Broadcast (Широковещательные сообщения)

Практическая часть показана в отдельной статье.

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.