Step by Step process for: how to code a Snack Bar

Snackbars provide lightweight feedback about an operation. They show a brief message at the bottom of the screen on mobile and lower left on larger devices. Snackbars appear above all other elements on screen and only one can be displayed at a time.

They automatically disappear after a timeout or after user interaction elsewhere on the screen, particularly after interactions that summon a new surface or activity. Snackbars can be swiped off screen.

Snackbar is a good and simple example of Material design. It is a newer version of the Toast.

When Android 5.0 was released, new design guidelines were also released with it, called material design. Material design in a way revamped the experience of an android user. It introduced many new design patterns and guidelines. But here we'll discuss only one new UX concept introduced in material design called Snackbar. This new concept is inspired from the Toast widget of android. Android Snackbar is just like a Toast message except that it has an option for user feedback. You can call it as a toast message with an action button. Usually this type of widget is used in a situation where user needs an option to reverse his actions like undo -ing an action. In this Android Snackbar Example I would show how to make an Android Snackbar using the new Design Support Library.

To make an Android Snackbar many libraries are available on the internet. But for this Android Snackbar Example we will be using the official Android design support library which was released in the beginning of June 15. Although a notable amount of effort has been invested in by some of the independent developers to make a library for snackbar, kudos to them.

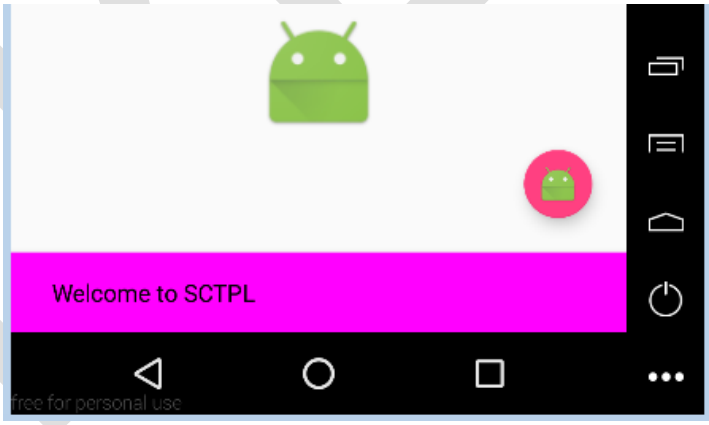
1. Simple Snackbar

Below is the syntax of a simple snackbar. The **make** function accepts three parameters. View, display message and duration of the message to be displayed.

Normally passing **CoordinatorLayout** as view param is the best option as it allows Snackbar to enable some features like swipe-to-dismiss and automatically moving of widgets like FloatingActionButton.

And the duration should be `LENGTH_SHORT`, `LENGTH_LONG` or `LENGTH_INDEFINITE`. When LENGTH_INDEFINITE is used, the snackbar will be displayed indefinite time and can be dismissed with swipe off.

```
Snackbar snackbar = Snackbar
        .make(
        coordinatorLayout,
        "Welcome to SCTPL",
        Snackbar.LENGTH_LONG
        );
snackbar.show();
```

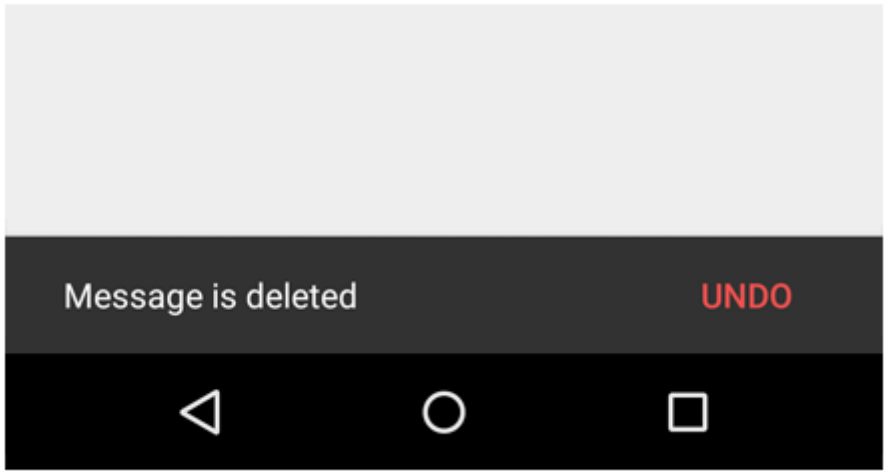


2. Snackbar with Action Callback

You can also mention a callback interaction method using `setAction()` method. This allows us to take certain action when user interacts with the snackbar.

```
Snackbar snackbar = Snackbar
        .make(coordinatorLayout,
              "Message is deleted",
              Snackbar.LENGTH_LONG)
        .setAction("UNDO",
                   new View.OnClickListener() {
                   @Override
                   public void onClick(View view) {
                   Snackbar snackbar1 = Snackbar.make(
                                        coordinatorLayout,
                                        "Message is restored!",
                                        Snackbar.LENGTH_SHORT);
                       snackbar1.show();
                   }
        });

snackbar.show();
```



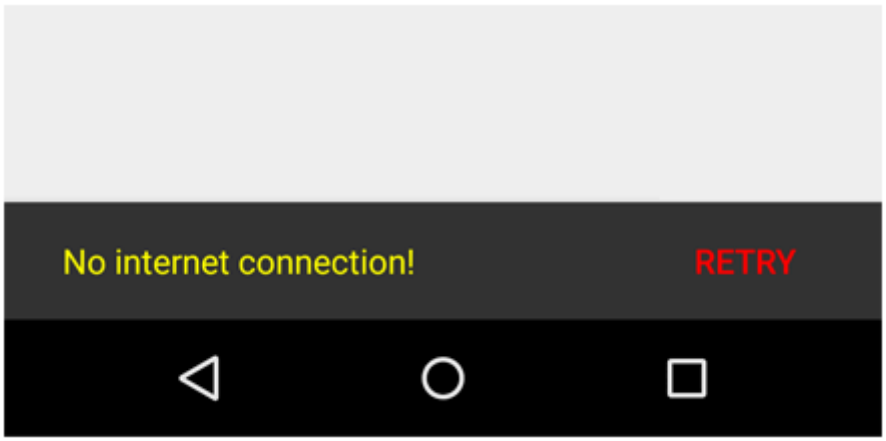
3. Customizing the Snackbar View

Snackbar comes with default **white** color text and **#323232** background color. You can override these colors as mentioned below.

```
Snackbar snackbar = Snackbar
        .make(coordinatorLayout,
              "No internet connection!",
              Snackbar.LENGTH_LONG)
        .setAction("RETRY", new View.OnClickListener() {
            @Override
            public void onClick(View view) {
            }
        });

// Changing message text color
snackbar.setActionTextColor(Color.RED);

// Changing action button text color
View sbView = snackbar.getView();
TextView textView = (TextView)
sbView.findViewById(android.support.design.R.id.snackbar_text);
textView.setTextColor(Color.YELLOW);
snackbar.show();
```



My Friends, let's start coding -

Android Snackbar Example using Design Support Library

Snackbar is a UI element used for user feedback. Generally used when instantaneous feedback is required from the user, after an action is performed. Snackbar appears in on the screen from bottom and can be dispersed by swiping, if a CoordinatorLayout is used as its parent (***will explain this part later in this document***).

To start off lets include the Android design support library in our project gradle file:

Go to Gradle Scripts > build.gradle file, check does it contain following lines of code:

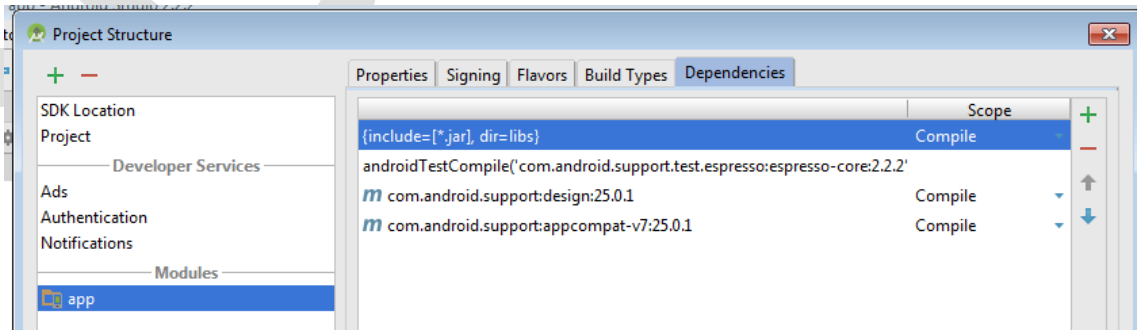
```
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {
        exclude group: 'com.android.support', module: 'support-annotations'
    })
    compile 'com.android.support:design:25.0.1'
    compile 'com.android.support:appcompat-v7:25.0.1'
}
```

Specifically these 2 lines of code. If you don't find them, either type them or **add them**#

#to add the above dependencies do this :
1> select app > press F4

2> would see Project Structure window

3> Click on +
(*see right side*)



4> Add 2 dependencies

- ✓ **com.android.support.design:25.x.x**
- ✓ **com.android.support : appcombat-v7:25.x.x**

// Refer the "SnackBarDemo" Project from the Workspace:

// code for the ***activity_main.xml*** file

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
   android:id="@+id/coordinatorLayout"
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">

   <RelativeLayout
      android:id="@+id/mainLayout"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:paddingBottom="@dimen/activity_vertical_margin"
      android:paddingLeft="@dimen/activity_horizontal_margin"
      android:paddingRight="@dimen/activity_horizontal_margin"
      android:paddingTop="@dimen/activity_vertical_margin">

      <CheckBox
         android:id="@+id/checkBox"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignParentTop="true"
         android:layout_centerHorizontal="true"
         android:layout_marginTop="40dp"
         android:text="Mark as done"/>

      <ImageView
         android:id="@+id/imageView"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_alignParentBottom="true"
         android:layout_centerHorizontal="true"
         android:layout_marginBottom="100dp"
         android:src="@drawable/someimage"/>

   </RelativeLayout>

   <android.support.design.widget.FloatingActionButton
      android:id="@+id/fab"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginBottom="20dp"
      android:layout_marginRight="20dp"
      android:src="@drawable/someimage"
      app:fabSize="mini"
      app:layout_anchor="@id/mainLayout"
      app:layout_anchorGravity="bottom|right|end"/>
</android.support.design.widget.CoordinatorLayout>
```

// Code for *MainActivity.java* file

```
package suvenconsultants.com.snackbardemo;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.support.design.widget.*;
import android.view.View;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;


public class MainActivity extends Activity {
   View.OnClickListener mOnClickListener;

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

      final CoordinatorLayout coordinatorLayout = (CoordinatorLayout)
            findViewById(R.id.coordinatorLayout);
      final CheckBox checkBox = (CheckBox) findViewById(R.id.checkBox);
      checkBox.setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {
         @Override
         public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {
            if (isChecked) {
               Snackbar snackbar = Snackbar
                     .make(coordinatorLayout,
                           "Welcome to SCTPL", Snackbar.LENGTH_LONG)
                     .setAction("Undo", mOnClickListener);
               snackbar.setActionTextColor(Color.RED);
               View snackbarView = snackbar.getView();
               snackbarView.setBackgroundColor(Color.MAGENTA);
               TextView textView = (TextView) snackbarView.findViewById(
                     android.support.design.R.id.snackbar_text);
               textView.setTextColor(Color.BLACK);
               snackbar.show();
            }
         }
      });

      mOnClickListener = new View.OnClickListener() {
         @Override
         public void onClick(View v) {
            checkBox.setChecked(false);
         }
      };
   }
}
```

- ➔ ***Nothing extra to be coded in Manifest file.***
- ➔ **Run the code. (View the output, carefully)**