Les permissions

Sommaire

1.1 - PRÉSENTATION

Une application Android doit être autorisée par programme ou par l'utilisateur à utiliser certaines données (BD contacts) ou certains services (Internet) ou certains matériels (Caméra) pour fonctionner.

Avec Marshmallow (Android 6, API 23) et Nougat (Android 7, API 24) la politique de sécurité des applications est renforcée. L'utilisateur est amené à intervenir de plus en plus.

Le système des permissions avec Android 6.x (API level 23) Marshmallow révolutionne le codage. Alors que précédemment il suffisait de préciser dans le Manifest la liste des permissions nécessaires à l'application, il faut dorénavant pour de nombreuses permissions l'accord de l'utilisateur. C'est le système du Requesting Runtime Permissions.

https://developer.android.com/training/permissions/requesting.html

https://inthecheesefactory.com/blog/things-you-need-to-know-about-android-m-permission-developer-edition/en

https://blog.xamarin.com/requesting-runtime-permissions-in-android-marshmallow/

Sauf ces permissions-ci qui ne nécessitent pas l'autorisation de l'utilisateur mais qui doivent être présentes dans le Manifest si nécessaire.

android.permission.ACCESS_LOCATION_EXTRA_COMMANDS
android.permission.**ACCESS_NETWORK_STATE**
android.permission.ACCESS_NOTIFICATION_POLICY
android.permission.ACCESS_WIFI_STATE
android.permission.ACCESS_WIMAX_STATE
android.permission.BLUETOOTH
android.permission.BLUETOOTH_ADMIN
android.permission.BROADCAST_STICKY
android.permission.**CHANGE_NETWORK_STATE**
android.permission.CHANGE_WIFI_MULTICAST_STATE
android.permission.CHANGE_WIFI_STATE
android.permission.CHANGE_WIMAX_STATE
android.permission.DISABLE_KEYGUARD
android.permission.EXPAND_STATUS_BAR
android.permission.FLASHLIGHT
android.permission.GET_ACCOUNTS
android.permission.GET_PACKAGE_SIZE
android.permission.**INTERNET**
android.permission.KILL_BACKGROUND_PROCESSES
android.permission.MODIFY_AUDIO_SETTINGS
android.permission.NFC
android.permission.READ_SYNC_SETTINGS
android.permission.READ_SYNC_STATS
android.permission.RECEIVE_BOOT_COMPLETED
android.permission.REORDER_TASKS
android.permission.REQUEST_INSTALL_PACKAGES
android.permission.SET_TIME_ZONE
android.permission.SET_WALLPAPER
android.permission.SET_WALLPAPER_HINTS
android.permission.SUBSCRIBED_FEEDS_READ
android.permission.TRANSMIT_IR
android.permission.USE_FINGERPRINT
android.permission.VIBRATE
android.permission.WAKE_LOCK
android.permission.WRITE_SYNC_SETTINGS
com.android.alarm.permission.SET_ALARM
com.android.launcher.permission.**INSTALL_SHORTCUT**
com.android.launcher.permission.UNINSTALL_SHORTCUT

1.2 - Permissions avec l'API 19 : KitKat

Écran des permissions pour une application donnée dans les Settings.

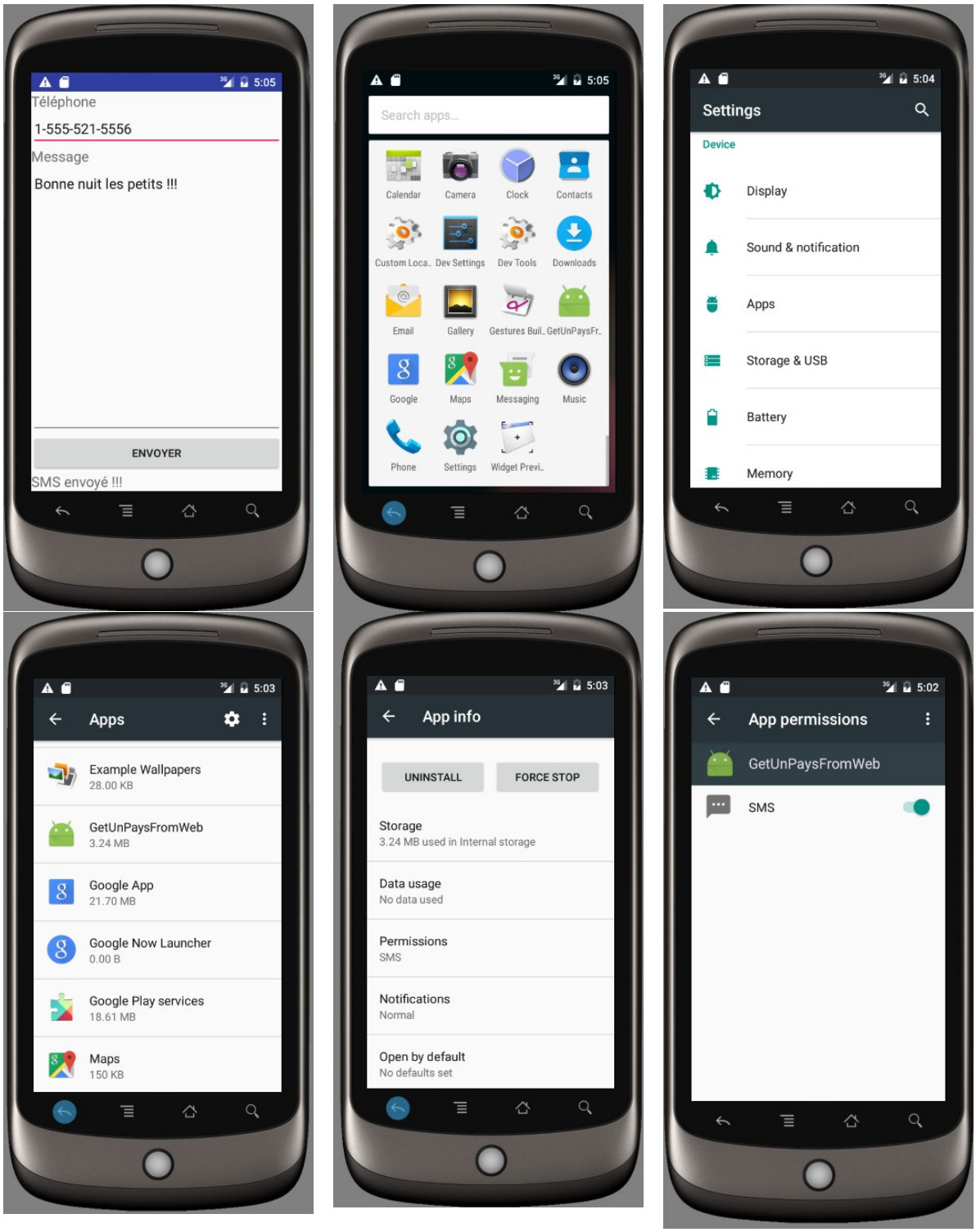
L'application peut :

- ✔ prendre des photos et enregistrer des vidéos,
- ✔ enregistrer des fichiers audio,
- ✔ visualiser et modifier le contenu de la carte SD,
- ✔ accéder au réseau.

L'utilisateur n'a pas à intervenir, les permissions sont paramétrées dans la Manifest.

1.3 - Permissions avec l'API 21 : Lollipop

L'envoi de SMS est autorisé ! Parce que le code l'a autorisé !
« Diagramme de navigation » pour accéder aux permissions d'une application.



1.4 - Permissions avec l'API 23 : Marshmallow

Avec l'API 23 tout change !
C'est à l'utilisateur de décider de donner ou pas l'autorisation d'accéder à tel service, à tel matériel, ...

Quelques références (officielles et d'autres utiles trouvées sur le web).

https://developer.android.com/guide/topics/security/permissions.html

https://developer.android.com/training/permissions/requesting.html

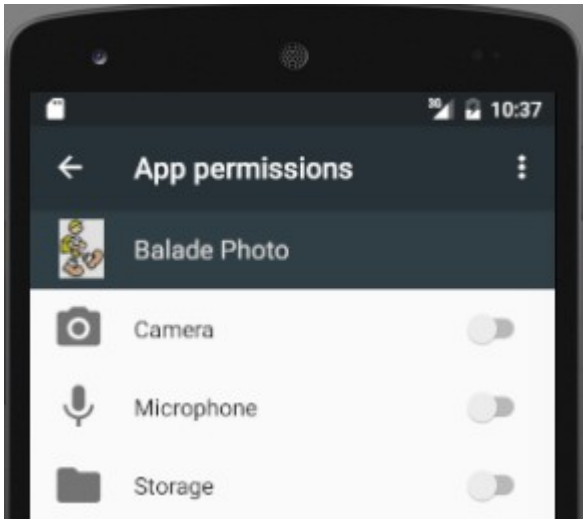
http://www.tutos-android.com/android-6-permissions

http://stackoverflow.com/questions/33666071/android-marshmallow-request-permission

1.5 - Un exemple avec la CAMERA

1.5.1 - Le paramétrage de l'application

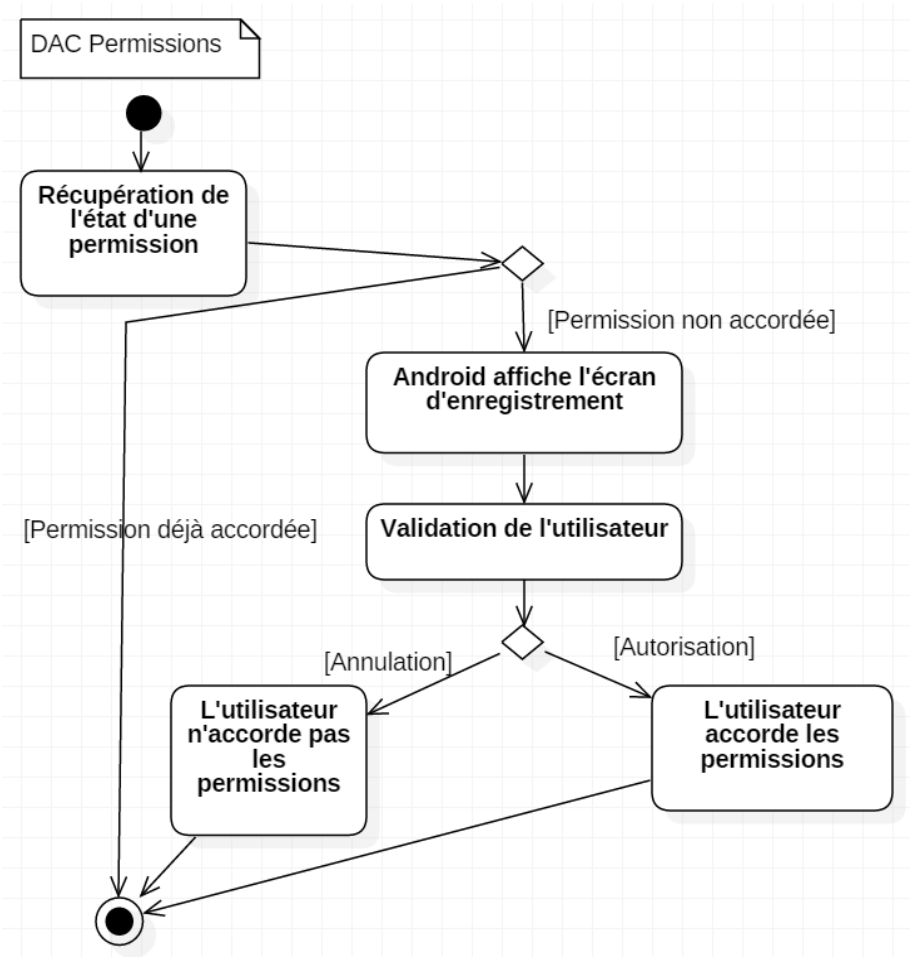
L'utilisateur n'a donné aucune permission pour la Camera ...



Dans le manifest :

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

1.5.2 - Le DAC


DAC Permissions
Récupération de l'état d'une permission
[Permission non accordée]
Android affiche l'écran d'enregistrement
[Permission déjà accordée]
Validation de l'utilisateur
[Annulation]
[Autorisation]
L'utilisateur n'accorde pas les permissions
L'utilisateur accorde les permissions

1.5.3 - L'exécution du code

L'utilisateur n'a pas donné l'autorisation d'utiliser la CAMERA. L'application demande à l'utilisateur s'il donne la permission de l'utiliser.

Est-ce vous donnez la permission à l'application « BaladePhoto » de prendre des photos et d'enregistrer des vidéos ?

Si l'utilisateur accorde la permission d'utiliser l'appareil photo, l'application affiche un Toast, et l'écran des permissions est le suivant :

si l'utilisateur ne donne pas la permission l'application affiche un Toast, et l'écran des permissions reste le même que précédemment.

1.5.4 - Syntaxes

Déclaration d'une constante à une valeur arbitraire.

```
private final int PERMISSION_REQUEST_USE_QQ_CHOSE = 1;
```

Récupère l'état d'une permission (accordée ou refusée).

https://developer.android.com/reference/android/support/v4/content/ContextCompat#checkSelfPermission(android.content.Context,%20java.lang.String)

int checkSelfPermission (Context context, String permission)
Determine whether you have been granted a particular permission.

```
int permissionCheckAction = ContextCompat.checkSelfPermission(contexte,
Manifest.permission.ACTION);
```

ACTION peut avoir la valeur suivante :

READ_EXTERNAL_STORAGE,
WRITE_EXTERNAL_STORAGE,
CAMERA,
RECORD_AUDIO,
etc.

Renvoie un int :

int PERMISSION_GRANTED
Constant Value: 0 (0x00000000)

int PERMISSION_DENIED
Constant Value: -1 (0xffffffff)

Faire quelque chose en fonction de l'état d'une permission.

```
if (permissionCheckAction == PackageManager.PERMISSION_GRANTED) {
    // Faire quelque chose, souvent ne rien faire
} else if (permissionCheckCamera == PackageManager.PERMISSION_DENIED) {
    // Faire autre chose
}
```

Ou en une seule fois demander à l'utilisateur d'accorder ou pas une permission en fonction de l'état d'une permission.

```
if (ContextCompat.checkSelfPermission(this, Manifest.permission.ACTION) !=
PackageManager.PERMISSION_GRANTED) {
    ActivityCompat.requestPermissions(contexte, new String[]{Manifest.permission.ACTION},
REQUEST_CODE);
}
```

La méthode ActivityCompat.requestPermissions() sollicite la méthode événementielle onRequestPermissionsResult().

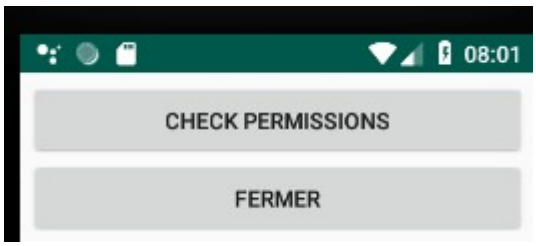
La méthode qui permet de modifier les permissions.

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

Cette méthode est appelée par ActivityCompat.requestPermissions() et à la suite de la validation (REFUSER ou AUTORISER) de l'utilisateur dans la boîte de dialogue.

Le tableau permissions[] contient la liste des droits
Le tableau grandResults contient le résultat des autorisations ou refus donnés par l'utilisateur

1.5.5 - Les codes



Créez une activité nommée « CameraPermissionsCheck ».

1.5.5.1 - permissions_check.xml

```
<?xml version='1.0' encoding='utf-8'?>
<ScrollView xmlns:android='http://schemas.android.com/apk/res/android'
    android:layout_width='match_parent'
    android:layout_height='match_parent'>

    <LinearLayout
        android:layout_width='match_parent'
        android:layout_height='match_parent'
        android:orientation='vertical'
        android:padding='5dp'>

        <Button
            android:id='@+id/buttonCheckPermissions'
            android:layout_width='match_parent'
            android:layout_height='wrap_content'
            android:text='Check Permissions' />

        <Button
            android:id='@+id/buttonFermer'
            android:layout_width='match_parent'
            android:layout_height='wrap_content'
            android:text='Fermer' />

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="Message"
            android:id="@+id/textViewMessage" />

    </LinearLayout>
</ScrollView>
```

1.5.5.2 - PermissionsCheck.java

Ce code teste l'état des permissions et demande à l'utilisateur de valider ou pas la permission d'utiliser la caméra à l'ouverture de l'activité ou sur le clic du bouton.
Cf la version simplifiée dans les Annexes.

```
import android.Manifest;
import android.app.Activity;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class CameraPermissionsCheck extends Activity implements
View.OnClickListener {

    private final int PERMISSION_REQUEST_USE_CAMERA = 1;

    private Button buttonCheckPermissions;
    private Button buttonFermer;
    private TextView textViewMessage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.permissions_check);

        buttonCheckPermissions = findViewById(R.id.buttonCheckPermissions
);
        buttonCheckPermissions.setOnClickListener(this);
        buttonFermer = findViewById(R.id.buttonFermer);
        buttonFermer.setOnClickListener(this);

        textViewMessage = findViewById(R.id.textViewMessage);

        permissions();
    } /// onCreate

    /*
    PERMISSIONS
     */
    private void permissions() {

        /*
        Permet de connaître l'état d'une permission
         */
        /*
        CAMERA
         */
        int permissionCheckCamera =
ContextCompat.checkSelfPermission(this,
                Manifest.permission.CAMERA);
```

```
        if (permissionCheckCamera == PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "Camera Granted",
Toast.LENGTH_LONG).show();
        } else if (permissionCheckCamera ==
PackageManager.PERMISSION_DENIED) {
            Toast.makeText(this, "Camera Denied",
Toast.LENGTH_LONG).show();
        }

        /*
        Permet de modifier l'état d'une permission
         */
        if (ContextCompat.checkSelfPermission(this,
Manifest.permission.CAMERA) == PackageManager.PERMISSION_DENIED) {
            ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.CAMERA}, PERMISSION_REQUEST_USE_CAMERA);
        }

    } /// permissions


    @Override
    public void onRequestPermissionsResult(int requestCode, String
permissions[], int[] grantResults) {
        /*
        Cette méthode est appelée par ActivityCompat.requestPermissions()
        Et à la suite de la validation de l'utilisateur de la boîte de
dialogue
        */
        switch (requestCode) {
            case PERMISSION_REQUEST_USE_CAMERA: {
                // If request is cancelled, the result arrays are empty.
                // Le tableau permissions[] contient la liste des droits
                // Le tableau grandResults contient le résultat des
autorisations ou refus donnés par l'utilisateur
                if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                    Toast.makeText(this, "Permission d'utiliser la Camera
accordée", Toast.LENGTH_LONG).show();
                } else {
                    Toast.makeText(this, "Permission d'utiliser la Camera
refusée", Toast.LENGTH_LONG).show();
                }
                return;
            }

        } /// switch
    } /// onRequestPermissionsResult

    @Override
    public void onClick(View v) {
        if (v == buttonCheckPermissions) {
            permissions();
        }
        if (v == buttonFermer) {
            finish();
        }

    } /// onClick

} /// class
```

1.5.6 - Exercice

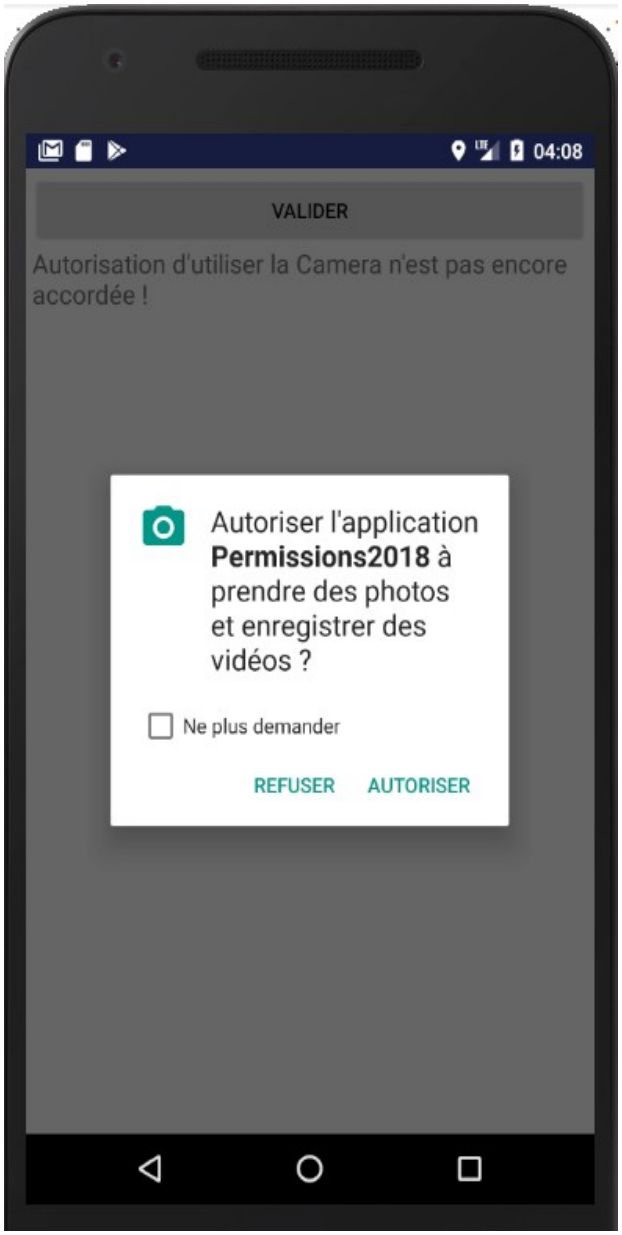
Attribuer des permissions à partir d'un seul écran composé de cases à cocher et d'un bouton de validation.

Note : il est fortement conseillé de relire ce qui précède, de le mettre en place, et de le compléter avant de passer à l'exercice.

1.6 - ANNEXES

1.6.1 - Exemple simplifié

1.6.1.1 - L'écran



1.6.1.2 - Le layout

```
<?xml version='1.0' encoding='utf-8'?>
<ScrollView xmlns:android='http://schemas.android.com/apk/res/android'
    android:layout_width='match_parent'
    android:layout_height='match_parent'>
    <LinearLayout
        android:layout_width='match_parent'
        android:layout_height='match_parent'
        android:orientation='vertical'
        android:padding='5dp'>
        <Button
            android:id='@+id/buttonValider'
            android:layout_width='match_parent'
            android:layout_height='wrap_content'
            android:text='Valider' />
        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="Message"
            android:id="@+id/textViewMessage" />
    </LinearLayout>
</ScrollView>
```

1.6.1.3 - L'activité

```
import android.Manifest;
import android.app.Activity;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class UnePermission extends Activity implements View.OnClickListener
{
    private final int PERMISSION_REQUEST_USE_CAMERA = 8;
    private Button buttonValider;
    private TextView textViewMessage;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.une_permission);
        buttonValider = findViewById(R.id.buttonValider);
        buttonValider.setOnClickListener(this);
        textViewMessage = findViewById(R.id.textViewMessage);
        permissions();
    } /// onCreate

    /*
    PERMISSIONS
     */
    private void permissions() {
        /*
        Permet de connaître l'état d'une permission
        ICI la CAMERA
         */
        int permissionCheckCamera = ContextCompat.checkSelfPermission(this,
                Manifest.permission.CAMERA);
        /*
        SI L'USAGE
         */
        if (permissionCheckCamera == PackageManager.PERMISSION_DENIED) {
            Toast.makeText(this, "Camera Denied",
Toast.LENGTH_LONG).show();
            textViewMessage.setText("Autorisation d'utiliser la Camera
n'est pas encore accordée !");
            ActivityCompat.requestPermissions(this, new String[]
{Manifest.permission.CAMERA}, PERMISSION_REQUEST_USE_CAMERA);
        } else {
            Toast.makeText(this, "Autorisation d'utiliser la Camera déjà
accordée !", Toast.LENGTH_LONG).show();
            textViewMessage.setText("Autorisation d'utiliser la Camera déjà
accordée !");
        }
    } /// permissions

    @Override
```

```
    public void onRequestPermissionsResult(int requestCode, String
permissions[], int[] grantResults) {
        /*
        Cette méthode est appelée par ActivityCompat.requestPermissions()
        */
        if (requestCode == PERMISSION_REQUEST_USE_CAMERA &&
grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
            Toast.makeText(this, "Permission d'utiliser la Camera
accordée", Toast.LENGTH_LONG).show();
            textViewMessage.setText("Permission d'utiliser la Camera
accordée");
        } else {
            Toast.makeText(this, "Permission d'utiliser la Camera refusée",
Toast.LENGTH_LONG).show();
            textViewMessage.setText("Permission d'utiliser la Camera
refusée");
        }
    } /// onRequestPermissionsResult

    @Override
    public void onClick(View v) {
        permissions();
    } /// onClick
} /// class
```

1.6.2 - Internet

```
        /*
        INTERNET
         */
        int permissionCheckInternet =
ContextCompat.checkSelfPermission(this,
                Manifest.permission.INTERNET);

        if (permissionCheckInternet == PackageManager.PERMISSION_GRANTED)
{
            Toast.makeText(this, "Internet Granted",
Toast.LENGTH_LONG).show();
        } else if (permissionCheckInternet ==
PackageManager.PERMISSION_DENIED) {
            Toast.makeText(this, "Internet Denied",
Toast.LENGTH_LONG).show();
        }
```