[SCA] PDA UI 库介绍

PDA UI 组件库收集提炼 SLS Android PDA 项目常用组件，可供新老项目快速使用以简化开发流程。

一、内置资源

1.1 颜色

1.2 图标

1.3 插图

色彩 / Color

主色

EE4D2D

FF5722

F53D2D
FF6633

辅助色

55CC77

FFC400

FF4742

1791F2

字体

191919

8C9AA8

间隔线

E8ECF5

背景

F3F5F8

图标

999999

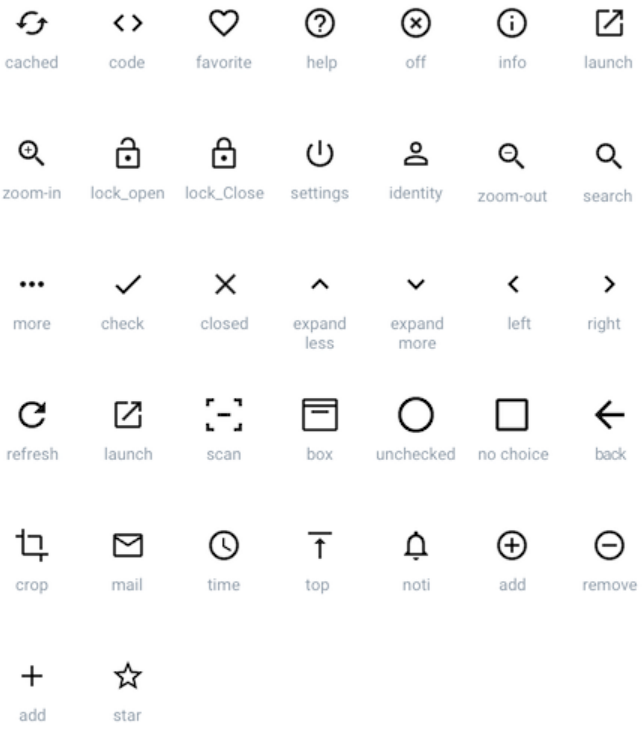
BDC6CF

FF5722

图标 / Icon

入口图标

线框图标



实底图标

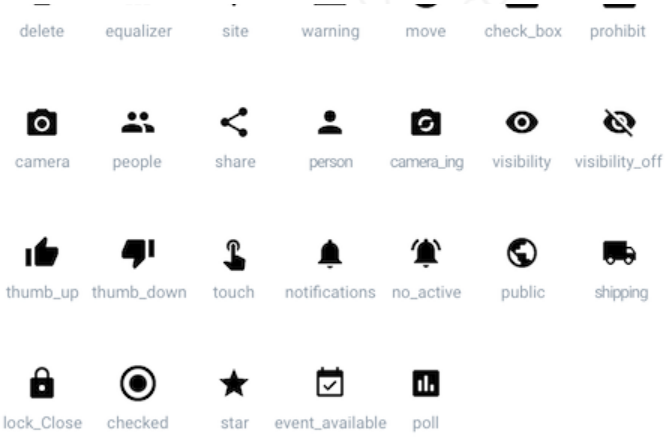

delete
equalizer
site
warning
move
check_box
prohibit
camera
people
share
person
camera_ing
visibility
visibility_off
thumb_up
thumb_down
touch
notifications
no_active
public
shipping
lock_Close
checked
star
event_available
poll

插图 / illustration

操作反馈插图

成功提示类文案

报错提示类文案

警告提示类文案

请扫描容器号，将商品收至容器

空态错误等类型插图

空态插图

404

网络错误保障等

二、基础控件

2.1 刷新标识 RoundRefreshDrawable

继承自 Drawable, 通过不断绘制两段圆弧来代表刷新状态的简单 Drawable. 可灵活集成至 View/TextView/ImageView 中。属性定义如下：

```
<declare-styleable name="RoundRefresh">
    <!-- Background color of the refresh ring. Default is black. -->
    <attr name="refreshBackgroundColor" format="color" />
    <!-- Foreground color of the refresh ring. Default is white. -->
    <attr name="refreshForegroundColor" format="color" />
    <!-- One cycle duration, unit is ms. -->
    <attr name="refreshDuration" format="integer" />
    <!-- Refresh circle stroke width. -->
    <attr name="refreshStrokeWidth" format="dimension" />
    <!-- Refresh circle radius. -->
    <attr name="refreshRadius" format="dimension" />
    <!-- Refresh circle foreground swipe angel, from 0 to 360. -->
    <attr name="refreshSwipeAngel" format="integer" />
</declare-styleable>
```

简单场景下，可有效减少 ProgressBar 的使用。

单独使用：

集成于 PdaTextView 中：

2.2 文本框/按钮 PdaTextView

继承自 AppCompatTextView，扩展实现粘性 drawableStart 及内置 RoundRefreshDrawable（可与粘性 drawableStart 同时开启）。属性定义如下：

```
<declare-styleable name="PdaTextView">
    <!-- Whether the drawableStart is sticky to the text. Default is true. -->
    <attr name="drawableStartSticky" format="boolean" />
    <!-- Where the refresh circle is showing relative to the text. Default is start. -->
    <attr name="refreshPosition">
        <enum name="top" value="0x30" />
        <enum name="bottom" value="0x50" />
        <enum name="start" value="0x00800003" />
    </attr>
    <attr name="refreshing" format="boolean" />
    <attr name="refreshBackgroundColor" />
    <attr name="refreshForegroundColor" />
    <attr name="refreshDuration" />
    <attr name="refreshStrokeWidth" />
    <attr name="refreshRadius" />
    <attr name="refreshSwipeAngel" />
</declare-styleable>
```

drawableStart 居中：

RoundRefreshDrawable 居中：

（开发缘由）当 TextView 需要左侧图标及文本共同居中显示时，一般有以下两种方法：

- a、当 TextView 的文本已知且固定时，添加 drawableStart，设置 TextView 的 gravity=start，通过调节 drawablePadding 来实现共同居中的效果；
- b、在 TextView 平级加一个 ImageView，外部再套一层 ViewGroup，来控制图标及文本的共同居中。

方法 a 虽然没有增加布局层级，但缺点很明显，文本不能任意设置、UI适配兼容性差；
方法 b 能适配各种情况，但增加了布局层级，同时 TextView 宽度不能为 match_parent，相关背景及点击事件可能要设置在上一级的 ViewGroup 上。

通过研读 Android 源码，本库提供了一种优雅的解决方式，既不增加布局层级，也能适配绝大多数情况。android.widget.TextView 类提供了一个半开放的 Api - `int getHorizontalOffsetForDrawables()`，在 onDraw() 进行绘制时，会使用该方法的返回值对 drawableStart 进行 offset，该方法有 hide 标记，但修饰符为 public，子类可重写实现自定义逻辑，达到 drawableStart 及文本共同居中的效果。

该功能默认开启，通过 `app:drawableStartSticky="false"` 可关闭。

2.3 输入框 PdaEditText

继承自 AppCompatEditText，在 AppCompatEditText 的基础上，添加下划线、字符计数、左标签（支持换行）、上标签（单行）以及字符验证等功能。属性定义如下：

```
<declare-styleable name="PdaEditText">
     <!-- The base color of the underline and the bottom text. Default is black. -->
     <attr name="pet_baseColor" format="color" />
     <!-- The underline's highlight color. Default equals to pet_baseColor. -->
     <attr name="pet_primaryColor" format="color" />
     <!-- The color for when something is wrong. (eg: exceeding max characters) -->
     <attr name="pet_errorColor" format="color" />
     <!-- Max characters count limit. Default(=0) means no limit. -->
     <attr name="pet_maxCharacters" format="integer" />
     <!--  Whether or not to show the underline. Hide by default. -->
     <attr name="pet_showUnderline" format="boolean" />
     <!-- Underline's color -->
     <attr name="pet_underlineColor" format="color" />
     <!-- Custom left label text -->
     <attr name="pet_labelLeft" format="string" />
     <!-- Custom top label text -->
     <attr name="pet_labelTop" format="string" />
     <!-- The label's text size. 12sp by default. -->
     <attr name="pet_labelTextSize" format="dimension" />
     <!-- The label's text color. Black by default. -->
     <attr name="pet_labelTextColor" format="reference|color" />
     <!-- The spacing between the main text and the label. 16dp by default. -->
     <attr name="pet_labelPadding" format="dimension" />
     <!-- The fixed width for pet_labelLeft. Default(=0) means no limit. -->
     <attr name="pet_labelLeftFixedWidth" format="dimension" />
     <!-- The bottom texts' size. 12sp by default. -->
     <attr name="pet_bottomTextSize" format="dimension" />
     <!-- The spacing between the main text and the bottom characters counter. 10dp by default. -->
     <attr name="pet_bottomTextPadding" format="dimension" />
     <!-- Whether to validate as soon as the text has changed. False by default -->
     <attr name="pet_autoValidate" format="boolean" />
     <!-- Auto validate on focus lost. False by default. -->
     <attr name="pet_validateOnFocusLost" format="boolean" />
     <!-- Whether check the characters count at the beginning. True by default. -->
     <attr name="pet_checkCharactersCountAtBeginning" format="boolean" />
     <!-- Whether input is required, red remark will be shown when this value if true. False by default. -->
     <attr name="pet_inputRequired" format="boolean" />
 </declare-styleable>
```

原理为通过增加相应 padding 来显示下划线、计数、标签等额外信息，达到单个控件实现复杂需求的效果。

基础用法：

基础用法

单项输入	后台预设内容
最多150px宽不行就折行处理	这里是填写后的内容宽度极值220

单项输入	单项输入
后台预设内容	后台预设内容

状态：

状态

标星必填 *	后台预设内容
完成输出	这里是填写后的内容
验证报错	这里是填写后的内容
不可编辑	不可编辑的内容

多行及计数：

2.4 侧滑菜单控件 HorizontalSwipeMenuLayout

继承自 FrameLayout，支持左滑/右滑菜单、自定义滑动菜单、RecyclerView 列表项自动协调。属性定义如下：

```
    <declare-styleable name="HorizontalSwipeMenuLayout">
        <!-- Left swipe menu layout id. -->
        <attr name="leftMenuId" format="reference" />
        <!-- Right swipe menu layout id. -->
        <attr name="rightMenuId" format="reference" />
        <!-- Whether the contentView can be clicked when left or right menu is open, default is false. -->
        <attr name="canClickContentWhenMenuOpen" format="boolean" />
    </declare-styleable>
```

单独使用时正常定义左/右 menu 即可，在 RecyclerView/ListView 中使用时共用一个 HorizontalSwipeMenuLayout.Coordinator 可自动协调所有 menu。

协调效果如下：

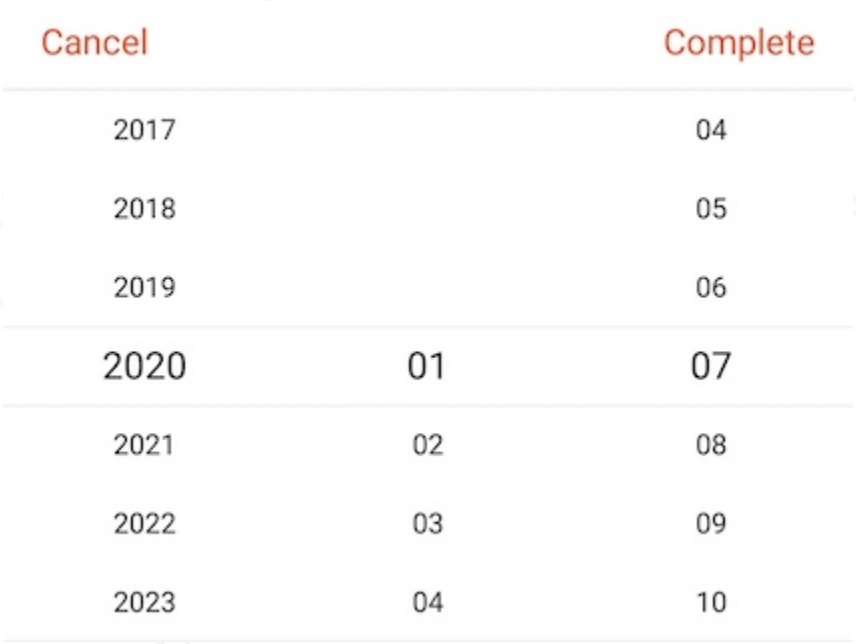
2.5 选择器 WheelView、TimePickerView

WheelView：自定义滚轮控件，继承自 View，设置 WheelAdapter 即可单独使用。属性定义如下：

```
    <declare-styleable name="WheelView">
        <attr name="wv_textSizeOut" format="dimension" />
        <attr name="wv_textSizeCenter" format="dimension" />
        <attr name="wv_textColorOut" format="color" />
        <attr name="wv_textColorCenter" format="color" />
        <attr name="wv_dividerColor" format="color" />
        <attr name="wv_itemHeight" format="dimension" />
        <attr name="wv_isCyclic" format="boolean" />
        <!--  -->
        <attr name="wv_maxItems" format="integer" />
    </declare-styleable>
```

TimePickerView：时间选择器，内部封装6个 WheelView，可进行完整年月日时分秒的选择，提供5种时间选择模式。

例：日期选择



2.6 导航栏 CommonTitle

继承自 RelativeLayout，封装一个 ImageView 及两个 TextView 组成常用导航栏。属性定义如下：

```
<declare-styleable name="CommonTitle">
    <attr name="titleStyle">
        <!-- Show title only. -->
        <enum name="titleOnly" value="0" />
        <!-- Show title and back icon. -->
        <enum name="commonGoBack" value="1" />
        <!-- Show title and both side icons. -->
        <enum name="showAll" value="2" />
    </attr>
    <!-- Set center title. -->
    <attr name="centerTitle" format="string" />
    <!-- Set right sub title. -->
    <attr name="rightTitle" format="string" />
    <!-- Set right iconsuggest just use one of rightTitle and rightIcon. -->
    <attr name="rightIcon" format="reference" />
    <attr name="showBottomDivider" format="boolean" />
</declare-styleable>
```

基础用法：

三、弹窗 & Popup & Adapter

3.1 单选 & 单选框

对于选项较少且选项已知的单选场景，我们可以使用 RadioGroup + RadioButton 来实现，如语言选择、环境选择。但当选项较多或者选项未知时，上述实现就会显得捉襟见肘。

本库使用 RecyclerView 来实现单选列表，将常用选中逻辑封装于 SingleChoiceAdapter 中；将视图创建抽象至 ChoiceViewHolder.Factory 中，并提供 6 种实现组合供选择；基本信息（title、desc、checked）的视图绑定在 ChoiceViewHolder 中自动完成。

使用时只需将数据列表及 ChoiceViewHolder.Factory 传入 SingleChoiceAdapter 即可，比使用 RadioGroup + RadioButton 更简洁。如本库提供的样式不能满足需求时，实现 ChoiceViewHolder.Factory 返回自定义 ChoiceViewHolder 即可扩展使用。

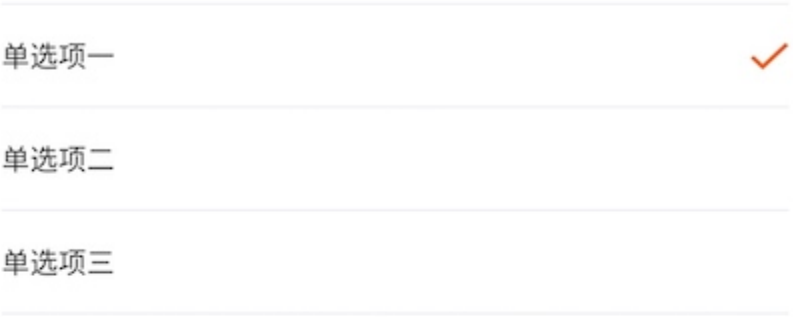
简单使用：

```
        ChoiceViewHolder.Factory tickFactory = SimpleChoiceViewHolderFactory.create(DRAWABLE_END_STYLE_TICK);
        RecyclerView recyclerBase = findViewById(R.id.recycler_radio_base);
        List<CheckedItem<String>> baseList = new ArrayList<>();
        baseList.add(new CheckedItem<>(null, true, ""));
        baseList.add(new CheckedItem<>(null, false, ""));
        baseList.add(new CheckedItem<>(null, false, ""));
        recyclerBase.setAdapter(new SingleChoiceAdapter<>(baseList, tickFactory));

        RecyclerView recyclerType1 = findViewById(R.id.recycler_radio_type1);
        List<CheckedItem<String>> typeList1 = new ArrayList<>();
        typeList1.add(new CheckedItem<>(null, false, "", ""));
        typeList1.add(new CheckedItem<>(null, true, "", ""));
        typeList1.add(new CheckedItem<>(null, false, "", ""));
        recyclerType1.setAdapter(new SingleChoiceAdapter<>(typeList1, tickFactory));
```

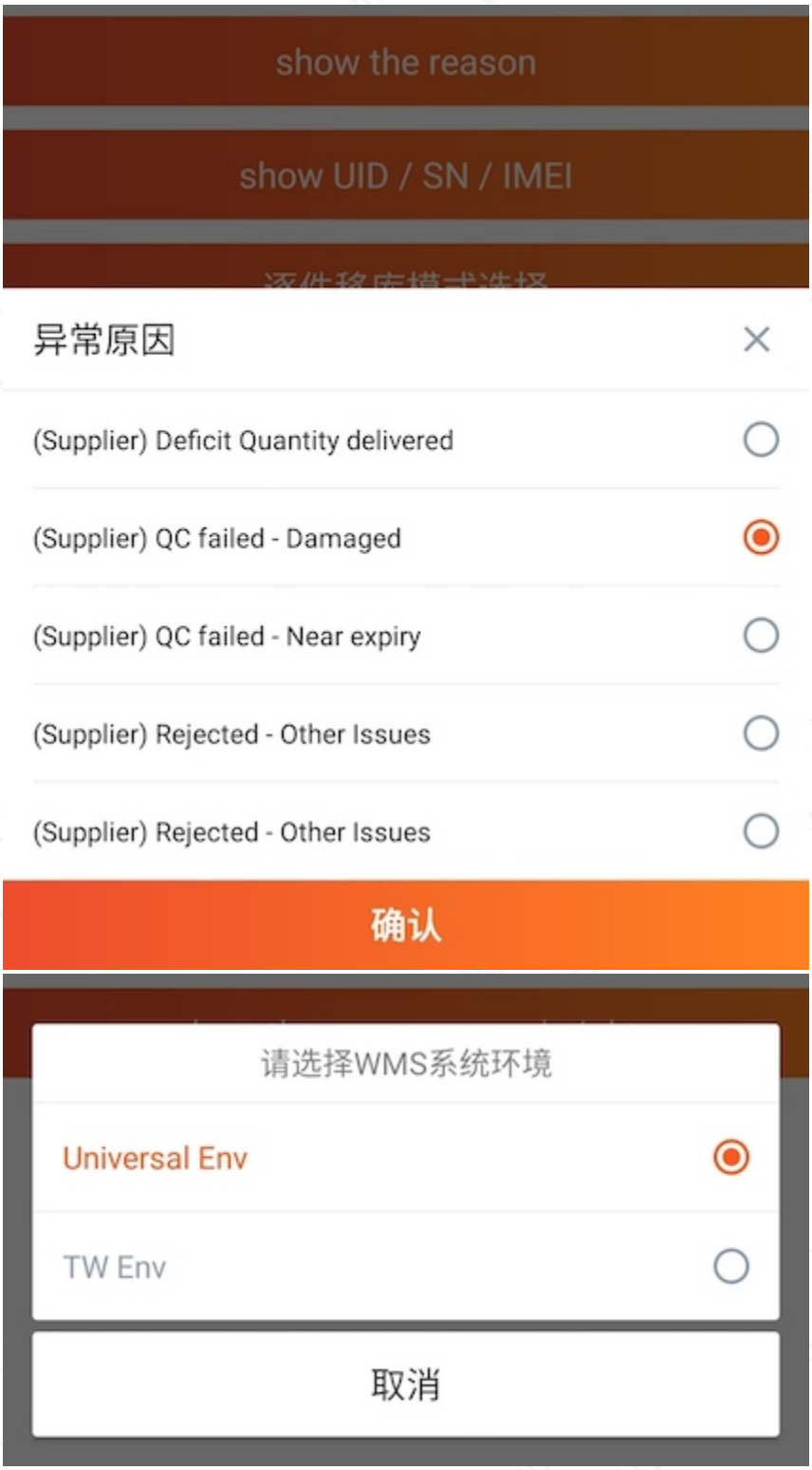
效果如下：

基础用法



类型

在 SingleChoiceAdapter 的基础上，封装了 SingleChoiceDialog 实现单选弹窗，提供以下两种风格：



3.2 复选

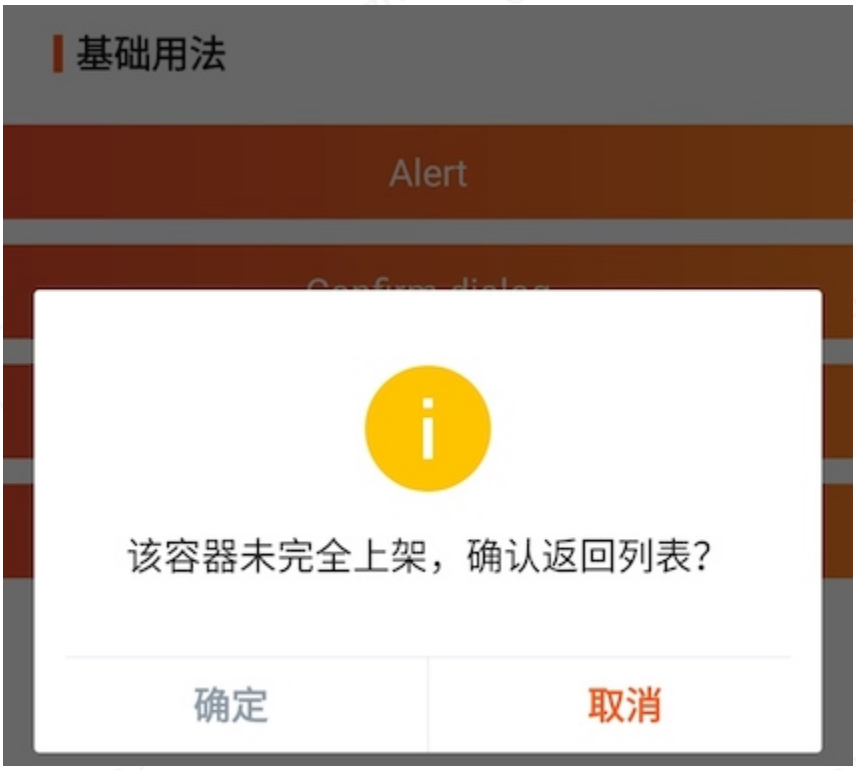
复选列表也使用 RecyclerView 来实现，将复选逻辑封装于 MultiChoiceAdapter 中，视图创建及绑定逻辑和 SingleChoiceAdapter 相同，都是使用 ChoiceViewHolder.Factory、ChoiceViewHolder。

复选列表：

3.3 对话框 NotifyDialog

对于常用的对话提示框，封装了 NotifyDialog，提供 Builder 模式对标题、描述、图标、左右按钮等信息进行设置。

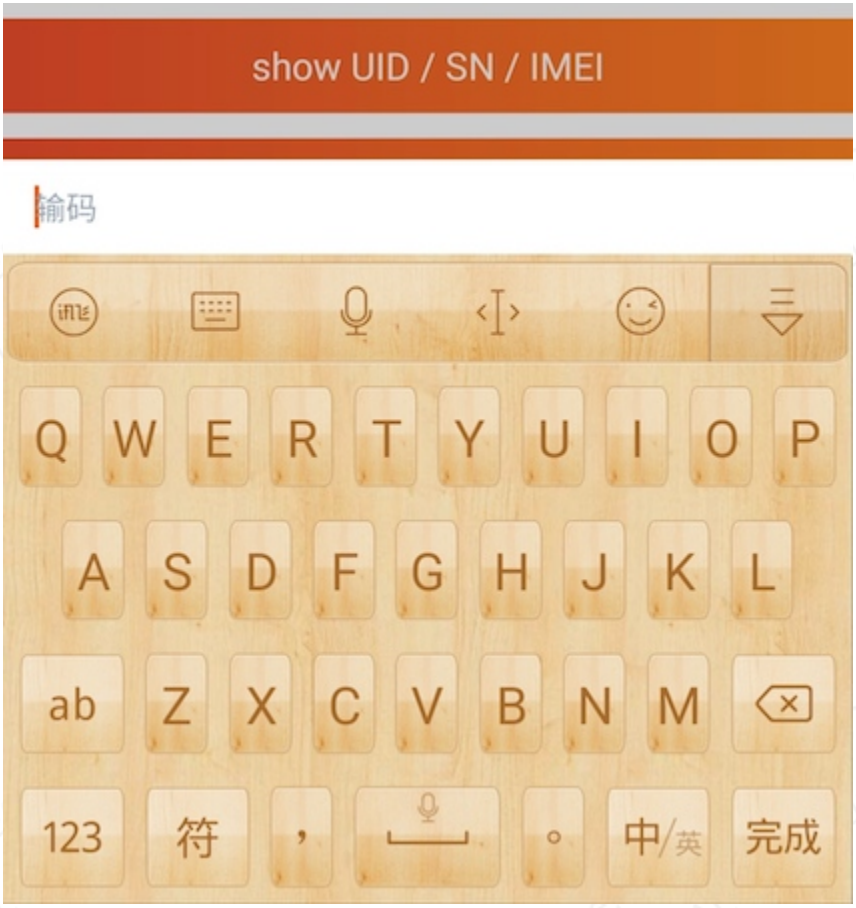
对话提示框:



3.4 Popup 输入框

本库移植了 WMS PDA 中常用的几种 Popup 输入框，其中 InputNumView 继承自 PopupWindow，通过 InputNumberHelper 辅助类来创建实际要展示的 Popup 输入框。

常用的有以下几种：


show UID / SN / IMEI
输码
Q W E R T Y U I O P
A S D F G H J K L
ab Z X C V B N M
123 符 ， 。 中/英 完成

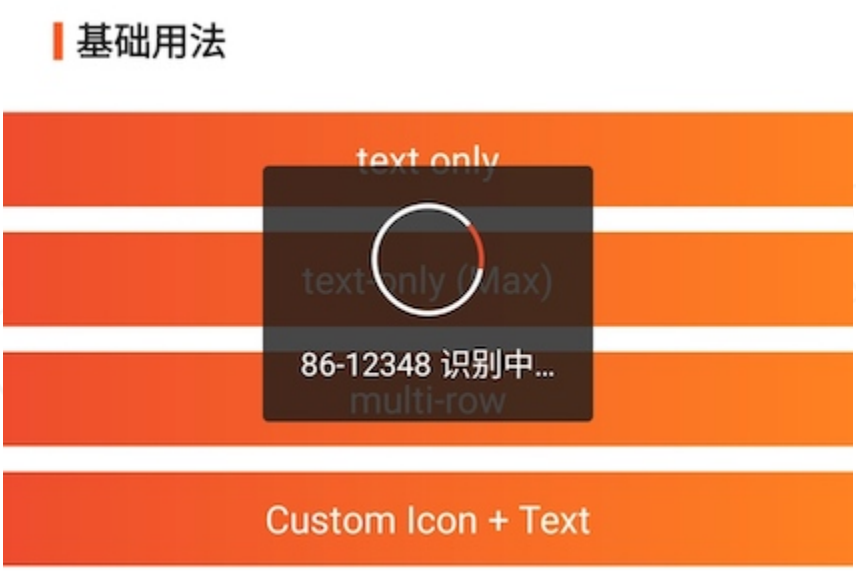
3.5 LoadingPopup

设计同学希望在 toast 提示中展示加载状态，使用系统 Toast 难以满足要求，本库采用 LoadingPopup 辅助类内部封装 PopupWindow 的方式来实现，简单高效。

简单使用方法如下：

```
    LoadingPopup mLoadingPopup = new LoadingPopup();

        //
        mLoadingPopup.showLoadingPopup(getActivity(), "86-12348 …");
        //
        mLoadingPopup.hidePopup();
```



由于各项目一般都有各自的异步实现，倒计时用法未封装至本库中，example 模块中使用 LoadingPopup 与 android.os.CountDownTimer 来实现倒计时用法：

```
        new CountDownTimer(3000, 1000) {
            @Override
            public void onTick(long millisUntilFinished) {
                long leftSec = (millisUntilFinished + 500) / 1000;
                mLoadingPopup.showLoadingPopup(getActivity(), "" + leftSec + "");
            }

            @Override
            public void onFinish() {
                mLoadingPopup.hidePopup();
            }
        }.start();
```

3.6 SimpleAdapter & ClipAdapter

SimpleAdapter：本库为 RecyclerView 封装了一套精简的 Adapter、ViewHolder -- SimpleAdapter、SimpleViewHolder，实现了基本的 Header、Footer 功能；使用方式与 BaseRecyclerViewAdapterHelper 类似，简单场景下可减少三方库的引入。

ClipAdapter：在 SimpleAdapter 的基础上，使用自定义 Footer 实现列表展开/收起功能，使用时只需传入折叠参数以及实现视图绑定即可。

四、后续计划

- 集成到项目中，在实际项目中进行锤炼与改进
- 在保持精简的同时，添加更多 feature
- 添加 androidx 版本，常规版本及 androidx 版本并行维护（当所有项目都切到 androidx 后，可以只维护 androidx 版本）