## Фильтры намерений

### Цель работы

Научиться использовать неявный вызов активности и создавать свои фильтры намерений (intent filters) для запуска независимых компонентов приложения.

### Методические указания

В данной работе мы будем создавать несколько активностей и налаживать между ними сложное взаимодействие.

Создадим новый проект. В файле расположения основного (и пока единственного окна) создадим две кнопки. При нажатии на первую мы хотим, чтобы открывалось окно, отображающее текущее время, а при тапе на вторую - окно, выводящее текущую дату.

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="horizontal">

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/btnTime"

android:text="Show time">

</Button>

<Button

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:id="@+id/btnDate"

android:text="Show date">

</Button>

</LinearLayout>

В основном классе создадим обработчик тапа по этим кнопкам. Обратите внимание, что мы создаем новый интент, однако не указываем конкретное имя класса активности, как на прошлой работе, а указываем только некую строчку. Эта строчка задает действие, которое мы хотим выполнить. По идее, в системе может быть несколько активностей, которые умеют делать одно и то же. В таком случае, система найдет все такие компоненты и предоставит пользователю выбор.

public void onClick(View v) {

Intent intent;

switch(v.getId()) {

case R.id.btnTime:

intent = new Intent("ru.startandroid.intent.action.showtime");

startActivity(intent);

break;

case R.id.btnDate:

intent = new Intent("ru.startandroid.intent.action.showdate");

startActivity(intent);

break;

}

}

Создадим вторую активность. Это будет окно, отображающее текущее время. Пока это пустое окно:

package com.example.intentfilter;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

public class ActivityTime extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_time);

}

}

И третью, отвечающую за отображение даты, тоже пустую:

package com.example.intentfilter;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

public class ActivityDate extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_date);

}

}

Во второй активности реализуем отображение текущей даты. Обратите внимание на работу со стандартными методами и классами Java для работы с датой и временем. Советуем обратиться к документации для более глубокого изучения стандартной библиотеки Java.

SimpleDateFormat sdf = new SimpleDateFormat("HH:mm:ss");

String time = sdf.format(new Date(System.*currentTimeMillis*()));

TextView tvTime = (TextView) findViewById(R.id.tvTime);

tvTime.setText(time);

Модифицируем файл манифеста. Здесь мы сразу сделаем несколько вещей. Во-первых, мы добавили атрибут label нашей активности. Это нужно для того, чтобы в окне выбора активности она имела приятное и понятное название. Во-вторых, мы создали новый элемент - это, собственно и есть фильтр. Он задает, какие действия умеет делать наша активность, то есть на какие неявные вызовы она может откликаться. Добавим необходимые атрибуты и впишем название действия также, как мы его обозначили в программном файле ранее:

<activity android:name=".ActivityTime" android:label="Time basic">

<intent-filter>

<action android:name="com.example.intent.action.showtime" />

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

</activity>

Самостоятельно модифицируйте третью активность таким образом, чтобы она откликалась на второе действие и выводила на экран такую дату в похожем кратком формате. Проверьте, что при тапе на обоих кнопках запускаются соответствующие активности.

Создадим еще одну активность. Эта четвертая активность будет откликаться на оба действия. На этом примере мы продемонстрируем, как можно проверять значение интента, вызвавшего данную активность.

package com.example.intentfilter;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

public class Info extends AppCompatActivity {

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_info);

}

}

Добавим фильтр в манифест. В данном случае, мы добавим оба действия в один фильтр:

<activity android:name=".Info" android:label="Info">

<intent-filter>

<action android:name="com.example.intent.action.showdate" />

<action android:name="com.example.intent.action.showtime" />

<category android:name="android.intent.category.DEFAULT" />

</intent-filter>

</activity>

Добавляем различные действия для разных интентов. Для этого мы получим объект-интент, вызвавший данную активность с помощью специального метода и проверим его свойство Action:

// получаем Intent, который вызывал это Activity

Intent intent = getIntent();

// читаем из него action

String action = intent.getAction();

String format = "", textInfo = "";

// в зависимости от action заполняем переменные

if (action.equals("com.example.intent.action.showtime")) {

format = "HH:mm:ss";

textInfo = "Time: ";

}

else if (action.equals("com.example.intent.action.showdate")) {

format = "dd.MM.yyyy";

textInfo = "Date: ";

}

// в зависимости от содержимого переменной format

// получаем дату или время в переменную datetime

SimpleDateFormat sdf = new SimpleDateFormat(format);

String datetime = sdf.format(new Date(System.*currentTimeMillis*()));

TextView tvDate = (TextView) findViewById(R.id.tvInfo);

tvDate.setText(textInfo + datetime);

Теперь все готово для запуска приложения. На основном экране мы должны увидеть две кнопки:



При нажатии на одну из них нам предоставляется выбор, какую активность запустить для выполнения соответствующего действия:

|  |  |
| --- | --- |
|  |  |

При повторном выбор того же действия система запомнит наш выбор:

|  |  |  |
| --- | --- | --- |
|  |  |  |

Таким образом можно использовать действия для запуска различных активностей. В данной работе мы определили собственное действие, существующее только в нашем приложении. Однако, точно таким же образом можно использовать стандартные действия, определенные самой системой для выполнения распространенных задач. Самостоятельно изучите их.

#### Передача информации в интент

Теперь потренируемся передавать информацию в запущенный по интенту компонент. Для этого создадим поле для текстового ввода в схеме главного окна:

<LinearLayout

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content">

<TextView

android:id="@+id/textView2"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Last Name">

</TextView>

<EditText

android:id="@+id/etLName"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="5dp">

</EditText>

</LinearLayout>

В коде главной активности добавим необходимый код для извлечения строки из поля ввода. Теперь добавим вызов метода интента после его создания, но до старта активности по этому интенту:

intent.putExtra("lname", etLName.getText().toString());

Эта инструкция добавит в структуру данных интента пользовательской текстовое поле с именем "lname" и значением из нашего текстового поля. Таких полей можно добавлять сколько угодно.

Теперь воспользуемся переданной информацией во второй активности. Создадим для нее текстовое поле:

<TextView

android:id="@+id/tvView"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:layout\_marginTop="20dp"

android:text="TextView"

android:textSize="20sp">

</TextView>

И запишем в него информацию, извлеченную из интента:

String lName = intent.getStringExtra("lname");

tvView.setText("Your name is: " + lName);

Общая схема работы нашего приложения теперь выглядит так. Обращаем внимание, что мы модифицировали только активность “Info”, две остальные не получают из интена дополнительных полей (хотя они туда все равно передаются).

|  |  |  |
| --- | --- | --- |
|  |  |  |

#### Возвращение результатов из интента

Часто требуется запустить активность для того, чтобы получить из нее какой-либо результат. Для этого изучим механизм возврата значения из активности. Добавим в главное окно надпись и кнопку, по нажатию на которую мы будем переходить в другую активность и вводить текст, а при возврате в главную активность этот текст будет отображаться.

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="fill\_parent"

android:layout\_height="fill\_parent"

android:orientation="vertical">

<Button

android:id="@+id/btnName"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:layout\_margin="20dp"

android:text="Input name">

</Button>

<TextView

android:id="@+id/tvName"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:text="Your name is ">

</TextView>

</LinearLayout>

В коде главной активности добавим в обработчик события код вызова новой активности. Обратите внимание, что мы вызываем активность при помощи другого метода.

intent = new Intent(this, NameActivity.class);

startActivityForResult(intent, 1);

break;

Теперь создадим вторую активность и расположим в ней необходимые элементы для удобного ввода текста:

<LinearLayout

xmlns:android="http://schemas.android.com/apk/res/android"

android:layout\_width="match\_parent"

android:layout\_height="match\_parent"

android:orientation="vertical">

<LinearLayout

android:id="@+id/linearLayout1"

android:layout\_width="match\_parent"

android:layout\_height="wrap\_content"

android:layout\_margin="10dp">

<TextView

android:id="@+id/textView1"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:text="Name">

</TextView>

<EditText

android:id="@+id/etName"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_marginLeft="10dp"

android:layout\_weight="1">

<requestFocus>

</requestFocus>

</EditText>

</LinearLayout>

<Button

android:id="@+id/btnOK"

android:layout\_width="wrap\_content"

android:layout\_height="wrap\_content"

android:layout\_gravity="center\_horizontal"

android:text="OK">

</Button>

</LinearLayout>

В коде главной активности создадим обработчик нажатия на кнопку. В нем мы создаем интент, в который поместим возвращаемую информацию. Обратите внимание на вызов метода finish().

@Override

public void onClick(View v) {

Intent intent = new Intent();

intent.putExtra("name", etName.getText().toString());

setResult(*RESULT\_OK*, intent);

finish();

}

Теперь возвращаемся в главную активность. Здесь мы должны определить метод, обрабатывающий обратный интент:

@Override

protected void onActivityResult(int requestCode, int resultCode, Intent data) {

if (data == null) {return;}

String name = data.getStringExtra("name");

tvName.setText("Your name is " + name);

}

Проверим работоспособность нашего приложения:

|  |  |  |
| --- | --- | --- |
|  |  |  |

### Контрольные вопросы

1. Какие существуют соглашения в порядке наименования действий?

Некоторые стандартные действия определяются в классе Intent как константы ACTION\_string. Чтобы назначить одно из этих действий этому атрибуту, нужно добавить "android.intent.action." в строку, следующую за ACTION\_ . Например, для ACTION\_MAIN используется "android.intent.action.MAIN".

1. Как передать информацию в активность используя неявный вызов?

intent = new Intent(Intent.ACTION\_VIEW,Uri.parse("https://www.google.ru"));

startActivity(intent);

1. Какие еще параметры можно задавать при создании неявного интента?

При создании неявного намерения мы используем не имя класса, а указываем параметры action, data, category с определёнными значениями.

1. Зачем нужна категория в интент-фильтрах? Какие существуют категории?

Категория – это строка, содержащая дополнительные сведения о том, каким компонентом должна выполняться обработка объекта Intent. В объект Intent можно поместить любое количество категорий.

1. ALTERNATIVE - Наличие данной категории говорит о том, что действие должно быть доступно в качестве альтернативного тому, которое выполняется по умолчанию для элемента этого типа данных. Например, если действие по умолчанию для контакта — просмотр, то в качестве альтернативы его также можно редактировать

2. SELECTED\_ALTERNATIVE - То же самое, что и ALTERNATIVE, но вместо одиночного действия с использованием утверждения намерения, которое описано выше, применяется в тех случаях, когда нужен список различных возможностей. Одной из функций фильтра намерений может стать динамическое заполнение контекстного меню с помощью действий.

3. BROWSABLE - Говорит о том, что действие доступно из браузера. Когда намерение срабатывает в браузере, оно всегда содержит данную категорию. Если вы хотите, чтобы приложение реагировало на действия, инициированные браузером (такие как перехват ссылок на конкретный сайт), то должны добавить в его манифест категорию BROWSABLE.

4. DEFAULT - Установите эту категорию, чтобы сделать компонент обработчиком по умолчанию для действия, выполняемого с указанным типом данных внутри Фильтра намерений. Это необходимо и для Активностей, которые запускаются с помощью явных Намерений

5. GADGET - Наличие этой категории указывает на то, что данная активность может запускаться внутри другой активности.

6. HOME - Устанавливая эту категорию и не указывая при этом действия, вы создаете альтернативу для стандартного домашнего экрана.

7. LAUNCHER - Используя эту категорию, вы помещаете Активность в окно для запуска приложений.

1. Зачем нужен элемент <requestFocus>?

Элемент requestFocus позволяет установить фокус на нужном компоненте.

1. Зачем нужны аргументы requestCode и resultCode в обратном интенте?

requestCode используется, чтобы отличать друг от друга пришедшие результаты. Задается в методе startActivityForResult, и проверяется потом в onActivityResult, чтобы точно знать, на какой вызов пришел ответ.

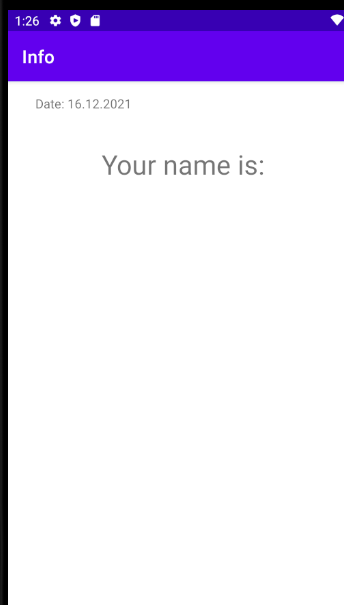
resultCode –статус вызова. Задается в методе setResult, и проверяется в onActivityResult, чтобы понять насколько успешно прошел вызов. Если при вызове что-то пошло не так, то вернется системная константа RESULT\_CANCELED.

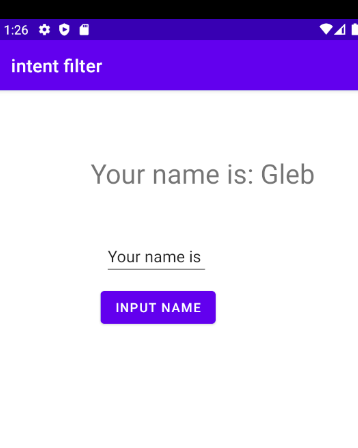
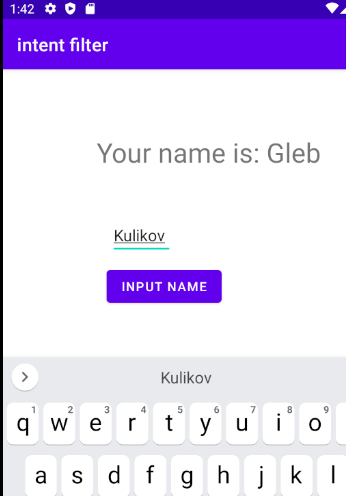
### Дополнительные задания

1. Создайте приложение, на главном окне которого будет расположено поле ввода текста и при нажатии на кнопку “перейти” будет запускаться браузер по введенному пользователем адресу.
2. Создайте приложение, отвечающее на какое-либо стандартное системное действие. Проверьте его работоспособность.
3. Создайте приложение, которое выводит текстовую надпись и предлагает выбрать цвет и выравнивание надписи. Выбор должен производиться в двух разных активностях. При возврате в основную активность форматирование надписи должно меняться.
4. (\*) Создайте приложение, запускающее приложение камеры. Когда пользователь делает снимок, он должен вернуться в наше приложение, и оно должно отобразить его в виде миниатюры на экране. Изображение выглядит как текст

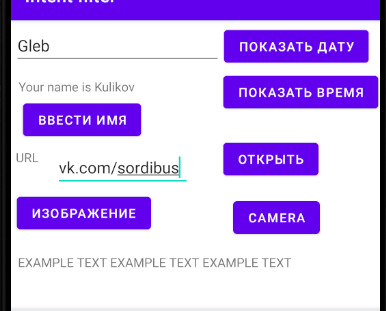
   Автоматически созданное описаниеИзображение выглядит как текст

   Автоматически созданное описаниеИзображение выглядит как квадрат

   Автоматически созданное описаниеИзображение выглядит как текст

   Автоматически созданное описаниеИзображение выглядит как текст

   Автоматически созданное описание



Изображение выглядит как текст

Автоматически созданное описаниеИзображение выглядит как текст, внутренний

Автоматически созданное описание