Android项目总结3

> 最近又独立开发了一个公司内部的小app，现在的感觉就是每一次独立开发一个新的app，都会遇到不同的问题，也都能独立解决这些问题。感觉这个状态还可以吧，同时打算把这些问题记录一下，以后再遇到同样问题的时候，有一个可以查阅的资料，也能帮助其它遇到同样问题的小伙伴们吧。

- 本次项目采用的网络框架：rxjava + retrofit2,这个就不在这里总结了，有空会单独总结一篇有关文章的。
- 获取屏幕宽度

```
public int getScreenWidth(){
        WindowManager wm = (WindowManager) this.getSystemService(Context.WINDOW_SERVICE);
        DisplayMetrics dm = new DisplayMetrics();
        wm.getDefaultDisplay().getMetrics(dm);
        int width = dm.widthPixels;         // 屏幕宽度（像素）
        int height = dm.heightPixels;       // 屏幕高度（像素）
        float density = dm.density;         // 屏幕密度（0.75 / 1.0 / 1.5）
        int densityDpi = dm.densityDpi;     // 屏幕密度dpi（120 / 160 / 240）
        // 屏幕宽度算法:屏幕宽度（像素）/屏幕密度
        int screenWidth = (int) (width / density);  // 屏幕宽度(dp)
        int screenHeight = (int) (height / density);// 屏幕高度(dp)
        return width;
    }
```

- 获取屏幕高度

```
public int getScreenHeight(){
        WindowManager wm = (WindowManager) this.getSystemService(Context.WINDOW_SERVICE);
        DisplayMetrics dm = new DisplayMetrics();
        wm.getDefaultDisplay().getMetrics(dm);
        int width = dm.widthPixels;         // 屏幕宽度（像素）
        int height = dm.heightPixels;       // 屏幕高度（像素）
        float density = dm.density;         // 屏幕密度（0.75 / 1.0 / 1.5）
        int densityDpi = dm.densityDpi;     // 屏幕密度dpi（120 / 160 / 240）
        // 屏幕宽度算法:屏幕宽度（像素）/屏幕密度
        int screenWidth = (int) (width / density);  // 屏幕宽度(dp)
        int screenHeight = (int) (height / density);// 屏幕高度(dp)
        getAndroiodScreenProperty();
        return height;
    }
```

- 获取进程名称

```
private static String getProcessName(int pid) {
        BufferedReader reader = null;
        try {
            reader = new BufferedReader(new FileReader("/proc/" + pid + "/cmdline"));
            String processName = reader.readLine();
            if (!TextUtils.isEmpty(processName)) {
                processName = processName.trim();
            }
            return processName;
        } catch (Throwable throwable) {
            throwable.printStackTrace();
        } finally {
            try {
                if (reader != null) {
                    reader.close();
                }
            } catch (IOException exception) {
                exception.printStackTrace();
            }
        }
        return null;
    }
```

- 不能滑动的ViewPager

```
public class NoScrollViewPager extends ViewPager {
    private boolean noScroll = true;

    public NoScrollViewPager(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public NoScrollViewPager(Context context) {
        super(context);
    }

    public void setNoScroll(boolean noScroll) {
        this.noScroll = noScroll;
    }

    @Override
    public void scrollTo(int x, int y) {
        super.scrollTo(x, y);
    }

    @Override
    public boolean onTouchEvent(MotionEvent arg0) {
        if (noScroll)
            return false;
        else
            return super.onTouchEvent(arg0);
    }

    @Override
    public boolean onInterceptTouchEvent(MotionEvent arg0) {
        if (noScroll)
            return false;
        else
            return super.onInterceptTouchEvent(arg0);
    }

    @Override
    public void setCurrentItem(int item, boolean smoothScroll) {
        super.setCurrentItem(item, smoothScroll);
    }

    @Override

    public void setCurrentItem(int item) {

        super.setCurrentItem(item, false);//表示切换的时候，不需要切换时间。

    }

}
```

- 自定义loadingView

LoadingBase.java

```
public abstract class LoadingBase extends View {

    public LoadingBase(Context context) {
        this(context, null);
    }

    public LoadingBase(Context context, AttributeSet attrs) {
        this(context, attrs, 0);
    }

    public LoadingBase(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        InitPaint();
    }

    public void startAnim() {
        stopAnim();
        startViewAnim(0f, 1f, 500);
    }

    public void startAnim(int time) {
        stopAnim();
        startViewAnim(0f, 1f, time);
    }
```

```
public void stopAnim() {
    if (valueAnimator != null) {
        clearAnimation();

        valueAnimator.setRepeatCount(0);
        valueAnimator.cancel();
        valueAnimator.end();
        if (OnStopAnim() == 0) {
            valueAnimator.setRepeatCount(0);
            valueAnimator.cancel();
            valueAnimator.end();
        }

    }
}

public ValueAnimator valueAnimator;

private ValueAnimator startViewAnim(float startF, final float endF, long time) {
    valueAnimator = ValueAnimator.ofFloat(startF, endF);
    valueAnimator.setDuration(time);
    valueAnimator.setInterpolator(new LinearInterpolator());


    valueAnimator.setRepeatCount(SetAnimRepeatCount());



    if (ValueAnimator.RESTART == SetAnimRepeatMode()) {
        valueAnimator.setRepeatMode(ValueAnimator.RESTART);

    } else if (ValueAnimator.REVERSE == SetAnimRepeatMode()) {
        valueAnimator.setRepeatMode(ValueAnimator.REVERSE);

    }

    valueAnimator.addUpdateListener(new ValueAnimator.AnimatorUpdateListener() {
        @Override
        public void onAnimationUpdate(ValueAnimator valueAnimator) {
            OnAnimationUpdate(valueAnimator);

        }
    });
    valueAnimator.addListener(new AnimatorListenerAdapter() {
        @Override
        public void onAnimationEnd(Animator animation) {
            super.onAnimationEnd(animation);

        }

        @Override
        public void onAnimationStart(Animator animation) {

            super.onAnimationStart(animation);
        }

        @Override
        public void onAnimationRepeat(Animator animation) {
            super.onAnimationRepeat(animation);
            OnAnimationRepeat(animation);
        }
    });
    if (!valueAnimator.isRunning()) {
        AinmIsRunning();
        valueAnimator.start();

    }

    return valueAnimator;
}


protected abstract void InitPaint();

protected abstract void OnAnimationUpdate(ValueAnimator valueAnimator);

protected abstract void OnAnimationRepeat(Animator animation);

protected abstract int OnStopAnim();

protected abstract int SetAnimRepeatMode();
```

```
    protected abstract int SetAnimRepeatMode();

    protected abstract int SetAnimRepeatCount();

    protected abstract void AinmIsRunning();


    public int dip2px(float dpValue) {
        final float scale = getContext().getResources().getDisplayMetrics().density;
        return (int) (dpValue * scale + 0.5f);
    }

    public float getFontlength(Paint paint, String str) {
        Rect rect = new Rect();
        paint.getTextBounds(str, 0, str.length(), rect);
        return rect.width();
    }

    public float getFontHeight(Paint paint, String str) {
        Rect rect = new Rect();
        paint.getTextBounds(str, 0, str.length(), rect);
        return rect.height();

    }

    public float getFontHeight(Paint paint) {
        Paint.FontMetrics fm = paint.getFontMetrics();
        return fm.descent - fm.ascent;
    }

}
```

LoadingView.java

```
public class LoadingView extends LoadingBase {

    private Paint mPaint;
    private Paint mPaintPro;

    private float mWidth = 0f;
    private float mPadding = 0f;
    private float startAngle = 0f;
    RectF rectF = new RectF();

    public LoadingView(Context context) {
        super(context);
    }

    public LoadingView(Context context, AttributeSet attrs) {
        super(context, attrs);
    }

    public LoadingView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
    }


    @Override
    protected void onMeasure(int widthMeasureSpec, int heightMeasureSpec) {
        super.onMeasure(widthMeasureSpec, heightMeasureSpec);

        if (getMeasuredWidth() > getHeight())
            mWidth = getMeasuredHeight();
        else
            mWidth = getMeasuredWidth();
        mPadding = 5;
    }

    @Override
    protected void onDraw(Canvas canvas) {
        super.onDraw(canvas);

        canvas.drawCircle(mWidth / 2, mWidth / 2, mWidth / 2 - mPadding, mPaintPro);
        rectF = new RectF(mPadding, mPadding, mWidth - mPadding, mWidth - mPadding);
        canvas.drawArc(rectF, startAngle, 100
                , false, mPaint);//第四个参数是否显示半径

    }


    private void initPaint() {
```

```
        mPaint = new Paint();
        mPaint.setAntiAlias(true);
        mPaint.setStyle(Paint.Style.STROKE);
        mPaint.setColor(Color.WHITE);
        mPaint.setStrokeWidth(8);

        mPaintPro = new Paint();
        mPaintPro.setAntiAlias(true);
        mPaintPro.setStyle(Paint.Style.STROKE);
        mPaintPro.setColor(Color.argb(100, 255, 255, 255));
        mPaintPro.setStrokeWidth(8);

    }


    public void setViewColor(int color) {
        mPaintPro.setColor(color);
        postInvalidate();
    }

    public void setBarColor(int color) {
        mPaint.setColor(color);
        postInvalidate();
    }


    @Override
    protected void InitPaint() {
        initPaint();
    }

    @Override
    protected void OnAnimationUpdate(ValueAnimator valueAnimator) {
        float value = (float) valueAnimator.getAnimatedValue();
        startAngle = 360 * value;

        invalidate();
    }

    @Override
    protected void OnAnimationRepeat(Animator animation) {

    }

    @Override
    protected int OnStopAnim() {
        return 0;
    }

    @Override
    protected int SetAnimRepeatMode() {
        return ValueAnimator.RESTART;
    }

    @Override
    protected void AinmIsRunning() {

    }

    @Override
    protected int SetAnimRepeatCount() {
        return ValueAnimator.INFINITE;
    }

}
```

- 自定义异常

```
public class NetworkException extends RuntimeException {



    public static final int REQUEST_OK = 100;
    public static final int REQUEST_FAIL = 101;
    public static final int METHOD_NOT_ALLOWED = 102;
    public static final int PARAMETER_ERROR = 103;
    public static final int UID_OR_PWD_ERROR = 104;
    public static final int SERVER_INTERNAL_ERROR = 105;
    public static final int REQUEST_TIMEOUT = 106;
    public static final int CONNECTION_ERROR = 107;
    public static final int VERIFY_EXPIRED = 108;
    public static final int NO_DATA = 109;


    public NetworkException(int resultCode) {
        this(getNetworkExceptionMessage(resultCode));
    }

    public NetworkException(String detailMessage) {
        super(detailMessage);
    }

    /**
     * 将结果码转换成对应的文本信息
     */
    private static String getNetworkExceptionMessage(int code) {
        String message = "";
        switch (code) {
            case REQUEST_OK:
                message = "请求成功";
                break;
            case REQUEST_FAIL:
                message = "请求失败";
                break;

            case METHOD_NOT_ALLOWED:
                message = "请求方式不允许";
                break;
            case PARAMETER_ERROR:
                message = "用户不存在";
                break;
            case UID_OR_PWD_ERROR:
                message = "用户名或密码错误";
                break;
            case SERVER_INTERNAL_ERROR:
                message = "服务器内部错误";
                break;
            case REQUEST_TIMEOUT:
                message = "请求超时";
                break;
            case CONNECTION_ERROR:
                message = "连接错误";
                break;
            case VERIFY_EXPIRED:
                message = "验证过期";
                break;
            case NO_DATA:
                message = "没有数据";
                break;
            case 110:
                message = "该用户已存在";
                break;
            default:
                message = "未知错误";
        }
        return message;
    }

}
```

- 自定义OnClickListener防止重复点击发生的问题

```
public abstract class NoDoubleClickListener implements View.OnClickListener {

    private static final int MIN_CLICK_DELAY_TIME = 1000;
    private long lastClickTime = 0;

    @Override
    public void onClick(View v) {
        long time = System.currentTimeMillis();
        if (time - lastClickTime > MIN_CLICK_DELAY_TIME) {
            lastClickTime = time;
            onNoDoubleClick(v);
        }

    }
    public abstract void onNoDoubleClick(View v);
}
```

- 创建notification并且添加点击事件

```
PendingIntent mainPendingIntent = null;

Intent mainIntent = new Intent(this, MainActivity.class);
mainPendingIntent = PendingIntent.getActivity(this, 0, mainIntent, PendingIntent.FLAG_UPDATE_CURRENT)


    NotificationManager mNotificationManager = (NotificationManager) getSystemService(NOTIFICATION_SE
    NotificationCompat.Builder mBuilder = new NotificationCompat.Builder(this);
    mBuilder.setContentTitle("您有新的订单,请登陆app查看")//设置通知栏标题
            .setContentIntent(getDefalutIntent(Notification.FLAG_AUTO_CANCEL)) //设置通知栏点击意图
            //  .setNumber(number) //设置通知集合的数量
            .setTicker("您有新的订单,请登陆app查看") //通知首次出现在通知栏，带上升动画效果的
            .setWhen(System.currentTimeMillis())//通知产生的时间，会在通知信息里显示，一般是系统获取到的时间
            .setContentIntent(mainPendingIntent)
            .setPriority(Notification.PRIORITY_DEFAULT) //设置该通知优先级
            .setAutoCancel(true)//设置这个标志当用户单击面板就可以让通知将自动取消
            .setOngoing(false)//ture，设置他为一个正在进行的通知。他们通常是用来表示一个后台任务,用户积极参与(如播放
            .setDefaults(Notification.DEFAULT_VIBRATE)//向通知添加声音、闪灯和振动效果的最简单、最一致的方式是使
            //Notification.DEFAULT_ALL  Notification.DEFAULT_SOUND 添加声音 // requires VIBRATE permiss
            .setSmallIcon(R.mipmap.ic_launcher);//设置通知小ICON


    mNotificationManager.notify(1, mBuilder.build());
```

- 获取Imei，兼容Android N(easypermission)

```
public class LoginActivity extends AppCompatActivity implements EasyPermissions.PermissionCallbacks {

    private String imei = "00000000000";
    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_login);
        String[] perms = {android.Manifest.permission.READ_PHONE_STATE};
        if (EasyPermissions.hasPermissions(this, perms)) {
            Log.e("lin", "---lin--->  imie  if");
            TelephonyManager telephonyManager = (TelephonyManager) this.getApplicationContext().getSy
            imei = telephonyManager.getDeviceId();
        } else {
            Log.e("lin", "---lin--->  imie  else");
            EasyPermissions.requestPermissions(this, "正在申请获取手机唯一编码",
                    100, perms);
        }
    }

    @Override
    public void onRequestPermissionsResult(int requestCode, String[] permissions, int[] grantResults)
        super.onRequestPermissionsResult(requestCode, permissions, grantResults);
        // Forward results to EasyPermissions
        EasyPermissions.onRequestPermissionsResult(requestCode, permissions, grantResults, this);
    }

    @Override
    public void onPermissionsGranted(int requestCode, List<String> perms) {
        TelephonyManager telephonyManager = (TelephonyManager) this.getApplicationContext().getSystem
        imei = telephonyManager.getDeviceId();
    }

    @Override
    public void onPermissionsDenied(int requestCode, List<String> perms) {

    }

}
```

- 倒计时控件

```
private TimeCount mTimeCount;
mTimeCount = new TimeCount(60000, 1000);
mTimeCount.start();

class TimeCount extends CountDownTimer {
    public TimeCount(long millisInFuture, long countDownInterval) {
        super(millisInFuture + 200 , countDownInterval);
    }

    @Override
    public void onFinish() {// 计时完毕
        tvGetCodeActivityLogin.setText("获取验证码");
        tvGetCodeActivityLogin.setClickable(true);
    }


    @Override
    public void onTick(long millisUntilFinished) {// 计时过程
        long time = millisUntilFinished / 1000;
        String timeString = String.valueOf(time);
        timeString = timeString + "S";
        tvGetCodeActivityLogin.setText(timeString);
        tvGetCodeActivityLogin.setClickable(false);//防止重复点击
    }
}
```

- 设置editext hint字的颜色值

```
CharSequence hint = edtAuthCodeActivityLogin.getHint();
SpannableString ss =  new SpannableString(hint);
AbsoluteSizeSpan ass = new AbsoluteSizeSpan(17, true);
edtAuthCodeActivityLogin.setHintTextColor(0xffdddddd);
ss.setSpan(ass, 0, ss.length(), Spanned.SPAN_EXCLUSIVE_EXCLUSIVE);
edtAuthCodeActivityLogin.setHint(new SpannedString("短信验证码"));
```

- App内采用的开源相册 Album

- 图片压缩框架 Luban
- fragment类似activity的onResume()

```
protected boolean isCreate = false;

@Override
    public void setUserVisibleHint(boolean isVisibleToUser) {
        super.setUserVisibleHint(isVisibleToUser);
        if (isVisibleToUser && isCreate) {
            getOrderList("-1");
        }
    }
```

- 利用反射设置tablayout下划线长度

```
tabLayoutOrderFragment.post(new Runnable() {
            @Override
            public void run() {
                setIndicator(tabLayoutOrderFragment,20,20);
            }
        });

public void setIndicator(TabLayout tabs, int leftDip, int rightDip) {
        Class<?> tabLayout = tabs.getClass();
        Field tabStrip = null;
        try {
            tabStrip = tabLayout.getDeclaredField("mTabStrip");
        } catch (NoSuchFieldException e) {
            e.printStackTrace();
        }

        tabStrip.setAccessible(true);
        LinearLayout llTab = null;
        try {
            llTab = (LinearLayout) tabStrip.get(tabs);
        } catch (IllegalAccessException e) {
            e.printStackTrace();
        }

        int left = (int) TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, leftDip, Resources.ge
        int right = (int) TypedValue.applyDimension(TypedValue.COMPLEX_UNIT_DIP, rightDip, Resources.

        for (int i = 0; i < llTab.getChildCount(); i++) {
            View child = llTab.getChildAt(i);
            child.setPadding(0, 0, 0, 0);
            LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(0, LinearLayout.LayoutPa
            params.leftMargin = left;
            params.rightMargin = right;
            child.setLayoutParams(params);
            child.invalidate();
        }

    }
```

- 封装一个链式调用的dialog

```
public class DialogUtils {

    private Context mContext;
    private AlertDialog mAlertDialog;
    private DialogUtils.Builder mBuilder;
    private boolean mHasShow = false;
    private String mTitle;
    private String mMessage;
    private int messageColor = -1;
    private SingleButtonCallback mPositiveCallback;
    private SingleButtonCallback mNegativeCallback = new SingleButtonCallback() {
        @Override
        public void onClick(@NonNull DialogUtils dialog, View.OnClickListener listener) {
            dismiss();
        }
    };
    private boolean mCancel;


    public DialogUtils(Context context) {
```

```
        this.mContext = context;
    }

    public void show() {
        if (!mHasShow) {
            mBuilder = new Builder();
        } else {
            mAlertDialog.show();
        }
        mHasShow = true;
    }

    public void dismiss() {
        mAlertDialog.dismiss();
    }

    public DialogUtils setTitle(String title) {
        this.mTitle = title;
        if (mBuilder != null) {
            mBuilder.setTitle(title);
        }
        return this;
    }

    public DialogUtils setMessage(String message) {
        this.mMessage = message;
        if (mBuilder != null) {
            mBuilder.setMessage(message);
        }
        return this;
    }

    public DialogUtils setMessageColor(int color) {
        this.messageColor = color;
        return this;
    }

    public DialogUtils setCanceledOnTouchOutside(boolean cancel) {
        this.mCancel = cancel;
        if (mBuilder != null) {
            mBuilder.setCanceledOnTouchOutside(mCancel);
        }
        return this;
    }

    public DialogUtils setPositive(SingleButtonCallback positiveCallback) {
        this.mPositiveCallback = positiveCallback;
        return this;
    }

    public DialogUtils setNegative(SingleButtonCallback negativeCallback) {
        this.mNegativeCallback = negativeCallback;
        return this;
    }


    public enum DialogAction {
        POSITIVE,
        NEGATIVE
    }

    public interface SingleButtonCallback {
        void onClick(@NonNull DialogUtils dialog, View.OnClickListener listener);
    }


    private class Builder {

        private TextView mTitleView;
        private TextView mMessageView;
        private TextView mPositive, mNegative;
        private Window mAlertDialogWindow;
        private RelativeLayout mDialog;

        private Builder() {
            mAlertDialog = new AlertDialog.Builder(mContext, R.style.Theme_AppCompat_Dialog).create()
            mAlertDialog.show();
            mAlertDialog.getWindow()
                    .clearFlags(WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE |
                            WindowManager.LayoutParams.FLAG_ALT_FOCUSABLE_IM);
            mAlertDialog.getWindow()
                    .setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_MASK_STATE);
```

```
            mAlertDialogWindow = mAlertDialog.getWindow();
            mAlertDialogWindow.setBackgroundDrawable(
                    new ColorDrawable(android.graphics.Color.TRANSPARENT));
            View contentView = LayoutInflater.from(mContext)
                    .inflate(R.layout.dialog_util_layout, null);
            contentView.setFocusable(true);
            contentView.setFocusableInTouchMode(true);
            mAlertDialogWindow.setBackgroundDrawableResource(R.drawable.material_dialog_window);
            mAlertDialogWindow.setContentView(contentView);
            mTitleView = (TextView) mAlertDialogWindow.findViewById(R.id.tv_title);
            mMessageView = (TextView) mAlertDialogWindow.findViewById(R.id.tv_hint);
            mPositive = (TextView) mAlertDialogWindow.findViewById(R.id.tv_sure);
            mNegative = (TextView) mAlertDialogWindow.findViewById(R.id.tv_cancel);
            mDialog = (RelativeLayout) mAlertDialogWindow.findViewById(R.id.dialog);
            Log.i("lin", "----lin----> 宽 " + MyApplication.get().getScreenWidth());
            Log.i("lin", "----lin----> 高 " + MyApplication.get().getScreenHeight());

            mDialog.setLayoutParams(new RelativeLayout.LayoutParams(MyApplication.get().getScreenWidt
            if (mTitle != null) {
                mTitleView.setText(mTitle);
            }
            if (mMessage != null) {
                mMessageView.setText(mMessage);
            }
            if (messageColor != -1) {
                mMessageView.setTextColor(messageColor);
            }
            mAlertDialog.setCanceledOnTouchOutside(mCancel);
            mAlertDialog.setCancelable(mCancel);
            if (mPositiveCallback != null) {
                mPositive.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        mPositiveCallback.onClick(DialogUtils.this, this);
                    }
                });
            }
            if (mNegativeCallback != null) {
                mNegative.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        mNegativeCallback.onClick(DialogUtils.this, this);
                    }
                });
            }

        }


        public void setTitle(String title) {
            mTitleView.setText(title);
        }

        public void setMessage(String message) {
            if (mMessageView != null) {
                mMessageView.setText(message);
            }
        }

        public void setCanceledOnTouchOutside(boolean canceledOnTouchOutside) {
            mAlertDialog.setCanceledOnTouchOutside(canceledOnTouchOutside);
            mAlertDialog.setCancelable(canceledOnTouchOutside);
        }

    }

}
```

- 将loadingView封装成loadingViewDialog

```
public class LoadingUtils {


    private Context mContext;
    private AlertDialog mAlertDialog;
    private LoadingUtils.Builder mBuilder;
    private boolean mHasShow = false;
    private String mMessage;
    private boolean mCancel;
```

```
    public LoadingUtils(Context context) {
        this.mContext = context;
    }

    public void show() {
        if (!mHasShow) {
            mBuilder = new LoadingUtils.Builder();
        } else {
            mAlertDialog.show();
        }
        mHasShow = true;
    }

    public void dismiss() {
        mAlertDialog.dismiss();
    }


    public LoadingUtils setMessage(String message) {
        this.mMessage = message;
        if (mBuilder != null) {
            mBuilder.setMessage(message);
        }
        return this;
    }

    public LoadingUtils setCanceledOnTouchOutside(boolean cancel) {
        this.mCancel = cancel;
        if (mBuilder != null) {
            mBuilder.setCanceledOnTouchOutside(mCancel);
        }
        return this;
    }

    private class Builder {

        private Window mAlertDialogWindow;
        private TextView mMessageView;
        private RelativeLayout mDialog;
        private LoadingView loadingView;

        private Builder() {
            mAlertDialog = new AlertDialog.Builder(mContext, R.style.Theme_AppCompat_Dialog).create()
            mAlertDialog.show();
            mAlertDialog.getWindow()
                    .clearFlags(WindowManager.LayoutParams.FLAG_NOT_FOCUSABLE |
                            WindowManager.LayoutParams.FLAG_ALT_FOCUSABLE_IM);
            mAlertDialog.getWindow()
                    .setSoftInputMode(WindowManager.LayoutParams.SOFT_INPUT_MASK_STATE);

            mAlertDialogWindow = mAlertDialog.getWindow();
            mAlertDialogWindow.setBackgroundDrawable(
                    new ColorDrawable(android.graphics.Color.TRANSPARENT));
            View contentView = LayoutInflater.from(mContext)
                    .inflate(R.layout.loading_view_layout, null);
            contentView.setFocusable(true);
            contentView.setFocusableInTouchMode(true);
            mAlertDialogWindow.setBackgroundDrawableResource(R.drawable.material_dialog_window);
            mAlertDialogWindow.setContentView(contentView);
            mDialog = (RelativeLayout) contentView.findViewById(R.id.rl_loading_view);
            mMessageView = (TextView) contentView.findViewById(R.id.tv_hint);
            loadingView = (LoadingView) contentView.findViewById(R.id.loading_view);

            loadingView.setViewColor(Color.argb(100, 255, 255, 255));
            loadingView.startAnim();
            loadingView.setBarColor(0xFF42a5f5);
            loadingView.startAnim();


            Log.i("lin", "----lin----> 宽 " + MyApplication.get().getScreenWidth());
            Log.i("lin", "----lin----> 高 " + MyApplication.get().getScreenHeight());

            mDialog.setLayoutParams(new FrameLayout.LayoutParams(MyApplication.get().getScreenWidth()

            if (mMessage != null) {
                mMessageView.setText(mMessage);
            }
            mAlertDialog.setCanceledOnTouchOutside(mCancel);
            mAlertDialog.setCancelable(mCancel);
        }
```

```
        }


        public void setMessage(String message) {
            if (mMessageView != null) {
                mMessageView.setText(message);
            }
        }

        public void setCanceledOnTouchOutside(boolean canceledOnTouchOutside) {
            mAlertDialog.setCanceledOnTouchOutside(canceledOnTouchOutside);
            mAlertDialog.setCancelable(canceledOnTouchOutside);
        }

    }

}
```

- 判断当前app是否有网络

```
public static boolean isNetworkAvailable(Context context) {
        ConnectivityManager cm = (ConnectivityManager) context
                .getSystemService(Context.CONNECTIVITY_SERVICE);
        if (cm == null) {

        } else {
            NetworkInfo[] info = cm.getAllNetworkInfo();
            if (info != null) {
                for (int i = 0; i < info.length; i++) {
                    if (info[i].getState() == NetworkInfo.State.CONNECTED) {
                        return true;
                    }
                }
            }
        }
        return false;
    }
```