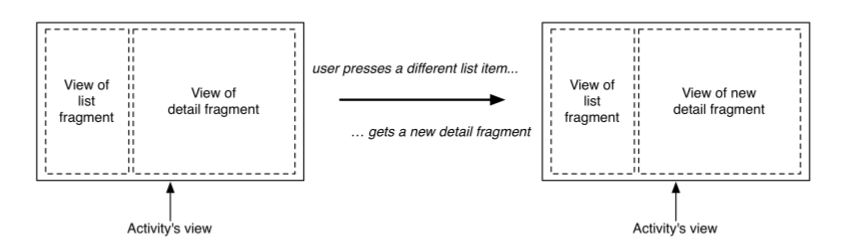
Android – Fragments

A fragment is a controller object that an activity can deputize to perform tasks. Most commonly, the task is managing a user interface. The user interface can be an entire screen or just one part of the screen.

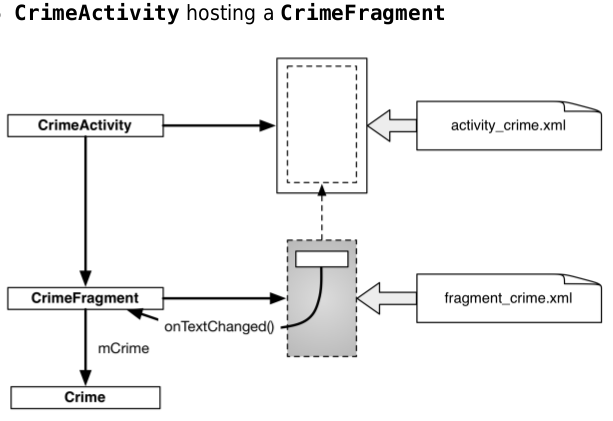
A fragment managing a user interface is known as a UI fragment. A UI fragment has a view of its own that is inflated from a layout file. The fragment’s view contains the interesting UI elements that the user wants to see and interact with.

The activity’s view contains a spot where the fragment’s view will be inserted. Or it might have several spots for the views of several fragments.



Using UI fragments separates the UI of your app into building blocks, which is useful for more thanjust list-detail applications. Working with individual blocks, it is easy to build tab interfaces, tack on animated sidebars, and more.

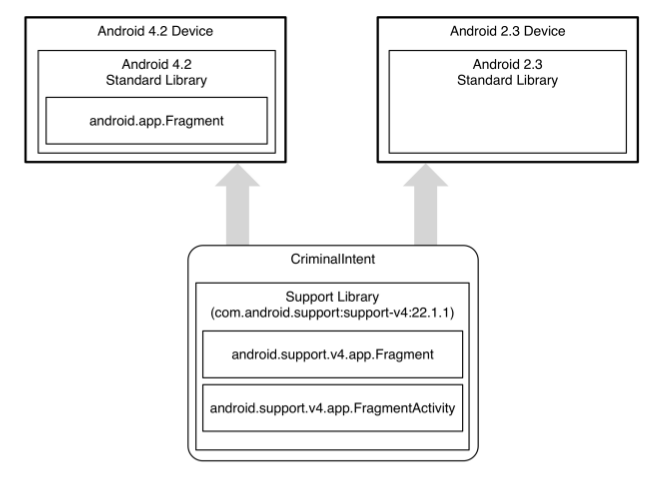
Here, the screen is managed by a UI fragment named CrimeFragment . An instance of CrimeFragment is hosted by an activity named CrimeActivity .



Fragments were introduced in API level 11. The support library includes a complete implementation of fragments that work all the way back to API level 4.

Note that when you use a support library class, it is not just used on older versions where no native class is available, it is also used on newer versions instead of the native class.

There are two key classes that we will use from the support library: the Fragment class (android.support.v4.app.Fragment) and the FragmentActivity class (android.support.v4.app.FragmentActivity). Using fragments requires activities that know how to manage fragments. The FragmentActivity class knows how to manage the support version of fragments.

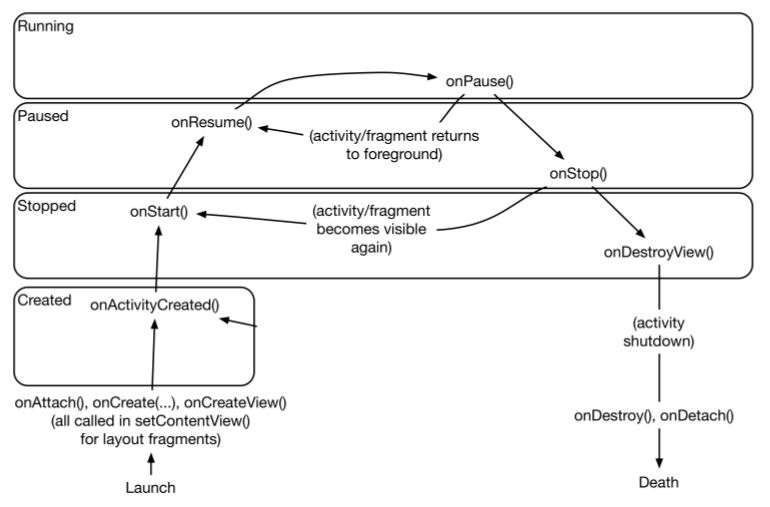


**To host a UI fragment, an activity must:**

• define a spot in its layout for the fragment’s view

• manage the lifecycle of the fragment instance

Fragment lifecycle :



**You have two options when it comes to hosting a UI fragment in an activity**:

• add the fragment to the activity’s layout

• add the fragment in the activity’s code

**The first approach** is known as using a layout fragment. It is **simple but inflexible**. If you **add the fragment to the activity’s layout**, you hardwire the fragment and its view to the activity’s view and cannot swap out that fragment during the activity’s lifetime.

**The second approach**, **adding the fragment to the activity’s code**, is **more complex**. But it is t**he only way to have control at runtime over your fragments**. You determine when the fragment is added to the activity and what happens to it after that. You can remove the fragment, replace it with another, and then add the first fragment back again.

**Here I will use the second approach.**

# FragmentActivity

1 – In app/java/<package\_name>/MainActivity.java, replace extends AppCompatActivity by FragmentActivity

# Defining a container view

Notice that the container view is completely generic, it does not name the CrimeFragment class. You can and will use this same layout to host other fragments.

2 - Locate MainActivity ’s layout at res/layout/activity\_main.xml . Open this file and replace the default layout with this FrameLayout :

<FrameLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

android:id="@+id/fragment\_container"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

Note that while activity\_main.xml consists solely of a container view for a single fragment, an activity’s layout can be more complex and define multiple container views as well as widgets of its own.

# Creating a UI Fragment

To create a layout file, right-click the res/layout folder in the project tool window and select New → Layout resource file. Name the file fragment\_crime.xml

<LinearLayout xmlns:android="<http://schemas.android.com/apk/res/android>"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<EditText android:id="@+id/crime\_title"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:hint="@string/crime\_title\_hint"/>

</LinearLayout>

# Creating the CrimeFragment class

public class CrimeFragment extends Fragment {

private Crime mCrime;

private EditText mTitleField;

...

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {

View v = inflater.inflate(R.layout.fragment\_crime, container, false);

mTitleField = (EditText)v.findViewById(R.id.crime\_title);

mTitleField.addTextChangedListener(new TextWatcher() {

@Override

public void beforeTextChanged(

CharSequence s, int start, int count, int after) {

// This space intentionally left blank

}

@Override

public void onTextChanged(

CharSequence s, int start, int before, int count) {

mCrime.setTitle(s.toString());

}

});

}

}

@Override

public void afterTextChanged(Editable s) {

// This one too

}

return v;

}

}

