**[](http://www.truiton.com/)**

* [Android Tutorials](http://www.truiton.com/category/android/)
* [Expo Navigator](http://www.truiton.com/expo-navigator/)
* [Shop](http://www.truiton.com/shop/)
* [About Us](http://www.truiton.com/about-us/)
* [Contact Us](http://www.truiton.com/contact-us/)

[Home](http://www.truiton.com/) » [Android](http://www.truiton.com/category/android/) » Android RecyclerView Tutorial

**Android RecyclerView Tutorial**

[*27 Feb, 2015*](http://www.truiton.com/2015/02/27/)*in*[*Android*](http://www.truiton.com/category/android/)*tagged*[*Material Design*](http://www.truiton.com/tag/material-design/)*by*[*Mohit Gupt*](http://www.truiton.com/author/mohitgupt/)*(updated on*[*April 25, 2015*](http://www.truiton.com/2015/02/27/)*)*

[](https://www.flickr.com/photos/10037058@N08/3696670712)

One of the latest widgets introduced in Android is RecyclerView. Many say its a replacement of existing ListView widget, also termed as ListView 2.0. I browsed though the entire world wide web, in search of a working Android RecyclerView example with explanation. Gotta say couldn’t find one. After a lot of struggle I finally made one and decided to write a tutorial for it my self. So here it goes, in this Android RecyclerView Tutorial I would demonstrate a working example of RecyclerView, with some basic functionality. Where I would show you how to include a RecyclerView into an Android app project with an onItemClick event.

Android RecyclerView falls under the material design of android. Hence to maintain backward compatibility, first I would suggest you to go through my article on [maintaining backward compatibility of material design](http://www.truiton.com/2015/02/android-material-design-backward-compatibility/). I will try to keep this tutorial on Android RecyclerView short and simple as it consists a huge number of classes and each one of them is designed to be customized.

**Android RecyclerView: A basic Introduction**

Just like ListView, [RecyclerView](https://developer.android.com/reference/android/support/v7/widget/RecyclerView.html) is used to display a large amount of similar items on screen. But since the Android team was building an enhancement, they added a bunch of new features to RecyclerView. Each one of these new features, give a platform to developers for implementing a highly custom made RecyclerView. One of the custom implementations of RecyclerView is the new Gmail app on Android. I took inspiration from it and tried to make a subset of it, in this Android RecyclerView Tutorial. Below I will discuss the 4 key classes of RecyclerView but first lets add its dependency in build.gradle(:app):-

build.gradle



|  |  |
| --- | --- |
| 1  2  3  4 | dependencies {      compile 'com.android.support:appcompat-v7:21.0.3'      compile 'com.android.support:recyclerview-v7:21.0.+'  } |

**RecyclerView with different layouts**

One of the new concepts introduced in RecyclerView is the Layout managers. This class basically defines the type of layout which RecyclerView should use. In RecyclerView you can define three types of [LayoutManager](https://developer.android.com/reference/android/support/v7/widget/RecyclerView.LayoutManager.html) (s).

1. [LinearLayoutManager](https://developer.android.com/reference/android/support/v7/widget/LinearLayoutManager.html) – This LayoutManager can be used to display linear lists, they could be vertical or horizontal.
2. [GridLayoutManager](https://developer.android.com/reference/android/support/v7/widget/GridLayoutManager.html) – Earlier in android GridView was the only widget to display grids, but now with RecyclerView, GridLayoutManager can be used to display grids.
3. [StaggeredGridLayoutManager](https://developer.android.com/reference/android/support/v7/widget/StaggeredGridLayoutManager.html) – Another great customization is StaggeredGridLayoutManager, this is used to display a staggered grid.

In this Android RecyclerView Tutorial for simplicity I will be using a LinearLayoutManager for displaying a vertical list of items.

**Android RecyclerView ViewHolder**

The concept of RecyclerView.ViewHolder is same as it was in the ListView. Simply said: when a view goes out of visible area it is kept for recycling. Earlier this was done through the custom view holder pattern. But now with RecyclerView a [ViewHolder](https://developer.android.com/reference/android/support/v7/widget/RecyclerView.ViewHolder.html) class is included in the adapter by default. Therefore now its a compulsion for everyone to implement this class. There are many advantages to this approach, like it keeps references to the views in your items, which in turn decreases the overhead of creating new references every time an item is displayed.

Have a look at the object, that I will be using to supply data to ViewHolder:

DataObject.java

Java



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27 | package com.truiton.recyclerviewexample;    public class DataObject {      private String mText1;      private String mText2;        DataObject (String text1, String text2){          mText1 = text1;          mText2 = text2;      }        public String getmText1() {          return mText1;      }        public void setmText1(String mText1) {          this.mText1 = mText1;      }        public String getmText2() {          return mText2;      }        public void setmText2(String mText2) {          this.mText2 = mText2;      }  } |

Also have please have look at the layout where RecyclerView needs to be added:

activity\_recycler\_view.xml

XHTML



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"      xmlns:tools="http://schemas.android.com/tools"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      tools:context=".RecyclerViewActivity">        <android.support.v7.widget.RecyclerView          android:id="@+id/my\_recycler\_view"          android:layout\_width="match\_parent"          android:layout\_height="match\_parent"          android:scrollbars="vertical" />    </RelativeLayout> |

**Android RecyclerView Adapter**

In general ListView implementations, the adapters were used to bind views with positions, usually in the getView() method. As developers we used to bind data into it. Here in RecyclerView the concept is same. But now the difference is, that new view holders are placed inside the adapter and the adapter now binds with ViewHolder, instead of views, as in previous adapters. In a way a direct binding between ViewHolder and position is made. To understand the concept mode deeply have a look at my custom implementation of RecyclerView.Adapter :

MyRecyclerViewAdapter

Java



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81 | package com.truiton.recyclerviewexample;    import android.support.v7.widget.RecyclerView;  import android.util.Log;  import android.view.LayoutInflater;  import android.view.View;  import android.view.ViewGroup;  import android.widget.TextView;    import java.util.ArrayList;    public class MyRecyclerViewAdapter extends RecyclerView          .Adapter<MyRecyclerViewAdapter          .DataObjectHolder> {      private static String LOG\_TAG = "MyRecyclerViewAdapter";      private ArrayList<DataObject> mDataset;      private static MyClickListener myClickListener;        public static class DataObjectHolder extends RecyclerView.ViewHolder              implements View              .OnClickListener {          TextView label;          TextView dateTime;            public DataObjectHolder(View itemView) {              super(itemView);              label = (TextView) itemView.findViewById(R.id.textView);              dateTime = (TextView) itemView.findViewById(R.id.textView2);              Log.i(LOG\_TAG, "Adding Listener");              itemView.setOnClickListener(this);          }            @Override          public void onClick(View v) {              myClickListener.onItemClick(getPosition(), v);          }      }        public void setOnItemClickListener(MyClickListener myClickListener) {          this.myClickListener = myClickListener;      }        public MyRecyclerViewAdapter(ArrayList<DataObject> myDataset) {          mDataset = myDataset;      }        @Override      public DataObjectHolder onCreateViewHolder(ViewGroup parent,                                                 int viewType) {          View view = LayoutInflater.from(parent.getContext())                  .inflate(R.layout.recyclerview\_item, parent, false);            DataObjectHolder dataObjectHolder = new DataObjectHolder(view);          return dataObjectHolder;      }        @Override      public void onBindViewHolder(DataObjectHolder holder, int position) {          holder.label.setText(mDataset.get(position).getmText1());          holder.dateTime.setText(mDataset.get(position).getmText2());      }        public void addItem(DataObject dataObj, int index) {          mDataset.add(dataObj);          notifyItemInserted(index);      }        public void deleteItem(int index) {          mDataset.remove(index);          notifyItemRemoved(index);      }        @Override      public int getItemCount() {          return mDataset.size();      }        public interface MyClickListener {          public void onItemClick(int position, View v);      }  } |

**Android RecyclerView onItemClick Event**

Sadly the standard implementation of RecyclerView does not have an onItemClick implementation. Instead they support touch events by adding an OnItemTouchListener through the addOnItemTouchListener method of RecyclerView class.

But I believe the suggested implementation is a little lengthy to implement. Also you may need to detect click events on the views in a RecyclerView item. Which may be a little difficult to implement through the standard OnItemTouchListener. Therefore I have implemented a custom recyclerview onitemclick event.

Here in the above code sample, I defined a custom interface MyClickListener with a method onItemClick, which is used for detecting clicks on the activity. The only drawback of this approach is, that I had to make a static reference to MyClickListener, inside the adapter. Although it may not create any problems, but still to avoid them I suggest you to initialize this interface inonResume() method of your activity.

The above code implementation will detect clicks on the item. If you need to detect clicks on particular views. It can be done simply by modifying the code in DataObjectHolder constructor. The layout for RecyclerView item:

recycleriew\_item.xml

XHTML



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25 | <?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:background="?android:attr/selectableItemBackground"      android:orientation="vertical">        <TextView          android:id="@+id/textView"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:paddingLeft="@dimen/activity\_horizontal\_margin"          android:paddingTop="5dp"          android:text="Large Text"          android:textAppearance="?android:attr/textAppearanceLarge" />        <TextView          android:id="@+id/textView2"          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:paddingBottom="5dp"          android:paddingLeft="@dimen/activity\_horizontal\_margin"          android:text="Small Text"          android:textAppearance="?android:attr/textAppearanceSmall" />  </LinearLayout> |

To display visual responses like ripples on screen when a click event is detected add selectableItemBackground resource in the layout (highlighted above). Next have a look at the main activity where all of this code would come into play:

RecyclerViewActivity.java

Java



|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62 | package com.truiton.recyclerviewexample;    import android.os.Bundle;  import android.support.v7.app.ActionBarActivity;  import android.support.v7.widget.LinearLayoutManager;  import android.support.v7.widget.RecyclerView;  import android.util.Log;  import android.view.View;    import java.util.ArrayList;      public class RecyclerViewActivity extends ActionBarActivity {      private RecyclerView mRecyclerView;      private RecyclerView.Adapter mAdapter;      private RecyclerView.LayoutManager mLayoutManager;      private static String LOG\_TAG = "RecyclerViewActivity";        @Override      protected void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_recycler\_view);            mRecyclerView = (RecyclerView) findViewById(R.id.my\_recycler\_view);          mRecyclerView.setHasFixedSize(true);          mLayoutManager = new LinearLayoutManager(this);          mRecyclerView.setLayoutManager(mLayoutManager);          mAdapter = new MyRecyclerViewAdapter(getDataSet());          mRecyclerView.setAdapter(mAdapter);          RecyclerView.ItemDecoration itemDecoration =                  new DividerItemDecoration(this, LinearLayoutManager.VERTICAL);          mRecyclerView.addItemDecoration(itemDecoration);            // Code to Add an item with default animation          //((MyRecyclerViewAdapter) mAdapter).addItem(obj, index);            // Code to remove an item with default animation          //((MyRecyclerViewAdapter) mAdapter).deleteItem(index);      }        @Override      protected void onResume() {          super.onResume();          ((MyRecyclerViewAdapter) mAdapter).setOnItemClickListener(new              MyRecyclerViewAdapter.MyClickListener() {              @Override              public void onItemClick(int position, View v) {                  Log.i(LOG\_TAG, " Clicked on Item " + position);              }          });      }        private ArrayList<DataObject> getDataSet() {          ArrayList results = new ArrayList<DataObject>();          for (int index = 0; index < 20; index++) {              DataObject obj = new DataObject("Some Primary Text " + index,                      "Secondary " + index);              results.add(index, obj);          }          return results;      }  } |

In the above code sample I have defined a basic Android RecyclerView. Where I have also shown, how to add and remove items at run time from the adapter. The good part here is, you need not to worry about animation. They are added and removed with default animation. But the most annoying thing is how to add dividers and spaces between items in RecyclerView?

**Android RecyclerView Spacing Between Items**

One of the biggest problems you might face while implementing Android RecyclerView is the spacing between items. Yes its true RecyclerView by default does not give any support for displaying dividers between items. Although you can add dividers by doing a custom implementation of RecyclerView.ItemDecoration class. Luckily I found a piece of of code in official samples. Link for the file:

[DividerItemDecoration.java](https://chromium.googlesource.com/android_tools/+/18728e9dd5dd66d4f5edf1b792e77e2b544a1cb0/sdk/extras/android/support/samples/Support7Demos/src/com/example/android/supportv7/widget/decorator/DividerItemDecoration.java)

After including divider decoration file and using it through the addItemDecoration method, your app would look something like this:

Video Player

In the above tutorial, I tried to make a RecyclerView inspired from the Gmail app. Although its not a complete implantation, but it gives a basic idea on how to use RecyclerView in Android. Hope this [Android RecyclerView Tutorial](http://www.truiton.com/2015/02/android-recyclerview-tutorial/) helped you. Connect with us through Facebook, Twitter and Google+ for more updates.

You may also like:

**[Wanna Know How to Make Rs. 6,302/Day?](http://engine.adzerk.net/r?e=&s=Q5Ajufka1zu493Jl8M2WGNRWvYg" \t "_blank)**

[Career Times](http://engine.adzerk.net/r?e=&s=Q5Ajufka1zu493Jl8M2WGNRWvYg" \t "_blank)

**[Android pick date time from EditText OnClick event](http://www.truiton.com/2013/03/android-pick-date-time-from-edittext-onclick-event/" \t "_self)**

**[Android Floating Label EditText](http://www.truiton.com/2015/06/android-floating-label-edittext/" \t "_self)**

**[Android FragmentPagerAdapter Example](http://www.truiton.com/2013/05/android-fragmentpageradapter-example/" \t "_self)**

**[Android Foreground Service Example](http://www.truiton.com/2014/10/android-foreground-service-example/" \t "_self)**

**[IITF 2014 Mobile app for Pragati Maidan Complex](http://www.truiton.com/2013/11/iitf-2014-mobile-app-pragati-maidan-complex/" \t "_self)**

.

**About**[**Mohit Gupt**](http://www.truiton.com/author/mohitgupt/)

An android enthusiast, and an iPhone user with a keen interest in development of innovative applications.

* [Web](http://www.truiton.com/)

 |

* [Twitter](http://twitter.com/mohitgupt)

 |

* [Facebook](https://www.facebook.com/Mr.Mohit.Gupt)

 |

* [Google+](https://www.google.com/+MohitGupt)

 |

* [More Posts (69)](http://www.truiton.com/author/mohitgupt/)

**Share this:**

* [10Click to share on Facebook (Opens in new window)10](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?share=facebook&nb=1)
* [Click to share on LinkedIn (Opens in new window)](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?share=linkedin&nb=1)
* [Click to share on Reddit (Opens in new window)](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?share=reddit&nb=1)
* [Click to share on Twitter (Opens in new window)](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?share=twitter&nb=1)
* [Click to share on Google+ (Opens in new window)](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?share=google-plus-1&nb=1)
* [1Click to share on Pinterest (Opens in new window)1](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?share=pinterest&nb=1)
* [Click to share on Pocket (Opens in new window)](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?share=pocket&nb=1)
* [Click to share on Tumblr (Opens in new window)](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?share=tumblr&nb=1)
* [More](http://www.truiton.com/2015/02/android-recyclerview-tutorial/)

**Leave a comment**

Top of Form

Your email address will not be published. Required fields are marked \*

Comment

Name \*

Email \*

Website



 Notify me of new posts by email.

Bottom of Form

**8 thoughts on “Android RecyclerView Tutorial”**

* 

[Reply ↓](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?replytocom=6658#respond)

Akash[*February 28, 2015 at 2:10 am*](http://www.truiton.com/2015/02/android-recyclerview-tutorial/#comment-6658)

You might want to mention something about<https://developer.android.com/reference/android/support/v7/widget/RecyclerView.ItemAnimator.html>

Definitely a very useful thing

* + 

[Reply ↓](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?replytocom=6664#respond)

[Mohit Gupt](http://www.truiton.com/) Post author[*February 28, 2015 at 7:13 pm*](http://www.truiton.com/2015/02/android-recyclerview-tutorial/#comment-6664)

Hi Akash,

Thanks for the suggestion.  
Will include in next tut.

* 

[Reply ↓](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?replytocom=7215#respond)

Tshepo[*May 31, 2015 at 8:41 pm*](http://www.truiton.com/2015/02/android-recyclerview-tutorial/#comment-7215)

Thanks for this cool tutorial.

I keep getting a nullpointer exception on the “myClickListener”, where do I assign it a value?

* + 

[Reply ↓](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?replytocom=7216#respond)

[Mohit Gupt](http://www.truiton.com/) Post author[*May 31, 2015 at 10:43 pm*](http://www.truiton.com/2015/02/android-recyclerview-tutorial/#comment-7216)

Hi,

Have a look at the onResume method of RecyclerViewActivity. It is set using the setOnItemClickListener method of MyRecyclerViewAdapter.  
Hope this helps.

-Mohit

* 

[Reply ↓](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?replytocom=7686#respond)

Anish Panthi[*July 20, 2015 at 7:41 am*](http://www.truiton.com/2015/02/android-recyclerview-tutorial/#comment-7686)

How can we add Header in RecyclerView? ListView has addHeaderView() method to add the header. Is there any way to add same in RecyclerView?

* 

[Reply ↓](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?replytocom=7848#respond)

hornet2319[*August 3, 2015 at 2:10 am*](http://www.truiton.com/2015/02/android-recyclerview-tutorial/#comment-7848)

how to create custom divider?

* + 

[Reply ↓](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?replytocom=7849#respond)

[Mohit Gupt](http://www.truiton.com/) Post author[*August 3, 2015 at 9:40 am*](http://www.truiton.com/2015/02/android-recyclerview-tutorial/#comment-7849)

you can create your own DividerItemDecoration.java and use it.

* 

[Reply ↓](http://www.truiton.com/2015/02/android-recyclerview-tutorial/?replytocom=9246#respond)

Shivam Goel[*January 4, 2016 at 4:16 pm*](http://www.truiton.com/2015/02/android-recyclerview-tutorial/#comment-9246)

Thanks for such an awesome article man.

**Post navigation**

* [← Android Material Design With Backward Compatibility](http://www.truiton.com/2015/02/android-material-design-backward-compatibility/)
* [Android RecyclerView vs ListView | Comparison →](http://www.truiton.com/2015/03/android-recyclerview-vs-listview-comparison/)

Top of Form

Search for: 

Bottom of Form

**Recent Posts**

* [Android Activity To Fragment Communication](http://www.truiton.com/2015/12/android-activity-fragment-communication/)
* [Android Floating Label EditText](http://www.truiton.com/2015/06/android-floating-label-edittext/)
* [Android Snackbar Example](http://www.truiton.com/2015/06/android-snackbar-example/)
* [Android Tabs Example – With Fragments and ViewPager](http://www.truiton.com/2015/06/android-tabs-example-fragments-viewpager/)
* [Android Palette – Pick colors from images](http://www.truiton.com/2015/05/android-palette-pick-colors-images/)

**Recent Comments**

* [Mohit Gupt](http://www.truiton.com/) on [Android Tabs Example – With Fragments and ViewPager](http://www.truiton.com/2015/06/android-tabs-example-fragments-viewpager/#comment-9549)
* [Mohit Gupt](http://www.truiton.com/) on [Capture and Record Android Screen using MediaProjection APIs](http://www.truiton.com/2015/05/capture-record-android-screen-using-mediaprojection-apis/#comment-9548)
* praveen on [Android Tabs Example – With Fragments and ViewPager](http://www.truiton.com/2015/06/android-tabs-example-fragments-viewpager/#comment-9547)
* Reshma on [Capture and Record Android Screen using MediaProjection APIs](http://www.truiton.com/2015/05/capture-record-android-screen-using-mediaprojection-apis/#comment-9537)
* aleksandar on [Android SwitchCompat Example](http://www.truiton.com/2015/03/android-switch-button-example/#comment-9533)

**Sponsored By:**

**Archives**

* [December 2015](http://www.truiton.com/2015/12/)
* [June 2015](http://www.truiton.com/2015/06/)
* [May 2015](http://www.truiton.com/2015/05/)
* [April 2015](http://www.truiton.com/2015/04/)
* [March 2015](http://www.truiton.com/2015/03/)
* [February 2015](http://www.truiton.com/2015/02/)
* [January 2015](http://www.truiton.com/2015/01/)
* [December 2014](http://www.truiton.com/2014/12/)
* [November 2014](http://www.truiton.com/2014/11/)
* [October 2014](http://www.truiton.com/2014/10/)
* [September 2014](http://www.truiton.com/2014/09/)
* [August 2014](http://www.truiton.com/2014/08/)
* [June 2014](http://www.truiton.com/2014/06/)
* [May 2014](http://www.truiton.com/2014/05/)
* [January 2014](http://www.truiton.com/2014/01/)
* [November 2013](http://www.truiton.com/2013/11/)
* [August 2013](http://www.truiton.com/2013/08/)
* [July 2013](http://www.truiton.com/2013/07/)
* [June 2013](http://www.truiton.com/2013/06/)
* [May 2013](http://www.truiton.com/2013/05/)
* [April 2013](http://www.truiton.com/2013/04/)
* [March 2013](http://www.truiton.com/2013/03/)
* [October 2012](http://www.truiton.com/2012/10/)

**Categories**

* [Android](http://www.truiton.com/category/android/)
* [iPhone](http://www.truiton.com/category/iphone/)
* [News](http://www.truiton.com/category/news/)
* [Uncategorized](http://www.truiton.com/category/uncategorized/)

**Follow Truiton**

**Follow Mohit Gupt**

**Funny Sticker Store**

[[](http://www.truiton.com/product/bug-in-my-code-sticker/)   
  
**Bug In My Code Sticker**   
$2.99 (Free Shipping Worldwide)](http://www.truiton.com/product/bug-in-my-code-sticker/)

**Recent Comments**

* [Mohit Gupt](http://www.truiton.com/) on [Android Tabs Example – With Fragments and ViewPager](http://www.truiton.com/2015/06/android-tabs-example-fragments-viewpager/#comment-9549)
* [Mohit Gupt](http://www.truiton.com/) on [Capture and Record Android Screen using MediaProjection APIs](http://www.truiton.com/2015/05/capture-record-android-screen-using-mediaprojection-apis/#comment-9548)
* praveen on [Android Tabs Example – With Fragments and ViewPager](http://www.truiton.com/2015/06/android-tabs-example-fragments-viewpager/#comment-9547)
* Reshma on [Capture and Record Android Screen using MediaProjection APIs](http://www.truiton.com/2015/05/capture-record-android-screen-using-mediaprojection-apis/#comment-9537)
* aleksandar on [Android SwitchCompat Example](http://www.truiton.com/2015/03/android-switch-button-example/#comment-9533)

**Recent Posts**

* [Android Activity To Fragment Communication](http://www.truiton.com/2015/12/android-activity-fragment-communication/)
* [Android Floating Label EditText](http://www.truiton.com/2015/06/android-floating-label-edittext/)
* [Android Snackbar Example](http://www.truiton.com/2015/06/android-snackbar-example/)
* [Android Tabs Example – With Fragments and ViewPager](http://www.truiton.com/2015/06/android-tabs-example-fragments-viewpager/)
* [Android Palette – Pick colors from images](http://www.truiton.com/2015/05/android-palette-pick-colors-images/)

**Search Truiton.com**

Top of Form

Search for: 

Bottom of Form

· © 2016 [Truiton](http://www.truiton.com/) · All Rights Reserved. ·  
· [Privacy Policy](http://www.truiton.com/privacy-policy/) | [Join Us](http://www.truiton.com/join-us/) ·