Building a Notification

This lesson explains how to create and issue a notification.

The examples in this class are based on the `NotificationCompat.Builder` (/reference/android/support/v4/app/NotificationCompat.Builder.html) class. `NotificationCompat.Builder` (/reference/android/support/v4/app/NotificationCompat.Builder.html) is in the Support Library (/). You should use `NotificationCompat` (/reference/android/support/v4/app/NotificationCompat.html) and its subclasses, particularly `NotificationCompat.Builder` (/reference/android/support/v4/app/NotificationCompat.Builder.html), to provide the best notification support for a wide range of platforms.

THIS LESSON TEACHES YOU TO

1. Create a Notification Builder
2. Define the Notification's Action
3. Set the Notification's Click Behavior
4. Issue the Notification

YOU SHOULD ALSO READ

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

Create a Notification Builder

When creating a notification, specify the UI content and actions with a `NotificationCompat.Builder` (/reference/android/support/v4/app/NotificationCompat.Builder.html) object. At bare minimum, a `Builder` (/reference/android/support/v4/app/NotificationCompat.Builder.html) object must include the following:

- A small icon, set by `setSmallIcon()`
- A title, set by `setContentTitle()`
- Detail text, set by `setContentText()`

For example:

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

Define the Notification's Action

Although actions are optional, you should add at least one action to your notification. An action takes users directly from the notification to an `Activity` (/reference/android/app/Activity.html) in your application, where they can look at the event that caused the notification or do further work. Inside a notification, the action itself is defined by a `PendingIntent` (/reference/android/app/PendingIntent.html) containing an `Intent` (/reference/android/content/Intent.html) that starts an `Activity` (/reference/android/app/Activity.html) in your application.

How you construct the `PendingIntent` (/reference/android/app/PendingIntent.html) depends on what type of `Activity` (/reference/android/app/Activity.html) you're starting. When you start an `Activity` (/reference/android/app/Activity.html) from a notification, you must preserve the user's expected navigation

experience. In the snippet below, clicking the notification opens a new activity that effectively extends the behavior of the notification. In this case there is no need to create an artificial back stack (see Preserving Navigation when Starting an Activity (navigation.html) for more information):

```
Intent resultIntent = new Intent(this, ResultActivity.class);
...
// Because clicking the notification opens a new ("special") activity, there's
// no need to create an artificial back stack.
PendingIntent resultPendingIntent =
    PendingIntent.getActivity(
    this,
    0,
    resultIntent,
    PendingIntent.FLAG_UPDATE_CURRENT
);
```

Set the Notification's Click Behavior

To associate the `PendingIntent` (/reference/android/app/PendingIntent.html) created in the previous step with a gesture, call the appropriate method of `NotificationCompat.Builder` (/reference/android/support/v4/app/NotificationCompat.Builder.html). For example, to start an activity when the user clicks the notification text in the notification drawer, add the `PendingIntent` (/reference/android/app/PendingIntent.html) by calling `setContentIntent()` (/reference/android/support/v4/app/NotificationCompat.Builder.html#setContentIntent(android.app.PendingIntent)). For example:

```
PendingIntent resultPendingIntent;
...
mBuilder.setContentIntent(resultPendingIntent);
```

Issue the Notification

To issue the notification:

- Get an instance of `NotificationManager`.
- Use the `notify()` method to issue the notification. When you call `notify()`, specify a notification ID. You can use this ID to update the notification later on. This is described in more detail in Managing Notifications.
- Call `build()`, which returns a `Notification` object containing your specifications.

For example:

```
NotificationCompat.Builder mBuilder;
...
// Sets an ID for the notification
int mNotificationId = 001;
// Gets an instance of the NotificationManager service
NotificationManager mNotifyMgr =
        (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
// Builds the notification and issues it.
mNotifyMgr.notify(mNotificationId, mBuilder.build());
```