动态申请权限库：easypermissions使用与源码解析

当接触了Android 6.0 7.0 之后，很多人就会发现，单纯的在AndroidManifest.xml中声明权限已经是不好使的了，因为Google为了保护用户的权限，需要应用动态的权限申请。

下面我们就介绍一个Google官方放出来的动态申请权限的库，GitHub地址，这个库可以简单的帮助我们完成动态申请权限的整个流程，并且可以将结果回调回来，因此我们可以在回调结果中加上我们的处理逻辑。

Installation 安装

EasyPermissions is installed by adding the following dependency to your build.gradle file:

```
dependencies {
    compile 'pub.devrel:easypermissions:0.4.0'
}
```

Usage 用法

Basic 基础

To begin using EasyPermissions, have your Activity (or Fragment) override the onRequestPermissionsResult method:

开始用这个库的时候，需要让Activity或者Fragment重写onRequestPermissionsResult这个方法：

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

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

        // Forward results to EasyPermissions
        // 远期回调回来的结果
        EasyPermissions.onRequestPermissionsResult(requestCode, permissions, grantResults, this);
    }
}
```

Request Permissions

The example below shows how to request permissions for a method that requires both CAMERA and ACCESS_FINE_LOCATION permissions. There are a few things to note:

- Using EasyPermissions#hasPermissions(...) to check if the app already has the required permissions. This method can take any number of permissions as its final argument.
- Requesting permissions with EasyPermissions#requestPermissions. This method will request the system permissions and show the rationale string provided if necessary. The request code provided should be unique to this request, and the method can take any number of permissions as its final argument.
- Use of the AfterPermissionGranted annotation. This is optional, but provided for convenience. If all of the permissions in a given request are granted, any methods annotated with the proper request code will be executed. This is to simplify the common flow of needing to run the requesting method after all of its permissions have been granted. This can also be achieved by adding logic on the onPermissionsGranted callback.

下面这个例子展示了如何申请权限用一个方法同时申请相机和申请定位的权限，这里有一些内容需要注意：

- 用EasyPermissions的hasPermissions方法检测一下，是否app已经具有要申请的权限。这个方法可以将许多方法作为最后的论证。

- 申请权限通过EasyPermissions的requestPermissions的方法。这个方法将会申请系统的权限，并且在如果有必要的条件下会展示出理由。这个申请权限的代码应该提供独一无二的申请，并且这些方法可以提供很多权限作为最终的定论。
- 应用AfterPermissionGranted这个标签，这是可以选择的，但是它提供了方便。如果所有权限在申请之后都被授予了，一些携带着这个标签的方法将会被执行。这是为了简化在所有权限被授予后需要运行请求方法的公共流程。这些同样也可以被实现通过添加合理的逻辑在onPermissionsGranted这个方法回调的时候。

```
@AfterPermissionGranted(RC_CAMERA_AND_LOCATION)
private void methodRequiresTwoPermission() {
    String[] perms = {Manifest.permission.CAMERA, Manifest.permission.ACCESS_FINE_LOCATION};
    if (EasyPermissions.hasPermissions(this, perms)) {
        // Already have permission, do the thing
        // ...
    } else {
        // Do not have permissions, request them now
        EasyPermissions.requestPermissions(this, getString(R.string.camera_and_location_rationale),
                RC_CAMERA_AND_LOCATION, perms);
    }
}
```

Optionally, for a finer control, you can have your Activity / Fragment implement the PermissionCallbacks interface.

随意的，通过一个好的控制，你可以让你的Activity/Fragment 实现PermissionCallbacks这个接口。

```
public class MainActivity extends AppCompatActivity implements EasyPermissions.PermissionCallbacks {

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

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

        // Forward results to EasyPermissions
        EasyPermissions.onRequestPermissionsResult(requestCode, permissions, grantResults, this);
    }

    @Override
    public void onPermissionsGranted(int requestCode, List<String> list) {
        // Some permissions have been granted
        // ...
    }

    @Override
    public void onPermissionsDenied(int requestCode, List<String> list) {
        // Some permissions have been denied
        // ...
    }
}
```

Required Permissions 必须的权限

In some cases your app will not function properly without certain permissions. If the user denies these permissions with the "Never Ask Again" option, you will be unable to request these permissions from the user and they must be changed in app settings. You can use the method EasyPermissions.somePermissionPermanentlyDenied(...) to display a dialog to the user in this situation and direct them to the system setting screen for your app:

无论如何你的app在没有某些权限的时候是不能正常运行的。如果用户否认这些权限，并且选择了再也不询问的选项，你将没有能力去申请那些权限通过用户，他们必须在app的设置中修改这些。你可以用这个方法EasyPermissions.somePermissionPermanentlyDenied(...)，展示一个对话框给用户在这个位置直接的告诉他们去系统的设置中去改变为了这个app：

```java
@Override
public void onPermissionsDenied(int requestCode, List<String> perms) {
    Log.d(TAG, "onPermissionsDenied:" + requestCode + ":" + perms.size());

    // (Optional) Check whether the user denied any permissions and checked "NEVER ASK AGAIN."
    // This will display a dialog directing them to enable the permission in app settings.
    if (EasyPermissions.somePermissionPermanentlyDenied(this, perms)) {
        new AppSettingsDialog.Builder(this).build().show();
    }
}

@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == AppSettingsDialog.DEFAULT_SETTINGS_REQ_CODE) {
        // Do something after user returned from app settings screen, like showing a Toast.
        Toast.makeText(this, R.string.returned_from_app_settings_to_activity, Toast.LENGTH_SHORT)
                .show();
    }
}
```