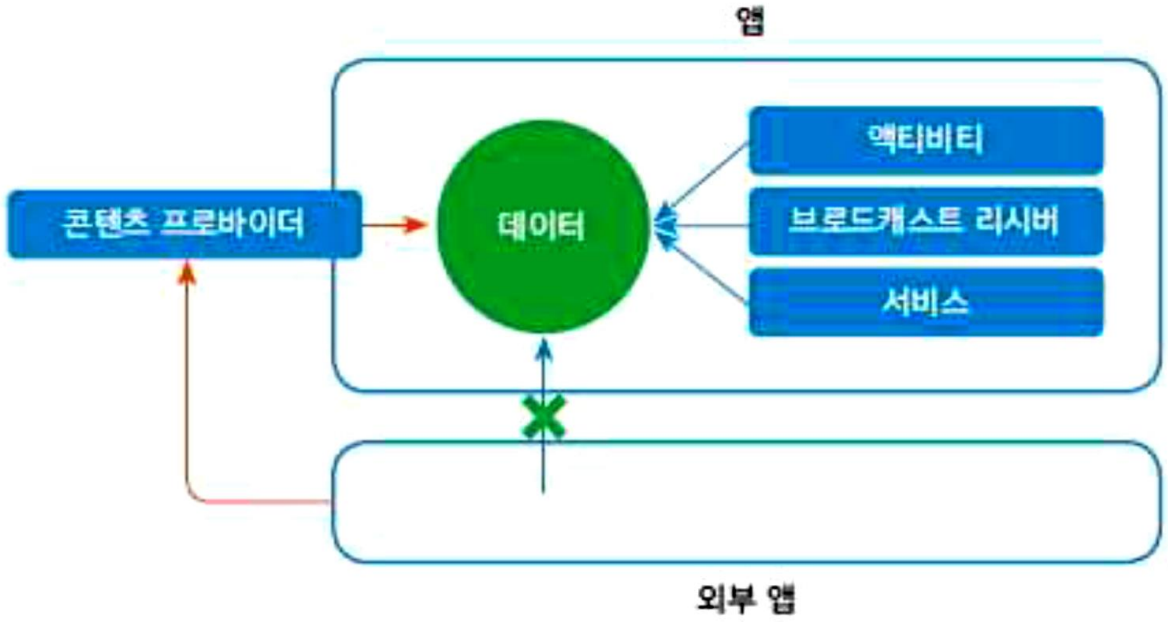
14.콘텐츠 프로바이더 컴포넌트

1. 콘텐츠 프로바이더 이해하기
2. 안드로이드 기본 앱과 연동하기
3. 카메라, 갤러리 앱과 연동하는 앱 만들기

1. 콘텐츠 프로바이더 이해하기

- 콘텐츠 프로바이더는 앱끼리 데이터를 연동하는 컴포넌트



1. 콘텐츠 프로바이더 이해하기

- 콘텐츠 프로바이더 작성하기
  - 콘텐츠 프로바이더는 ContentProvider 클래스를 상속
  - onCreate(), getType(), query(), insert(), update(), delete() 함수를 재정의해서 작성

```
class MyContentProvider : ContentProvider() {
    override fun delete(uri: Uri, selection: String?, selectionArgs: Array<String>?): Int {
        return 0
    }

    override fun getType(uri: Uri): String? {
        return null
    }

    override fun insert(uri: Uri, values: ContentValues?): Uri? {
        return null
    }

    override fun onCreate(): Boolean {
        return false
    }
```

```
    override fun query(
        uri: Uri, projection: Array<String>?, selection: String?,
        selectionArgs: Array<String>?, sortOrder: String?
    ): Cursor? {
        return null
    }

    override fun update(
        uri: Uri, values: ContentValues?, selection: String?,
        selectionArgs: Array<String>?
    ): Int {
        return 0
    }
}
```

1. 콘텐츠 프로바이더 이해하기

■ 콘텐츠 프로바이더 작성하기

- 콘텐츠 프로바이더도 안드로이드 컴포넌트이므로 매니페스트에 등록
- name 속성뿐만 아니라 authorities 속성도 반드시 선언
- authorities 속성은 외부에서 이 콘텐츠 프로바이더를 이용할 때 식별값

• 매니페스트에 콘텐츠 프로바이더 등록

```
<provider
    android:name=".MyContentProvider"
    android:authorities="com.example.test_provider"
    android:enabled="true"
    android:exported="true"></provider>
```

1. 콘텐츠 프로바이더 이해하기

콘텐츠 프로바이더 이용하기

- 콘텐츠 프로바이더는 인텐트와 상관이 없습니다.
- 외부 앱에서 콘텐츠 프로바이더를 사용하려면 먼저 매니페스트에 해당 앱에 관한 패키지 공개 설정
- 콘텐츠 프로바이더를 사용할 때는 ContentResolver 객체를 이용
  - public final int delete(Uri url, String where, String[] selectionArgs)
  - public final Uri insert(Uri url, ContentValues values)
  - public final Cursor query(Uri uri, String[] projection, String selection,String[] selectionArgs, String sortOrder)

• 패키지 공개 설정

```
<queries>
    <!--   둘 중 하나만 선언하면 됩니다.  -->
    <!--   <provider android:authorities="com.example.test_provider" /> -->
    <package android:name="com.example.test_outter" />
</queries>
```

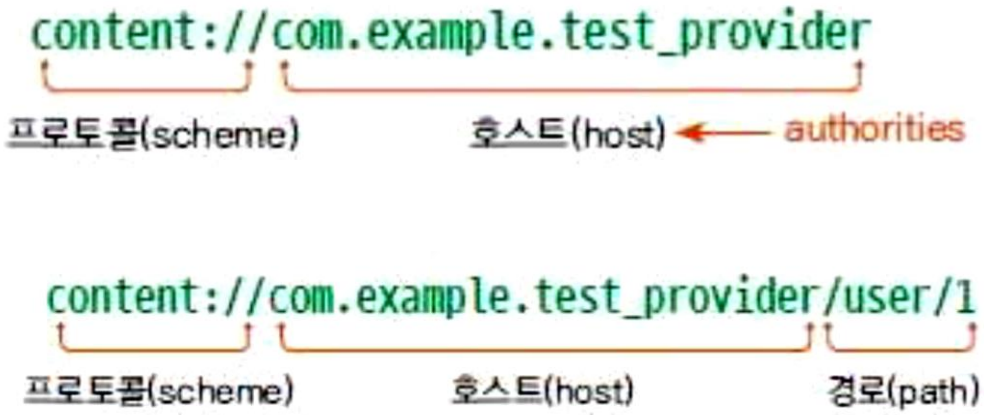
• 시스템의 콘텐츠 프로바이더 사용

```
contentResolver.query(
    Uri.parse("content://com.example.test_provider"),
    null, null, null, null)
```

1. 콘텐츠 프로바이더 이해하기

콘텐츠 프로바이더 이용하기

- Uri 객체의 URL 문자열은 프로토콜명과 콘텐츠 프로바이더의 식별자로 등록된 authorities값



2. 안드로이드 기본 앱과 연동하기

- 주소록 앱 연동하기
  - 주소록 앱에서 데이터를 가져오려면 퍼미션을 설정

• 주소록 앱 사용 퍼미션 설정

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

  - 주소록의 목록 화면을 띄우는 코드
    - ContactsContract.Contacts.CONTENT_URI: 모든 사람의 데이터
    - ContactsContract.CommonDataKinds.Phone.CONTENT_URI: 전화번호가 있는 사람

• 주소록 목록 출력

```
val intent = Intent(Intent.ACTION_PICK, ContactsContract.CommonDataKinds.Phone.CONTENT_URI)
requestContactsLauncher.launch(intent)
```

· 사람

2. 안드로이드 기본 앱과 연동하기

- 주소록 앱 연동하기
  - 목록에서 한 사람을 선택하여 되돌아오면 ActivityResultCallback의 onActivityResult() 함수가 자동으로 실행

• 주소록에서 사용자가 한 사람을 선택했을 때 실행되는 함수

```
requestContactsLauncher = registerForActivityResult(
    ActivityResultContracts.StartActivityForResult())
{
    if (it.resultCode == RESULT_OK) {
        Log.d("kkang", "${it.data?.data}")
    }
}
```

  - 주소록 앱이 전달한 문자열

```
content://com.android.contacts/contacts/lookup/0r82-270667DB186FDB1C89B9.3114i12251c758
ec39928/1144
```

2. 안드로이드 기본 앱과 연동하기

- **주소록 앱 연동하기**
  - 식별값을 조건으로 주소록 앱에 필요한 데이터를 구체적으로 다시 요청

• 콘텐츠 프로바이더로 필요한 데이터 요청

```
requestContactsLauncher = registerForActivityResult(
    ActivityResultContracts.StartActivityForResult())
{
    if (it.resultCode == RESULT_OK) {
        val cursor = contentResolver.query(
            it!!.data!!.data!!,
            arrayOf<String>(
                ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME,
                ContactsContract.CommonDataKinds.Phone.NUMBER
            ),
            null,
            null,
            null
        )
        Log.d("kkang", "cursor size....${cursor?.count}")
        if (cursor!!.moveToFirst()) {
            val name = cursor?.getString(0)
            val phone = cursor?.getString(1)
            binding.resultContact.text = "name: $name, phone: $phone"
        }
    }
}
```

2. 안드로이드 기본 앱과 연동하기

갤러리 앱 연동하기

- 갤러리 앱 연동은 인텐트로 갤러리 앱의 목록 화면을 띄우거나 갤러리 앱의 콘텐츠 프로바이더로 데이터를 가져오는 작업
- 이미지 작업 시 고려 사항
  - 안드로이드에서 이미지는 Drawable이나 Bitmap 객체로 표현합니다.
  - Bitmap 객체는 BitmapFactory로 생성합니다.
  - BitmapFactory로 이미지를 생성할 때는 OOM 오류를 고려해야 합니다.
  - Glide나 Picasso 같은 이미지 처리 라이브러리를 이용하는 것이 효율적일 수 있습니다.
- Bitmap 이미지는 BitmapFactory 클래스의 'decode'로 시작하는 다음과 같은 함수로 생성
  - BitmapFactory.decodeByteArray(): byte[] 배열의 데이터로 비트맵 생성
  - BitmapFactory.decodeFile(): 파일 경로를 매개변수로 지정하면 그 파일에서 데이터를 읽을 수 있는 FileInputStream을 만들어 decodeStream() 함수 이용
  - BitmapFactory.decodeResource(): 리소스 이미지로 비트맵 생성
  - BitmapFactory.decodeStream(): InputStream으로 읽은 데이터로 비트맵 생성

2. 안드로이드 기본 앱과 연동하기

■ 갤러리 앱 연동하기

- OOM이란 앱의 메모리가 부족해서 발생하는 오류
- 용량이 큰 이미지를 불러올 때 발생
- 이미지 크기를 줄일 때는 BitmapFactory.Option 객체의 inSampleSize 속성을 이용

• 옵션을 지정해 비트맵 생성

```
val option = BitmapFactory.Options()
option.inSampleSize = 4
val bitmap = BitmapFactory.decodeStream(inputStream, null, option)
```

- 갤러리 앱 연동 방법
  - 앱의 사진 목록을 출력하는 코드

• 사진 목록 출력

```
val intent = Intent(Intent.ACTION_PICK, MediaStore.Images.Media.EXTERNAL_CONTENT_URI)
intent.type = "image/*"
requestGalleryLauncher.launch(intent)
```

2. 안드로이드 기본 앱과 연동하기

▪ 갤러리 앱 연동하기

- 이미지 정보를 얻어 실제 화면에 출력되는 크기와 비교해서 inSampleSize값을 계산

• 적절한 비율로 이미지 크기 줄이기

```
private fun calculateInSampleSize(fileUri: Uri, reqWidth: Int, reqHeight: Int): Int {
    val options = BitmapFactory.Options()
    options.inJustDecodeBounds = true
    try {
        var inputStream = contentResolver.openInputStream(fileUri)
        BitmapFactory.decodeStream(inputStream, null, options)
        inputStream!!.close()
        inputStream = null
    } catch (e: Exception) {
        e.printStackTrace()
    }
```

옵션만 설정하고자 true로 지정합니다.

각종 이미지 정보가 옵션에 설정됩니다.

```
    val (height: Int, width: Int) = options.run { outHeight to outWidth }
    var inSampleSize = 1
    // inSampleSize 비율 계산
    if (height > reqHeight || width > reqWidth) {
        val halfHeight: Int = height / 2
        val halfWidth: Int = width / 2
        while (halfHeight / inSampleSize >= reqHeight &&
                halfWidth / inSampleSize >= reqWidth) {
            inSampleSize *= 2
        }
    }
    return inSampleSize
}
```

2. 안드로이드 기본 앱과 연동하기

- 이미지 불러오는 코드

• 이미지를 불러오는 코드

```
requestGalleryLauncher = registerForActivityResult(
    ActivityResultContracts.StartActivityForResult())
{
    try {
        // inSampleSize 비율 계산, 지정
        val calRatio = calculateInSampleSize(it!!.data!!.data!!,
            resources.getDimensionPixelSize(R.dimen.imgSize),
            resources.getDimensionPixelSize(R.dimen.imgSize))
        val option = BitmapFactory.Options()
        option.inSampleSize=calRatio

        // 이미지 로딩
        var inputStream = contentResolver.openInputStream(it!!.data!!.data!!)
        val bitmap = BitmapFactory.decodeStream(inputStream, null, option)
        inputStream!!.close()
        inputStream = null
        bitmap?. let {
            binding.galleryResult.setImageBitmap(bitmap)
        } ?: let {
            Log.d("kkang", "bitmap null")
        }
    } catch (e: Exception) {
        e.printStackTrace()
    }
}
```

2. 안드로이드 기본 앱과 연동하기

■ 카레라 앱 연동하기

- 카메라 앱을 연동하여 사진을 촬영하고 그 결과를 돌려받는 방법은 다음 2가지
  - 사진 데이터를 가져오는 방법
  - 사진 파일을 공유하는 방법
- 사진 데이터를 가져오는 방법
  - 인텐트로 카메라 앱의 사진 촬영 액비티비를 실행

• 사진 촬영 액티비티 실행

```
val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
requestCameraThumbnailLauncher.launch(intent)
```

2. 안드로이드 기본 앱과 연동하기

■ 카메라 앱 연동하기

- 카메라 앱에서 넘어온 사진 데이터는 ActivityResult Callback에서 획득

• 사진 데이터 가져오기

```
requestCameraThumbnailLauncher = registerForActivityResult(
    ActivityResultContracts.StartActivityForResult())
{
    val bitmap = it?.data?.extras?.get("data") as Bitmap
}
```

2. 안드로이드 기본 앱과 연동하기

■ 카메라 앱 연동하기

- 사진 파일을 공유하는 방법

1. 앱에서 사진을 저장할 파일을 만듭니다.
2. 사진 파일 정보를 포함한 인텐트를 전달해 카메라 앱을 실행
3. 카메라 앱으로 사진을 촬영하여 공유된 파일에 저장
4. 카메라 앱을 종료하면서 성공 또는 실패를 반환
5. 카메라 앱이 저장한 사진 파일을 앱에서 이용

- 사진 파일을 공유하는 방법을 이용하려면 먼저 앱에서 외장 메모리에 파일을 생성
- getExternalStoragePublicDirectory() 또는 getExternal FilesDir() 함수를 이용

• 외장 메모리 사용 퍼미션

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

2. 안드로이드 기본 앱과 연동하기

■ 카메라 앱 연동하기

▪ 사진파일 공유하는 방법

- API 레벨 24 버전부터는 file:// 프로토콜로 구성된 URI를 외부에 노출하지 못함
- 앱끼리 파일을 공유하려면 content:// 프로토콜을 이용하고 이 URI에 임시로 접근할 수 있는 권한을 부여
- FileProvider 클래스는 androidx 라이브러리에서 제공하며 XML 설정을 기반으로 해서 content:// 프로토콜로 구성된 URI를 생성해 줌

• 파일 프로바이더용 XML 파일

```
<paths xmlns:android="http://schemas.android.com/apk/res/android">
    <external-path name="myfiles" path="Android/data/com.example.test16/files/Pictures" />
</paths>
```

앱의 패키지명

2. 안드로이드 기본 앱과 연동하기

■ 카메라 앱 연동하기

▪ 사진파일 공유하기

▪ 파일 프로바이더용 XML 파일을 매니페스트 파일에 등록

• 매니페스트에 파일 프로바이더용 XML 파일 등록

```
<provider
    android:name="androidx.core.content.FileProvider"
    android:authorities="com.example.test16.fileprovider"
    android:exported="false"
    android:grantUriPermissions="true">
    <meta-data
        android:name="android.support.FILE_PROVIDER_PATHS"
        android:resource="@xml/file_paths"></meta-data>
</provider>
```

2. 안드로이드 기본 앱과 연동하기

- **카메라 앱 연동하기**
  - 사진파일 공유하기
    - 파일 생성

• 파일 만들기

```
val timeStamp: String = SimpleDateFormat("yyyyMMdd_HHmmss").format(Date())
val storageDir: File? = getExternalFilesDir(Environment.DIRECTORY_PICTURES)
val file = File.createTempFile(
    "JPEG_${timeStamp}_",
    ".jpg",
    storageDir
)
filePath = file.absolutePath
```

- FileProvider를 이용해 Uri 객체를 만들고 이를 카메라 앱을 실행하는 인텐트의 엑스트라 데이터로 설정

• 카메라 앱을 실행하는 인텐트

```
val photoURI: Uri = FileProvider.getUriForFile(
    this,
    "com.example.test16.fileprovider", file
)
val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
intent.putExtra(MediaStore.EXTRA_OUTPUT, photoURI);
requestCameraFileLauncher.launch(intent)
```

2. 안드로이드 기본 앱과 연동하기

■ 카메라 앱 연동하기

▪ 사진파일 공유하기

▪ 카메라 앱에서 사진을 촬영한 후 다시 앱으로 돌아왔을 때 실행되는 코드

• 비트맵 이미지 생성

```
requestCameraFileLauncher = registerForActivityResult(
    ActivityResultContracts.StartActivityForResult())
{
    val option=BitmapFactory.Options()
    option.inSampleSize = 10
    val bitmap = BitmapFactory.decodeFile(filePath, option)
    bitmap?.let {
        binding.cameraFileResult.setImageBitmap(bitmap)
    }
}
```

2. 안드로이드 기본 앱과 연동하기

- 지도 앱 연동하기

• 지도 앱을 실행하는 인텐트

```
val intent =
    Intent(Intent.ACTION_VIEW, Uri.parse("geo:
    37.5662952,126.9779451"))
startActivity(intent)
```

2. 안드로이드 기본 앱과 연동하기

■ 전화 앱 연동하기

• 전화를 거는 퍼미션 설정

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

• 전화 앱을 실행하는 인텐트

```
val intent = Intent(Intent.ACTION_CALL, Uri.parse("tel:02-120"))
startActivity(intent)
```

실습 : 카메라, 갤러리 앱과 연동하는 앱 만들기

- 1단계. 모듈 생성하고 빌드 그래들 작성하기
  - Ch16_Provider라는 이름으로새로운모듈을만듭니다.
  - 뷰바인딩기법을이용하도록설정
- 2단계. 실습 파일 복사하기
  - res 디렉터리 아래에 drawable, layout, values 디렉터리를 Ch16_Provider 모듈의 같은 위치에 복사
  - 코틀린 파일이 있는 디렉터리에서 MainActivity.kt 파일을 복사
- 3단계. 파일 프로바이더용 XML 작성하기
  - res에 xml 디렉터리를 만들고 그 아래에 file_paths.xml 파일을 만들어 코드를 작성
- 4단계. 매니페스트 작성하기
  - 카메라 앱과 파일 정보를 공유하기 위해 매니페스트 파일에 <provider>를 추가

실습 : 카메라, 갤러리 앱과 연동하는 앱 만들기

- 5단계. 메인 액티비티 작성하기
  - MainActivity.kt 파일을 열고 코드를 작성
- **6단계. 앱 실행하기**