Navigation externe

1. Objectifs

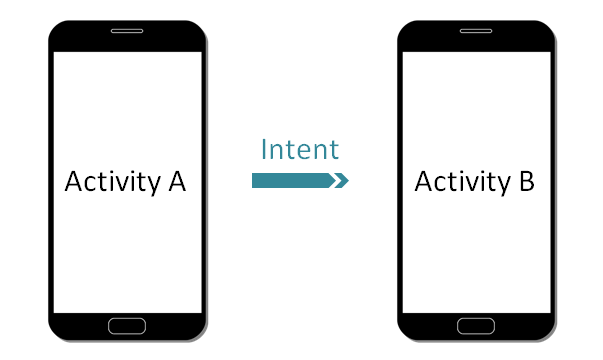
Savoir faire appel à des contrôleurs externes.

1. Intent
   1. Présentation

Permet de faire une demande d’action pour :

* Démarrer une activité
* Démarrer un service
* Diffuser un message à d’autres applications
  1. Intent explicite

Un intent explicite peut demander à une activité A de lancer une activité B se trouvant dans la même application.



package fr.eni.t2

​

import android.content.Intent

import androidx.appcompat.app.AppCompatActivity

import android.os.Bundle

import fr.eni.t2.databinding.ActivityMainBinding

​

class MainActivity : AppCompatActivity()

{

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

val bd = ActivityMainBinding.inflate(layoutInflater)

setContentView(bd.root)

​

bd.btnEnvoyer.setOnClickListener {

var intent = Intent(this, SecondActivity::class.java)

startActivity(intent)

}

}

}

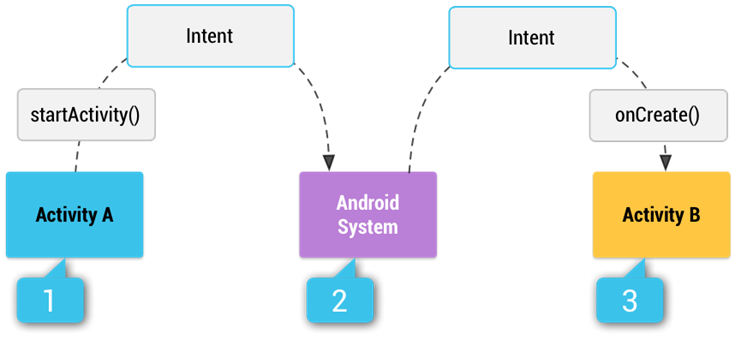
* 1. Intent implicite

Les intentions implicites ne spécifient pas directement les composants Android qui doivent être appelés, elles spécifient uniquement les actions à exécuter. Un Uri peut être utilisé avec l'intention implicite afin de spécifier le type de données sur lequel une action est à effectuer.

Par exemple :

Intent intent = new Intent(ACTION\_VIEW,Uri.parse("http://www.eni-ecole.fr"));

Cela ouvrira le navigateur Web pour ouvrir la page www.eni-ecole.fr. Si plusieurs composants différents sont trouvés alors l'utilisateur devra sélectionner le composant à utiliser.



class MainActivity2 : AppCompatActivity() {

​

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

val bd = ActivityMain2Binding.inflate(layoutInflater)

setContentView(bd.root)

​

bd.btnEnvoyer.setOnClickListener {

startActivity(intent)

var intent = Intent(Intent.ACTION\_VIEW, Uri.parse("https://www.eni-ecole.fr"))

this.startActivity(intent)

}

}

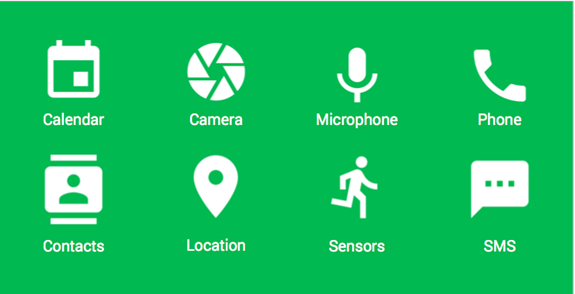
}

* 1. Permissions

Les autorisations d'application contribuent à protéger la confidentialité des utilisateurs en protégeant l'accès aux éléments suivants :

Données restreintes, telles que l'état du système et les coordonnées d'un utilisateur.

Actions restreintes, telles que la connexion à un appareil couplé et l'enregistrement audio.



La première chose à faire pour avoir la permission d'accéder à un élément est d'ajouter la permission necessaire dans le ficheir AndroidManifest.xml.

<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android"

package="fr.eni.envoiunsms">

​

<uses-permission android:name="android.permission.CAMERA" />

​

<application

android:allowBackup="true"

android:icon="@mipmap/ic\_launcher"

android:label="@string/app\_name"

android:roundIcon="@mipmap/ic\_launcher\_round"

android:supportsRtl="true"

android:theme="@style/Theme.EnvoiUnSMS">

<activity

android:name=".MainActivity"

android:exported="true">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

</application>

​

</manifest>

IHM contenant le bouton pour afficher la caméra :

<?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">

​

<ImageView android:id="@+id/camera\_icon"

android:layout\_width="48dp"

android:layout\_height="48dp"

android:layout\_marginEnd="12dp"

android:src="@drawable/ic\_launcher\_background"

app:layout\_constraintStart\_toStartOf="parent"

app:layout\_constraintEnd\_toStartOf="@+id/camera\_lbl"

app:layout\_constraintTop\_toTopOf="parent"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintHorizontal\_chainStyle="packed"/>

​

<TextView android:id="@+id/camera\_lbl"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Camera"

android:textSize="24sp"

app:layout\_constraintBottom\_toBottomOf="parent"

app:layout\_constraintStart\_toEndOf="@+id/camera\_icon"

app:layout\_constraintEnd\_toEndOf="parent"

app:layout\_constraintTop\_toTopOf="parent" />

​

</androidx.constraintlayout.widget.ConstraintLayout>

Contrôleur pour gérer la caméra :

package fr.eni.envoiunsms

​

import android.Manifest

import android.content.Intent

import android.net.Uri

import android.os.Bundle

import android.provider.MediaStore

import android.provider.Settings

import androidx.activity.result.contract.ActivityResultContracts

import androidx.appcompat.app.AppCompatActivity

import com.google.android.material.snackbar.BaseTransientBottomBar.LENGTH\_LONG

import com.google.android.material.snackbar.Snackbar

import fr.eni.envoiunsms.databinding.ActivityMainBinding

​

class MainActivity : AppCompatActivity() {

​

private val binding: ActivityMainBinding by lazy { ActivityMainBinding.inflate(layoutInflater) }

private val cameraPermission =

registerForActivityResult(ActivityResultContracts.RequestPermission()) { granted ->

with(binding.root) {

when {

granted -> useCamera()

shouldShowRequestPermissionRationale(Manifest.permission.CAMERA) -> {

//this option is available starting in API 23

Snackbar.make(this, "Permission denied, show more info!", LENGTH\_LONG).show()

}

else -> Snackbar.make(this, "Permission denied", LENGTH\_LONG).show()

​

}

}

}

​

override fun onCreate(savedInstanceState: Bundle?) {

super.onCreate(savedInstanceState)

setContentView(binding.root)

​

binding.cameraIcon.setOnClickListener {

cameraPermission.launch(Manifest.permission.CAMERA)

//openSettings()

}

}

​

fun useCamera()

{

val cameraIntent = Intent(MediaStore.ACTION\_IMAGE\_CAPTURE)

startActivity(cameraIntent)

}

​

private fun openSettings() {

Intent(Settings.ACTION\_APPLICATION\_DETAILS\_SETTINGS).apply {

addCategory(Intent.CATEGORY\_DEFAULT)

data = Uri.parse("package:$packageName")

}.run(::startActivity)

}

}

1. Conclusion

Dans ce chapitre nous avons vu comment faire appel à une ressource externe, comment avoir des droits permettant de bien faire fonctionner une application, notamment en demandant spécifiquement à l'utilisateur les droits nécessaires.