Android ListeView, ListeActivity une Introduction

jean-michel Douin, douin au cnam point fr
version : 16 Octobre 2012

Notes de cours

Sommaire

- **Vue**
  - ***ListView***
    - **Présentation**
  - **Adapter la Vue d'une liste d'items aux souhaits du programmeur**
    - **Le patron Stratégie**
  - **Utiliser des stratégies d'affichage standard**
  - **Installer sa propre stratégie**
    - **Accès aux ressources décrites en XML**
      - **Optimiser ces accès (cache)**
  - **Filtrage, tri**

- **Contrôleur**
  - ***ListActivity***
  - ***onListItemClick***
  - ***onItemLongClick***

Technique …

  - **Chargement d'une liste et *AsyncTask***

Bibliographie utilisée

http://developer.android.com/resources/index.html

Le cours de Victor Matos
http://grail.cba.csuohio.edu/~matos/notes/cis-493/Android-Syllabus.pdf

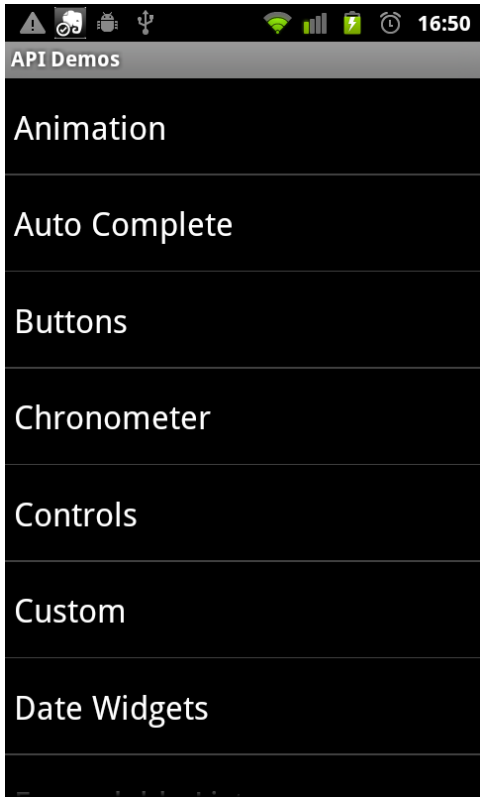
http://www.vogella.com/android.html

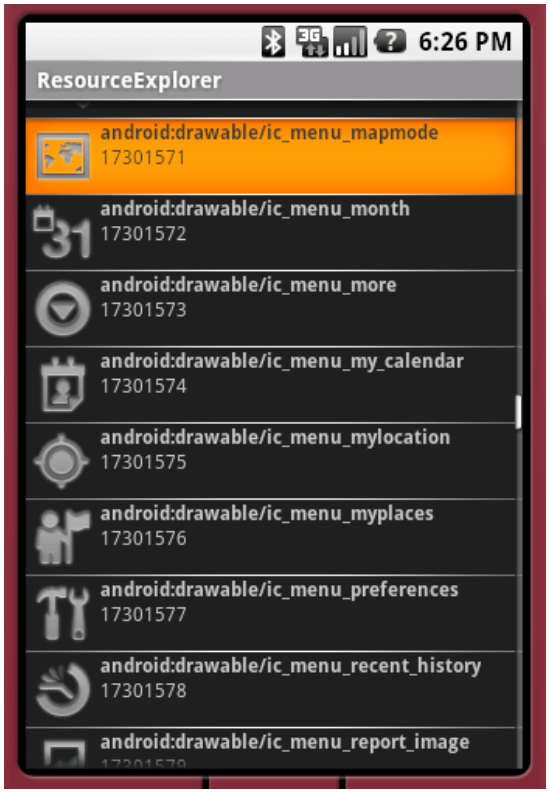
Plusieurs livres
- Android A Programmers Guide - McGraw Hill
- Professional Android Application Development – Wrox
- Le livre de Mark Murphy - Pearson

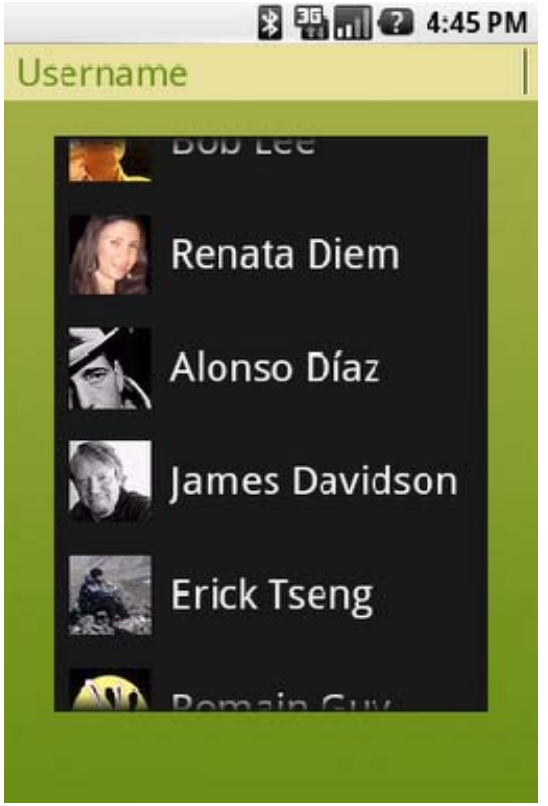
Une vidéo youtube Google IO the world of ListView
http://www.youtube.com/watch?v=wDBM6wVEO70

Présentation d'une liste d'items

- **Un composant graphique commun**
  - **À un grand nombre d'applications**







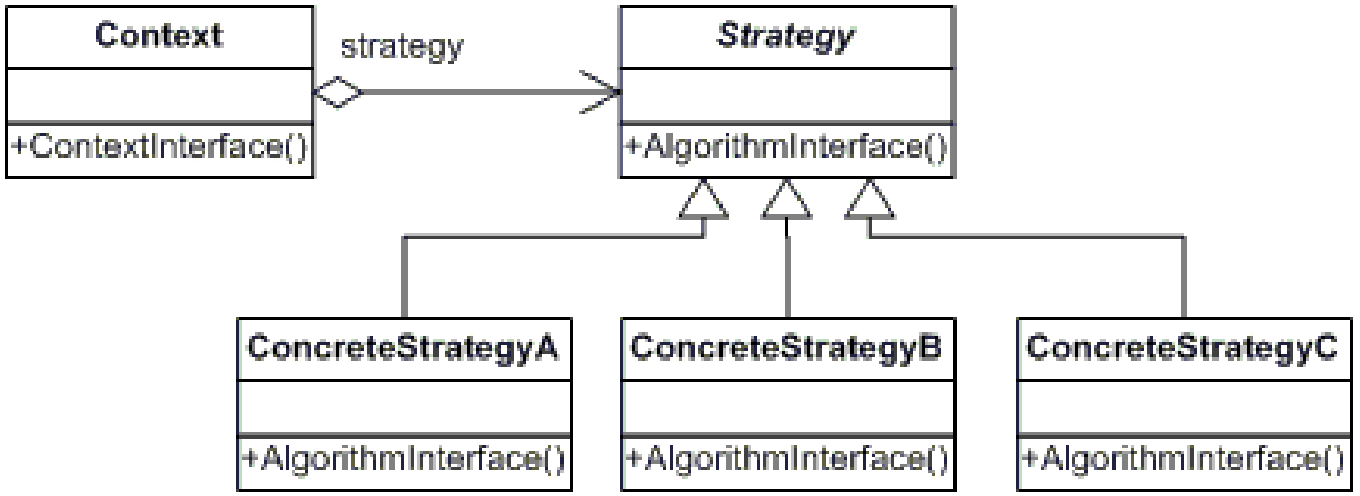
ListView Vue

- **Une liste de vues présentant des items**
- **Chaque item s'affiche selon un format,**
  - **Un adaptateur,**
    - **une stratégie d'affichage est choisie**
- **Il existe des stratégies (adaptateurs) standard**
  - **Une liste d'items constituée de noms séparés d'un trait**
    - **BaseAdapter, SimpleAdapter, ArrayAdapter**
  - **Une liste d'items, reflet d'une interrogation dans une base de données**
    - **CursorAdapter**
  - **Des adaptateurs créés par le programmeur**

Contrôleur, MVC

- **Contrôleur**
  - A chaque clic sur l'un des items une action est déclenchée
    - onListItemClick au sein de l'activity
- **M La liste des objets Java représente le modèle**
  - Tout changement d'état du modèle sera notifié à la vue
- **V La vue de la liste et sa stratégie d'affichage**
  - Stratégie comme le patron strategy

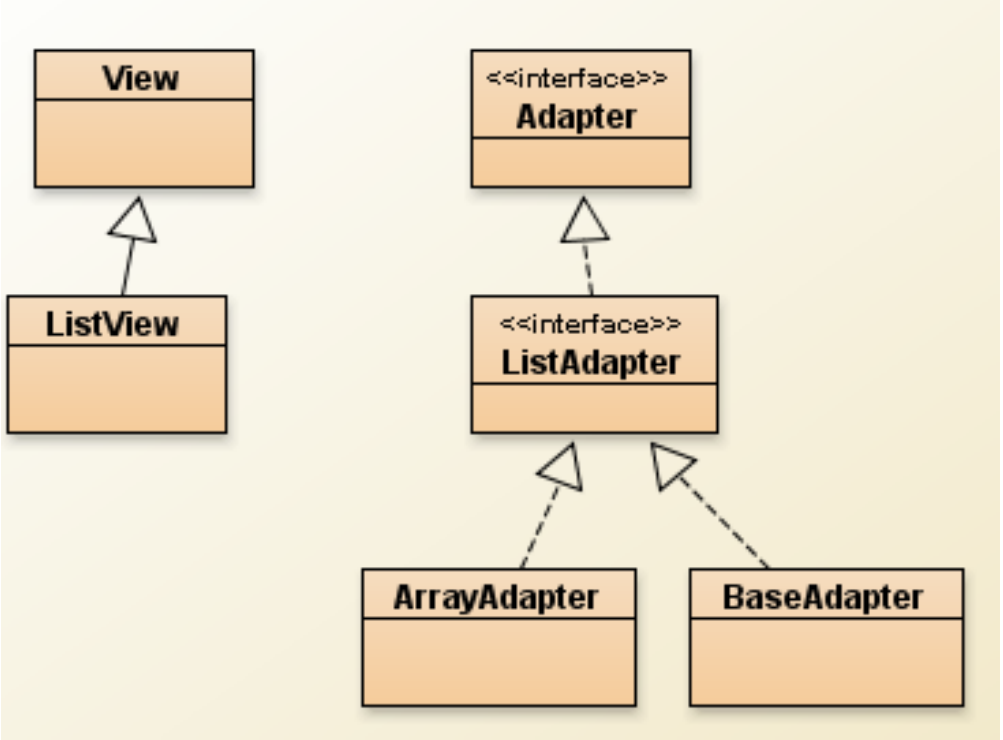
Le patron stratégie, l'original



- **Strategy**
  - **Une interface commune pour tous les affichages**
  - **Android : *ListAdapter***
- **ConcreteStrategy**
  - **Quelle stratégie pour quel rendu ?**
  - **Android : *ArrayAdapter, BaseAdapter, CursorAdapter, MonAdaptateur* …**
- **Context**
  - **Utilisation d'une stratégie choisie par le client**
  - **Android : *ListView***

- http://www.dofactory.com/Patterns/PatternStrategy.aspx

Usage du patron Strategy



- Le patron *Strategy*
- Son usage (ici simplifiée) avec le nom des classes des API Android

Adapter, ListAdapter + ListView

```
public interface Adapter{

   public int getCount(); // le nombre d'items

   public Object getItem(int position);

   public View getView(int position, View convertView);
}
```

```
public class ListView extends View{
   private Adapter adapter;

   public void setAdapter(Adapter adapter){
      this.adapter = adapter;
   }

   public Adapter getAdapter(){
      return adapter;
   }
```

ListView liste = *// déclaration en XML*
liste.setAdapter(new ArrayAdapter….

Un premier exemple et son affichage

- **La ListView en XML**
  - **Le fichier res/layout/liste.xml**
- **Une activity,**
  - **Pour ce premier exemple,**
    - **un affichage (sans contrôle, sans clics de l'utilisateur)**
    - **Quelques instructions de la méthode onCreate**
- **Le modèle**
  - **Un diplôme Cnam constitué de plusieurs unités d'enseignement**

Le fichier res/layout/liste.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent" >

        <ListView
            android:id="@+id/listeId"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1" >

        </ListView>
</LinearLayout>
```

  - La liste est un composant graphique

- **Pour ce premier exemple**
  - Le « format » de chaque ligne de cette liste est choisi parmi les formats prédéfinis
    - android.R.layout.simple_list_item_1
    - android.R.layout.simple_list_item_2

    - Cf. http://developer.android.com/reference/android/R.html

L'activity, ici sans contrôle

```
public class ListePremierExemple extends Activity {
    private static final String[] itemsCC=
        {"NFP121", "RSX116", "NSY102","SMB116","SMB117","UARS01"};
    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.liste);
      ListView lv =(ListView)findViewById(R.id.listeId);
      lv.setAdapter(new ArrayAdapter<String>(
                          this,
                          android.R.layout.simple_list_item_1,
                          itemsCC));
```

- **Le modèle :**
  - **ici un certificat de compétences constitué de 6 unités**
    - **NFP121, RSX116, NSY102, SMB116, SMB117, UARS01**
    - **Unités enseignées au Cnam (présentiel et à distance)**
    - **Recherche Google certificat intégrateur applications mobiles**

Exécution, le rendu

- **Un rendu simple d'un item de texte par ligne**
  - **android.R.layout.simpl_list_1 est prédéfini**
    - **Serait-ce un TextView ?**

android.R.simple_list_item_1 ???

simple_list_item_1:

```
<TextView xmlns:android="http://schemas.android.com/apk/res/android"
 android:id="@android:id/text1"
        style="?android:attr/listItemFirstLineStyle"
        android:paddingTop="2dip"
        android:paddingBottom="3dip"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content" />
```

simple_list_item_2:

```
<TwoLineListItem xmlns:android="http://schemas.android.com/apk/res/
android"
        android:paddingTop="2dip"
        android:paddingBottom="2dip"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">

        <TextView android:id="@android:id/text1"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                style="?android:attr/listItemFirstLineStyle"/>

        <TextView android:id="@android:id/text2"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:layout_below="@android:id/text1"
                style="?android:attr/listItemSecondLineStyle" />
</TwoLineListItem>
```

- afficher le texte des messages précédents -

- **La ligne de la liste est décrite ainsi**

http://developer.android.com/reference/android/R.html

int	simple_expandable_list_item_2	
int	simple_gallery_item	
int	simple_list_item_1	
int	simple_list_item_2	
int	simple_list_item_activated_1	A version of `simple_list_item_1` that is able to change its background state to indicate when it is activated (that is checked by a ListView).
int	simple_list_item_activated_2	A version of `simple_list_item_2` that is able to change its background state to indicate when it is activated (that is checked by a ListView).
int	simple_list_item_checked	
int	simple_list_item_multiple_choice	
int	simple_list_item_single_choice	
int	simple_selectable_list_item	A simple ListView item layout which can contain text and support (single or multiple) item selection.
int	simple_spinner_dropdown_item	
int	simple_spinner_item	
int	test_list_item	
int	two_line_list_item	

- **Un extrait**

Un deuxième exemple

Le premier exemple s'est enrichi

- **Ajoutons pour chaque item**
  - **Une icône pour la période d'enseignement**
    - **1er ou 2ème semestre**
  - **Le lieu d'enseignement en présentiel**
    - **Prévoir que ce lieu puisse changer…**

Description xml, d'une ligne

Chaque ligne contient :

- **L'intitulé de l'unité**
  - **textView**
- **Une image suivie du lieu d'enseignement**
  - **imageView suivie textView**

- **Le fichier res/layout/uecnam.xml**


```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:orientation="vertical" android:weightSum="1">
    <TextView android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="TextView" android:layout_width="match_parent"
        android:id="@+id/nomId" android:textSize="32dp"></TextView>
    <TableRow android:layout_height="wrap_content" android:id="@+id/tableRow1"
        android:layout_width="match_parent">
        <ImageView android:layout_height="wrap_content" android:src="@drawable/icon"
            android:layout_width="wrap_content" android:id="@+id/imageId"></ImageView>
        <TextView android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="TextView" android:layout_width="wrap_content"
            android:layout_height="wrap_content" android:id="@+id/lieuId"
            android:textSize="24dp"></TextView>
    </TableRow>
</LinearLayout>
```

Proposons maintenant notre propre Adapter

- **Selon le patron Strategy,**
  - **Une classe implémentant l'interface ListAdapter**

Aidons nous de l'existant …

- **Il existe une sous classe qu'il suffit de dériver**
  - **BaseAdapter**
    - **Il suffira de définir getCount, getItem, getItemId et getView**
    - **Nous l'appellerons UECnamAdapter**
  - **Notre modèle évolue**
    - **Diplôme, UE deviennent des objets métiers**
    - **Un diplôme est constitué d'unités d'enseignement (UE)**
      - **Chaque diplôme délivre une liste d'UE**
    - **Une UE est constitué d'un intitulé, du lieu d'enseignement et de la période**

L'activité ne change guère, tjs quelques lignes

```
public class ListeDeuxiemeExemple extends Activity {


    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.liste);
      ListView lv =(ListView)findViewById(R.id.listeId);

      lv.setAdapter(new UECnamAdapter(this,DiplomeCnam.CCY114));

    }
```

- **Une stratégie concrète nommée UECnamAdapter**
  - **Il reste les méthodes**
    - **getCount, getItem, getItemId et getView à définir**

La classe UECnamAdapter

```
public static class UECnamAdapter extends BaseAdapter{
    private Context ctxt;
    private DiplomeCnam diplome;

    public UECnamAdapter(Context ctxt, DiplomeCnam diplome){
        this.ctxt = ctxt;
        this.diplome = diplome;
    }
    public int getCount() {
        return diplome.getListe().size();
    }
    public Object getItem(int position) {
        return diplome.getListe().get(position);
    }
    public long getItemId(int position) {
        return position;
    }
```

- **Tout diplôme propose la liste des unités qui le compose**
  - **Une liste est retournée**
    - **List<UE> getListe() est une méthode d'instance de la classe DiplomeCnam**
    - **getCount, getItem, getItemId sont immédiats**

La classe UECnamAdapter, méthode getView

- **Cette méthode est appelée par Android**
  - **À chaque affichage de la liste et seulement sur les lignes présentées**
    - **Notre exemple présente 5 unités sur 6**
      - **5 appels de getView**

  - **Principe :**
    - **Allons chercher le fichier XML défini pour l'UE (uecnam.xml)**
    - **Réalisons l'adéquation**
      - **ListView/Modèle**
        - » Ligne de la liste/ UE du diplôme
      - **Affectons les items de chaque View avec chaque certains attributs de l'UE**

La méthode getView

```
public View getView(int position, View convertView, ViewGroup parent) {

   LayoutInflater inflater = null;
   inflater = (LayoutInflater)ctxt.getSystemService(Context.LAYOUT_INFLATER_SERVICE);

   View itemView = inflater.inflate(R.layout.uecnam, parent, false);
   TextView nomView =(TextView)itemView.findViewById(R.id.nomId);
   ImageView imageView = (ImageView)itemView.findViewById(R.id.imageId);
   TextView lieuView=(TextView)itemView.findViewById(R.id.lieuId);

   nomView.setText(diplome.getListe().get(position).getNom());
   lieuView.setText(diplome.getListe().get(position).getLieu());
   String periode = diplome.getListe().get(position).getPeriode();
   if(periode.equals("1"))
       imageView.setImageResource(R.drawable.orange01);
   else
       imageView.setImageResource(R.drawable.green02);

  return itemView;
}
```

- **Au complet, explications**
  - **LayoutInflater lecture du fichier XML**
  - **Inflate(le fichier XML, la racine, booléen (complétion d'un arbre existant)**

Prohibitif en temps d'exécution !

- **Inflate transforme à la volée le fichier XML en arbre Java**
  - En version naïve cette transformation se produit à chaque appel
  - A chaque manipulation de l'utilisateur
    - …
  - Ce qui pourrait être pénalisant en temps d'exécution

```
public View getView(int position, View convertView, ViewGroup parent) {
```

- **Le système peut avoir créé la vue**
- **Celle-ci est transmise à la méthode getView de l'adapter**
  - Un contrat est en place
    - Si le paramètre convertView == null, la vue doit être créée
      - Alors les liens sur les vues de la ligne sont conservées
    - Sinon une référence de la vue est transmise et peut donc être utilisée
      - Nous utiliserons alors les éléments du cache
      - Le cache est à installer

Gestion du cache

- **Une classe interne et statique au sein de l'adapter**

```
private static class CacheView{
    public TextView  nom;
    public ImageView image;
    public TextView  lieu;
}

public View getView(int position, View convertView, ViewGroup parent) {
  View itemView = convertView;
  if(convertView == null){
```

Alors la vue doit être créée les liens sur les vues de la ligne sont placées dans un cache

Gestion du cache au complet

```
public View getView(int position, View convertView, ViewGroup parent) {
  View itemView = convertView;
  if(convertView == null){
    LayoutInflater inflater = null;
    inflater = (LayoutInflater)ctxt.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
    itemView = inflater.inflate(R.layout.uecnam, parent, false);
    CacheView cache = new CacheView();
    cache.nom  = (TextView)itemView.findViewById(R.id.nomId);
    cache.image= (ImageView)itemView.findViewById(R.id.imageId);
    cache.lieu = (TextView)itemView.findViewById(R.id.lieuId);
    itemView.setTag(cache);
  }

  CacheView cache = (CacheView)itemView.getTag();
  cache.nom.setText(diplome.getListe().get(position).getNom());
  cache.lieu.setText(diplome.getListe().get(position).getLieu());
  String periode = diplome.getListe().get(position).getPeriode();
  if(periode.equals("1"))
      cache.image.setImageResource(R.drawable.orange01);
    else
      cache.image.setImageResource(R.drawable.green02);


 return itemView;
}
```

- Le champ tag d'une vue permet de memoriser une reference
- Ici le cache entre deux affichages

Tri, filtrage

- **La classe Adapter s'en charge**
- **Ou bien le modèle propose ce type de méthodes**

- **Ordre alphabétique des UE**
- **Ordre en fonction de la période (1er ou 2ème semestre)**

- **Etc..**

En exemple de tri, ici selon le nom et la période

- **Il suffit de proposer, selon les critères retenus**
  - **Une implémentation de l'interface Comparator entre deux UE**

Le lieu d'enseignement change …

- **Ce qui peut arriver, le modèle change**
  - **La vue est prévenue ….**
  - **Changement du modèle**
    - **RSX116 est maintenant programmé en studio d'enregistrement en 17 2 6**
    - **Appel du mutateur, changement de lieu**
  - **Appel de la méthode de l'adapter**
    - `((BaseAdapter)lv.getAdapter()).notifyDataSetChanged();`
  - **Adéquation par Android du modèle et de la vue**
  - **Réactualisation, appel de getView automatique**

Une classe dédiée ListActivity

- **Simplification :**
  - **usage des ListView par une sous classe d'Activity : ListActivity**

```
public class ListeTroisiemeExemple extends ListActivity {

    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      // setContentView(R.layout.list); inutile
      // il faut une liste dont l'id est list  (@android:id/list)
      setListAdapter(new UECnamAdapter(this,DiplomeCnam.CCY114));

    }
```

Avec une Activity,
Le 2ème exemple

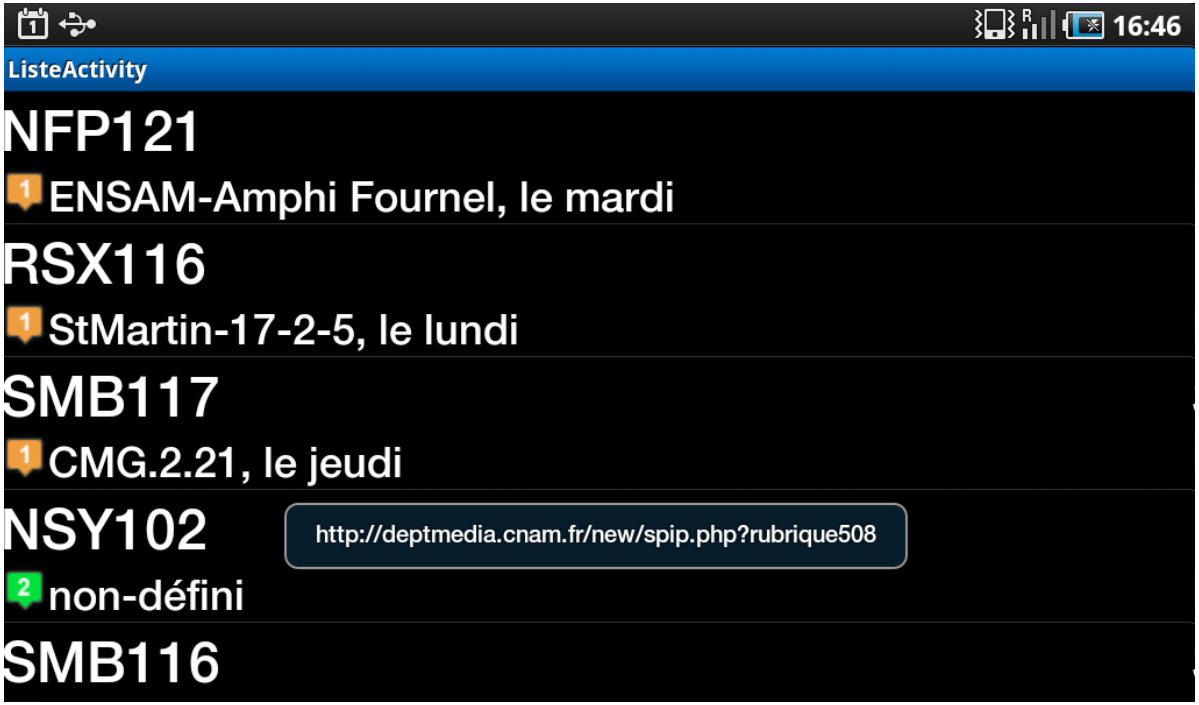
```
public class ListeDeuxiemeExemple extends Activity {


    @Override
    public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.liste);
      ListView lv =(ListView)findViewById(R.id.listeId);

      lv.setAdapter(new UECnamAdapter(this,DiplomeCnam.CCY114));

    }
```

Contrôleur, comportement

- **Au clic, contextuel**
  - **A chaque clic un toast est présenté, l'URL de l'UE est affiché**
    - **À terme le site de l'unité pourrait être présenté …**
  - **Méthode onListItemClick, classe ListActivity**

```
protected void onListItemClick(ListView lv, View v, int position, long id){
    String url = DiplomeCnam.CCY114.getListe().get(position).getURL();
    Toast.makeText(this,url,Toast.LENGTH_LONG).show();
}
```

A chaque clic, un Toast est affiché



- **Ici un clic sur RSX116 affiche l'URL de l'unité**
- **Une deuxième activité pourrait être exécutée**
  - **Par exemple une WebView …**

Conclusion

- **C’est une brève présentation, les outils pour faire le TP**

- **Il en reste**
  - **SimpleAdapter qui n’est pas si simple …**
  - **CursorAdapter reflet du cursor lors d’une requête sur une table de BdD**
  - **…**

Chargement des items d'une liste et AsyncTask

- **Si le chargement des items d'une liste devient coûteux en temps d'exécution**

Alors ce chargement pourrait être effectué en tache de fond

- **Construction de la liste item par item en tâche de fond**
  - **Usage de AsyncTask**
  - **Reprenons le premier exemple …**

Une exemple d'usage d'AsyncTask

```
public class ListeQuatriemeExemple extends ListActivity {

    private static final String[] itemsCC=
    {"NFP121", "RSX116", "NSY102","SMB116","SMB117","UARS01"};


public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);

  setListAdapter(new ArrayAdapter<String>(
                       this,
                       android.R.layout.simple_list_item_1,
                       new ArrayList<String>()));

  new ConstructionListe().execute();

}
```

- **ConstructionListe extends AsyncTask**

ConstructionListe extends AsyncTask

```
private class ConstructionListe extends AsyncTask<Void, String, Void>{

    protected Void doInBackground(Void... args) {
          for(String item : itemsCC){
            SystemClock.sleep(5000);
            publishProgress(item);
          }
        return null;
    }

    protected void onProgressUpdate(String... item) {
          ((ArrayAdapter<String>)getListAdapter()).add(item[0]);
        }
```

CursorAdapter et les contacts

- http://www.java2s.com/Code/Android/UI/UsingSimpleCursorAdapter.htm