falke2 карма 21,0 рейтинг 0,0
31 голос

29 ноября 2011 в 00:16

ListView в Android: Кастомизация списков

перевод

Разработка под Android*

Продолжение статьи о ListView в Android, в котором мы рассмотрим более сложные примеры его использования, такие, как иконки на элементах списка и добавление чекбоксов к этим элементам. Так же мы рассмотрим возможности по оптимизации кода.

Напомню, что статья является переводом этой статьи с разрешения ее автора.

Пример: ListActivity с собственным шаблоном.

Вы можете создать свой собственный шаблон для элементов списка и применить его к своему Адаптеру. Шаблон будет одинаковым для каждого элемента списка, но дальше мы разберем как сделать его более гибким. В нашем примере мы добавим иконку к каждому элементу списка.

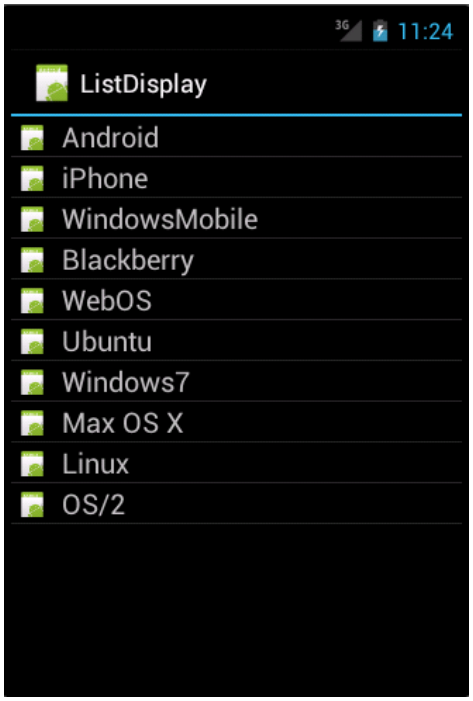
Создайте файл шаблона «rowlayout.xml» в папке res/layout вашего проекта «de.vogella.android.listactivity».

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

    <ImageView
        android:id="@+id/icon"
        android:layout_width="22px"
        android:layout_height="22px"
        android:layout_marginLeft="4px"
        android:layout_marginRight="10px"
        android:layout_marginTop="4px"
        android:src="@drawable/ic_launcher" >
    </ImageView>

    <TextView
        android:id="@+id/label"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@+id/label"
        android:textSize="20px" >
    </TextView>

</LinearLayout>
```

Измените свою Деятельность на следующую. Код почти такой же, как и в предыдущем примере, единственная разница в том, что мы используем наш собственный шаблон в ArrayAdapter и указываем адаптеру какой элемент пользовательского интерфейса будет содержать текст. Мы не делали этого в предидущей статье, поскольку мы использовали стандартный шаблон.

```
package de.vogella.android.listactivity;

import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MyListActivity extends ListActivity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        String[] values = new String[] { "Android", "iPhone", "WindowsMobile",
                "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
```

```
                "Linux", "OS/2" };
        // Использование собственного шаблона
        ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
                R.layout.rowlayout, R.id.label, values);
        setListAdapter(adapter);
    }

    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        String item = (String) getListAdapter().getItem(position);
        Toast.makeText(this, item + " selected", Toast.LENGTH_LONG).show();
    }
}
```



Пример: ListActivity с гибким шаблоном

Оба предыдущих примера используют один шаблон сразу для всех строк. Если вы хотите изменить вид определенных строк, вам нужно определить свой адаптер и заместить метод getView().

Этот метод ответственен за создание отдельных элементов вашего ListView. getView() возвращает Вид. Этот Вид фактически является Шаблоном (ViewGroup) и содержит в себе другие Виды, например, ImageView или TextView. С getView() вы так же можете изменить параметры индивидуальных видов.

Чтобы прочитать шаблон из XML в getView(), вы можете использовать системный сервис LayoutInflator.

В этом примере мы расширяем ArrayAdapter, но так же мы можем реализовать непосредственно BaseAdapter.

Определение простого адаптера

Очень просто создать свой Адаптер, не обращая внимания на его оптимизацию. Просто получайте в своей Деятельности данные, которые хотите отобразить и сохраняйте их в элемент списка. В вашем getView() установите ваш предопределенный шаблон для элементов и получите нужные вам элементы с помощью findViewById(). После этого вы можете определить их свойства.

Наш пример использует две картинки: «no.png» и «ok.png». Я положил их в папку «res/drawable-mdpi». Используйте свои картинки. Если не нашли таковых, то просто скопируйте «icon.png» и, с помощью графического редактора, немного измените их.

Создайте класс «MySimpleArrayAdapter», который будет служить нашим Адаптером.

```
package de.vogella.android.listactivity;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
```

```
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class MySimpleArrayAdapter extends ArrayAdapter<String> {
    private final Context context;
    private final String[] values;

    public MySimpleArrayAdapter(Context context, String[] values) {
        super(context, R.layout.rowlayout, values);
        this.context = context;
        this.values = values;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        LayoutInflater inflater = (LayoutInflater) context
                .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(values[position]);
        // Изменение иконки для Windows и iPhone
        String s = values[position];
        if (s.startsWith("Windows7") || s.startsWith("iPhone")
                || s.startsWith("Solaris")) {
            imageView.setImageResource(R.drawable.no);
        } else {
            imageView.setImageResource(R.drawable.ok);
        }

        return rowView;
    }
}
```

Чтобы использовать этот Адаптер, измените класс MyList на следующее

```
package de.vogella.android.listactivity;

import android.app.ListActivity;
import android.os.Bundle;

public class MyListActivity extends ListActivity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        String[] values = new String[] { "Android", "iPhone", "WindowsMobile",
                "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
                "Linux", "OS/2" };
        MySimpleArrayAdapter adapter = new MySimpleArrayAdapter(this, values);
        setListAdapter(adapter);
    }

}
```

Когда вы запустите это приложение, вы увидите список с элементами, с разными значками на некоторых из них.

Оптимизация производительности вашего собственного адаптера

Создание Java объектов для каждого элемента — это увеличение потребления памяти и временные затраты. Как уже говорилось, Андроид стирает элементы (виды) вашего списка, которые уже не отображаются и делегируют управление ими в метод getView() через параметр convertView.

Ваш Адаптер может использовать этот вид и избежать «раздутие» Шаблона для этого элемента. Это сохраняет память и уменьшает загрузку процессора.

В вашей реализации вы должны проверять convertView на наличие содержимого и переназначать его, отправляя

новые данные в существующий Шаблон, если convertView не пустой.

Наша реализация так же использует модель ViewHolder. Метод findViewById() достаточно ресурсоемок, так что нужно избегать его, если в нем нет прямой необходимости.

ViewHolder сохраняет ссылки на необходимые в элементе списка Шаблоны. Этот ViewHolder прикреплен к элементу методом setTag(). Каждый Вид может содержать примененную ссылку. Если элемент очищен, мы можем получить ViewHolder через метод getTag(). Это выглядит нагруженным, но, на самом деле, работает быстрее, чем повторяющиеся вызовы findViewById().

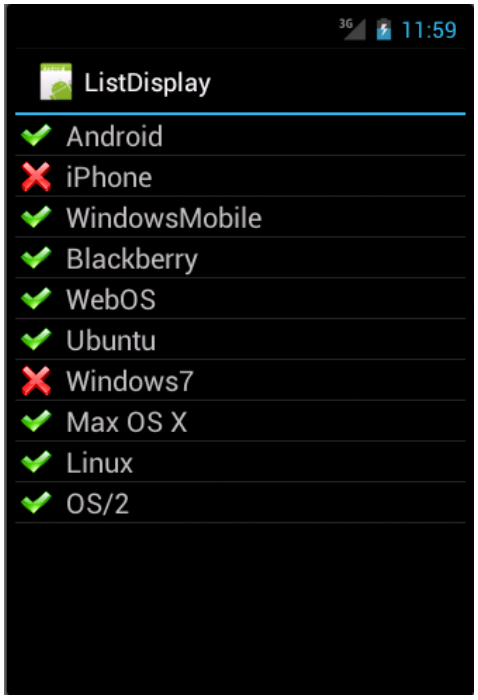
Обе техники (переназначение существующих видов и модель ViewHolder) увеличивают производительность примерно на 15%, особенно на больших объемах данных.

Продолжая использовать проект «de.vogella.android.listactivity», создайте класс «MyArrayAdapter.java».

```
package de.vogella.android.listactivity;

import android.app.Activity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.ImageView;
import android.widget.TextView;

public class MyArrayAdapter extends ArrayAdapter<String> {
    private final Activity context;
    private final String[] names;

    public MyArrayAdapter(Activity context, String[] names) {
        super(context, R.layout.rowlayout, names);
        this.context = context;
        this.names = names;
    }

    // Класс для сохранения во внешний класс и для ограничения доступа
    // из потомков класса
    static class ViewHolder {
        public ImageView imageView;
        public TextView textView;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        // ViewHolder буферизирует оценку различных полей шаблона элемента

        ViewHolder holder;
        // Очищает сущетсвующий шаблон, если параметр задан
        // Работает только если базовый шаблон для всех классов один и тот же
        View rowView = convertView;
        if (rowView == null) {
            LayoutInflater inflater = context.getLayoutInflater();
            rowView = inflater.inflate(R.layout.rowlayout, null, true);
            holder = new ViewHolder();
            holder.textView = (TextView) rowView.findViewById(R.id.label);
            holder.imageView = (ImageView) rowView.findViewById(R.id.icon);
            rowView.setTag(holder);
        } else {
            holder = (ViewHolder) rowView.getTag();
        }

        holder.textView.setText(names[position]);
        // Изменение иконки для Windows и iPhone
        String s = names[position];
        if (s.startsWith("Windows7") || s.startsWith("iPhone")
                || s.startsWith("Solaris")) {

            holder.imageView.setImageResource(R.drawable.no);
```

```
        } else {
            holder.imageView.setImageResource(R.drawable.ok);
        }

        return rowView;
    }
}
```

```
package de.vogella.android.listactivity;

import android.app.ListActivity;
import android.os.Bundle;

public class MyListActivity extends ListActivity {
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        String[] values = new String[] { "Android", "iPhone", "WindowsMobile",
                "Blackberry", "WebOS", "Ubuntu", "Windows7", "Max OS X",
                "Linux", "OS/2" };
        setListAdapter(new MyArrayAdapter(this, values));
    }

}
```



Продвинутые ListActivity

Обработка долгого нажатия на элементе

Вы так же можете добавить LongItemClickListener к виду. Для этого получите ListView через метод getListView() и определите обработку длительного нажатия через метод setOnItemLongClickListener().

```
package de.vogella.android.listactivity;

import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemLongClickListener;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.Toast;

public class MyList extends ListActivity {

/** Вызывается, если деятельность создана первый раз. */
```

```
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        // Создает массив строк для расположения их в ListActivity
        String[] names = new String[] { "Linux", "Windows7", "Eclipse", "Suse",
                "Ubuntu", "Solaris", "Android", "iPhone", "Linux", "Windows7",
                "Eclipse", "Suse", "Ubuntu", "Solaris", "Android", "iPhone" };
        ArrayAdapter<String> adapter = new TwoLayoutsArrayAdapter(this, names);
        setListAdapter(adapter);
        ListView list = getListView();
        list.setOnItemLongClickListener(new OnItemLongClickListener() {

            @Override
            public boolean onItemLongClick(AdapterView<?> parent, View view,
                    int position, long id) {
                Toast.makeText(MyList.this,
                        "Item in position " + position + " clicked",
                        Toast.LENGTH_LONG).show();
                // Возвращает "истину", чтобы завершить событие клика, чтобы
                // onListItemClick больше не вызывался
                return true;
            }
        });
    }

    @Override
    protected void onListItemClick(ListView l, View v, int position, long id) {
        super.onListItemClick(l, v, position, id);
        // Получение элемента, который был нажат
        Object o = this.getListAdapter().getItem(position);
        String keyword = o.toString();
        Toast.makeText(this, "You selected: " + keyword, Toast.LENGTH_SHORT)
                .show();

    }

}
```

Элементы, взаимодействующие с моделью данных

Ваш шаблон элемента списка так же может содержать Виды, взаимодействующие с моделью данных. Например, вы можете использовать Checkbox в элементе списка и, если чекбокс включен, вы можете менять данные, отображаемые в элементе.

Мы до сих пор используем тот же проект. Создайте шаблон элемента списка «rowbuttonlayout.xml».

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="wrap_content">
    <TextView android:text="@+id/label" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:id="@+id/label"
        android:textSize="30px"></TextView>
    <CheckBox android:id="@+id/check" android:layout_width="wrap_content"
        android:layout_height="wrap_content" android:layout_marginLeft="4px"
        android:layout_marginRight="10px" android:layout_alignParentRight="true"
        ></CheckBox>
</RelativeLayout>
```

создайте для этого примера класс Model, который содержит название элемента и его содержимое, если он чекнут.

```
package de.vogella.android.listactivity;

public class Model {

    private String name;
    private boolean selected;

    public Model(String name) {
        this.name = name;
```

```
        selected = false;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public boolean isSelected() {
        return selected;
    }

    public void setSelected(boolean selected) {
        this.selected = selected;
    }

}
```

Создайте следующий Адаптер. Этот Адаптер добавит обработку изменения Checkbox. Если чекбокс включен, то данные в модели тоже меняются. Искомый Checkbox получает свою модель через метод setTag().

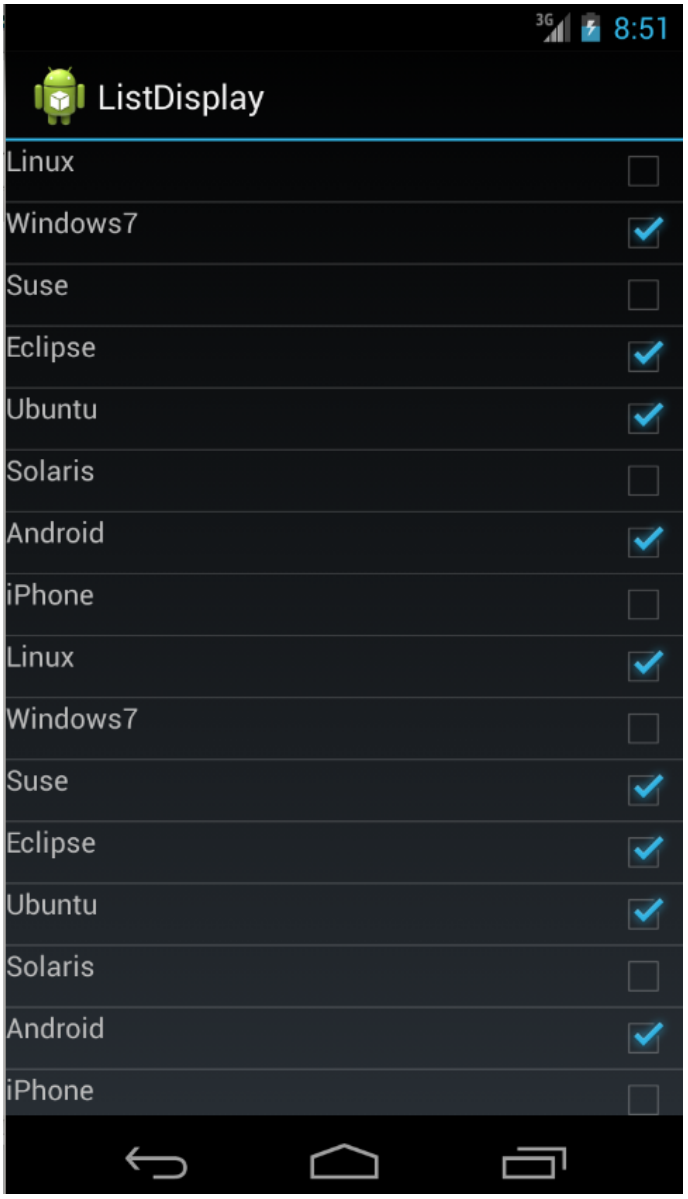
```
package de.vogella.android.listactivity;

import java.util.List;

import android.app.Activity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;

public class InteractiveArrayAdapter extends ArrayAdapter<Model> {

    private final List<Model> list;
    private final Activity context;

    public InteractiveArrayAdapter(Activity context, List<Model> list) {
        super(context, R.layout.rowbuttonlayout, list);
        this.context = context;
        this.list = list;
    }

    static class ViewHolder {
        protected TextView text;
        protected CheckBox checkbox;
    }

    @Override
    public View getView(int position, View convertView, ViewGroup parent) {
        View view = null;
        if (convertView == null) {
            LayoutInflater inflator = context.getLayoutInflater();
            view = inflator.inflate(R.layout.rowbuttonlayout, null);
            final ViewHolder viewHolder = new ViewHolder();
            viewHolder.text = (TextView) view.findViewById(R.id.label);
            viewHolder.checkbox = (CheckBox) view.findViewById(R.id.check);
            viewHolder.checkbox
                    .setOnCheckedChangeListener(new CompoundButton.OnCheckedChangeListener() {

                        @Override
                        public void onCheckedChanged(CompoundButton buttonView,
                                boolean isChecked) {
```

```
                            Model element = (Model) viewHolder.checkbox
                                    .getTag();
                            element.setSelected(buttonView.isChecked());

                        }
                    });
            view.setTag(viewHolder);
            viewHolder.checkbox.setTag(list.get(position));
        } else {
            view = convertView;
            ((ViewHolder) view.getTag()).checkbox.setTag(list.get(position));
        }
        ViewHolder holder = (ViewHolder) view.getTag();
        holder.text.setText(list.get(position).getName());
        holder.checkbox.setChecked(list.get(position).isSelected());
        return view;
    }
}
```

В завершение измените свой ListView на следующий.

```
package de.vogella.android.listactivity;

import java.util.ArrayList;
import java.util.List;

import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;

public class MyList extends ListActivity {


/** Вызывается, если деятельность создана первый раз. */

    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        // Создает массив строк, чтобы передать их в ListActivity
        ArrayAdapter<Model> adapter = new InteractiveArrayAdapter(this,
                getModel());
        setListAdapter(adapter);
    }

    private List<Model> getModel() {
        List<Model> list = new ArrayList<Model>();
        list.add(get("Linux"));
        list.add(get("Windows7"));
        list.add(get("Suse"));
        list.add(get("Eclipse"));
        list.add(get("Ubuntu"));
        list.add(get("Solaris"));
        list.add(get("Android"));
        list.add(get("iPhone"));
        // Первоначальный выбор одного из элементов
        list.get(1).setSelected(true);
        return list;
    }

    private Model get(String s) {
        return new Model(s);
    }

}
```

Когда вы запустите ваше приложение, вам будет доступна отметка элементов, которая будет отражаться на вашей модели.



Мультивыбор

Так же можно сделать одиночный и мультивыбор. Посмотрите следующие сниппеты для примера. Чтобы получить выбранные элементы используйте listView.getCheckedItemPosition() или listView.getCheckedItemPositions(). Вы так же можете использовать listView.getCheckedItemIds(), чтобы получить ID выбранных элементов.

```
package de.vogella.android.listactivity;

import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MyList extends ListActivity {


/** Вызывается, если деятельность создана первый раз. */

    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        // Создает массив строк, которые передаются в ListActivity
        String[] names = new String[] { "Linux", "Windows7", "Eclipse", "Suse",
                "Ubuntu", "Solaris", "Android", "iPhone", "Linux", "Windows7",
                "Eclipse", "Suse", "Ubuntu", "Solaris", "Android", "iPhone" };
        setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_multiple_choice,
                android.R.id.text1, names));
        ListView listView = getListView();
        listView.setChoiceMode(ListView.CHOICE_MODE_MULTIPLE);
```

```
    }

}
```

```
package de.vogella.android.listactivity;

import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MyList extends ListActivity {


/** Вызывается при первом создании деятельности. */

    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        // Создает массив строк, которые передаются в ListActivity
        String[] names = new String[] { "Linux", "Windows7", "Eclipse", "Suse",
                "Ubuntu", "Solaris", "Android", "iPhone", "Linux", "Windows7",
                "Eclipse", "Suse", "Ubuntu", "Solaris", "Android", "iPhone" };
        setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_single_choice,
                android.R.id.text1, names));
        ListView listView = getListView();
        listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
    }

}


package de.vogella.android.listactivity;

import android.app.ListActivity;
import android.os.Bundle;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MyList extends ListActivity {


/** Создается при первом создании деятельности. */

    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        // Создает массив строк, которые передаются в ListActivity
        String[] names = new String[] { "Linux", "Windows7", "Eclipse", "Suse",
                "Ubuntu", "Solaris", "Android", "iPhone", "Linux", "Windows7",
                "Eclipse", "Suse", "Ubuntu", "Solaris", "Android", "iPhone" };
        setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_single_choice,
                android.R.id.text1, names));
        ListView listView = getListView();
        listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE);
    }

}
```

Хедер и Футер

Вы можете поместить произвольные элементы вокруг своего списка. Например, вы можете создать шаблон со списком между двумя TextView. Если вы так сделаете, то вы должны указать id "android:id/list" к ListView, т.к. ListActivity ищет Вид с таким идентификатором. В таком случае один TextView всегда будет видимым над ListView (Хедер), а другой будет виден внизу. Если вы хотите использовать Футер и Хедер только в конце/начале списка,

чтобы они не были фиксированными, то нужно использовать view.setHeaderView() или view.setFooterView(), например:

```
package de.vogella.android.listactivity;

import android.app.ListActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.ListView;

public class MyList extends ListActivity {



/**Вызывается при первом создании активности. */

    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        // Создает массив строк, которые передаются в ListActivity
        String[] names = new String[] { "Linux", "Windows7", "Eclipse", "Suse",
                "Ubuntu", "Solaris", "Android", "iPhone", "Linux", "Windows7",
                "Eclipse", "Suse", "Ubuntu", "Solaris", "Android", "iPhone" };
        View header = getLayoutInflater().inflate(R.layout.header, null);
        View footer = getLayoutInflater().inflate(R.layout.footer, null);
        ListView listView = getListView();
        listView.addHeaderView(header);
        listView.addFooterView(footer);
        setListAdapter(new ArrayAdapter<String>(this,
                android.R.layout.simple_list_item_single_choice,
                android.R.id.text1, names));

    }
}
```

SimpleCursorAdapter

Если вы работаете с базой данных или же с контентом непосредственно, вы можете использовать SimpleCursorAdapter, чтобы перенести данные в ваш ListView.

Создайте новый проект «de.vogella.android.listactivity.cursor» с деятельностью «MyListActivity». Создайте такую деятельность.

```
package de.vogella.android.listactivity.cursor;

import android.app.ListActivity;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.provider.ContactsContract;
import android.widget.ListAdapter;
import android.widget.SimpleCursorAdapter;

public class MyListActivity extends ListActivity {

/** Вызывается при первом создании деятельности */

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        Cursor mCursor = getContacts();
        startManagingCursor(mCursor);
        // Создание нового адаптера для списка, связанного с курсором
        // SimpleListAdapter создан для определения курсора.
        ListAdapter adapter = new SimpleCursorAdapter(this, // Связь.
                android.R.layout.two_line_list_item, // Определения шаблона элемента
                mCursor, // Переход к курсору, который надо запомнить.
                // Массив курсоров, которые надо запомнить.
                new String[] { ContactsContract.Contacts._ID,
                        ContactsContract.Contacts.DISPLAY_NAME },
```

```
                // Массив, связывающий запомненные курсоры и шаблоны с ними связанные
                new int[] { android.R.id.text1, android.R.id.text2 });

        // Запоминаем адаптер.
        setListAdapter(adapter);
    }

    private Cursor getContacts() {
        // Выполняем запрос
        Uri uri = ContactsContract.Contacts.CONTENT_URI;
        String[] projection = new String[] { ContactsContract.Contacts._ID,
                ContactsContract.Contacts.DISPLAY_NAME };
        String selection = ContactsContract.Contacts.IN_VISIBLE_GROUP + " = '"
                + ("1") + "'";
        String[] selectionArgs = null;
        String sortOrder = ContactsContract.Contacts.DISPLAY_NAME
                + " COLLATE LOCALIZED ASC";

        return managedQuery(uri, projection, selection, selectionArgs,
                sortOrder);
    }

}
```

Убедитесь, что вы дали приложению доступ к контактам. (Используйте «android.permission.READ_CONTACTS» в AndroidManifest.xml).

Спасибо за внимание. Комментарии и поправки к переводу приветствуются, т.к. даже в исходнике встречаются ошибки и опечатки.

Прошу прощения за репост, изначально не отметил как перевод, поскольку недавно здесь. Большое спасибо за наводку jeston, принял к сведению и научился на ошибках.

свой шаблон ListView, ListView и чекбоксы, сложный список, оптимизация ListView

+29 104088 287 Lars Vogel falke2 0,0

Комментарии (16)

НЛО прилетело и опубликовало эту надпись здесь

falke2 29 ноября 2011 в 17:53 # ↑ 0

Сигнал к тому, чтобы перманентно переводить статьи с этого ресурса? :)

НЛО прилетело и опубликовало эту надпись здесь

falke2 29 ноября 2011 в 18:04 # ↑ 0

Отписал Ларсу с просьбой, чтобы знал :) Посмотрим, что ответит, а то как-то некрасиво получается, что он не знает об этом :)
Думаю, предполагается положительный ответ.

falke2 29 ноября 2011 в 18:49 # ↑ +3

Ларс одобряет, собирается твитнуть :)

iAmGeorge 29 ноября 2011 в 21:08 # +2

Этот перевод ужасен

falke2 29 ноября 2011 в 21:16 # ↑ +1

Прошу прощения, буду совершенствоваться в будущем.

ArtRoman 1 декабря 2011 в 03:06 # ↑ 0

Больше всего смущает Деятельность, и немного орфография. Да и перевод слишком в лоб, словно от гугл-переводчика.

falke2 1 декабря 2011 в 09:28 # ↑ 0

Ну а как еще перевести Activity? Думаю, в дальнейшем такого рода слова не буду переводить. Ну и да, не слишком художественно получилось техдок переводить :)

ArtRoman 1 декабря 2011 в 21:02 # ↑ 0

Никак. Активити есть активити, у разработчиков это на слуху. Вы же не переводили TextView, ListView. Название класса Model оставили. Activity — это тоже класс, но базовый. Хедер и Футер тоже под вопросом, но это ещё хотя бы привычно.

ArtRoman 1 декабря 2011 в 21:09 # ↑ 0

Опять же, о художественности тут никто не говорит, всё-таки, действительно, техдок)
Но многие фразы просто некорректны, например, «Так же можно сделать одиночный и мультивыбор». Можно кого сделать одиночный? Мультивыбор — множественный выбор. И тут лучше не “и”, а “или” (сделать оба сразу не получится). А можно просто сказать по-человечески: Вы можете выбрать один или несколько элементов. В общем, автосгенерированный Javadoc это одно, а статья пишется несколько иным языком. Но это всё не по теме блога, можно продолжить в личке.

evgajukov 4 мая 2012 в 17:39 # 0

А как поступить, если в списке нужно элементы группировать по определенным свойствам и для каждой группы выводить в этом списке заголовок с элементами?
Например:
есть задача отображать список встреч следующим образом:
*** Сегодня**
— задача 1
— задача 2
— …
— задача N
*** Завтра**
— задача N+1
— задача N+2
— и т.д.

debugger88 4 мая 2012 в 17:50 # ↑ 0

ExpandableListView

evgajukov 4 мая 2012 в 17:55 # ↑ 0

О! Большое спасибо!

evgajukov 4 мая 2012 в 17:54 # 0

О! Большое спасибо!

Severovostok 30 октября 2012 в 23:53 # 0

Быть может я что-то не так понял, но в статье сказано: «15 % faster then using the findViewById() method», а тут указано «увеличивают производительность примерно на 175%, особенно на больших объемах данных».

Только зарегистрированные пользователи могут оставлять комментарии. Войдите, пожалуйста.