Fondamentaux d'Android

2. Première interface utilisateur interactive

Hatem Aziza - September 2024

Developers

Introduction

L'interface utilisateur (UI) qui apparaît sur l'écran d'un appareil Android se compose d'une hiérarchie d'objets appelés vues : chaque élément de l'écran est un fichier **View**. La classe View représente l'élément de base de tous les composants de l'interface utilisateur et la classe de base pour les classes qui fournissent des composants interactifs de l'interface utilisateur tels que des boutons, des cases à cocher et des champs de saisie de texte. Les sous-classes de View couramment utilisées décrites dans plusieurs leçons comprennent :

- **TextView** pour afficher du texte.
- **EditText** pour permettre à l'utilisateur de saisir et de modifier du texte.
- **Button** et d'autres éléments cliquables (tels que **RadioButton**, **CheckBox**, et **Spinner**) pour fournir un comportement interactif.
- **ScrollView** et **RecyclerView** pour afficher les éléments déroulants.
- **ImageView** pour afficher des images.
- **ConstraintLayout** et **LinearLayout** pour contenir d'autres éléments View et les positionner.

Ce que vous apprendrez

- Comment créer une application avec un comportement interactif.
- Comment utiliser l'éditeur de mise en page pour concevoir une mise en page.
- Comment modifier la mise en page en XML.

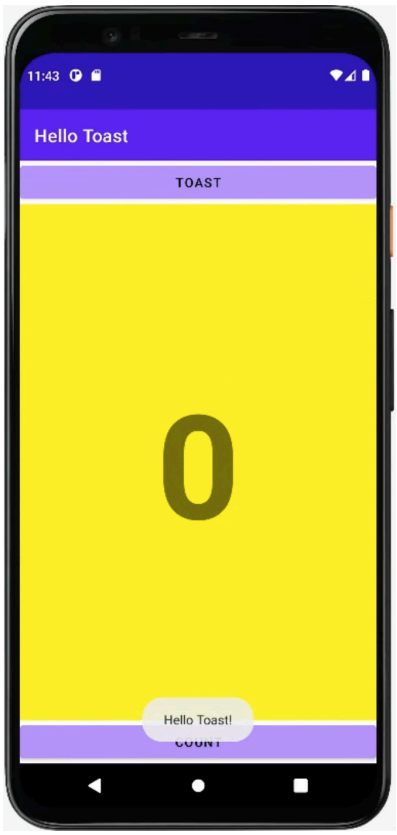
Ce que tu feras

- Créez une application et ajoutez deux éléments Button et un TextView
- Manipulez chaque élément du ConstraintLayout pour les contraindre aux marges et aux autres éléments.
- Modifiez les attributs des éléments de l'interface utilisateur.
- Modifiez la mise en page de l'application en XML.
- Extrayez les chaînes codées en dur dans les ressources de chaînes.
- Implémentez des méthodes de gestion des clics pour afficher des messages à l'écran lorsque l'utilisateur appuie sur chaque élément Button

Présentation de l'application

L'application **HelloToast** se compose de deux éléments Button et d'un TextView. Lorsque l'utilisateur appuie sur le premier Button, il affiche un court message (Toast) sur l'écran. En appuyant sur la seconde, Button vous augmentez le compteur de « clics » affiché dans le TextView, qui commence à zéro.

Voici à quoi ressemblera l'application terminée :



Tâche 1 : Créer et explorer un nouveau projet

Créez le projet Android Studio

1. Démarrez Android Studio et créez un nouveau projet avec les paramètres suivants :

Attribut	Valeur
Modèle	**Activité vide**

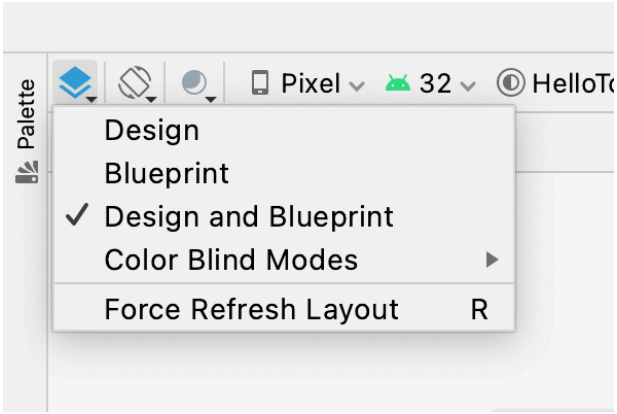
Nom	**Bonjour Toast**
Nom du paquet	**com.exemple.hellotoast**
Langue	**Java**
SDK minimal	**API 21 : Android 5.0 (Lollipop)**
Utiliser les anciennes bibliothèques Android.support	Décoché

2. Sélectionnez **Exécuter** > **Exécuter l'application** ou cliquez sur l' icône Exécuter dans la barre d'outils pour créer et exécuter l'application sur l'émulateur ou votre appareil.

Tâche 2 : Ajouter des éléments View dans l'éditeur de mise en page

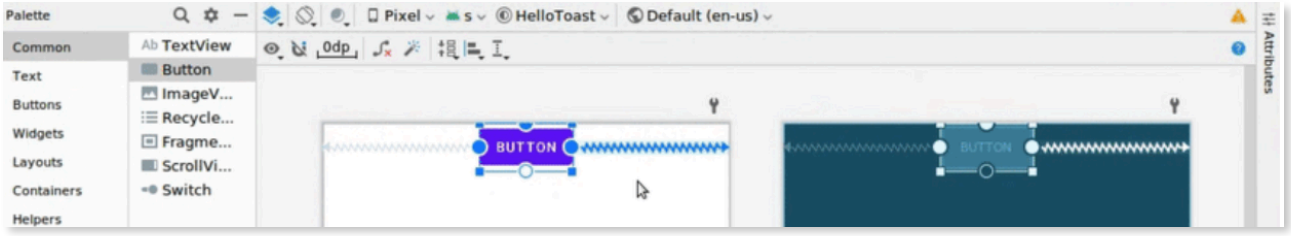
Examinez les contraintes des éléments

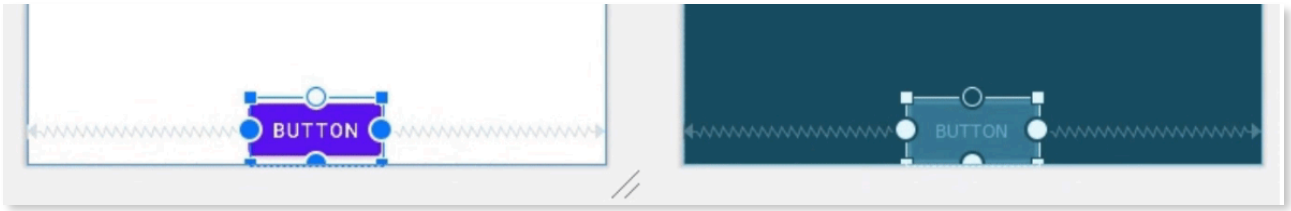
1. Ouvrez **activity_main.xml** depuis le volet **Projet** > **Android** s'il n'est pas déjà ouvert.
2. Cliquez sur le bouton **Sélectionner une surface de conception** dans la barre d'outils et choisissez **Conception et plan (Design and Blueprint)**.



3. Sélectionnez **TextView** dans le volet Arborescence des composants . Le TextView « Hello World » est mis en surbrillance dans les volets de conception et de plan et les contraintes de l'élément sont visibles.
4. Explorez les différentes contraintes.

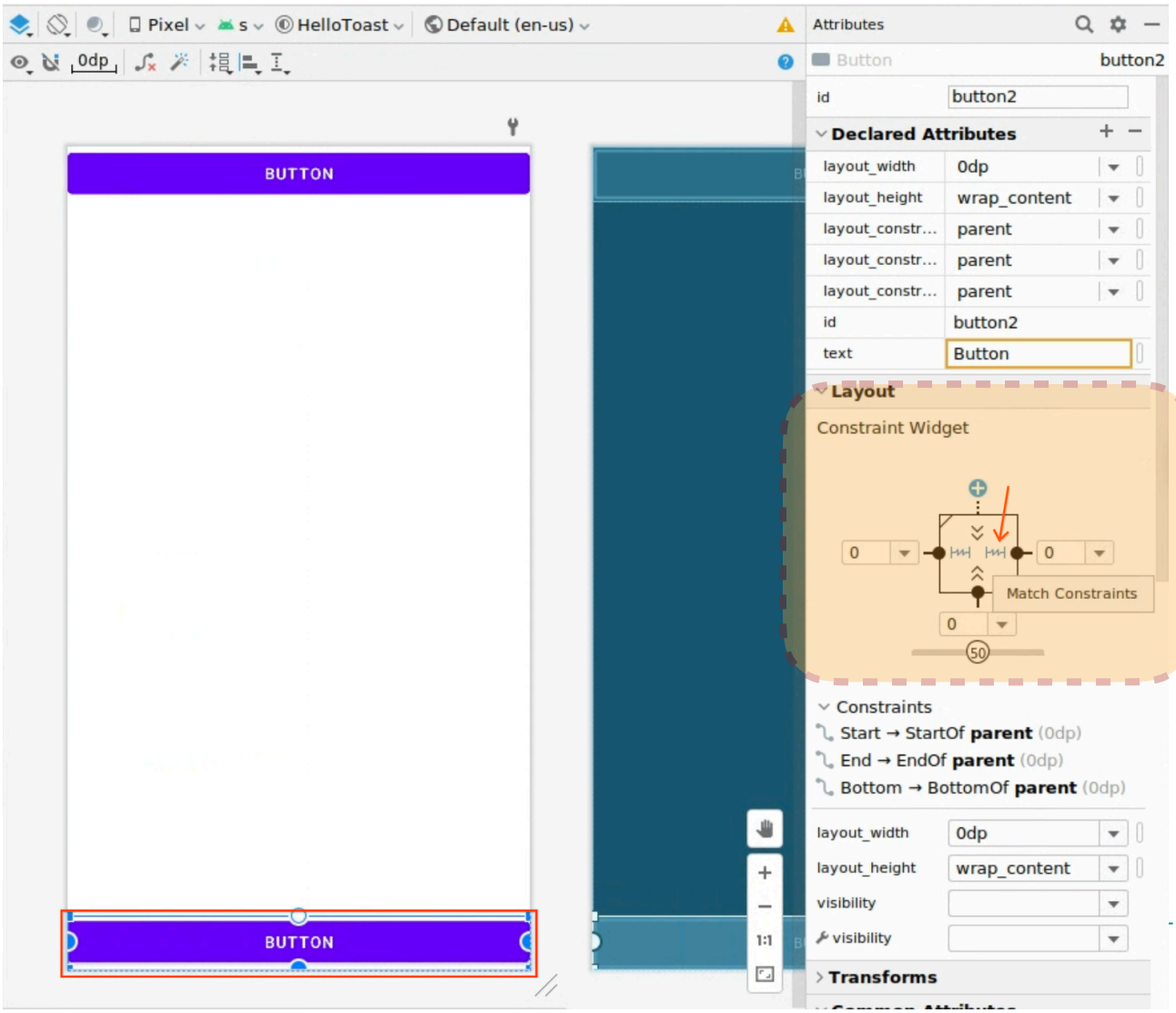
Ajouter deux bouton au layout





Tâche 3 : Modifier les attributs des éléments de l'interface utilisateur

Changer la taille du bouton



Modifiez les attributs du bouton

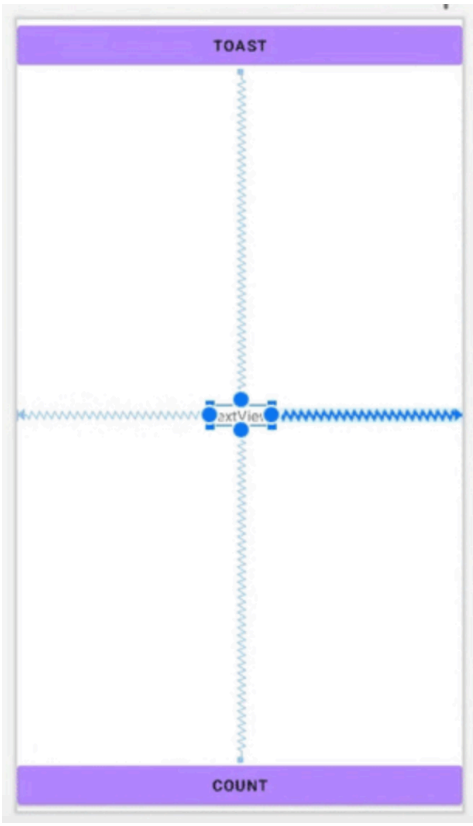
Chaque View a besoin d'un identifiant unique. Et pour être utiles, les éléments Button ont besoin de texte. Les éléments View peuvent également avoir des arrière-plans qui peuvent être des couleurs ou des images.

Le volet Attributs offre un accès à tous les attributs que vous pouvez attribuer à un élément View. Vous pouvez saisir des valeurs pour chaque attribut, telles que les attributs **android:id**, **backgroundTint**, **textColor** et .**text**

1. Après avoir sélectionné le premier Button, modifiez le champ **android:id** en **bouton_toast**, qui est utilisé pour identifier l'élément dans le layout. Lorsque la boîte de dialogue Renommer apparaît, sélectionnez **Refactor** . Cela modifie toutes les instances ID du projet avec le nouveau nom.
2. Définissez l'attribut **backgroundTint** sur **@color/purple_200**.
3. Définissez l'attribut **textColor** sur **@android:color/black**.
4. Modifiez l'attribut text en Toast .
5. Effectuez les mêmes modifications d'attribut pour le second Button, en utilisant **button_count** comme **ID**, **Count** pour l'attribut **text** et les mêmes couleurs.

Tâche 4 : Ajouter un TextEdit et définir ses attributs

Ajouter un TextView et des contraintes



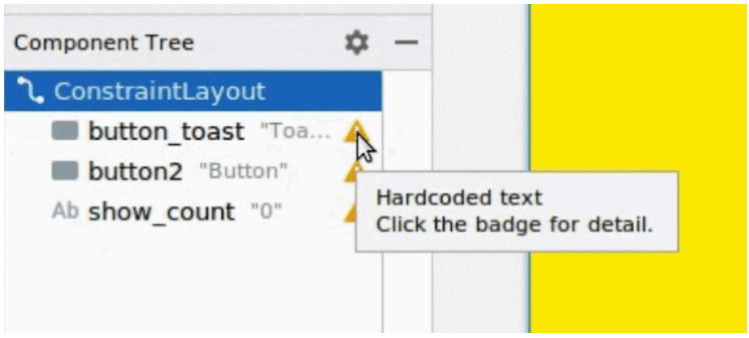
Définissez les attributs TextView

Définissez les attributs pour le TextView comme décrit ci-dessous.

1. Définissez le **ID** sur **show_count** .
2. Remplacez les contrôles de taille de vue horizontale et verticale (**layout_width** et **layout_height**) par **match_parent**.
3. Réglez le **text** sur **0** .
4. Faites défiler le volet jusqu'à Attributs communs et développez l'attribut **textAppearance**.
5. Réglez le **textSize** sur **160sp** .
6. Définissez le **textColor** sur **@color/purple_500**.
7. Réglez le **textStyle** sur **B** (gras).
8. Faites défiler le volet jusqu'à All Attributes et définissez le **background** sur **#FFFF00** pour une nuance de jaune.
9. Faites défiler vers le bas **gravity**, développez **gravity** et sélectionnez le **centre** .

Tâche 5 : Modifier le layout en XML

La mise en page de l'application Hello Toast est presque terminée ! Cependant, un point d'exclamation apparaît à côté de chaque élément de l'interface utilisateur dans l' arborescence des composants . Passez votre pointeur sur ces points d'exclamation pour voir les messages d'avertissement, comme indiqué ci-dessous. Le même avertissement apparaît pour les trois éléments : les chaînes codées en dur doivent utiliser des ressources.



1. ouvrez le fichier activity_main.xml s'il ne l'est pas déjà, puis cliquez sur l' onglet **Code**.
2. Cliquez une fois sur le mot ("Toast"(le premier avertissement en surbrillance).
3. Cliquez sur l'ampoule qui apparaît au début de la ligne et sélectionnez **Extraire la ressource de chaîne** dans le menu contextuel.
4. Entrez **button_label_toast** pour le **nom de la ressource**.

5. Cliquez sur OK. Une ressource chaîne est créée dans le fichier **res/values/strings.xml** et la chaîne dans votre code est remplacée par une référence à la ressource : **@string/button_label_toast**

6. Extrayez les chaînes restantes :
   button_label_count pour "**Count**"et **count_initial_value** pour "**0**".

7. Dans le volet **Projet** > **Android** , développez les valeurs dans **res** , puis double-cliquez sur **strings.xml** pour voir vos ressources de chaîne dans le fichier strings.xml :

```
<resources>
    <string name="app_name">Hello Toast</string>
    <string name="button_label_toast">Toast</string>
    <string name="button_label_count">Count</string>
    <string name="count_initial_value">0</string>
</resources>
```

Tâche 6 : Ajouter des gestionnaires onClick pour les boutons

Dans cette tâche, vous ajoutez une méthode Java pour chaque Button de MainActivity qui s'exécute lorsque l'utilisateur appuie sur le bouton Button.

Ajoutez l' attribut et le gestionnaire onClick à chaque bouton

```
<Button
    android:id="@+id/button_toast"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:backgroundTint="@color/purple_200"
    android:text="@string/button_label_toast"
    android:textColor="@android:color/black"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.498"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    android:onClick="showToast"/>
<Button
    android:id="@+id/button_count"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:backgroundTint="@color/purple_200"
    android:text="@string/button_label_count"
    android:textColor="@android:color/black"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    android:onClick="countUp" />
```

Modifiez le gestionnaire du bouton Toast

```
public void showToast(View view) {
  Toast toast = Toast.makeText(this, R.string.toast_message,
                                          Toast.LENGTH_SHORT);
  toast.show();
}
```

Modifiez le gestionnaire du bouton Count

```
public class MainActivity extends AppCompatActivity {
  private int mCount = 0;
  private TextView mShowCount;
```

```
@Override
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  mShowCount = (TextView) findViewById(R.id.show_count);
}
```

```
public void countUp(View view) {
  mCount++;
  if (mShowCount != null) {
    mShowCount.setText(Integer.toString(mCount));
  }
}
```