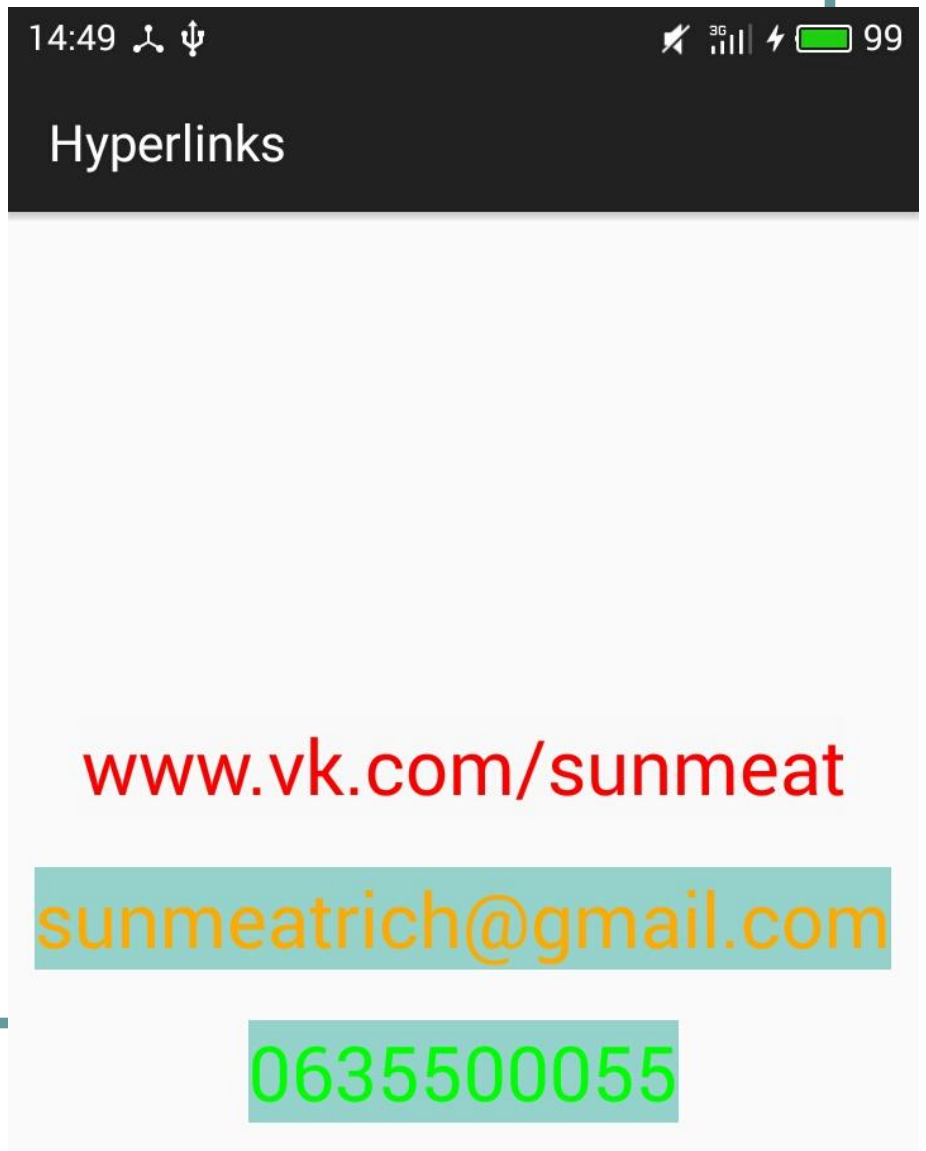
Broadcast receivers

android

Создание гиперссылок

XML-разметка:

https://git.io/vPpN6



Вспомним про интенты

Намерение (интент) - это механизм для описания одного действия – например, выбора фотографии, отправки письма, совершения звонка, запуска браузера с переходом по указанному адресу, запуска сервиса или другого активити в приложении, и тд. Сегодня мы рассмотрим применение интентов для трансляции сообщений по системе. Любое приложение способно зарегистрировать **широковещательный приёмник** (broadcast receiver) и отслеживать интенты с возможностью на них реагировать. Это позволяет создавать приложения, использующие событийную модель, в основе которой лежат внутренние, системные или сторонние события, передаваемые внешними программами.

http://developer.alexanderklimov.ru/android/theory/intent.php

Примеры неявных интентов

```
// открытие браузера
Uri uriUrl = Uri.parse("http://google.com/");
Intent launchBrowser = new Intent(Intent.ACTION_VIEW, uriUrl);
startActivity(launchBrowser);

// отправка поискового запроса
Intent search = new Intent(Intent.ACTION_WEB_SEARCH);
search.putExtra(SearchManager.QUERY, "android");
startActivity(search);

// запуск камеры
Intent intent = new Intent(android.provider.MediaStore.ACTION_IMAGE_CAPTURE);
startActivity(intent);

// запустить одну из программ с передачей номера телефона
Intent dial = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:0635500055"));
startActivity(dial);
```

Виды интентов: явный

```
// код первого активити
ArrayList<File> fileList = new ArrayList<>();
Intent intent = new Intent(MainActivity.this,
SecondActivity.class); // явный вызов
intent.putExtra("FILES_LIST", fileList);
startActivity(intent);

// код второго активити
ArrayList<File> filelist = (ArrayList<File>)getIntent()
.getSerializableExtra("FILES_LIST");
```

Виды интентов: неявный

Неявные намерения — это механизм, позволяющий запрашивать компоненты приложений с помощью действий. Можно попросить систему запустить активити, выполняющее заданное действие, не зная ничего ни о самом активити, ни о приложении. Например, можно набрать телефонный номер, но не знать, какую программу дозвона выберет пользователь на своём телефоне. При создании неявного интента необходимо назначить действие, которое должно выполниться, а также при желании указать вспомогательный путь **URI** к тем данным, что нужно обработать.

Действия неявных интентов

- ACTION_ANSWER
- ACTION_CALL
- ACTION_DELETE
- ACTION_EDIT
- ACTION_INSERT
- ACTION_HEADSET_PLUG
- ACTION_MAIN
- ACTION_PICK
- ACTION_SEARCH
- ACTION_SEND
- ACTION_SENDTO
- ACTION_SYNC
- ACTION_TIMEZONE_CHANGED
- ACTION_VIEW
- ACTION_WEB_SEARCH

http://developer.alexanderklimov.ru/android/theory/intent.php

Широковещательные сообщения

В системе Android существует понятие **широковещательных сообщений**, которые можно как отправлять, так и принимать. Для отправки сообщений, предназначенных не какому-то отдельному приложению, объекту или компоненту, а всем подряд – применяются опять же таки интенты. И любая программа, оборудованная специальным рисивером, может поймать это сообщение и предпринять свои ответные действия на основе полученной информации.

Создание сообщения

Сообщения может посылать операционная система, наше приложение или чужие приложения.

Для создания сообщения необходимо оформить объект-интент, установив ему дополнительные сведения (действие, данные и категорию). Строка действия должна быть уникальной, чтобы рисиверы смогли точно идентифицировать переданное сообщение. Обычно, строка-идентификатор действия составляется по правилам именования пакетов Java. Например, так:

```
public static final String MY_MESSAGE =
"com.sunmeat.thirdproject.MESSAGE_1";
```

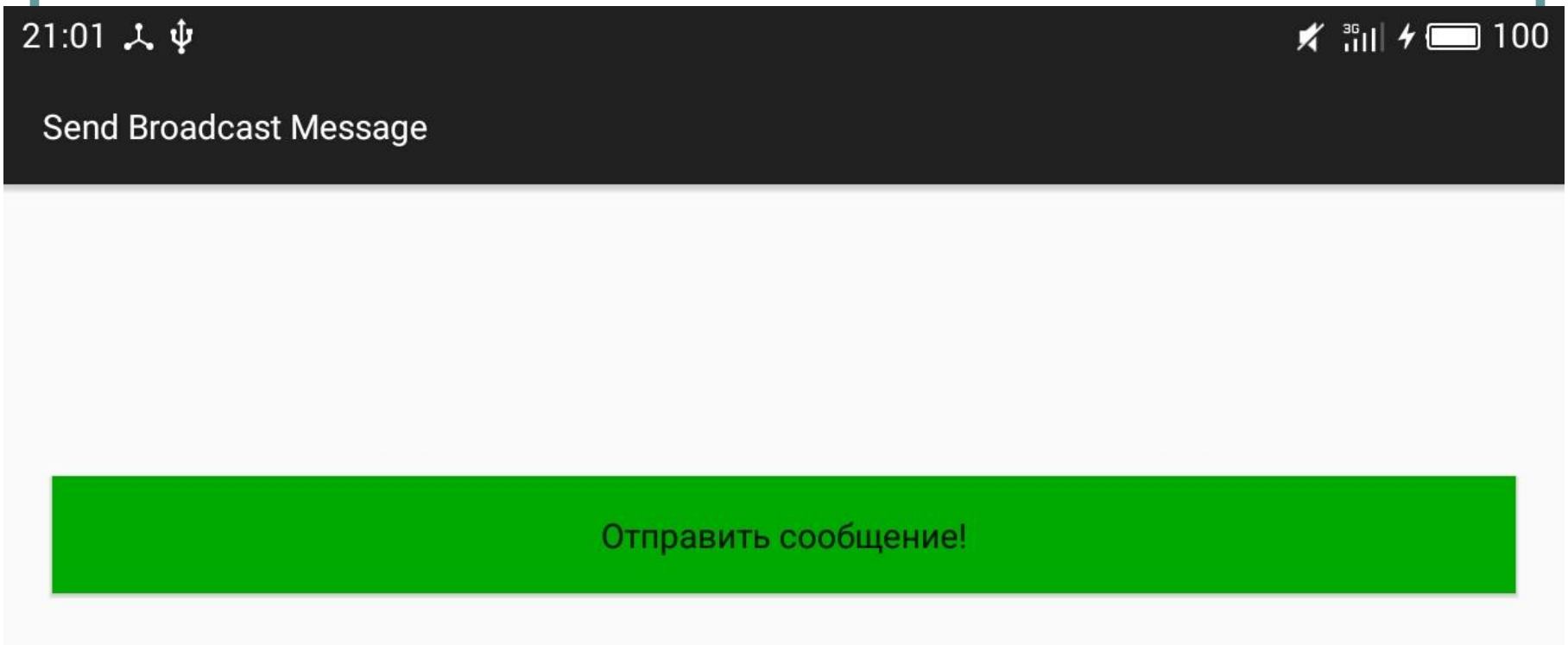
Отправка сообщения

```
Intent message = new Intent();
intent.setAction(MY_MESSAGE);


intent.putExtra("name", "Alex");
intent.putExtra("age", "28");


sendBroadcast(message);
```

Пример кода

MainActivity.java и activity_main.xml:

https://git.io/vXvVj

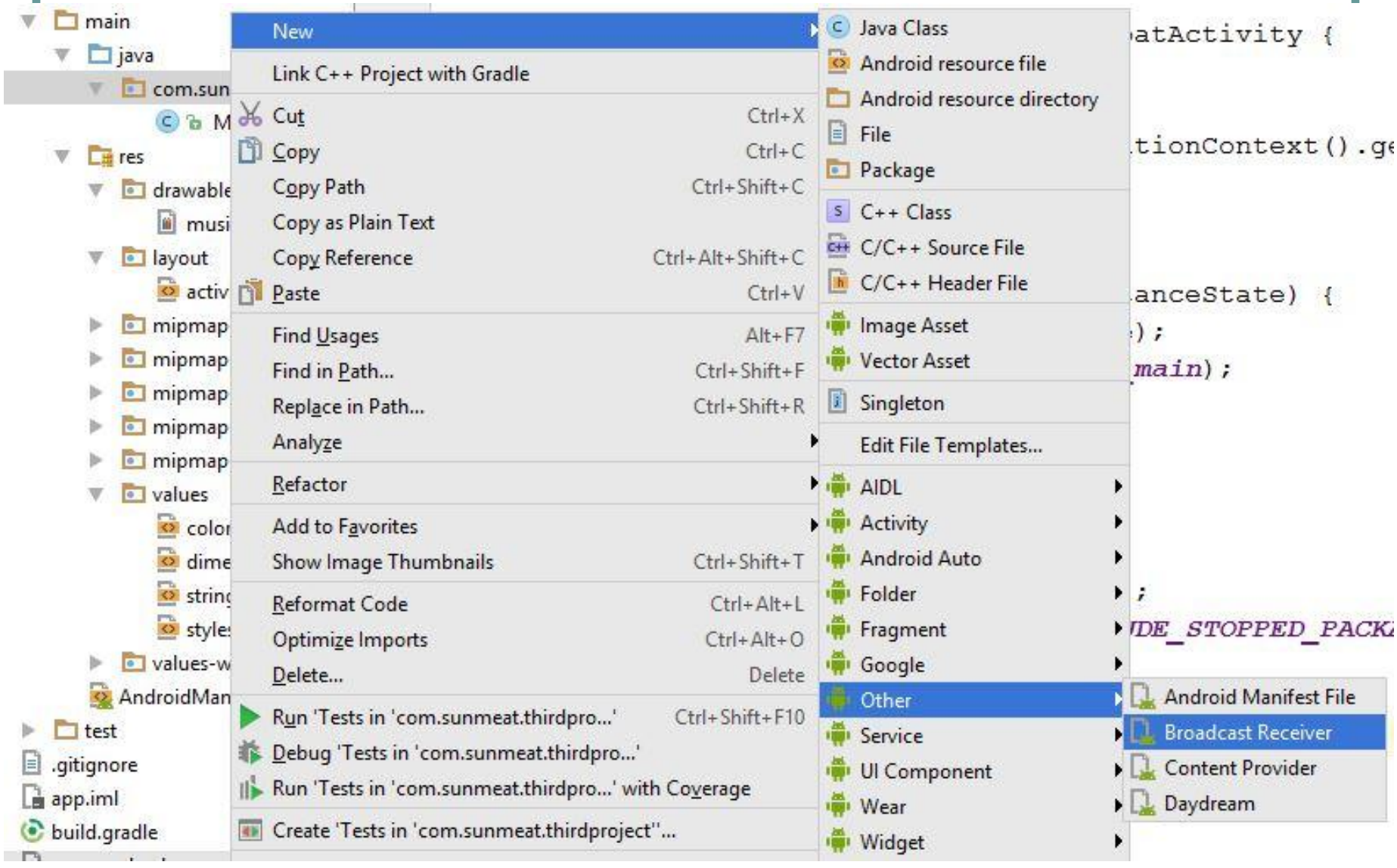


Нужен рисивер

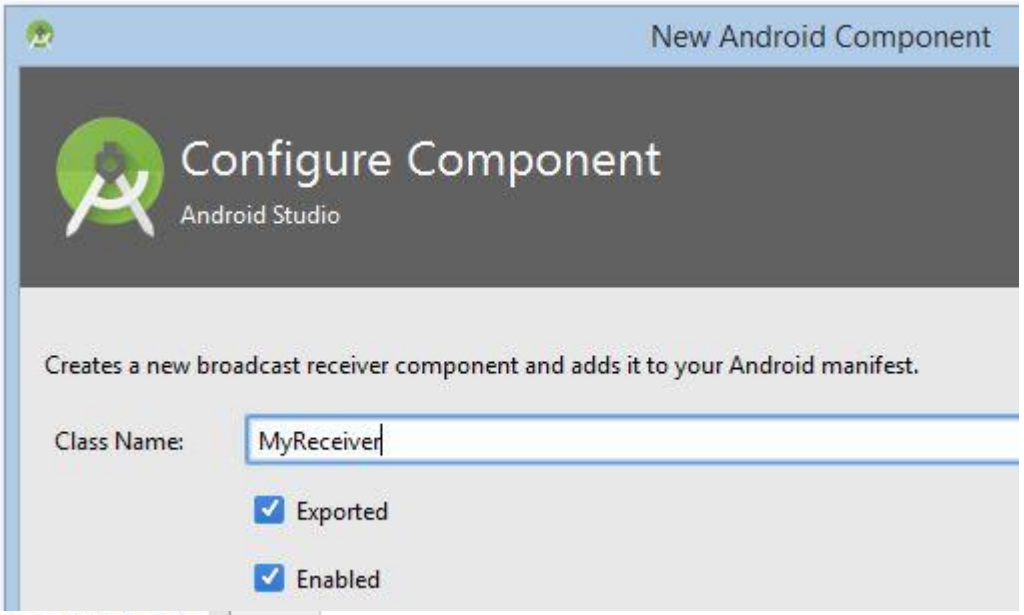
Сообщение-то отправлено, однако ушло оно в никуда, так как ни одно приложение в системе не оборудовано рисивером для него. Попробуем создать рисивер прямо в этом же приложении, и будем сами принимать свои же сообщения.

Рисивер представляет собой обычный Java-класс на основе **BroadcastReceiver**. Можно создать класс вручную и наполнить его необходимыми методами, но в студии есть готовый шаблон, который поможет немного сэкономить время.

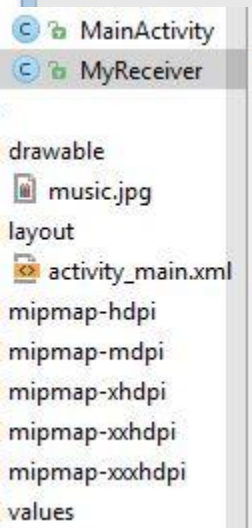
Создание рисивера



Прописка в манифесте




```
<receiver
    android:name=".MyReceiver"
    android:enabled="true"
    android:exported="true">
</receiver>
```




```
public class MyReceiver extends BroadcastReceiver {
    public MyReceiver() {
    }

    @Override
    public void onReceive(Context context, Intent intent) {
        // TODO: This method is called when the BroadcastReceiver is receiving
        // an Intent broadcast.
        throw new UnsupportedOperationException("Not yet implemented");
    }
}
```

Отредактируем файлы

```
<receiver
    android:name=".MyReceiver"
    android:enabled="true"
    android:exported="true">

    <intent-filter>
        <action android:name="com.sunmeat.thirdproject.MSG1" />
    </intent-filter>

</receiver>
```

```
@Override
public void onReceive(Context context, Intent intent) {
    Toast.makeText(context, "Обнаружено сообщение: " +
                    intent.getStringExtra("someData"),
            Toast.LENGTH_LONG).show();
}
```

Системные сообщения

На практике, чаще встречается потребность принимать сообщения, а не отправлять их. В первую очередь это касается сообщений от системы.

Примерами таких сообщений могут быть:

- Низкий заряд батареи
- Нажатие на кнопку камеры
- Установка нового приложения

Способы создания приёмника

Рисивер можно создать двумя способами - через манифест и программно через метод **registerReceiver()**. Между двумя способами есть существенная разница. Рисивер, заданный в манифесте, известен системе, которая сканирует файлы манифеста всех установленных приложений. Поэтому, даже если приложение не запущено, оно всё равно сможет отреагировать на поступающее сообщение. Рисивер, созданный программно, может работать только в том случае, когда активити приложения запущено. Некоторые системные сообщения могут обрабатываться только программными рисиверами. Это не лишено смысла, например, если приложение не запущено, ему нет смысла принимать сообщения о заряде батареи.

Следим за уровнем громкости

Нажимаем на аппаратные кнопочки уровня громкости (вверх-вниз):

```
<intent-filter>
    <action android:name="android.media.VOLUME_CHANGED_ACTION" />
</intent-filter>
```

```
@Override
public void onReceive(Context context, Intent intent) {
    int volume = (Integer) intent.getExtras()
            .get("android.media.EXTRA_VOLUME_STREAM_VALUE");
    Toast.makeText(context, volume + "", Toast.LENGTH_SHORT).show();
}
```

Следим за питанием

После запуска отключаем кабель питания:

```
<intent-filter>
    <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED" />
</intent-filter>
```

```
@Override
public void onReceive(Context context, Intent intent) {
    if (intent.getAction()
            .equalsIgnoreCase("android.intent.action.ACTION_POWER_DISCONNECTED")) {

        Toast.makeText(context, "Не отключай меня... я кушать хочу ))",
                Toast.LENGTH_LONG).show();
    }
}
```

https://developer.android.com/training/monitoring-device-state/battery-monitoring.html

Следим за вай-фаем

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


<intent-filter>
    <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
</intent-filter>
```

MyReceiver.java:

https://git.io/vXvF3

Следим за смсками

```
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.READ_SMS" />
<uses-permission android:name="android.permission.SEND_SMS" />
```

```
<intent-filter>
    <action android:name="android.provider.Telephony.SMS_RECEIVED" />
</intent-filter>
```

MyReceiver.java:

https://git.io/vXfeW

Следим за входящими

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

```
<intent-filter>
    <action android:name="android.intent.action.PHONE_STATE" />
</intent-filter>
```

MyReceiver.java:

https://git.io/vXJU8

Системный интент TIME_TICK

Системное событие TIME_TICK срабатывает каждую минуту:

https://git.io/vXJLM

Запуск после перезагрузки

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

 <receiver
     android:name=".MyReceiver"
     android:enabled="true"
     android:permission="android.permission.RECEIVE_BOOT_COMPLETED">
     <intent-filter>
         <action android:name="android.intent.action.BOOT_COMPLETED" />
     </intent-filter>
 </receiver>
```

```
@Override
public void onReceive(Context context, Intent intent) {
    String action = intent.getAction();
    if (action.equalsIgnoreCase("android.intent.action.BOOT_COMPLETED")) {
        Intent i = new Intent(context, MainActivity.class);
        i.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        context.startActivity(i);
    }
}
```

http://developer.alexanderklimov.ru/android/theory/boot.php

Жизненный цикл рисиверов

У рисиверов только один колбэк-метод:

void onReceive(Context context, Intent intent);

Когда сообщение прибывает, Android вызывает метод **onReceive()** и передаёт в него интент, содержащий сообщение. Рисивер активен только во время выполнения этого метода. Выполняющийся в настоящее время код в методе **onReceive()** является приоритетным процессом и будет сохранён (защищён от уничтожения системой), кроме случаев критического недостатка памяти. Когда программа возвращается из метода **onReceive()**, рисивер становится неактивным – и такой процесс может быть уничтожен системой в любое время, когда память, которую он потребляет, будет необходима другим процессам.

http://developer.alexanderklimov.ru/android/theory/broadcast.php

Другие действия для рисиверов

Некоторые из встроенных действий, представленных как константы в классе **Intent**, которые используются для того, чтобы проследить изменения состояния устройства:

- **ACTION_DATE_CHANGED** и **ACTION_TIME_CHANGED** — запускаются при ручном изменении пользователем даты или времени на устройстве
- **ACTION_SCREEN_OFF** и **ACTION_SCREEN_ON** — передаются, когда экран выключается или включается
- **ACTION_TIMEZONE_CHANGED** — передаётся при изменении текущего часового пояса

Практика

Создать рисивер в другом приложении. Сделать так, чтобы одно приложение отправляло сообщение, а другое – получало его.

Практика

Keep the screen on:

https://developer.android.com/training/scheduling/wakelock.html#screen

Практика

Активити ставит себя на передний план каждые 5 секунд:

http://stackoverflow.com/questions/3801562/how-to-bring-an-activity-to-front-in-android