TP : Notifications

Introduction

Qu'est-ce qu'une notification ?

Une notification est un message que l'on peut afficher à l'utilisateur en dehors de notre application. Quand le système reçoit une notification, elle s'affiche d'abord comme une icône dans la barre qui se situe en haut d'un téléphone Android. On peut voir les détails de cette notification en déployant la liste des notifications.

A quoi ça sert ?

Les notifications sont utilisées pour nous informer, à travers un court message, de certains événements tels que l'arrivée d'un nouvel e-mail, d'un SMS, d'une nouvelle actualité, d'une mise à jour... Elles peuvent aussi nous permettre de produire de multiples actions comme nous faire démarrer une application, répondre à un message ou même gérer une application de musique.

Objectif du TP

Le but de ce TP est d'apprendre à créer des notifications, on va réaliser une application simple avec un bouton qui va envoyer une notification au système. Nous allons également voir une autre manière d'afficher notre notification.

Le TP va se faire à partir du code initial lors de la création d'un projet sous Android Studio.
Pour obtenir la version finale de ce TP :
https://drive.google.com/file/d/0Byle-MBcX9LWTlpvc3YwVklsVXc/view?usp=sharing

Déroulement du TP

Commençons par créer un projet Android Studio de cette façon :
"Start a new Android Studio project" -> "Minimum SDK API 9: Android 2.3" -> "Empty Activity"

Je vais vous présenter la manière la plus simple de faire une notification, étape par étape.

Création de la notification

Etape 1.1 : Création d'un objet "NotificationCompat.Builder"

Il s'agit d'une seule ligne pour avoir un objet qui va nous permettre de concevoir une notification.

```
NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this);
```

Pensez bien à importer la classe NotificationCompat (et les autres classes à venir).
Android Studio vous le rappelera de toute façon.

Etape 1.2 : Configuration des propriétés de notre notification

A l'aide de l'objet que nous avons créé précedemment, on va pouvoir spécifier les propriétés nécessaires à notre notification, il y a 3 propriétés indispensables :
- Une icône, spécifiée avec setSmallIcon()
- Un titre, spécifié avec setContentTitle()
- Un texte, spécifié avec setContentText()

Il y a plein d'autres propriétés possibles. Je vous invite à aller voir ce lien pour plus de détails.

Voici le code indiquant les 3 propriétés dont nous avons besoin :

```
mBuilder.setSmallIcon(R.drawable.notification_icon);
mBuilder.setContentTitle("Titre de ma notification");
mBuilder.setContentText("Texte de ma notification");
```

Si à ce stade, Android Studio ne connaît pas la ressource "notification_icon", c'est normal car elle n'existe pas encore. Il va juste falloir ajouter une image qui va représenter l'icône de notre notification dans notre projet.

Je vous invite donc à télécharger une icône de base sur ce lien.
Il faudra mettre cette image dans le dossier drawable, pour trouver ce dossier, à partir du répertoire de nos projets Android Studio, il faut suivre ce chemin :
<NomDeVotreApplication>/app/src/main/res/drawable

Etape 1.3 : Envoi de la notification au système

Afin de délivrer la notification, il faut d'abord créer un objet NotificationManager récupérant le service de notification. Ensuite, il suffira d'appéler une méthode notify() pour envoyer notre notification. Voici le code pour se faire :

```
int notificationID = 1;
NotificationManager mNotificationManager =
        (NotificationManager) getSystemService(Context.NOTIFICATION_SERVICE);
mNotificationManager.notify(notificationID, mBuilder.build());
```

La variable notificationID permet d'attacher un ID à notre notification afin de modifier celle-ci plus tard.

Uniquement avec ces quelques lignes de code, on a une simple notification.
On va maintenant tester cela en créant un bouton qui va envoyer la notification que l'on vient de concevoir.

Mise en pratique

Etape 2.1 : Ajout du bouton

Nous allons faire simple, on va sur notre vue activity_main.xml et on ajoute un bouton avec comme ID 'button_notification', voici le code si nécessaire :

```
<Button
    android:text="Envoyer une notification"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/button_notification"
    android:layout_below="@+id/textView"
    android:layout_alignParentLeft="true"
    android:layout_alignParentStart="true" />
```

Il est conseillé de mettre le texte du bouton dans la fichier strings.xml mais il s'agit ici d'une petite application de tutoriel.

On va maintenant sur le fichier MainActivity.java et dans la méthode onCreate(), on initialise notre bouton et on lui met un écouteur d'évenement onClick() :

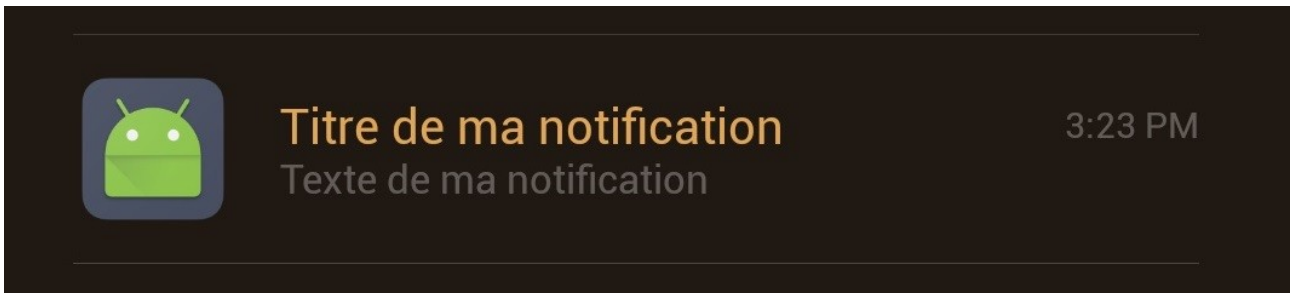
```
final Button button = (Button) findViewById(R.id.button_notification);
button.setOnClickListener(new View.OnClickListener() {
    public void onClick(View v) {

    }
});
```

Etape 2.2 : Envoi de la notification lorsqu'on appuie sur notre bouton

Il s'agit simplement de mettre notre invocation de méthode notify() dans la méthode onClick que l'on vient d'écrire. Il vous sera sûrement demandé de mettre certaines variables (mBuilder, notificationID et mNotificationManager) en "final" afin qu'elles soient accessibles à partir d'une classe interne comme celle de l'écouteur de notre bouton.

Dernière étape : Test de l'application

Il ne reste plus qu'à essayer de lancer notre application et d'appuyer sur le bouton pour voir si cela nous envoie bien une notification. Cela devrait ressembler à ça :



Un autre affichage pour notre notification

Actuellement, la notification que l'on vient de faire nous permet uniquement de communiquer une petite information. Nous allons modifier l'affichage de cette notification afin, par exemple, de l'agrandir pour y ajouter plus d'informations qu'une ligne de texte.

Etape 3.1 : Création d'un objet "NotificationCompat.InboxStyle"

La création de cet objet représente un "expanded layout" qui sera ajouté à notre notification.

```
NotificationCompat.InboxStyle inboxStyle =
        new NotificationCompat.InboxStyle();
```

Etape 3.2 : Configuration de l'"expanded layout"

On va pouvoir réaliser trois choses avec l'objet "NotificationCompat.InboxStyle", on peut lui spécifier un titre, un texte et différentes lignes. Voici les différentes méthodes à utiliser :

- addLine() pour l'ajout de lignes
- setBigContentTitle() pour spécifier le titre
- setSummaryText() pour spécifier le texte

Pour avoir un peu plus de détails, voir ce lien.
On va donc écrire quelque chose comme ci-dessous :

```
inboxStyle.setBigContentTitle("Titre de l'expanded layout");
inboxStyle.setSummaryText("Texte de l'expanded layout");
inboxStyle.addLine("Ligne 1");
inboxStyle.addLine("Ligne 2");
inboxStyle.addLine("Ligne 3");
inboxStyle.addLine("Ligne 4");
```

Etape 3.3 : Ajout de l'"expanded layout" à la notification

Maintenant, la dernière chose à faire est d'ajouter l'objet que l'on vient de configurer à la notification, il faut utiliser la méthode setStyle() sur notre objet "NotificationCompat.Builder".

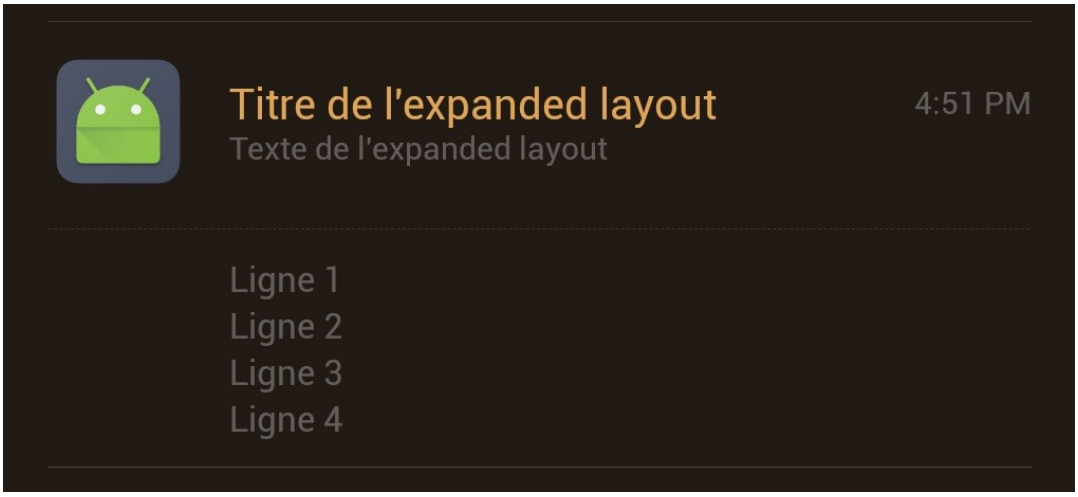
```
mBuilder.setStyle(inboxStyle);
```

Dernière étape : Test de l'application

On peut maintenant essayer cela en lançant à nouveau notre application et en appuyant sur le bouton pour voir notre nouvelle notification.

Il peut y avoir des particularités pour visualiser ces nouvelles informations.
Par exemple, pour les téléphones utilisant MIUI, il faut glisser la notification avec deux doigts vers le bas ou les écarter vers l'extérieur.

Voici ce que cela donne sur l'interface MIUI :



Fin du TP

C'est tout pour ce TP qui, je l'espère, vous a permis de comprendre comment créer une simple notification et de voir une autre façon de l'afficher. Cependant, il existe bien d'autres choses à faire avec les notifications mais nous n'avons pas le temps de les aborder ici. Je vous invite alors à visiter ces liens pour en apprendre davantage sur les notifications :

https://developer.android.com/guide/topics/ui/notifiers/notifications.html
https://material.google.com/patterns/notifications.html