Работа с широковещательными сообщениями

Цель работы

Познакомится с сервисами Android Studio и научиться создавать приложения принимающие и отправляющие сообщения.

Задания для выполнения

1. Создайте приложение, которое будет отправлять и принимать сообщения.

Методические указания

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.

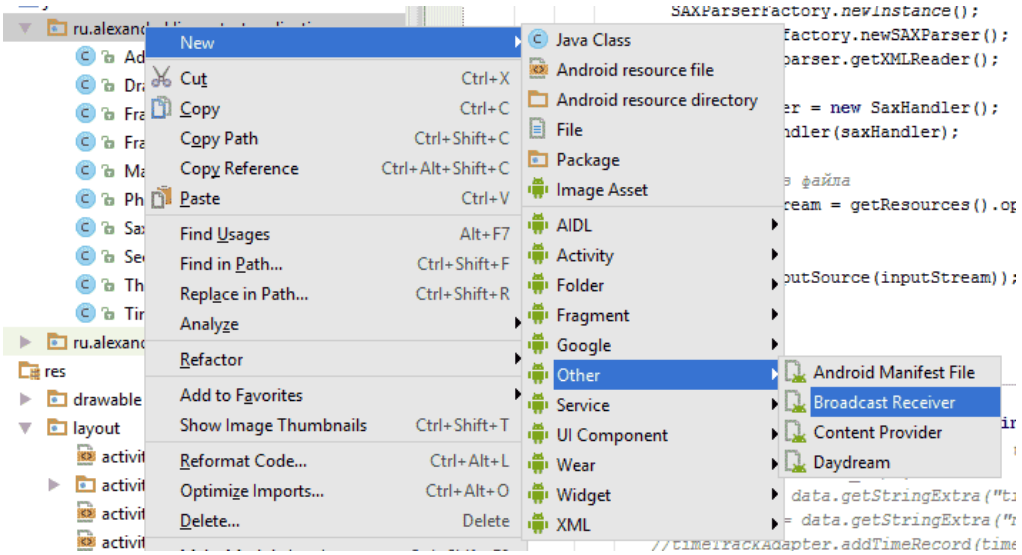
```
public static final String WHERE_MY_CAT_ACTION = "ru.alexanderklimov.action.CAT";
public static final String ALARM_MESSAGE = "Срочно пришлите кота!";

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.

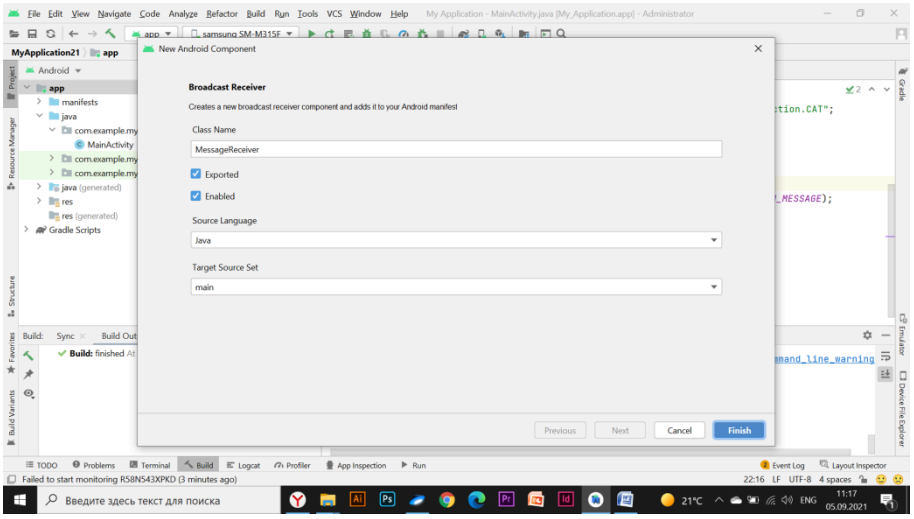
Приёмник представляет собой обычный Java-класс на основе BroadcastReceiver. Вы можете создать вручную класс и наполнить

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>
```

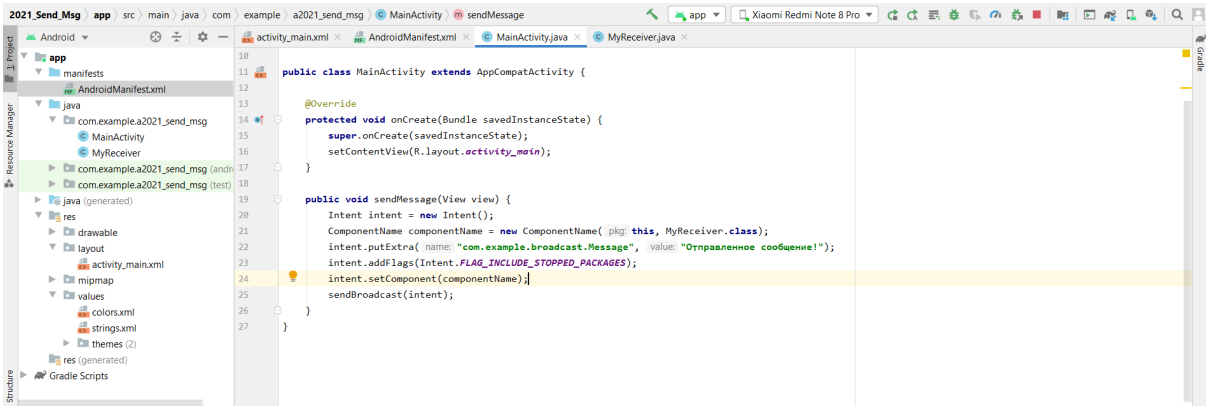
Вернёмся в класс приёмника и модифицируем метод onReceive().

```
@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.

Если приложение работает не корректно, необходимо реализовать следующие изменения.

Когда приложение запускается в фоновом режиме, оно потребляет некоторые ограниченные ресурсы устройства, такие как оперативная память. Это может привести к ухудшению работы пользователя, особенно если пользователь использует ресурсоемкое приложение, такое как игра или просмотр видео. Чтобы улучшить пользовательский интерфейс, Android 8.0 (уровень API 26) накладывает ограничения на то, что приложения могут делать во время работы в фоновом режиме. В этом скриншоте приводятся изменения в коде приложения, чтобы оно хорошо работало в соответствии с новыми ограничениями.



Материалы

- http://developer.alexanderklimov.ru/android/notification.php
- http://developer.alexanderklimov.ru/android/toast.php
- http://developer.alexanderklimov.ru/android/theory/services-theory.php
- https://metanit.com/java/android/11.2.php
- http://developer.alexanderklimov.ru/android/broadcast.php
- https://developer.android.com/about/versions/oreo/background.html