Fragments

A **Fragment**is a piece of an activity which enable more modular activity design. It will not be wrong if we say, a fragment is a kind of **sub-activity**.

* A fragment has its own layout and its own behaviour with its own life cycle callbacks.
* You can add or remove fragments in an activity while the activity is running.
* You can combine multiple fragments in a single activity to build a multi-pane UI.
* A fragment can be used in multiple activities.
* Fragment life cycle is closely related to the life cycle of its host activity which means when the activity is paused, all the fragments available in the activity will also be stopped.
* A fragment can implement a behaviour that has no user interface component.
* Fragments were added to the Android API in Honeycomb version of Android which API version 11.

## Fragment Life Cycle

Android fragments have their own life cycle very similar to an android activity. This section briefs different stages of its life cycle.



#### FRAGMENT LIFECYCLE

Here is the list of methods which you can to override in your fragment class −

* **onAttach()**The fragment instance is associated with an activity instance.The fragment and the activity is not fully initialized. Typically you get in this method a reference to the activity which uses the fragment for further initialization work.
* **onCreate()** The system calls this method when creating the fragment. You should initialize essential components of the fragment that you want to retain when the fragment is paused or stopped, then resumed.
* **onCreateView()** The system calls this callback when it's time for the fragment to draw its user interface for the first time. To draw a UI for your fragment, you must return a **View** component from this method that is the root of your fragment's layout. You can return null if the fragment does not provide a UI.
* **onActivityCreated()**The onActivityCreated() is called after the onCreateView() method when the host activity is created. Activity and fragment instance have been created as well as the view hierarchy of the activity. At this point, view can be accessed with the findViewById() method. example. In this method you can instantiate objects which require a Context object
* **onStart()**The onStart() method is called once the fragment gets visible.
* **onResume()**Fragment becomes active.
* **onPause()** The system calls this method as the first indication that the user is leaving the fragment. This is usually where you should commit any changes that should be persisted beyond the current user session.
* **onStop()**Fragment going to be stopped by calling onStop()
* **onDestroyView()**Fragment view will destroy after call this method
* **onDestroy()**onDestroy() called to do final clean up of the fragment's state but Not guaranteed to be called by the Android platform.

## How to use Fragments?

This involves number of simple steps to create Fragments.

* First of all decide how many fragments you want to use in an activity. For example let's we want to use two fragments to handle landscape and portrait modes of the device.
* Next based on number of fragments, create classes which will extend the *Fragment* class. The Fragment class has above mentioned callback functions. You can override any of the functions based on your requirements.
* Corresponding to each fragment, you will need to create layout files in XML file. These files will have layout for the defined fragments.
* Finally modify activity file to define the actual logic of replacing fragments based on your requirement.

**Single Frame Fragment**

Single frame fragment is designed for small screen devices such as hand hold devices(mobiles) and it should be above android 3.0 version.

## Example

This example will explain you how to create your own *Fragments*. Here we will create two fragments and one of them will be used when device is in landscape mode and another fragment will be used in case of portrait mode. So let's follow the following steps to similar to what we followed while creating *Hello World Example* −

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | You will use Android StudioIDE to create an Android application and name it as *MyFragments* under a package *com.example.myfragments*, with blank Activity. |
| 2 | Modify main activity file *MainActivity.java* as shown below in the code. Here we will check orientation of the device and accordingly we will switch between different fragments. |
| 3 | Create a two java files *PM\_Fragment.java* and *LM\_Fragement.java* under the package *com.example.myfragments* to define your fragments and associated methods. |
| 4 | Create layouts files *res/layout/lm\_fragment.xml* and *res/layout/pm\_fragment.xml* and define your layouts for both the fragments. |
| 5 | Modify the default content of *res/layout/activity\_main.xml* file to include both the fragments. |
| 6 | Define required constants in *res/values/strings.xml* file |
| 7 | Run the application to launch Android emulator and verify the result of the changes done in the application. |

Following is the content of the modified main activity file **MainActivity.java** −

package com.example.myfragments;

import android.app.Activity;

import android.app.FragmentManager;

import android.app.FragmentTransaction;

import android.content.res.Configuration;

import android.os.Bundle;

public class MainActivity extends Activity {

/\*\* Called when the activity is first created. \*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

Configuration config = getResources().getConfiguration();

FragmentManager fragmentManager = getFragmentManager();

FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();

/\*\*

\* Check the device orientation and act accordingly

\*/

if (config.orientation == Configuration.ORIENTATION\_LANDSCAPE) {

/\*\*

\* Landscape mode of the device

\*/

LM\_Fragement ls\_fragment = new LM\_Fragement();

fragmentTransaction.replace(android.R.id.content, ls\_fragment);

}else{

/\*\*

\* Portrait mode of the device

\*/

PM\_Fragement pm\_fragment = new PM\_Fragement();

fragmentTransaction.replace(android.R.id.content, pm\_fragment);

}

fragmentTransaction.commit();

}

}

Create two fragment files **LM\_Fragement.java** and **PM\_Fragment.java**

Following is the content of **LM\_Fragement.java** file −

package com.example.myfragments;

import android.app.Fragment;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

/\*\*

\* Created by TutorialsPoint7 on 8/23/2016.

\*/

public class LM\_Fragement extends Fragment {

@Override

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

/\*\*

\* Inflate the layout for this fragment

\*/

return inflater.inflate(R.layout.lm\_fragment, container, false);

}

}

Following is the content of **PM\_Fragement.java** file −

package com.example.myfragments;

import android.app.Fragment;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

/\*\*

\* Created by TutorialsPoint7 on 8/23/2016.

\*/

public class PM\_Fragement extends Fragment {

@Override

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

/\*\*

\* Inflate the layout for this fragment

\*/

return inflater.inflate(R.layout.pm\_fragment, container, false);

}

}

Create two layout files **lm\_fragement.xml** and **pm\_fragment.xml** under *res/layout* directory.

Following is the content of **lm\_fragement.xml** file −

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

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

android:orientation="vertical"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:background="#7bae16">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="@string/landscape\_message"

android:textColor="#000000"

android:textSize="20px" />

<!-- More GUI components go here -->

</LinearLayout>

Following is the content of **pm\_fragment.xml** file −

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

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

android:orientation="horizontal"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:background="#666666">

<TextView

android:layout\_width="fill\_parent"

android:layout\_height="wrap\_content"

android:text="@string/portrait\_message"

android:textColor="#000000"

android:textSize="20px" />

<!-- More GUI components go here -->

</LinearLayout>

Following will be the content of **res/layout/activity\_main.xml** file which includes your fragments −

<?xml version="1.0" encoding="utf-8"?>

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

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

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="horizontal">

<fragment

android:name="com.example.fragments"

android:id="@+id/lm\_fragment"

android:layout\_weight="1"

android:layout\_width="0dp"

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

<fragment

android:name="com.example.fragments"

android:id="@+id/pm\_fragment"

android:layout\_weight="2"

android:layout\_width="0dp"

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

</LinearLayout>

Make sure you have following content of **res/values/strings.xml** file −

<?xml version="1.0" encoding="utf-8"?>

<resources>

<string name="app\_name">My Application</string>

<string name="landscape\_message">This is Landscape mode fragment</string>

<string name="portrait\_message">This is Portrait mode fragment></string>

</resources>

Let's try to run our modified **MyFragments** application we just created. I assume you had created your **AVD** while doing environment set-up. To run the app from Android Studio, open one of your project's activity files and click Run icon from the tool bar. Android Studio installs the app on your AVD and starts it and if everything is fine with your set-up and application, it will display Emulator window where you will click on Menu button to see the following window. Be patience because it may take sometime based on your computer speed −



To change the mode of the emulator screen, let's do the following −

* **fn+control+F11** on Mac to change the landscape to portrait and vice versa.
* **ctrl+F11** on Windows.
* **ctrl+F11** on Linux.

Once you changed the mode, you will be able to see the GUI which you have implemented for landscape mode as below −



This way you can use same activity but different GUI's through different fragments. You can use different type of GUI components for different GUI's based on your requirements.

# list Fragment

Static library support version of the framework's ListFragment. Used to write apps that run on platforms prior to Android 3.0. When running on Android 3.0 or above, this implementation is still used.

**The basic implementation of list fragment is for creating list of items in fragments**



#### LIST IN FRAGMENTS

## Example

This example will explain you how to create your own list fragment based on arrayAdapter. So let's follow the following steps to similar to what we followed while creating Hello World Example −

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | You will use Android Studio to create an Android application and name it as *SimpleListFragment* under a package *com.example.tutorialspoint7.myapplication*, with blank Activity. |
| 2 | Modify the string file, which has placed at *res/values/string.xml* to add new string constants |
| 3 | Create a layout called *list\_fragment.xml* under the directory *res/layout* to define your list fragments. and add fragment tag(<fragment>) to your activity\_main.xml |
| 4 | Create a myListFragment.java, which is placed at *java/myListFragment.java* and it contained *onCreateView()*,*onActivityCreated()* and*OnItemClickListener()* |
| 5 | Run the application to launch Android emulator and verify the result of the changes done in the application. |

Before start coding i will initialize of the string constants inside *string.xml* file under *res/values directory*

<?xml version="1.0" encoding="utf-8"?>

<resources>

<string name="app\_name">ListFragmentDemo</string>

<string name="action\_settings">Settings</string>

<string name="hello\_world">Hello world!</string>

<string name="imgdesc">imgdesc</string>

<string-array name="Planets">

<item>Sun</item>

<item>Mercury</item>

<item>Venus</item>

<item>Earth</item>

<item>Mars</item>

<item>Jupiter</item>

<item>Saturn</item>

<item>Uranus</item>

<item>Neptune</item>

</string-array>

</resources>

Following will be the content of **res/layout/activity\_main.xml** file. it contained linear layout and fragment tag.

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<fragment

android:id="@+id/fragment1"

android:name="com.example.tutorialspoint7.myapplication.MyListFragment"

android:layout\_width="match\_parent"

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

</LinearLayout>

Following will be the content of **res/layout/list\_fragment.xml** file. it contained linear layout,list view and text view

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<ListView

android:id="@android:id/list"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" >

</ListView>

<TextView

android:id="@android:id/empty"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content" >

</TextView>

</LinearLayout>

following will be the content of **src/main/java/myListFragment.java**file.before writing to code, need to follow few steps as shown below

* Create a class MyListFragment and extend it to ListFragment.
* Inside the **onCreateView()** method , inflate the view with above defined list\_fragment xml layout.
* Inside the **onActivityCreated()** method , create a arrayadapter from resource ie using String array R.array.planet which you can find inside the string.xml and set this adapter to listview and also set the onItem click Listener.
* Inside the **OnItemClickListener()** method , display a toast message with Item name which is being clicked.

package com.example.tutorialspoint7.myapplication;

import android.annotation.SuppressLint;

import android.app.ListFragment;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.ViewGroup;

import android.widget.AdapterView;

import android.widget.AdapterView.OnItemClickListener;

import android.widget.ArrayAdapter;

import android.widget.Toast;

public class MyListFragment extends ListFragment implements OnItemClickListener {

@Override

public View onCreateView(LayoutInflater inflater,

ViewGroup container, Bundle savedInstanceState) {

View view = inflater.inflate(R.layout.list\_fragment, container, false);

return view;

}

@Override

public void onActivityCreated(Bundle savedInstanceState) {

super.onActivityCreated(savedInstanceState);

ArrayAdapter adapter = ArrayAdapter.createFromResource(getActivity(),

R.array.Planets, android.R.layout.simple\_list\_item\_1);

setListAdapter(adapter);

getListView().setOnItemClickListener(this);

}

@Override

public void onItemClick(AdapterView<?> parent, View view, int position,long id) {

Toast.makeText(getActivity(), "Item: " + position, Toast.LENGTH\_SHORT).show();

}

}

Following code will be the content of MainActivity.java

package com.example.tutorialspoint7.myapplication;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

public class MainActivity extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

}

following code will be the content of manifest.xml, which has placed at res/AndroidManifest.xml

<?xml version="1.0" encoding="utf-8"?>

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

package="com.example.tutorialspoint7.myapplication">

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:supportsRtl="true"

android:theme="@style/AppTheme">

<activity android:name=".MainActivity">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

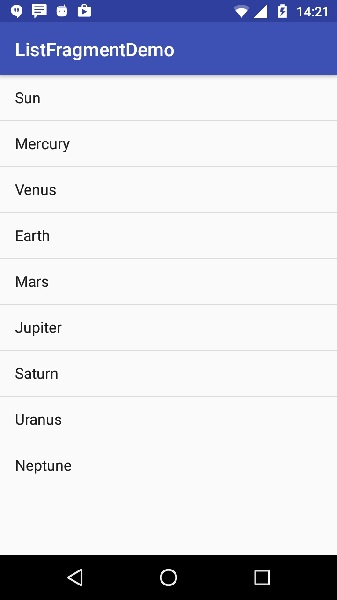
</activity>

</application>

</manifest>

## Running the Application

Let's try to run our **SimpleListFragment** application we just created. I assume you had created your **AVD** while doing environment set-up. To run the app from Android Studio, open one of your project's activity files and click Run Eclipse Run Icon icon from the toolbar. Android installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window −



# Fragment Transition

## What is a Transition?

Activity and Fragment transitions in Lollipop are built on top of a relatively new feature in Android called Transitions. Introduced in KitKat, the transition framework provides a convenient API for animating between different UI states in an application. The framework is built around two key concepts: scenes and transitions. A scene defines a given state of an application's UI, whereas a transition defines the animated change between two scenes.

When a scene changes, a Transition has two main responsibilities −

* Capture the state of each view in both the start and end scenes.
* Create an Animator based on the differences that will animate the views from one scene to the other.

## Example

This example will explain you how to create your custom animation with fragment transition . So let's follow the following steps to similar to what we followed while creating Hello World Example −

|  |  |
| --- | --- |
| **Step** | **Description** |
| 1 | You will use Android Studio to create an Android application and name it as *fragmentcustomanimations* under a package *com.example.fragmentcustomanimations*, with blank Activity. |
| 2 | Modify the activity\_main.xml, which has placed at *res/layout/activity\_main.xml* to add a Text View |
| 3 | Create a layout called *fragment\_stack.xml.xml* under the directory *res/layout* to define your fragment tag and button tag |
| 4 | Create a folder, which is placed at *res/* and name it as animation and add fragment\_slide\_right\_enter.xml fragment\_slide\_left\_exit.xml, ,fragment\_slide\_right\_exit.xml and fragment\_slide\_left\_enter.xml |
| 5 | In MainActivity.java, need to add fragment stack, fragment manager, and onCreateView() |
| 6 | Run the application to launch Android emulator and verify the result of the changes done in the application. |

following will be the content of *res.layout/activity\_main.xml* it contained TextView

<?xml version="1.0" encoding="utf-8"?>

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

android:id="@+id/text"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:gravity="center\_vertical|center\_horizontal"

android:text="@string/hello\_world"

android:textAppearance="?android:attr/textAppearanceMedium" />

Following will be the content of **res/animation/fragment\_stack.xml** file. it contained frame layout and button

<?xml version="1.0" encoding="utf-8"?>

<LinearLayout

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

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical" >

<fragment

android:id="@+id/fragment1"

android:name="com.pavan.listfragmentdemo.MyListFragment"

android:layout\_width="match\_parent"

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

</LinearLayout>

Following will be the content of **res/animation/fragment\_slide\_left\_enter.xml** file. it contained set method and object animator

<?xml version="1.0" encoding="utf-8"?>

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

<objectAnimator

android:interpolator="@android:interpolator/decelerate\_quint"

android:valueFrom="100dp" android:valueTo="0dp"

android:valueType="floatType"

android:propertyName="translationX"

android:duration="@android:integer/config\_mediumAnimTime" />

<objectAnimator

android:interpolator="@android:interpolator/decelerate\_quint"

android:valueFrom="0.0" android:valueTo="1.0"

android:valueType="floatType"

android:propertyName="alpha"

android:duration="@android:integer/config\_mediumAnimTime" />

</set>

following will be the content of **res/animation/fragment\_slide\_left\_exit.xml** file.it contained set and object animator tags.

<?xml version="1.0" encoding="utf-8"?>

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

<objectAnimator

android:interpolator="@android:interpolator/decelerate\_quint"

android:valueFrom="0dp" android:valueTo="-100dp"

android:valueType="floatType"

android:propertyName="translationX"

android:duration="@android:integer/config\_mediumAnimTime" />

<objectAnimator

android:interpolator="@android:interpolator/decelerate\_quint"

android:valueFrom="1.0" android:valueTo="0.0"

android:valueType="floatType"

android:propertyName="alpha"

android:duration="@android:integer/config\_mediumAnimTime" />

</set>

Following code will be the content of **res/animation/fragment\_slide\_right\_enter.xml**file.it contained set and object animator tags

<?xml version="1.0" encoding="utf-8"?>

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

<objectAnimator

android:interpolator="@android:interpolator/decelerate\_quint"

android:valueFrom="-100dp" android:valueTo="0dp"

android:valueType="floatType"

android:propertyName="translationX"

android:duration="@android:integer/config\_mediumAnimTime" />

<objectAnimator

android:interpolator="@android:interpolator/decelerate\_quint"

android:valueFrom="0.0" android:valueTo="1.0"

android:valueType="floatType"

android:propertyName="alpha"

android:duration="@android:integer/config\_mediumAnimTime" />

</set>

following code will be the content of **res/animation/fragment\_slide\_right\_exit.xml**file, it contained set and object animator tags

<?xml version="1.0" encoding="utf-8"?>

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

<objectAnimator

android:interpolator="@android:interpolator/decelerate\_quint"

android:valueFrom="0dp" android:valueTo="100dp"

android:valueType="floatType"

android:propertyName="translationX"

android:duration="@android:integer/config\_mediumAnimTime" />

<objectAnimator

android:interpolator="@android:interpolator/decelerate\_quint"

android:valueFrom="1.0" android:valueTo="0.0"

android:valueType="floatType"

android:propertyName="alpha"

android:duration="@android:integer/config\_mediumAnimTime" />

</set>

following code will be the content of **src/main/java/MainActivity.java** file. it contained button listener, stack fragment and onCreateView

package com.example.fragmentcustomanimations;

import android.app.Activity;

import android.app.Fragment;

import android.app.FragmentTransaction;

import android.os.Bundle;

import android.view.LayoutInflater;

import android.view.View;

import android.view.View.OnClickListener;

import android.view.ViewGroup;

import android.widget.Button;

import android.widget.TextView;

/\*\*

\* Demonstrates the use of custom animations in a FragmentTransaction when

\* pushing and popping a stack.

\*/

public class FragmentCustomAnimations extends Activity {

int mStackLevel = 1;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.fragment\_stack);

// Watch for button clicks.

Button button = (Button)findViewById(R.id.new\_fragment);

button.setOnClickListener(new OnClickListener() {

public void onClick(View v) {

addFragmentToStack();

}

});

if (savedInstanceState == null) {

// Do first time initialization -- add initial fragment.

Fragment newFragment = CountingFragment.newInstance(mStackLevel);

FragmentTransaction ft = getFragmentManager().beginTransaction();

ft.add(R.id.simple\_fragment, newFragment).commit();

}

else

{

mStackLevel = savedInstanceState.getInt("level");

}

}

@Override

public void onSaveInstanceState(Bundle outState) {

super.onSaveInstanceState(outState);

outState.putInt("level", mStackLevel);

}

void addFragmentToStack() {

mStackLevel++;

// Instantiate a new fragment.

Fragment newFragment = CountingFragment.newInstance(mStackLevel);

// Add the fragment to the activity, pushing this transaction

// on to the back stack.

FragmentTransaction ft = getFragmentManager().beginTransaction();

ft.setCustomAnimations(R.animator.fragment\_slide\_left\_enter,

R.animator.fragment\_slide\_left\_exit,

R.animator.fragment\_slide\_right\_enter,

R.animator.fragment\_slide\_right\_exit);

ft.replace(R.id.simple\_fragment, newFragment);

ft.addToBackStack(null);

ft.commit();

}

public static class CountingFragment extends Fragment {

int mNum;

/\*\*

\* Create a new instance of CountingFragment, providing "num"

\* as an argument.

\*/

static CountingFragment newInstance(int num) {

CountingFragment f = new CountingFragment();

// Supply num input as an argument.

Bundle args = new Bundle();

args.putInt("num", num);

f.setArguments(args);

return f;

}

/\*\*

\* When creating, retrieve this instance's number from its arguments.

\*/

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

mNum = getArguments() != null ? getArguments().getInt("num") : 1;

}

/\*\*

\* The Fragment's UI is just a simple text view showing its

\* instance number.

\*/

@Override

public View onCreateView(LayoutInflater inflater,

ViewGroup container,Bundle savedInstanceState) {

View v = inflater.inflate(R.layout.hello\_world, container, false);

View tv = v.findViewById(R.id.text);

((TextView)tv).setText("Fragment #" + mNum);

tv.setBackgroundDrawable(getResources().

getDrawable(android.R.drawable.gallery\_thumb));

return v;

}

}

}

following will be the content of**AndroidManifest.xml**

<?xml version="1.0" encoding="utf-8"?>

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

package="com.example.fragmentcustomanimations"

android:versionCode="1"

android:versionName="1.0" >

<application

android:allowBackup="true"

android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name"

android:theme="@style/AppTheme" >

<activity

android:name="com.example.fragmentcustomanimations.MainActivity"

android:label="@string/app\_name" >

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

</manifest>

## Running the Application

Let's try to run our **Fragment Transitions** application we just created. I assume you had created your **AVD** while doing environment set-up. To run the app from Android Studio, open one of your project's activity files and click Run Eclipse Run Icon icon from the toolbar. Android installs the app on your AVD and starts it and if everything is fine with your setup and application, it will display following Emulator window:



If click on new fragment, it going to be changed the first fragment to second fragment as shown below

