Логирование в Android

[](http://2.bp.blogspot.com/-a4Qhj-HQit0/UGNAXD5a2sI/AAAAAAAAAN8/xhfJfceAbtc/s1600/icon.png)

Важный этап любой разработки приложений – это логирование. Не существует серьезных проектов, которые были бы написаны хотя бы без одного бага. Чтобы отследить все перемещения и изменения объектов в приложении используется лог. Например: запросы и ответы от сервера, значения массивов и коллекций, вывод обработчиков ошибок и другое. Рассмотрим, как реализовать логирование в приложение так, чтобы это было удобно.

В Android существует свой класс для работы c логами: android.util.Log. Его мы и будем использовать для вывода информации в LogCat DDMS. Но мы его немного модифицируем для более удобного использования.

Во-первых, каждый раз, когда мы выводим в лог какую-нибудь информацию, затрачивается небольшое количество времени. Если выводить в лог очень часто, то это время может сказаться на производительности приложения. Поэтому, перед публикацией нашего приложения, необходимо все наши логи отключить. А если их достаточно много? В этой статье мы рассмотрим класс, позволяющий отключить все логи изменив всего один параметр.

Во-вторых, публикация приложения невозможна, если в манифесте приложения установлен парметр android:debuggable="true", который мы использовали для того, чтобы отследить какие-либо участки кода в приложении. В статье я приведу пример, как в зависимости от этого параметра, выводить или не выводить информацию в лог.

В-третьих, иногда необходимо переопределить вывод лога не в консоль, а в другой поток вывода. На примере данной статьи мы попробуем разработать приложение, которое будет выводить наш лог в файл на SD-карте.

**Разработка**

Создадим новый класс для логирования: LogSystem, с одним атрибутом класса, который будет отвечать за возможность вывода информации в лог.

[?](http://snowpard-android-ru.blogspot.ru/2012/09/android.html)

|  |
| --- |
| public final class LogSystem {   private static boolean mLoggingEnabled = true;  private LogSystem() {     }     public static void setDebugLogging(boolean enabled) {    mLoggingEnabled = enabled;   }  } |

Переопределим все методы вывода андроидовского класса Log:

public static int v(String tag, String msg) {

        int result = 0;

        if (mLoggingEnabled) {

         result = Log.v(tag, msg);

        }

        return result;

    }

 public static int v(String tag, String msg, Throwable tr) {

    int result = 0;

       if (mLoggingEnabled) {

        result = Log.v(tag, msg, tr);

       }

       return result;

 }

     public static int d(String tag, String msg) {

      int result = 0;

         if (mLoggingEnabled) {

          result = Log.d(tag, msg);

         }

         return result;

    }

    public static int d(String tag, String msg, Throwable tr) {

     int result = 0;

        if (mLoggingEnabled) {

         result = Log.d(tag, msg, tr);

        }

        return result;

    }

    public static int i(String tag, String msg) {

     int result = 0;

        if (mLoggingEnabled) {

         result = Log.i(tag, msg);

        }

        return result;

    }

    public static int i(String tag, String msg, Throwable tr) {

     int result = 0;

        if (mLoggingEnabled) {

         result = Log.i(tag, msg, tr);

        }

        return result;

    }

    public static int w(String tag, String msg) {

     int result = 0;

        if (mLoggingEnabled) {

         result = Log.w(tag, msg);

        }

        return result;

    }

    public static int w(String tag, String msg, Throwable tr) {

     int result = 0;

        if (mLoggingEnabled) {

         result = Log.w(tag, msg, tr);

        }

        return result;

    }

    public static int w(String tag, Throwable tr) {

     int result = 0;

        if (mLoggingEnabled) {

         result = Log.w(tag, tr);

        }

        return result;

    }

    public static int e(String tag, String msg) {

     int result = 0;

        if (mLoggingEnabled) {

         result = Log.e(tag, msg);

        }

        return result;

    }

    public static int e(String tag, String msg, Throwable tr) {

     int result = 0;

        if (mLoggingEnabled) {

         result = Log.e(tag, msg, tr);

        }

        return result;

    }

Мы разработали управляемый лог. Атрибут mLoggingEnabled указывает приложению, надо ли выводить сообщение или нет.

Для того чтобы использовать наш класс, достаточно только вызвать любой соответствующий статический метод.

Рассмотрим следующий пример. Мы хотим отследить весь жизненный цикл приложения.

|  |
| --- |
| public class MainActivity extends Activity {      public static final String TAG = MainActivity.class.getSimpleName();        @Override      public void onCreate(Bundle savedInstanceState) {          super.onCreate(savedInstanceState);          LogSystem.i(TAG, "onCreate()");  }     @Override   protected void onDestroy() {    super.onDestroy();    LogSystem.i(TAG, "onDestroy()");   }     @Override   protected void onPause() {    super.onPause();    LogSystem.i(TAG, "onPause()");   }     @Override   protected void onResume() {    super.onResume();    LogSystem.i(TAG, "onResume()");   }     @Override   protected void onStart() {    super.onStart();    LogSystem.i(TAG, "onStart()");   }     @Override   protected void onStop() {    super.onStop();    LogSystem.i(TAG, "onStop()");   }    } |

**Debuggable**

Для того чтобы выводить или не выводить информацию в лог в зависимости от значения аттрибута android:debuggable, необходимо в методе onCreate() главного Activity написать:

|  |
| --- |
| boolean isDebuggable = (0 != (getApplicationInfo().flags & ApplicationInfo.FLAG\_DEBUGGABLE));  LogSystem.setDebugLogging(isDebuggable); |

**Лог в файл**

Для логирования в файл на SD-карте, сначала необходимо добавитьpermission в манифест приложения:

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />

Теперь доработаем класс LogSystem, чтобы можно было выводить лог в файл.

private static final String PATH = "sdcard/log.dat";

|  |
| --- |
| public static int logInFile(String tag, String msg)   {    int result = 0;    File file = new File(PATH);    try {     if (!file.exists()) {        file.createNewFile();       }     String timeLog = new SimpleDateFormat("dd.MM.yy hh:mm:ss").format(new Date());     BufferedWriter bw = new BufferedWriter(new FileWriter(file,true));     bw.append(timeLog+" (" + tag + ")\t" + msg + "\n");     bw.close();     result = 1;    } catch (IOException e) {     e.printStackTrace();    }    return result;   } |

Мы добавили статический атрибут, который будет указывать путь к файлу нашего лога на SD-карте. И разработали метод для записи информации в этот файл.

Модернизируем наше приложение с учетом новых изменений. ЗаменимLogSystem.i(TAG, "onCreate()"); в методе onCreate() нашего Activity наLogSystem.logInFile(TAG, " onCreate ()");

Пример записи в логе будет следующий:

19.09.12 14:12:01 (MainActivity)              onCreate()

**Возможные доработки**

* Другой способ вывода в файл

Если мы хотим использовать статические методы v(), d(), e(), i(), w() для вывода сразу в файл, то можно изменить класс LogSystem следующим образом (на примере метода e()):

|  |
| --- |
| public static int logInFile(int priority, String tag, String msg)   {    int result = 0;    File file = new File(PATH);    try {     if (!file.exists()) {        file.createNewFile();       }     String timeLog = new SimpleDateFormat("dd.MM.yy hh:mm:ss").format(new Date());     BufferedWriter bw = new BufferedWriter(new FileWriter(file,true));     bw.append(priority + "\t" + timeLog + " (" + tag + ")\t" + msg + "\n");     bw.close();     result = 1;    } catch (IOException e) {     e.printStackTrace();    }    return result;   }  public static int e(String tag, String msg) {       int result = 0;          if (mLoggingEnabled) {           result = logInFile(Log.ERROR, tag, msg);          }          return result;      } |

* Усовершенствование методов вывода

Так же возможно усовершенствовать методы вывода, например, дополнительным параметром int placeForLog, который показывал бы, куда выводить лог

public static int e(String tag, String msg, int placeForLog) {

     int result = 0;

        if (mLoggingEnabled) {

  switch (placeForLog) {

   case 0:

           result = logInFile(Log.ERROR, tag, msg);

    break;

   case 1:

    result = Log.e(tag, msg);

    break;

        }

        return result;

    }

**Ссылки**

Исходные коды данного проекта можно скачать отсюда: [zip](http://www.snowpard.org/examples/article_b4/LogExample.zip)