Broadcast (Широковещательные сообщения)

Практическая часть показана в [отдельной статье](http://developer.alexanderklimov.ru/android/broadcast.php).

Есть два основных класса сообщений, которые могут быть получены приёмником широковещательных сообщений:

* нормальные сообщения о намерениях (Normal broadcasts) — посылаемые вызовом метода **context.sendBroadcast()** и являющиеся полностью асинхронными. Все получатели сообщения выполняются в неопределённом порядке, часто в одно и то же время. Это более эффективно, но означает, что получатели не могут использовать результат или прервать сообщение;
* порядковые сообщения о намерениях (Ordered broadcasts), которые посылаются методом **Context.sendOrderedBroadcast()**. Эти сообщения посылаются одному получателю за один раз. Поскольку каждое полученное сообщение выполняется по очереди, он может в случае необходимости полностью прервать сообщение, чтобы его не успели передать другим приёмникам. Приёмниками сообщений можно управлять с помощью атрибута **android:priority** фильтра сообщений; приёмники сообщений, имеющие одинаковый приоритет, будут выполнены в произвольном порядке.

Приёмник широковещательных сообщений — это компонент для получения внешних событий и реакции на них. Инициализировать передачи могут другие приложения или службы. Класс **BroadcastReceiver** является базовым для класса, в котором должны происходить получение и обработка сообщений, посылаемых клиентским приложением с помощью вызова метода **sendBroadcast()**. Вы можете или динамически зарегистрировать экземпляр класса **BroadcastReceiver** с помощью метода **Context.registerReceiver()**, или статически создать его в элементе **<receiver>** в файле манифеста приложения.

Для объекта **BroadcastReceiver** нет никаких возможностей видеть или фиксировать намерения, используемые в методе **startActivity()**. Аналогично, когда вы передали намерение для запуска активности через объект **BroadcastReceiver**, вы не сможете найти или запустить требуемую активность. Эти две операции семантически полностью различаются: запуск активности через намерение является приоритетной операцией для системы, изменяющей содержимое экрана устройства, с которым в настоящее время взаимодействует пользователь. Передача широковещательных сообщений для системы является фоновой работой, о которой обычно не знает пользователь и которая, соответственно, имеет более низкий приоритет.

Жизненный цикл приемников широковещательных сообщений

Приемник широковещательных сообщений имеет единственный метод обратного вызова:

void onReceive(Context context, Intent intent)

Когда широковещательное сообщение прибывает для получателя сообщения, Android вызывает его методом **onReceive()** и передаёт в него объект **Intent**, содержащий сообщение. Приёмник широковещательных сообщений является активным только во время выполнения этого метода. Процесс, который в настоящее время выполняет **BroadcastReceiver**, т. е. выполняющийся в настоящее время код в методе обратного вызова **onReceive()**, как полагает система, является приоритетным процессом и будет сохранён, кроме случаев критического недостатка памяти в системе.

Когда программа возвращается из метода **onReceive()**, приёмник становится неактивным и система полагает, что работа объекта **BroadcastReceiver** закончена. Процесс с активным широковещательным получателем защищён от уничтожения системой. Однако процесс, содержащий неактивные компоненты, может быть уничтожен системой в любое время, когда память, которую он потребляет, будет необходима другим процессам.

Это представляет проблему, когда ответ на широковещательное сообщение занимает длительное время. Если метод **onReceive()**порождает отдельный поток, а затем возвращает управление, то полный процесс, включая и порожденный поток, система Android считает неактивным (если другие компоненты приложения не активны в процессе), и считает этот процесс кандидатом на уничтожение.

В частности, вы не можете отобразить диалог или осуществить связывание со службой внутри экземпляра **BroadcastReceiver**. Для первого случая необходимо вместо этого использовать методы класса **NotificationManager**. Во втором случае можно использовать вызов метода **Context.startService()**, чтобы послать команду для запуска службы.

Решение этой проблемы возможно, если запустить в методе **onReceive()** отдельную службу вместе с **BroadcastReceiver** и позволить службе выполнять задание, чтобы сохранить содержание процесса активным в течение всего времени вашей операции.

Приёмники системных событий

Android использует широковещательные сообщения для системных событий, таких как уровень зарядки батареи, сетевые подключения, входящие звонки, изменения часового пояса, состояние подключения данных, входящие сообщения SMS или обращения по телефону. Вы можете использовать эти сообщения, чтобы добавлять к вашим собственным проектам новые функциональные возможности, основанные на системных событиях.

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

* **ACTION\_BOOT\_COMPLETED** — передается один раз, когда устройство завершило свою загрузку. Требует разрешения **RECEIVE\_BOOT\_COMPLETED**
* **ACTION\_CAMERA\_BUTTON** — передается при нажатии пользователем клавиши **Camera**
* **ACTION\_DATE\_CHANGED** и **ACTION\_TIME\_CHANGED** - запускаются при изменении даты или времени на устройстве вручную пользователем
* **ACTION\_SCREEN\_OFF** и **ACTiON\_SCREEN\_ON** — передаются, когда экран выключается или включается
* **ACTION\_TIMEZONE\_CHANGED** — передается при изменении текущего часового пояса

Использование широковещательных сообщений

Широковещательные сообщения также используются для уведомления слушателей системных или прикладных событий. Широковещательные сообщения делают приложение более открытым; передавая события, использующие сообщения, вы открываете компоненты своего приложения для сторонних приложений, и сторонние разработчики реагируют на события без необходимости изменять ваше оригинальное приложение. В своём приложении вы можете прослушивать широковещательные сообщения других приложений, заменить или улучшить функциональность собственного (или стороннего) приложения или реагировать на системные изменения и события приложений.

Прослушивание событий приёмниками широковещательных сообщений

Чтобы создать приёмник широковещательных сообщений, его необходимо зарегистрировать либо в коде, либо в манифесте приложения при помощи фильтра намерений, чтобы определить, какие сообщения приёмник должен прослушивать. Для этого надо в элементе **<application>** добавить элемент **<receiver>**, определяющий имя класса приёмника широковещательных сообщений для его регистрации. Элемент **<receiver>** должен также включать фильтр намерений **<intent-filter>**, в котором нужно указать действие в виде строки.

Если регистрация была сделана через манифест, приложение не обязано работать, чтобы ваш приёмник среагировал на трансляцию намерения. Приложение запустится автоматически, когда подходящие намерение будет транслировано. Т.е. система сама сканирует содержимое манифеста всех приложений и делает за нас всю работу. Это хорошее решение, позволяющее экономить ресурсы. Такой подход позволяет создавать приложения, способные реагировать на события даже после завершения или принудительного завершения.

Чтобы создать новый приёмник широковещательных сообщений, необходимо создать класс, расширяющий базовый класс **BroadcastReceiver** и реализовать метод обратного вызова **onReceive()** обработчика событий.

public class PlayerReceiver extends BroadcastReceiver

{

private static final String TYPE = "type";

private static final int ID\_ACTION\_PLAY = 0;

private static final int ID\_ACTION\_STOP = 1;

@Override

public void onReceive(Context context, Intent intent)

{

int type = intent.getIntExtra(TYPE, ID\_ACTION\_STOP);

switch (type)

{

case ID\_ACTION\_PLAY:

// выполнение полученного намерения

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

break;

}

}

}

Метод **onReceive()** будет выполнен при получении широковещательного намерения, если полученное намерение соответствует фильтру. Приложения с зарегистрированными приёмниками широковещательных намерений будут запущены автоматически при получении соответствующего намерения. Метод должен быть завершён в течение пяти секунд, иначе появится диалоговое окно о принудительном закрытии.

Также можно зарегистрировать широковещательный приёмник не через манифест, а программно. Приёмник, зарегистрированный таким способом, будет отвечать на поступающие намерения только в том случае, если компонент приложения, которому он принадлежит, выполняется в этот момент.

Это может быть полезным, когда приёмник используется для обновления элементов пользовательского интерфейса внутри активности, запуска сервисов или уведомления через **NotificationManager**. В таких случаях вы можете отменять регистрацию широковещательного приёмника, если активность не отображается на экране или находится в неактивном состоянии.

В коде программы можете написать приблизительно такой код (обычно используют метод **onResume()**):

// Создаём и регистрируем широковещательный приёмник

IntentFilter filter = new IntentFilter(NEW\_CAT\_DETECTED);

CatDetectedBroadcastReceiver receiver = new CatDetectedBroadcastReceiver(receiver, filter);

Для отмены регистрации используется метод **unregisterReceiver()** в контексте приложения, передавая ему в качестве параметра экземпляр широковещательного приёмника (обычно в методе **onPause()**):

unregisterReceiver(receiver);

Трансляция упорядоченных намерений

Если важно, чтобы приёмники получали намерения в определённом порядке или могли влиять на транслируемое намерение, можно использовать метод **sendOrderedBroadcast()**:

String requiredPermission = "ru.alexanderklimov.MY\_BROADCAST\_PERMISSION";

sendOrderedBroadcast(intent, requiredPermission);

С помощью этого метода ваше намерение дойдёт до всех зарегистрированных приёмников, обладающих необходимым доступом (если он был указан), в порядке их приоритета. Приоритет задаётся с помощью атрибута **android:priority** внутри узла фильтра намерений в манифесте. Чем больше значение, тем выше приоритет.

Производить упорядоченные трансляции с использованием приоритетов рекомендуется только для тех приёмников, которым необходим конкретный порядок приёма сообщений.

Ограничения в Android 8.0 Oreo (API 26)

Часть широковещательных сообщений, задаваемых через манифест, запретили. Теперь нужно регистрировать их программно.

Broadcast (Широковещательные сообщения)

[Передача сообщений](http://developer.alexanderklimov.ru/android/broadcast.php#send)  
[Приёмники широковещательных сообщений](http://developer.alexanderklimov.ru/android/broadcast.php#receiver)  
[Периодическое срабатывание каждую минуту](http://developer.alexanderklimov.ru/android/broadcast.php#time_pick)  
[Автостарт Activity или Service при загрузке (перезагрузке) девайса](http://developer.alexanderklimov.ru/android/broadcast.php#boot)  
[Следим за питанием](http://developer.alexanderklimov.ru/android/broadcast.php#power)

В Android существует понятие широковещательных сообщений, которые можно отправлять или принимать. Оба процесса между собой не связаны и их можно использовать по отдельности.

Передача сообщений

Для начала научимся отправлять сообщения. В одном из уроков мы учились запускать другую активность с помощью намерения **Intent**. Но намерения можно использовать для отправки сообщений, предназначенные не какому-то отдельному приложению, объекту или компоненту, а всем. И любая программа, оборудованная специальным приёмником, может поймать это сообщение и предпринять свои шаги на основе полученной информации.

Для понимания, представьте, что радистка Кэт отправляет сообщение: "Срочно пришлите кота! Хочу быть сильной независимой женщиной. А ваш Штирлиц - фашист!". Возможно в этом сообщении содержится шифровка, что нужно прислать жену, столик для жены разведчика в кафе заказан, а Штирлиц - козёл! Но это не важно для нашего урока.

Любой человек, имеющий специальный оборудованный радиоприёмник, может принять это сообщение. Так же поступают и программы. Они обзаводятся приёмниками и прослушивают определённый тип сообщений.

Сообщения может создавать сама система, а также ваша программа и чужие программы.

Передача сообщений весьма проста в реализации. В вашем приложении необходимо создать сообщение, которое вы хотите передать. Установите при необходимости поля **action**, **data** и **category** (действие, данные и категорию) вашего сообщения и путь, который позволяет приёмникам широковещательных сообщений точно определять "своё" сообщение. В этом сообщении строка действия **ACTION** должна быть уникальной, чтобы идентифицировать передаваемое действие. В таких случаях создают строку-идентификатор действия по правилам именования пакетов Java. Например, для обнаружения кота в большом здании:

public static final String NEW\_CAT\_DETECTED = "ru.alexanderklimov.action.NEW\_CAT";

Далее вы создаёте объект **Intent**, загружаете в него нужную информацию и вызываете метод **sendBroadcast()**, передав ему в качестве параметра созданный объект **Intent**. Дополнительные данные можно использовать в **extras** как необязательные параметры.

Виртуальный код для обнаружения кота:

Intent intent = new Intent(NEW\_CAT\_DETECTED);

// Или так

// Intent intent = new Intent();

// intent.setAction(NEW\_CAT\_DETECTED);

intent.putExtra("catname", CatName);

intent.putExtra("longitude", currentLongitude);

intent.putExtra("latitude", currentLatitude);

sendBroadcast(intent);

В этом примере мы создали намерение с уникальной строкой, передали дополнительные данные (имя кота и его координаты), отправили сообщение. Другое приложение, связанное с картами, может принять сообщение и показать кота на карте.

Существуют также родственные методы **sendStickyBroadcast()** и **sendOrderedBroadcast()**.

Для старых устройств этого было вполне достаточно, но начиная с Android 3.0, в целях безопасности сообщения будут игнорироваться остановленными приложениями, чтобы они не запускались. Поэтому следует добавлять дополнительный флаг, разрешающий запуск активности.

intent.addFlags(Intent.FLAG\_INCLUDE\_STOPPED\_PACKAGES);

Мы напишем простой пример, который будет отправлять сообщения и также создадим приёмник для его получения. О приёмнике мы поговорим подробно во второй части урока. А пока получим первое представление о нём.

Создайте новый проект и разместите на экране кнопку с надписью "Отправить сообщение". Присвойте атрибуту **onClick** название метода, в котором будет происходит отправка широковещательного сообщения.

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Отправить сообщение"

android:id="@+id/button"

android:layout\_gravity="center\_horizontal"

android:onClick="sendMessage" />

В классе активности создаём уникальную строку и реализуем метод для щелчка кнопки. Также добавим дополнительные данные - первую часть послания радистки.

public static final String WHERE\_MY\_CAT\_ACTION = "ru.alexanderklimov.action.CAT";

public static final String ALARM\_MESSAGE = "Срочно пришлите кота!";

public void sendMessage(View view) {

Intent intent = new Intent();

intent.setAction(WHERE\_MY\_CAT\_ACTION);

intent.putExtra("ru.alexanderklimov.broadcast.Message", ALARM\_MESSAGE);

intent.addFlags(Intent.FLAG\_INCLUDE\_STOPPED\_PACKAGES);

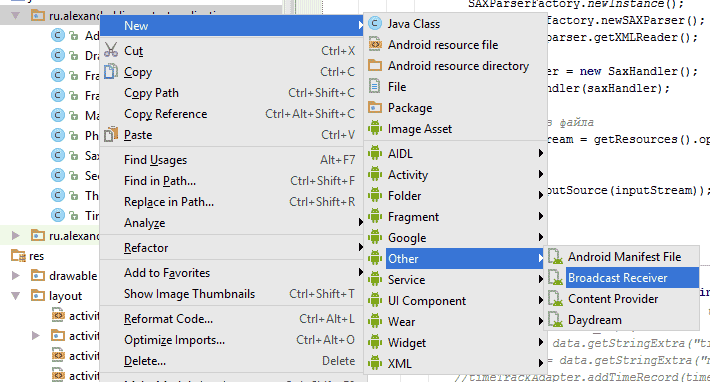
sendBroadcast(intent);

}

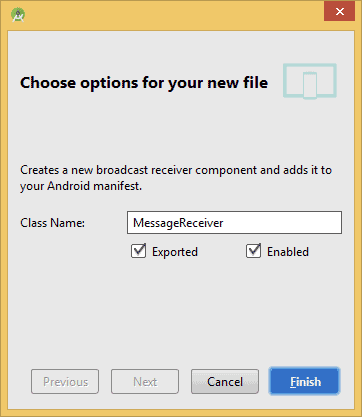
Запустив пример, вы можете нажать на кнопку и отправлять сообщение. Только ваше сообщение уйдёт в никуда, так как ни одно приложение не оборудовано приёмником для него. Исправим ситуацию и создадим приёмник в своём приложении. Мы будем сами принимать свои же сообщения.

Приёмник представляет собой обычный Java-класс на основе **BroadcastReceiver**. Вы можете создать вручную класс и наполнить его необходимыми методами. Раньше так и поступали. Но в студии есть готовый шаблон, который поможет немного сэкономить время.

Щёлкаем правой кнопкой мыши на названии пакета и выбираем **New | Other | Broadcast Receiver**



В диалоговом окне задаём имя приёмника, остальные настройки оставляем без изменений.



Студия создаст изменения в двух местах. Во-первых, будет создан класс **MessageReceiver**:

package ru.alexanderklimov.testapplication;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

public class MessageReceiver extends BroadcastReceiver {

public MessageReceiver() {

}

@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");

}

}

Во-вторых, в манифесте будет добавлен новый блок.

<receiver

android:name=".MessageReceiver"

android:enabled="true"

android:exported="true" >

</receiver>

В него следует добавить фильтр, по которому он будет ловить сообщения.

<receiver

android:name=".MessageReceiver"

android:enabled="true"

android:exported="true">

<intent-filter>

<action android:name="ru.alexanderklimov.action.CAT" />

</intent-filter>

</receiver>

Вернёмся в класс приёмника и модифицируем метод **onReceive()**.

@Override

public void onReceive(Context context, Intent intent) {

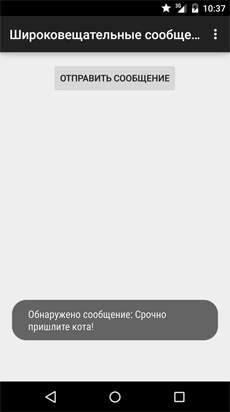
Toast.makeText(context, "Обнаружено сообщение: " +

intent.getStringExtra("ru.alexanderklimov.broadcast.Message"),

Toast.LENGTH\_LONG).show();

}

Снова запустим пример и ещё раз отправим сообщение. Так как наше приложение теперь оборудовано не только передатчиком, но и приёмником, то оно должно уловить сообщение и показать его нам.



Вы можете создать другое приложение с приёмником, чтобы одно приложение посылало сообщение, а другое принимало.

Приёмники широковещательных сообщений

Вот плавно мы перешли к приёмникам широковещательных сообщений. На самом деле вам не так часто придётся рассылать сообщения, гораздо чаще встречается потребность принимать сообщения. В первую очередь, сообщения от системы. Примерами таких сообщений могут быть:

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

Приёмник можно создать двумя способами - через манифест (мы использовали этот способ в примере) и программно через метод **registerReceiver()**. Между двумя способами есть существенная разница. Приёмник, заданный в манифесте, известен системе, которая сканирует файлы манифеста всех установленных приложений. Поэтому, даже если ваше приложение не запущено, оно всё равно сможет отреагировать на поступающее сообщение.

Приёмник, созданный программно, может работать только в том случае, когда активность вашего приложения активна. Казалось, это является недостатком и нет смысла использовать такой подход. Но всё не так просто. Некоторые системные сообщения могут обрабатываться только приёмниками, созданными программно. И в этом есть свой резон. Например, если ваше приложение не запущено, ему нет смысла принимать сообщения о заряде батареи. Иначе заряд батареи будет расходоваться ещё быстрее при лишней бесполезной работе. Информацию о заряде батареи ваше приложение может получить, когда в этом есть необходимость. Следует сверяться с документацией, какой вид приёмника следует использовать.

При программной регистрации приёмника мы можем также снять регистрацию, когда больше не нуждаемся в нём с помощью метода **unregisterBroadcastReceiver()**.

Периодическое срабатывание каждую минуту

Рассмотрим пример периодического срабатывания приёмника каждую минуту с помощью системного намерения **android.intent.action.TIME\_TICK**. Приёмник будет создан программно. Добавим на экран активности две кнопки для регистрации и отмены регистрации широковещательного сообщения.

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:onClick="registerBroadcastReceiver"

android:text="Регистрация" />

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:onClick="unregisterBroadcastReceiver"

android:text="Отмена" />

Создадим вручную новый класс **TimeBroadcastReceiver**, наследующий от **BroadcastReceiver**:

package ru.alexanderklimov.testapplication;

import android.content.BroadcastReceiver;

import android.content.Context;

import android.content.Intent;

import android.widget.Toast;

import java.text.Format;

import java.text.SimpleDateFormat;

import java.util.Date;

public class TimeBroadcastReceiver extends BroadcastReceiver {

public TimeBroadcastReceiver() {

}

@Override

public void onReceive(Context context, Intent intent) {

StringBuilder msgStr = new StringBuilder("Текущее время: ");

Format formatter = new SimpleDateFormat("hh:mm:ss a");

msgStr.append(formatter.format(new Date()));

Toast.makeText(context, msgStr, Toast.LENGTH\_SHORT).show();

}

}

Вы можете создать класс приёмника и через шаблон, как мы это сделали в предыдущем примере. Но в этом случае удалите запись о нём в манифесте, так как нам он не понадобится. Но если вы забудете сделать это, то ничего страшного не произойдёт, так как там не прописаны фильтры.

Откроем код главной активности и зарегистрируем (а также снимем регистрацию) приёмник:

package ru.alexanderklimov.testapplication;

import android.os.Bundle;

import android.app.Activity;

import android.content.IntentFilter;

import android.view.Menu;

import android.view.View;

import android.widget.Toast;

public class MainActivity extends Activity {

private TimeBroadcastReceiver mTimeBroadCastReceiver = new TimeBroadcastReceiver();

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

}

// регистрируем широковещательный приёмник

// для намерения "android.intent.action.TIME\_TICK".

// Данное намерение срабатывает каждую минуту

public void registerBroadcastReceiver(View view) {

this.registerReceiver(mTimeBroadCastReceiver, new IntentFilter(

"android.intent.action.TIME\_TICK"));

Toast.makeText(getApplicationContext(), "Приёмник включен",

Toast.LENGTH\_SHORT).show();

}

// Отменяем регистрацию

public void unregisterBroadcastReceiver(View view) {

this.unregisterReceiver(mTimeBroadCastReceiver);

Toast.makeText(getApplicationContext(), "Приёмник выключён", Toast.LENGTH\_SHORT)

.show();

}

}

Запускаем проект и нажимаем на первую кнопку, чтобы включить рассылку широковещательного сообщения. Теперь каждую минуту будет срабатывать запуск всплывающего сообщения с текущим временем. Даже если вы переключитесь на другое приложение, то всё равно будете видеть сообщения.

Это один из примеров, когда приёмник следует регистрировать программно. Я видел часто на форумах вопросы, почему не работает данное намерение **android.intent.action.TIME\_TICK**. А не надо было его регистрировать в манифесте.

В нашем примере мы устанавливали и снимали регистрацию через нажатия кнопок. Обычно включают регистрацию в методе **onResume()**, а снимают регистрацию в методе **onPause()**.

Необходимо помнить, что программная регистрация широковещательного сообщения создаётся в основном потоке приложения и это может послужить источником ошибок, если операции в **BroadcastReceiver** занимают длительное время. Как вариант, используйте сервисы. Почитайте на эту тему [статью (en)](http://gmariotti.blogspot.ru/2013/02/antipattern-freezing-ui-with-broadcast.html).

Автостарт Activity или Service при загрузке (перезагрузке) девайса

Ещё один полезный пример, который часто используется приложениями.

Если ваше приложение (сервис) должно запускаться сразу после перезагрузки устройства, то используйте намерение **android.intent.action.BOOT\_COMPLETED**:

public class BootReceiver extends BroadcastReceiver {

Context mContext;

private final String BOOT\_ACTION = "android.intent.action.BOOT\_COMPLETED";

@Override

public void onReceive(Context context, Intent intent) {

mContext = context;

String action = intent.getAction();

if (action.equalsIgnoreCase(BOOT\_ACTION)) {

// здесь ваш код

// например, запускаем уведомление

Intent intent = new Intent(context, ru.alexanderklimov.NotifyService.NotifyService.class);

context.startService(intent);

// в общем виде

//для Activity

Intent activivtyIntent = new Intent(context, MyActivity.class);

activivtyIntent.addFlags(Intent.FLAG\_ACTIVITY\_NEW\_TASK);

context.startActivity(activivtyIntent);

//для Service

Intent serviceIntent = new Intent(context, MyService.class);

context.startService(serviceIntent);

}

}

}

Мы создали отдельный класс для широковещательного сообщения. Также нужно создать разрешение и зарегистрировать приёмник в манифесте.

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

<receiver android:enabled="true" android:name=".BootReceiver"

android:permission="android.permission.RECEIVE\_BOOT\_COMPLETED">

<intent-filter>

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

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

</receiver>

Смотри также [Автозапуск приложения при загрузке](http://developer.alexanderklimov.ru/android/theory/boot.php)

Следим за питанием

Нет, речь пойдёт не о правильном питании кота. Имеется в виду питание от электричества. Если ваше устройство отключить от зарядки, то система оповещает об этом событии через широковещательное намерение **android.intent.action.ACTION\_POWER\_DISCONNECTED**.

Не станем заводить новый приёмник, а откроем манифест и добавим дополнительный фильтр к приёмнику сообщений от радистки Кэт.

<receiver

android:name=".MessageReceiver"

android:enabled="true"

android:exported="true">

<intent-filter>

<action android:name="ru.alexanderklimov.action.CAT" />

</intent-filter>

**<intent-filter>**

**<action android:name="android.intent.action.ACTION\_POWER\_DISCONNECTED"></action>**

**</intent-filter>**

</receiver>

А в классе **MessageReceiver** добавим код для метода.

@Override

public void onReceive(Context context, Intent intent) {

//Toast.makeText(context, "Обнаружено сообщение: " +

// intent.getStringExtra("ru.alexanderklimov.broadcast.Message"),

// Toast.LENGTH\_LONG).show();

if (intent.getAction().equalsIgnoreCase("android.intent.action.ACTION\_POWER\_DISCONNECTED")) {

String message = "Обнаружено сообщение "

+ intent.getAction();

Toast.makeText(context, message,

Toast.LENGTH\_LONG).show();

}

}

Пример нужно проверять на реальном устройстве. Подключите устройство к питанию, а затем выдерните кабель. На экране появится сообщение.