

## Лекция #34. Выбор или съёмка изображений в приложении.

Получение изображения с помощью пользовательского ввода - обычная задача в разработке под Android. Однако реализовать его далеко не так просто. На более ранних этапах разработки Android, чтобы позволить пользователю сделать снимок и предварительно просмотреть его, требовалось немало усилий. То же самое касается выбора изображения через приложение галереи.

Но со временем Google предоставляет все больше и больше API, которые упрощают эти задачи и, следовательно, нашу повседневную жизнь как разработчиков. В настоящее время у нас есть доступ к **ActivityResult API**, который позволяет нам использовать **ActivityResultContract API**. На более ранних этапах, если мы хотели получить результат от намерения, нам нужно было переопределить функцию onActivityResult(), установить коды запроса и так далее.

Благодаря API ActivityResult мы теперь можем упростить этот процесс, сэкономить много кода и получить более четкий способ получения данных с помощью **ActivityResultContract**. В этой лекции мы сосредоточимся на следующих двух контрактах:

ActivityResultContracts.TakePicture()  
ActivityResultContracts.GetContent()

Как следует из названия, **TakePicture()** принимает в качестве параметра **Uri** файла. Затем он автоматически открывает собственное приложение камеры, заботится об обработке разрешений и обо всем остальном, о чем нам раньше приходилось заботиться самим.

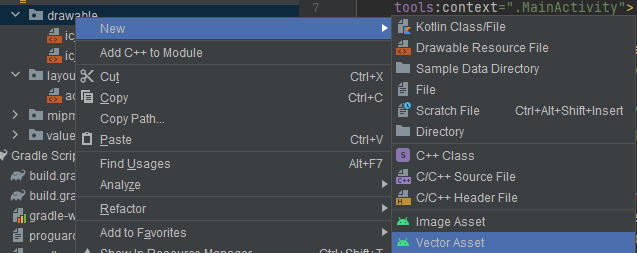
**GetContent()**, с другой стороны, предлагает пользователю выбрать фрагмент контента. В нашем случае мы хотим, чтобы пользователь выбрал изображение из своей галереи. Чтобы указать, какой тип mime следует выбрать, мы можем добавить фильтр типа, например "image/\*"

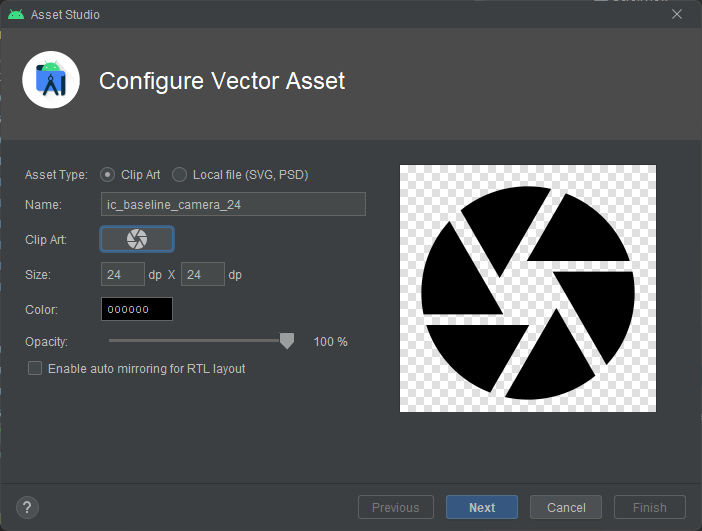
Далее мы реализуем **MaterialCardView** как изображение предварительного просмотра и кнопку, которая запускает наш диалог для выбора изображения через приложение галереи или получения изображения приложением камеры соответствующего устройства. Чтобы дать вам краткое введение в ActivityResult API, мы реализуем упомянутый пример.

Мы начинаем реализацию с настройки нашего проекта с шаблоном EmptyActivity. Откройте файл app-level build.gradle и убедитесь, что включены следующие зависимости:

implementation "androidx.activity:activity-ktx:1.4.0"  
implementation "androidx.fragment:fragment-ktx:1.3.6"

Добавьте в ресурсы drawable векторный ресурс с именем **ic\_baseline\_camera\_24**:





После этого замените макет activity\_main.xml следующим кодом:

<?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">  
  
 <com.google.android.material.card.MaterialCardView  
 android:id="@+id/image\_preview\_card"  
 android:layout\_width="0dp"  
 android:layout\_height="0dp"  
 android:layout\_margin="16dp"  
 app:cardCornerRadius="8dp"  
 app:cardElevation="8dp"  
 app:layout\_constraintBottom\_toTopOf="@+id/linearLayout"  
 app:layout\_constraintDimensionRatio="16:9"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintHorizontal\_bias="0.5"  
 app:layout\_constraintStart\_toStartOf="parent"  
 app:layout\_constraintTop\_toTopOf="parent">  
  
 <ImageView  
 android:layout\_width="80dp"  
 android:layout\_height="80dp"  
 android:layout\_gravity="center"  
 android:alpha="0.25"  
 android:src="@drawable/ic\_baseline\_camera\_24"  
 app:tint="@color/black" />  
  
 <ImageView  
 android:id="@+id/image\_preview"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:scaleType="centerCrop"  
 tools:src="@tools:sample/backgrounds/scenic" />  
 </com.google.android.material.card.MaterialCardView>  
  
 <LinearLayout  
 android:id="@+id/linearLayout"  
 android:layout\_width="0dp"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginEnd="16dp"  
 android:layout\_marginBottom="16dp"  
 android:gravity="center\_vertical"  
 android:orientation="horizontal"  
 app:layout\_constraintBottom\_toBottomOf="parent"  
 app:layout\_constraintEnd\_toEndOf="parent"  
 app:layout\_constraintStart\_toStartOf="parent">  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/take\_image\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginStart="16dp"  
 android:layout\_marginEnd="8dp"  
 android:text="Take image" />  
  
 <com.google.android.material.button.MaterialButton  
 android:id="@+id/select\_image\_button"  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:layout\_marginEnd="16dp"  
 android:text="Select gallery image" />  
 </LinearLayout>  
  
</androidx.constraintlayout.widget.ConstraintLayout>

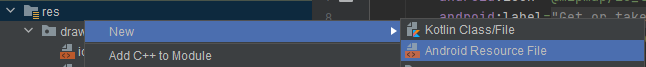
Как видите, у нас есть MaterialCardView для предварительного просмотра изображения. Включенные кнопки отвечают за запуск **TakePicture()** или **GetContent()**.

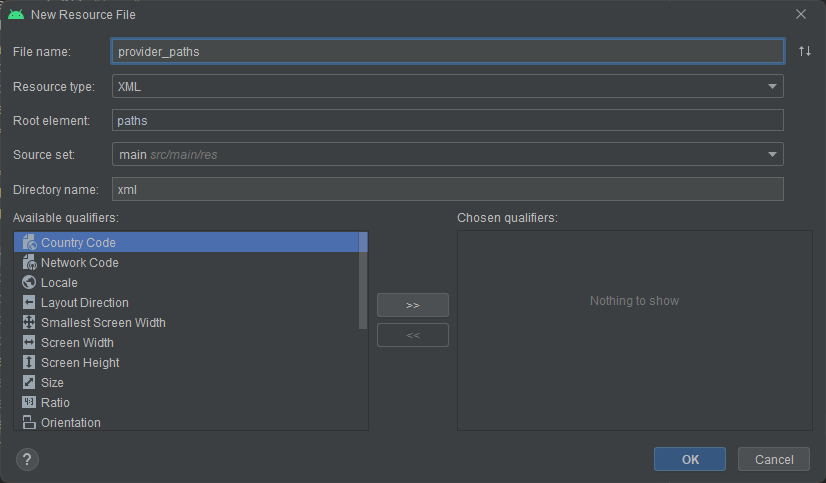
**ActivityResultAPI** работает с так называемыми «контрактами».

В этой лекции мы будем придерживаться заранее подготовленных контрактов, которые предоставляет нам Google. Чтобы продолжить, нам сначала нужно добавить **FileProvider** в наш **AndroidManifest**. Просто скопируйте следующий код:

<application…>  
 <provider  
 android:name="androidx.core.content.FileProvider"  
 android:authorities="${applicationId}.provider"  
 android:exported="false"  
 android:grantUriPermissions="true">  
 <meta-data  
 android:name="android.support.FILE\_PROVIDER\_PATHS"  
 android:resource="@xml/provider\_paths" />  
 </provider>  
</application>

Затем создайте новую папку **XML** в каталоге значений и добавьте файл с именем **provider\_paths.xml**:





<?xml version="1.0" encoding="utf-8"?>  
<paths xmlns:android="http://schemas.android.com/apk/res/android">  
 <cache-path  
 name="cached\_files"  
 path="." />  
 <files-path  
 name="images"  
 path="." />  
</paths>

Добавив их, приложение теперь может взаимодействовать с файловой системой. Теперь мы готовы к собственно логике. Адаптируйте код вашей MainActivity к следующему:

package com.awkitsune.getortakepicture  
  
import android.net.Uri  
import android.os.Bundle  
import android.widget.ImageView  
import androidx.activity.result.contract.ActivityResultContracts  
import androidx.appcompat.app.AppCompatActivity  
import androidx.core.content.FileProvider  
import androidx.lifecycle.*lifecycleScope*import com.google.android.material.button.MaterialButton  
import java.io.File  
  
class MainActivity : AppCompatActivity() {  
 private val previewImage by *lazy* **{** findViewById<ImageView>(R.id.*image\_preview*) **}** private var latestTempUri: Uri? = null  
  
 private val takeImageResult = registerForActivityResult(ActivityResultContracts.TakePicture()) **{** isSuccess **->** if (isSuccess) {  
 latestTempUri?.*let* **{** uri **->** previewImage.setImageURI(uri)  
 **}** }  
 **}** private val selectImageFromGalleryResult = registerForActivityResult(ActivityResultContracts.GetContent()) **{** uri: Uri? **->** uri?.*let* **{** previewImage.setImageURI(uri) **}  
 }** private fun getTmpFileUri(): Uri {  
 val tmpFile = File.createTempFile("tmp\_image\_file", ".png", *cacheDir*).*apply* **{** createNewFile()  
 deleteOnExit()  
 **}** return FileProvider.getUriForFile(  
 *applicationContext*,  
 "${BuildConfig.*APPLICATION\_ID*}.provider",  
 tmpFile  
 )  
 }  
  
 private fun takeImage() {  
 *lifecycleScope*.launchWhenStarted **{** getTmpFileUri().*let* **{** uri **->** latestTempUri = uri  
 takeImageResult.launch(uri)  
 **}  
 }** }  
  
 private fun selectImageFromGallery() = selectImageFromGalleryResult.launch("image/\*")  
  
 override fun onCreate(savedInstanceState: Bundle?) {  
 super.onCreate(savedInstanceState)  
 setContentView(R.layout.*activity\_main*)  
  
 setClickListeners()  
 }  
  
 private fun setClickListeners() {  
 findViewById<MaterialButton>(R.id.*take\_image\_button*).setOnClickListener **{** takeImage() **}** findViewById<MaterialButton>(R.id.*select\_image\_button*).setOnClickListener **{** selectImageFromGallery() **}** }  
}

Как видите, реализация довольно проста. Мы определяем контракты как их собственные переменные. Очевидно, **takeImageResult** означает получение изображения через приложение камеры. **selectImageFromGallerResult** для получения изображения через приложение галереи соответственно.

