Тема 12. PUSH-уведомление[1]

Кроме Toast-уведомлений, существует также другой тип уведомлений, который выводится в системной строке состояния в виде значка с небольшим текстом. Если открыть окно уведомлений, то можно увидеть расширенную текстовую информацию об уведомлении.

Когда пользователь открывает расширенное сообщение, Android запускает объект Intent, который определён в соответствии с уведомлением. Можно также конфигурировать уведомление с добавлением звука, вибрации и мигающих индикаторов на мобильном устройстве.

Этот вид уведомления удобен в том случае, когда приложение работает в фоновом режиме и должно уведомить пользователя о каком-либо важном событии. Уведомление будет висеть до тех пор, пока пользователь не отреагирует на него, в отличие от Toast-сообщения, которое исчезнет через несколько секунд. Фоновое приложение создаёт уведомление в строке состояния, но не запускает активность самостоятельно для получения пользовательского взаимодействия. Это должен сделать только сам пользователь в удобное ему время.

Чтобы создать уведомление в строке состояния, необходимо использовать два класса:

Notification – определяем свойства уведомления строки состояния: значок, расширенное сообщение и дополнительные параметры настройки (звук и др.)

NotificationManager – системный сервис Android, который управляет всеми уведомлениями. Экземпляр NotificationManager создается при помощи вызова метода *getSystemService()*, а затем, когда надо показать уведомление пользователю, вызывается метод *notify()*. Недавно появился более простой способ через метод *from()*.

Начиная с Android 3, для уведомлений используется класс Notification.Builder. Добавим на экран активности кнопку и создадим простой пример для демонстрации работы уведомления.

```
package ru.schoolbreaker.testapplication;
import android.app.Notification;
import android.app.NotificationManager;
import android.app.PendingIntent;
import android.content.Context;
import android.content.Intent;
import android.content.res.Resources;
import android.graphics.BitmapFactory;
import android.support.v7.app.ActionBarActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
```

```
import android.view.View;
public class MainActivity extends ActionBarActivity {
// Идентификатор уведомления
private static final int NOTIFY_ID = 101;
@Override
protected void onCreate(Bundle savedInstanceState) {
super.onCreate (savedInstanceState);
setContentView(R.layout.activity_main); }
public void onClick(View view) {
Context context = getApplicationContext();
Intent notificationIntent = new Intent(context, MainActiv ity.class);
PendingIntent contentIntent = PendingIntent.getActivity(c ontext,
0, notificationIntent,
PendingIntent.FLAG_CANCEL_CURRENT);
Resources res = context.getResources();
Notification.Builder builder = new Notification.Builder(c ontext);
builder.setContentIntent(contentIntent)
.setSmallIcon(R.drawable.ic_launcher_cat)
// большая картинка
setLargeIcon(BitmapFactory.decodeResource(res, R .drawable.hungrycat))
//.setTicker(res.getString(R.string.warning)) // текст в строке состояния
.setTicker("Последнее китайское предупреждение!")
.setWhen(System.currentTimeMillis())
.setAutoCancel(true)
//.setContentTitle(res.getString(R.string.notifyt itle)) // Заголовок уведомления
.setContentTitle("Напоминание")
//.setContentText(res.getString(R.string.notifyte xt))
.setContentText("Пора покормить кота"); // Текст у ведомления
// Notification notification = builder.getNotification(); // до API 16
Notification notification = builder.build();
NotificationManager notificationManager = (NotificationMa nager) context
.getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(NOTIFY_ID, notification); } }
```

Для начала вам надо создать идентификатор уведомления. Он нужен для того, чтобы можно было различать уведомления друг от друга. Ведь вы можете создать идеальное приложение, которое уведомляло бы хозяина, что кота надо покормить (первое уведомление), погладить (второе уведомление), почистить лоток (третье уведомление). А если у вас будет один идентификатор, то каждое новое уведомление затрёт предыдущее и хозяин не увидит свои недоработки. Для идентификатора используйте какое-нибудь число.

Далее формируется внешний вид и поведение уведомления через построитель Notification.Builder. Вы можете задать текст уведомлений, значки и прочие атрибуты:

Метод setSmallIcon() устанавливает маленький значок, который выводится в строке состояния, а также в правой части открытого уведомления.

Метод setLargeIcon() устанавливает большой значок, который выводится в открытом уведомлении слева.

Метод setTicker() выводит временную строку в строке состояния, которая затем исчезает. Остаётся только маленький значок (см. выше)

Остальные методы понятны по названиям. В примере закомментированы «правильные варианты» использования строк через ресурсы, но, чтобы было понятно, в примерах сразу показаны нужные строки.

Также нам нужны объекты Intent и PendingIntent, которые описывают намерения и целевые действия. В нашем случае мы хотим запустить нашу активность, когда пользователь среагирует на уведомление.

Начиная с API 16, вместо устаревшего метода getNotification() следует использовать метод build().

Далее надо сформировать уведомление с помощью специального менеджера. Ссылку на NotificationManager можно получить через вызов метода getSystemService(), передав ему в качестве параметра строковую константу NOTIFICATION_SERVICE, определённую в классе Context

Выводится уведомление с помощью метода notify() – своеобразный аналог метода show() у Toast из предыдущего урока.

На первой картинке мы видим, как появился маленький значок. Также появляется текст «Последнее китайское предупреждение!», который быстро исчезает, рис. 1.

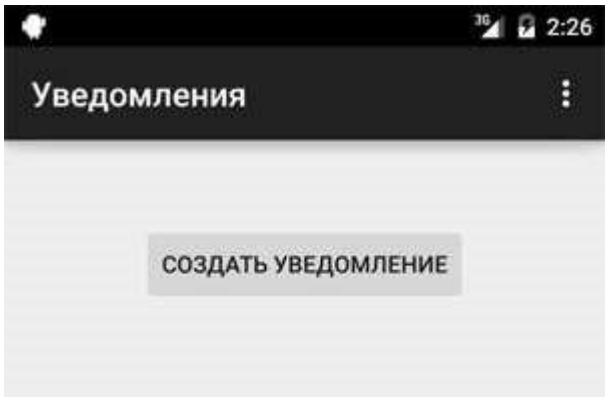

Рис. 1. Создание уведомлений

Далее мы можем открыть уведомление, чтобы увидеть более подробную информацию, рис. 2.

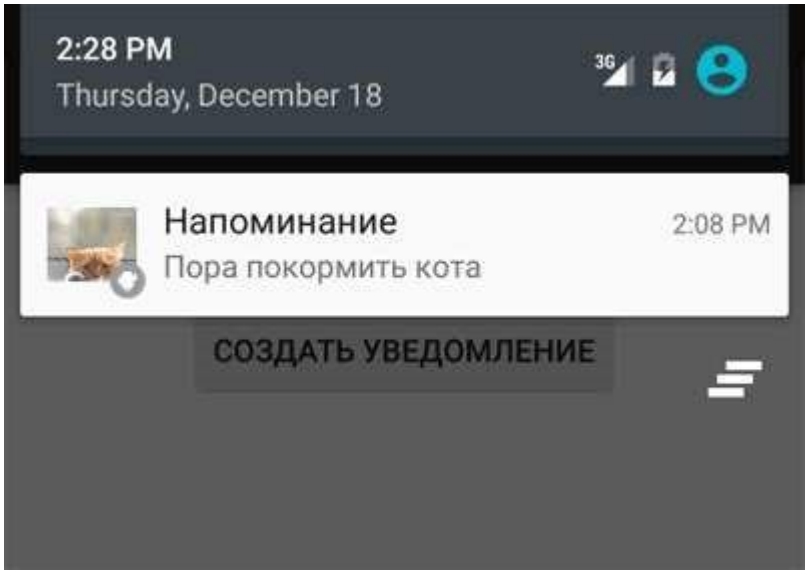

Рис. 2. Просмотр уведомления

В примере показан избыточный код, чтобы вы сразу получили представление о возможностях уведомлений. Часть методов с префиксом set можно пропустить.

Ниже показан укороченный код с использованием библиотеки совместимости для старых устройств. Обратите также внимание на упрощённый вариант доступа к менеджеру уведомлений через методNotificationManagerCompat.from(this).

```
Notification.Builder builder = new NotificationCompat.Builder(thi s);
// оставим только самое необходимое
builder.setContentIntent(contentIntent)
.setSmallIcon(R.drawable.ic_launcher_cat)
.setContentTitle("Напоминание")
.setContentText("Пора покормить кота"); // Текст уведомле ния
Notification notification = builder.build();
NotificationManagerCompat notificationManager = NotificationManag erCompat.from(this);
notificationManager.notify(NOTIFY_ID, notification);
```

Как мы уже упоминали, если вам нужно обновить уведомление, то просто ещё раз отправьте его устройству под этим же идентификатором.

Если коснуться уведомления, то запустится наша программа (даже если она была перед этим закрыта).

Совсем не обязательно запускать своё приложение, хотя это является распространённой практикой. Можете задать нужное поведение, например, запустить свой сайт по указанному адресу. Переделаем код:

```
Context context = getApplicationContext();
Intent notificationIntent = new Intent(Intent.ACTION_VIEW,
Uri.parse("http://developer.schoolbreaker.ru/android/"));
PendingIntent pendingIntent = PendingIntent.getActivity(context, 0,
notificationIntent, PendingIntent.FLAG_CANCEL_CURRENT);
Notification.Builder builder = new Notification.Builder(context)
```

```
.setContentTitle("Посетите мой сайт")
.setContentText("http://developer.schoolbreaker.ru/androi d/")
.setTicker("Внимание!").setWhen(System.currentTimeMillis())
.setContentIntent(pendingIntent)
.setDefaults(Notification.DEFAULT_SOUND).setAutoCancel(tr ue)
.setSmallIcon(R.drawable.ic_launcher);
NotificationManager notificationManager = (NotificationManager)
getSystemService(Context.NOTIFICATION_SERVICE);
notificationManager.notify(NOTIFY_ID, builder.build());
```

Обратите внимание, что на этот раз мы не указали картинку для большого значка и система подставляет в этом случае маленький значок, растягивая его до нужных размеров. Также появился новый метод ***setDefaults()***, о котором говорится ниже.

Также можно вывести индикатор прогресса, чтобы указать текущий ход выполнения задачи. Можно установить бесконечное выполнение:

```
setProgress(100, 50, false);
```

Если ваша цель – только вывести уведомление, но не запускать активность при нажатии на самом уведомлении, то используйте вызов намерения без параметров:

```
Intent intent = new Intent();
```

Вы можете из программы удалить своё уведомление, посланное по глупости.

```
notificationManager.cancel(NOTIFY_ID);
```

Можно добавить вибрацию, звуковой сигнал или мерцание светодиодами для ваших уведомлений при помощи настроек по умолчанию. В свойстве defaults вы можете сочетать следующие константы:

Notification.DEFAULT_LIGHTS
Notification.DEFAULT_SOUND
Notification.DEFAULT_VIBRATE

Чтобы к уведомлению добавить звук и вибрации по умолчанию, используйте код:

```
notification.defaults = Notification.DEFAULT_SOUND | Notification.DEFAULT_VIBRATE;
```

Если хотите установить сразу все значения по умолчанию, задействуйте константу *Notification.DEFAULT_ALL*.

Использование звуковых оповещений для уведомления пользователя о событиях, связанных с устройством (например, входящий звонок), стало привычным. Большинство стандартных событий, от входящих звонков до новых сообщений и низкого заряда батареи, объявляются с помощью звуковых мелодий. Android позволяет проигрывать любой звуковой файл на телефоне в качестве уведомления. Чтобы это сделать, нужно присвоить свойству sound путь URI:

```
notification.sound = ringURI;
```

Также можно использовать собственный звуковой файл, загруженный на устройстве или добавленный в проект в качестве ресурса.

```
Uri ringURI  = RingtoneManager.getDefaultUri(RingtoneManager.TYPE_NOTIFICATION);
notification.sound  = ringURI;
```

С SD-карты:

```
notification.sound = Uri.parse("file:///sdcard/cat.mp3");
// если знаем точный путь!
```

Объект Notification включает в себя свойства для настройки цвета и частоты мерцания светодиодов устройства. Здесь стоит обратить внимание, что конкретные модели устройств могут не содержать светодиодные индикаторы или иметь другие цвета.

Свойство ledARGB может устанавливать цвет для светодиодной подсветки. Свойства ledOffMS и ledOnMS позволяют регулировать частоту и поведение светодиодов. Вы можете включить светодиоды, присвоив свойству ledOnMS значение 1, а ledOffMS – 0. Присвоив им обоим значения 0, светодиоды можно выключить.

Настроив работу со светодиодами, необходимо также добавить флаг FLAG_SHOW_LIGHTS к свойству flags объекта Notification.

В следующем фрагменте кода показано, как включить на устройстве красный светодиод:

```
notification.ledARGB  = Color.RED;
notification.ledOffMS  = 0;
otification.ledOnMS  = 1;
notification.flags  = notification.flags  | Notification.FLAG_SHOW_  LIGHTS;
```

В настоящее время эмулятор Android не умеет визуально показывать активность светодиодов.

Вы можете делать уведомления текущими и/или настойчивыми, устанавливая флаги *FLAG_INSISTENT* и *FLAG_ONGOING_EVENT*. Уведомления, помеченные как текущие, используются для представления событий, которые выполняются в данный момент времени (например, загрузка файла, фоновое проигрывание музыки). Текущие уведомления необходимы для сервисов, работающих на переднем плане. Пример установки флагов:

```
notification.flags  = notification.flags  | Notification.FLAG_ONGOI NG_EVENT;
```

В расширенной статусной строке текущие события отделены от обычных, чтобы вы сразу могли их отличить.

Настойчивые уведомления непрерывно повторяют звуковые сигналы, вибрируют и мерцают светодиодами, пока не будут остановлены. Подобные уведомления, как правило, используются для событий, которые требуют

немедленного и своевременного внимания, таких как входящий звонок, срабатывание будильника или время кормёжки кота. В следующем фрагменте кода показано, как сделать уведомление настойчивым:

```
notification.flags  = notification.flags  | Notification.FLAG_INSIS TENT;
```

В методе *getActivity()* может понадобиться изменить флаг, например

```
PendingIntent pendingIntent = PendingIntent.getActivity(context,  0
,intent,  Intent.FLAG_ACTIVITY_NEW_TASK);
```

Существуют и другие флаги. Хотя в большинстве случаев используется просто 0.

Не сразу бывает заметно, но на самом деле, когда при нажатии на уведомлении у вас запускается активность, то запускается не старая активность, которая была на экране до этого, а новая. Это можно увидеть в примере, если, например, есть текстовое поле с текстом (создайте такую активность). Введите какой-нибудь текст в активности, а потом создайте уведомление, вызывающее активность. Вы увидите, что запустится новая активность с пустым текстовым полем, хотя мы ожидали увидеть запущенную активность. Если вам нужен именно этот вариант, то используйте флаги для намерения.

```
Intent intent = new Intent(context, MainActivity.class);
intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP | Intent.FLAG_ACTIVITY_SINGLE_TOP);
```

Либо вы можете прописать в манифесте для нужной активности атрибут *android:launchMode="singleTop"*.