

Création des interfaces simples

Plan

- Introduction
- Les vues
- Les layouts
 - LinearLayout
 - FrameLayout
 - TableLayout
 - GridLayout
 - RelativeLayout
 - ScrollView
- Les ressources

Introduction

- La création d'interface utilisateur sous Android peut s'effectuer de deux manières :
 - La création statique avec XML,
 - Constitué de plusieurs éléments comme : boutons, texte, zone d'édition
 - La création dynamique avec java
 - Activité qui crée et affiche le contenu statique
 - Interactions utilisateur
 - Traitements à effectuer

Les vues

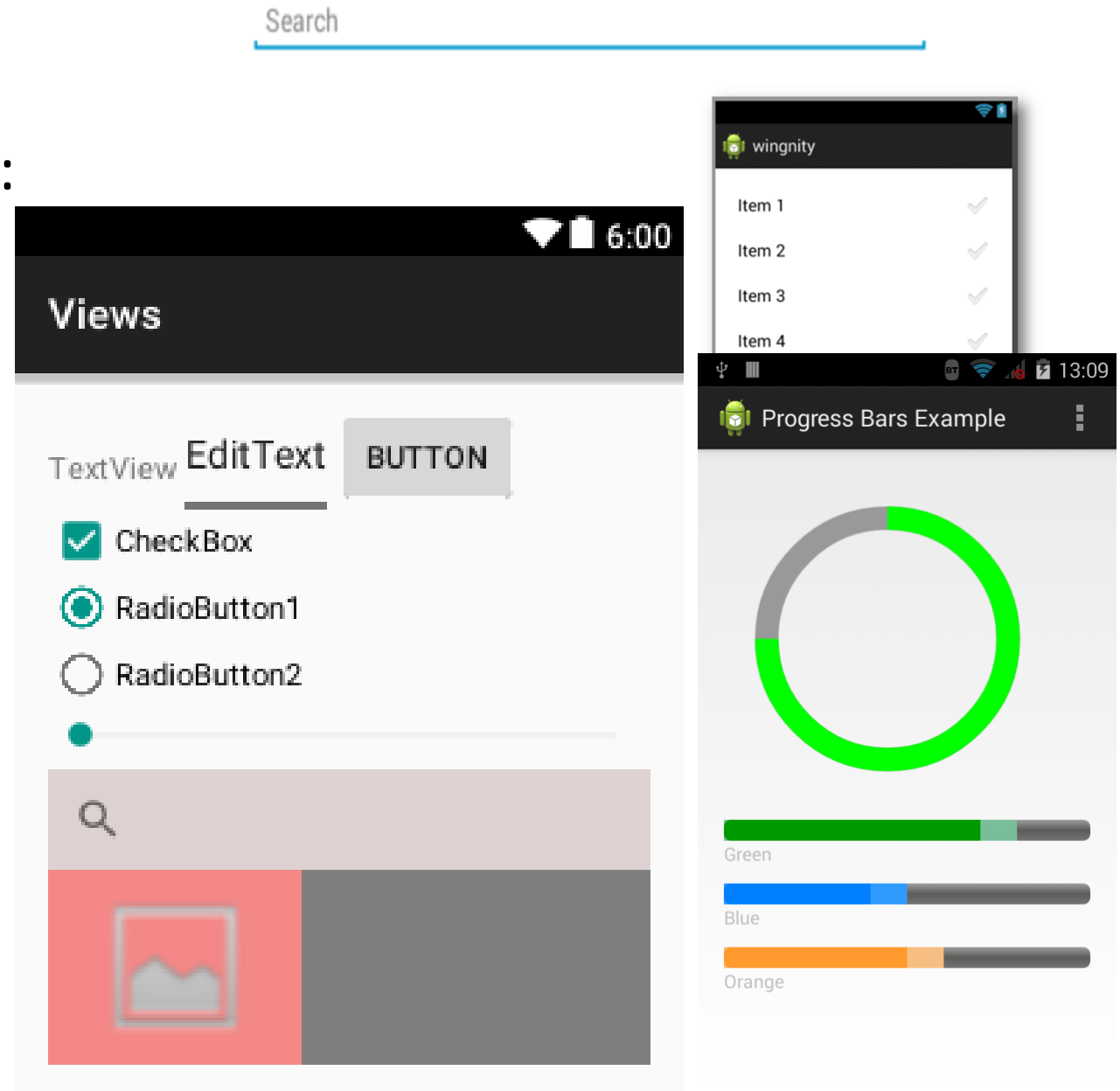
- Les vues Android représentent :

- Les éléments d'un formulaire

- TextView
 - EditText
 - Button
 - Checkbox
 - RadioButton
 - CheckedTextView
 - ProgressBar
 - SeekBar
 - SearchView
 - SearchTextView

- Les éléments multimédia

- ImageView
 - ImageButton
 - VideoView



Les vues

- Les vues Android représentent :
 - Les éléments d'un formulaire
 - TextView : affiche une chaîne
 - EditText : permet la saisie d'une chaîne
 - Button : bouton cliquable
 - Checkbox : case à cocher
 - RadioButton : bouton radio regroupable dans un RadioGroup
 - CheckedTextView : chaîne cochable (implante Checkable)
 - ProgressBar : barre de progression (horizontale, circulaire), variante avec étoiles de notation avec RatingBar
 - SeekBar : barre de réglage
 - SearchView : champ de recherche avec proposition de suggestions
 - Les éléments multimédia
 - ImageView : affichage d'une ressource image
 - ImageButton : bouton avec image
 - VideoView : affichage contrôlable de vidéo

Les vues

- Tous les éléments basiques d'une vue héritent de la classe View
- La construction des interfaces graphiques qui sont composées de plusieurs vues se fait dans un fichier XML

Vues dans un fichier XML

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

```
<Button
    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button 1" />
```

```
<Button
    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button 2" />
```

```
<Button
    android:id="@+id/button3"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Button 3"
    android:layout_weight="1"/>
```

```
</LinearLayout>
```

Déclarer des identifiants

- Un identifiant correspond à un nom unique affecté à un élément d'une vue
- Grâce à cet identifiant, on peut mettre en place les interactions et les traitements pour l'élément possédant cet identifiant
- Pour associer un identifiant à un élément d'une, il faut utiliser l'attribut suivant :
 - ***android:id="@+id/nom_identifiant"***
 - **android:id** : Nom de l'attribut
 - **@+** : Indique la déclaration d'un nouvel identifiant
 - **id** : indique la catégorie de l'identifiant
 - **nom_identifiant** : correspond à l'identifiant de l'élément
- La syntaxe permettant d'accéder à un identifiant depuis un programme java
 - **R.id.nom_identifiant**
- Accéder à un identifiant depuis un fichier XML
 - **@id/nom_identifiant**

Identifiant des vues

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

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 2" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 3"
        android:layout_weight="1"/>

</LinearLayout>
```

Accéder à un identifiant depuis un fichier java

```
public class MainActivity extends Activity {  
  
    Button button;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        addListenerOnButton();  
    }  
  
    public void addListenerOnButton() {  
        button = (Button) findViewById(R.id.button1);  
        button.setOnClickListener(new OnClickListener() {  
            @Override  
            public void onClick(View arg0) {  
                ...  
            }  
        });  
    }  
}
```

Spécifier la taille des éléments

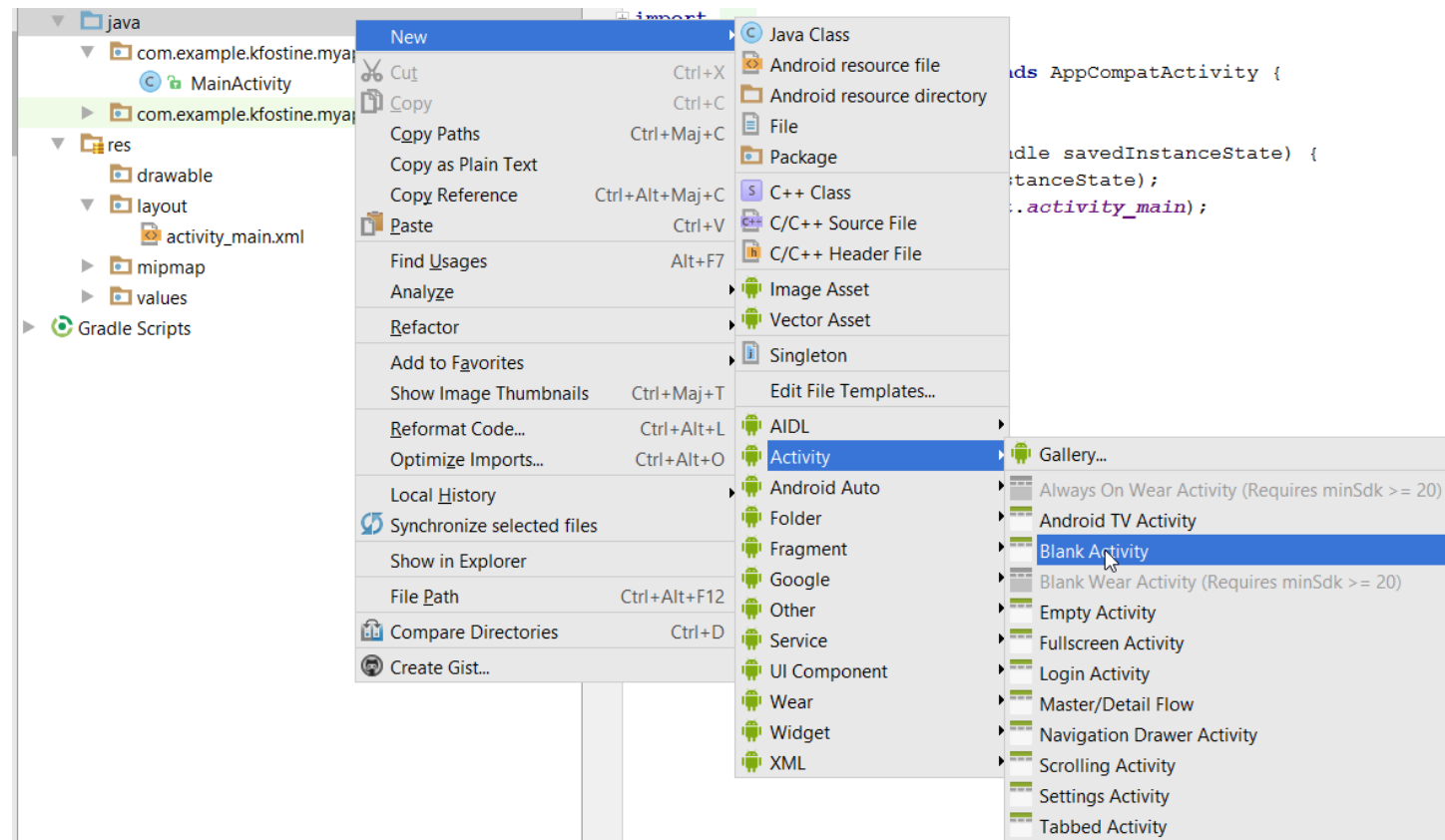
- À chaque déclaration d'élément d'une vue, il faut spécifier sa largeur (android:layout_width) et sa hauteur (android:layout_height)
- On peut spécifier ces valeurs de différentes manières :
 - match_parent : la taille de l'élément est égale à celle de l'élément parent
 - Bouton ayant la même largeur que son conteneur
 - wrap_content : la taille de l'élément est égale à celle de son contenu
 - Bouton ayant la même largeur que son contenu additionné des espacements internes(padding)
 - En spécifiant une valeur : Définir la taille d'un élément à l'aide d'une valeur fixe
 - android:layout_width="15dp"
 - Les valeurs sont à spécifier en dp (density-indépendent pixel et non en px)
 - Les tailles spécifiées en dp conservent les mêmes proportions quelque soient la densité de l'écran

Combiner avec les activités

- Une fois la partie statique (xml) d'une interface déclarée, il faut créer une activité (classe java)
- Cette classe doit hériter de la classe Activity
- Elle doit redéfinir au minimum la méthode onCreate
- Elle doit lier l'activité à l'interface (xml) via la méthode setContentView

Créer une nouvelle activité

- Clic droit sur le dossier java du projet
- New/Activity/Blank Activity



Création d'une activité

New Android Activity

Customize the Activity

←

+

Blank Activity

Creates a new blank activity with an app bar.

Activity Name:

Main2Activity

Layout Name:

activity_main2

Title:

Main2Activity

☐ Launcher Activity

☐ Use a Fragment

Hierarchical Parent:

Package name:

The name of the activity class to create

Activity Name is not a valid class name

Previous

Next

Cancel

Finish

Contenu minimum d'une activité

```
public class MyAndroidAppActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```

Remarquer l'utilisation du fichier R.java pour récupérer le layout correspondant à l'interface

Le fichier manifest

- L'activité doit être déclarée dans le manifeste de l'application

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

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="My Application"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Activité affichée
au lancement de
l'application

Les layouts

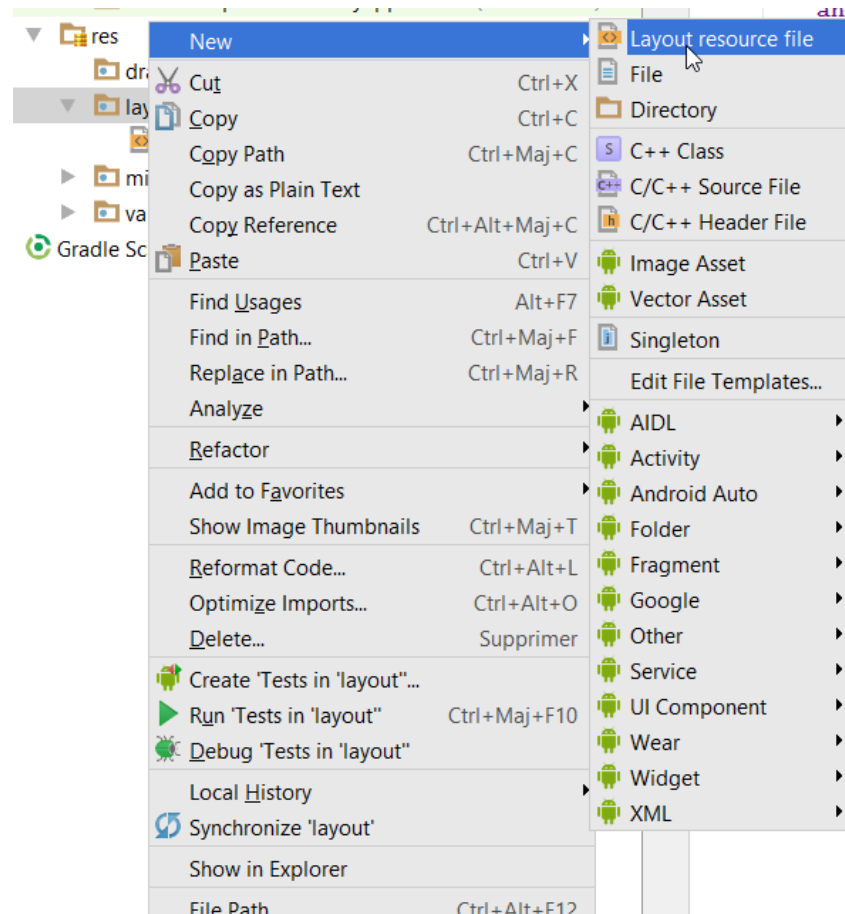
- Les layouts facilitent l'organisation des différents éléments qui composent une interface utilisateur.
- Ils servent de conteneur aux éléments d'une vue
- Tous les layouts Android héritent de la classe ViewGroup
- La classe ViewGroup héritent de la classe View
- Les principaux Layout sont :
 - LinearLayout
 - FrameLayout
 - TableLayout
 - GridLayout
 - RelativeLayout
 - ScrollView
 - ...

LinearLayout

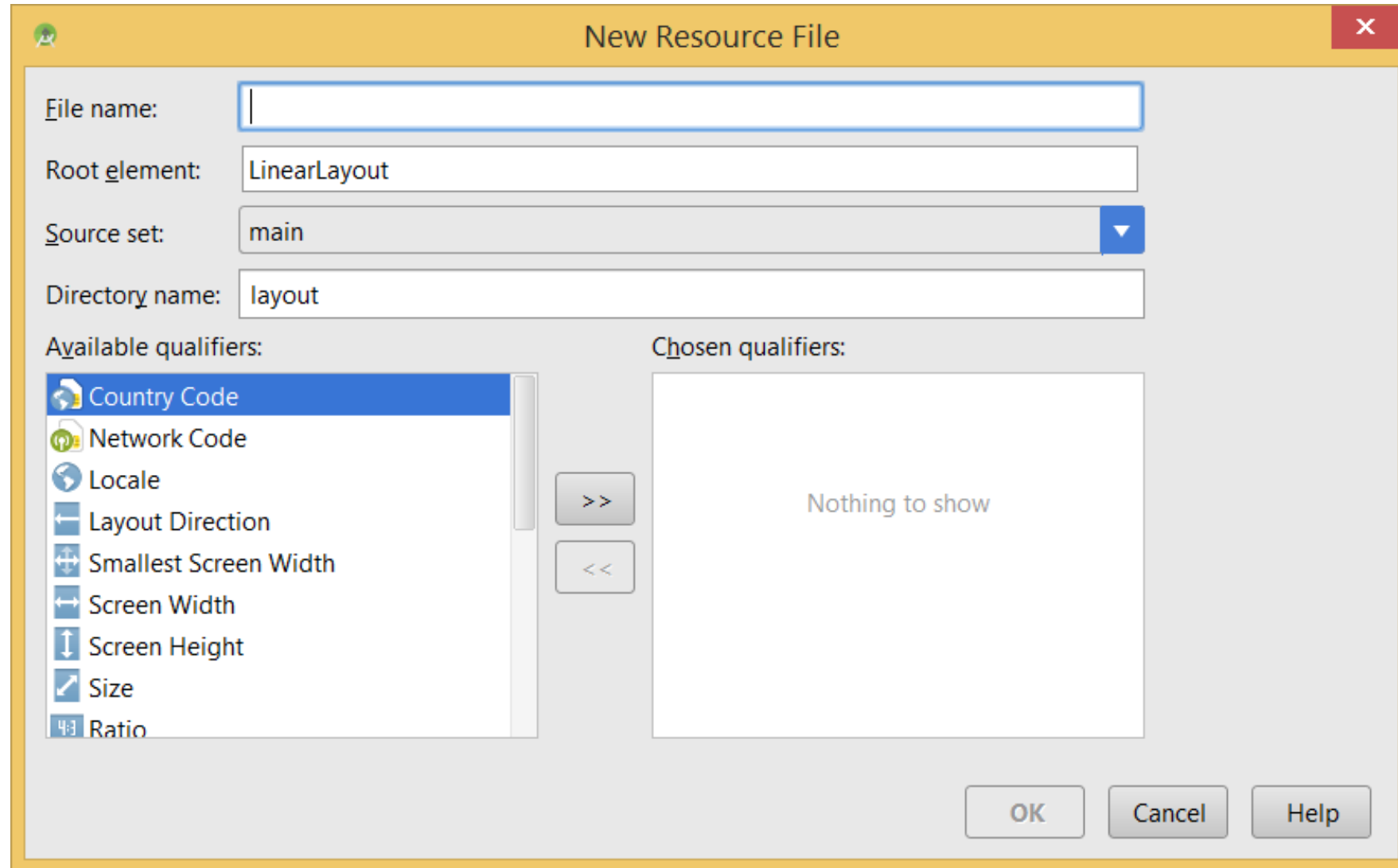
- Le LinearLayout permet d'aligner les vues dans l'ordre des déclarations dans une direction verticale ou horizontale
- On peut déclarer les attributs suivants :
 - L'orientation du layout
 - `android:orientation="vertical"` > ou `android:orientation="horizontal"` >
 - La gravité des éléments :
 - `android:layout_gravity="right"` : spécifie le positionnement d'un élément dans son conteneur
 - `android:gravity="right"` : spécifie le positionnement du contenu d'un élément (Exemple le positionnement du contenu d'un text dans un bouton)
 - Le poids des éléments : Indique à un élément l'espace qu'il peut occuper. Plus le poids d'un élément est grand, plus il peut s'allonger et occuper l'espace disponible.
 - `android:layout_weight="1"` :
 - La taille de la zone à allonger doit être définie à 0dp.
 - Le poids par défaut est 0.

Création d'un linearLayout

- Clic droit sur le dossier res/layout du projet
- New/Layout resource File



Création d'un LinearLayout

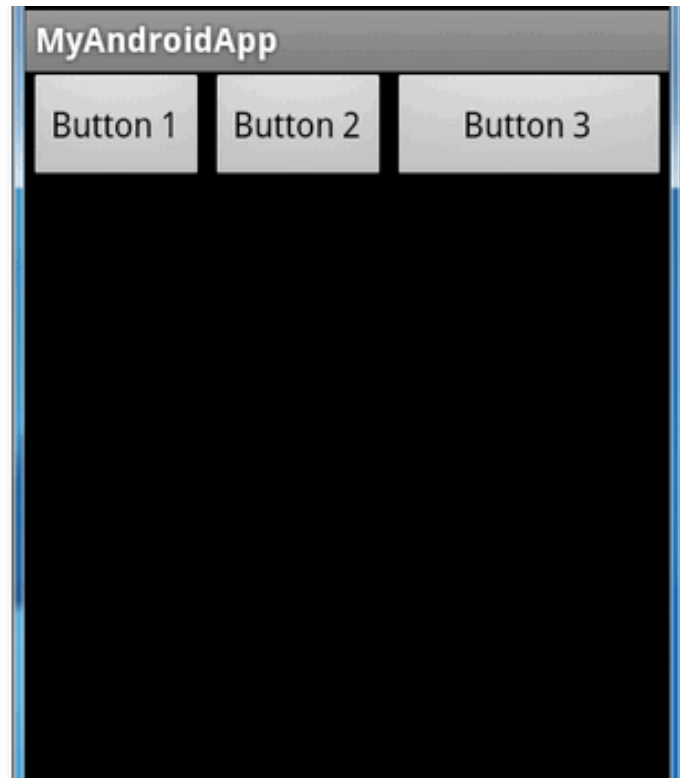


The screenshot shows the 'New Resource File' dialog box with the following fields and options:

- File name:** An empty text input field.
- Root element:** A text input field containing 'LinearLayout'.
- Source set:** A dropdown menu showing 'main'.
- Directory name:** A text input field containing 'layout'.
- Available qualifiers:** A list box containing the following items:
 - Country Code
 - Network Code
 - Locale
 - Layout Direction
 - Smallest Screen Width
 - Screen Width
 - Screen Height
 - Size
 - Ratio
- Chosen qualifiers:** An empty box with the text 'Nothing to show'.
- Navigation buttons:** '>>' and '<<' buttons between the available and chosen qualifiers lists.
- Bottom buttons:** 'OK', 'Cancel', and 'Help' buttons.

LinearLayout

- Exercice 1
 - Écrire une application Android qui affiche l'interface graphique suivante (à utiliser seulement l'onglet text de Android Studio pour construire l'interface – Il est interdit de toucher à l'onglet Design)



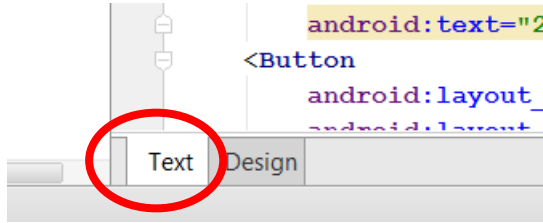
Solution

- Créer un nouveau projet Android (Empty Activity)
- Créer le fichier xml (layout : LinearLayout) nommé main
 - Clic droit sur le dossier res/layout
 - New/Layout resource File
 - File name : main
 - Root Element : LinearLayout
- Écrire le code de la diapo 23 pour éditer le fichier main.xml
- Modifier l'activité MainActivity.java en écrivant le code de la diapo 24

Solution

File : res/layout/main.xml

À basculer de l'onglet
« Design » à l'onglet « Text »



Écrire ce code à l'intérieur
de l'élément LinearLayout



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

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 2" />

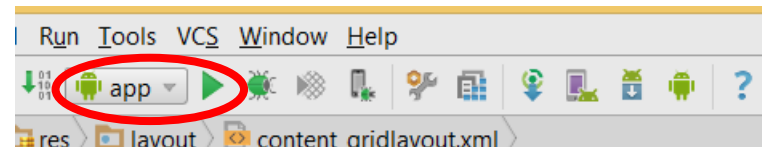
    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 3"
        android:layout_weight="1"/>

</LinearLayout>
```

Solution : Fichier : java/MainActivity.java

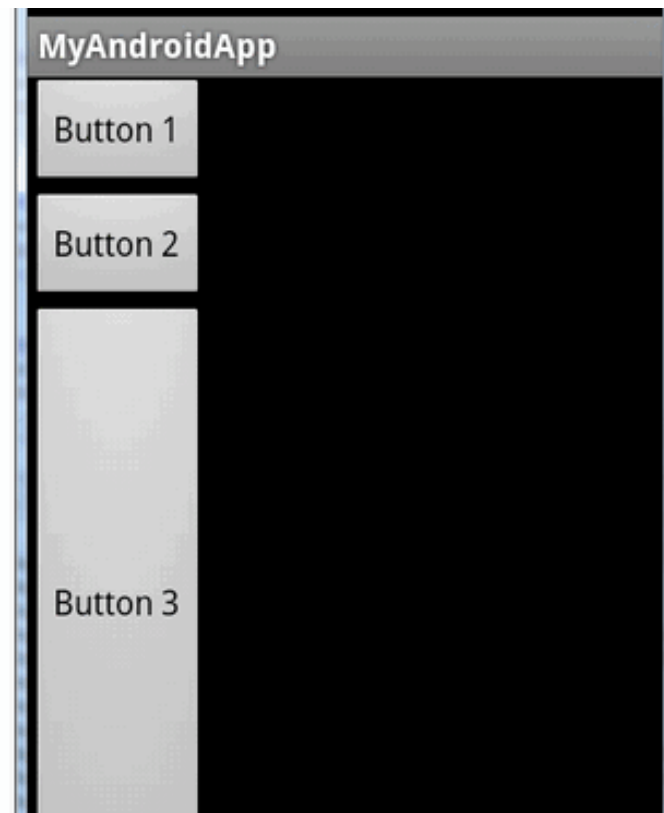
```
public class MyAndroidAppActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```

Exécuter le projet et
choisir un émulateur



LinearLayout

- Exercice 2
 - Écrire une application Android qui affiche l'interface graphique suivante (à utiliser seulement l'onglet text de Android Studio pour construire l'interface – Il est interdit de toucher à l'onglet Design)

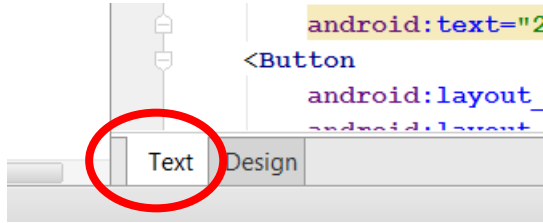


Solution

- Créer un nouveau projet Android (Empty Activity)
- Créer le fichier xml (layout : LinearLayout) nommé main
 - Clic droit sur le dossier res/layout
 - New/Layout resource File
 - File name : main
 - Root Element : LinearLayout
- Écrire le code de la diapo 27 pour éditer le fichier main.xml
- Modifier l'activité MainActivity.java en écrivant le code de la diapo 28

Solution

À basculer de l'onglet
« Design » à l'onglet « Text »



Écrire ce code à l'intérieur
de l'élément LinearLayout



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

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 1" />

    <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 2" />

    <Button
        android:id="@+id/button3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Button 3"
        android:layout_weight="1"/>

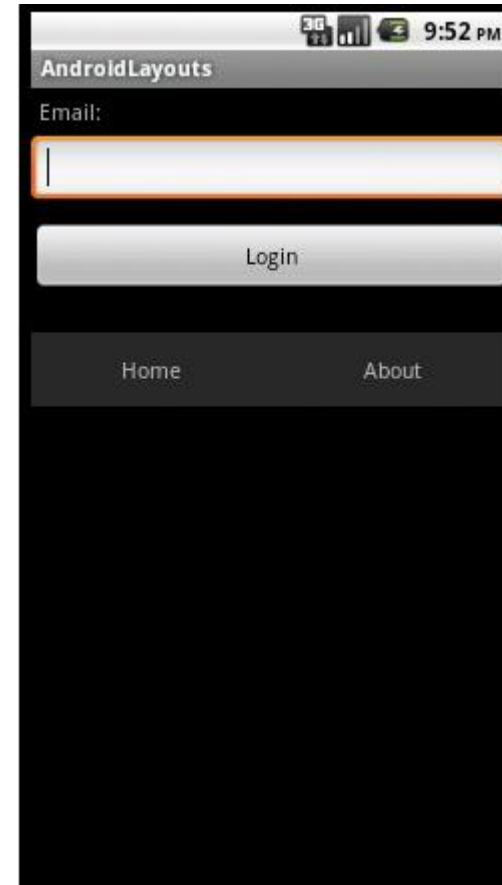
</LinearLayout>
```

Solution : Fichier : java/MainActivity.java

```
public class MyAndroidAppActivity extends Activity {  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```

LinearLayout

- Exercice 3
 - Écrire une application Android qui affiche l'interface graphique suivante (à utiliser seulement l'onglet text de Android Studio pour construire l'interface – Il est interdit de toucher à l'onglet Design)



Indices de solution

- Créer un nouveau projet Android
- Créer le fichier xml (layout : LinearLayout) nommé main
 - Clic droit sur le dossier res/layout
 - New/Layout resource File
 - File name : main
 - Root Element : LinearLayout
- Écrire le code pour éditer le fichier main.xml
 - Il y a un LinearLayout à l'intérieur d'un autre LinearLayout pour les deux derniers boutons
- Modifier l'activité MainActivity.java pour lier l'activité au fichier main.xml

FrameLayout

- Le FrameLayout est le conteneur le plus simple
- Il représente un espace qui affiche l'objet de votre choix
- Un élément ajouté à un FrameLayout se positionne en haut à gauche du layout. Cette position peut-être changé à l'aide de l'attribut `android:gravity`.
- Vous avez la possibilité d'ajouter plusieurs éléments dans un même FrameLayout et de modifier la visibilité de ces éléments pour afficher ou cacher plusieurs éléments au même emplacement

Exercice 4

- Écrire une application Android qui affiche l'écran de Login(fig 1) pour demander l'utilisateur à saisir le nom d'utilisateur et mot de passe. Si le nom d'utilisateur est « user » et le mot de passe est « pass » en cliquant sur le bouton « Login », le programme doit afficher l'écran correspondant à l'image de chien (fig 2)
- Il faut utiliser Frame Layout pour garder les deux vues affichées
- À utiliser seulement l'onglet text de Android Studio pour construire l'interface – Il est interdit de toucher à l'onglet Design

Fig 1

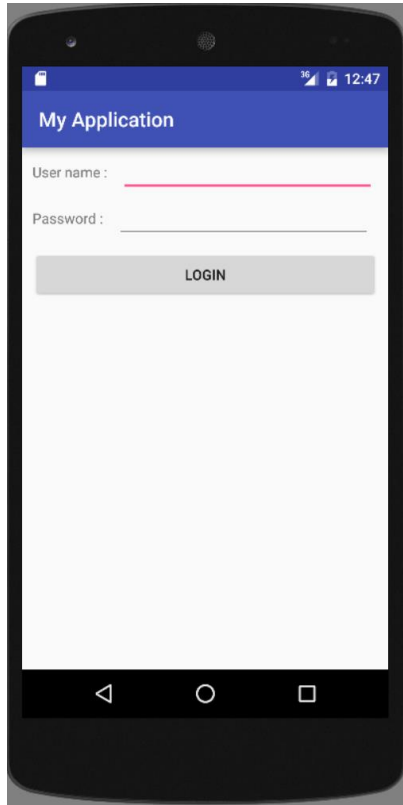
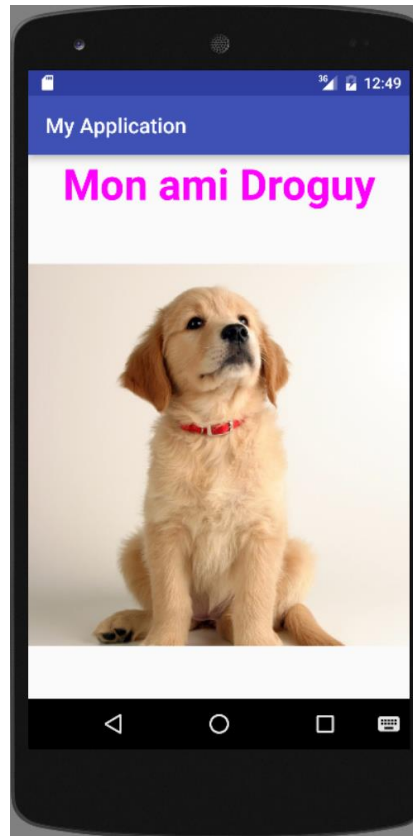


Fig. 2



Solution

- Créer un nouveau projet Android (Empty Activity)
- Copier l'image « Chien.jpg » disponible dans le dossier public/Fostiné Kerlyne/420-254-MO/Semaine 3/ dans le dossier res/drawable du projet
- Créer le fichier xml (layout : FrameLayout) nommé activity_main1
 - Clic droit sur le dossier res/layout
 - New/Layout resource File
 - File name : activity_main1
 - Root Element : FrameLayout
- Écrire le code des diapos 34 à 36 pour éditer le fichier activity_main1.xml
- Modifier l'activité MainActivity.java en écrivant le code de la diapo 37

Solution – Fichier : activity_main1

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

    <LinearLayout
        android:id="@+id/idLayoutLogin"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:padding="10dp"
                android:text="User name : "/>
            <EditText
                android:id="@+id/idTextUserName"
                android:layout_width="250dp"
                android:layout_height="wrap_content" />
        </LinearLayout>
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
```

Zone d'écran
de login



Zone d'écran
de login

Solution – Fichier : activity_main1

```
        android:padding="10dp"
        android:text="Password : "/>
<EditText
    android:id="@+id/idTextPassword"
    android:layout_width="250dp"
    android:layout_height="wrap_content"
    android:inputType="textPassword"/>
</LinearLayout>
<Button
    android:id="@+id/idButtonLogin"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="10dp"
    android:text="Login"/>
```

```
</LinearLayout>
```

```
<LinearLayout
```

```
    android:id="@+id/idLayoutImage"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:visibility="gone">
```

```
<TextView
```

```
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Mon ami Droguy"
    android:fontFamily="Arial"
    android:textColor="#FF00FF"
    android:textSize="40dp"
    android:textStyle="bold"
```

Zone d'écran
d'image de chien

Solution – Fichier : activity_main1

Zone d'écran
d'image de chien

```
        android:layout_gravity="center"/>
    <ImageView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1"
        android:src="@drawable/chien"/>
    </LinearLayout>
</FrameLayout>
```

Solution – Fichier : MainActivity.java

```
public class MainActivity extends AppCompatActivity {

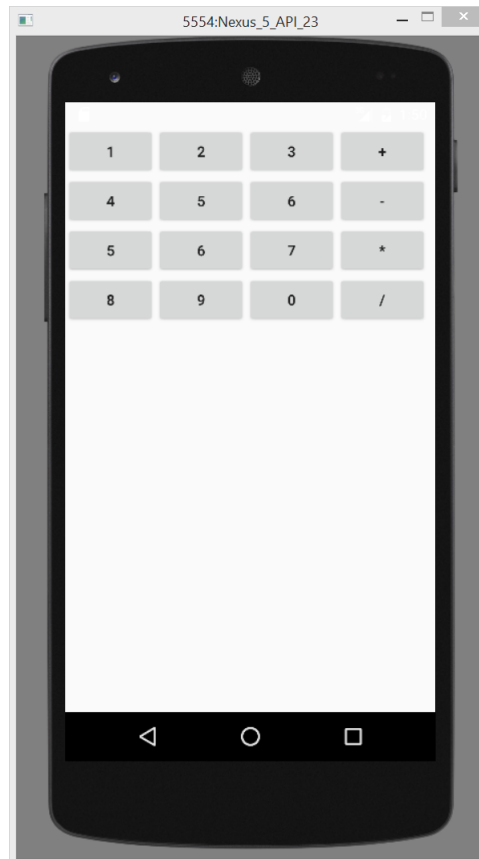
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main1);
        Button btnLogin = (Button) findViewById(R.id.idButtonLogin);
        btnLogin.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                TextView txtUserName = (TextView) findViewById(R.id.idTextUserName);
                TextView txtPassword = (TextView) findViewById(R.id.idTextPassword);
                String userName = txtUserName.getText().toString();
                String password = txtPassword.getText().toString();
                if(userName.equalsIgnoreCase("user") && password.equalsIgnoreCase("pass")){
                    LinearLayout layoutLogin = (LinearLayout) findViewById(R.id.idLayoutLogin);
                    LinearLayout layoutImage = (LinearLayout) findViewById(R.id.idLayoutImage);
                    layoutLogin.setVisibility(View.GONE);
                    layoutImage.setVisibility(View.VISIBLE);
                }
            }
        });
    }
}
```

TableLayout

- Le TableLayout positionne tous ses fils en ligne et colonne à la manière d'un tableau
- TableLayout hérite de LinearLayout
- Il permet d'avoir des cellules vides
- Il ne permet pas d'étendre une cellule sur plusieurs lignes ou plusieurs colonnes
- Pour créer des lignes dans un TableLayout, on utilise TableRow
- Chaque élément de type TableRow représente un élément avec 0 ou plusieurs colonnes
- L'élément « Space » permet de laisser une case vide
 - `<Space/>`

Exercice 5

- Écrire une application Android qui utilise le layout `TableLayout` pour afficher l'écran suivant:
- À utiliser seulement l'onglet text de Android Studio pour construire l'interface – Il est interdit de toucher à l'onglet Design



Solution

- Créer un nouveau projet Android (Empty Activity)
- Créer le fichier xml (layout : TableLayout) nommé activity_main1
 - Clic droit sur le dossier res/layout
 - New/Layout resource File
 - File name : content_calculatrice_tablelayout
 - Root Element : TableLayout
 - Écrire le code des diapos 41-42 pour éditer le fichier content_calculatrice_tablelayout
- Modifier l'activité MainActivity.java en écrivant le code de la diapo 43

Solution – Fichier : content calculatrice tablelayout

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

```
    <TableRow>  
        <Button  
            android:padding="3dp"  
            android:text="1"/>  
        <Button  
            android:padding="3dp"  
            android:text="2"/>  
        <Button  
            android:padding="3dp"  
            android:text="3"/>  
        <Button  
            android:padding="3dp"  
            android:text="+"/>  
    </TableRow>
```

 Ligne 1

```
    <TableRow>  
        <Button  
            android:padding="3dp"  
            android:text="4"/>  
        <Button  
            android:padding="3dp"  
            android:text="5"/>  
        <Button  
            android:padding="3dp"  
            android:text="6"/>  
        <Button  
            android:padding="3dp"  
            android:text="-"/>  
    </TableRow>
```

 Ligne 2

Solution – Fichier : content_calculatrice_tablelayout

```
<TableRow>
  <Button
    android:padding="3dp"
    android:text="5"/>
  <Button
    android:padding="3dp"
    android:text="6"/>
  <Button
    android:padding="3dp"
    android:text="7"/>
  <Button
    android:padding="3dp"
    android:text="*"/>
</TableRow>
```



Ligne 3

```
<TableRow>
  <Button
    android:padding="3dp"
    android:text="8"/>
  <Button
    android:padding="3dp"
    android:text="9"/>
  <Button
    android:padding="3dp"
    android:text="0"/>
  <Button
    android:padding="3dp"
    android:text="/" />
</TableRow>
```



Ligne 4

```
</TableLayout>
```

Solution – Fichier : MainActivity.java

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.content_calculatrice_tablelayout);  
    }  
}
```

Exercice 5 avec une case vide



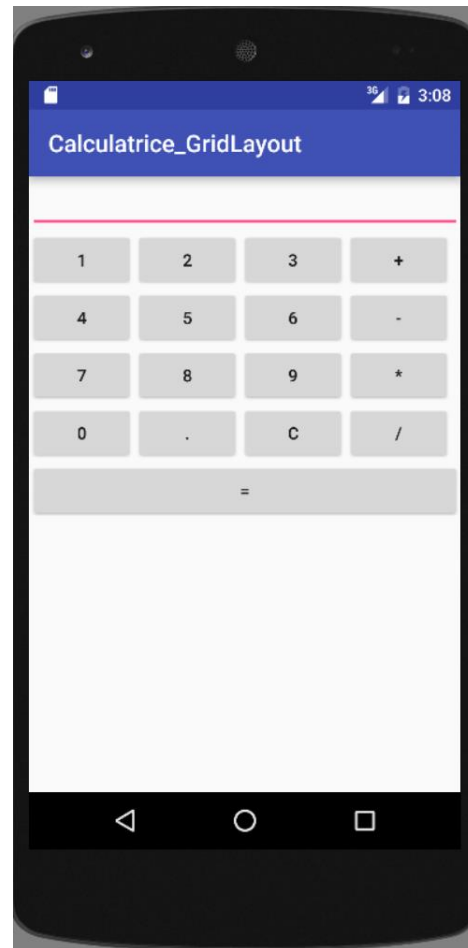
```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">
    <TableRow>
        <Button
            android:padding="3dp"
            android:text="1"/>
        <Button
            android:padding="3dp"
            android:text="2"/>
        <Space/>
        <Button
            android:padding="3dp"
            android:text="+"/>
    </TableRow>
    <TableRow>
        <Button
            android:padding="3dp"
            android:text="4"/>
        <Button
            android:padding="3dp"
            android:text="5"/>
        <Button
```

GridLayout

- Le GridLayout permet de diviser l'écran en ligne et en colonne
- Il permet de laisser des cases vides
- Il permet d'étendre une ligne ou une colonne sur plusieurs cases
- L'élément « Space » permet de laisser une case vide
 - `<Space/>`

Exercice 6

- Écrire une application Android qui utilise le layout GridLayout pour afficher l'écran suivant:
- À utiliser seulement l'onglet text de Android Studio pour construire l'interface – Il est interdit de toucher à l'onglet Design



Solution

- Créer un nouveau projet Android (Empty Activity)
- Créer le fichier xml (layout : GridLayout) nommé activity_main1
 - Clic droit sur le dossier res/layout
 - New/Layout resource File
 - File name : activity_main1
 - Root Element : GridLayout
- Écrire le code des diapos 48 à 51 pour éditer le fichier activity_main1.xml
- Modifier l'activité MainActivity.java en écrivant le code de la diapo 52

Solution – Fichier : content_gridlayout.xml

```
<?xml version="1.0" encoding="utf-8"?>
<GridLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent"
    android:columnCount="4"
    android:rowCount="6">
    <EditText
        android:layout_column="0"
        android:layout_row="0"
        android:layout_columnSpan="4"
        android:layout_gravity="fill_horizontal"/>
    <Button
        android:layout_column="0"
        android:layout_row="1"
        android:padding="3dp"
        android:text="1"/>
    <Button
        android:layout_column="1"
        android:layout_row="1"
        android:padding="3dp"
        android:text="2"/>
    <Button
        android:layout_column="2"
        android:layout_row="1"
        android:padding="3dp"
        android:text="3"/>
    <Button
        android:layout_column="3"
        android:layout_row="1"
        android:padding="3dp"
        android:text="+"/>
```



Ligne 1

Ligne 2

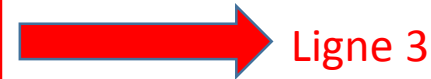
Solution – Fichier : content_gridlayout.xml

```
<Button  
    android:layout_column="0"  
    android:layout_row="2"  
    android:padding="3dp"  
    android:text="4"/>
```

```
<Button  
    android:layout_column="1"  
    android:layout_row="2"  
    android:padding="3dp"  
    android:text="5"/>
```

```
<Button  
    android:layout_column="2"  
    android:layout_row="2"  
    android:padding="3dp"  
    android:text="6"/>
```

```
<Button  
    android:layout_column="3"  
    android:layout_row="2"  
    android:padding="3dp"  
    android:text="-"/>
```



Ligne 3

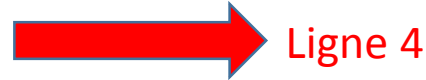
Solution – Fichier : content_gridlayout.xml

```
<Button
    android:layout_column="0"
    android:layout_row="3"
    android:padding="3dp"
    android:text="7"/>

<Button
    android:layout_column="1"
    android:layout_row="3"
    android:padding="3dp"
    android:text="8"/>

<Button
    android:layout_column="2"
    android:layout_row="3"
    android:padding="3dp"
    android:text="9"/>

<Button
    android:layout_column="3"
    android:layout_row="3"
    android:padding="3dp"
    android:text="*/>
```



Ligne 4

Solution – Fichier : content_gridlayout.xml

```
<Button  
    android:layout_column="0"  
    android:layout_row="4"  
    android:padding="3dp"  
    android:text="0"/>
```

```
<Button  
    android:layout_column="1"  
    android:layout_row="4"  
    android:padding="3dp"  
    android:text="."/>
```

```
<Button  
    android:layout_column="2"  
    android:layout_row="4"  
    android:padding="3dp"  
    android:text="C"/>
```

```
<Button  
    android:layout_column="3"  
    android:layout_row="4"  
    android:padding="3dp"  
    android:text="/" />
```

Ligne 5

```
<Button  
    android:layout_column="0"  
    android:layout_row="5"  
    android:padding="3dp"  
    android:text="="  
    android:layout_columnSpan="4"  
    android:layout_gravity="fill_horizontal"/>
```

Ligne 6

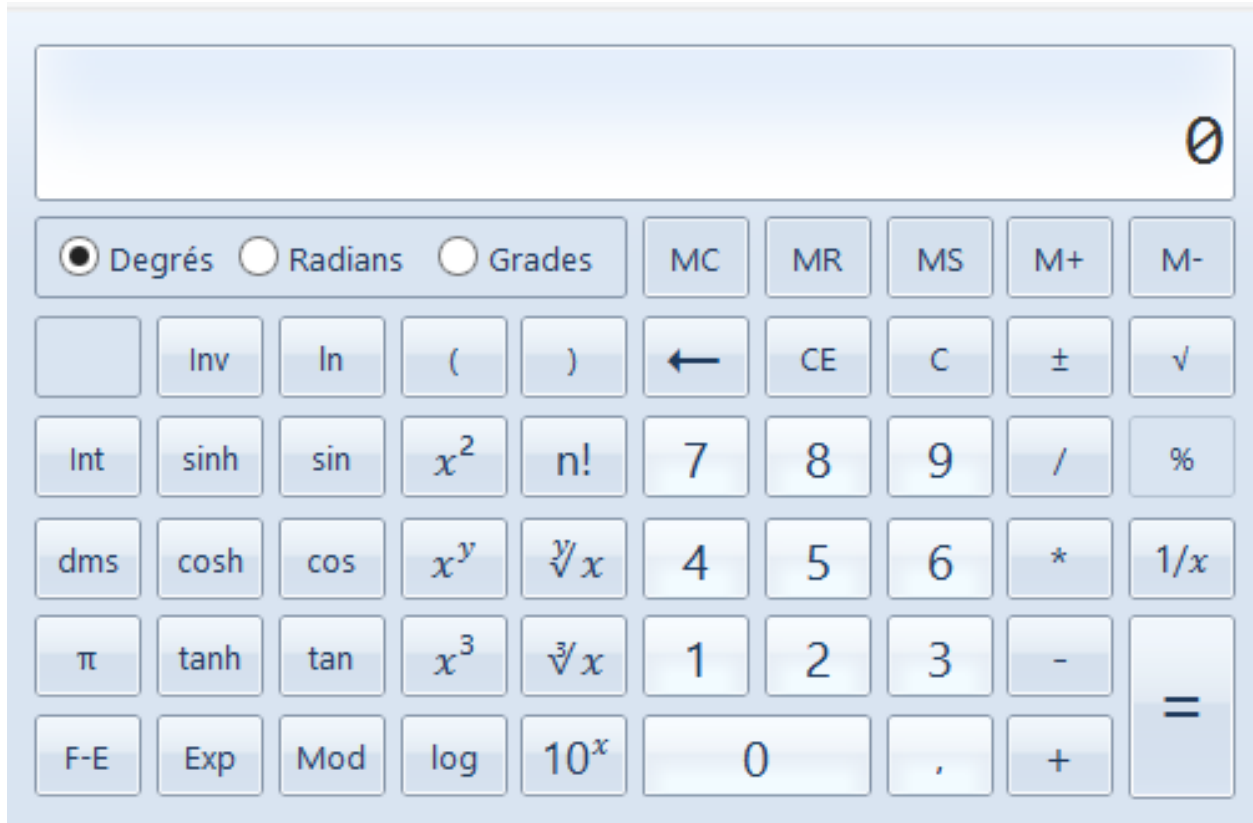
```
</GridLayout>
```

Solution – Fichier : MainActivity.java

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.content_gridlayout);  
    }  
}
```

Exercice 7

- Écrire une application Android qui utilise les layouts vus à date pour reproduire l'interface suivante
- À utiliser seulement l'onglet text de Android Studio pour construire l'interface – Il est interdit de toucher à l'onglet Design



Pour certains éléments, il faut utiliser ImageButton et n'oubliez pas de mettre les images dans le dossier res/drawable.

RelativeLayout

- Le Relative permet de placer les composants les uns en fonction des autres
- Un élément peut être positionné en
 - fonction de son conteneur
 - en fonction d'un autre élément de la vue

RelativeLayout

- Positionnement relatif au conteneur
 - Positionnement en fonction des 4 bords du conteneur
 - `android:layout_alignParentTop` : aligner le bord haut de l'élément avec le bord haut du conteneur
 - `android:layout_alignParentBottom` : aligner le bord inférieur de l'élément avec le bord inférieur du conteneur
 - `android:layout_alignParentLeft` : aligner le bord gauche de l'élément avec le bord gauche du conteneur
 - `android:layout_alignParentRight` : aligner le bord droit de l'élément avec le bord droit du conteneur
 - `android:layout_alignParentEnd` : aligner la fin de l'élément avec la fin du conteneur
 - `android:layout_alignParentStart` : aligner le début de l'élément avec le début du conteneur
 - On peut combiner plusieurs valeurs de positionnement

RelativeLayout

- Positionnement relatif aux autres éléments
 - Plusieurs options de positionnement différent existent. Il faut ajouter l'attribut voulu avec comme valeur l'identifiant de l'élément cible
 - `android:layout_above` : aligner le bord haut de l'élément avec le bord haut du conteneur
 - `android:layout_below` : aligner le bord inférieur de l'élément avec le bord inférieur du conteneur
 - `android:layout_toLeftOf` : aligner le bord gauche de l'élément avec le bord gauche du conteneur
 - `android:layout_toRightOf` : aligner le bord droit de l'élément avec le bord droit du conteneur
 - `android:layout_alignTop` : aligner le bord haut de l'élément avec le bord haut de l'élément référencé
 - `android:layout_alignBottom` : aligner le bord inférieur de l'élément avec le bord inférieur de l'élément référencé
 - `android:layout_alignLeft` : aligner le bord gauche de l'élément avec le bord gauche de l'élément référencé
 - `android:layout_alignRight` : aligner le bord droit de l'élément avec le bord droit de l'élément référencé
 - `android:layout_alignBaseline` : indique que les lignes de base de cet élément sont alignés avec celles de l'élément référencé
 - On peut combiner plusieurs valeurs de positionnement

Exercice 8

- Écrire une application Android qui utilise RelativeLayout pour reproduire l'écran de Login suivant :
- À utiliser seulement l'onglet text de Android Studio pour construire l'interface – Il est interdit de toucher à l'onglet Design



Solution

- Créer un nouveau projet Android (Empty Activity)
- modifier le fichier xml nommé activity_main.xml qui se trouve dans le dossier res/layout (Ce fichier a déjà un RelativeLayout)
- Écrire le code des diapos 59 à 60 pour éditer le fichier activity_main.xml

Solution – Fichier : activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.kfostine.login_relativelayout.MainActivity">

    <EditText
        android:id="@+id/idEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:hint="Adresse email"/>

    <EditText
        android:id="@+id/idPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@id/idEmail"
        android:layout_marginTop="10dp"
        android:hint="Mot de passe"
    />
```

Solution – Fichier : activity_main.xml

```
<Button
    android:id="@+id/idValider"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@id/idPassword"
    android:layout_below="@id/idPassword"
    android:text="Valider"
/>
```

```
<Button
    android:id="@+id/idAnnuler"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@id/idValider"
    android:layout_toLeftOf="@id/idValider"
    android:text="Annuler"
/>
```

```
</RelativeLayout>
```

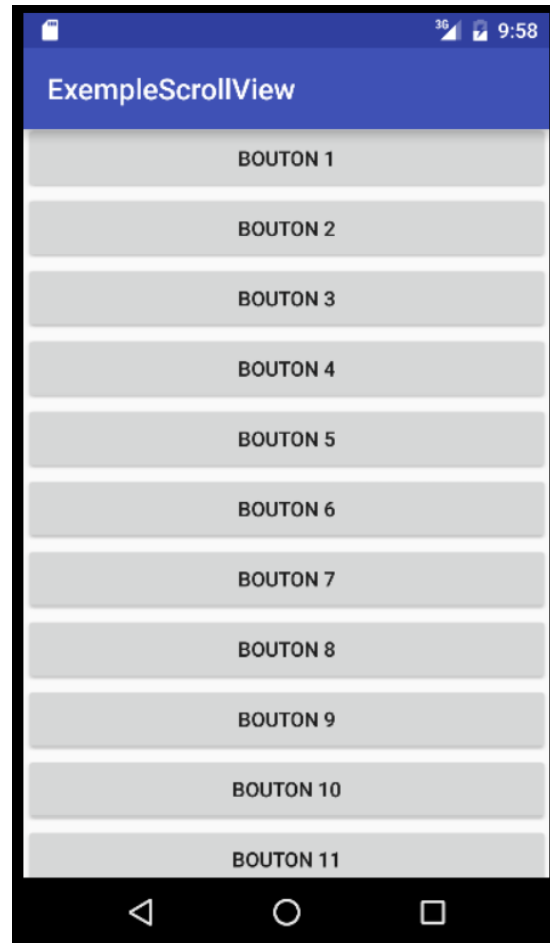
ScrollView

- Un ScrollView permet au conteneur d'être Scrollable pour dépasser la taille de l'écran de l'appareil
- Un ScrollView ne doit contenir qu'un seul fils (en général un autre layout)

Il ne faut jamais utiliser un ScrollView et une ListView pour empêcher d'avoir des conflits de Scroll entre la listView et le ScrollView

Exercice 9

- Écrire une application Android qui affiche 15 boutons alignés verticalement dans un ScrollView pour reproduire l'écran suivant :
- À utiliser seulement l'onglet text de Android Studio pour construire l'interface – Il est interdit de toucher à l'onglet Design



Solution

- Créer un nouveau projet Android (Empty Activity)
- Créer le fichier xml (layout : ScrollView) nommé main
 - Clic droit sur le dossier res/layout
 - New/Layout resource File
 - File name : main
 - Root Element : ScrollView
- Écrire le code des diapos 64 à 65 pour éditer le fichier main.xml
- Modifier l'activité MainActivity.java en écrivant le code de la diapo 66

Solution – Fichier : main.xml

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

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

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Bouton 1"/>

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Bouton 2"/>

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Bouton 3"/>

        <Button
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:text="Bouton 4"/>

    </LinearLayout>

</ScrollView>
```


Solution – Fichier : main.xml

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bouton 5"/>
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bouton 6"/>
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bouton 7"/>
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bouton 8"/>
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bouton 9"/>
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bouton 10"/>
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bouton 11"/>
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bouton 12"/>
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bouton 13"/>
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bouton 14"/>
```

```
<Button
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:text="Bouton 15"/>
```

```
</LinearLayout>
```

```
</ScrollView>
```

Solution – Fichier : MainActivity.java

```
public class MainActivity extends AppCompatActivity {  
  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
    }  
}
```

Les ressources

- Les ressources sont :
 - Les textes (constantes affichés dans une application Android)
 - Les images
 - Les éléments d'animation
 - Les éléments de rotation d'écran
 - Les couleurs
 - Les éléments de menus

Les ressources

- Les ressources doivent être externalisées (placées dans le dossier /res) pour :
 - Faciliter la maintenance (mise à jour des ressources)
 - Permettre l'utilisation de langues différentes dans une application
 - Faciliter la gestion de plusieurs tailles d'écran
 - Gérer l'orientation de l'écran
 - Gérer les spécificités propre à des pays différents
 - Gérer les versions d'Android
 - ...

Les ressources

- Toutes les ressources doivent se trouver dans le dossier /res de l'application Android
 - Le nom de fichier des ressources ne peut contenir que des lettres en minuscules, un point ou un underscore(_)

Les types de ressources

- **Les animations** : Définit des animations pré-déterminées
 - Les animations Tween (Tweening animation) sont enregistrées dans le dossier **res/anim/** et sont accessibles depuis la classe **R.anim** dans un fichier Java et depuis **"@anim/"** dans un fichier XML
 - Les animations de trames (Frame animation) sont enregistrées dans le dossier **res/drawable/** et sont accessibles depuis la classe **R.drawable** dans un fichier Java et depuis **"@drawable/"** dans un fichier XML
- **Les couleurs** : Définit les ressources de couleur qui changent en fonction de l'état de la vue.
 - Les couleurs sont enregistrées dans le dossier **res/color/** et sont accessible à partir de la classe **R.color** dans un fichier Java et depuis **"@color/"** dans un fichier XML
- **Les ressources Drawable** : Définit différents graphiques avec des bitmaps (images) ou XML.
 - Les images ou les graphiques sont enregistrés dans le dossier **res/drawable/** et sont accessibles à partir de la classe **R.drawable** dans un fichier Java et depuis **"@drawable/"** dans un fichier XML

Les types de ressources

- **Les ressources Layout** : Définit les layouts de l'application
 - Les layouts sont enregistrés dans le dossier **res/layout/** et sont accessibles à partir de la classe **R.layout** dans un fichier Java et depuis "**@layout/**" dans un fichier XML
- **Les menus** : Définit le contenu des menus de l'application Android
 - Les menus sont enregistrés dans le dossier **res/menu/** et sont accessibles à partir de la classe **R.menu** dans un fichier Java et depuis "**@menu/**" dans un fichier XML
- Les ressources String : Définit les chaînes de caractères qui sont affichés sur les composants d'interfaces graphiques des applications Android (étiquettes, message d'une boîte de dialogue, ...), les tableaux de chaînes et les pluriels (et incluent le formatage et le style associés à la chaîne).
 - Les chaînes de caractères sont enregistrés dans le dossier **res/values/strings** et sont accessibles à partir de la classe **R.string**, **R.array**, et **R.plural** dans un fichier Java et depuis "**@string/**", "**@array/**", et "**@plural/**" dans un fichier XML

Les types de ressources

- **Les styles** : Définit l'apparence et le format des éléments d'interfaces Utilisateur
 - Les styles sont enregistrés dans le dossier **res/values/styles** et sont accessibles à partir de la classe **R.style** dans un fichier Java et depuis **"@style/"** dans un fichier XML
- **Les fonts** : Définit les familles de police incluant des polices personnalisées en XML
 - Les fonts sont enregistrés dans le dossier **res/font/** et sont accessibles à partir de la classe **R.font** dans un fichier Java et depuis **"@font/"** dans un fichier XML
- **D'autres types de ressources** : Définit des valeurs telles que des booléens, des nombres entiers, des dimensions, des couleurs et d'autres tableaux.
 - Ils sont enregistrés dans le dossier **res/values/dimens**, **res.values/bools**, **res/values/integers**, **res/values/colors** mais accessibles à partir de sous-classes R originales (telles que **R.bool**, **R.integer**, **R.dimens**, etc.)

Le dossier Drawable

- Le dossier Drawable est en grande partie dédiée à des images de l'application
- Pour gérer les différentes résolutions d'une image dans une application il faut disposer du groupe de dossiers **/res/drawable-xxx**
- Xxx peut être remplacé par :
 - **ldpi** : écran à basse résolution(environ 120dpi)
 - **mdpi** : écran avec une moyenne résolution(environ 160dpi)
 - **hdpi** : écran avec une haute résolution(environ 240dpi)
 - **xhdpi** : écran avec une très haute résolution(environ 320dpi)
 - **xxhdpi** : écran avec une très très haute résolution(environ 480dpi)
 - **xxxhdpi** : écran avec une haute résolution(environ 640dpi)
 - **nodpi** : contient les images qui ne sont pas dépendantes de la résolution de l'appareil

Les chaines de caractères

- Tous les chaines de caractères doivent se trouver dans un ou plusieurs fichiers (strings.xml) se trouvant dans le dossier /res/values/strings
- Exemple de strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <!--Simple string-->
  <string name="hello">Hello World, ResourcesTuto!</string>
  <string name="app_name">ResourcesTuto</string>
  <string name="simpleString">Good Morning My name is Bond, James Bond</string>

  <!--String with parameters ($s for string, $d for decimal (any decimal: integer, double, float...)-->
  <string name="param_message">Hello, i love %1$s, i hate %2$s, i am %3$d years old</string>

  <!--String array-->
  <string-array name="i_like_array">
    <item>chocolate</item>
    <item>television</item>
    <item>internet</item>
    <item>nicotine</item>
    <item>hug</item>
    <item>Santa Claus</item>
  </string-array>

  <!--Plurals String-->
  <plurals name="singleOrPlural">
    <item quantity="one">I am alone on moon</item>
    <item quantity="other">I am 6 900 000 000 on earth</item>
  </plurals>
</resources>
```

Les chaines de caractères

Les chaînes simples s'utilisent, soit dans le code :

```
getString(R.string.simpleString)
```

soit dans le `layout.xml` pour déclarer un composant en utilisant son identifiant :

```
<TextView  
    android:id="@+id/simpleStringTV"  
    android:layout_width="fill_parent"  
    android:layout_height="wrap_content"  
    android:text="@string/hello"  
>
```

Les chaines de caractères

Les chaînes avec paramètres s'utilisent avec un formateur.

Dans le fichier *strings.xml*, nous avons déclaré la chaîne :

```
<!--String with parameters ($s for string, $d for decimal (any decimal:integer, double, float...)-->  
<string name="param_message">Hello, i love %1$s, i hate %2$s, i am %3$d years old</string>
```

Nous la manipulons ainsi en Java :

```
// Then instanciate a string with parameter  
String parameterString=String.format(getString(R.string.param_message),"peace","war",2);  
// And set the text view  
TextView textViewWithParam=(TextView)findViewById(R.id.parameterStringTV);  
textViewWithParam.setText(parameterString);
```

Les chaines de caractères

Les chaînes peuvent être regroupées sous forme de tableaux :

```
<!--String array-->
<string-array name="i_like_array">
  <item>chocolate</item>
  <item>television</item>
  <item>internet</item>
  <item>nicotine</item>
  <item>hug</item>
  <item>Santa Claus</item>
</string-array>
```

Dans le code, celui-ci se manipule ainsi :

```
// Instanciate the resources object:
Resources resources=getResources();
String[] planets = resources.getStringArray(R.array.i_like_array);
```

Les chaines de caractères

Les chaînes peuvent aussi posséder un genre singulier ou pluriel :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <plurals name="numberOfSongsAvailable">
    <item quantity="one">One song found.</item>
    <item quantity="other">%d songs found.</item>
  </plurals>
```


et s'utilisent dans le code ainsi :

```
int songCount = 4;
Resources res = getResources();
String messageSong = getQuantityString(R.plurals.numberOfSongsAvailable, songCount);
```

Contenu de messageSong

4 songs found

Gestion du pluriel

- L'attribut peut avoir les valeurs suivantes
 - **zero** : Cas de 0 élément
 - **one** : Cas d'un élément
 - **two** : Cas de deux éléments
 - **few** : cas d'un petit nombre d'éléments (trois ou quatre par exemple)
 - **many** : cas d'un grand nombre d'éléments (dix, douze par exemple)
 - **other** : autres cas 
- La méthode `getEntityString(int id, int quantite)` permet de récupérer la bonne chaine

Les chaines de caractères

Il faut faire attention aux signes ` et `

```
<string name="good_example">"This'll work"</string>  
<string name="good_example_2">This\'ll also work</string>  
<string name="bad_example">This doesn't work</string>
```


Les images

Les formats acceptés sont PNG et JPEG (GIF n'est pas recommandé).

Il suffit de déposer les fichiers image dans res\drawable pour qu'ils soient référencés par le système.

Ainsi l'image res/drawable/myimage.png peut être utilisée,

soit dans le code :

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

soit dans la description des IHM dans le fichier de layout :

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

Les couleurs

Elles se placent sous *res/values* et peuvent se nommer *color.xml* :

```
<resources>
  <color name="opaque_red">#f00</color>
  <color name="translucent_red">#80ff0000</color>
</resources>
```

Chaque couleur est décrite par son nom et son code RGB hexadécimal.

Dans le code, elles s'utilisent comme ça :

```
Resources res = getResources();
int color = res.getColor(R.color.opaque_red);
```

et dans un fichier de ressources :

```
<TextView
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:textColor="@color/translucent_red"
  android:text="Hello"/>
```

Les styles

La convention souhaite que ce fichier soit enregistré sous `res\values\styles.xml`, il peut alors être référencé dans votre application par la balise `@style/MonStyle`.

Les styles et les thèmes d'une application peuvent être surchargés ou définis :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="ToDoTheme" parent="@android:style/Theme.Black">
    <item name="android:textSize">12sp</item>
  </style>
</resources>
```

ou bien :

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <style name="MyTheme">
    <item name="android:textSize">12sp</item>
    <item name="android:textColor">#111</item>
  </style>
  <style name="MySubTheme" parent="MyTheme"; >
    <item name="android:textSize">8sp</item>
  </style>
</resources>
```

Et il sera utilisé dans le descripteur de ressources (le layout.xml) de la manière suivante :

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

L'internationalisation



- L'une des spécificités de Google Play est la facilité de rendre une application disponible à l'international
- Pour permettre à une application d'être accessible et utilisable par beaucoup d'utilisateurs il faut gérer le maximum de langues possible
 - Lors du lancement d'une application sur un téléphone la langue utilisée est celle de l'appareil si l'application gère cette langue sinon à la langue par défaut de l'application
- La traduction des chaînes de caractères est la brique essentielle pour internationaliser une application
- Il faut fournir des versions images, sons, vidéos adaptées aux différentes localisations

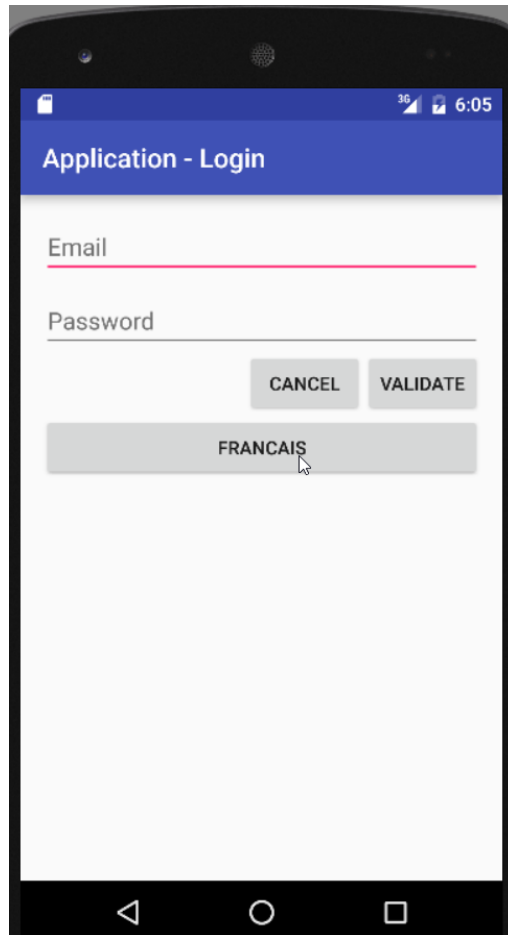
L'internationalisation

- Le dossier res/values est utilisé pour supporter la langue par défaut de l'application
 - Utiliser values pour stocker les fichiers de langue anglaise
 - Pour les autres langues, il faut créer les dossiers suivants :
 - Français : values-fr
 - Espagnol : values-es
 - Japonais : values-ja

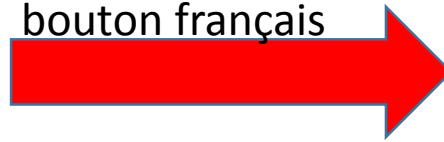
Il faut créer un dossier par langue supportée par l'application et fournir les fichiers traduits dans les langues cibles

Exercice 10

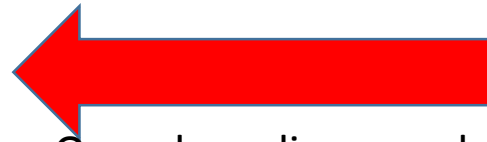
- Écrire une application Android qui refait l'exercice 9 mais en permettant à l'utilisateur de passer de la langue française à la langue Anglaise quand on clique sur le bouton « Anglais » et le texte du bouton devrait changer de Anglais à Français



Quand on clique sur le bouton français



Quand on clique sur le bouton English



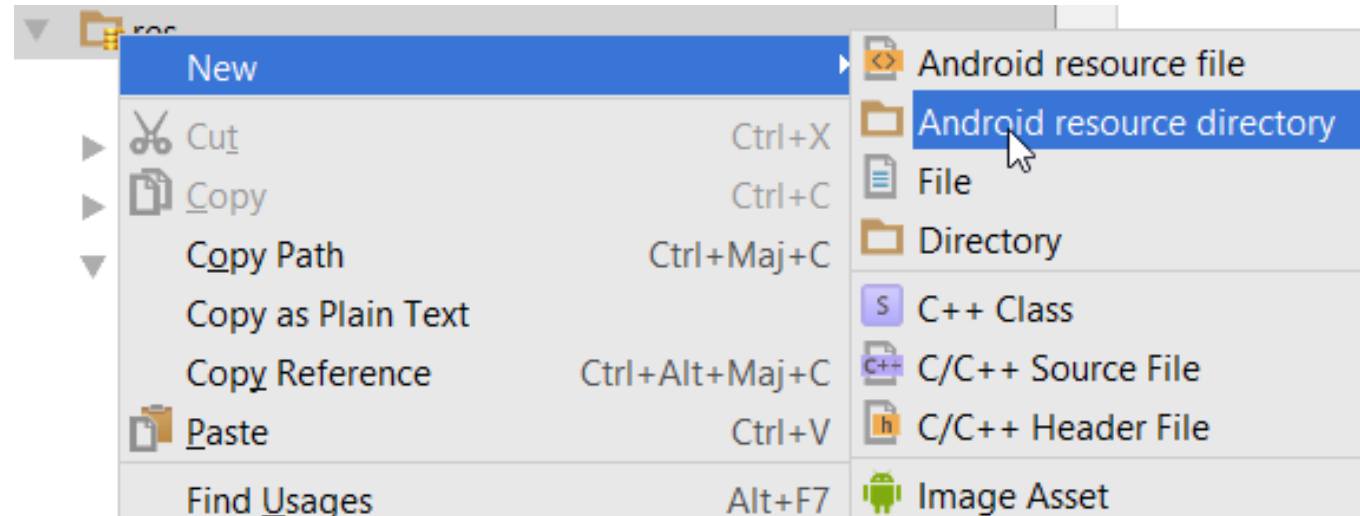
Solution

- Remplir le fichier res/values/strings pour les chaines de caractères en anglais

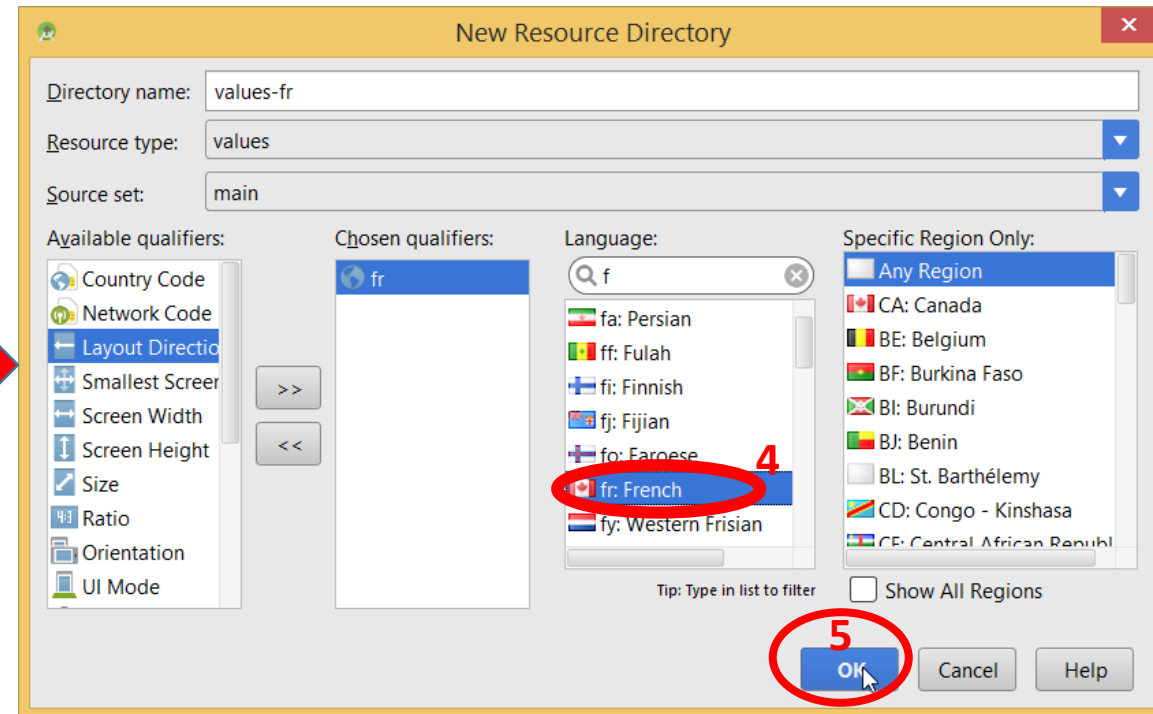
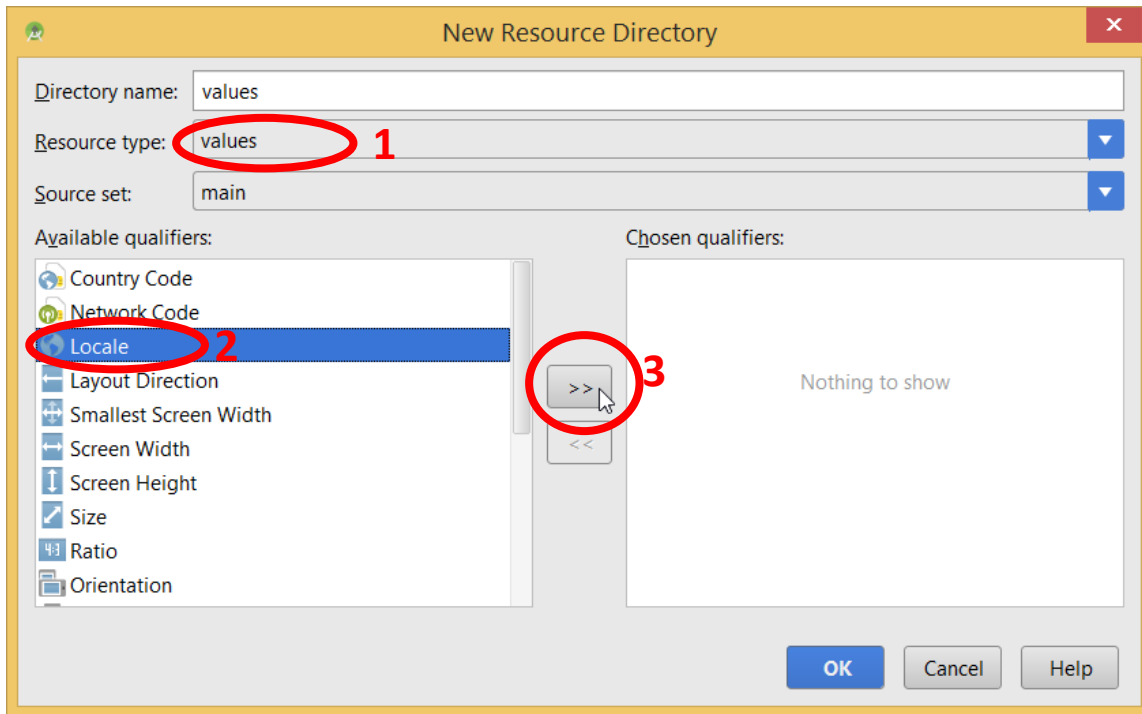
```
<resources>
    <string name="app_name">Application - Login</string>
    <string name="text_email">Email</string>
    <string name="text_password">Password</string>
    <string name="text_cancel">Cancel</string>
    <string name="text_validate">Validate</string>
    <string name="text_language">Francais</string>
</resources>
```

Solution

- Le dossier res/values est utilisé pour la langue anglaise
- Créer un dossier res/values-fr pour la langue française
 - Clic droit sur le dossier res
 - New/Android resource directory

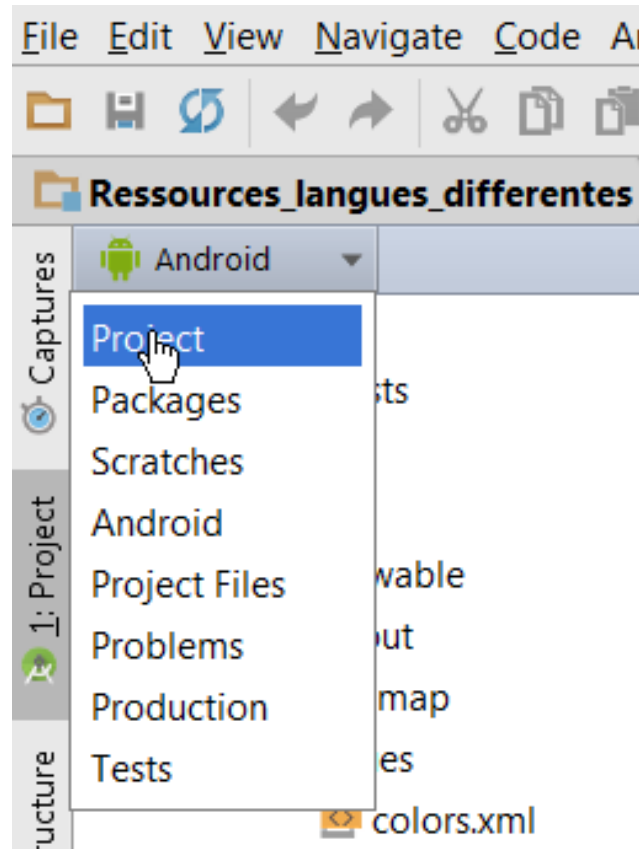


Solution



Solution

- Passer de l'explorateur Android à Project pour voir le nouveau dossier créé dans l'explorateur de projet



Solution

- Copier le fichier res/values/strings.xml dans res/values-fr/
- Modifier le fichier res/values-fr/string.xml de manière que son contenu puisse ressembler à ceci :

```
<resources>
    <string name="app_name">Application - Connection</string>
    <string name="text_email">Adresse courriel</string>
    <string name="text_password">Mot de passe</string>
    <string name="text_cancel">Annuler</string>
    <string name="text_validate">Valider</string>
    <string name="text_language">English</string>
</resources>
```

Solution – Éditer le fichier res/layout/activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.kfostine.ressources_langues_différentes.MainActivity">

    <EditText
        android:id="@+id/idEmail"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:hint="@string/text_email"/>

    <EditText
        android:id="@+id/idPassword"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_below="@id/idEmail"
        android:layout_marginTop="10dp"
        android:hint="@string/text_password"
    />
```

Solution – Éditer le fichier res/layout/activity_main.xml

```
<Button
    android:id="@+id/idValider"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignRight="@id/idPassword"
    android:layout_below="@id/idPassword"
    android:text="@string/text_validate"
/>
```

```
<Button
    android:id="@+id/idAnnuler"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@id/idValider"
    android:layout_toLeftOf="@id/idValider"
    android:text="@string/text_cancel"
/>
```

```
<Button
    android:id="@+id/idLangue"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_below="@id/idValider"
    android:text="@string/text_language"
/>
```

```
</RelativeLayout>
```

Solution – Programmer le bouton pour changer de langue

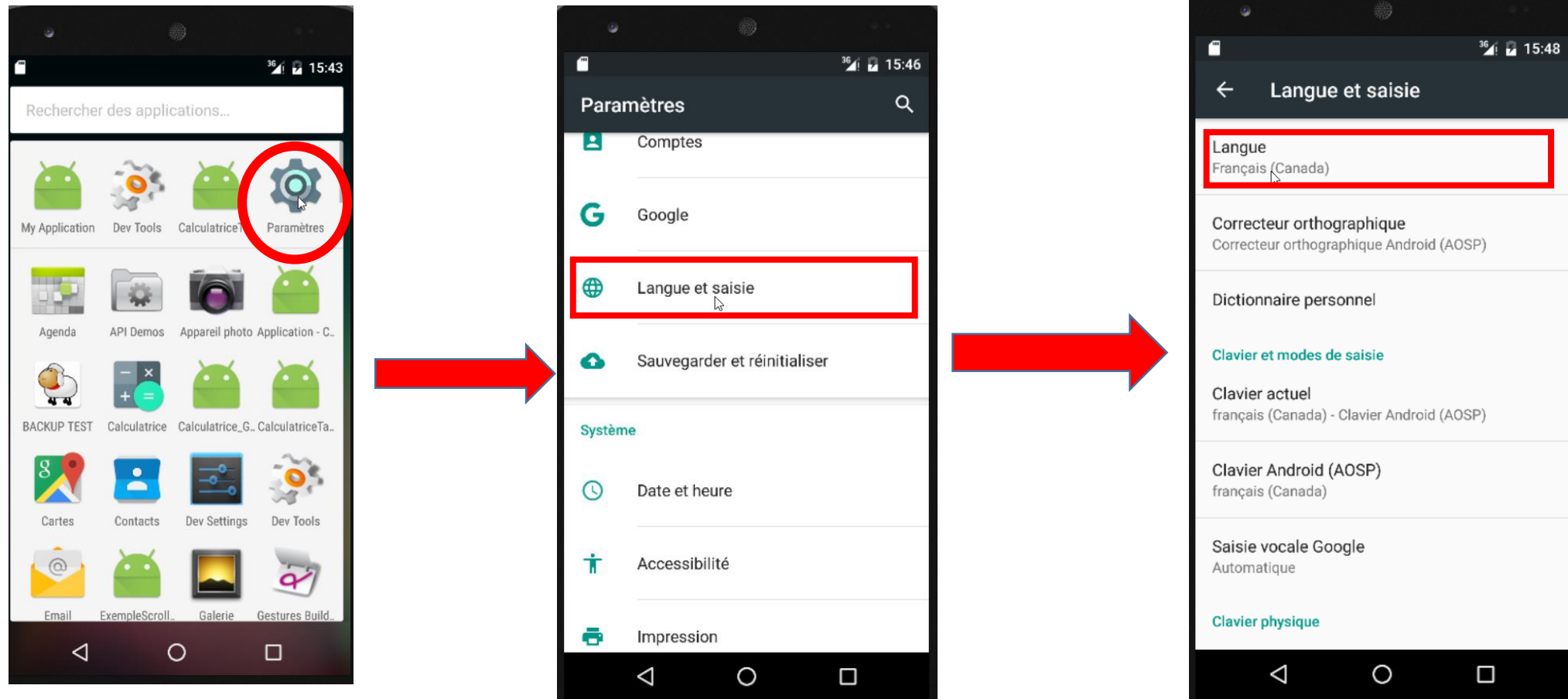
```
public class MainActivity extends AppCompatActivity {
    Button btnLangue;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        btnLangue = (Button) findViewById(R.id.idLangue);
        btnLangue.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String langueVoulue = btnLangue.getText().toString();
                String codeLangue;
                if(langueVoulue.equalsIgnoreCase("Francais")){
                    codeLangue = "fr";
                }
                else{
                    codeLangue = "en";
                }

                Locale locale = new Locale(codeLangue);
                Locale.setDefault(locale);
                Configuration config = new Configuration();
                config.locale = locale;
                getBaseContext().getResources().updateConfiguration(config,
                    getBaseContext().getResources().getDisplayMetrics());
            }
        });
    }
}
```



Test de l'application

- Pour tester l'application, il faut aller dans les paramètres et changer la langue du système de l'émulateur pour rouler l'application une fois en français et une autre fois en anglais.
- Après, Tester l'application avec les boutons Français/Anglais



Références

- <https://www.mkyong.com/android/android-linearlayout-example/>
- <https://developer.android.com/guide/topics/resources/available-resources.html>
- Guide de développement d'applications Java pour Smartphones et Tablettes – Sylvain Hébuterne et Sebastien Perochon – Eni – 2015