Android Material Design working with Tabs

by **Ravi Tamada** / **590** Comments

Android Design Support Library made our day easier by providing backward compatibility to number of material design components all the way back to Android 2.1. In Design support Library the components like navigation drawer, floating action button, snackbar, tabs, floating labels and animation frameworks were introduced. In this article we are going to learn how to implement material tabs in your apps.

Before going further, I suggest have a look at this tabs docs that defines do's and don'ts while implementing tabs.

DOWNLOAD CODE

VIDEO DEMO

1. Making the App Material

We'll start this by creating a new project and applying the material theme. If you are not aware of android material design, my previous article Android Getting Started with Material Design gives you a good start.

1. In Android Studio, go to **File ⇒ New Project** and fill all the details required to create a new project. When it prompts to select a default activity, select **Blank Activity** and proceed.

2. Open **build.gradle** and add android design support library **com.android.support:design:23.0.1**

build.gradle

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:23.0.1'
    compile 'com.android.support:design:23.0.1'
}
```

3. Open **colors.xml** located under **res ⇒ values** and add the below color values.

colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#125688</color>
    <color name="colorPrimaryDark">#125688</color>
    <color name="textColorPrimary">#FFFFFF</color>
    <color name="windowBackground">#FFFFFF</color>
    <color name="navigationBarColor">#000000</color>
    <color name="colorAccent">#c8e8ff</color>
</resources>
```

4. Add the below dimensions to **dimens.xml** located under **res ⇒ values**.

dimens.xml

```
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
    <dimen name="tab_max_width">264dp</dimen>
    <dimen name="tab_padding_bottom">16dp</dimen>
    <dimen name="tab_label">14sp</dimen>
    <dimen name="custom_tab_layout_height">72dp</dimen>
</resources>
```

5. Open **styles.xml** located under **res ⇒ values** and add below styles. The styles defined in this styles.xml are common to all the android versions.

styles.xml

```
<resources>

    <style name="MyMaterialTheme" parent="MyMaterialTheme.Base">

    </style>

    <style name="MyMaterialTheme.Base" parent="Theme.AppCompat.Light.DarkActionBar">
        <item name="windowNoTitle">true</item>
        <item name="windowActionBar">false</item>
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
    </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 5.0**

styles.xml

```
<resources>

    <style name="MyMaterialTheme" parent="MyMaterialTheme.Base">
        <item name="android:windowContentTransitions">true</item>
        <item name="android:windowAllowEnterTransitionOverlap">true</item>
        <item name="android:windowAllowReturnTransitionOverlap">true</item>
        <item name="android:windowSharedElementEnterTransition">@android:transition/move</i
        <item name="android:windowSharedElementExitTransition">@android:transition/move</ite
    </style>

</resources>
```

7. Finally open **AndroidManifest.xml** and modify the theme to our customized theme by changing the `android:theme` attribute value.

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

AndroidManifest.xml

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

    <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.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Run the app and verify the material theme by observing the notification bar color. If you see the notification bar color changed, it means that the material design theme is applied successfully.

Now we have our app material ready. So let's start adding the tabs. But before that we'll create few fragment activities for testing purpose. All these fragment activities contains very simple UI with only one TextView.

8. Under your main package create a fragment named **OneFragment.java** and add the below code.

OneFragment.java

```
package info.androidhive.materialtabs.fragments;

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

import info.androidhive.materialtabs.R;


public class OneFragment extends Fragment{

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

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

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_one, container, false);
    }

}
```

9. Open **fragment_one.xml** located under **res ⇒ layout** and do the below changes.

fragment_one.xml

```
<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="info.androidhive.materialtabs.fragments.OneFragment">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/one"
        android:textSize="40dp"
        android:textStyle="bold"
        android:layout_centerInParent="true"/>

</RelativeLayout>
```

10. Likewise create few more fragment activities with same code we used for OneFragment.java. I have created **TwoFragment.java**, **ThreeFragment.java**, **FourFragemnt.java .....** upto **TenFragment.java**

2. Fixed Tabs

Fixed tabs should be used when you have limited number of tabs. These tabs are fixed in position. In android design support library lot of new elements like CoordinatorLayout, AppBarLayout, TabLayout and lot more were introduced. I won't cover all of these as it's not the agenda of this article.

11. Open the layout file of main activity (**activity_main.xml**) and add below layout code.

`app:tabMode` – Defines the mode of the tab layout. In our case the value should be "**fixed**"

activity_main.xml

```
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:theme="@style/ThemeOverlay.AppCompat.Dark.ActionBar">

        <android.support.v7.widget.Toolbar
            android:id="@+id/toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:layout_scrollFlags="scroll|enterAlways"
            app:popupTheme="@style/ThemeOverlay.AppCompat.Light" />

        <android.support.design.widget.TabLayout
            android:id="@+id/tabs"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            app:tabMode="fixed"
            app:tabGravity="fill"/>
    </android.support.design.widget.AppBarLayout>

    <android.support.v4.view.ViewPager
        android:id="@+id/viewpager"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"  />
</android.support.design.widget.CoordinatorLayout>
```

12. Open **MainActivity.java** and do the below changes.

`tabLayout.setupWithViewPager(viewPager);` – Assigns the ViewPager to TabLayout.

`setupViewPager()` – Defines the number of tabs by setting appropriate fragment and tab name.

`ViewPagerAdapter` – Custom adapter class provides fragments required for the view pager.

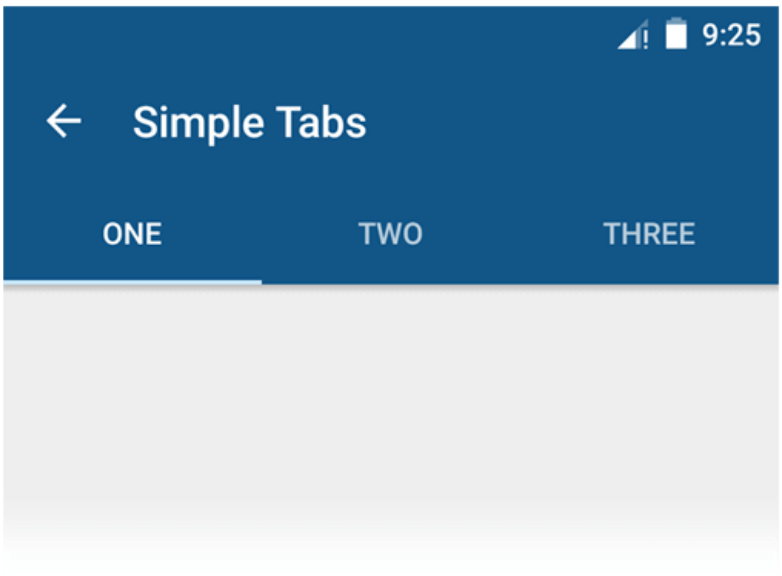
MainActivity.java

```
package info.androidhive.materialtabs.activity;

import android.os.Bundle;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;

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

import info.androidhive.materialtabs.R;
import info.androidhive.materialtabs.fragments.OneFragment;
import info.androidhive.materialtabs.fragments.ThreeFragment;
import info.androidhive.materialtabs.fragments.TwoFragment;

public class MainActivity extends AppCompatActivity {

    private Toolbar toolbar;
    private TabLayout tabLayout;
    private ViewPager viewPager;

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

```
        setContentView(R.layout.activity_main);

        toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);

        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        viewPager = (ViewPager) findViewById(R.id.viewpager);
        setupViewPager(viewPager);

        tabLayout = (TabLayout) findViewById(R.id.tabs);
        tabLayout.setupWithViewPager(viewPager);
    }

    private void setupViewPager(ViewPager viewPager) {
        ViewPagerAdapter adapter = new ViewPagerAdapter(getSupportFragmentManager());
        adapter.addFragment(new OneFragment(), "ONE");
        adapter.addFragment(new TwoFragment(), "TWO");
        adapter.addFragment(new ThreeFragment(), "THREE");
        viewPager.setAdapter(adapter);
    }

    class ViewPagerAdapter extends FragmentPagerAdapter {
        private final List<Fragment> mFragmentList = new ArrayList<>();
        private final List<String> mFragmentTitleList = new ArrayList<>();

        public ViewPagerAdapter(FragmentManager manager) {
            super(manager);
        }

        @Override
        public Fragment getItem(int position) {
            return mFragmentList.get(position);
        }

        @Override
        public int getCount() {
            return mFragmentList.size();
        }

        public void addFragment(Fragment fragment, String title) {
            mFragmentList.add(fragment);
            mFragmentTitleList.add(title);
        }

        @Override
        public CharSequence getPageTitle(int position) {
            return mFragmentTitleList.get(position);
        }
    }
}
```

Now run the app. You should able to see the tabs displayed with swipe functionality between the tabs.

Android Material Design Tab Layout

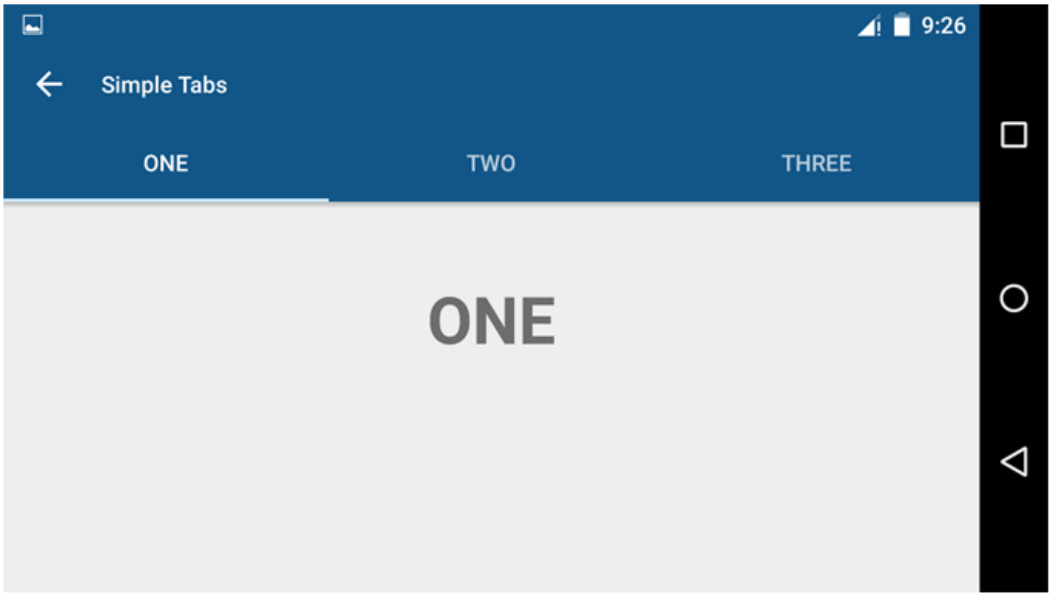


www.androidhive.info

2.1 Full Width Tabs

If you want the tabs to be occupied the fullwidth of the screen, you need to assign `app:tabGravity="fill"` to our TabLayout.

Android Tab Layout Landscape View

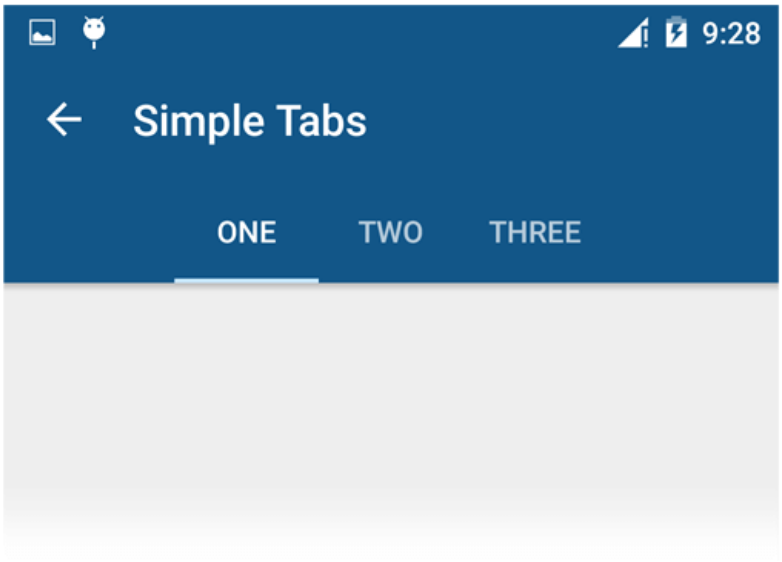


www.androidhive.info

2.2 Center Aligned Tabs

If you want to keep your tabs horizontally centered, assign `app:tabGravity="center"` to TabLayout.

Android Tab Layout - Gravity Center



www.androidhive.info

3. Scrollable Tabs

The scrollable tabs should be used when you have many number of tabs where there is insufficient space on the screen to fit all of them. To make the tabs scrollable, set `app:tabMode` to `TabLayout.MODE_SCROLLABLE`

13. Open **activity_main.xml** and change the `app:tabMode` to **scrollable**.

```
<android.support.design.widget.TabLayout
        android:id="@+id/tabs"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        app:tabMode="scrollable"/>
```

14. Edit **MainActivity.java** and add few fragments to ViewPager in **setupViewPager()** method. I have added total of 10 fragments to ViewPager. After the changes, your main activity should look like below.

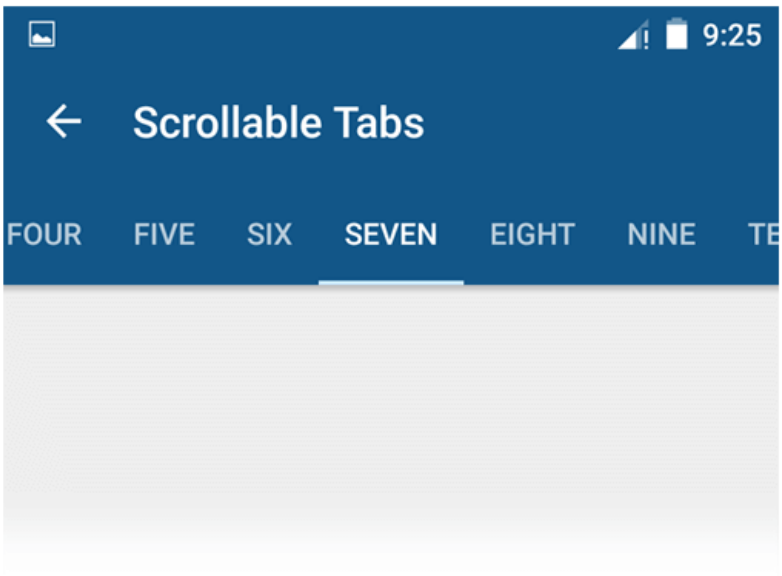
MainActivity.java

```
package info.androidhive.materialtabs.activity;

import android.os.Bundle;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
```

```
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;

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

public class MainActivity extends AppCompatActivity {

    private Toolbar toolbar;
    private TabLayout tabLayout;
    private ViewPager viewPager;

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

        toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        viewPager = (ViewPager) findViewById(R.id.viewpager);
        setupViewPager(viewPager);

        tabLayout = (TabLayout) findViewById(R.id.tabs);
        tabLayout.setupWithViewPager(viewPager);
    }

    private void setupViewPager(ViewPager viewPager) {
        ViewPagerAdapter adapter = new ViewPagerAdapter(getSupportFragmentManager());
        adapter.addFrag(new OneFragment(), "ONE");
        adapter.addFrag(new TwoFragment(), "TWO");
        adapter.addFrag(new ThreeFragment(), "THREE");
        adapter.addFrag(new FourFragment(), "FOUR");
        adapter.addFrag(new FiveFragment(), "FIVE");
        adapter.addFrag(new SixFragment(), "SIX");
        adapter.addFrag(new SevenFragment(), "SEVEN");
        adapter.addFrag(new EightFragment(), "EIGHT");
        adapter.addFrag(new NineFragment(), "NINE");
        adapter.addFrag(new TenFragment(), "TEN");
        viewPager.setAdapter(adapter);
    }

    class ViewPagerAdapter extends FragmentPagerAdapter {
        private final List<Fragment> mFragmentList = new ArrayList<>();
        private final List<String> mFragmentTitleList = new ArrayList<>();

        public ViewPagerAdapter(FragmentManager manager) {
            super(manager);
        }

        @Override
        public Fragment getItem(int position) {
            return mFragmentList.get(position);
        }

        @Override
        public int getCount() {
            return mFragmentList.size();
        }

        public void addFrag(Fragment fragment, String title) {
            mFragmentList.add(fragment);
            mFragmentTitleList.add(title);
        }

        @Override
        public CharSequence getPageTitle(int position) {
            return mFragmentTitleList.get(position);
        }
    }
}
```

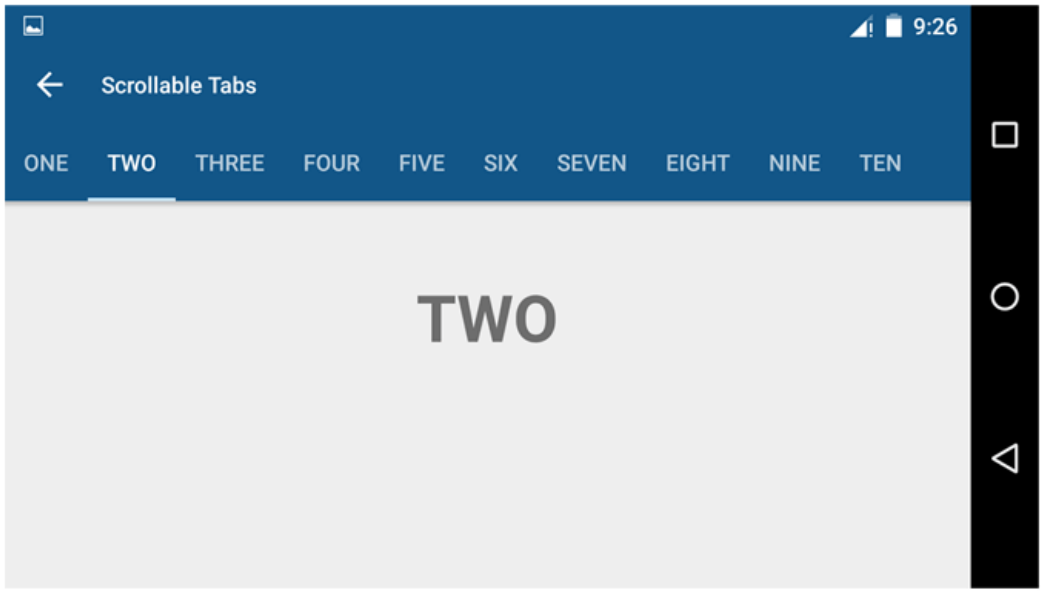
Now if you run the app, you can see the more number of tabs with scrollable functionality.

Android Scrollable Tab Layout



www.androidhive.info

Android Scrollable Tab Layout



www.androidhive.info

4. Tabs with Icon & Text

Sometimes you might wanted to add an icon to Tab. Earlier adding an icon to tab is tedious process. But with the design support library it is very easy. All you have to do is call `setIcon()` method by passing appropriate icon. The icon will be placed in front of tab label.

```
tabLayout.getTabAt(0).setIcon(tabIcons[0]);
tabLayout.getTabAt(1).setIcon(tabIcons[1]);
```

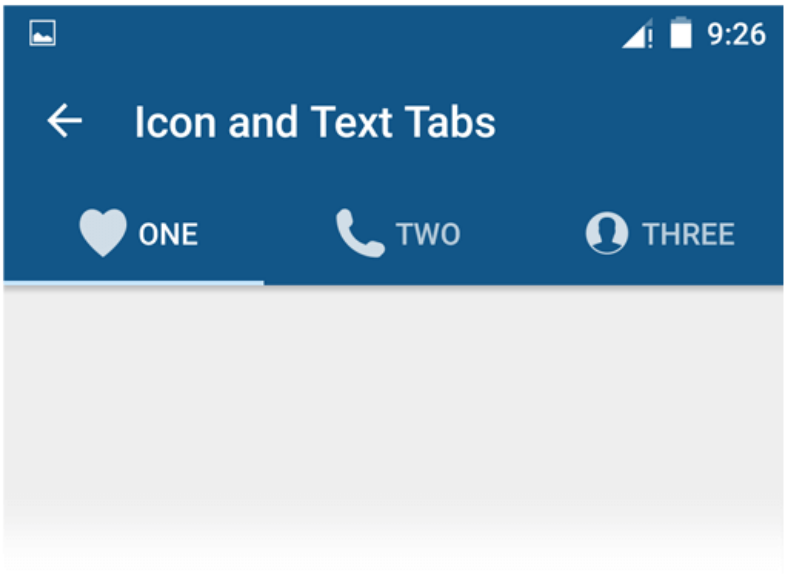
15. Open your **MainActivity.java** and modify the code as below. Here I have added a new method called **setupTabIcons()** in which I have set all the tab icons.

MainActivity.java

```
import android.os.Bundle;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;

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

public class MainActivity extends AppCompatActivity {

    private Toolbar toolbar;
    private TabLayout tabLayout;
    private ViewPager viewPager;
    private int[] tabIcons = {
            R.drawable.ic_tab_favourite,
            R.drawable.ic_tab_call,
            R.drawable.ic_tab_contacts
    };

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

        toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        viewPager = (ViewPager) findViewById(R.id.viewpager);
        setupViewPager(viewPager);

        tabLayout = (TabLayout) findViewById(R.id.tabs);
        tabLayout.setupWithViewPager(viewPager);
        setupTabIcons();
    }

    private void setupTabIcons() {
        tabLayout.getTabAt(0).setIcon(tabIcons[0]);
        tabLayout.getTabAt(1).setIcon(tabIcons[1]);
        tabLayout.getTabAt(2).setIcon(tabIcons[2]);
    }

    private void setupViewPager(ViewPager viewPager) {
        ViewPagerAdapter adapter = new ViewPagerAdapter(getSupportFragmentManager());
        adapter.addFrag(new OneFragment(), "ONE");
        adapter.addFrag(new TwoFragment(), "TWO");
        adapter.addFrag(new ThreeFragment(), "THREE");
        viewPager.setAdapter(adapter);
    }

    class ViewPagerAdapter extends FragmentPagerAdapter {
        private final List<Fragment> mFragmentList = new ArrayList<>();
        private final List<String> mFragmentTitleList = new ArrayList<>();

        public ViewPagerAdapter(FragmentManager manager) {
            super(manager);
        }

        @Override
        public Fragment getItem(int position) {
            return mFragmentList.get(position);
        }

        @Override
        public int getCount() {
            return mFragmentList.size();
        }

        public void addFrag(Fragment fragment, String title) {
            mFragmentList.add(fragment);
            mFragmentTitleList.add(title);
        }

        @Override
```

```
    public CharSequence getPageTitle(int position) {
        return mFragmentTitleList.get(position);
    }
  }
}
```

Android Tab Layout with Icon and Text



www.androidhive.info

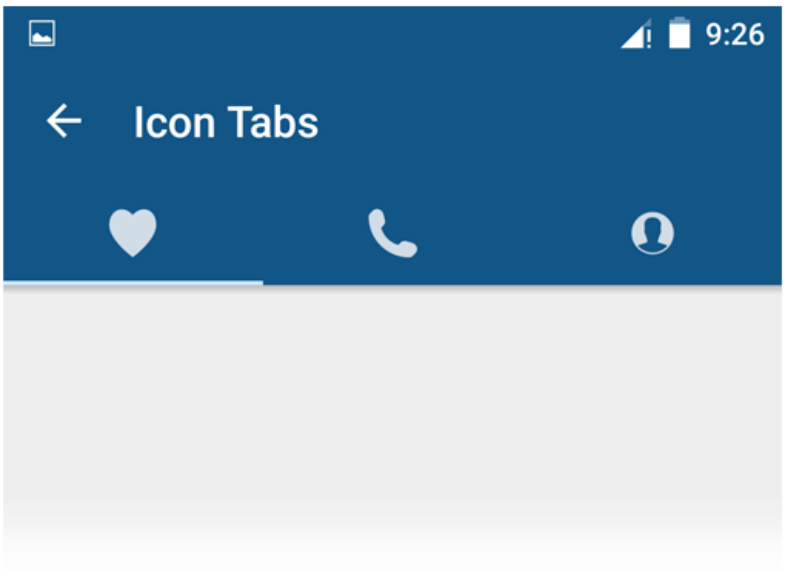
5. Tabs with only Icons

Setting only icon to tab is same as setting text and icon except the method `getPageTitle()` in ViewPagerAdapter class returns **null** instead of tab label.

16. Open **MainActivity.java** and modify the **getPageTitle()** method as below and run the project.

```
@Override
public CharSequence getPageTitle(int position) {

    // return null to display only the icon
    return null;
}
```

Android Tab Layout with only Icons



www.androidhive.info

6. Custom Tab View with Icon & Text

Setting a custom view to the tab is very useful when you are not able to achieve desired output by following the methods provided by tab layout. While setting a custom view to tab, make sure that you follow the specs suggested by android for tabs.

When we set the tab an icon and text, you can see the icon is horizontally aligned with tab text. But if you want to place the icon above the tab label, you have to use a custom view to achive it.

17. Under **res ⇒ values**, create an xml file named **fonts.xml** and add below string value. This xml file defines the font family for the tab label.

fonts.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="font_fontFamily_medium">sans-serif</string>
</resources>
```

18. Under **res ⇒ values-v21**, create another xml named **fonts.xml**.

fonts.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="font_fontFamily_medium">sans-serif-medium</string>
</resources>
```

19. Open **activity_main.xml** and set the custom height to TabLayout. Setting this height is important as placing icon above the tab label takes more space than normal.

```
<android.support.design.widget.TabLayout
        android:id="@+id/tabs"
        android:layout_width="match_parent"
        android:layout_height="@dimen/custom_tab_layout_height"
        app:tabMode="fixed"
        app:tabGravity="fill"/>
```

20. Create an xml layout named **custom_tab.xml** under **res ⇒ layout**. In this layout we have defined the custom view for the tab.

custom_tab.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/tab"
    android:textColor="@color/colorAccent"
    android:textSize="@dimen/tab_label"
    android:fontFamily="@string/font_fontFamily_medium"/>
```

21. Open **MainActivity.java** and modify the code as below. Here if you observe `setupTabIcons()` method, we have rendered **custom_tab.xml** layout in each tab using below lines of code.

```
TextView tabOne = (TextView) LayoutInflater.from(this).inflate(R.layout.custom_tab, null);
tabOne.setText("ONE");
tabOne.setCompoundDrawablesWithIntrinsicBounds(0, R.drawable.ic_tab_favourite, 0, 0);
tabLayout.getTabAt(0).setCustomView(tabOne);
```

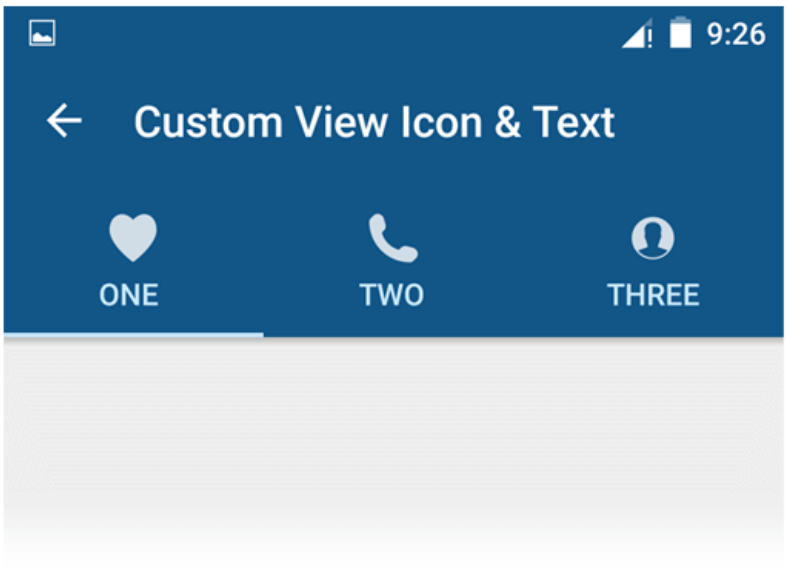
MainActivity.java

```
import android.os.Bundle;
import android.support.design.widget.TabLayout;
import android.support.v4.app.Fragment;
import android.support.v4.app.FragmentManager;
import android.support.v4.app.FragmentPagerAdapter;
import android.support.v4.view.ViewPager;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.LayoutInflater;
import android.widget.TextView;

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

import info.androidhive.materialtabs.R;
import info.androidhive.materialtabs.fragments.OneFragment;
import info.androidhive.materialtabs.fragments.ThreeFragment;
import info.androidhive.materialtabs.fragments.TwoFragment;

public class MainActivity extends AppCompatActivity {

    private Toolbar toolbar;
    private TabLayout tabLayout;
    private ViewPager viewPager;
    private int[] tabIcons = {
            R.drawable.ic_tab_favourite,
            R.drawable.ic_tab_call,
            R.drawable.ic_tab_contacts
    };

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

        toolbar = (Toolbar) findViewById(R.id.toolbar);
        setSupportActionBar(toolbar);
```

```
        getSupportActionBar().setDisplayHomeAsUpEnabled(true);

        viewPager = (ViewPager) findViewById(R.id.viewpager);
        setupViewPager(viewPager);

        tabLayout = (TabLayout) findViewById(R.id.tabs);
        tabLayout.setupWithViewPager(viewPager);
        setupTabIcons();
    }

    private void setupTabIcons() {

        TextView tabOne = (TextView) LayoutInflater.from(this).inflate(R.layout.custom_tab, null);
        tabOne.setText("ONE");
        tabOne.setCompoundDrawablesWithIntrinsicBounds(0, R.drawable.ic_tab_favourite, 0, 0);
        tabLayout.getTabAt(0).setCustomView(tabOne);

        TextView tabTwo = (TextView) LayoutInflater.from(this).inflate(R.layout.custom_tab, null);
        tabTwo.setText("TWO");
        tabTwo.setCompoundDrawablesWithIntrinsicBounds(0, R.drawable.ic_tab_call, 0, 0);
        tabLayout.getTabAt(1).setCustomView(tabTwo);

        TextView tabThree = (TextView) LayoutInflater.from(this).inflate(R.layout.custom_tab, null);
        tabThree.setText("THREE");
        tabThree.setCompoundDrawablesWithIntrinsicBounds(0, R.drawable.ic_tab_contacts, 0, 0);
        tabLayout.getTabAt(2).setCustomView(tabThree);
    }

    private void setupViewPager(ViewPager viewPager) {
        ViewPagerAdapter adapter = new ViewPagerAdapter(getSupportFragmentManager());
        adapter.addFrag(new OneFragment(), "ONE");
        adapter.addFrag(new TwoFragment(), "TWO");
        adapter.addFrag(new ThreeFragment(), "THREE");
        viewPager.setAdapter(adapter);
    }

    class ViewPagerAdapter extends FragmentPagerAdapter {
        private final List<Fragment> mFragmentList = new ArrayList<>();
        private final List<String> mFragmentTitleList = new ArrayList<>();

        public ViewPagerAdapter(FragmentManager manager) {
            super(manager);
        }

        @Override
        public Fragment getItem(int position) {
            return mFragmentList.get(position);
        }

        @Override
        public int getCount() {
            return mFragmentList.size();
        }

        public void addFrag(Fragment fragment, String title) {
            mFragmentList.add(fragment);
            mFragmentTitleList.add(title);
        }

        @Override
        public CharSequence getPageTitle(int position) {
            return mFragmentTitleList.get(position);
        }
    }
}
```

Now if you run the app, you can see the icon placed above the tab label.

Android Tab Layout with Custom View



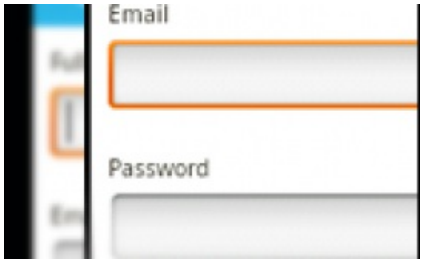
www.androidhive.info

I hope this article provided useful information about the tab layout using design support library. If you have any queries please do comment below.

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

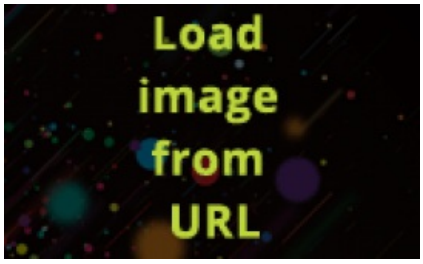


Android Login and Registration Screen Design

by **Ravi Tamada**

Android Material Design Floating Action Button

by **Ravi Tamada**



Android Loading Image from URL (HTTP)

by **Ravi Tamada**

Android Fullscreen Image Slider with Swipe and Pinch Zoom Gestures

by **Ravi Tamada**

SEARCH HERE

Search the site

WE'RE SOCIAL

AndroidHive
43K likes

Like Page

Be the first of your friends to like this

POPULAR ANDROID TUTORIALS

1. Android SQLite Database Tutorial - 1,655,192 views
2. How to connect Android with PHP, MySQL - 1,613,911 views
3. Android JSON Parsing Tutorial - 1,513,990 views
4. Android Push Notifications using Firebase Cloud Messaging FCM & PHP - 1,439,078 views
5. Android Sliding Menu using Navigation Drawer - 1,368,953 views
6. Android Login and Registration with PHP, MySQL and SQLite - 1,213,007 views
7. Android Custom ListView with Image and Text - 1,089,274 views
8. Android GPS, Location Manager Tutorial - 903,654 views
9. Android Tab Layout with Swipeable Views - 804,300 views
10. Android Expandable List View Tutorial - 732,079 views

ABOUT ANDROIDHIVE

Androidhive is independent online publication that covers android programming tutorials, app reviews and more. We are a team of some of the internet's most experienced android developers, user-experience designers and digital marketers.

Although we have different interests and capabilities, what keeps up together is: **Love for Android**.

We have our channels on various digital avenues such as Google+, Facebook, YouTube.

QUICK LINKS

- **Advertise with us**
- **Privacy Policy**
- **Terms of Service**
- **Sitemap**
- **Team Androidhive**

CONTACT US

For any business and advertisement related queries, contact us here:

Email: advertise@androidhive.info

Address:KPHB Phase 9,
Kukatpally, Hyderabad,
India