Вступление

Что такое тост? Представьте себе картину. За столом собралась большая куча народа и весело отмечает день рождения кота. Стоит шум и гам. Соседи громко разговаривают между собой и не обращают внимания на других. И тут из-за стола поднимается всеми уважаемый человек со стаканом вина и вилочкой стучит по стеклу стакана, чтобы привлечь внимание присутствующих. Шум смолкает и человек произносит тост. Также и в телефоне, когда вы увлечены какой-то задачей, вдруг всплывает сообщение, привлекая ваше внимание. Это и есть Toast. Второй пример - когда вы заряжаете специальный хлеб (тосты) в тостер, то они через определенное время подпрыгивают, сигнализируя о своей готовности. Посмотрим, как это работает в Android.

Теория

Всплывающее уведомление (Toast Notification) является сообщением, которое появляется на поверхности окна приложения, заполняя необходимое ему количество пространства, требуемого для сообщения. При этом текущая деятельность приложения остается работоспособной для пользователя. В течение нескольких секунд сообщение плавно закрывается. Всплывающее уведомление также может быть создано службой, работающей в фоновом режиме. Как правило, всплывающее уведомление используется для показа коротких текстовых сообщений.

Практика

Для создания всплывающего уведомления необходимо инициализировать объект **Toast** при помощи метода **Toast.makeText()**, а затем вызвать метод **show()** для отображения сообщения на экране:

```
Toast toast = Toast.makeText(getApplicationContext(),
   "Пора покормить кота!", Toast.LENGTH_SHORT);
toast.show();
```

У метода **makeText()** есть три параметра:

- Контекст приложения;
- Текстовое сообщение;
- Продолжительность времени показа уведомления. Можно использовать **только** две константы;

Константы для указания продолжительности показа сообщения

- **LENGTH_SHORT** — (По умолчанию) показывает текстовое уведомление на короткий промежуток времени;
- **LENGTH_LONG** — показывает текстовое уведомление в течение длительного периода времени.

Если покопаться в исходниках Android, то можно найти такие строчки:

```
private static final int LONG_DELAY = 3500; // 3.5 seconds
private static final int SHORT_DELAY = 2000; // 2 seconds
```

Как видите, уведомления выводятся на 3 с половиной секунды или на 2 секунды. Других вариантов нет, не пытайтесь использовать другие значения - у вас ничего не получится.

Настройка позиции на экране

По умолчанию стандартное всплывающее уведомление появляется в нижней части экрана. Изменить место появления уведомления можно с помощью метода **setGravity(int, int, int)**. Метод принимает три параметра:

- стандартная константа для размещения объекта в пределах большего контейнера (например, GRAVITY.CENTER, GRAVITY.TOP и др.);
- смещение по оси X;
- смещение по оси Y.

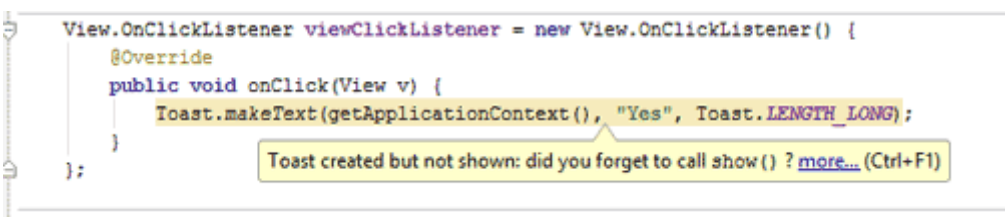
Например, если вы хотите, чтобы уведомление появилось в центре экрана, то используйте следующий код:

```
toast.setGravity(Gravity.CENTER, 0, 0);
```

Если нужно сместить уведомление направо, то просто увеличьте значение второго параметра. Для смещения вниз нужно увеличить значение последнего параметра. Соответственно, для смещения вверх и влево используйте отрицательные значения.

Не забывайте про метод show()

Типичная ошибка начинающих программистов - забывают добавить вызов метода **show()** для отображения сообщения на экране. К счастью, в студии, если вы пропустите метод **show()**, то строка будет подсвечена, а при подведении указателя мыши к строке увидите:



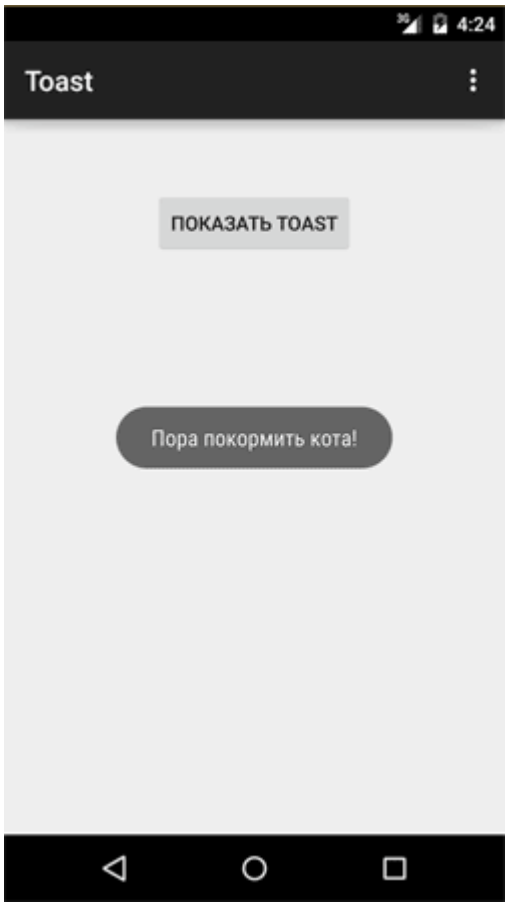
Пример

Создайте новый проект или используйте любой старый проект из предыдущих занятий. Добавьте на экран активности кнопку и присвойте ей текст **Показать Toast**, а также присвойте атрибуту **android:onClick** значение **showToast**. Теперь напишем код:

```
public void showToast(View view) {
    //создаем и отображаем текстовое уведомление
    Toast toast = Toast.makeText(getApplicationContext(),
            "Пора покормить кота!",
            Toast.LENGTH_SHORT);
    toast.setGravity(Gravity.CENTER, 0, 0);
    toast.show();
}
```

Запустите приложение и нажмите кнопку. В центре экрана появится на короткое время текстовое сообщение, которое само исчезнет.

В Android 4.4 внешний вид всплывающего сообщения изменился, прямоугольник закруглили (http://developer.alexanderklimov.ru/android/theory/whatsnew.php#kitekat). Раньше был прямоугольник.



Для закрепления материала напишем еще один пример. Удалим предыдущий код для щелчка кнопки и напишем такой код:

```
int duration = Toast.LENGTH_LONG;
Toast toast2 = Toast.makeText(getApplicationContext(),
                R.string.catfood,
                duration);
toast2.setGravity(Gravity.TOP, 0, 0);
toast2.show();
```

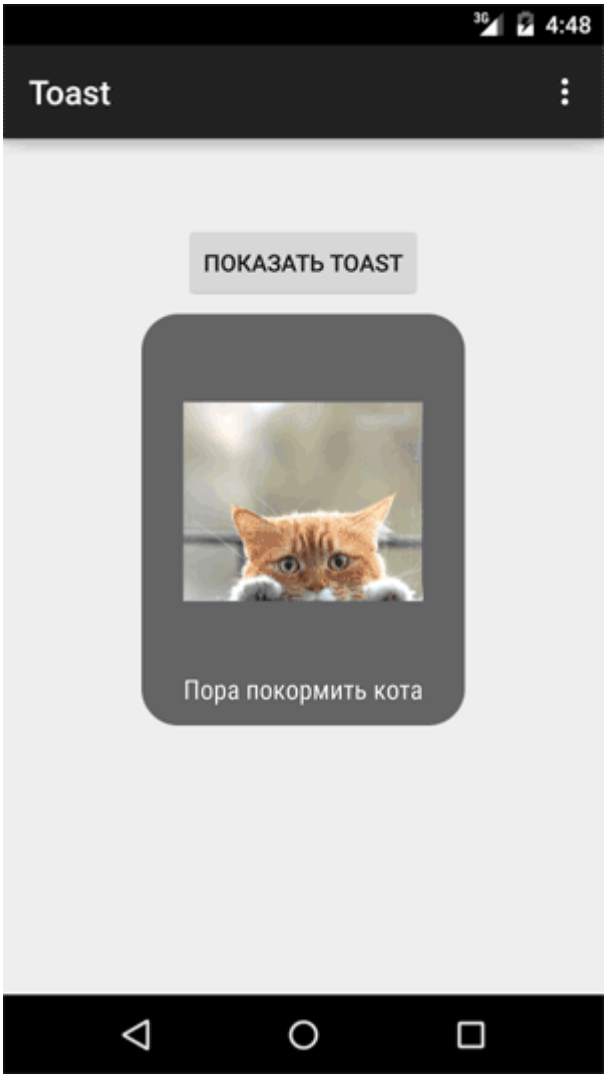
Я подумал, что вы можете не заметить сообщение, которое показывается слишком мало времени. Поэтому на этот раз я использовал константу **LENGTH_LONG**, чтобы вы успели обратить внимание на сообщение и покормить наконец несчастного голодного кота. Помимо этого, я поместил текст сообщения в XML-ресурсы, как это рекомендуется всегда делать. Кроме того, сообщение будет выводиться в верхней части экрана.

Добавляем картинку

Как правило, для **Toast** используются короткие текстовые сообщения. При необходимости вы можете добавить к сообщению и картинку. Используя метод **setView()**, принадлежащий объекту **Toast**, вы можете задать любое представление (включая разметку) для отображения.

Начнем с приготовлений. Подготовьте картинку и разместите её в папке **res/drawable**, как мы делали в уроке с "Hello Kitty". Картинка будет доступна приложению как ресурс через название файла без расширения. Например, я добавил в папку файл с изображением кота **hungrycat.jpg** и могу получить к нему доступ через выражение **R.drawable.hungrycat**. Чтобы изображение появилось в стандартном Toast-сообщении, нам потребуется программно создать объект класса **ImageView** и задать для него изображение из ресурсов с помощью метода **setImageResource**. Сам по себе стандартный внешний вид **Toast** состоит из контейнера **LinearLayout**, в который нужно добавить созданный объект **ImageView**. Можно задать также позицию, в которую следует вывести изображение. Если указать значение 0, то изображение будет показано выше текста. Код для создания **Toast** с изображением выглядит следующим образом:

```
public void showToast(View view) {
    Toast toast3 = Toast.makeText(getApplicationContext(),
            R.string.catfood, Toast.LENGTH_LONG);
    toast3.setGravity(Gravity.CENTER, 0, 0);
    LinearLayout toastContainer = (LinearLayout) toast3.getView();
    ImageView catImageView = new ImageView(getApplicationContext());
    catImageView.setImageResource(R.drawable.hungrycat);
    toastContainer.addView(catImageView, 0);
    toast3.show();
}
```



Создание собственных всплывающих уведомлений

В предыдущем примере мы получили доступ к контейнеру через метод **getView()**. Можно пойти от обратного - подготовить свой контейнер и внедрить его в объект **Toast** через метод **setView()**.

Если простого текстового сообщения недостаточно для уведомления пользователя приложения, можно создать собственный дизайн разметки своего уведомления. Для получения разметки из XML-файла и работы с ней в программе используется класс **LayoutInflater** и его методы **getLayoutInflater()** или **getSystemService()**, которые возвращают объект **LayoutInflater**. Затем вызовом метода **inflate()** получают корневой объект **view** этой разметки. Например, для файла разметки уведомления с именем **custom_layout.xml** и его корневого элемента с идентификатором **android:id="@+id/toast_layout"** код будет таким:

```
LayoutInflater inflater = getLayoutInflater();
View layout = inflater.inflate(R.layout.custom_layout,
                (ViewGroup) findViewById(R.id.toast_layout));
```

Параметры, передаваемые в метод **inflate()**:

- идентификатор ресурса схемы размещения (custom_layout.xml);
- идентификатор ресурса корневого представления (toast_layout).

После получения корневого представления из него можно получить все дочерние представления уже известным методом **findViewByid()** и определить информационное наполнение для этих элементов.

Затем создается объект **Toast** и устанавливаются нужные свойства, например, **Gravity** и продолжительность времени показа уведомления.

```
Toast toast = new Toast(getApplicationContext());
toast.setGravity(Gravity.CENTER_VERTICAL, 0, 0);
toast.setDuration(Toast.LENGTH_LONG);
```

После этого вызывается метод **setView()**, которому передаётся разметка уведомления, и метод **show()**, чтобы отобразить уведомление с собственной разметкой:

```
toast.setView(layout);
toast.show();
```

Вам нужно создать разметку **custom_layout.xml**.

Определите два дочерних элемента **ImageView** и **TextView**:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/toast_layout"
    android:orientation="horizontal"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:padding="10dp"
    android:background="#DAAA">
<ImageView android:id="@+id/imageView"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:layout_marginRight="10dp"/>
<TextView android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="match_parent"
    android:text="@string/catfood"
    android:textColor="#FFF777">
</LinearLayout>
```

Запустите проект на выполнение. При нажатии кнопки вызова должно появиться на несколько секунд окно уведомления с текстовым сообщением и значком.

Использование уведомлений Toast в рабочих потоках

Как элемент графического интерфейса **Toast** должен быть вызван в потоке GUI, иначе существует риск выброса межпотокового исключения. В листинге объект **Handler** используется для гарантии того, что уведомление **Toast** было вызвано в потоке GUI.

```
private void mainProcessing() {
    Thread thread = new Thread(null, doBackgroundThreadProcessing,
            "Background");
        thread.start();
}

private Runnable doBackgroundThreadProcessing = new Runnable() {
    public void run() {
        backgroundThreadProcessing();
    }
};

private void backgroundThreadProcessing() {
    handler.post(doUpdateGUI);
}

// Объект Runnable, который вызывает метод из потока GUI
private Runnable doUpdateGUI = new Runnable() {
    public void run() {
        Context context = getApplicationContext();
        String msg = "To open mobile development!";
        int duration = Toast.LENGTH_SHORT;
        Toast.makeText(context, msg, duration).show();
    }
};
```

Дополнительные сведения

Напоследок хочу предупредить об одной потенциальной проблеме. При вызове сообщения нужно указывать контекст в первом параметре метода **makeText()**. В интернете и, возможно и у меня на сайте будет попадаться пример **makeText(MainActivity.this, ...)**. Ошибки в этом нет, так как класс **Activity** является потомком **Context** и в большинстве случаев пример будет работать. Но иногда я получаю письма от пользователей, которые жалуются на непонятное поведение сообщения, когда текст не выравнивается, обрезается и т.д. Это связано с тем, что активность может использовать определённую тему или стиль, которые вызывают такой побочный эффект. Поэтому я рекомендую вам использовать метод **getApplicationContext()**.