TP_B1 : INITIATION IHM ANDROID

Objectif du TP :

- Principe de l'activity sous Android
- Principe du MVC sous Android

CE QU'IL FAUT SAVOIR ET COMPRENDRE :

- **Une activité (Activity) est une Vue de notre future application.**
- **La classe MainActivity hérite de la classe Activity**
- **Chaque Activity est indépendante**
- **Toute activité possède au moins une méthode OnCreate**

CRÉATION DE NOTRE PREMIER TEXTVIEW

Recopier les différentes instructions suivantes CONSCIENCIEUSEMENT

Nous voulons que notre application souhaite la bienvenue aux utilisateurs en disant « Bienvenue à LansLeVillage ! ».

Pour cela, nous allons avoir besoin d'un champ "texte". Sur Android, un champ de ce type est un **TextView** (sur Android, tous les éléments sont basés sur la classe **View**).
Nous créons donc notre TextView grâce à la ligne suivante :

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

Il faut pour cela importer la bibliothèque correspondante :

```
import android.widget.TextView;
```

On applique un texte à notre TextView :

```
tv.setText("Bienvenue à LansLeVillage !");
```

Enfin, on applique le TextView à la vue courante :

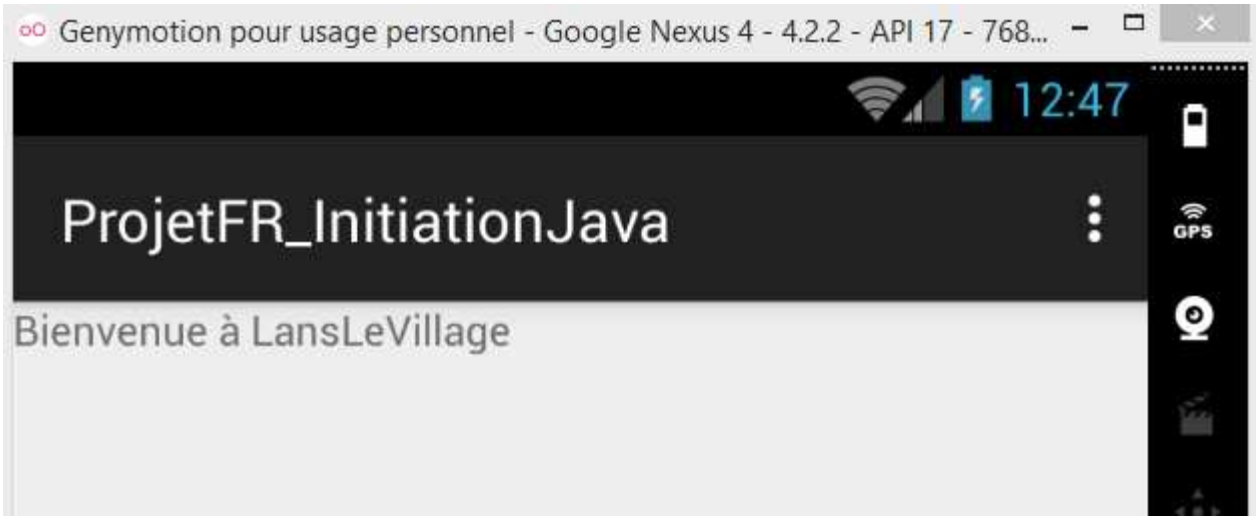
```
setContentView(tv);
```

Vous remarquerez la présence de la ligne suivante :

```
super.onCreate(savedInstanceState);
```

Elle permet de dire à Android comment lancer l'application (on appelle la méthode onCreate de la classe parente en passant le paramètre *savedInstanceState*, qui est le statut précédent de l'application ➔ *on y reviendra ! sur cette notion de statut)*

Lancer votre application sur votre téléphone en faisant Run sur la classe MainActivity

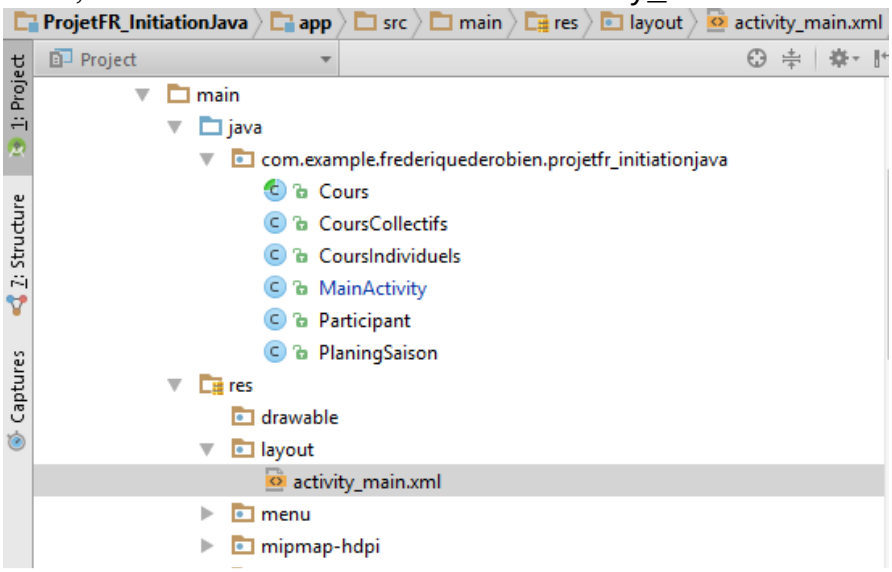


BRAVO ! mais il ne FAUT SURTOUT PAS FAIRE comme cela !

L'organisation du code d'Android est basé sur le modèle MVC

1) Gestion de la VUE dans un fichier à part (`activity_main.xml`)

Lors des développements Android, il faut SEPARER les différents éléments :
soit ici la déclaration des éléments de la vue (TextView) de son exécution.
Aussi, il existe un fichier xml nommé `activity_main.xml`

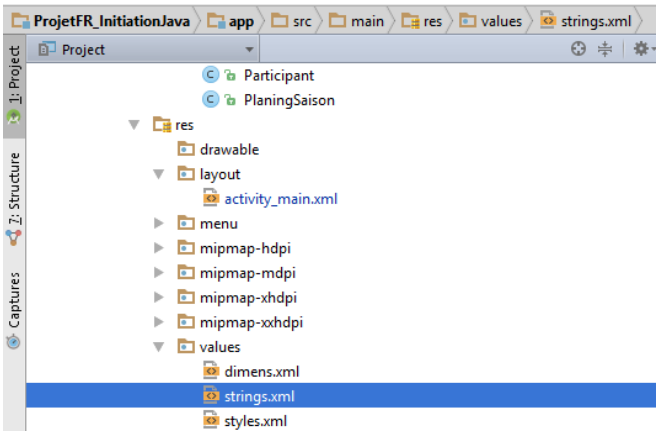


Vous constatez que dans ce fichier, il y a déjà un TextView pour afficher `HelloWorld !`

Lancer votre application sur votre AVD(Android Virtual Device) en faisant Run sur la classe MainActivity. Supprimer le code précédemment ajouté…

Comment dans le code, fait-il le lien entre le textView du fichier `activity_main.xml` et le fichier `MainActivity.java`

2) Gestion des DONNEES dans un fichier à part (`strings.xml`)



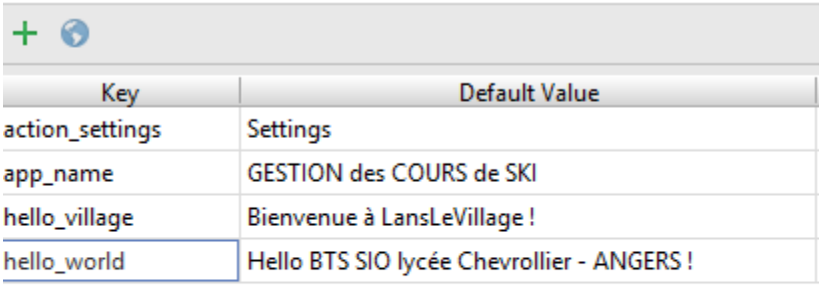
Vous constatez que la chaine HelloWorld est référencée dans le fichier strings.xml

Changer le Hello world ! en Hello BTS SIO Lycée Chevrollier - ANGERS

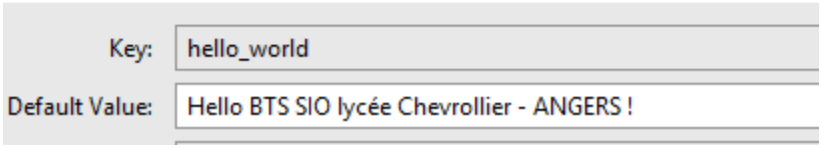
```
<string name="hello_world">Hello BTS SIO Lycée Chevrollier - ANGERS!</string>
```

cliquer sur *open editor* du fichier strings.xml, vous voyez bien ce couple clé/valeur.

Sur la « terre », sélectionner le **français** et changer Hello par *Bienvenue* BTS SIO….



Key	Default Value
action_settings	Settings
app_name	GESTION des COURS de SKI
hello_village	Bienvenue à LansLeVillage !
hello_world	Hello BTS SIO lycée Chevrollier - ANGERS !



Lancer votre application sur votre AVD en faisant Run sur la classe MainActivity.

Quelle langue est choisie ici ? Pourquoi ?

Comment dans le code, fait-il le lien entre le textView du fichier `activity_main.xml` et le fichier `strings.xml`

ⓘ `Chaque String a un identifiant et une valeur (couple clé/valeur)`
`Key(clé) : "@+id/app_name"  Valeur : "@string/app_name"`

3) Gestion du CONTROLEUR ?

Qui est le controleur ici ?

Faites un schéma qui explique le modèle MVC ?

TRAVAIL à faire :

- Créer une nouvelle zone de Texte afin d'afficher « Bienvenue à LansLeVillage ! » en respectant le modèle MVC.

Problème : vos 2 textView se chevauchent ?

Il faut gérer les Layout « mise en page de notre activity ». Actuellement, vos 2 textView sont dans une `relativeLayout` (cadre qui prend toute la largeur et hauteur de la configuration de votre AVD)

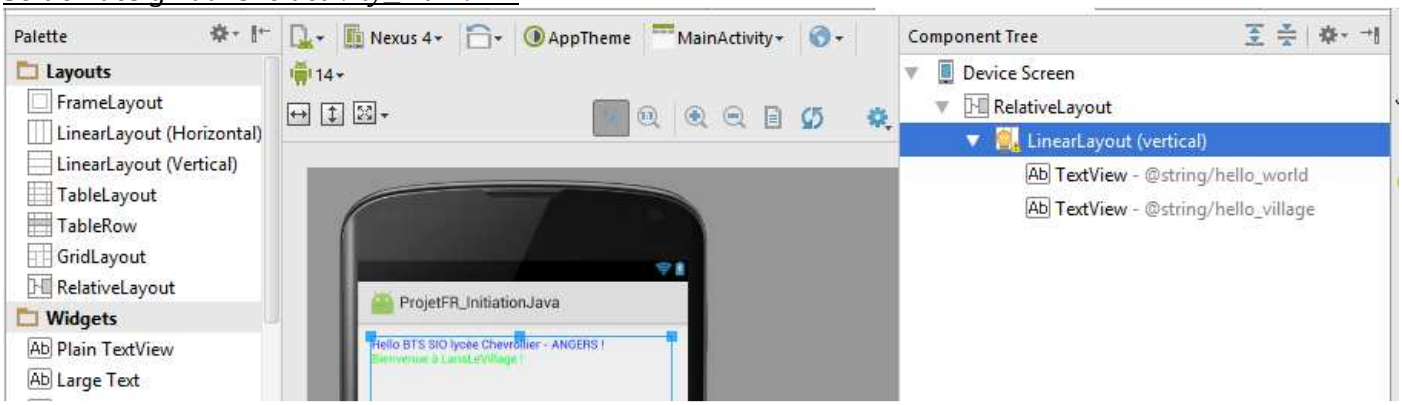
A l'intérieur de ce relativeLayout, nous allons organiser les éléments graphiques en choisissant un `LinearLayout` de manière **verticale** (orientation) pour que mes éléments graphiques *soit les uns en dessous des autres.*

2 manières de modifer les choses :

Soit en text dans le activity_main.xml

```
<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:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
tools:context=".MainActivity">

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

        <TextView android:text="@string/hello_world"
android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#ff0b29ff" />

        <TextView android:text="@string/hello_village"
android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#ff05ff20" />
    </LinearLayout>

</RelativeLayout>
```

Soit en design dans le activity_main.xml



✍ Que signifie **android:layout_width="match_parent" ?**

TRAVAIL à faire :

- Changer l'affichage du nom du projet en mettant « gestion des cours de SKI » au lieu du « ProjetFR_InitiationJava »