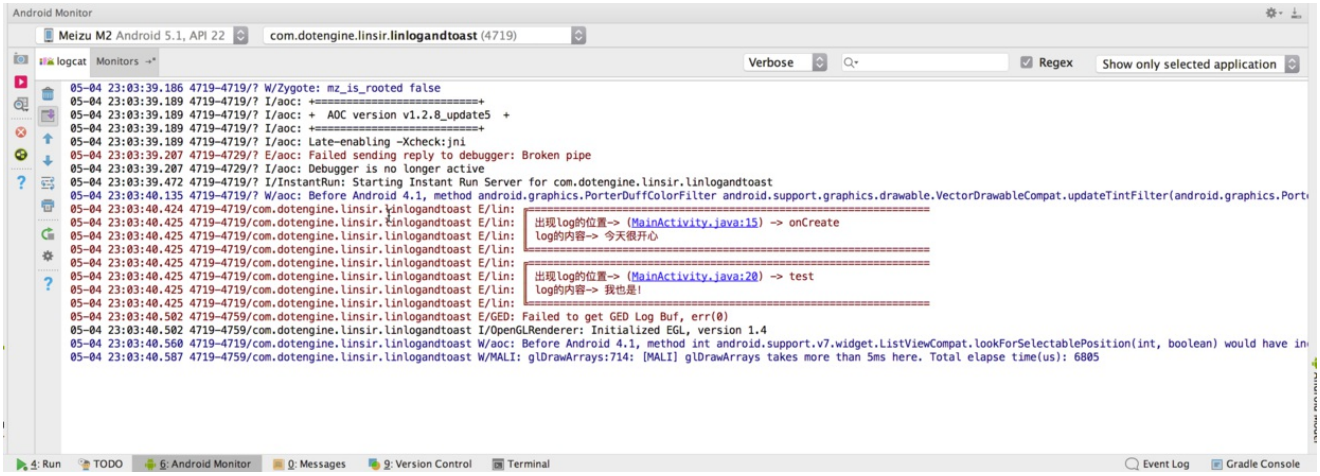
> 没错，是我闲的无聊写了一个用来Log和Toast的库，这个库目前能实现将Log和Toast变得更简单，并且可以实现Log的快速定位，以及Debug和Release的切换，可以非常简单的配置在Realease情况下不打印Log。 当然这些都还非常基础~也准备在接下来的时间里，封装一些网络操作的框架，还有一些BaseAdapter的框架。

效果图：



配置方法

```
compile 'com.linsir:linLog:1.0.0'
```

经过以上的配置，已经可以正常的使用了，如果我们想配置的更加轻便的话，是可以这样的：

```
public class App extends Application {
    public static final boolean DEBUG = BuildConfig.DEBUG;
    @Override public void onCreate() {
        super.onCreate();
        LinToast.init(getApplicationContext());
        LinLog.init(DEBUG, "lin");
}
```

```
defaultConfig {
        buildConfigField("boolean", "LOG", "true")
    }
    buildTypes {
        release {
            buildConfigField("boolean", "LOG", "false")
        }
    }
```

好了，以上便完全配置完成了~

使用

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        LinLog.lLog("今天很开心");
        LinToast.showToast("啦啦啦啦~~~");
        test();
    }

    private void test(){
        LinLog.lLog("我也是！");
    }
}
```

源码

源码下载

整个代码也不超过100行，就是进行了简单的封装，以及 `Thread.currentThread().getStackTrace()` 这样一个方法，便可以获取到打印Log的位置，以及一些基本的信息，然后展示出来就可以了。

```
/**
 *  Created by linSir
 *  date at 2017/5/3.
 *  describe: 一个专门用来展示log的工具类
 */

public class LinLog {

    private static boolean Debug = true;
    private static String Tag = "null";

    public static void init(boolean debug, String tag) {
        LinLog.Debug = debug;
        LinLog.Tag = tag;
    }

    public static void lLog(String text){
        if (!Debug){
            return;
        }
        String dividingLine = "╔═══════════════════════════════════════════════════════════════════════\n";
        String dividingLine2 = "╚═══════════════════════════════════════════════════════════════════════\n";
        Log.e(Tag,dividingLine);
        setUpContent(text);
        Log.e(Tag,dividingLine2);
    }

    private static void setUpContent(String content) {
        StackTraceElement targetStackTraceElement = getStackTraceElement();
        Log.e(Tag, "╟ 出现log的位置-> (" + targetStackTraceElement.getFileName() + ":"
                + targetStackTraceElement.getLineNumber() + ")" + " -> " + targetStackTraceElement.
        Log.e(Tag, "╟ log的内容-> "+content);
    }

    private static StackTraceElement getStackTraceElement(){
        StackTraceElement targetStackTrace = null;
        boolean shouldTrace = false;
        StackTraceElement[] stackTrace = Thread.currentThread().getStackTrace();
        for (StackTraceElement stackTraceElement : stackTrace) {
            boolean isLogMethod = stackTraceElement.getClassName().equals(LinLog.class.getName());
            if (shouldTrace && !isLogMethod) {
                targetStackTrace = stackTraceElement;
                break;
            }
            shouldTrace = isLogMethod;
        }
        return targetStackTrace;
    }
}
```

```
public class LinToast {

    private static LinToast linToast;
    private static Context mContext;
    private static Toast mToast;

    public static void init(Context context) {
        mContext = context.getApplicationContext();
        mToast = Toast.makeText(context,"",Toast.LENGTH_SHORT);
    }

    public static void showToast(String txt) {
        mToast.setText(txt);
        mToast.setDuration(Toast.LENGTH_SHORT);
        mToast.show();
    }
}
```

总结

其实整体流程还是挺简单的，就是新建一个Library，然后写一下逻辑，写完之后，上传到bintray.com，然后就可以了。然后说句题外话，非常欢迎大家有事没事，引用一下这个库，增加一下下载量，也欢迎大家上我的github提issue或者star,follow的。 总体感觉，写一个这样的库还是挺有意义的吧，可以把项目中经常用到的工具类封装一下，日后用着也方便，大家可以点开源码看一下，自己也尝试着写一下。之后我也会持续更新一些网络框架的封装，还有BaseAdapter的封装~如果大家在写类似东西的时候，遇到了问题也欢迎和我讨论。

欢迎大家点□~~ GitHub地址，欢迎star，follow~~