## 用Toolbar作为Appbar或Actionbar

**Add a Toolbar to an Activity**

These steps describe how to set up a [Toolbar](https://developer.android.com/reference/android/support/v7/widget/Toolbar.html) as your activity's app bar:

1. Add the [v7 appcompat](https://developer.android.com/tools/support-library/features.html#v7-appcompat) support library to your project, as described in [Support Library Setup](https://developer.android.com/tools/support-library/setup.html).
2. Make sure the activity extends [AppCompatActivity](https://developer.android.com/reference/android/support/v7/app/AppCompatActivity.html):

JAVA

public class MyActivity extends AppCompatActivity {  
  // ...  
}

**Note:** Make this change for every activity in your app that uses a [**Toolbar**](https://developer.android.com/reference/android/support/v7/widget/Toolbar.html) as an app bar.

1. In the app manifest, set the [<application>](https://developer.android.com/guide/topics/manifest/application-element.html) element to use one of appcompat's [NoActionBar](https://developer.android.com/reference/android/support/v7/appcompat/R.style.html" \l "Theme_AppCompat_NoActionBar) themes. Using one of these themes prevents the app from using the native [ActionBar](https://developer.android.com/reference/android/app/ActionBar.html) class to provide the app bar. For example:

<application  
    android:theme="@style/Theme.AppCompat.Light.NoActionBar"  
    />

1. Add a [Toolbar](https://developer.android.com/reference/android/support/v7/widget/Toolbar.html) to the activity's layout. For example, the following layout code adds a [Toolbar](https://developer.android.com/reference/android/support/v7/widget/Toolbar.html) and gives it the appearance of floating above the activity:

<android.support.v7.widget.Toolbar  
   android:id="@+id/my\_toolbar"  
   android:layout\_width="match\_parent"  
   android:layout\_height="?attr/actionBarSize"  
   android:background="?attr/colorPrimary"  
   android:elevation="4dp"  
   android:theme="@style/ThemeOverlay.AppCompat.ActionBar"  
   app:popupTheme="@style/ThemeOverlay.AppCompat.Light"/>

The [Material Design specification](https://www.google.com/design/spec/what-is-material/elevation-shadows.html#elevation-shadows-shadows) recommends that app bars have an elevation of 4 dp.

Position the toolbar at the top of the activity's [layout](https://developer.android.com/guide/topics/ui/declaring-layout.html), since you are using it as an app bar.

1. In the activity's [onCreate()](https://developer.android.com/reference/android/app/Activity.html" \l "onCreate(android.os.Bundle)) method, call the activity's [setSupportActionBar()](https://developer.android.com/reference/android/support/v7/app/AppCompatActivity.html" \l "setSupportActionBar(android.support.v7.widget.Toolbar)) method, and pass the activity's toolbar. This method sets the toolbar as the app bar for the activity. For example:

JAVA

@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity\_my);  
    **Toolbar myToolbar = (Toolbar) findViewById(R.id.my\_toolbar);  
    setSupportActionBar(myToolbar);**  
}

Your app now has a basic action bar. By default, the action bar contains just the name of the app and an overflow menu. The options menu initially contains just the **Settings** item. You can add more actions to the action bar and the overflow menu, as described in [Adding and Handling Actions](https://developer.android.com/training/appbar/actions.html).

**Use App Bar Utility Methods**

Once you set the toolbar as an activity's app bar, you have access to the various utility methods provided by the [v7 appcompat](https://developer.android.com/tools/support-library/features.html#v7-appcompat) support library's [ActionBar](https://developer.android.com/reference/android/support/v7/app/ActionBar.html) class. This approach lets you do a number of useful things, like hide and show the app bar.

To use the [ActionBar](https://developer.android.com/reference/android/support/v7/app/ActionBar.html) utility methods, call the activity's [getSupportActionBar()](https://developer.android.com/reference/android/support/v7/app/AppCompatActivity.html" \l "getSupportActionBar()) method. This method returns a reference to an appcompat [ActionBar](https://developer.android.com/reference/android/support/v7/app/ActionBar.html) object. Once you have that reference, you can call any of the [ActionBar](https://developer.android.com/reference/android/support/v7/app/ActionBar.html) methods to adjust the app bar. For example, to hide the app bar, call [ActionBar.hide()](https://developer.android.com/reference/android/support/v7/app/ActionBar.html" \l "hide()).

**Add Action Buttons**

All action buttons and other items available in the action overflow are defined in an XML [menu resource](https://developer.android.com/guide/topics/resources/menu-resource.html). To add actions to the action bar, create a new XML file in your project's res/menu/ directory.

Add an [<item>](https://developer.android.com/guide/topics/resources/menu-resource.html#item-element) element for each item you want to include in the action bar, as shown in this code example of a menu XML file:

<menu xmlns:android="http://schemas.android.com/apk/res/android" >  
  
    <!-- "Mark Favorite", should appear as action button if possible -->  
    <item  
        android:id="@+id/action\_favorite"  
        android:icon="@drawable/ic\_favorite\_black\_48dp"  
        android:title="@string/action\_favorite"  
        app:showAsAction="ifRoom"/>  
  
    <!-- Settings, should always be in the overflow -->  
    <item android:id="@+id/action\_settings"  
          android:title="@string/action\_settings"  
          app:showAsAction="never"/>  
  
</menu>

The app:showAsAction attribute specifies whether the action should be shown as a button on the app bar. If you set app:showAsAction="ifRoom" (as in the example code's *favorite* action), the action is displayed as a button if there is room in the app bar for it; if there is not enough room, excess actions are sent to the overflow menu. If you set app:showAsAction="never" (as in the example code's *settings* action), the action is always listed in the overflow menu, not displayed in the app bar.

The system uses the action's icon as the action button if the action is displayed in the app bar. You can find many useful icons on the [Material Icons](https://www.google.com/design/icons/) page.

You can declare items for the options menu from either your [Activity](https://developer.android.com/reference/android/app/Activity.html)subclass or a [Fragment](https://developer.android.com/reference/android/app/Fragment.html) subclass. If both your activity and fragment(s) declare items for the options menu, they are combined in the UI. The activity's items appear first, followed by those of each fragment in the order in which each fragment is added to the activity. If necessary, you can re-order the menu items with the android:orderInCategory attribute in each <item> you need to move.

To specify the options menu for an activity, override [onCreateOptionsMenu()](https://developer.android.com/reference/android/app/Activity.html" \l "onCreateOptionsMenu(android.view.Menu))(fragments provide their own [onCreateOptionsMenu()](https://developer.android.com/reference/android/app/Fragment.html" \l "onCreateOptionsMenu(android.view.Menu, android.view.MenuInflater)) callback). In this method, you can inflate your menu resource ([defined in XML](https://developer.android.com/guide/topics/ui/menus#xml)) into the [Menu](https://developer.android.com/reference/android/view/Menu.html)provided in the callback. For example:

@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    MenuInflater inflater = [**getMenuInflater()**](https://developer.android.com/reference/android/app/Activity.html#getMenuInflater());  
    inflater.inflate(R.menu.game\_menu, menu);  
    return true;  
}

You can also add menu items using [add()](https://developer.android.com/reference/android/view/Menu.html#add(int, int, int, int)) and retrieve items with [findItem()](https://developer.android.com/reference/android/view/Menu.html" \l "findItem(int)) to revise their properties with [MenuItem](https://developer.android.com/reference/android/view/MenuItem.html) APIs.

If you've developed your application for Android 2.3.x and lower, the system calls [onCreateOptionsMenu()](https://developer.android.com/reference/android/app/Activity.html" \l "onCreateOptionsMenu(android.view.Menu)) to create the options menu when the user opens the menu for the first time. If you've developed for Android 3.0 and higher, the system calls [onCreateOptionsMenu()](https://developer.android.com/reference/android/app/Activity.html" \l "onCreateOptionsMenu(android.view.Menu)) when starting the activity, in order to show items to the app bar.

**Respond to Actions**

When the user selects one of the app bar items, the system calls your activity's [onOptionsItemSelected()](https://developer.android.com/reference/android/app/Activity.html" \l "onOptionsItemSelected(android.view.MenuItem)) callback method, and passes a [MenuItem](https://developer.android.com/reference/android/view/MenuItem.html) object to indicate which item was clicked. In your implementation of [onOptionsItemSelected()](https://developer.android.com/reference/android/app/Activity.html" \l "onOptionsItemSelected(android.view.MenuItem)), call the [MenuItem.getItemId()](https://developer.android.com/reference/android/view/MenuItem.html" \l "getItemId()) method to determine which item was pressed. The ID returned matches the value you declared in the corresponding [<item>](https://developer.android.com/guide/topics/resources/menu-resource.html#item-element) element's android:id attribute.

For example, the following code checks to see which action the user selected. If the method does not recognize the user's action, it invokes the superclass method:

JAVA

@Override  
public boolean onOptionsItemSelected(MenuItem item) {  
    switch (item.getItemId()) {  
        case R.id.action\_settings:  
            // User chose the "Settings" item, show the app settings UI...  
            return true;  
  
        case R.id.action\_favorite:  
            // User chose the "Favorite" action, mark the current item  
            // as a favorite...  
            return true;  
  
        default:  
            // If we got here, the user's action was not recognized.  
            // Invoke the superclass to handle it.  
            return super.onOptionsItemSelected(item);  
  
    }  
}

## 使用Toast——限时浮现式弹窗

Toast是Android中用来显示显示信息的一种机制，和Dialog不一样的是，Toast是没有焦点的，而且Toast显示的时间有限，过一定的时间就会自动消失。下面用一个实例来看看如何使用Toast。

 1.默认效果



代码

Toast.makeText(getApplicationContext(), "默认Toast样式",  
     Toast.LENGTH\_SHORT).show();

 2.自定义显示位置效果



代码

toast = Toast.makeText(getApplicationContext(),  
     "自定义位置Toast", Toast.LENGTH\_LONG);  
   toast.setGravity(Gravity.CENTER, 0, 0);  
   toast.show();

 3.带图片效果



 代码

toast = Toast.makeText(getApplicationContext(),  
     "带图片的Toast", Toast.LENGTH\_LONG);  
   toast.setGravity(Gravity.CENTER, 0, 0);  
   LinearLayout toastView = (LinearLayout) toast.getView();  
   ImageView imageCodeProject = new ImageView(getApplicationContext());  
   imageCodeProject.setImageResource(R.drawable.icon);  
   toastView.addView(imageCodeProject, 0);  
   toast.show();

 4.完全自定义效果



代码

LayoutInflater inflater = getLayoutInflater();  
   View layout = inflater.inflate(R.layout.custom,  
     (ViewGroup) findViewById(R.id.llToast));  
   ImageView image = (ImageView) layout  
     .findViewById(R.id.tvImageToast);  
   image.setImageResource(R.drawable.icon);  
   TextView title = (TextView) layout.findViewById(R.id.tvTitleToast);  
   title.setText("Attention");  
   TextView text = (TextView) layout.findViewById(R.id.tvTextToast);  
   text.setText("完全自定义Toast");  
   toast = new Toast(getApplicationContext());  
   toast.setGravity(Gravity.RIGHT | Gravity.TOP, 12, 40);  
   toast.setDuration(Toast.LENGTH\_LONG);  
   toast.setView(layout);  
   toast.show();

 5.其他线程



 代码

new Thread(new Runnable() {  
    public void run() {  
     showToast();  
    }  
   }).start();

## 使用DialogFragment与FragmentManager进行对话框显示

**Basic Dialog**

The simplest use of DialogFragment is as a floating container for the fragment's view hierarchy. A simple implementation may look like this:

public static class MyDialogFragment extends DialogFragment {

int mNum;

/\*\*

\* Create a new instance of MyDialogFragment, providing "num"

\* as an argument.

\*/

static MyDialogFragment newInstance(int num) {

MyDialogFragment f = new MyDialogFragment();

// Supply num input as an argument.

Bundle args = new Bundle();

args.putInt("num", num);

f.setArguments(args);

return f;

}

@Override

public void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

mNum = getArguments().getInt("num");

// Pick a style based on the num.

int style = DialogFragment.STYLE\_NORMAL, theme = 0;

switch ((mNum-1)%6) {

case 1: style = DialogFragment.STYLE\_NO\_TITLE; break;

case 2: style = DialogFragment.STYLE\_NO\_FRAME; break;

case 3: style = DialogFragment.STYLE\_NO\_INPUT; break;

case 4: style = DialogFragment.STYLE\_NORMAL; break;

case 5: style = DialogFragment.STYLE\_NORMAL; break;

case 6: style = DialogFragment.STYLE\_NO\_TITLE; break;

case 7: style = DialogFragment.STYLE\_NO\_FRAME; break;

case 8: style = DialogFragment.STYLE\_NORMAL; break;

}

switch ((mNum-1)%6) {

case 4: theme = android.R.style.Theme\_Holo; break;

case 5: theme = android.R.style.Theme\_Holo\_Light\_Dialog; break;

case 6: theme = android.R.style.Theme\_Holo\_Light; break;

case 7: theme = android.R.style.Theme\_Holo\_Light\_Panel; break;

case 8: theme = android.R.style.Theme\_Holo\_Light; break;

}

setStyle(style, theme);

}

@Override

public View onCreateView(LayoutInflater inflater, ViewGroup container,

Bundle savedInstanceState) {

View v = inflater.inflate(R.layout.fragment\_dialog, container, false);

View tv = v.findViewById(R.id.text);

((TextView)tv).setText("Dialog #" + mNum + ": using style "

+ getNameForNum(mNum));

// Watch for button clicks.

Button button = (Button)v.findViewById(R.id.show);

button.setOnClickListener(new OnClickListener() {

public void onClick(View v) {

// When button is clicked, call up to owning activity.

((FragmentDialog)getActivity()).showDialog();

}

});

return v;

}

}

An example showDialog() method on the Activity could be:

void showDialog() {

mStackLevel++;

// DialogFragment.show() will take care of adding the fragment

// in a transaction. We also want to remove any currently showing

// dialog, so make our own transaction and take care of that here.

FragmentTransaction ft = getFragmentManager().beginTransaction();

//每个fragment都有一个类似id的tag可用于查找，可以通过xml或代码设置

Fragment prev = getFragmentManager().findFragmentByTag("dialog");

if (prev != null) {

ft.remove(prev);

}

ft.addToBackStack(null);

// Create and show the dialog.

DialogFragment newFragment = MyDialogFragment.newInstance(mStackLevel);

//通过代码顺便设置了newFragment的tag为dialog

newFragment.show(ft, "dialog");

}

This removes any currently shown dialog, creates a new DialogFragment with an argument, and shows it as a new state on the back stack. When the transaction is popped, the current DialogFragment and its Dialog will be destroyed, and the previous one (if any) re-shown. Note that in this case DialogFragment will take care of popping the transaction of the Dialog is dismissed separately from it.

**Alert Dialog**

Instead of (or in addition to) implementing [Fragment.onCreateView(LayoutInflater, ViewGroup, Bundle)](https://developer.android.com/reference/android/app/Fragment.html" \l "onCreateView(android.view.LayoutInflater,%20android.view.ViewGroup,%20android.os.Bundle)) to generate the view hierarchy inside of a dialog, you may implement[onCreateDialog(Bundle)](https://developer.android.com/reference/android/app/DialogFragment.html#onCreateDialog(android.os.Bundle)) to create your own custom Dialog object.

This is most useful for creating an [AlertDialog](https://developer.android.com/reference/android/app/AlertDialog.html), allowing you to display standard alerts to the user that are managed by a fragment. A simple example implementation of this is:

public static class MyAlertDialogFragment extends DialogFragment {

public static MyAlertDialogFragment newInstance(int title) {

MyAlertDialogFragment frag = new MyAlertDialogFragment();

Bundle args = new Bundle();

args.putInt("title", title);

frag.setArguments(args);

return frag;

}

@Override

public Dialog onCreateDialog(Bundle savedInstanceState) {

int title = getArguments().getInt("title");

return new AlertDialog.Builder(getActivity())

.setIcon(R.drawable.alert\_dialog\_icon)

.setTitle(title)

.setPositiveButton(R.string.alert\_dialog\_ok,

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {

((FragmentAlertDialog)getActivity()).doPositiveClick();

}

}

)

.setNegativeButton(R.string.alert\_dialog\_cancel,

new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int whichButton) {

((FragmentAlertDialog)getActivity()).doNegativeClick();

}

}

)

.create();

}

}

The activity creating this fragment may have the following methods to show the dialog and receive results from it:

void showDialog() {

DialogFragment newFragment = MyAlertDialogFragment.newInstance(

R.string.alert\_dialog\_two\_buttons\_title);

newFragment.show(getFragmentManager(), "dialog");

}

public void doPositiveClick() {

// Do stuff here.

Log.i("FragmentAlertDialog", "Positive click!");

}

public void doNegativeClick() {

// Do stuff here.

Log.i("FragmentAlertDialog", "Negative click!");

}

Note that in this case the fragment is not placed on the back stack, it is just added as an indefinitely running fragment. Because dialogs normally are modal, this will still operate as a back stack, since the dialog will capture user input until it is dismissed. When it is dismissed, DialogFragment will take care of removing itself from its fragment manager.

## 使用Notification系统服务发送系统提示

**Add the support library**

Although most projects created with Android Studio include the necessary dependencies to use [NotificationCompat](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.html), you should verify that your module-level build.gradle file includes the following dependency:

dependencies {  
    implementation "com.android.support:support-compat:27.1.1"  
}

**Create a basic notification**

Set the notification content

To get started, you need to set the notification's content and channel using a[NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html) object. The following example shows how to create a notification with the following:

* A small icon, set by [setSmallIcon()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setSmallIcon(int)). This is the only user-visible content that's required.
* A title, set by [setContentTitle()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setContentTitle(java.lang.CharSequence)).
* The body text, set by [setContentText()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setContentText(java.lang.CharSequence)).
* The notification priority, set by [setPriority()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setPriority(int)). The priority determines how intrusive the notification should be on Android 7.1 and lower. (For Android 8.0 and higher, you must instead set the channel importance—shown in the next section.)

NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this, CHANNEL\_ID)  
        .setSmallIcon(R.drawable.notification\_icon)  
        .setContentTitle(textTitle)  
        .setContentText(textContent)  
        .setPriority(NotificationCompat.PRIORITY\_DEFAULT);

Notice that the [NotificationCompat.Builder](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html) constructor requires that you provide a channel ID. This is required for compatibility with Android 8.0 (API level 26) and higher, but is ignored by older versions.

By default, the notification's text content is truncated to fit one line. If you want your notification to be longer, you can enable an expandable notification by adding a style template with [setStyle()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setStyle(android.support.v4.app.NotificationCompat.Style)). For example, the following code creates a larger text area:

NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this, CHANNEL\_ID)  
        .setSmallIcon(R.drawable.notification\_icon)  
        .setContentTitle("My notification")  
        .setContentText("Much longer text that cannot fit one line...")  
        **.setStyle(new NotificationCompat.BigTextStyle()  
                .bigText("Much longer text that cannot fit one line..."))**  
        .setPriority(NotificationCompat.PRIORITY\_DEFAULT);

For more information about other large notification styles, including how to add an image and media playback controls, see [Create a Notification with Expandable Detail](https://developer.android.com/training/notify-user/expanded.html).

**Create a channel and set the importance**

Before you can deliver the notification on Android 8.0 and higher, you must register your app's [notification channel](https://developer.android.com/training/notify-user/channels.html) with the system by passing an instance of [NotificationChannel](https://developer.android.com/reference/android/app/NotificationChannel.html) to [createNotificationChannel()](https://developer.android.com/reference/android/app/NotificationManager.html" \l "createNotificationChannel(android.app.NotificationChannel)). So the following code is blocked by a condition on the [SDK\_INT](https://developer.android.com/reference/android/os/Build.VERSION.html#SDK_INT) version:

private void createNotificationChannel() {  
    // Create the NotificationChannel, but only on API 26+ because  
    // the NotificationChannel class is new and not in the support library  
    if (Build.VERSION.SDK\_INT >= Build.VERSION\_CODES.O) {  
        CharSequence name = getString(R.string.channel\_name);  
        String description = getString(R.string.channel\_description);  
        int importance = NotificationManager.IMPORTANCE\_DEFAULT;  
        NotificationChannel channel = new NotificationChannel(CHANNEL\_ID, name, importance);  
        channel.setDescription(description);  
        // Register the channel with the system; you can't change the importance  
        // or other notification behaviors after this  
        NotificationManager notificationManager = getSystemService(NotificationManager.class);  
        notificationManager.createNotificationChannel(channel);  
    }  
}  
Because you must create the notification channel before posting any notifications on Android 8.0 and higher, you should execute this code as soon as your app starts. It's safe to call this repeatedly because creating an existing notification channel performs no operation.

Notice that the [NotificationChannel](https://developer.android.com/reference/android/app/NotificationChannel.html) constructor requires an importance, using one of the constants from the [NotificationManager](https://developer.android.com/reference/android/app/NotificationManager) class. This parameter determines how to interrupt the user for any notification that belongs to this channel—though you must also set the *priority* with [setPriority()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setPriority(int)) to support Android 7.1 and lower (as shown above).

Although you must set the notification importance/priority as shown here, the system does not guarantee the alert behavior you'll get. In some cases the system might change the importance level based other factors, and the user can always redefine what the importance level is for a given channel.

For more information about what the different levels mean, read about[notification importance levels](https://developer.android.com/training/notify-user/channels.html#importance).

Set the notification's tap action

Every notification should respond to a tap, usually to open an activity in your app that corresponds to the notification. To do so, you must specify a content intent defined with a [PendingIntent](https://developer.android.com/reference/android/app/PendingIntent.html) object and pass it to [setContentIntent()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setContentIntent(android.app.PendingIntent)).

The following snippet shows how to create a basic intent to open an activity when the user taps the notification:

// Create an explicit intent for an Activity in your app  
**Intent intent = new Intent(this, AlertDetails.class);  
intent.setFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK | Intent.FLAG\_ACTIVITY\_CLEAR\_TASK);  
PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);**  
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this, CHANNEL\_ID)  
        .setSmallIcon(R.drawable.notification\_icon)  
        .setContentTitle("My notification")  
        .setContentText("Hello World!")  
        .setPriority(NotificationCompat.PRIORITY\_DEFAULT)  
        // Set the intent that will fire when the user taps the notification  
        **.setContentIntent(pendingIntent)**  
        .setAutoCancel(true);

Notice this code calls [setAutoCancel()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "setAutoCancel(boolean)), which automatically [removes the notification](https://developer.android.com/training/notify-user/build-notification#Removing) when the user taps it.

The [setFlags()](https://developer.android.com/reference/android/content/Intent.html" \l "setFlags(int)) method shown above helps preserve the user's expected navigation experience after they open your app via the notification. But whether you want to use that depends on what type of activity you're starting, which may be one of the following:

* An activity that exists exclusively for responses to the notification. There's no reason the user would navigate to this activity during normal app use, so the activity starts a new task instead of being added to your app's existing [task and back stack](https://developer.android.com/guide/components/activities/tasks-and-back-stack.html). This is the type of intent created in the sample above.
* An activity that exists in your app's regular app flow. In this case, starting the activity should create a back stack so that the user's expectations for the [Back and Up buttons](https://developer.android.com/design/patterns/navigation.html) is preserved.

For more about the different ways to configure your notification's intent, read[Start an Activity from a Notification](https://developer.android.com/training/notify-user/navigation.html).

**Show the notification**

To make the notification appear, call [NotificationManagerCompat.notify()](https://developer.android.com/reference/android/support/v4/app/NotificationManagerCompat.html" \l "notify(int, android.app.Notification)), passing it a unique ID for the notification and the result of [NotificationCompat.Builder.build()](https://developer.android.com/reference/android/support/v4/app/NotificationCompat.Builder.html" \l "build()). For example:

NotificationManagerCompat notificationManager = NotificationManagerCompat.from(this);  
// notificationId is a unique int for each notification that you must define  
notificationManager.notify(notificationId, mBuilder.build());  
Remember to save the notification ID that you pass to [NotificationManagerCompat.notify()](https://developer.android.com/reference/android/support/v4/app/NotificationManagerCompat.html" \l "notify(int, android.app.Notification)) because you'll need it later if you want to [update](https://developer.android.com/training/notify-user/build-notification#Updating) or [remove the notification](https://developer.android.com/training/notify-user/build-notification#Removing).

如何在Notification中添加除了单击响应外的其他功能，如Action按键和直接回复窗口等，参考Android Developers Guide。