

Extract text on image and translate android app source code is available now.

See example in Play Store



Inducesmile

Mobile developer community for beginners and students

[ANDROID](#)
[KOTLIN](#)
[IOS \(SWIFT\)](#)
[HYBRID APP ▾](#)
[MOBILE GAMES](#)
[IOT - VR](#)
[TIPS](#)
[CODING HELP](#)
[BUY CODE](#)

[Home](#) ▶ [Android](#) ▶ Add Header To Android RecyclerView



ADD HEADER TO ANDROID RECYCLERVIEW

In this tutorial, we are going to learn how to add header to android recyclerview. If you have worked with Android ListView, we will notice that there is a build in addHeaderView () method that you can easily use to add a header to ListView. This is not the case when you work with Android RecyclerView.

There might may few ways to achieve this but in this add header to android recyclerview tutorial, we are going use ViewType property to achieve this.

If you have done it in a different way, we will be glad if you can share your solution with us in the comment section.

Another difficult thing to achieve when you work with android RecyclerView is the ability to create multiple sections. Although there are many Android third party libraries you can use. If you are



Scan & Translate Android App
Source code

[VIEW APP SAMPLE](#)

[WRITE FOR US](#)

ANDROID SOURCE CODE POLL

Which of the below listed source codes should we publish next month?

Multi Restaurant Food Ordering App (27%, 39 Votes)

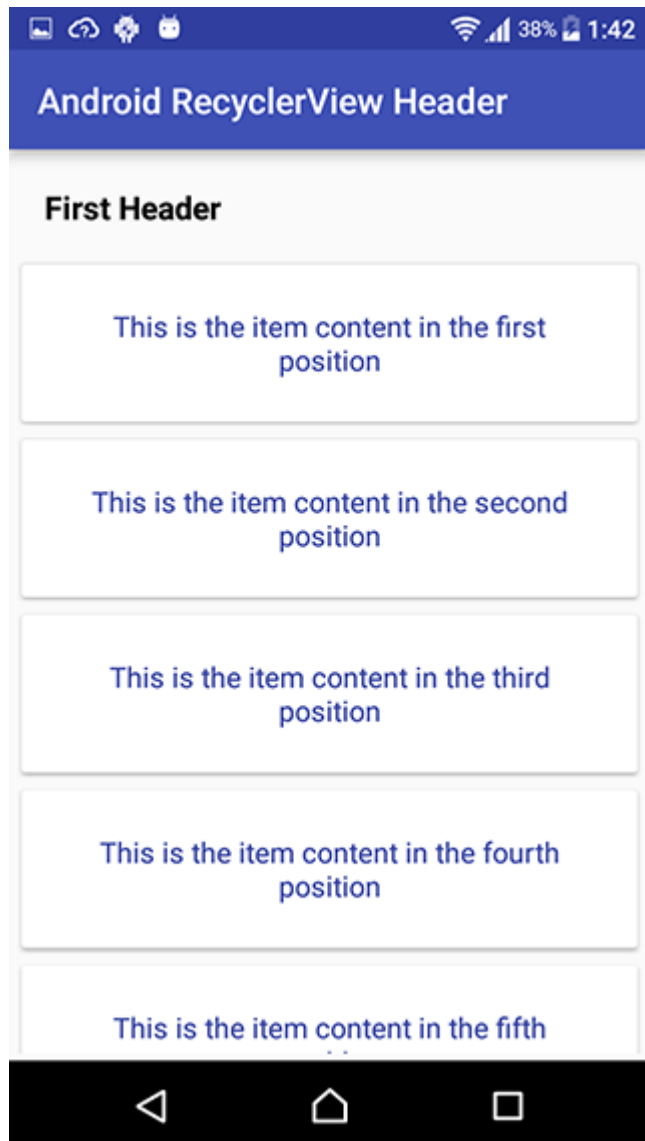
Stock and Inventory App (18%, 26 Votes)

Android Dating App (16%, 23

Accept

Adding header to android recyclerview will involved creating two different android ViewHolder instances. One for the HeaderView and the other for the ItemView.

To understand what we plan to achieve in this tutorial, I have added a screen shot for the application.



CREATE NEW ANDROID PROJECT

Lets start to soil our hands in code. Start up your IDE. For this tutorial, I am using the following tools and environment, feel free to use what

Android E-Book App (12%, 17 Votes)



Appointment Booking App (10%, 15 Votes)



Business & Service Provider Finder (6%, 9 Votes)



Total Voters: **147**

LIKE US ON FACEBOOK



Induce smile

5,305 likes

Like Page

Be the first of your friends to like this

RECENT POSTS

- Camera Translator All Language App
- Why MVVM and How to execute MVVM combined with Data Binding and LiveData in Four Simple Steps
- Camera Translator All Languages – Android Project idea source code for students
- Note Scan Android App -Project idea source code for students
- How To Build Android WhatsApp Clone Chat Application
- Android Car Rental App with Web Admin Panel – Project Idea Source Code For Students

Accept

Android Studio

Sony F model (Android 6)

Min SDK 14

Target SDK 24

- Android Retrofit 2 with JSON API Example
- Android Room with LiveData Example Tutorial
- Android UI App for Food Ordering and Delivery from Multiple Restaurants

To create a new android application project, follow the steps as stipulated below.

Go to File menu

Click on New menu

Click on Android Application

Enter Project name: AndroidRecyclerViewHeader

Package: com.inducesmile.androidrecyclerviewheader

Select Blank Activity

Name your activity: MainActivity

Keep other default selections

Continue to click on next button until Finish button is active, then click on Finish Button.

BUILD.GRADLE

Since we are going to use Android CardView, we are going to add the library in our project to enable us access android cradview. Open your app level build.gradle and add the code below to the file.

```
apply plugin: 'com.android.application'
```



Accept

```
buildToolsVersion "24.0.0"
defaultConfig {
    applicationId "com.inducesmile.androidrecyclervi
    minSdkVersion 14
    targetSdkVersion 24
    versionCode 1
    versionName "1.0"
}
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('progua
    }
}
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:24.1.1'
    compile 'com.android.support:design:24.1.1'
    compile 'com.android.support:support-v4:24.1.1'
    compile 'com.android.support:cardview-v7:24.1.1'
}
```

STRINGS.XML

We are going to update our project strings.xml file located in the values folder inside the res folder. Open the file and add the code below to it.

```
<resources>
    <string name="app_name">Android RecyclerView Header<
    <string name="sample">Lorem Ipsum is simply dummy te
</resources>
```

COLORS.XML

Accept

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <color name="colorBlack">#000000</color>
</resources>
```

CUSTOMRECYCLERVIEWWADAPTER.JAVA

We are going to create Recyclerview adapter. The adapter will be use to bind our data source to the itemview of the RecyclerView.

In the project package folder, right click and select create a new java file. Name the file CustomRecyclerViewAdapter.java.

We will initialize two class member variables that will hold the a value for the HeaderView and ItemView.

In the onCreateViewHolder(ViewGroup parent, int viewType), we will use the viewType to differentiate between the headerView and the itemview.

Each instance of the ViewHolder class is created and the view object is passed as a parameter.

In onBindViewHolder(RecyclerView.ViewHolder holder, int position), we check if the holder is an instance of the HeaderView or the ItemView before using it.

The complete code for the class is as shown below.

```
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import java.util.List;
```

Accept

```
private List<ItemObject> itemObjects;
public CustomRecyclerViewAdapter(List<ItemObject> it
    this.itemObjects = itemObjects;
}
@Override
public RecyclerView.ViewHolder onCreateViewHolder(Vi
    if (viewType == TYPE_HEADER) {
        View layoutView = LayoutInflater.from(parent
        return new HeaderViewHolder(layoutView);
    } else if (viewType == TYPE_ITEM) {
        View layoutView = LayoutInflater.from(parent
        return new ItemViewHolder(layoutView);
    }
    throw new RuntimeException("No match for " + vie
}
@Override
public void onBindViewHolder(RecyclerView.ViewHolder
    ItemObject mObject = itemObjects.get(position);
    if(holder instanceof HeaderViewHolder){
        ((HeaderViewHolder) holder).headerTitle.setT
    }else if(holder instanceof ItemViewHolder){
        ((ItemViewHolder) holder).itemContent.setTex
    }
}
private ItemObject getItem(int position) {
    return itemObjects.get(position);
}
@Override
public int getItemCount() {
    return itemObjects.size();
}
@Override
public int getItemViewType(int position) {
    if (isPositionHeader(position))
        return TYPE_HEADER;
    return TYPE_ITEM;
}
private boolean isPositionHeader(int position) {
    return position == 0;
}
}
```

Accept

LAYOUT FILES FOR THE ADAPTER

If you take a close look at the CustomRecyclerViewAdapter, you will see that we created two different layout file for the headerView and itemView. This is important in situations where the header will appear and look different from the items.

We are going to create the two different layout files.

HEADER_LAYOUT.XML

The header_layout.xml is responsible for the header UI appearance. In this case, it will contain a single TextView widget which will server as a header title.

Create a new layout file inside the layout folder and name it header_layout.xml. Copy and add the code below to the file.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="16sp"
    android:orientation="vertical">
    <TextView
        android:id="@+id/header_id"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="18sp"
        android:textStyle="bold"
        android:textColor="@color/colorBlack"
        android:text="@string/app_name"/>
</LinearLayout>
```

ITEM_LAYOUT.XML

Accept

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">
    <android.support.v7.widget.CardView xmlns:android="h
        xmlns:card_view="http://schemas.android.com/apk/
        android:id="@+id/direction_card_view"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="2dp"
        card_view:cardElevation="2dp"
        card_view:cardUseCompatPadding="true">
        <TextView
            android:id="@+id/item_content"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:padding="24dp"
            android:textColor="@color/colorPrimaryDark"
            android:textSize="16sp"
            android:gravity="center"
            android:text="@string/sample" />
        </android.support.v7.widget.CardView>
    </LinearLayout>

```

VIEWHOLDER CLASSES

Just like with the layout files, we are going to create two different ViewHolder classes which our adapter will make use of. This is the trick with adding header to android RecyclerView.

Let create our two ViewHolder classes now.

HEADERVIEWHOLDER.JAVA

In the project package folder, create a new file and name it

Accept


```
import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.widget.TextView;
public class HeaderViewHolder extends RecyclerView.ViewHolder
    public TextView headerTitle;
    public HeaderViewHolder(View itemView) {
        super(itemView);
        headerTitle = (TextView)itemView.findViewById(R.
    }
}
```

ITEMVIEWHOLDER.JAVA

In the project package folder, create a new file and name it ItemViewHolder.java. Open the file and add the code below to it.

```
import android.support.v7.widget.RecyclerView;
import android.view.View;
import android.widget.TextView;
public class ItemViewHolder extends RecyclerView.ViewHolder
    public TextView itemContent;
    public ItemViewHolder(View itemView) {
        super(itemView);
        itemContent = (TextView)itemView.findViewById(R.
    }
}
```

ITEMOBJECT.JAVA

We are going to wrap our data in an object so we need to create a new java file and we will name it ItemObject.java. Since it is an entity object it will contain only a constructor and get method.

Open the file and add the code below.

Accept

```
        this.contents = contents;
    }
    public String getContents() {
        return contents;
    }
}
```

MAINACTIVITY CLASS AND ITS LAYOUT FILE

We have create more of the layout files and classes we will use in this application. Now we will focus on how to integrate and bring everything together to achieve the aim we had from the beginning.

We will first start with the main layout file.

ACTIVITY_MAIN.XML

The activity_main.xml is the main layout and it is used to display the UI view of the MainActivity class.

It is a simple layout file that will contain a RecyclerView widget. Open the layout file and add the code below.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.co
xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_margin="4dp"
    tools:context="com.inducesmile.androidrecyclerviewhe
    <android.support.v7.widget.RecyclerView
        android:id="@+id/add_header"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">
```

Accept

MAINACTIVITY.JAVA

The MainActivity class will obtain an instance of the RecyclerView widget. It will call its setLayoutManager() method and we will pass an instance of a LinearLayoutManager.

Then, we will create a instance of our CustomRecyclerViewAdapter and pass it a parameter in setAdapter() method of the RecyclerView class.

Finally, we will prepare a data source that we will feed to our custom adapter object as one of its parameters.

The complete code for this class is shown below.

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import java.util.ArrayList;
import java.util.List;
public class MainActivity extends AppCompatActivity {
    private static final String TAG = MainActivity.class
    private RecyclerView addHeaderRecyclerView;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        addHeaderRecyclerView = (RecyclerView)findViewByIdBy
        LinearLayoutManager layoutManager = new Li
        addHeaderRecyclerView.setLayoutManager(linearLayoutManager);
        addHeaderRecyclerView.setHasFixedSize(true);
        CustomRecyclerViewAdapter customAdapter = new Cu
        addHeaderRecyclerView.setAdapter(customAdapter);
    }
    private List<ItemObject> getDataSource(){
        List<ItemObject> data = new ArrayList<ItemObject>
        data.add(new ItemObject("First Header"));
        data.add(new ItemObject("This is the item conten
        data.add(new ItemObject("This is the item conten
```

Accept

```
}  
}  
◀ ▶
```

This brings us to the end of this tutorial. I hope that you have learn something. Run your app and see for yourself.

You can download the code for this tutorial below. If you are having hard time downloading the tutorial, kindly contact me.

Remember to subscribe with your email address to be among the first to receive my new android blog post once it is published.

OTHER INTERESTING POSTS:

Android Dictionary
Application with Search
Suggest and Text to
Speech Feature

Android Navigation Drawer
with Material Design

How to create Android Unit
Converter – Unit Converter
Example Tutorial

Android Light Sensor -
Environment Sensors

Accept

Populate Android Spinner
Data from Remote MYSQL
Database

Using Braintree Payment in
Android to accept Payment

Tags: [Add Header To Android RecyclerView](#), [Android HeaderView](#),
[headerView in Recyclerview](#)


ABOUT THE AUTHOR



Inducesmile

I learn and write about Android, iOS, Javascript, Php, Node.js, React Native, Mobile Game, Virtual Reality and Internet of Things



 **Inducesmile**
1,932
● 1 ● 9 ● 15

2 COMMENTS



Malavan | October 7, 2017

[Log in to Reply](#)

Thank you..... very much for this tutorial....



nasir | April 8, 2018

[Log in to Reply](#)

Sir...how can i download this tutorial?

ADD A COMMENT

Accept



INDUCESMILE – ANDROID TUTORIAL, ANDROID APPS, ANDROID STUDIO, ANDROID SDK, ANDROID DEVELOPMENT COPYRIGHT © 2018.

[PROGRAMMING HELP](#)

[ANDROID TUTORIALS](#)

[CONTACT US](#)

[PRIVACY POLICY](#)

Accept