4.7. Широковещательные сообщения (Broadcast). Создание трансляций.

Сайт:	Samsung Innovation Campus	Напечатано::	Murad Rezvan
Курс:	Мобильная разработка на Kotlin	Дата:	понедельник, 3 июня 2024, 17:56
Книга:	4.7. Широковещательные сообщения (Broadcast). Создание трансляций.		

Оглавление

1. Широковещательные трансляции

Наряду с регистрацией трансляций мобильные приложения могут посылать сообщения в информационное пространство системы, где подписанные на эти посылки приложения их читают. Отсылка может быть широковещательной в полном смысле, то есть адресованной всем, кто подписан; иметь ограничения на получателей; а может осуществляться внутри приложения. В любом случае сообщение запаковывается в намерение, и уже это намерение транслируется подписчикам. В конструктор при создании намерения передаётся имя трансляции. Именем может быть любая уникальная строковая константа, идентифицирующая передаваемую трансляцию. Часто имя трансляции начинают с имени пакета приложения, делающего рассылку

```
val intent = Intent("ru.myitacademy.samsung.postmaster.Letter")
```

но это не обязательно. В упражнении 4.6.3 трансляция имеет имя "Козьма_Прутков" и не имеет никакого отношения к имени пакета приложения. Однако, вероятность того, что в системе встретится ещё одна рассылка с подобным именем, очень мала, что соответствует условию предотвращения коллизии.

При необходимости в намерение добавляются передаваемые данные

```
intent.putExtra("text","Здравствуйте, дорогие читатели!")
```

Операционная система Android для каждого случая имеет свои механизмы адресации трансляций в зависимости от её типа.

- *Неявная трансляция.* Отправляется в общее пространство сообщений системы. Доступна системным процессам и приложениям, имеющим приёмник, зарегистрированный на её получение. Подписку может оформить любой процесс, которому известно имя трансляции.
- *Упорядоченная трансляция.* Отправляется конкретному приёмнику, имеющему соответственные разрешения. Если подписчиков на трансляцию более одного, то отправка происходит по очереди в порядке приоритетности. Во время упорядоченной отправки сообщений приёмники могут изменять передаваемые данные, а так же могут в любой момент прервать трансляцию.
- *Локальная трансляция.* Используется в случае, когда отправитель и приёмник находятся в одном приложении. Сохраняет конфиденциальность пересылаемой информации в контексте одного приложения и не выходит за рамки своего процесса, тем самым оптимизируя расход ресурсов устройства.

> С версии 1.1.0-alpha01 класс androidx.localbroadcastmanager.content.LocalBroadcastManager исключён из инструментов разработки как устаревший с рекомендациями от Google об использовании LiveData для решения задачи локальных рассылок.

1.1. Отправка сообщения всем подписчикам

Неявная трансляция используется для отправки сообщения в систему и доступна всем, кто подписан на данную рассылку. В этом случае управление трансляцией полностью принадлежит запускающему приложению. Приёмники широковещательной трансляции могут только регистрировать сообщения и не имеют возможности остановить трансляцию или отправить ответную реакцию за пределы своего контекста. Механизмом отправки неявных широковещательных трансляций выступает метод активности sendBroadcast(), получающий в качестве обязательного аргумента некоторое неявное намерение.

```
sendBroadcast(intent)
```

Можно объявлять намерение непосредственно в вызове метода

```
        sendBroadcast(Intent().also { intent ->
            intent.action = "ru.myitacademy.samsung.postmaster.Letter"
            intent.putExtra("text","Здравствуйте, дорогие читатели!")
        })
```

В качестве необязательного аргумента метода может быть передан список разрешений, которыми должен обладать приёмник для получения данной трансляции.

```
sendBroadcast(intent,android.Manifest.permission.WRITE_VOICEMAIL)
```

В таком случае рассылку смогут получать только приёмники, зарегистрированные в приложении с заданными разрешениями в манифесте. Подробнее о запуске трансляций по разрешениям можно ознакомиться в документации.

Метод `sendBroadcast()` является асинхронным фоновым процессом и срабатывает при запуске подписанных на его трансляцию приёмников.

1.2. Упражнение 4.7.1 Рассылка широковещательной трансляции

Сделаем приложение, делающее рекламные рассылки. Текст рекламы будет вводиться перед рассылкой, отправка трансляции будет выполняться по нажатию на кнопку.

1. Создайте новый проект с именем *Advertising*, в качестве корневой разметки активности установите вертикальную линейную разметку и поместите в нее элементы (подписи записаны в файле res/values/string.xml)

- текстовое поле с подписью "Введите текст рекламы" (константа adv_text);
- текстовое поле для ввода текста;
- кнопку с надписью "Рекламировать всем" (константа allbutton).

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:gravity="center"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/adv_text"/>

    <EditText
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/advtxt"/>

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/allbtn"
        android:text="@string/allbutton"/>

</LinearLayout>
```

У поля ввода и кнопки установите атрибут `id`, поскольку в коде к ним будет обращение.

2. В классе активности найдите кнопку методом `findViewById()` и установите на неё обработчик нажатия

```
  findViewById<Button>(R.id.allbtn).setOnClickListener {
    }
```

В обработчик поместите создание и настройку намерения с именем *"ru.myitacademy.samsung.advertising.POST"* и данными, считанными из текстового поля ввода

```
val intent = Intent().also { intent ->
            intent.action = "ru.myitacademy.samsung.advertising.POST"
            intent.putExtra("hot_news", findViewById<EditText>(R.id.advtxt).text.toString())
        }
```

и сразу после создания запустите трансляцию

```
sendBroadcast(intent)
```

3. Для того, чтобы рекламу кто-то читал, создайте приложение с двумя приёмниками, читающими трансляцию "ru.myitacademy.samsung.advertising.POST" и отображающими тосты с сообщением в разных областях экрана

```
class MyReceiver: BroadcastReceiver() {
    override fun onReceive(p0: Context?, p1: Intent) {
       Toast.makeText(p0,p1.getStringExtra("hot_news"),Toast.LENGTH_SHORT).show()
    }
}

class MyReceiver2: BroadcastReceiver() {
    override fun onReceive(p0: Context?, p1: Intent) {
       val t = Toast.makeText(p0,p1.getStringExtra("hot_news"),Toast.LENGTH_SHORT)
           t.setGravity(Gravity.TOP,0,0)
           t.show()
    }
}
```

и зарегистрируйте оба приёмника в методе onCreate()

```
registerReceiver(MyReceiver(),
            IntentFilter("ru.myitacademy.samsung.advertising.POST")
        )
registerReceiver(MyReceiver2(),
            IntentFilter("ru.myitacademy.samsung.advertising.POST")
        )
```

4. Запустите сначала приложение с приёмниками, а затем приложение Advertising. Можно добавить подписку на "ru.myitacademy.samsung.advertising.POST" в приложении Subscriber и запустить оба приложения с приёмниками, чтобы проследить асинхронность получения трансляции.

Новые приёмники можно посмотреть здесь

Приложение Advertising будет изменяться в следующих упражнениях.

1.3. Упорядоченные трансляции

При упорядоченной трансляции сообщение передаётся всем подписчикам по очереди в порядке приоритета.

Приоритет приёмника устанавливается в свойстве `priority` фильтра намерения (`android:priority` при статической регистрации) и представляется целым числом из отрезка [-1000; 1000]. Значением по умолчанию является 0.

```
val intentFilter =  IntentFilter("ru.myitacademy.samsung.postmaster.Letter")
        intentFilter.priority = -100
```

Первыми получают трансляцию приёмники с наибольшим значением поля `priority`. Если у нескольких получателей одинаковый уровень приоритета, то порядок передачи сообщения между ними определяется случайным образом.

Все подписчики имеют доступ к изменению данных, передаваемых в сообщении и их дальнейшей передаче, а так же имеют возможность прерывания трансляции. Это значит, что приёмники с приоритетом ниже, чем у остановившего трансляцию, сообщение не получат.

Если при трансляции не планируется возврата данных обратно в широковещательный канал, для запуска вещания используют метод sendOrderedBroadcast(Intrnt, String?) с двумя аргументами.

- Первый аргумент - обязательно непустое намерение, содержащее саму трансляцию;
- второй аргумент - разрешения, требующиеся от приёмников для чтения сообщения. Если имеет значение `null`, то читать трансляцию могут все подписчики в порядке приоритета.

1.4. Упражнение 4.7.2 Упорядочение рассылок

Добавим в проект Advertising возможность упорядоченной рассылки.

1. В обработчике кнопки рассылки замените вызов `sendBroadcast(intent)` на вызов упорядоченной трансляции

```
findViewById<Button>(R.id.allbtn).setOnClickListener {
            myIntent.putExtra("hot_news", findViewById<EditText>(R.id.advtxt).text.toString())
            //sendBroadcast(intent)
            sendOrderedBroadcast(myIntent,null)}
    }
```

Разрешения устанавливать не нужно, по этому второй аргумент примет значение `null`.

Проведите тестирование работы приложения на заготовленных в упражнении 4.7.1 приёмниках. Ничего не изменилось, поскольку у приёмников не установлен приоритет.

2. В приложении Tests первому приёмнику установите приоритет -1, второму 1.

```
val if1 =  IntentFilter("ru.myitacademy.samsung.advertising.POST")
           if1.priority = -1
            registerReceiver(MyReceiver(),if1)

            registerReceiver(MyReceiver2(),
                             IntentFilter("ru.myitacademy.samsung.advertising.POST").apply { priority = 1 }
            )
```

В приложении Subscriber приоритет устанавливать не нужно.

Subscriber работает между приёмниками Tests, поскольку у него приоритет по умолчанию, который равен 0. Таким образом, порядок приоритетов такой: 1, 0, -1.

Если во втором параметре метода `sendOrderedBroadcast()` передать разрешения, то получать рассылку будут только те приёмники, у которых имеются данные разрешения. Реализуем выборочную рассылку.

3. Добавьте в разметку рассылающего приложения ещё одну кнопку

```
 <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/selectbtn"
        android:text="@string/selectbtn"
        android:onClick="selectPost"/>
```

В активности определите метод `selectPost()`, являющийся обработчиком нажатия новой кнопки

```
fun selectPost(view: View){
    sendOrderedBroadcast(myIntent,Manifest.permission.INTERNET)
}
```

То есть по этой кнопке получать рассылку будут только те приёмники, у которых подключено разрешение на использование сети интернет.

Сообщения получает только Subscriber, поскольку только у него есть разрешение на выход в интернет.

1.5. Изменение данных в трансляции

Упорядоченная трансляция позволяет приёмникам управлять жизненным циклом трансляции и изменять передаваемые в ней данные. Если один из получателей остановит трансляцию, то приёмники с более низким приоритетом её уже не получат и данные будут возвращены отправителю.
Если цепочка трансляций не будет оборвана, то все приёмники получат возможность не только прочитать сообщение, но и изменить передаваемые в трансляции данные, в любом случае, результат будет возвращён в посылающее приложение.

Это значит, что у источника широковещательной трансляции тоже должен быть свой приёмник. В методе `onReceive()` каждого приёмника, участвующего в упорядоченной трансляции, доступны все значения полей трансляции и методы их получения и изменения:

- resultCode. Код результата. Передаётся приёмнику из предыдущего участника и передаётся дальше в методе `setResult()`.
- resultData. Передаваемые по трансляции данные. Передаются в виде непустой текстовой строки, могут быть преобразованы в процессе трансляции. Передаётся в следующий приёмник методом `setResult()`.
- ResultExtras. Данные трансляции, извлекаемые методом getResultExstras(Boolean). Логический параметр указывает, нужно ли добавлять данные в коллекцию передаваемых данных. Если false, то данные не будут передаваться в следующий приёмник. Передача производится в методе `setResult()`. Данные из `ResultExtras` получаются через геттеры в зависимости от типа размещаемых данных:

```
// у отправителя
val extras = Bundle()
  extras.putInt("putInt",45)
  extras.putString("putStr","myStr")


//у получателя
var res = getResultExtras(true)
var i = res.getInt("putInt",-5)
var str = res.getString("putStr")
```

После получения параметров результата из предыдущего узла трансляции, приёмник в рамках работы метода `onReseive()` может внести изменения в эти данные и передать их дальше.

- Все три свойства результата передаются по трансляции методом setResult(Int, String, Bundle)
- Остановить трансляцию может любой приёмник, участвующий в передаче, вызовом метода abortBroadcast().

Трансляция с возвращаемыми данными посылается перегрузкой метода sendOrderedBroadcast(Intent, String?, BroadcastReceiver?, Handler?, int, String?, Bundle?)

Обязательным параметром является первый - намерение, несущее сообщение. Все остальные аргументы могут принимать значение `null`.

Параметр	Назначение
intent: Intent	Обязательно непустое намерение, несущее данные трансляции
permission: String?	Разрешения, которые должны иметь приёмники, чтобы иметь доступ к трансляции. Если null, то трансляцию получают все подписчики в приоритетном порядке.
receiver : BroadcastReceiver?	Приёмник "обратной связи". После завершения трансляции получает преобразованные данные трансляции. Если не установлен, то данные в транслятор не возвращаются.
handler: Handler?	Организатор работы результирующего приёмника. Если null, то приёмник работает в главном потоке.
resultCode: Int	Начальное значение кода результата трансляции. Может принимать любое целочисленное значение, в том числе константы Activity.RESULT_OK (-1), Activity.RESULT_CANCEL (0) или Activity.RESULT_FIRST_USER (1). Изменяется приёмниками и передаётся дальше по трансляции. Финальное значение возвращается в приёмник отправителя.
initStr: String?	Начальное значение передаваемых по трансляции текстовых данных. Как любое начальное значение может принимать значение null.
data: Bundle?	Начальное состояние результата трансляции, может быть любым объектом. Применяется при передаче несериализованных объектов. Может быть пустым.

1.6. Упражнение 4.7.3 Передача данных по трансляции

Внесём изменения в Advertising для организации передачи данных по трансляции.

1. Добавьте в разметку активности ещё одну кнопку с надписью "Рассылка с обратной связью" и обработчиком нажатия `backPost()` для запуска цепочки трансляций.

```
<Button
       android:layout_width="wrap_content"
       android:layout_height="wrap_content"
       android:id="@+id/tbtn"
       android:text="@string/btn"
       android:onClick="backPost"/>
```

2. В классе активности определите метод `backPost()`, запускающий упорядоченную трансляцию с возвратом значения.

```
 fun backPost(view:View){
     sendOrderedBroadcast(myIntent,  //намерение остаётся прежнее
                                         null,          //разрешений на прослушивание трансляции не требуется
                                         MyReceiver(), // приёмник для получения данных обратно
                                         null,  //запускается приёмник в главном потоке
                                         0,  //начальное значение resultCode = 0,
  // в нём будет происходить подсчёт выполненных трансляций
                                        "NewPost", // строка - данные трансляции
                                         Bundle().apply{    // передаваемые по трансляции данные
  //начальное значение считывается из текстового поля ввода на активности
                                                             putString("strEx",findViewById<EditText>
(R.id.advtxt).text.toString())})
 }
```

3. Создайте в приложении приёмник для получения результата из трансляции.

```
class MyReceiver : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
     }
}
```

Данные по трансляции будут передаваться в объекте resultExtras, по этому создайте переменную `res: Bundle` и получите в неё объект данных.

```
 var res = getResultExtras(true)
```

Теперь можно получить итоговое значение всей цепочки трансляции:

```
 var str = res.getString("strEx")
```

и поместить её в Toast на экране, выводя так же количество выполненных пересылок, которые накапливаются в параметре resultCode:

```
Toast.makeText(context, str + " \n" +
                "сделано" + resultCode + "пересылок", Toast.LENGTH_LONG).show()
        Log.d("Control_toast",str)
```

Рассылка готова, осталось создать участников трансляции.

4. Запустите новый проект и опишите в нём три приёмника

```
class MyReceiver1 : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
            Toast.makeText(context,"Первый прочёл!",Toast.LENGTH_SHORT).apply{
            setGravity(Gravity.TOP,0,0)
            show() }
      }
 }

class MyReceiver2 : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
            Toast.makeText(context,"Второй прочёл!",Toast.LENGTH_SHORT).apply{
            setGravity(Gravity.CENTER,0,0)
            show()}
      }
}

class MyReceiver3 : BroadcastReceiver() {
    override fun onReceive(context: Context, intent: Intent) {
            Toast.makeText(context,"Третий прочёл",Toast.LENGTH_SHORT).apply{
            setGravity(Gravity.BOTTOM,0,0)
            show()}
      }
}
```

5. В активности создайте четыре приёмника.

```
    val r1 = MyReceiver1()
    val r2 = MyReceiver2()
    val r3 = MyReceiver1()
    val r4 = MyReceiver3()
```

Зарегистрируйте их в методе `onCreate()` с одним фильтром и разными приоритетами, переопределите `onDestroy()` для отмены регистрации.

```
        val filter = IntentFilter("ru.myitacademy.samsung.advertising.POST")
        registerReceiver(r1, filter.apply { priority = 17 })
        registerReceiver(r2, filter.apply { priority = -7 })
        registerReceiver(r3, filter.apply { priority = 0 })
        registerReceiver(r4, filter.apply { priority = 7 })
```

6. В методе `onReceive()` первого класса приёмников получите из предыдущей трансляции значение `resultCode` в переменную `resCode: Int`

```
 var resCode = resultCode
```

данные трансляции в переменную `res: Bundle` и прочитайте из неё текстовую строку в переменную `str: String`

```
      var res = getResultExtras(true)
      var str = res.getString("strEx")
```

Объект данных обязательно нужно прочитать в переменную, поскольку он будет передаваться дальше по трансляции. Если данные текущим приёмником передаваться не будут, то объект можно не создавать, а сразу считать с него текст:

```
var str = getResultExtras(true).getString("strEx")
```

7. Теперь приёмник может изменять полученные значения из предыдущей трансляции:

```
      str += " \nПолезная информация" // изменили передаваемый текст
      res.putString("strEx",str) // записали его в передаваемые данные
      setResult(++resCode, resultData, res) //передали трансляцию дальше,
    // при этом увеличили счётчик количества передач на 1.
```

8. Внесите изменения в оставшиеся классы приёмников и запустите сначала приложение с приёмниками, а потом приложение рассылки. Обратите внимание на последовательность работы приёмников и возвращённое значение в итоговом Toast.

9. Добавьте в классы приёмников по очереди прерывание трансляции

```
abortBroadcast()
```

и проследите за состоянием данных трансляции.

Начать тур для пользователя на этой странице