Широковещательные рассылки

Приложения Android могут отправлять и получать широковещательные рассылки (*англ.* broadcast) от операционной системы и от других приложений. Широковещательные рассылки отправляются, когда происходит какое-либо событие, представляющее интерес для других приложений.

Android отправляет широковещательные рассылки при возникновении различных системных событий, например, когда система переключается в режим полета и выходит из него, когда начинается зарядка и т.д. Системные рассылки отправляются всем приложениям, которые подписаны на получение события.

Широковещательные рассылки можно использовать в качестве системы обмена сообщениями между приложениями, однако система оптимизирует доставку широковещательных рассылок, поэтому сроки доставки не гарантируются. Если задержка доставки критична для приложения, то лучше использовать другие методы связи, например, привязываемые сервисы.

Прием рассылок

Приложения могут подписываться на получение определенных широковещательных рассылок: когда подходящая рассылка отправляется, система автоматически направит его в приложения, которые подписались на получение этого конкретного типа рассылок. Есть два варианта подписки: через манифест приложения, или динамически в коде приложения.

Подписка через манифест приложениям

Этот вариант не всегда подходит, потому что большая часть неявных рассылок (не предназначенных конкретному приложению) больше не может быть получена при таком варианте подписке. Список неявных рассылок, которые всё же можно принимать, приведён в документации (*англ.*) Однако, рассылки, направленные конкретному приложению, доходят корректно.

Чтобы подписаться на рассылку, нужно объявить в манифесте тег `receiver`:

```
<receiver android:name=".MyBroadcastReceiver" android:exported="true">
    <intent-filter>
        <action android:name="com.example.broadcast.MY_NOTIFICATION" />
    </intent-filter>
</receiver>
```

Здесь `name` – это имя класса, который будет принимать рассылки; если должны приниматься рассылки от системы и других приложений, атрибут `exported` должен быть установлен в `true`.

Фильтры намерений (в теге `intent-filter`) перечисляют те рассылки, которые приложение желает принимать.

Класс, который будет принимать рассылки, должен быть унаследован от класса `BroadcastReceiver` и должен реализовывать метод `onReceive`, который будет вызываться при получении широковещательной рассылки. Пример такого класса:

```
class MyBroadcastReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
        Log.d("My Broadcast Receiver", "Получена рассылка ${intent.action}")
    }
}
```

Система регистрирует приемник при установке приложения. Приемник становится отдельной точкой входа в приложение, то есть система может запустить приложение и доставить рассылку, если приложение в данный момент не запущено. Однако в этом случае главная активность не создаётся, поэтому не получится передать ей какую-либо информацию или указания.

Если многие приложения через манифест подписались на одну и ту же широковещательную рассылку, это может привести к массовому запуску, что существенно повлияет на производительность устройства. Поэтому рекомендуется по возможности подписываться на рассылку в коде приложения, тогда рассылка будет получена только во время его работы. Некоторые широковещательные рассылки, например, `CONNECTIVITY_ACTION`, могут быть приняты только приемниками, которые зарегистрированы в коде.

Поскольку метод `onReceive()` выполняется в основном потоке, он должен выполняться и возвращать результат быстро.

Подписка в коде приложения

Чтобы зарегистрировать приемник в коде, нужно:

1. Создать экземпляр класса, унаследованного от `BroadcastReceiver`, как, например, `MyBroadcastReceiver` из примера выше:

   ```
   val br = MyBroadcastReceiver()
   ```

2. Создать объект `IntentFilter` с указанием *строки действия* рассылки, которая будет приниматься приемником:

   ```
   val filter = IntentFilter(SOME_BROADCAST)
   ```

3. Зарегистрировать приемник рассылок с помощью метода `registerReceiver`:

   ```
   ContextCompat.registerReceiver(this, br, filter, ContextCompat.RECEIVER_EXPORTED)
   ```

Здесь `RECEIVER_EXPORTED` – это аналог атрибута `exported` из манифеста, чтобы можно было принимать рассылки от системы и других приложений. Если конкретный приемник должен принимать только рассылки, отправленные самим приложением, используется флаг `RECEIVER_NOT_EXPORTED`.

Для прекращения работы приемника нужно вызвать метод `unregisterReceiver`:

```
unregisterReceiver(br)
```

Регистрировать приемник и удалять регистрацию следует в парных событиях: например, `onCreate` и `onDestroy`, или `onResume` и `onPause`. Если нарушить это правило и регистрировать приемник, например, в `onResume`, а удалять регистрацию в `onDestroy`, то приемник может быть зарегистрирован много раз, а разрегистрирован только один раз, что может приводить к утечкам ресурсов или повышенным системным расходам ресурсов.

Отправка рассылок

Приложения могут отправить рассылку с помощью двух методов:

- `sendBroadcast()` отправляет рассылку всем приемникам, но порядок приемников не определен. Приемники не могут узнать результаты работы других приемников или прервать рассылку.
- `sendOrderedBroadcast()` отправляет рассылку приемникам по очереди. Каждый приемник может передать результат следующему приемнику, или же прервать рассылку. Порядок приемников задаётся атрибутом `priority` в фильтре намерений, если значение одинаковое, то порядок может оказаться любым.

Для отправки рассылки нужно создать намерение и отправить его:

```
val intent = Intent()
intent.action = "com.example.broadcast.MY_NOTIFICATION"
intent.putExtra("info", "Какая-то информация")
sendBroadcast(intent)
```

Строка действия (action) должна быть сформирована в стиле имени пакета приложения и однозначно идентифицировать широковещательное событие. Пространство имен для действий широковещательных рассылок является глобальным, и если не использовать имя пакета, то можно случайно получить рассылку от другого приложения, которое выбрало аналогичное имя для действия.

К рассылке можно добавить дополнительную информацию с помощью метода `putExtra()`. Также можете ограничить приемники рассылки одним приложением, для этого нужно указать имя пакета приложения в свойстве `package`.