Лекция #35 Адаптеры

Адаптеры в Android — это мост между представлением адаптера (например, ListView) и базовыми данными для этого представления. Это важная концепция в архитектуре Android, которая также требуется для сертификации Android приложения в Play Store.

Почему стоит использовать адаптеры? Без них для реализации функциональности ListView вам потребуется:

- Создать TextView в группе ScrollView.
- Затем вам нужно будет реализовать концепцию разбивки на страницы для содержимого TextView.
- Вам также придется написать дополнительный код для идентификации события щелчка в определенной строке в TextView.

Вы можете спросить, зачем нам разбивка содержимого?

Представьте, что вы создаете приложение, которому нужно отображать все ваши электронные письма, и пользователь сможет их прокручивать. В среднем современный пользователь получает 10 электронных писем в день. Если вы не реализуете разбиение содержимого, у вас будут серьезные проблемы с производительностью при извлечении всех этих данных и отображении их на экране мобильного устройства. Для этих целей и существуют адаптеры в Android и представления адаптеров!

Давайте теперь разберемся с внутренней работой адаптеров в Android и как они действуют как загрузчик данных для представления адаптера.

Адаптеры вызывают метод getView(), который возвращает представление для каждого элемента в представлении адаптера. Формат макета и соответствующие данные для элемента в представлении адаптера устанавливаются в методе getView(). Если getView() будет возвращать новое представление каждый раз, когда он вызывается, это приведёт к серьёзным проблемам с производительностью. Создание нового представления в Android очень затратно, так как вам нужно пройтись по иерархии представлений (используя метод findViewbyID()), а затем раздуть (inflate) представление, чтобы окончательно отобразить его на экране. Так же это очень сильно нагружает сборщик мусора в Android тем, что, когда пользователь прокручивает список, создается новое представление; старое представление (поскольку оно не переработано) лишается ссылки и становится кандидатом на удаление сборщиком мусора. Так что в целях оптимизации Android перерабатывает представления и повторно использует представление, которое выходит из фокуса.

Предположим, что на приведенном выше рисунке мы отображаем месяцы года в ListView. Для начала на экране показаны месяцы с января по май. При прокрутке представления месяц январь выходит за пределы области отображения экрана мобильного устройства. Как только январское представление исчезает с экрана, представление адаптера (в данном случае ListView) отправляет представление чему-то, что называется переработчиком (recycler). Поэтому при прокрутке вверх вызывается метод getView() для получения следующего представления (которое это июнь). Этот метод getView() имеет параметр convertview, который указывает на неиспользуемое представление в переработчике. Через convertview адаптер пытается завладеть неиспользуемым представлением и повторно использовать его для отображения нового представления (в данном случае это июнь).

Обзор пользовательских адаптеров Android ListView

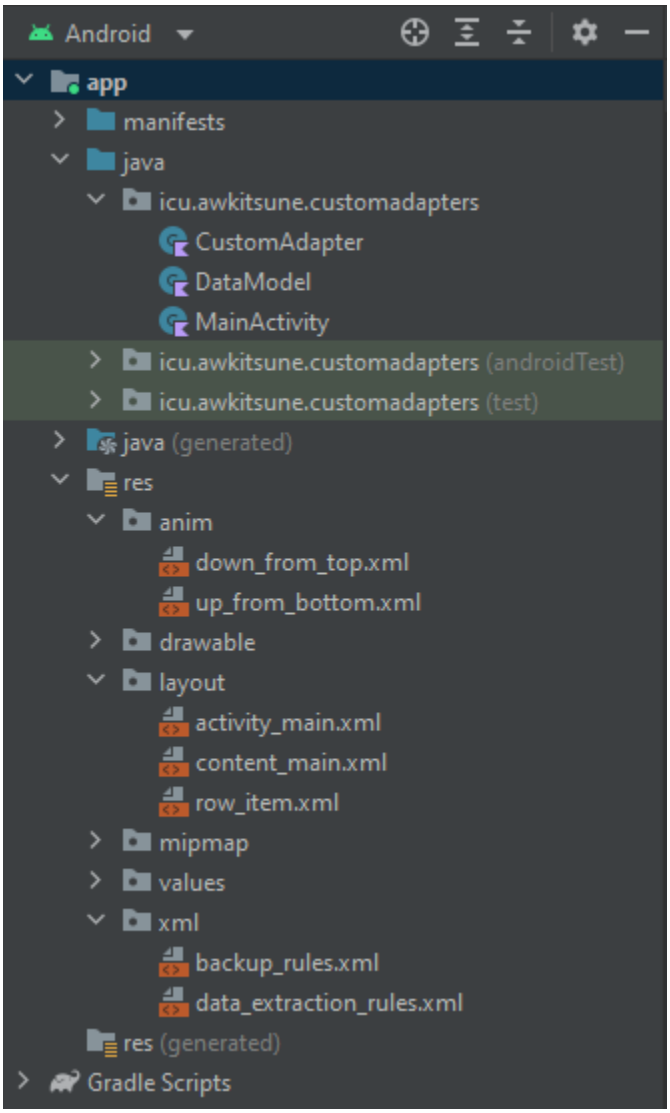
Самый простой адаптер для заполнения представления из списка ArrayList — это `ArrayAdapter`. Существуют и другие адаптеры, такие как CursorAdapter, который напрямую привязывается к набору результатов из локальной базы данных SQLite и использует курсор в качестве источника данных.

При создании экземпляра ListView строки заполняются таким образом, что заполняется всё видимое пространство списка. После этого в памяти не

создаются новые элементы строки. Когда пользователь прокручивает список, элементы, которые покидают экран, сохраняются в памяти для последующего использования, а затем каждая новая строка, появляющаяся на экране, повторно использует более старую строку, хранящуюся в памяти.

Создадим приложение, состоящее из списка строк, отображающих текстовые описания и значок информации. Если щелкнуть на строку, отобразится SnackBar с текстовыми элементами этой строки. Если щелкнуть информацию, отобразится SnackBar с информацией, относящейся к этой строке.

Структура проекта



Создание View Template

Создадим макет xml, который представляет элементы в строке в индивидуальном порядке. `row_item.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="match_parent"
    android:orientation="vertical"
    android:padding="16dp">

    <TextView
        android:id="@+id/name"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentTop="true"
        android:text="Marshmallow"
        android:textAppearance="?android:attr/textAppearanceSmall"/>

    <TextView
        android:id="@+id/type"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@+id/name"
        android:layout_marginTop="5dp"
        android:text="Android 6.0" />

    <ImageView
        android:id="@+id/item_info"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentEnd="true"
        android:layout_alignParentRight="true"
        android:layout_centerVertical="true"
        android:src="@android:drawable/ic_dialog_info" />

    <LinearLayout
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_centerInParent="true">

        <TextView
            android:id="@+id/version_heading"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="API: "
            android:textStyle="bold" />

        <TextView
            android:id="@+id/version_number"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="23"
            android:textStyle="bold" />
    </LinearLayout>

</RelativeLayout>
```

Мы создадим пользовательский ListView путем создания подкласса ArrayAdapter с DataModel в качестве объекта. `getView()` — это метод, который возвращает фактическое представление, используемое как строка в ListView в определенной позиции. `activity_main.xml` содержит ListView, как показано ниже.

`activity_main.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ListView
```

```
        android:id="@+id/list"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Модель данных, содержащаяся в ArrayList, показана ниже.

`DataModel.kt`

```
package icu.awkitsune.customadapters

class DataModel(var name: String, var type: String, var version_number: String, var feature: String)
```

CustomAdapter, который заполняет DataModel в ListView, показан ниже.
`CustomAdapter.kt`

```
package icu.awkitsune.customadapters

import android.content.Context
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.view.animation.Animation
import android.view.animation.AnimationUtils
import android.widget.ArrayAdapter
import android.widget.ImageView
import android.widget.TextView
import com.google.android.material.snackbar.Snackbar


class CustomAdapter(dataSet: ArrayList<DataModel>, context: Context) :
    ArrayAdapter<DataModel>(context, R.layout.row_item, dataSet), View.OnClickListener {
    private var mContext: Context

    // View lookup cache
```

```
private class ViewHolder {
    var txtName: TextView? = null
    var txtType: TextView? = null
    var txtVersion: TextView? = null
    var info: ImageView? = null
}

override fun onClick(v: View) {
    val position = v.tag as Int
    val `object`: Any? = getItem(position)
    val dataModel = `object` as DataModel?
    when (v.id) {
        R.id.item_info -> Snackbar.make(
            v,
            "Release date " + dataModel!!.feature,
            Snackbar.LENGTH_LONG
        )
            .setAction("No action", null).show()
    }
}

private var lastPosition = -1

init {
    mContext = context
}

override fun getView(position: Int, convertView: View?, parent: ViewGroup): View {
    var convertView: View? = convertView
    val dataModel = getItem(position)

    val viewHolder: ViewHolder
    val result: View?
    if (convertView == null) {
        viewHolder = ViewHolder()
        val inflater = LayoutInflater.from(context)
        convertView = inflater.inflate(R.layout.row_item, parent, false)
        viewHolder.txtName = convertView.findViewById(R.id.name)
```

```
            viewHolder.txtType = convertView.findViewById(R.id.type)
            viewHolder.txtVersion = convertView.findViewById(R.id.version_number)
            viewHolder.info = convertView.findViewById(R.id.item_info) as ImageView
            result = convertView
            convertView.tag = viewHolder
        } else {
            viewHolder = convertView.tag as ViewHolder
            result = convertView
        }
        val animation: Animation = AnimationUtils.loadAnimation(
            mContext,
            if (position > lastPosition) R.anim.up_from_bottom else R.anim.down_from_top
        )
        result?.startAnimation(animation)
        lastPosition = position
        viewHolder.txtName!!.text = dataModel!!.name
        viewHolder.txtType!!.text = dataModel.type
        viewHolder.txtVersion!!.text = dataModel.version_number
        viewHolder.info?.setOnClickListener(this)
        viewHolder.info?.tag = position

        return convertView!!
    }
}
```

В приведенном выше коде мы добавили `onClickListener` в ImageView, который отображает SnackBar при нажатии с описанием для соответствующей строки. Также строки списка анимируются при прокрутке. Два XML-файла ресурсов анимации приведены ниже.

`down_from_top.xml`

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="@android:anim/decelerate_interpolator">
    <translate
        android:fromXDelta="0%" android:toXDelta="0%"
        android:fromYDelta="-100%" android:toYDelta="0%"
```

```
        android:duration="400" />
</set>
```

up_from_bottom.xml

```
<?xml version="1.0" encoding="utf-8"?>
<set xmlns:android="http://schemas.android.com/apk/res/android"
    android:shareInterpolator="@android:anim/decelerate_interpolator">
    <translate
        android:fromXDelta="0%" android:toXDelta="0%"
        android:fromYDelta="100%" android:toYDelta="0%"
        android:duration="400" />
</set>
```

MainActivity.kt, где для CustomAdapter задано значение ListView, определено ниже. Наряду с этим заполняется случайный ArrayList объектов DataModel.
MainActivity.kt

```
package icu.awkitsune.customadapters

import android.os.Bundle
import android.view.View
import android.widget.AdapterView.OnItemClickListener
import android.widget.ListView
import androidx.appcompat.app.AppCompatActivity
import com.google.android.material.snackbar.Snackbar


class MainActivity : AppCompatActivity() {
    lateinit var dataModels: ArrayList<DataModel>
    lateinit var listView: ListView
    private lateinit var adapter: CustomAdapter

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
```

```
        listView = findViewById<View>(R.id.list) as ListView
        dataModels = ArrayList()
        dataModels.add(DataModel("Apple Pie", "Android 1.0", "1", "September 23, 2008"))
        dataModels.add(DataModel("Banana Bread", "Android 1.1", "2", "February 9, 2009"))
        dataModels.add(DataModel("Cupcake", "Android 1.5", "3", "April 27, 2009"))
        dataModels.add(DataModel("Donut", "Android 1.6", "4", "September 15, 2009"))
        dataModels.add(DataModel("Eclair", "Android 2.0", "5", "October 26, 2009"))
        dataModels.add(DataModel("Froyo", "Android 2.2", "8", "May 20, 2010"))
        dataModels.add(DataModel("Gingerbread", "Android 2.3", "9", "December 6, 2010"))
        dataModels.add(DataModel("Honeycomb", "Android 3.0", "11", "February 22, 2011"))
        dataModels.add(DataModel("Ice Cream Sandwich", "Android 4.0", "14", "October 18, 2011"))
        dataModels.add(DataModel("Jelly Bean", "Android 4.2", "16", "July 9, 2012"))
        dataModels.add(DataModel("Kitkat", "Android 4.4", "19", "October 31, 2013"))
        dataModels.add(DataModel("Lollipop", "Android 5.0", "21", "November 12, 2014"))
        dataModels.add(DataModel("Marshmallow", "Android 6.0", "23", "October 5, 2015"))
        adapter = CustomAdapter(dataModels!!, applicationContext)
        listView.adapter = adapter
        listView.onItemClickListener = OnItemClickListener { parent, view, position, id ->
            val dataModel = dataModels!![position]
            Snackbar.make(
                view,
                """${dataModel.name} ${dataModel.type}
                    API: ${dataModel.version_number}""".trimMargin(),
                Snackbar.LENGTH_LONG)
                .setAction("No action", null)
                .show()
        }
    }
}
```

Пример того, как должна работать программа ниже:

https://imgur.com/a/CYx5L6b