PROGRAMMATION DE COMPOSANTS MOBILES (ANDROID)

Wieslaw Zielonka

ViewBinding dans RecyclerView.Adapter

Rappel

ViewBinding permet s'affranchir de `findViewById()`. Dans une activité c'est simple.

Ajouter dans la section android de `build.gradle` (Module) :

```
buildFeatures {
    viewBinding true
}
```

Si le fichier layout est `activity_main.xml` et contient un TextView dont id est **`tx`** alors :

```
private val binding  by lazy { ActivityMainBinding.inflate(layoutInflater)

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)

    setContentView(binding.root)

et à partir de maintenant
```

binding.tx -> la référence vers le TextView tx

view binding dans RecyclerView.Adapter

Dans le `RecyclerView.Adapter` on crée une nouvelle `View` à chaque appel de la méthode `onCreateViewHolder()` et on remplit le contenu dans `onBidViewHolder()`.

Donc dans on `bindViewholder()` on utilise les appels à `findViewById()` pour retrouver les View qui forme un item de la liste à afficher.

view binding dans RecyclerView.Adapter

On suppose que le fichier layout pour un item dans `RecyclerView` est par exemple `author_item_layout.xml`.

ViewBinding produira automatiquement la classe
`AuthorItemLayoutBinding`
une classe dérivée de `ViewBinding`.

`Etape 1 )`

Dans le `RecyclerView.Adapter` on modifie la définition de holder :

```
class VH(val binding: AuthorItemLayoutBinding) : RecyclerView.ViewHolder( binding.root )
```

le holder contient maintenant une propriété supplémentaire : `binding`

view binding dans RecyclerView.Adapter

Etape 2) dans onCreateViewHolder() :

```
override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): VH {

    /* création de  view et de binding en même temps */
    val binding = AuthorItemLayoutBinding
                  .inflate( LayoutInflater
                  .from(parent.getContext()), parent, false )
```

A partir de ce moment **binding.id_de_view** permet d'accéder à une view dans le layout. Par exemple si le layout contient un CheckBox dont id est **check** alors on pourra installer un listener :

```
    binding.check.setOnClickListener {  ....  }

    on termine onCreateViewHolder() avec :

    return VH( binding )
}
```

view binding dans RecyclerView.Adapter

Etape 3) onBindViewHolder() :

```
override fun onBindViewHolder(holder: VH, position: Int) {

    /* holder.itemView c'est toujours la view à la racine
      * dans le fichier xml */
```

Maintenantant avec **holder.binding.id_de view** on peut récupérer n'importe quel view dans le layout, par exemple si le layout contient deux TextView dont id est **nom** et **prénom** :

```
    holder.binding.prenom.text = ....
    holder.binding.nom.text =  ....
```

permet de mettre de valeur à afficher dans ces deux TextView ou encore mieux s'il y a beaucoup de View à remplir :

```
with( holder.binding ){
    prénom.text = ....
    nom.text = ....
}
```