# 10. Permissions, Google Maps, Location

## Permissions

Операционная система Android устроена таким образом, что для выполнения некоторых операций или доступа к определенным ресурсам, приложение должно иметь разрешение на это.

Разрешения могут быть двух типов: **normal** и **dangerous**. Отличие между ними в том, что dangerous разрешения опасны, т.к. могут быть использованы для получения ваших личных данных или информации о вас, или еще каким-то способом могут навредить вам. Примеры dangerous разрешений - это доступ к контактам или смс.

Полный список существующих разрешений можно посмотреть [здесь](https://developer.android.com/reference/android/Manifest.permission.html). Характеристика Protection level подскажет насколько опасно это разрешение. А [здесь](https://developer.android.com/guide/topics/permissions/normal-permissions.html) можно сразу просмотреть весь список **normal** разрешений.

Если приложению необходимо получить какое-либо разрешение, то оно должно быть указано в AndroidManifest.xml, в корневом теге <manifest>. Тег разрешения - <uses-permission>.

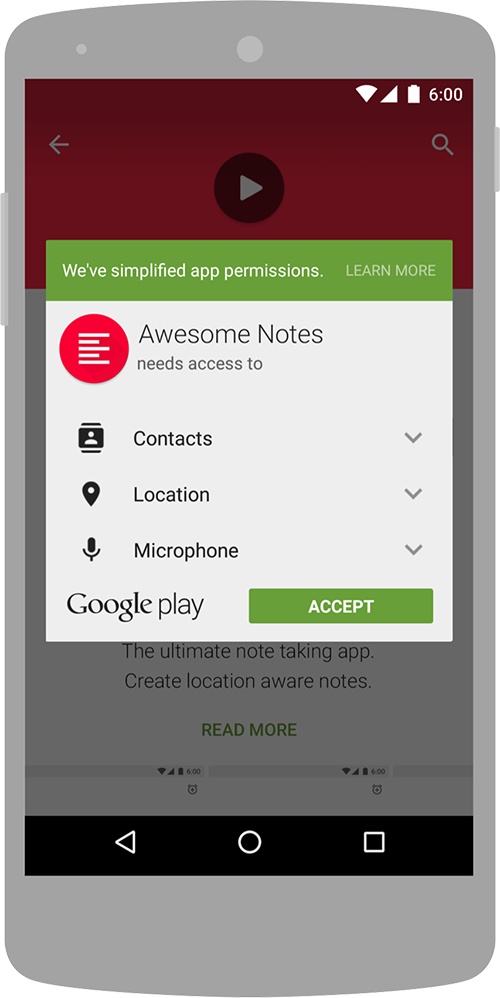
Вот пример манифеста с разрешениями:

|  |  |
| --- | --- |
|  | <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.test.someapplication">    <uses-permission android:name="android.permission.INTERNET" />  <uses-permission android:name="android.permission.READ\_CONTACTS" />  <uses-permission android:name="android.permission.BLUETOOTH" />  <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  <uses-permission android:name="android.permission.CAMERA" />  <uses-permission android:name="android.permission.SEND\_SMS" />    <application  ...  ></application>    </manifest> |

Здесь мы указываем, что приложению понадобятся разрешения на работу с интернет, контактами, bluetooth, локацией, камерой и смс. Пользователю необходимо будет подтвердить, что он предоставляет приложению эти разрешения.

## До Android 6

До выхода Android 6 все было просто и легко. Когда пользователь устанавливал приложение с манифестом, который мы рассмотрели чуть выше, то он видел такой экран:



Система показывает разрешения, которые были прописаны в манифесте. Сначала те, которые могут быть опасными с точки зрения приватности (отправка смс, доступ к камере/местоположению/контактам), а затем - обычные (интернет, bluetooth).

Таким образом пользователь видит, на что претендует приложение, и может примерно понять все ли в порядке. Если, например, приложение калькулятор при установке просит у вас доступ к контактам и смс, то скорее всего, что-то не так с этим приложением и оно может быть опасным для ваших данных.

Нажав кнопку Install, пользователь автоматически подтверждает свое согласие, что приложению будут предоставлены эти запрашиваемые разрешения. И далее, когда приложение, например, пытается в коде получить список контактов, то оно без проблем их получает.

Если же в манифесте не указать разрешение READ\_CONTACTS, то его не будет и в списке тех разрешений, которые подтверждает пользователь. Соответственно, система не предоставит этому приложению доступ к контактам. И при попытке получить список контактов, будет ошибка:

*java.lang.SecurityException: Permission Denial: opening provider com.android.providers.contacts.ContactsProvider2*

## Android 6

С выходом Android 6 механизм подтверждения поменялся. Теперь при установке приложения пользователь больше не видит списка запрашиваемых разрешений. Приложение автоматически получает все требуемые normal разрешения, а dangerous разрешения необходимо будет программно запрашивать в процессе работы приложения.

Т.е. теперь недостаточно просто указать в манифесте, что вам нужен, например, доступ к контактам. Когда вы в коде попытаетесь запросить список контактов, то получите ошибку SecurityException: Permission Denial. Потому что вы явно не запрашивали это разрешение, и пользователь его не подтверждал.

Перед выполнением операции, требующей разрешения, необходимо спросить у системы, есть ли у приложения разрешение на это. Т.е. подтверждал ли пользователь, что он дает приложению это разрешение. Если разрешение уже есть, то выполняем операцию. Если нет, то запрашиваем это разрешение у пользователя.

Давайте посмотрим, как это выглядит на практике.

Проверка текущего статуса разрешения выполняется методом checkSelfPermission

|  |  |
| --- | --- |
| 1 | int permissionStatus = ContextCompat.checkSelfPermission(this, Manifest.permission.READ\_CONTACTS); |

На вход метод требует Context и название разрешения. Он вернет константу PackageManager.PERMISSION\_GRANTED (если разрешение есть) или PackageManager.PERMISSION\_DENIED (если разрешения нет).

Если разрешение есть, значит мы ранее его уже запрашивали, и пользователь подтвердил его. Можем получать список контактов, система даст нам доступ.

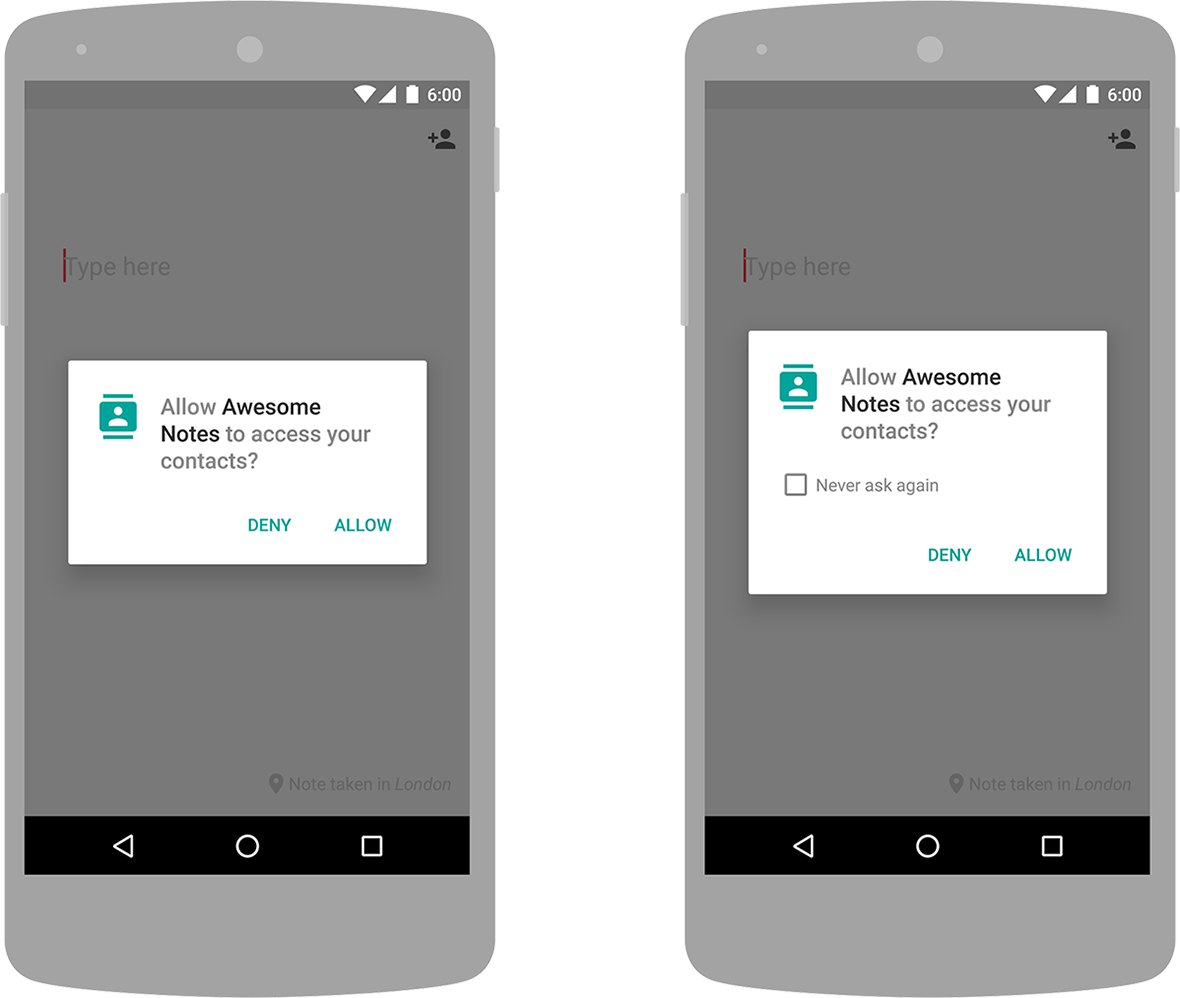
Если разрешения нет, то нам надо его запросить. Это выполняется методом [requestPermissions](https://developer.android.com/reference/android/support/v4/content/ContextCompat.html#checkSelfPermission(android.content.Context,%20java.lang.String)). Схема его работы похожа на метод startActivityForResult. Мы вызываем метод, передаем ему данные и request code, а ответ потом получаем в определенном onResult методе.

Добавим запрос разрешения к уже имеющейся проверке.

|  |  |
| --- | --- |
|  | int permissionStatus = ContextCompat.checkSelfPermission(this, Manifest.permission.READ\_CONTACTS);    if (permissionStatus == PackageManager.PERMISSION\_GRANTED) {  readContacts();  } else {  ActivityCompat.requestPermissions(this, new String[] {Manifest.permission.READ\_CONTACTS},  REQUEST\_CODE\_PERMISSION\_READ\_CONTACTS);  } |

Проверяем разрешение READ\_CONTACTS. Если оно есть, то читаем контакты. Иначе запрашиваем разрешение READ\_CONTACTS методом [requestPermissions](https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#requestPermissions(android.app.Activity,%20java.lang.String%5B%5D,%20int)). На вход метод требует Activity, список требуемых разрешений, и request code. Обратите внимание, что для разрешений используется массив. Т.е. вы можете запросить сразу несколько разрешений.

После вызова метода requestPermissions система покажет следующий диалог



Здесь будет отображено разрешение, которое мы запросили методом requestPermissions. Пользователь может либо подтвердить его (ALLOW), либо отказать (DENY). Если будет запрошено сразу несколько разрешений, то на каждое из них будет показан отдельный диалог. И пользователь может какие-то разрешения подтвердить, а какие-то нет.

Решение пользователя мы получим в методе [onRequestPermissionsResult](https://developer.android.com/reference/android/support/v4/app/ActivityCompat.OnRequestPermissionsResultCallback.html#onRequestPermissionsResult(int,%20java.lang.String%5B%5D,%20int%5B%5D))

|  |  |
| --- | --- |
|  | @Override  public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  switch (requestCode) {  case REQUEST\_CODE\_PERMISSION\_READ\_CONTACTS:  if (grantResults.length > 0  && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {  // permission granted  readContacts();  } else {  // permission denied  }  return;  }  } |

Проверяем, что requestСode тот же, что мы указывали в requestPermissions. В массиве permissions придут название разрешений, которые мы запрашивали. В массиве grantResults придут ответы пользователя на запросы разрешений.

Мы проверяем, что массив ответов не пустой и берем первый результат из него (т.к. мы запрашивали всего одно разрешение). Если пользователь подтвердил разрешение, то выполняем операцию. Если же пользователь отказал, то дальнейшие действия зависят от логики вашего приложения.

В итоге схема получения разрешения состоит из трех действий:

- проверка текущего состояния разрешения

- запрос на получение разрешения, если оно еще не было получено

- обработка ответа на запрос

Далее поговорим про некоторые дополнительные возможности, нюансы и прочие мелочи.

## Манифест

При использовании новой схемы разрешений вам все равно необходимо указывать разрешение в манифесте. Если его там не указать и сделать запрос на это разрешение, то вам просто сразу придет отказ без всякого диалога.

## Всегда проверяйте разрешение

Каждый раз (а не только первый) перед выполнением операции, требующей определенного разрешения, необходимо проверять, что это разрешение есть. Потому что, даже если пользователь уже давал это разрешение, он всегда может зайти в настройки приложения и отменить его. И если вы после этого не выполните проверку, то получите ошибку при выполнении операции.

## Don’t ask again

Когда вы первый раз делаете запрос на какое-либо разрешение, пользователь может отказать. При последующих запросах этого же разрешения, в диалоге появится чекбокс Don’t ask again

Если пользователь включит этот чекбокс, то при последующих ваших запросах диалог не будет отображаться, а в onRequestPermissionsResult сразу будет приходить отказ.

## Объяснение для пользователя

Когда вы запрашиваете разрешение, пользователю должно быть очевидно, зачем приложению понадобилось это разрешение, и у него не должно возникать вопросов. Но случаи бывают разные, и вы можете решить, что вам надо явно объяснить пользователю, почему приложению понадобилось это разрешение.

Диалог, который показывается при запросе разрешения, - системный, вы не можете менять его содержимое и добавлять туда свой текст. Но вы можете сделать свой диалог или что-то подобное и показать его перед тем, как будете делать запрос разрешения.

Есть метод [shouldShowRequestPermissionRationale](https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#shouldShowRequestPermissionRationale(android.app.Activity,%20java.lang.String)), который может быть полезен в данной ситуации. Передаете ему название разрешения, а он вам в виде boolean ответит, надо ли показывать объяснение для пользователя.

Т.е. вы сначала проверяете наличие разрешения. Если его нет, то вызываете shouldShowRequestPermissionRationale, чтобы решить, надо ли показывать объяснение пользователю. Если не надо, то делаете запрос разрешения. А если надо, то показываете ваш диалог с объяснением, а после этого диалога делаете запрос разрешения.

Алгоритм работы метода shouldShowRequestPermissionRationale прост.

Если вы еще ни разу не запрашивали это разрешение, то он вернет false. Т.е. перед первым запросом разрешения ничего объяснять не надо.

Если вы ранее уже запрашивали это разрешение и пользователь отказал, то метод вернет true. Т.е. пользователь не понимает, почему он должен давать это разрешение, и надо ему это объяснить.

Если пользователь ставил галку Don’t ask again, то метод вернет false. Запрос полномочий все равно не будет выполнен. Объяснять что-то не имеет смысла.

Разумеется, вы можете показывать дополнительную информацию согласно вашим правилам и не использовать метод shouldShowRequestPermissionRationale.

## Группы

Dangerous разрешения собраны в группы. Список групп можно посмотреть [здесь](https://developer.android.com/reference/android/Manifest.permission_group.html). Если вы запросили одно разрешение из группы и пользователь предоставил вам его, то вы автоматически получаете все разрешения этой группы.

Например, разрешения READ\_CONTACTS и WRITE\_CONTACTS принадлежат группе CONTACTS. И если пользователь уже подтверждал разрешение на READ\_CONTACTS, то при проверке WRITE\_CONTACTS вы получите PERMISSION\_GRANTED.

## Google Maps

1. Рассказать коротко про сервисы гугл .Настроить серисы, добавить карты <https://developers.google.com/android/guides/setup>   
   implementation 'com.google.android.gms:play-services-maps:15.0.1'

Процесс создания ключа: <https://developers.google.com/maps/documentation/android-sdk/signup>

## Location

com.google.android.gms:play-services-location:15.0.1

Разобрать пример <https://github.com/googlesamples/android-play-location/tree/master/LocationUpdates>

Алгоритм:

1. Проверить permissions
2. Проверить, включена ли в настройках локация
3. Начать получать данные

#### Задание:

1. В уроке с мостами реализовать экран с картой.
2. Реализовывать предлагается с SupportMapFragment
3. Запросить разрешения на геолокацию. Если разрешение дано - включить встроенный слой с геопозицией пользователя   
   mMap.setMyLocationEnabled(true);
4. Продвинутый уровень:   
   Разобрать пример <https://github.com/googlesamples/android-play-location/tree/master/LocationUpdates>   
   Не использовать встроенные в карту средства отображения локации. Вместо этого использовать FusedLocationProviderClient. Необходимо получать реальную геопозицию пользователя и отображать ее на карте. При этом каждый раз, когда приходят новые данные по локации - двигать маркер. Не забывать приостанавливать получение локации в onPause и возобновлять в onResume.

#### Полезные ссылки:

1. <https://startandroid.ru/ru/blog/508-android-permissions.html> - про разрешения на русском
2. <https://developer.android.com/guide/topics/permissions/overview> - про разрешения официальная документация и далее по блоку
3. <https://developers.google.com/maps/documentation/android-sdk/start> - про гугл мэпс. На уроке рассказал лишь часть, остальное - самостоятельно
4. Все про локацию в официальном гайде : <https://developer.android.com/training/location/>
5. Пример с локацией: <https://github.com/googlesamples/android-play-location/tree/master/LocationUpdates>

# 0. Permissions, Google Maps, Location

## Permissions

Операционная система Android устроена таким образом, что для выполнения некоторых операций или доступа к определенным ресурсам, приложение должно иметь разрешение на это.

Разрешения могут быть двух типов: **normal** и **dangerous**. Отличие между ними в том, что dangerous разрешения опасны, т.к. могут быть использованы для получения ваших личных данных или информации о вас, или еще каким-то способом могут навредить вам. Примеры dangerous разрешений - это доступ к контактам или смс.

Полный список существующих разрешений можно посмотреть [здесь](https://developer.android.com/reference/android/Manifest.permission.html). Характеристика Protection level подскажет насколько опасно это разрешение. А [здесь](https://developer.android.com/guide/topics/permissions/normal-permissions.html) можно сразу просмотреть весь список **normal** разрешений.

Если приложению необходимо получить какое-либо разрешение, то оно должно быть указано в AndroidManifest.xml, в корневом теге <manifest>. Тег разрешения - <uses-permission>.

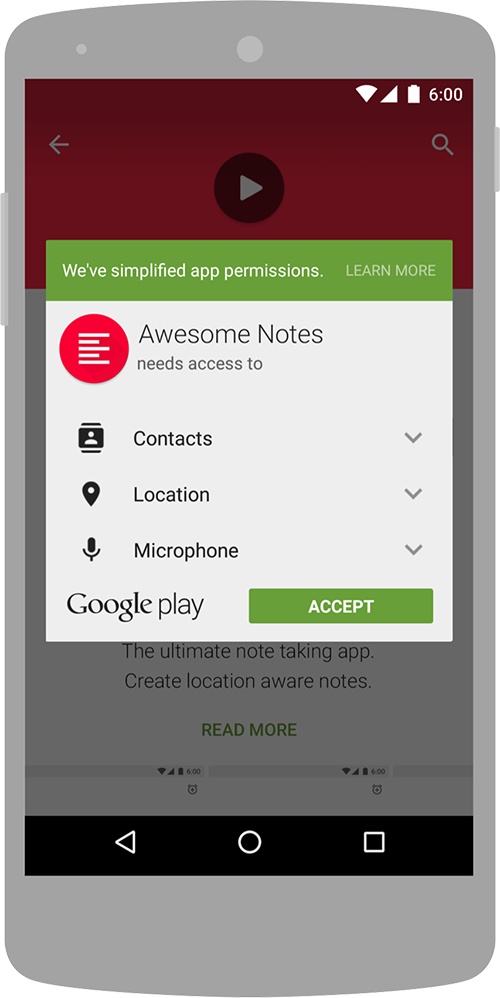
Вот пример манифеста с разрешениями:

|  |  |
| --- | --- |
|  | <?xml version="1.0" encoding="utf-8"?>  <manifest xmlns:android="http://schemas.android.com/apk/res/android"  package="com.test.someapplication">    <uses-permission android:name="android.permission.INTERNET" />  <uses-permission android:name="android.permission.READ\_CONTACTS" />  <uses-permission android:name="android.permission.BLUETOOTH" />  <uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />  <uses-permission android:name="android.permission.CAMERA" />  <uses-permission android:name="android.permission.SEND\_SMS" />    <application  ...  ></application>    </manifest> |

Здесь мы указываем, что приложению понадобятся разрешения на работу с интернет, контактами, bluetooth, локацией, камерой и смс. Пользователю необходимо будет подтвердить, что он предоставляет приложению эти разрешения.

## До Android 6

До выхода Android 6 все было просто и легко. Когда пользователь устанавливал приложение с манифестом, который мы рассмотрели чуть выше, то он видел такой экран:



Система показывает разрешения, которые были прописаны в манифесте. Сначала те, которые могут быть опасными с точки зрения приватности (отправка смс, доступ к камере/местоположению/контактам), а затем - обычные (интернет, bluetooth).

Таким образом пользователь видит, на что претендует приложение, и может примерно понять все ли в порядке. Если, например, приложение калькулятор при установке просит у вас доступ к контактам и смс, то скорее всего, что-то не так с этим приложением и оно может быть опасным для ваших данных.

Нажав кнопку Install, пользователь автоматически подтверждает свое согласие, что приложению будут предоставлены эти запрашиваемые разрешения. И далее, когда приложение, например, пытается в коде получить список контактов, то оно без проблем их получает.

Если же в манифесте не указать разрешение READ\_CONTACTS, то его не будет и в списке тех разрешений, которые подтверждает пользователь. Соответственно, система не предоставит этому приложению доступ к контактам. И при попытке получить список контактов, будет ошибка:

*java.lang.SecurityException: Permission Denial: opening provider com.android.providers.contacts.ContactsProvider2*

## Android 6

С выходом Android 6 механизм подтверждения поменялся. Теперь при установке приложения пользователь больше не видит списка запрашиваемых разрешений. Приложение автоматически получает все требуемые normal разрешения, а dangerous разрешения необходимо будет программно запрашивать в процессе работы приложения.

Т.е. теперь недостаточно просто указать в манифесте, что вам нужен, например, доступ к контактам. Когда вы в коде попытаетесь запросить список контактов, то получите ошибку SecurityException: Permission Denial. Потому что вы явно не запрашивали это разрешение, и пользователь его не подтверждал.

Перед выполнением операции, требующей разрешения, необходимо спросить у системы, есть ли у приложения разрешение на это. Т.е. подтверждал ли пользователь, что он дает приложению это разрешение. Если разрешение уже есть, то выполняем операцию. Если нет, то запрашиваем это разрешение у пользователя.

Давайте посмотрим, как это выглядит на практике.

Проверка текущего статуса разрешения выполняется методом checkSelfPermission

|  |  |
| --- | --- |
| 1 | int permissionStatus = ContextCompat.checkSelfPermission(this, Manifest.permission.READ\_CONTACTS); |

На вход метод требует Context и название разрешения. Он вернет константу PackageManager.PERMISSION\_GRANTED (если разрешение есть) или PackageManager.PERMISSION\_DENIED (если разрешения нет).

Если разрешение есть, значит мы ранее его уже запрашивали, и пользователь подтвердил его. Можем получать список контактов, система даст нам доступ.

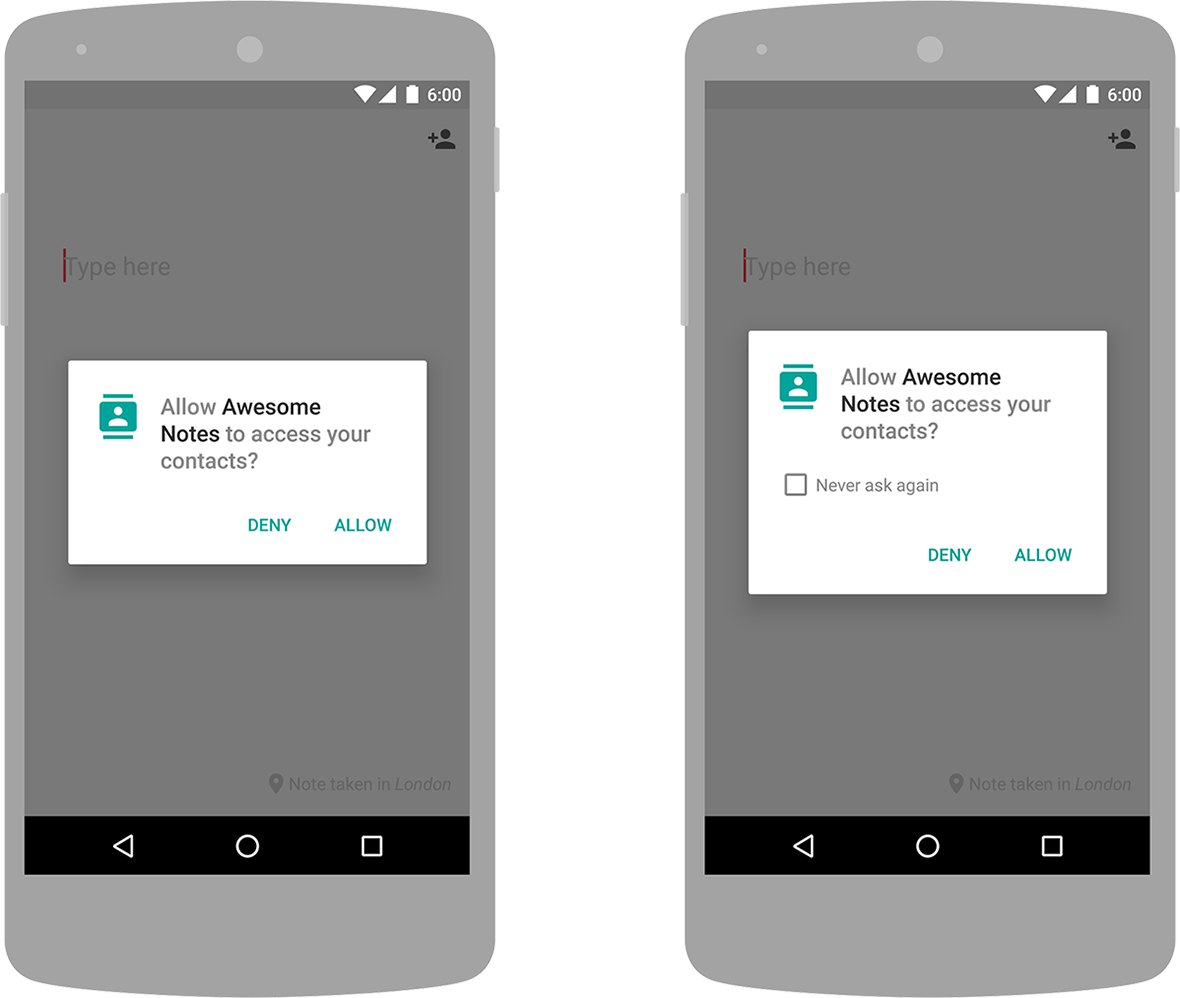
Если разрешения нет, то нам надо его запросить. Это выполняется методом [requestPermissions](https://developer.android.com/reference/android/support/v4/content/ContextCompat.html#checkSelfPermission(android.content.Context,%20java.lang.String)). Схема его работы похожа на метод startActivityForResult. Мы вызываем метод, передаем ему данные и request code, а ответ потом получаем в определенном onResult методе.

Добавим запрос разрешения к уже имеющейся проверке.

|  |  |
| --- | --- |
|  | int permissionStatus = ContextCompat.checkSelfPermission(this, Manifest.permission.READ\_CONTACTS);    if (permissionStatus == PackageManager.PERMISSION\_GRANTED) {  readContacts();  } else {  ActivityCompat.requestPermissions(this, new String[] {Manifest.permission.READ\_CONTACTS},  REQUEST\_CODE\_PERMISSION\_READ\_CONTACTS);  } |

Проверяем разрешение READ\_CONTACTS. Если оно есть, то читаем контакты. Иначе запрашиваем разрешение READ\_CONTACTS методом [requestPermissions](https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#requestPermissions(android.app.Activity,%20java.lang.String%5B%5D,%20int)). На вход метод требует Activity, список требуемых разрешений, и request code. Обратите внимание, что для разрешений используется массив. Т.е. вы можете запросить сразу несколько разрешений.

После вызова метода requestPermissions система покажет следующий диалог



Здесь будет отображено разрешение, которое мы запросили методом requestPermissions. Пользователь может либо подтвердить его (ALLOW), либо отказать (DENY). Если будет запрошено сразу несколько разрешений, то на каждое из них будет показан отдельный диалог. И пользователь может какие-то разрешения подтвердить, а какие-то нет.

Решение пользователя мы получим в методе [onRequestPermissionsResult](https://developer.android.com/reference/android/support/v4/app/ActivityCompat.OnRequestPermissionsResultCallback.html#onRequestPermissionsResult(int,%20java.lang.String%5B%5D,%20int%5B%5D))

|  |  |
| --- | --- |
|  | @Override  public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {  switch (requestCode) {  case REQUEST\_CODE\_PERMISSION\_READ\_CONTACTS:  if (grantResults.length > 0  && grantResults[0] == PackageManager.PERMISSION\_GRANTED) {  // permission granted  readContacts();  } else {  // permission denied  }  return;  }  } |

Проверяем, что requestСode тот же, что мы указывали в requestPermissions. В массиве permissions придут название разрешений, которые мы запрашивали. В массиве grantResults придут ответы пользователя на запросы разрешений.

Мы проверяем, что массив ответов не пустой и берем первый результат из него (т.к. мы запрашивали всего одно разрешение). Если пользователь подтвердил разрешение, то выполняем операцию. Если же пользователь отказал, то дальнейшие действия зависят от логики вашего приложения.

В итоге схема получения разрешения состоит из трех действий:

- проверка текущего состояния разрешения

- запрос на получение разрешения, если оно еще не было получено

- обработка ответа на запрос

Далее поговорим про некоторые дополнительные возможности, нюансы и прочие мелочи.

## Манифест

При использовании новой схемы разрешений вам все равно необходимо указывать разрешение в манифесте. Если его там не указать и сделать запрос на это разрешение, то вам просто сразу придет отказ без всякого диалога.

## Всегда проверяйте разрешение

Каждый раз (а не только первый) перед выполнением операции, требующей определенного разрешения, необходимо проверять, что это разрешение есть. Потому что, даже если пользователь уже давал это разрешение, он всегда может зайти в настройки приложения и отменить его. И если вы после этого не выполните проверку, то получите ошибку при выполнении операции.

## Don’t ask again

Когда вы первый раз делаете запрос на какое-либо разрешение, пользователь может отказать. При последующих запросах этого же разрешения, в диалоге появится чекбокс Don’t ask again

Если пользователь включит этот чекбокс, то при последующих ваших запросах диалог не будет отображаться, а в onRequestPermissionsResult сразу будет приходить отказ.

## Объяснение для пользователя

Когда вы запрашиваете разрешение, пользователю должно быть очевидно, зачем приложению понадобилось это разрешение, и у него не должно возникать вопросов. Но случаи бывают разные, и вы можете решить, что вам надо явно объяснить пользователю, почему приложению понадобилось это разрешение.

Диалог, который показывается при запросе разрешения, - системный, вы не можете менять его содержимое и добавлять туда свой текст. Но вы можете сделать свой диалог или что-то подобное и показать его перед тем, как будете делать запрос разрешения.

Есть метод [shouldShowRequestPermissionRationale](https://developer.android.com/reference/android/support/v4/app/ActivityCompat.html#shouldShowRequestPermissionRationale(android.app.Activity,%20java.lang.String)), который может быть полезен в данной ситуации. Передаете ему название разрешения, а он вам в виде boolean ответит, надо ли показывать объяснение для пользователя.

Т.е. вы сначала проверяете наличие разрешения. Если его нет, то вызываете shouldShowRequestPermissionRationale, чтобы решить, надо ли показывать объяснение пользователю. Если не надо, то делаете запрос разрешения. А если надо, то показываете ваш диалог с объяснением, а после этого диалога делаете запрос разрешения.

Алгоритм работы метода shouldShowRequestPermissionRationale прост.

Если вы еще ни разу не запрашивали это разрешение, то он вернет false. Т.е. перед первым запросом разрешения ничего объяснять не надо.

Если вы ранее уже запрашивали это разрешение и пользователь отказал, то метод вернет true. Т.е. пользователь не понимает, почему он должен давать это разрешение, и надо ему это объяснить.

Если пользователь ставил галку Don’t ask again, то метод вернет false. Запрос полномочий все равно не будет выполнен. Объяснять что-то не имеет смысла.

Разумеется, вы можете показывать дополнительную информацию согласно вашим правилам и не использовать метод shouldShowRequestPermissionRationale.

## Группы

Dangerous разрешения собраны в группы. Список групп можно посмотреть [здесь](https://developer.android.com/reference/android/Manifest.permission_group.html). Если вы запросили одно разрешение из группы и пользователь предоставил вам его, то вы автоматически получаете все разрешения этой группы.

Например, разрешения READ\_CONTACTS и WRITE\_CONTACTS принадлежат группе CONTACTS. И если пользователь уже подтверждал разрешение на READ\_CONTACTS, то при проверке WRITE\_CONTACTS вы получите PERMISSION\_GRANTED.

## Google Maps

1. Рассказать коротко про сервисы гугл .Настроить серисы, добавить карты <https://developers.google.com/android/guides/setup>   
   implementation 'com.google.android.gms:play-services-maps:15.0.1'

Процесс создания ключа: <https://developers.google.com/maps/documentation/android-sdk/signup>

## Location

com.google.android.gms:play-services-location:15.0.1

Разобрать пример <https://github.com/googlesamples/android-play-location/tree/master/LocationUpdates>

Алгоритм:

1. Проверить permissions
2. Проверить, включена ли в настройках локация
3. Начать получать данные

#### Задание:

1. В уроке с мостами реализовать экран с картой.
2. Реализовывать предлагается с SupportMapFragment
3. Запросить разрешения на геолокацию. Если разрешение дано - включить встроенный слой с геопозицией пользователя   
   mMap.setMyLocationEnabled(true);
4. Продвинутый уровень:   
   Разобрать пример <https://github.com/googlesamples/android-play-location/tree/master/LocationUpdates>   
   Не использовать встроенные в карту средства отображения локации. Вместо этого использовать FusedLocationProviderClient. Необходимо получать реальную геопозицию пользователя и отображать ее на карте. При этом каждый раз, когда приходят новые данные по локации - двигать маркер. Не забывать приостанавливать получение локации в onPause и возобновлять в onResume.

#### Полезные ссылки:

1. <https://startandroid.ru/ru/blog/508-android-permissions.html> - про разрешения на русском
2. <https://developer.android.com/guide/topics/permissions/overview> - про разрешения официальная документация и далее по блоку
3. <https://developers.google.com/maps/documentation/android-sdk/start> - про гугл мэпс. На уроке рассказал лишь часть, остальное - самостоятельно
4. Все про локацию в официальном гайде : <https://developer.android.com/training/location/>
5. Пример с локацией: <https://github.com/googlesamples/android-play-location/tree/master/LocationUpdates>