**Navigation and Fragment Activities**

**Aims:**

To introduce developing with Fragments and a Navigation Graph

**Objectives:**

* Using a navigation graph
* Managing fragment activities
* Sharing a view model between fragments

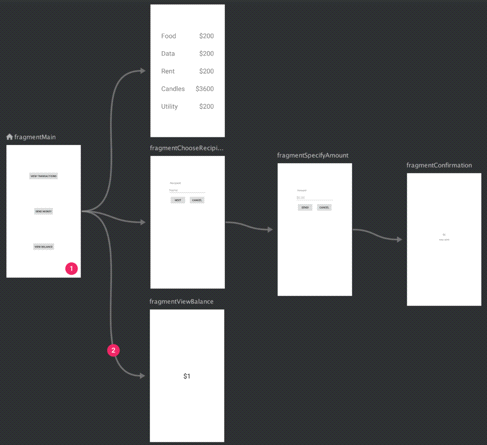
**1. Introduction**

Here we’ll use Android Studio to develop an Android app with a more sophisticated user interface that will manage the navigation between fragment activities.

Fragments are partial activities that are hosted by a main activity. The main activity can host data which can be shared between the fragments.

Firstly, we’ll set up navigation, using the following components…

* **Navigation Graph**: which is an XML resource with navigation **destinations**, and **actions** for navigating between them.



* **NavHostFragment**: which is a widget you add to your main activity layout for displaying the destinations for the navigation graph.
* **NavController**: which is an object that keeps track of the current position within the navigation graph and is used for swapping destinations in the NavHostFragment as you move through a navigation graph.

**2. Starting a new project**

Let’s get started…

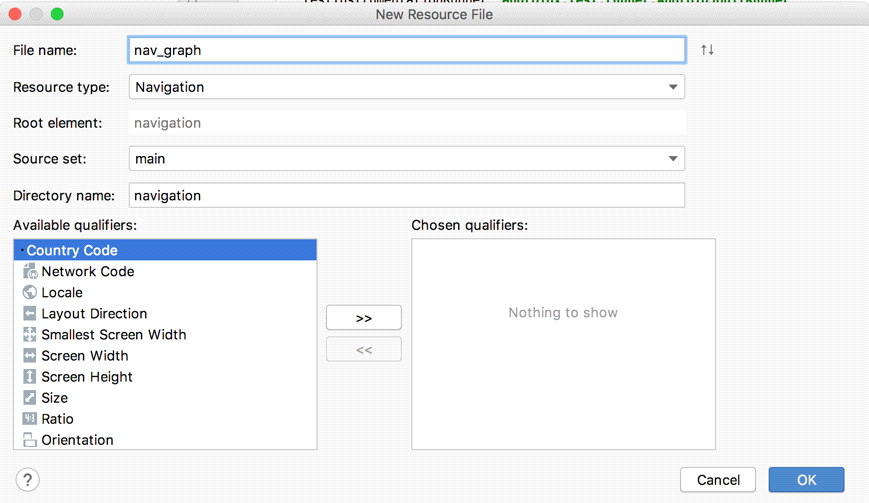
From windows start Android Studio and start a new Android Studio project.

Choose an **Empty Activity**, name the project ‘**MyFragmentsApp**’ and ensure the language is ‘**Java**’ before you click [Finish].

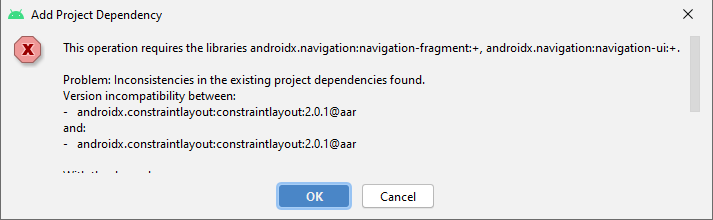
Wait for Gradle sync and indexing to finish (see bottom right of Android Studio).

**3. Creating a Navigation Graph**

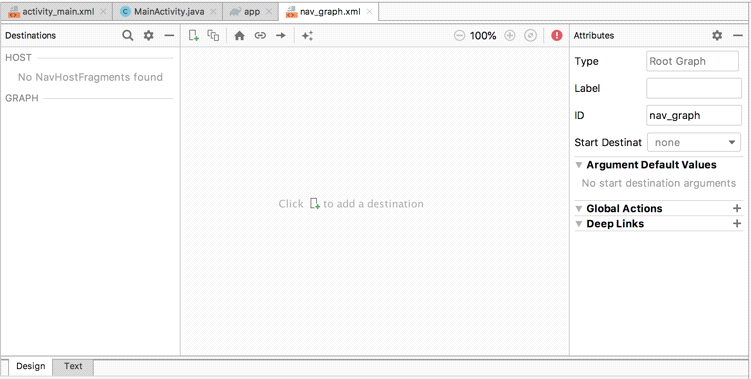
Right click on ‘**res**’ directory to create a new Android Resource File, from the resulting dialog set ‘File Name’: to be ‘**nav\_graph**’ and select ‘Resource type :’ to be ‘navigation’.



You maybe be prompted with a message to add additional libraries, click [OK].



In the Design View you will see an empty navigation graph.



which represents the XML file…

*<?***xml version="1.0" encoding="utf-8"***?>*<**navigation xmlns:android="http://schemas.android.com/apk/res/android"  
 xmlns:app="http://schemas.android.com/apk/res-auto" android:id="@+id/nav\_graph"**>  
  
</**navigation**>

**4. Create Some Fragments**

From the Project explorer right click on …

**app/java/com.*your-id*. myfragmentsapp**

to create a **New / Fragment / Fragment (Blank)**

In the resulting dialog select the ‘Fragment Name:’ to be ‘**FirstFragment**’

Similarly create a **New / Fragment / Fragment (Blank)**

In the resulting dialog select the ‘Fragment class name:’ to be ‘**SecondFragment**’

You will see new java files **FirstFragment** and **SecondFragment**, for example with the code…

**public class** FirstFragment **extends** Fragment {  
 *…. Some code…*

}

You will also see new XML files

res/layout/**fragment\_first.xml** which works with the first fragment

res/layout/ **fragment\_second.xml** which works with the second fragment

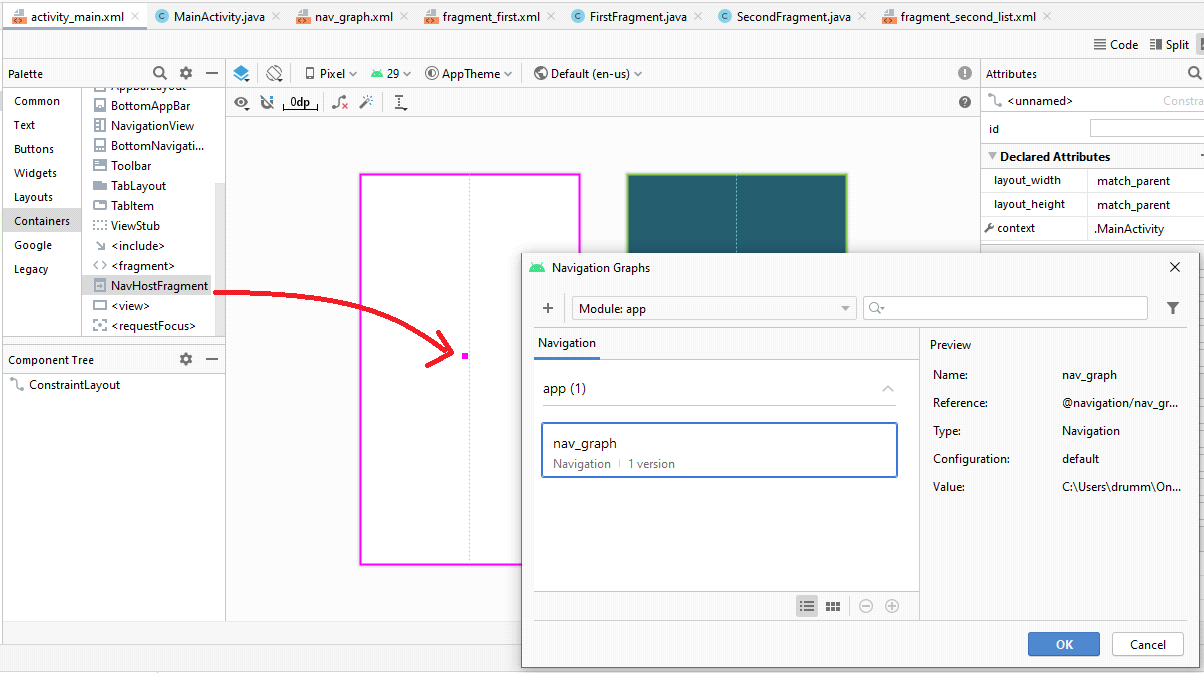
for example, fragment\_first’s XML is

*<?***xml version="1.0" encoding="utf-8"***?>*<**FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"  
 .. some xml …**  
 *<!--* ***TODO: Update blank fragment layout*** *-->* <**TextView  
 android:layout\_width="match\_parent"  
 android:layout\_height="match\_parent"  
 android:text="@string/hello\_blank\_fragment"** />  
  
</**FrameLayout**>

**5. Host your navigation graph**

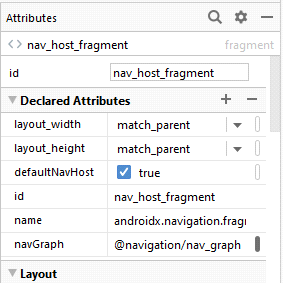
An activity needs to ‘host’ your navigation graph, for example we’ll choose the main activity to be the host.

Delete the TextView in **activity\_main.xml** file, hence drag and drop from Containers a **NavHostFragment** onto your design view, hence when prompted, choosing the **nav\_graph** you created earlier.



From the nav\_host\_fragment’s attribute inspector, change…

* the id to **nav\_host\_fragment**.
* **layout\_width** and **layout\_height** to **match\_parent**.



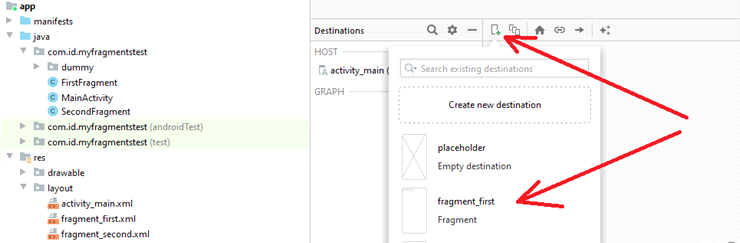
**6. Bring Fragments into Navigation Graph using Designer**

Now from the Project explorer double click on **res/navigation/nav\_graph.xml**

Click on the

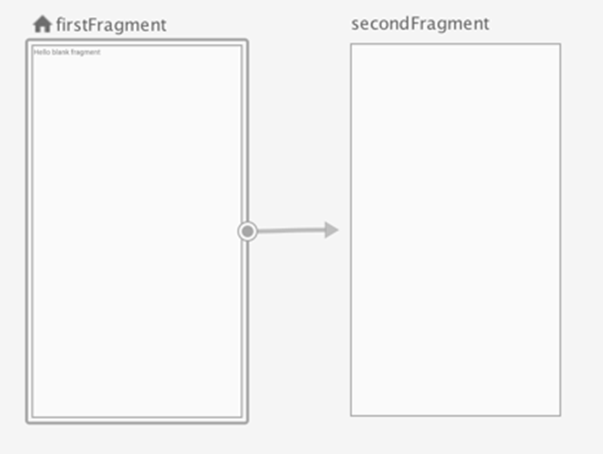


Hence, choose to add your **fragment\_first** to the graph



Similar add **fragment\_second\_list** to the graph**.**

You can click on the first fragment and click on the blue circle to the right of it to drag an arrow to your second fragment.



Note how this adds XML to nav\_graph.xml to define an **action**…

<**action  
 android:id="@+id/action\_firstFragment\_to\_secondFragment"  
 app:destination="@id/secondFragment"** />

**7. Nav Controller to use actions and move between fragments**

Here we’ll add a button to move from the first fragment activity to the second fragment activity.

Go to **fragment\_first.xml**

Delete the default TextView (*tip: it’s often easier to do this from the Component Tree*).

Add a layout, e.g. a **Linear Layout (vertical)**

Add a **Button** and give it the id ‘**next\_button**’ and the text ‘Next’.

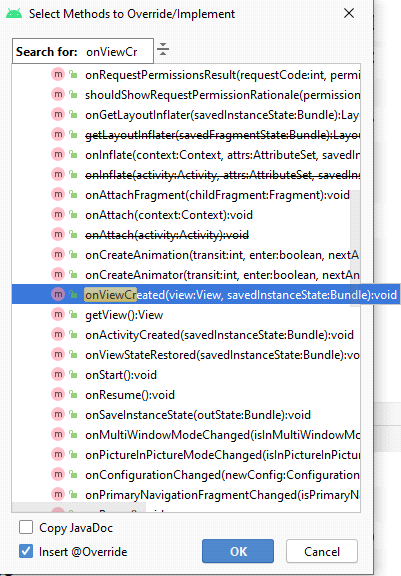
*Ctrl-S to save all.*

Go to **FirstFragment.java**, add the field variables…

**private** NavController **navController**;  
**private** Button **nextButton**;

From the Android Studio’s main menu and choose **Code->Override Methods**…

Hence type *onViewCreated*, select the method and click [OK] to override this method…



Hence add to the resulting **onViewCreated()**,some code to bind field references to objects…

**navController** = Navigation.*findNavController*(view);  
**nextButton** = view.findViewById(R.id.***next\_button***);

and add an OnClickListener for the button…

**nextButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
   
 }  
});

Hence in the implemented onClick() method, add code that uses the **action** to move from the first fragment to the second fragment.

**navController**.navigate(R.id.***action\_firstFragment\_to\_secondFragment***);

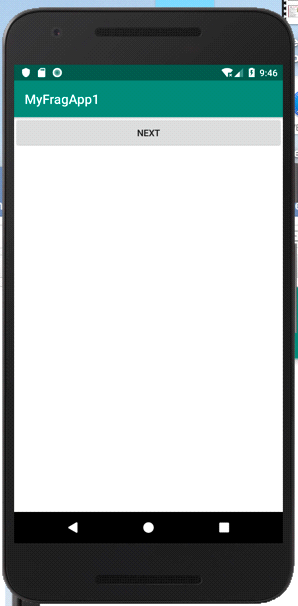
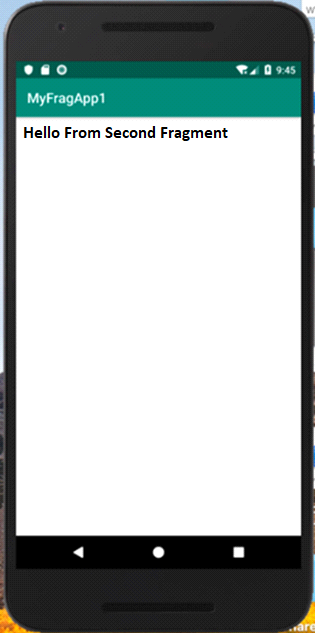
So, your code for the onViewCreated() method should be…

@Override  
**public void** onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {  
 **super**.onViewCreated(view, savedInstanceState);  
 **navController** = Navigation.*findNavController*(view);  
 **nextButton** = view.findViewById(R.id.***next\_button***);  
 **nextButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **navController**.navigate(R.id.***action\_firstFragment\_to\_secondFragment***);  
 }  
 });  
}

Go to **fragment\_second.xml**

Change the default **TextView**’s **id** to ‘textViewFrag2’ and set its **text** to “hello from second fragment”.

Run your application…

Notice how you can use your button to move to the next fragment and use the back arrow to get back to the start.

**8. Sharing a model between fragments**

Typically when we create an app we want to share data between fragments, for example in a game the first fragment might ask for the player’s name, the next fragment might be a game where the player gains a score, and the next fragment presents the name and score with a congratulations or commiserations message..

The modern way to share data is via a shared model. A model is simply and object to hold some data, it could just be as simple as a string or an instance of a complex class which you define.

From the Project explorer right click on …

**app/java/com.*your-id*. myfragmentsapp**

to create a **New / Java Class** and give it the name MyModel

Add to the resulting **MyModel** class private fields for **name** and **age**, and from Android Studio's main menu use **Code->Generate** to generate thier getters and setters to give....

**public class** MyModel {  
 **private** String name;  
 **private int** age;  
  
 **public** String getName() {  
 **return** name;  
 }  
  
 **public void** setName(String name) {  
 **this**.name = name;  
 }  
  
 **public int** getAge() {  
 **return** age;  
 }  
  
 **public void** setAge(**int** age) {  
 **this**.age = age;  
 }  
}

Though note later you could add in here whatever variables, objects, lists, etc you want for your application.

To make this a shared model we can’t just instantiate it in an activity, instead Android’s ‘jetpack’ libraries provides a…

* **ViewModel** class which we can subclass to manage the interactions between our fragments and our model.
* **MutableLiveData** class to make our model observable, so when one fragment changes the data in the model, other fragments can be notified and observe the change.

From the Project explorer right click on …

**app/java/com.*your-id*. myfragmentsapp**

to create a **New / Java Class** and give it the name **MyViewModel** and specify its **subclass** to be **ViewModel**.

Add to the resulting **MyViewModel** class code to give…

**public class** MyViewModel **extends** ViewModel {

**private** MutableLiveData<MyModel> **myLiveModel**;  
  
 **public** MyViewModel()  
 {  
 **myLiveModel** = **new** MutableLiveData<MyModel>();  
 **myLiveModel**.setValue(**new** MyModel());  
 }

LiveData<MyModel> getMyModel() {  
 **return myLiveModel**;  
 }

}

The above code

* declares a field variable myLiveModel
* binds it to an instance MutableLiveData<MyModel>
* hence associates this with an instance of MyModel
* and provides a useful getter method to myLiveModel

Next, we want to access the **MyViewModel** in a fragment.

For example, in **FirstFragment.java** add to the **FirstFragment**

class the field variables…

MyModel **myModel**;  
MyViewModel **myViewModel**;

From the Android Studio’s main menu and choose **Code->Override Methods**…

Hence type *onActivityCreated*and select this method and click [OK] to override this method…

Add to the **onActivityCreated()** method the following…

**myViewModel** = **new** ViewModelProvider(requireActivity()).get(MyViewModel.**class**);  
**myModel**=**myViewModel**.getMyModel().getValue();  
**myModel**.setName(**"Fred Bloggs"**);  
**myModel**.setAge(25);

The code uses **ViewModelProvider** to bind to the view model, hence enables to model to be accessed and changed.

Hence, we also want to access the view model fromanother fragment.

For example, in **SecondFragment.java** add to the **SecondFragment**

class the field variables…

MyViewModel **myViewModel**;  
MyModel **myModel**;  
TextView **tv**;

Override the **SecondFragment** class’s **onViewCreated()** and add …

**tv** = view.findViewById(R.id.***textViewFrag2***);  
  
**myViewModel** = **new** ViewModelProvider(requireActivity()).get(MyViewModel.**class**);  
**myViewModel**.getMyModel().observe(getViewLifecycleOwner(), **new** Observer<MyModel>() {  
 @Override  
 **public void** onChanged(MyModel myModel) {  
 **tv**.setText(**"Name is "** + myModel.getName()+**" of age "** + myModel.getAge());  
 }  
});

You can see the **myViewModel**.getMyModel().observe() method lets you specify an Observer object to handle changes to the model, in this case setting the second fragment’s text view to show values set in the first fragment.

Run the app.

**EXERCISE**

Create a new app with a storyboard including three colourful fragments which present to the user…

* a splash screen maybe with an image and colourful text,
* a form where a user can enter (for example) their name, age, height, weight, weekly units of alcohol and smoking/non-smoking
* and a screen to show recommendations for a healthier lifestyle based on the data

Think about what data your model should hold and how you might set it from form and access it and use to performs appropriate calculations. Google around for some suitable formula and information.

**COMPLETE CODE EXAMPLE**

**MyModel.java**

**package** com.example.navapp;  
  
**public class** MyModel {  
 **protected** String **name**;  
 **protected int age**;  
}

**MyViewModel.java**

**package** com.example.navapp;  
**import** androidx.lifecycle.LiveData;  
**import** androidx.lifecycle.MutableLiveData;  
**import** androidx.lifecycle.ViewModel;  
  
**public class** MyViewModel **extends** ViewModel {  
 **private** MutableLiveData<MyModel> **myLiveModel**;  
  
 **public** MyViewModel()  
 {  
 **myLiveModel** = **new** MutableLiveData<MyModel>();  
 **myLiveModel**.setValue(**new** MyModel());  
 }  
 LiveData<MyModel> getMyModel() {  
 **return myLiveModel**;  
 }  
}

**FirstFragment.java**

**public class** FirstFragment **extends** Fragment {  
  
 // after other code

Button **myButton**;  
 NavController **navController**;  
 MyModel **myModel**;  
 MyViewModel **myViewModel**;  
  
 @Override  
 **public void** onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {  
 **super**.onViewCreated(view, savedInstanceState);  
  
  
 **navController** = Navigation.*findNavController*(view);  
 **myButton** = view.findViewById(R.id.***button***);  
 **myButton**.setOnClickListener(**new** View.OnClickListener() {  
 @Override  
 **public void** onClick(View v) {  
 **navController**.navigate(R.id.***action\_firstFragment\_to\_secondFragment***);  
 }  
 });  
  
 **myViewModel** = **new** ViewModelProvider(requireActivity()).get(MyViewModel.**class**);  
  
 **myModel**=**myViewModel**.getMyModel().getValue();  
 **myModel**.**age** = 25;  
 **myModel**.**name** = **"Fred Bloggs"**;  
 }  
}

**SecondFragment.java**

**public class** SecondFragment **extends** Fragment {  
  
 // after other code

MyModel **myModel**;  
 MyViewModel **myViewModel**;  
 @Override  
 **public void** onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {  
 **super**.onViewCreated(view, savedInstanceState);  
 TextView tv = view.findViewById(R.id.***textViewFrag2***);  
  
 **myViewModel** = **new** ViewModelProvider(requireActivity()).get(MyViewModel.**class**);  
 **myViewModel**.getMyModel().observe(getViewLifecycleOwner(), **new** Observer<MyModel>() {  
 @Override  
 **public void** onChanged(MyModel myModel) {  
 tv.setText(myModel.**name** + **" student "** + myModel.**age**);  
 }  
 });  
 }  
}