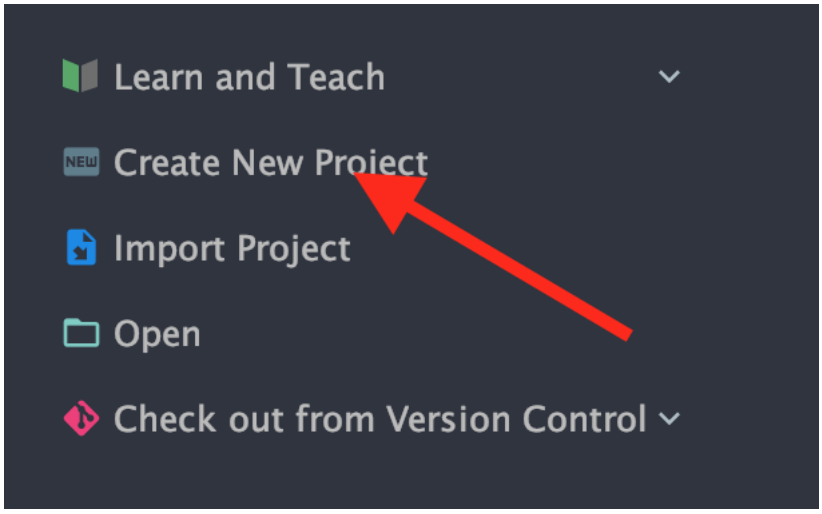
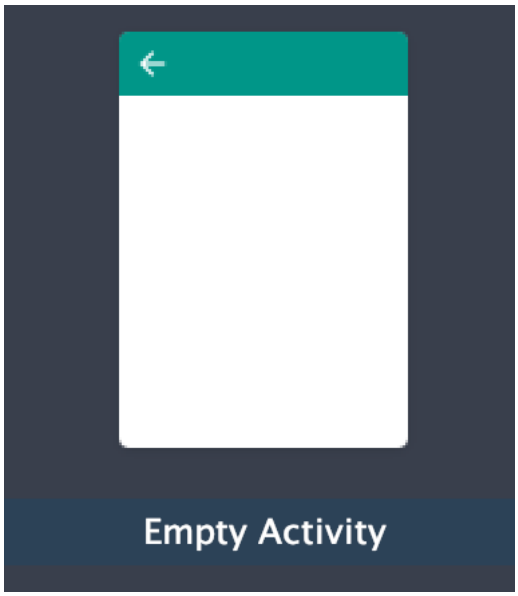
V. Notre Première APP ANDROID

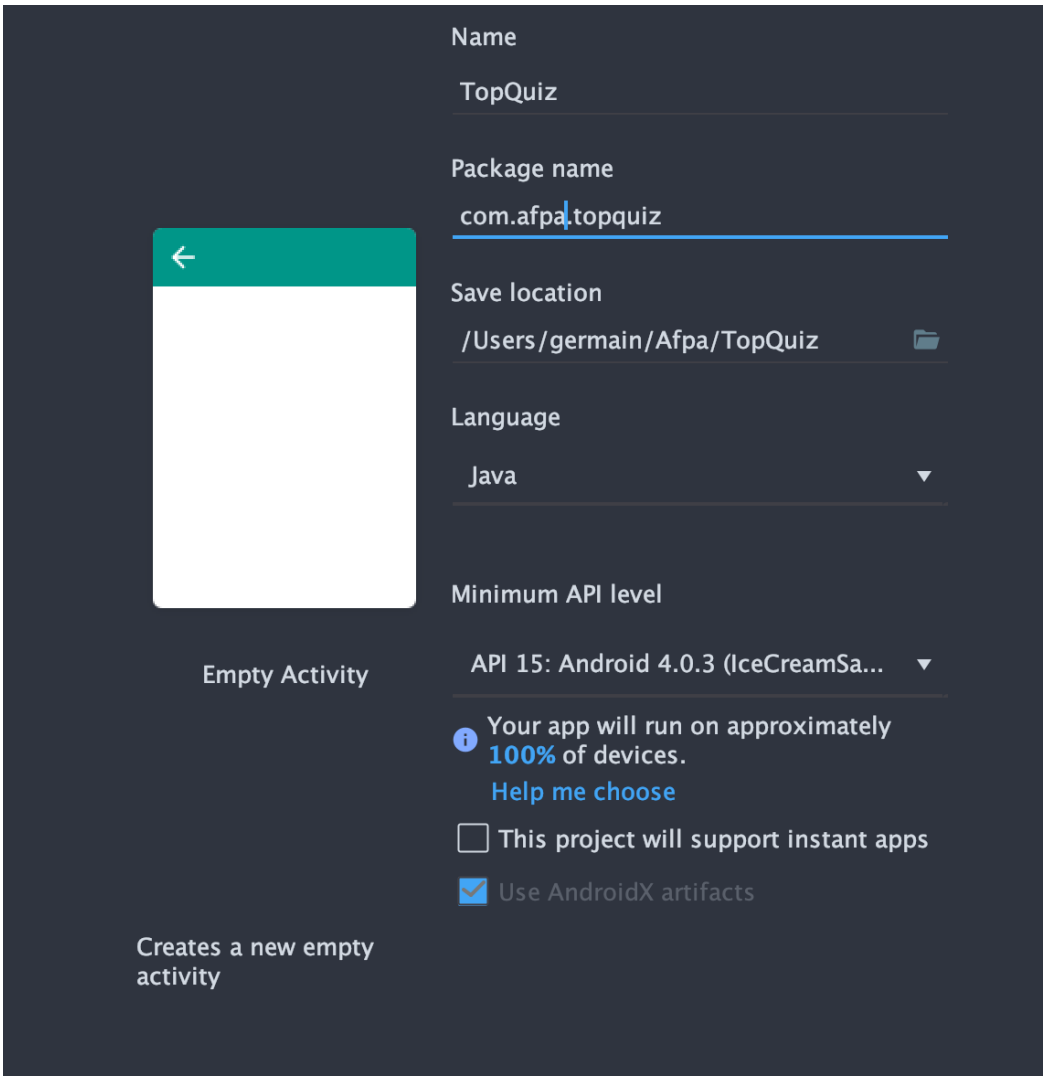
Sur la première page ou dans File cliquez sur **Create new Project**



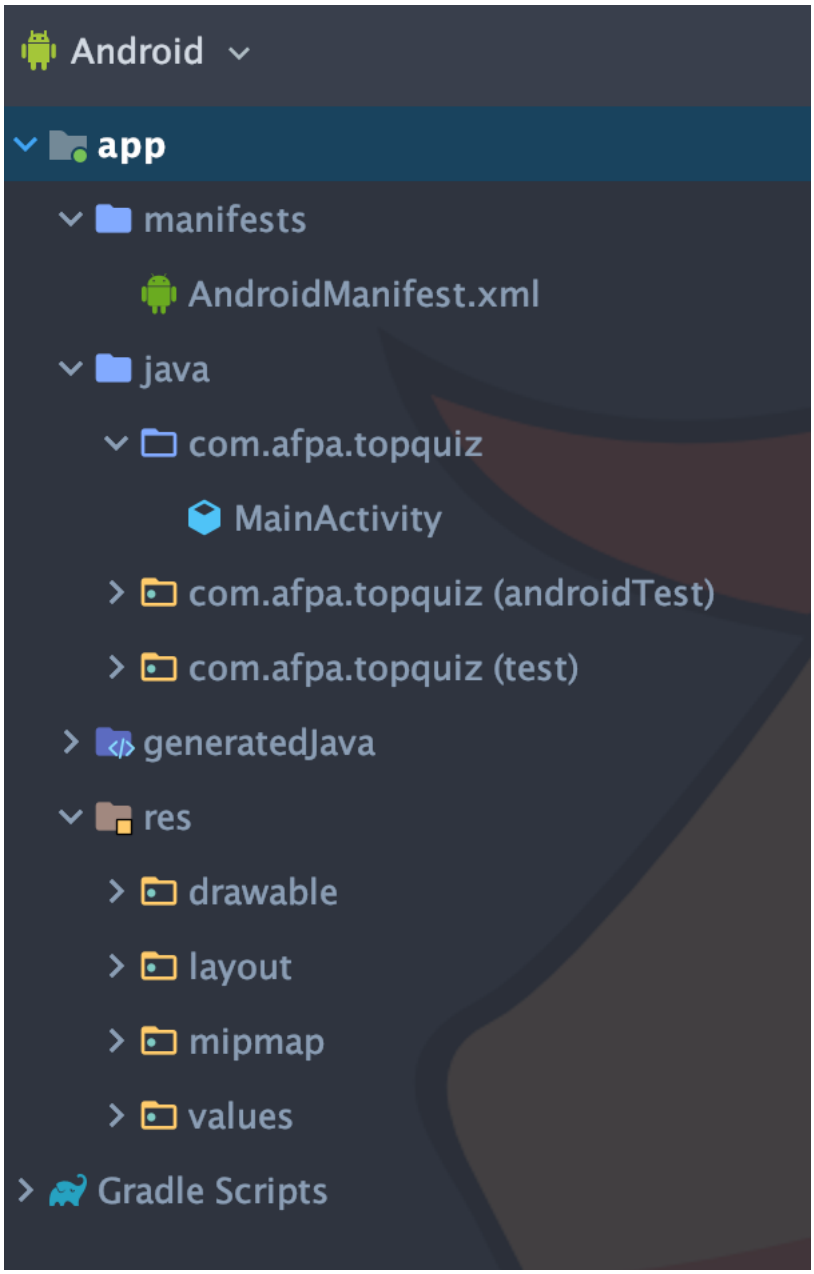
Choisissez android dans la colonne de gauche puis **empty activity**



Remplissez le nom du **projet**, du **Package** principal et le **dossier** où sera enregistré votre projet



Et laissez travailler **Gradle** et on obtient un projet android:

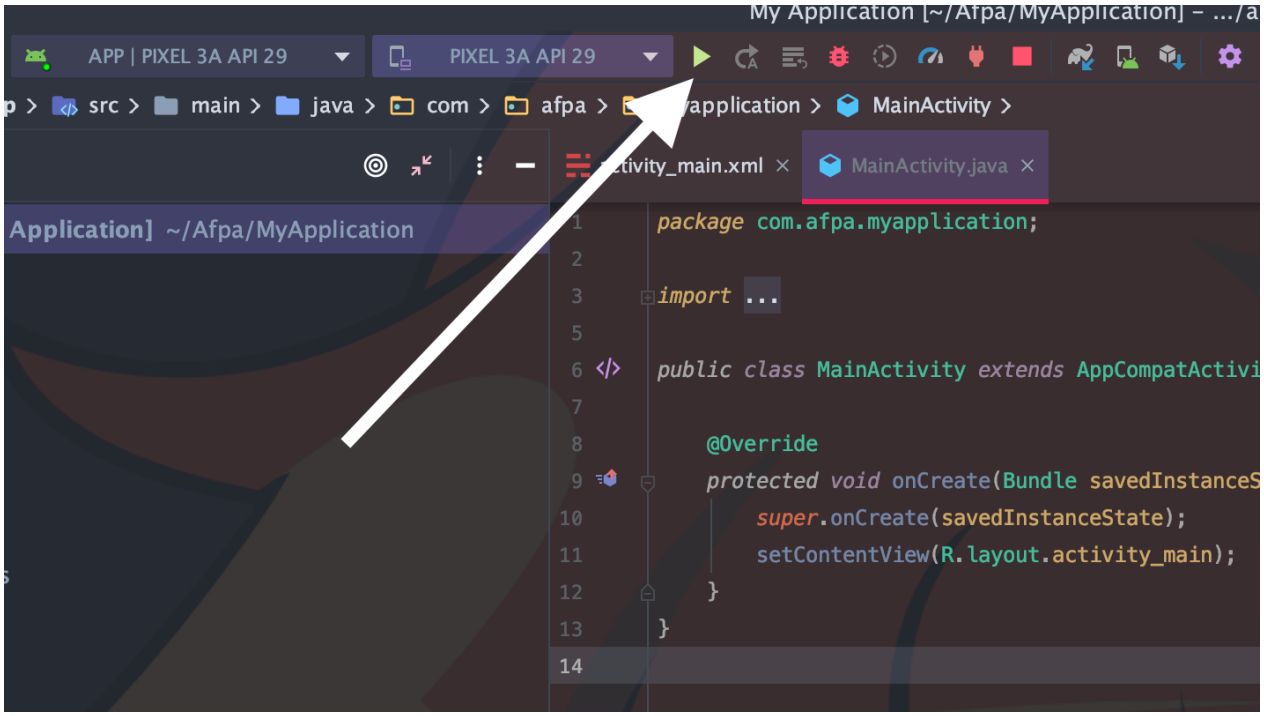


- Dans le dossier **Java** nous trouverons le code source java
- Dans le dossier **res** nous trouverons 4 dossiers :
  - **drawable**: le dossier des images et des logos
  - **layout**: le dossier des *mise en page*
  - **mipmap**: le dossier pour l'icône de l'App
  - **values**: le dossier des paramètre de l'App

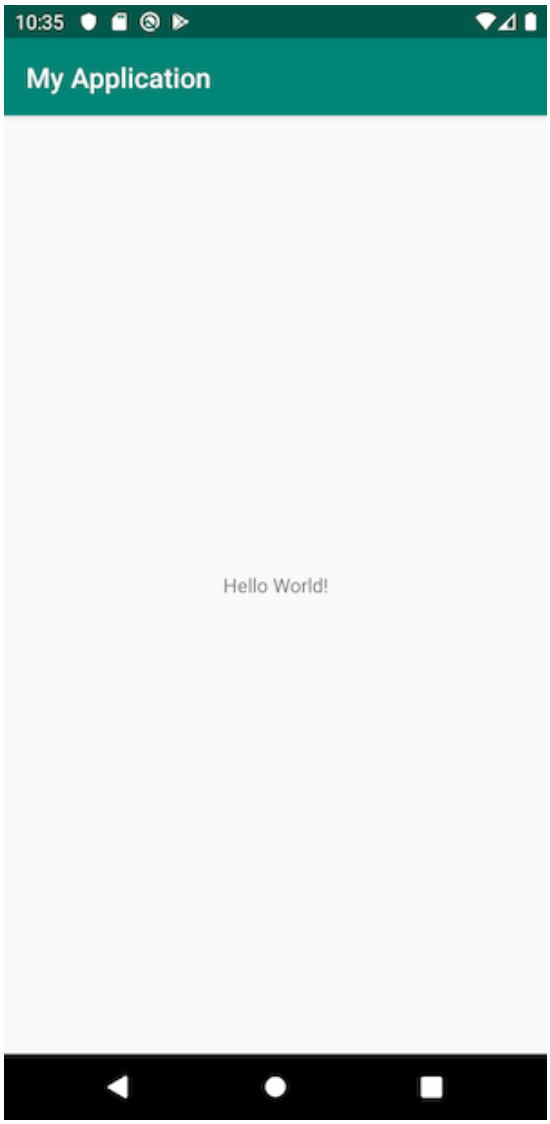
par exemple, on peut trouver dans `values/strings` l'entête de notre appli "My Application" que l'on modifiera en "Mon Appli" par la suite.

```
<resources>
    <string name="app_name">Mon appli</string>
</resources>
```

à ce niveau si tout s'est bien déroulé vous pouvez lancer votre application dans l'émulateur en cliquant sur la flèche verte.

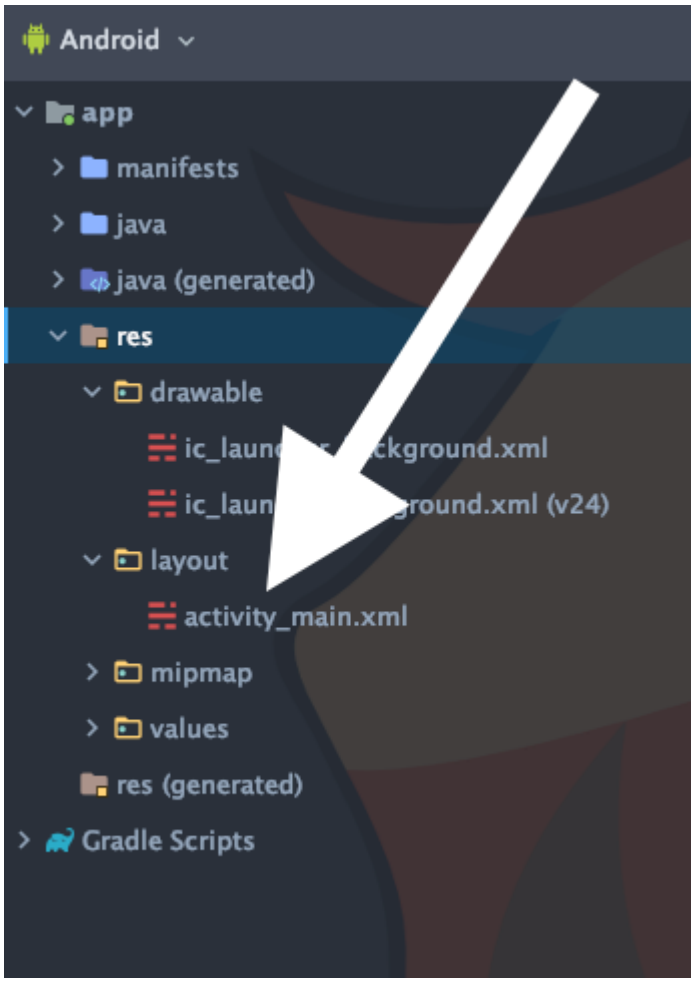


Et la magie opère :



Nous allons maintenant modifier cette application pour la rendre interactive.

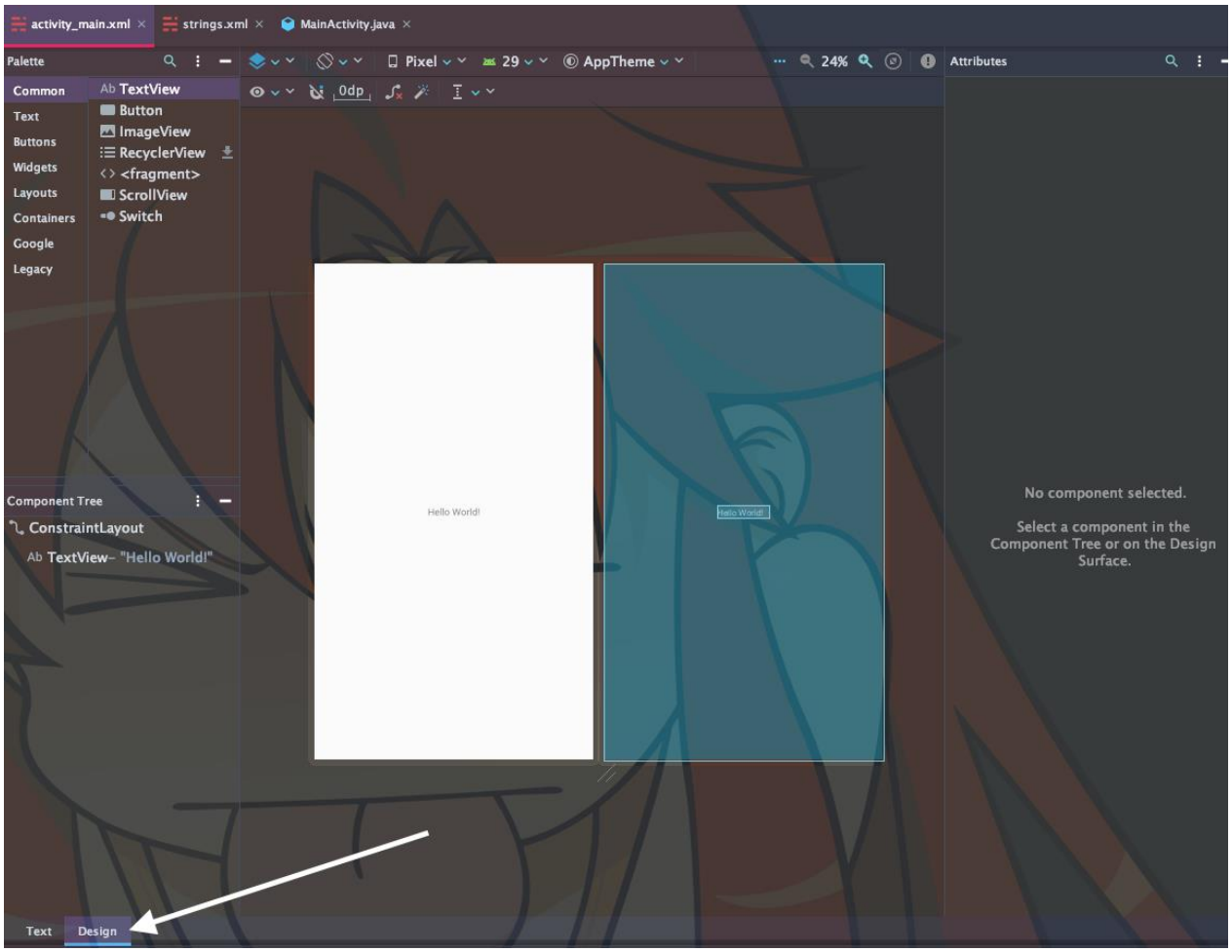
Ouvrons le fichier activity_main.xml



Voici ce que vous devez trouver à l'intérieur:

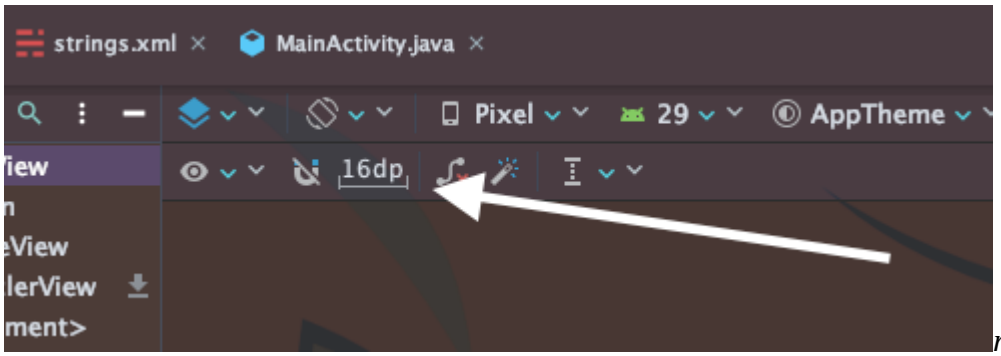
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
        xmlns:app="http://schemas.android.com/apk/res-auto"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Hello World!"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent"/>

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

En bas de la fenètre vous pouvez trouver un bouton **Design** qui vous permet de passer en mode layout:



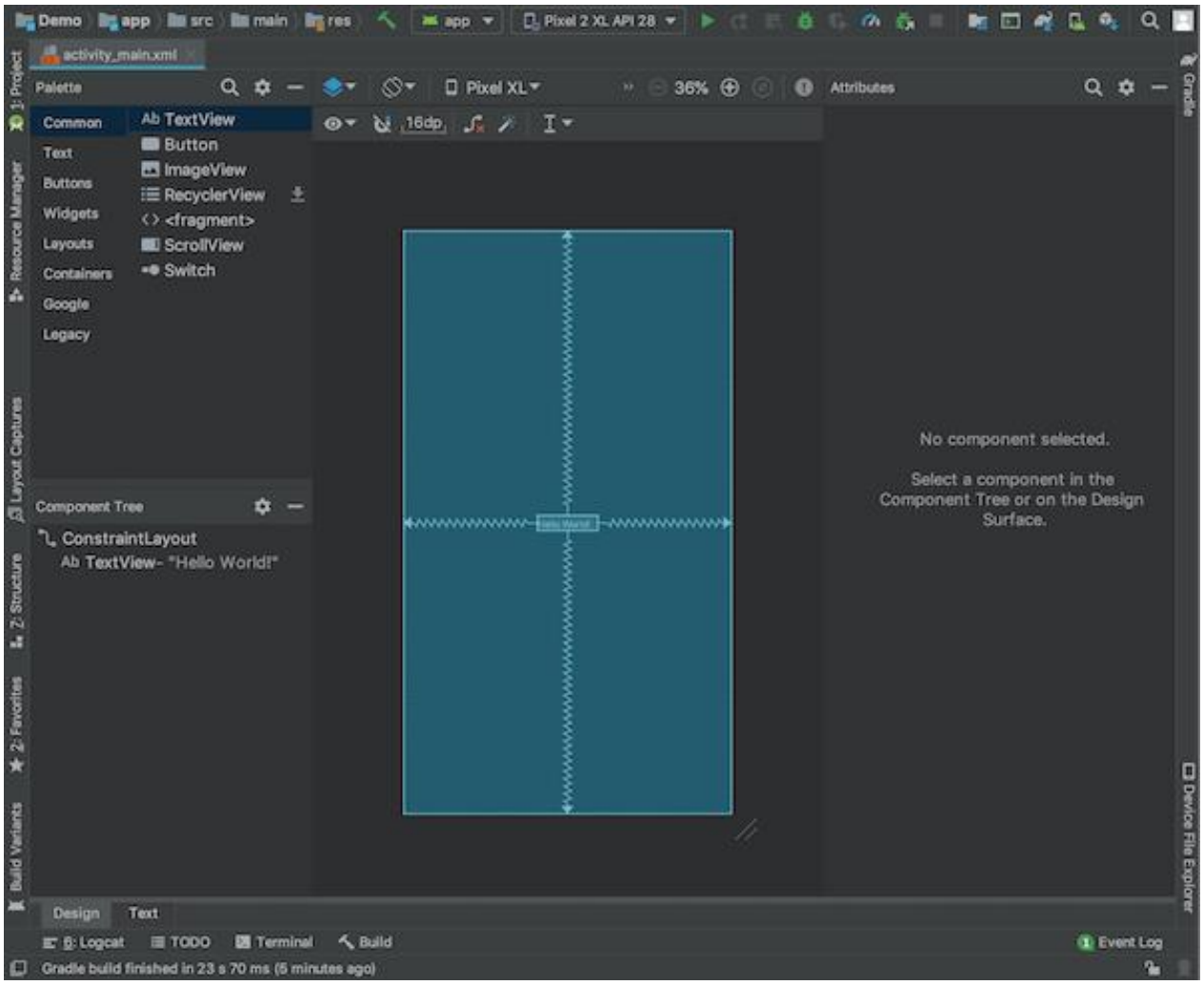
Cliquez sur  et **show all constraints** pour afficher les contraintes.

Réglons les marges par défaut à **16dp**

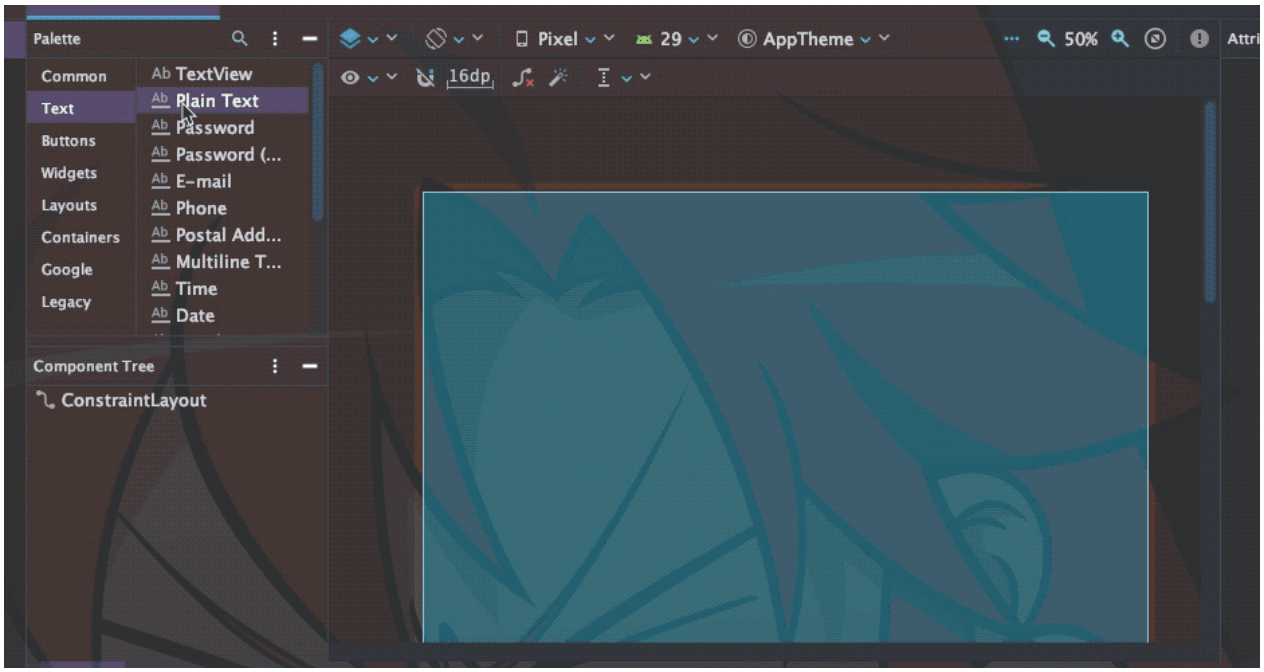


margin

Supprimez le **TextView**

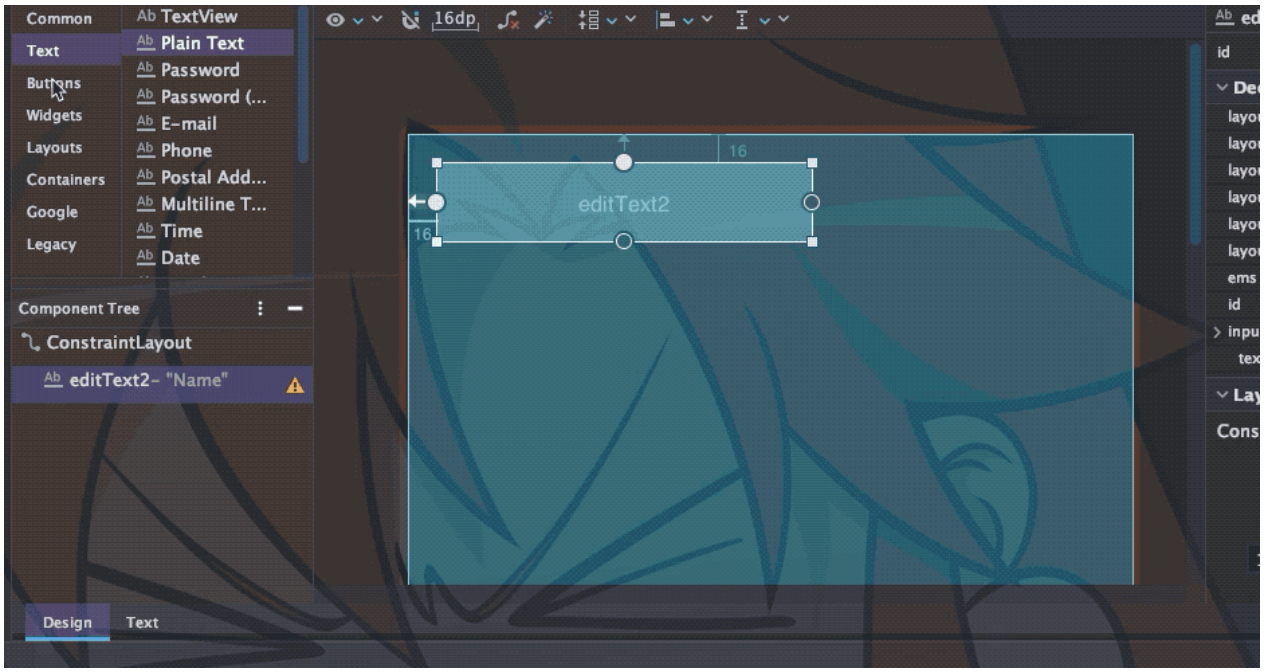


et recréez un Plain Text ainsi que 2 contraintes



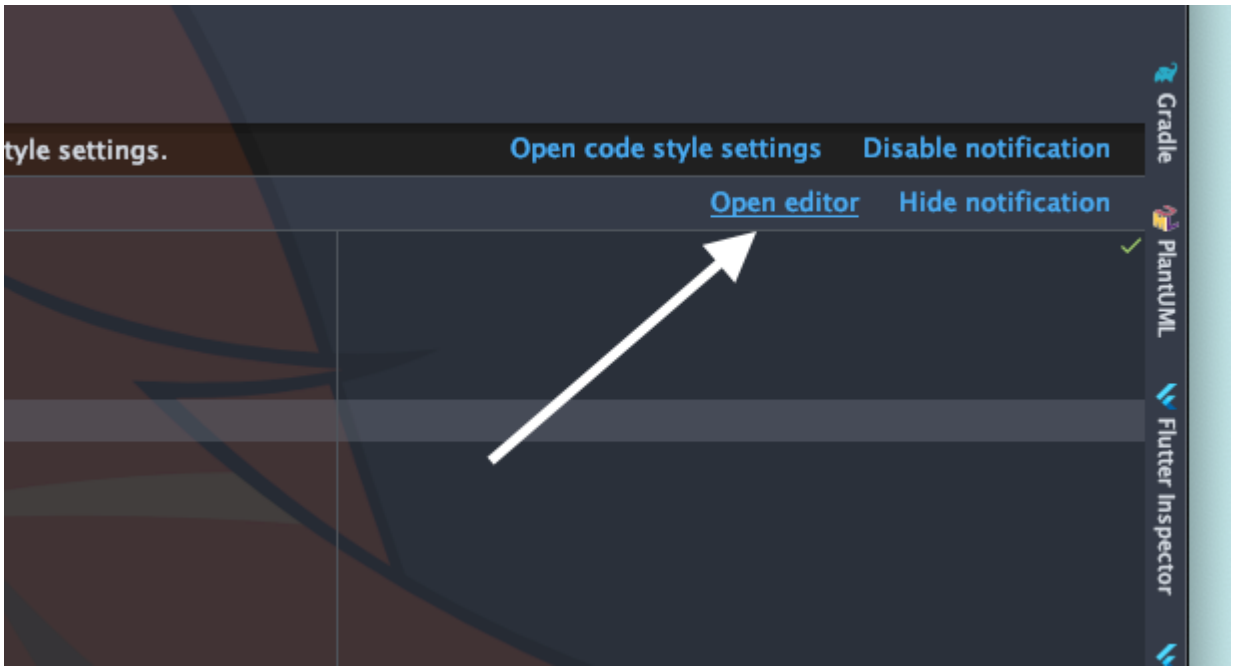
Ajout d'un bouton

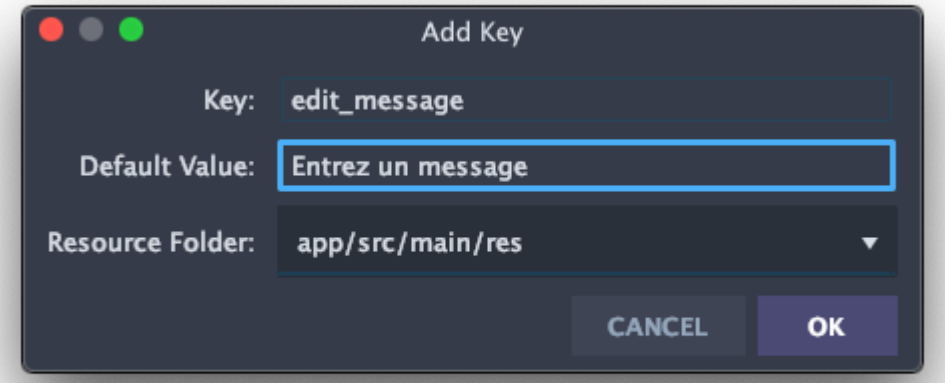
Puis à la manière de SceneBuilder ajoutez un bouton



Changeons les "UI Strings"

Dans le dossier res allez au fichier strings.xml et cliquez sur Open Editor et ajoutez une string pour notre champ text en cliquant sur **+**

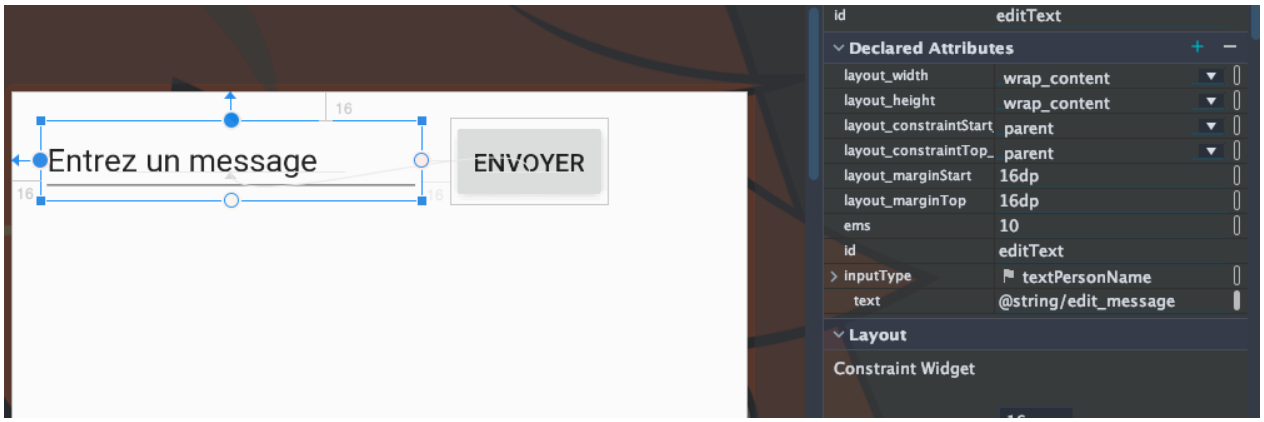




puis faites la même chose pour le bouton:

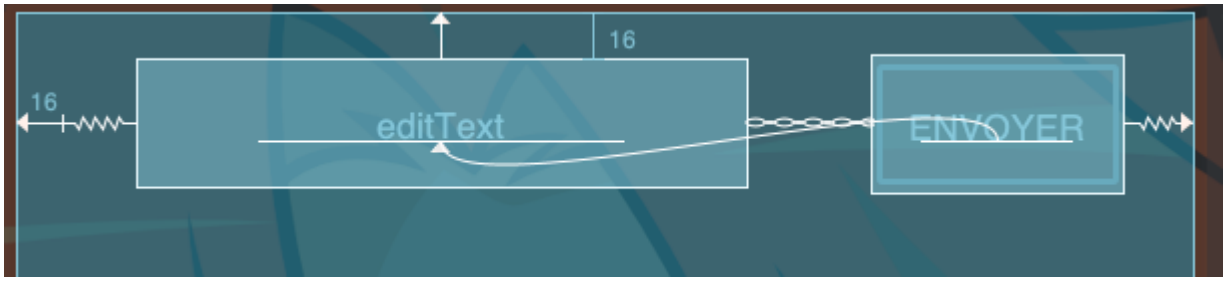
- **key** = button_send
- **Default value** = Envoyer

Faites le lien avec nos éléments. Dans le fichier activity_main.xml, après avoir sélectionné le champ text allez dans la colonne de droite et supprimez la valeur de **text** qui doit être Name. Puis cliquez sur le petit bouton qui affiche **Pick a Resource** quand on le survole. Une fenêtre s'ouvre et sélectionnez notre string `edit_message`. Enfin, faites la même chose pour le bouton.

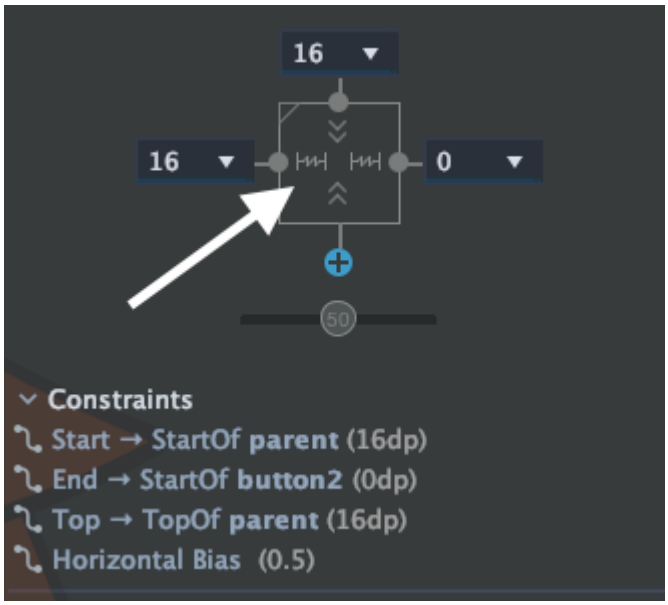


Rendons le champ adaptable

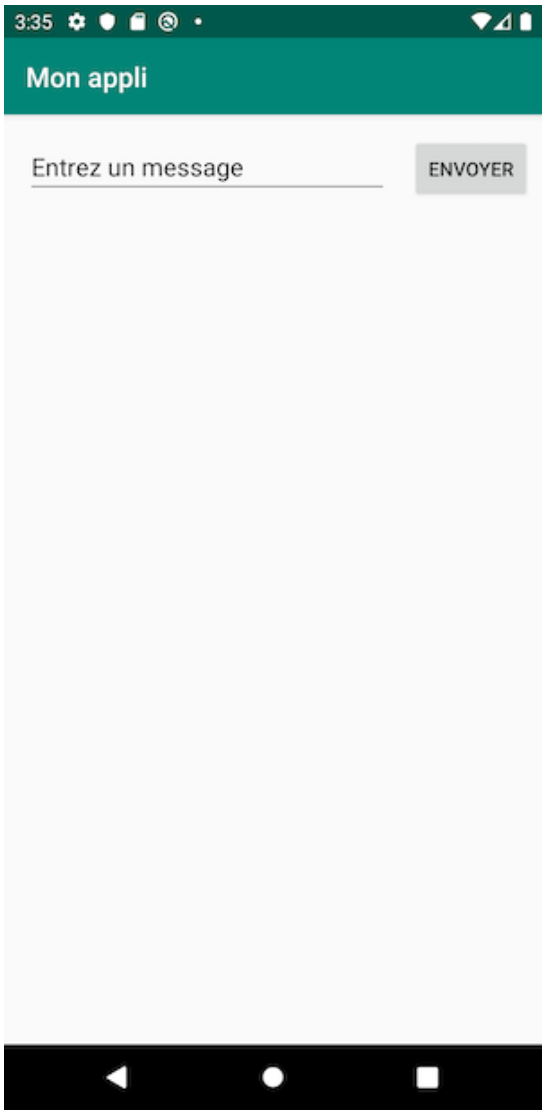
Sélectionnez les deux objets, clique droit et **Chains > Create Horizontal Chain** .



Réglez la marge droite du champ à 16dp et cliquez sur la contrainte en la réglant sur **Match Constraint**



Testez votre appli…



Ajoutons une action

Dans la `MainActivity.java` ajoutez la méthode `sendMessage`

```
public void sendMessage(View view) {
        // Notre action ici
    }
```

Dans le layout sélectionnez le bouton et dans `onClick` déroulez le menu jusqu'à notre nouvelle méthode.

Construisons une `Intent`

Un `Intent` est un objet qui fournit une liaison d'exécution entre des composants distincts, tels que deux activités. L'`Intent` représente l'intention d'une application de faire quelque chose. Vous pouvez utiliser des intentions pour une grande variété de tâches, mais dans cette leçon, votre intention démarre une autre activité.

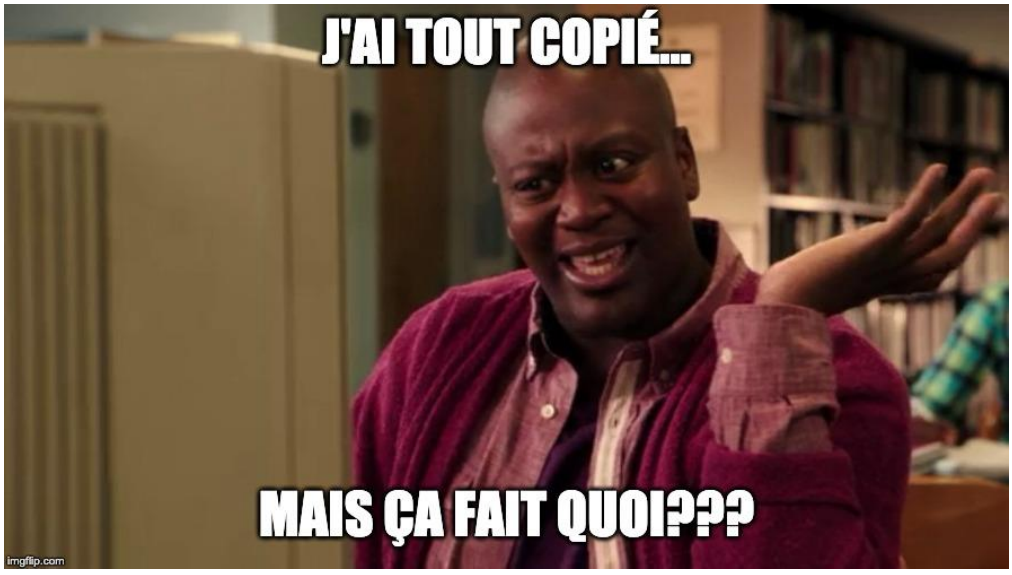
Dans MainActivity, ajoutons une constante EXTRAT_MESSAGE et la méthode sendMessage(). Cette constante est utilisée comme clé d'identification afin d'éviter les conflits si plusieurs applications interagissent.

```
public class MainActivity extends AppCompatActivity {
public static final String EXTRA_MESSAGE = "com.afpa.myapplication.MESSAGE";
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    public void sendMessage(View view) {
        Intent intent = new Intent(this, DisplayMessageActivity.class);
        EditText editText = (EditText) findViewById(R.id.editText);
        String message = editText.getText().toString();
        intent.putExtra(EXTRA_MESSAGE, message);
        startActivity(intent);
    }
```

Ajoutez les imports que IntelliJ vous propose.

Nécessairement `DisplayMessageActivity` va rester en rouge tant que nous ne l'aurons pas créé.

Que venons-nous d'écrire ?



- Le constructeur d'`Intent` prend deux paramètres, un `context` et une classe. La classe est la classe vers où nous allons nous diriger en cliquant sur le bouton.
- La méthode `putExtra` ajoute la valeur du champ EditText à l'intention.
- La méthode `startActivity()` démarre une instance de `DisplayMessageActivity` qui est spécifiée dans l'Intention.

Il ne nous reste qu'à créer notre classe.

Créons notre deuxième activité

C'est parti, clic droit sur app, **New > Activity > Empty Activity** et complétez le champ **Activity Name** avec `DisplayMessageActivity`.

Intellij va faire le travail pour nous :

- Créer le fichier DisplayMessageActivity
- Créer le layout activity_display_message.xml
- Ajouter les éléments dans AndroidManifest.xml

Dans le layout ainsi généré ajoutez un `TextView` que vous pourrez personnaliser avec ses attirbuts dans la colonne de droite (n'oubliez pas les contraintes).

Enfin, ajoutons les méthodes à notre classe.

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

        // Récupère les infos de Intent
        Intent intent = getIntent();
        String message = intent.getStringExtra(MainActivity.EXTRA_MESSAGE);

        // Trouve le TextView et modifie le texte.
        TextView textView = findViewById(R.id.textView);
        textView.setText(message);
    }
```

Attention aux imports

Pour finir, une flèche retour…

Ouvrez `AndroidManifest.xml`, trouvez `<activity android:name=".DisplayMessageActivity">` et modifiez son contenu par:

```
<activity android:name=".DisplayMessageActivity"
          android:parentActivityName=".MainActivity">
    <!-- Le tag meta-data est requis uniquement si vous utilisez l'api15 et
en dessous -->
    <meta-data
        android:name="android.support.PARENT_ACTIVITY"
        android:value=".MainActivity" />
</activity>
```

Lancez l'app…