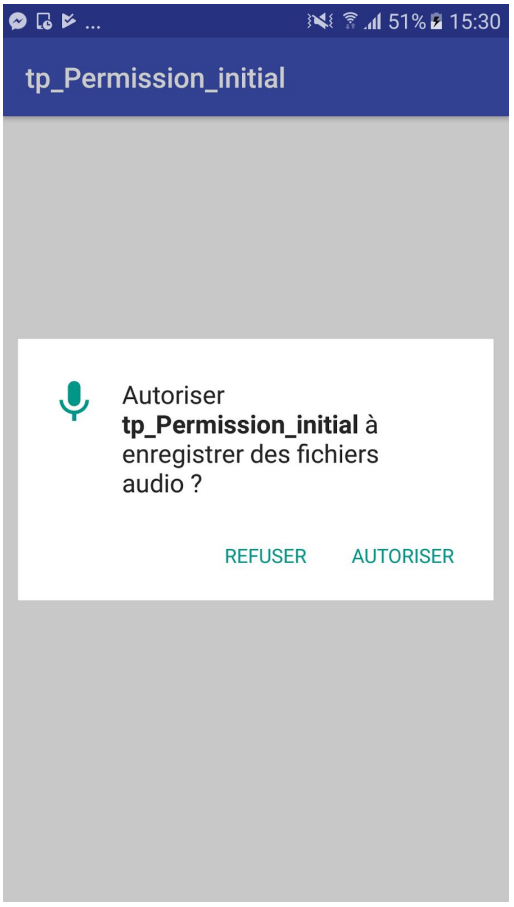
Android-TP: Les Permissions

ELAZZAM Mustaphat FI
24/10/2017 à 14:30

Résumé

Vous vous apprêtez à développer une application Android dans laquelle vous souhaitez accéder à la position, les contacts, l'agenda ou encore les photos? Vous souhaitez savoir si vous avez le droit de le faire et si oui sous quelles conditions?

Sous Android 6 et plus, l'utilisateur confirmera chaque permission pendant l'exécution au run Time (une popup lui demandera d'accorder la permission à votre application) si la permission touche à la vie privée de l'utilisateur. Si ça n'est pas le cas, la permission sera automatiquement accordée. Dans les versions antérieures à Android 6, la permission sera demandée à l'installation de l'application.



Dans ce TP, nous verrons tout d'abord comment déclarer une permission, puis comment demander une permission au run Time, et enfin comment gérer les retours de l'utilisateur

Pré-requis

Les pré-requis nécessaires :

- Savoir programmer en Java
- Connaître les bases en Android
- Etre muni d'un appareil disposant de la version android minimum 6.0.0

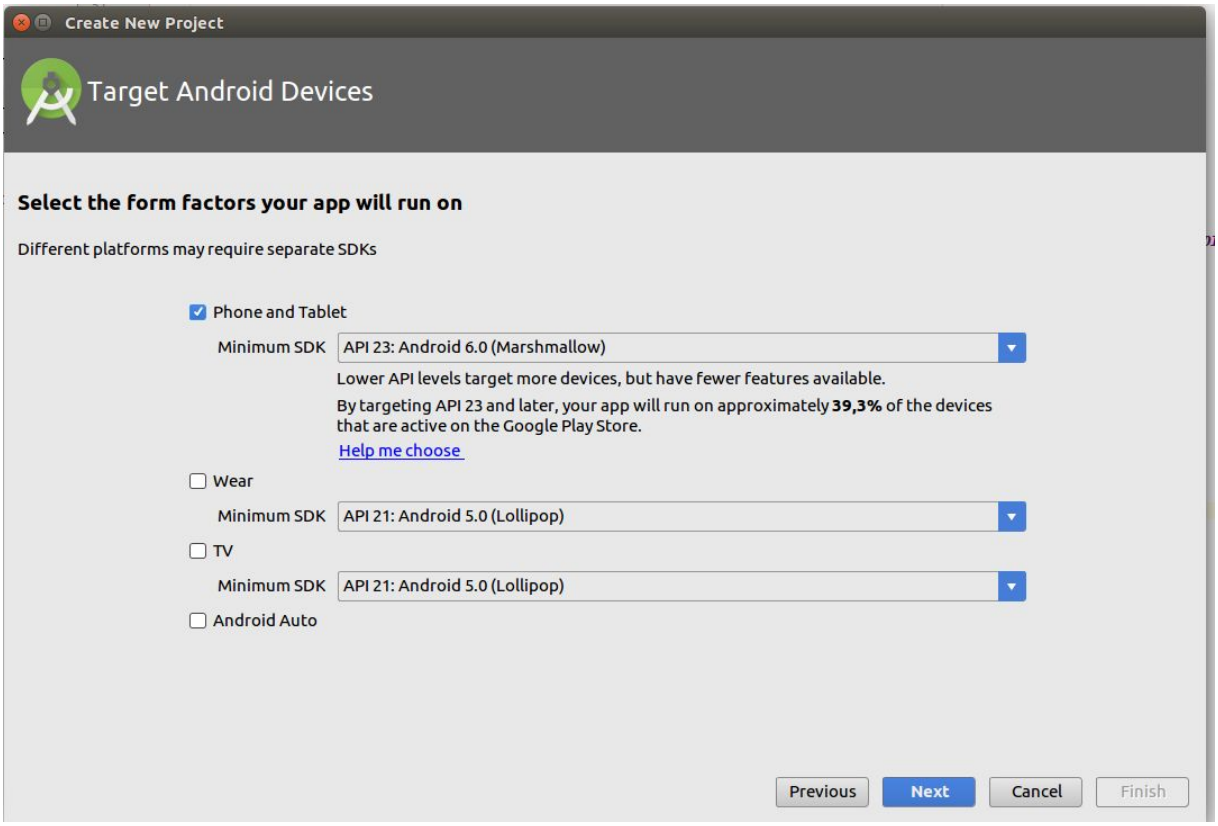
Code source

Code source **initial** disponible à (URL...)

Code source **final** disponible à (URL...)

Explications du TP

Création du projet : Créer un projet SDK 23 "Android 6.0 " (version minimum requise pour que votre application fonctionne).



Etape 1: Déclarer une permission

Première chose à faire après la création du projet, dans le manifest de votre application, utilisez la balise <uses-permission >. C'est partie ! :)

activity_main.xml × | MainActivity.java × | AndroidManifest.xml × | app ×

manifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.elazzam.android_permission_record_final">


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


    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Android_permission_record_final"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

il existe deux types de permissions, dangereuses et normales.

Les permissions dangereuses "touchent à la vie privée de l'utilisateur". Dans ce cas, une popup vous demandera de donner ce type de permission.

Permissions dangereuses

- Agenda
- Appareil photo
- Capteurs corporels
- Contacts
- Microphone
- Position (GPS)
- SMS
- Stockage
- Téléphone

Les permissions normales "ne touchent pas à la vie privée de l'utilisateur". Comme par exemple le flash. Dans ce cas, on déclare la permission et elle sera donnée de manière automatique.

Etape 2 : Demander la Permission au Run Time

A partir d'Android 6.0, l'utilisateur donne les permissions au moment où l'application utilise la fonctionnalité cible. Il peut alors autoriser une permission donnée et en refuser une autre.

1- La fonction **checkSelfPermission()** permet de vérifier si l'application dispose de la permission nécessaire à notre action.

checkSelfPermission

```
int checkSelfPermission (Context context,
                String permission)
```

Determine whether *you* have been granted a particular permission.

Parameters	
`context`	`Context`
`permission`	`String`: The name of the permission being checked.

Returns	
`int`	`PERMISSION_GRANTED` if you have the permission, or `PERMISSION_DENIED` if not.

2-La fonction **shouldShowRequestPermissionRationale()** permet de savoir si une permission a déjà était demandée. Si oui, il est préférable de ne pas la demander de nouveau, ou alors, il vaudrait ajouter une alerte et expliquer à l'utilisateur en quoi l'autorisation est nécessaire et lui proposer d'activer la permission. S'il accepte nous lui affichons la pop-up système.

```
boolean shouldShowRequestPermissionRationale (Activity activity,
                String permission)
```

Parameters	
`activity`	`Activity`: The target activity.
`permission`	`String`: A permission your app wants to request.

Returns	
`boolean`	Whether you can show permission rationale UI.

3- La fonction **RequestPermission()** permet de demander la permission

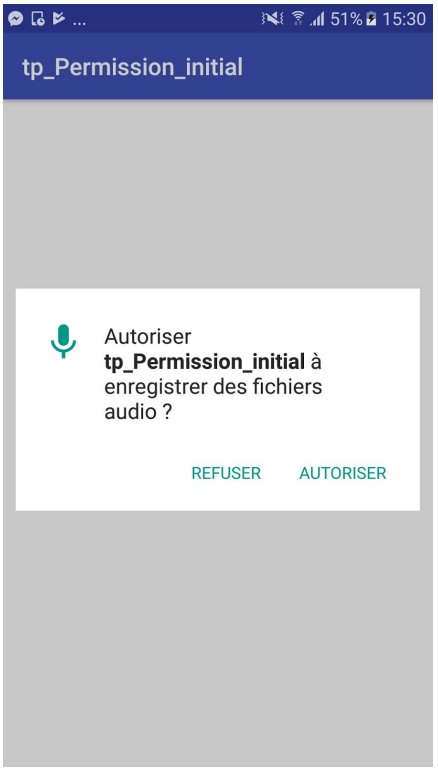
```
void requestPermissions (Activity activity,
                String[] permissions,
                int requestCode)
```

Parameters	
`activity`	`Activity`: The target activity.
`permissions`	`String`: The requested permissions. Must me non-null and not empty.
`requestCode`	`int`: Application specific request code to match with a result reported to `onRequestPermissionsResult(int, String[], int[])`. Should be >= 0.

Exemple d'implémentation:

```
if (ContextCompat.checkSelfPermission(this, Manifest.permission.RECORD_AUDIO)  != PackageManager.PERMISSION_GRANTED){
    if (ActivityCompat.shouldShowRequestPermissionRationale(this, Manifest.permission.RECORD_AUDIO)){
        //permission deja demander mais elle a etait refuser soit on fait rien soit
        //on explique a l utilisateur pk cette permission est obligatoir pour le fonctionnement de l'application
        //avec une alerte si il accepte on redemande la permission

        AlertAudio();

    }else{

        //si on jamais demander la permission c'est l'occasion pour la demander
        ActivityCompat.requestPermissions(this,new String[]{Manifest.permission.RECORD_AUDIO},MY_PERMISSION_REQUEST_RECORD_AUDIO );

    }

}else {

        //permission accorder
}
```

Etape 3: Gérer La réponse de l'utilisateur



A l'affichage de la pop-up système, deux choix sont proposés à l'utilisateur,

- autoriser
- refuser

Nous allons donc devoir récupérer la réponse de l'utilisateur et vérifier que la permission a été acceptée ou refusée.

La méthode **onRequestPermissionResult()** (à surcharger) est un rappel du résultat des demandes d'autorisations. Cette méthode est invoquée pour chaque appel de requestPermissions().

```
void onRequestPermissionsResult (int requestCode,
                String[] permissions,
                int[] grantResults)
```

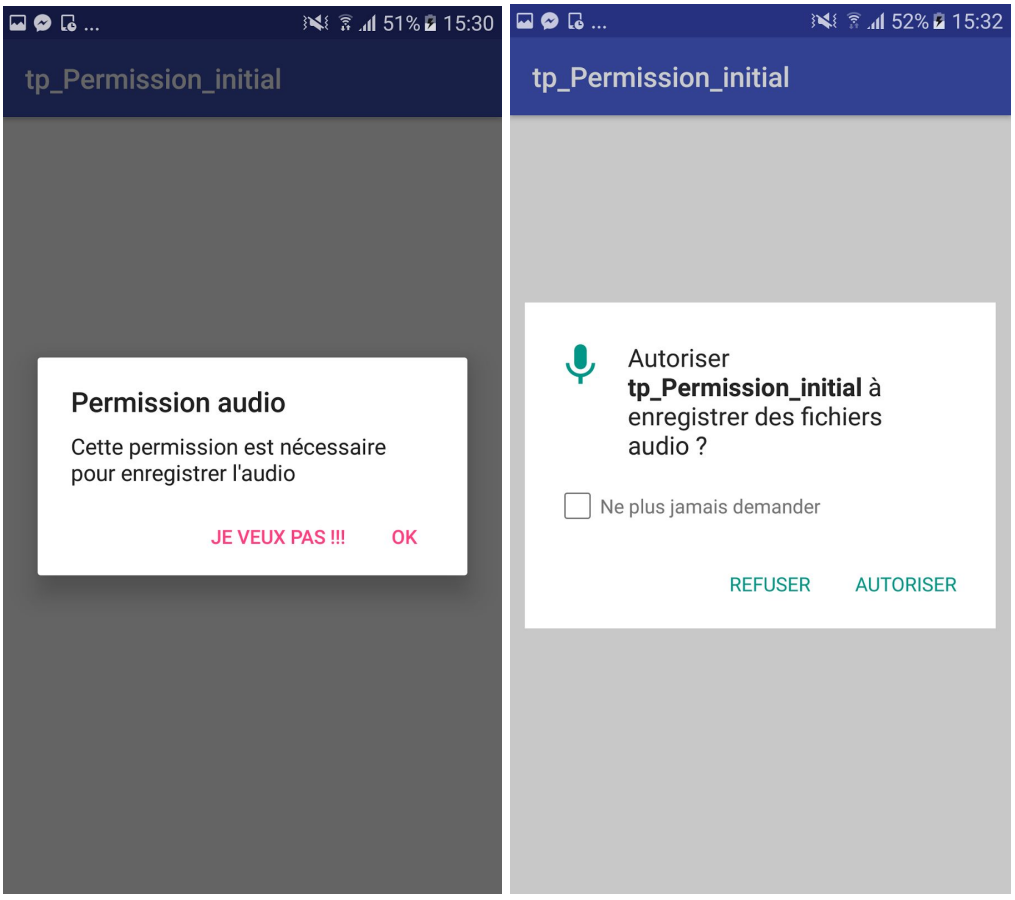
Parameters	
`requestCode`	`int`: The request code passed in `requestPermissions(android.app.Activity, String[], int)`
`permissions`	`String`: The requested permissions. Never null.
`grantResults`	`int`: The grant results for the corresponding permissions which is either `PERMISSION_GRANTED` or `PERMISSION_DENIED`. Never null.

1-Gérer une autorisation **grantResults.**

Nous allons vérifier la valeur de la première case du tableau **grantResults,** Si la valeur est égale à PackageManager.PERMISSION_GRANTED, c'est que la permission a été acceptée par l'utilisateur.

2- Gérer un refus

Dans le cas où l'utilisateur refuse la permission; on peut lui expliquer en quoi la permission est nécessaire. S'il accepte, on peut alors lui demander une seconde fois la permission (dans le cas où il n'a pas coché la case "Ne plus jamais demander" lors de la première demande de permission.



Nous allons devoir gérer les deux cas suivants:

- L'utilisateur a cliqué sur "**refuser":** nous allons appeler la méthode AlerteAutio().
- L'utilisateur a cliqué sur "**refuser**" et il a coché "**Ne plus jamais demander**"**.** Dans ce cas, on arrête de demander la permission et on désactive l'option.

Pour différencier les résultats, on appelle de nouveau la méthode **shouldShowRequestPermissionRationale()** et on vérifie son résultat.

- Si elle renvoie false, cela signifie que l'utilisateur a **refusé et qu'il a coché "Ne plus jamais demander"**. On ne peut plus lui demander la permission.
- Si elle renvoie true, c'est que l'utilisateur a **refusé** la permission **sans cocher "Ne plus jamais demander"**. On peut alors lui expliquer une deuxième fois l'intérêt d'accepter cette permission.

Exemple d'implémentation:

```
@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    switch (requestCode) {
        case MY_PERMISSION_REQUEST_RECORD_AUDIO: {
            if (grantResults[0] == PackageManager.PERMISSION_DENIED) {
                boolean showRationale = ActivityCompat.shouldShowRequestPermissionRationale(this, permissions[0]);
                if (showRationale) {
                    AlertAudio();
                } else if (!showRationale) {
                    Toast.makeText(getApplicationContext(), "pour donner la permission allez sur reglage -> application -> android_permission_initial -> autorisations -> autoriser du microphone",
                            Toast.LENGTH_LONG).show();

                    //l'utilisateur a choisi ne plus jamais redemander
                    //vous pouvez soit activer un peu de recul,
                    //désactiver les fonctionnalités de votre application
                    //ou ouvrez une autre boîte de dialogue expliquant encore une fois l'autorisation et la direction de l'application "je vous conseil pas de le faire"
                }
            }

            else{
                //permission accordée
            }

        }


    }
}
```

Informations complémentaires

- Demander une permission est complexe car nécessite de vérifier plusieurs cas.
- A tout moment un utilisateur peut activer ou désactiver une permission.
- Il convient de respecter les choix de l'utilisateur et de vérifier tous les cas d'usage avant de demander une permission.