SCIENCES SORBONNE UNIVERSITÉ

Lists

Renaultᵃlrde.epita.fr

Using Lists

The most used component in Android Applications

- Contacts
- Mails
- Pictures
- Social Network

Ready-to-use component

Model-View-Controller

- Model: the data to be displayed
- View: the GUI of the cell
- Controller: the relation between a data and its display

ListActivity

A ListActivity is the simplest component to build a list

- A unique ListView
- No need to define a layout for each cell
- Abstractions to manipulation the underlying list
  - **notifyDataSetChanged: datas have been modified, the GUI must be refreshed**
  - **notifyDataSetInvalidated: invalidate the underlying datas**

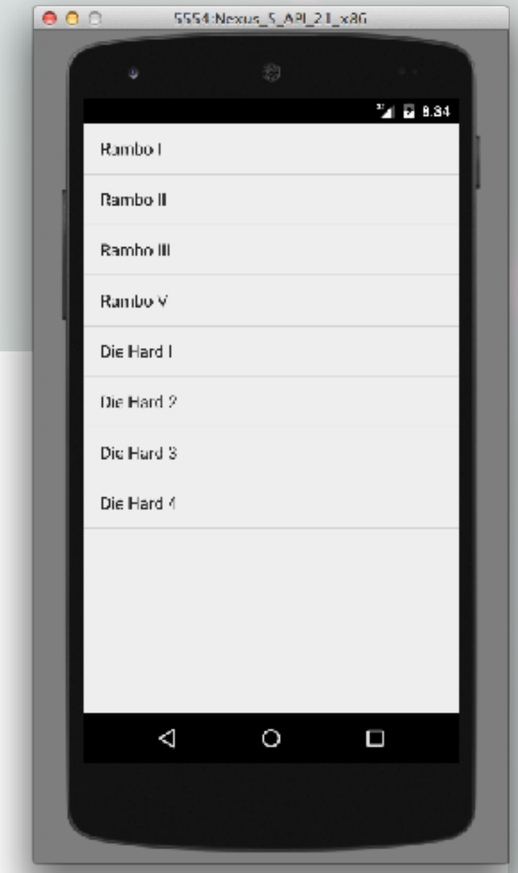
An Adapter is responsible of the management between the model and the view

- use a DataSetObserver to track modifications
- implement android.widget.adapter

ArrayAdapter

A ready-to-use adapter

- requirements: your data must be organized as an array
- Your data will be displayed using the toString method




```
public class MainActivity extends ListActivity {
   ArrayAdapter<String> adapter;
   String[] values = {
    "Rambo I", "Rambo II", "Rambo III", "Rambo V",
    "Die Hard I", "Die Hard 2", "Die Hard 3",
    "Die Hard 4"};

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        adapter = new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_1, values);
        setListAdapter(adapter);
    }
}
```

ListActivity and User Event

Detect a click

```
@Override
public void onListItemClick(ListView l, View v,
                            int position, long id) {
    // do something with the data
    Log.v("MainActivity", values[position]);
}
```

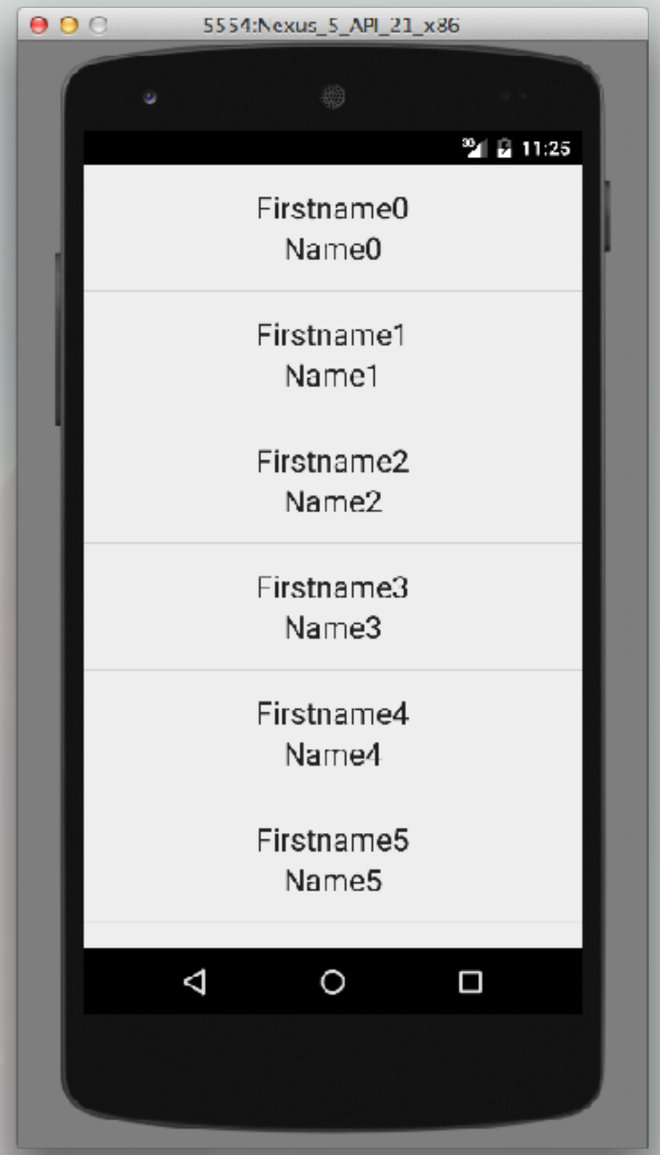
Detect a long click

```
getListView().setOnItemLongClickListener(
    new AdapterView.OnItemLongClickListener() {
      @Override
      public boolean onItemLongClick(AdapterView<?> parent,
                          View view, int position, long id) {
          return true;
      }
});
```

Slightly More complex Lists

- Handle lists with multiple text fields
- A GUI for each cell
- A specific adapter for matching the elements of the view

SimpleAdapter is the solution

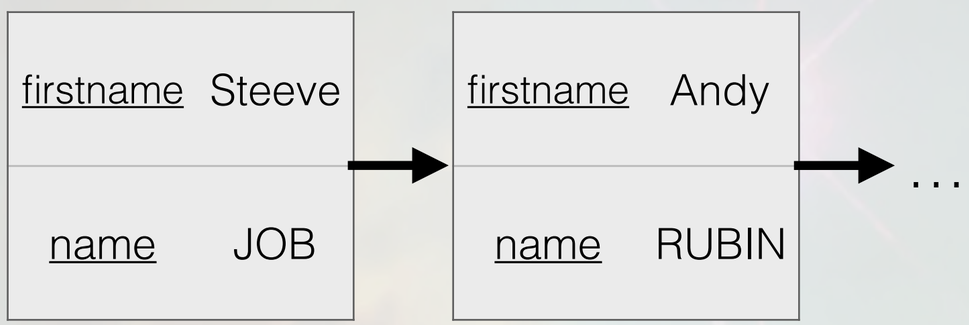


SimpleAdapter

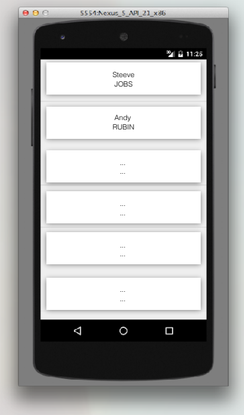
- Binding between static data and static GUI description
- Data must be a list of associative containers
- Each container is one row to display

Each container contains the description of a cell

INPUT



OUTPUT

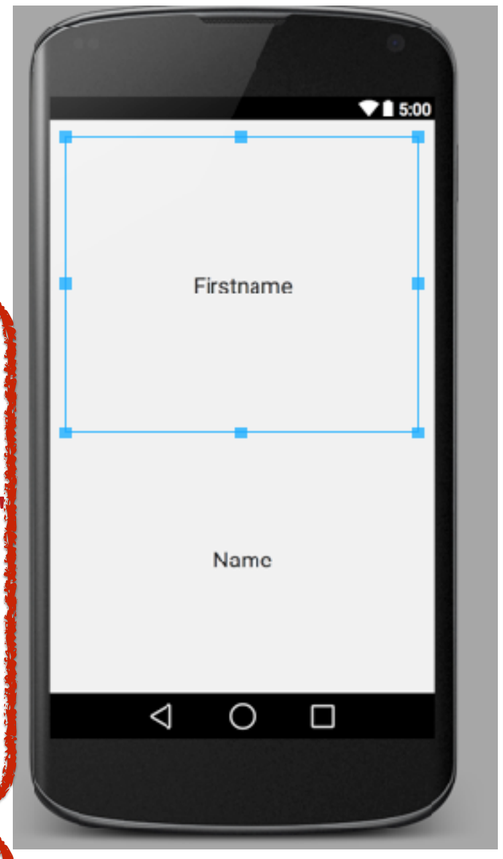


Define a Cell GUI


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

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Firstname"
        android:id="@+id/textView_firstname"
        android:layout_weight="0.50"
        android:gravity="center_vertical|center_horizontal" />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:textAppearance="?android:attr/textAppearanceLarge"
        android:text="Name"
        android:id=« @+id/textView_name"
        android:layout_weight="0.50"
        android:gravity="center_vertical|center_horizontal" />
</LinearLayout>
```



Populate the List


```
SimpleAdapter adapter;
ArrayList<HashMap<String, String>> data =
      new ArrayList<HashMap<String, String>>();
```

Declaration

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    for (int i = 0; i < 20; ++i)
        addItem("Name"+i, "Firstname"+i);
```

Prepare Data

```
    adapter = new SimpleAdapter(
            this,
            data,
            R.layout.cell_content,
            new String[]{"name", "firstname"},
            new int[]{R.id.textView_name, R.id.textView_firstname});
```

Instantiation

```
    setListAdapter(adapter);
}
```

setup

```
private void addItem(String record_name, String record_fname) {
    HashMap<String,String> item = new HashMap<String,String>();
    item.put("name", record_name);
    item.put("firstname", record_fname);
    data.add(item);
}
```

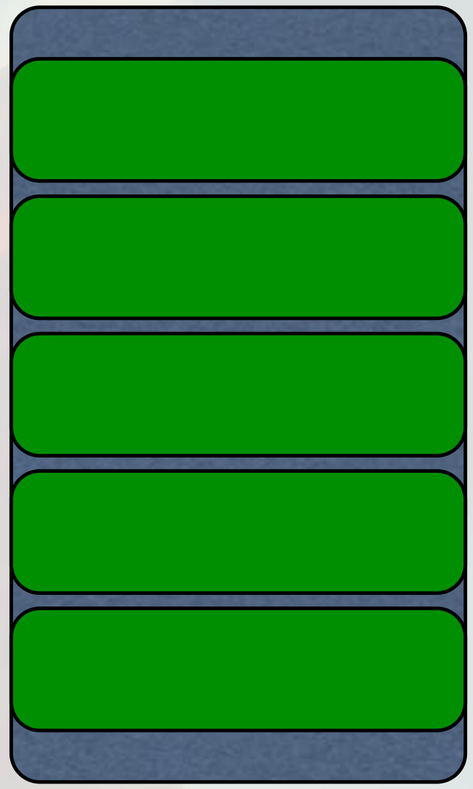
Prepare Data

More Complex Lists

Overload ArrayAdapter

Specify dynamically each cell

- getView(int position, View cV, View parent):
  - position: index of the element in the list
  - cV: reusable View, if different from null



Recycle Views:

- Avoid excessives allocations
- Reset the view before usage!

Define a complex Cell GUI (rowlayout)


```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="wrap_content" >

    <ImageView
        android:contentDescription="row"
        android:id="@+id/icon"
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:layout_marginLeft="6dp"
        android:layout_marginRight="6dp"
        android:layout_marginTop="5dp"
        android:layout_marginBottom="5dp"
        android:src="@drawable/ic_launcher_foreground" >
    </ImageView>

    <TextView
        android:id="@+id/label"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginTop="20dp"
        android:layout_marginBottom="20dp"
        android:layout_marginLeft="10dp"
        android:text="text"
        android:textSize="26sp" >
    </TextView>

</LinearLayout>
```

Overload ArrayAdapter


```
public class MyAdapter extends ArrayAdapter<String> {

    private Integer[] images = {
            R.drawable.one, R.drawable.two, R.drawable.three
    };

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater)
                getContext().getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View rowView = inflater.inflate(R.layout.rowlayout, parent, false);

        TextView textView = (TextView) rowView.findViewById(R.id.label);
        ImageView imageView = (ImageView) rowView.findViewById(R.id.icon);

        textView.setText(getItem(position));

        if(convertView == null )
            imageView.setImageResource(images[position]);
        else
            rowView = (View)convertView;

        return rowView;
    }

    public MyAdapter(Context context, String[] values) {
        super(context, R.layout.rowlayout, values);
    }
}
```

Import these ressources !

call the super constructor

Setup the Adaptator


```
public class MainActivity extends ListActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        String[] values = new String[]{ "one", "two", "three"};
        MyAdapter a = new MyAdapter(this, values);
        setListAdapter(adaptateur);
    }
}
```

Header for the view

Adding an Header for your list is easy

- Just define R.layout.header
- and use addHeaderView

```
ListView lv = getListView();
LayoutInflater inflater = getLayoutInflater();
View header = inflater.inflate(R.layout.header, lv, false);
lv.addHeaderView(header, null, false);
```

You can also use addFooterView to define a footer

Summary

A lot of adapter exit

- SimpleAdapter:
  - **static binding between data and GUI of a row**

- ArrayAdapter:
  - **If not overloaded, acts like a simple adapter**
  - **Otherwise can be specified to handle complex views**

- SimpleCursorAdapter and CursorAdapter
  - **Create lists from data base**

These Adapter works the same in every activity even if not ListActivity.

- You just have to grab the reference to the listView and apply setListAdapter