Android常用工具类

工具类 utils 工具类 快速开发

1.日志工具类L.java

```
package com.moyu.utils;
import android.util.Log;
/**
 * Log统一管理类
 */
public class L
{
    private L()
    {
        /* cannot be instantiated */
        throw new UnsupportedOperationException("cannot be
instantiated");
    }

    public static boolean isDebug = true;// 是否需要打印bug，可以在
application的onCreate函数里面初始化
    private static final String TAG = "way";

    // 下面四个是默认tag的函数
    public static void i(String msg)
    {
        if (isDebug)
            Log.i(TAG, msg);
    }

    public static void d(String msg)
    {
        if (isDebug)
            Log.d(TAG, msg);
    }

    public static void e(String msg)
    {
        if (isDebug)
            Log.e(TAG, msg);
    }

    public static void v(String msg)
```

```
        {
            if (isDebug)
                Log.v(TAG, msg);
        }

        // 下面是传入自定义tag的函数
        public static void i(String tag, String msg)
        {
            if (isDebug)
                Log.i(tag, msg);
        }

        public static void d(String tag, String msg)
        {
            if (isDebug)
                Log.i(tag, msg);
        }

        public static void e(String tag, String msg)
        {
            if (isDebug)
                Log.i(tag, msg);
        }

        public static void v(String tag, String msg)
        {
            if (isDebug)
                Log.i(tag, msg);
        }
    }
```

2.Toast统一管理类

```
    package com.moyu.utils;

    import android.content.Context;
    import android.widget.Toast;

    /**
     * Toast统一管理类
     *
     */
    public class T
    {
        private T()
        {
            /* cannot be instantiated */
```

```
        throw new UnsupportedOperationException("cannot be
instantiated");
    }

    public static boolean isShow = true;

    /**
     * 短时间显示Toast
     *
     * @param context
     * @param message
     */
    public static void showShort(Context context, CharSequence
message)
    {
        if (isShow)
            Toast.makeText(context, message,
Toast.LENGTH_SHORT).show();
    }

    /**
     * 短时间显示Toast
     *
     * @param context
     * @param message
     */
    public static void showShort(Context context, int message)
    {
        if (isShow)
            Toast.makeText(context, message,
Toast.LENGTH_SHORT).show();
    }

    /**
     * 长时间显示Toast
     *
     * @param context
     * @param message
     */
    public static void showLong(Context context, CharSequence
message)
    {
        if (isShow)
            Toast.makeText(context, message,
Toast.LENGTH_LONG).show();
    }

    /**
     * 长时间显示Toast
```

```
     *
     * @param context
     * @param message
     */
    public static void showLong(Context context, int message)
    {
        if (isShow)
            Toast.makeText(context, message,
Toast.LENGTH_LONG).show();
    }

    /**
     * 自定义显示Toast时间
     *
     * @param context
     * @param message
     * @param duration
     */
    public static void show(Context context, CharSequence message,
int duration)
    {
        if (isShow)
            Toast.makeText(context, message, duration).show();
    }

    /**
     * 自定义显示Toast时间
     *
     * @param context
     * @param message
     * @param duration
     */
    public static void show(Context context, int message, int
duration)
    {
        if (isShow)
            Toast.makeText(context, message, duration).show();
    }

}
```

3.SharedPreferences封装类SPUtils

```
package com.moyu.utils;

import java.lang.reflect.InvocationTargetException;
import java.lang.reflect.Method;
```

```
import java.util.Map;

import android.content.Context;
import android.content.SharedPreferences;

public class SPUtils
{
    /**
     * 保存在手机里面的文件名
     */
    public static final String FILE_NAME = "share_data";

    /**
     * 保存数据的方法，我们需要拿到保存数据的具体类型，然后根据类型调用不同的保存方法
     *
     * @param context
     * @param key
     * @param object
     */
    public static void put(Context context, String key, Object object)
    {

        SharedPreferences sp = context.getSharedPreferences(FILE_NAME,
                Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sp.edit();
         //instanceof用来判断内存中实际对象A是不是B类型
        //出现这种情况经常是需要强制转换的时候
        if (object instanceof String)
        {
            editor.putString(key, (String) object);
        } else if (object instanceof Integer)
        {
            editor.putInt(key, (Integer) object);
        } else if (object instanceof Boolean)
        {
            editor.putBoolean(key, (Boolean) object);
        } else if (object instanceof Float)
        {
            editor.putFloat(key, (Float) object);
        } else if (object instanceof Long)
        {
            editor.putLong(key, (Long) object);
        } else
        {
            editor.putString(key, object.toString());
        }
```

```
        SharedPreferencesCompat.apply(editor);
    }

    /**
     * 得到保存数据的方法，我们根据默认值得到保存的数据的具体类型，然后调用相对于的方法获取值
     *
     * @param context
     * @param key
     * @param defaultObject
     * @return
     */
    public static Object get(Context context, String key, Object defaultObject)
    {
        SharedPreferences sp = context.getSharedPreferences(FILE_NAME,
                Context.MODE_PRIVATE);

        if (defaultObject instanceof String)
        {
            return sp.getString(key, (String) defaultObject);
        } else if (defaultObject instanceof Integer)
        {
            return sp.getInt(key, (Integer) defaultObject);
        } else if (defaultObject instanceof Boolean)
        {
            return sp.getBoolean(key, (Boolean) defaultObject);
        } else if (defaultObject instanceof Float)
        {
            return sp.getFloat(key, (Float) defaultObject);
        } else if (defaultObject instanceof Long)
        {
            return sp.getLong(key, (Long) defaultObject);
        }

        return null;
    }

    /**
     * 移除某个key值已经对应的值
     * @param context
     * @param key
     */
    public static void remove(Context context, String key)
    {
        SharedPreferences sp = context.getSharedPreferences(FILE_NAME,
```

```
                Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sp.edit();
        editor.remove(key);
        SharedPreferencesCompat.apply(editor);
    }

    /**
     * 清除所有数据
     * @param context
     */
    public static void clear(Context context)
    {
        SharedPreferences sp =
context.getSharedPreferences(FILE_NAME,
                Context.MODE_PRIVATE);
        SharedPreferences.Editor editor = sp.edit();
        editor.clear();
        SharedPreferencesCompat.apply(editor);
    }

    /**
     * 查询某个key是否已经存在
     * @param context
     * @param key
     * @return
     */
    public static boolean contains(Context context, String key)
    {
        SharedPreferences sp =
context.getSharedPreferences(FILE_NAME,
                Context.MODE_PRIVATE);
        return sp.contains(key);
    }

    /**
     * 返回所有的键值对
     *
     * @param context
     * @return
     */
    public static Map<String, ?> getAll(Context context)
    {
        SharedPreferences sp =
context.getSharedPreferences(FILE_NAME,
                Context.MODE_PRIVATE);
        return sp.getAll();
    }

    /**
```

```
     * 创建一个解决SharedPreferencesCompat.apply方法的一个兼容类
     *
     * @author zhy
     *
     */
    private static class SharedPreferencesCompat
    {
        private static final Method sApplyMethod =
findApplyMethod();

        /**
         * 反射查找apply的方法
         *
         * @return
         */
        @SuppressWarnings({ "unchecked", "rawtypes" })
        private static Method findApplyMethod()
        {
            try
            {
                Class clz = SharedPreferences.Editor.class;
                return clz.getMethod("apply");
            } catch (NoSuchMethodException e)
            {
            }

            return null;
        }
        /**
         * 如果找到则使用apply执行，否则使用commit
         *
         * @param editor
         */
        public static void apply(SharedPreferences.Editor editor)
        {
            try
            {
                if (sApplyMethod != null)
                {
                    sApplyMethod.invoke(editor);
                    return;
                }
            } catch (IllegalArgumentException e)
            {
            } catch (IllegalAccessException e)
            {
            } catch (InvocationTargetException e)
            {
            }
```

```
                editor.commit();
            }
        }
    }
```

对SharedPreference的使用做了建议的封装，对外公布出put，get，remove，clear等等方法；
注意一点，里面所有的commit操作使用了SharedPreferencesCompat.apply进行了替代，目的是尽可能的使用apply代替commit。首先说下为什么，因为commit方法是同步的，并且我们很多时候的commit操作都是UI线程中，毕竟是IO操作，尽可能异步；所以我们使用apply进行替代，apply异步的进行写入。

4.单位转换类DensityUtils

```
package com.moyu.utils;

import android.content.Context;
import android.util.TypedValue;

/**
 * 常用单位转换的辅助类
 */
public class DensityUtils
{
    private DensityUtils()
    {
        /* cannot be instantiated */
        throw new UnsupportedOperationException("cannot be
instantiated");
    }

    /**
     * dp转px
     *
     * @param context
     * @param val
     * @return
     */
    public static int dp2px(Context context, float dpVal)
    {
        return (int)
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP,
                dpVal,
context.getResources().getDisplayMetrics());
    }
```

```
    /**
     * sp转px
     *
     * @param context
     * @param val
     * @return
     */
    public static int sp2px(Context context, float spVal)
    {
        return (int)
TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_SP,
                spVal,
context.getResources().getDisplayMetrics());
    }

    /**
     * px转dp
     *
     * @param context
     * @param pxVal
     * @return
     */
    public static float px2dp(Context context, float pxVal)
    {
        final float scale =
context.getResources().getDisplayMetrics().density;
        return (pxVal / scale);
    }

    /**
     * px转sp
     *
     * @param fontScale
     * @param pxVal
     * @return
     */
    public static float px2sp(Context context, float pxVal)
    {
        return (pxVal /
context.getResources().getDisplayMetrics().scaledDensity);
    }
}
```

5.SD卡相关辅助类 SDCardUtils

```
package com.moyu.utils;
```

```
import java.io.File;

import android.os.Environment;
import android.os.StatFs;

/**
 * SD卡相关的辅助类
 */
public class SDCardUtils
{
    private SDCardUtils()
    {
        /* cannot be instantiated */
        throw new UnsupportedOperationException("cannot be
instantiated");
    }

    /**
     * 判断SDCard是否可用
     *
     * @return
     */
    public static boolean isSDCardEnable()
    {
        return Environment.getExternalStorageState().equals(
                Environment.MEDIA_MOUNTED);
    }

    /**
     * 获取SD卡路径
     *
     * @return
     */
    public static String getSDCardPath()
    {
        return
Environment.getExternalStorageDirectory().getAbsolutePath()
                + File.separator;
    }

    /**
     * 获取SD卡的剩余容量 单位byte
     *
     * @return
     */
    public static long getSDCardAllSize()
    {
        if (isSDCardEnable())
        {
```

```
            StatFs stat = new StatFs(getSDCardPath());
            // 获取空闲的数据块的数量
            long availableBlocks = (long)
stat.getAvailableBlocks() - 4;
            // 获取单个数据块的大小（byte）
            long freeBlocks = stat.getAvailableBlocks();
            return freeBlocks * availableBlocks;
        }
        return 0;
    }

    /**
     * 获取指定路径所在空间的剩余可用容量字节数，单位byte
     *
     * @param filePath
     * @return 容量字节 SDCard可用空间，内部存储可用空间
     */
    public static long getFreeBytes(String filePath)
    {
        // 如果是sd卡的下的路径，则获取sd卡可用容量
        if (filePath.startsWith(getSDCardPath()))
        {
            filePath = getSDCardPath();
        } else
        {// 如果是内部存储的路径，则获取内存存储的可用容量
            filePath =
Environment.getDataDirectory().getAbsolutePath();
        }
        StatFs stat = new StatFs(filePath);
        long availableBlocks = (long) stat.getAvailableBlocks() -
4;
        return stat.getBlockSize() * availableBlocks;
    }

    /**
     * 获取系统存储路径
     *
     * @return
     */
    public static String getRootDirectoryPath()
    {
        return Environment.getRootDirectory().getAbsolutePath();
    }
}
```

6.屏幕相关辅助类 ScreenUtils

```
package com.zhy.utils;

import android.app.Activity;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.Rect;
import android.util.DisplayMetrics;
import android.view.View;
import android.view.WindowManager;

/**
 * 获得屏幕相关的辅助类
 */
public class ScreenUtils
{
    private ScreenUtils()
    {
        /* cannot be instantiated */
        throw new UnsupportedOperationException("cannot be
instantiated");
    }

    /**
     * 获得屏幕高度
     *
     * @param context
     * @return
     */
    public static int getScreenWidth(Context context)
    {
        WindowManager wm = (WindowManager) context
                .getSystemService(Context.WINDOW_SERVICE);
        DisplayMetrics outMetrics = new DisplayMetrics();
        wm.getDefaultDisplay().getMetrics(outMetrics);
        return outMetrics.widthPixels;
    }

    /**
     * 获得屏幕宽度
     *
     * @param context
     * @return
     */
    public static int getScreenHeight(Context context)
    {
        WindowManager wm = (WindowManager) context
                .getSystemService(Context.WINDOW_SERVICE);
        DisplayMetrics outMetrics = new DisplayMetrics();
```

```
        wm.getDefaultDisplay().getMetrics(outMetrics);
        return outMetrics.heightPixels;
    }

    /**
     * 获得状态栏的高度
     *
     * @param context
     * @return
     */
    public static int getStatusHeight(Context context)
    {

        int statusHeight = -1;
        try
        {
            Class<?> clazz =
Class.forName("com.android.internal.R$dimen");
            Object object = clazz.newInstance();
            int height =
Integer.parseInt(clazz.getField("status_bar_height")
                    .get(object).toString());
            statusHeight =
context.getResources().getDimensionPixelSize(height);
        } catch (Exception e)
        {
            e.printStackTrace();
        }
        return statusHeight;
    }

    /**
     * 获取当前屏幕截图，包含状态栏
     *
     * @param activity
     * @return
     */
    public static Bitmap snapShotWithStatusBar(Activity activity)
    {
        View view = activity.getWindow().getDecorView();
        view.setDrawingCacheEnabled(true);
        view.buildDrawingCache();
        Bitmap bmp = view.getDrawingCache();
        int width = getScreenWidth(activity);
        int height = getScreenHeight(activity);
        Bitmap bp = null;
        bp = Bitmap.createBitmap(bmp, 0, 0, width, height);
        view.destroyDrawingCache();
        return bp;
```

```
    }

    /**
     * 获取当前屏幕截图，不包含状态栏
     *
     * @param activity
     * @return
     */
    public static Bitmap snapShotWithoutStatusBar(Activity
activity)
    {
        View view = activity.getWindow().getDecorView();
        view.setDrawingCacheEnabled(true);
        view.buildDrawingCache();
        Bitmap bmp = view.getDrawingCache();
        Rect frame = new Rect();

activity.getWindow().getDecorView().getWindowVisibleDisplayFrame(f
rame);
        int statusBarHeight = frame.top;

        int width = getScreenWidth(activity);
        int height = getScreenHeight(activity);
        Bitmap bp = null;
        bp = Bitmap.createBitmap(bmp, 0, statusBarHeight, width,
height
                - statusBarHeight);
        view.destroyDrawingCache();
        return bp;
    }
}
```

7.App相关辅助类

```
package com.zhy.utils;

import android.content.Context;
import android.content.pm.PackageInfo;
import android.content.pm.PackageManager;
import android.content.pm.PackageManager.NameNotFoundException;

/**
 * 跟App相关的辅助类
 *
 *
 *
```

```
 */
public class AppUtils
{

    private AppUtils()
    {
        /* cannot be instantiated */
        throw new UnsupportedOperationException("cannot be
instantiated");

    }

    /**
     * 获取应用程序名称
     */
    public static String getAppName(Context context)
    {
        try
        {
            PackageManager packageManager =
context.getPackageManager();
            PackageInfo packageInfo =
packageManager.getPackageInfo(
                    context.getPackageName(), 0);
            int labelRes = packageInfo.applicationInfo.labelRes;
            return context.getResources().getString(labelRes);
        } catch (NameNotFoundException e)
        {
            e.printStackTrace();
        }
        return null;
    }

    /**
     * [获取应用程序版本名称信息]
     *
     * @param context
     * @return 当前应用的版本名称
     */
    public static String getVersionName(Context context)
    {
        try
        {
            PackageManager packageManager =
context.getPackageManager();
            PackageInfo packageInfo =
packageManager.getPackageInfo(
                    context.getPackageName(), 0);
            return packageInfo.versionName;
```

```
            } catch (NameNotFoundException e)
            {
                e.printStackTrace();
            }
            return null;
        }
    }
```

8.软键盘相关辅助类KeyBoardUtils

```
package com.zhy.utils;

import android.content.Context;
import android.view.inputmethod.InputMethodManager;
import android.widget.EditText;

/**
 * 打开或关闭软键盘
 *
 * @author zhy
 *
 */
public class KeyBoardUtils
{
    /**
     * 打开软键盘
     *
     * @param mEditText 输入框
     * @param mContext 上下文
     */
    public static void openKeybord(EditText mEditText, Context
mContext)
    {
        InputMethodManager imm = (InputMethodManager) mContext
                .getSystemService(Context.INPUT_METHOD_SERVICE);
        imm.showSoftInput(mEditText,
InputMethodManager.RESULT_SHOWN);
        imm.toggleSoftInput(InputMethodManager.SHOW_FORCED,
                InputMethodManager.HIDE_IMPLICIT_ONLY);
    }

    /**
     * 关闭软键盘
     *
     * @param mEditText  输入框
     * @param mContext 上下文
     */
    public static void closeKeybord(EditText mEditText, Context
mContext)
    {
        InputMethodManager imm = (InputMethodManager) mContext
                .getSystemService(Context.INPUT_METHOD_SERVICE);

        imm.hideSoftInputFromWindow(mEditText.getWindowToken(),
0);
    }
}
```

9.网络相关辅助类 NetUtils

```
package com.zhy.utils;

import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;

/**
 * 跟网络相关的工具类
 *
 *
 *
 */
public class NetUtils
{
    private NetUtils()
    {
        /* cannot be instantiated */
        throw new UnsupportedOperationException("cannot be
instantiated");
    }

    /**
     * 判断网络是否连接
     *
     * @param context
     * @return
     */
    public static boolean isConnected(Context context)
    {

        ConnectivityManager connectivity = (ConnectivityManager)
context
                .getSystemService(Context.CONNECTIVITY_SERVICE);

        if (null != connectivity)
        {

            NetworkInfo info =
connectivity.getActiveNetworkInfo();
            if (null != info && info.isConnected())
            {
                if (info.getState() ==
```

```
NetworkInfo.State.CONNECTED)
                {
                    return true;
                }
            }
        }
        return false;
    }

    /**
     * 判断是否是wifi连接
     */
    public static boolean isWifi(Context context)
    {
        ConnectivityManager cm = (ConnectivityManager) context
                .getSystemService(Context.CONNECTIVITY_SERVICE);

        if (cm == null)
            return false;
        return cm.getActiveNetworkInfo().getType() ==
ConnectivityManager.TYPE_WIFI;

    }

    /**
     * 打开网络设置界面(请检查网络)
     */
    public static void openSetting(Activity activity)
    {
        Intent intent = new Intent("/");
        ComponentName cm = new
ComponentName("com.android.settings",
                "com.android.settings.WirelessSettings");
        intent.setComponent(cm);
        intent.setAction("android.intent.action.VIEW");
        activity.startActivityForResult(intent, 0);
    }

}
```

10.Http相关辅助类 HttpUtils (主要基于HttpURLConnection)

```
package com.zhy.utils;

import java.io.BufferedReader;
import java.io.ByteArrayOutputStream;
import java.io.IOException;
```

```
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.HttpURLConnection;
import java.net.URL;

/**
 * Http请求的工具类
 *
 * @author zhy
 *
 */
public class HttpUtils
{

    private static final int TIMEOUT_IN_MILLIONS = 5000;

    public interface CallBack
    {
        void onRequestComplete(String result);
    }


    /**
     * 异步的Get请求
     *
     * @param urlStr
     * @param callBack
     */
    public static void doGetAsyn(final String urlStr, final
CallBack callBack)
    {
        new Thread()
        {
            public void run()
            {
                try
                {
                    String result = doGet(urlStr);
                    if (callBack != null)
                    {
                        callBack.onRequestComplete(result);
                    }
                } catch (Exception e)
                {
                    e.printStackTrace();
                }

            };
```

```
        }.start();
    }

    /**
     * 异步的Post请求
     * @param urlStr
     * @param params
     * @param callBack
     * @throws Exception
     */
    public static void doPostAsyn(final String urlStr, final
String params,
            final CallBack callBack) throws Exception
    {
        new Thread()
        {
            public void run()
            {
                try
                {
                    String result = doPost(urlStr, params);
                    if (callBack != null)
                    {
                        callBack.onRequestComplete(result);
                    }
                } catch (Exception e)
                {
                    e.printStackTrace();
                }

            };
        }.start();

    }

    /**
     * Get请求，获得返回数据
     *
     * @param urlStr
     * @return
     * @throws Exception
     */
    public static String doGet(String urlStr)
    {
        URL url = null;
        HttpURLConnection conn = null;
        InputStream is = null;
        ByteArrayOutputStream baos = null;
        try
```

```
            {
                url = new URL(urlStr);
                conn = (HttpURLConnection) url.openConnection();
                conn.setReadTimeout(TIMEOUT_IN_MILLIONS);
                conn.setConnectTimeout(TIMEOUT_IN_MILLIONS);
                conn.setRequestMethod("GET");
                conn.setRequestProperty("accept", "*/*");
                conn.setRequestProperty("connection", "Keep-Alive");
                if (conn.getResponseCode() == 200)
                {
                    is = conn.getInputStream();
                    baos = new ByteArrayOutputStream();
                    int len = -1;
                    byte[] buf = new byte[128];

                    while ((len = is.read(buf)) != -1)
                    {
                        baos.write(buf, 0, len);
                    }
                    baos.flush();
                    return baos.toString();
                } else
                {
                    throw new RuntimeException(" responseCode is not
    200 ... ");
                }

            } catch (Exception e)
            {
                e.printStackTrace();
            } finally
            {
                try
                {
                    if (is != null)
                        is.close();
                } catch (IOException e)
                {
                }
                try
                {
                    if (baos != null)
                        baos.close();
                } catch (IOException e)
                {
                }
                conn.disconnect();
            }
```

```
            return null ;

        }

        /**
         * 向指定 URL 发送POST方法的请求
         *
         * @param url
         *             发送请求的 URL
         * @param param
         *             请求参数，请求参数应该是 name1=value1&name2=value2 的
    形式。
         * @return 所代表远程资源的响应结果
         * @throws Exception
         */
        public static String doPost(String url, String param)
        {
            PrintWriter out = null;
            BufferedReader in = null;
            String result = "";
            try
            {
                URL realUrl = new URL(url);
                // 打开和URL之间的连接
                HttpURLConnection conn = (HttpURLConnection) realUrl
                        .openConnection();
                // 设置通用的请求属性
                conn.setRequestProperty("accept", "*/*");
                conn.setRequestProperty("connection", "Keep-Alive");
                conn.setRequestMethod("POST");
                conn.setRequestProperty("Content-Type",
                        "application/x-www-form-urlencoded");
                conn.setRequestProperty("charset", "utf-8");
                conn.setUseCaches(false);
                // 发送POST请求必须设置如下两行
                conn.setDoOutput(true);
                conn.setDoInput(true);
                conn.setReadTimeout(TIMEOUT_IN_MILLIONS);
                conn.setConnectTimeout(TIMEOUT_IN_MILLIONS);

                if (param != null && !param.trim().equals(""))
                {
                    // 获取URLConnection对象对应的输出流
                    out = new PrintWriter(conn.getOutputStream());
                    // 发送请求参数
                    out.print(param);
                    // flush输出流的缓冲
                    out.flush();
                }
```

```
            // 定义BufferedReader输入流来读取URL的响应
            in = new BufferedReader(
                    new InputStreamReader(conn.getInputStream()));
            String line;
            while ((line = in.readLine()) != null)
            {
                result += line;
            }
        } catch (Exception e)
        {
            e.printStackTrace();
        }
        // 使用finally块来关闭输出流、输入流
        finally
        {
            try
            {
                if (out != null)
                {
                    out.close();
                }
                if (in != null)
                {
                    in.close();
                }
            } catch (IOException ex)
            {
                ex.printStackTrace();
            }
        }
        return result;
    }
}
```

11.Logger日志工具管理类

```
package org.iti.eyescare.util;

import java.util.Hashtable;

import android.util.Log;

public class MyLogger {

    private final static String LOG_TAG = "APP_LOG_TAG";
    private final static boolean APP_LOG_OUTPUT = true;
    private static Hashtable<String, MyLogger> sLoggerTable;
```

```
    static {
        sLoggerTable = new Hashtable<String, MyLogger>();
    }

    private String mClassName;

    public static MyLogger getLogger(String className) {
        MyLogger classLogger = (MyLogger)
sLoggerTable.get(className);
        if (classLogger == null) {
            classLogger = new MyLogger(className);
            sLoggerTable.put(className, classLogger);
        }
        return classLogger;
    }

    private MyLogger(String name) {
        mClassName = name;
    }

    public void v(String log) {
        if (APP_LOG_OUTPUT) {
            Log.v(LOG_TAG, "{Thread:" +
Thread.currentThread().getName() + "}"
                    + "[" + mClassName + ":] " + log);
        }
    }

    public void d(String log) {
        if (APP_LOG_OUTPUT) {
            Log.d(LOG_TAG, "{Thread:" +
Thread.currentThread().getName() + "}"
                    + "[" + mClassName + ":] " + log);
        }
    }

    public void i(String log) {
        if (APP_LOG_OUTPUT) {
            Log.i(LOG_TAG, "{Thread:" +
Thread.currentThread().getName() + "}"
                    + "[" + mClassName + ":] " + log);
        }
    }

    public void i(String log, Throwable tr) {
        if (APP_LOG_OUTPUT) {
            Log.i(LOG_TAG,
                    "{Thread:" + Thread.currentThread().getName()
```

```
    + "}" + "["
                                + mClassName + ":] " + log + "\n"
                                + Log.getStackTraceString(tr));
            }
        }

        public void w(String log) {
            if (APP_LOG_OUTPUT) {
                Log.w(LOG_TAG, "{Thread:" +
    Thread.currentThread().getName() + "}"
                        + "[" + mClassName + ":] " + log);
            }
        }

        public void w(String log, Throwable tr) {
            if (APP_LOG_OUTPUT) {
                Log.w(LOG_TAG,
                        "{Thread:" + Thread.currentThread().getName()
    + "}" + "["
                                + mClassName + ":] " + log + "\n"
                                + Log.getStackTraceString(tr));
            }
        }

        public void e(String log) {
            if (APP_LOG_OUTPUT)
                Log.e(LOG_TAG, "{Thread:" +
    Thread.currentThread().getName() + "}"
                        + "[" + mClassName + ":] " + log);
        }

        public void e(String log, Throwable tr) {
            if (APP_LOG_OUTPUT)
                Log.e(LOG_TAG,
                        "{Thread:" + Thread.currentThread().getName()
    + "}" + "["
                                + mClassName + ":] " + log + "\n"
                                + Log.getStackTraceString(tr));
        }

    }
```

12.应用数据清除管理器DataCleanManager

```
    package org.iti.eyescare.util;

    import java.io.File;
```

```
import java.math.BigDecimal;

import android.content.Context;
import android.os.Environment;
import android.text.TextUtils;

/**
 * 本应用数据清除管理器
 *
 * @author Tailyou
 *
 */
public class DataCleanManager {

    public static String getTotalCacheSize(Context context) throws
Exception {
        long cacheSize = getFolderSize(context.getCacheDir());
        if (Environment.getExternalStorageState().equals(
                Environment.MEDIA_MOUNTED)) {
            cacheSize +=
getFolderSize(context.getExternalCacheDir());
        }
        return getFormatSize(cacheSize);
    }

    public static void clearAllCache(Context context) {
        deleteDir(context.getCacheDir());
        if (Environment.getExternalStorageState().equals(
                Environment.MEDIA_MOUNTED)) {
            deleteDir(context.getExternalCacheDir());
        }
    }

    private static boolean deleteDir(File dir) {
        if (dir != null && dir.isDirectory()) {
            String[] children = dir.list();
            for (int i = 0; i < children.length; i++) {
                boolean success = deleteDir(new File(dir,
children[i]));
                if (!success) {
                    return false;
                }
            }
        }
        return dir.delete();
    }

    /**
     * * 清除本应用内部缓存(/data/data/com.xxx.xxx/cache) * *
```

```
     *
     * @param context
     */
    public static void cleanInternalCache(Context context) {
        deleteFilesByDirectory(context.getCacheDir());
    }

    /**
     * * 清除本应用所有数据库(/data/data/com.xxx.xxx/databases) * *
     *
     * @param context
     */
    public static void cleanDatabases(Context context) {
        deleteFilesByDirectory(new File("/data/data/"
                + context.getPackageName() + "/databases"));
    }

    /**
     * * 按名字清除本应用数据库 * *
     *
     * @param context
     * @param dbName
     */
    public static void cleanDatabaseByName(Context context, String
dbName) {
        context.deleteDatabase(dbName);
    }

    /**
     * * 清除本应用
SharedPreference(/data/data/com.xxx.xxx/shared_prefs) *
     *
     * @param context
     */
    public static void cleanSharedPreference(Context context) {
        deleteFilesByDirectory(new File("/data/data/"
                + context.getPackageName() + "/shared_prefs"));
    }

    /**
     * * 清除/data/data/com.xxx.xxx/files下的内容 * *
     *
     * @param context
     */
    public static void cleanFiles(Context context) {
        deleteFilesByDirectory(context.getFilesDir());
    }

    /**
```

```
     * * 清除外部cache下的内容
(/mnt/sdcard/android/data/com.xxx.xxx/cache)
     *
     * @param context
     */
    public static void cleanExternalCache(Context context) {
        if (Environment.getExternalStorageState().equals(
                Environment.MEDIA_MOUNTED)) {
            deleteFilesByDirectory(context.getExternalCacheDir());
        }
    }

    /**
     * * 清除自定义路径下的文件，使用需小心，请不要误删。而且只支持目录下的文
件删除 * *
     *
     * @param filePath
     * */
    public static void cleanCustomCache(String filePath) {
        deleteFilesByDirectory(new File(filePath));
    }

    /**
     * * 清除本应用所有的数据 * *
     *
     * @param context
     * @param filepath
     */
    public static void cleanApplicationData(Context context,
String... filepath) {
        cleanInternalCache(context);
        cleanExternalCache(context);
        cleanDatabases(context);
        cleanSharedPreference(context);
        cleanFiles(context);
        if (filepath == null) {
            return;
        }
        for (String filePath : filepath) {
            cleanCustomCache(filePath);
        }
    }

    /**
     * * 删除方法 这里只会删除某个文件夹下的文件，如果传入的directory是个文
件，将不做处理 * *
     *
     * @param directory
     */
```

```
    private static void deleteFilesByDirectory(File directory) {
        if (directory != null && directory.exists() &&
directory.isDirectory()) {
            for (File item : directory.listFiles()) {
                item.delete();
            }
        }
    }

    /**
     * Context.getExternalFilesDir() -->
     * SDCard/Android/data/你的应用的包名/files/目录，一般放一些长时间保
存的数据
     *
     * Context.getExternalCacheDir() -->
     * SDCard/Android/data/你的应用包名/cache/目录，一般存放临时缓存数据
     *
     * @param file
     * @return
     * @throws Exception
     */
    public static long getFolderSize(File file) throws Exception {
        long size = 0;
        try {
            File[] fileList = file.listFiles();
            for (int i = 0; i < fileList.length; i++) {
                // 如果下面还有文件
                if (fileList[i].isDirectory()) {
                    size = size + getFolderSize(fileList[i]);
                } else {
                    size = size + fileList[i].length();
                }
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return size;
    }

    /**
     * 删除指定目录下文件及目录
     *
     * @param deleteThisPath
     * @param filepath
     * @return
     */
    public static void deleteFolderFile(String filePath, boolean
deleteThisPath) {
        if (!TextUtils.isEmpty(filePath)) {
```

```
            try {
                File file = new File(filePath);
                if (file.isDirectory()) {// 如果下面还有文件
                    File files[] = file.listFiles();
                    for (int i = 0; i < files.length; i++) {
                        deleteFolderFile(files[i].getAbsolutePath(), true);
                    }
                }
                if (deleteThisPath) {
                    if (!file.isDirectory()) {// 如果是文件，删除
                        file.delete();
                    } else {// 目录
                        if (file.listFiles().length == 0) {// 目录下没有文件或者目录，删除
                            file.delete();
                        }
                    }
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
    }

    /**
     * 格式化单位
     *
     * @param size
     * @return
     */
    public static String getFormatSize(double size) {
        double kiloByte = size / 1024;
        if (kiloByte < 1) {
            return size + "Byte";
        }

        double megaByte = kiloByte / 1024;
        if (megaByte < 1) {
            BigDecimal result1 = new BigDecimal(Double.toString(kiloByte));
            return result1.setScale(2, BigDecimal.ROUND_HALF_UP)
                    .toPlainString() + "KB";
        }

        double gigaByte = megaByte / 1024;
        if (gigaByte < 1) {
            BigDecimal result2 = new BigDecimal(Double.toString(megaByte));
```

```
            return result2.setScale(2, BigDecimal.ROUND_HALF_UP)
                    .toPlainString() + "MB";
        }

        double teraBytes = gigaByte / 1024;
        if (teraBytes < 1) {
            BigDecimal result3 = new
BigDecimal(Double.toString(gigaByte));
            return result3.setScale(2, BigDecimal.ROUND_HALF_UP)
                    .toPlainString() + "GB";
        }
        BigDecimal result4 = new BigDecimal(teraBytes);
        return result4.setScale(2,
BigDecimal.ROUND_HALF_UP).toPlainString()
                + "TB";
    }

}
```

13.加解密工具 AESUtils

```
package org.iti.algorithm;

import java.security.SecureRandom;

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class AESUtils {

    /**
     * 加密方法
     *
     * @param seed
     *            密钥
     * @param clearText
     *            明文
     * @return
     */
    public static String encrypt(String seed, String clearText) {
        byte[] result = null;
        try {
            byte[] rawkey = getRawKey(seed.getBytes());
            result = encrypt(rawkey, clearText.getBytes());
```

```
        } catch (Exception e) {
            e.printStackTrace();
        }
        String content = toHex(result);
        return content;

    }

    /**
     * 解密方法
     *
     * @param seed
     * @param encrypted
     * @return
     */
    public static String decrypt(String seed, String encrypted) {
        byte[] rawKey;
        try {
            rawKey = getRawKey(seed.getBytes());
            byte[] enc = toByte(encrypted);
            byte[] result = decrypt(rawKey, enc);
            String coentn = new String(result);
            return coentn;
        } catch (Exception e) {
            e.printStackTrace();
            return null;
        }

    }

    private static byte[] getRawKey(byte[] seed) throws Exception
{
        KeyGenerator kgen = KeyGenerator.getInstance("AES");
        SecureRandom sr = SecureRandom.getInstance("SHA1PRNG");
        sr.setSeed(seed);
        kgen.init(128, sr);
        SecretKey sKey = kgen.generateKey();
        byte[] raw = sKey.getEncoded();
        return raw;
    }

    private static byte[] encrypt(byte[] raw, byte[] clear) throws
Exception {
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
        Cipher cipher =
Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, skeySpec, new
IvParameterSpec(
                new byte[cipher.getBlockSize()]));
```

```
        byte[] encrypted = cipher.doFinal(clear);
        return encrypted;
    }

    private static byte[] decrypt(byte[] raw, byte[] encrypted)
            throws Exception {
        SecretKeySpec skeySpec = new SecretKeySpec(raw, "AES");
        Cipher cipher =
Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(Cipher.DECRYPT_MODE, skeySpec, new
IvParameterSpec(
                new byte[cipher.getBlockSize()]));
        byte[] decrypted = cipher.doFinal(encrypted);
        return decrypted;
    }

    public static String toHex(String txt) {
        return toHex(txt.getBytes());
    }

    public static String fromHex(String hex) {
        return new String(toByte(hex));
    }

    public static byte[] toByte(String hexString) {
        int len = hexString.length() / 2;
        byte[] result = new byte[len];
        for (int i = 0; i < len; i++)
            result[i] = Integer.valueOf(hexString.substring(2 * i,
2 * i + 2),
                    16).byteValue();
        return result;
    }

    public static String toHex(byte[] buf) {
        if (buf == null)
            return "";
        StringBuffer result = new StringBuffer(2 * buf.length);
        for (int i = 0; i < buf.length; i++) {
            appendHex(result, buf[i]);
        }
        return result.toString();
    }

    private static void appendHex(StringBuffer sb, byte b) {
        final String HEX = "0123456789ABCDEF";
        sb.append(HEX.charAt((b >> 4) & 0x0f)).append(HEX.charAt(b
& 0x0f));
    }
```

```
}
```

14.Apache HttpClient的封装

基于Apache HttpClient的封装，支持HTTP GET、POST请求，支持多文件上传

```
package org.iti.eyescare.util;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.nio.charset.Charset;
import java.security.KeyStore;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Map;
import java.util.Map.Entry;

import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.HttpVersion;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.conn.ClientConnectionManager;
import org.apache.http.conn.scheme.PlainSocketFactory;
import org.apache.http.conn.scheme.Scheme;
import org.apache.http.conn.scheme.SchemeRegistry;
import org.apache.http.conn.ssl.SSLSocketFactory;
import org.apache.http.entity.mime.HttpMultipartMode;
import org.apache.http.entity.mime.MultipartEntityBuilder;
import org.apache.http.entity.mime.content.FileBody;
import org.apache.http.impl.client.DefaultHttpClient;
import
org.apache.http.impl.conn.tsccm.ThreadSafeClientConnManager;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.params.BasicHttpParams;
import org.apache.http.params.HttpConnectionParams;
import org.apache.http.params.HttpParams;
import org.apache.http.params.HttpProtocolParams;
import org.apache.http.protocol.HTTP;
import org.apache.http.util.EntityUtils;
```

```
/**
 * 基于Apache HttpClient的封装，支持HTTP GET、POST请求，支持多文件上传
 *
 * @author Tailyou
 *
 */
public class ApacheHttpUtil {
    // 设置URLConnection的连接超时
    private final static int CONNET_TIMEOUT = 30 * 1000;
    // 设置URLConnection的读取超时
    private final static int READ_TIMEOUT = 30 * 1000;

    /**
     * HTTP GET请求
     *
     * @param url
     *            请求链接
     * @return HTTP GET请求结果
     */
    public static String get(String url) {
        return get(url, null);
    }

    /**
     * HTTP GET请求
     *
     * @param url
     *            请求链接
     * @param params
     *            HTTP GET请求的QueryString封装map集合
     * @return HTTP GET请求结果
     */
    public static String get(String url, Map<String, String>
params) {
        try {
            String realUrl = generateUrl(url, params);
            HttpClient client = getNewHttpClient();
            HttpGet getMethod = new HttpGet(realUrl);
            HttpResponse response = client.execute(getMethod);
            if (response.getStatusLine().getStatusCode() ==
HttpStatus.SC_OK) {
                StringBuilder builder = new StringBuilder();
                BufferedReader reader = new BufferedReader(
                        new
InputStreamReader(response.getEntity().getContent()));
                for (String s = reader.readLine(); s != null; s =
reader
                        .readLine()) {
```

```
                    builder.append(s);
                }
                String result = builder.toString();
                return result;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    /**
     * HTTP POST请求
     *
     * @param url
     *            请求链接
     * @param params
     *            HTTP POST请求body的封装map集合
     * @return
     *
     */
    public static String post(String url, Map<String, String>
params) {
        try {
            HttpClient client = getHttpClient();
            HttpPost postMethod = new HttpPost(url);
            List<BasicNameValuePair> pairs = new
ArrayList<BasicNameValuePair>();
            if (params != null && params.size() > 0) {
                Iterator<Entry<String, String>> iterator =
params.entrySet()
                        .iterator();
                while (iterator.hasNext()) {
                    Entry<String, String> param = iterator.next();
                    String key = param.getKey();
                    String value = param.getValue();
                    BasicNameValuePair pair = new
BasicNameValuePair(key, value);
                    pairs.add(pair);
                }
                postMethod
                        .setEntity(new UrlEncodedFormEntity(pairs,
HTTP.UTF_8));
            }
            HttpResponse response = client.execute(postMethod);
            if (response.getStatusLine().getStatusCode() ==
HttpStatus.SC_OK) {
                String result =
EntityUtils.toString(response.getEntity());
```

```
                return result;
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return null;
    }

    /**
     * HTTP POST请求上传文件，支持多文件上传
     *
     * @param url
     *            请求链接
     * @param params
     *            HTTP POST请求文本参数map集合
     * @param files
     *            HTTP POST请求文件参数map集合
     * @return HTTP POST请求结果
     * @throws IOException
     */
    public static String post(String url, Map<String, String>
params,
            Map<String, String> files) throws IOException {
        MultipartEntityBuilder multipartEntityBuilder =
MultipartEntityBuilder
                .create();

multipartEntityBuilder.setMode(HttpMultipartMode.BROWSER_COMPATIBL
E);
        multipartEntityBuilder.setCharset(Charset.forName("UTF-
8"));
        if (params != null && !params.isEmpty()) {
            for (Map.Entry<String, String> entry :
params.entrySet()) {
                multipartEntityBuilder.addTextBody(entry.getKey(),
                        entry.getValue());
            }
        }

        if (files != null && !files.isEmpty()) {
            for (Map.Entry<String, String> entry :
files.entrySet()) {
                File file = new File(entry.getValue());

multipartEntityBuilder.addBinaryBody(entry.getKey(), file);
            }
        }

        HttpClient client = getNewHttpClient();
```

```
        HttpPost post = new HttpPost(url);
        HttpEntity httpEntity = multipartEntityBuilder.build();
        post.setEntity(httpEntity);
        HttpResponse response = client.execute(post);

        StringBuffer sb = new StringBuffer();
        if (response.getStatusLine().getStatusCode() ==
HttpStatus.SC_OK) {
            HttpEntity result = response.getEntity();
            if (result != null) {
                InputStream is = result.getContent();
                BufferedReader br = new BufferedReader(
                        new InputStreamReader(is));
                String tempLine;
                while ((tempLine = br.readLine()) != null) {
                    sb.append(tempLine);
                }
            }
        }
        post.abort();
        return sb.toString();
    }

    /**
     * 上传图片
     *
     * @param url
     * @param file
     * @return
     * @throws IOException
     */
    public static String post(String url, File file) throws
IOException {
        HttpClient client = getNewHttpClient();
        HttpPost post = new HttpPost(url);

        MultipartEntityBuilder multipartEntityBuilder =
MultipartEntityBuilder
                .create();

multipartEntityBuilder.setMode(HttpMultipartMode.BROWSER_COMPATIBL
E);
        multipartEntityBuilder.setCharset(Charset.forName("UTF-
8"));
        multipartEntityBuilder.addTextBody("fileName",
file.getName());
        multipartEntityBuilder.addPart("uploadFile", new
FileBody(file));
```

```
        HttpEntity httpEntity = multipartEntityBuilder.build();
        post.setEntity(httpEntity);

        HttpResponse response = client.execute(post);
        StringBuffer sb = new StringBuffer();
        if (response.getStatusLine().getStatusCode() ==
HttpStatus.SC_OK) {
            HttpEntity result = response.getEntity();
            if (result != null) {
                InputStream is = result.getContent();
                BufferedReader br = new BufferedReader(
                        new InputStreamReader(is));
                String tempLine;
                while ((tempLine = br.readLine()) != null) {
                    sb.append(tempLine);
                }
            }
        }
        post.abort();
        return sb.toString();
    }

    /**
     * 获取HttpClient
     *
     * @return
     */
    public static HttpClient getHttpClient() {
        BasicHttpParams httpParams = new BasicHttpParams();
        HttpConnectionParams.setConnectionTimeout(httpParams,
CONNET_TIMEOUT);
        HttpConnectionParams.setSoTimeout(httpParams,
READ_TIMEOUT);
        HttpClient httpClient = new DefaultHttpClient(httpParams);
        return httpClient;
    }

    /**
     * 获取HttpClient
     *
     * @return
     */
    private static HttpClient getNewHttpClient() {
        try {
            KeyStore trustStore = KeyStore.getInstance(KeyStore
                    .getDefaultType());
            trustStore.load(null, null);
            SSLSocketFactory sf = new
SSLSocketFactory(trustStore);
```

```
sf.setHostnameVerifier(SSLSocketFactory.ALLOW_ALL_HOSTNAME_VERIFIE
R);

            HttpParams params = new BasicHttpParams();
            HttpProtocolParams.setVersion(params,
HttpVersion.HTTP_1_1);
            HttpProtocolParams.setContentCharset(params,
HTTP.UTF_8);
            HttpConnectionParams.setConnectionTimeout(params,
CONNET_TIMEOUT);
            HttpConnectionParams.setSoTimeout(params,
READ_TIMEOUT);

            SchemeRegistry registry = new SchemeRegistry();
            registry.register(new Scheme("http",
PlainSocketFactory
                    .getSocketFactory(), 80));
            registry.register(new Scheme("https", sf, 443));
            ClientConnectionManager ccm = new
ThreadSafeClientConnManager(
                    params, registry);
            return new DefaultHttpClient(ccm, params);
        } catch (Exception e) {
            return new DefaultHttpClient();
        }
    }

    /**
     * 根据基础url和参数拼接请求地址
     *
     * @param url
     * @param params
     * @return
     */
    private static String generateUrl(String url, Map<String,
String> params) {
        StringBuilder urlBuilder = new StringBuilder(url);
        if (null != params) {
            urlBuilder.append("?");
            Iterator<Entry<String, String>> iterator =
params.entrySet()
                    .iterator();
            while (iterator.hasNext()) {
                Entry<String, String> param = iterator.next();
                String key = param.getKey();
                String value = param.getValue();
                urlBuilder.append(key).append('=').append(value);
                if (iterator.hasNext()) {
```

```
                    urlBuilder.append('&');
                }
            }
        }
        return urlBuilder.toString();
    }
}
```

15.上传下载图片ImageUtil

```
package org.iti.eyescare.util;

import java.io.File;
import java.io.IOException;
import java.io.InputStream;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;

import org.iti.eyescare.constants.Constants;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;

public class ImageUtil {

    /**
     * 上传图片
     *
     * @param filePath
     *            文件路径
     * @param count
     *            失败重传次数
     * @return
     */
    public static String uploadPic(File file, int count) {
        String filePathServer = "";
        if (count > 0) {
            try {
                String result =
ApacheHttpUtil.post(Constants.UPLOAD_FILE_URL,
                        file);
                Response resp = Response.convert(result);
                filePathServer = resp.getResponResult();
            } catch (IOException e1) {
                e1.printStackTrace();
                uploadPic(file, count--);
```

```
                }
            }
            return filePathServer;
        }

        /**
         * 根据图片在服务器上的地址加载图片
         *
         * @param urlStr
         * @param reqWidth
         * @param reqHeight
         * @return
         * @throws MalformedURLException
         * @throws IOException
         */
        public static Bitmap decodeSampledBitmapFromUrl(String urlStr,
                int reqWidth, int reqHeight) throws
    MalformedURLException,
                IOException {
            final BitmapFactory.Options options = new
    BitmapFactory.Options();
            options.inJustDecodeBounds = true;
            BitmapFactory
                    .decodeStream(getInputStreamFromUrl(urlStr), null,
    options);
            options.inSampleSize = calculateInSampleSize(options,
    reqWidth,
                    reqHeight);
            options.inJustDecodeBounds = false;
            options.inPreferredConfig = Bitmap.Config.RGB_565;
            options.inPurgeable = true;
            options.inInputShareable = true;
            return
    BitmapFactory.decodeStream(getInputStreamFromUrl(urlStr), null,
                    options);
        }

        /**
         * 从url中得到流
         *
         * @param urlStr
         * @return
         * @throws MalformedURLException
         * @throws IOException
         */
        public static InputStream getInputStreamFromUrl(String urlStr)
                throws MalformedURLException, IOException {
            URL url = new URL(urlStr);
            HttpURLConnection urlConn = (HttpURLConnection)
```

```
url.openConnection();
        InputStream inputStream = urlConn.getInputStream();
        return inputStream;
    }

    /**
     * 计算inSampleSize
     *
     * @param options
     * @param reqWidth
     * @param reqHeight
     * @return
     */
    public static int calculateInSampleSize(BitmapFactory.Options
options,
            int reqWidth, int reqHeight) {
        final int height = options.outHeight;
        final int width = options.outWidth;
        int inSampleSize = 1;

        if (height > reqHeight || width > reqWidth) {
            if (width > height) {
                inSampleSize = Math.round((float) height / (float)
reqHeight);
            } else {
                inSampleSize = Math.round((float) width / (float)
reqWidth);
            }
        }
        return inSampleSize;
    }
}
```

16.对象转换成json字符串以及反序列化

```
package org.iti.eyescare.util;

import java.io.Serializable;
import java.util.Collection;
import java.util.List;
import java.util.Map;
import java.util.Set;

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

public class JsonUtil implements Serializable {
```

```
    private static final long serialVersionUID =
-7609521898676321656L;

    /**
     * 从对象转换成json字符串
     *
     * @param obj
     * @return
     */
    public static final String toJson(Object obj) {
        Gson gson = new Gson();
        return gson.toJson(obj);
    }

    /**
     * 反序列化
     *
     * @param clazz
     *            目标类型
     * @param json
     *            json字符串
     * @return
     */
    public static final <T> T toObj(TypeToken<T> typeToken, String
json) {
        Gson gson = new Gson();
        return gson.fromJson(json, typeToken.getType());
    }

    /**
     * 反序列化
     *
     * @param json
     * @return
     */
    public static <T> Collection<T> toCollection(String json,
            TypeToken<Collection<T>> typeToken) {
        return JsonUtil.toObj(typeToken, json);
    }

    /**
     * 反序列化
     *
     * @param json
     *            json字符串
     * @return
     */
    public static <T> List<T> toList(String json,
```

```
    TypeToken<List<T>> typeToken) {
            return JsonUtil.toObj(typeToken, json);
        }

        /**
         * 反序列化
         *
         * @param json
         *            json字符串
         * @return
         */
        public static <T> Set<T> toSet(String json, TypeToken<Set<T>>
    typeToken) {
            return JsonUtil.toObj(typeToken, json);
        }

        /**
         * 反序列化
         *
         * @param json
         *            json字符串
         * @return
         */
        public static <K, V> Map<K, V> toMap(String json,
                TypeToken<Map<K, V>> typeToken) {
            return JsonUtil.toObj(typeToken, json);
        }
    }
```

源码下载

源码 请点击
链接：https://github.com/Morcal/sampleCode/tree/master/AndroidUtils