Les Notifications

**Simon Cimetière (FA)  
22/11/2018**

# Résumé

Présentez ici en 10 lignes maxi de quoi parle le TP, et ce qu’on apprend à y faire. Par exemple :

Avec la démocratisation d’applications en tout genre, on ne prête même plus attention a toute ces rappels automatiques : Les Notifications.

*Les notifications sont utilisées généralement pour informer l’utilisateur qu’il y a quelque chose d’important sur l’application qui la lance.*

*Dans ce TP, nous verrons de quoi est composée une notification et nous verrons comment construire les différentes parties d’une notification.*

# Pré-requis

* *Aucun pré requis pour ce TP.*

# Code source

Code source **initial** : <https://github.com/Alakathion/TP_notification>

Code source **final** : https://github.com/Alakathion/TP\_notification\_fini

# Explications du TP

## Etape 1: créer le déclencheur de la notification

La première étape pour créer une notification est de savoir comment vous souhaitez la déclencher.  
Pour ce TP, nous allons juste créer 2 boutons : l’un pour ajouter une notification et l’autre pour la supprimer.

Dans un premier temps, modifiez la fonction onCreate() afin de recuperer les 2 boutons presents sur la page. Puis definissez un Listener sur le clic de ces boutons. Pour commencer, vous pouvez juste afficher un Toaster afin de voir si vos boutons sont bien instancié.

*Toast.makeText(getBaseContext(), "Ajout d'une notification", Toast.LENGTH\_SHORT).show();*

## Etape 2: créer une Notification.

Une fois l’étape 1 réalisé, créer une nouvelle méthode createNotification() dans la MainActivity. Cette fonction va permettre de créer une notification et de l’afficher sur le téléphone.

Pour cela, il va d’abord instancier un objet de Type NotificationManager. C’est cet objet qui va permettre « d’envoyer » la notification sur le téléphone. Pour cela, il faut récupérer le service de notification du téléphone comme suis :

getSystemService(*NOTIFICATION\_SERVICE*);

Pour la suite, il va vous falloir une Intent, ainsi qu’une PendingIntent . pour cela créer une Intent avec cette instance de mainActivity comme packageContext et MainActivity.class comme second argument.

Avect cette Intent, créer un PendingIntent comme suis :

final PendingIntent pendingIntent = PendingIntent.*getActivity*(this,  
 1, launchNotifiactionIntent,  
 PendingIntent.*FLAG\_ONE\_SHOT*);

Maintenant, nous allons pouvoir créer le Builder de la classe Notification. Pour cela, il suffit de créer un nouvel objet Notification.Builder et de lui fournir notre context de MainActivity.

Maintenant il ne reste plus qu’as rajouter toute les information que nous voulons afficher dans notre notification.

Dans un premier temps nous allons rajouter deux choses :

setWhen(System.*currentTimeMillis*())  
setTicker("premiere notification")

Ces 2 méthodes sont très importantes pour la notification. La première permet de définir Quand la notification va se lancer. La seconde est une information qui est utile pour l’accessibilité de la notification.

Pour la suite, a vous de voir. Il existe differentes methodes très explicite. Par exemple *setSmallIcon, setContentTitle, setContentText.*

Une fois votre notification prète, pensez a rajouter l’Intent que nous avons definis precedement, comme suis :

setContentIntent(pendingIntent)

Avant de terminez votre fonction, vous pouvez donc votre notificationManager et appeler la méthode notify dessus en renseignant un ID a votre notification et votre builder.build().

Pour pouvoir le tester sur votre téléphone, vous n’avez plus qu’as appelé la méthode createNotification dans le Listener du bouton associer.

## Etape 2: Supprimer une Notification.

Dans certains cas vous pourriez avoir besoin de supprimer une notification envoyé directement a l’intérieur de votre application.

Créer donc une nouvelle méthode deleteNotification().

Pour pouvoir supprimer une notification, il vous faut d’abord récupérer votre notificationManager. Parton du principe que vous ne l’avez pas garder a porté de main.

Pour le récupérer il va donc falloir faire comme dans le createNotification, a la différence que vous ne voulez plus juste récupérer le NOTIFICATION\_SERVICE, mais celui de votre contexte précisément. Pour cela, il suffit de le récupérer comme suis :

getBaseContext().*NOTIFICATION\_SERVICE*

Une fois le Manager récuper, il suffit d’appeler la méthode cancel avec l’ID de votre notification.

## Etape 3: Pour aller plus loin sur les Notifications.

Il existe de nombreuses méthodes utilisables sur les notifications. Pour finir, Nous allons en voir 2 autres.

La première est le set Style. Elle permet énormément de chose et notamment de gérer les notifications étendues. Pour cela il vous devez appeler cette méthode sur le builder. Cette methode prend en paramètre le contenue de votre notification étendue. Par exemple, vous pouvez lui fournir du texte visible comme ceci :

CharSequence cs = "Bonjour ceci \n est ma premiere notification etendu car je veux tester le rendu de celle-ci » ;

build.setStyle(new Notification.BigTextStyle().bigText(cs))

Notez que bigText prend en paramètre un objet CharSequence et non un String.

La deuxième deuxième, et dernière méthode que nous verrons ici est l’ajout d’un bouton d’action au sein même de la notification.

Pour cela, il faut utiliser la méthode addAction. Celle-ci prend 3 arguments :

* Une icône.
* Un titre.
* Une action.

Pour les deux premier, aucun problème. Pour l’action a exécuter, vous pouvez par exemple demandez l’ouverture de votre application. Pour cela il faudra récupérer le contexte de l’application et faire une activité. Voici la définition d’une activité de base ouvrant l’application qui a envoyer la notification.

PendingIntent.*getActivity*(getApplicationContext(), 0, getIntent(), 0, null)

# Informations complémentaires

* https://developer.android.com/reference/android/app/Notification