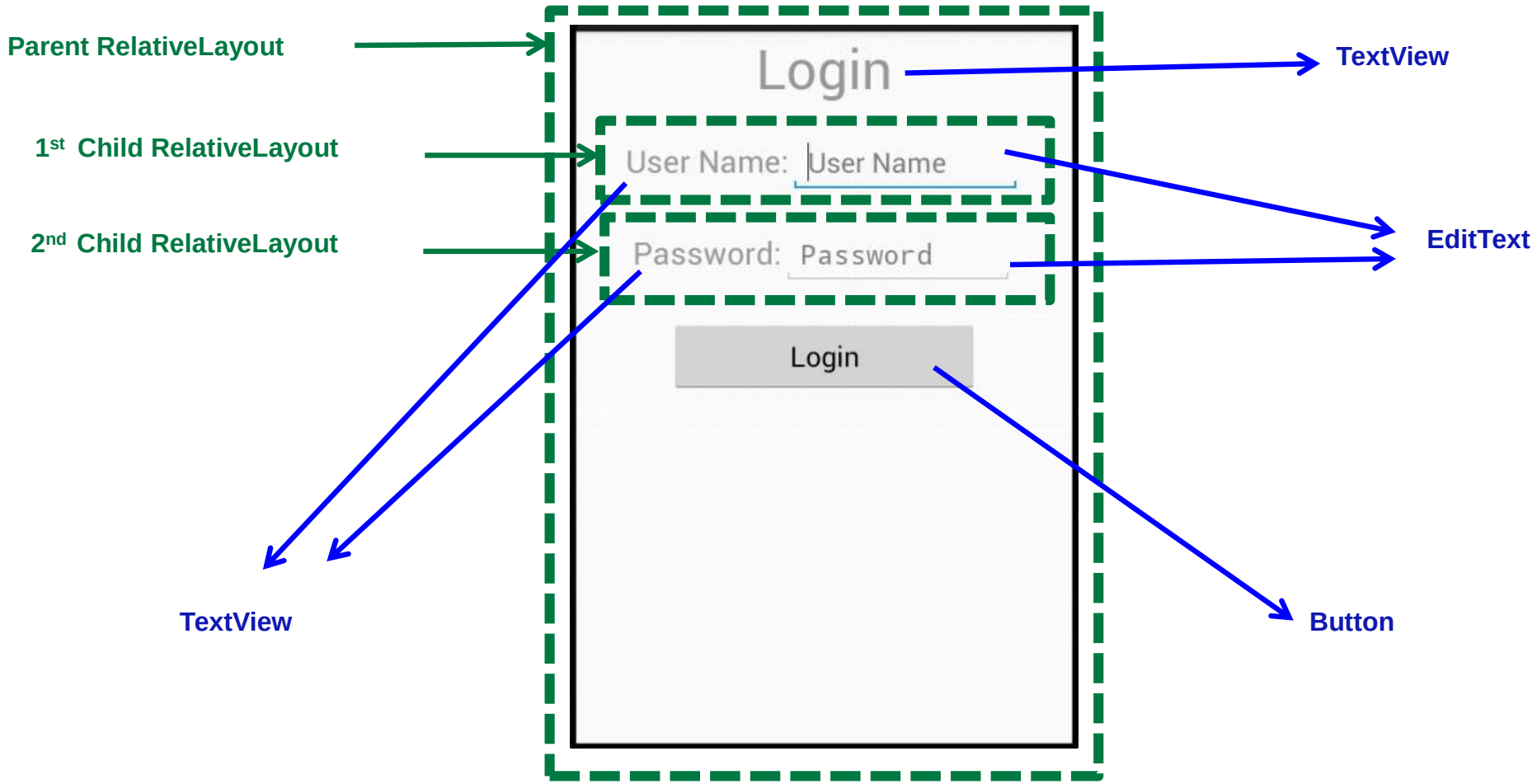
PARTIE 1

RelativeLayout



Copyright - Tout droit réservé - OFPPT

RelativeLayout



Copyright - Tout droit réservé - OFPPT

TableLayout

Dernier layout de base, il permet d'organiser les éléments en tableau, comme en HTML, mais sans les bordures.

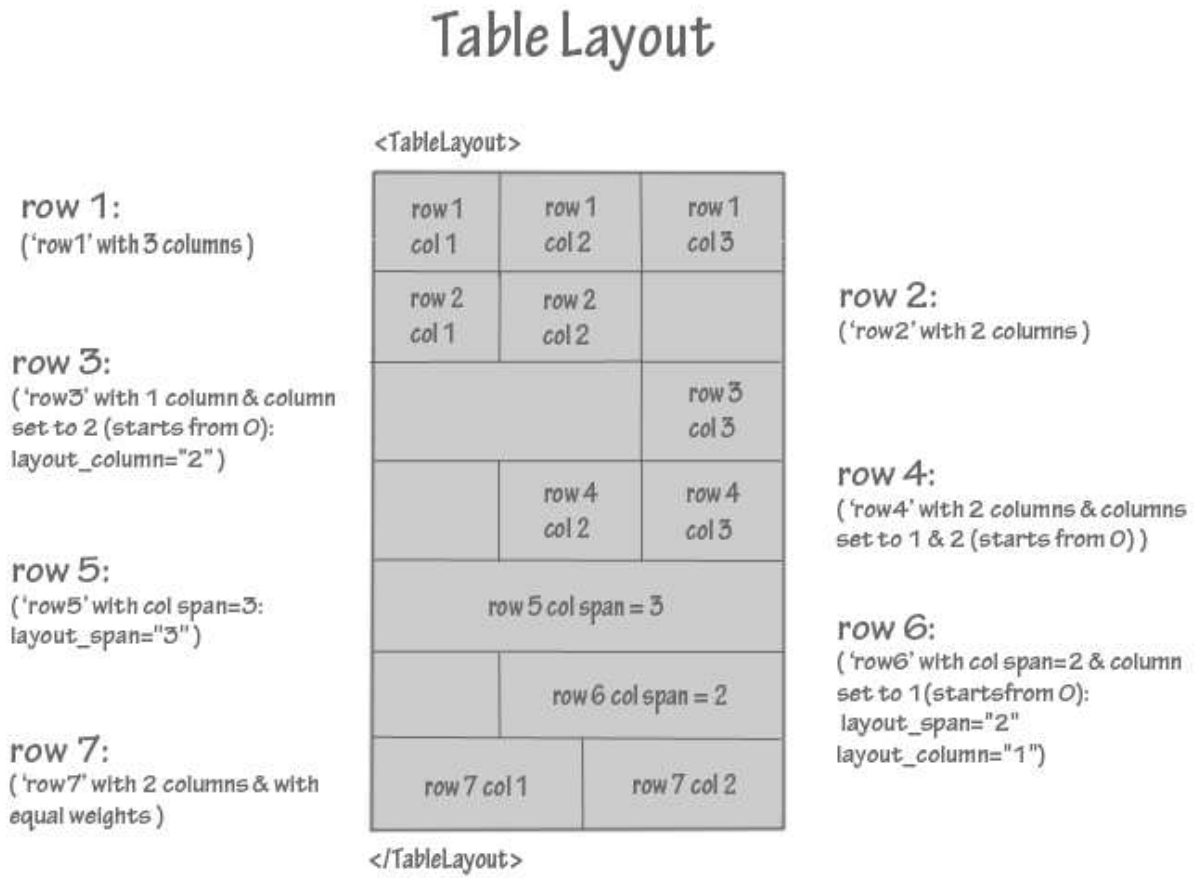
Code XML

```
<?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>
…
…
</TableRow>
 <TableRow>
…..
….
</TableRow>
</TableLayout>
```


Copyright - Tout droit réservé - OFPPT

PARTIE 1

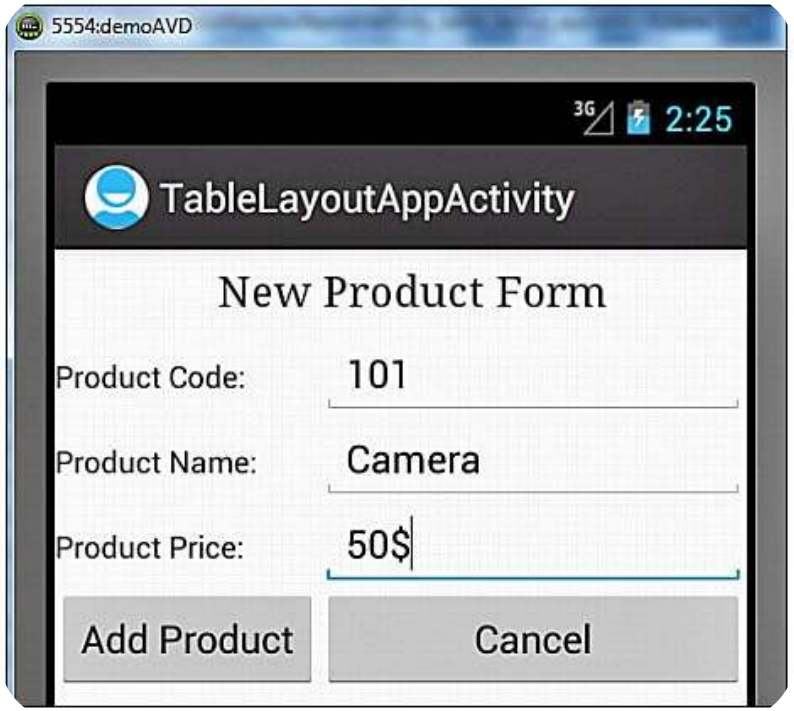
TableLayout



Copyright - Tout droit réservé - OFPPT

PARTIE 1

TableLayout



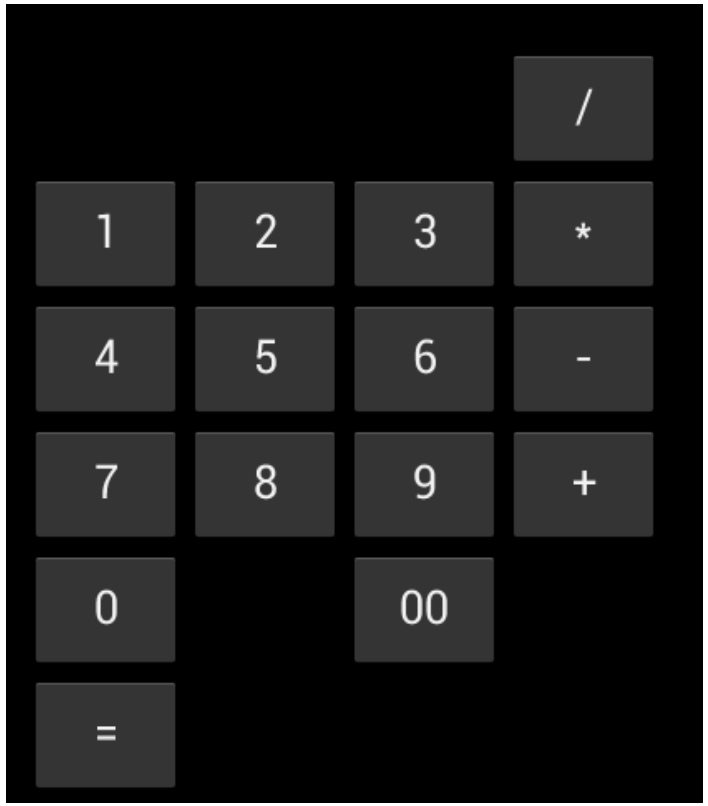
Code XML

```
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
  android:orientation="vertical"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:stretchColumns="1">
  <TableRow android:padding="5dp">
    <TextView
      android:layout_height="wrap_content"
      android:text="New Product Form"
      android:typeface="serif"
      android:layout_span="2"
      android:gravity="center_horizontal"
      android:textSize="20dp" />
  </TableRow>
  <TableRow>
    <TextView
      android:layout_height="wrap_content"
      android:text="Product Code:"
      android:layout_column="0"/>
    <EditText
      android:id="@+id/prod_code"
      android:layout_height="wrap_content"
      android:layout_column="1"/>
  </TableRow>
```

Copyright - Tout droit réservé - OFPPT

GridLayout

Ce layout vient compléter le tablelayout en proposant de fusionner des cellules.



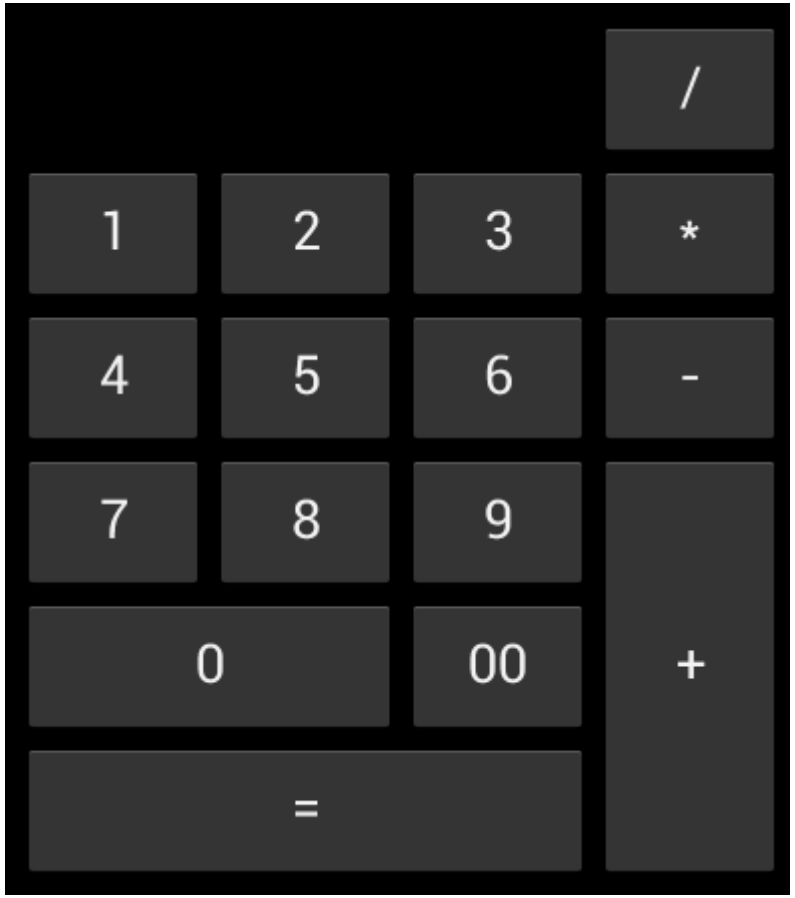
Code XML

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:layout_gravity="center"
   android:columnCount="4"
   android:orientation="horizontal" >
   <Button
      android:layout_column="3"
      android:text="/" />
   <Button android:text="1" />
  <!-- Other numbers -->
   <Button android:text="9" />
   <Button
      android:layout_rowSpan="3"
      android:text="+" />
   <Button
      android:layout_columnSpan="2"
      android:text="0" />
   <Button android:text="00" />
   <Button
      android:layout_columnSpan="3"
      android:text="=" />
</GridLayout>
```


Copyright - Tout droit réservé - OFPPT

PARTIE 1

GridLayout



Code XML

```
<Button
    android:layout_column="3"
    android:text="/" />

<Button android:text="1" />


<Button
    android:layout_gravity="fill"
    android:layout_rowSpan="3"
    android:text="+" />

<Button
    android:layout_columnSpan="2"
    android:layout_gravity="fill"
    android:text="0" />
```

Copyright - Tout droit réservé - OFPPT

FrameLayout

Il s'agit du layout le plus basique. Il permet de superposer des éléments, par exemple le logo Android avec un texte sur cette image.



Code XML

```
<FrameLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent" >
    <ImageView
      android:src="@drawable/ic_launcher"
      android:scaleType="fitCenter"
      android:layout_height="match_parent"
      android:layout_width="match_parent"/>

    <TextView
      android:text="Tutorial Android"
      android:textSize="30sp"
      android:textStyle="bold"
      android:textColor="#003399"
      android:layout_height="match_parent "
      android:layout_width="match_parent "
      android:gravity="center"/>
  </FrameLayout>
```

PARTIE 1

Copyright - Tout droit réservé - OFPPT

Notions communes aux différents layout

- **android:layout_gravity** : pour positionner un élément à gauche, au milieu ou à droite par rapport à son conteneur (permet de centrer du texte au milieu de l'écran) ;
- **android:inputType** : les zones de saisie de texte peuvent être typées : simple texte , numéro de téléphone, adresse email, mot de passe, ...

Par exemple android:inputType="textPassword"

- **android:hint** : permet de saisir dans une zone de texte une explication sur la zone de texte. Par exemple, indiquer qu'il s'agit de saisir son login ;
- **android:layout_weight** : permet de donner du poids à un élément, ce qui signifie que l'élément occupera plus ou moins d'espace.

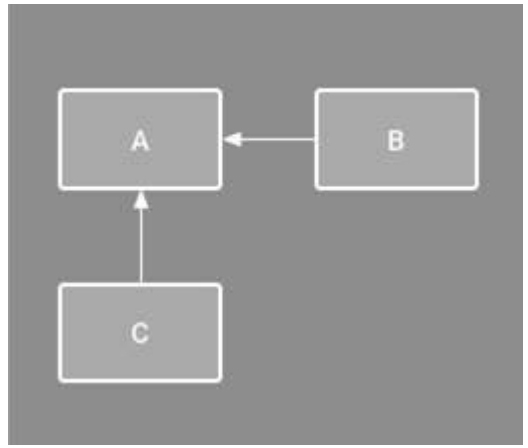
Copyright - Tout droit réservé - OFPPT

ConstraintLayout

- Disposition par défaut recommandée pour Android ;
- Résoudre le problème coûteux du trop grand nombre de mises en page imbriquées, tout en permettant un comportement complexe ;
- Positionner et dimensionner les vues à l'intérieur de la mise en page en utilisant un ensemble de contraintes.

Qu'est-ce qu'une contrainte ?

Une restriction ou une limitation des propriétés d'une vue que la mise en page tente de respecter.



Copyright - Tout droit réservé - OFPPT

Contraintes de positionnement relatif

Peut mettre en place une contrainte relative au conteneur parent.

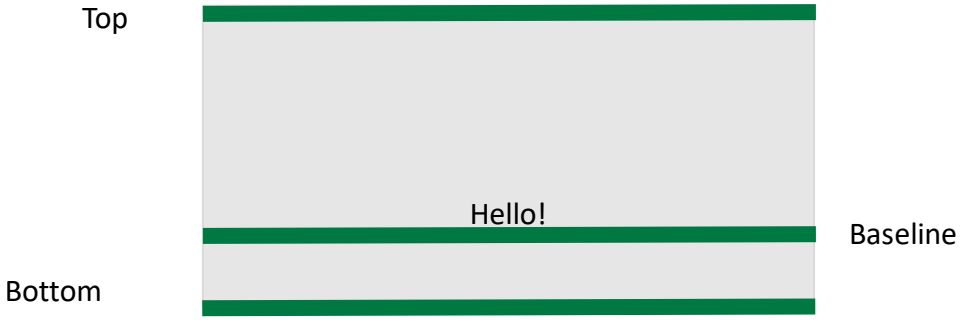
Format :

```
layout_constraint<SourceConstraint>_to<TargetConstraint>Of
```

Exemple attributes on a TextView:

```
app:layout_constraintTop_toTopOf="parent"
app:layout_constraintLeft_toLeftOf="parent"
```







PARTIE 1

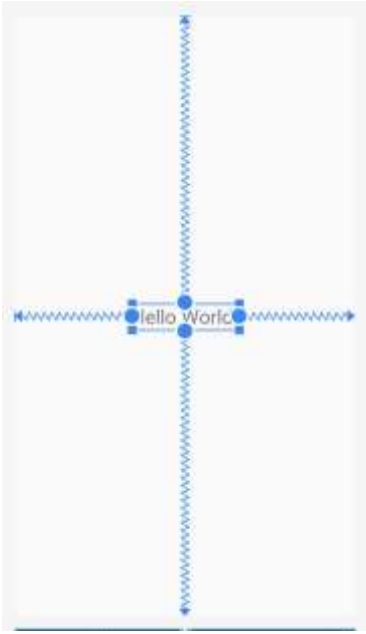
Copyright - Tout droit réservé - OFPPT

Exemple Simple ConstraintLayout

```
<androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <TextView
        ...
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

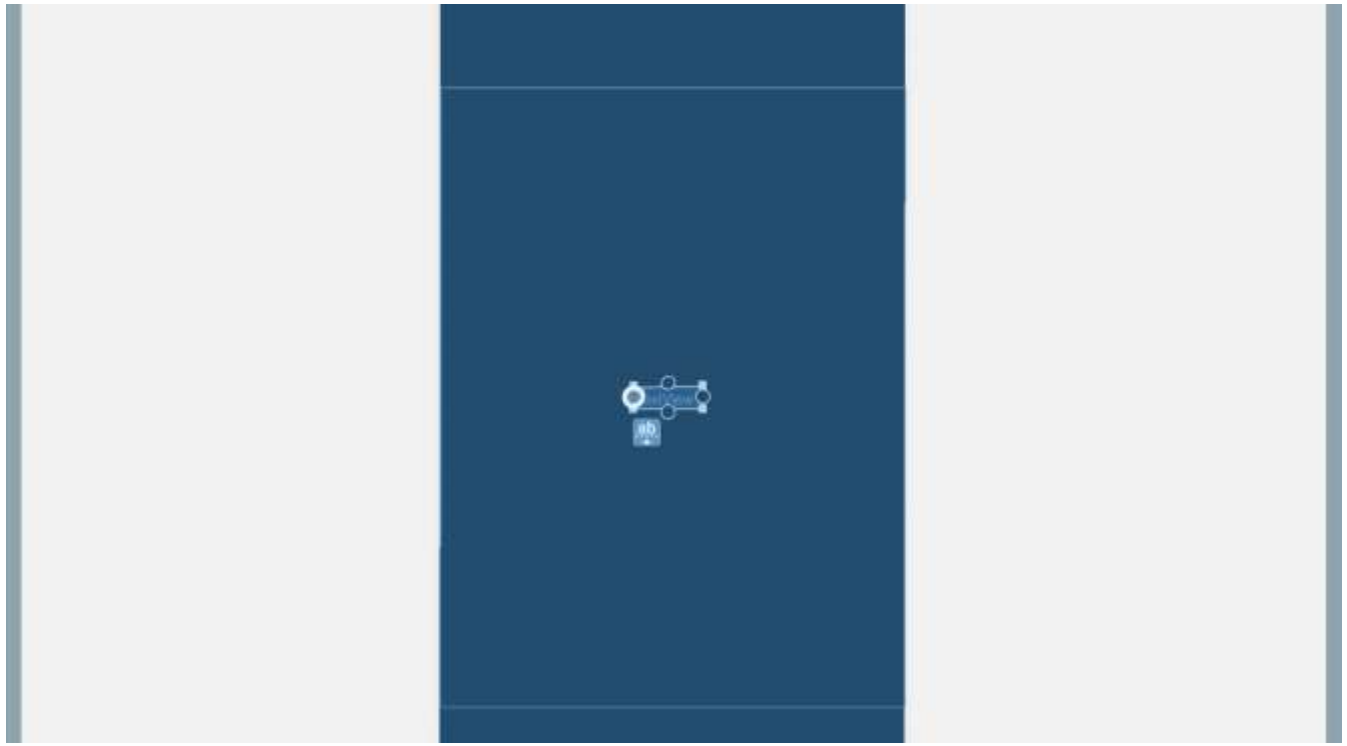
</androidx.constraintlayout.widget.ConstraintLayout>
```



Copyright - Tout droit réservé - OFPPT

Layout Editor dans Android Studio

Il est possible de cliquer et de faire glisser pour ajouter des contraintes à une vue.



Copyright - Tout droit réservé - OFPPT

Constraint Widget dans Layout Editor



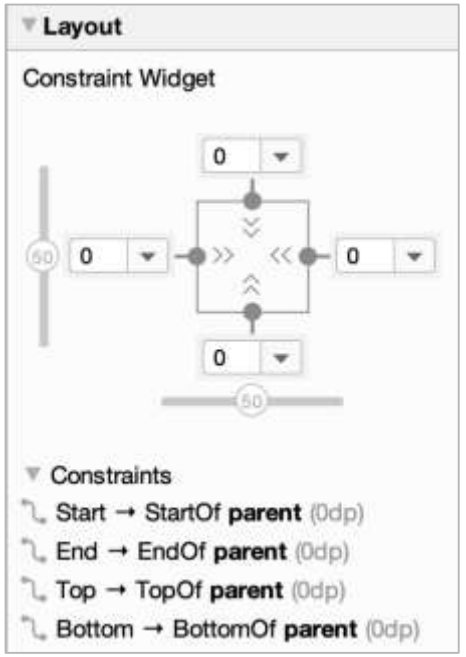
Fixed



Wrap content



Match constraints

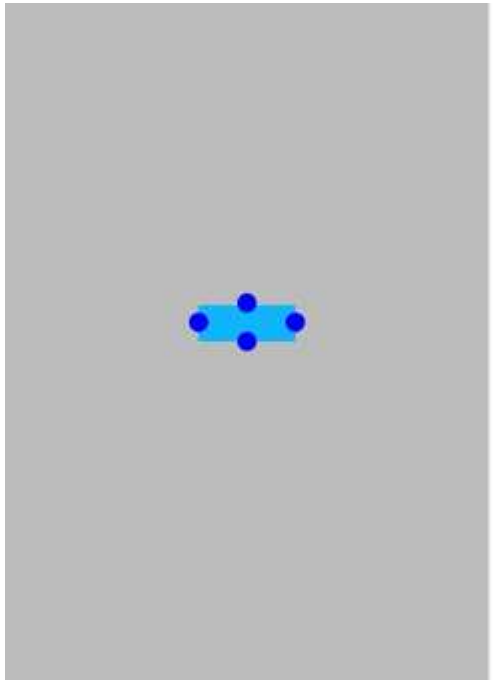


Dans l'éditeur de modèle, le widget Contrainte apparaît dans la fenêtre Attributs, à droite de l'écran. Trois types de symboles (illustrés plus haut) indiquent quel type de contrainte a été placé au début, à la fin, en haut et en bas de la vue actuellement sélectionnée.

PARTIE 1

Copyright - Tout droit réservé - OFPPT

Wrap content pour width et height



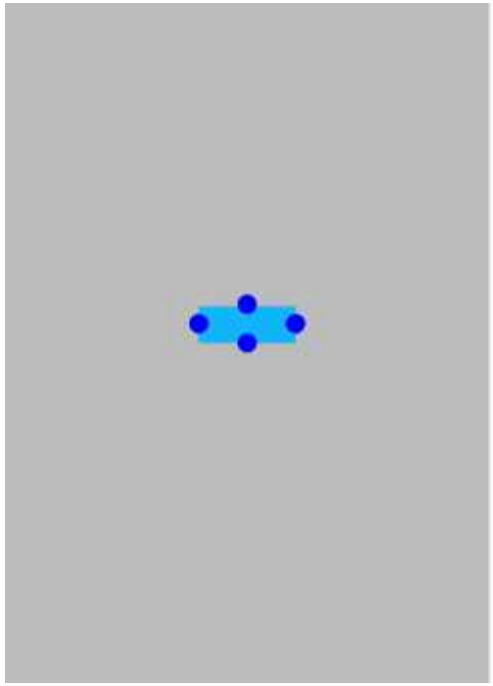


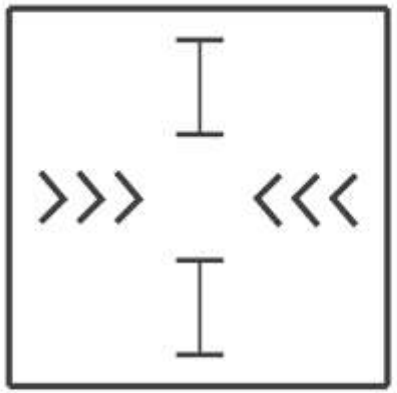
```
layout_width    wrap_content
layout_height   wrap_content
```

Le carré à droite représente une vue avec wrap_content sur sa hauteur et sa largeur.

Copyright - Tout droit réservé - OFPPT

Wrap content pour width, fixed height



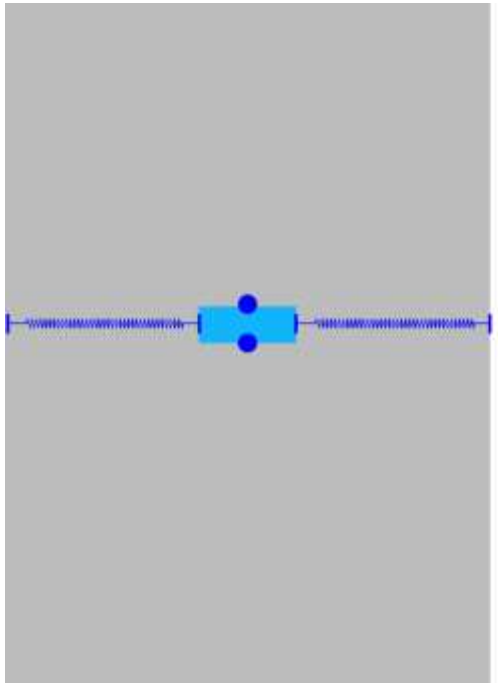


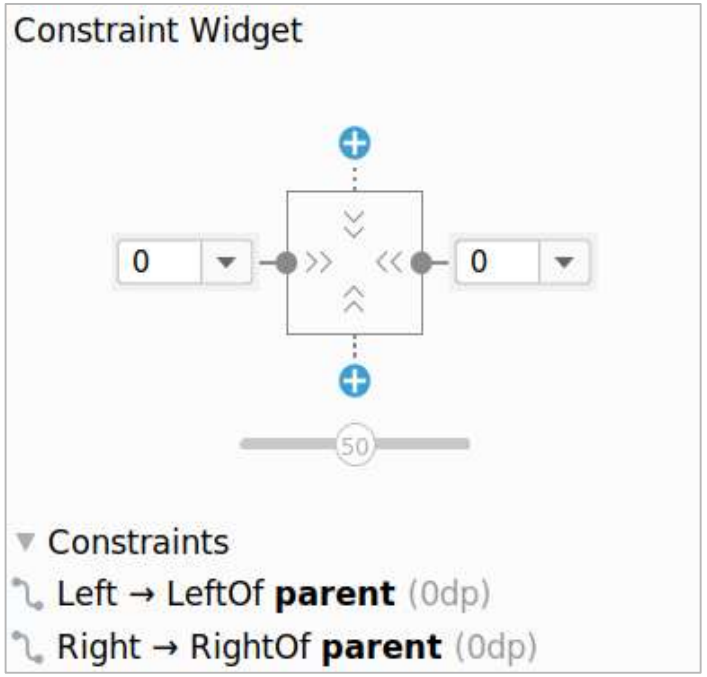
```
layout_width    wrap_content
layout_height   48dp
```

Il y a aussi un autre symbole ressemblant à un tuyau qui indique que la dimension a une taille fixe en dp (48dp) pour la hauteur de la vue. La largeur de la vue est wrap_content comme dans la diapositive précédente.

Copyright - Tout droit réservé - OFPPT

Center a view horizontally



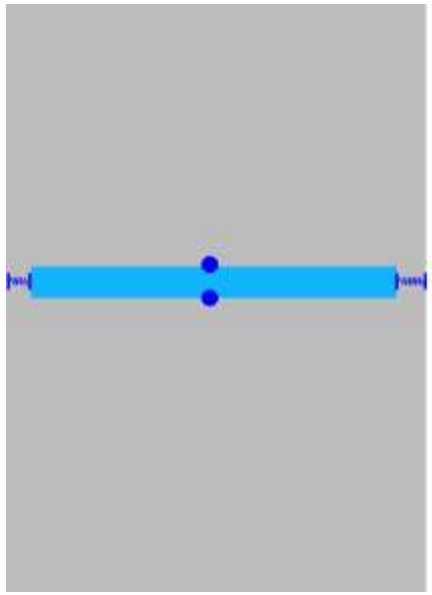


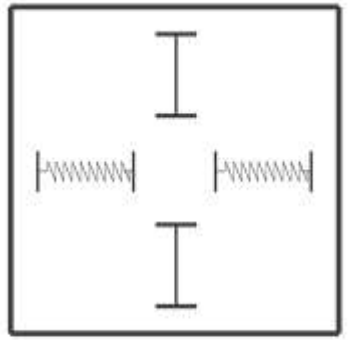
Une vue peut être soumise à plusieurs contraintes qui déterminent dans quelle mesure elle affectera la position finale. Dans ce cas, la vue est soumise à des contraintes de début et de fin dans le sens horizontal. Il y a également un biais de 50 (remarquer le curseur gris dans l'appel), de sorte que les contraintes sont appliquées de manière égale. Le résultat final est que la vue est centrée horizontalement dans le parent.

Copyright - Tout droit réservé - OFPPT

Utiliser match_constraint

Il est impossible d'utiliser match_parent sur une vue enfant, utiliser match_constraint à la place.





```
layout_width    0dp(match_constraint)
layout_height   48dp
```

Dans LinearLayout, il est possible d'utiliser match_parent pour indiquer que l'on souhaite qu'une vue occupe tout l'espace disponible. Dans ConstraintLayout, il est impossible d'utiliser match_parent sur une vue enfant, il faut donc utiliser match_constraint. En fonction de la version d'Android Studio utilisée, il se peut que la valeur 0dp (match_constraint) soit utilisée.

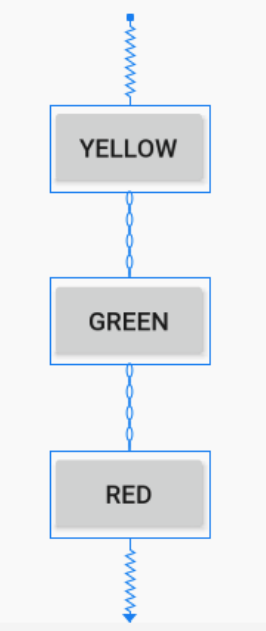
Copyright - Tout droit réservé - OFPPT

Chains

- Permet de positionner les vues les unes par rapport aux autres ;
- Peuvent être liées horizontalement ou verticalement ;
- Fournissent la plupart des fonctionnalités de LinearLayout.

Créer une chaîne dans l'éditeur de mise en page

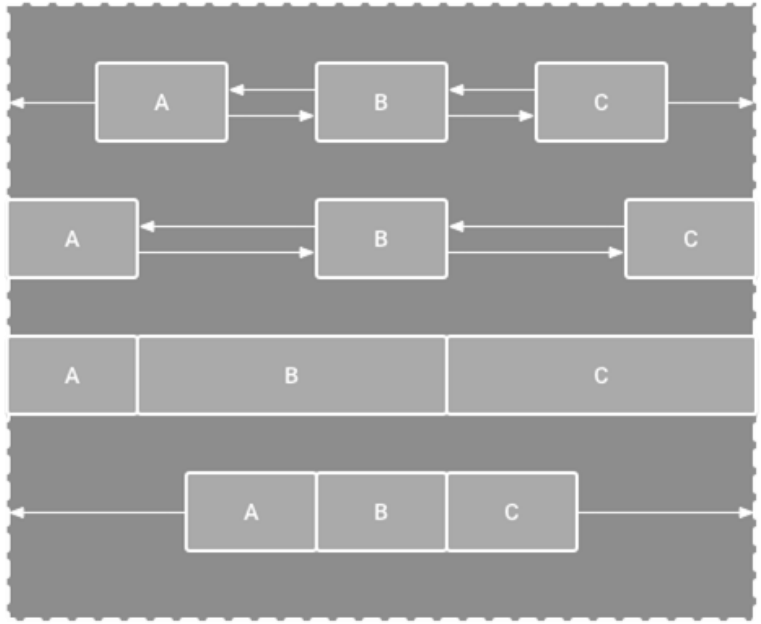
1. Sélectionner les objets à inclure dans la chaîne ;
2. Cliquer avec le bouton droit de la souris et sélectionner Chaînes ;
3. Créer une chaîne horizontale ou verticale.



Copyright - Tout droit réservé - OFPPT

Styles de chaînes

Ajuster l'espace entre les vues avec ces différents styles de chaînes.



Spread Chain

Spread Inside Chain

Weighted Chain

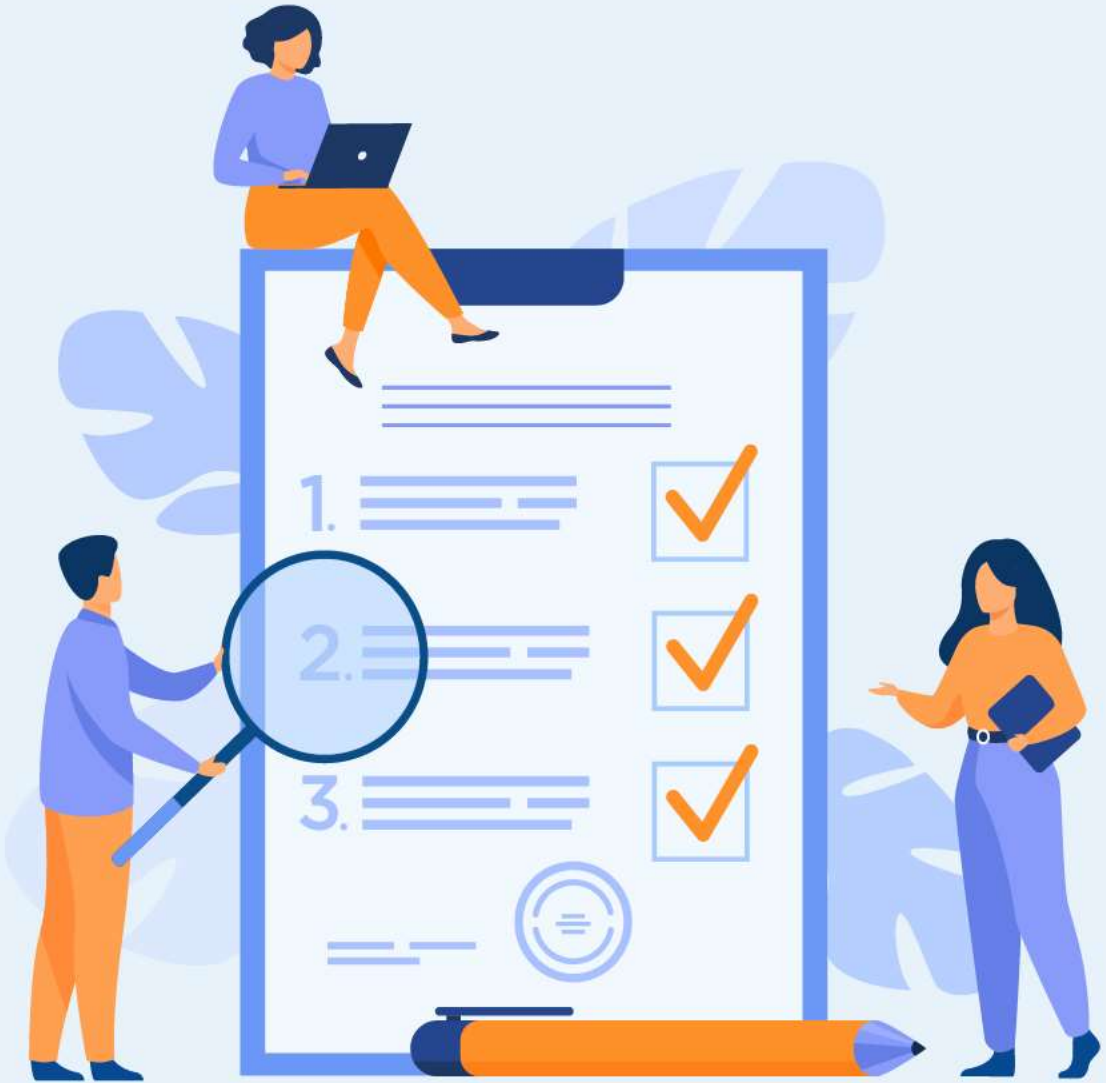
Packed Chain

PARTIE 1

Copyright - Tout droit réservé - OFPPT

CHAPITRE 3

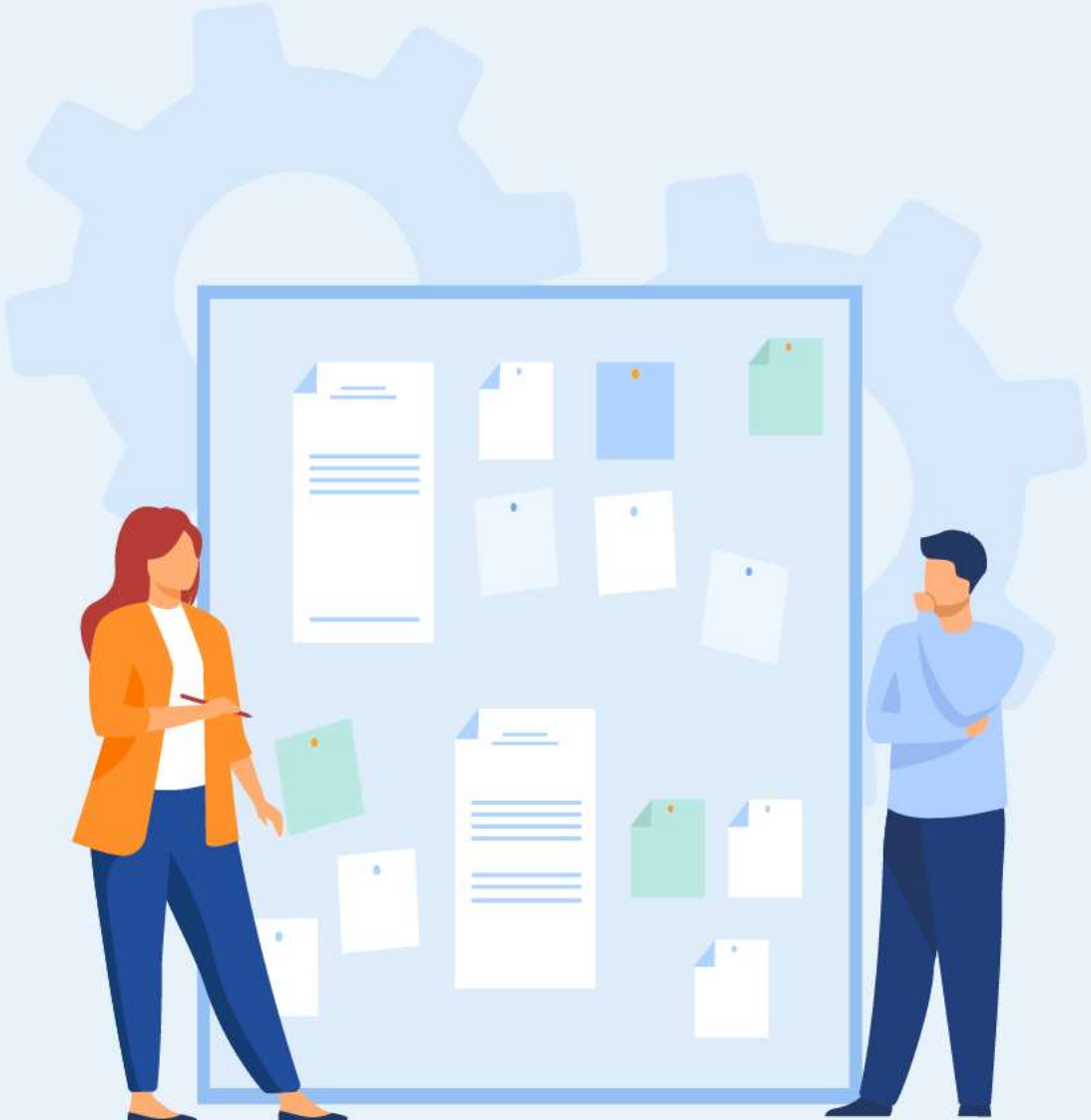
Gérer les évènements



Ce que vous allez apprendre dans ce chapitre :

- Définition d’un évènement
- Présentation des types des événements

1 heure



CHAPITRE 3

Gérer les évènements

1. **Définition d’un évènement**
2. Présentation des types des événements

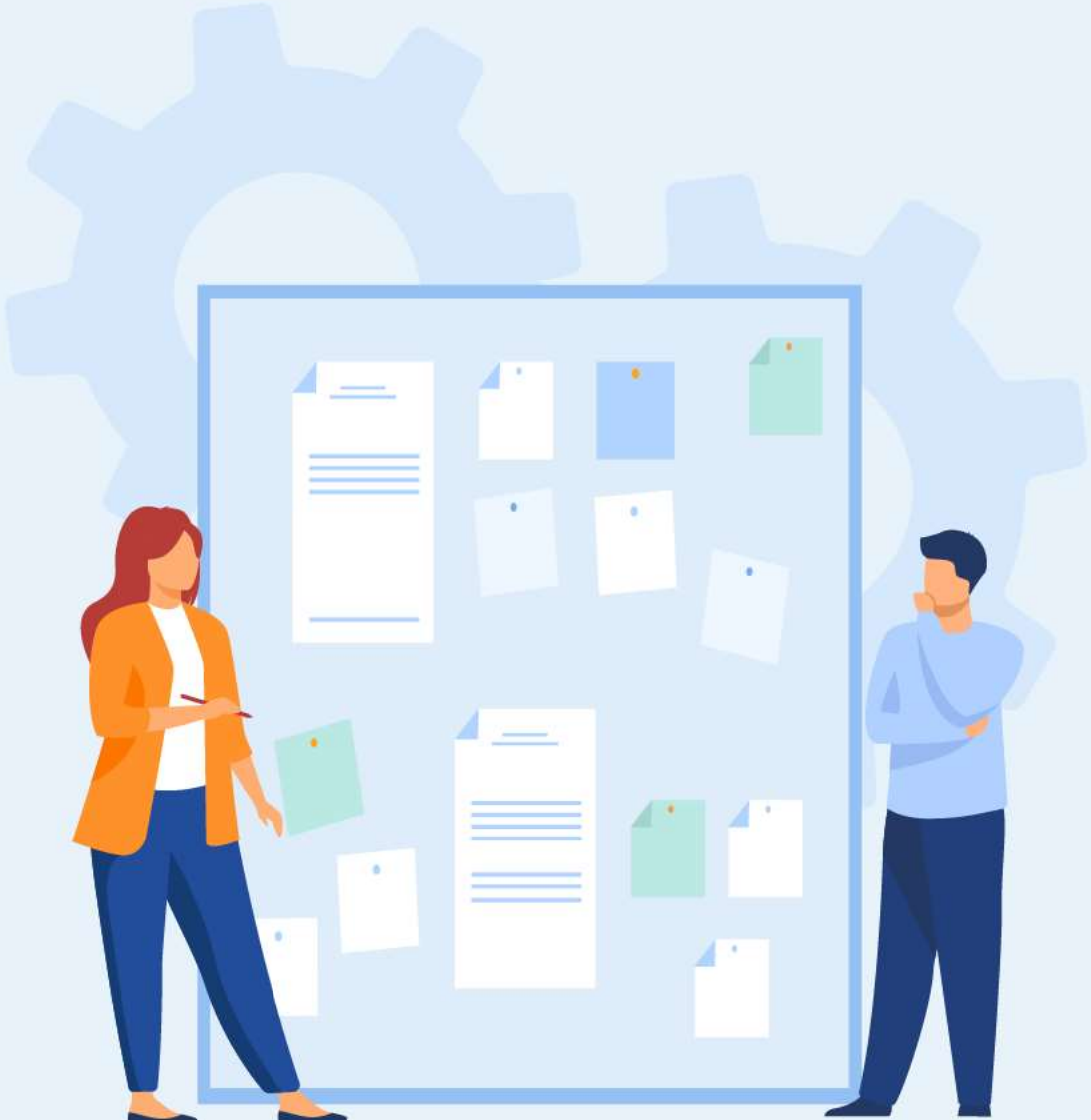
UI events

Les événements IU sont des actions qui doivent être traitées dans la couche IU, soit par l'IU, soit par le ViewModel. Le type d'événement le plus courant est l'événement utilisateur. L'utilisateur produit des événements utilisateur en interagissant avec l'application, par exemple en tapant sur l'écran ou en faisant des gestes. L'interface utilisateur consomme ensuite ces événements à l'aide de rappels tels que les écouteurs onClick().

Mots clés :

- **UI** : Code de vue ou de composition qui gère l'interface utilisateur ;
- **Événements d'interface** : Actions qui doivent être traitées dans la couche d'interface utilisateur ;
- **Événements utilisateurs** : Événements produits par l'utilisateur lors de son interaction avec l'application.

Copyright - Tout droit réservé - OFPPT



CHAPITRE 3

Gérer les évènements

1. Définition d’un évènement
2. **Présentation des types des événements**

Écouteurs d'événements (Event listeners)

Un écouteur d'événements est une interface de la classe View qui contient une seule méthode de rappel. Ces méthodes seront appelées par le framework Android lorsque la vue à laquelle l'écouteur a été enregistré est déclenchée par une interaction de l'utilisateur avec l'élément de l'interface utilisateur.

Les interfaces des écouteurs d'événements comprennent les méthodes de rappel suivantes :

onClick()

De **View**.**OnClickListener**. Il est appelé lorsque l'utilisateur touche l'élément (en mode tactile), ou se concentre sur l'élément à l'aide des touches de navigation ou de la boule de commande et appuie sur la touche "entrée" appropriée ou sur la boule de commande.

onLongClick()

De **View**.**OnLongClickListener**. Cette fonction est appelée lorsque l'utilisateur touche et maintient l'élément (en mode tactile), ou se focalise sur l'élément à l'aide des touches de navigation ou de la boule de commande et appuie et maintient la touche "entrée" appropriée ou appuie et maintient la boule de commande (pendant une seconde).

onFocusChange()

De **View**.**OnFocusChangeListener**. Cette fonction est appelée lorsque l'utilisateur navigue vers l'élément ou s'en éloigne, à l'aide des touches de navigation ou du trackball.

onKey()

Tiré de **View**.**OnKeyListener**. Elle est appelée lorsque l'utilisateur se concentre sur l'élément et appuie ou relâche une touche matérielle sur le périphérique.

onTouch()

Tiré de **View**.**OnTouchListener**. Cette fonction est appelée lorsque l'utilisateur effectue une action qualifiée d'événement tactile, y compris une pression, un relâchement ou tout autre geste de déplacement sur l'écran (dans les limites de l'élément).

onCreateContextMenu()

De **View**.**OnCreateContextMenuListener**. Elle est appelée lorsqu'un menu contextuel est en cours de création (à la suite d'un "clic long" prolongé).

Copyright - Tout droit réservé - OFPPT

Exemple (Java)

L'exemple ci-dessous montre comment enregistrer un écouteur de clic pour un bouton.

```
// Create an anonymous implementation of OnClickListener
private OnClickListener mCorkyListener = new OnClickListener() {
    public void onClick(View v) {
      // do something when the button is clicked
    }
};

protected void onCreate(Bundle savedValues) {
    ...
    // Capture our button from layout
    Button button = (Button)findViewById(R.id.corky);
    // Register the onClick listener with the implementation above
    button.setOnClickListener(mCorkyListener);
    ...
}
```

Il peut également être plus pratique d'implémenter le OnClickListener dans le cadre de votre activité. Cela évitera le chargement supplémentaire de la classe et l'allocation d'objets. Par exemple :

```
public class ExampleActivity extends Activity implements OnClickListener {
    protected void onCreate(Bundle savedValues) {
        ...
        Button button = (Button)findViewById(R.id.corky);
        button.setOnClickListener(this);
    }

    // Implement the OnClickListener callback
    public void onClick(View v) {
      // do something when the button is clicked
    }
    ...
}
```

PARTIE 1

Copyright - Tout droit réservé - OFPPT

Exemple (Suite)

Noter que la fonction de rappel onClick() de l'exemple précédent n'a pas de valeur de retour, mais que certaines autres méthodes d'écoute d'événements doivent renvoyer un booléen. La raison dépend de l'événement. Pour les quelques méthodes qui le font, voici pourquoi :

- **onLongClick()** - Cette méthode renvoie un booléen pour indiquer si l'événement a été consommé et s'il ne doit pas être poursuivi. En d'autres termes, renvoyer true pour indiquer que l'événement a été traité et qu'il doit s'arrêter ici ; renvoyer false si l'événement n'a pas été traité et/ou s'il doit être transmis à d'autres auditeurs on-click.
- **onKey()** - Cette fonction renvoie un booléen pour indiquer si l'événement a été consommé et s'il ne doit pas être poursuivi. En d'autres termes, retourner true pour indiquer que l'événement a été traité et qu'il doit s'arrêter ici ; retourner false si l'événement n'a pas été traité et/ou s'il doit être transmis à d'autres auditeurs on-key.
- **onTouch()** - Cette fonction renvoie un booléen pour indiquer si l'écouteur consomme cet événement. Ce qui est important, c'est que cet événement peut avoir plusieurs actions qui se suivent. Ainsi, si la fonction renvoie false lorsque l'événement d'action vers le bas est reçu, elle indique que l'événement n'a pas été consommé et qu'elle n'est pas non plus intéressée par les actions ultérieures de cet événement. Ainsi, aucune autre action de l'événement ne vous sera demandée, comme un geste du doigt ou l'éventuel événement d'action vers le haut.

Copyright - Tout droit réservé - OFPPT

Exemple (Kotlin)

L'exemple ci-dessous montre comment enregistrer un écouteur de clic pour un bouton.

```
protected void onCreate(savedValues: Bundle) {
   ...
   val button: Button = findViewById(R.id.corky)
   // Register the onClick listener with the implementation above
   button.setOnClickListener { view ->
      // do something when the button is clicked
   }
   ...
}
```

Il peut également s'avérer plus pratique d'implémenter le OnClickListener dans le cadre de l'activité. Cela évitera le chargement supplémentaire de la classe et l'allocation d'objets. Par exemple :

```
class ExampleActivity : Activity(), OnClickListener {

   protected fun onCreate(savedValues: Bundle) {
      val button: Button = findViewById(R.id.corky)
      button.setOnClickListener(this)
   }

   // Implement the OnClickListener callback
   fun onClick(v: View) {
      // do something when the button is clicked
   }
}
```


Copyright - Tout droit réservé - OFPPT

Event Handlers

Lorsque l'on crée un composant personnalisé à partir de View, il est possible de définir plusieurs méthodes de rappel utilisées comme gestionnaires d'événements par défaut. Certains des callbacks communs utilisés pour le traitement des événements :

- **onKeyDown(int, KeyEvent)** - Appelé lorsqu'un nouvel événement de touche se produit.
- **onKeyUp(int, KeyEvent)** - Appelé lorsqu'un événement lié à l'activation d'une touche se produit.
- **onTrackballEvent(MotionEvent)** - Appelé lorsqu'un événement de mouvement de la boule de commande se produit.
- **onTouchEvent(MotionEvent)** - Appelé lorsqu'un événement de mouvement de l'écran tactile se produit.
- **onFocusChanged(boolean, int, Rect)** - Appelé lorsque la vue prend ou perd le focus.

Il existe d'autres méthodes à connaître, qui ne font pas partie de la classe View, mais qui peuvent avoir un impact direct sur la façon dont les événements sont gérés. Ainsi, lors de la gestion d'événements plus complexes dans un layout, il convient de tenir compte de ces autres méthodes :

- **Activity.dispatchTouchEvent(MotionEvent)** : Cette méthode permet à votre activité d'intercepter tous les événements tactiles avant qu'ils ne soient envoyés à la fenêtre ;
- **ViewGroup.onInterceptTouchEvent(MotionEvent)** : Cette méthode permet à un ViewGroup d'observer les événements lorsqu'ils sont distribués aux vues enfant ;
- **ViewParent.requestDisallowInterceptTouchEvent(boolean)** : Appeler ceci sur une vue parent pour indiquer qu'elle ne doit pas intercepter les événements tactiles avec **onInterceptTouchEvent(MotionEvent)**.

Copyright - Tout droit réservé - OFPPT



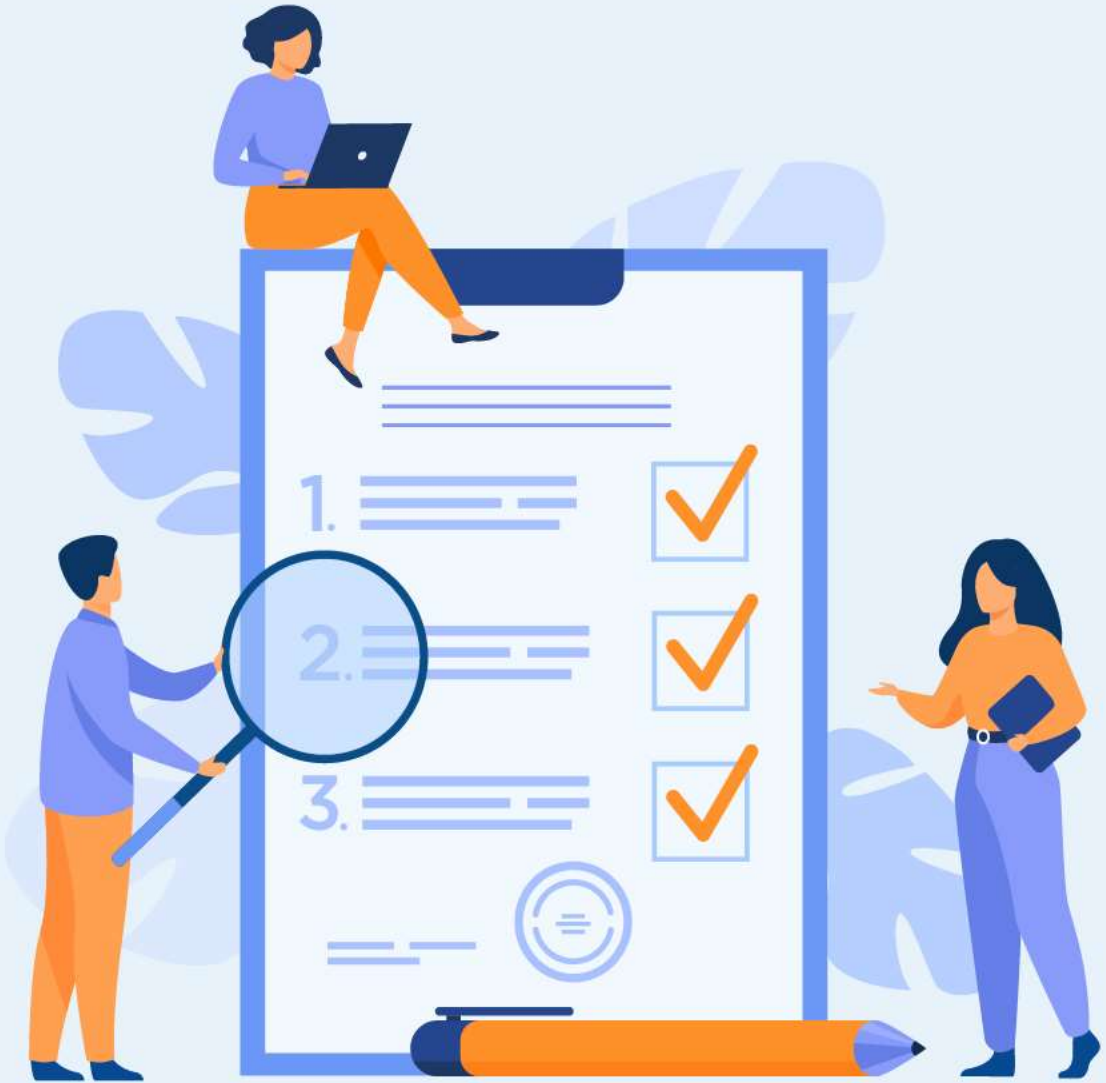
PARTIE 2

Créer des interfaces graphiques avancées

Dans ce module, vous allez :

- Intégration correcte des styles et des ressources ;
- Utilisation des Canvas ;
- Implémenter des vues en utilisant le Material design ;
- Gestion correcte des animations entre les activités.

9 heures



CHAPITRE 1

Coder des custom view

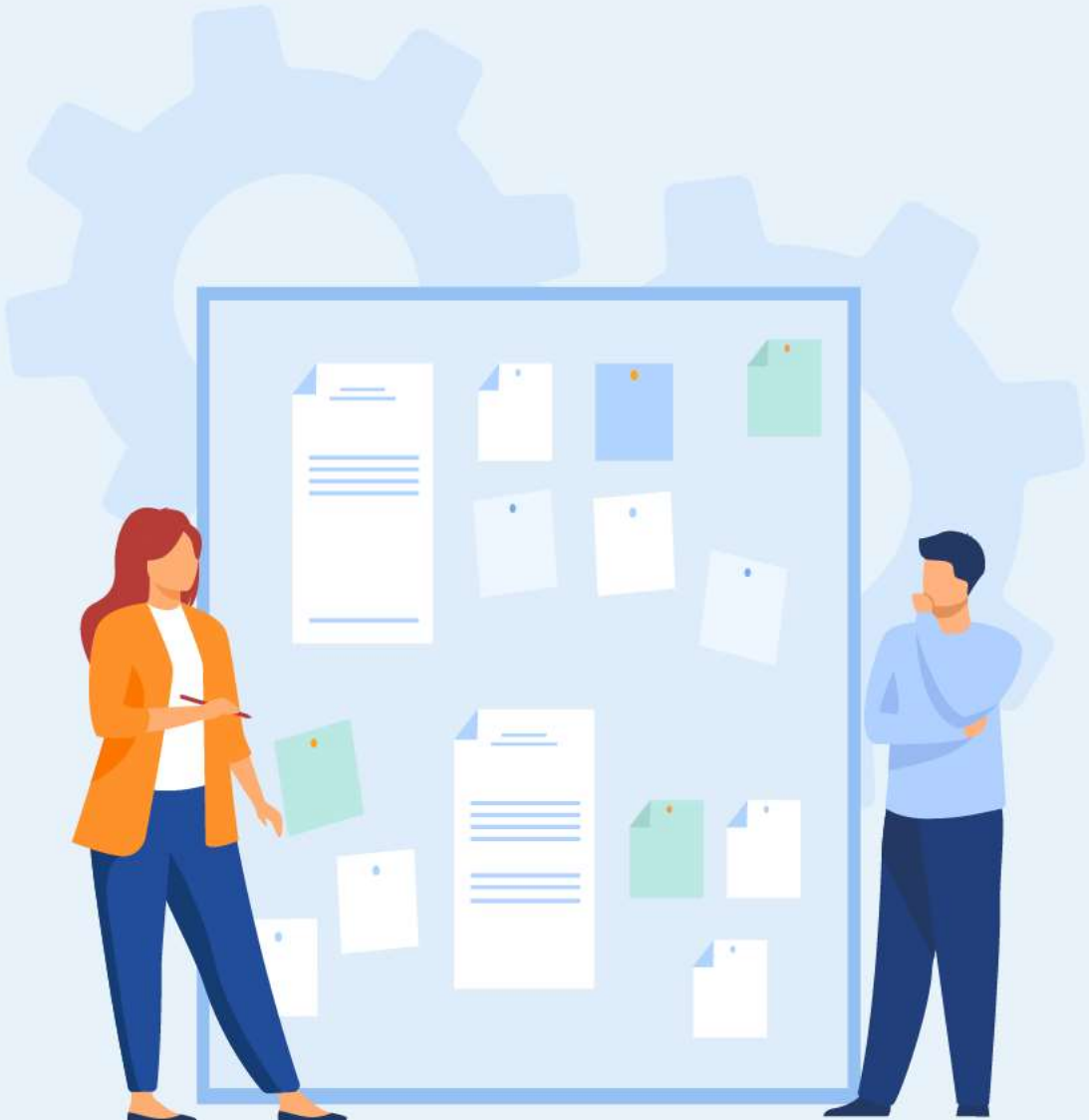
Ce que vous allez apprendre dans ce chapitre :

- Manipulation des fichiers des styles ;
- Utilisation des fichiers ressources ;
- Intégration de Material design ;
- Maitrise des Canvas.

3 heures

OFPPT
La Voie de l'Avenir

WEBFORCE
BE THE CHANGE



CHAPITRE 1

Coder des custom view

1. **Manipulation des fichiers des styles**
2. Utilisation des fichiers ressources
3. Intégration de Material design
4. Maitrise des Canvas

Drawables

Drawable : classe Android générique utilisée pour représenter tout type de graphique.

Tous les drawables sont stockés dans le dossier du projet res/drawable.

Classes Drawable

Bitmap File
Nine-Patch File
Layer List Drawable
Shape Drawable
State List Drawable
Level List Drawable
Transition Drawable
Vector Drawable
... et autres

Bitmaps

- Format PNG (.png), JPG (.jpg), ou GIF (.gif) ;
- BMP non compressé (.bmp) ;
- WebP (4.0 et plus) ;
- Crée un type de données BitmapDrawable ;
- Placé directement dans res/drawables.

Référencement des dessins

XML: `@[package:]drawable/filename`

```
<ImageView
    android:layout_height="wrap_content"
    android:layout_width="wrap_content"
    android:src="@drawable/myimage" />
```

Code Java : `R.drawable.filename`

```
Resources res = getResources();
Drawable drawable =
res.getDrawable(R.drawable.myimage);
```

PARTIE 2

Copyright - Tout droit réservé - OFPPT

Nine-Patch Files

Les fichiers de neuf patchs (.9.png) sont des PNG avec des régions extensibles.

S'étire seulement en plus grand, pas en plus petit, donc commencer avec une petite image.

Souvent utilisé pour les arrière-plans des éléments d'interface utilisateur

Exemple : l'arrière-plan d'un bouton change de taille en fonction de la longueur de l'étiquette.

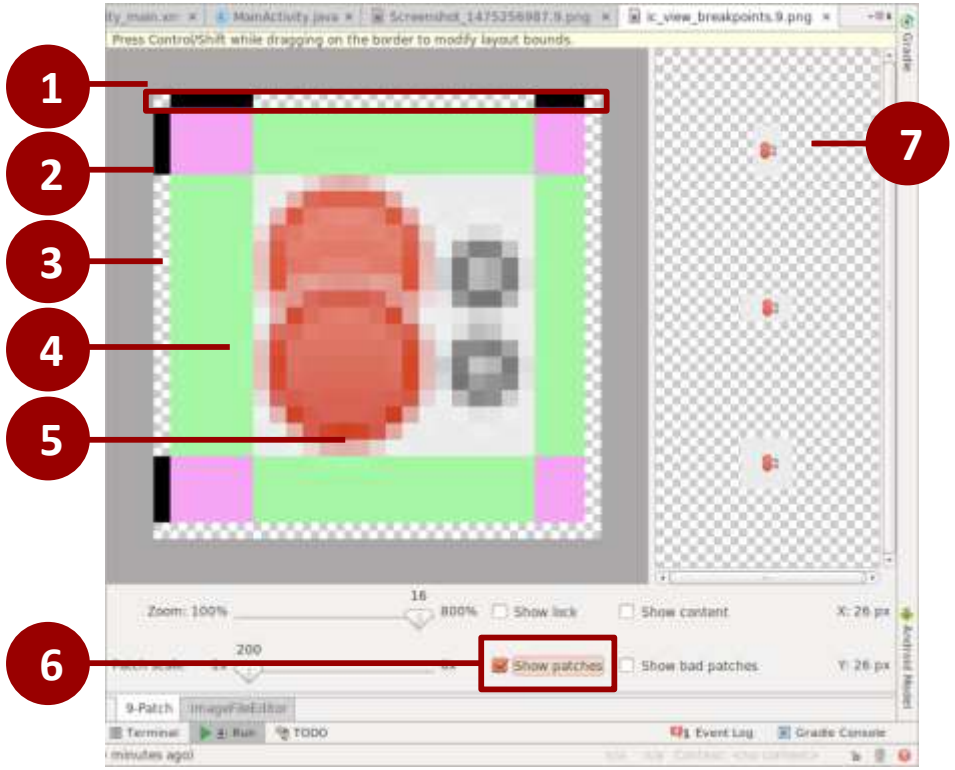
Création de fichiers de type Nine-Patch

1. Placer un petit fichier PNG dans res/drawable ;
2. Cliquer avec le bouton droit de la souris et choisir Créer un fichier 9-Patch ;
3. Double-clic sur le fichier 9-Patch pour ouvrir l'éditeur ;
4. Spécifier les régions extensibles (diapositive suivante).

Copyright - Tout droit réservé - OFPPT

Édition des fichiers de type Nine-Patch

1. Bordure pour marquer les régions extensibles en largeur ;
2. Régions extensibles marquées pour la hauteur Rose == les deux directions ;
3. Cliquer pour rendre les pixels noirs. Shift-clic (ctrl-clic sur Mac) pour annuler le marquage ;
4. Zone extensible ;
5. Non extensible ;
6. Cocher Montrer les patchs ;
7. Aperçu de l'image étirée.



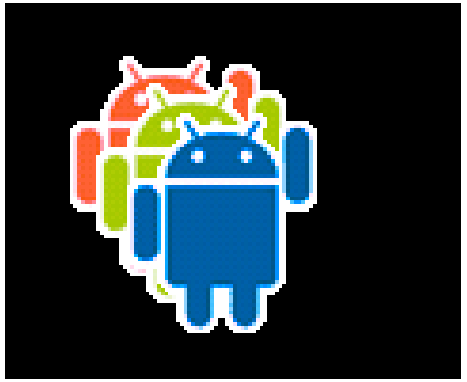
Copyright - Tout droit réservé - OFPPT

Liste des couches

- Les images peuvent être créées en couches, comme avec des outils de dessin tels que Gimp ;
- Dans Android, chaque couche est représentée par un drawable ;
- Les calques sont organisés et gérés en XML ;
- La liste et les éléments peuvent avoir des propriétés ;
- Les couches sont dessinées les unes sur les autres dans l'ordre défini dans le fichier XML ;
- LayerDrawable

Création de la liste des couches

```
<layer-list>
  <item>
   <bitmap android:src="@drawable/android_red"
    android:gravity="center" />
  </item>
  <item android:top="10dp" android:left="10dp">
   <bitmap android:src="@drawable/android_green"
    android:gravity="center" />
  </item>
  <item android:top="20dp" android:left="20dp">
   <bitmap android:src="@drawable/android_blue"
    android:gravity="center" />
  </item>
</layer-list>
```



Copyright - Tout droit réservé - OFPPT

Shape Drawables & GradientDrawable

- Définit une forme et ses propriétés en XML :
  - Rectangle, ovale, anneau, ligne.
- Stylisée avec des attributs tels que <corners>, <gradient>, <padding>, <size>, <solid>et <stroke> ;

  Voir Shape Drawables pour plus d'attributs ;
- Peut être gonflé pour un GradientDrawable.

Création d'un GradientDrawable

```
<shape … android:shape="rectangle">
 <gradient
   android:startColor="@color/white"
   android:endColor="@color/blue"
   android:angle="45"/>
 <corners android:radius="8dp" />
</shape>
```




```
Resources res = getResources();
Drawable shape = res.getDrawable(R.drawable.gradient_box);

TextView tv = (TextView)findViewByID(R.id.textview);
tv.setBackground(shape);
```

PARTIE 2

Copyright - Tout droit réservé - OFPPT

Transition Drawables

- Drawable qui peut faire un fondu enchaîné entre deux autres drawables ;
- Chaque graphique représenté par <item> à l'intérieur de <selector> ;
- Représenté par TransitionDrawable dans le code Java ;
- Transition vers l'avant en appelant startTransition() ;
- Transition vers l'arrière avec reverseTransition() ;

Création de dessins de transition

```
<transition ...>
  <selector> <item android:drawable="@drawable/on" />
             <item android:drawable="@drawable/off" />
  </selector>
</transition>
```

```
<ImageButton
    android:id="@+id/button"
    android:src="@drawable/transition" />
```

```
ImageButton button = findViewById(R.id.button);
TransitionDrawable drawable =
        (TransitionDrawable) button.getDrawable();
drawable.startTransition(500);
```


Copyright - Tout droit réservé - OFPPT

Vector drawables

- Créer une transition qui s'adapte à toutes les tailles d'écran ;
- API Android niveau 21 et plus ;
- Utiliser Vector Asset Studio pour créer (diapositives ci-dessous) ;
- VectorDrawable.

Créer un Vector drawable

```
<vector ...
    android:height="256dp" android:width="256dp"
    android:viewportWidth="32" android:viewportHeight="32">
  <path android:fillColor="@color/red"
      android:pathData="M20.5,9.5
                        c-1.955,0,-3.83,1.268,-4.5,3
                        c-0.67,-1.732,-2.547,-3,-4.5,-3 ...
/>
</vector>
```

pathData pour la forme du cœur



Copyright - Tout droit réservé - OFPPT

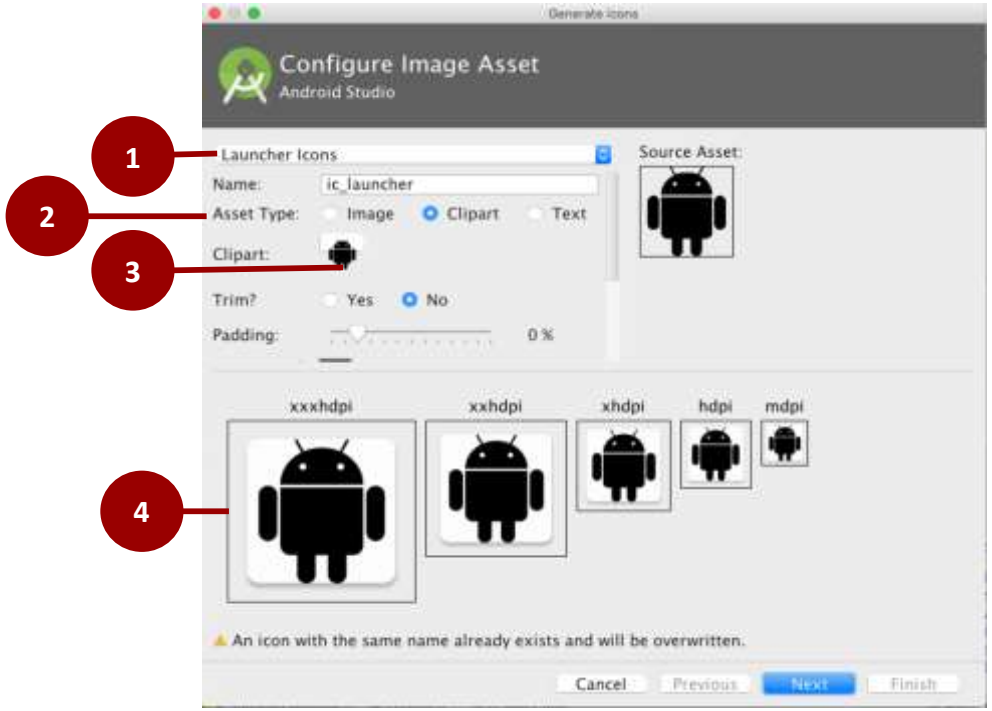
Qu'est-ce que Image Asset Studio ?

- Créer des icônes à partir d'icônes matérielles, d'images et de texte ;
- Icônes de lanceur, de barre d'action, d'onglet, de notification ;
- Générer un ensemble d'icônes pour une densité d'écran généralisée ;
- Stockées dans le dossier /res ;

- Pour lancer Image Asset Studio :
    1. Cliquer avec le bouton droit de la souris sur le dossier res de votre projet ;
    2. Choisir New > Image Asset.

Utiliser Image Asset Studio

1. Choisir le type d'icône et changer le nom ;
2. Choisir une image, un clip ou un texte ;
3. Cliquer sur l'icône pour choisir l'image clipart ;
4. Inspecter les actifs pour plusieurs tailles d'écran.



PARTIE 2

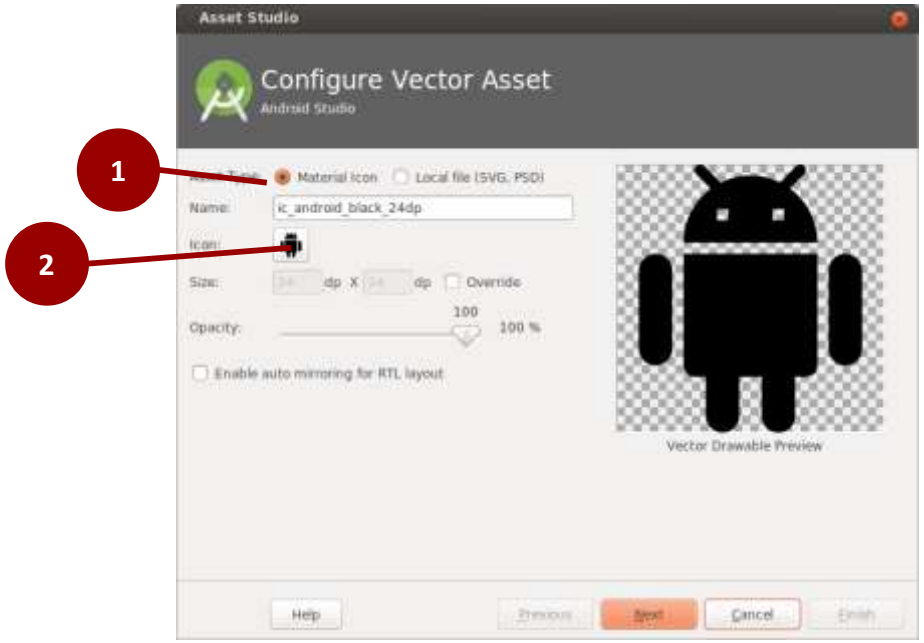
Copyright - Tout droit réservé - OFPPT

Qu'est ce que Vector Asset Studio ?

- Créer des icônes à partir d'icônes matérielles ou fournir vos propres dessins vectoriels pour l'API 21 et les versions ultérieures ;
- Lanceur, barre d'action, onglet, icônes de notification ;
- Générer un dessin vectoriel à l'échelle ;
- Stocké dans le dossier res ;
- Pour lancer Image Asset Studio :
    1. Cliquer avec le bouton droit de la souris sur le dossier res de votre projet;
    2. Choisir New > Vector Asset.

Utiliser Vector Asset Studio

1. Choisir dans la bibliothèque d'icônes matérielles ou fournir votre propre dessin vectoriel SVG ou PSD ;
2. Ouvrir la bibliothèque d'icônes matérielles.



PARTIE 2

Copyright - Tout droit réservé - OFPPT

Images, mémoire et performances

- Utiliser la plus petite résolution d'image nécessaire ;
- Redimensionner, recadrer, compresser ;
- Dessins vectoriels pour les images simples ;
- Utiliser des bibliothèques : Glide ou Picasso ;
- Choisir des formats d'image adaptés au type et à la taille de l'image ;
- Utiliser des formats d'image avec perte et ajuster la qualité lorsque cela est possible ;
- Pour en savoir plus sur la compression des données pour les développeurs, consulter le site Understanding Compression.

Copyright - Tout droit réservé - OFPPT

Qu'est-ce qu'un style ?

- Collection d'attributs qui définissent l'aspect visuel d'une vue ;
- Réduire la duplication ;
- Rendre le code plus compact ;
- Gérer l'aspect visuel de nombreux composants avec un seul style.

Les styles réduisent le désordre

```
<TextView
  android:layout_width="match_parent"
  android:layout_height="wrap_content"
  android:textColor="#00FF00"
  android:typeface="monospace"
  android:text="@string/hello" />
```

```
<TextView
    style="@style/CodeFont"
    android:text="@string/hello" />
```


Copyright - Tout droit réservé - OFPPT

Définir les styles dans styles.xml

styles.xml est dans res/values

```
<resources>
    <style name="CodeFont">
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```

Héritage : Parent

Définir un style parent...

```
<resources>
    <style name="CodeFont">
        <item name="android:layout_width">match_parent</item>
        <item name="android:layout_height">wrap_content</item>
        <item name="android:textColor">#00FF00</item>
        <item name="android:typeface">monospace</item>
    </style>
</resources>
```


Copyright - Tout droit réservé - OFPPT

Héritage : Définir l'enfant

Définir un enfant avec Codefont comme parent

```
<resources>
    <style name="RedCode" parent="@style/Codefont>
        <item name="android:textColor">#FF0000</item>
    </style>
</resources>
```


Copyright - Tout droit réservé - OFPPT

Themes

- Un thème est un style appliqué à une activité entière ou même à l'application entière ;
- Les thèmes sont appliqués dans AndroidManifest.xml

  <Application android:theme="@style/AppTheme">

Personnaliser le thème de l'application du projet

```
<!-- Thème de base de l'application. -->
<style name="AppTheme"
       parent="Theme.AppCompat.Light.DarkActionBar">
<!-- Try: Theme.AppCompat.Light.NoActionBar -->
   <!-- Customize your theme here. -->
   <item name="colorPrimary">@color/colorPrimary</item>
   <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
   <item name="colorAccent">@color/colorAccent</item>
</style>
```


Copyright - Tout droit réservé - OFPPT

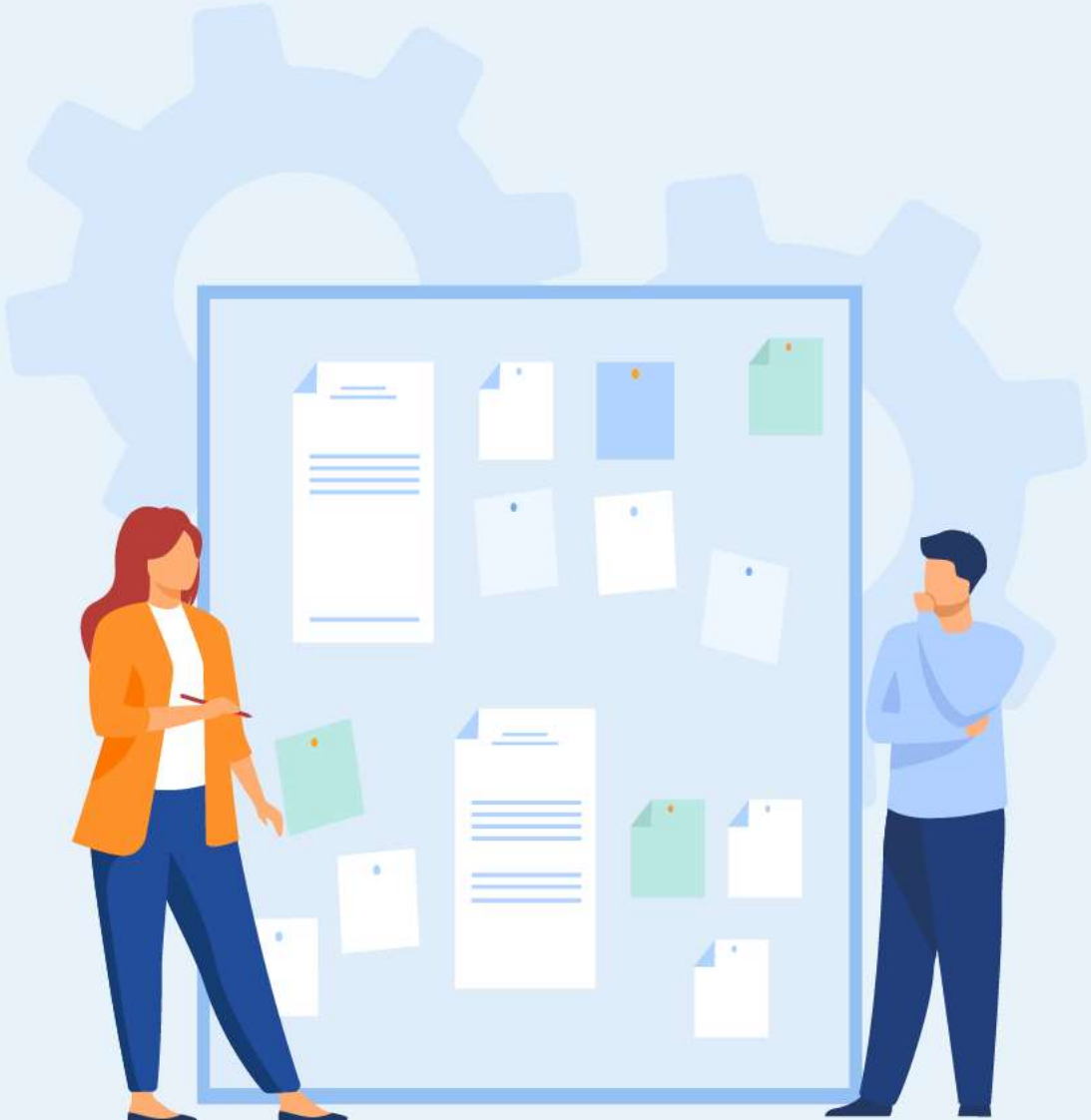
Ressources sur les styles et les thèmes

La plateforme Android dispose d'une collection de styles et de thèmes intégrés :

- Android Styles
- Android Themes
- Styles and Themes Guide
- DayNight Theme Guide



Copyright - Tout droit réservé - OFPPT



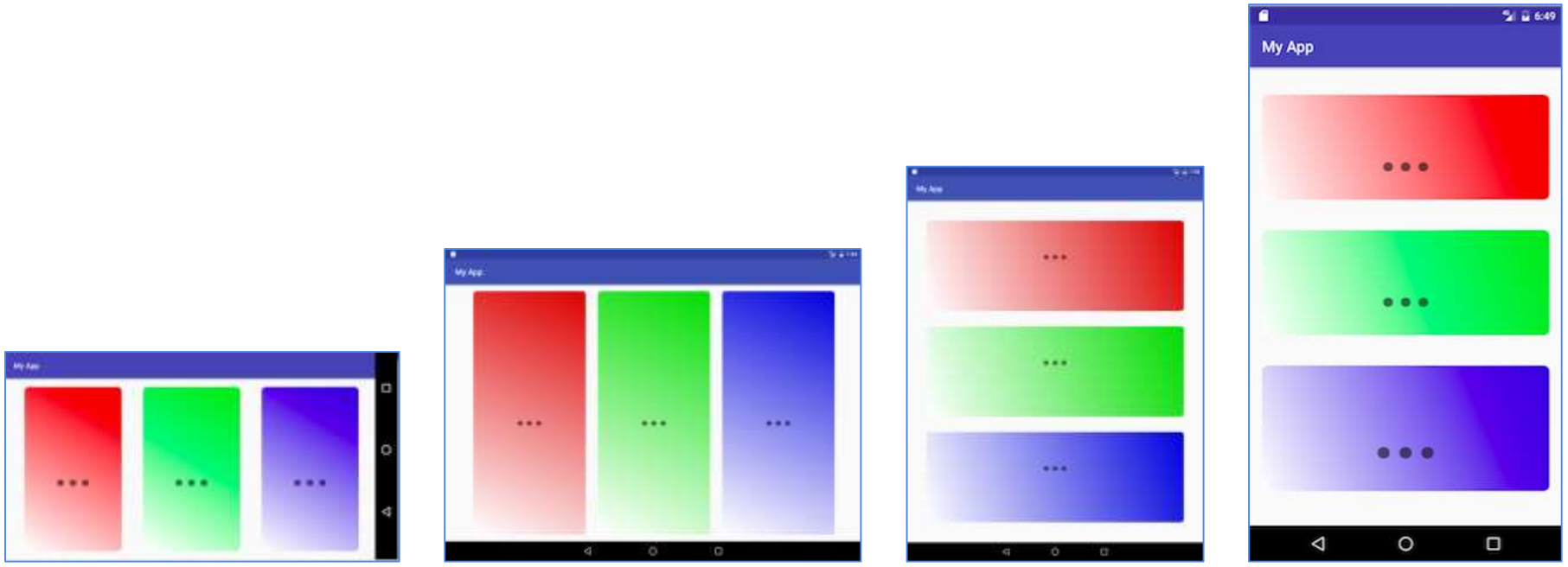
CHAPITRE 1

Coder des custom view

1. Manipulation des fichiers des styles
2. **Utilisation des fichiers ressources**
3. Intégration de Material design
4. Maitrise des Canvas

Qu'est-ce qu'une mise en page adaptative ?

Des mises en page qui s'adaptent à des écrans de tailles, d'orientations et de dispositifs différents.



Copyright - Tout droit réservé - OFPPT

Disposition adaptative

- La mise en page s'adapte à la configuration :
  - Taille de l'écran ;
  - Orientation du dispositif ;
  - Locale ;
  - Version d'Android installée.
- Fournit des ressources alternatives :
  - Chaînes de caractères localisées.
- Utilise des mises en page flexibles :
  - GridLayout.

Copyright - Tout droit réservé - OFPPT

Dossiers de ressources d'une petite application

```
MyProject/
    src/
    res/
        drawable/
            graphic.png
        layout/
            activity_main.xml
            list_iteminfo.xml
        mipmap/
            ic_launcher_icon.png
        values/
            strings.xml
```

Placer les ressources dans le dossier res de du projet

Répertoires de ressources communes

- `drawable/`, `layout/`, `menu/`
- `values/`— Fichiers XML de valeurs simples, telles que chaîne ou couleur
- `xml/`— des fichiers XML arbitraires
- `raw/`— des fichiers arbitraires dans leur forme brute
- `mipmap/`—drawables pour différentes densités d'icônes de lanceur
- Liste Compléte

Copyright - Tout droit réservé - OFPPT

Que sont les ressources alternatives ?

Des configurations de dispositifs différentes peuvent nécessiter des ressources différentes :

- Chaînes de caractères localisées ;
- Résolutions d'image ;
- Dimensions de la mise en page.

Android charge automatiquement les ressources appropriées

Créer des dossiers de ressources alternatives



Utiliser des dossiers alternatifs pour les ressources en fonction de la configuration des appareils.

Copyright - Tout droit réservé - OFPPT

Noms des dossiers de ressources alternatives

Les noms des dossiers de ressources ont le format ressources name-config qualifier.

drawable-hdpi	mise en page pour les écrans à haute densité
layout-land	mise en page pour l'orientation paysage
layout-v7	mise en page pour la version de la plate-forme
values-fr	tous les fichiers de valeurs pour la langue française

Liste des répertoires et des qualificatifs et détail de leur utilisation.

Orientation de l'écran

- Utiliser res/layout et fournir des alternatives pour le paysage si nécessaire :
  - res/layout-port pour les mises en page spécifiques au portrait ;
  - res/layout-land pour les mises en page spécifiques au paysage.
- Éviter les dimensions codées en dur pour réduire le besoin de mises en page spécialisées.

Copyright - Tout droit réservé - OFPPT

Mise en page simple et adaptable

GridLayout

- Dans values/integer.xml:

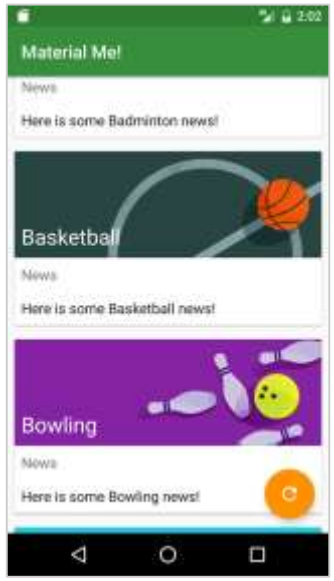
```
<integer name="grid_column_count">1</integer>
```

- Dans values/integer.xml-land:

```
<integer name="grid_column_count">2</integer>
```



Landscape



Portrait

Largeur minimale (Smallest-width)

- Smallest-width (sw) dans le nom du dossier indique la largeur minimale du périphérique :
  - res/values-swndp, où n est la plus petite largeur ;
  - Exemple : res/values-sw600dp/dimens.xml ;
  - Ne change pas avec l'orientation.
- Android utilise la ressource la plus proche (sans la dépasser) de la plus petite largeur de l'appareil.

Copyright - Tout droit réservé - OFPPT

Version plateforme

- Niveau d'API supporté par le dispositif :
    - res/drawables-v14 contient les éléments à dessiner pour les périphériques qui prennent en charge le niveau 14 de l'API et les niveaux supérieurs.
- Certaines ressources ne sont disponibles que pour les versions plus récentes :
    - Le format d'image WebP nécessite le niveau 14 de l'API (Android 4.0).
- Niveau d'API Android.

Localisation

- Fournir des chaînes (et d'autres ressources) pour des locales spécifiques :
    - res/values-es/strings.xml.
- Augmente l'audience potentielle de votre application ;
- La langue est basée sur les paramètres de l'appareil ;
- Localisation.

Copyright - Tout droit réservé - OFPPT

Ressources couleurs

Un moyen de nommer et de normaliser les couleurs dans l'ensemble de l'application.

Dans `res/values/colors.xml`:

```
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    ...
</resources>
```

Spécifié en tant que couleurs hexadécimales sous la forme de `#AARRGGBB`.

Ressources de dimension

Un moyen de nommer et de normaliser les valeurs des dimensions dans les mises en page.

- Déclarer les valeurs de vos dimensions dans `res/values/dimens.xml`:

```
<resources>
    <dimen name="top_margin">16dp</dimen>
</resources>
```

- Il faut les appeler @dimen/<name> dans les mises en page ou R.dimen.<name> dans le code :

```
<TextView …
    android:layout_marginTop="@dimen/top_margin" />
```

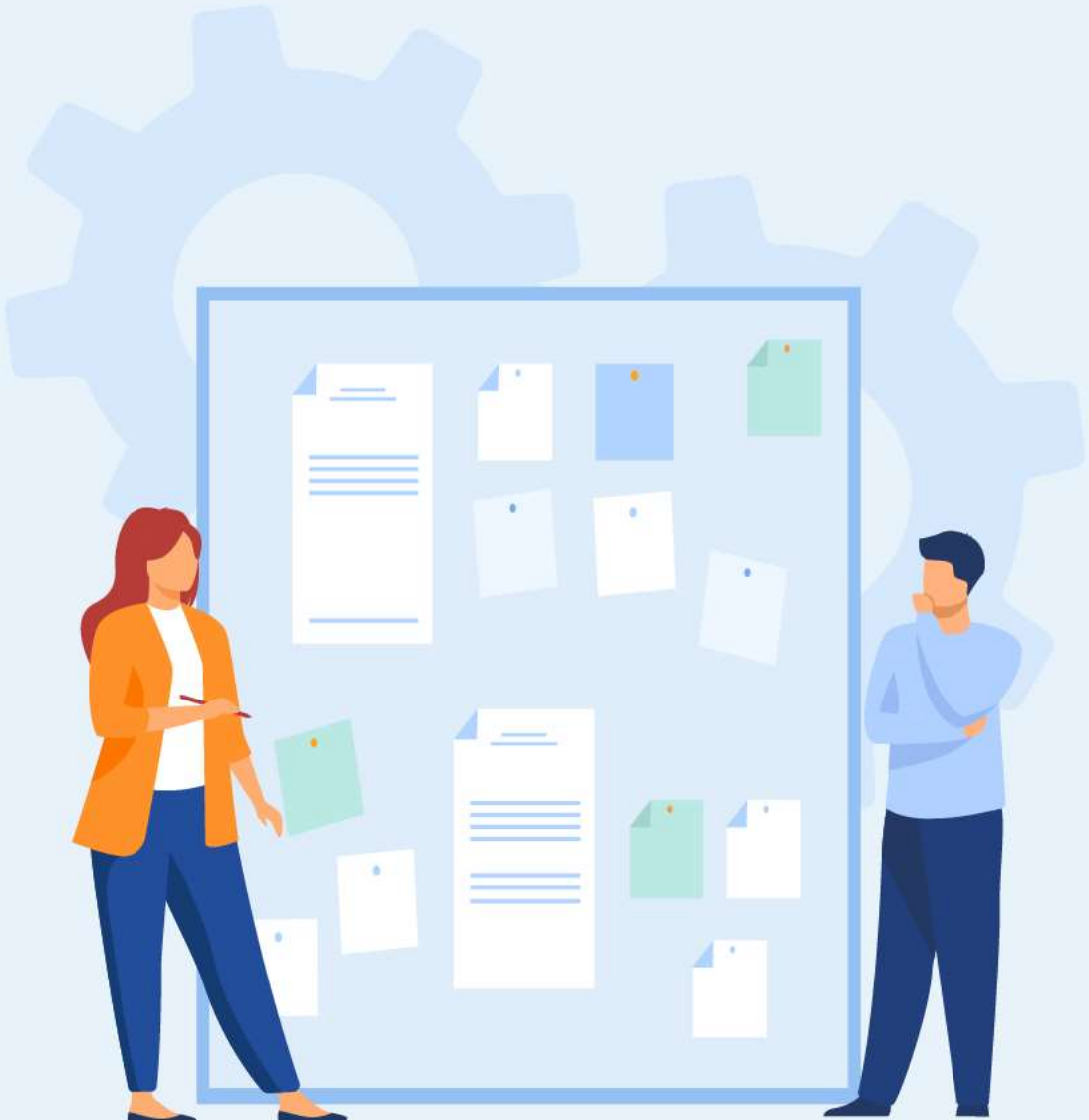
PARTIE 2

Copyright - Tout droit réservé - OFPPT

Ressources par défaut

- Toujours fournir des ressources par défaut :
    - nom de répertoire sans qualificatif ;
    - res/layout, res/values, res/drawables....
- Android utilise les ressources par défaut lorsqu'aucune ressource spécifique ne correspond à la configuration ;
- Localisation avec les ressources.

Copyright - Tout droit réservé - OFPPT



CHAPITRE 1

Coder des custom view

1. Manipulation des fichiers des styles
2. Utilisation des fichiers ressources
3. **Intégration de Material design**
4. Maitrise des Canvas

Qu'est-ce que le Material Design ?

- Directives de conception ;
- Langage visuel ;
- Combiner les principes classiques d'un bon design avec l'innovation et les possibilités de la technologie et de la science ;
- Spécifications du design des matériaux.



Métaphore du matériel

- Environnement tridimensionnel contenant de la lumière, de la matière et des ombres ;
- Les surfaces et les bords fournissent des repères visuels ancrés dans la réalité ;
- Les principes fondamentaux de la lumière, de la surface et du mouvement expliquent comment les objets se déplacent, interagissent et existent dans l'espace et les uns par rapport aux autres.



Copyright - Tout droit réservé - OFPPT

Le design matériel dans l'application

Les éléments de l'application Android doivent se comporter de la même manière que les matériaux du monde réel.

- projeter des ombres ;
- Occuper l'espace ;
- Interagir les uns avec les autres.



Bold, graphic, intentional

- Choisir délibérément les couleurs ;
- Remplir l'écran d'un bout à l'autre ;
- Utiliser une typographie à grande échelle ;
- Utiliser intentionnellement les espaces blancs ;
- Mettre l'accent sur l'action de l'utilisateur ;
- Rendre les fonctionnalités évidentes.



Copyright - Tout droit réservé - OFPPT

Imagerie

Les images aident à communiquer et à différencier l'application.

devraient être :

- Pertinents ;
- Informatifs ;
- Séduisants.

Bonnes pratiques :

- Utilisation avec du texte ;
- Images originales ;
- Fournir un point d'intérêt ;
- Construire un récit.

Typographie



PARTIE 2

Copyright - Tout droit réservé - OFPPT

Roboto typeface

Roboto est la police de caractères standard sur Android ;

Roboto a 6 niveaux de poids :

- Mince
- Léger
- Régulier
- Moyen
- Gras
- Noir

Roboto Thin
Roboto Light
Roboto Regular
Roboto Medium
Roboto Bold
Roboto Black
Roboto Thin Italic
Roboto Light Italic
Roboto Italic
Roboto Medium Italic
Roboto Bold Italic
Roboto Black Italic

Styles de police et échelle

- Un trop grand nombre de tailles prête à confusion et ne donne pas une bonne apparence ;
- Un ensemble limité de tailles qui fonctionnent bien ensemble.

Display 4	Light 112sp
Display 3	Regular 56sp
Display 2	Regular 45sp
Display 1	Regular 34sp
Headline	Regular 24sp
Title	Medium 20sp
Subheading	Regular 16sp (Device), Regular 15sp (Desktop)
Body 2	Medium 14sp (Device), Medium 13sp (Desktop)
Body 1	Regular 14sp (Device), Regular 13sp (Desktop)
Caption	Regular 12sp
Button	MEDIUM (ALL CAPS) 14sp

Copyright - Tout droit réservé - OFPPT

Définition de l'apparence du texte

```
android:textAppearance=
    "@style/TextAppearance.AppCompat.Display3"
```

Les polices en tant que ressources

- Bundle fonts en tant que ressources dans le paquet d'application (APK) ;
- Créer un dossier de polices dans la ressource, ajouter le fichier XML de la police à la police ;
- Pour accéder à la ressource police :
    - @font/myfont ;
    - R.font.myfont.
- Android 8.0 (niveau 26 de l'API) - Android 4.1 (niveau 16 de l'API) et versions ultérieures, utiliser la bibliothèque de support 26 ;
- Voir Polices en XML.

Copyright - Tout droit réservé - OFPPT

Polices téléchargeables

- Télécharger les polices depuis l'application du fournisseur
  - Réduit la taille de l'APK ;
  - Augmente le taux de réussite de l'installation de l'application ;
  - Améliore la santé générale du système, économise les données cellulaires, la mémoire du téléphone et l'espace disque.
- Android 8.0 (niveau 26 de l'API) - Niveau 14 de l'API et plus, utiliser la bibliothèque de support 26 ;
- Voir les polices téléchargeables

Couleur

- Nuances vives ;
- Environnements atténués ;
- Ombres profondes ;
- Hautes lumières.



Copyright - Tout droit réservé - OFPPT

Palette de couleurs

- Material Design recommande d'utiliser :
  - une couleur primaire ;
  - ainsi que des nuances ;
  - et une couleur d'accentuation.
- Créer une expérience utilisateur audacieuse pour l'application :
  - Palette de couleurs du Material Design.



Palette de couleurs pour le projet

- Android Studio crée une palette de couleurs pour l'utilisateur ;
- Définition de l'AppTheme dans styles.xml :
  - colorPrimary-AppBar, marquage ;
  - colorPrimaryDark-barre d'état, contraste ;
  - colorAccent-attire l'attention de l'utilisateur, switchs, FAB.
- Couleurs définies dans colors.xml ;
- Outil de sélection des couleurs.



PARTIE 2

Copyright - Tout droit réservé - OFPPT

PARTIE 2

Couleur et contraste du texte

- Contraste pour la séparation visuelle ;
- Contraste pour la lisibilité ;
- Contraste pour l'accessibilité ;
- Tout le monde ne voit pas les couleurs de la même façon ;
- Le thème gère le texte par défaut :
  - Theme.AppCompat.Light : texte proche du noir ;
  - Theme.AppCompat.Light.DarkActionBar : texte proche du blanc.

Good choice

Good choice

Bad choice

Bad choice

Bad choice

Good choice

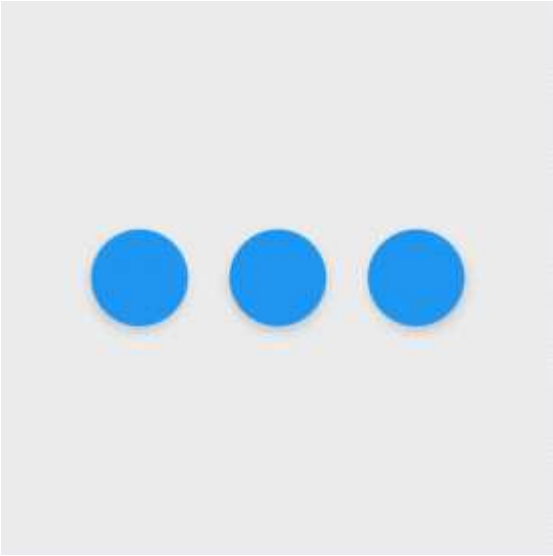
Mouvement

Le mouvement en Material Design décrit :

- Les relations spatiales ;
- Fonctionnalité ;
- Intention.

Le mouvement est :

- Responsive ;
- Naturel ;
- Conscient ;
- Intentionnel.



Copyright - Tout droit réservé - OFPPT

PARTIE 2

Le mouvement dans l'application

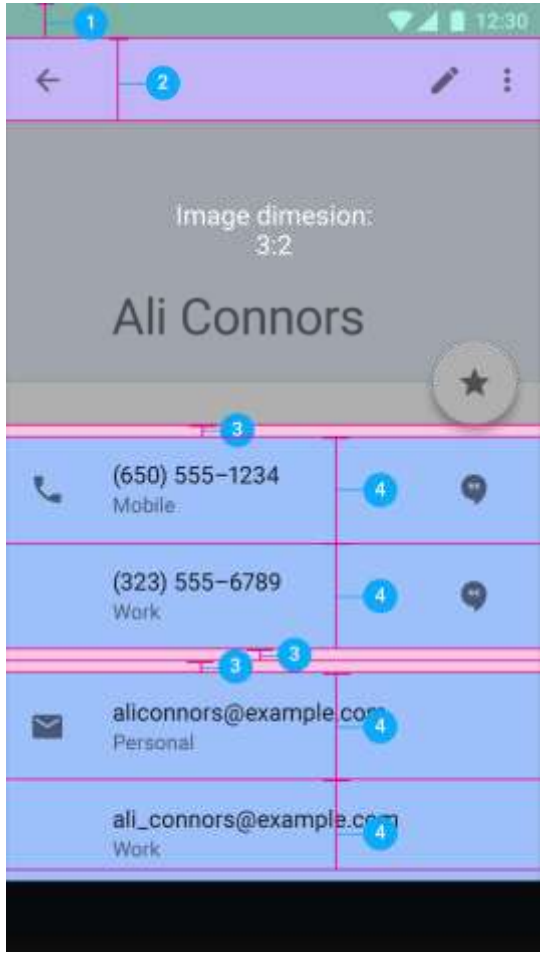
- Maintenir la continuité ;
- Mettre en évidence des éléments ou des actions ;
- Offrir une transition naturelle entre les actions ou les états ;
- Attirer l'attention ;
- Organiser les transitions ;
- Offrir une rétroaction réactive.

Touch feedback



Responsive interaction

Layout

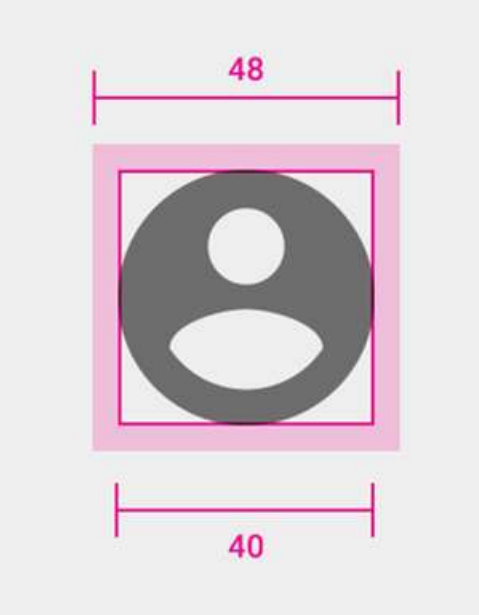


Copyright - Tout droit réservé - OFPPT

PARTIE 2

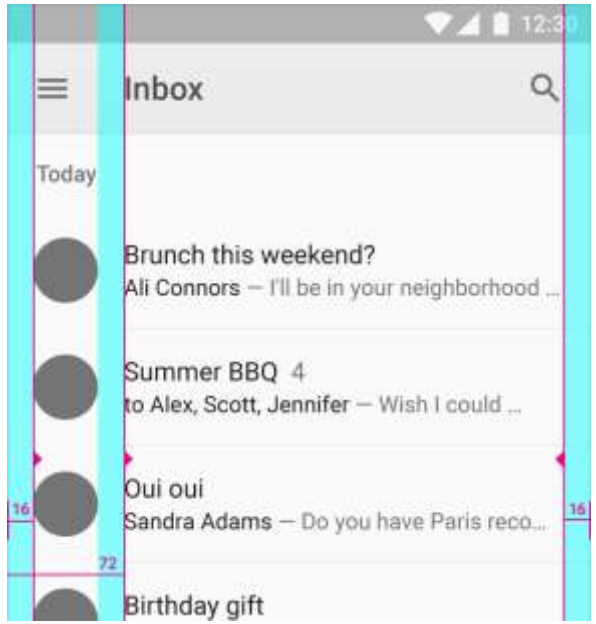
Layout pour le Material Design

- Pixels indépendants de la densité pour les vues-dp ;
- Pixels évolutifs pour le texte-sp ;
- Les éléments s'alignent sur une grille avec un espacement cohérent ;
- Planifier la disposition des éléments ;
- Utiliser des modèles de mise en page courants.

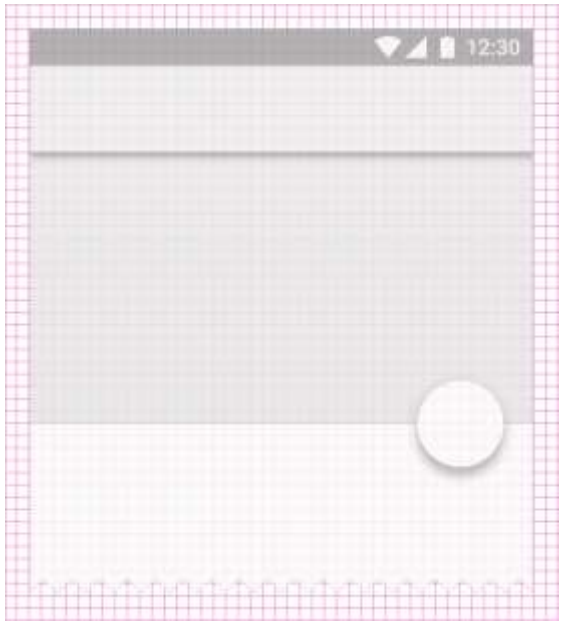


c) Sizing

Planification de layout



a) Spacing



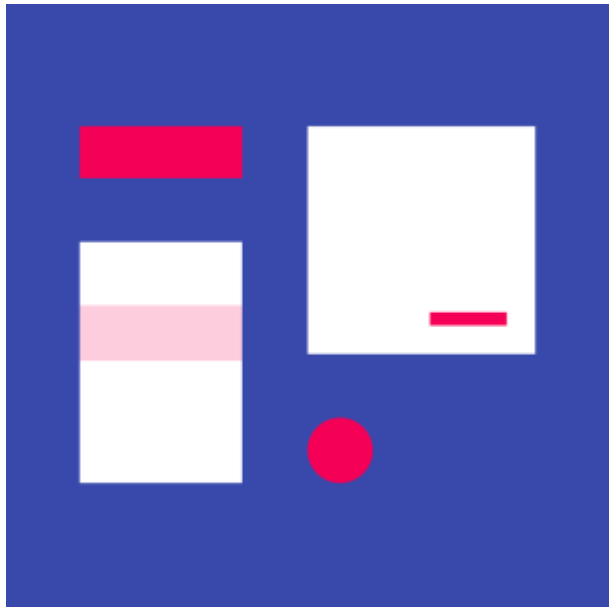
b) Grid alignment

Copyright - Tout droit réservé - OFPPT

Composants

Material Design dispose de lignes directrices sur l'utilisation et la mise en œuvre des composants Android :

- Bottom Navigation
- Buttons
- Cards
- Chips
- Data Tables
- Dialogs
- Dividers
- Sliders
- Snackbar
- Toasts
- Steppers
- Subheaders
- Text Fields
- Toolbars



- Expansion Panels
- Grid Lists
- Lists
- Menus
- Pickers
- Progress Bars
- Selection Controls





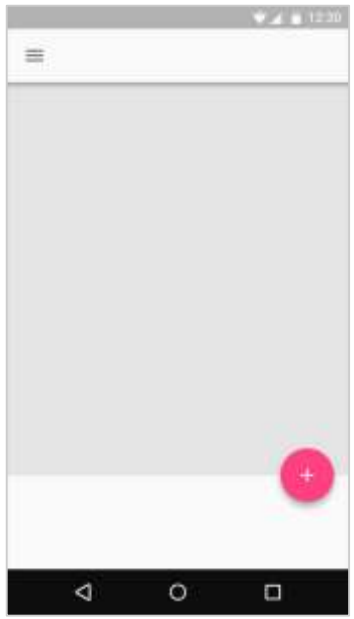




Copyright - Tout droit réservé - OFPPT

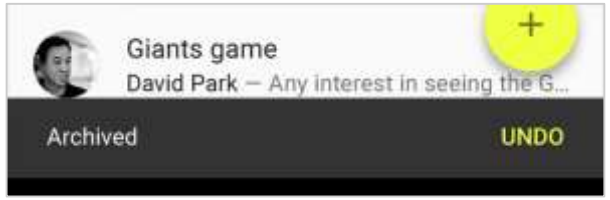
La cohérence favorise l'intuition de l'utilisateur

FAB

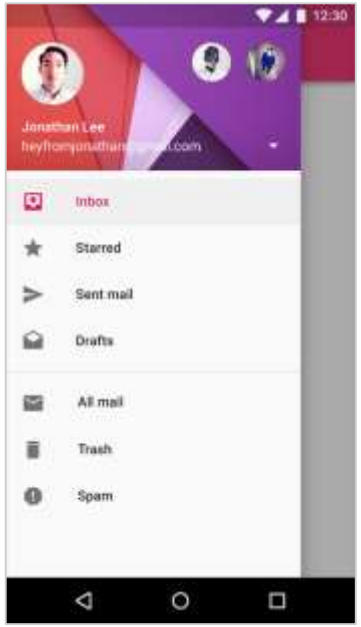




Tabs



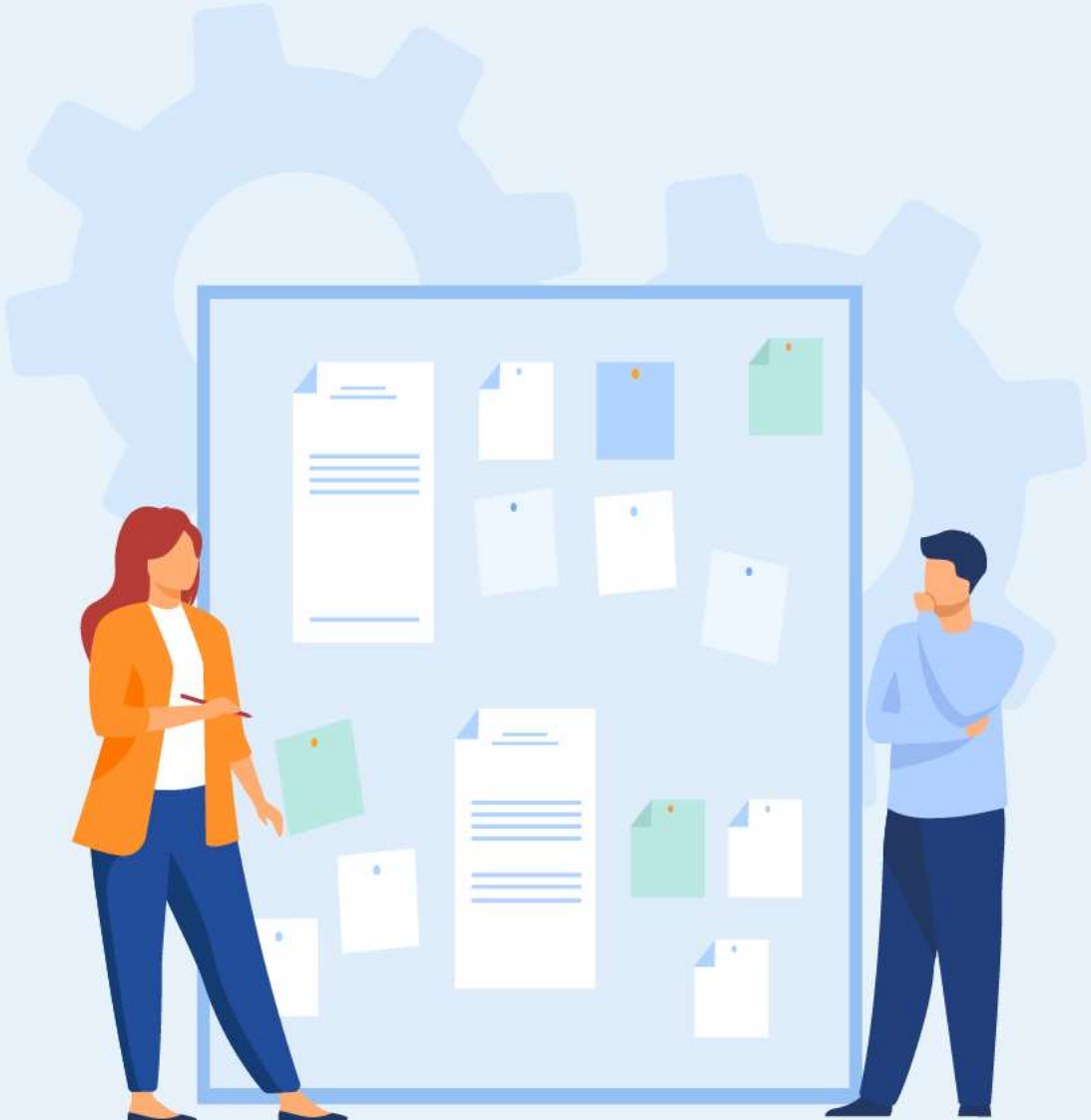
Snackbar



Navigation Drawer

PARTIE 2

Copyright - Tout droit réservé - OFPPT



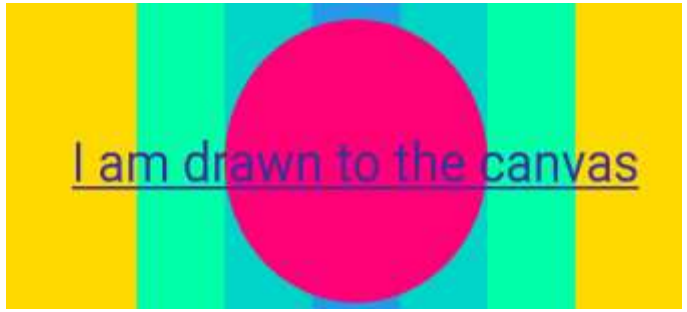
CHAPITRE 1

Coder des custom view

1. Manipulation des fichiers des styles
2. Utilisation des fichiers ressources
3. Intégration de Material design
4. **Maitrise des Canvas**

Pourquoi utiliser un canevas ?

Écrire sur un canevas pour réaliser des dessins plus complexes que ceux qui sont possibles avec les vues prédéfinies.



Classe Canvas

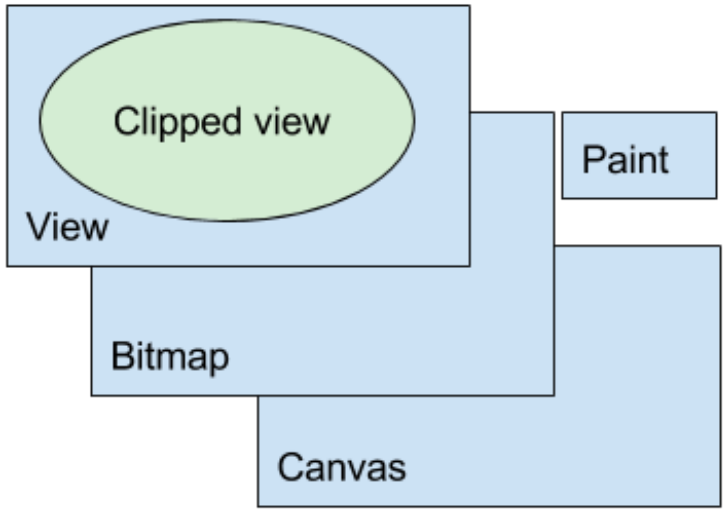
- Classe Canvas : surface de dessin logique pour le dessin 2D ;
- Utiliser Canvas lorsque l'application doit redessiner régulièrement ;
- La méthode onDraw() s'exécute sur le thread de l'interface utilisateur ;
- Le détourage définit les parties visibles par l'utilisateur ;
- Utiliser l'outil Profile GPU Rendering pour contrôler les performances de ses opérations de dessin.

Copyright - Tout droit réservé - OFPPT

Classes requises pour dessiner

Il faut une vue, un Bitmap, un Canvas et un objet Paint.

- La vue affiche le bitmap ;
- Bitmap est une surface de dessin physique ;
- Canvas possède une API pour dessiner sur le bitmap ;
- Paint donne un style à ce que l'on dessine.



Deux approches pour dessiner sur Canvas

2 façons de dessiner sur le canevas :

- **ImageView** : Dessiner sur le canevas dans le gestionnaire **onClick**

Le dessin peut changer lorsque l'utilisateur touche l'ImageView.

- **Custom View** : Dessiner sur le canevas dans la méthode **onDraw()**

Permet une interaction plus complexe avec l'utilisateur.

Copyright - Tout droit réservé - OFPPT

Utiliser un canvas dans un ImageView

Il est possible d'utiliser une ImageView pour dessiner sur un canevas.

Définir le gestionnaire onClick de l'ImageView pour qu'il dessine sur le canevas.

Pour ce faire, il faut utiliser :

- ImageView ;
- Bitmap ;
- Canvas ;
- Paint.

Etapes

1. Créer une ImageView ;
2. Créer un objet Paint pour définir le style de dessin ;
3. Implémenter le gestionnaire onClick pour l'ImageView afin de dessiner sur un canvas.

Copyright - Tout droit réservé - OFPPT

Créer ImageView en XML

```
<ImageView
      android:id="@+id/myimageview"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:onClick="drawSomething"/>
```

- drawSomething() va dessiner sur le canvas.

Paint

- Créer un objet Paint ;
- Stocker la façon de dessiner : couleur, style, épaisseur du trait, taille du texte ;
- L'objet Paint peut persister en dehors du gestionnaire onClick ; il n'est pas nécessaire de le recréer à chaque déclenchement du onClick.

Copyright - Tout droit réservé - OFPPT

Exemple de code Paint

```
private Paint mPaint = new Paint();
private Paint mPaintText =
        new Paint(Paint.UNDERLINE_TEXT_FLAG);
mPaint.setColor(Color.RED);
mPaintText.setColor(Color.BLUE);
mPaintText.setTextSize(70);
```

Le gestionnaire onClick dans l'ImageView

Définir le gestionnaire onClick pour dessiner sur le canvas :

1. Créer un bitmap ;
2. Associer Bitmap à ImageView ;
3. Créer un canevas avec un bitmap ;
4. Dessiner sur le Canvas ;
5. Appeler invalidate() sur la vue pour forcer le redessin.

Copyright - Tout droit réservé - OFPPT

Exemple de code onClick

```
mBitmap = Bitmap.createBitmap(
       vWidth, vHeight, Bitmap.Config.ARGB_8888);
mImageView.setImageBitmap(mBitmap);
mCanvas = new Canvas(mBitmap);
mCanvas.drawColor(mColorBackground);
mCanvas.drawText(getString(R.string.my_string),
                 100, 100, mPaintText);
view.invalidate();
```



Utiliser une vue personnalisée

Utiliser une vue personnalisée pour permettre au dessin de changer en réponse à une interaction plus complexe avec l'utilisateur.

- Hériter de l'une des classes View ;
- Redéfinir onDraw() et 9 pour dessiner ;
- Remplacer onTouchEvent() pour gérer les contacts de l'utilisateur.

Copyright - Tout droit réservé - OFPPT

Exemple de scénario d'utilisation

Par exemple, l'utilisateur dessine une image sur le canvas.



Aperçu des étapes à suivre pour dessiner

1. Créer un Bitmap ;
2. Associer le Bitmap à la vue ;
3. Créer un Canvas avec un objet Bitmap ;
4. Utiliser Paint pour le style de dessin ;
5. Dessiner dans onDraw() ;
6. Dessiner quand la taille change dans onSizeChanged() ;
7. Appeler invalidate() sur la vue ...demande à Android de redessiner l'affichageu.

Copyright - Tout droit réservé - OFPPT

1. Créer une classe de vue personnalisée

- Créer une classe de vue personnalisée ;
- Étendre n'importe quelle vue qui possède les fonctionnalités dont le client a besoin :

```
public class MyCanvasView extends View {...}
```

2. Utiliser setContentView()

Définir la vue du contenu de l'activité pour l'instance de la vue personnalisée :

```
MyCanvasView myCanvasView;
myCanvasView = new MyCanvasView(this);
setContentView(myCanvasView);
```

3. Initialiser l'objet Paint

Dans le constructeur de la vue personnalisée :

- Créer l'objet Paint et définir les propriétés initiales ;
- Initialiser les variables membres ;
- Obtenir une référence au contexte ;

Impossible de créer un Canvas car la vue non inflatée n'a pas de taille.

Exemple de code Paint :

```
mPaint = new Paint();
   mPaint.setColor(backgroundColor);
   mPaint.setAntiAlias(true);
   mPaint.setDither(true);
   mPaint.setStyle(Paint.Style.STROKE); // default: FILL
   mPaint.setStrokeJoin(Paint.Join.ROUND); // default: MITER
   mPaint.setStrokeCap(Paint.Cap.ROUND); // default: BUTT
   mPaint.setStrokeWidth(12); // default: Hairline-width
```


Copyright - Tout droit réservé - OFPPT

4. Créer un bitmap dans onSizeChanged()

Dans la vue personnalisée, redéfinir onSizeChanged() :

- Créer un Bitmap ;
- Crée un Canvas avec le Bitmap ;
- Appelé lorsque la taille de la vue est attribuée pour la première fois, ou lorsque sa taille change ;
- Dans onSizeChanged(), calculer les positions, les dimensions, tout ce qui est lié à la taille de la vue.

Exemple onSizeChanged()

```
@Override
protected void onSizeChanged(int width, int height,
                             int oldWidth, int oldHeight) {
   super.onSizeChanged(width, height, oldWidth, oldHeight);
   mBitmap = Bitmap.createBitmap(width, height,
                                 Bitmap.Config.ARGB_8888);
   mCanvas = new Canvas(mBitmap);
   mCanvas.drawColor(mDrawColor);
}
```

5. Code de redéfinition de onDraw()

Peut être simple ou complexe, doit être rapide, fonctionne sur UI Thread.

```
@Override
protected void onDraw(Canvas canvas) {
   super.onDraw(canvas);
   canvas.drawBitmap(mBitmap, 0, 0, mPaint);
   canvas.drawPath(mPath, mPaint);
}
```


Copyright - Tout droit réservé - OFPPT

6. Redéfinition de onTouchEvent()

- Dessiner en réponse au mouvement de l'utilisateur ;
- Dans l'exemple suivant, **invalidate()** est à l'intérieur des déclarations de cas parce qu'il y a beaucoup d'autres types d'événements de mouvement passés dans cet écouteur, et nous ne voulons pas invalider la vue pour ceux-ci.

```
switch (event.getAction()) {
        case MotionEvent.ACTION_DOWN:
            touchStart(x, y);
            break;
        case MotionEvent.ACTION_MOVE:
            touchMove(x, y);
            invalidate();
            break;
        case MotionEvent.ACTION_UP:
            touchUp();
            invalidate();
            break;
```


Copyright - Tout droit réservé - OFPPT

Exemple de code

```
private void touchMove(float x, float y) {
   float dx = Math.abs(x - mX); // distance moved
   float dy = Math.abs(y - mY);
   if (dx >= TOUCH_TOLERANCE || dy >= TOUCH_TOLERANCE) {
       // QuadTo() adds a quadratic bezier from the last point,
       // approaching control point (x1,y1), and ending at (x2,y2).
       mPath.quadTo(mX, mY, (x + mX)/2, (y + mY)/2);
       mX = x;
       mY = y;
   }
}
```


Copyright - Tout droit réservé - OFPPT

Opérations sur Canvas

- Remplir le Canvas avec de la couleur ;
- Dessiner des formes, telles que des rectangles, des arcs et des chemins ;
- Dessiner du texte stylisé par les propriétés d'un objet Paint ;
- Styliser des formes et du texte à l'aide d'objets Paint.

Autres opérations

- Sauvegarder et restaurer l'état du Canvas ;
- Appliquer des transformations, telles que la translation, la mise à l'échelle ou des transformations personnalisées ;
- Clip : appliquer une forme ou un chemin pour définir les parties visibles de Canvas.

Copyright - Tout droit réservé - OFPPT

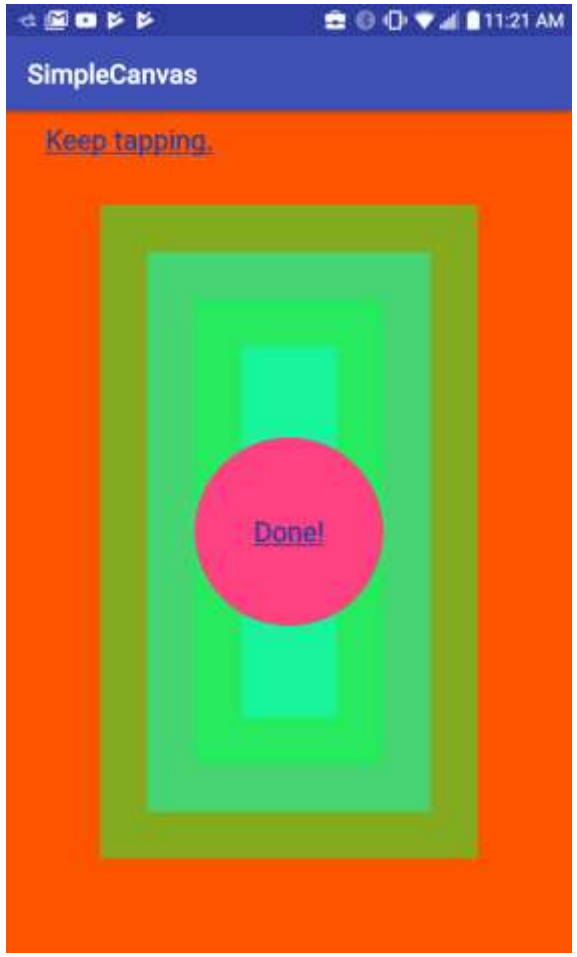
Remplir le Canvas avec de la couleur

```
mCanvas.drawColor(mColorBackground);
```

Dessiner des formes primitives

- Formes primitives :
  drawRect(), drawOval(), drawArc()
- Rempli, souligné, ou les deux : setStyle()

```
mPaint.setStyle(Paint.Style.FILL_AND_STROKE);
mCanvas.drawCircle (x, y, radius, mPaint);
mRect.set (x, y, width, height);
mCanvas.drawRect(mRect, mPaint);
```



Copyright - Tout droit réservé - OFPPT

Dessiner des formes complexes

- Dessiner des formes plus complexes à l'aide de la classe Path ;
- Définir une forme en ajoutant des lignes et des courbes à un objet Path, puis dessiner la forme en utilisant drawPath() ;
- Les chemins peuvent être soulignés, remplis, ou les deux, selon la méthode setStyle().

Dessiner du texte

- Dessiner du texte avec drawText() ;
- Spécifier la police de caractères avec setTypeface() ;
- Définir la couleur du texte avec setColor().

```
mPaintText.setColor(textColor);
mPaintText.setTypeface(Typeface.DEFAULT_BOLD);
mCanvas.drawText(getString(R.string.keep_tapping),
                                100, 100, mPaintText);
```


Copyright - Tout droit réservé - OFPPT

Transformations: translate()

- Utiliser translate() pour déplacer l'origine du canvas ;
- Utiliser pour dessiner la même forme à différents endroits :

```
canvas.translate(dx, dy);
```

Transformations: rotate()

- Utiliser rotate() pour faire pivoter le canvas d'un certain nombre de degrés :

```
canvas.rotate(180);
```

Transformations: skew()

- Incliner le canvas en appelant la méthode skew() ;
- Avec skew(), il est possible d'obtenir des effets de texte intéressants :

```
canvas.skew(0.2f, 0.3f);
```



Combiner les transformations

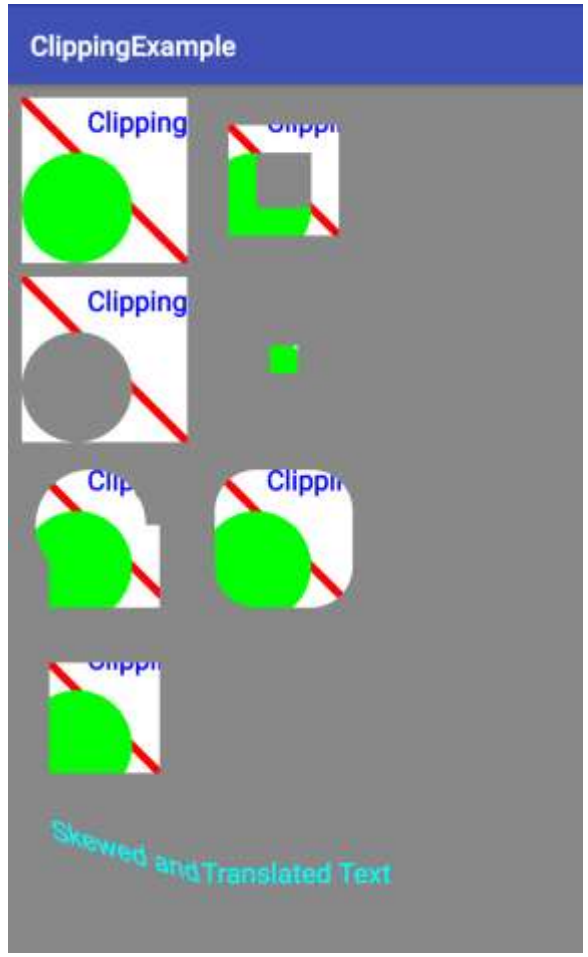
```
mPaint.setTextSize(120);
canvas.translate(100, 1800);
canvas.skew(0.2f, 0.3f);
canvas.drawText("Transformed", 400, 60, mPaint);
```


Copyright - Tout droit réservé - OFPPT

Clipping

Le clipping est un moyen de définir les régions d'une image, d'un canvas ou d'un bitmap qui sont dessinées ou non à l'écran.

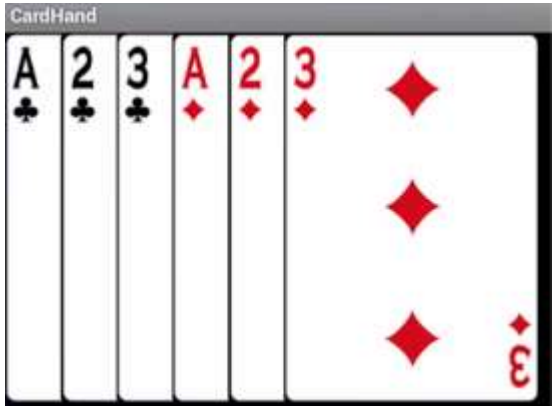
- Réduire la surimpression ;
- Améliorer les performances en dessinant moins ;
- Créer des effets et des animations intéressants pour l'interface utilisateur.



Copyright - Tout droit réservé - OFPPT

À quoi sert le clippage ?

- Ne dessiner que les parties visibles des cartes ;
- Afficher uniquement les parties pertinentes d'une image ;
- Définir une région d'écrêtage de n'importe quelle forme.





Copyright - Tout droit réservé - OFPPT

clipRect()

- Utiliser canvas.clipRect() pour définir une région d'écrêtage rectangulaire.

```
canvas.clipRect(x, Y, right, bottom);
```

clipPath()

- Utiliser canvas.clipPath() pour créer une région d'écrêtage personnalisée :

```
mPath.addCircle(radius, x, y, Path.Direction.CCW);
canvas.clipPath(mPath, Region.Op.DIFFERENCE);
```

- CCW signifie dessiner un cercle dans le sens inverse des aiguilles d'une montre ;
- Region.Op.DIFFERENCE indique comment appliquer la région au canvas.

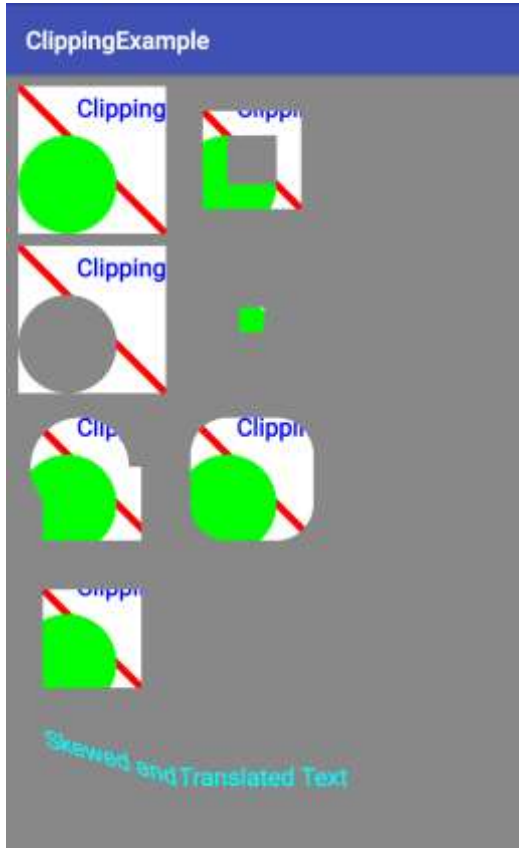
Copyright - Tout droit réservé - OFPPT

quickReject()

- quickReject() vérifie si un rectangle ou un chemin spécifié se trouverait complètement en dehors des régions actuellement visibles, après que toutes les transformations ont été appliquées;
- Décider efficacement quels objets ne doivent pas être dessinés du tout

Écrêtage complexe

- Combinaison ou intersection de plusieurs régions de découpage.



Copyright - Tout droit réservé - OFPPT

Pile d'états de dessin

- Le contexte d'activité maintient une pile d'états de dessin ;
- Chaque état comprend les transformations et les régions d'écrêtage actuellement appliquées ;
- Impossible de supprimer les régions d'écrêtage ;
- Annuler une transformation en la renversant est source d'erreurs.

Sauvegarde et restauration

- Sauvegarder l'état du Canvas en utilisant save() ;
- Appliquer les transformations ;
- Dessiner ;
- Restaurer l'état précédent du Canvas en utilisant restore().

```
canvas.save();
mPaint.setTextSize(120);
canvas.translate(100, 1800);
canvas.skew(0.2f, 0.3f);
canvas.drawText("Skewing", 400, 60, mPaint);
canvas.restore();
```

```
canvas.save();
mPaint.setColor(Color.CYAN);
canvas.translate(600, 1800);
canvas.drawText("Save/Restore", 400, 60, mPaint);
canvas.restore();
```

Copyright - Tout droit réservé - OFPPT

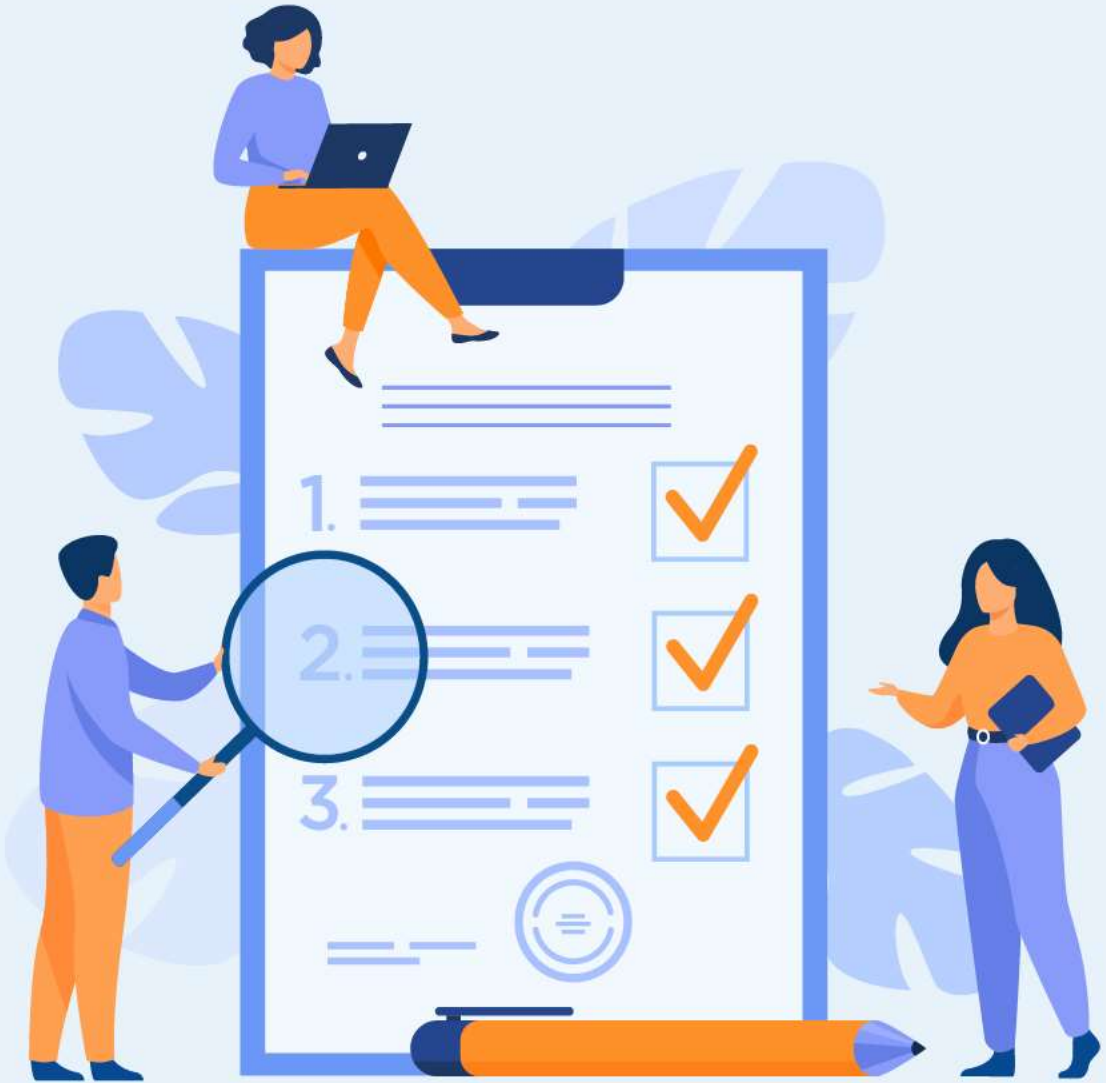
Résumé du Canvas

- Il faut disposer d'une View, d'un Canvas, d'une Paint et d'un objet Bitmap ;
- Le Bitmap est la surface de dessin physique. Le Canvas fournit une API pour dessiner sur la Bitmap, le Paint sert à styliser ce que le dessin, et la View affiche la Bitmap ;
- Il faut créer un Bitmap, l'associer à une vue, créer un Canvas avec un objet Paint pour le Bitmap, puis dessiner ;
- Vous devez **invalider()** la vue lorsque vous avez fini de dessiner, afin que le système Android redessine l'affichage ;
- Tous les dessins sont effectués sur le thread de l'interface utilisateur, donc les performances sont importantes.

Résumé des Clips

- Le contexte d'une activité maintient un état qui préserve les transformations et les régions d'écrêtage pour le Canvas ;
- Utiliser **canvas.save()** et **canvas.restore()** pour dessiner et revenir à l'état d'origine de votre Canvas ;
- Pour dessiner plusieurs formes sur un Canvas, il faut soit calculer leur emplacement, soit déplacer (translater) l'origine de votre surface de dessin ; ce dernier point peut faciliter la création de méthodes utilitaires pour les séquences de dessin répétées ;
- Les régions d'écrêtage peuvent être n'importe quelle forme, combinaison de formes ou chemin ;
- Ajouter, soustraire et croiser des régions d'écrêtage pour obtenir exactement la région dont vous avez besoin ;
- Vous pouvez appliquer des transformations au texte

Copyright - Tout droit réservé - OFPPT



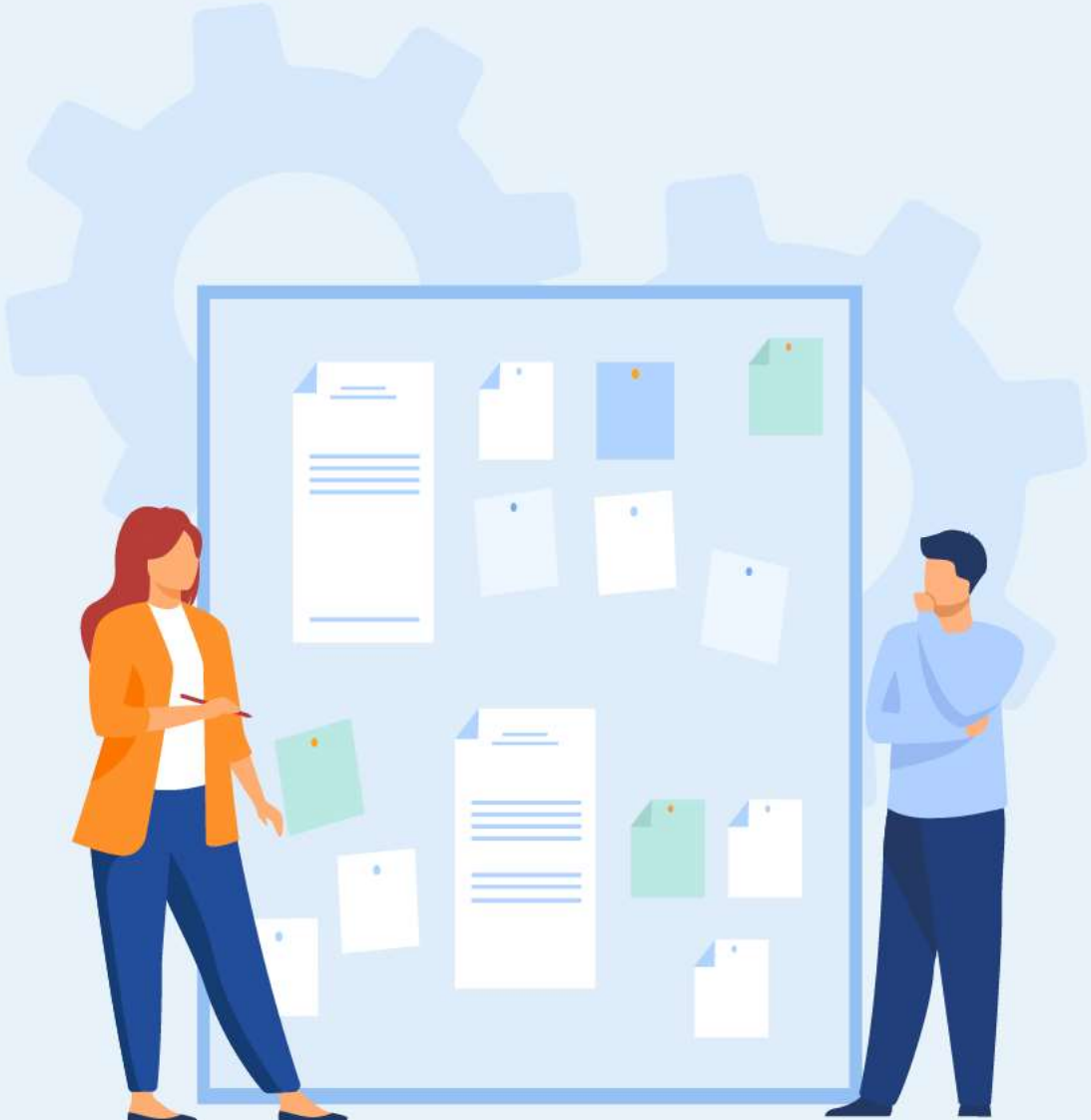
CHAPITRE 2

Gérer des animations entre les activités

Ce que vous allez apprendre dans ce chapitre :

- Définition d’une animation ;
- Utilisation des transitions customisées ;
- Maitrise des animations (Explode, Slide, Fade).

4 heures



CHAPITRE 2

Gérer des animations entre les activités

1. **Définition d’une animation**
2. Utilisation des transitions customisées
3. Maitrise des animations (Explode, Slide, Fade)

Qu'est-ce que l'animation ?

L'animation est une technique permettant de créer l'illusion d'un objet en mouvement en montrant une série d'images discrètes qui changent dans le temps.



Copyright - Tout droit réservé - OFPPT

Exemples d'animation

- Un flip book a une image différente sur chaque page - lorsque la page est tournée rapidement, les yeux perçoivent le mouvement ;
- La claymation est un type d'animation en stop-motion ;
- Les animations d'interface utilisateur, telles que la projection d'un élément de liste ;
- ... et des millions de jeux mobiles avec des animations de personnages, d'environnements et d'interfaces utilisateur.

Qu'est-ce qu'une fréquence d'image ?

- Un cadre est une image dans une séquence animée (un écran dans une séquence d'écrans d'interface utilisateur sur un appareil) ;
- La cadence d'image (fréquence d'images) est la vitesse à laquelle les images sont affichées ;
- Le taux de rafraîchissement est la fréquence à laquelle le système Android redessine l'écran.

Copyright - Tout droit réservé - OFPPT

Si la fréquence d'images est...

- ...plus lent que le taux de rafraîchissement, l'animation peut bégayer ;
- ...plus rapide que le taux de rafraîchissement, l'application gaspille des ressources ;
- ...identique au taux de rafraîchissement de l'écran, les animations sont fluides et aucune ressource n'est gaspillée.

Idéalement, la fréquence d'images doit correspondre étroitement à la fréquence de rafraîchissement de l'écran.

Gestion de la fréquence d'images

- Heureusement, le système Android gère la fréquence d'images ;
- Dans la plupart des situations, la gestion de la fréquence d'images des animations n'est pas nécessaire.

Copyright - Tout droit réservé - OFPPT

Types d'animation pour Android

- View animation
- Property animation
- Drawable animation
- Physics-based animation
    - Spring animation
    - Fling animation

View animation

- View animation est un système d'animation des vues ;
- Il s'agit d'un ancien système limité aux objets View ;
- Relativement facile à mettre en place ;
- Offre suffisamment de possibilités pour répondre aux besoins de nombreuses applications.

Copyright - Tout droit réservé - OFPPT

Spécificités de View-animation

- Ne modifie que l'endroit où la vue est dessinée, pas la vue elle-même :
    - Si un bouton est animé pour se déplacer à l'écran, l'endroit où l'utilisateur touche le bouton ne change pas. Il faut mettre en œuvre une logique propre pour gérer cela.
- Moins de temps et moins de code requis que d'autres animations telles que l'animation des propriétés.

Animation des propriétés

- Animation des propriétés disponible depuis Android 3.0 (niveau 11 de l'API) ;
- Animation des propriétés de tout objet ;
- Extensible pour animer les propriétés de types personnalisés ;
- Il est préférable d'utiliser l'animation des propriétés plutôt que l'animation de la vue.

Copyright - Tout droit réservé - OFPPT

Animation des propriétés

- Animer presque tout, comme la couleur, la taille, la position ;
- Modifie la valeur d'une propriété sur une durée déterminée ;
- Attribuer des animateurs aux propriétés à animer ;
- Définir les aspects de l'animation, tels que la durée ou le taux de changement (interpolation).

Exemple : Animer un cercle pour qu'il devienne plus grand en augmentant son rayon au fil du temps.

Propriété à animer

Pour animer un objet dont la valeur peut être définie par une méthode "setter".

- La propriété ne doit pas nécessairement être un attribut existant ;
- La propriété à animer doit avoir une méthode "setter" ;
- Définir la méthode "setter" si elle n'existe pas déjà.

Copyright - Tout droit réservé - OFPPT

Animer une propriété existante

Pour animer la taille du texte d'une vue de texte :

- TextView a une propriété **textSize** et une méthode **setTextSize**() ;
- Animez la propriété **textSize** pour que le texte soit plus grand ou plus petit.

Animer une propriété virtuelle

Pour animer la couleur du texte, la couleur d'arrière-plan et la rotation du texte ensemble :

- Définir **setMyStrangeProperty**() pour modifier les valeurs des attributs **textColor**, background et rotation ;
- Animer l'attribut **myStrangeProperty**.

Copyright - Tout droit réservé - OFPPT

Définir les caractéristiques de l'animation

Le système d'animation des propriétés permet de définir les caractéristiques suivantes d'une animation :

- Durée ;
- Interpolation du temps (Interpolation temporelle) ;
- Nombre de répétitions et comportement ;
- Paramètres de l'animateur ;
- Délai de rafraîchissement des images.

Durée

- Combien de temps dure une animation ;
- La durée par défaut est de 300 millisecondes.

Copyright - Tout droit réservé - OFPPT

Interpolation temporelle

- Spécifier comment les valeurs de la propriété sont calculées en fonction du temps écoulé de l'animation ;
- Déplacement linéaire ou accélération/décélération ;
- Utiliser les interpolateurs fournis par le système ou créer ses propres interpolateurs.

Nombre de répétitions et comportement

- Indiquer si une animation doit être répétée ou non lorsqu'elle atteint la fin de sa durée ;
- Combien de fois doit-elle être répétée ?
- Jouer en sens inverse ;
- Jouer en avant puis en arrière, pour le nombre de répétitions.

Copyright - Tout droit réservé - OFPPT

Objet AnimatorSet

- Regrouper les animations en ensembles logiques qui sont joués ensemble, séquentiellement ou après des délais spécifiques.

Exemple : Coordonner plusieurs balles rebondissantes

Qu'est-ce qu'un interpolateur de temps ?

L'interpolateur temporel définit la manière dont les valeurs d'une animation évoluent dans le temps :

- Spécifier que les animations se déroulent de façon linéaire (régulière) ;
- Utiliser un temps non linéaire, par exemple, accélérer au début et décélérer à la fin de l'animation ;
- Utiliser un interpolateur prédéfini ou définir son propre interpolateur.

Copyright - Tout droit réservé - OFPPT

Interpolateurs prédéfinis

- LinearInterpolator : Le taux de changement est constant ; chaque image modifie l'objet de la même manière ;
- AccelerateDecelerateInterpolator : Le taux de changement commence et se termine lentement, mais s'accélère au milieu ;
- AnticipateInterpolator : Le changement commence vers l'arrière puis s'élance vers l'avant, comme si un élastique claquait ;
- BounceInterpolator : Change les rebonds à la fin ;
- PathInterpolator : Suit un chemin indiqué par les utilisateurs
- ...et bien d'autres encore.

Pour implémenter son propre interpolateur, utiliser l'interface TimeInterpolator.

Classes pour l'animation des propriétés

- Un Animator gère l'animation ;
- ObjectAnimator met à jour les propriétés avec de nouvelles valeurs ;
- TypeEvaluator calcule les valeurs d'une propriété donnée à partir des données de synchronisation, de la valeur de départ et de la valeur finale : IntEvaluator, FloatEvaluator ou ArgbEvaluator.
- Un TimeInterpolator

Copyright - Tout droit réservé - OFPPT

Création d'instances d'ObjectAnimator

ObjectAnimator possède des méthodes de fabrique (factory) pour créer une instance qui s'anime de différentes manières :

- ofArgb(... , int... values) anime entre les valeurs de couleur ;
- ofFloat(... , Path path) anime des coordonnées le long d'une trajectoire ;
- ofFloat(... , float... values) anime entre des valeurs flottantes ;
- ofInt(... , int... values) anime entre des valeurs int.
- ...et plus encore

1. Créer une instance d'ObjectAnimator

- Choisir la méthode de fabrique et fournir les arguments :
  - Objet à animer ;
  - Nom de la propriété sous forme de chaîne ;
  - Valeur de départ ;
  - Valeur finale.

```
ObjectAnimator animator =  ObjectAnimator.ofFloat(mView, "radius", 0, 50) ;
```


Copyright - Tout droit réservé - OFPPT

2. Définir l'interpolateur

- Optionnellement, définir un interpolateur (linéaire par défaut)

```
animator.setInterpolator(new AccelerateInterpolator()) ;
```

3. Définir la durée en millisecondes

```
animator.setDuration(400);
```


Copyright - Tout droit réservé - OFPPT

4. Démarrer l'animation

```
animator.start();
```

Paramètre requis pour la propriété animée

Pour que l'ObjectAnimator mette à jour les propriétés correctement :

- La propriété doit avoir une fonction "setter" setPropertyName() d'un type correspondant, par exemple, si le nom de la propriété est radius, il faut une méthode setRadius() ;
- Si la propriété View, appeler la méthode invalidate() dans le callback onAnimationUpdate().

Copyright - Tout droit réservé - OFPPT

Écouteurs d'animation

Écouter les événements importants pendant le processus d'animation :

```
ValueAnimator fadeAnim =
    ObjectAnimator.ofFloat(newBall, "alpha", 1f, 0f);
fadeAnim.setDuration(250);
fadeAnim.addListener(new AnimatorListenerAdapter() {
    public void onAnimationEnd(Animator animation) {
        balls.remove(((ObjectAnimator)animation).getTarget());
    }
}
```


Copyright - Tout droit réservé - OFPPT

AnimatorSet

- AnimatorSet exécute les animations les unes par rapport aux autres ;
- Exécuter les animations avant, après ou en même temps que d'autres animations.

AnimatorSet pour combiner les animations

Séquence d'animation pour l'exemple de code sur les 2 diapositives suivantes :

1. Jouer bounceAnim ;
2. Jouer squashAnim1, squashAnim2, stretchAnim1, et stretchAnim2 en même temps ;
3. Jouer bounceBackAnim.

Copyright - Tout droit réservé - OFPPT

Code pour AnimatorSet (1)

```
// Create bouncer AnimatorSet
AnimatorSet bouncer = new AnimatorSet();
bouncer.play(bounceAnim).before(squashAnim1);
bouncer.play(squashAnim1).with(squashAnim2);
bouncer.play(squashAnim1).with(stretchAnim1);
bouncer.play(squashAnim1).with(stretchAnim2);
bouncer.play(bounceBackAnim).after(stretchAnim2);
bouncer.start();
```

Code pour AnimatorSet (2)

```
// Create fadeAnim animator
ValueAnimator fadeAnim = ObjectAnimator.ofFloat(newBall,
                                    "alpha", 1f, 0f);
fadeAnim.setDuration(250);

// Create and play animatorSet
AnimatorSet animatorSet = new AnimatorSet();
animatorSet.play(bouncer).before(fadeAnim);
animatorSet.start();
```


Copyright - Tout droit réservé - OFPPT

Animation des propriétés en XML

Enregistrer dans le répertoire res/animator/

Classes avec les balises XML correspondantes :

ValueAnimator	<animator>
ObjectAnimator	<objectAnimator>
AnimatorSet	<set>

Pour en savoir plus, consultez le site Property Animation

Exemple

```
<set android:ordering="sequentially">
    <set>
        <objectAnimator
            android:propertyName="x"
            android:duration="500"
            android:valueTo="400"
            android:valueType="intType"/>
        <objectAnimator
            android:propertyName="y"
            android:duration="500"
            android:valueTo="300"
            android:valueType="intType"/>
    </set>
    <objectAnimator
        android:propertyName="alpha"
        android:duration="500"
        android:valueTo="1f"/>
</set>
```


Copyright - Tout droit réservé - OFPPT

Exécution de l'exemple XML

- Gonfler les ressources XML en un objet AnimatorSet ;
- Définir les objets cibles pour toutes les animations avant de les lancer.

```
AnimatorSet set = (AnimatorSet) AnimatorInflater.loadAnimator(myContext,
    R.anim.property_animator);
set.setTarget(myObject);
set.start();
```


Copyright - Tout droit réservé - OFPPT

Gérer des animations entre les activités
Drawable animation

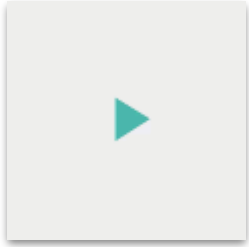
Définition

- L'animation Drawable (animation dessinée) affiche les ressources Drawable l'une après l'autre, comme un rouleau de film ;
- Utile pour animer des choses qui sont plus faciles à représenter avec des ressources Drawable, comme une progression de bitmaps.



Créer une animation dessinée

- La façon la plus simple de créer est à partir d'un fichier XML ;
- Enregistrer le fichier dans res/drawable ;
- Le fichier XML contient l'élément <animation-list> comme noeud racine ;
- Chaque <item>enfant définit une ressource dessinable et la durée d'un cadre pour un cadre.



PARTIE 2

Copyright - Tout droit réservé - OFPPT

Exemple XML : rocket_thrust .xml ⇒ R.drawable.rocket_thrust

```
<animation-list
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:oneshot="true">
    <item android:drawable="@drawable/rocket_thrust1"
        android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust2"
        android:duration="200" />
    <item android:drawable="@drawable/rocket_thrust3"
        android:duration="200" />
</animation-list>
```


Copyright - Tout droit réservé - OFPPT

Exemple d'une AnimationDrawable

```
AnimationDrawable rocketAnimation;

public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.main);
  ImageView rocketImage = (ImageView)
                          findViewById(R.id.rocket_image);

rocketImage.setBackgroundResource(R.drawable.rocket_thrust);
  rocketAnimation = (AnimationDrawable)
                          rocketImage.getBackground();
}
```

Démarrer une AnimationDrawable

```
public boolean onTouchEvent(MotionEvent event) {
  if (event.getAction() == MotionEvent.ACTION_DOWN) {
    rocketAnimation.start();
    return true;
  }
  return super.onTouchEvent(event);
}
```

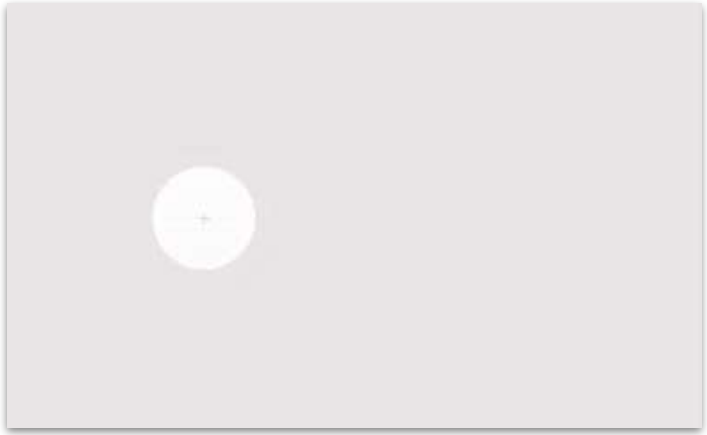

Copyright - Tout droit réservé - OFPPT

Définition

- L'animation basée sur la physique s'appuie sur les lois de la physique pour donner un haut degré de réalisme à l'animation ;
- Utilise les principes fondamentaux de la physique pour créer des animations.

Poussé par la force

Par exemple, une animation est entraînée par une force et s'arrête lorsque la force atteint l'équilibre.



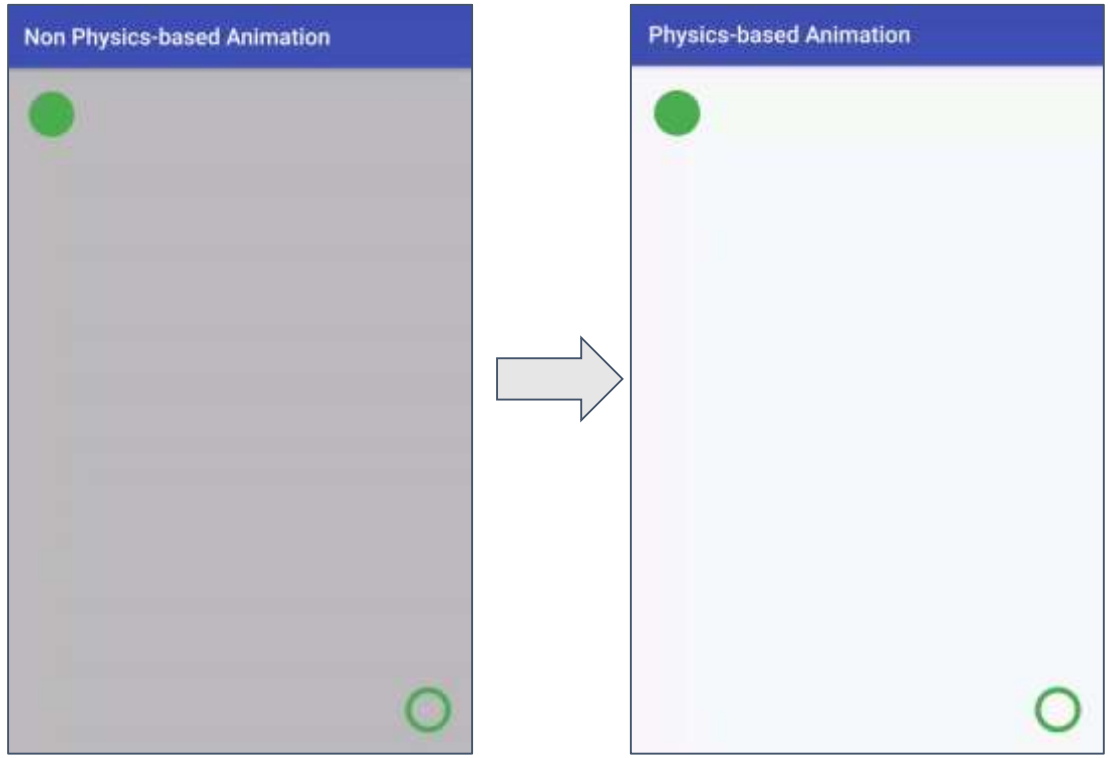
Spring release effect

Copyright - Tout droit réservé - OFPPT

Avantages de l'animation basée sur la physique

- Aspect naturel :
  - Flexible et imitant les mouvements en temps réel.
- Correction de trajectoire :
  - Conserve l'élan lorsque la cible change et se termine par un mouvement plus fluide.
- Réduit le janking visuel :
  - Apparaît plus réactif, moins de perturbations visuelles.

Non basé sur la physique vs. basé sur la physique



Copyright - Tout droit réservé - OFPPT

android.support.animation

L'API android.support.animation a été introduite dans la version 25.3.0 de la bibliothèque de support.

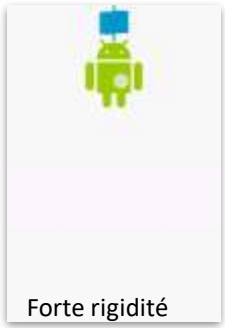
Elle comprend des classes pour les animations basées sur la physique :

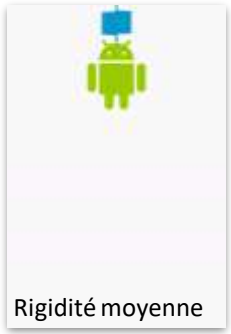
```
dependencies {
    // Java language implementation
    implementation "androidx.dynamicanimation:dynamicanimation: version_number
    // Kotlin
    implementation "androidx.dynamicanimation:dynamicanimation-ktx: version_number "
}
```

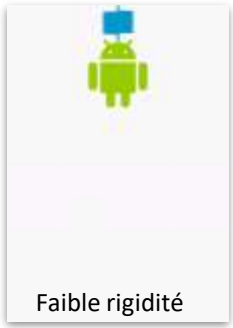

Copyright - Tout droit réservé - OFPPT

Animation de ressorts

- Dans l'animation du ressort, personnaliser la rigidité du ressort, le taux d'amortissement et la position finale ;
- La force du ressort met à jour la valeur et la vitesse de l'animation à chaque image ;
- L'animation se poursuit jusqu'à ce que la force du ressort atteigne l'équilibre.









Copyright - Tout droit réservé - OFPPT

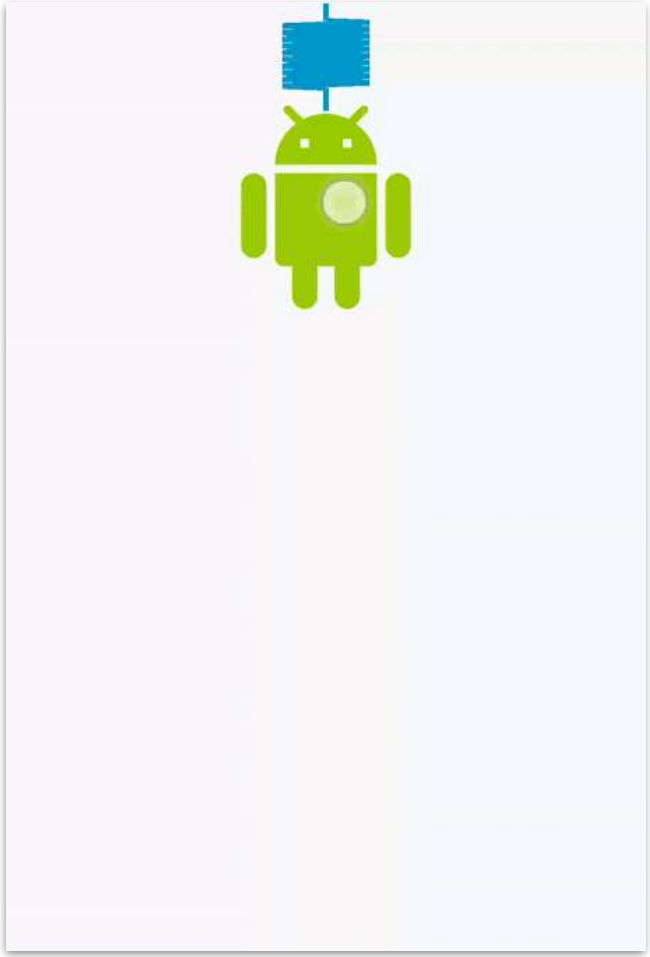
Animation de ressorts

```
final SpringAnimation anim = new
           SpringAnimation(
               this, DynamicAnimation.Y, 10);

anim.setStartVelocity(10000);

anim.getSpring().setStiffness(STIFFNESS_LOW);

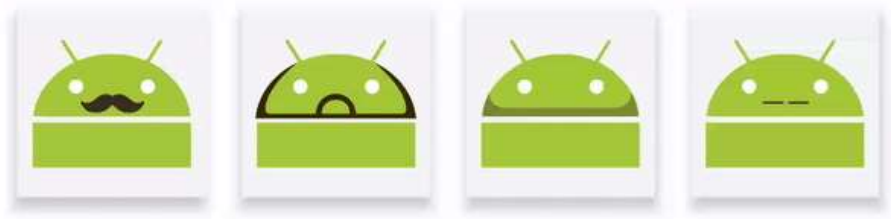
anim.start();
```



Copyright - Tout droit réservé - OFPPT

Animation de flottement

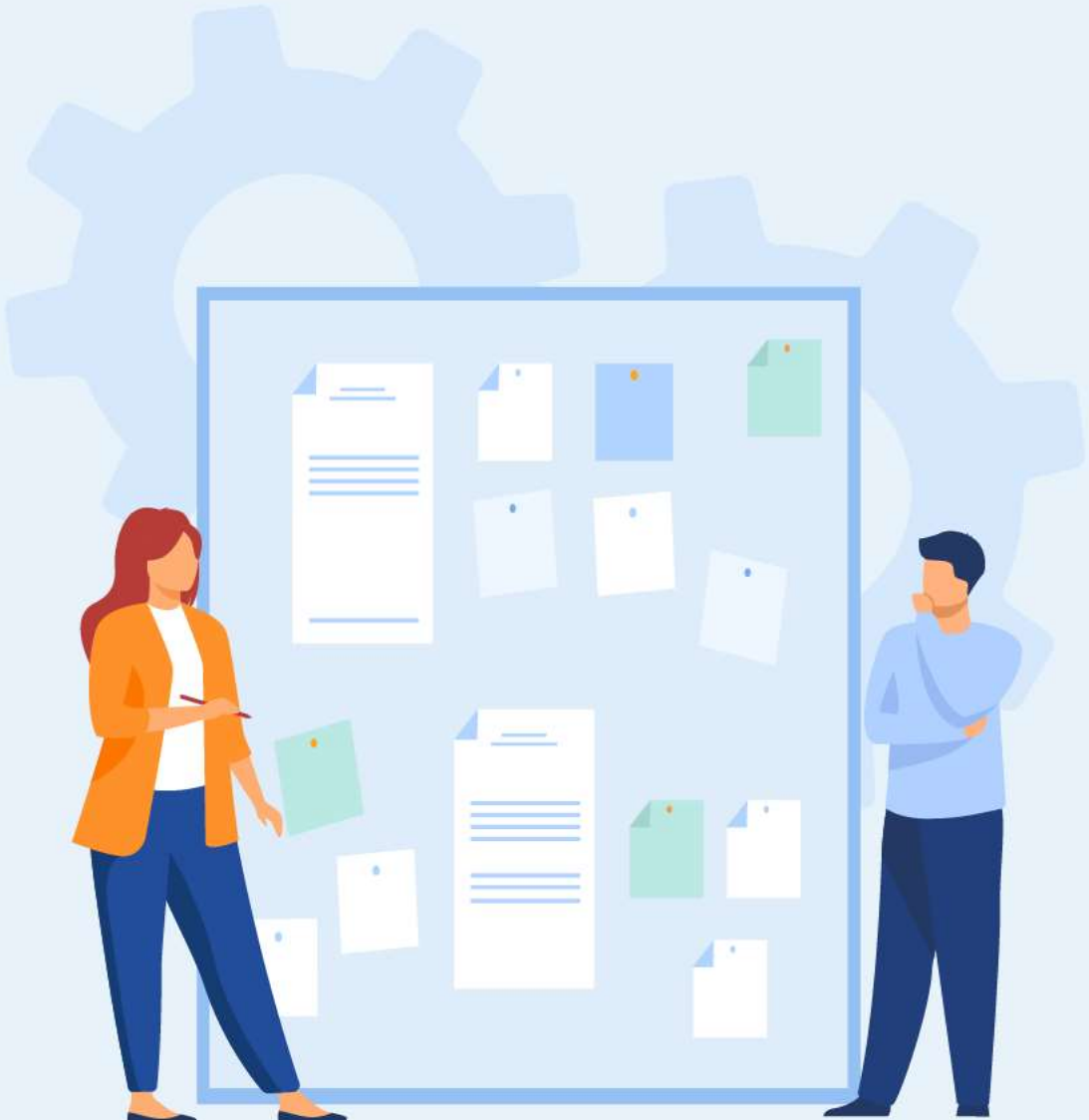
- L'animation du flottement (Fling animation) utilise une force de friction proportionnelle à la vitesse de l'objet ;
- Utilisée pour mettre fin à l'animation de manière progressive ;
- L'élan initial provenant de la vitesse du geste ralentit progressivement ;
- L'animation se termine lorsque la vitesse ne change plus de façon visible.



Code pour l'animation fling

```
FlingAnimation fling = new FlingAnimation(
                this, DynamicAnimation.ROTATION_X);
fling.setStartVelocity(150)
        .setMinValue(0)
        .setMaxValue(1000)
        .setFriction(0.1f)
        .start();
```


Copyright - Tout droit réservé - OFPPT



CHAPITRE 2

Gérer des animations entre les activités

1. Définition d’une animation
2. **Utilisation des transitions customisées**
3. Maitrise des animations (Explode, Slide, Fade)

Android Transition Framework

- Android Transition Framework permet d'animer toutes sortes de mouvements dans l'interface utilisateur en fournissant simplement la disposition de départ et la disposition d'arrivée.
- Android Transition Framework permet de :
    - Animer automatiquement de la vue de départ à la vue d'arrivée en fournissant les mises en page ;
    - Utiliser des animations communes prédéfinies telles que **Translate**, **Resize** et **Fade**.;
    - Charger des animations intégrées à partir de fichiers de ressources de mise en page ;
    - Appliquer un ou plusieurs effets d'animation à un groupe de vues à la fois ;
    - Utiliser des scènes chargées à partir de mises en page complexes ou générées par un programme ;
    - Contrôler le cycle de vie et la progression d'une animation en fournissant les callbacks ;
    - Créer des animations de transition personnalisées.

Transitions d'activités

Au moment d'ouvrir et de fermer une activité, Android joue une transition par défaut. Cette transition peut être personnalisée pour refléter la marque et/ou différencier l'application. Les transitions d'activité sont disponibles à partir d'Android 5.0 Lollipop (niveau 21 de l'API). Les transitions d'activité comportent deux parties : la transition d'entrée et la transition de sortie. La transition d'entrée indique comment l'activité et ses vues seront animées à l'ouverture de l'activité. La transition de sortie, quant à elle, décrit comment l'activité et ses vues sont animées lorsque l'activité est fermée ou qu'une nouvelle activité est ouverte. Android prend en charge les transitions intégrées suivantes :

- **Explode** : Cette fonction déplace les vues vers l'intérieur ou l'extérieur du centre ;
- **Fade** : Cette vue apparaît ou disparaît lentement ;
- **Slide** : Cette fonction déplace les vues vers l'intérieur ou l'extérieur à partir des bords.

Il existe deux façons d'ajouter des transitions d'activité : via le XML et via le code.

Copyright - Tout droit réservé - OFPPT

Ajout de transitions d'activité via XML

- Les transitions d'activité peuvent être ajoutées via XML.
- La première étape consiste à activer les transitions du contenu de la fenêtre. Pour ce faire, il suffit d'ajouter le thème de l'activité dans themes.xml comme suit :

```
<item name="android:windowActivityTransitions">true</item>
```

- Ensuite, ajouter les transitions d'entrée et de sortie avec les attributs de style **android:windowEnterTransition** et **android:windowExitTransition**. Par exemple, si les transitions par défaut de **@android:transition/** sont utilisées, les attributs à ajouter sont les suivants :

```
<item name="android:windowEnterTransition">
@android:transition/slide_left</item>

<item name="android:windowExitTransition">
@android:@transition/explode</item>
```

Le fichier **themes.xml** sera alors le suivant :

```
<style name="AppTheme"
        parent="Theme.AppCompat.Light.DarkActionBar">
        ...
        <item name="android:windowActivityTransitions">true</item>
        <item name="android:windowEnterTransition">
        @android:@transition/slide_left</item>
        <item name="android:windowExitTransition">
        @android:@transition/explode</item>
</style>
```

- Les transitions d'activité sont activées avec :

```
<item name="android:windowActivityTransitions"> true </item>
```

- L'attribut<itemame="android:windowEnterTransition">@android:transition/slide_left</item> définit la transition d'entrée ;
- tandis que l'attribut @android:@transition/explode est la transition de sortie, telle que définie par l'attribut <itemame="android:windowExitTransition"> @android:transition/explode</item>.

PARTIE 2

Copyright - Tout droit réservé - OFPPT

Ajout de transitions d'activité via Code

- Les transitions d'activité peuvent également être ajoutées de manière programmatique. La première étape consiste à activer les transitions de contenu de la fenêtre. Pour cela, appeler la fonction suivante dans l'activité avant l'appel à **setContentView**() :

```
window.requestFeature(Window.FEATURE_CONTENT_TRANSITIONS)
```

- Ensuite, ajouter les transactions d'entrée et de sortie avec **window.enterTransition** et **window.exitTransition**, respectivement. À partir du package android.transition, il est possible d'utiliser les transitions intégrées **Explode(), Slide()** et **Fade().** Par exemple, pour utiliser **Explode()** comme transition d'entrée et **Slide()** comme transition de sortie, il suffit d'ajouter le code suivant :

```
window.enterTransition = Explode()
window.exitTransition = Slide()
```

Il ne faut pas oublier d'inclure dans ces appels une vérification de Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP si le SDK minimum supporté par l'application est inférieur à 21.

Démarrer une activité avec une transition d'activité

Une fois les transitions d'activité ajoutées à une activité (soit par XML, soit par codage), il est possible d'activer la transition à l'ouverture de l'activité. Au lieu de l'appel **startActivity(intent)**, il faut passer un bundle avec l'animation de la transition. Pour ce faire, démarrer l'activité avec le code suivant :

```
startActivity(intent,ActivityOptions
.makeSceneTransitionAnimation(this).toBundle())
```

L'argument ActivityOptions.makeSceneTransitionAnimation(this).toBundle() va créer un bundle avec la transition d'entrée et de sortie spécifiée dans l'activité (via XML ou avec du code).

Copyright - Tout droit réservé - OFPPT

Ajouter une transition d'élément partagé

1. Il arrive qu'une application passe d'une activité à une autre, et qu'un élément commun soit présent dans les deux activités. Dans une application de cinéma, par exemple, une activité contenant une liste de films (avec une image miniature) peut ouvrir une nouvelle activité avec les détails du film sélectionné, ainsi qu'une image en taille réelle en haut. L'ajout d'une transition d'élément partagé pour l'image permettra de lier la vignette de l'activité de la liste à l'image de l'activité des détails. La transition d'élément partagé comporte deux parties : la transition d'entrée et la transition de sortie. Ces transitions peuvent être réalisées par le biais de XML ou de code. La première étape consiste à activer une transition de contenu de fenêtre. Pour cela, il suffit d'ajouter le thème de l'activité à themes.xml avec les éléments suivants :

```
<item name="android:windowContentTransitions">true</item>
```

Il est possible de le faire de manière programmatique en appelant la fonction suivante dans votre activité avant l'appel à **setContentView()** :

```
window.requestFeature(Window.FEATURE_CONTENT_TRANSITIONS
)
```

L'attribut **android:windowContentTransitions** avec une valeur **true** et **window.requestFeature(Window.FEATURE_CONTENT_TRANSITIONS)** activera la transition du contenu de la fenêtre.

2. Ensuite, il est possible d'ajouter l'élément partagé enter transition et l'élément partagé exit transition. Si les fichiers **enter_transition.xml** et **exit_transition.xml** se trouvent dans le répertoire **res/transitions**, il suffit d'ajouter l'attribut de style suivant pour ajouter l'élément partagé enter transition :

```
<item name="android:windowSharedElementEnterTransition">
    @transition/enter_transition</item>
```

Il est aussi possible de le faire par code avec les lignes suivantes :

```
val enterTransition = TransitionInfater.from(this)
 .infateTransition(R.transition.enter_transition)
window.sharedElementEnterTransition = enterTransition
```

L'attribut **windowSharedElementEnterTransition** et **window.sharedElementEnterTransition** définissent la transition d'entrée dans le fichier **enter_transition.xml**.

PARTIE 2

Copyright - Tout droit réservé - OFPPT

Ajouter une transition d'élément partagé

3. Pour ajouter la transition de sortie de l'élément partagé, il suffit d'ajouter les attributs de style suivants :

```
<item name="android:windowSharedElementExitTransition">
@transition/exit_transition</item>
```

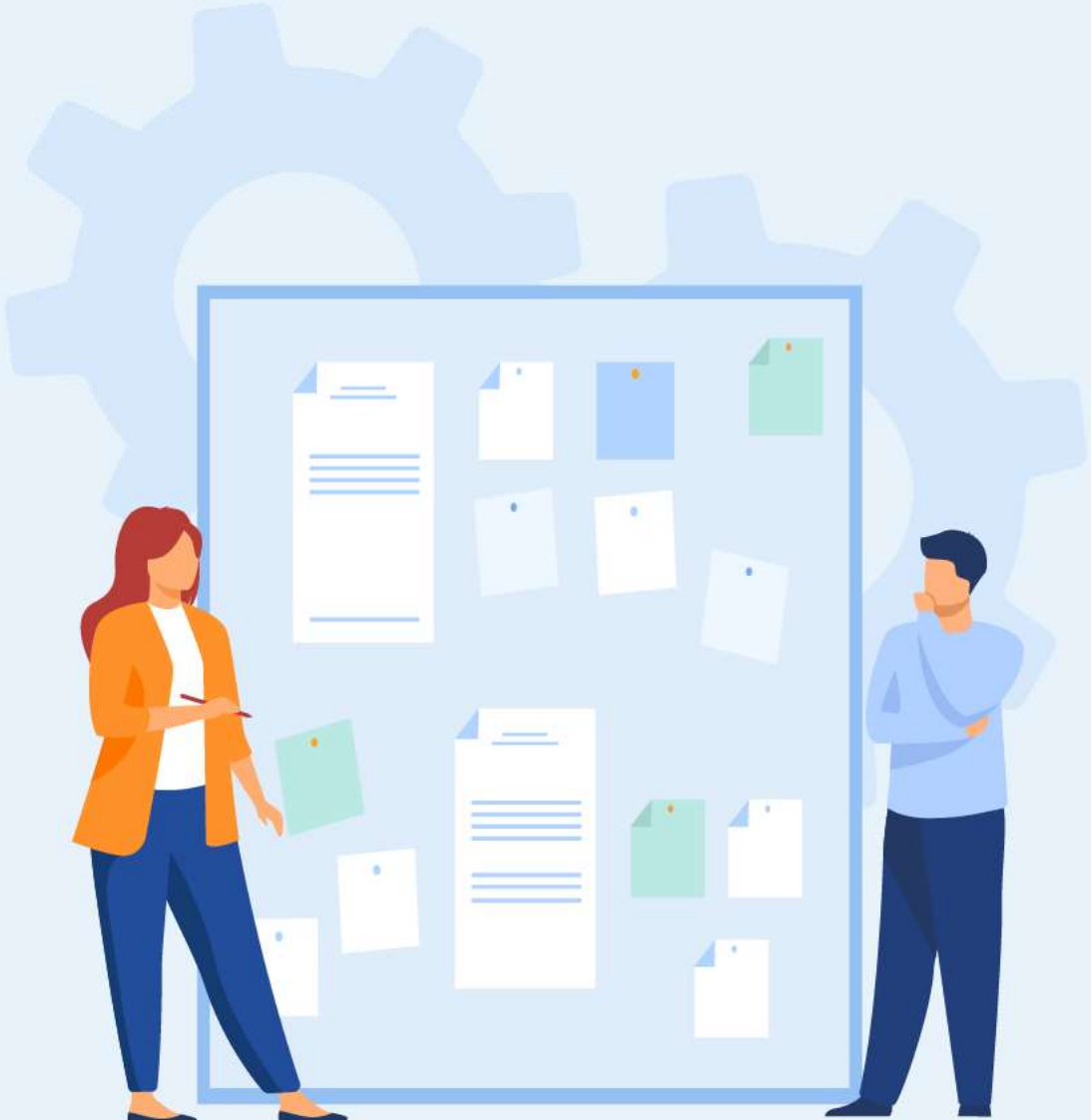
Cela peut être fait de manière programmatique avec les lignes de code suivantes :

```
val exitTransition = TransitionInfater.from(this)
  .infateTransition(R.transition.exit_transition)
window.sharedElementExitTransition = exitTransition
```

Les attributs **windowSharedElementExitTransition**

et **window.sharedElementExitTransition** définissent la transition de sortie dans le fichier **exit_transition.xml**.

Copyright - Tout droit réservé - OFPPT



CHAPITRE 2

Gérer des animations entre les activités

1. Définition d’une animation
2. Utilisation des transitions customisées
3. **Maitrise des animations (Explode, Slide, Fade)**

Exemple : explode animation sur une transition Activities

1. Premièrement, créer un XML de transition pour la transition explode nommé explode.xml dans le dossier res/transition. Comme suit :

```
<?xml version="1.0" encoding="utf-8"?>
<transitionSet
xmlns:android="http://schemas.android.com/apk/res/android">
    <explode
        android:duration="320" />
</transitionSet>
```

2. Ensuite, dans le fichier styles.xml de l'AppTheme, définir android:windowContentTransitions comme true et référencer la transition explode pour **Enter** et **Exit**, comme ci-dessous :

```
<item name="android:windowContentTransitions">true</item>
<item name="android:windowEnterTransition">@transition/explode</item>
<item name="android:windowExitTransition">@transition/explode</item>
```

3. Maintenant, démarrer l'activité :

```
ActivityOptions options = ActivityOptions.makeSceneTransitionAnimation(this);
Intent intent = new Intent(MainActivity.this, SecondActivitiy.class);
startActivity(intent, options.toBundle());
```


Copyright - Tout droit réservé - OFPPT

Exemple : fade animation sur une transition Activities

Il est possible de créer ses propres fichiers d'animation .xml pour faire apparaître une nouvelle activité et faire disparaître l'activité en cours :

fade_in.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator"
    android:fromAlpha="0.0" android:toAlpha="1.0"
    android:duration="500" />
```

fade_out.xml :

```
<?xml version="1.0" encoding="utf-8"?>
<alpha xmlns:android="http://schemas.android.com/apk/res/android"
    android:interpolator="@android:anim/accelerate_interpolator"
    android:fromAlpha="1.0" android:toAlpha="0.0"
    android:fillAfter="true"
    android:duration="500" />
```

Utiliser les animations dans un code comme celui-ci : (dans l'activité)

```
Intent i = new Intent(this, NewlyStartedActivity.class);
startActivity(i);
overridePendingTransition(R.anim.fade_in, R.anim.fade_out);
```

Le code ci-dessus permet de faire disparaître l'activité actuellement active et de faire apparaître l'activité nouvellement lancée, ce qui permet une transition en douceur.

Copyright - Tout droit réservé - OFPPT

Exemple : slide animation sur une transition Activities

1. Utiliser les fichiers xml dans res/anim/ :

Ceci est pour l'animation **de gauche à droite** :

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
  <translate android:fromXDelta="-100%" android:toXDelta="0%"
          android:fromYDelta="0%" android:toYDelta="0%"
          android:duration="700"/>
</set>
```

Ceci est pour l'animation **de droite à gauche** :

```
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="false">
  <translate
    android:fromXDelta="0%" android:toXDelta="100%"
    android:fromYDelta="0%" android:toYDelta="0%"
    android:duration="700" />
</set>
```

2. Dans le code, utiliser l'intention de gauche à droite :

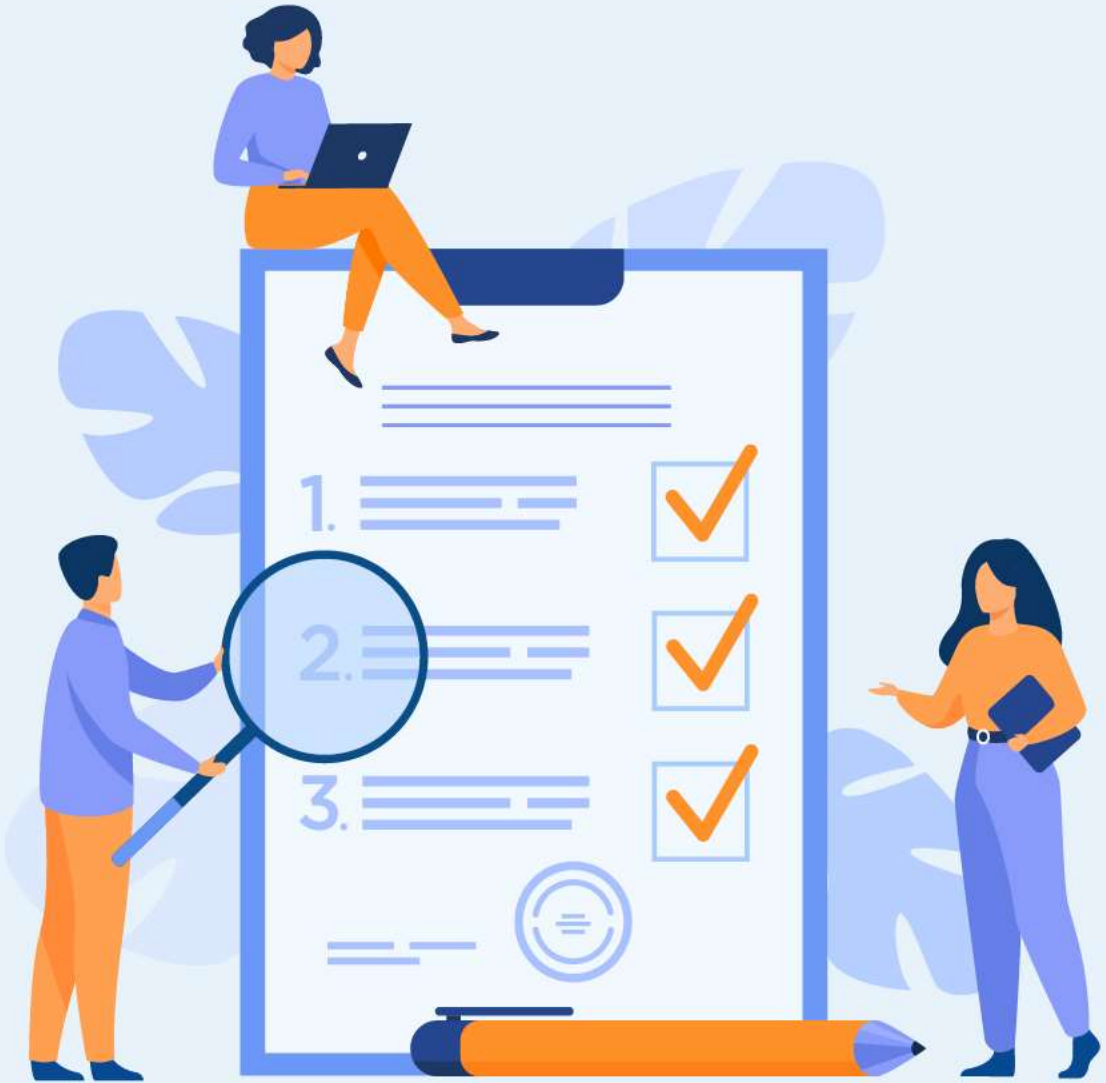
```
this.overridePendingTransition(R.anim.animation_enter,
                R.anim.animation_leave);
```

3. Dans le code, utiliser l'intention de droite à gauche :

```
this.overridePendingTransition(R.anim.animation_leave,
                R.anim.animation_enter);
```

PARTIE 2

Copyright - Tout droit réservé - OFPPT



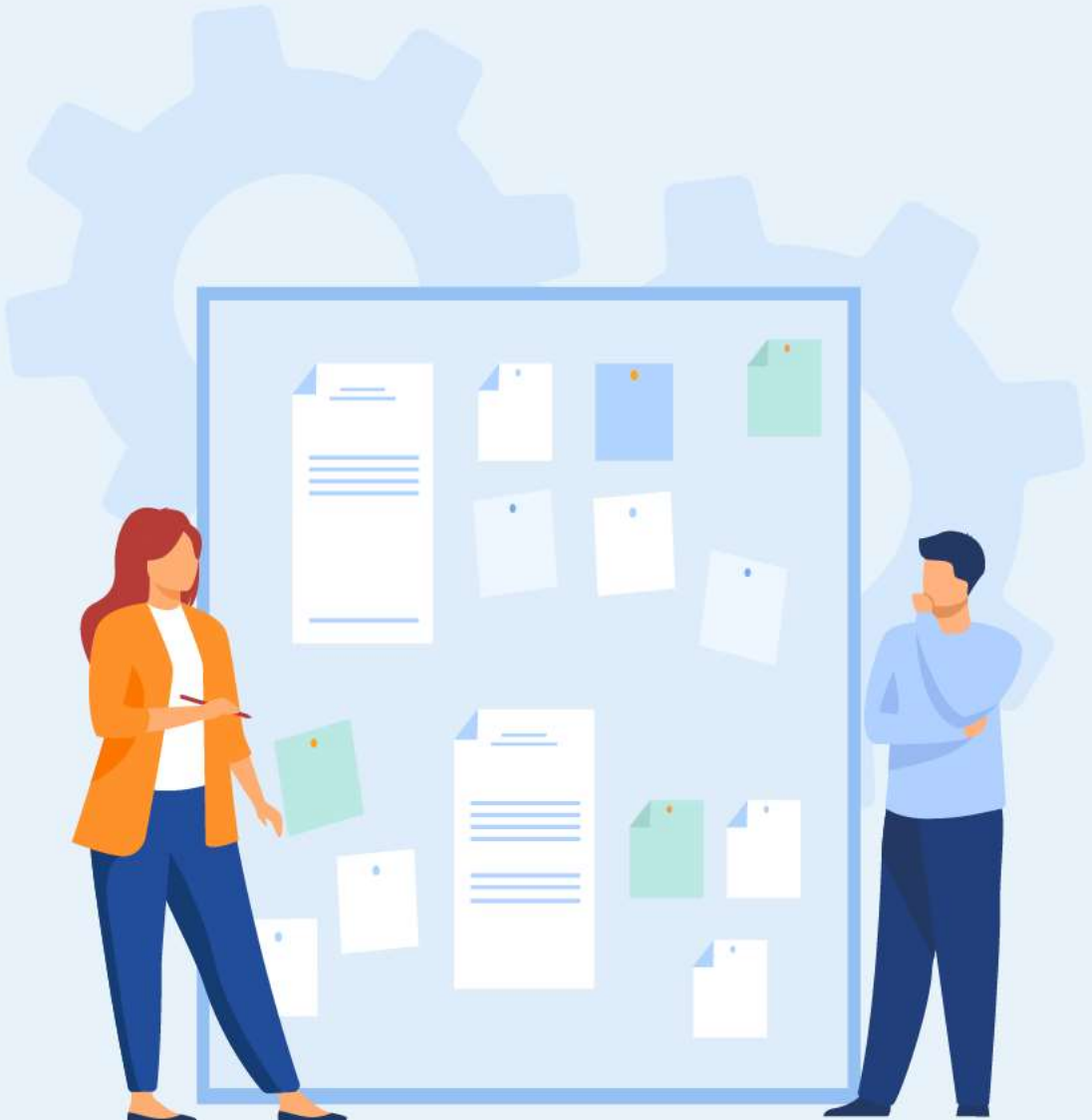
CHAPITRE 3

Créer des effets d’animations

Ce que vous allez apprendre dans ce chapitre :

- Présentation des différents effets d’animation ;
- Création des effets d’animations (Crossfade, Circular reveal, Card flip, Zoom animation).

2 heures



CHAPITRE 3

Créer des effets d’animations

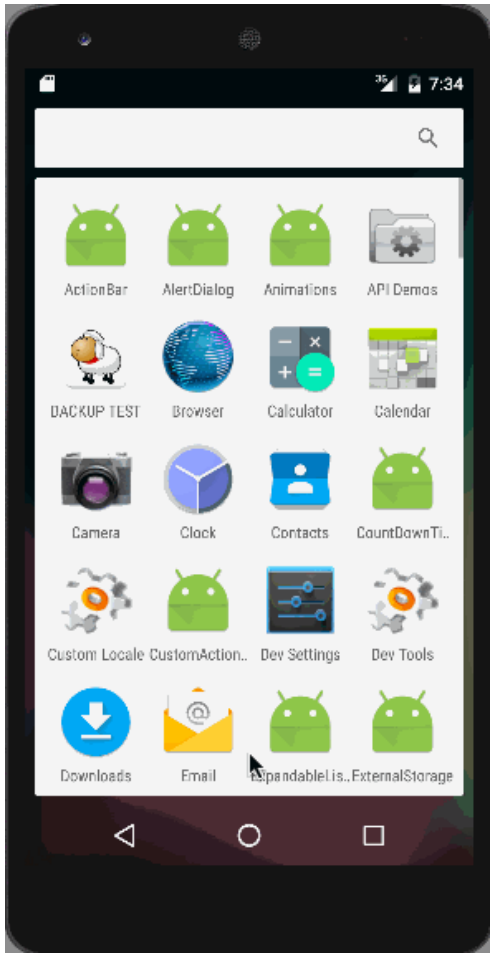
1. **Présentation des différents effets d’animation**
2. Création des effets d’animations (Crossfade, Circular reveal, Card flip, Zoom animation)

Ajouter une transition d'élément partagé

L'animation dans les applications Android est le processus de création de mouvement et de changement de forme. Les méthodes de base de l'animation qui seront abordées sont les suivantes :

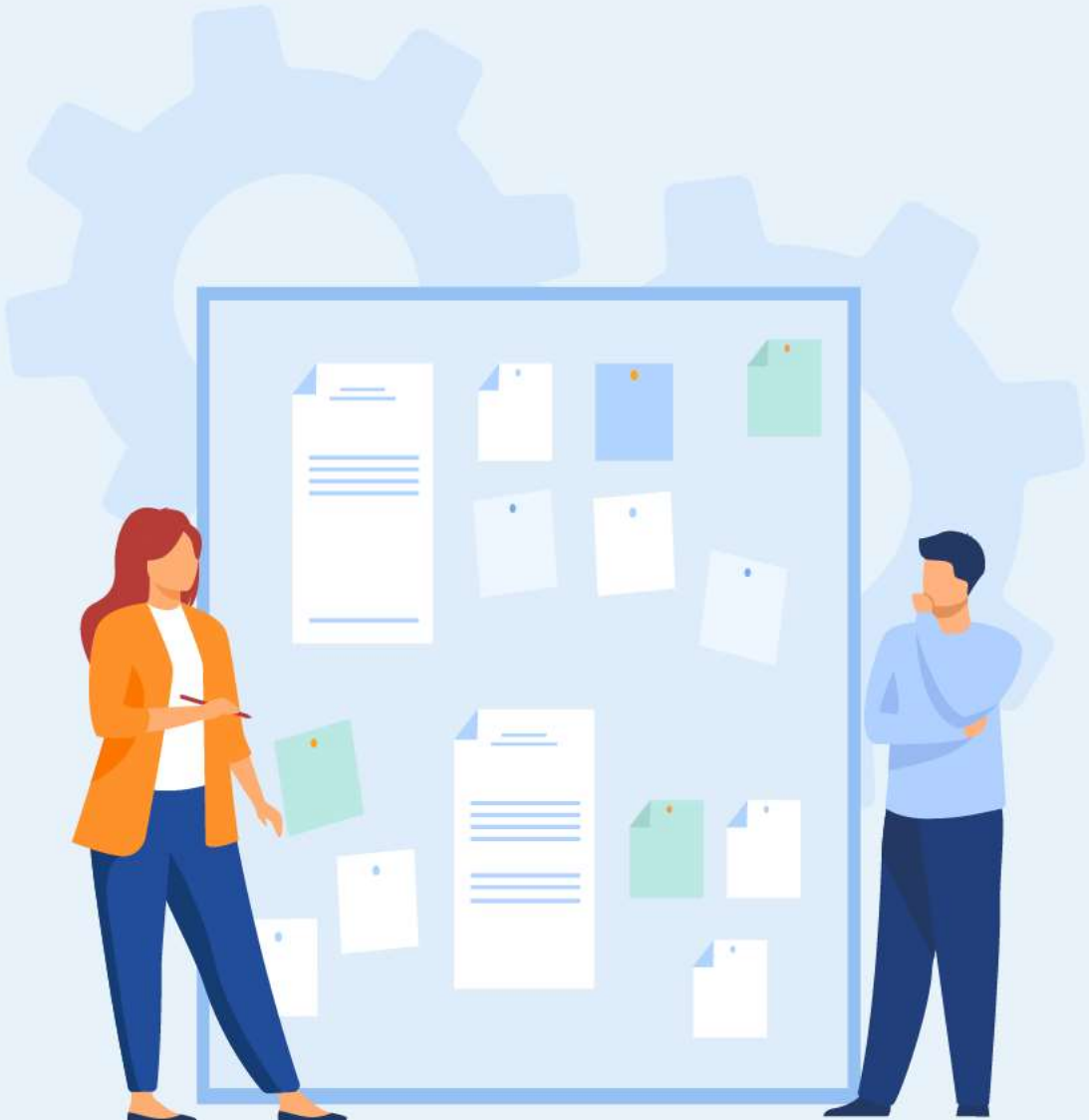
1. Fade In Animation
2. Fade Out Animation
3. Cross Fading Animation
4. Blink Animation
5. Zoom In Animation
6. Zoom Out Animation
7. Rotate Animation
8. Move Animation
9. Slide Up Animation
10. Slide Down Animation
11. Bounce Animation
12. Sequential Animation
13. Together Animation

Démonstration des animations



PARTIE 2

Copyright - Tout droit réservé - OFPPT



CHAPITRE 3

Créer des effets d’animations

1. Présentation des différents effets d’animation
2. **Création des effets d’animations (Crossfade, Circular reveal, Card flip, Zoom animation)**

Exemple d'animation Android XML

```
<?xml version="1.0" encoding="utf-8"?>
<scale xmlns:android="https://schemas.android.com/apk/res/android"
  android:interpolator="@android:anim/accelerate_decelerate_interpolator"
  android:duration="300"
  android:fillAfter="true"
  android:fromXScale="0.0"
  android:fromYScale="0.0"
  android:toXScale="1.0"
  android:toYScale="1.0" />
```

- **android:interpolator** : C'est le taux de changement dans l'animation. Il est possible de définir les interpolateurs en utilisant le temps comme contrainte. Dans le code xml ci-dessus, un interpolateur intégré est assigné ;
- **android:duration** : Durée de l'animation dans laquelle l'animation doit se terminer. Elle est ici de 300ms. C'est généralement la durée idéale pour afficher la transition à l'écran. Le début et la fin de l'animation sont définis à l'aide de :
  android:fromTRANSFORMATION
  android:toTRANSFORMATION
- **TRANSFORMATION** : est la transformation à spécifier. Dans le cas présent, la transformation commence avec une échelle x et y de 0 et se termine avec une échelle x et y de 1 ;
- **android:fillAfter** : propriété qui spécifie si la vue doit être visible ou cachée à la fin de l'animation. Nous l'avons définie comme visible dans le code ci-dessus. Si elle prend la valeur false, l'élément revient à son état précédent après l'animation ;
- **android:startOffset** : Il s'agit du temps d'attente avant le début d'une animation. Cette propriété est principalement utilisée pour exécuter plusieurs animations de manière séquentielle ;
- **android:repeatMode** : Cette propriété est utile pour répéter l'animation ;
- **android:repeatCount** : Cette propriété définit le nombre de répétitions de l'animation. Si nous fixons cette valeur à l'infini, l'animation sera répétée à l'infini.

Copyright - Tout droit réservé - OFPPT

Charger l'animation

```
Animation animation;
animation = AnimationUtils.loadAnimation(getApplicationContext(),
        R.anim.sample_animation);
```

Démarrer l'animation

```
sampleTextView.startAnimation(animation);
```


Copyright - Tout droit réservé - OFPPT

Fade In Animation

```
<set xmlns:android="https://schemas.android.com/apk/res/android"
   android:fillAfter="true" >

   <alpha
      android:duration="1000"
      android:fromAlpha="0.0"
      android:interpolator="@android:anim/accelerate_interpolator"
      android:toAlpha="1.0" />

</set>
```

Ici, alpha fait référence à l'opacité d'un objet. Un objet avec des valeurs alpha plus faibles est plus transparent, tandis qu'un objet avec des valeurs alpha plus élevées est moins transparent, plus opaque. Le fondu en animation n'est rien d'autre que l'augmentation de la valeur alpha de 0 à 1.

Fade Out Animation

```
<set xmlns:android="https://schemas.android.com/apk/res/android"
   android:fillAfter="true" >

   <alpha
      android:duration="1000"
      android:fromAlpha="1.0"
      android:interpolator="@android:anim/accelerate_interpolator"
      android:toAlpha="0.0" />

</set>
```

Une Animation android fade out est exactement l'inverse d'un fade in, où il faut diminuer la valeur alpha de 1 à 0.

Copyright - Tout droit réservé - OFPPT

Cross Fading Animation

Le cross fading consiste à réaliser une animation en fade in sur un TextView pendant que l'autre TextView est en fade out. Cela peut être fait en utilisant fade_in.xml et fade_out.xml sur les deux TextViews. Le code sera présenté dans le fichier MainActivity.java.

Blink Animation

```
<set xmlns:android="https://schemas.android.com/apk/res/android">
   <alpha android:fromAlpha="0.0"
      android:toAlpha="1.0"
      android:interpolator="@android:anim/accelerate_interpolator"
      android:duration="600"
      android:repeatMode="reverse"
      android:repeatCount="infinite"/>
</set>
```

Ici, le fade in et le fade out sont exécutés à l'infini en mode inverse à chaque fois.

Copyright - Tout droit réservé - OFPPT

Zoom In Animation

```
<set xmlns:android="https://schemas.android.com/apk/res/android"
   android:fillAfter="true" >

   <scale
      xmlns:android="https://schemas.android.com/apk/res/android"
      android:duration="1000"
      android:fromXScale="1"
      android:fromYScale="1"
      android:pivotX="50%"
      android:pivotY="50%"
      android:toXScale="3"
      android:toYScale="3" >
   </scale>

</set>
```

Pour le zoom à partir du centre de l'élément, il faut utiliser pivotX="50%" et pivotY="50%".

Zoom Out Animation

```
<set xmlns:android="https://schemas.android.com/apk/res/android"
   android:fillAfter="true" >

   <scale
      xmlns:android="https://schemas.android.com/apk/res/android"
      android:duration="1000"
      android:fromXScale="1.0"
      android:fromYScale="1.0"
      android:pivotX="50%"
      android:pivotY="50%"
      android:toXScale="0.5"
      android:toYScale="0.5" >
   </scale>

</set>
```

Remarquer que android:from et android:to sont opposés dans zoom_in.xml et zoom_out.xml.

Copyright - Tout droit réservé - OFPPT

Rotate Animation

```
<set xmlns:android="https://schemas.android.com/apk/res/android">
   <rotate android:fromDegrees="0"
      android:toDegrees="360"
      android:pivotX="50%"
      android:pivotY="50%"
      android:duration="600"
      android:repeatMode="restart"
      android:repeatCount="infinite"
      android:interpolator="@android:anim/cycle_interpolator"/>

</set>
```

Une balise from/toDegrees est utilisée ici pour spécifier les degrés et un interpolateur cyclique est utilisé.

Move Animation

```
<set
   xmlns:android="https://schemas.android.com/apk/res/android"
   android:interpolator="@android:anim/linear_interpolator"
   android:fillAfter="true">

  <translate
      android:fromXDelta="0%p"
      android:toXDelta="75%p"
      android:duration="800" />
</set>
```


Copyright - Tout droit réservé - OFPPT

Slide Up Animation

```
<set xmlns:android="https://schemas.android.com/apk/res/android"
   android:fillAfter="true" >

   <scale
      android:duration="500"
      android:fromXScale="1.0"
      android:fromYScale="1.0"
      android:interpolator="@android:anim/linear_interpolator"
      android:toXScale="1.0"
      android:toYScale="0.0" />

</set>
```

Il faut définir android:fromYScale="1.0" et android:toYScale="0.0" à l'intérieur de la balise scale.

Slide Down Animation

```
<set xmlns:android="https://schemas.android.com/apk/res/android"
   android:fillAfter="true">

   <scale
      android:duration="500"
      android:fromXScale="1.0"
      android:fromYScale="0.0"
      android:toXScale="1.0"
      android:toYScale="1.0" />

</set>
```


Copyright - Tout droit réservé - OFPPT

Sequential Animation

```
<set xmlns:android="https://schemas.android.com/apk/res/android"
   android:fillAfter="true"
   android:interpolator="@android:anim/linear_interpolator" >


   <!-- Move -->
   <translate
      android:duration="800"
      android:fillAfter="true"
      android:fromXDelta="0%p"
      android:startOffset="300"
      android:toXDelta="75%p" />
   <translate
      android:duration="800"
      android:fillAfter="true"
      android:fromYDelta="0%p"
      android:startOffset="1100"
      android:toYDelta="70%p" />
   <translate
      android:duration="800"
      android:fillAfter="true"
      android:fromXDelta="0%p"
      android:startOffset="1900"
      android:toXDelta="-75%p" />
   <translate
      android:duration="800"
      android:fillAfter="true"
      android:fromYDelta="0%p"
      android:startOffset="2700"
      android:toYDelta="-70%p" />

     <!-- Rotate 360 degrees -->
     <rotate
      android:duration="1000"
      android:fromDegrees="0"
      android:interpolator="@android:anim/cycle_interpolator"
      android:pivotX="50%"
      android:pivotY="50%"
      android:startOffset="3800"
      android:repeatCount="infinite"
      android:repeatMode="restart"
      android:toDegrees="360" />

</set>
```

Ici, un android:startOffset différent est utilisé pour les transitions afin de les garder séquentielles.

Copyright - Tout droit réservé - OFPPT

Bounce Animation

```
<set xmlns:android="https://schemas.android.com/apk/res/android"
   android:fillAfter="true"
   android:interpolator="@android:anim/bounce_interpolator">

   <scale
      android:duration="500"
      android:fromXScale="1.0"
      android:fromYScale="0.0"
      android:toXScale="1.0"
      android:toYScale="1.0" />

</set>
```

Ici, l'interpolateur de rebond est utilisé pour compléter l'animation de manière rebondissante.

Together Animation

```
<set xmlns:android="https://schemas.android.com/apk/res/android"
   android:fillAfter="true"
   android:interpolator="@android:anim/linear_interpolator" >
   <!-- Move -->
   <scale
      xmlns:android="https://schemas.android.com/apk/res/android"
      android:duration="4000"
      android:fromXScale="1"
      android:fromYScale="1"
      android:pivotX="50%"
      android:pivotY="50%"
      android:toXScale="4"
      android:toYScale="4" >
   </scale>
   <!-- Rotate 180 degrees -->
   <rotate
      android:duration="500"
      android:fromDegrees="0"
      android:pivotX="50%"
      android:pivotY="50%"
      android:repeatCount="infinite"
      android:repeatMode="restart"
      android:toDegrees="360" />
</set>
```

Ici, android:startOffset est supprimé pour qu'ils se produisent simultanément.

Copyright - Tout droit réservé - OFPPT