

## 2.8. Элементы пользовательского интерфейса

Сайт: [Samsung Innovation Campus](https://innovationcampus.ru)  
Курс: Мобильная разработка на Kotlin  
Книга: 2.8. Элементы пользовательского интерфейса

Напечатано:: Murad Rezvan  
Дата: понедельник, 3 июня 2024, 17:47

## Описание

Продолжаем изучение элементов пользовательского интерфейса. В этой теме мы познакомимся со спиннер, ресурс-массив, адаптеры

## Оглавление

[2.8.1. Спиннеры](#)

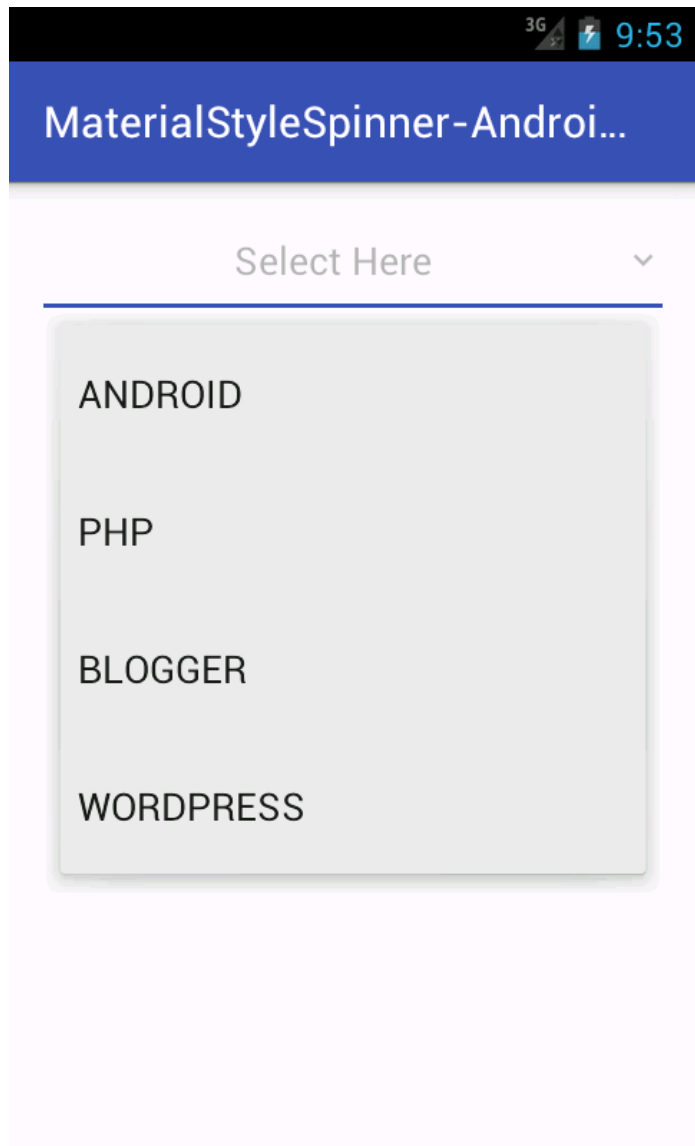
[2.8.2. Ресурс-массив](#)

[2.8.3. Адаптеры](#)

[Упражнение 2.8.](#)

## 2.8.1. Спиннеры

Следующими распространенными элементами пользовательского интерфейса являются так называемые спиннеры. Спиннеры предоставляют быстрый и удобный способ выбора одного значения из заданного набора. В состоянии по умолчанию спиннер показывает свое исходное значение. Если дотронуться до спиннера, раскроется меню с другими доступными значениями, из которых пользователь может выбрать нужное. Говоря простыми словами, спиннер – это выпадающий список, позволяющий выбрать одно значение из нескольких. Он позволяет сэкономить место на экране и очень распространен в программировании.



Вы можете добавить спиннер в свою разметку с помощью объекта [Spinner](#). Например следующим образом:

```
<Spinner
    android:id="@+id/spinner"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
</Spinner>
```

Чтобы заполнить **Spinner** данными, необходимо указать [SpinnerAdapter](#) в исходном коде **Activity** или **Fragment** (о них мы будем говорить дальше).

Ключевыми классами для работы со спиннерами являются следующие:

- [Spinner](#)
- [SpinnerAdapter](#)
- [AdapterView.OnItemSelectedListener](#)

## 2.8.2. Ресурс-массив

Одним из наиболее часто используемых элементов интерфейса являются списки. Небольшой список можно организовать очень просто. Создать массив строк в ресурсах. Спиннеры также являются в некотором роде списками. Варианты наполнения спиннера, могут могут быть различными, но должны осуществляться через класс [SpinnerAdapter](#) или его наследников. Про адаптеры мы поговорим немного далее. Например, в качестве наполнения спиннера может быть предварительно определенный файл строковых ресурсов (например, в файле /values/list.xml):

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="languages_array">
        <item>C++</item>
        <item>Java</item>
        <item>Kotlin</item>
        <item>Pascal</item>
        <item>Fortran</item>
        <item>MatLab</item>
        <item>SQL</item>
        <item>XML</item>
    </string-array>
</resources>
```

Доступ к ресурс массивам в коде программы осуществляется следующим образом:

```
val array: Array = resources.getStringArray(R.array.languages_array)
```

Кроме строковых массивов в ресурсах могут храниться массивы других типов, например цветов:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <array name="rgb">
        <item>@color/red</item>
        <item>@color/green</item>
        <item>@color/blue</item>
    </array>
</resources>
```

Доступ к ресурс-массивам других типов осуществляется аналогичным образом.

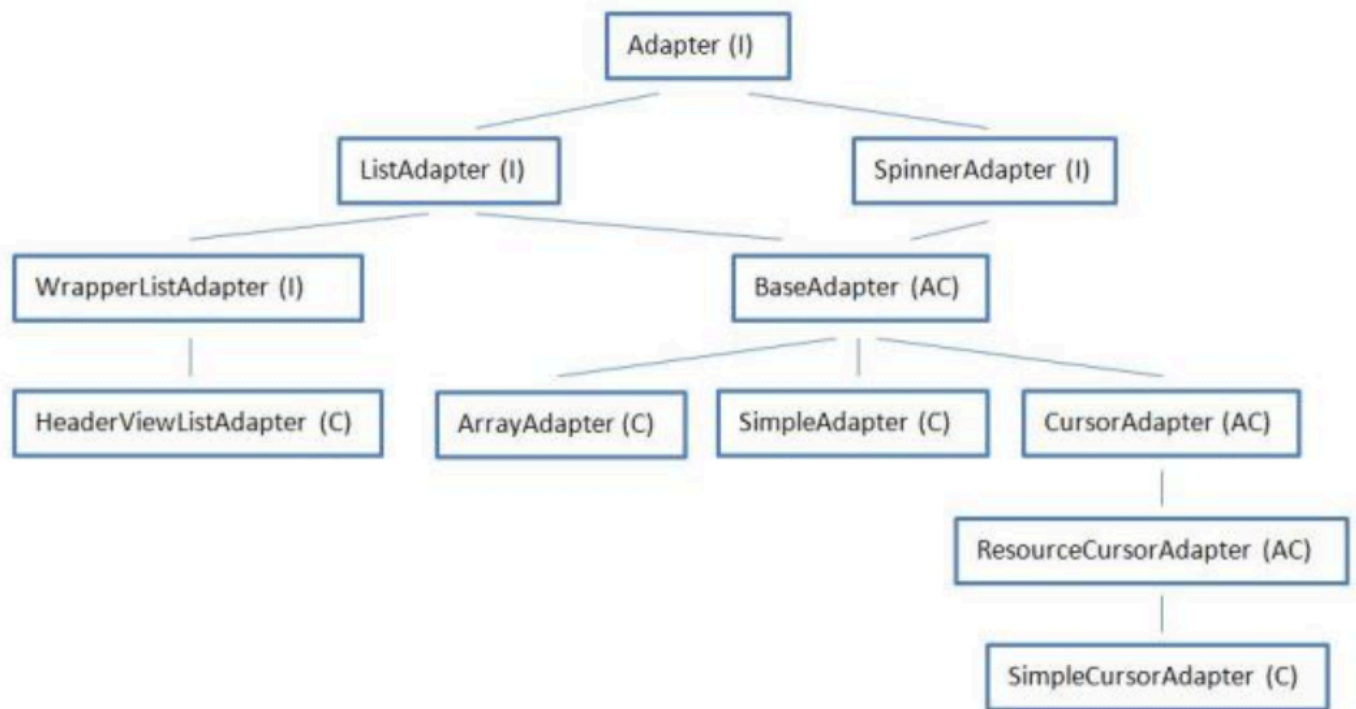
## 2.8.3. Адаптеры

Как уже отмечалось ранее, одним из наиболее часто используемых элементов интерфейса являются списки. Для вывода списка может использоваться компонент `Spinner`, а чтобы определить его содержание и структуру — связанный с ним адаптер.

Зачем все так сложно реализовано? Почему бы просто не передать данные `Spinner` напрямую? Ответ прост: чтобы экономно расходовать оперативную память. Дело в том, что мобильные устройства обладают малым количеством оперативной памяти, а операционная система Android использует всю имеющуюся (не важно сколько ее на устройстве), так как она построена вокруг виртуальной машины Java, которая сама решает, когда очищать ресурсы.

Таким образом `Adapter` — некий мост между набором данных и объектом, использующим эти данные. Также адаптер отвечает за создание View-компонента для набора данных.

Ниже приведена схема классов для адаптеров. Разберем ее подробнее некоторые из них.



Интерфейс `Adapter` описывает ключевые методы, которые должны содержать адаптеры: `getCount`, `getItem` и т.д.

Интерфейс `SpinnerAdapter` нужен для построения выпадающих списков - спиннеров. Он содержит метод `getDropDownView`, который возвращает элемент выпадающего списка. Подробнее с методами интерфейса можно ознакомиться на [официальном сайте](#)

Интерфейс `ListAdapter` используется в `ListView` (метод `setAdapter`). Содержит описание методов для работы с разделителями списка.

Интерфейсы `WrapperListAdapter` используются для работы с вложенными адаптерами и содержит метод `getWrappedAdapter`, позволяющий выделить из основного адаптера вложенный.

Класс `ArrayAdapter` — это готовый адаптер, принимающий список или массив объектов, перебирающий его и вставляющий строковое значение в указанный `TextView`.

Класс `SimpleAdapter` принимает на вход список `Map`-объектов, где каждый `Map`-объект — это список атрибутов. Кроме этого на вход принимает два массива — `from[]` и `to[]`. В `to` указываем `id` экранных элементов, а в `from` имена(`key`) из объектов `Map`, значения которых будут вставлены в соответствующие (из `from`) экранные элементы.

Абстрактный класс `CursorAdapter` реализует абстрактные методы класса `BaseAdapter`, содержит свои методы по работе с курсором и оставляет наследникам методы по созданию и наполнению View: `newView`, `bindView`.

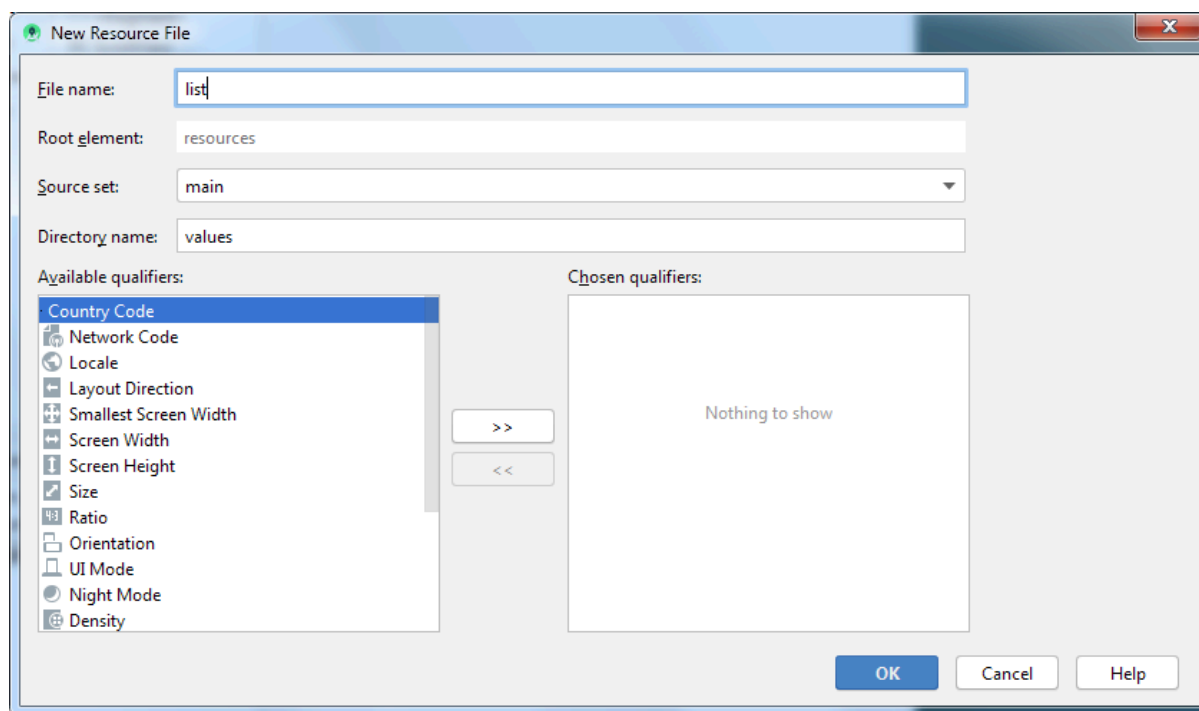
Как вы видите, использование определенного адаптера зависит от задачи разработки. В практическом примере мы будем использовать `SpinnerAdapter`.

## Упражнение 2.8.

Разработаем приложение, в котором будет находиться спиннер, наполненный некоторыми языками программирования. При нажатии на элемент спиннера WebView будет открываться страница Wikipedii с информацией о выбранном языке. Данные для спиннера будем получать из следующего ресурс-массива:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="languages_array">
        <item>C++</item>
        <item>Java</item>
        <item>Kotlin</item>
        <item>Pascal</item>
        <item>Fortran</item>
        <item>MatLab</item>
        <item>SQL</item>
        <item>XML</item>
    </string-array>
</resources>
```

В ресурсы добавим (лучше создать новый файл ресурсов) массив языков программирования из таблицы, например, */values/list.xml* и запишем туда наш ресурс-массив.



Для того, чтобы наше приложение имело выход в интернет в файл *Manifest.xml* нужно добавить строчку

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Разметку *activity\_main.xml* организуем следующим образом:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/linear_layout"
    android:gravity = "center">

    <TextView
        android:id="@+id/txtView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Выберете язык программирования"
        android:textSize="20dp" />

    <Spinner
        android:id="@+id/spinner"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignBottom="@id/txtView"/>

    <WebView
        android:id="@+id/webview"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
    ></WebView>

</LinearLayout>
```

Наконец код на языке Kotlin может выглядеть следующим образом:



```

package ru.samsung.itacademy.mdev

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.webkit.WebView
import android.webkit.WebViewClient
import android.widget.*

class MainActivity : AppCompatActivity() {

    lateinit var webView: WebView

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
        // access the items of the list
        val languages = resources.getStringArray(R.array.languages_array)

        // access the spinner
        val spinner = findViewById<Spinner>(R.id.spinner)
        if (spinner != null) {
            val adapter = ArrayAdapter(this,
                android.R.layout.simple_spinner_item, languages)
            spinner.adapter = adapter

            spinner.onItemSelectedListener = object :
                AdapterView.OnItemSelectedListener {

                override fun onItemSelected(parent: AdapterView<*>,
                    view: View, position: Int, id: Long) {

                    webView = findViewById(R.id.webview)
                    webView.webViewClient = object : WebViewClient() {
                        override fun shouldOverrideUrlLoading(view: WebView?, url: String?): Boolean {
                            view?.loadUrl(url)
                            return true
                        }
                    }
                    webView.loadUrl("https://ru.wikipedia.org/wiki/" + languages[position])
                }
            }
        }
    }
}

```

Как мы видим, здесь задействован `ArrayAdapter` - наследник `SpinnerAdapter`. В итоге вид приложения будет следующим.

SpinnerInKotlin

Выберете язык программирования

Kotlin


Википедия

# Kotlin

🌐✎

**Kotlin** (Ко́тлин) — статически типизированный [язык программирования](#), работающий поверх [Java Virtual Machine](#) и разрабатываемый компанией [JetBrains](#). Также компилируется в [JavaScript](#) и в [исполняемый код](#) ряда платформ через инфраструктуру [LLVM](#). Язык назван в честь [острова Котлин](#) в [Финском заливе](#), на котором расположен город [Кронштадт](#)<sup>[4]</sup>.

Kotlin



Класс языка	объектно-ориентированный язык программирования и язык JVM <sup>[d]</sup>
Автор	JetBrains
Расширение файлов	<div><div>.kt</div>или<div>.kts</div></div>
Выпуск	<div><div>•</div>1.3.72 (15 апреля 2020)<sup>[1]</sup></div>

Проект с приложением можно посмотреть [здесь](#)

[Начать тур для пользователя на этой странице](#)