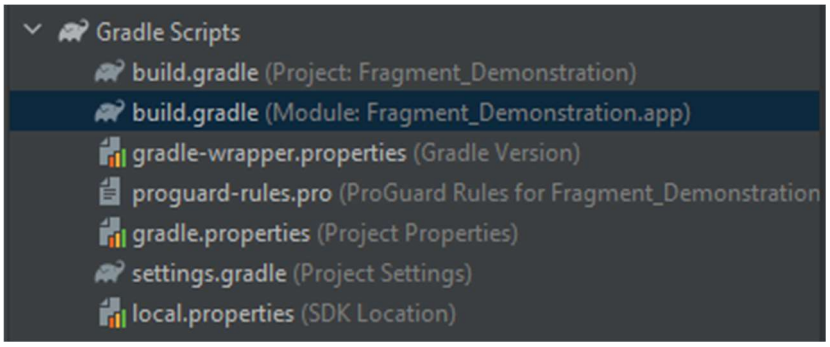
Лекция #25. Фрагменты

Организация приложения на основе нескольких **activity** не всегда может быть оптимальной. Мир ОС Android довольно сильно фрагментирован и состоит из многих устройств. И если для мобильных аппаратов с небольшими экранами взаимодействие между разными **activity** выглядит довольно неплохо, то на больших экранах - планшетах, телевизорах окна **activity** смотрелись бы не очень в силу большого размера экрана. Собственно поэтому и появилась концепция **фрагментов**.

Фрагмент представляет кусочек визуального интерфейса приложения, который может использоваться повторно и многократно. У фрагмента может быть собственный файл **layout**, у фрагментов есть свой собственный жизненный цикл. Фрагмент существует в контексте **activity** и имеет свой жизненный цикл, вне **activity** обособлено он существовать не может. Каждая **activity** может иметь несколько **фрагментов**.

Для начала работы с фрагментами создадим новый проект с пустой **MainActivity**. И вначале создадим первый фрагмент. Но сразу стоит отметить, что не вся функциональность фрагментов по умолчанию может быть доступна в проекте, поскольку располагается в отдельной библиотеке - **AndroidX Fragment library**. И вначале необходимо подключить к проекту эту библиотеку в файле **build.gradle**.



Найдем в нем секцию dependencies, которая выглядит по умолчанию примерно так:

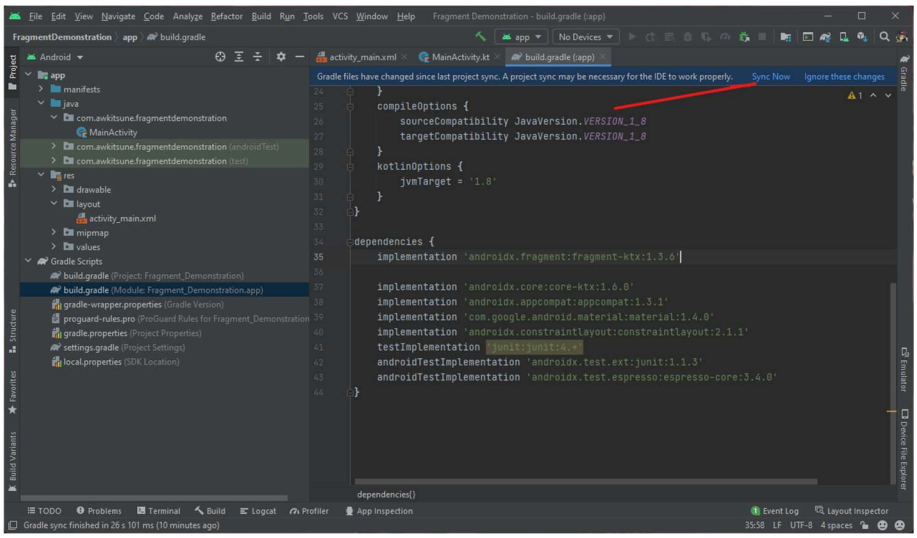
```
dependencies {

    implementation 'androidx.core:core-ktx:1.6.0'
    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
}
```

В ее начало добавим строку

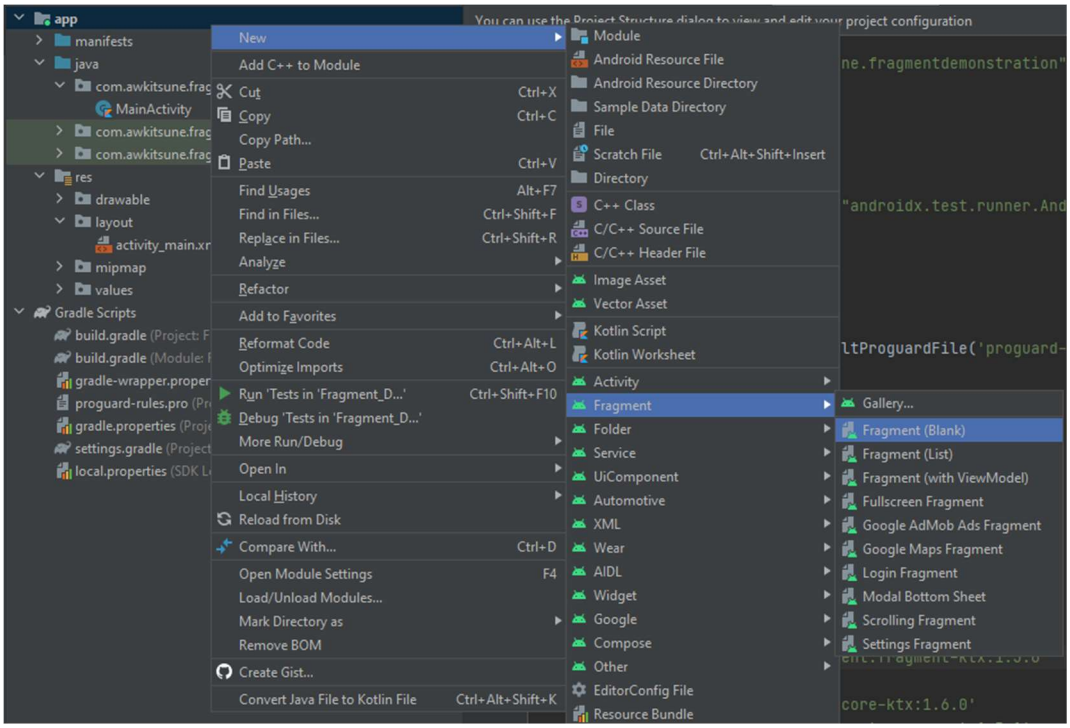
```
implementation 'androidx.fragment:fragment-ktx:1.3.6'
```

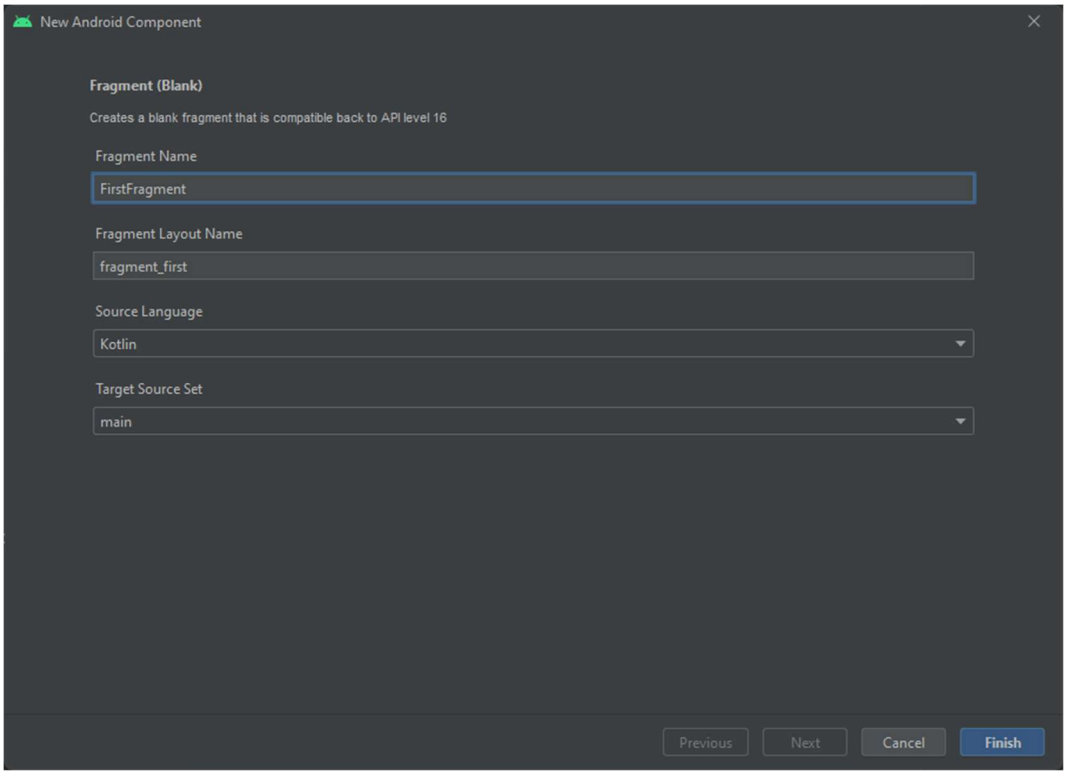
И затем нажмем на появившуюся ссылку Sync Now.



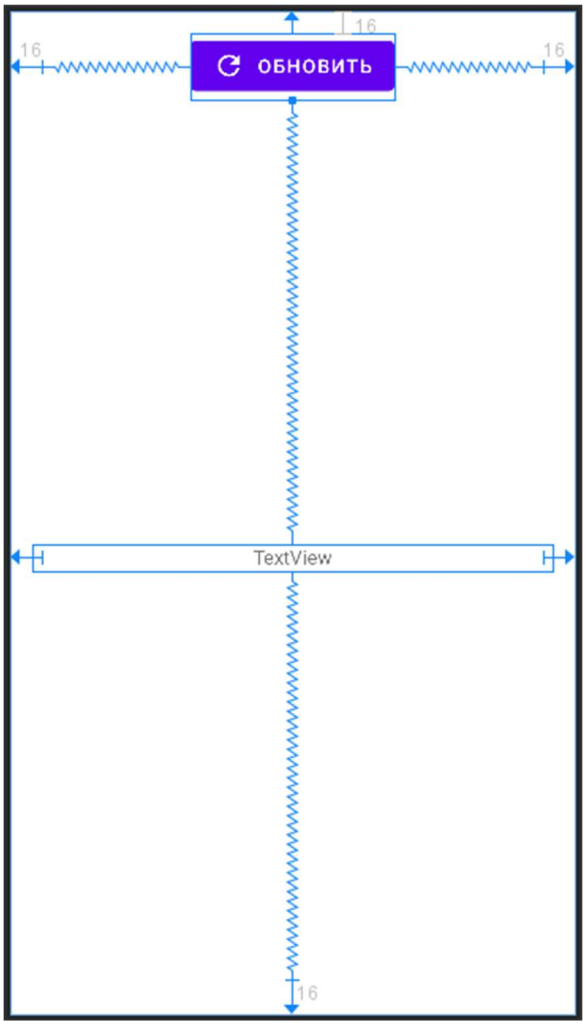
Фактически фрагмент - это обычный класс **Java**, который наследуется от класса **Fragment**. Однако как и класс **Activity**, фрагмент может использовать **xml-файлы layout** для определения графического интерфейса. И таким образом, мы можем добавить по отдельности **класс Java**, который представляет фрагмент, и **файл xml** для хранения в нем разметки интерфейса, который будет использовать фрагмент.

Итак, добавим в папку проект новый фрагмент





И определим в нём следующую разметку:



Фрагменты содержат те же **элементы управления**, что и **activity**. В частности, здесь определены **кнопка и текстовое поле**, которые будут составлять интерфейс фрагмента.

Теперь **изменим класс** фрагмента. Перейдём в класс **FirstFragment.kt** и определим у него следующее содержание:

```
package com.awkitsune.fragmentdemonstration

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup

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

Добавление фрагмента в Activity

Для использования фрагмента добавим его в **MainActivity**. Для этого изменим файл **activity_main.xml**, которая определяет интерфейс для **MainActivity**:

```
<?xml version="1.0" encoding="utf-8"?>
<FrameLayout 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:id="@+id/frameLayoutMain"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity" />
```

Для добавления фрамента применяется элемент **FrameLayout**. Код класса **MainActivity** тоже подвергнется изменениям:

```
package com.awkitsune.fragmentdemonstration

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

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

        val firstFragment = FirstFragment()

        supportFragmentManager.beginTransaction().apply {
            replace(R.id.frameLayoutMain, firstFragment)
            commit()
        }
    }
}
```

Здесь мы используем **supportFragmentManager** для загрузки нашего фрагмента через метод **beginTransaction().apply**

Если мы запустим приложение, то мы увидим фактически тот же самый интерфейс, который мы могли бы сделать и через **activity**, только в данном случае интерфейс будет определен во фрагменте:

Добавление логики к фрагменту

Фрагмент определяет кнопку. Теперь добавим к этой кнопки некоторое действие. Для этого изменим класс **FirstFragment**:

```
package com.awkitsune.fragmentdemonstration

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.Button
import android.widget.TextView
import java.util.*

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

```
    override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
        super.onViewCreated(view, savedInstanceState)

        val updateButton: Button = view.findViewById(R.id.updateButton)
        val dateText: TextView = view.findViewById(R.id.textView)

        updateButton.setOnClickListener {
            val curDate: String = Date().toString()
            dateText.text = curDate
        }
    }
}
```

Здесь переопределен метод **onViewCreated** класса Fragment, который вызывается после создания объекта View для визуального интерфейса, который представляет данный фрагмент. Созданный объект View передается в качестве первого параметра. И далее мы можем получить конкретные элементы управления в рамках этого объекта View, в частности, TextView и Button, и выполнить с ними некоторые действия. В данном случае в обработчике нажатия кнопки в текстовом поле выводится текущая дата.

Важно что теперь мы привязываем действие к кнопке в классе фрагмента обращаясь к методу кнопки **setOnClickListener**:

```
updateButton.setOnClickListener {
    val curDate: String = Date().toString()
    dateText.text = curDate
}
```