

Отчёт по проделанной работе на семинаре по мобильной разработке (25.02.2022).

(выполнил Валяев Георгий Анатольевич)

Сегодня уведомления, в свою очередь, пожалуй, дают весьма ценную возможность следить за новостным полем развития событий во множестве приложений как на мобильном телефоне, так и на других устройствах с Операционными системами. При разработке сервисов приложения создание и логика работы уведомлений стоят на одном уровне с самими сервисами, так как они взаимосвязаны по некоторым пунктам.

Отдельное слово стоит сказать всплывающим уведомлениям и push-уведомлениям, которые дали немного разнообразия по функционалу и более приятный глазу интерфейс временного уведомления. Именно вариант всплывающего уведомления мы и реализуем в этой методичке, при этом разработав 2 варианта: текстовый и с картинкой, что более интересно смотрится.

В файле XML главной активности мы создадим по традиции текстовое окно с информацией о функционале приложения, 2 кнопки, при нажатии которых появится текстовое всплывающее уведомление и уведомление с картинкой:

```
toast_custom.xml × activity_main.xml ×
1 <?xml version="1.0" encoding="utf-8"?>
2 <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3   xmlns:tools="http://schemas.android.com/tools"
4   android:layout_width="fill_parent"
5   android:layout_height="fill_parent"
6   android:orientation="vertical"
7   tools:context=".MainActivity">
8
9   <TextView
10     android:id="@+id/tvToastInfo"
11     android:layout_width="wrap_content"
12     android:layout_height="wrap_content"
13     android:layout_gravity="center"
14     android:layout_marginTop="300px"
15     android:textSize="50px"
16     android:text="Toast pop-up notifications and images"
17     android:textColor="@color/purple_700">
18 </TextView>
19
20   <Button
21     android:id="@+id/btnToast"
22     android:layout_width="wrap_content"
23     android:layout_height="wrap_content"
24     android:text="Create Toast_notify"
25     android:layout_gravity="center"
26     android:layout_marginTop="200px">
27 </Button>
28
29   <Button
30     android:id="@+id/btnImageToast"
31     android:layout_width="wrap_content"
32     android:layout_height="wrap_content"
33     android:text="Create Image_toast_notify"
34     android:layout_gravity="center"
35     android:layout_marginTop="200px">
36 </Button>
37
38 </LinearLayout>
```

Довольно просто. Коснёмся теперь MainActivity.java:

```

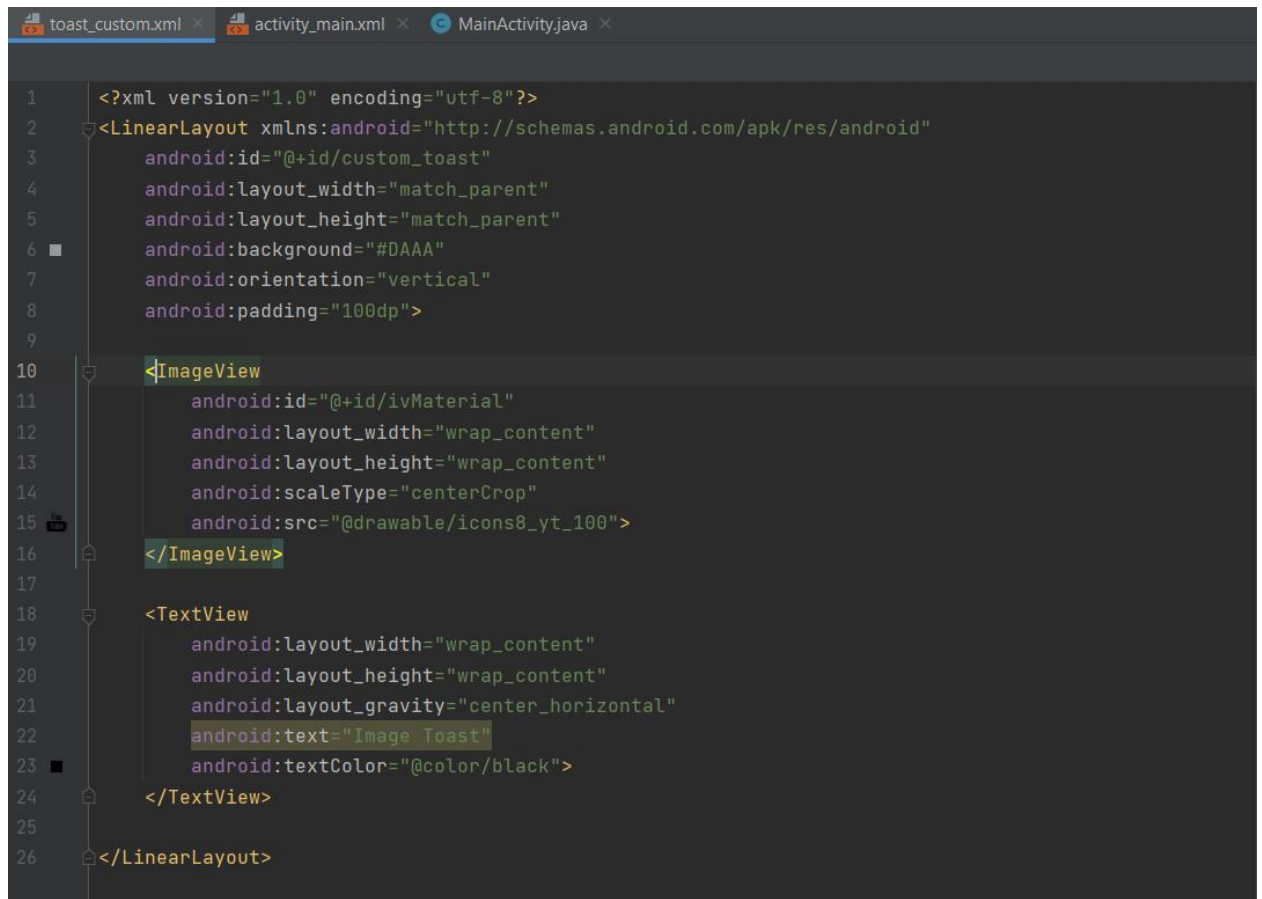
1  package com.example.toast_notify;
2
3  import ...
12
13 public class MainActivity extends AppCompatActivity {
14
15     @Override
16     protected void onCreate(Bundle savedInstanceState) {
17         super.onCreate(savedInstanceState);
18         setContentView(R.layout.activity_main);
19
20         Button btnToast = findViewById(R.id.btnToast);
21         Button btnImToast = findViewById(R.id.btnImageToast);
22
23         btnToast.setOnClickListener(new View.OnClickListener() {
24             @Override
25             public void onClick(View view) {
26                 String message = "It's a Toast message";
27                 Toast toast = Toast.makeText(getApplicationContext(), message, Toast.LENGTH_SHORT);
28                 toast.setGravity(Gravity.CENTER, xOffset: 100, yOffset: 0);
29                 toast.show();
30             }
31         });
32
33         btnImToast.setOnClickListener(new View.OnClickListener() {
34             @Override
35             public void onClick(View view) {
36                 LayoutInflater inflater = getLayoutInflater();
37                 View layout = inflater.inflate(R.layout.toast_custom,
38                     (ViewGroup) findViewById(R.id.custom_toast));
39                 Toast toast = new Toast(getApplicationContext());
40                 toast.setGravity(Gravity.CENTER_VERTICAL, xOffset: 0, yOffset: 0);
41                 toast.setDuration(Toast.LENGTH_LONG);
42                 toast.setView(layout);
43                 toast.show();
44             }
45         });
46     }
47 }

```

Итак, здесь мы инициализируем 2 переменные типа Button, для которых и реализуем обработчик нажатия на кнопку с выводом текстового всплывающего уведомления (с заданием расположения по центру). В случае с картинкой всё сложнее: через `getLayoutInflater()` и `inflater.inflate()` мы реализовали получение нашей картинки из специально созданного XML-

файла (toast_custom.xml), структуру которого рассмотрим ниже. После этого мы формируем объект уведомления Toast, задаём расположение и отправляем в качестве параметра метода .setView() полученную картинку с текстом.

А вот и структура нашего прикладного для уведомления файла XML:

A screenshot of an IDE window showing the XML structure for a custom toast notification. The window has three tabs: 'toast_custom.xml', 'activity_main.xml', and 'MainActivity.java'. The 'toast_custom.xml' tab is active, displaying the following XML code:

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:id="@+id/custom_toast"
4      android:layout_width="match_parent"
5      android:layout_height="match_parent"
6      android:background="#DAAA"
7      android:orientation="vertical"
8      android:padding="100dp">
9
10     <ImageView
11         android:id="@+id/ivMaterial"
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:scaleType="centerCrop"
15         android:src="@drawable/icons8_yt_100">
16     </ImageView>
17
18     <TextView
19         android:layout_width="wrap_content"
20         android:layout_height="wrap_content"
21         android:layout_gravity="center_horizontal"
22         android:text="Image Toast"
23         android:textColor="@color/black">
24     </TextView>
25
26 </LinearLayout>
```

Здесь мы просто описали структуру нашего всплывающего уведомления с картинкой: картинка и сообщение ниже о событии с расположением по центру экрана приложения.

Рассмотрим итоговый результат работы приложения:

9:21 ⓘ



Toast_notify

Toast pop-up notifications and images

CREATE TOAST_NOTIFY

CREATE IMAGE_TOAST_NOTIFY



9:21 ⓘ



Toast_notify

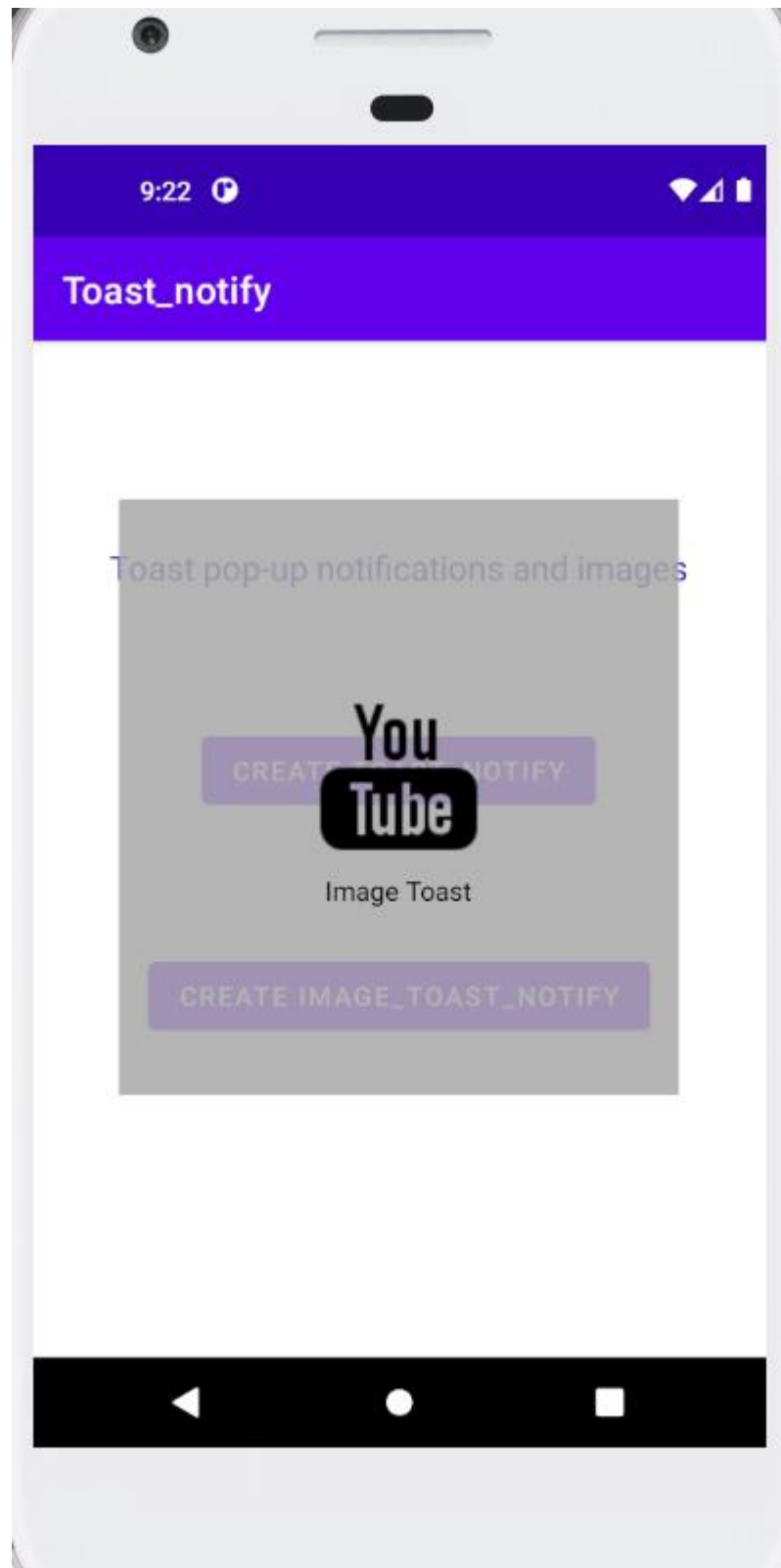
Toast pop-up notifications and images

CREATE TOAST_NOTIFY

CREATE IMAGE_TOAST_NOTIFY

It's a Toast message





Таким образом, работа приложения была реализована!

Контрольные вопросы

1. Чем сервисы отличаются от Activity?
2. Возможно ли в Toast Notification разместить LinearLayout?

3. Сколько времени отображается всплывшее сообщение созданное с помощью объекта **Toast**?

1) В основном, сервисы имеют возможность пережить активность приложения и работать в фоновом режиме, выполняя при этом какие-то рутинные и объёмные длительные операции, также имея доступ к удалённым ресурсам: обработки информации при сетевых запросах к веб-серверу, после чего могут начать работу сами уведомления. Сервисы могут быть в работе очень длительное время до тех пор, пока их кто-то не отключит, или они сами себя не остановят.

2) Такое возможно. В нашей работе мы напрямую связываем объект Toast и специально созданный визуал для уведомления (layout):

```
LayoutInflater inflater = getLayoutInflater();  
View layout = inflater.inflate(R.layout.toast_custom,  
    (ViewGroup) findViewById(R.id.custom_toast));  
Toast toast = new Toast(getApplicationContext());
```

3) Такого типа уведомление активно в приложении в пределах от 2 до 3,5 секунд, так как этот параметр задаётся в методе `Toast.makeText(getApplicationContext(), "Текст уведомления", Toast.LENGTH_SHORT OR Toast.LENGTH_LONG)`.