Downloads | Tips | Write for Us

HOME > MATERIAL DESIGN > Android Getting Started with Material Design

Android Getting Started with Material Design

by Ravi Tamada / April 11, 2015 / 437 Comments

You might have heard of android Material Design which was introduced in Android Lollipop version. In Material Design lot of new things were introduced like **Material Theme**, new **widgets**, **custom shadows**, **vector drawables** and custom **animations**. If you haven't working on Material Design yet, this article will give you a good start.

In this tutorial we are going to learn the basic steps of Material Design development i.e writing the custom theme and implementing the navigation drawer using the RecyclerView.

Go through the below links to get more knowledge over Material Design.

> Material Design Specifications

> Creating Apps with Material Design

SEARCH HERE

Search the site

WE'RE SOCIAL

AndroidHive
36,879 likes

Like Page

Be the first of your friends to li

Subscribe to Newsletter

Join our 746,498 subscribers and get access to the latest android tutorials, freebies, scripts and much more!

1. Downloading Android Studio

Before going further, download the Android Studio and do the necessary setup as I am going to use Android Studio for all my tutorial from now on. If you are trying the Android Studio for the first time, go the overview doc to get complete overview of android studio.

2. Material Design Color Customization

Material Design provides set of properties to customize the Material

Design Color theme. But we use five primary attributes to customize overall theme.

`colorPrimaryDark` – This is darkest primary color of the app mainly applies to notification bar background.

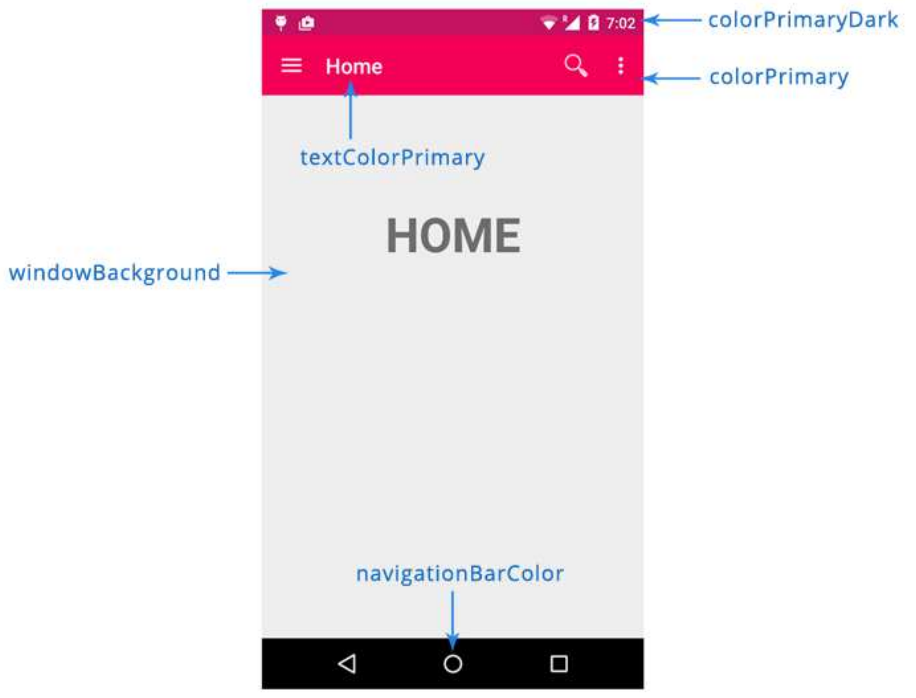
`colorPrimary` – This is the primary color of the app. This color will be applied as toolbar background.

`textColorPrimary` – This is the primary color of text. This applies to toolbar title.

`windowBackground` – This is the default background color of the app.

`navigationBarColor` – This color defines the background color of footer navigation bar.

Android Material Design



www.androidhive.info

You can go through this material design color patterns and choose the one that suits your app.

3. Creating Material Design Theme

1. In Android Studio, go to **File** ⇒ **New Project** and fill all the details

7. Android Login and Registration with PHP, MySQL and SQLite - 915,071 views

8. Android GPS, Location Manager Tutorial - 696,482 views

9. Android Tab Layout with Swipeable Views - 662,815 views

10. Android working with Google Maps V2 - 587,539 views

required to create a new project. When it prompts to select a default activity, select **Blank Activity** and proceed.

2. Open **res** ⇒ **values** ⇒ **strings.xml** and add below string values.

```
strings.xml
<resources>
    <string name="app_name">Material Design</string>
    <string name="action_settings">Settings</string>
    <string name="action_search">Search</string>
    <string name="drawer_open">Open</string>
    <string name="drawer_close">Close</string>

    <string name="nav_item_home">Home</string>
    <string name="nav_item_friends">Friends</string>
    <string name="nav_item_notifications">Messages</s

    <!-- navigation drawer item labels  -->
    <string-array name="nav_drawer_labels">
        <item>@string/nav_item_home</item>
        <item>@string/nav_item_friends</item>
        <item>@string/nav_item_notifications</item>
    </string-array>

    <string name="title_messages">Messages</string>
    <string name="title_friends">Friends</string>
    <string name="title_home">Home</string>
</resources>
```

3. Open **res** ⇒ **values** ⇒ **colors.xml** and add the below color values. If you don't find colors.xml, create a new resource file with the name.

```
colors.xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#F50057</color>
    <color name="colorPrimaryDark">#C51162</color>
    <color name="textColorPrimary">#FFFFFF</color>
    <color name="windowBackground">#FFFFFF</color>
    <color name="navigationBarColor">#000000</color>
    <color name="colorAccent">#FF80AB</color>
</resources>
```

4. Open **res** ⇒ **values** ⇒ **dimens.xml** and add below dimensions.

```
dimens.xml
<resources>
    <!-- Default screen margins, per the Android Desi
    <dimen name="activity_horizontal_margin">16dp</di
    <dimen name="activity_vertical_margin">16dp</dime
    <dimen name="nav_drawer_width">260dp</dimen>
</resources>
```

5. Open **styles.xml** under **res** ⇒ **values** and add below styles. The styles defined in this styles.xml are common to all the android versions. Here I am naming my theme as **MyMaterialTheme**.

```
styles.xml
<resources>

    <style name="MyMaterialTheme" parent="MyMaterialT

    </style>

    <style name="MyMaterialTheme.Base" parent="Theme.
        <item name="windowNoTitle">true</item>
        <item name="windowActionBar">false</item>
        <item name="colorPrimary">@color/colorPrimary
        <item name="colorPrimaryDark">@color/colorPri
        <item name="colorAccent">@color/colorAccent</
    </style>

</resources>
```

6. Now under **res**, create a folder named **values-v21**. Inside values-v21, create another **styles.xml** with the below styles. These styles are specific to **Android Lollipop** only.

```
styles.xml
<resources>

    <style name="MyMaterialTheme" parent="MyMaterialT
        <item name="android:windowContentTransitions"
        <item name="android:windowAllowEnterTransitio
        <item name="android:windowAllowReturnTransiti
        <item name="android:windowSharedElementEnterT
        <item name="android:windowSharedElementExitTr
    </style>

</resources>
```

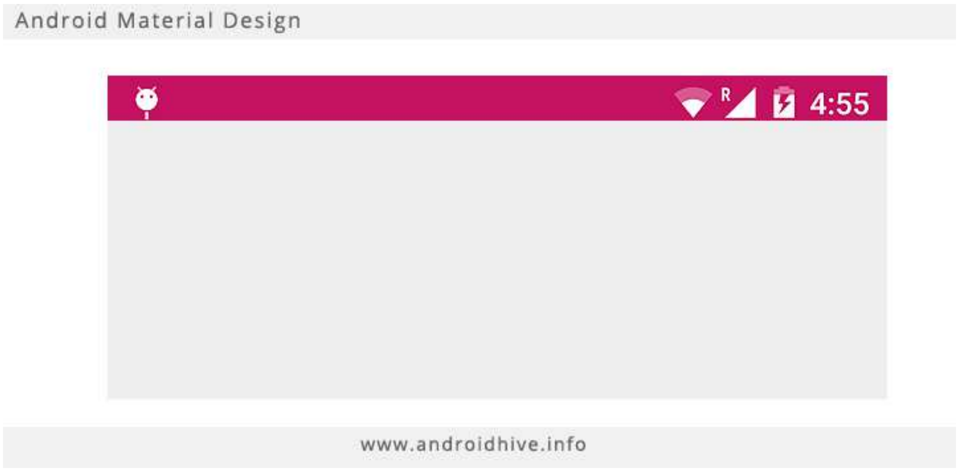
7. Now we have the basic Material Design styles ready. In order to apply the theme, open **AndroidManifest.xml** and modify the `android:theme` attribute of **<application>** tag.

```
android:theme="@style/MyMaterialTheme"
```

So after applying the theme, your **AndroidManifest.xml** should look like below.

```
AndroidManifest.xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/a
    package="info.androidhive.materialdesign" >

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/MyMaterialTheme" >
        <activity
            android:name=".activity.MainActivity"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.

                <category android:name="android.inten
            </intent-filter>
        </activity>
    </application>

</manifest>
```

Now if you run the app, you can see the notification bar color changed to the color that we have mentioned in our styles.



3.1 Adding the Toolbar (Action Bar)

Adding the toolbar is very easy. All you have to do is, create a separate layout for the toolbar and include it in other layout wherever you want the toolbar to be displayed.

8. Create an xml file named **toolbar.xml** under **res** ⇒ **layout** and add `android.support.v7.widget.Toolbar` element. This create the toolbar with specific height and theming.

toolbar.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.Toolbar xmlns:android="htt
    xmlns:local="http://schemas.android.com/apk/res-a
    android:id="@+id/toolbar"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:minHeight="?attr/actionBarSize"
    android:background="?attr/colorPrimary"
    local:theme="@style/ThemeOverlay.AppCompat.Dark.A
    local:popupTheme="@style/ThemeOverlay.AppCompat.L
```

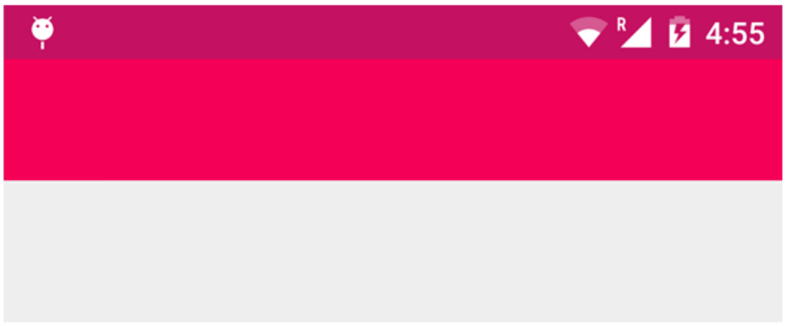
9. Open the layout file of your main activity (**activity_main.xml**) and add the **toolbar** using **<include/>** tag.

activity_main.xml

```
<RelativeLayout xmlns:android="http://schemas.android
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <LinearLayout
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:orientation="vertical">

        <include
            android:id="@+id/toolbar"
            layout="@layout/toolbar" />
    </LinearLayout>


</RelativeLayout>
```

Run the app and see if the toolbar displayed on the screen or not.

Android Material Design - Toolbar



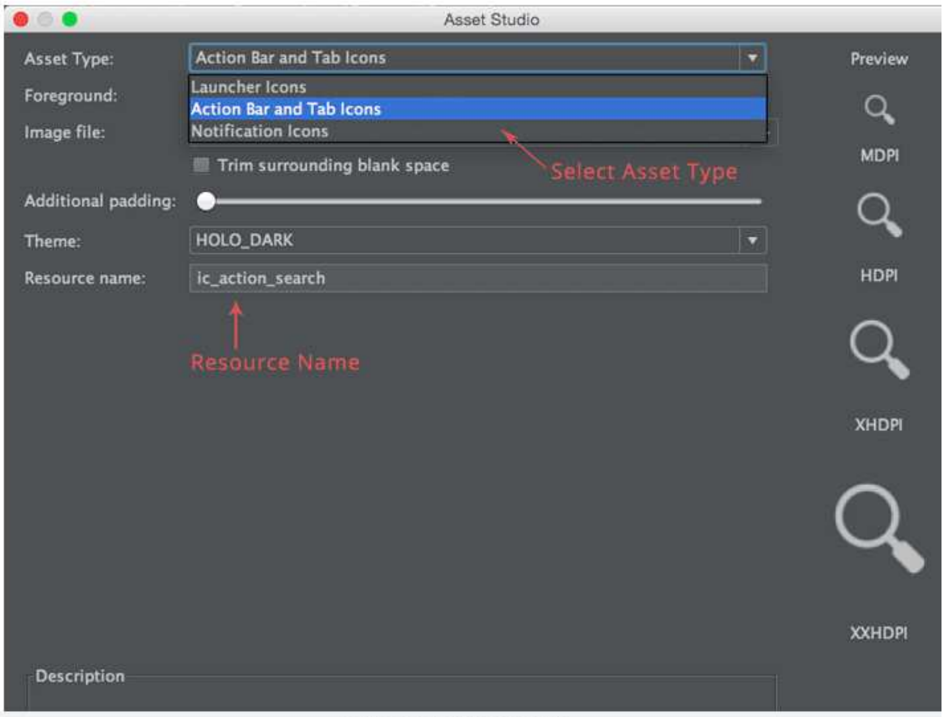
www.androidhive.info

Now let's try to add a toolbar title and enable the action items.

10. Download this search icon and import it into Android Studio as a Image Asset.

11. To import the Image Asset in Android Studio, right click on **res** ⇒ **New** ⇒ **Image Asset**. It will show you a popup window to import the resource. Browse the search icon that you have downloaded in the above step, select **Action Bar and Tab Icons** for Asset Type and give the resource name as **ic_search_action** and proceed.

Android Studio Importing Image Asset


Asset Studio

Asset Type: Action Bar and Tab Icons

Foreground:

Image file:

Trim surrounding blank space

Additional padding:

Theme: HOLO_DARK

Resource name: ic_action_search

Description

www.androidhive.info

12. Once the icon is imported, open **menu_main.xml** located under **res ⇒ menu** and add the search menu item as mentioned below.

menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/r
    xmlns:app="http://schemas.android.com/apk/res-aut
    xmlns:tools="http://schemas.android.com/tools"
    tools:context=".MainActivity">

    <item
        android:id="@+id/action_search"
        android:title="@string/action_search"
        android:orderInCategory="100"
        android:icon="@drawable/ic_action_search"
        app:showAsAction="ifRoom" />

    <item
        android:id="@+id/action_settings"
        android:title="@string/action_settings"
        android:orderInCategory="100"
        app:showAsAction="never" />
</menu>
```

13. Now open your **MainActivity.java** and do the below changes.

> Extend the activity from `AppCompatActivity`

> Enable the toolbar by calling **setSupportActionBar()** by passing the toolbar object.

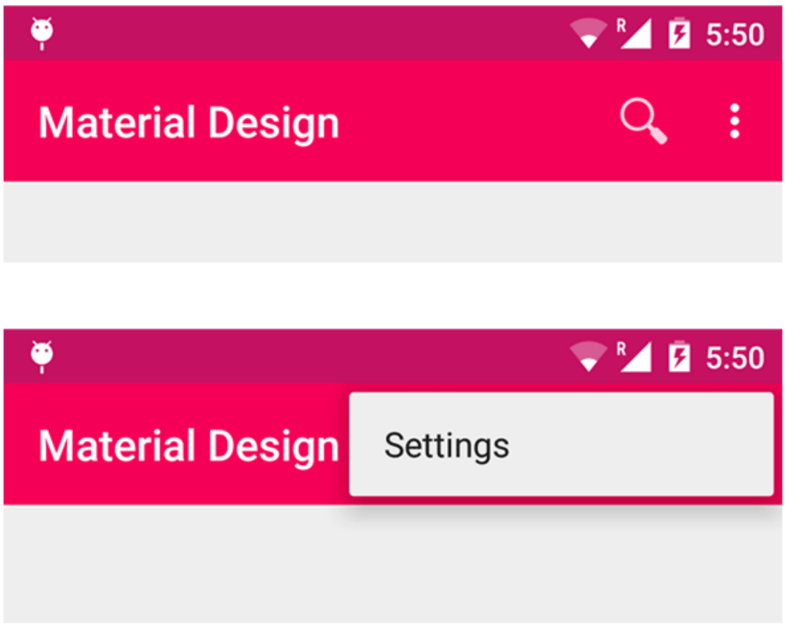
> Override **onCreateOptionsMenu()** and **onOptionsItemSelected()** methods to enable toolbar action items.

MainActivity.java

```
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;

public class MainActivity extends AppCompatActivity {

    private Toolbar mToolbar;

    @Override
    protected void onCreate(Bundle savedInstanceState
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        mToolbar = (Toolbar) findViewById(R.id.toolba

        setSupportActionBar(mToolbar);
        getSupportActionBar().setDisplayShowHomeEnabl
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the a
        getMenuInflater().inflate(R.menu.menu_main, m
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem ite
        // Handle action bar item clicks here. The ac
        // automatically handle clicks on the Home/Up
        // as you specify a parent activity in Androi
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }
}
```

After doing the above changes, if you run the app, you should see the search icon and action overflow icon.

Android Material Design Toolbar Action Items



www.androidhive.info

3.2 Adding Navigation Drawer

Adding navigation drawer is same as that we do before lollipop, but instead if using ListView for menu items, we use RecyclerView in material design. So let's see how to implement the navigation drawer with RecyclerView.

14. In your project's java folder, create three packages named **activity**, **adapter**, **model** and move your **MainActivity.java** to activity package. This will keep your project organized.

15. Open `build.gradle` located under your **app** module and add below dependencies. After adding the dependencies, goto **Build ⇒ Rebuild Project** to download required libraries.

build.gradle
```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:22.2.0'
    compile 'com.android.support:recyclerview-v7:22.2
}
```

16. Under **model** package, create a class named **NavDrawerItem.java** with the below code. This model class is POJO class that defines each row in navigation drawer menu.

NavDrawerItem.java
```
package info.androidhive.materialdesign.model;

/**
 * Created by Ravi on 29/07/15.
 */
public class NavDrawerItem {
    private boolean showNotify;
    private String title;


    public NavDrawerItem() {

    }

    public NavDrawerItem(boolean showNotify, String t
        this.showNotify = showNotify;
        this.title = title;
    }

    public boolean isShowNotify() {
        return showNotify;
    }

    public void setShowNotify(boolean showNotify) {
        this.showNotify = showNotify;
    }

    public String getTitle() {
        return title;
    }

    public void setTitle(String title) {
        this.title = title;
    }
}
```

17. Under **res** ⇒ **layout**, create an xml layout named **nav_drawer_row.xml** and add the below code. The layout renders each row in navigation drawer menu. If you want to customize the navigation drawer menu item, you have to do the changes in this file. For now it has only one TextView.

nav_drawer_row.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:clickable="true">

    <TextView
        android:id="@+id/title"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingLeft="30dp"
        android:paddingTop="10dp"
        android:paddingBottom="10dp"
        android:textSize="15dp"
        android:textStyle="bold" />

</RelativeLayout>
```

18. Download this profile icon and paste it in your drawable folder. This step is optional, but this icon used in the navigation drawer header part.

19. Create another xml layout named **fragment_navigation_drawer.xml** and add the below code. This layout renders the complete navigation drawer view. This layout contains a header section to display the user information and a RecyclerView to display the list view.

```
fragment_navigation_drawer.xml
<RelativeLayout xmlns:android="http://schemas.android
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@android:color/white">


    <RelativeLayout
        android:id="@+id/nav_header_container"
        android:layout_width="match_parent"
        android:layout_height="140dp"
        android:layout_alignParentTop="true"
        android:background="@color/colorPrimary">

        <ImageView
            android:layout_width="70dp"
            android:layout_height="70dp"
            android:src="@drawable/ic_profile"
            android:scaleType="fitCenter"
            android:layout_centerInParent="true" />

    </RelativeLayout>


    <android.support.v7.widget.RecyclerView
        android:id="@+id/drawerList"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/nav_header_containe
        android:layout_marginTop="15dp" />


</RelativeLayout>
```

20. As the RecyclerView is customized, we need an adapter class to render the custom xml layout. So under adapter package, create a class named **NavigationDrawerAdapter.java** and paste the below code. This adapter class inflates **nav_drawer_row.xml** and renders the **RecycleView** drawer menu.

```
import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import java.util.Collections;
import java.util.List;

/**
 * Created by Ravi Tamada on 12-03-2015.
 */
public class NavigationDrawerAdapter extends Recycler
    List<NavDrawerItem> data = Collections.emptyList(
    private LayoutInflater inflater;
    private Context context;

    public NavigationDrawerAdapter(Context context, L
        this.context = context;
        inflater = LayoutInflater.from(context);
        this.data = data;
    }

    public void delete(int position) {
        data.remove(position);
        notifyItemRemoved(position);
    }

    @Override
    public MyViewHolder onCreateViewHolder(ViewGroup
        View view = inflater.inflate(R.layout.nav_dra
        MyViewHolder holder = new MyViewHolder(view);
        return holder;
    }

    @Override
    public void onBindViewHolder(MyViewHolder holder,
        NavDrawerItem current = data.get(position);
        holder.title.setText(current.getTitle());
    }

    @Override
    public int getItemCount() {
        return data.size();
    }

    class MyViewHolder extends RecyclerView.ViewHolde
        TextView title;

        public MyViewHolder(View itemView) {
            super(itemView);
            title = (TextView) itemView.findViewById(
        }
    }
}
```

21. Under **activity** package, create a fragment named **FragmentDrawer.java**. In Android Studio, to create a new fragment, **Right click on activity** ⇒ **New** ⇒ **Fragment** ⇒ **Fragment (Blank)** and give your fragment class name.

```
FragmentDrawer.java
/**
 * Created by Ravi on 29/07/15.
 */

import android.content.Context;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.GestureDetector;
import android.view.LayoutInflater;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup;

import java.util.ArrayList;
import java.util.List;

import info.androidhive.materialdesign.R;
import info.androidhive.materialdesign.adapter.Naviga
import info.androidhive.materialdesign.model.NavDrawe

public class FragmentDrawer extends Fragment {

    private static String TAG = FragmentDrawer.class.
    private RecyclerView recyclerView;
    private ActionBarDrawerToggle mDrawerToggle;
    private DrawerLayout mDrawerLayout;
    private NavigationDrawerAdapter adapter;
    private View containerView;
    private static String[] titles = null;
    private FragmentDrawerListener drawerListener;

    public FragmentDrawer() {

    }

    public void setDrawerListener(FragmentDrawerListe
        this.drawerListener = listener;
    }

    public static List<NavDrawerItem> getData() {
        List<NavDrawerItem> data = new ArrayList<>();


        // preparing navigation drawer items
        for (int i = 0; i < titles.length; i++) {
            NavDrawerItem navItem = new NavDrawerItem
            navItem.setTitle(titles[i]);
            data.add(navItem);
        }
        return data;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // drawer labels
        titles = getActivity().getResources().getStri
    }

    @Override
    public View onCreateView(LayoutInflater inflater,
                             Bundle savedInstanceStat
        // Inflating view layout
        View layout = inflater.inflate(R.layout.fragm
        recyclerView = (RecyclerView) layout.findView

        adapter = new NavigationDrawerAdapter(getActi
        recyclerView.setAdapter(adapter);
        recyclerView.setLayoutManager(new LinearLayou
        recyclerView.addOnItemTouchListener(new Recyc
            @Override
            public void onClick(View view, int positi
                drawerListener.onDrawerItemSelected(v
                mDrawerLayout.closeDrawer(containerVi
            }

            @Override
            public void onLongClick(View view, int po

            }
        }));

        return layout;
    }


    public void setUp(int fragmentId, DrawerLayout dr
        containerView = getActivity().findViewById(fr
        mDrawerLayout = drawerLayout;
        mDrawerToggle = new ActionBarDrawerToggle(get
            @Override
            public void onDrawerOpened(View drawerVie
                super.onDrawerOpened(drawerView);
                getActivity().invalidateOptionsMenu()
            }

            @Override
            public void onDrawerClosed(View drawerVie
                super.onDrawerClosed(drawerView);
                getActivity().invalidateOptionsMenu()
            }

            @Override
            public void onDrawerSlide(View drawerView
                super.onDrawerSlide(drawerView, slide
                toolbar.setAlpha(1 - slideOffset / 2)
            }
        };

        mDrawerLayout.setDrawerListener(mDrawerToggle
        mDrawerLayout.post(new Runnable() {
            @Override
            public void run() {
                mDrawerToggle.syncState();
            }
        });

    }

    public static interface ClickListener {
        public void onClick(View view, int position);

        public void onLongClick(View view, int positi
    }

    static class RecyclerTouchListener implements Rec

        private GestureDetector gestureDetector;
        private ClickListener clickListener;

        public RecyclerTouchListener(Context context,
            this.clickListener = clickListener;
            gestureDetector = new GestureDetector(con
                @Override
                public boolean onSingleTapUp(MotionEv
                    return true;
                }
                @Override
                public void onLongPress(MotionEvent e
                    View child = recyclerView.findChi
                    if (child != null && clickListene
                        clickListener.onLongClick(chi
                    }
                }
            });
        }

        @Override
        public boolean onInterceptTouchEvent(Recycler

            View child = rv.findChildViewUnder(e.getX
            if (child != null && clickListener != nul
                clickListener.onClick(child, rv.getCh
            }
            return false;
        }

        @Override
        public void onTouchEvent(RecyclerView rv, Mot
        }

        @Override
        public void onRequestDisallowInterceptTouchEv

        }


    }

    public interface FragmentDrawerListener {
        public void onDrawerItemSelected(View view, i
    }
}
```

22. Finally open main activity layout (**activity_main.xml**) and modify the layout as below. In this layout we are adding `android.support.v4.widget.DrawerLayout` to display the navigation drawer menu.

Also you have to give the correct path of your FragmentDrawer in <fragment> element.

```
actiivty_main.xml
<android.support.v4.widget.DrawerLayout xmlns:android
    xmlns:app="http://schemas.android.com/apk/res-aut
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/drawer_layout"
    android:layout_width="match_parent"
    android:layout_height="match_parent">


    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:orientation="vertical">

        <LinearLayout
            android:id="@+id/container_toolbar"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:orientation="vertical">

            <include
                android:id="@+id/toolbar"
                layout="@layout/toolbar" />
        </LinearLayout>

        <FrameLayout
            android:id="@+id/container_body"
            android:layout_width="fill_parent"
            android:layout_height="0dp"
            android:layout_weight="1" />


    </LinearLayout>


    <fragment
        android:id="@+id/fragment_navigation_drawer"
        android:name="info.androidhive.materialdesign
        android:layout_width="@dimen/nav_drawer_width
        android:layout_height="match_parent"
        android:layout_gravity="start"
        app:layout="@layout/fragment_navigation_drawe
        tools:layout="@layout/fragment_navigation_dra

</android.support.v4.widget.DrawerLayout>
```

Now we have all the layout files and java classes ready in place. Let's do the necessary changes in MainActivity to make the navigation drawer functioning.

23. Open your **MainActivity.java** and do the below changes.

> Implement the activity from **FragmentDrawer.FragmentDrawerListener** and add the **onDrawerItemSelected()** override method.

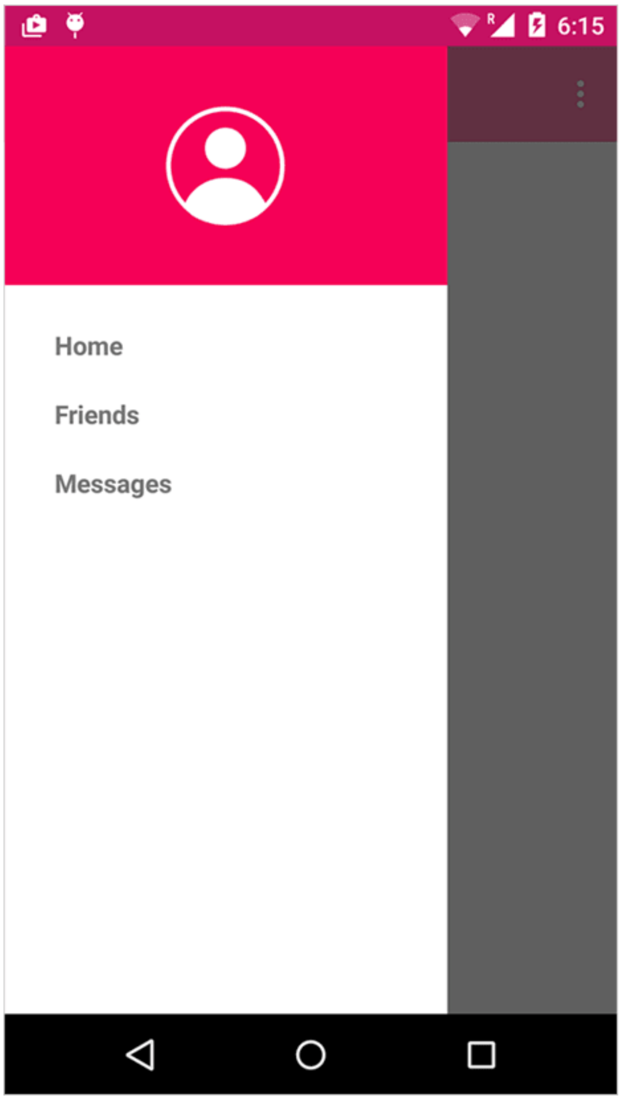
> Create an instance of **FragmentDrawer** and set the drawer selected listeners.

MainActivity.java

```
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

public class MainActivity extends AppCompatActivity i

    private Toolbar mToolbar;
    private FragmentDrawer drawerFragment;

    @Override
    protected void onCreate(Bundle savedInstanceState
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mToolbar = (Toolbar) findViewById(R.id.toolba

        setSupportActionBar(mToolbar);
        getSupportActionBar().setDisplayShowHomeEnabl

        drawerFragment = (FragmentDrawer)
                getSupportFragmentManager().findFragm
        drawerFragment.setUp(R.id.fragment_navigation
        drawerFragment.setDrawerListener(this);
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the a
        getMenuInflater().inflate(R.menu.menu_main, m
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem ite
        // Handle action bar item clicks here. The ac
        // automatically handle clicks on the Home/Up
        // as you specify a parent activity in Androi
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onDrawerItemSelected(View view, int p
        }
}
```

Now if you run the app, you can see the navigation drawer with a header and few list items in it.

Android Material Design Navigation Drawer



www.androidhive.info

3.3 Implementing Navigation Drawer Item Selection

Although navigation drawer is functioning, you can see the selection of drawer list items not working. This is because we are yet to implement the click listener on RecyclerView items.

As we have three menu items in navigation drawer (**Home**, **Friends** & **Messages**), we need to create three separate fragment classes for each menu item.

24. Under res layout, create an xml layout named **fragment_home.xml** and add below code.

fragment_home.xml

```
<RelativeLayout xmlns:android="http://schemas.android
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context="info.androidhive.materialdesign.ac


    <TextView
        android:id="@+id/label"
        android:layout_alignParentTop="true"
        android:layout_marginTop="100dp"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:gravity="center_horizontal"
        android:textSize="45dp"
        android:text="HOME"
        android:textStyle="bold"/>

    <TextView
        android:layout_below="@id/label"
        android:layout_centerInParent="true"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:textSize="12dp"
        android:layout_marginTop="10dp"
        android:gravity="center_horizontal"
        android:text="Edit fragment_home.xml to chang

</RelativeLayout>
```

25. Under **activity** package, create a fragment class named **HomeFragment.java** and add below code.

HomeFragment.java

```
import android.app.Activity;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
```

```
public class HomeFragment extends Fragment {

    public HomeFragment() {
        // Required empty public constructor
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

    }

    @Override
    public View onCreateView(LayoutInflater inflater,
                             Bundle savedInstanceStat
        View rootView = inflater.inflate(R.layout.fra


        // Inflate the layout for this fragment
        return rootView;
    }

    @Override
    public void onAttach(Activity activity) {
        super.onAttach(activity);
    }

    @Override
    public void onDetach() {
        super.onDetach();
    }
}
```

26. Create two more fragment classes named **FriendsFragment.java**, **MessagesFragment.java** and respected layout files named **fragment_friends.xml** and **fragment_messages.xml** and add the code from above two steps.

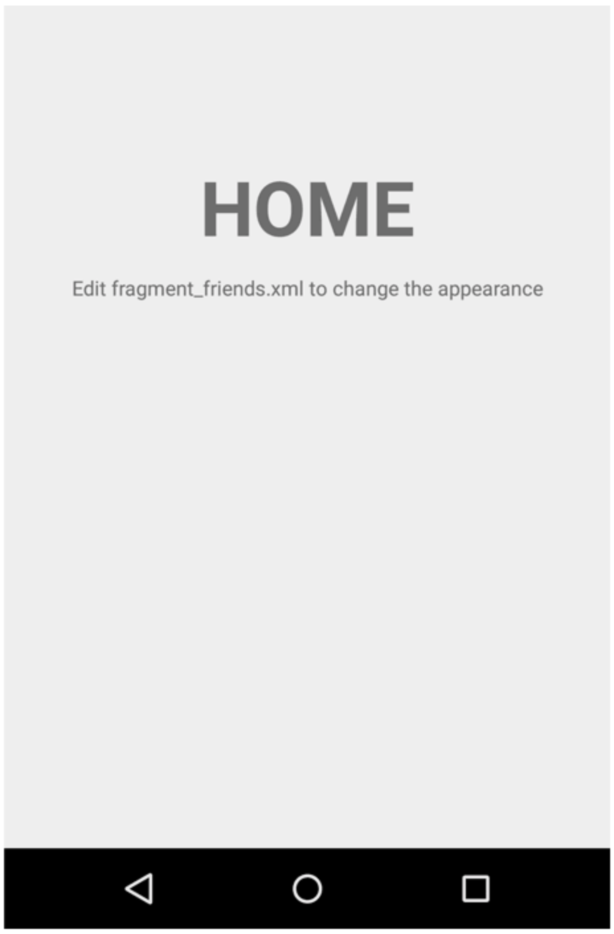
27. Now open **MainActivity.java** and do the below changes. In the below code

> displayView() method displays the fragment view respected the navigation menu item selection. This method should be called in **onDrawerItemSelected()** to render the respected view when a navigation menu item is selected.

```
MainActivity.java
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentTransaction;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarActivity;
import android.support.v7.widget.Toolbar;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.Toast;

public class MainActivity extends ActionBarActivity i

    private static String TAG = MainActivity.class.ge

    private Toolbar mToolbar;
    private FragmentDrawer drawerFragment;

    @Override
    protected void onCreate(Bundle savedInstanceState
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        mToolbar = (Toolbar) findViewById(R.id.toolba

        setSupportActionBar(mToolbar);
        getSupportActionBar().setDisplayShowHomeEnabl

        drawerFragment = (FragmentDrawer)
                getSupportFragmentManager().findFragm
        drawerFragment.setUp(R.id.fragment_navigation
        drawerFragment.setDrawerListener(this);

        // display the first navigation drawer view o
        displayView(0);
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the a
        getMenuInflater().inflate(R.menu.menu_main, m
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem ite
        // Handle action bar item clicks here. The ac
        // automatically handle clicks on the Home/Up
        // as you specify a parent activity in Androi
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        if(id == R.id.action_search){
            Toast.makeText(getApplicationContext(), "
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

    @Override
    public void onDrawerItemSelected(View view, int p
            displayView(position);
    }

    private void displayView(int position) {
        Fragment fragment = null;
        String title = getString(R.string.app_name);
        switch (position) {
            case 0:
                fragment = new HomeFragment();
                title = getString(R.string.title_home
                break;
            case 1:
                fragment = new FriendsFragment();
                title = getString(R.string.title_frie
                break;
            case 2:
                fragment = new MessagesFragment();
                title = getString(R.string.title_mess
                break;
            default:
                break;
        }

        if (fragment != null) {
            FragmentManager fragmentManager = getSupp
            FragmentTransaction fragmentTransaction =
            fragmentTransaction.replace(R.id.containe
            fragmentTransaction.commit();

            // set the toolbar title
            getSupportActionBar().setTitle(title);
        }
    }
}
```

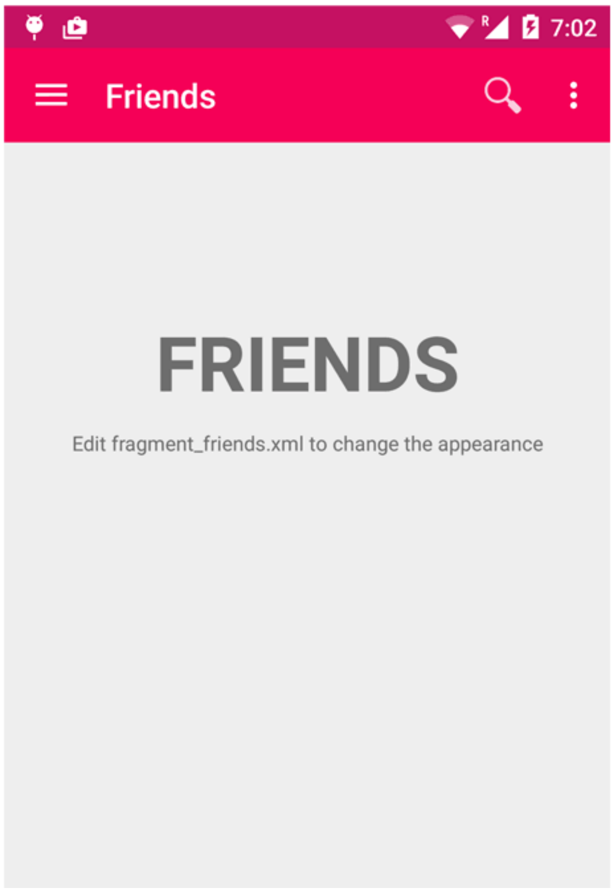
Now if you run the app, you can see the selection of navigation drawer menu is working and respected view displayed below the toolbar.

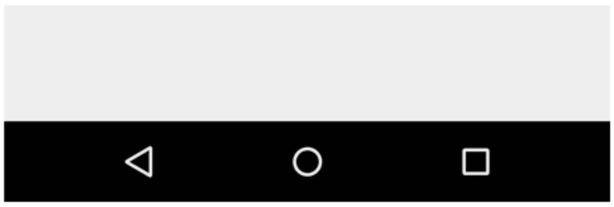
Android Material Design Navigation Drawer



www.androidhive.info

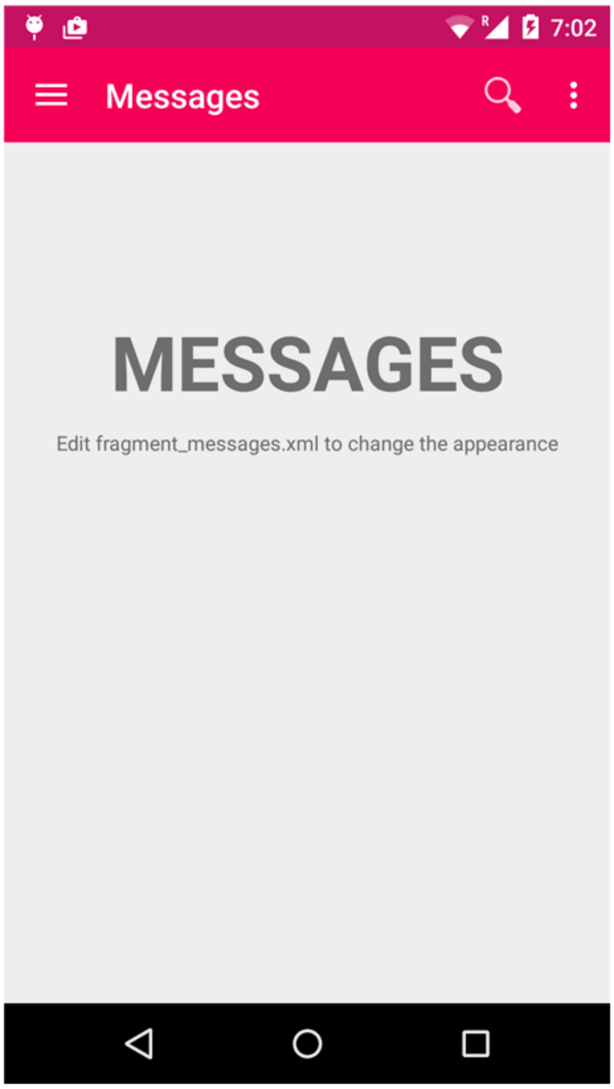
Android Material Design Navigation Drawer





www.androidhive.info

Android Material Design Navigation Drawer



www.androidhive.info

What's Next?

Below are few more material components you can add to your app. These were implemented using recent Android Design Support Library.

1. Material Design Tab Layout

If you want to add tabs to your app, Android Material Design Tabs covers different aspects of Tab Layout.

2. Floating Labels for EditText

Learn how floating labels works on EditText with a simple form validation example.

3. Floating Action Button (FAB)

Add the Floating Action Button to your which displays in circular shape floating on the top of the UI.

4. Snackbar

Add the Snackbar to your app to give immediate feedback about any operation that user performed.

Change Log

Updated On	29th July 2015 (Latest support library)

Email SUBSCRIBE

ABOUT THE AUTHOR

Ravi Tamada

Ravi is hardcore Android programmer and Android programming has been his passion since he compiled his first hello-world program. Solving real problems of Android developers through tutorials has always been interesting part for him.

RELATED POSTS

Retrofit Example in Andro
A type-safe HTTP client for Android and J

Android Working with Retrofit HTTP Library

by Ravi Tamada

Android Building Free Wallpapers App – Part 1

by Ravi Tamada

Android Building Free Wallpapers App – Part 2

by Ravi Tamada

Android Layouts: Linear Layout, Relative Layout and Table Layout

by Ravi Tamada

Comments Community 1 Login

Recommend 47 Share Sort by Newest

Join the discussion…

Bruno Logerfo • 4 days ago
Thanks, nice tutorial. I think i followed it correctly but my navigation drawer has no itens. any ideas?
• Reply • Share ›

Sam • 5 days ago
Hi Ravi,

is there a way to handle the back button?
• Reply • Share ›

maxx • 16 days ago
How to add the menu content dynamically? Bcz I am getting the content from the Rest API, So it will change dynamically, So can you please guide this.
• Reply • Share ›

Meshileya Israel • 21 days ago
Thanks for this amazing tutorials.....really appreciates
• Reply • Share ›

Ivan • 22 days ago
Beautiful tutorial. Thanks!
• Reply • Share ›

Ravi Tamada Mod Ivan • 22 days ago
You are welcome :)
• Reply • Share ›

Maher Nabeel • 22 days ago
by providing to us such well explained tutorials, you're truly a successful person. Thanks man.
• Reply • Share ›

Ravi Tamada Mod Maher Nabeel • 22 days ago
Thanks for the appreciation Maher.
• Reply • Share ›

Muhammad Sufiyan • 23 days ago
really great post, helped me a lot.. and others posts too are great :)