В прошлом уроке мы научились получать список установленных приложений. Теперь нужно их как-то показать пользователю.

Давайте подумаем, как это сделать. Отображать простые данные, такие, как текст, мы уже научились. Тут же у нас есть массив (список) данных. Отображение списков работает несколько иначе.

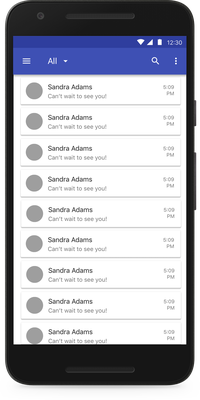
RecyclerView

В любой ОС есть компонент, предназначенный для отображения списков. В iOS это TableView, в Android — RecyclerView (раньше был ListView, но он уже давно в прошлом).

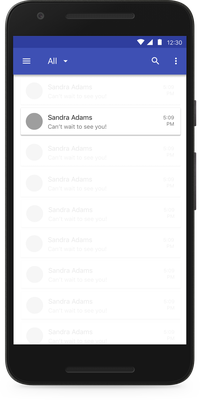
Современные приложения в 99% случаев содержат списки в том или ином виде. Лента ВКонтакте или Facebook — список. Каталог приложений в Google Play — список. Списки бывают различной сложности и разного вида, но они есть почти везде.

Отображение списков — сложная задача для ОС. Почему?

Давайте разберемся, как устроен список. Представьте интерфейс приложения со списком. Он выглядит примерно так:

[](https://skillberg.com/media/uploads/2017/11/17/919af74bc023402d8d95d9c3f28bca52.png)Список в Android

Список состоит из множества элементов, строк/ячеек:

[](https://skillberg.com/media/uploads/2017/11/17/15c9afd467c349699365390407e790ec.png)Одна ячейка в списке

Каждая ячейка — это View с некоторым количеством (в нашем случае их четыре) дочерних View.

Представьте, что у нас 10 элементов в списке. Получается, нужно создать 10 ячеек, или 40 View суммарно. Вроде не страшно?

А теперь представьте, что их не 10, а 1000. Получается, нужно создать 1000 ячеек, или 4000 View. А если 100000? Каждый View сам по себе занимает не особо много памяти, но 4000 View, да ещё и со сложной структурой займет очень много памяти.

Плюс создание View — достаточно "дорогая" операция. Когда надо создать пару десятков View, это несущественно, однако, если их количество измеряется тысячами или десятками тысяч, то создание такого количества View займет очень много места (и, скорее всего, не хватит памяти).

Как же выйти из этой ситуации?

Первое, что приходит в голову — создать столько ячеек, сколько влезает в экран, и создавать новые ячейки по мере появления их на экране, а старые, соответственно, уничтожать. Таким образом мы решаем проблему с памятью — ведь, получается, в памяти будет храниться всего 10 (или сколько там влезет в экран?) ячеек, что несущественно, и создать несколько десятков View будет не очень долго, так?

К сожалению, не так. С памятью-то проблему мы частично решили, но нужно учитывать несколько нюансов. Во-первых, постоянное создание и уничтожение объектов, особенно при быстром листании списка, будет подкидывать много работы сборщику мусора (ведь нужно освободить память от тех ячеек, которые нам больше не нужны).

Сборщик мусора, конечно, работает относительно быстро, но если он будет вызываться постоянно, интерфейс начнет подлагивать, плюс это создаст дополнительную нагрузку на CPU, что плохо скажется на потреблении заряда аккумулятора.

Но это еще полбеды. Вы же помните, я упомянул выше, что создание View — достаточно дорогая операция? Создание несколько десятков View — это не особо "тяжело" для системы, но не в том случае, когда у нас создаются десятки View каждую секунду. В общем, проблема все еще есть, и существенная.

Так как же её решить? Выход достаточно прост: мы все так же создаем столько ячеек, сколько требуется для отображения на экране, но те ячейки, которые скрылись из вида мы не уничтожаем, а переиспользуем! Отсюда, кстати, и название компонента, который отвечает за списки — RecyclerView.

Таким образом, у нас создается, скажем, 10 ячеек (количество варьируется в зависимости от того, сколько влезает в экран), и больше новые ячейки не создаются, а используются старые.

Ровно такой же алгоритм используется почти везде: в Android, iOS, Windows Phone и т.д.

Выглядит сложно? Не пугайтесь — на самом деле, все просто. Все, что требуется от нас — сообщить RecyclerView, сколько элементов в нашем списке, сказать ему, как создавать списки, и показать, как привязать данные к конкретной ячейке. Всё!

AppCompat

И последнее, о чем я обязательно расскажу, прежде чем мы перейдем к практике — это библиотеки AppCompat.

Android очень динамично развивается — постоянно добавляются новые фичи, новые элементы интерфейса и т.д. Когда-то давно для реализации некоторых фич приходилось использовать сторонние библиотеки, например, в старых версиях Android не было ActionBar, и чтобы использовать его на Android вторых версий нужно было использовать ActionBarSherlock.

Это не очень хорошо, потому что это лишняя сторонняя зависимость — за его баги не отвечает команда Android, разработку таких библиотек могут рано или поздно забросить, да и в целом это неудобно.

Или бывало даже такое, что выпускались две версии приложения — для Android 2 и для Android 3+.

Спустя какое-то время команда Android все же поняла, что это плохо сказывается на приложениях, выпускаемых для Android, и решила выпустить библиотеки AppCompat (**Application Compatibility** — совместимость приложений). Различные компоненты этих библиотек позволяют использовать самые новые фичи (в том числе и интерфейсные) на старых версиях Android, причем совместимость обеспечивается абсолютно прозрачно для разработчика.

Почему я рассказываю об этом прямо сейчас? RecyclerView появился только с релизом Android 5, и в более ранних версиях его не было. До Android 5 все использовали ListView, и если бы RecyclerView не было в AppCompat, нам бы сейчас пришлось либо реализовывать и ListView и RecyclerView, что не очень хорошо, либо использовать старый ListView, либо вообще отказаться от поддержки устройств на базе Android 4 (а на момент написания курса их очень много!).

Но, хвала Гуглу, мы можем просто подключить AppCompat и спокойно использовать RecyclerView :)

Использование RecyclerView

Наконец-то практика!

Для начала, как я и сказал выше, нужно подключить соответствующую библиотеку. Откройте build.gradle, соответствующий нашему приложению (не проекту!). В нем в разделе dependencies вы увидите примерно такие зависимости (версии могут отличаться):

dependencies {

implementation fileTree(dir: 'libs', include: ['\*.jar'])

implementation 'com.android.support:appcompat-v7:26.0.2'

testImplementation 'junit:junit:4.12'

androidTestImplementation('com.android.support.test.espresso:espresso-core:3.0.1', {

exclude group: 'com.android.support', module: 'support-annotations'

})

}

Как видите, тут уже есть зависимость от "общей" библиотеки AppCompat. Добавьте туда строчку implementation 'com.android.support:recyclerview-v7:26.0.2' (версия может быть иной), чтобы блок dependenciesвыглядел вот так:

dependencies {

implementation fileTree(dir: 'libs', include: ['\*.jar'])

implementation 'com.android.support:appcompat-v7:26.0.2'

implementation 'com.android.support:recyclerview-v7:26.0.2'

testImplementation 'junit:junit:4.12'

androidTestImplementation('com.android.support.test.espresso:espresso-core:3.0.1', {

exclude group: 'com.android.support', module: 'support-annotations'

})

}

После этого синхронизируйте проект, нажав на **Sync now** в правом верхнем углу.

Теперь откройте файл лэйаута нашей Activity, и добавьте туда RecyclerView:

<?xml version="1.0" encoding="utf-8"?>

<FrameLayout 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="com.test.packages.MainActivity">

<android.support.v7.widget.RecyclerView

android:id="@+id/apps\_rv"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent" />

</FrameLayout>

Теперь "найдём" свежесозданный RecyclerView в Java-коде (как обычно, в методе onCreate()):

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

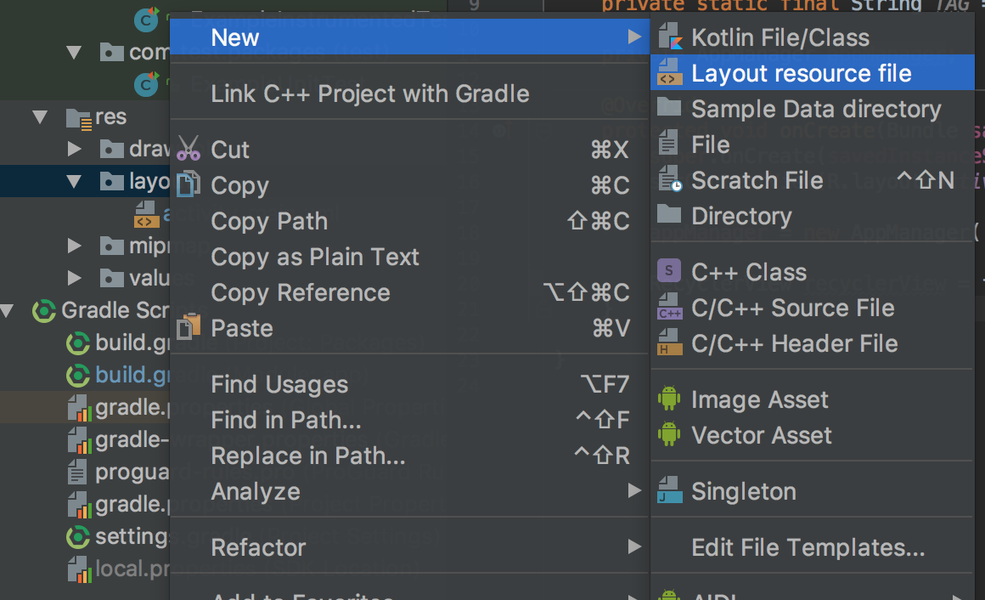
setContentView(R.layout.activity\_main);

appManager = new AppManager(this);

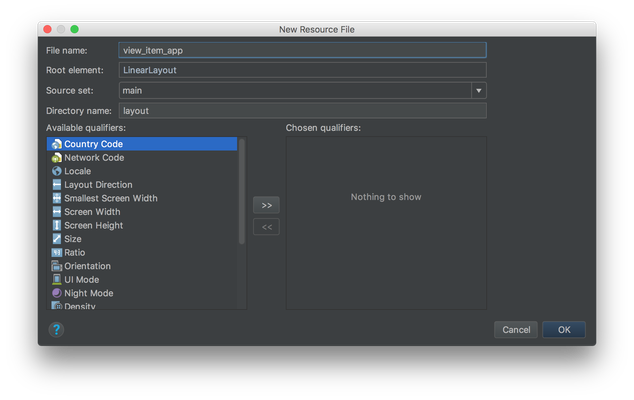
RecyclerView recyclerView = findViewById(R.id.apps\_rv);

}

Так, RecyclerView у нас есть. Теперь нужно создать лэйаут для элементов списка, который мы будем отображать. Нажмите правой кнопкой на директорию res/layout, выберите **New**, затем **Layout resource file**.

[](https://skillberg.com/media/uploads/2017/11/17/36db1097f7454ae688b126136c14fd61.png)Создание нового файла лэйаута

В появившемся окне в качестве имени файла укажите view\_item\_app и нажмите на кнопку **ОК**:

[](https://skillberg.com/media/uploads/2017/11/17/591319b6df3540788339f469003cc4aa.png)Создание нового файла лэйаута

Как вы помните из схемы, которую я приводил в начале этого урока, нам нужно отобразить иконку приложения, его название и версию.

Для начала с этим вполне справится такая схема:

<LinearLayout> <!-- Горизонтальный -->

<ImageView />

<LinearLayout> <!-- Вертикальный -->

<TextView />

<TextView />

</LinearLayout>

</LinearLayout>

Замените код лэйаута (того, что мы создали, а не Activity!) на следующий:

<?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"

android:orientation="horizontal">

<ImageView

android:id="@+id/icon\_iv"

android:layout\_width="72dp"

android:layout\_height="72dp"

android:adjustViewBounds="true"

android:scaleType="fitXY" />

<LinearLayout

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:orientation="vertical">

<TextView

android:id="@+id/name\_tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

<TextView

android:id="@+id/version\_tv"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content" />

</LinearLayout>

</LinearLayout>

Итак, у нас есть список приложений, которые надо отобразить. Есть лэйаут ячейки. Осталось научить RecyclerView создавать ячейку и отображать в ней данные.

Создание RecyclerView.Adapter

О том, как создавать и заполнять данными ячейки мы "говорим" RecyclerView в специальном классе, который называется Adapter. Давайте создадим этот класс:

package com.test.packages;

import android.support.v7.widget.RecyclerView;

import android.view.View;

import android.view.ViewGroup;

/\*\*

\* Адаптер

\*/

public class AppsAdapter extends RecyclerView.Adapter<AppsAdapter.ViewHolder> {

// В этом методе мы создаем новую ячейку

@Override

public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

return null;

}

// В этом методе мы привязываем данные к ячейке

@Override

public void onBindViewHolder(ViewHolder holder, int position) {

}

// В этом методе мы возвращаем количество элементов списка

@Override

public int getItemCount() {

return 0;

}

/\*\*

\* View holder

\* Хранит информацию о ячейке

\*/

static class ViewHolder extends RecyclerView.ViewHolder {

public ViewHolder(View itemView) {

super(itemView);

}

}

}

В первую очередь, нужно создать ячейку. Это мы сделаем в методе onCreateViewHolder().

У нас есть XML-файл разметки, и нужно каким-то образом преобразовать его во View. Для этого есть такой класс, как LayoutInflater.

LayoutInflater. Создание View

XML-файл — это всего лишь описание того, как будет выглядеть View. А LayoutInflater на основе этого описания создает объект View, с которым мы уже можем работать из Java-кода.

На самом деле, мы уже сталкивались с ним, но неявно — когда в Activity вызывается метод setContentView(), "внутри" LayoutInflater создает View из указанного XML-файла и передает его в Activity. Теперь же нам нужно создать его руками.

Сначала давайте получим этот самый LayoutInflater. Для него нам понадобится Context. Контекст мы можем взять из View, который передается в метод в качестве параметра:

Context context = parent.getContext();

Теперь получаем LayoutInflater. Способ первый: через Context.getSystemService():

LayoutInflater layoutInflater = (LayoutInflater) context.getSystemService(Context.LAYOUT\_INFLATER\_SERVICE);

Способ второй, более удобный:

LayoutInflater layoutInflater = LayoutInflater.from(context);

Теперь создаем View из XML-разметки:

View view = layoutInflater.inflate(R.layout.view\_item\_app, parent, false);

* Первый аргумент — ссылка на ресурс с XML-разметкой.
* Второй аргумент — View, который будет выступать "контейнером" для созданного. Он передается в качестве аргумента в метод onCreateViewHolder().
* Третий аргумент сообщает LayoutInflater, что не нужно прямо сейчас добавлять созданный View в родительский контейнер. Если не передать его, или передать true, приложение упадет, так как RecyclerView попытается еще раз добавить его, а двух родительских View у одного View быть не может.

Наконец, создаем ViewHolder:

ViewHolder viewHolder = new ViewHolder(view);

И возвращаем его из метода. Итого, получается вот такой код:

@Override

public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {

Context context = parent.getContext();

LayoutInflater layoutInflater = LayoutInflater.from(context);

View view = layoutInflater.inflate(R.layout.view\_item\_app, parent, false);

ViewHolder viewHolder = new ViewHolder(view);

return viewHolder;

}

Теперь нужно разобраться, что такое ViewHolder.

RecyclerView.ViewHolder

Помните, я говорил, что создание View — "дорогая" операция, если выполнять её часто? Помимо создания, нам нужно будет "привязывать" данные к этим View, например, устанавливать текст в TextView. Чтобы сделать это, нужно сначала "найти" эти TextView (и другие View), используя findViewById().

Это тоже достаточно дорогая операция, поэтому лучше сделать её один раз при создании View и сохранить "ссылки" на найденные View. Именно для этого нам пригодится ViewHolder.

Как вы помните, у нас три View, в которых мы отображаем данные: ImageView для иконки, TextView для текста и TextView для версии. Соответственно, нужно создать три поля во ViewHolder:

private final ImageView iconIv;

private final TextView nameTv;

private final TextView versionTv;

После этого в конструкторе найти все три View. Получается вот такой ViewHolder:

static class ViewHolder extends RecyclerView.ViewHolder {

private final ImageView iconIv;

private final TextView nameTv;

private final TextView versionTv;

public ViewHolder(View itemView) {

super(itemView);

iconIv = itemView.findViewById(R.id.icon\_iv);

nameTv = itemView.findViewById(R.id.name\_tv);

versionTv = itemView.findViewById(R.id.version\_tv);

}

}

Отображение данных

Осталось последнее: отобразить данные. Создайте новое приватное поле в адаптере:

private List<AppInfo> apps = new ArrayList<>();

Здесь будет храниться список приложений. Мы заранее создаем пустой список, потому что устанавливать данные мы будем не сразу, и может так получиться, что RecyclerView попробует обратиться к этим данным, а там null. И приложение упадёт.

Теперь создайте setter для данных:

public void setApps(List<AppInfo> apps) {

this.apps = apps;

}

Теперь нам нужен метод getItemCount(). Из него мы должны вернуть общее количество ячеек, которые мы хотим отобразить. В нашем случае это количество приложений, то есть код метода будет таким:

@Override

public int getItemCount() {

return apps.size();

}

И финальный шаг: отображение данных. За это отвечает метод onBindViewHolder(). В него нам передается ViewHolder, в котором мы будем отображать данные, и индекс элемента, который, собственно, надо отобразить. Тут все просто:

@Override

public void onBindViewHolder(ViewHolder holder, int position) {

AppInfo appInfo = apps.get(position);

holder.nameTv.setText(appInfo.getName());

holder.versionTv.setText(appInfo.getVersionName());

holder.iconIv.setImageDrawable(appInfo.getIcon());

}

Привязка адаптера к RecyclerView

Осталось "привязать" адаптер к RecyclerView и передать в него данные делается это вот так:

AppsAdapter appsAdapter = new AppsAdapter();

recyclerView.setLayoutManager(new LinearLayoutManager(this));

recyclerView.setAdapter(appsAdapter);

* В первой строчке мы создаем адаптер.
* Во второй — устанавливаем LayoutManager. Он отвечает за то, каким образом будут отображаться ячейки. В нашем случае — вертикально друг под другом.
* В третьей — устанавливаем адаптер в RecyclerView.

После этого передаем данные в адаптер и сообщаем адаптеру, что данные изменились. Если не уведомить его об изменении данных, то они попросту не отобразятся.

В итоге, onCreate() в нашей MainActivity выглядит вот так:

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

appManager = new AppManager(this);

List<AppInfo> installedApps = appManager.getInstalledApps();

AppsAdapter appsAdapter = new AppsAdapter();

RecyclerView recyclerView = findViewById(R.id.apps\_rv);

recyclerView.setLayoutManager(new LinearLayoutManager(this));

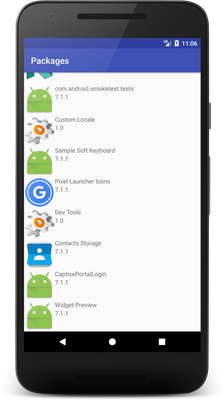
recyclerView.setAdapter(appsAdapter);

appsAdapter.setApps(installedApps);

appsAdapter.notifyDataSetChanged();

}

А приложение — вот так:

[](https://skillberg.com/media/uploads/2017/11/17/51c68d0a3bfa47d68a704ae5f5e175d1.png)Список приложений в RecyclerView

Ура! Мы сделали первое настоящее приложение :)

Выглядит оно достаточно страшно, и почти ничего не умеет, но мы исправим это в следующих уроках ;)

ПРИМЕРЫ КОДА

[ИСХОДНЫЙ КОД УРОКА](https://github.com/Skillberg/PackagesAndroid/tree/lesson-12)