Andorid Jetpack架构之Navigation

官方介绍：

处理应用内导航所需的一切

<https://developer.android.google.cn/guide/navigation/>

Andorid Studio配置：

<https://developer.android.google.cn/guide/navigation/navigation-getting-started>

1. 添加gradle依赖：

AndroidX:

def nav\_version = "2.1.0"

implementation "androidx.navigation:navigation-fragment:$nav\_version"

implementation "androidx.navigation:navigation-ui:$nav\_version

JAVA:

def nav\_version = "1.0.0"

implementation "android.arch.navigation:navigation-fragment:$nav\_version" // For Kotlin use navigation-fragment-ktx

implementation "android.arch.navigation:navigation-ui:$nav\_version" // For Kotlin use navigation-ui-ktx

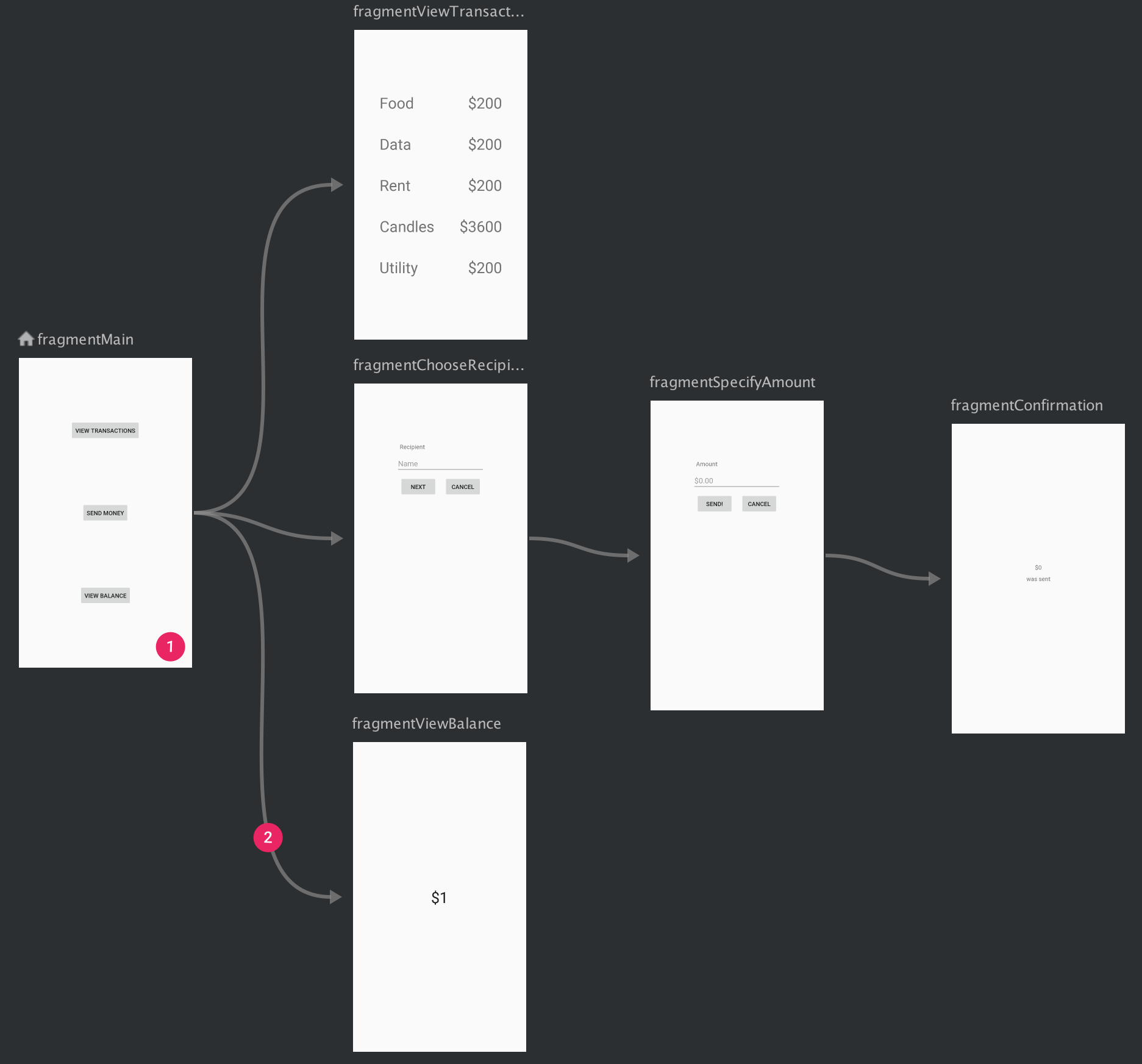
创建导航图：

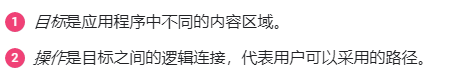
导航发生在应用程序的目标之间，即用户可以导航到应用程序中的任何位置。这些目的地是通过action连接的。

一个导航图是一个包含所有的目的地和行动的资源文件。该图代表您应用的所有导航路径。

图1显示了一个示例应用程序的导航图的直观表示，该应用程序包含通过五个动作连接的六个目的地。每个目标都由预览缩略图表示，而连接操作则由箭头表示，箭头表示用户如何从一个目标导航到另一个目标

例如：



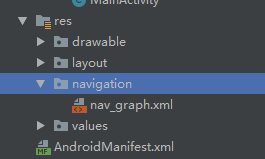


要将导航图添加到项目中，请执行以下操作

File-> New > Android Resource File -> New Resource File （比如文件名nav\_graph）

Resource type选择Navigation

创建完成会生成如下文件：



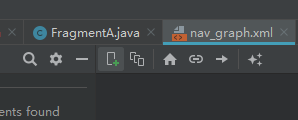
<navigation xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto"  
 xmlns:tools="http://schemas.android.com/tools"  
 android:id="@+id/nav\_graph"  
 app:startDestination="@id/fragmentA">

app:startDestination="@id/fragmentA"表示启动开始fragment

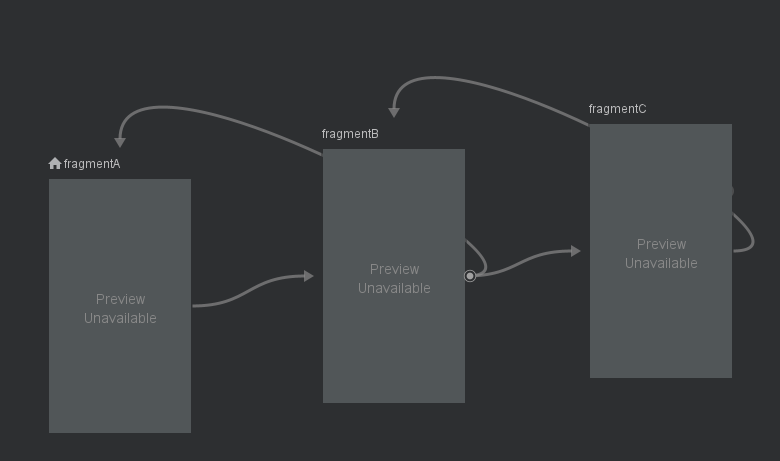
2. 创建不同页面Fragment：

正常创建FragmentA，FragmentB，FragmentC

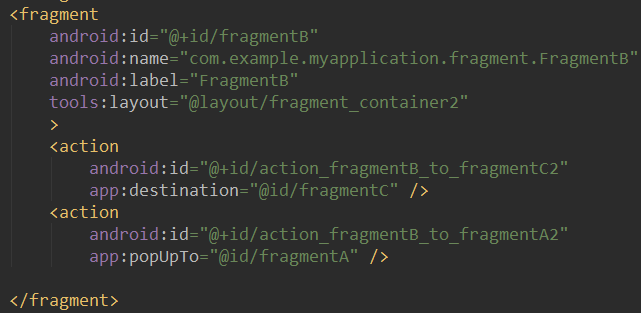
在nav\_graph视图中添加



加入3个Fragment，如下：



拖拽调转：



添加Action如下：

app:destination="@id/fragmentC"

表示跳转下一个Fragment，id为fragmentC

app:popUpTo="@id/fragmentA"

表示返回上一个Fragment，id为fragmentA

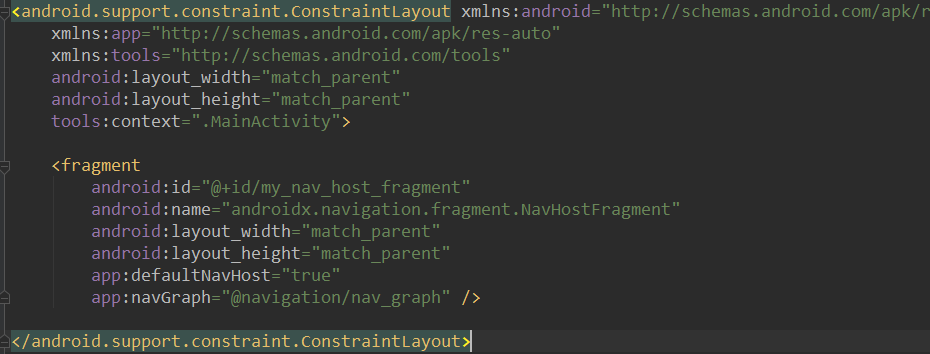
tools:layout="@layout/fragment\_container2"

用于显示视图，不加默认design不显示对应layout

3. 配置好Fragment跳转，我们需要一个Holder/Main将其串起来：

在Activity中通过 Navigation **展示一系列的Fragment**，但是我们需要告诉Navigation 和Activity，这一系列的 Fragment **展示在哪**——NavHostFragment应运而生，我把它的作用归纳为 **导航界面的容器**

在MainActivity中activity\_mian.xml添加HostFragment（androidx.navigation.fragment.NavHostFragment）

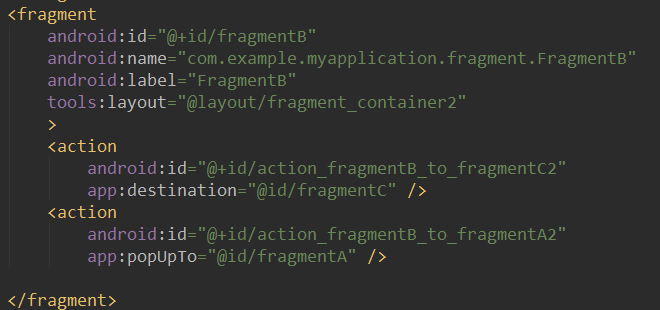


修改MainActivity的onSupportNavigateUp方法：

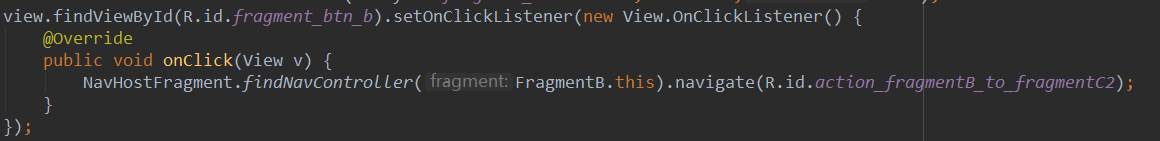
@Override  
public boolean onSupportNavigateUp() {  
 return Navigation.*findNavController*(this, R.id.*my\_nav\_host\_fragment*).navigateUp();  
}

获取NavHostFragment然后将Activity的返回事件交给NavHostFragment处理，NavHostFragment会把fragment进行后退，在页面之间进行返回

处理Fragment之间调转目标页面：



修改FragmentB：



跳转下一个页面，action指定action\_fragmentB\_to\_fragmentC2，而action\_fragmentB\_to\_fragmentC2配置的是FragmentC

Fragment之间传递数据：

Google不建议Fragment传递大量序列化的数据，使用Fragment之间传递数据可以参考使用ViewModel（因为是在activity里面传递数据，所以使用ViewModelStore可以获取数据，还能保证数据不会丢失，不需要序列化）

之间介绍过ViewModel，这里直接使用

1. 创建我们的ViewModel，并在Activity中实例化

MainViewModel viewModel = ViewModelProviders.*of*(this).get(MainViewModel.class);

使用该方法会绑定一个与activity关联的ViewModelStore（具体是一个Fragment，并且retainInstance）

此时在Activity里面的NavHostFragment启动开始的FragmentA跳转FragmentB，在Fragment的onAttach里面获取ViewModel里面的数据：

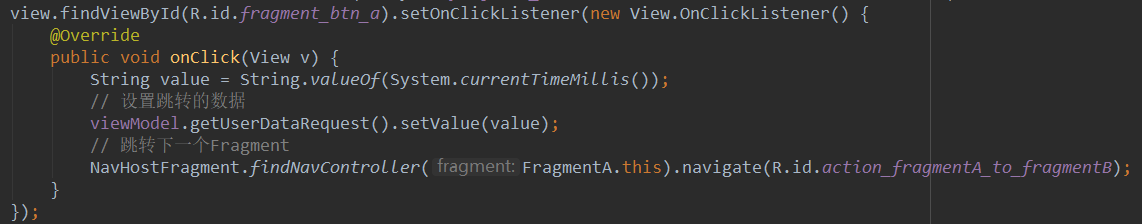
比如：

1.FragmentA有一个按钮，点击之后设置ViewModel的数据，然后跳转FragmentB

2.在FragmentB里面获取ViewModel的数据，并打印最新数据

1. ：

FragmentA



2. ：

