Android
Fragment et MVC

Rémi Forax

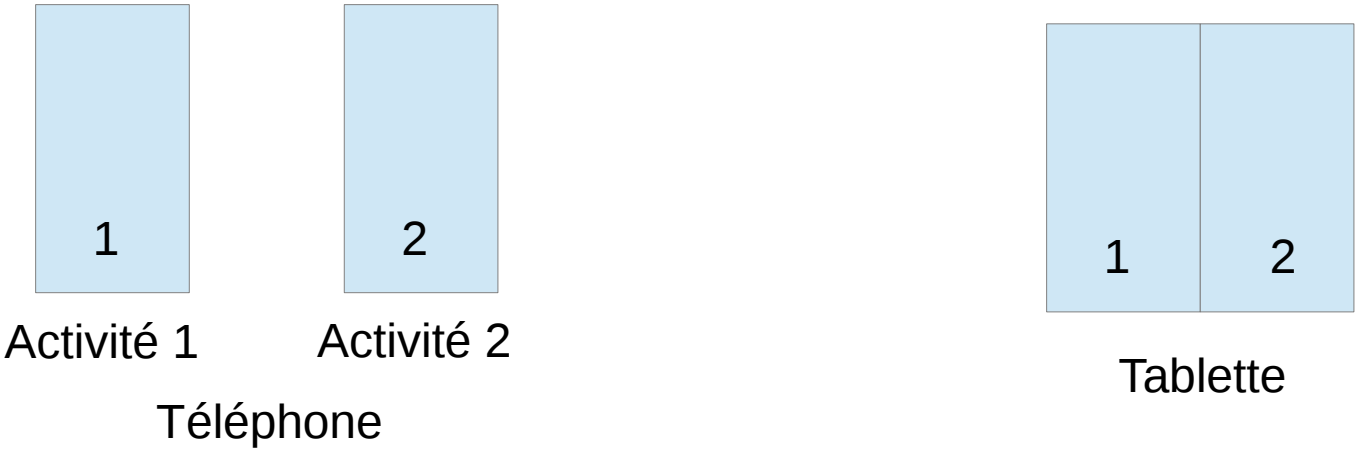
Fragment

Fragment

Sous-activité introduite par la version 11 (Android 3.0) et possède aussi un cycle de vie

Il permet de découpe un écran en plusieurs partie, plusieurs fragments.

Permet de gérer les tablettes et les téléphones de la même façon



La classe Fragment

De même que pour une Activity, un fragment à un layout en XML et une classe associée, cette classe doit hérité de la classe android.app.Fragment

Un fragment peut mourir puis être reconstruit

- Il doit exister un constructeur sans paramètre
- Si on veut garder des informations
  - getArguments/setArguments(Bundle)

Un exempe simple

```
public class SimpleFragment extends Fragment {
    private String name;

    public SimpleFragment() {
      // called when reconstructing a fragment
    }

    public static SimpleFragment create(String name) {
      SimpleFragment fragment = new SimpleFragment();
      Bundle arguments = new Bundle();
      arguments.putString("name", name);
      fragment.setArguments(arguments);
      return fragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Bundle arguments = getArguments();
        name = (arguments != null)? arguments.getString("name", "empty"): "empty";
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        View layout = inflater.inflate(R.layout.simple, container, false);
        TextView text = (TextView)layout.findViewById(R.id.text);
        text.setText(name);
        return layout;
    }
}
```

Factory method
pour passer des paramètres

décodage des paramètres

Cycle de vie d'un fragment

Un fragment est soit décalré en XML avec la balise <fragment> soit ajouté dynamiquement en utilisant un FragmentManager

Une activité gère un BackStack interne de fragment

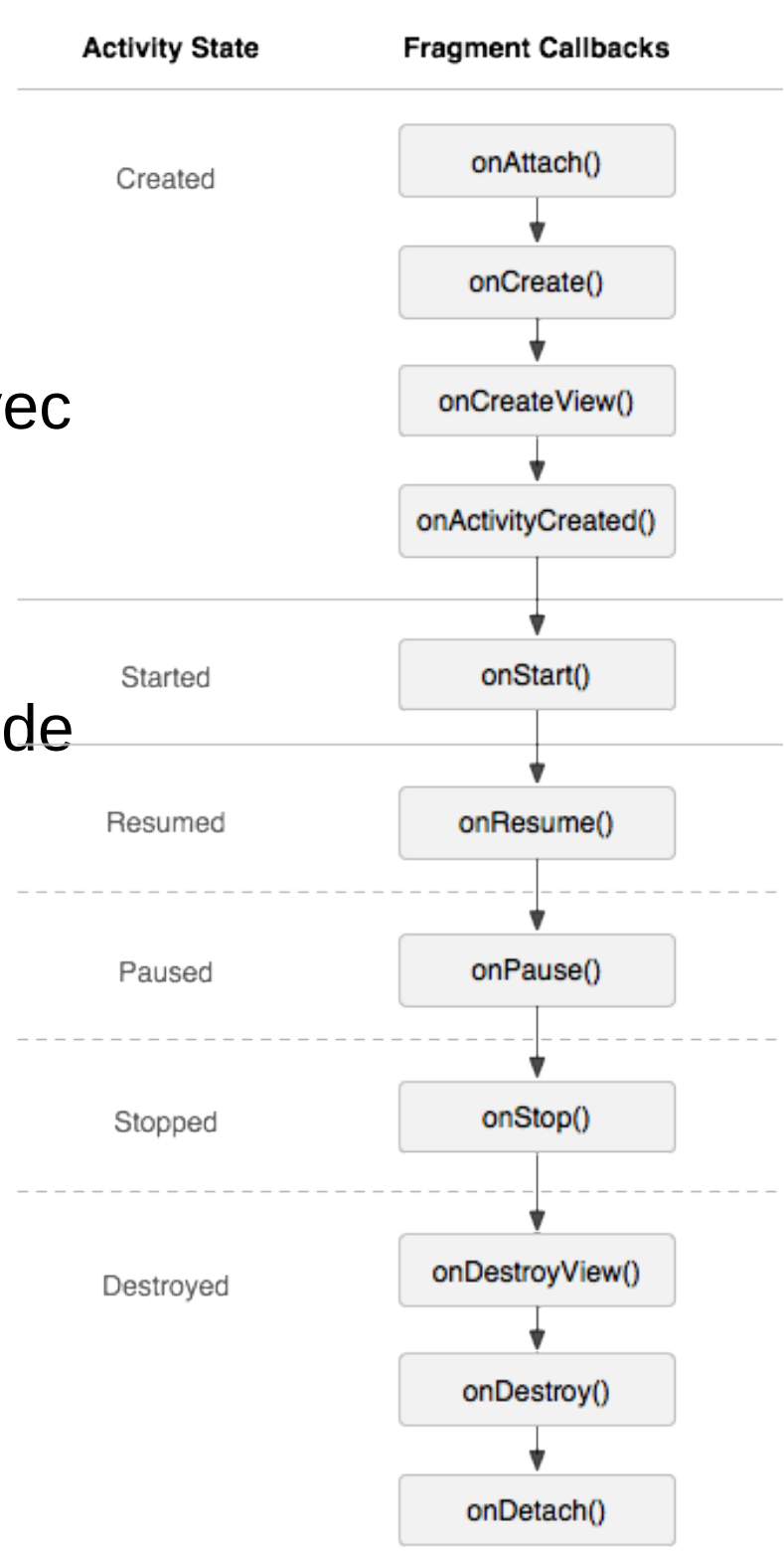
- Création d'une transaction

  fragmentManager.beginTransaction()
- Modification d'une transaction

  transaction.add/remove/replace
- Gestion du BackStack interne

  transaction.addToBackStack/popBackStack
- Validation de la transaction

  transaction.commit()



Gestion du backstack

```
public class SimpleFragment extends Fragment {
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle ...) {
        View layout = inflater.inflate(R.layout.simple, container, false);
        ...
        Button button = ...
        button.addOnClickListener(new OnCLickListener() {
          public void onClick(View view) {
            SimpleFragment fragment = create(editText.getText());
            FragmentTransaction transaction = getFragmentManager().beginTransaction();
            transaction.replace(R.id.simple_fragment, fragment);
            transaction.setTransition(
                FragmentTransaction.TRANSIT_FRAGMENT_OPEN);
            transaction.addToBackStack(null);
            transaction.commit();
          }
        });
        return layout;
    }
}
```

Adaptation au matériel avec des fragments

On créé plusieurs layout adaptés au matériel dans res/layout-X/mylayout.xml

- Un LinearLayout avec deux fragments
- Un FrameLayout avecun seul fragment

A l'exécution, on peut tester si un layout existe ou non car findViewById(...) renvoie null

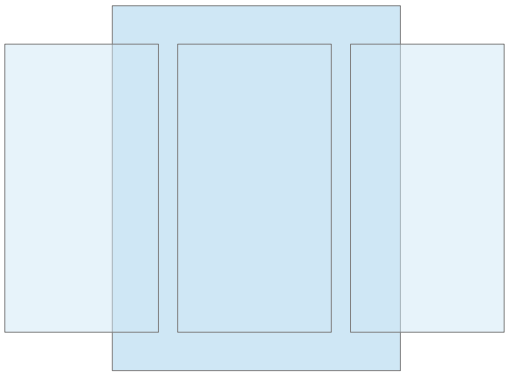
On adapte le comportement d'affichage

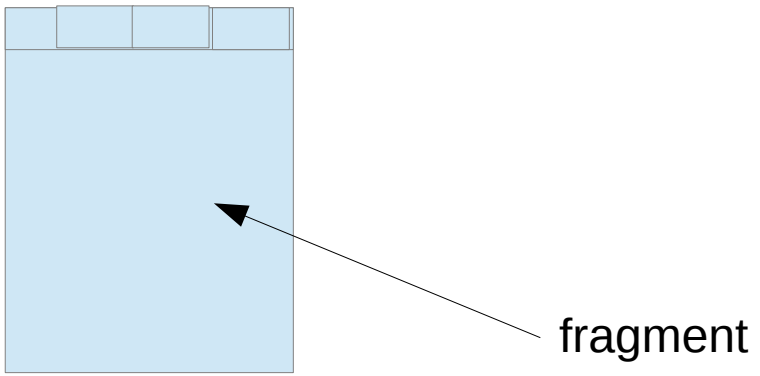
- Si deux fragments, mise à jour du second fragment
- Si un seul fragment, lancement d'une nouvelle activité avec le second fragment (Intent)

Autres usages des fragments

Une activité peut afficher plusieurs fragments, ce qui permet de gérer par exemple un caroussel

Les onglets de la menubar permettent aussi de changer de fragment dans une activité





Donnée & Orientation

Donnée et changement d'orientation

Un changement d'orientation du device détruit l'activité courante pour la recréer

- Car l'écran graphique doit s'adapter à la nouvelle configuration
- Même mécanisme que si l'application est détuite par le système

Il est possible de passer des données entre les deux activités

Données actives d'une activité (Bundle)

Gestion par défaut

Si l'on appel super.onCreate(bundle) dans onCreate() alors la configuration de chaque composant (texte en cours d'édition par ex.) est sauvegardé et restaurer

Si on veut sauvegarder des états supplémentaires, il faut le à la main

- un Bundle permet de stocker uniquement des valeurs de type primitif, pas des objets !

```
public class MainActivity1 extends Activity {
  @Override protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Data data = new Data();
    if (savedInstanceState != null) {
      data = new Data(savedInstanceState);
    }
    EditText emailAddressEdit = (EditText)findViewById(R.id.editText1);
    emailAddressEdit.setText(data.emailAddress);
    EditText messageEdit = (EditText)findViewById(R.id.editText2);
    messageEdit.setText(data.message);
  }
  @Override protected void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    EditText emailAddressEdit = (EditText)findViewById(R.id.editText1);
    String emailAddress = emailAddressEdit.getText().toString();
    EditText messageEdit = (EditText)findViewById(R.id.editText2);
    String message = messageEdit.getText().toString();
    data.store(new Data(emailAddress, message));
  }
  static class Data {
    private final String emailAddress;
    private final String message;
    public Data(Bundle bundle) {
      emailAddress = bundle.getString("emailAdress");
      message = bundle.getString("message");
    }
    public void store(Bundle bundle) {
      bundle.putString("emailAdress", emailAddress);
      bundle.putString("message", message);
    }
```

Changement d'orientation **rapide**

Si l'application utilise des bitmaps ou des ressources chères à recalculer, le changement d'orientation va être lent

avec Android <3.0

Il est possible de passer un objet entre l'activité actuelle et l'activité qui sera créée après le changement d'orientation

- redéfinir onRetainNonConfigurationInstance()
- Et dans onCreate, tester si getLastNonConfigurationInstance() est null ou pas

avec Android >=3.0

L'activité meurt mais le fragment “top-level” reste

- Le fragment doit être en-avant plan (pas dans le BackStack)
- Le fragment doit être marqué comme setRetainInstance()

Avec une activité

```
public class MainActivity2 extends Activity {
  static class Data {
    ...
  }
  private transient Data data;
  @Override protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    Data data = (Data)getLastNonConfigurationInstance();
    if (data == null) {
      data = new Data();
      if (savedInstanceState != null) {
        data.load(savedInstanceState);
      }
    }
    this.data = data;
    EditText emailAddressEdit = (EditText)findViewById(R.id.editText1);
    emailAddressEdit.setText(data.emailAddress);
    EditText messageEdit = (EditText)findViewById(R.id.editText2);
    messageEdit.setText(data.message);
  }
  @Override public Object onRetainNonConfigurationInstance() {
    return data;
  }
```

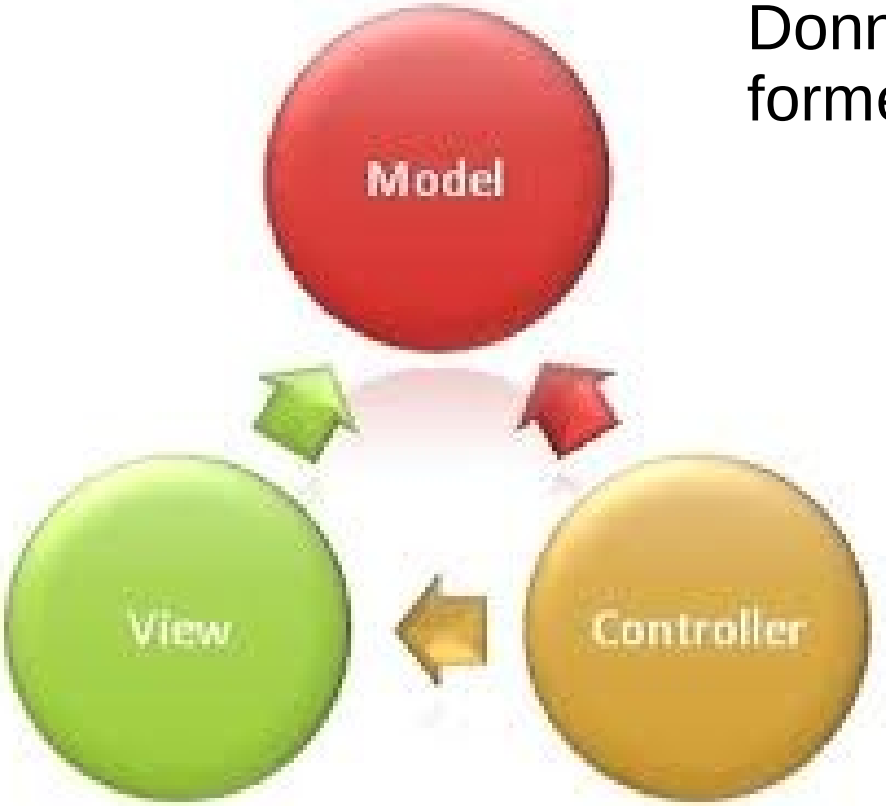
Avec un fragment

```
public class MainFragment extends Fragment {
  static class Data {
    ...
  }
  private transient Data data;
  @Override public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setRetainInstance(true);
  }
  @Override public View onCreateView(LayoutInflater inflater, ViewGroup container,
                                     Bundle savedInstanceState) {
    return inflater.inflate(R.layout.fragment_main, container);
  }
  @Override public void onActivityCreated(Bundle savedInstanceState) {
    super.onActivityCreated(savedInstanceState);
    Data data = new Data();
    if (savedInstanceState != null) {
      data = new Data(savedInstanceState);
    }
    Activity activity = getActivity();
    EditText emailAddressEdit = (EditText)activity.findViewById(R.id.editText1);
    emailAddressEdit.setText(data.emailAddress);
    ...
  }
  @Override public void onSaveInstanceState(Bundle outState) {
    super.onSaveInstanceState(outState);
    Activity activity = getActivity();
    EditText emailAddressEdit = (EditText)activity.findViewById(R.id.editText1);
    ...
    Data data = new Data(emailAdress, message);
    data.store(outState);
  }
}
```

MVC

Modèle Vue Controleur

Model-View-Controller



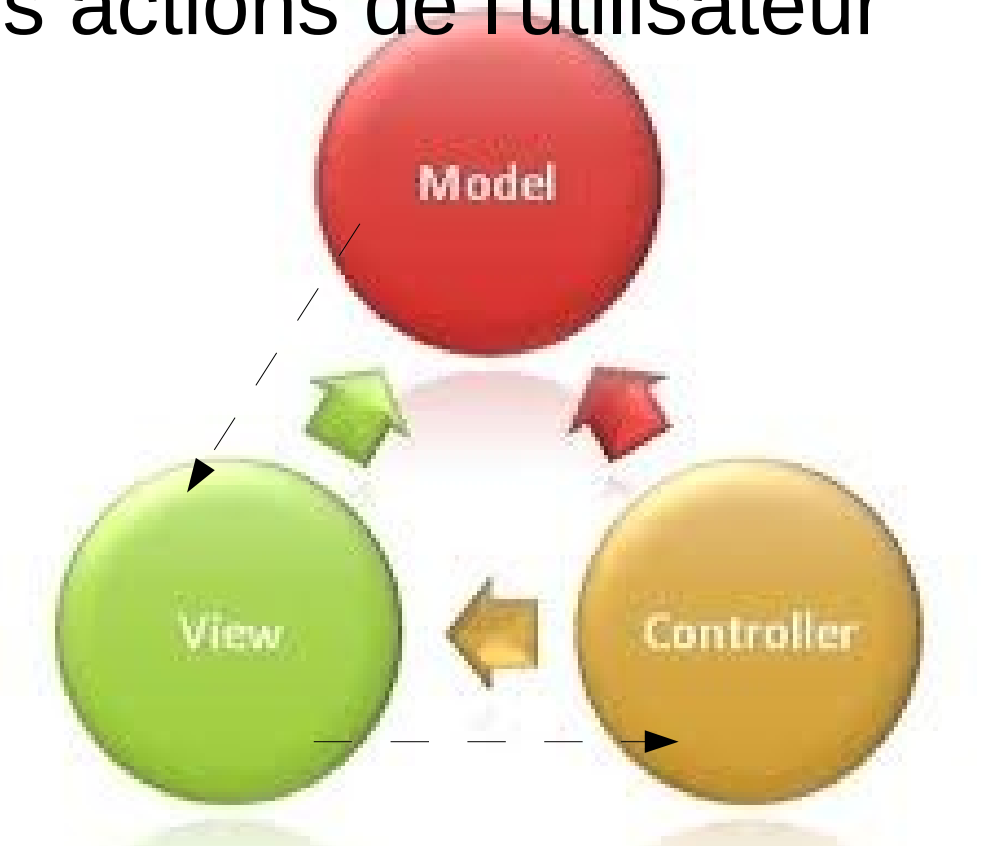
Donnée métier dans une forme adaptée à la vue

Affichage graphique des données

Action sur demande de l'utilisateur qui va modifier les donnée

MVC avec les évènements

- La vue s'enregistre sur le modèle pour être prévenue des modifications de celui-ci
- Le controleur s'enregistre sur la vue pour être prévenu des actions de l'utilisateur



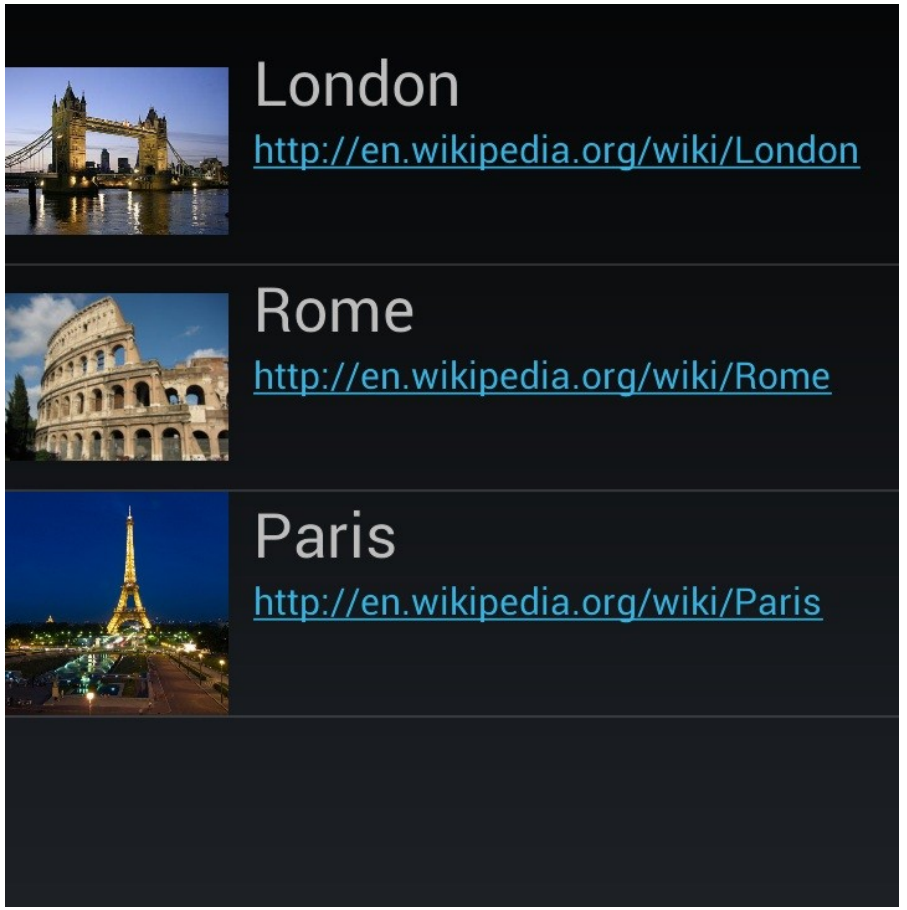
Un exemple de simple

ListView utilise le design pattern MVC

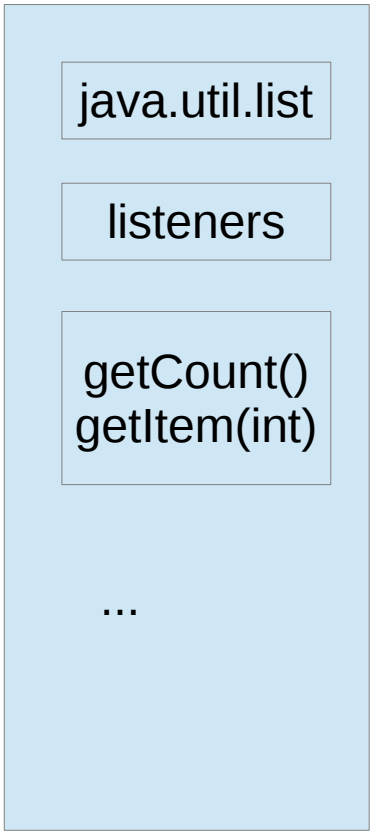
- La vue: ListView
- Le modèle: ListAdapter
- Les controleurs: tous les listeners que vous voulez implanter

ListView / ListAdapter

Le ListAdapter est le modèle des ListView



ListView



ListAdapter

ListView simple

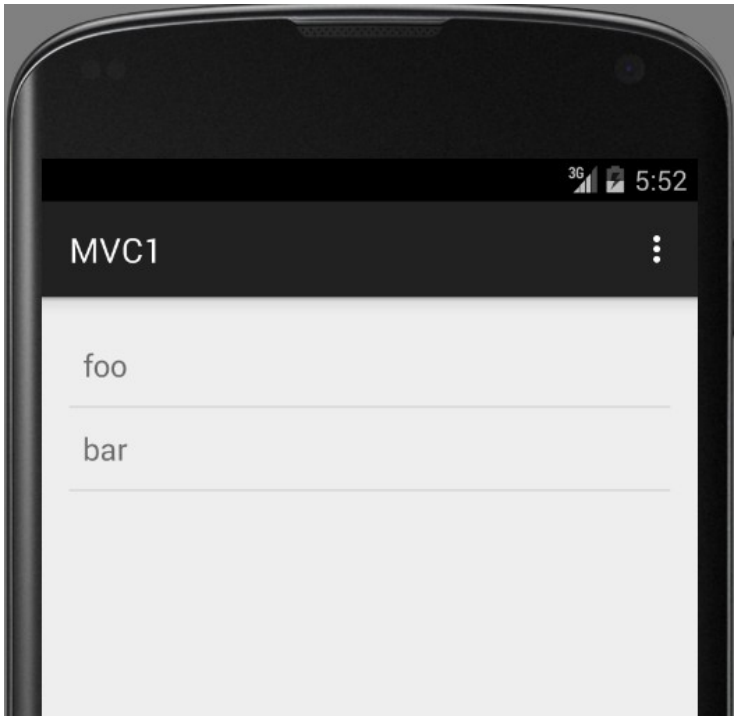
Au niveau de l'activité

```
public class MainActivity {
  @Override
  protected void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.activity_main);

    ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1);
    adapter.add("foo");
    adapter.add("bar");

    ListView listView = (ListView)findViewById(R.id.mylist);
    listView.setListAdapter(adapter);
```

ListView simple

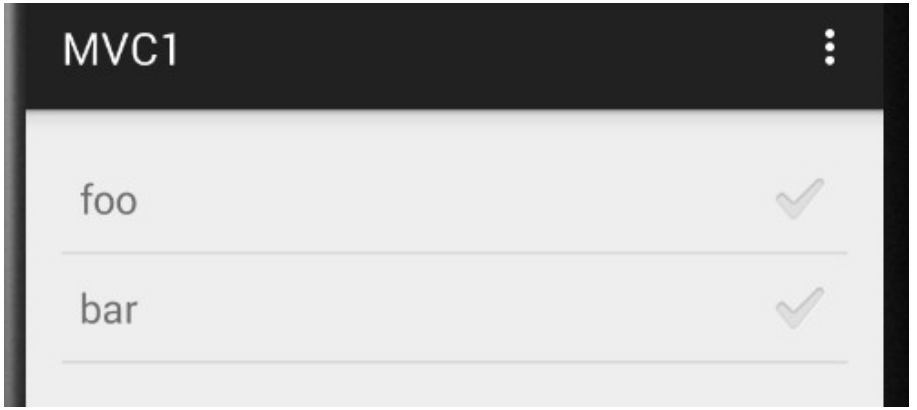
Dans le fichier res/layout/activity_XXX.xml

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

  <ListView android:id="@+id/mylist"
    android:layout_width="fill_parent"
    android:layout_height="0dip"
    android:layout_weight="1"  />

</LinearLayout>
```



Le Layout de chaque item

Permet de changer l'affiche de chaque item




```
ListView listView = (ListView)findViewById(R.id.mylist);
ListAdapter adapter = new ArrayAdapter<String>(this,
    android.R.layout.simple_list_item_checked);
adapter.add("foo");
adapter.add("bar");
listView.setListAdapter(adapter);
```

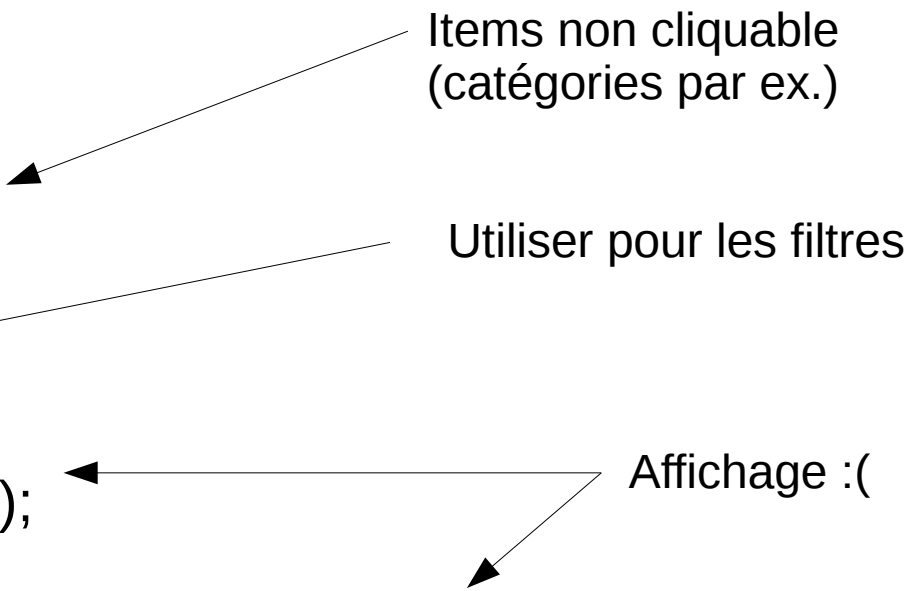
Layout pour chaque item

Le ListAdapter d'Android

```
public interface ListAdapter {
  void unregisterDataSetObserver(DataSetObserver observer);
  void registerDataSetObserver(DataSetObserver observer);

  int getCount();
  Object getItem(int position);

  …  // plein d'autres méthodes
       // qui ne font pas partie du design pattern




}
```

Le ListAdapter d'Android

```
public interface ListAdapter {
  void unregisterDataSetObserver(DataSetObserver observer);
  void registerDataSetObserver(DataSetObserver observer);

  int getCount();
  Object getItem(int position);
  boolean isEmpty();

  boolean areAllItemsEnabled()
  boolean isEnabled(int position)

  boolean hasStableIds();
  long getItemId(int position);

  int getViewTypeCount();
  int getItemViewType(int position);

  View getView(int position, View convertView, ViewGroup parent);
}
```



Le modèle

void unregisterDataSetObserver(DataSetObserver observer)
void registerDataSetObserver(DataSetObserver observer)

Un DataOverver est une ListView qui veut être au courant des modifications

- Pas besoin d'implanter ces méthodes si le modèle est non mutable

int getCount()

Nombre total d'item de la liste

- Cette méthode ne doit pas être en O(n2) !

Object getItem(int position)

Renvoie un Item à une position, les appels ne s'effectue par focément dans l'ordre

- Cette méthode ne doit pas être en O(n2) !

boolean isEmpty()

Si le modèle est vide, la vue peut utiliser un autre layout

- souvent return getCount() == 0

Liste & Catégories

Une ListView permet de gérer des “catégories” qui ne sont pas des Views cliquables

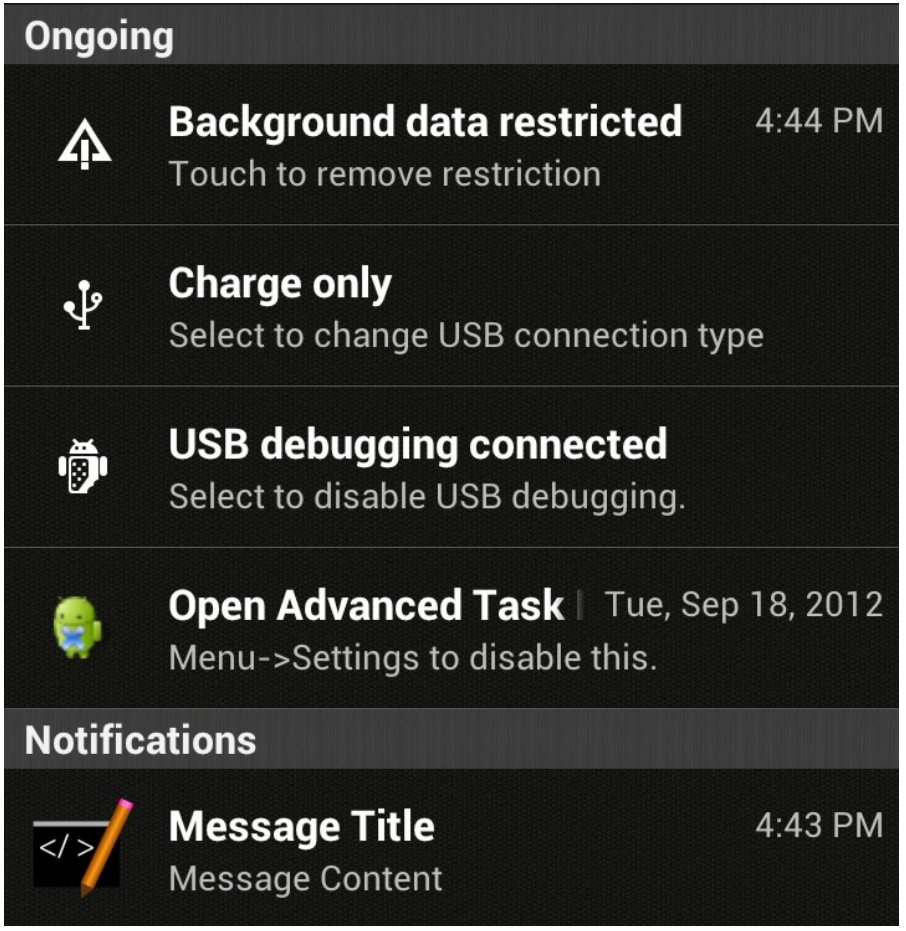
boolean areAllItemsEnabled()

- Vrai si pas de catégorie

boolean isEnabled(int position)

- false si c'est l'item est une catégorie

Ici, il y a 7 items, isEnabled(0) et isEnabled(5) renvoie false



Type de vue

ListView peut utiliser des plusieurs type de views

int getViewTypeCount()

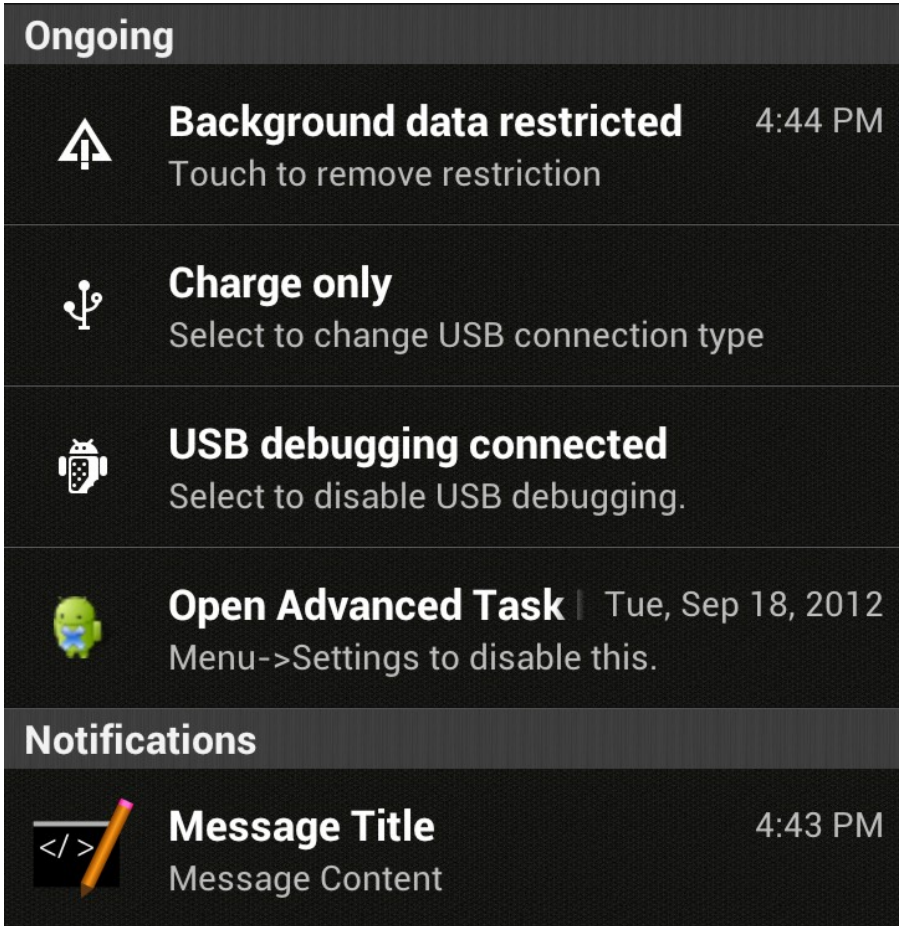
- Nombre de type de vues

int getItemViewType(int position)

- Type de vue pour une position

View getView(int position,
View convertView,
ViewGroup parent)

- ConvertView est recyclée en fonction de son type



Vues et Recyclage de vues

La méthode getView

View getView(int position, View convertView, ViewGroup parent)

associe une vue à un item donné.

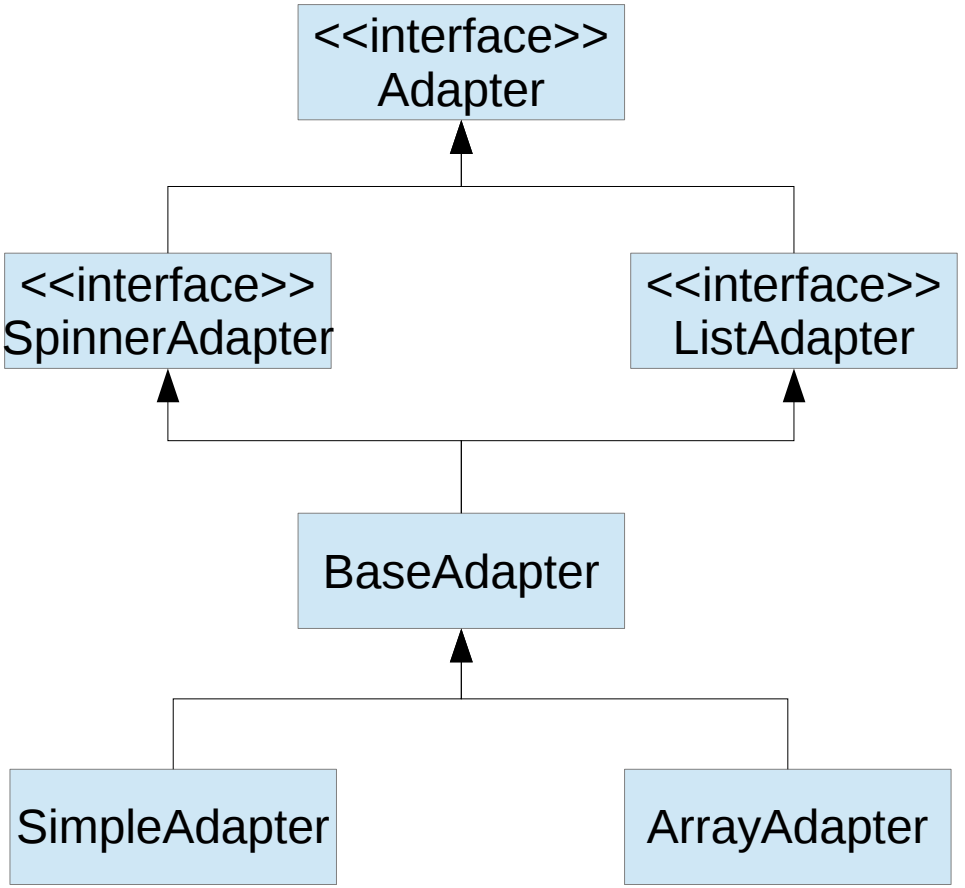
Seul les vues associées à des items visibles sont crées pour éviter d'allouer trop d'élement

- Les vues sont recyclées, la vue d'un élement qui n'est plsu visible peut être utiliser pour un élement qui devient visible

  convertView correspond une vue recyclée et donc peut être null

Les différents modèles existant

Hierarchie des adapteurs existants



Les différents modèles existant

Adapter

- Interface de Base des ListAdapter et SpinnerAdapter

ListAdapter étend Adapter

- Les items peuvent être enable ou non

SpinnerAdapter

- Vue spécifique pour bouton “drop down”: getDropDownView

BaseAdapter implante ListAdapter, SpinnerAdapter

- Implante les listeners, tous est enable par défaut, et il n'y a qu'un seul type de view

SimpleAdapter étend BaseAdapter

- Classe spécialisé pour des données **non mutable** sous forme d'une List de Map (la Map définie les valeurs pour les colonnes)

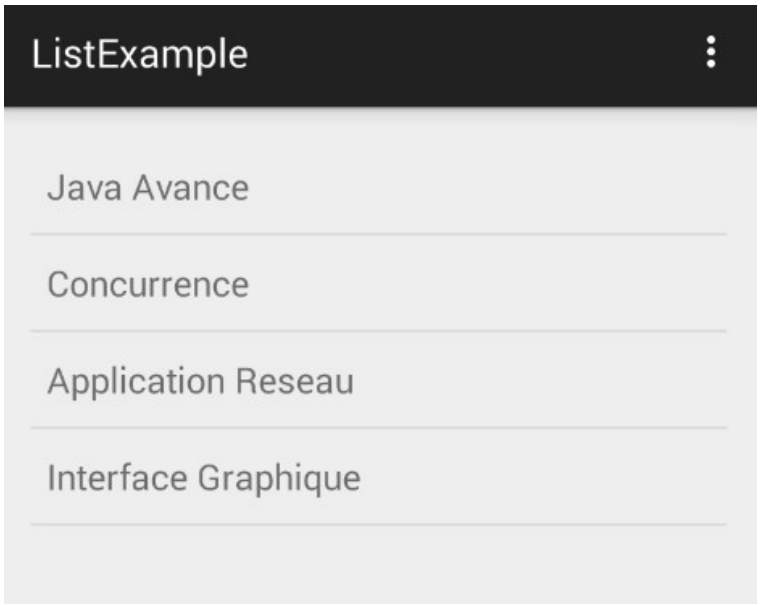
ArrayAdapter étend BaseAdapter

- Classe spécialisée pour les données **mutable** sous forme de liste

Modèle non mutable

“Faire une liste de cours”

On déclare les noms des cours dans les ressources



On créé un ListAdapter remplie avec les ressources

Le fichier de ressources

- Il est possible de spécifier un contenu statique en tant que ressource string-array
- Le fichier peu avoir n'importe quel nom mais doit être dans le répertoire res/values

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
  <string-array name="class_array">
      <item>Java Avance</item>
      <item>Concurrence</item>
      <item>Application Reseau</item>
      <item>Interface Graphique</item>
  </string-array>
</resources>
```

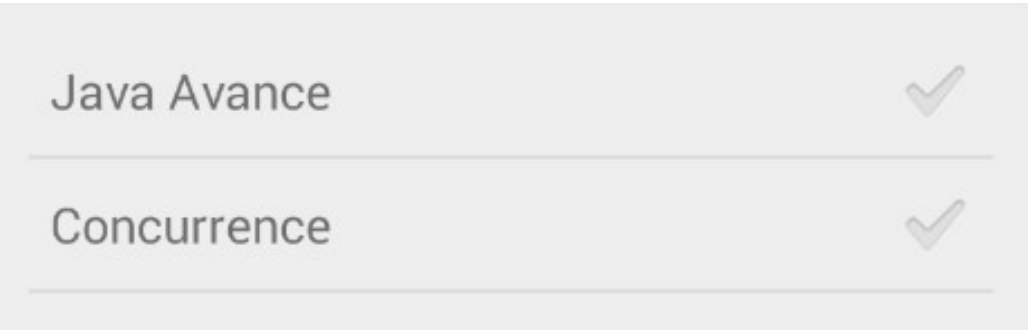
Modèle non mutable

ArrayAdapter.createFromResource permer tde créer un modèle à partie d'un tableau de String

```
@Override
protected void onCreate(Bundle savedInstanceState){
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  ListAdapter adapter =
     ArrayAdapter.createFromResource(this,
          R.array.class_array,
          android.R.layout.simple_list_item_1);
  ListView listView = (ListView)findViewById(R.id.mylist);
  listView.setAdapter(adapter);
```

Modèle mutable simple

On veut maintenant pouvoir cocher (ou décocher) les cours



- On doit avoir un modèle qui sauvegarde si chaque item est coché ou non
- On doit changer la représentation des items pour afficher graphiquement la coche

Liste de cours

Il faut se créer son propre modèle i.e.
son propre ListAdapter

```
protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  Resources res = getResources();
  String[] classes = res.getStringArray(R.array.class_array);
  LayoutInflater layoutInflater = getLayoutInflater();

  ListAdapter adapter = new ClassListAdapter(classes, layoutInflater);
  ListView listView = (ListView)findViewById(R.id.mylist);
  listView.setAdapter(adapter);
}
```

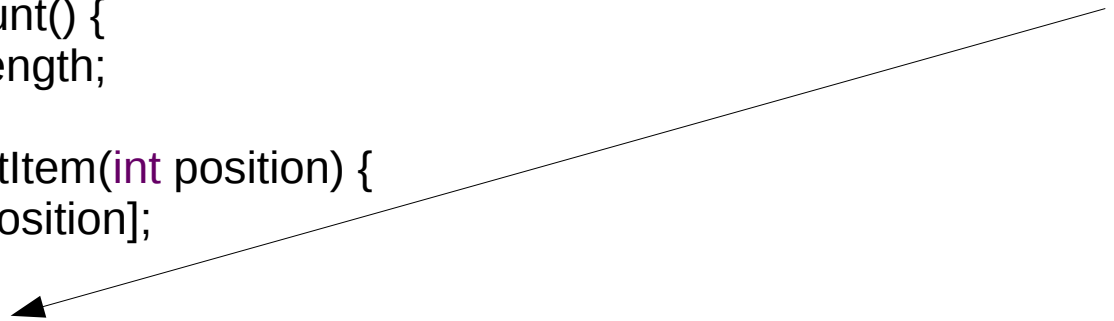
Modèle mutable

```
class ClassListAdapter extends BaseAdapter {
  private final Object[] values;
  private final boolean[] checked;
  private final LayoutInflater layoutInflater;

  public ClassListAdapter(Object[] values, LayoutInflater layoutInflater) {
    this.values = values;
    this.checked = new boolean[values.length];
    this.layoutInflater = layoutInflater;
  }

  public int getCount() {
    return values.length;
  }
  public Object getItem(int position) {
    return values[position];
  }
  public long getItemId(int position) {
    return 0;  // not stable
  }
  ...
```

Personaliser l'affichage

```
class ClassListAdapter extends BaseAdapter {
  private final Object[] values;
  private final boolean[] checked;
  private final LayoutInflater layoutInflater;

  public int getCount() {
    return values.length;
  }
  public Object getItem(int position) {
    return values[position];
  }

  public View getView(int position, View convertView, ViewGroup parent) {
    CheckedTextView view;
    if (convertView == null) {
      view = (CheckedTextView)layoutInflater.inflate(
        android.R.layout.simple_list_item_checked, parent, false);
    } else {
      view = (CheckedTextView)convertView;
    }
    view.setText(values[position].toString());
    view.setChecked(checked[position]);
    return view;
  }
```

On doit pour cela implanter getView()



Et comment réagir à la selection par l'utilisateur

Il faut implanter un controleur

- On écoute l'évènement OnItemClickListener
- On **délègue** au modèle pour qu'il se modifie
- Le modèle **signalera** alors une **modification** à la **vue**

Le controleur doit appeler le modèle

```
protected void onCreate(Bundle savedInstanceState) {
  ...
  Resources res = getResources();
  String[] classes = res.getStringArray(R.array.class_array);
  LayoutInflater layoutInflater = getLayoutInflater();
  final ClassListAdapter adapter =
       new ClassListAdapter(classes, layoutInflater);
  ListView listView = (ListView)findViewById(R.id.mylist);
  listView.setAdapter(adapter);
  listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    public void onItemClick(AdapterView<?> parent,
                                    View view, int position, long id) {
      adapter.check(position);
    }
  });
}
```

Modification du modèle et envoie d'une notification

```
class ClassListAdapter extends BaseAdapter {
  private final Object[] values;
  private final boolean[] checked;
  private final LayoutInflater layoutInflater;

  public int getCount() {
    return values.length;
  }
  public Object getItem(int position) {
    return values[position];
  }

  public void check(int position) {
    checked[position] = !checked[position];
    notifyDataSetChanged();
  }
}
```

Envoie d'une notification aux vues

Modèle mutable moins simple

On veut lorsque l'utilisateur clique sur la checkbox, n'afficher qu'une ligne sur 2

Modèle mutable stateless

En tant que classe interne de l'activité (Activity)

```
/* not static */ class MyAdapter extends BaseAdapter {
  private boolean onlyEven;
  private final LayoutInflater layoutInflater = getLayoutInflater();

  @Override
  public int getCount() {
    return (onlyEven)? 500: 1000;
  }

  @Override
  public Object getItem(int position) {
    return (onlyEven)? position * 2: position;
  }

  @Override
  public long getItemId(int position) {
    return position;
  }
```

Modèle mutable stateless

```
  @Override
  protected void onCreate(Bundle bundle) {
    ...
    final MyAdapter adapter = new MyAdapter();
    final CheckBox checked = (CheckBox)findViewById(R.id.checkBox);
    adapter.setOnlyEven(checked.isChecked());
    checked.setOnClickListener(new OnClickListener() {
      @Override
      public void onClick(View v) {
        adapter.setOnlyEven(checked.isChecked());
      }
  });
  setListAdapter(adapter);
}

/* not static */ class MyAdapter extends BaseAdapter {
  private boolean onlyEven;

  public void setOnlyEven(boolean onlyEven) {
    this.onlyEven = onlyEven;
    notifyDataSetChanged();
  }
```

Le controleur modifie le modèle

Le modèle notifie l'ensemble des vues enregistrées

Et pour changer la vue graphique

```
@Override
public View getView(int position, View convertView, ViewGroup parent) {
  LinearLayout view;
  if (convertView == null) {
    view = (LinearLayout)layoutInflater.inflate(
                    R.layout.list_item, parent, false);
  } else {
    view = (LinearLayout)convertView;
  }
  ((TextView)view.getChildAt(0)).setText(
      getItem(position).toString());
  ((TextView)view.getChildAt(1)).setText(
      "another text" + getItem(position).toString());
  return view;
 }
}
```

```
<LinearLayout
   xmlns:android="http://schemas.andro...."
   android:layout_width="fill_parent"
   android:layout_height="fill_parent"
   android:orientation="vertical" >

   <TextView android:id="@+id/text1"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:textIsSelectable="true"
      android:textSize="14sp"
      />
   <TextView android:id="@+id/text2"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:textIsSelectable="false"
      android:textStyle="italic"
      android:textSize="12sp"
      />
</LinearLayout>
```

Filtre & ids Stable

Un Liste Adapter possède la notion d'ID unique pour un item donné

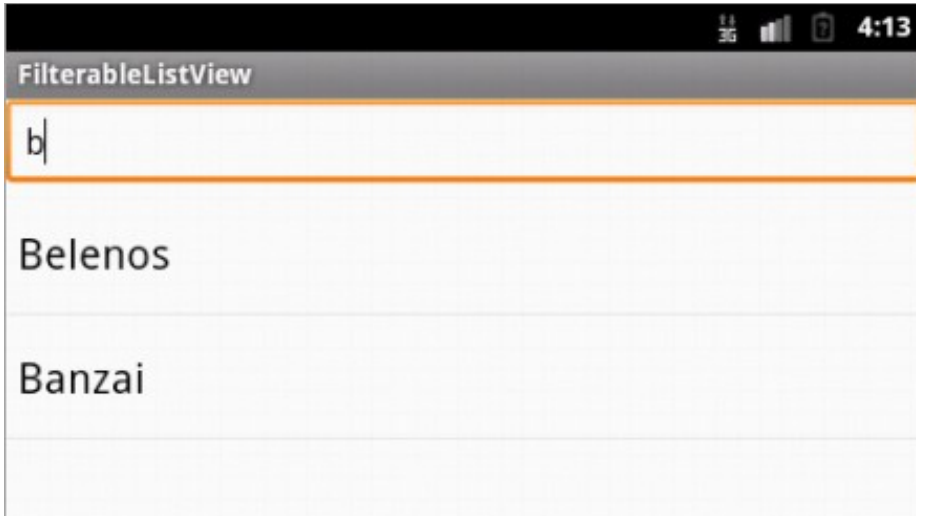
- boolean hasStableIds()
  - Indique so chaque item possède un ID unique (comme pour les BDDs)
    - Dans ce cas, la vue peut sauvegarder son état si l'activité meurt
    - Mais, il faut être capable de trouver un ID unique pour chaque Item
- long getItemId(int position)
  - Renvoie l'ID pour une position donnée

Filtre

La classe ArrayAdapter possède un mécanisme de filtre qui permet de filtrer suivant les premières lettres d'un mot donné

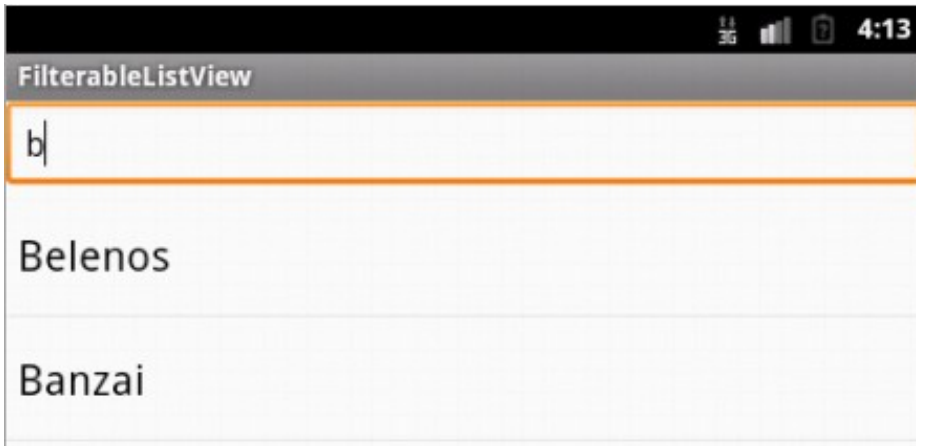
Adapter.getFilter().filter(startOfTheText)

L'implantation de ArrayAdapter.hasStableIds() renvoie false



Filtrer une ListView

```
public class MainActivity extends ListActivity {
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    final ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
          android.R.layout.simple_list_item_1,
          Arrays.asList("Belenos", "Toutatis", "CornoFulgur", "Banzai"));
    EditText filterText = (EditText) findViewById(R.id.filterEdit);
    filterText.addTextChangedListener(new TextWatcher() {
      public void afterTextChanged(Editable s) {
        // do nothing
      }
      public void beforeTextChanged(CharSequence s, int start, int count, int after) {
        // do nothing
      }
      public void onTextChanged(CharSequence s, int start, int before, int count) {
        adapter.getFilter().filter(s);
      }
    });
    setListAdapter(adapter);
  }
```



ListActivity

Raccourci dans le cas où l'activité n'est qu'une seule ListView

- Le layout est déjà fourni
  (la ListView est nommée @android:id/list)
  donc pas besoin d'appeler setContentView !
- Fournie une méthode setListAdapter(adapter) qui fournit l'adapter pour cette liste

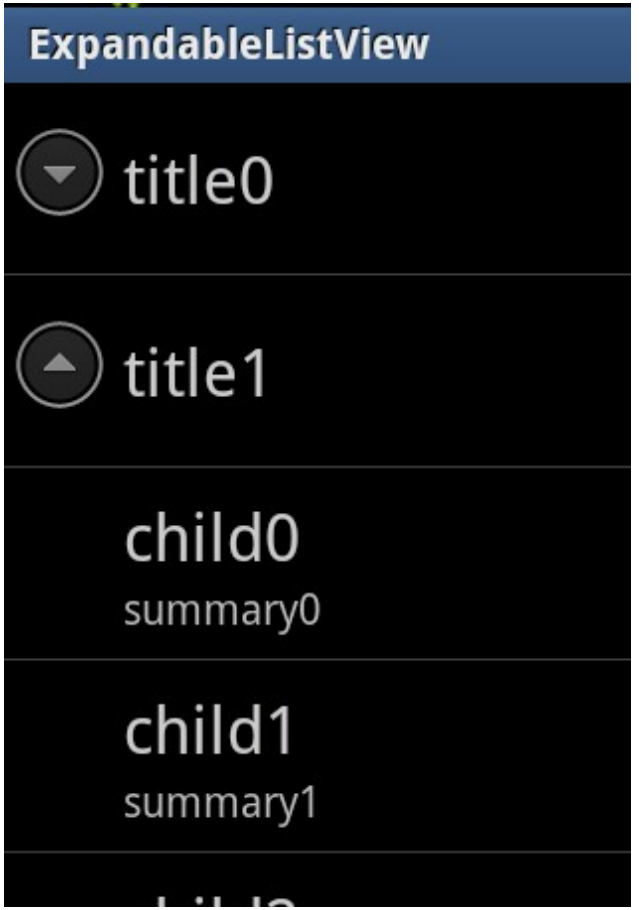
En héritant de ListActivity

On utilise directement setListAdapter() sur l'activité

```
public class MainActivity extends ListActivity {
  @Override
  protected void onCreate(Bundle bundle) {
    super.onCreate(bundle);
    setContentView(R.layout.activity_main);

    ListAdapter adapter = new ArrayAdapter<String>(this,
        android.R.layout.simple_list_item_1);
    adapter.add("foo");
    adapter.add("bar");
    this.setListAdapter(adapter);
```

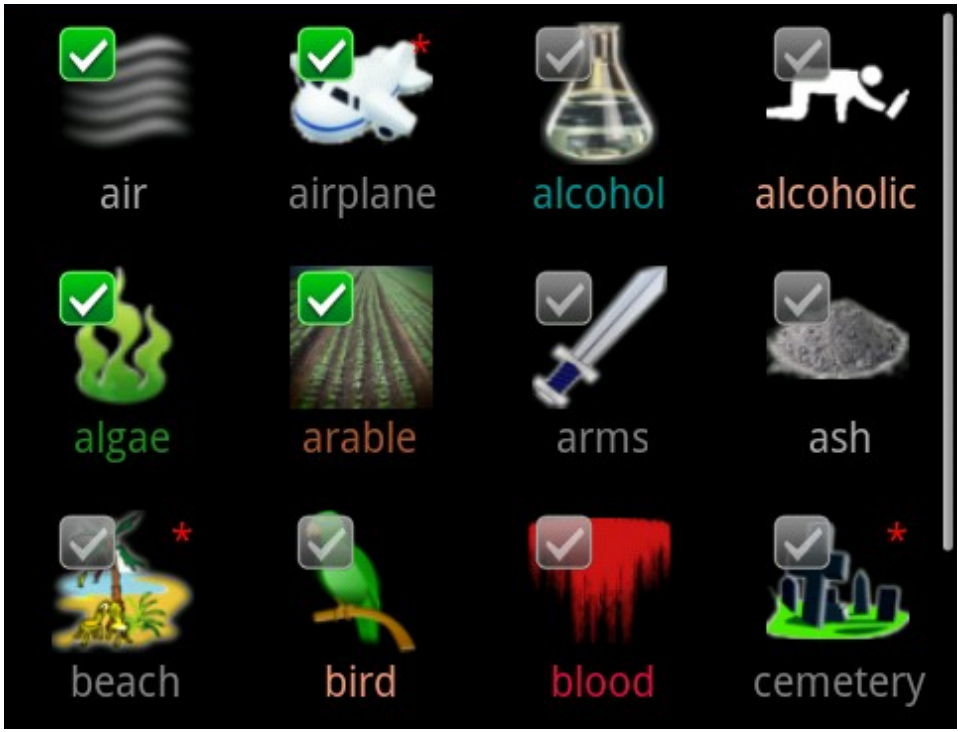
ExpandableListView

- Sous-classe de ListView (beurk)
- Permet deux niveaux de regroupement
- Le premier niveau contient un bouton qui permet d'afficher le second niveau



GridView

Un GridView permet de mettre des items sur une grille



Utilise aussi un ListAdapter

Un exemple de GridView

Un GridView en utilisant un fichier XML de Layout

```
<?xml version="1.0" encoding="utf-8"?>
<GridView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:columnWidth="90dp"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
 />
```

Largeur d'une colonne

Espaces entre les colonnes

Agrandissement

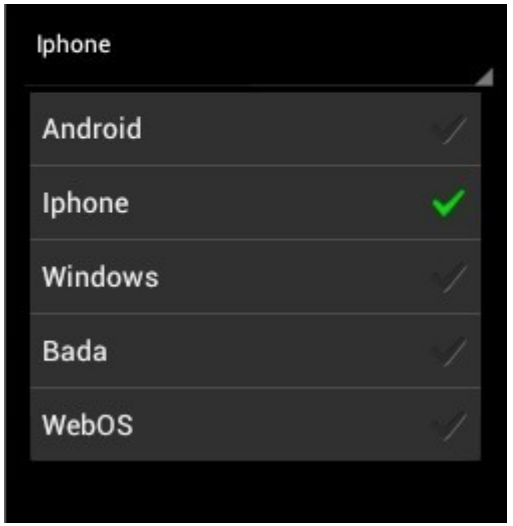
L'adapteur du GridView

```
public class MyGridViewActivity extends Activity {
  ...
  class ImageAdapter extends BaseAdapter {
    public int getCount() { return 10; }
    public Object getItem(int position) { return null; }
    public long getItemId(int position) { return 0; }

    public View getView(int position, View convertView, ViewGroup parent) {
        ImageView imageView;
        if (convertView == null) {  // if it's not recycled, initialize attributes
            imageView = new ImageView(MyGridViewActivity.this);
            imageView.setLayoutParams(new GridView.LayoutParams(60, 60));
            imageView.setScaleType(ImageView.ScaleType.CENTER_CROP);
            imageView.setPadding(8, 8, 8, 8);
        } else {
           imageView = (ImageView) convertView;
        }
        imageView.setImageResource(imageIds[position % imageIds.length]);
        return imageView;
    }

    private static final int[] imageIds = {
      R.drawable.image1, R.drawable.image2, R.drawable.image3
    };
```

Spinner

Un Spinner possède juste une méthode supplémentaire qui fournit la vue du bouton drop down

```
interface SpinnerAdapter extends Adapter {
    View getDropDownView(int position,
            View convertView,
            ViewGroup parent);
}
```



Et dans l'activité

On créé l'adapteur que l'on donne au Spinner

```
@Override
protected void onCreate(Bundle savedInstanceState){
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);

  Spinner spinner = (Spinner) findViewById(R.id.spinner);
  ArrayAdapter<CharSequence> adapter =
    ArrayAdapter.createFromResource(this,
      R.array.phones_array,
      android.R.layout.simple_spinner_item);
  adapter.setDropDownViewResource(
     android.R.layout.simple_spinner_dropdown_item);
  spinner.setAdapter(adapter);
```