UNIVERSITÉ CÔTE D'AZUR

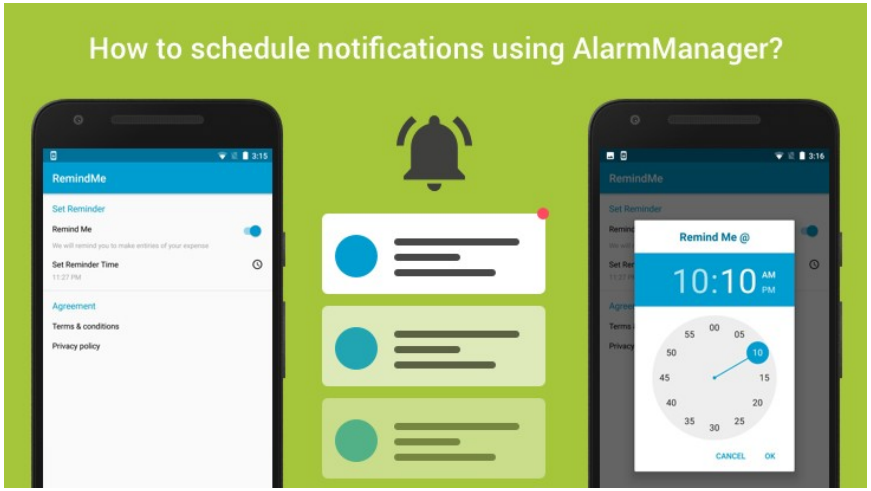
IMPLÉMENTER DES NOTIFICATIONS SUR ANDROID

frederic.rallo@univ-cotedazur.fr

RAPPEL SUR LES NOTIFICATIONS

Une notification est un message qu'Android affiche en dehors de l'UI

Ils servent à transmettre à l'utilisateur des rappels, des messages transmis par d'autres utilisateurs ou d'autres informations en temps réel provenant de votre application.



L'utilisateur peut alors appuyer sur la notification pour ouvrir votre application ou effectuer une action directement depuis la notification.

DES TÂCHES ASYNCHRONES

- Les notifications font partie de la famille des tâches asynchrones
- On en a déjà utilisé avec les listeners sur les boutons !
- Étudions ce code :

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    String title = ((EditText)findViewById( R.id.edit_text_title)).getText().toString();
    String message = ((EditText)findViewById( R.id.edit_text_message)).getText().toString();
    findViewById( R.id.buttonNotify1).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           sendNotificationOnChannel( title, message, CHANNEL_1_ID, NotificationCompat.PRIORITY_LOW
);
        }
    });
}
```

- Le compilateur n’en veut pas, voyez-vous pourquoi ?

DES TÂCHES ASYNCHRONES

- **Le compilateur exige que l'on place le mot final devant la déclaration des String title et message… pourquoi ?**

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    final String title = ((EditText)findViewById( R.id.edit_text_title)).getText().toString();
    final String message = ((EditText)findViewById( R.id.edit_text_message)).getText().toString();
    findViewById( R.id.buttonNotify1).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           sendNotificationOnChannel( title, message, CHANNEL_1_ID, NotificationCompat.PRIORITY_LOW
);
        }
    });
}
```

- **Nous sommes ici dans la méthode onCreate(). Cela signifie que l'on explique ce que l'on souhaite obtenir au moment de la création.**
- **On souhaite connaitre la valeur des String title et message**
- **On explique ce qu'il se passera lorsqu'on appuiera sur le bouton.**

DES TÂCHES ASYNCHRONES

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    final String title = ((EditText)findViewById( R.id.edit_text_title)).getText().toString();
    final String message = ((EditText)findViewById( R.id.edit_text_message)).getText().toString();
    findViewById( R.id.buttonNotify1).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           sendNotificationOnChannel( title, message, CHANNEL_1_ID, NotificationCompat.PRIORITY_LOW
);
        }
    });
}
```

- **En mettant le mot clé final on dit que l'on veut faire référence à la valeur title et message qui existait au moment de la création de l'activité**
- **Le mot final est là pour nous rappeler que la valeur qu'on passera en paramètre n'a pas été changée**

DES TÂCHES ASYNCHRONES

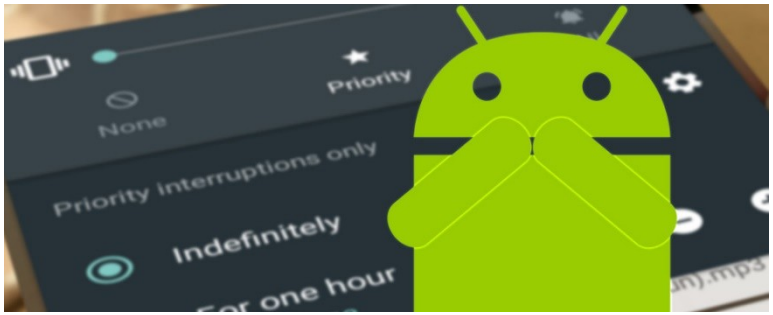
- **Le code suivant est très différent**

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    findViewById( R.id.buttonNotify1).setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
           String title = ((EditText)findViewById( R.id.edit_text_title)).getText().toString();
           String message = ((EditText)findViewById( R.id.edit_text_message)).getText().toString();
           sendNotificationOnChannel( title, message, CHANNEL_1_ID, NotificationCompat.PRIORITY_LOW
);
        }
    });
}
```

- **On souhaite connaitre la valeur des String title et message après avoir cliqué !**
- **On écrit une méthode dans le onCreate() qui ne sera exécutée que plus tard (peut-être jamais)**
- **C'est une méthode asynchrone !**

DES TÂCHES ASYNCHRONES

- **Pour réaliser une notification, c'est le même principe :**
- **On prépare un channel pour lui permettre de partir**
- **On prépare la notification**
- **On l'expédie dans le channel**



- **…**
- **Le système se charge de son « voyage »**
  - Elle peut voyager avec une priorité LOW, DEFAULT ou HIGH
  - Les channels eux aussi disposent de priorités LOW, DEFAULT et HIGH

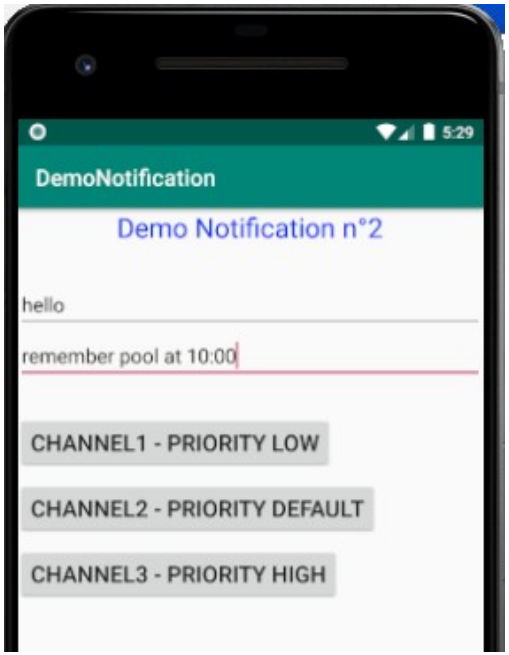
NOTIFICATION CHANNEL

- **Un canal de notification permet d'avoir des paramètres qui s'appliquent à de multiples notifications ayant un thème similaire.**
- **depuis Android 8.0 (API 26), il faut créer un canal de notification.**
- **Créer le canal de notification avant de publier des notifications**
- **exécuter ce code dès le démarrage de votre application.**
  - Dans le tutoriel, on expliquera comment faire !

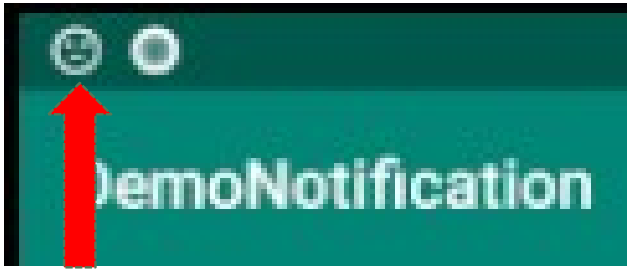
NOTIFICATION CHANNEL

On souhaite afficher un titre et une description en notification

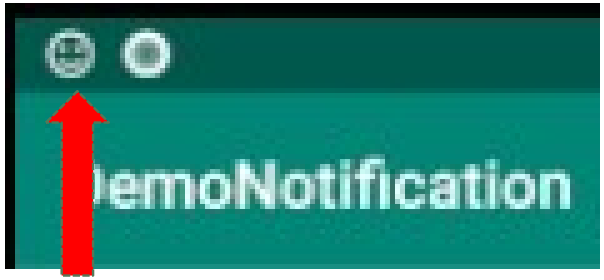
On peut choisir de faire transiter ces informations sur des channels différents. C'est une façon d'organiser les notifications (options à deux étoiles)



Résultat lorsqu'on utilise le channel 1



Résultat lorsqu'on utilise le channel 2

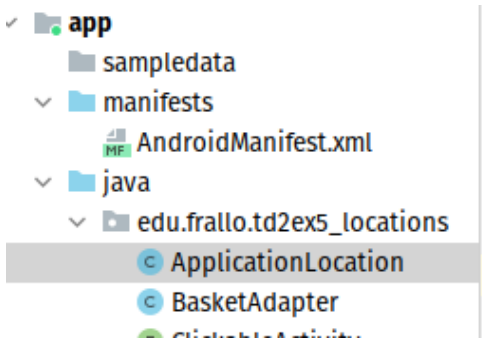


Résultat lorsqu'on utilise le channel 3

CRÉATION D'UNE NOTIFICATION

Créer les canaux de communication

- **Dans une classe qui étend Application**




```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools">
4
5      <uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
6      <application
7          android:name="edu.frallo.td2ex5_locations.ApplicationLocation"
8          android:allowBackup="true"
```

CLASSE APPLICATIONLOCATION EXTENDS APPLICATION

```
public class ApplicationLocation extends Application {
    static final String CHANNEL_1_ID = "channel LOW";
    public static final String CHANNEL_2_ID = "channel DEFAULT";
    public static final String CHANNEL_3_ID = "channel HIGH";
    private static NotificationManager notificationManager;


    @Override
    public void onCreate() {
       super.onCreate();
       createNotificationChannels();
    }


     private void createNotificationChannels() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {  // Créer le NotificationChannel, seulement pour API 26+
           NotificationChannel channel1 = createNotificationChannel(        CHANNEL_1_ID,"Channel 1",
                                                                            NotificationManager.IMPORTANCE_LOW,
                                                                            "This Channel has low priority");

           NotificationChannel channel2 = createNotificationChannel(        CHANNEL_2_ID,"Channel 2",
                                                                            NotificationManager.IMPORTANCE_DEFAULT,
                                                                            "This Channel has default priority");
           NotificationChannel channel3 = createNotificationChannel(        CHANNEL_3_ID,"Channel 2",
                                                                            NotificationManager.IMPORTANCE_HIGH,
                                                                            "This Channel has high priority");

           // Enregister le canal sur le système : attention de ne plus rien modifier après
           NotificationManager manager = getSystemService(NotificationManager.class);
           Objects.requireNonNull(manager).createNotificationChannel(channel1);
           Objects.requireNonNull(manager).createNotificationChannel(channel2);
           Objects.requireNonNull(manager).createNotificationChannel(channel3);
       }
     }
```

CLASSE APPLICATIONLOCATION EXTENDS APPLICATION

```
public class ApplicationLocation extends Application {
    static final String CHANNEL_1_ID = "channel LOW";
    public static final String CHANNEL_2_ID = "channel DEFAULT";
    public static final String CHANNEL_3_ID = "channel HIGH";
    private static NotificationManager notificationManager;


    private NotificationChannel createNotificationChannel(String channelId, CharSequence name, int importance, String
    channelDescription) {
       // Créer le NotificationChannel, seulement pour API 26+
       if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
          NotificationChannel channel = new NotificationChannel(channelId, name, importance);
          channel.setDescription(channelDescription);
          return channel;
       }
       return null;
    }
```

ACTIVITÉ QUI CRÉE UNE NOTIFICATION

```
public class DisplayActivity extends AppCompatActivity {
    private final String TAG = "frallo " + getClass().getSimpleName();

    @Override
    protected void onCreate(Bundle savedInstanceState) {
      //some code
    }



    private void sendNotificationOnChannel(String title, String content, String channelId, int priority) {
        NotificationCompat.Builder notification = new NotificationCompat.Builder(this, channelId)
                .setContentTitle(title)
                .setContentText(content)
                .setPriority(priority)
                .setCategory(NotificationCompat.CATEGORY_MESSAGE);
        switch (channelId) {
            case CHANNEL_1_ID:
                notification.setSmallIcon(R.drawable.channel1);
                break;
            case CHANNEL_2_ID:
                notification.setSmallIcon(R.drawable.channel2);
                break;
            case CHANNEL_3_ID:
                notification.setSmallIcon(R.drawable.channel3);
                break;
        }

        if (ActivityCompat.checkSelfPermission(getApplicationContext()
                                    , android.Manifest.permission.POST_NOTIFICATIONS) != PackageManager.PERMISSION_GRANTED) {
            // TODO: for another course... ;-)
             Log.d(TAG, "permission needed to send notification !");
            return;
        }
        NotificationManagerCompat.from(this).notify(0, notification.build());
    }
```

Pour bien faire, il faudrait demander l'autorisation d'envoyer des notifications

CRÉATION D'UNE NOTIFICATION

Créez d'abord deux attributs :

- **Un String CHANNEL_ID correspondant à l'identifiant du canal**
- **Un entier NOTIFICATION_ID correspondant à l'identifiant de la notification**

```
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    CharSequence name = "Notification channel name";
    int importance = NotificationManager.IMPORTANCE_DEFAULT;
    NotificationChannel channel = new NotificationChannel(CHANNEL_ID, name, importance);
    channel.setDescription("Notification channel description");
    // Enregister le canal sur le système : attention de ne plus rien modifier après
    NotificationManager notificationManager = getSystemService(NotificationManager.class);
    Objects.requireNonNull(notificationManager).createNotificationChannel(channel);
}
```

ENVOYER UNE NOTIFICATION

- On crée une NotificationCompat.Builder
- On lui donne les attributs souhaités (y compris priorité)
- On appelle la méthode statique notify de NotificationManagerCompat

```
private void sendNotificationOnChannel(String title, String message, String channelId, int priority) {
    NotificationCompat.Builder notification =
            new NotificationCompat.Builder( getApplicationContext(), channelId)
                                    .setSmallIcon( R.drawable.channel1 )
                                    .setContentTitle( title )
                                    .setContentText( "id=" + ++notificationId + " - " + message )
                                    .setPriority( priority );
NotificationManagerCompat.from(this).notify( notificationId , notification.build() );
}
```