



Mikael Desertot  
mikael.desertot@uphf.fr

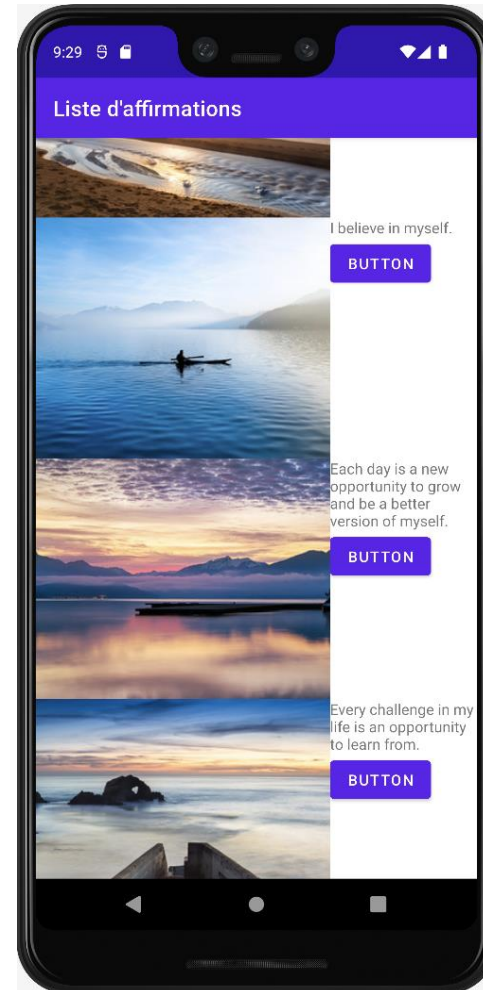
# Transformation de l'exemple RecyclerView

A decorative green L-shaped line consisting of a vertical segment on the left and a horizontal segment extending to the right, positioned below the title.

Mise en œuvre des Fragments

# Objectif

- L'application de base affirmation (transformée) disposait d'une liste d'image+texte+bouton.
- Un clic sur un bouton ouvrait un AlertDialog
- Maintenant le clic devra ouvrir une nouvelle vue (fragment) qui affichera seulement le texte de l'item cliqué



# Gestion du build

---

- Aidez vous des fichier fournis pour compléter vos build.gradle. A noter :
  - Permet de gérer la navigation entre fragments  
implementation **'androidx.navigation:navigation-fragment:2.5.3'**  
implementation **'androidx.navigation:navigation-ui:2.5.3'**
  - Autorise la génération des objets binding des layout (aa\_bb.xml → AaBbBinding)

```
buildFeatures {  
    viewBinding = true  
}
```

# Layouts

---

- Dupliquer activity\_main.xml et renommer la copie en fragment\_list.xml par exemple
- Modifier activity\_main.xml pour gérer les fragments

```
<FrameLayout 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">
```

```
<androidx.fragment.app.FragmentContainerView
  android:id="@+id/nav_host_fragment"
  android:name="androidx.navigation.fragment.NavHostFragment"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  app:defaultNavHost="true"/>
```

```
</FrameLayout>
```

# Layouts

---

- fragment\_list.xml deviendra la fragment principal et reprendra le layout de la liste existant

```
<FrameLayout 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=".ListFragment">
```

```
<androidx.recyclerview.widget.RecyclerView
  android:id="@+id/recycler_view"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  app:layoutManager="LinearLayoutManager"
  android:scrollbars="vertical"/>
```

```
</FrameLayout>
```

# Activité et Fragments

---

- Dupliquez MainActivity qui gèrait la liste et renommez la copie en ListFragment par exemple. Cette activité sera transformée en fragment par la suite
- La MainActivity contiendra juste le code nécessaire pour gérer les fragments.
- Ce code est simple et réutilisable dans d'autres applications
- La spécificité des vues est contenues dans les fragments

# MainActivity.kt

---

```
class MainActivity : AppCompatActivity() {  
  
    private lateinit var navController: NavController  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
  
        val binding = ActivityMainBinding.inflate(layoutInflater)  
        setContentView(binding.root)  
  
        val navHostFragment = supportFragmentManager  
            .findFragmentById(R.id.nav_host_fragment) as NavHostFragment  
  
        navController = navHostFragment.navController  
  
        setupActionBarWithNavController(navController)  
    }  
  
    override fun onSupportNavigateUp(): Boolean {  
        return navController.navigateUp() || super.onSupportNavigateUp()  
    }  
}
```



# ListFragment

---

- Transformer le recycler view du fragment pour intégrer les modifs suivantes
- Très similaire à la version Activity

```
class ListFragment : Fragment() {  
  
    private var _binding: FragmentListBinding? = null  
    private val binding get() = _binding!!  
    private lateinit var recyclerView: RecyclerView  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
    }  
}
```

On hérite de fragment

Objet généré depuis le layout

Rien de spécial !


# ListFragment

---

```
override fun onCreateView(  
    inflater: LayoutInflater,  
    container: ViewGroup?,  
    savedInstanceState: Bundle?,  
): View {
```


Récupération de  
l'arbre des vues

```
    _binding = FragmentListBinding.inflate(inflater, container, false)  
    val view = binding.root  
    return view  
}
```



# ListFragment

---

```
override fun onCreateView(view: View, savedInstanceState: Bundle?) {  
    val myDataset = Datasource().loadAffirmations()  Récupération  
    recyclerView = binding.recyclerView des données  
    recyclerView.adapter = ItemAdapter(requireContext(), myDataset)  
    recyclerView.setHasFixedSize(true)  
}
```

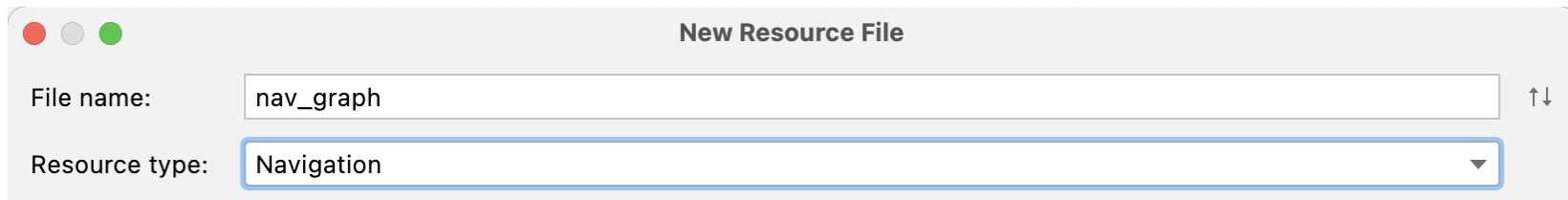
  

```
override fun onDestroyView() {  
    super.onDestroyView()  
    _binding = null  
}
```

# Creation d'un nav\_graph

---

- Clic droit sur res
  - New Android Resource File



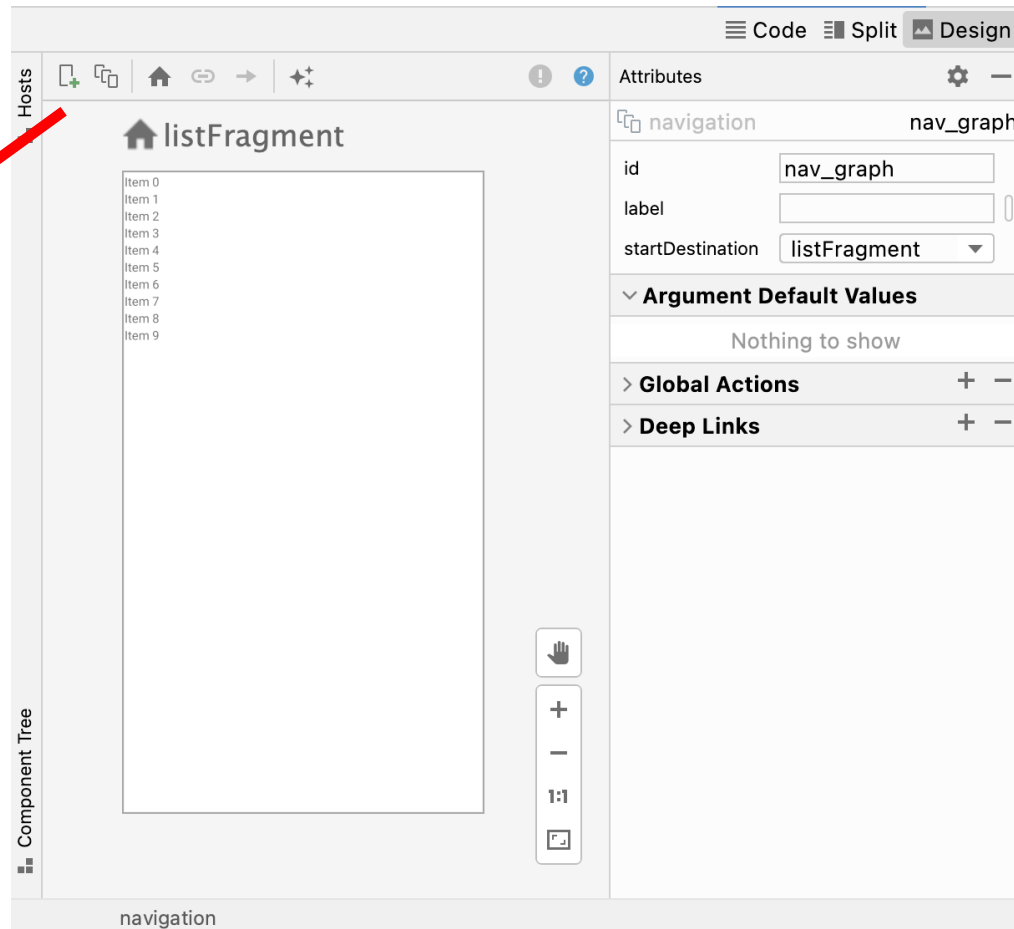
- Ajouter ce navGraph au  
FragmentContainerView

```
app:navGraph="@navigation/nav_graph"/>
```

# nav\_graph.xml

- Permet de gérer l'enchainement des vues (fragments) de l'application
- Outil graphique générant le xml (jetez un œil au xml)

Ajouter un  
fragment



# A partir de la → tester

---

- Seul changement c'est le menu en haut qui montre qu'on a un fragment
- Etape suivant : quand on clic sur le bouton on veut aller à un autre fragment
- Que faut il faire
  - Ajouter un layout pour le nouveau fragment
  - Ajouter un fragment (kt)
  - Faire un lien dans le nav\_graph avec le nouveau fragment, passer la chaine en parametre
  - Modifier le bouton pour changer le AlertDialog en lien vers le nouveau fragment

# Nouveau layout

---

- On s'en fiche c'est simple... juste un texte

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="match_parent"
  android:layout_height="match_parent">


  <TextView
    android:id="@+id/textViewDetail"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"/>
</FrameLayout>
```

# DetailFragment.kt

---

```
class DetailFragment : Fragment() {  
    companion object {  
        val KEY = "sentence"  
    }  
}
```

Définition d'un mot clé  
pour le passage du  
paramètre




```
private var _binding: FragmentDetailBinding? = null
```

```
private val binding get() = _binding!!
```

```
private lateinit var sentence: String
```

```
override fun onCreate(savedInstanceState: Bundle?) {  
    super.onCreate(savedInstanceState)  
    arguments?.let {  
        sentence = it.getString(KEY).toString()  
    }  
}
```

Récupération de la  
valeur d'un paramètre





# DetailFragment.kt

---

## ■ onCreateView évident

```
override fun onCreateView(  
    inflater: LayoutInflater,  
    container: ViewGroup?,  
    savedInstanceState: Bundle?  
): View? {  
  
    _binding = FragmentDetailBinding.inflate(inflater, container, false)  
    val view = binding.root  
    return view  
}
```

# DetailFragment.kt

---

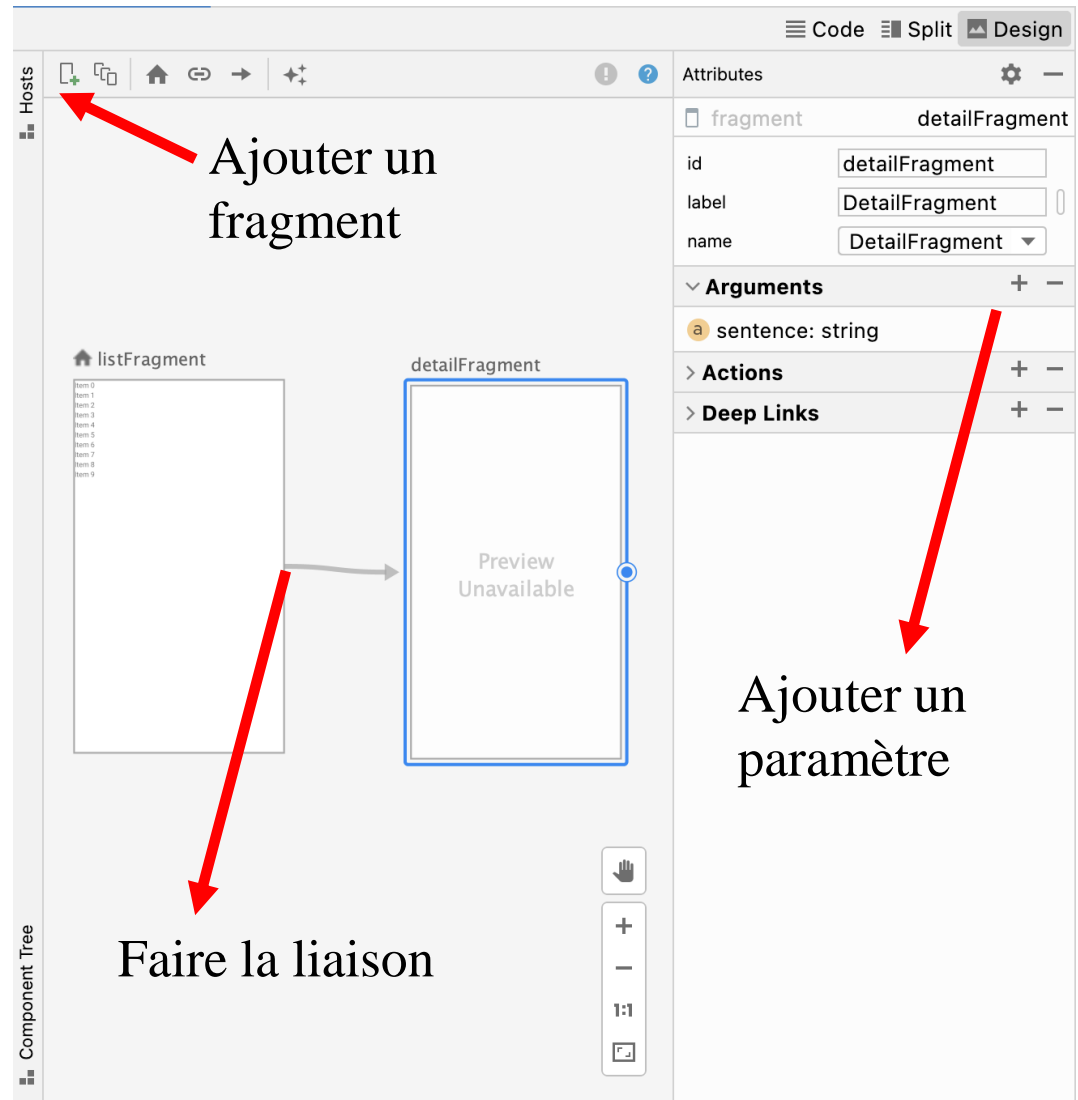
```
override fun onCreateView(view: View, savedInstanceState: Bundle?) {  
    _binding?.textViewDetail?.text = sentence  
}
```

Affecter le texte au  
textView

```
override fun onDestroyView() {  
    super.onDestroyView()  
    _binding = null  
}  
}
```

# Ajout dans la navigation de l'app

- Ajouter le nouveau fragment
- Faire un lien (glisser pour amener la flèche vers le second fragment)
- Gérer le paramètre String



# Modification du bouton

---

```
holder.button.setOnClickListener(){  
    val action = ListFragmentDirections.  
        actionListFragmentToDetailFragment(  
            context.resources.getString(item.stringResourceId))  
    holder.view.findNavController().navigate(action)  
}
```

Généré depuis le nav\_graph

Généré depuis la flèche entre les fragments

Création de l'action

Déclenchement de l'action

# Tester

---

- Vue fragment et navigation

