Модуль 4. Алгоритмы и структуры данных

Тема 4.3. Адаптеры в Android

2 часа

Оглавление

4.3. Адаптеры в Android

Одним из наиболее часто используемых элементов интерфейса является **список**. Их реализация при создании Android приложений имеет ряд особенностей: для вывода списка используется компонент `ListView`, а чтобы определить его содержание и структуру, в которой они будут храниться - свяанный с ним адаптер.

Сразу возникает вопрос — «Зачем так сложно реализовано? Почему бы просто не передать данные ListView напрямую?» Ответ очень простой: «Чтобы сохранить драгоценную оперативную память». Дело в том, что мобильные устройства обладают малым количеством оперативной памяти, а операционная система Android использует всю имеющуюся (не важно сколько её на устройстве), так как она построена вокруг виртуальной машины Java, которая сама решает, когда очищать ресурсы. Причем если виртуальная машина Java посчитает, что ваше приложение использует много ресурсов, то она его закроет без спроса.

Программист может реализовать свой адаптер, но чаще всего используются готовые адаптеры. Базовый адаптер **BaseAdapter** и его потомки [1]:

- **ArrayAdapter<T>** - предназначен для работы с ListView. Данные представлены в виде массива, которые размещаются в отдельных элементах TextView;
- **ListAdapter** - адаптер между ListView и данными. Строго говоря, это класс-интерфейс, который можно использовать и в ArrayAdapter и в SimpleAdapter и т.д.;
- **SpinnerAdapter** - адаптер для связки данных с элементом Spinner. Это тоже интерфейс, как ListAdapter и работает по схожему принципу;
- **SimpleAdapter** - адаптер, позволяющий заполнить данными список более сложной структуры, например, два текста в одной строке списка;
- **SimpleCursorAdapter** - дополняет ResourceCursorAdapter и создаёт компоненты TextView/ImageView из столбцов, содержащихся в курсоре. Компоненты определяются в ресурсах;
- **CursorAdapter** - предназначен для работы с ListView, предоставляет данные для списка через курсор, который должен иметь колонку с именем "_id";
- **ResourceCursorAdapter** - этот адаптер дополняет CursorAdapter и может создавать виды из ресурсов;
- **HeaderViewListAdapter** - расширенный вариант ListAdapter, когда ListView имеет заголовки.
- **WrapperListAdapter** - еще один адаптер для списков.

4.3.1. Адаптер ArrayAdapter

Рассмотрим первый простой приемр, который объяснит многое из вышесказанного.

Требуется вывести название двенадцати месяцев на экран устройства в виде списка. Реализуем это с помощью адаптера ArrayAdapter и ListActivity.

Самым простым способом вывода списка на экран, является его вывод через ListActivity. В этом случае создаётся активность (файл разметки создавать не нужно), внутри которой на всю поверхность растянут компонент ListView. Внутри конструктора активности происходит заполнение списка. Заполнять список можно любыми адаптерами (SimpleAdapter, ArrayAdapter, ListAdapter и т. д.). Использовать ListActivity удобно тогда, когда вам нужно вывести на экран исключительно список, без других элементов интерфейса.

Как уже было сказано, ArrayAdapter предназначен для работы с ListView и данные представляет в виде массива, каждый элемент которого размещается в отдельном элементе TextView.

Списки выводятся с помощью ListView, а провайдерами (поставщиками) контента выступают

адаптеры, которые готовят элементы списков для вывода на экран.

Создадим новый проект , родителем главной активности назначим ListActivity.

Далее создадим массив в котором будут храниться месяцы:

```
String[] myArr = {"Январь", "Февраль", "Март", "Апрель", "Май", "Июнь",
        "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"};
```

Всё готово для вывода нашего списка на экран!

Делаем следующее:

1) Создаём и заполняем ArrayAdapter;

2) Назначаем адаптер.

Получаем следующий код который располагается внутри onCreate нашей активности:

```
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 MainActivity extends ListActivity {

   String[] myArr = {"Январь", "Февраль", "Март", "Апрель", "Май", "Июнь",
           "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"};

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

       ArrayAdapter<String> monthAdapter = new ArrayAdapter<String>(this,
android.R.layout.simple_list_item_1, myArr);

       setListAdapter(monthAdapter);
   }
}
```

Сохраняем и запускаем приложение - видим наш список на экране устройства (рисунок 1):

Рисунок 1

Можете поводить пальцем по экрану и, если все элементы на экран не вошли, вы увидите как смещаются элементы списка вверх или вниз.

А теперь ответ на главный вопрос «Зачем?».

Обратимся к снимку экрана (рисунок 1). Как вы можете заметить, на нем изображены всего 6 элементов списка. На самом деле система создала 7 визуальных экземпляров списка. При передвижении списка вверх или вниз система не создает новые элементы. Она использует уже созданные. Представим, что пользователь начал водить по экрану пальцем и необходимо вывести новые элементы списка. Запускается следующий механизм - верхние элементы скрываются с экрана, заполняются новыми данными и отображаются внизу экрана. Если пронумеровать все визуализированные элементы, то можно получить следующую картину подмены (рисунок 2):

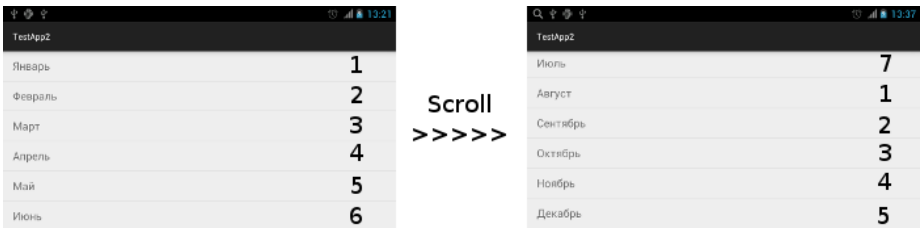

Рисунок 2

Не важно сколько элементов в вашем массиве (списке), который нужно вывести на экран. Будет создано столько визуальных элементов списка, сколько вмещается на экран +1. Если бы этого механизма не существовало, то для 1000 000 текстовых элементов массива системе пришлось бы создать 1000 000 экземпляров TextView для их отображения. Это повлекло бы крах приложения, так как память тут же бы закончилась (конечно, если на вашем устройстве нет нескольких лишних гигабайт оперативной памяти). А в нашем случае создаётся несколько визуальных экземпляров и, собственно, массив под эти 1000 000 элементов! Подмена и перезаполнение визуальных элементов происходит настолько быстро, что пользователь этого никогда не заметит.

Как же теперь отследить пункт на который нажал пользователь? Для этих целей в классе ListActivity реализован метод (слушатель) — onListItemClick. Для нашего примера можно написать следующий код слушателя, который будет отображать значение нажатого элемента:

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

Теперь при нажатии на какой либо элемент на экране будет появляться название выбранного месяца. Обратите внимание на метод getListAdapter(), с его помощью осуществляется доступ к адаптеру с помощью которого осуществляется вывод информации на экран. С его помощью также можно получить доступ к данным, которые выводятся на экран, в частности с помощью метода `getItem`.

Если захочется динамически изменить элемент отображаемого массива, используйте следующий код:

```
monthArr[11] = "!TEST!";
monthAdapter.notifyDataSetInvalidated();
```

С помощью метода «*notifyDataSetInvalidated*» вы информируете адаптер о том, что элементы массива изменились и требуется обновить их отображение. Добавьте этот код в слушатель клика и посмотрите что произойдёт!

Задание 4.3.1

Разработать приложение для отображению списка планет солнечной системы. При нажатии на элемент списка происходит отображение названия планеты выбранного элемента. Внизу экрана должна быть кнопка, при нажатии на которую в список планет будет добавляться новая планета с названием «Неизвестная планета».

4.3.2. SimpleAdapter

Что делать, если необходимо вывести более сложный список элементов? Например, с использованием нескольких надписей, изображениями и переключателями?

В этом случае можно использовать готовый адаптер SimpleAdapter. Работу с ним можно разделить на следующие этапы:

1. Создание разметки;
2. Создание списка данных для отображения в адаптере. Элементы списка состоят из объектов типа HashMap (ключ - значение);
3. Создание массива атрибутов для сопоставления элементов списка с элементами разметки;
4. Создание массива идентификаторов разметки;
5. Создание самого адаптера.

Усложним предыдущий пример: каждый элемент списка будет иметь кроме названия его описание. Начнем:

1) В основном файле разметки остается:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
        xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <ListView
        android:id="@+id/listView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
            android:layout alignParentBottom="true"
        android:layout_alignParentEnd="true"
        android:layout_alignParentLeft="true"
        android:layout_alignParentRight="true"
        android:layout_alignParentStart="true"
        android:layout_alignParentTop="true" />

</RelativeLayout>
```

Создадим разметку для адаптера в новом файле adapter_item.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="match_parent"
        android:orientation="vertical" >

        <ImageView
        android:id="@+id/imageView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" />

        <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Месяц"
        android:textAppearance="?android:attr/textAppearanceLarge" />

        <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Среднесуточная температура"
        android:textAppearance="?android:attr/textAppearanceMedium" />

        <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Количество дней"
         android:textAppearance="?android:attr/textAppearanceMedium" />

        <CheckBox
        android:id="@+id/checkBox"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:checked="true"
        android:text="Мне нравится этот месяц" />

</LinearLayout>
```

Мы получим следующий вид (рисунок 3):

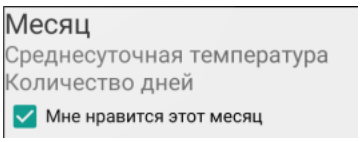

Рисунок 3

2) Создание списка данных:

```
       // Названия месяцев
        String[] monthArr = { "Январь", "Февраль", "Март", "Апрель", "Май",
"Июнь",
                        "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь",
"Декабрь" };
        // Среднесуточная температура
        String[] tempArr = { "-12.7", "-11.3", "-4.5", "7.7", "19.3",
```

```
"23.9",
                        "23.5", "22.8", "16.0", "5.2", "-0.3", "-9.3" };
        // Количество дней
        String[] dayCArr = { "31", "28", "31", "30", "31", "30", "31", "31",
"30",
                        "31", "30", "31" };

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

                // Готовим данные
                ArrayList<HashMap<String, Object>> data =
                        new ArrayList<HashMap<String,
Object>>(monthArr.length);
                HashMap<String, Object> map;

                for (int i = 0; i < monthArr.length; i++) {
                        map = new HashMap<String, Object>();
                        map.put("month", monthArr[i]);
                        map.put("temp", tempArr[i]);
                        map.put("day", dayCArr[i]);
                        map.put("like", true);
                        map.put("img", R.drawable.sun);
                        data.add(map);
                }

        }
```

В папку res/drawable сохраняем файл sun с изображением солнышка.

3) Создание массива атрибутов:

```
         // Атрибуты элементов
          String[] from = { "month", "temp", "day", "like", "img" };
```

4) Создание массива идентификаторов разметки:

```
       // Идентификаторы
        int[] to = { R.id.textView, R.id.textView2, R.id.textView3,
R.id.checkBox, R.id.imageView };
```

5) Создание адаптера с заполненными данными и назначение списку:

```
       // Создание адаптера
        SimpleAdapter adapter = new SimpleAdapter(this, data,
                R.layout.adapter_item, from, to);
        ListView lv = (ListView) findViewById(R.id.listView);
        lv.setAdapter(adapter);
```

В представленном примере на экран выводится список месяцев с указанием названия, среднесуточной температурой, количества дней в неделе, изображения и отметки «нравится месяц»:

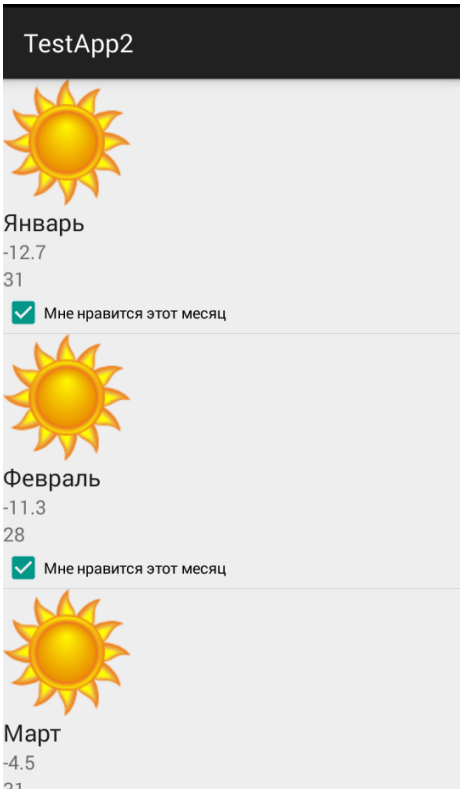

Рисунок 4

В итоге весь код `MainActivity` будет выглядеть так:

```
// Названия месяцев
String[] monthArr =
{"Январь","Февраль","Март","Апрель","Май","Июнь","Июль","Август","Сентябрь",
"Октябрь","Ноябрь","Декабрь"};
// Среднесуточная температура
String[] tempArr = {"-12.7","-11.3","-
4.5","7.7","19.3","23.9","23.5","22.8","16.0","5.2","-0.3","-9.3"};
// Количество дней
String[] dayCArr =
{"31","28","31","30","31","30","31","31","30","31","30","31"};

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

        // Готовим данные
        ArrayList<HashMap<String, Object>> data = new
ArrayList<HashMap<String, Object>>(monthArr.length);
        HashMap<String, Object> map;
        for (int i = 0; i < monthArr.length; i++) {
        map = new HashMap<String, Object>();
        map.put("month", monthArr[i]);
        map.put("temp", tempArr[i]);
        map.put("day", dayCArr[i]);
        map.put("like", true);
        map.put("img", R.drawable.sun);
```

```
        data.add(map);
        }
        // Атрибуты элементов
        String[] from = { "month", "temp", "day", "like", "img" };
        // Идентификаторы
        int[] to = { R.id.textView, R.id.textView2, R.id.textView3,
R.id.checkBox, R.id.imageView };
        // Создание адаптера
        SimpleAdapter adapter = new SimpleAdapter(this, data,
R.layout.adapter_item, from, to);

        ListView lv = (ListView) findViewById(R.id.listView);
        lv.setAdapter(adapter);
}
```

Тема адаптеров большая и интересная. Например, вы можете всегда реализовать собственный адаптер на основе ArrayAdapter.

Задание 4.3.2

Разработайте приложение, в результате работы которого на экран будет выведена информация о всех командах выигравших «Кубок Гагарина» с момента основания КХЛ. Обязательно укажите название команды, герб команды, сезоны победы, родной город команды. Для вывода информации используйте SimpleAdapter.

4.3.3 Собственный адаптер на основе ArrayAdapter

Мы разобрались как работать с двумя видами адаптеров, а теперь давайте научимся полностью контролировать работу адаптера. Возьмём под контроль ArrayAdapter, будем сами назначать, что ему показывать и где [3]. В новом примере адаптер будет снова показывать погоду, но на этот раз сохранять состояние CheckBox и для месяцев с отрицательной температурой будет выводиться другая картинка из файла snow. Разметка адаптера остаётся та же.

На этот раз для хранения параметров каждого месяца будем использовать отдельный класс MyMonth. Создайте новый файл класса:

```
public class MyMonth {
   String month = "";    // Название месяца
   double temp = 0.;     // Средняя температура
   int days = 0;         // Количество дней
   boolean like = true; // Нравится месяц
}
```

Для заполнения массива месяцев (данные которого будут выводиться с помощью адаптера), в главной активности, создадим метод создания массива из объектов *MyMonth*. Конструктор активности остаётся практически без изменений. Код главной активности:

```
package com.itschool.samsung.myadapter;

import android.app.Activity;
import android.os.Bundle;
import android.widget.ListView;

public class MainActivity extends Activity {

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

        MyMonthAdapter adapter = new MyMonthAdapter(this, makeMonth());
        ListView lv = (ListView) findViewById(R.id.listView);
        lv.setAdapter(adapter);
    }

    // Метод создания массива месяцев
    MyMonth[] makeMonth() {
        MyMonth[] arr = new MyMonth[12];

        // Названия месяцев
        String[] monthArr = {"Январь", "Февраль", "Март", "Апрель", "Май",
"Июнь", "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"};
        // Среднесуточная температура
        double[] tempArr = {-12.7, -11.3, -4.5, 7.7, 19.3, 23.9, 23.5, 22.8,
16.0, 5.2, -0.3, -9.3};
        // Количество дней
        int[] dayArr = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};

        // Сборка месяцев
        for (int i = 0; i < arr.length; i++) {
            MyMonth month = new MyMonth();
            month.month = monthArr[i];
            month.temp = tempArr[i];
            month.days = dayArr[i];
            arr[i] = month;
        }
        return arr;
    }
}
```

Теперь переходим к непосредственному созданию класса своего адаптера. Наш адаптер будет являться потомком класса ArrayAdapter<MyMonth>. В классе всего два метода: конструктор и метод заполнения адаптера. Код класса:

```
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ArrayAdapter;
import android.widget.CheckBox;
import android.widget.ImageView;
import android.widget.TextView;

public class MyMonthAdapter extends ArrayAdapter<MyMonth> {
```

```
    public MyMonthAdapter(Context context, MyMonth[] arr) {
        super(context, R.layout.adapter_item, arr);
    }

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

        final MyMonth month = getItem(position);

        if (convertView == null) {
            convertView =
LayoutInflater.from(getContext()).inflate(R.layout.adapter_item, null);
        }

        // Заполняем адаптер
        ((TextView)
convertView.findViewById(R.id.textView)).setText(month.month);
        ((TextView)
convertView.findViewById(R.id.textView2)).setText(String.valueOf(month.temp)
);
        ((TextView)
convertView.findViewById(R.id.textView3)).setText(String.valueOf(month.days)
);
        // Выбираем картинку для месяца
        if (month.temp > 0.)
            ((ImageView)
convertView.findViewById(R.id.imageView)).setImageResource(R.drawable.sun);
        else
            ((ImageView)
convertView.findViewById(R.id.imageView)).setImageResource(R.drawable.snow);

        CheckBox ch = (CheckBox) convertView.findViewById(R.id.checkBox);
        ch.setChecked(month.like);
        ch.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                month.like = ((CheckBox) v).isChecked();
            }
        });
        return convertView;
    }
}
```

Обратите внимание что в конструктор адаптера передаётся массив объектов, данными которого и необходимо заполнить список, и внутри он передаётся в конструктор родителя.

В методе getView происходит заполнение каждого выводимого элемента. В первой строке происходит выделение объекта MyMonth, данные которого будут выводиться. Метод *getItem* позволяет получить элементы используемого массива. Поле *position* - сообщаем номер текущего создаваемого элемента. Далее происходит проверка, создан ли текущий экземпляр адаптера или его необходимо создать.

В следующих строках происходит заполнение надписей и выбор отображаемой картинки (картинка выбирается в зависимости от температуры месяца).

Далее происходит работа с галочкой. Она заполняется в соответствии с текущим состоянием поля like текущего месяца. Также на неё назначается слушатель события OnClick, который изменяет значение поля like текущего месяца если пользователь нажал на галочку. Таким образом, в

результате выполнения программы в массиве arr сохранено состояние визуальных элементов адаптера и это можно использовать в дальнейшей обработке.

Полностью проект приведен в материалах. Вид экрана:

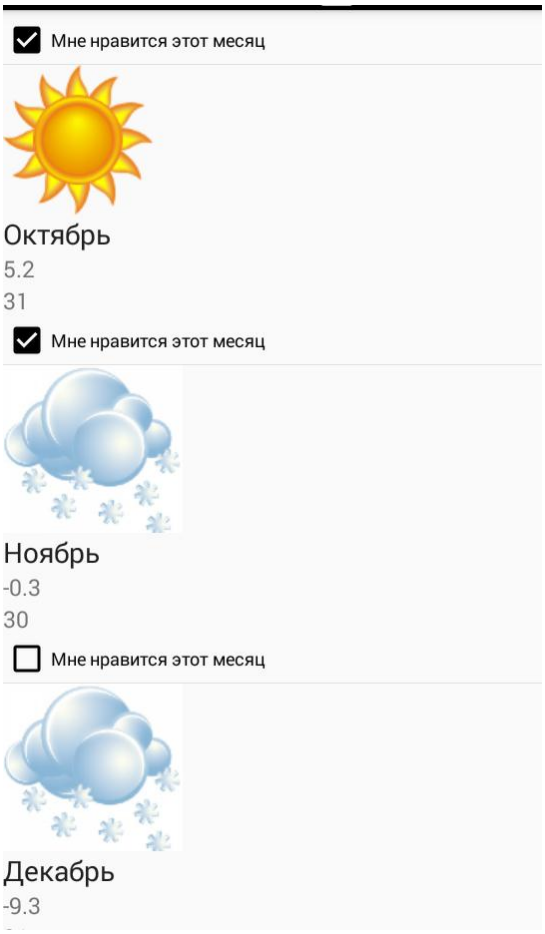

Рисунок 5

Задание 4.3.3

Реализовать адаптер с разметкой из задания 4.3.2, родителем которого является ArrayAdapter. Добавьте в разметку CheckBox, с надписью “Нравится”. Сохранять состояние статуса “Нравится” при его изменении.

Источники

[1] Климов А. Освой программирование играючи http://developer.alexanderklimov.ru/

[2] Seth D. Bergmann, “Compiler Design: Theory, Tools, and Examples”, 2010. Описание принципов создания компиляторов. Первое издание использует в примерах язык Pascal, второе - C++.

[3] Соколов А. ListView в Android: Кастомизация списков http://habrahabr.ru/post/133575/

Благодарности

Компания Samsung Electronics выражает благодарность за участие в подготовке данного материала преподавателю IT ШКОЛЫ SAMSUNG Язовцеву Игорю Алексеевичу.