



globsyn

*Taking People To The Next Level*

# globalsyn



**globsyn**   
**finishing school**

# User Interface design



---

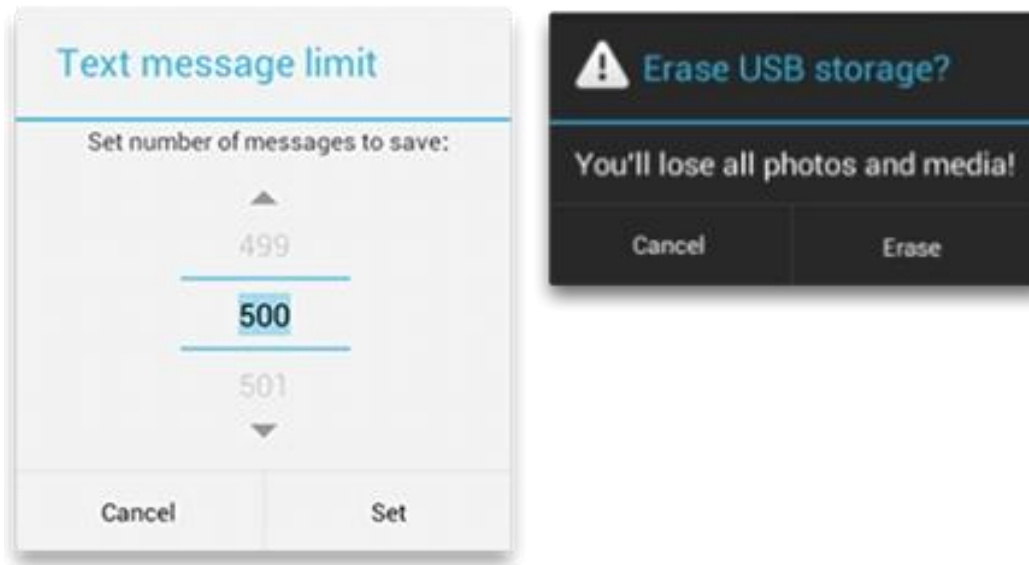
## | User Interface design

♣ Topics to be covered in this session:

- Dialog
- Settings
- Notifications

# Dialogs

Dialog is a small window that is used to prompt the user to enter a value to value or make a decision.



Dialog windows do not occupy the entire space of a screen and are used as modal events.

# Usage of Dialogs

- Dialogs are very vital. They are used to obtain an Yes/No (Cancel / OK) input from the user in form of prompts.
- Dialogs are also used to obtain small, input values from the users. Without Dialogs, this same process could take loads of time, processing capacity and effort to complete.
- Similarly, Dialogs are easy to implement with less hassles.

# Designing a Dialog Class

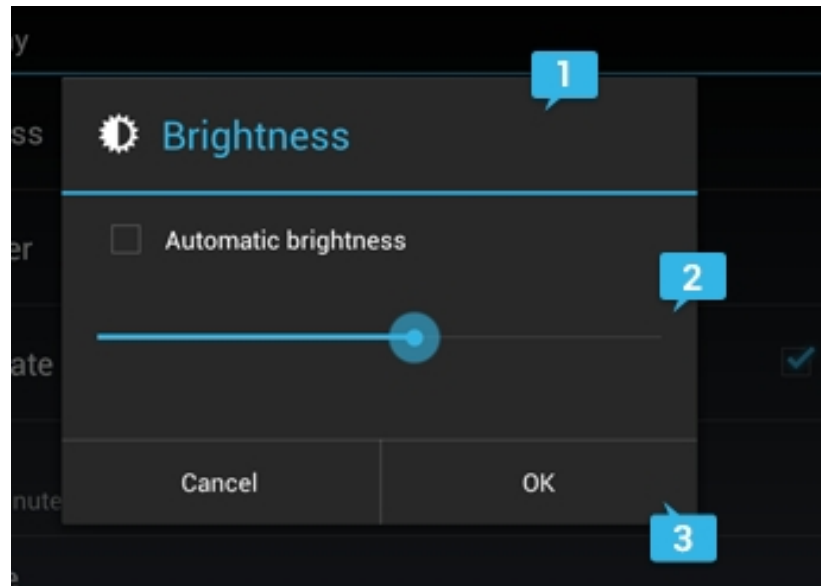
Any Dialog Class has three main areas.

- ♣ **The Title Region:** This is an optional field that can be used to give some title to the Dialog.
- ♣ **Content Area:** This area displays the content of the Dialog. It could be a Text display or a slider in case of changing Brightness and so on.



# Designing a Dialog Class Contd.

♣ **Action Buttons** : Usually, there are two Actions Buttons in most of the Dialog Boxes – Cancel and Ok. The dismissive action, Cancel is always placed on the left with the affirmative action, OK being placed on the right.



# Designing the Dialogs

- There is a class called as Dialog based on which we will design our Dialog box.
- Using a predefined instantiation of the Dialog class such as AlertDialog or TimePickerDialog, you can instantiate your Dialog class.
- To define the style of your dialog, you can use the DialogFragment class.
- The DialogFragment works just like a normal Fragment with many built-in options that supports embedding and reusing of the User Interface.

# What is a Dialog Fragment?

- 1 A Dialog Fragment as mentioned above is a class when extended provides custom layouts and designs.
- 2 You can manage an AlertDialog within a DialogFragment to do suitable actions.
- 3 After creating that, you can use a show() method to display that DialogFragment.

# Building AlertDialog

- 1 The AlertDialog is the major instantiation that is used in the creation of your Dialogs.
- 2 As mentioned earlier, it contains three major parts



# Building AlertDialog Contd.

```
AlertDialog.Builder builder = new AlertDialog.Builder(getActivity());  
builder.setMessage(R.string.dialog_message).setTitle(R.string.dialog  
_title);  
AlertDialog dialog = builder.create();
```

- | Take a look at the three lines of code above.
- | In the first line, using AlertDialog.Builder, a new Builder class has been instantiated.
- | In the second line, the Dialog was chained with various other methods to obtain favorable characteristics.
- | Finally, the dialog was created.

# Inserting Buttons

- | The Dialog would become meaningless without Buttons. Hence, you have to use them.
- | There are three options for creating and inserting Buttons. They are
  - ♣ **setPositiveButton()** – Used for the OK Button. It accepts the Value.
  - ♣ **setNegativeButton()** – Used for the Cancel Button. Rejects the Dialog values.
  - ♣ **setNeutralButton()** – Neither creates nor destroys. A good example is “reset” or “remind later”

# Create a Custom Layout

- 1 You may not like the Layout that the default Dialog box provides. If that is the case, you can use `setView()` method to change that layout.
- 2 Let us assume that we have created a new Layout file in XML and have named it `dialog_signin.xml`.
- 3 Now, create a new builder using `AlertDialog.Builder` and insert the following line to obtain a custom layout,  
`builder.setView(inflater.inflate(R.layout.dialog_signin,null))`

# Displaying the Dialog

- There is no use in creating the Dialog if you are not going to use it. Hence, you need a way to display the dialog.
- The show() method can provide you just that.

```
public void confirmFireMissiles() {  
    DialogFragment newFragment = new FireMissilesDialogFragment();  
    newFragment.show(getSupportFragmentManager(), "missiles");  
}
```

- By using the above code, you would have displayed the FireMissiles dialog box.



# Settings

- 1 You probably would have seen the Settings in an Android mobile phone. It contains various options that modify the working of the phone.
- 2 Similarly, you can also decide to give Settings for your application to allow the user customize certain things.
- 3 Settings can be implemented using the android built-in Preferences API.

# Why Should You Use Settings?

- 1 If you let the users predefine and select certain Settings on their own, it would reduce your work load drastically.
- 2 Also, by letting the users choose the Settings, you can give them a sense of control over the app.
- 3 Although Settings are important, do not give high prominence to them. When it comes to an Action Bar, keep them in the stack overflow.

# How Should You Define a Setting?

- Generally, Settings are not a very prominent part of the User Interface Design. They are not compulsory but are used anyway.
- First of all, be very clear. Do not make every single, irrelevant detail, a Setting. Having too many options to modify would make the user experience overwhelming.

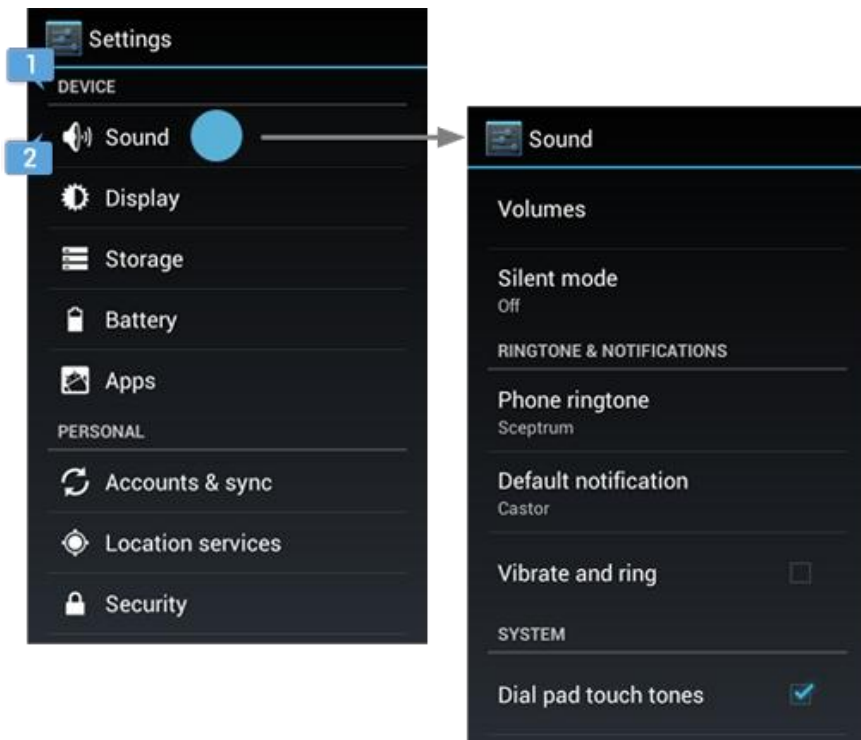
## How Should You Define a Setting? Contd.

- First, take any preference that you want to make as a Setting. Now, think and rationalize if many users would want to change it. If not, do not make it a setting.
- If yes, then check if there is an exact solution to the users. If there is an exact answer, go to your source code and modify your program to suit the exact answer rather than providing a Setting where users can change it.
- If there is no exact solution, create the setting.

# Grouping Related Settings together

- 1 The presence of large number of settings is more a harm than good.
- 2 The presence of that many settings will make the app look heavy and it might end up confounding the users which is no good at all.
- 3 So, use sub dividers and separate activities to reduce the number of settings that you have used. This will give a sense of organization to your Settings screen.

# Example for Efficient Settings Grouping



- So, how do you efficiently group Settings?
- In this example, the various Sound settings are grouped into a separate page.
- Similarly, various subsections like Device and Personal are present to reduce the complexity.

# Various Design Patterns

- | You need to design separate and individual design patterns to suit different settings items.
- | For instance, let us take a look at some of the common Design Patterns that are used in Android Settings.
  - ♣ **Checkbox**– A Checkbox is used to either select or not select a particular field. For instance, “Auto-Rotate” is a select or a not-select type of field and Checkboxes can be used with them.

# Various Design Patterns Contd.

- ♣ **Multiple Choices**– For instance, take Font Size. You will provide, four choices, Very Large, Large, Small and Very Small in that setting. Hence, there you should use Multiple Choice.
- ♣ **Slider**– Again, this is another very important design pattern that is used in Android. It can be used for Volumes, Brightness and a whole lot of other values.
- ♣ **On/Off Switch**– Used to Switch On/Off a service. It could be Wi-Fi, Bluetooth or anything related.



# Defining Settings Using Preferences

- 1 The Settings in Android are not defined or modified using the View API. They use the Preferences API.
- 2 For instance, if you want to add a CheckBox element for a particular Settings, you have to create a `<checkboxpreference>` subclass.
- 3 All these must be contained inside the `<PreferenceScreen>` class.
- 4 Like `checkboxpreference`, there are other elements like `Listpreference` and so on.

# CheckBoxPreference Subclass

- Like mentioned before, `<CheckboxPreference>` is a subclass that has to be defined if you need an item inside the Check Box of an element's setting.
- For each item, you need to declare a similar `CheckBoxPreference` with the four attributes, key, title, summary and defaultvalue

```
<CheckBoxPreference
    android:key="pref_sync"
    android:title="@string/pref_sync"
    android:summary="@string/pref_sync_summ"
    android:defaultValue="true" />
```

# Grouping Settings Together

- As mentioned earlier, there are two ways to group the settings together. The first method is to group them through titles.
- Inside the `<preferenceScreen>`, you can open a `<PreferenceCategory>` subclass with only the key and title.
- Inside the `<PreferenceCategory>`, you can usually declare all your preferences. Now, all the elements inside this `PreferenceCategory` will be grouped together under a single name.

## Grouping Settings Together Contd.

- 1 The second method to group the Settings would be under a new screen.
- 2 To do that, you need a `<PreferenceScreen>` subclass that will contain all the `ListPreference` and `CheckBoxPreference` and other suitable items. All the items listed under this `PreferenceScreen` will be grouped into a separate screen.

# Guidelines for Writing Setting Description

- There are Guidelines that you need to follow before you write the text for Settings.
- That is, your sentences should be imperative and direct. They should not be passive.
- They should not be a description but a status.

For example, “Vibrate on Touch” is a better term than “Activate Haptical Feedback on a touch gesture”. Hence, it is used.

# Notifications

---

- Notifications are display messages about various information that keep the user informed about the happenings of the system.
- Notifications can be provided for instances like New E-mails, new Text messages, change in network signals and so on.

# How do You Access Notifications?

The Notifications first appear in the Notification area on top of your UI. When the user pulls down the Notification area, the Notification drawer opens.



The image shows the Notification area in Android.

# Views of Notifications

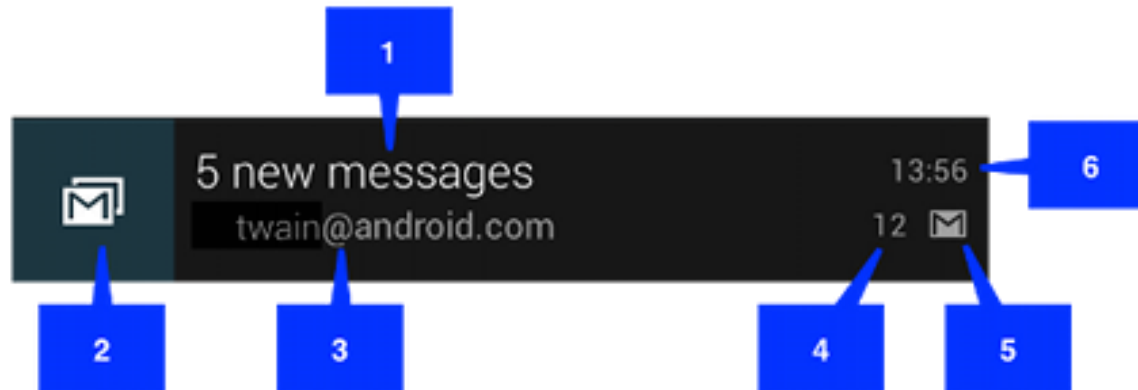
There are two views of Notification,

- ♣ The Standard View is the view that you see when you drag down the Navigation drawer.
- ♣ The Big View is something that you see once you “expand” the notification in the drawer. This function of “expanding” the notification is only available in Android 4.1 JellyBean and not on earlier versions.



# Standard View of Notification

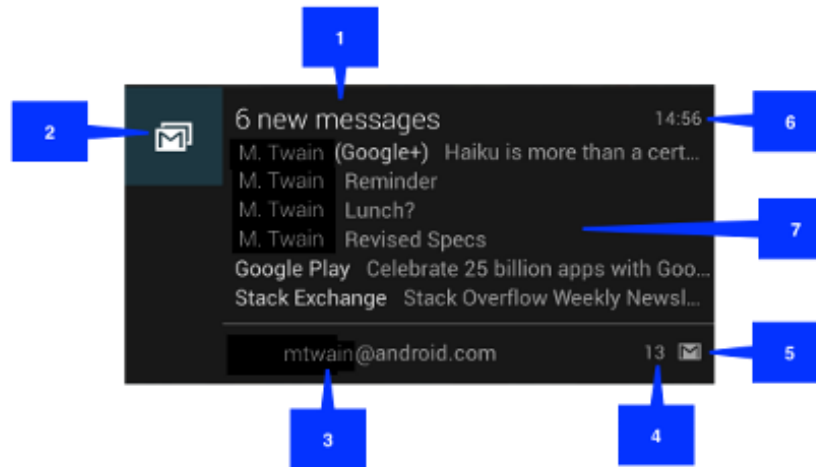
The Standard Notification View contains the following components



1 – Content, 2- Large Icon of the Content type, 3 – Content Text, 4 – Content Info, 5 – Small Icon of the Content, 6 – Time of issue of the notification

# Big View of Notification

The Notification in the drawer can be expanded with the help of a touch gesture.



The only difference between this and normal notification is the fact that Big View supports large amount of data view.

# How to Create Notifications?

- Creating Notifications is carried out by the `NotificationCompat.Builder.build()` object which returns an `Notification`.
- If you want to issue a new notification, you need to use the `NotificationManager.notify()` method.
- During the declaration of any notification, keep in mind that all these notifications should contain `setSmallIcon`, `setContentTitle` and `setContentText` functions.

# How to Create Notifications? Contd.

You can create a new notification using the following lines of code.

```
NotificationCompat.Builder mBuilder =  
    new NotificationCompat.Builder(this)  
        .setSmallIcon(R.drawable.notification_icon)  
        .setContentTitle("My notification")  
        .setContentText("Hello World!");
```

After creation, you must define a PendingIntent which will call upon an activity which suits your notification.

This is generally the standard view. If you want to apply Big View, you need to apply the Builder.setStyle attribute to BigView

# How to Remove Notifications?

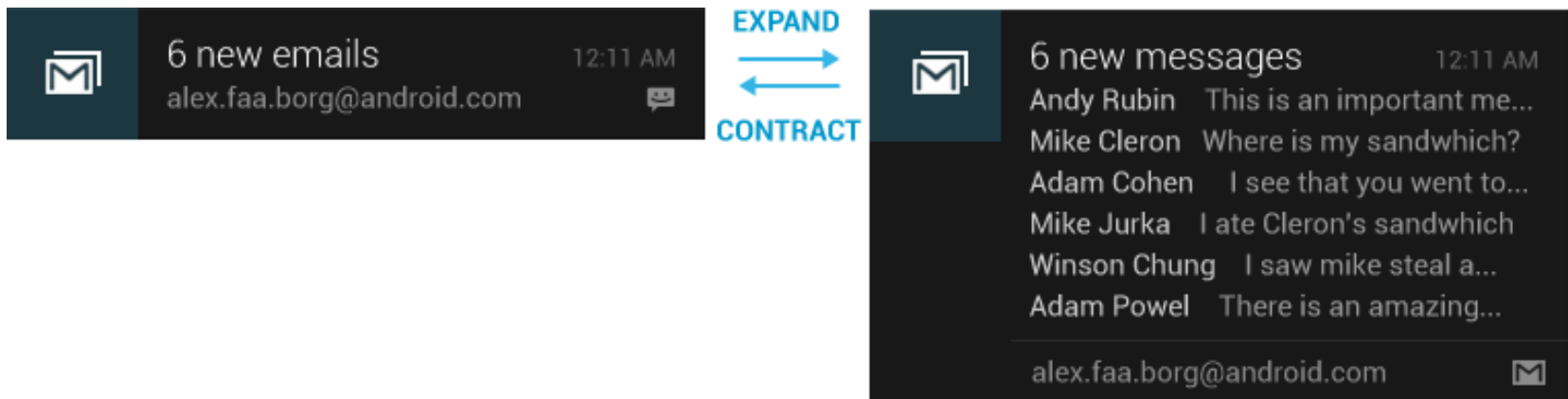
- 1 So, a Notification has been reported. How do you remove it? There are 4 possible ways to do that.
- 2 You click the Clear All button at the bottom of the Notification drawer.
- 3 You view the notification which will activate the `setAutoCancel()` method.
- 4 You call the `cancel()` method for that notification.
- 5 You call `cancelAll()` method that will clear all notifications.

# Priority in Notifications

- Through the latest Android Update, you can now define priority values for Notifications. High priority notifications will obviously be rated higher than others.
- The six different priority flag values for Notifications are Max, High, Default, Low and Min in that order with Max meaning Maximum priority and Min meaning least priority.

# Stacking Similar Notifications

- Assume that you have received two new text messages. Rather than displaying them as two separate notifications, now the Android will display them as one notification with multiple numbers.
- If you have received 6 emails, the Android notification drawer will not show 6 notifications, it will show a single notification saying that you have “6 new Emails”



# References

Android Developers :

**developer.android.com/reference/android/app/Notification.html**


Sai Geetha, CIT:

saigeethamn.blogspot.com/2009/09/**android-developer**-tutorial-for.html

Yong Mook : [www.mkyong.com/android/android-custom-dialog-example/](http://www.mkyong.com/android/android-custom-dialog-example/)

Phil Nickinson: <http://www.androidcentral.com/how-enable-developer-settings-android-42>





*Taking People To The Next Level ...*