



**Institut Universitaire de Technologie**

**Département Informatique**

Site de Bourg-en-Bresse



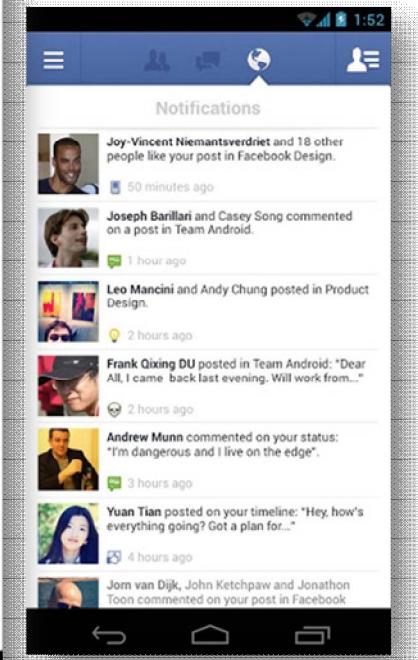
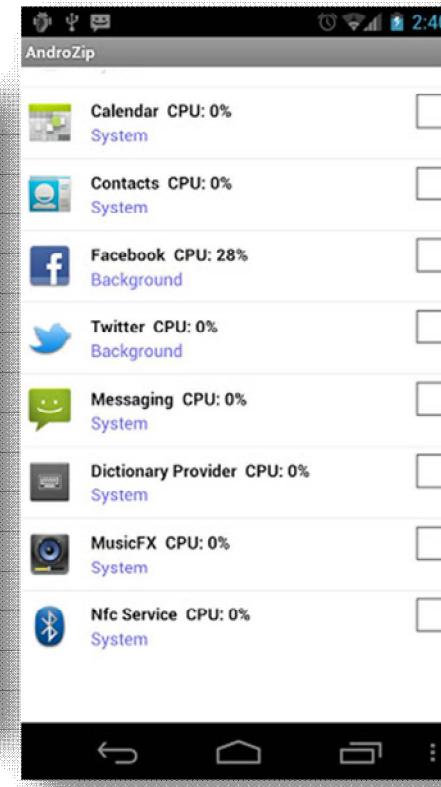
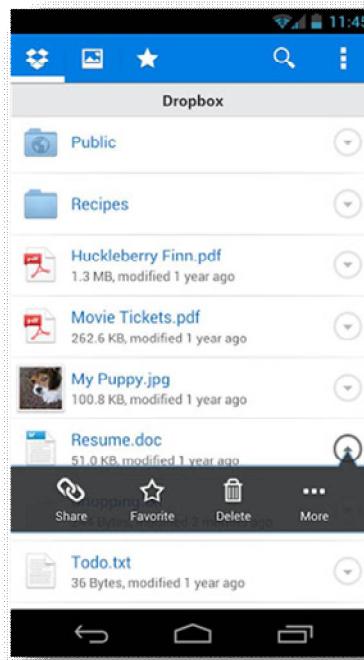
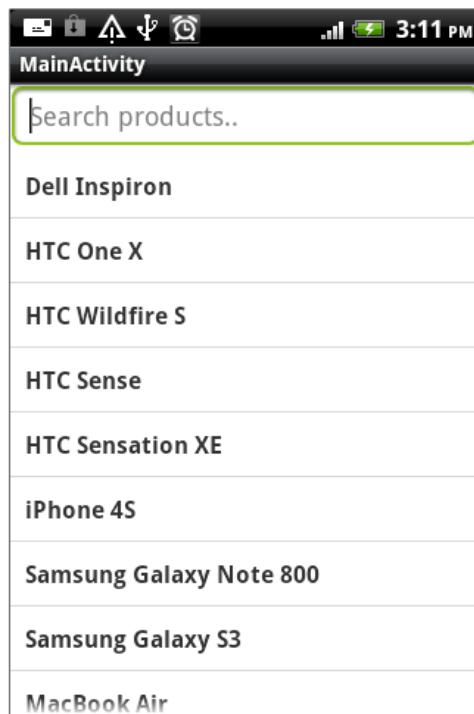
ANDROID

## TP 5 : Les ListView

Semestre 4

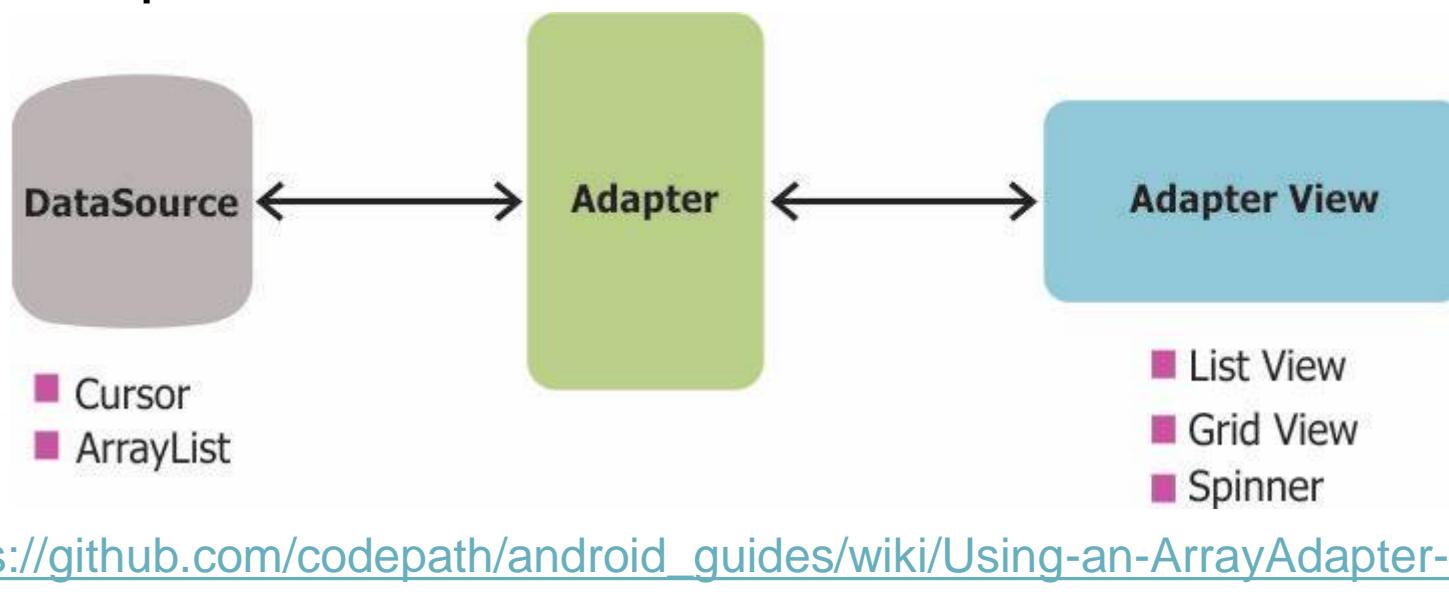
[lionel.buathier@univ-lyon1.fr](mailto:lionel.buathier@univ-lyon1.fr)

# Introduction aux ListView



# 1. Définition d'une ListView

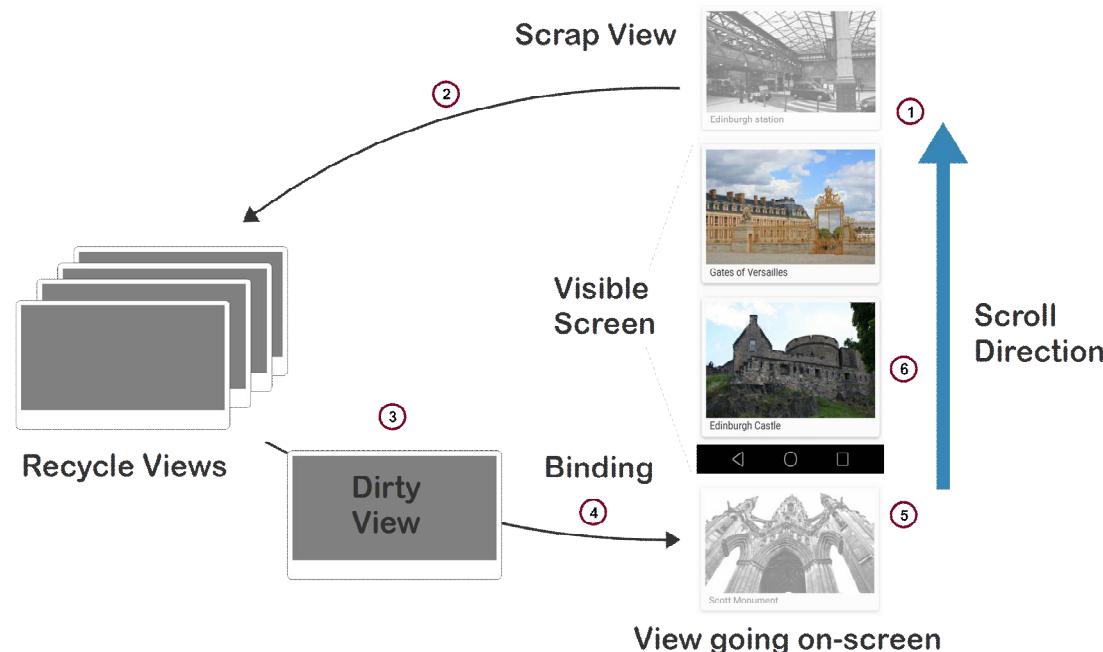
- ▶ Une liste de vues affiche des éléments nommés « items »
- ▶ Chaque item s'affiche selon une stratégie d'affichage définie par un adaptateur
- ▶ L'adapter fait le lien entre les données et l'interface



[https://github.com/codepath/android\\_guides/wiki/Using-an-ArrayAdapter-with-ListView](https://github.com/codepath/android_guides/wiki/Using-an-ArrayAdapter-with-ListView)  49

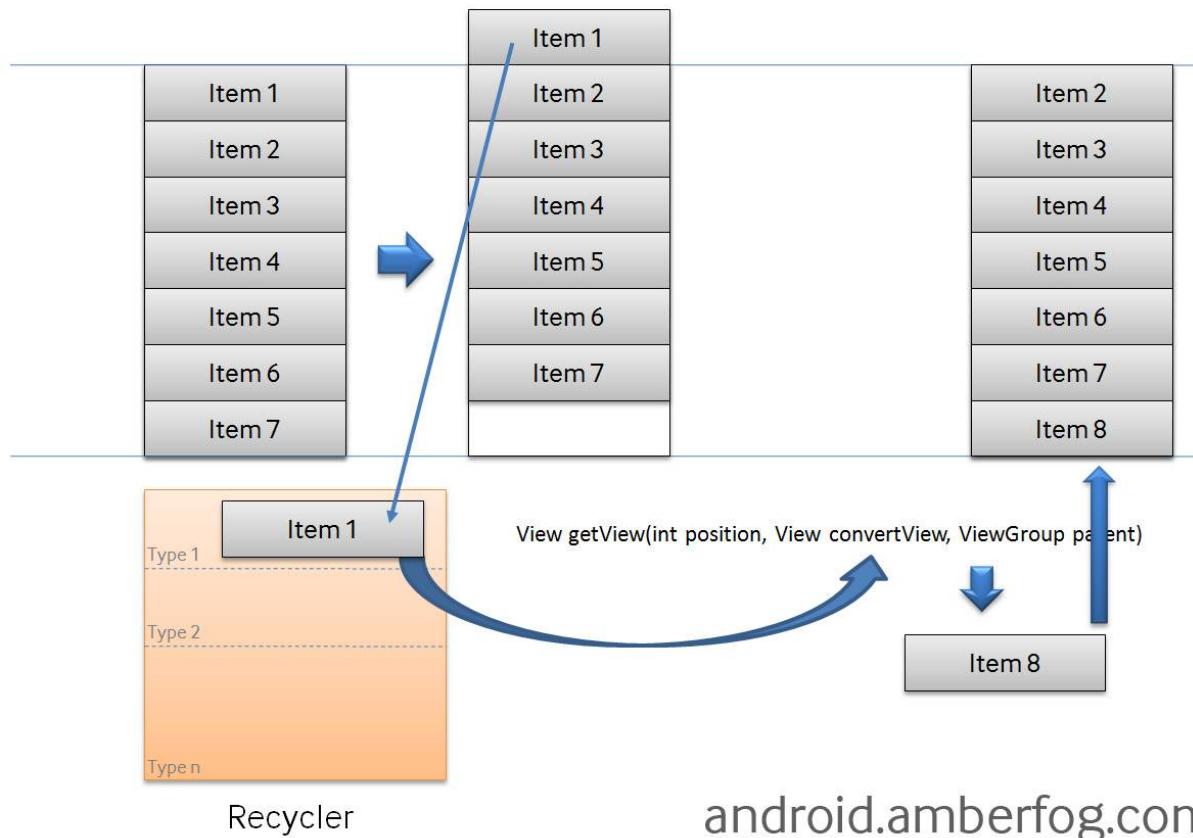
# 1. Recyclage des éléments de la liste

- ▶ Seulement une partie de la liste est affichée à l'écran
- ▶ Lors d'un scroll, les éléments sortants sont mis en cache



[https://github.com/codepath/android\\_guides/wiki/Using-an-ArrayAdapter-with-ListView](https://github.com/codepath/android_guides/wiki/Using-an-ArrayAdapter-with-ListView)  50

# 1. Recyclage des éléments de la liste



[https://github.com/codepath/android\\_guides/wiki/Using-an-ArrayAdapter-with-ListView](https://github.com/codepath/android_guides/wiki/Using-an-ArrayAdapter-with-ListView) 

# 1. Stratégies d'affichage (Adapters)

- ▶ Il existe des stratégies d'affichage (Standard Adapters)
  - Une liste d'items constituée de noms séparés par trait horizontal
    - ⇒ SimpleAdapter, ArrayAdapter
  - Une liste d'items, reflet d'une interrogation dans une base de données
    - ⇒ CursorAdapter
- ▶ On peut créer sa propre stratégie (Custom Adapters)
  - en héritant de BaseAdapter



## 2. Affichage dans une ListActivity

- ▶ On peut Afficher une ListView dans une ListActivity en utilisant :
  - ⇒ une ListView (dont le nom est obligatoirement @android:id/list)
  - ⇒ un ArrayAdapter
  - ⇒ le layout par défaut est : simple\_list\_item\_1
  - ⇒ Assez restrictif et limité, donc à éviter



### 3. Affichage simple dans une Activity

- ▶ On peut également afficher une ListView dans une Activity en utilisant :
  - ⇒ une ListView (dont le nom est libre)
  - ⇒ un ArrayAdapter (qui gère une liste d'objets, mais n'affiche qu'une ligne avec le texte retourné par `toString()`)
  - ⇒ le layout par défaut est : `android.R.layout.simple_list_item_1`
- ▶ Ressources :

<http://developer.android.com/guide/topics/ui/declaring-layout.html#AdapterViews>

1<sup>e</sup> partie de :

<http://tutos-android-france.com/listview-afficher-une-liste-delements/>



## Exercice 1 : affichage d'une liste textuelle

Dans une Activity normale, afficher :

- ▶ une liste d'éléments (textuels) comportant, par exemple, le nom de différents cocktails
- ▶ un toast avec le nom du cocktail, lorsqu'on clique sur un item de la liste

⇒ Utilisation d'une ListView et d'un ArrayAdapter



## 4. Affichage personnalisé avec SimpleAdapter

- ▶ Tout en utilisant les adaptateurs standards, on peut personnaliser l'affichage d'une liste (ajout d'icônes, polices, etc.) pour la rendre plus ergonomique, en créant son propre layout (et éventuellement son adapter - cf. p. suivante)
- ▶ On peut également proposer une nouvelle vue avec un affichage plus détaillé lorsqu'on clique sur un item
- ▶ Ressource :

<http://www.tutomobile.fr/personnaliser-une-listview-tutoriel-android-n%C2%B07/04/07/2010/>

- ▶ **Astuce** : comment recréer un id à partir d'un nom d'icone

```
int id = getApplicationContext().getResources().getIdentifier(nom_icone,"mipmap","nom.du.package");
```



56

## Exercice 2: affichage d'une liste personnalisée

Dans une Activity normale, afficher :

- ▶ une liste personnalisée comportant :
    - Le nom des éléments (ex. équipes d'un championnat sportif)
    - Une icône (le fanion de ces équipes)
  - ▶ un contenu plus détaillé lorsqu'on sélectionne un item  
(ex : nom de l'entraîneur, classement de l'équipe, etc.)
- ⇒ Utilisation d'une ListView et d'un SimpleAdapter



## 5. Données spécifiques => "Custom" Adapter

- ▶ L'adaptateur personnalisé hérite de la classe BaseAdapter.
- ▶ Cette classe doit implémenter obligatoirement trois méthodes :
  - Object **getItem** (int position)
  - long **getItemId** (int position)
  - View **getView** (int position, View convertView, ViewGroup parent) est la plus délicate à utiliser.
    - ⇒ appelée à chaque fois qu'un item est affiché à l'écran.
- ▶ **Astuce** : comment recréer un id à partir d'un nom d'icone dans getView():

```
int id = getApplicationContext().getResources().getIdentifier(nom_icone,"mipmap","nom.du.package");
myImgView.setImageResource(getApplicationContext().getResources().getDrawable(resID));
```



# Exemples de Custom adapters

## ▶ Ressources :

<http://www.codeofaninja.com/2013/02/android-listview-tutorial.html>

[https://github.com/codepath/android\\_guides/wiki/Using-an-ArrayAdapter-with-ListView](https://github.com/codepath/android_guides/wiki/Using-an-ArrayAdapter-with-ListView)

2<sup>e</sup> partie de : <http://tutos-android-france.com/listview-afficher-une-liste-delements/>

<http://tutorielsandroid.com/tuto-android-creer-un-listview-avec-un-adaptateur-personnalise/>



## Exercice 3: personnalisation d'un adaptateur

- ▶ On souhaite maintenant afficher une liste de contacts enregistrés dans une collection d'une classe spécifique.
- ▶ Le formulaire de saisie sera celui utilisé dans le TP précédent. Un bouton permettra d'ajouter un nouveau contact. Lors de la validation du formulaire, ce contact sera ajouté à la liste en cours.
- ▶ Un appui court sur un contact permettra d'en afficher l'aperçu.
- ▶ Un appui long sur un contact permettra de le supprimer.
- ▶ Définir votre propre adaptateur, qui fera le lien entre vos données spécifiques et les éléments d'affichage.



## Exercice 3: sauvegarde des données

- ▶ Pour sauvegarder la liste complète incluant les données ajoutées, on peut l'enregistrer dans le système de fichier propre à l'application (et non accessible aux autres). C'est Android qui gère le chemin d'accès.
- ▶ Le principe est d'enregistrer la liste avec la méthode WriteObject sur un objet ObjectOutputStream :

<http://developer.android.com/reference/java/io/ObjectOutputStream.html>

La contrainte est d'utiliser des objets qui implémentent l'interface "Serializable".

- ▶ Un exemple sur un objet très simple :

<http://www.vogella.com/tutorials/JavaSerialization/article.html>

