By creating an Activity with a Fragment, 3 XML files are created. There is one file for the Activity and two for the Fragment. Specifically, activity\_main.xml is the layout file for the Activity and fragment\_main.xml and content\_main.xml are used for the Fragment.

Two Java source files are created. One is MainActivity.java which is the code for the Activity. The other is MainActivityFragment.java which is the code for the Fragment.

Taking the high-level view, this is how the code and layout files work:

* Activity\_main.xml is the layout for the activity and a full screen “shell” for displaying the fragment. It includes the XML layout file content\_main.xml

Content\_main.xml is the layout for the Fragment

Fragment\_main.xml is the layout for what should be displayed in the Fragment and all real user interface widgets would go there

MainActivity.java includes an onCreate() method that sets the content view to activity\_main.xml.

MainActivityFragment.java defines the what should happen in the Fragment. The layout file fragment\_main.xml is inflated and displayed.

Taking a closer look, the Activity layout file has includes the content layout. The content layout is a Fragment and when the fragment code runs, it inflates the fragment layout.

The layout for the Fragment, fragment\_main.xml includes a TextView with the text “Hello World.” It is the default layout that you have seen for previous activities.

Using a Layout for Fragment Display

Fragments can be embedded directly into an XML layout for an activity. This is the approach that the generated code from Android Studio takes. You use the Fragment class name to identify the fragment. When the layout is inflated, the fragment class runs and the user interface is displayed. In the generated Android Studio code, the reference to MainActivityFragment in the layout means that the code for MainActivityFragment.java will run.

Content\_main.xml includes a direct reference to the Fragment class. It is important to note that the name of the class must be used. The full class name includes the package name. The full class name is com.talkingandroid.hour4fragmentexample.MainActivityFragment.

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

xmlns:tools="http://schemas.android.com/tools"

xmlns:app="http://schemas.android.com/apk/res-auto" android:id="@+id/fragment"

android:name="com.talkingandroid.activitywithfragment.MainActivityFragment"

android:layout\_width="match\_parent" android:layout\_height="match\_parent"

app:layout\_behavior="@string/appbar\_scrolling\_view\_behavior"

tools:layout="@layout/fragment\_main" />

In addition to including a fragment in the layout file, you can work with fragments directly using a FragmentTransaction.

A FragmentTransaction object is created with the method getSupportFragmentManager().

This snippet creates a FragmentTransaction that adds an ExampleFragment to layout that is specided by r.id.container.

getSupportFragmentManager().beginTransaction()

.add(R.id.container, new ExampleFragment())

.commit();

A FragmentTransaction object uses the following methods to manage fragments in a layout:

* Add(): Adds a Fragment to a layout.

Replace(): Replaces a Fragment within a layout.

Remove(): Removes a Fragment from a layout.

AddToBackStack(): Adds the set of commands in the transaction to the back stack. The fragment will not be destroyed when it is removed or replaced, it will be stopped. If the user navigates using the Back button, the fragment on the stack will be redisplayed.

SetTransition(): Sets an animation for the FragmentTransaction.

Commit(): Commits the transaction. This causes the transaction to take effect. The AddToBackStack() method is associated with all of the commands that are committed.