**Уведомления в Android Notification + PendingIntent на примере**

В этом уроке мы научимся создавать уведомления (Notifications) с помощью PendingIntent, а также разберем различные настройки и кастомизацию наших уведомлений.

**Краткий обзор PendingIntent в Android**

Для начала нам нужно понять для чего нужен объект PendingIntent.

И так, PendingIntent является оболочкой для объектов [Intent](http://prologistic.com.ua/znakomstvo-s-android-intent-zapusk-activity-s-pomoshh-yu-intent.html" \t "_blank) (намерений) и определяет действие, которое должно выполниться в будущем. Как правило, он используется в для запуска уведомлений или в паре с AlarmManager для повторяющихся действий.

Из соображений безопасности, передающийся в PendingIntent объект Intent должен явно задавать имя компонента. Каждый явный Intent в свою очередь должен обрабатываться такими компонентами, как Activity, BroadcastReceiver или Service. Для этого PendingIntent использует следующие методы:

* PendingIntent.getActivity() — получение PendingIntent для запуска начать Activity.
* PendingIntent.getBroadcast() — получение PendingIntent для запуска объектов Broadcast.
* PendingIntent.getService() — получение PendingIntent для запуска службы.

Рассмотрим создание PendingIntent на примере:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | Intent intent = new Intent(this, SomeActivity.class);    // Упаковываем объект Intent в PendingIntent  PendingIntent pendingIntent = PendingIntent.getActivity(                this, 1, intent, PendingIntent.FLAG\_UPDATE\_CURRENT);  try {      // Запускаем созданный pendingIntent      pendingIntent.send();  } catch (PendingIntent.CanceledException e) {      e.printStackTrace();  } |

Рассмотрим подробнее параметры, которые мы передали в методе getActivity():

* Первый параметр this — контекст, в котором PendingIntent начинает свою работу.
* Второй параметр 1 (requestCode) — код запроса, с которым отправляем наш PendingIntent.
* Третий параметр intent — явное намерение с указанием объекта Activity, который должен быть запущен.
* Четвертый параметр PendingIntent.FLAG\_UPDATE\_CURRENT — один из возможных флагов объекта PendingIntent. В примере выше мы использовали FLAG\_UPDATE\_CURRENT, который обновит значение PendingIntent (если такой уже существует) последним объектом Intent. Также есть такой флаг, как FLAG\_CANCEL\_CURRENT, который отменяет уже существующий PendingIntent.

**Уведомления в Android. Объекты Notification**

В Android уже есть класс Toast, который является удобным способом оповестить пользователя с помощью различных уведомлений, но проблема заключается в том, что эти сообщения не являются постоянными и в течении нескольких секунд исчезают. Да, сейчас в Android используется [виджет SnackBar](http://prologistic.com.ua/primer-ispol-zovaniya-snackbar-v-android-sozdaem-kastomny-j-snackbar.html" \t "_blank) для этих целей, однако мы говорим об объектах Notification, которые подразумевают другой способ оповещения пользователя.

С помощью объекта Notification мы можем показать пользователю сообщение вне пользовательского интерфейса нашего приложения и использовать различные способы его вызова.

**Создаем Notification в Android. Наиболее используемые методы**

Уведомления создается с помощью класса NotificationManager, как показано ниже:

|  |  |
| --- | --- |
| 1  2 | NotificationManager notificationManager = (NotificationManager)    getSystemService (NOTIFICATION\_SERVICE); |

Также предоставляется интерфейс Notification.Builder для создания объекта уведомления:

|  |  |
| --- | --- |
| 1 | NotificationCompat.Builder mNotificationBuilder = new NotificationCompat.Builder(this) |

Теперь давайте рассмотрим наиболее используемые методы при создании уведомлений:

* Метод build() — компонует все заданные параметры и возвращает новый объект Notification.
* Метод NotificationCompat.Builder setAutoCancel(boolean autoCancel) — устанавливает поведение уведомления таким образом, чтобы оно автоматически отменялось, когда пользователь нажимает кнопку на панели
* Метод setContent (RemoteViews views) предоставляет работу с RemoteViews (уведомления на экране блокировки) вместо стандартной панели уведомлений
* Метод setContentInfo (CharSequence info) устанавливает большой размер текста в правой части уведомления
* Метод setContentIntent (PendingIntent intent) предоставляет объект PendingIntent для отправки после нажатия пользователем на уведомление.
* Метод setContentText (CharSequence text) устанавливает текст во второй строке уведомления
* Метод setContentTitle (CharSequence title) устанавливает текст в первой строке уведомления (установка заголовка уведомления)
* Метод setDefaults (int defaults) устанавливает параметры оповещений по умолчанию. Например, mBuilder.setDefaults (Notification.DEFAULT\_LIGHTS | Notification.DEFAULT\_SOUND)
* Метод setLargeIcon (Bitmap icon) устанавливает большой значок для уведомления
* Метод setSmallIcon (int icon) устанавливает маленький значок на панели уведомлений
* Метод setTicker (CharSequence tickerText) устанавливает текст, который будет отображаться в строке состояния, когда уведомление только появится
* Метод setVibrate (long[] pattern) устанавливает паттерн вибрации (сколько будет длиться и какие интервалы), когда уведомление только появится
* Метод setWhen (long when) устанавливает время, когда произошло событие в уведомлении (объекты Notification сортируются по этому времени в панели уведомлений).

Интерфейс Notification.Builder позволяет добавлять до трех кнопок с определёнными действиями на одно уведомление.

Начиная с версии Android API 4.1 и выше Android появилась большие уведомления с расширенным функционалом.

**Отменяемые уведомления в Android**

Android позволяет нам отменять уведомления с помощью вызова метода cancel() для конкретного уведомления (получаем по id) или вызвать метод NotificationManager.cancelAll() для удаления всех ранее полученных уведомлений.

**Создаем уведомление (Notification) в Android**

В нашем тестовом приложении мы создадим объект Intent для просмотра конкретной веб-страницы, обернем его в PendingIntent и заставим вызываться по нажатию пользователя. Также добавим функцию отмены уведомления.

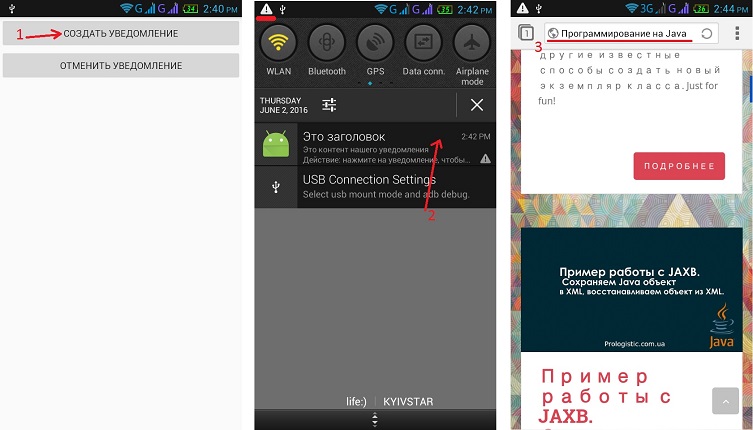
[Создадим проект в Android](http://prologistic.com.ua/pishem-hello-world-na-android-rabotaem-v-android-studio.html) и добавим в макет **activity\_main.xml** два виджета [Button](http://prologistic.com.ua/sozdaem-knopku-v-android-primer-raboty-s-vidzhetom-button-v-android.html" \t "_blank).Первая кнопка будет отвечать за создание уведомления, а другая за его отмену:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | <?xml version="1.0" encoding="utf-8"?>  <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"      xmlns:tools="http://schemas.android.com/tools"      android:layout\_width="match\_parent"      android:layout\_height="match\_parent"      tools:context="ua.com.prologistic.materialdesignapp.MainActivity">        <Button          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Создать уведомление"          android:id="@+id/btn\_create\_notification"          android:layout\_alignParentTop="true"          android:layout\_alignParentRight="true"          android:layout\_alignParentEnd="true"          android:layout\_alignParentLeft="true"          android:layout\_alignParentStart="true" />        <Button          android:layout\_width="wrap\_content"          android:layout\_height="wrap\_content"          android:text="Отменить уведомление"          android:id="@+id/btn\_cancel\_notification"          android:layout\_below="@+id/btn\_create\_notification"          android:layout\_alignRight="@+id/btn\_create\_notification"          android:layout\_alignEnd="@+id/btn\_create\_notification"          android:layout\_alignParentLeft="true"          android:layout\_alignParentStart="true" />  </RelativeLayout> |

Теперь идем в MainActivity, инициализируем кнопки и создаем методы отправки и отмены уведомления:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78 | package ua.com.prologistic.materialdesignapp;    import android.app.NotificationManager;  import android.app.PendingIntent;  import android.content.Context;  import android.content.Intent;  import android.graphics.BitmapFactory;  import android.net.Uri;  import android.os.Bundle;  import android.support.v7.app.AppCompatActivity;  import android.support.v7.app.NotificationCompat;  import android.view.View;  import android.widget.Button;    public class MainActivity extends AppCompatActivity implements View.OnClickListener {      // идентификатор нашего уведомления      private int NOTIFICATION\_ID = 1010;      private NotificationManager notificationManager;    *@Override*      protected void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          setContentView(R.layout.activity\_main);            Button btnCreate = (Button) findViewById(R.id.btn\_create\_notification);          Button btnCancel = (Button) findViewById(R.id.btn\_cancel\_notification);            btnCreate.setOnClickListener(this);          btnCancel.setOnClickListener(this);        }      public void createNotification(String url) {          // инициализируем интерфейс NotificationCompat.Builder          NotificationCompat.Builder builder = new NotificationCompat.Builder(this);          builder.setSmallIcon(android.R.drawable.ic\_dialog\_alert);            // создаем Intent с действием на просмотр веб-страниц          // и явно указываем какую ссылку мы хотим открыть          Intent intent = new Intent(Intent.ACTION\_VIEW, Uri.parse(url));          PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, intent, 0);            // устанавливаем объект PendingIntent, большую иконку, заголовки и контент          builder.setContentIntent(pendingIntent);          builder.setLargeIcon(BitmapFactory.decodeResource(getResources(), R.mipmap.ic\_launcher));          builder.setContentTitle("Это заголовок");          builder.setContentText("Это контент нашего уведомления");          builder.setSubText("Действие: нажмите на уведомление, чтобы открыть сайт");            notificationManager = (NotificationManager) getSystemService(NOTIFICATION\_SERVICE);            // отображаем объект Notification в панели уведомлений          notificationManager.notify(NOTIFICATION\_ID, builder.build());      }        public void cancelNotification() {          // получаем из контекста сервис уведомлений и отменяем уведомление по его Id          notificationManager = (NotificationManager) getApplicationContext().getSystemService(Context.NOTIFICATION\_SERVICE);          notificationManager.cancel(NOTIFICATION\_ID);      }    *@Override*      public void onClick(View view) {          int id = view.getId();            switch (id) {              case R.id.btn\_create\_notification: {                  // задаем ссылку, которая будет                  // открываться по нажатию на уведомление                  createNotification("http://prologistic.com.ua");                  break;              }              case R.id.btn\_cancel\_notification: {                  cancelNotification();                  break;              }          }      }  } |

Как видите, нам не потребовалось никаких дополнительных разрешений в манифесте. Теперь просто запустим приложение и посмотрим результат:

[](http://prologistic.com.ua/wp-content/uploads/2016/06/res.jpg)Как видно на картинке выше, мы нажали на кнопку «Создать уведомление», потом оно появилось в панели уведомлений со значком Alert. Потом мы опустили «шторку уведомлений» и увидели скомпонованный объект Notification. После нажатия на уведомление мы перешли на указанный сайт.

Вам осталось лишь отменить уведомление: просто нажмите в приложении кнопку «Отменить» и NotificationManager уберет его по указанному id.