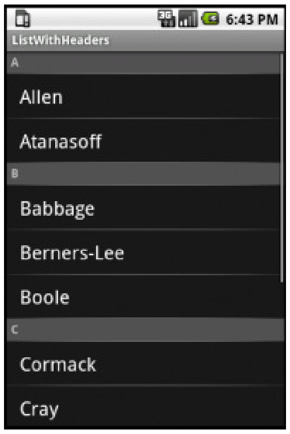
Just Another Android Blog

Home About

Alphabetically ordered ListView with labelled sections

Posted on October 5, 2010 by Lewis

ListView is the most complex widget in Android. I recently found myself needing to implement a ListView to show a list of contacts separated by alphabetical headers:



Surprisingly, header functionality isn't included in Android. A number of custom solutions exist:

http://github.com/commonsguy/cwac-merge

http://github.com/commonsguy/cw-advandroid/tree/master/ListView/Sections/

http://jsharkey.org/blog/2008/08/18/separating-lists-with-headers-in-android-09/

http://www.helloandroid.com/tutorials/how-add-divider-items-listview

In this example we'll see how I've used Jeff Sharkey's SeparatedListAdapter to implement a sorted list of labelled contacts.

We need four classes:

SeparatedListAdapter.java – a slightly modified version of Jeff Sharkey's
ListWithHeaders.java - our android activity
ListItemContainer.java – holds objects which implement the ListItemInterface
ListItemInterface.java - objects which implement this interface will be correctly sorted and labelled
ListItemObject.java – an example implementation of the ListItemInterface

Let's look at these classes in more detail:

ListItemInterface.java

Objects that implement this interface specify a header label for themselves. ListItem objects are also comparable – this is so they can be sorted in some order within their respective sections.

Archives

Select Month

```
public interface ListItemInterface extends Comparable {

    public String getLabel();
    public int compareTo(Object arg0);
}
```

ListItemObject.java

This is an example of an object which implements the ListItemInterface – the label method for this example simply returns the first character of the name field.

```
public class ListItemObject implements ListItemInterface {
    public String name;

    public ListItemObject(String name) {
        this.name = name;
    }

    public int compareTo(Object arg0) {
        ListItemObject c = (ListItemObject) arg0;
        return this.getName().compareTo(c.getName());
    }

    public String getName(){
        return name;
    }

    @Override
    public String toString() {
        return name;
    }

    @Override
    public String getLabel() {
        // TODO Auto-generated method stub
        return Character.toString(name.charAt(0));
    }

}
```

ListItemContainer.java

This class holds objects in an ArrayList, the key function of this class is to return a map of header labels to ArrayLists.

e.g.

"A" –> {Arnold,Archie}
"B" –> {Barry, Bob}

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

public class ListItemContainer<T extends ListItemInterface> {

    ArrayList<T> objects;

    public ListItemContainer(){
        objects = new ArrayList<T>();
    }

    public void addData(T c)
    {
        objects.add(c);
        Collections.sort(objects);
    }

    public Map<String, ArrayList<T>> getSortedData()
    {
        Map<String, ArrayList<T>> results = new HashMap<String, ArrayList<T>>
        ArrayList<T> contacts = null;
        String currletter = null;

        for (T c : objects)
        {
            if (!c.getLabel().equals(currletter))
            {
                contacts = new ArrayList<T>();
                currletter = c.getLabel();
                results.put(currletter,contacts);
                System.out.println("making new letter:" + currletter);
            }
            System.out.println("adding" + c.toString());
            contacts.add(c);
        }
        return results;

    }
}
```

ListWithHeaders.java

The makeAdapter() method takes the data returned by the ListItemContainer and iterates through it – creating and populating a new adapter for each key-value pair and adding the adapter to the SeparatedListAdapter.

```
import java.util.ArrayList;
import java.util.Iterator;
import java.util.Map;
import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;

public class ListWithHeaders extends ListActivity {

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

        ListItemContainer<ListItemObject> data = new ListItemContainer<ListIt
        data.addData(new ListItemObject("Babbage"));
        data.addData(new ListItemObject("Boole "));
        data.addData(new ListItemObject("Berners-Lee"));
        data.addData(new ListItemObject("Atanasoff "));
        data.addData(new ListItemObject("Allen"));
        data.addData(new ListItemObject("Cormack"));
        data.addData(new ListItemObject("Cray"));
        data.addData(new ListItemObject("Dijkstra "));
        data.addData(new ListItemObject("Dix"));
        data.addData(new ListItemObject("Dewey"));
        data.addData(new ListItemObject("Erdos"));

        Map<String, ArrayList<ListItemObject>> sortedContacts = data.getSorte
        SeparatedListAdapter adapter = this.makeAdapter(sortedContacts);

        this.setListAdapter(adapter);
        this.setContentView(R.layout.main);
    }

    public <T extends ListItemInterface> SeparatedListAdapter makeAdapter(Map
    {
        Iterator it = sortedObjects.entrySet().iterator();
        SeparatedListAdapter adapter = new SeparatedListAdapter(this);
        String label = null;
        ArrayList<T> section = new ArrayList<T>();
        while (it.hasNext()) {
            Map.Entry<String,ArrayList<T>> pairs = (Map.Entry<String,ArrayLis
            section =  pairs.getValue();
            label = pairs.getKey();
            adapter.addSection(label, new ArrayAdapter<T>(this, R.layout.list_
        }
        return adapter;
    }

}
```

You'll need to put these XML files in your res/layout folder:

```
<?xml version="1.0" encoding="utf-8"?>
<!-- list_item.xml -->
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/list_item_title"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:paddingTop="10dip"
    android:paddingBottom="10dip"
    android:paddingLeft="15dip"
    android:textAppearance="?android:attr/textAppearanceLarge"
    />

<?xml version="1.0" encoding="utf-8"?>
<!-- list_header.xml -->
<TextView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/list_header_title"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:paddingTop="2dip"
    android:paddingBottom="2dip"
    android:paddingLeft="5dip"
    style="?android:attr/listSeparatorTextViewStyle" />
```

SeparatedListAdapter.java – does all the hard work!

```
import java.util.LinkedHashMap;
import java.util.Map;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Adapter;
import android.widget.ArrayAdapter;
import android.widget.BaseAdapter;

    public class SeparatedListAdapter extends BaseAdapter {

        public final Map<String, Adapter> sections = new LinkedHashMap<String
        public final ArrayAdapter<String> headers;
        public final static int TYPE_SECTION_HEADER = 0;

        public SeparatedListAdapter(Context context) {
            headers = new ArrayAdapter<String>(context, R.layout.list_header)
        }

        public void addSection(String section, Adapter adapter) {
            this.headers.add(section);
            this.sections.put(section, adapter);
        }

        public Object getItem(int position) {
            for (Object section : this.sections.keySet()) {
                Adapter adapter = sections.get(section);
                int size = adapter.getCount() + 1;

                // check if position inside this section
                if (position == 0)
                    return section;
                if (position < size)
                    return adapter.getItem(position - 1);

                // otherwise jump into next section
                position -= size;
            }
            return null;
        }

        public int getCount() {
            // total together all sections, plus one for each section header
            int total = 0;
            for (Adapter adapter : this.sections.values())
                total += adapter.getCount() + 1;
            return total;
        }

        public int getViewTypeCount() {
            // assume that headers count as one, then total all sections
            int total = 1;
            for (Adapter adapter : this.sections.values())
                total += adapter.getViewTypeCount();
            return total;
        }

        public int getItemViewType(int position) {
            int type = 1;
            for (Object section : this.sections.keySet()) {
                Adapter adapter = sections.get(section);
                int size = adapter.getCount() + 1;

                // check if position inside this section
                if (position == 0)
                    return TYPE_SECTION_HEADER;
                if (position < size)
                    return type + adapter.getItemViewType(position - 1);

                // otherwise jump into next section
                position -= size;
                type += adapter.getViewTypeCount();
            }
            return -1;
        }
```

Posted in Android examples | Tagged alphabetical, Android, contacts, headers, ListView, ordering, sections | Leave a comment