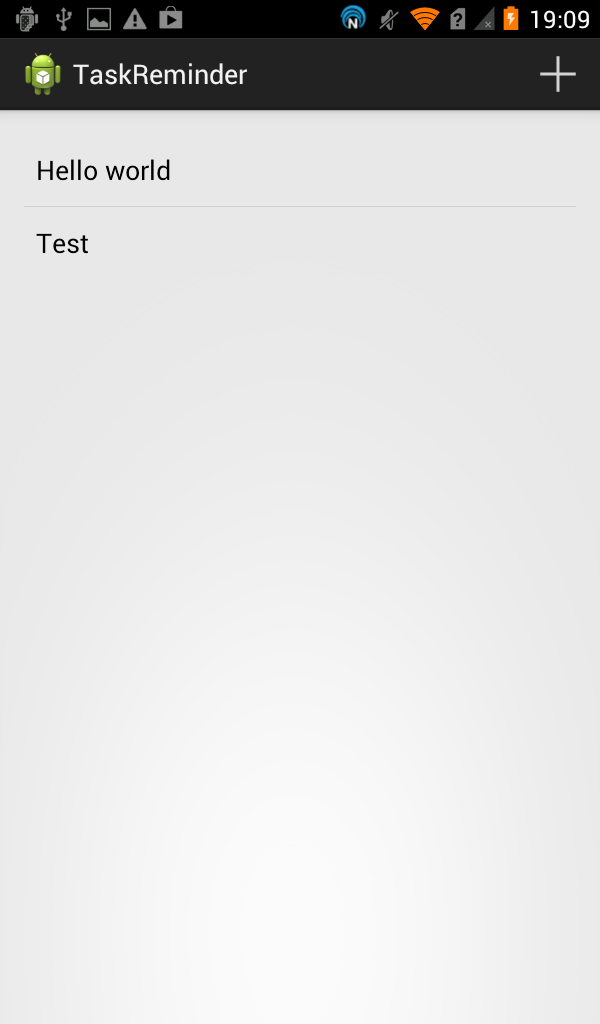
# Mon application : Pense Bête

L’objectif de ce TP est de développer une application permettant de stocker une liste de taches à effectuer. Pensez à télécharger le pack d’icônes mis à disposition par Google : <https://developer.android.com/design/downloads/index.html#action-bar-icon-pack>.

## Explications

Afficher d’arriver au résultat final, vous devez :

* Utiliser l’objet « ListView » permettant de gérer une vue en liste.
* Une classe héritant de BaseAdapter permettant de gérer l’accès aux données (lecture, enregistrement, …).
* Gérer l’évènement appui long pour supprimer un élément
* Ajouter un bouton dans l’ActionBar pour insérer de nouvelles taches.

## Layout : ListView

Permet d’afficher facilement une liste d’élément.

Les données sont gérés au travers d’un Adapter. <http://developer.android.com/guide/topics/ui/layout/listview.html>

### Adapter

Un « Adapter » est un moyen de faire le lien entre les données et l’affichage. <http://developer.android.com/reference/android/widget/Adapter.html>

Généralement, Il faut créer son propre Adapter héritant de BaseAdapter. Cette méthode nécessite d’implémenter quelques méthodes afin d’avoir un fonctionnement total, tels que : « getCount() », « getItem() », « getItemId() » et « getView ». En dehors de « getView », les méthodes se limitent à une seule ligne de code.

« getView() » est un peu plus complexe, car le principe de cette méthode est de peupler la vue correspondant à la ligne passée en paramètre. Il faut ainsi créer une nouvelle vue à partir d’une vue grâce au « LayoutInflater »[[1]](#footnote-1) et la remplir grâce aux données stockées dans notre liste.

Il ne restera plus qu’à lier votre Adapter à la vue :



## Sérialiser des données

Dans ce projet, afin de réduire au maximum les problématiques liées au parsing des données. Nous allons stocker nos données en utilisant le format JSON[[2]](#footnote-2).

L’écriture et la lecture des données s’effectuent à l’intérieur de l’Adapter.

### Ecrire dans un fichier

Le plus simple est d’utiliser un « FileOutputStream » permettant d’écrire une chaine de caractères dans un fichier. Cette chaine de caractères est issue de l’objet « JSONArray » en ayant au préalablement transformé (sérialisé) nos objets « Task » en « JSONObject » (Méthode de l’objet « Task » disponible ci-après).



### JSON



### Lire dans un fichier

Il ne s’agit pas de la méthode la plus optimisé, ni de la plus propre. Cependant elle permet de comprendre facilement le cheminement entre la lecture du fichier et la génération du tableau de taches.

Grâce à l’objet « BufferedReader » nous pouvons récupérer ligne par ligne le contenu du fichier et le stocker dans une « String ». L’objet « JSONArray » peut être construit avec cette chaine de caractère. Un fois construit, il suffit de l’itérer afin de créer de nouveaux objets « Task » et ainsi reconstruire la liste de taches.

Vous noterez que la variable « array » appartient à notre Adapter, et que l’appel de la méthode « notifyDataSetChanged() » permet d’indiquer à la vue que l’affichage doit être rafraichi.



## ActionBar

C:\Users\Steven\Desktop\tmp\actionbar.png

Il s’agit d’une apparue depuis la version 3.0 d’Android. Elle permet de se passer du bouton physique de menu. Le principe d’utilisation reste globalement le même qu’à l’époque des menus classiques :

Il suffit d’ajouter une entrée dans le fichier xml contenu dans le dossier « menu » correspondant à votre vue.



Puis de contrôler les événements dans la méthode « onOptionsItemSelected » de l’activité :



En appelant la méthode « add\_task() » à écrire pour qu’elle permette l’ajout d’une tache (Pensez à utiliser une pop-up : Objet « Dialog »).

## Dialog

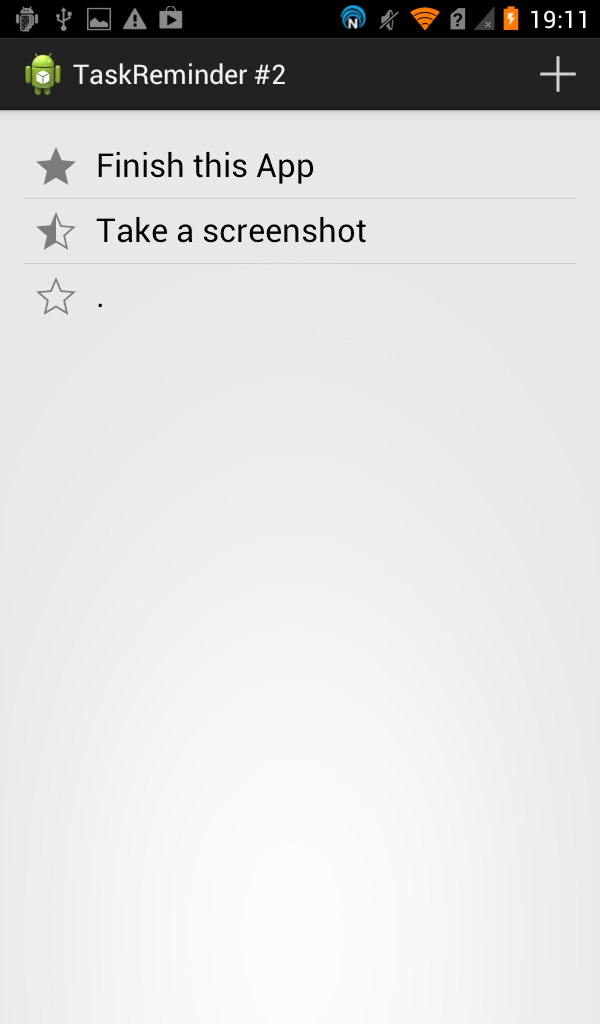
Les « Dialog » sont des petites fenêtres s’occupant de poser une question à l’utilisateur ou bien demander la saisie d’une information.

<https://developer.android.com/guide/topics/ui/dialogs.html>

Grâce à une classe dédiée à la construction de cet objet, il est possible de créer une fenêtre en trois étapes. Voici un exemple correspondant à notre besoin pour l’appui long :



## Aller plus loin : Ajouter une notion de priorité

Dans ce projet, nous avons utilisé un « itemLayout » standard fourni par Android. Pour aller plus loin, il serait intéressant de créer son propre layout permettant l’ajout d’une icône comme la capture d’écran à gauche de paragraphe.

Il sera aussi nécessaire de modifier la fenêtre Dialog d’ajout afin d’insérer un « Spinner » permettant de sélectionner le niveau de priorité.

## Résultat

Le projet « TaskReminder » et la version améliorée « TaskReminder2 » sont disponibles sur le dépôt suivant : <https://github.com/steven-martins/tp-android>

1. http://developer.android.com/reference/android/view/LayoutInflater.html [↑](#footnote-ref-1)
2. <http://en.wikipedia.org/wiki/JSON> [↑](#footnote-ref-2)