Lecture 14: Camera
IN721: Design and Development of Applications for Mobile Devices
Semester One, 2020

Kaiako: Grayson Orr

Te Kura Matatini ki Otago, Ōtepoti, Aotearoa

Wednesday, 6 May

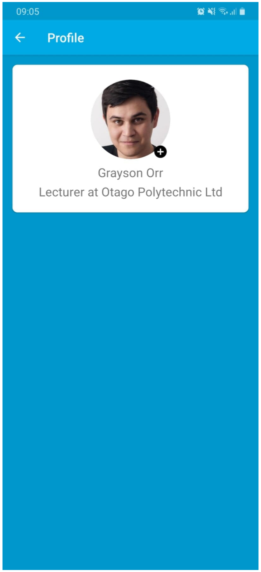
Lecture 13: SQLite Topics

- SQLite
- Formatting date & time

Lecture 14: Camera Topics

- Camera API

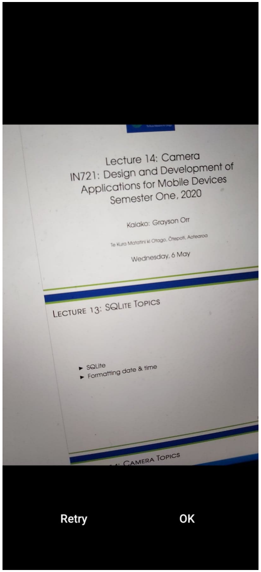
Today's Practical


09:05

Profile

Grayson Orr

Lecturer at Otago Polytechnic Ltd


Lecture 14: Camera
IN721: Design and Development of Applications for Mobile Devices
Semester One, 2020

Kaiako: Grayson Orr

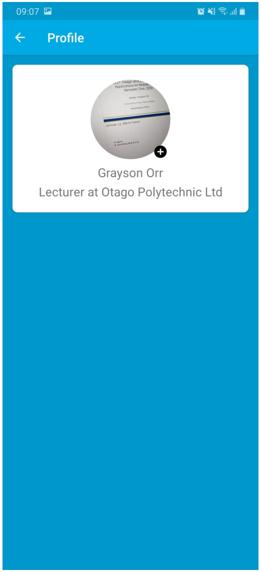
Te Kura Matatini ki Otago, Ōtepoti, Aotearoa

Wednesday, 6 May

Lecture 13: SQLite Topics

- SQLite
- Formatting date & time

Camera Topics


09:07

Profile

Grayson Orr

Lecturer at Otago Polytechnic Ltd

Camera

- Support for various cameras & camera features available on devices
- Enables users to capture pictures & videos in their applications

Camera - Considerations

- Camera requirements - does your application requirement a camera?
- Customized camera - if yes, how will your application use the camera?
- Foreground services requirement - when does your application interact with the camera?
- Storage - what the accessibility constraints for your images?

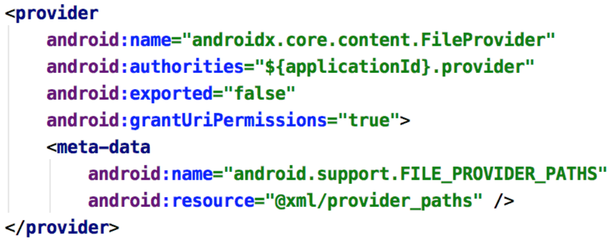
Camera - Permissions

- Declare new permissions in AndroidManifest.xml
  - Camera
  - Storage
- Other permissions to consider:
  - Audio recording
  - Location

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

CAMERA - PROVIDER

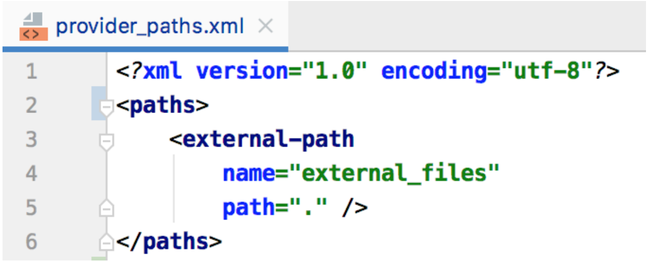
- Declare provider in AndroidManifest.xml
- @xml/provider_paths



```xml
<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>
```

Camera - Provider

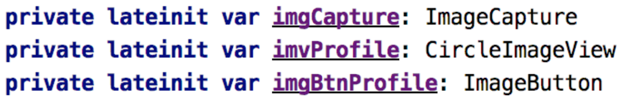
- Create a new file called provider_paths.xml in xml resource directory



```xml
<?xml version="1.0" encoding="utf-8"?>
<paths>
    <external-path
        name="external_files"
        path="." />
</paths>
```

CAMERA

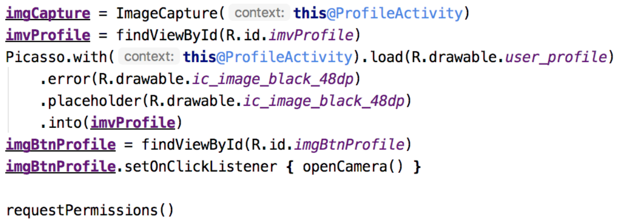
- Create a class called ProfileActivity.kt
- Extends BaseActivity.kt
- Declare the following variables:



```kotlin
private lateinit var imgCapture: ImageCapture
private lateinit var imvProfile: CircleImageView
private lateinit var imgBtnProfile: ImageButton
```

Camera

- ProfileActivity.kt onCreate
- openCamera - opens the camera via an intent
- requestPermissions - we will look at this soon

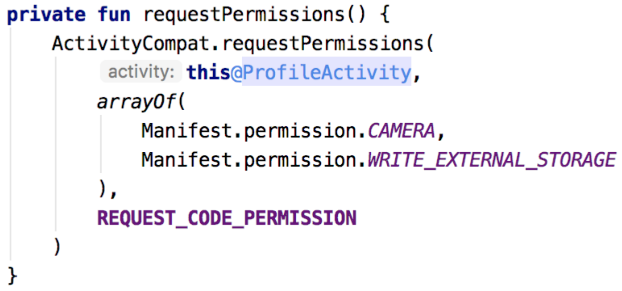


```
imgCapture = ImageCapture( context: this@ProfileActivity)
imvProfile = findViewById(R.id.imvProfile)
Picasso.with( context: this@ProfileActivity).load(R.drawable.user_profile)
    .error(R.drawable.ic_image_black_48dp)
    .placeholder(R.drawable.ic_image_black_48dp)
    .into(imvProfile)
imgBtnProfile = findViewById(R.id.imgBtnProfile)
imgBtnProfile.setOnClickListener { openCamera() }

requestPermissions()
```

Camera

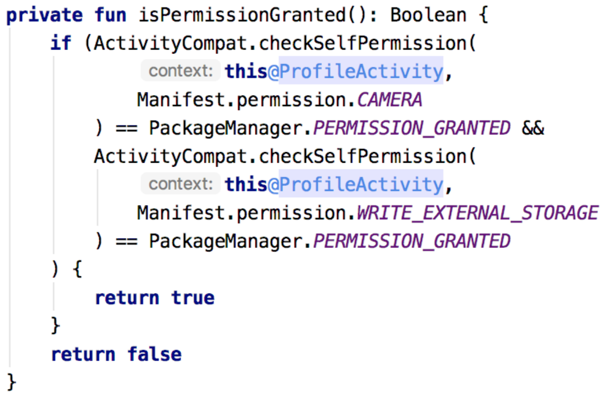
- Private function - requestPermissions
- ArrayList of permissions - we declare the permissions in AndroidManifest.xml
- REQUEST_CODE_PERMISSIONS = 1
  - Create a companion object or constant
  - Used to check user action for corresponding request
  - User will be prompt to allow or deny these permissions



```
private fun requestPermissions() {
    ActivityCompat.requestPermissions(
        activity: this@ProfileActivity,
        arrayOf(
            Manifest.permission.CAMERA,
            Manifest.permission.WRITE_EXTERNAL_STORAGE
        ),
        REQUEST_CODE_PERMISSION
    )
}
```

Camera

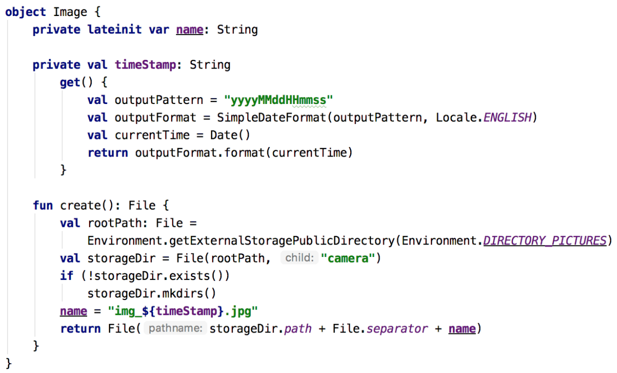
- Private function - isPermissionGranted
- Check if camera & write external storage permissions have been allowed/denied by the user
  - Returns a boolean value



```
private fun isPermissionGranted(): Boolean {
    if (ActivityCompat.checkSelfPermission(
            context: this@ProfileActivity,
            Manifest.permission.CAMERA
        ) == PackageManager.PERMISSION_GRANTED &&
        ActivityCompat.checkSelfPermission(
            context: this@ProfileActivity,
            Manifest.permission.WRITE_EXTERNAL_STORAGE
        ) == PackageManager.PERMISSION_GRANTED
    ) {
        return true
    }
    return false
}
```

Camera

- Create Image object
- Declare two variables - image's name & timestamp (getter)
- Function - create
  - Environment - access to environment variables



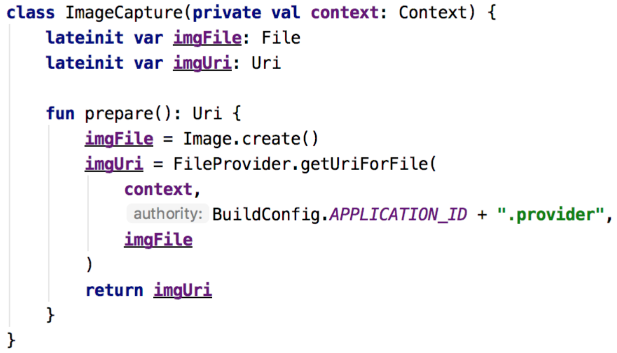
```kotlin
object Image {
    private lateinit var name: String

    private val timeStamp: String
        get() {
            val outputPattern = "yyyyMMddHHmmss"
            val outputFormat = SimpleDateFormat(outputPattern, Locale.ENGLISH)
            val currentTime = Date()
            return outputFormat.format(currentTime)
        }

    fun create(): File {
        val rootPath: File =
            Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_PICTURES)
        val storageDir = File(rootPath,  child: "camera")
        if (!storageDir.exists())
            storageDir.mkdirs()
        name = "img_${timeStamp}.jpg"
        return File( pathname: storageDir.path + File.separator + name)
    }
}
```

Camera

- Create a class called ImageCapture
- Function - prepare
  - Get the File object
  - FileProvider - facilitates secure sharing of files
    - Creates a content URI instead of a file URI
    - Content URI - allows you to grant read & write access using temp permissions
    - getUriForFile - returns a content URI for a given File object

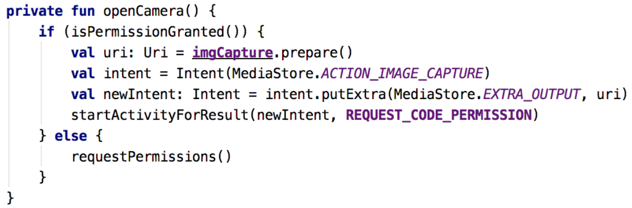


```kotlin
class ImageCapture(private val context: Context) {
    lateinit var imgFile: File
    lateinit var imgUri: Uri

    fun prepare(): Uri {
        imgFile = Image.create()
        imgUri = FileProvider.getUriForFile(
            context,
            authority: BuildConfig.APPLICATION_ID + ".provider",
            imgFile
        )
        return imgUri
    }
}
```

Camera

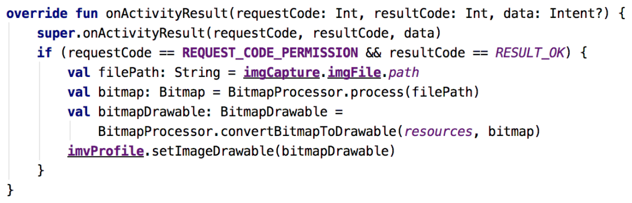
- Private function - openCamera
- Check if the permissions have been allowed by the user
- Open camera via an intent
  - Indicate a content resolver Uri
  - Used to store the requested image or video
- Call override function - startActivityForResult
  - Pass the intent to start & request code as its arguments
  - Request code - if >= 0, this code will be returned in onActivityResult when the activity exit
- If permissions have not been allowed by the user, call requestPermissions



```
private fun openCamera() {
    if (isPermissionGranted()) {
        val uri: Uri = imgCapture.prepare()
        val intent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
        val newIntent: Intent = intent.putExtra(MediaStore.EXTRA_OUTPUT, uri)
        startActivityForResult(newIntent, REQUEST_CODE_PERMISSION)
    } else {
        requestPermissions()
    }
}
```

Camera

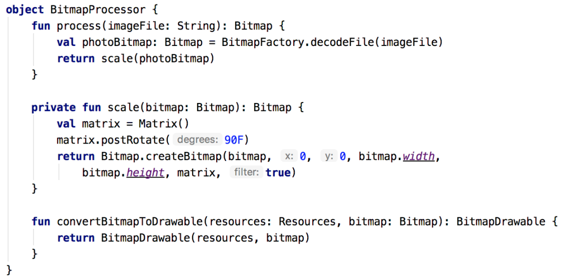
- Override function - onActivityResult
- Check if the request code = REQUEST_CODE_PERMISSIONS & result code equals RESULTS_OK
- Get the file path
- Process file path into bitmap
- Convert the bitmap into a drawable
- Set circle image view to the drawable



```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)
    if (requestCode == REQUEST_CODE_PERMISSION && resultCode == RESULT_OK) {
        val filePath: String = imgCapture.imgFile.path
        val bitmap: Bitmap = BitmapProcessor.process(filePath)
        val bitmapDrawable: BitmapDrawable =
            BitmapProcessor.convertBitmapToDrawable(resources, bitmap)
        imvProfile.setImageDrawable(bitmapDrawable)
    }
}
```

Camera

- Create an object called BitmapProcessor
- function - process
  - BitmapFactory - creates Bitmap objects
  - decodeFile - decode a file path into a bitmap
- private function - scale
  - Matrix - create an identity matrix
  - postRotate - post-concatenates the matrix with the specified rotation
- function - convertBitmapToDrawable



```
object BitmapProcessor {
    fun process(imageFile: String): Bitmap {
        val photoBitmap: Bitmap = BitmapFactory.decodeFile(imageFile)
        return scale(photoBitmap)
    }

    private fun scale(bitmap: Bitmap): Bitmap {
        val matrix = Matrix()
        matrix.postRotate( degrees: 90F)
        return Bitmap.createBitmap(bitmap,  x: 0,  y: 0, bitmap.width,
            bitmap.height, matrix,  filter: true)
    }

    fun convertBitmapToDrawable(resources: Resources, bitmap: Bitmap): BitmapDrawable {
        return BitmapDrawable(resources, bitmap)
    }
}
```

Camera - Alternatives

- Camera2
- CameraX
- There are variety of camera libraries written on top of Camera, Camera2 & CameraX APIs
- Resources - Code Samples

Practical

- Series of tasks covering today’s lecture
- Worth 2% of your final mark for the Design and Development of Applications for Mobile Devices course
- Deadline: Friday, 12 June at 5pm