LECTURE 18: MAKING RUNTIME PERMISSION REQUESTS IN ANDROID

BY LINA HAMMAD & AHMAD BARGHASH

In this lecture, we will learn, how to request a dangerous READ_EXTERNAL_STORAGE permission at run time, which is necessary since Android 6.0 (Marshmallow), whereas before this permission would be granted at the installation process. For this we will use the checkSelfPermission and requestPermissions methods and show a rationale to the user at the appropriate time with the shouldShowRequestPermissionRationale method.

HOW TO REQUEST PERMISSIONS IN ANDROID APPLICATION?

- Starting from **Android 6.0 (API 23)**, users are not asked for permissions at the time of installation rather developers need to request for the permissions at the run time. Only the permissions that are **defined in the manifest file** can be requested at run time.

- Types of Permissions:

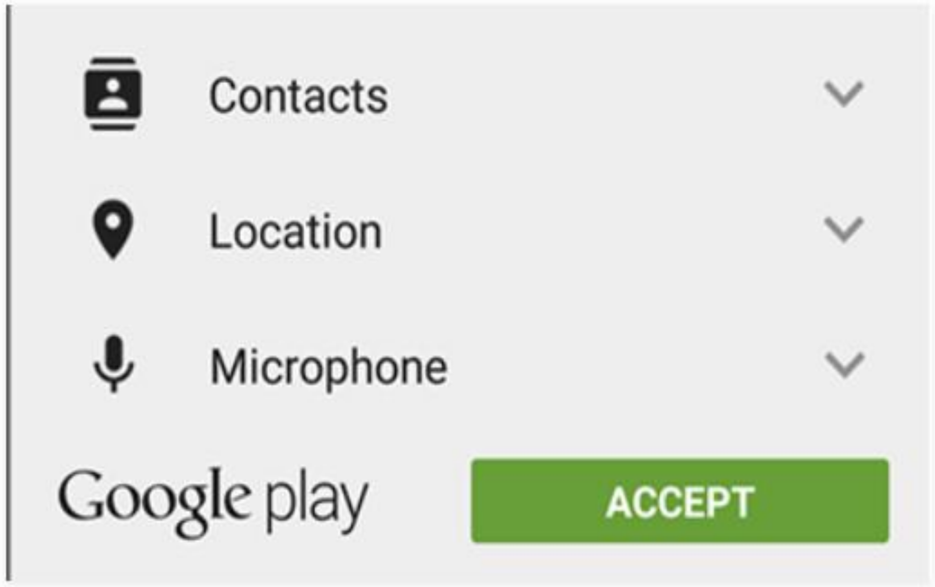
1. **Install-Time Permissions:** If the Android 5.1.1 (API 22) or lower, the permission is requested at the installation time at the Google Play Store.

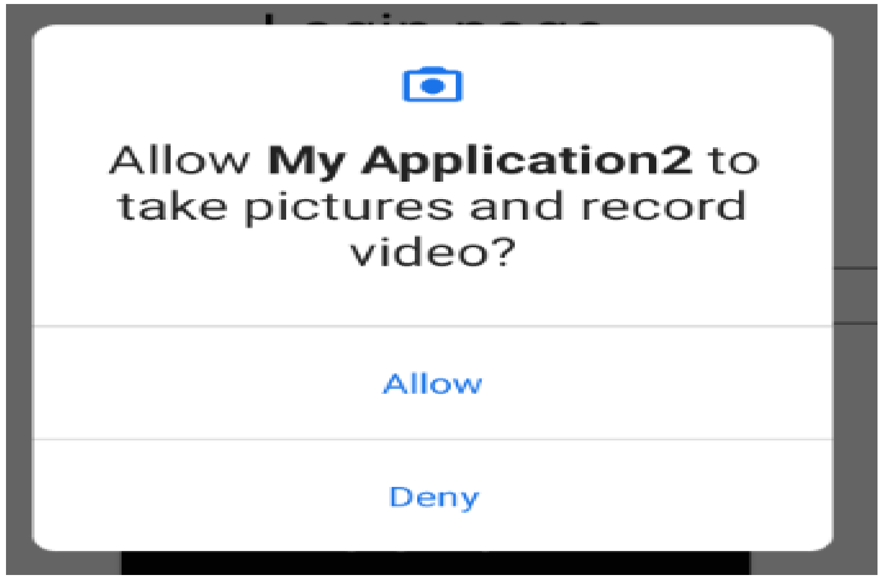
   - If the user **Accepts** the permissions, the app is installed. Else the app **installation is cancelled**.

2. **Run-Time Permissions:** If the Android 6 (API 23) or higher, the permission is requested at the run time during the running of the app.

   - If the user **Accepts** the permissions, then that feature of the app can be used. Else to use the feature, the app **requests the permission again**.

TYPES OF PERMISSIONS




Allow **My Application2** to take pictures and record video?

Install-Time Permissions from Google Play Store

Run-Time Permissions during the runnnig of the app

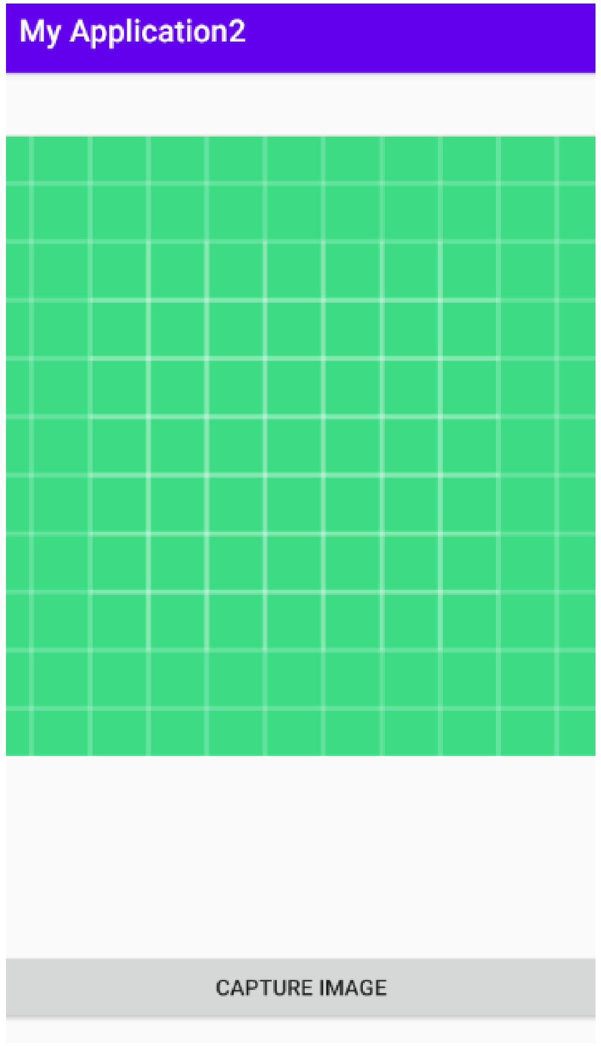
STEPS FOR REQUESTING PERMISSIONS AT RUN TIME

- In this article, we will discuss how to request permissions in an Android Application at run time.

- **Steps for Requesting permissions at run time :**
  1. Declare the permission in Android Manifest file.
  2. Modify activity_main.xml file.
  3. Modify MainActivity.kt by Checking whether permission is already granted or not. If permission isn't already granted, request user for the permission.

EXAMPLE OF THIS LECTURE

- This application that we want to build in this lecture:



DECLARE THE PERMISSION IN ANDROID MANIFEST FILE

- In Android permissions are declared in **AndroidManifest.xml** file using the **uses-permission** tag.
- <uses-permission android:name=”android.permission.PERMISSION_NAME”/>
- In the following example we are declaring storage and camera permission.

```
<!--Declaring the required permissions-->
<!--Adding Camera, Write External Storage Permission-->
<uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

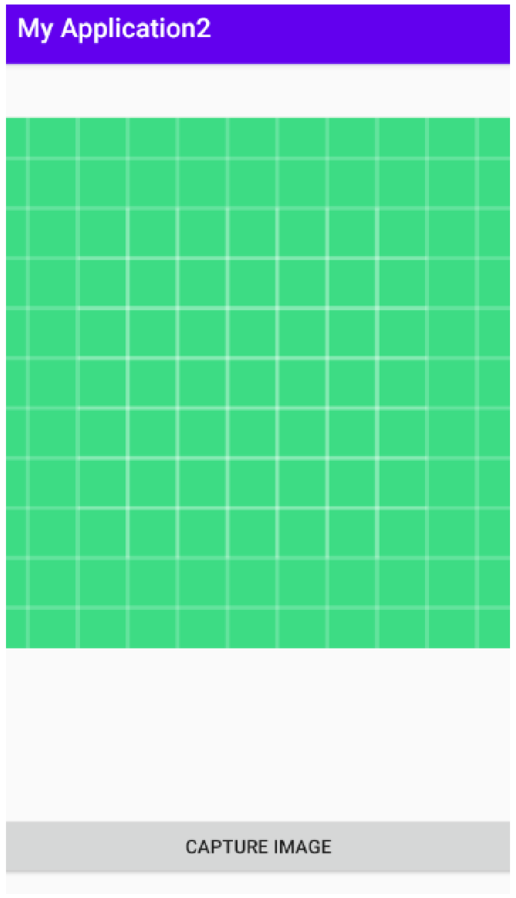
- Note: declare the required permissions before the application tag.

MODIFY ACTIVITY_MAIN.XML

- Modify activity_main.xml file to add one button to capture a picture and request permission on button click and add an ImageView to view the captured image.

```
<ImageView
    android:id="@+id/image_view"
    android:layout_width="match_parent"
    android:layout_height="396dp"
    android:scaleType="centerCrop"
    android:src="@drawable/ic_launcher_background"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<Button
    android:id="@+id/capture_btn"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="32dp"
    android:text="Capture Image"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/image_view" />
```



MODIFY MAINACTIVITY.KT

- Now, in MainActivity.kt we will need to:

1. Check whether permission is granted or not. If permission isn't already granted, request user for the permission.
   - In order to use any service or feature, the permissions are required. Hence, we have to ensure that the permissions are given for that. If not, then the permissions are requested.
2. We will program the button to open the phone camera, capture an image, save it in the gallery, return the captured image to the application and view it in the ImageView.

CHECK FOR PERMISSIONS

- **Check for permissions:** Beginning with Android 6.0 (API level 23), the user has the right to revoke permissions from any app at any time, even if the app targets a lower API level. So to use the service, the app needs to check for permissions every time.
- Syntax:

```
//if system os is Marshmallow or Above, we need to request runtime permission
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M){// M is the version code; it is equals to 23. SDK_INT
is the version
    if (checkSelfPermission(Manifest.permission.CAMERA) == PackageManager.PERMISSION_DENIED ||
        checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE) ==
PackageManager.PERMISSION_DENIED){
        //permission was not enabled
       //show popup to request permission
    }
    else{
        //permission already granted
    }
}
else{
    //system OS is < marshmallow, so there is no need to request a run time permission

}
```

REQUEST PERMISSIONS

- **Request Permissions:** When **PERMISSION_DENIED** is returned from the **checkSelfPermission()** method in the above syntax, we need to prompt the user for that permission. Android provides several methods that can be used to request permission, such as **requestPermissions()**.
- Syntax:

```
//permission was not enabled
val permission = arrayOf(Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE)

//show popup to request permission
requestPermissions(permission, PERMISSION_CODE) //PERMISSION_CODE is a global variable with value equals to 1000
```

CHECK AND REQUEST PERMISSIONS

```
//button click
var capture_btn = findViewById<Button>(R.id.capture_btn)
capture_btn.setOnClickListener {
    //if system OS is Marshmallow or Above, we need to request runtime permission
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M){
        if (checkSelfPermission(Manifest.permission.CAMERA) == PackageManager.PERMISSION_DENIED ||
            checkSelfPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE) == PackageManager.PERMISSION_DENIED){

            //permission was not enabled
            val permission = arrayOf(Manifest.permission.CAMERA, Manifest.permission.WRITE_EXTERNAL_STORAGE)
            //show popup to request permission
            requestPermissions(permission, PERMISSION_CODE) //PERMISSION_CODE is a global variable with value equals to 1000
        }
        else{
            //permission already granted
            openCamera()
        }
    }
    else{
        //system OS is < marshmallow
        openCamera()
    }
}
```

OVERRIDE ON REQUEST PERMISSIONS RESULT()

- Override **onRequestPermissionsResult()** method**: onRequestPermissionsResult()** is called when user grant or decline the permission. **RequestCode** is one of the parameteres of this function which is used to check user action for corresponding request. Here a toast message is shown indicating the permission and user action.
- As we mentioned in the previous slides, a permission request is made via a call to the requestPermissions() method of the ActivityCompat class. When this method is called, the permission request is handled asynchronously, and a method named onRequestPermissionsResult() is called when the task is completed.
- The requestPermissions() method takes as arguments a reference to the current activity, together with the identifier of the permission being requested and a request code. The request code can be any integer value and will be used to identify which request has triggered the call to the onRequestPermissionsResult() method.

OVERRIDE ON REQUEST PERMISSION SRESULT()

- **onRequestPermissionsResult Syntax:**

```
override fun onRequestPermissionsResult(requestCode: Int, permissions: Array<out String>, grantResults: IntArray) {
    //called when user presses ALLOW or DENY from Permission Request Popup
    when(requestCode){
        PERMISSION_CODE -> { //PERMISSION_CODE is a global variable with value equals to 1000
            if (grantResults.size > 0 && grantResults[0] == PackageManager.PERMISSION_GRANTED){
                //permission from popup was granted
                openCamera()
            }
            else{
                //permission from popup was denied
                Toast.makeText(this, "Permission denied", Toast.LENGTH_SHORT).show()
            }
        }
    }
}
```

OPEN CAMERA FUNCTION

```kotlin
fun openCamera() {
    val values = ContentValues()
    values.put(MediaStore.Images.Media.TITLE, "New Picture") //title of the captured image
    values.put(MediaStore.Images.Media.DESCRIPTION, "From the Camera") //description of the captured image
    image_uri = contentResolver.insert(MediaStore.Images.Media.EXTERNAL_CONTENT_URI, values) //image_uri is a
         global variable, declared as the following var image_uri: Uri? = null


    //camera intent
    val cameraIntent = Intent(MediaStore.ACTION_IMAGE_CAPTURE)
    cameraIntent.putExtra(MediaStore.EXTRA_OUTPUT, image_uri)
    startActivityForResult(cameraIntent, IMAGE_CAPTURE_CODE) //IMAGE_CAPTURE_CODE is a global variable with value equals to 1000
}
```

- Notes:
  - ContentValues class is used to store a set of values that the ContentResolver can process.
  - `image_uri` variable to send all the values information to the camera application.

VIEW THE CAPTURED IMAGE IN THE IMAGEVIEW

```
override fun onActivityResult(requestCode: Int, resultCode: Int, data: Intent?) {
    super.onActivityResult(requestCode, resultCode, data)

    //called when image was captured from camera intent
    if (resultCode == Activity.RESULT_OK){
        //set image captured to image view
        image_view.setImageURI(image_uri)
    }
}
```

FINAL RESULT

THE FULL LIST OF PERMISSIONS

- The full list of permissions that fall into the dangerous category is contained in the following table:

Permission Group	Permission
Calendar	READ_CALENDAR WRITE_CALENDAR
Camera	CAMERA
Contacts	READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS
Location	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION
Microphone	RECORD_AUDIO

Permission Group	Permission
Phone	SMS SEND_SMS RECEIVE_SMS READ_SMS RECEIVE_WAP_PUSH RECEIVE_MMS
Storage	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE

SUMMARY

- Prior to the introduction of Android 6.0 the only step necessary for an app to request permission to access certain functionality was to add an appropriate line to the application's manifest file. The user would then be prompted to approve the permission at the point that the app was installed. This is still the case for most permissions, with the exception of a set of permissions that are considered dangerous. Permissions that are considered dangerous usually have the potential to allow an app to violate the user's privacy such as allowing access to the microphone, contacts list or external storage.

- As outlined in this lecture, apps based on Android 6 or later must now request dangerous permission approval from the user when the app launches in addition to including the permission request in the manifest file.