

PROJECT: [AW] NAGOYA – HMI

LEADER : ANH TRAN

MENTOR: TAM NGUYEN

ANDROID RESERCHING REPORT

VERSION 0.1

**HISTORICAL**

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Date | Author | Notes |
| 0.1 | 19/11/2020 | Quan Truong | Create document |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**CONTENT**

[**2.** **Using the Fragment methods in the host Activity** 6](#_Toc56724145)

**I. UI CONTROLS**

1. **EditText, TextView, Button, RadioButton, CheckBox, Spinner**

-Pass.

1. **Date Picker**

* Have 2 mode:  
  android:datePickerMode="calendar"

android:datePickerMode="calendar"

* Get day, month, year from Calendar:  
   Calendar calendar = Calendar.getInstance();  
   calendar.setTimeInMillis(System.currentTimeMillis());  
   int year = calendar.get(Calendar.YEAR);  
   int month = calendar.get(Calendar.MONTH);  
   int day = calendar.get(Calendar.DAY\_OF\_MONTH);

1. **Time Picker**

* Have 2 mode:  
  android:timePickerMode="clock"

android:timePickerMode="spinner"

* + - Get time from TimePiker:

timePicker.setOnTimeChangedListener

**4. RecycleView**

\*Folow the folowing step:

1. Add the support library

2. Add the RecyclerView in the layout XML file

3. Create a custom row Layout

4. Create the ﻿ RecyclerView.Adapter ﻿to populate data into the RecyclerView

5. Create the ﻿ViewHolder to ﻿provide a reference to the views for each data item

6. Bind the Adapter to the RecyclerView in the Activity

**5. Toolbar Search**

\*Folow the folowing step:

1. Create toolbar layout

2. Include toolbar layout to layout XML file

3. Create ArrayList temp = ArrayList<model> currently

3. Clear ArrayList<model> currently

4 . Use TextWatcher to compare between keyword (EditText) and element in ArrayList temp: *toLowerCase*().*trim*().*contains(…)*

5. Set Adapter (recycleview) by ArrayList temp

**II. ANDROID CORE BUILDING: FRAGMENT**

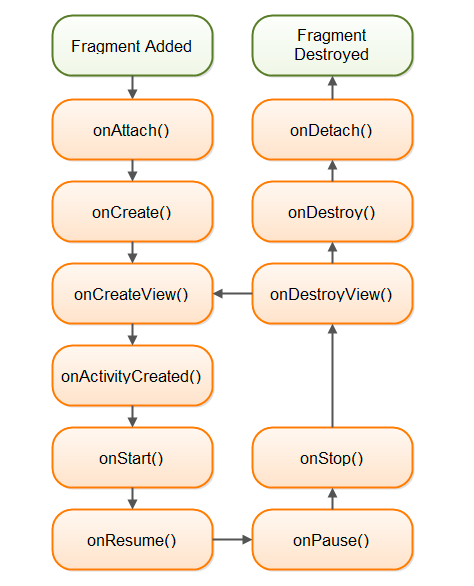
**1. Fragment LifeCycle**

\*Fragment lifecycle states:

The Fragment is added by an Activity (which acts as the host of the Fragment). Once added, the Fragment goes through three states, as shown in the figure below:

* Active (or resumed)
* Paused
* Stopped

Because a Fragment is always hosted by an Activity, the Fragment lifecycle is directly affected by the host Activity lifecycle. For example, when the Activity is paused, so are all Fragments in it, and when the Activity is destroyed, so are all Fragments.



* [onAttach()](https://developer.android.com/reference/android/app/Fragment.html#onAttach(android.content.Context)): Called when a Fragment is first attached to a host Activity. Use this method to check if the Activity has implemented the required listener callback for the Fragment (if a listener interface was defined in the Fragment). After this method, [onCreate()](https://developer.android.com/reference/android/app/Fragment.html#onCreate(android.os.Bundle)) is called.
* [onCreate()](https://developer.android.com/reference/android/app/Fragment.html#onCreate(android.os.Bundle)): Initialize essential components and variables of the Fragment in this callback. The system calls this method when the Fragment is created. Anything initialized in onCreate() is preserved if the Fragment is paused and resumed.
* [onCreateView()](https://developer.android.com/reference/android/app/Fragment.html#onCreateView(android.view.LayoutInflater,%20android.view.ViewGroup,%20android.os.Bundle)): Inflate the XML layout for the Fragment in this callback. The system calls this method to draw the Fragment UI for the first time. As a result, the Fragment is visible in the Activity. To draw a UI for your Fragment, you must return the root [View](https://developer.android.com/reference/android/view/View.html) of your Fragment layout. Return null if the Fragment does not have a UI.
* [onActivityCreated()](https://developer.android.com/reference/android/app/Fragment.html#onActivityCreated(android.os.Bundle)): Called when the Activity onCreate() method has returned. Use it to do final initialization, such as retrieving views or restoring state. It is also useful for a Fragment that uses [setRetainInstance()](https://developer.android.com/reference/android/app/Fragment.html#setRetainInstance(boolean)) to retain its instance, as this callback tells the Fragment when it is fully associated with the new Activity instance. The onActivityCreated() method is called after [onCreateView()](https://developer.android.com/reference/android/app/Fragment.html#onCreateView(android.view.LayoutInflater,%20android.view.ViewGroup,%20android.os.Bundle)) and before [onViewStateRestored()](https://developer.android.com/reference/android/app/Fragment.html#onViewStateRestored(android.os.Bundle)).
* [onStart()](https://developer.android.com/reference/android/app/Fragment#onStart()) makes the fragment visible to the user (based on its containing activity being started).
* [onResume()](https://developer.android.com/reference/android/app/Fragment.html#onResume()): Called by the Activity to resume a Fragment that is visible to the user and actively running.
* [onPause()](https://developer.android.com/reference/android/app/Fragment.html#onPause()): Save any data and states that need to survive beyond the destruction of the Fragment. The system calls this method if any of the following occurs:
  1. The user navigates backward.
  2. The Fragment is replaced or removed, or another operation is modifying the Fragment.
  3. The host Activity is paused.
* onStop() — This method called when the fragment is no longer visible; the fragment will get change with other fragment or it gets removed from activity or fragment’s activity called stop.
* [onDestroyView()](https://developer.android.com/reference/android/app/Fragment.html#onDestroyView()): Called when the View previously created by onCreateView() has been detached from the Fragment. This call can occur if the host Activity has stopped, or the Activity has removed the Fragment. Use it to perform some action, such as logging a message, when the Fragment is no longer visible. The next time the Fragment needs to be displayed, a new View is created. The onDestroyView() method is called after [onStop()](https://developer.android.com/reference/android/app/Fragment.html#onStop()) and before [onDestroy()](https://developer.android.com/reference/android/app/Fragment.html#onDestroy()).
* onDestroy() :  This method called when the fragment does its final clean up.
* onDetach() :  This method called when the fragment is detached from its host activity.

1. **Using the Fragment methods in the host Activity**

Likewise, your Activity can call methods in the Fragment by acquiring a reference to the Fragment from FragmentManager, using [findFragmentById()](https://developer.android.com/reference/android/app/FragmentManager.html#findFragmentById(int)). For example, to call the getSomeData() method in the Fragment, acquire a reference first:

ExampleFragment fragment = (ExampleFragment)

getFragmentManager().findFragmentById(R.id.example\_fragment);

// ...

mData = fragment.getSomeData();

### Adding the Fragment to the back stack

A significant difference between an Activity and a Fragment is how activities and fragments use their respective back stacks, so that the user can navigate back with the **Back** button (as discussed in [Tasks and Back Stack](https://developer.android.com/guide/components/tasks-and-back-stack.html)).

* For an Activity, the system automatically maintains a back stack of activities.
* For a Fragment, the hosting Activity maintains a back stack, and you have to explicitly add a Fragment to that back stack by calling [addToBackStack()](https://developer.android.com/reference/android/app/FragmentTransaction.html#addToBackStack(java.lang.String)) during any transaction that adds the Fragment.

Keep in mind that when your app replaces or removes a Fragment, it's often appropriate to allow the user to navigate backward and "undo" the change. To allow the user to navigate backward through Fragment transactions, call [addToBackStack()](https://developer.android.com/reference/android/support/v4/app/FragmentTransaction.html#addToBackStack(java.lang.String)) before you commit the [FragmentTransaction](https://developer.android.com/reference/android/support/v4/app/FragmentTransaction.html):

fragmentTransaction.add(R.id.fragment\_container, fragment);

fragmentTransaction.addToBackStack(null);

fragmentTransaction.commit();

**Tip**: The addToBackStack() method takes an optional string parameter that specifies a unique name for the transaction. Specify null because the name isn't needed unless you plan to perform advanced Fragment operations using the [FragmentManager.BackStackEntry](https://developer.android.com/reference/android/support/v4/app/FragmentManager.BackStackEntry.html) interface.

When removing a Fragment, remember that the hosting Activity maintains a back stack for the Fragment (if you add the transaction to it, as described above). However, the Fragment is not destroyed. If the user navigates back to restore the Fragment, it restarts.

### Sending data to a Fragment from an Activity

To send data to a Fragment from an Activity, set a [Bundle](https://developer.android.com/reference/android/os/Bundle.html) and use the Fragment method [setArguments(Bundle)](https://developer.android.com/reference/android/app/Fragment.html#setArguments(android.os.Bundle)) to supply the construction arguments for the Fragment.

For example, if the user previously made a choice in the Fragment, and you want to send that choice back to the Fragment when starting it from the Activity, you would create the arguments Bundle and insert the String value of the choice into the Bundle mapping for the key ("choice").

The best practice for initializing the data in a Fragment is to perform this initialization in the Fragment in a factory method. As you learned in the lesson on fragments, you can create the Fragment instance with a newinstance() method in the Fragment itself:

public static SimpleFragment newInstance() {

return new SimpleFragment();

}

Then instantiate the Fragment in an Activity by calling the newInstance() method in the Fragment:

SimpleFragment fragment = SimpleFragment.newInstance();

1. **Sending data from a Fragment to its host Activity**

To have a Fragment communicate to its host Activity, follow these steps in the Fragment:

* 1. *Define a listener interface, with one or more callback methods to communicate with the Activity.*
  2. *Override the*[*onAttach()*](https://developer.android.com/reference/android/app/Fragment.html#onAttach(android.app.Activity))*lifecycle method to make sure the host Activity implements the interface.*
  3. *Call the interface callback method to pass data as a parameter.*

In the host Activity, follow these steps:

1. *Implement the interface defined in the Fragment. (All the Activity classes that use the Fragment have to implement the interface.)*
2. *Implement the Fragment callback method(s) to retrieve the data.*

The following is an example of defining the OnFragmentInteractionListener interface in the Fragment, including the onRadioButtonChoice() callback to communicate to the host Activity. The Activity must implement this interface. The onAttach() method gets a reference to the listener if the Activity implemented this interface; if not, this method throws an exception:

// Interface definition and onFeedbackChoice() callback.

interface OnFragmentInteractionListener {

void onRadioButtonChoice(int choice);

}

@Override

public void onAttach(Context context) {

super.onAttach(context);

if (context instanceof OnFragmentInteractionListener) {

mListener = (OnFragmentInteractionListener) context;

} else {

throw new ClassCastException(context.toString()

+ " must implement OnFragmentInteractionListener");

}

}

The following shows how the Fragment uses the onCheckedChanged() listener for checked radio buttons in the Fragment, and uses the onRadioButtonChoice() callback to provide data to the host Activity.

public void onCheckedChanged(RadioGroup group, int checkedId) {

View radioButton = radioGroup.findViewById(checkedId);

int index = radioGroup.indexOfChild(radioButton);

switch (index) {

case YES: // User chose "Yes."

textView.setText(R.string.yes\_message);

mRadioButtonChoice = YES;

mListener.onRadioButtonChoice(YES);

break;

case NO: // User chose "No."

textView.setText(R.string.no\_message);

mRadioButtonChoice = NO;

mListener.onRadioButtonChoice(NO);

break;

default: // No choice made.

mRadioButtonChoice = NONE;

mListener.onRadioButtonChoice(NONE);

break;

}

}

To use the Fragment callback method(s) to retrieve data, the Activity must implement the interface defined in the Fragment class:

public class MainActivity extends AppCompatActivity

implements SimpleFragment.OnFragmentInteractionListener {

//...

}

The Activity can then use the onRadioButtonChoice() callback to get the data.

@Override

public void onRadioButtonChoice(int choice) {

mRadioButtonChoice = choice;

Toast.makeText(this, "Choice is " + Integer.toString(choice),

LENGTH\_SHORT).show();

}

Some data in a Fragment is may be relevant to the Activity that hosts it. Your Activity code can use a callback to retrieve relevant data from the Fragment. The Activity can then send that data to the Fragment when recreating the Fragment.