Лекция #27. Взаимодействие между фрагментами

Одна activity может использовать несколько фрагментов, например, с одной стороны список, а с другой - детальное описание выбранного элемента списка. В такой конфигурации activity использует два фрагмента, которые между собой должны взаимодействовать. Рассмотрим базовые принципы взаимодействия фрагментов в приложении.

Создадим новый проект с пустой MainActivity. Далее создадим разметку layout для фрагментов. Пусть у нас в приложении будет два фрагмента. Добавим в папку **res/layout** новый xml-файл **fragment_list.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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
    <ListView
        android:id="@+id/countriesList"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent">
    </ListView>
</androidx.constraintlayout.widget.ConstraintLayout>
```

Здесь определен элемент ListView для вывода списка объектов.

И также добавим для другого фрагмента файл разметки **fragment_detail.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"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="16dp">
    <TextView
        android:id="@+id/detailsText"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"

        android:text="Не выбрано"
        android:textSize="22sp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Оба фрагмента будут предельно простыми: один будет содержать список, а второй - текстовой поле. Логика приложения будет такова: при выборе элемента в списке в одном фрагменте выбранный элемент должен отобразиться в текстовом поле, которое находится во втором фрагменте.

Затем добавим в проект в одну папку с MainActivity собственно классы фрагментов. Добавим новый класс **ListFragment** со следующим содержимым:

```
package com.awkitsune.fragmentdemonstration2

import android.content.Context;
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import androidx.fragment.app.Fragment;
import android.widget.AdapterView.OnItemClickListener
import java.lang.ClassCastException


class ListFragment: Fragment(){
    internal interface OnFragmentSendDataListener {
        fun onSendData(data: String?)
    }

    private var fragmentSendDataListener: OnFragmentSendDataListener? = null
    var countries = arrayOf<String?>("Бразилия", "Аргентина", "Колумбия",
"Чили", "Уругвай")


    override fun onAttach(context: Context) {
        super.onAttach(context)
        try {
            fragmentSendDataListener = context as OnFragmentSendDataListener
        } catch (e: ClassCastException) {
            throw ClassCastException(
```

```
                context.toString()
                        + " должен реализовывать интерфейс
OnFragmentInteractionListener"
            )
        }
    }

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        val view: View = inflater.inflate(R.layout.fragment_list, container,
false)

        // получаем элемент ListView
        val countriesList = view.findViewById<ListView>(R.id.countriesList)

        // создаем адаптер
        val adapter: ArrayAdapter<String?> =
            ArrayAdapter((context)!!, android.R.layout.simple_list_item_1,
countries)

        // устанавливаем для списка адаптер
        countriesList.adapter = adapter

        // добавляем для списка слушатель
        countriesList.onItemClickListener =
            OnItemClickListener { parent, v, position, id ->
                // получаем выбранный элемент
                val selectedItem: String = parent.getItemAtPosition(position)
as String

                // Посылаем данные Activity
                fragmentSendDataListener!!.onSendData(selectedItem)
            }
        return view
    }
}
```

Фрагменты не могут напрямую взаимодействовать между собой. Для этого надо обращаться к контексту, в качестве которого выступает класс Activity. Для обращения к activity, как правило, создается вложенный интерфейс. В данном случае он называется **OnFragmentSendDataListener** с одним методом:

```
interface OnFragmentSendDataListener {
    fun onSendData(data: String?)
}

private var fragmentSendDataListener: OnFragmentSendDataListener? = null
```

При обработке нажатия на элемент в списке мы можем отправить Activity данные о выбранном объекте:

```
val selectedItem: String = parent.getItemAtPosition(position) as String
fragmentSendDataListener!!.onSendData(selectedItem)
```

Таким образом, при выборе объекта в списке MainActivity получит выбранный объект.

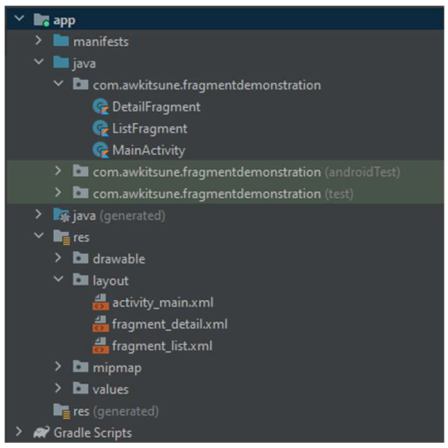
Теперь создадим фрагмент **DetailFragment** и изменим его класс:

```
package com.awkitsune.fragmentdemonstration2

import androidx.fragment.app.Fragment

import android.widget.TextView

import android.os.Bundle

import android.view.ViewGroup

import android.view.LayoutInflater
import android.view.View


class DetailFragment: Fragment() {
    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        return inflater.inflate(R.layout.fragment_detail, container, false)
    }

    // обновление текстового поля
    fun setSelectedItem(selectedItem: String?) {
        val view = view?.findViewById<TextView>(R.id.detailsText)
        view?.text = selectedItem
    }
}
```

Задача этого фрагмента - вывод некоторой информации. Так как он не должен передавать никакую информацию другому фрагменту, здесь мы модем ограничиться только переопределением метода onCreateView(), который в качестве визуального интерфейса устанавливает разметку из файла **fragment_detail.xml**

Но чтобы имитировать взаимодействие между двумя фрагментами, здесь также определен метод **setSelectedItem()**, который обновляет текст на текстовом поле.

В итоге получится следующая структура:



Теперь изменим файл разметки **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"
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/listFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:name="com.awkitsune.fragmentdemonstration2.ListFragment"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintBottom_toTopOf="@+id/detailFragment"
        app:layout_constraintRight_toRightOf="parent"/>

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/detailFragment"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:name="com.awkitsune.fragmentdemonstration2.DetailFragment"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toBottomOf="@id/listFragment"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintRight_toRightOf="parent"/>

</androidx.constraintlayout.widget.ConstraintLayout>
```

С помощью двух элементов **FragmentContainerView** в MainActivity добавляются два выше определенных фрагмента.

И в конце изменим код **MainActivity**:

```
package com.awkitsune.fragmentdemonstration2

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle

class MainActivity : AppCompatActivity(),
ListFragment.OnFragmentSendDataListener{
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
```

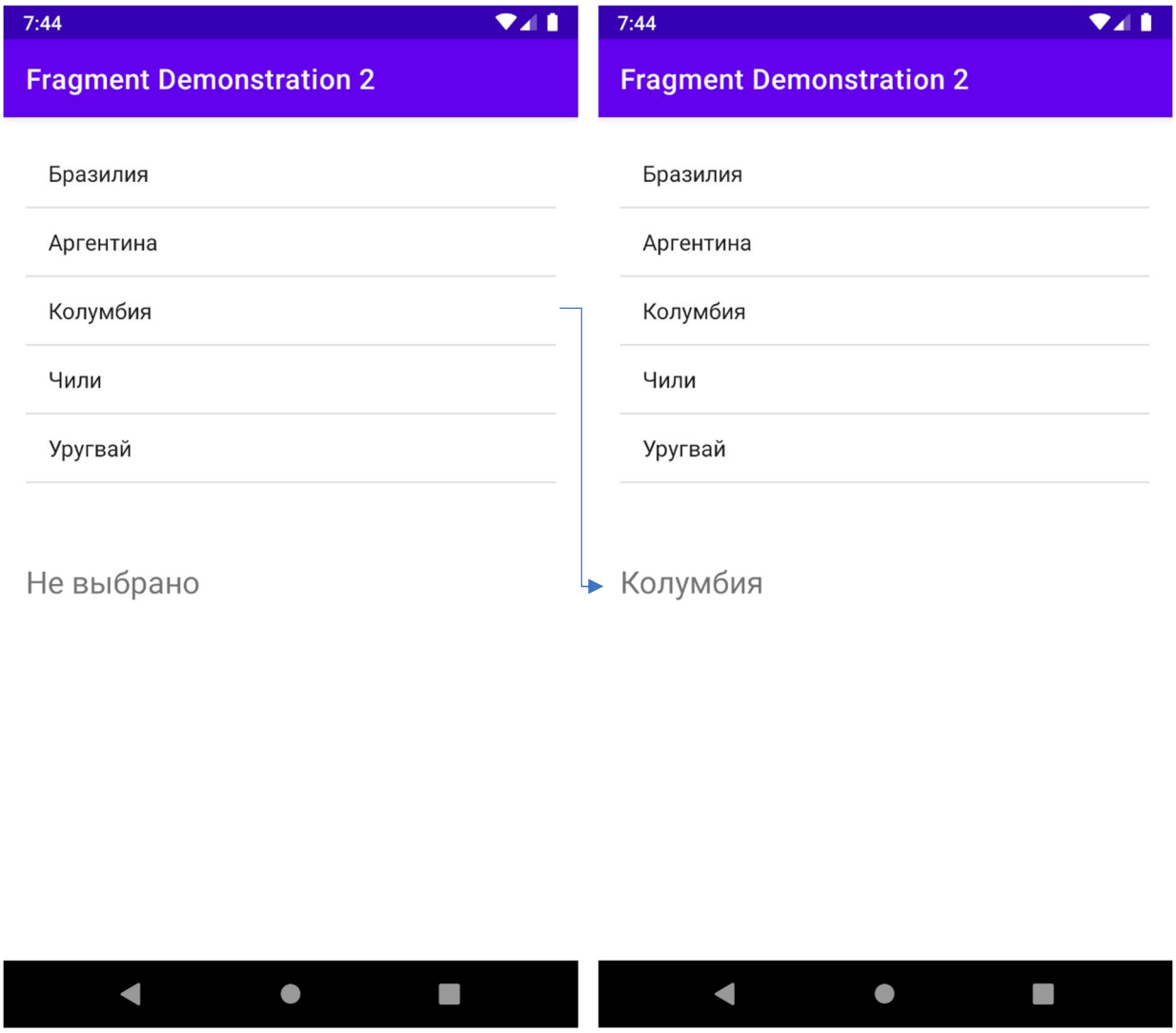
```
        setContentView(R.layout.activity_main)
    }

    override fun onSendData(selectedItem: String?) {
        val fragment = supportFragmentManager
            .findFragmentById(R.id.detailFragment) as DetailFragment?
        fragment?.setSelectedItem(selectedItem)
    }
}
```

Для взаимодействия фрагмента ListFragment с другим фрагментом через MainActivity надо, чтобы эта activity реализовывала интерфейс OnFragmentSendDataListener. Для этого реализуем метод **onSendData()**, который получает фрагмент DetailFragment и вызывает у него метод **setSelectedItem()**

В итоге получится, что при выборе в списке во фрагменте ListFragment будет срабатывать слушатель списка и в частности его метод **onItemClick(AdapterView<?> parent, View v, int position, long id)**, который вызовет метод **fragmentSendDataListener.onSendData(selectedItem);**

fragmentSendDataListener устанавливается как MainActivity, поэтому при этом будет вызван метод setSelectedItem у фрагмента DetailFragment. Таким образом, произойдет взаимодействие между двумя фрагментами.

Если мы запустим проект, то на экран будут выведены оба фрагмента, которые смогут взаимодействовать между собой.


7:44
Fragment Demonstration 2
Бразилия
Аргентина
Колумбия
Чили
Уругвай
Не выбрано
7:44
Fragment Demonstration 2
Бразилия
Аргентина
Колумбия
Чили
Уругвай
Колумбия