Présentation de l'environnement

1. Objectifs

Les objectifs de ce chapitre sont les suivants :

* Savoir envoyer des données à l'ihm en utilisant findViewById
* Savoir envoyer des données à l'ihm en utilisant le viewBinding
* Savoir envoyer des données à l'ihm en utilisant le dataBinding

1. Présentation

Avoir une ihm c'est bien, avoir une ihm dynamique dont le contenu change en fonction des besoins et des actions de l'utilisateur c'est mieux. Pour se faire il existe différentes méthodes. Ces différentes méthodes présenté dans ce chapitre.

1. La méthode findViewById
   1. Présentation

La fonction findViewById est utilisée dans le contrôleur et elle permet de récupérer UN objet représentant UN élément graphique d'une IHM dans le but d’interagir avec celui-ci.

* 1. Mise en place

Dans cette mise en place, la fonction findViewById sera utilisée pour interagir avec l'ihm dont le code est le suivant :

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

​

<TextView

android:id="@+id/tv\_titre"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

/>

​

<EditText

android:id="@+id/et\_nom"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Nom"

app:layout\_constraintTop\_toBottomOf="@id/tv\_titre"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<EditText

android:id="@+id/et\_prenom"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Prénom"

app:layout\_constraintTop\_toBottomOf="@id/et\_nom"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<EditText

android:id="@+id/et\_numero"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Numéro"

app:layout\_constraintTop\_toBottomOf="@id/et\_prenom"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<EditText

android:id="@+id/et\_rue"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Rue"

app:layout\_constraintTop\_toBottomOf="@id/et\_numero"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<EditText

android:id="@+id/et\_ville"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Ville"

app:layout\_constraintTop\_toBottomOf="@id/et\_rue"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<EditText

android:id="@+id/et\_code\_postal"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Code postal"

app:layout\_constraintTop\_toBottomOf="@id/et\_ville"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<Button

android:id="@+id/btn\_envoyer"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toBottomOf="@+id/et\_code\_postal"

app:layout\_constraintLeft\_toLeftOf="parent"

android:text="Envoyer"/>

​

</androidx.constraintlayout.widget.ConstraintLayout>

Voici comment récupérer des objets représentant chacun un élément de l'ihm, puis comment leur valeur est exploitée :

package fr.eni.t2

​

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.util.Log

import android.widget.Button

import android.widget.EditText

import android.widget.TextView

import android.widget.Toast

​

class MainActivity : AppCompatActivity()

{

lateinit var tvTitre:TextView

lateinit var etNom:EditText

lateinit var etPrenom:EditText

lateinit var etNumero:EditText

lateinit var etRue:EditText

lateinit var etVille:EditText

lateinit var etCodePostal:EditText

lateinit var btnEnvoyer:Button

​

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(R.layout.activity\_main)

​

tvTitre = findViewById(R.id.tv\_titre)

etNom = findViewById<EditText>(R.id.et\_nom)

etPrenom = findViewById<EditText>(R.id.et\_prenom)

etNumero = findViewById<EditText>(R.id.et\_numero)

etRue = findViewById<EditText>(R.id.et\_rue)

etVille = findViewById<EditText>(R.id.et\_ville)

etCodePostal = findViewById<EditText>(R.id.et\_code\_postal)

btnEnvoyer = findViewById<Button>(R.id.btn\_envoyer)

​

tvTitre.text = "Formulaire 53D";

​

btnEnvoyer.setOnClickListener {

Log.i("ACOS","Bonjour ${etPrenom} " +

"${etNom.text}, vous habitez à ${etNumero.text} " +

"${etRue.text} ${etCodePostal} ${etVille.text} ")

}

}

}

Dans l'exemple précèdent la valeur d'un EditText nommé etNom est récuperée pour ensuite l'afficher dans un log mais il aurait été aussi possible de modifier cet EditText comme ceci :

etNom.text = "Nouvelle valeur"

* 1. Conclusion

On remarquera dans cette mise en place le nombre important de fois qu'est utilisé findViewById(), cela peut devenir fastidieux à partir du moment où l'IHM se complexifie.

1. La méthode ViewBinding
   1. Présentation

Permet de récupérer simplement  un objet dans le contrôleur contenant TOUS les éléments graphiques sous forme d'objets.

* 1. Configuration

Dans le fichier build.gradle :

android {

...

buildFeatures {

viewBinding = true

}

}

* 1. Mise en place

Dans cette mise en place, la méthode viewBinding sera utilisée pour interagir avec l'ihm dont le code est le suivant :

<?xml version="1.0" encoding="utf-8"?>

<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

<TextView

android:id="@+id/tv\_titre"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

/>

<EditText

android:id="@+id/et\_nom"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Nom"

app:layout\_constraintTop\_toBottomOf="@id/tv\_titre"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

<EditText

android:id="@+id/et\_prenom"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Prénom"

app:layout\_constraintTop\_toBottomOf="@id/et\_nom"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

<EditText

android:id="@+id/et\_numero"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Numéro"

app:layout\_constraintTop\_toBottomOf="@id/et\_prenom"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

<EditText

android:id="@+id/et\_rue"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Rue"

app:layout\_constraintTop\_toBottomOf="@id/et\_numero"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

<EditText

android:id="@+id/et\_ville"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Ville"

app:layout\_constraintTop\_toBottomOf="@id/et\_rue"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

<EditText

android:id="@+id/et\_code\_postal"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="Code postal"

app:layout\_constraintTop\_toBottomOf="@id/et\_ville"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

<Button

android:id="@+id/btn\_envoyer"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toBottomOf="@+id/et\_code\_postal"

app:layout\_constraintLeft\_toLeftOf="parent"

android:text="Envoyer"/>

</androidx.constraintlayout.widget.ConstraintLayout>

Voici comment récupérer un objet contenant les objets représentant chacun un élément de l'ihm, puis comment leur valeur est exploitée :

package fr.eni.t2

​

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.util.Log

import fr.eni.t2.databinding.ActivityMainBinding

​

class MainActivity : AppCompatActivity()

{

private lateinit var bd: ActivityMainBinding

​

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

bd = ActivityMainBinding.inflate(layoutInflater)

setContentView(bd.root)

​

bd.tvTitre.text = "Formulaire 53D";

​

bd.btnEnvoyer.setOnClickListener {

Log.i("ACOS","Bonjour ${bd.etPrenom.text} " +

"${bd.etNom.text}, vous habitez à ${bd.etNumero.text} " +

"${bd.etRue.text} ${bd.etCodePostal.text} ${bd.etVille.text} ")

}

}

}

Dans l'exemple précèdent la valeur d'un EditText nommé etNom est récuperée pour ensuite l'afficher dans un log mais il aurait été aussi possible de modifier cet EditText comme ceci :

Bd.etNom.text = "Nouvelle valeur"

* 1. Conclusion

On remarquera dans cette mise en place qu'un seul objet est récupéré pour manipuler tous les éléments de l'ihm. Cette méthode est l'une de celle à privilégier (avec le dataBinding) grâce à sa simplicité.

1. La méthode dataBinding
   1. Présentation

Permet d'injecter des données dans l'IHM et d'utiliser le ViewBinding.

* 1. Configuration

Dans le fichier build.gradle :

android {

...

buildFeatures {

dataBinding = true

}

* 1. Mise en place

Dans cette mise en place, la méthode viewBinding sera utilisée pour interagir avec l'ihm dont le code est le suivant, attention, une ihm doit obligatoirement être englobée par la balise <layout> pour pouvoir utiliser le dataBinding.

<?xml version="1.0" encoding="utf-8"?>

<layout>

<data>

<variable

name="personne"

type="fr.eni.t2.Personne" />

​ <variable

name="titre"

type="String" />

</data>

​<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

tools:context=".MainActivity">

​

<TextView

android:id="@+id/tv\_titre"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

app:layout\_constraintLeft\_toLeftOf="parent"

app:layout\_constraintRight\_toRightOf="parent"

app:layout\_constraintTop\_toTopOf="parent"

android:text="@{titre}"

/>

​

<EditText

android:id="@+id/et\_nom"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@={personne.nom}"

app:layout\_constraintTop\_toBottomOf="@id/tv\_titre"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<EditText

android:id="@+id/et\_prenom"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@={personne.prenom}"

app:layout\_constraintTop\_toBottomOf="@id/et\_nom"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<EditText

android:id="@+id/et\_numero"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@={personne.numero}"

app:layout\_constraintTop\_toBottomOf="@id/et\_prenom"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<EditText

android:id="@+id/et\_rue"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@={personne.rue}"

app:layout\_constraintTop\_toBottomOf="@id/et\_numero"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<EditText

android:id="@+id/et\_ville"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@={personne.ville}"

app:layout\_constraintTop\_toBottomOf="@id/et\_rue"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<EditText

android:id="@+id/et\_code\_postal"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:text="@={personne.codePostal}"

app:layout\_constraintTop\_toBottomOf="@id/et\_ville"

app:layout\_constraintLeft\_toLeftOf="parent"

/>

​

<Button

android:id="@+id/btn\_envoyer"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

app:layout\_constraintTop\_toBottomOf="@+id/et\_code\_postal"

app:layout\_constraintLeft\_toLeftOf="parent"

android:text="Envoyer"/>

​</androidx.constraintlayout.widget.ConstraintLayout>

​</layout>

Voici comment récupérer un objet contenant les objets représentant chacun un élément de l'ihm, puis comment leur valeur est exploitée, il est aussi à noter que dans le code ci-après on injecte aussi des données dans l'ihm.

package fr.eni.t2

​

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import android.util.Log

import fr.eni.t2.databinding.ActivityMainBinding

​

data class Personne(var nom:String="Nom",var prenom:String="Prénom",var numero:String="Numéro",var rue:String="Rue",var codePostal:String="Code Postal",var ville:String="Ville")

​

class MainActivity : AppCompatActivity()

{

private lateinit var bd: ActivityMainBinding

var unePersonne:Personne = Personne()

​

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

bd = ActivityMainBinding.inflate(layoutInflater)

setContentView(bd.root)

​

bd.tvTitre.text = "Formulaire 53D";

​

bd.personne = unePersonne

​

bd.btnEnvoyer.setOnClickListener {

Log.i("ACOS","Bonjour ${unePersonne.nom} " +

"${unePersonne.prenom}, vous habitez à ${unePersonne.numero} " +

"${unePersonne.rue} ${unePersonne.codePostal} ${unePersonne.ville} ")

}

}

}

* 1. Conclusion

On remarquera dans cette mise en place qu'un seul objet est récupéré pour manipuler tous les éléments de l'ihm. Et que l'on peut injecter des données dans l'ihm. Par la suite, il sera question de combiner la méthode du dataBinding et les ViewModel afin d'arriver à un modèle architectural très performant en terme de praticité et de puissance. Cette méthode est l'une de celle à privilégier (avec le viewBinding) grâce à sa simplicité.

1. ViewBinding vs DataBinding
   1. ViewBinding

Liaison des vues au code

* 1. DataBinding

Liaison des données aux vues + ViewBinding (liaison des vues au code)

Il y a trois différences importantes

* Avec le ViewBinding, les mises en page n'ont pas besoin de balise <layout>
* Vous ne pouvez pas utiliser le Viewbinding pour lier des mises en page avec des données au format XML

Les principaux avantages du ViewBinding sont la rapidité et l'efficacité. Il a un temps de génération plus court car il évite les problèmes de  performances associés à la liaison de données en raison des processeurs d'annotations affectant le temps de génération.

En bref, il n'y a rien que le ViewBinding puisse faire que le DataBinding ne puisse faire (bien qu'au prix de temps d'une génération plus longue) et il y a beaucoup de liaisons que le ViewBinding ne peut pas faire.

* Il est possible d'activer les deux dans le build.gradle afin de pouvoir utiliser le dataBinding sur certaines ihms grâce à la balise layout et afin de pouvoir utiliser le viewBinding par défaut sur les autres ihms.