Displaying Progress in a Notification

Notifications can include an animated progress indicator that shows users the status of an ongoing operation. If you can estimate how long the operation takes and how much of it is complete at any time, use the "determinate" form of the indicator (a progress bar). If you can't estimate the length of the operation, use the "indeterminate" form of the indicator (an activity indicator).

Progress indicators are displayed with the platform's implementation of the `ProgressBar` `(/reference/android/widget/ProgressBar.html)` class.

To use a progress indicator, call `setProgress()` `(/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean))`. The determinate and indeterminate forms are described in the following sections.

THIS LESSON TEACHES YOU TO

1. Display a Fixed-duration progress Indicator
2. Display a Continuing Activity Indicator

YOU SHOULD ALSO READ

- Notifications API Guide
- Intents and Intent Filters
- Notifications Design Guide

Display a Fixed-duration Progress Indicator

To display a determinate progress bar, add the bar to your notification by calling `setProgress(max, progress, false)` `(/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean))` and then issue the notification. The third argument is a boolean that indicates whether the progress bar is indeterminate (**true**) or determinate (**false**). As your operation proceeds, increment `progress`, and update the notification. At the end of the operation, `progress` should equal `max`. A common way to call `setProgress()` `(/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean))` is to set `max` to 100 and then increment `progress` as a "percent complete" value for the operation.

You can either leave the progress bar showing when the operation is done, or remove it. In either case, remember to update the notification text to show that the operation is complete. To remove the progress bar, call `setProgress(0, 0, false)` `(/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean))`. For example:

```
...
mNotifyManager =
        (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
mBuilder = new NotificationCompat.Builder(this);
mBuilder.setContentTitle("Picture Download")
    .setContentText("Download in progress")
    .setSmallIcon(R.drawable.ic_notification);
// Start a lengthy operation in a background thread
new Thread(
    new Runnable() {
        @Override
        public void run() {
            int incr;
            // Do the "lengthy" operation 20 times
            for (incr = 0; incr <= 100; incr+=5) {
                    // Sets the progress indicator to a max value, the
                    // current completion percentage, and "determinate"
                    // state
                    mBuilder.setProgress(100, incr, false);
                    // Displays the progress bar for the first time.
                    mNotifyManager.notify(0, mBuilder.build());
                        // Sleeps the thread, simulating an operation
                        // that takes time
                        try {
                            // Sleep for 5 seconds
                            Thread.sleep(5*1000);
                        } catch (InterruptedException e) {
                            Log.d(TAG, "sleep failure");
                        }
            }
            // When the loop is finished, updates the notification
            mBuilder.setContentText("Download complete")
            // Removes the progress bar
                    .setProgress(0,0,false);
            mNotifyManager.notify(ID, mBuilder.build());
        }
    }
// Starts the thread by calling the run() method in its Runnable
).start();
```

The resulting notifications are shown in figure 1. On the left side is a snapshot of the notification during the operation; on the right side is a snapshot of it after the operation has finished.

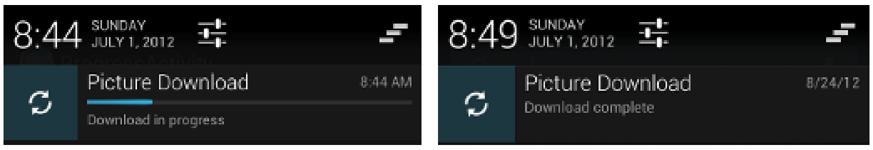

Figure 1. The progress bar during and after the operation.

Display a Continuing Activity Indicator

To display a continuing (indeterminate) activity indicator, add it to your notification with `setProgress(0, 0, true)` (/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean)) and issue the notification. The first two arguments are ignored, and the third argument declares that the indicator is indeterminate. The result is an indicator that has the same style as a progress bar, except that its animation is ongoing.

Issue the notification at the beginning of the operation. The animation will run until you modify your notification. When the operation is done, call `setProgress(0, 0, false)` (/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean)) and then update the notification to remove the activity indicator. Always do this; otherwise, the animation will run even when the operation is complete. Also remember to change the notification text to indicate that the operation is complete.

To see how continuing activity indicators work, refer to the preceding snippet. Locate the following lines:

```
// Sets the progress indicator to a max value, the current completion
// percentage, and "determinate" state
mBuilder.setProgress(100, incr, false);
// Issues the notification
mNotifyManager.notify(0, mBuilder.build());
```

Replace the lines you've found with the following lines. Notice that the third parameter in the `setProgress()` (/reference/android/support/v4/app/NotificationCompat.Builder.html#setProgress(int, int, boolean)) call is set to `true` to indicate that the progress bar is indeterminate:

```
 // Sets an activity indicator for an operation of indeterminate length
```

```
mBuilder.setProgress(0, 0, true);
// Issues the notification
mNotifyManager.notify(0, mBuilder.build());
```

The resulting indicator is shown in figure 2:

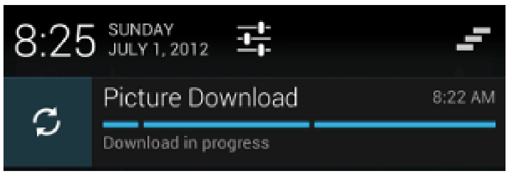

Figure 2. An ongoing activity indicator.