RxJava (http://developer.alexanderklimov.ru/android/rx/)

Советы (http://developer.alexanderklimov.ru/android/tips-android.php)

Статьи (http://developer.alexanderklimov.ru/android/articles-android.php)

Книги (http://developer.alexanderklimov.ru/android/books.php)

Java (http://developer.alexanderklimov.ru/android/java/java.php)

Kotlin (http://developer.alexanderklimov.ru/android/kotlin/)

Дизайн (http://developer.alexanderklimov.ru/android/design/)

Отладка (http://developer.alexanderklimov.ru/android/debug/)

Open Source (http://developer.alexanderklimov.ru/android/opensource.php)

Полезные ресурсы (http://developer.alexanderklimov.ru/android/links.php)

Broadcast (Широковещательные сообщения)

Передача сообщений
Приёмники широковещательных сообщений
Периодическое срабатывание каждую минуту
Автостарт Activity или Service при загрузке (перезагрузке) девайса
Следим за питанием

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.

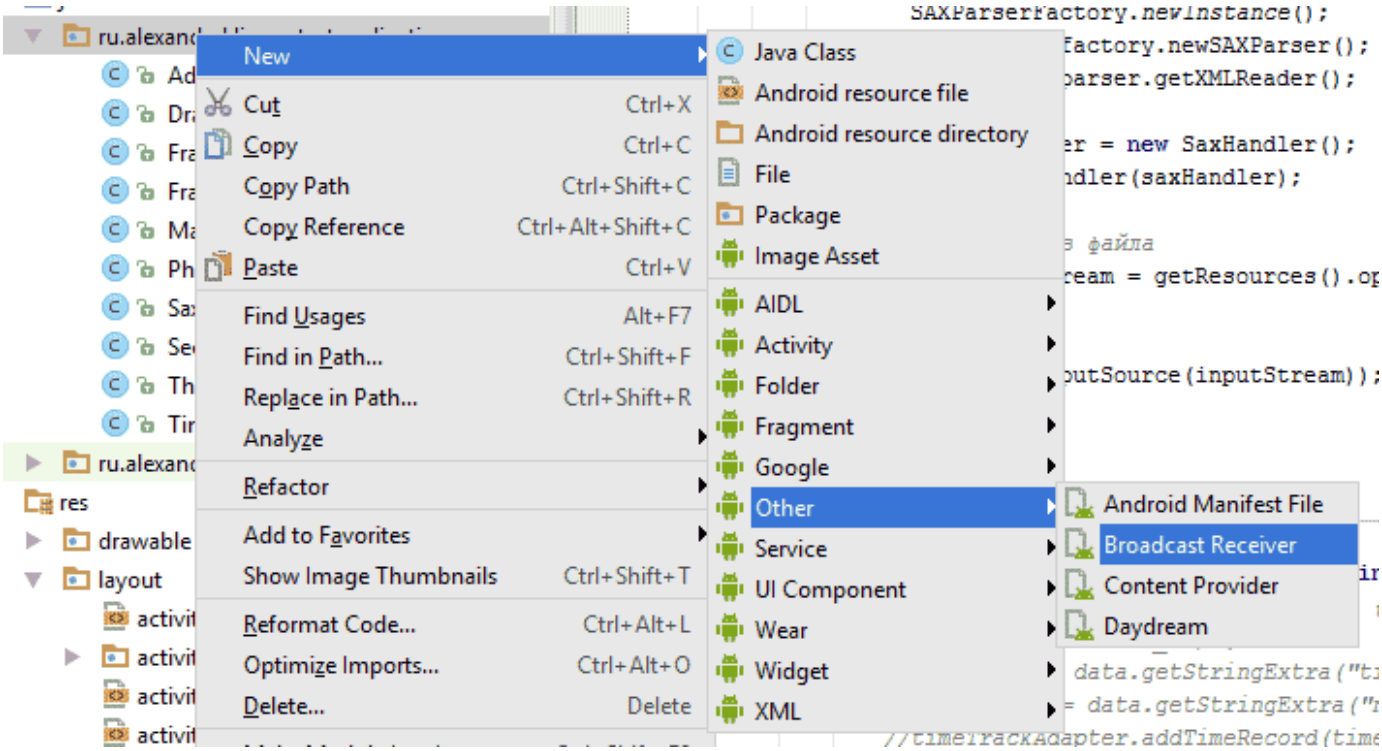
```
public static final String WHERE_MY_CAT_ACTION = "ru.alexanderklimov.action.CAT";
public static final String ALARM_MESSAGE = "Срочно пришлите кота!";

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);
}
```

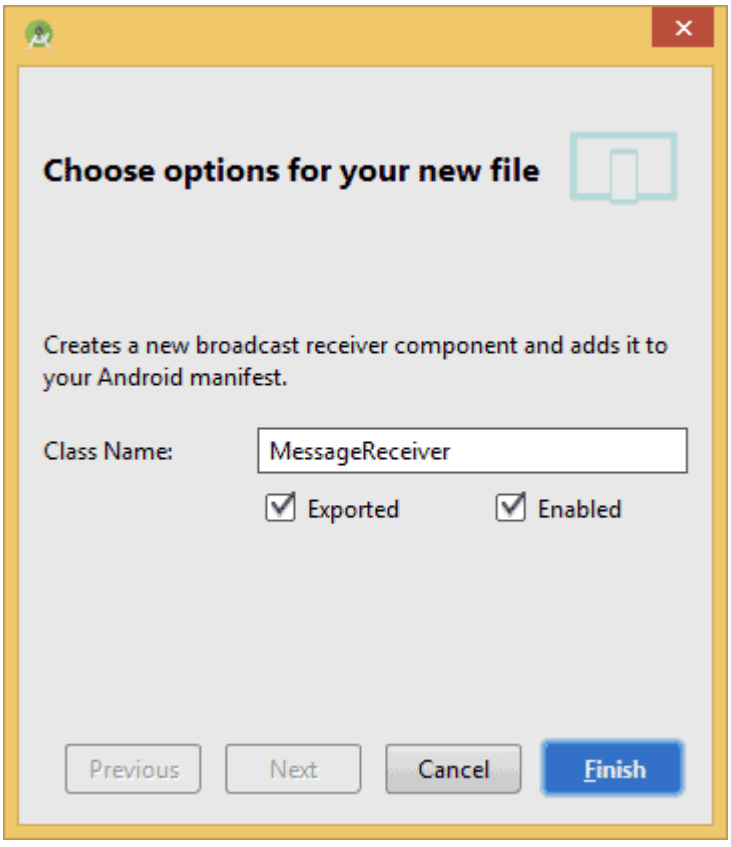
Запустив пример, вы можете нажать на кнопку и отправлять сообщение. Только ваше сообщение уйдёт в никуда, так как ни одно приложение не оборудовано приёмником для него. Исправим ситуацию и создадим приёмник в своём приложении. Мы будем сами принимать свои же сообщения.

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>
```

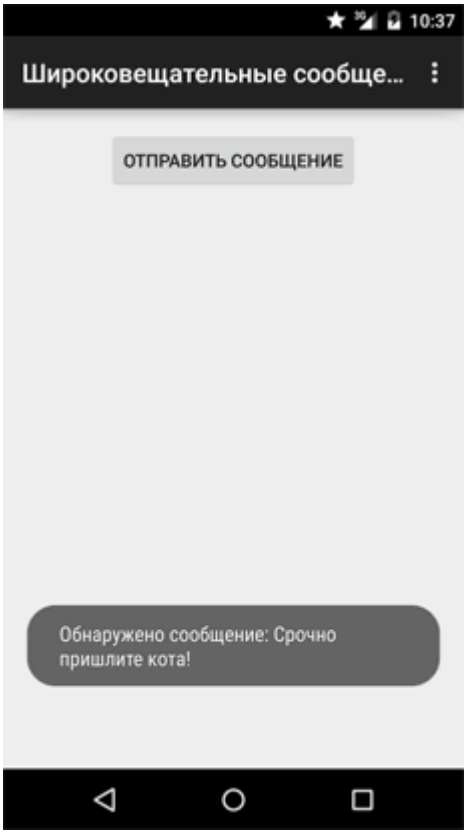
В него следует добавить фильтр, по которому он будет ловить сообщения.

```
<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.NotifyServic
e.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_DISCONNECTE
D")) {
        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.

Другие примеры

Теория (http://developer.alexanderklimov.ru/android/theory/broadcast.php)

Получаем показания батареи в реальном времени (http://developer.alexanderklimov.ru/android/battery.php#getlevel)

Отслеживание состояния соединения Wi-Fi (http://developer.alexanderklimov.ru/android/wifi.php#statechange)

DownloadManager (http://developer.alexanderklimov.ru/android/downloadmanager.php)

Секретный код (http://developer.alexanderklimov.ru/android/secretcode.php)

Реклама