03-4. Navigation Architecture Component

Android Architecture Component

Principles of Navigation

- **fixed starting destination**
  - starting destination 은 유저가 앱을 런치 했을 때 보이는 화면
  - starting destination 은 back button 에 의해 화면이 전환될 때 앱 내에서 가장 마지막에 보이는 화면

- **navigation stack**
  - 앱의 navigation 상태는 stack 구조를 가져야 함
  - start destination 은 stack 의 bottom 에 위치해야 하며, 현재 유저 화면에 보이고 있는 destination 이 stack 의 top 에 위치해야 함

- **The Up button never exits your app**
  - Up button에 의해 앱이 종료되게 만들면 안됨
  - deep link 에 의해 up button 이 제공된다면, up button 에 의해 다른 앱으로 화면이 전환되게 하지 않음

- **Up and Back are equivalent within your app's task**

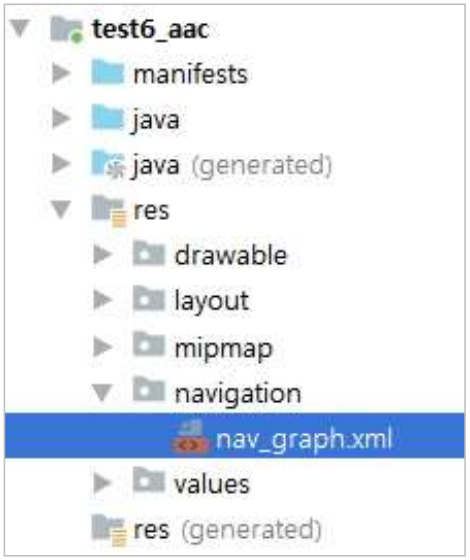
Navigation Components

- Navigation Component 는 화면간 이동을 간편하게 작성하게 하는 JetPack 의 Library 혹은 Tool
- 하나의 main activity에 여러 개의 fragment 로 화면을 설계하는 것을 목적으로 함
- activity가 navigation host가 되고 fragment 가 destination 이 됨
- 여러 개의 activity가 있다면 각 activity 를 위한 navigation graph를 별도로 지정

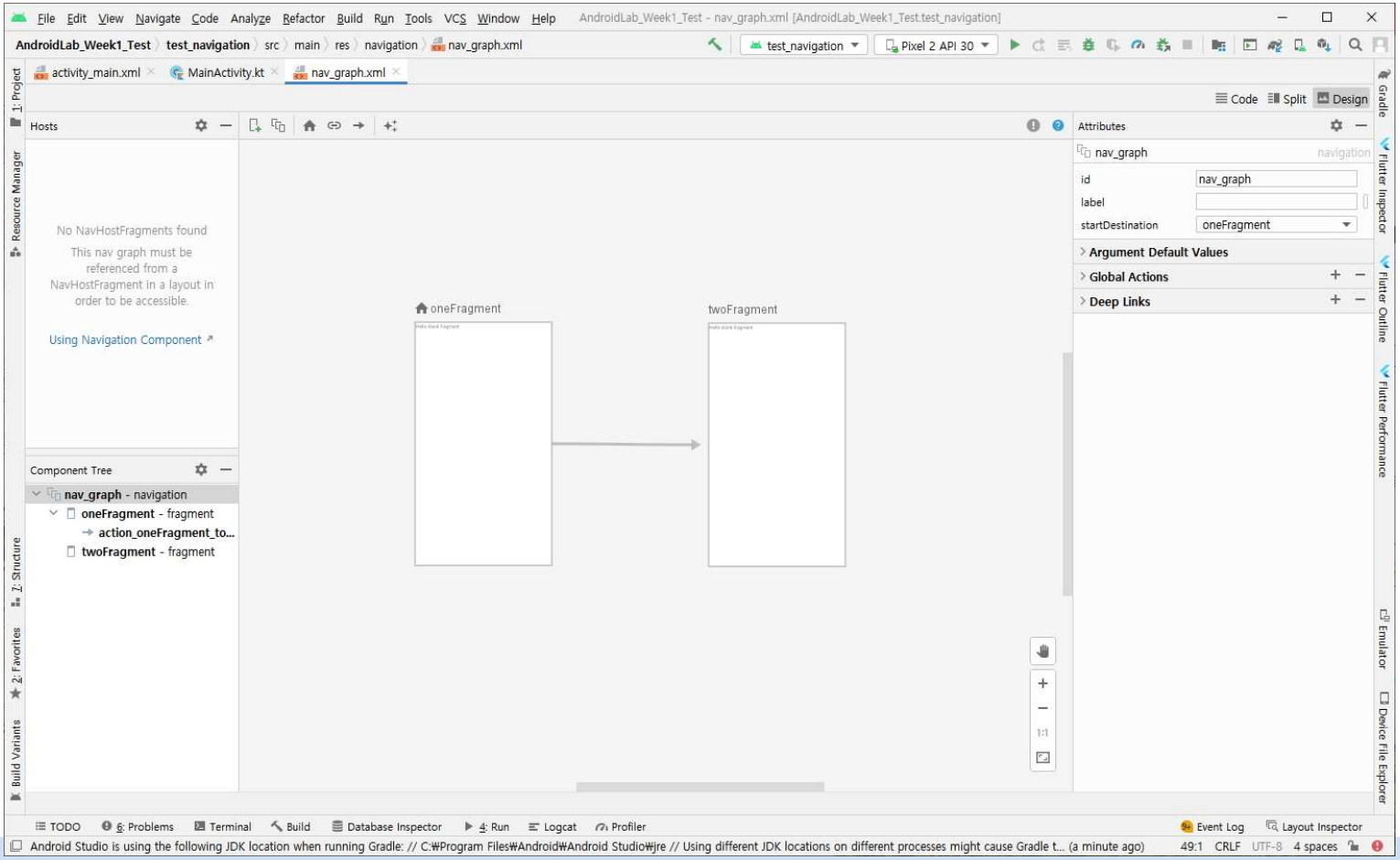
```
implementation 'androidx.navigation:navigation-fragment-ktx:$nav_version'
implementation 'androidx.navigation:navigation-ui-ktx:$nav_version'
```

Navigation Components

•용어 정리
- •Navigation graph : navigation 을 도식화 시킨 화면 (XML)
- •Navigation editor : Android Studio 에서 제공하는 navigation graph 를 위한 Tool
- •Destination : 화면 단위
- •Action : destination 간 연결
- •Argument : destination 간 navigation 시 전달 데이터



Navigation Editor



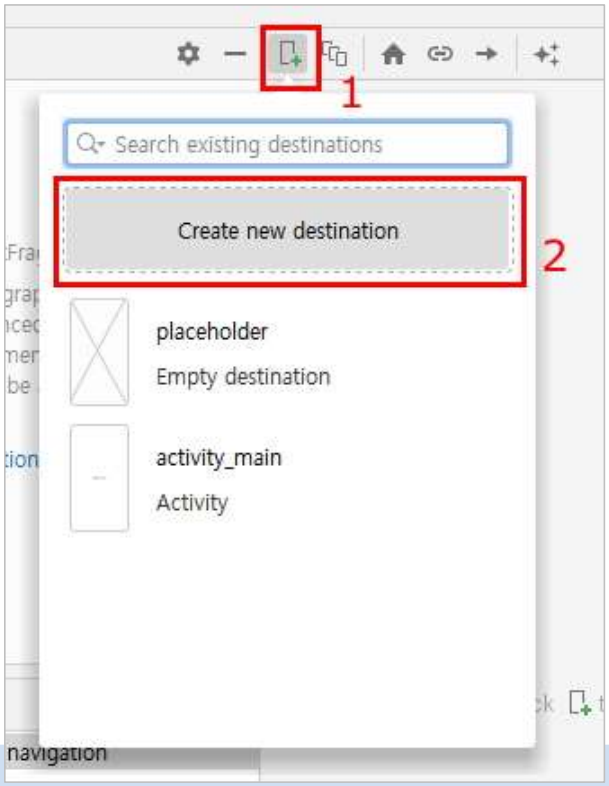
Navigation Graph

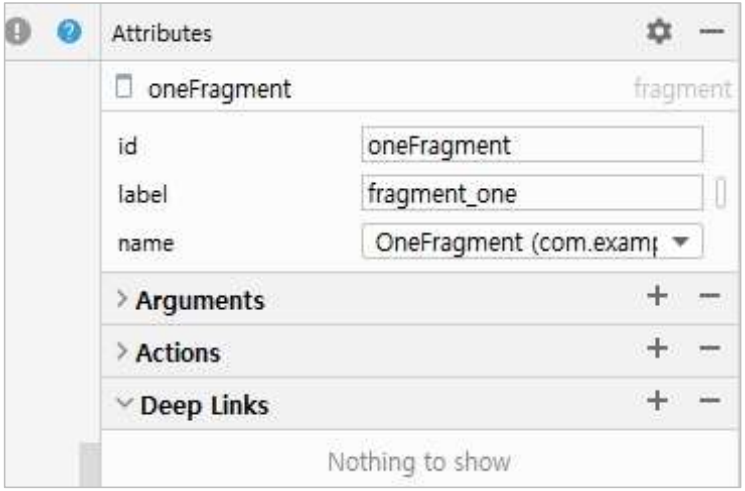
- Navigation 내용이 정의된 XML 파일

```
<navigation 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:id="@+id/nav_graph"
    app:startDestination="@id/oneFragment">


</navigation>
```

New Destination

•Create new destination 을 클릭하여 새로운 fragment를 작성
•존재하는 activity 혹은 fragment 리스트 중에 하나를 선택할 수도 있음

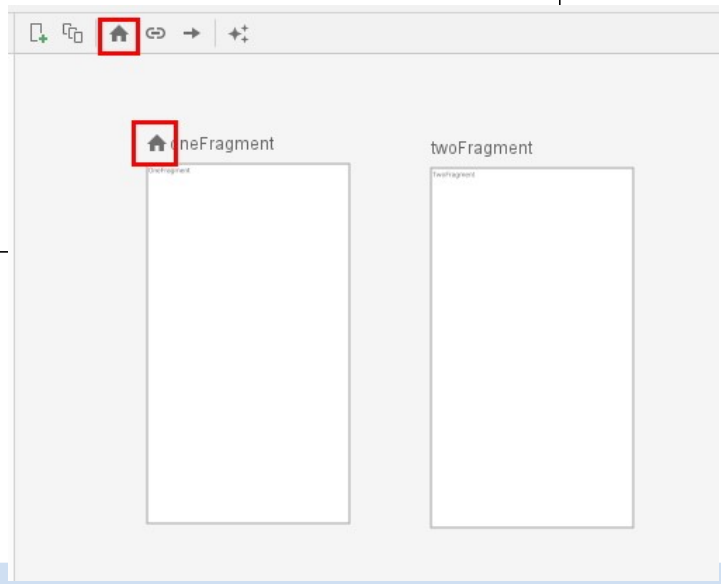




New Destination

•navigation graph

```
<navigation 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:id="@+id/nav_graph"
    app:startDestination="@id/oneFragment">

    <fragment
        android:id="@+id/oneFragment"
        android:name="com.example.test_navigation.OneFragment"
        android:label="fragment_one"
        tools:layout="@layout/fragment_one" />
</navigation>
```



•특정 Destination 을 start destination 으로 지정

NavHostFragment

- host 의 navigation 관리는 NavHost 인터페이스 구현체가 담당
- NavHost 인터페이스를 구현한 NavHostFragment를 Activity에서 이용하여 navigation 을 관리
- navGraph 속성으로 navigation graph 지정

```
<androidx.fragment.app.FragmentContainerView
    android:id="@+id/nav_host_fragment"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="match_parent"
    android:layout_height="0dp"
    android:layout_weight="1"
    app:defaultNavHost="true"
    app:navGraph="@navigation/nav_graph" />
```

<action>

- Navigation graph 에서 destination 간 연결 표현
- NavController 에 의한 destination 이동에 이용


```
<fragment>
    <action
        android:id="@+id/action_oneFragment_to_twoFragment"
        app:destination="@id/twoFragment" />
</fragment>
```

NavController

- Navigation 을 제어하기 위한 클래스로 navigate() 함수로 화면 이동
- action id 로 화면 이동과 destination id 로 화면 이동 방법 제공

```
val controller = Navigation.findNavController(it)
controller.navigate(R.id.action_oneFragment_to_twoFragment)
```

```
val controller = Navigation.findNavController(it)
controller.navigate(R.id.twoFragment)
```

NavController

•FragmentContainerView 의 app:defaultNavHost="true" 속성에 의해 Back button 이 클릭되면 이전 Destination 으로 자동 이동

```
<androidx.fragment.app.FragmentContainerView

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

•navigateUp() 혹은 popBackStack() 함수로 이전 Destination 으로 이동 가능

```
val controller = Navigation.findNavController(it)
controller.navigateUp()
```

Menu navigate

- menu id 와 destination id 를 동일하게 등록하여 화면 이동
- onNavDestinationSelected() 함수를 이용하여 MenuItem 과 NavController를 연결

```
<menu xmlns:android="http://schemas.android.com/apk/res/android">
    <item
        android:id="@+id/oneFragment"
        android:title="one"/>
    <item
        android:id="@+id/twoFragment"
        android:title="two"/>
</menu>
```

```
override fun onOptionsItemSelected(item: MenuItem): Boolean {
    NavigationUI.onNavDestinationSelected(item, Navigation.findNavController(this,
R.id.nav_host_fragment))
    return super.onOptionsItemSelected(item)
}
```

Bundle data

•화면전환시 Bundle 로 데이터 전송
•navigate() 함수의 두번째 매개변수로 Bundle 전달

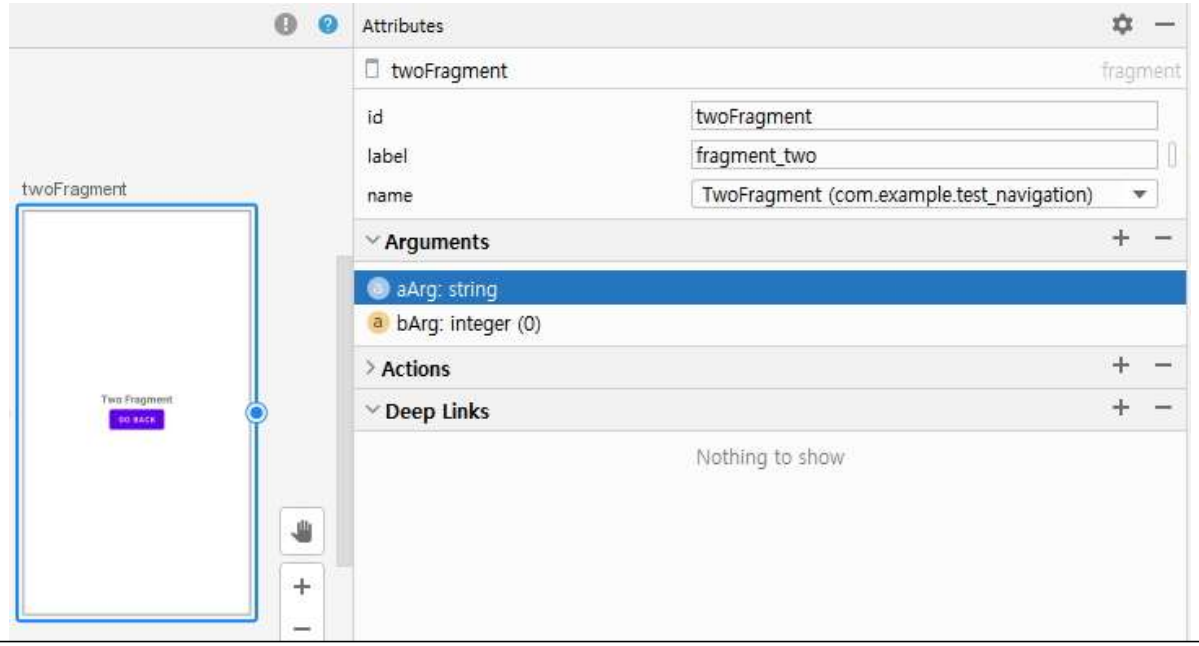
```
val bundle = Bundle()
bundle.putString("aArg", "kkang")
bundle.putInt("bArg", 20)

val controller = Navigation.findNavController(it)
controller.navigate(R.id.twoFragment, bundle)
```

•데이터 획득

```
val bundle = arguments

val aArg = bundle?.getString("aArg")
val bArg = bundle?.getInt("bArg")
```

<argument>

•Nav graph 에서 argument 를 등록




```
<fragment
   android:id="@+id/twoFragment">
   <argument       android:name="aArg"       app:argType="string" />
   <argument
      android:name="bArg"       app:argType="integer"       android:defaultValue="0" />
</fragment>
```

SafeArgs

•destination 에 argument 등록은 화면 이동시 전송되는 데이터의 구조를 파악하기 위한 것
•navigate 시에 데이터를 포함시키지 않았다면 선언된 default 값이 획득이 되고 default 값이 선언되지 않았다면 null
•SafeArgs 는 destination 에 선언된 대로 argument 가 맞게 전달 되는지에 대한 검증
    •argument 의 개수
    •argument 의 이름
    •Default 가 선언여부

•build.gradle (Project Level)

```
classpath"androidx.navigation:navigation-safe-args-gradle-plugin:$nav_version"
```

•build.gradle (Modulet Level)

```
plugins {
    id "androidx.navigation.safeargs.kotlin"
}
```

Directions

•safeargs를 위한 클래스가 자동으로 만들어 짐
•OneFragemtn -> TwoFragment 인 경우, OneFragment + Directions 클래스명으로 만들어짐
•Directions 클래스에 action id 값으로 데이터를 등록하기 위한 함수가 자동 generate
•예를 들어 action id 가 action_oneFragment_to_twoFragment 이면 함수명은 actionOneFragmentToTwoFragment()

```
val directions = OneFragmentDirections.actionOneFragmentToTwoFragment(aArg = "hello")

controller.navigate(directions)
```

Args

- 데이터를 받는 Fragment 를 위한 Args 클래스 자동 generate
- Fragment 클래스명 + Args

```
bundle?.let {
    TwoFragmentArgs.fromBundle(bundle).also {
        binding.twoTextView.text = "aArg : ${it.aArg}, bArg : ${it.bArg}"
    }
}
```

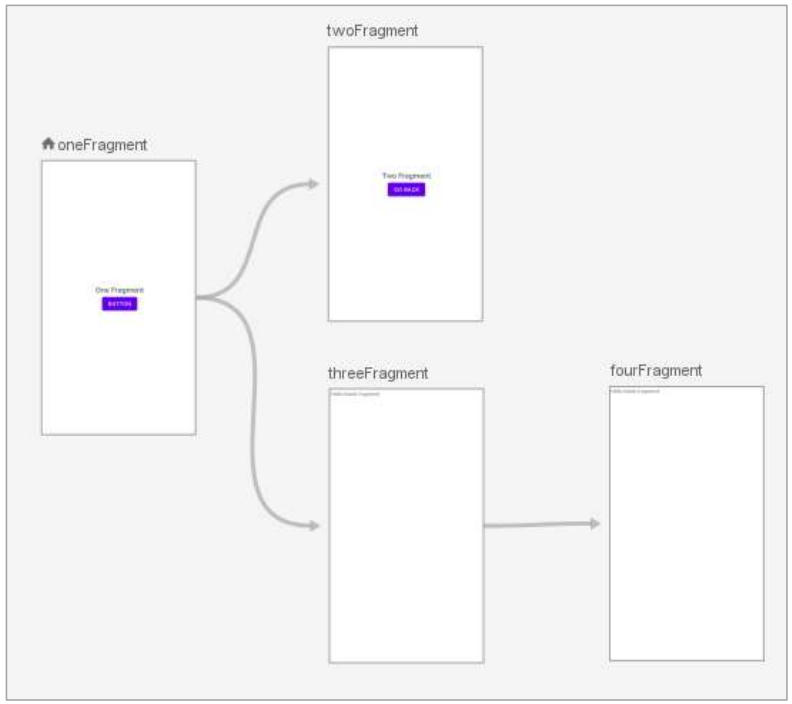
DestinationChangeListener

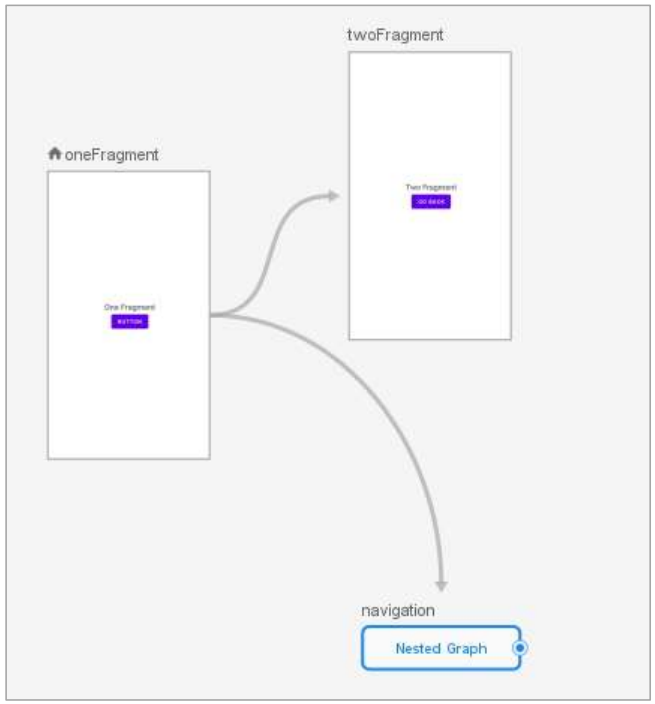
- NavController 에 의해 destination 으로 navigate 되는 순간을 이벤트로 감지가 가능

```
controller.addOnDestinationChangedListener { controller, destination, arguments ->
    Log.d("kkang", "navigation changed.. to : ${destination.id}")
}
```

Nested Graph

•navigation 이 복잡하게 나오는 경우 group 으로 묶어 조금 더 효과적으로 관리하는 기법





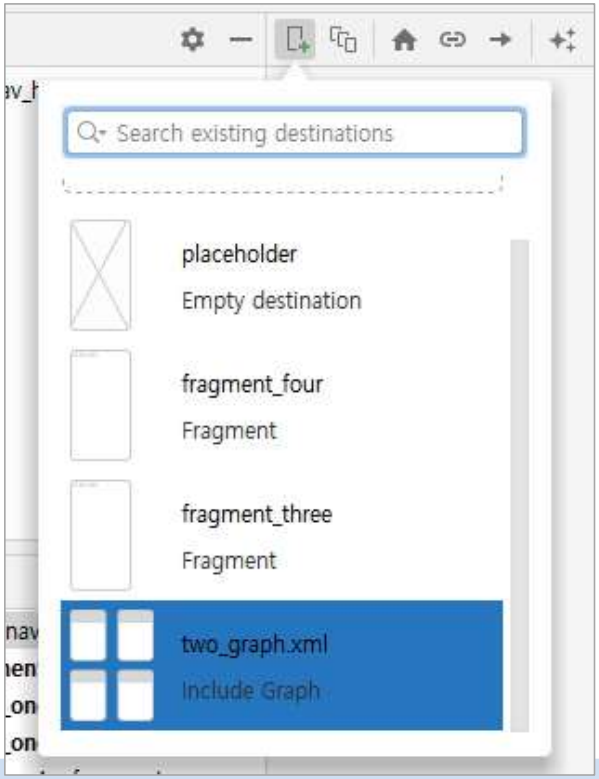
Nested Graph

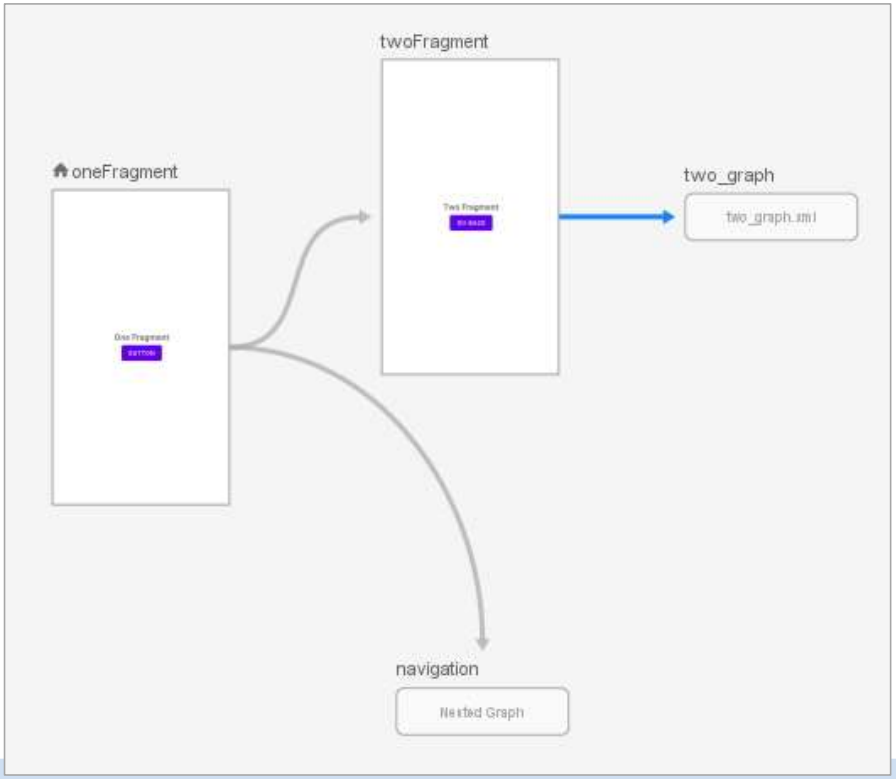
- Nav graph 에는 두개의 graph 처럼 나오지만 xml 은 하나
- <navigation> 이 중첩

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

    <fragment />
    <fragment />

    <navigation android:id="@+id/navigation"
        app:startDestination="@id/threeFragment">

        <fragment />
        <fragment  />

    </navigation>
</navigation>
```

<include>

- navigation 이 복잡한 경우 여러 xml로 작성
- XML 파일을 <include> 로 포함





감사합니다

단단히 마음먹고 떠난 사람은
산꼭대기에 도착할 수 있다.
산은 올라가는 사람에게만 정복된다.

윌리엄 셰익스피어
William Shakespeare