UNIVERSITÀ
DEGLI STUDI
DE L'AQUILA

disim

Mobile Programming Laboratory

ANDROID
Permissions

Teachers

Ing. Tarquini Francesco, Ph.D

Ph.D in Computer Science Engineering

francesco.tarquini@univaq.it

Ing. D'Errico Leonardo

Ph.D Student in Computer Science Engineering

leonardo.derrico@graduate.univaq.it

Teaching Materials

Available on MOODLE platform

http://www.didattica.univaq.it

Google Drive Repository

https://drive.google.com/drive/folders/1ISqZfn0i9Ub3eWNXbvW00rd0hD9ya8OL?usp=sharing

Topics

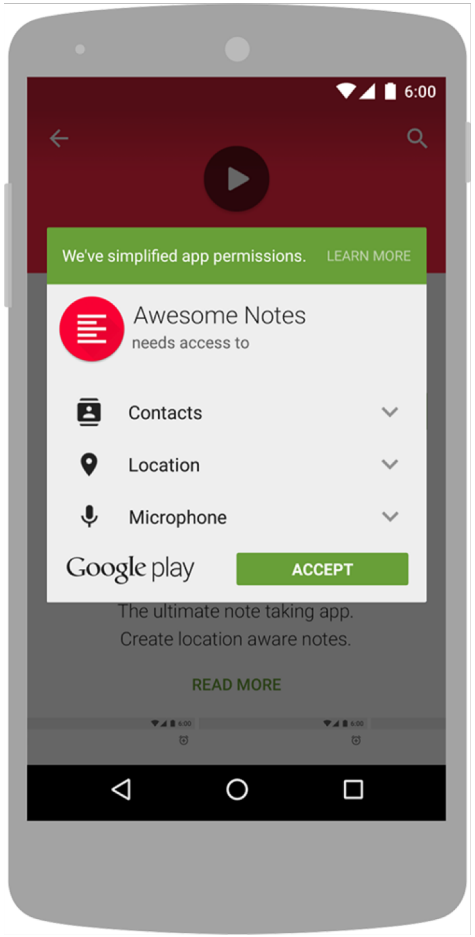
- Permissions
  - Manifest
  - Runtime permissions

Permissions

The purpose of a permission is to protect the privacy of an Android user.

Android apps must request permission to access sensitive user data (such as contacts and SMS), as well as certain system features (such as camera and internet).

Depending on the feature, the system might grant the permission automatically or might prompt the user to approve the request.



Permissions - Manifest

An app must publicize the **permissions** it requires by including **<uses-permission>** tags in the **app manifest**.

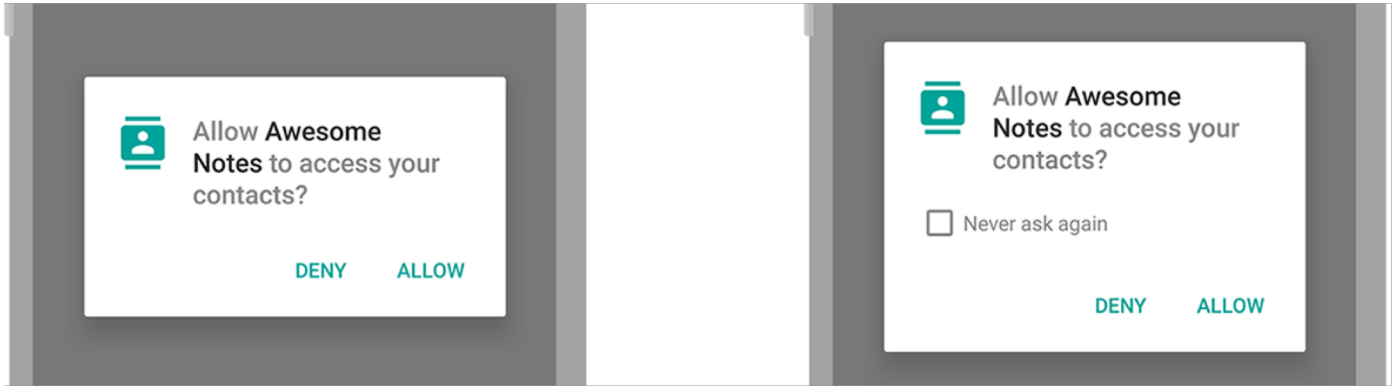
If your app lists **normal** permissions in its manifest (that is, permissions that don't pose much risk to the user's privacy or the device's operation), the system automatically grants those permissions to your app.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
  package="it.univaq.mobileprogramming.helloword">

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

        ...

</manifest>
```

Permissions - Runtime Permissions

If your app lists **dangerous** permissions in its manifest (that is, permissions that could potentially affect the user's privacy or the device's normal operation), such as the SEND_SMS permission above, the user must explicitly agree to grant those permissions.

For example, the ability to read the user's contacts is a dangerous permission. If an app declares that it needs a dangerous permission, the user has to explicitly grant the permission to the app. Until the user approves the permission, your app cannot provide functionality that depends on that permission.



Permissions - Runtime Permissions

If your app lists **dangerous** permissions in its manifest (that is, permissions that could potentially affect the user's privacy or the device's normal operation), such as the SEND_SMS permission above, the user must explicitly agree to grant those permissions.

For example, the ability to read the user's contacts is a dangerous permission. If an app declares that it needs a dangerous permission, the user has to explicitly grant the permission to the app. Until the user approves the permission, your app cannot provide functionality that depends on that permission.

Permissions - Runtime Permissions

GROUP	PERMISSIONS
CALENDAR	READ_CALENDAR WRITE_CALENDAR
CONTACTS	READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE CALL_PHONE READ_CALL_LOG WRITE_CALL_LOG ADD_VOICEMAIL USE_SIP PROCESS_OUTGOING_CALLS
SENSORS	BODY_SENSORS
SMS	SEND_SMS RECEIVE_SMS READ_SMS RECEIVE_WAP_PUSH RECEIVE_MMS
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE

Permissions - Runtime Permissions

STEP 1: Check of the permissions

The developer has to check if the user granted the permission yet.

To do this, Android introduced in Support Library the **ContextCompat** class and its **checkSelfPermissions()** method

```
int result = ContextCompat.checkSelfPermission(<context>, <permission>);
```

For example:

```
int result = ContextCompat.checkSelfPermission(getApplicationContext(), Manifest.permission.WRITE_EXTERNAL_STORAGE);
```

The returned result is an integer value mapped in the two constants define in PackageManager class

```
PackageManager.PERMISSION_GRANTED
PackageManager.PERMISSION_DENIED
```

Permissions - Runtime Permissions

STEP 2: Request permission

If the app doesn't already have the permission it needs, the app must call one of the requestPermissions() methods.

These methods belongs to **ActivityCompat** class.

Two methods exist:

shouldShowRequestPermissionRationale(): the developer can show to the user an explanation (asynchronously)

requestPermissions(): without explanation

Permissions - Runtime Permissions

STEP 2: Request permission

The requestPermissions() method requires the following inputs:

```
ActivityCompat.requestPermissions(Context context, String[] permissions, int requestCode);
```

For example:

```
ActivityCompat.requestPermissions(getApplicationContext(), new String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, REQUEST_CODE);
```

context: is the application context

permissions: is a String array of all required permissions

requestCode: an integer to identify a specific request

Permissions - Runtime Permissions

STEP 3: Response

The developer can analyze the user response in the AppCompatActivity callback method, onRequestPermissionsResult():

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
  super.onRequestPermissionsResult(requestCode, permissions, grantResults);
}
```

In **grantResults** there are the result values to match with the previous constants of all permissions required in the same order of the String array input named permissions.

The **requestCode** is the same defined in requestPermissions method.