На самом деле, было бы хорошо подгружать погоду в сервисе — таким образом мы можем периодически подгружать её в фоне, а также показывать информацию в уведомлении.

Получаем геолокацию в сервисе

В первую очередь перенесём код запроса геолокации из MainActivity в наш сервис.

Запрос разрешений останется в Activity, но вся работа, связанная с гео будет находиться в сервисе.

Кроме того, удалим весь лишний код из сервиса. В итоге получим это:

public class WeatherService extends Service {

private static final String TAG = "WeatherService";

private LocationManager locationManager;

@Nullable

@Override

public IBinder onBind(Intent intent) {

return null;

}

@Override

public void onCreate() {

super.onCreate();

Notification notification = createNotification();

startForeground(1, notification);

setupLocation();

}

@Override

public void onDestroy() {

locationManager.removeUpdates(locationListener);

super.onDestroy();

}

@Override

public int onStartCommand(Intent intent, int flags, int startId) {

return START\_STICKY;

}

/\*\*

\* Создаём уведомление

\*/

private Notification createNotification() {

NotificationCompat.Builder builder = new NotificationCompat.Builder(this);

builder.setSmallIcon(R.drawable.ic\_notification);

builder.setContentTitle("Title");

builder.setContentText("Text");

builder.setOngoing(true);

Intent mainIntent = new Intent(this, MainActivity.class);

PendingIntent pendingIntent = PendingIntent.getActivity(

this,

0,

mainIntent,

PendingIntent.FLAG\_UPDATE\_CURRENT

);

builder.setContentIntent(pendingIntent);

return builder.build();

}

/\*\*

\* Подписываемся на обновления гео

\*/

private void setupLocation() {

// Получаем LocationManager

locationManager = (LocationManager) getSystemService(LOCATION\_SERVICE);

// Получаем лучший провайдер

Criteria criteria = new Criteria();

String bestProvider = locationManager.getBestProvider(criteria, true);

Log.v(TAG, "Best provider: " + bestProvider);

if (bestProvider != null) {

// На всякий случай проверим, не убрал ли пользователь разрешение на ГЕО

if (ActivityCompat.checkSelfPermission(this,

Manifest.permission.ACCESS\_FINE\_LOCATION) != PackageManager.PERMISSION\_GRANTED

&& ActivityCompat.checkSelfPermission(this,

Manifest.permission.ACCESS\_COARSE\_LOCATION) != PackageManager.PERMISSION\_GRANTED) {

return;

}

// Получаем последнюю доступную позицию

Location lastKnownLocation = locationManager.getLastKnownLocation(bestProvider);

Log.v(TAG, "Last location: " + lastKnownLocation);

// Подписываемся на обновления

locationManager.requestLocationUpdates(

bestProvider, // провайдер

0, // мин. время

0, // мин. расстояние

locationListener

);

}

}

/\*\*

\* Слушатель для обновления гео

\*/

private final LocationListener locationListener = new LocationListener() {

@Override

public void onLocationChanged(Location location) {

Log.v(TAG, "Location changed: " + location);

}

@Override

public void onStatusChanged(String provider, int status, Bundle extras) {

Log.v(TAG, "Status changed: " + provider + ", status: " + status);

}

@Override

public void onProviderEnabled(String provider) {

Log.v(TAG, "Provider enabled: " + provider);

}

@Override

public void onProviderDisabled(String provider) {

Log.v(TAG, "Provider disabled: " + provider);

}

};

}

В MainActivity добавим новый метод, который будет запускать сервис:

private void startWeatherService() {

Intent intent = new Intent(this, WeatherService.class);

startService(intent);

}

Будем вызывать его в checkAndRequestGeoPermission() и onRequestPermissionsResult():

private void checkAndRequestGeoPermission() {

int permissionCheck = ContextCompat.checkSelfPermission(this,

Manifest.permission.ACCESS\_FINE\_LOCATION);

if (permissionCheck != PackageManager.PERMISSION\_GRANTED) {

ActivityCompat.requestPermissions(this,

new String[]{Manifest.permission.ACCESS\_FINE\_LOCATION},

REQUEST\_CODE\_LOCATION\_PERMISSION);

} else {

startWeatherService();

}

}

@Override

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions, @NonNull int[] grantResults) {

super.onRequestPermissionsResult(requestCode, permissions, grantResults);

if (requestCode == REQUEST\_CODE\_LOCATION\_PERMISSION) {

if (grantResults.length > 0

&& grantResults[0] == PackageManager.PERMISSION\_GRANTED) {

startWeatherService();

} else {

// Нет гео

// Попробуем показать ещё раз

checkAndRequestGeoPermission();

}

}

}

Попробуйте запустить приложение и увидите, что мы успешно получаем геолокацию из сервиса:

11-16 23:58:01.671 4552-4552/com.skillberg.weather2 V/WeatherService: Best provider: gps

11-16 23:58:01.678 4552-4552/com.skillberg.weather2 V/WeatherService: Last location: Location[gps 37.421998,-121.084000 acc=20 et=+18s4ms alt=0.0 {Bundle[EMPTY\_PARCEL]}]

Делаем запросы к API в сервисе

Добавим инстанс Api в сервис:

private final Api api = ApiFactory.createApi();

А ещё нам пригодится поле для текущей погоды:

@Nullable

private CurrentWeather currentWeather;

Создадим новый метод:

private void getCurrentWeather(double latitude, double longitude) {

}

В нём создадим уже знакомый нам запрос к API:

Call<CurrentWeather> call = api.getCurrentWeather(

latitude,

longitude,

Constants.API\_KEY,

Constants.DEFAULT\_UNITS

);

Однако, в отличие от примера с тестами, мы будем делать асинхронный вызов. Для этого нам понадобится метод enqueue(). Он принимает в качестве параметра Callback с методами:

* onResponse() — будет вызван, если метод выполнился успешно, либо неуспешно с HTTP-ошибкой.
* onFailure() — будет вызван, если произошла ошибка сети.

lang=java call.enqueue(new Callback() { @Override public void onResponse(@NonNull Call call, @NonNull Response response) { if (response.isSuccessful()) { CurrentWeather currentWeather = response.body();

Log.i(TAG, "Got weather: " + currentWeather);

WeatherService.this.currentWeather = currentWeather;

setCurrentWeather();

} else {

Log.e(TAG, "Failed to get current weather. Code: " + response.code());

}

}

@Override

public void onFailure(@NonNull Call<CurrentWeather> call, @NonNull Throwable t) {

Log.e(TAG, "Failed to get current weather: " + t.getMessage());

}

});

Также добавим новый метод setCurrentWeather(), который будет вызываться, когда мы получили новую погоду:

private void setCurrentWeather() {

Notification notification = createNotification();

startForeground(1, notification);

}

Поскольку мы будем показывать текущую погоду в уведомлении, нам потребуются соответствующие строковые ресурсы:

<string name="title\_updating\_weather">Updating</string>

<string name="text\_updating\_weather">Updating weather…</string>

Такой текст будет показываться, когда обновление погоды в процессе.

<string name="title\_current\_weather">%1$d° — %2$s</string>

<string name="text\_current\_weather">Min: %1$d°, max: %2$d°</string>

А такой — когда получили погоду.

* В заголовке будут **температура** и **город**.
* В тексте будет **минимальная** и **максимальная** температура.

Ну и, конечно, обновим код показа уведомления:

private Notification createNotification() {

NotificationCompat.Builder builder = new NotificationCompat.Builder(this);

builder.setSmallIcon(R.drawable.ic\_notification);

if (currentWeather == null) {

builder.setContentTitle(getString(R.string.title\_updating\_weather));

builder.setContentText(getString(R.string.text\_updating\_weather));

} else {

builder.setContentTitle(getString(

R.string.title\_current\_weather,

(int) currentWeather.getMain().getTemp(),

currentWeather.getCityName()));

builder.setContentText(getString(

R.string.text\_current\_weather,

(int) currentWeather.getMain().getMinTemp(),

(int) currentWeather.getMain().getMaxTemp()

));

}

builder.setOngoing(true);

// ...

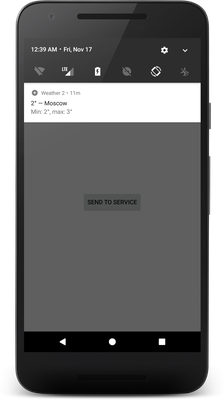
И, конечно же, обновим информацию о погоде в методе setupLocation() сразу после получения последней геопозиции:

Location lastKnownLocation = locationManager.getLastKnownLocation(bestProvider);

Log.v(TAG, "Last location: " + lastKnownLocation);

getCurrentWeather(lastKnownLocation.getLatitude(), lastKnownLocation.getLongitude());

Запустите приложение и увидите, что погода получена успешно:

[](https://skillberg.com/media/uploads/2017/11/17/ac77ec4a8f084bfda59709c1fa9b5a76.png)Текущая погода в уведомлении

Увеличиваем интервал получения погоды

Как вы помните, в запросе обновления геопозиции мы указали нули в качестве минимального времени и расстояния.

Это не очень хорошая практика, так как нам не нужна такая точность. Нам достаточно изменения позиции на 10 километров и минимального интервала в 60 минут:

locationManager.requestLocationUpdates(

bestProvider, // провайдер

TimeUnit.HOURS.toMillis(1), // мин. время

10000, // мин. расстояние

locationListener

);

Кроме того, не забудьте добавить получение погоды в метод onLocationChanged():

@Override

public void onLocationChanged(Location location) {

Log.v(TAG, "Location changed: " + location);

getCurrentWeather(location.getLatitude(), location.getLongitude());

}

BroadcastReceiver

В Android существует механизм **BroadcastReceiver**, позволяющий обмениваться сообщениями между приложениями.

Принцип действия таков:

* Приложение подписывается на определённые сообщения.
* Другое (или это же) приложение отправляет сообщение, используя метод sendBroadcast().
* Когда приходит сообщение, вызывается метод onReceive() в BroadcastReceiver.

Существует два способа регистрации ресивера:

* В AndroidManifest.xml.
* В рантайме, например, в Activity.

У каждого Broadcast-сообщения есть так называемый Action — уникальный идентификатор. Есть предопределённые **Action**, но вы можете создать свой **Action**.

Регистрация в манифесте

К примеру, нам нужно запустить приложение, как только система загрузилась. Для этого нужно подписаться на android.intent.action.BOOT\_COMPLETED.

Давайте создадим BroadcastReceiver, который будет вызываться при запуске системы:

public class BootReceiver extends BroadcastReceiver {

@Override

public void onReceive(Context context, Intent intent) {

}

}

В методе onReceive() мы запустим сервис:

if (Intent.ACTION\_BOOT\_COMPLETED.equals(intent.getAction())) {

context.startService(new Intent(context, WeatherService.class));

}

Зачем мы проверяем **Action**? Дело в том, что в некоторых случаях система может вызвать BroadcastReceiver для другого Action. Всегда проверяйте, что произошло именно то событие, которое ждёте!

**Обратите внимание:** BroadcastReceiver предназначен для выполнения очень краткосрочных операций (не более нескольких секунд). Если вам нужно выполнить длительную операцию, запускайте сервис, иначе система убьёт BroadcastReceiver. Кроме того, код будет выполнен на UI-потоке.

Осталось зарегистрировать BroadcastReceiver в манифесте:

<receiver android:name=".BootReceiver">

</receiver>

Внутрь этого тега добавьте специальный тег <intent-filter> — он указывает системе, по каким критериям система должна вызвать наш BroadcastReceiver:

<intent-filter>

<action android:name="android.intent.action.BOOT\_COMPLETED" />

</intent-filter>

Кроме того, чтобы система вызвала наш ресивер, потребуется разрешение:

<uses-permission android:name="android.permission.RECEIVE\_BOOT\_COMPLETED" />

Готово! Запустите приложение, перезагрузите устройство, и вы увидите, как приложение запустилось!

**Обратите внимание:** в последних версиях Android очень сильно ограничили список системных экшенов, которые можно получить при регистрации в манифесте. Ознакомьтесь со [списком](https://developer.android.com/guide/components/broadcast-exceptions.html) тех Action-ов, которые можно регистрировать в манифесте. Все остальные нужно регистрировать в рантайме.

**Обратите внимание:** ни один BroadcastReceiver не будет вызван, если пользователь ни разу не запускал приложение после установки.

Регистрация в рантайме

Когда сервис получил погоду, нужно отправить информацию о погоде в MainActivity. Хороший способ сделать это — отправить **Broadcast**.

Создадим для этого собственный **Action** в MainActivity:

public static final String ACTION\_GOT\_WEATHER = "com.skillberg.weather2.ACTION\_GOT\_WEATHER";

Мы можем передать данные внутри **Broadcast** — будем передавать текущую погоду таким образом. Создадим ещё и константу для **Extra**:

public static final String EXTRA\_CURRENT\_WEATHER = "current\_weather";

Создадим сабкласс BroadcastReceiver и его инстанс внутри MainActivity:

private class WeatherReceiver extends BroadcastReceiver {

@Override

public void onReceive(Context context, Intent intent) {

if (ACTION\_GOT\_WEATHER.equals(intent.getAction())) {

Log.i(TAG, "Got weather!");

}

}

}

private final WeatherReceiver weatherReceiver = new WeatherReceiver();

Поскольку мы будем передавать данные о погоде через **Intent**, нужно, чтобы объект погоды (и дочерние объекты) реализовывал интерфейс Parcelable. Вы уже умеете это делать, так что сделайте это самостоятельно.

Теперь в ресивере получим погоду:

@Override

public void onReceive(Context context, Intent intent) {

if (ACTION\_GOT\_WEATHER.equals(intent.getAction())) {

CurrentWeather currentWeather = intent.getParcelableExtra(EXTRA\_CURRENT\_WEATHER);

Log.i(TAG, "Got weather: " + currentWeather);

}

}

Осталось зарегистрировать **ресивер** (и не забыть разрегистрировать, когда Activity завершается). В зависимости от ситуации это можно делать в различных методах, в нашем случае лучше всего будет регистрировать в onStart() и разрегистрировать в onStop():

@Override

protected void onStart() {

super.onStart();

IntentFilter intentFilter = new IntentFilter(ACTION\_GOT\_WEATHER);

registerReceiver(weatherReceiver, intentFilter);

}

@Override

protected void onStop() {

unregisterReceiver(weatherReceiver);

super.onStop();

}

Готово!

Теперь отправим **broadcast** из сервиса, из метода setCurrentWeather(). Сделать это очень легко:

Intent intent = new Intent(MainActivity.ACTION\_GOT\_WEATHER);

intent.putExtra(MainActivity.EXTRA\_CURRENT\_WEATHER, currentWeather);

sendBroadcast(intent);

Всё! Запустите приложение и посмотрите в логи:

11-17 01:35:57.126 3883-3883/com.skillberg.weather2 I/WeatherService: Got weather: CurrentWeather{main=ForecastMain{temp=1.5, pressure=1019.0, humidity=93.0, minTemp=1.0, maxTemp=2.0}, wind=ForecastWind{speed=7.0, degree=300.0}, cityName='Moscow'}

11-17 01:35:57.136 3883-3883/com.skillberg.weather2 I/Weather/MainActivity: Got weather: CurrentWeather{main=ForecastMain{temp=1.5, pressure=1019.0, humidity=93.0, minTemp=1.0, maxTemp=2.0}, wind=ForecastWind{speed=7.0, degree=300.0}, cityName='Moscow'}

SharedPreferences

Помимо хранения данных в файлах или базах данных SQLite, в Android есть простой механизм для сохранения настроек — SharedPreferences.

По сути, это XML-файл, хранящийся внутри приватной директории приложения.

Давайте сохранять время последнего обновления погоды в SharedPreferences и получать его в MainActivity.

В методе setCurrentWeather в WeatherService добавьте такой код:

SharedPreferences sharedPreferences = getSharedPreferences("weather", MODE\_PRIVATE);

sharedPreferences

.edit()

.putLong("last\_update", System.currentTimeMillis())

.apply();

В первой строчке получаем инстанс SharedPreferences.

Первый параметр метода getSharedPreferences() — имя файла, в котором будут храниться настройки, второй — режим доступа к файлу настроек. Мы не хотим никому давать доступ к файлу, поэтому используем Context.MODE\_PRIVATE.

Во второй строчке мы добавляем новое значение в настройки. Первый параметр — **ключ сохраняемого элемента**, используя который мы можем получить доступ к элементу, второй — **сохраняемое значение**.

Теперь в методе onCreate() в MainActivity получим значение, сохранённое в настройках:

SharedPreferences sharedPreferences = getSharedPreferences("weather", MODE\_PRIVATE);

long lastUpdate = sharedPreferences.getLong("last\_update", 0);

Log.i(TAG, "Last update: " + lastUpdate);

Собственно, тут тоже всё очевидно. В методе getLong() первый параметр — **ключ**, второй — **значение по умолчанию**, которое будет возвращено в случае, если с таким ключом ещё ничего не было сохранено.

Запустите приложение и посмотрите в логи:

11-17 01:56:05.099 4225-4225/com.skillberg.weather2 I/Weather/MainActivity: Last update: 0

Само собой, при первом запуске мы получим 0, то есть значение по умолчанию.

Перезапустите приложение:

11-17 01:56:50.824 4299-4299/com.skillberg.weather2 I/Weather/MainActivity: Last update: 1510872965519

Супер!

ПРИМЕРЫ КОДА

[ИСХОДНЫЙ КОД УРОКА](https://github.com/Skillberg/WeatherAndroid2/tree/lesson-35)