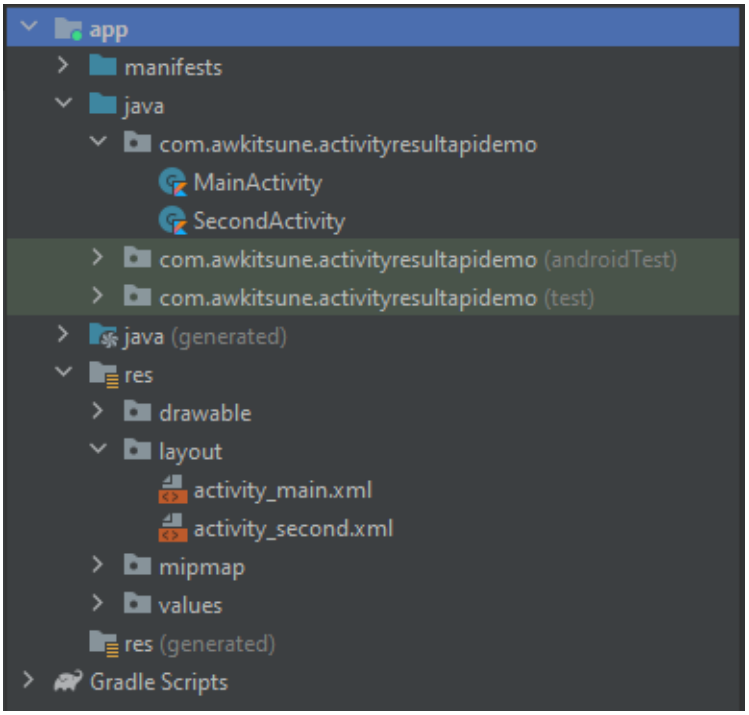
Лекция #20. Получение результата из Activity

В прошлой теме было рассмотрено как вызывать новую **Activity** и передавать ей некоторые **данные**. Но мы можем не только передавать данные запускаемой activity, но и ожидать от нее некоторого результата работы.

К примеру, пусть у нас в проекте будут две activity: **MainActivity** и **SecondActivity**. А для каждой activity есть свой файл интерфейса: **activity_main.xml** и **activity_second.xml** соответственно.



В прошлой теме мы вызывали новую activity с помощью метода **startActivity()**. Для получения же результата работы запускаемой **activity** необходимо использовать **Activity Result API**.

Activity Result API предоставляет компоненты для регистрации, запуска и обработки результата другой Activity. Одним из преимуществ применения Activity Result API является то, что он отвязывает результат Activity от самой Activity. Это позволяет получить и обработать результат, даже если Activity, которая возвращает результат, в силу ограничений памяти или в силу других причин завершила свою работу. Вкратце рассмотрим основные моменты применения **Activity Result API**.

Регистрация функции для получения результата

Для регистрации функции, которая будет обрабатывать результат, **Activity Result API** предоставляет метод **registerForActivityResult()**. Этот метод в качестве параметров принимает
объекты **ActivityResultContract** и **ActivityResultCallback** и возвращает объект **ActivityResultLauncher**, который применяется для запуска другой **activity**.

```
ActivityResultLauncher<I> registerForActivityResult (
                ActivityResultContract<I, O> contract,
                ActivityResultCallback<O> callback)
```

ActivityResultContract определяет контракт: данные какого типа будут подаваться на вход и какой тип будет представлять результат.

ActivityResultCallback представляет интерфейс с единственным методом **onActivityResult()**, который определяет обработку полученного результата. Когда вторая activity закончит работу и возвратит результат, то будет как раз вызываться этот метод. Результат передается в метод в качестве параметра. При этом тип параметра должен соответствовать типу результата, определенного в **ActivityResultContract**. Например:

```
var mStartForResult =
registerForActivityResult(StartActivityForResult(),
    ActivityResultCallback<Any?> {
        // обработка result
    })
```

Класс **ActivityResultContracts** предоставляет ряд встроенных типов контрактов. Например, в листинге кода выше применяется встроенный тип **ActivityResultContracts.StartActivityForResult**, который в качестве входного объекта устанавливает объект Intent, а в качестве типа результата - тип ActivityResult.

Запуск activity для получения результата

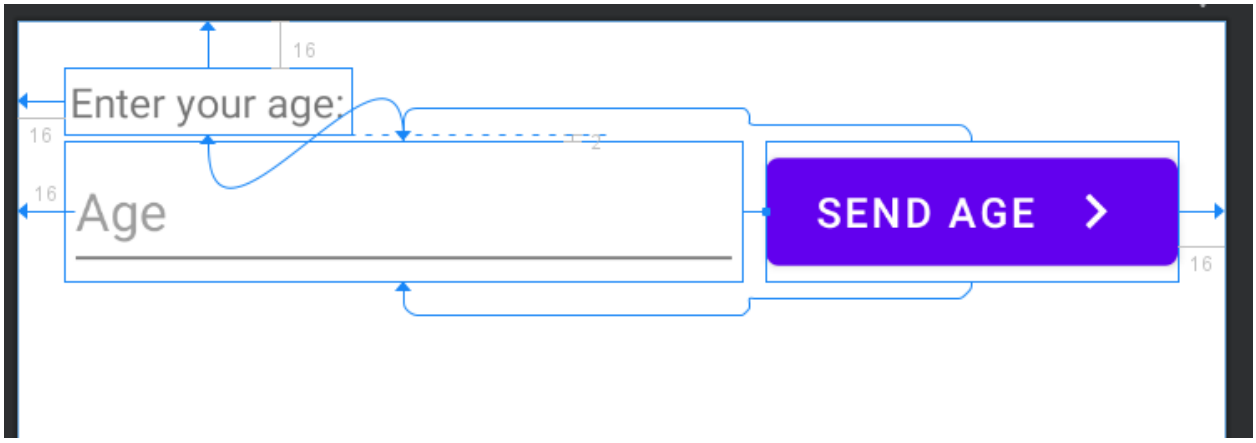
Метод **registerForActivityResult()** регистрирует функцию-колбек и возвращает объект **ActivityResultLauncher**. С помощью этого мы можем запустить activity. Для этого у объекта ActivityResultLauncher вызывается метод **launch()**:

```
mStartForResult?.launch(intent);
```

В метод **launch()** передается объект того типа, который определен объектом **ActivityResultContracts** в качестве входного.

Практическое применение Activity Result API

Итак, определим в файле activity_main.xml следующий интерфейс:



Для ввода данных здесь определен элемент **EditText**, а для отправки - кнопка.

Определим в классе **MainActivity** запуск второй **activity**:

```
package com.awkitsune.activityresultapidemo

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

import android.view.View
import android.widget.EditText

import android.widget.TextView

import androidx.activity.result.ActivityResultLauncher
import
androidx.activity.result.contract.ActivityResultContracts.StartActivit
yForResult
```

```
class MainActivity : AppCompatActivity() {
    companion object{
        const val AGE_KEY: String = "AGE"
        const val ACCESS_MESSAGE: String = "ACCESS_MESSAGE"
    }

    var mStartForResult: ActivityResultLauncher<Intent?>? =
registerForActivityResult(StartActivityForResult()) {
            result ->
                val textView = findViewById<TextView>(R.id.textView)

                if (result.resultCode == RESULT_OK) {
                    val intent: Intent? = result.data
                    val accessMessage: String? =
intent!!.getStringExtra(ACCESS_MESSAGE)
                    textView.text = accessMessage
                }
                else {
                    textView.text = "Ошибка доступа"
                }
    }

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

    fun onButtonClick(view: View){
        var ageBox: EditText = findViewById(R.id.editTextAge)
        var age: String = ageBox.text.toString()

        var intent: Intent = Intent(this@MainActivity,
SecondActivity::class.java)
        intent.putExtra(AGE_KEY, age)

        mStartForResult?.launch(intent)
    }
}
```

Вкратце рассмотрим главные моменты этого кода. Прежде всего, мы определяем объект **ActivityResultLauncher**, с помощью которого будем запускать вторую **activity** и передавать ей данные:

```
var mStartForResult: ActivityResultLauncher<Intent?>? =
registerForActivityResult(StartActivityForResult()) {
        result ->
            val textView = findViewById<TextView>(R.id.textView)

            if (result.resultCode == RESULT_OK) {
```

```
                val intent: Intent? = result.data
                val accessMessage: String? =
intent!!.getStringExtra(ACCESS_MESSAGE)
                textView.text = accessMessage
            }
            else {
                textView.text = "Ошибка доступа"
            }
}
```

Объект **ActivityResultLauncher** типизируется типом **Intent**, так как объект этого типа будет передаваться в метод **launch()** при запуске второй activity.

Второй аргумент метода **registerForActivityResult()** - объект **ActivityResultCallback** типизируется типом результата - типом ActivityResult и определяет функцию-колбек **onActivityResult()**, которая получает результат и обрабатывает его. В данном случае обработка состоит в том, что мы выводим в текстовое поле ответ от второй activity.

При обработке мы проверяем полученный код результата:

```
if (result.resultCode == Activity.RESULT_OK)
```

В качестве результата, как правило, применяются встроенные константы **Activity.RESULT_OK** и **Activity.RESULT_CANCELED**. На уровне условностей **Activity.RESULT_OK** означает, что activity успешно обработала запрос, а **Activity.RESULT_CANCELED** - что activity отклонила обработку запроса.

С помощью метода **getData()** результата получаем переданные из второй activity данные в виде объекта Intent:

```
var intent: Intent? = result.data
```

Далее извлекаем из Intent строку, которая имеют ключ **ACCESS_MESSAGE**, и выводим ее в текстовое поле.

Таким образом, мы определили объект **ActivityResultLauncher**. Далее в обработчике нажатия **onClick** с помощью этого объекта запускаем вторую **activity** - **SecondActivity**:

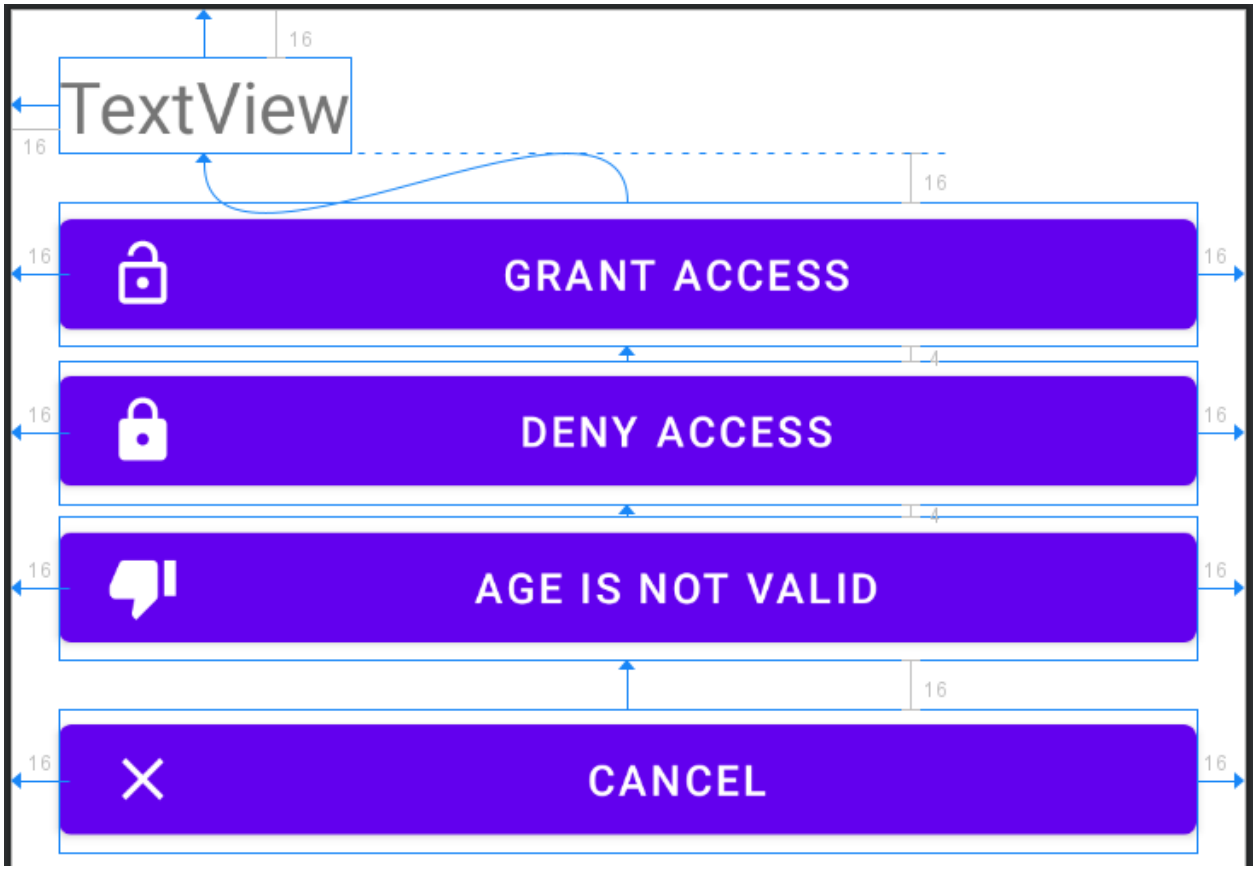
```
fun onButtonClick(view: View){
    var ageBox: EditText = findViewById(R.id.editTextAge)
    var age: String = ageBox.text.toString()

    var intent: Intent = Intent(this@MainActivity,
SecondActivity::class.java)
    intent.putExtra(AGE_KEY, age)
```

```
    mStartForResult?.launch(intent)
}
```

В обработчике нажатия кнопки **onClick()** получаем введенный в текстовое поле возраст, добавляем его в объект **Intent** с ключем **AGE_KEY** и запускаем **SecondActivity** с помощью метода **launch()**

Теперь перейдем к **SecondActivity** и определим в файле **activity_second.xml** набор кнопок:



А в классе **SecondActivity** определим обработчики для этих кнопок:

```
package com.awkitsune.activityresultapidemo

import androidx.appcompat.app.AppCompatActivity

import android.content.Intent;
import android.os.Bundle
import android.view.View
import android.widget.TextView

class SecondActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_second)
```

```
        var extras: Bundle? = intent.extras
        if (extras != null){
            var textViewAge: TextView = findViewById(R.id.textViewAge)
            var age: String? = extras.getString(MainActivity.AGE_KEY)
            if (age!= null){
                textViewAge.text = "Age: $age"
            }
        }
    }
    fun onButtonGrantClick(view: View){
        sendMessage("Access allowed")
    }

    fun onButtonDenyClick(view: View){
        sendMessage("Access denied")
    }

    fun onButtonNotValidClick(view: View){
        sendMessage("Invalid age")
    }

    fun onButtonCancelClick(view: View){
        setResult(RESULT_CANCELED)
        finish()
    }

    fun sendMessage(message: String){
        var data: Intent = Intent()
        data.putExtra(MainActivity.ACCESS_MESSAGE, message)
        setResult(RESULT_OK, data)
        finish()
    }
}
```

Три кнопки вызывают метод **sendMessage()**, в который передают отправляемый ответ. Это и будет то сообщение, которое получить **MainActivity** в методе **onActivityResult**.

Для возврата результата необходимо вызвать метод **setResult()**, в который передается два параметра:

- числовой код результата
- отправляемые данные

После вызова метода **setResult()** нужно вызвать метод **finish**, который уничтожит текущую **activity**.

Одна кнопка вызывает обработчик **onCancelClick()**, в котором передается в **setResult** только код результата - **RESULT_CANCELED**.

То есть условно говоря, мы получаем в **SecondActivity** введенный в **MainActivity** возраст и с помощью нажатия определенной кнопки возвращаем некоторый результат в виде сообщения.

В зависимости от нажатой кнопки на **SecondActivity** мы будем получать разные результаты в **MainActivity**: