android开发之应用Crash自动抓取Log_自动保存崩溃日志到本地

标签： android android开发 异常处理

2015-05-09 16:58 4040人阅读 评论(3) 收藏 举报

分类：

Android进阶（21）

应用发生crash之后要查看log，判断问题出在什么地方，可是一旦应用发布出去，就要想办法把用户的崩溃日志拿到分析。

所以要在发生crash之后抓取log，然后上传到服务器，方便开发者查看，现在都有很多第三方做这方面的服务，这里说下如何自己来实现。

其实原理很简单，应用出现异常后，会由默认的异常处理器来处理异常，

我们要做的就是把这个任务接管过来，自己处理异常，包括收集日志，保存到本地，然后上传到服务器。

下面是自己实现的异常处理类。

[java] view plain copy print ?

```java
public class CrashHandler implements UncaughtExceptionHandler {
    public static final String TAG = "CrashHandler";

    // 系统默认的UncaughtException处理类
    private Thread.UncaughtExceptionHandler mDefaultHandler;
    // CrashHandler实例
    private static CrashHandler INSTANCE = new CrashHandler();
    // 程序的Context对象
    private Context mContext;
    // 用来存储设备信息和异常信息
    private Map<String, String> infos = new HashMap<String, String>();

    // 用于格式化日期,作为日志文件名的一部分
    private DateFormat formatter = new SimpleDateFormat("yyyy-MM-dd_HH-mm-ss");
    private String nameString;

    /** 保证只有一个CrashHandler实例 */
    private CrashHandler() {
    }

    /** 获取CrashHandler实例 ,单例模式 */
    public static CrashHandler getInstance() {
        return INSTANCE;
    }

    /**
     * 初始化
     *
     * @param context
```

加载插

```
     */
    public void init(Context context) {
        mContext = context;
        // 获取系统默认的UncaughtException处理器
        mDefaultHandler = Thread.getDefaultUncaughtExceptionHandler();
        // 设置该CrashHandler为程序的默认处理器
        Thread.setDefaultUncaughtExceptionHandler(this);
        nameString = BmobUserManager.getInstance(mContext).getCurrentUserName();
    }

    /**
     * 当UncaughtException发生时会转入该函数来处理
     */
    @Override
    public void uncaughtException(Thread thread, Throwable ex) {
        if (!handleException(ex) && mDefaultHandler != null) {
            // 如果用户没有处理则让系统默认的异常处理器来处理
            mDefaultHandler.uncaughtException(thread, ex);
        } else {
            try {
                Thread.sleep(3000);
            } catch (InterruptedException e) {
                Log.e(TAG, "error : ", e);
            }
            // 退出程序
            android.os.Process.killProcess(android.os.Process.myPid());
            System.exit(1);
        }
    }

    /**
     * 自定义错误处理,收集错误信息 发送错误报告等操作均在此完成.
     *
     * @param ex
     * @return true:如果处理了该异常信息;否则返回false.
     */
    private boolean handleException(Throwable ex) {
        if (ex == null) {
            return false;
        }
        WonderMapApplication.getInstance().getSpUtil().setCrashLog(true);// 每次进入应用检查，是否有log，有则上传
        // 使用Toast来显示异常信息
        new Thread() {
            @Override
            public void run() {
                Looper.prepare();
                Toast.makeText(mContext, "很抱歉,程序出现异常,正在收集日志，即将退出", Toast.LENGTH_LONG)
                        .show();
                Looper.loop();
            }
        }.start();
        // 收集设备参数信息
        collectDeviceInfo(mContext);
        // 保存日志文件
```

```
84.             String fileName = saveCrashInfo2File(ex);
85.             return true;
86.         }
87.
88.         /**
89.          * 收集设备参数信息
90.          *
91.          * @param ctx
92.          */
93.         public void collectDeviceInfo(Context ctx) {
94.             try {
95.                 PackageManager pm = ctx.getPackageManager();
96.                 PackageInfo pi = pm.getPackageInfo(ctx.getPackageName(),
97.                         PackageManager.GET_ACTIVITIES);
98.                 if (pi != null) {
99.                     String versionName = pi.versionName == null ? "null"
100.                             : pi.versionName;
101.                     String versionCode = pi.versionCode + "";
102.                     infos.put("versionName", versionName);
103.                     infos.put("versionCode", versionCode);
104.                 }
105.             } catch (NameNotFoundException e) {
106.                 Log.e(TAG, "an error occured when collect package info", e);
107.             }
108.             Field[] fields = Build.class.getDeclaredFields();
109.             for (Field field : fields) {
110.                 try {
111.                     field.setAccessible(true);
112.                     infos.put(field.getName(), field.get(null).toString());
113.                     Log.d(TAG, field.getName() + " : " + field.get(null));
114.                 } catch (Exception e) {
115.                     Log.e(TAG, "an error occured when collect crash info", e);
116.                 }
117.             }
118.         }
119.
120.         /**
121.          * 保存错误信息到文件中
122.          *
123.          * @param ex
124.          * @return 返回文件名称,便于将文件传送到服务器
125.          */
126.         private String saveCrashInfo2File(Throwable ex) {
127.
128.             StringBuffer sb = new StringBuffer();
129.             for (Map.Entry<String, String> entry : infos.entrySet()) {
130.                 String key = entry.getKey();
131.                 String value = entry.getValue();
132.                 sb.append(key + "=" + value + "\n");
133.             }
134.
135.             Writer writer = new StringWriter();
136.             PrintWriter printWriter = new PrintWriter(writer);
137.             ex.printStackTrace(printWriter);
138.             Throwable cause = ex.getCause();
139.             while (cause != null) {
```

```
            cause.printStackTrace(printWriter);
            cause = cause.getCause();
        }
        printWriter.close();
        String result = writer.toString();
        L.d(WModel.CrashUpload, result);
        sb.append(result);
        try {
            long timestamp = System.currentTimeMillis();
            String time = formatter.format(new Date());
            String fileName = nameString + "-" + time + "-" + timestamp
                    + ".log";
            if (Environment.getExternalStorageState().equals(
                    Environment.MEDIA_MOUNTED)) {
                String path = WMapConstants.CrashLogDir;
                File dir = new File(path);
                if (!dir.exists()) {
                    dir.mkdirs();
                }
                FileOutputStream fos = new FileOutputStream(path + fileName);
                fos.write(sb.toString().getBytes());
                fos.close();
            }
            return fileName;
        } catch (Exception e) {
            Log.e(TAG, "an error occured while writing file...", e);
        }
        return null;
    }

}
```

使用方式如下：

在Application的onCreate中加上下面

```
[java] view plain  copy  print  ?
CrashHandler crashHandler = CrashHandler.getInstance();
crashHandler.init(this);
```

这样一来，应用发生crash，自动保存log到本地了。

其实这样还有一个好处，就是不用在logcat里面翻来翻去找日志，直接到本地文件夹打开看就是了。