

# 移动平台应用开发

## 实验报告告

学 院 计算机学院  
年 级 2017  
班 级 1 班  
学 号 1711425  
姓 名 曹元议

2020 年 4 月 18 日

# 目录

一、实验目的.....	1
二、实验内容.....	1
2.1 Android 常用控件 .....	1
2.2 Android 常用界面布局 .....	2
2.3 Android 常用事件处理方法 .....	3
三、实验步骤及实验结果.....	4
3.1 个人信息编辑界面的实现.....	6
3.1.1 总体布局设计.....	7
3.1.2 子布局及控件的设计.....	10
3.1.3 菜单和返回键的设计 .....	45
3.2 个人信息展示界面的实现.....	48
3.2.1 布局设计.....	48
3.2.2 菜单和返回键的设计 .....	55
3.3 登录界面的实现.....	57
3.4 欢迎界面的实现.....	66
3.5 自定义对话框的实现.....	68
四、实验遇到的问题及其解决方法.....	74
五、实验结论.....	76

## 一、实验目的

熟悉 Android 开发中的 UI 设计，包括了解和熟悉常用控件的使用、界面布局和事件处理等内容。

## 二、实验内容

本次实验需要设计一个 UI 界面，要求如下：

1. 至少一个嵌套布局
2. 至少有 5 种不同的控件，该 5 种控件至少涵括 3 个类别（文本框、按钮、列表、进度条、选择器、菜单、对话框），控件总数为 10 个以上
3. 每个控件都有可操作的控件处理方法

我在这次实验要设计的内容是按照上述的实验要求做一个个人信息编辑的界面，但是后来觉得其中涉及到的所有元素都放在这一个界面当中显得太单调，就又自学并尝试了一些额外的内容，额外做了欢迎页、登录界面、个人信息显示的界面。由于这次实验主要目的是熟悉 UI 设计，所以这次实验中程序所进行的一切操作都只是简单的界面级别的操作，并不会以文件或者数据库的方式向本地保存任何内容，因此程序运行时对其中内容做的任何更改在程序退出后都将失效。也就是本次运行了程序，编辑了个人信息，下次运行程序其中的个人信息和初始的状态一样。

我在本次实验中实现的功能除了上述要求的功能之外，还有以下的拓展功能：

1. **Activity** 的简单切换以及带渐入渐出效果的 **Activity** 切换
2. **Activity** 间信息的简单传递
3. 简单的换肤功能
4. 简单的国际化功能
5. 再按一次退出程序功能

以下的内容当中，红色标记的是我在本次实验（对应上述要求的 **Activity**）中使用到的，绿色标记的只是超出上述要求的 **Activity** 中使用到的。

### 2.1 Android 常用控件

Android 的常用控件简单分类如下：

文本框: `TextView`、`EditText`  
按钮: `Button`、`RadioButton`、`RadioGroup`、`CheckBox`、`ImageButton`  
列表: `ListView`、`ExpandableListView`、`Spinner`、`AutoCompleteTextView`、  
`GridView`、`ImageView`  
进度条: `ProgressBar`、`ProgressDialog`、`SeekBar`、`RatingBar`  
选择器: `DatePicker` (`DatePickerDialog`)、`TimePicker`  
(`TimePickerDialog`)  
菜单: `Menu`、`ContextMenu`  
对话框: `Dialog`、`ProgressDialog`

常用的控件有文本框、按钮和列表等。

可以直接在 Java 类中为控件设置属性，也可以在 xml 文件中设置属性。每个控件都有一系列的属性，例如 id、size、text、color 等等，其他控件属性的设置可以查看 API。

(关于 `ImageView`，我使用了一个第三方开源类，继承自 `ImageView`，可以将图片以圆形形状显示)

## 2.2 Android 常用界面布局

**FrameLayout:** 最简单的一个布局对象。它里面只显示一个显示对象。Android 屏幕元素中所有的显示对象都将会固定在屏幕的左上角，不能指定位置。但允许有多个显示对象，但后一个将会直接在前一个之上进行覆盖显示，把前一个部份或全部挡住（除非后一个是透明的）。

**LinearLayout:** 以单一方向对其中的显示对象进行排列显示，如以垂直排列显示，则布局管理器中将只有一列；如以水平排列显示，则布局管理器中将只有一行。同时，它还可以对个别的显示对象设置显示比例。

**TableLayout:** 继承自 `LinearLayout`。以拥有任意行列的表格对显示对象进行布局，每个显示对象被分配到各自的单元格之中，但单元格的边框线不可见。

**AbsoluteLayout:** 允许以坐标的方式，指定显示对象的具体位置，左上角的坐标为(0, 0)，向下及向右，坐标值变大。这种布局管理器由于显示对象的位置定死了，所以在不同的设备上，有可能会出现最终的显示效果不一致。

**RelativeLayout:** 允许通过指定显示对象相对于其它显示对象或父级对象的相对位置来布局。如一个按钮可以放于另一个按钮的右边，或者可以放在布局管理器的中央。

**ConstraintLayout:** 是 `RelativeLayout` 的升级版，可以减少对布局层次的嵌

套，进而提高 app 的性能。这种布局管理器和 `RelativeLayout` 一样都是通过相对位置来对控件进行布局的。相比 `RelativeLayout` 而言，`ConstraintLayout` 对位置的描述更加灵活，也可以更方便地在可视化视图中拖拽控件。

`ScrollView`: 继承自 `FrameLayout`。当绘制的控件超出手机屏幕尺寸的时候，可以采用（垂直）滚动的方式，使控件显示。

布局中可以放置控件，而每个布局又可以嵌套其他布局，这种思想和之前学习 java 的布局是一样的。

## 2.3 Android 常用事件处理方法

`void onClick(View v)` 一个普通的点击按钮事件

`boolean onKeyDown(int keyCode, KeyEvent event)`  
用于在多个事件连续时发生，用于按键重复

`boolean onKeyUp(int keyCode, KeyEvent event)` 用于在按键进行按下时发生

`boolean onTouchEvent(MotionEvent event)` 触摸屏事件，当在触摸屏上有动作发生

`boolean onKeyLongPress(int keyCode, KeyEvent event)` 当长时间按时发生

`boolean onKey(View v, int keyCode, KeyEvent event)` 捕获针对 `EditText` 的按键动作

`void onClick(DialogInterface dialog, int which)` 针对对话框中控件的单击事件

`void onCheckedChanged(RadioGroup group, int checkedId)` `RadioGroup` 中选中的 `RadioButton` 改变时发生

`void onItemSelected(AdapterView<?> parent, View view, int position, long id)` `Spinner` 的某一项被选中时发生

`void onCheckedChanged(CompoundButton buttonView, boolean isChecked)` 当 `CheckBox` 选中的状态变化时发生

`void beforeTextChanged(CharSequence s, int start, int count, int after)` `EditText` 的内容改变之前（将要改变时）发生

```
void onTextChanged(CharSequence s, int start, int before, int count)
```

EditText 的内容改变时（改变过程中）发生

```
void afterTextChanged(Editable s) EditText 的内容改变之后发生
```

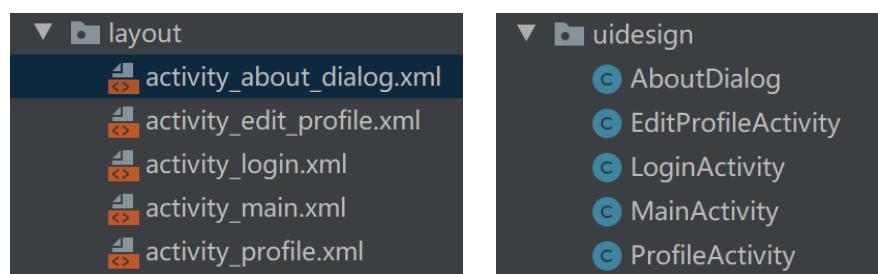
```
boolean onOptionsItemSelected(MenuItem item) 当某个菜单项被选中时  
发生
```

### 三、实验步骤及实验结果

本次实验的步骤为：编写代码→测试实验结果。

本次实验的编程部分主要分为两个步骤：设计页面布局、实现控件处理方法。

本次实验涉及到的布局文件和与布局相对应的 Java 文件如下图所示：



其中，布局相对应的 Java 文件都在包 com.example.uidesign 中。

activity\_edit\_profile.xml 和 EditProfileActivity.java 对应本次实验按照要求设计的个人信息编辑的界面，由于篇幅所限，下面我会着重描述这个界面的相关设计，其他界面主要选择性展示额外功能和有特殊设置的控件。

activity\_about\_dialog.xml 和 AboutDialog.java 在自定义对话框上，

activity\_login.xml 和 LoginActivity.java 用在登录界面上，

activity\_main.xml 和 MainActivity.java 用在欢迎界面上，

activity\_profile.xml 和 ProfileActivity.java 用在个人信息展示界面上。

以下是本次实验的工程文件中的 `AndroidManifest.xml` 文件，可以看到包括

AboutDialog 以内有 5 个 Activity。

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="com.example.uidesign">
4
5      <application
6          android:allowBackup="true"
7          android:icon="@mipmap/ic_launcher"
8          android:label="@string/app_name"
9          android:roundIcon="@mipmap/ic_launcher_round"
```

```
10     android:supportsRtl="true"
11     android:theme="@style/AppTheme">
12         <activity android:name=".EditProfileActivity" />
13         <activity android:name=".ProfileActivity" />
14         <activity
15             android:name=".AboutDialog"
16             android:theme="@style/Theme.AppCompat.DayNight.Dialog" />
17         <activity android:name=".LoginActivity" />
18         <activity
19             android:name=".MainActivity"
20             android:theme="@style/Theme.AppCompat.DayNight.NoActionBar">
21             <intent-filter>
22                 <action android:name="android.intent.action.MAIN" />
23
24                 <category android:name="android.intent.category.LAUNCHER">
25             />
26             </intent-filter>
27         </activity>
28     </application>
29 </manifest>
```

属性 `android:name="android.intent.action.MAIN"` 表示这个 `Activity` 是程序启动时加载的 `Activity`。

一个贯穿整个程序的类——`Profile` 类，在 `com.example.profile` 包中。用于获取和设置個人資料。其中，类变量和成员变量的定义如下：

```
public static final String[] star_sign = {"摩羯座", "水瓶座", "双鱼座", "白羊座", "金牛座",
                                         "双子座", "巨蟹座", "狮子座", "处女座", "天秤座", "天蝎座", "射手座"};
public static final String zodiac = "猴鸡狗猪鼠牛虎兔龙蛇马羊";
public static final String[] provinces = {"北京", "天津", "河北", "山西", "内蒙古", "辽宁", "吉林",
                                         "黑龙江", "上海", "江苏", "浙江", "安徽", "福建", "江西", "山东", "河南", "湖北", "湖南", "广东",
                                         "广西", "海南", "重庆", "四川", "贵州", "云南", "西藏", "陕西", "甘肃", "青海", "宁夏", "新疆",
                                         "台湾", "香港", "澳门"};
public static final String[] jobs = {"学生", "从政", "领域大佬", "普通上班族", "程序员",
                                    "老板", "搬砖工人", "待在家", "其他"};
public static final String[] hobbies = {"旅行和摄影", "文学和艺术", "体育运动", "美食", "影视和音乐",
                                       "休闲娱乐", "社交", "学习", "宅家", "睡觉"};
```

```
public final String USERID = "1711425"; // 用户名
private String password = "123456"; // 密码
private String nickname; // 昵称
private LoginFrom from; // 登录方式
private Sex sex; // 性别
private String motto; // 个性签名
private int location; // 所在地(索引)
private int job; // 职业身份(索引)
private String email; // 电子邮箱
private String phone; // 手机号
private int birthYear; // 出生年
private int birthMonth; // 出生月份
private int birthDay; // 出生日
private int likes; // 点赞数
private boolean[] hobby; // 数组内某个元素为true则表示拥有相应的兴趣爱好
```

上图中出现的用户名和密码是在下面介绍的登录界面当中要输入的唯一正确的用户名的密码。其中枚举类型 `Sex` 和 `LoginFrom`, 分别在 `com.example.profile` 包的 `Sex.java` 和 `LoginFrom.java` 文件中, 定义如下:

```
public enum Sex { FEMALE, MALE };

public enum LoginFrom {
    WEIXIN, WEIBO, QQ, NAME
}
```

在后续部分代码的描述中会出现 `Profile` 类里面的成员函数调用, 由于这些函数定义大多不涉及实现原理, 就不一一介绍和注释了, 函数名可以反映这个函数的具体作用。

在 `com.example.profile` 包中还有一个 `ShareProfile` 类, 用于定义在多个 `Activity` 之间共享的变量 (有点类似 C/C++ 的全局变量) 定义如下:

```
public class ShareProfile {
    public static Profile profile = new Profile(); // 用于展示的个人资料
    public static int theme = 0; // 设置的主题
    public static boolean liked = false; // 是否点赞过
}
```

### 3.1 个人信息编辑界面的实现

在 `EditProfileActivity.java` 的类 `EditProfileActivity` 中, 定义了一个 `Profile` 类的成员变量 `tempProfile`, 用于临时保存在此 `Activity` 中做的更改。

在 `onCreate()` 函数中, 会通过 `Profile` 类的成员函数 `copyProfile()`, 将参

数对应的 **Profile** 类的实例的内容拷贝到调用该函数的 **Profile** 类对象当中，定义如下：

```
public void copyProfile(Profile temp) {  
    this.nickname = temp.nickname;  
    this.sex = temp.sex;  
    this.location = temp.location;  
    this.job = temp.job;  
    this.birthDay = temp.birthDay;  
    this.birthMonth = temp.birthMonth;  
    this.birthYear = temp.birthYear;  
    this.motto = temp.motto;  
    this.email = temp.email;  
    this.phone = temp.phone;  
    this.hobby = temp.hobby;  
    this.hobby = new boolean[hobbies.length];  
    System.arraycopy(temp.hobby, srcPos: 0, this.hobby, destPos: 0, temp.hobby.length);  
    this.likes = temp.likes;  
}
```

在 **onCreate()** 函数中，变量 **tempProfile** 通过调用 **copyProfile()** 函数将 **ShareProfile.profile** 中的内容拷贝给 **tempProfile** 当中，以 **tempProfile** 当中的内容对 **EditProfileActivity** 的显示进行初始化，做的一切更改也保存在 **tempProfile** 中。如下图所示：

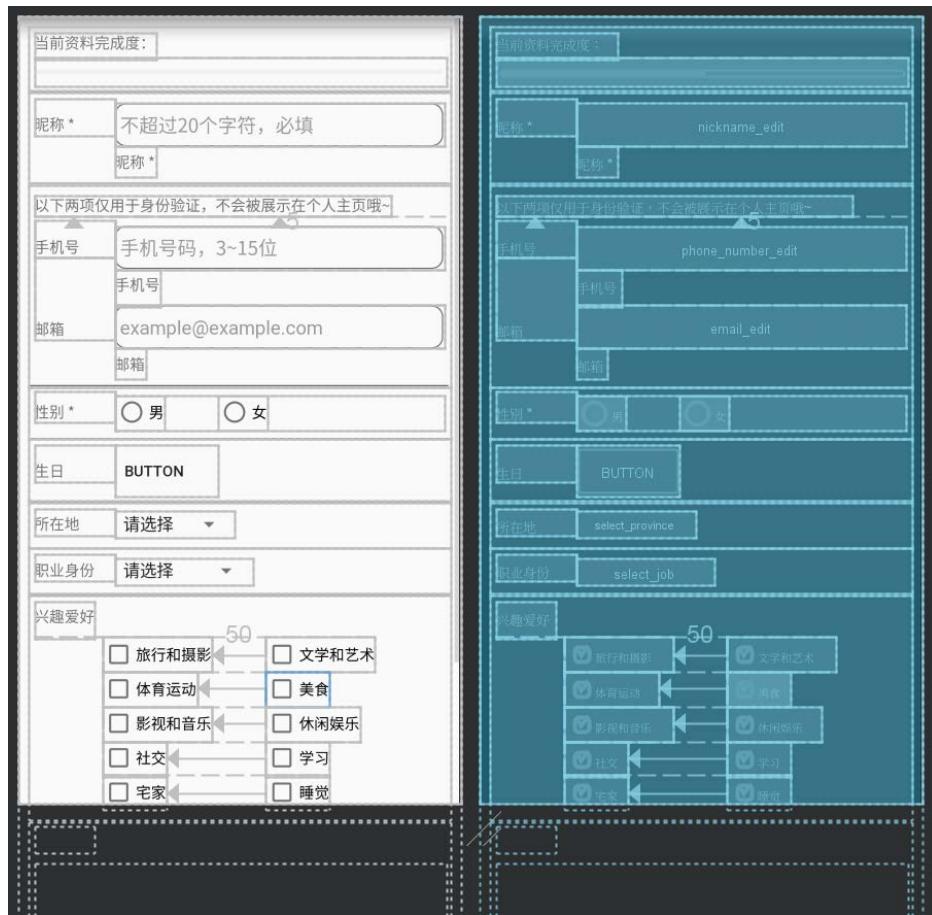
```
/* 生成临时资料，保存时将临时资料中的内容copy过去 */  
tempProfile.copyProfile(ShareProfile.profile);
```

为了方便起见，保存功能的实现是 **ShareProfile.profile** 通过调用 **copyProfile()** 函数将 **tempProfile** 的内容拷贝给 **ShareProfile.profile**。这样个人信息展示界面 **ProfileActivity** 就能使用 **ShareProfile.profile** 当中的内容进行展示了。

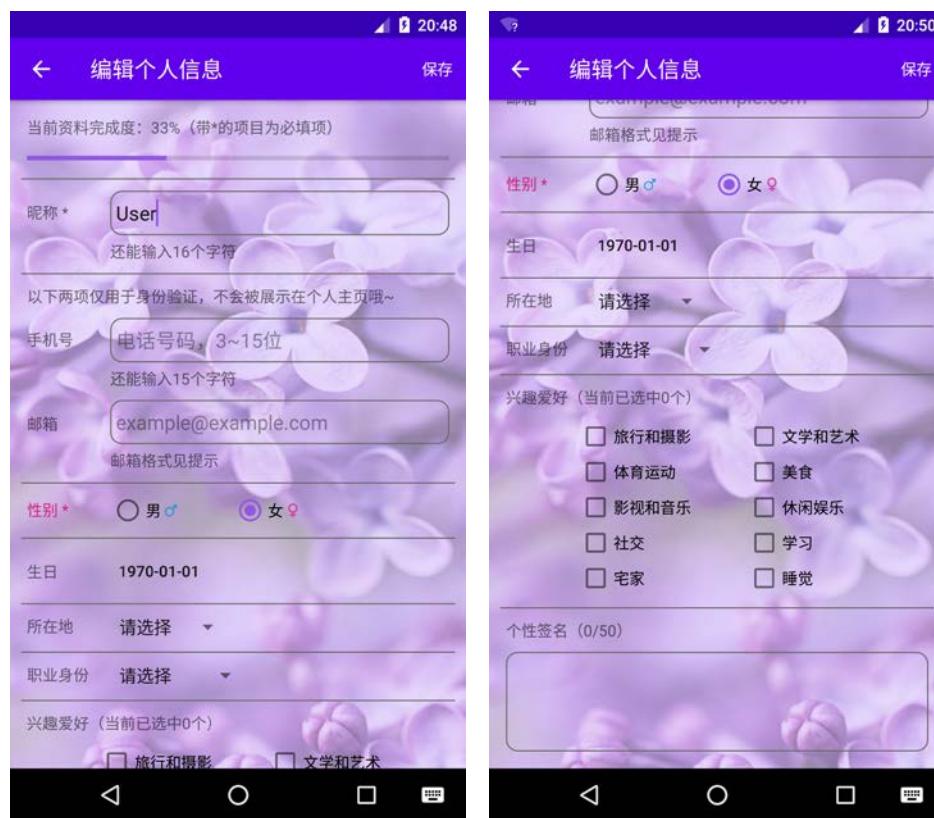
**EditProfileActivity** 类还定义了一个成员变量 **hasError**，用于记录编辑个人信息的过程中产生的错误，如果这个变量的值不为 0，则无法保存。

### 3.1.1 总体布局设计

从 **activity\_edit\_profile.xml** 文件的设计视图可以看到这个界面的大致布局情况如下。可以看出在这个视图当中，由于屏幕长度的原因没有把显示所有的控件，但是这个布局超出了屏幕的一部分，是因为最外层使用了 **ScrollView**，表示下面还有没显示的控件，这些没显示的控件通过滚动页面可以看到。

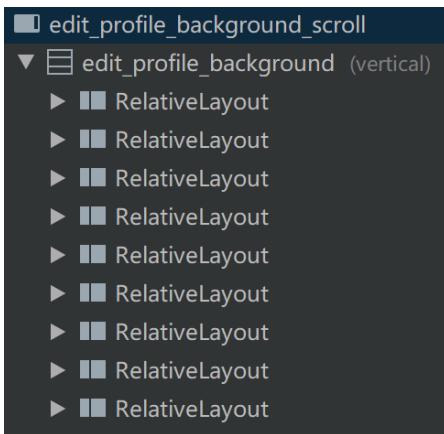


以下是运行程序之后这个界面的实际效果：



其中，标题“编辑个人信息”通过 `EditProfileActivity.java` 的 `EditProfileActivity` 类的 `onCreate()` 函数调用的 `setTitle()` 函数动态设置。

进入 `activity_edit_profile.xml` 的编辑和预览视图，整体布局结构如下：



首先最外层的布局 `edit_profile_background_scroll` 是 `ScrollView` 实现界面的滚动，里面嵌套了一层 `id` 为 `edit_profile_background` 的 `LinearLayout`，这个 `LinearLayout` 里面又嵌套了若干个 `RelativeLayout`。以下是最外层的 `ScrollView` 的属性设置：

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools"
4      android:id="@+id/edit_profile_background_scroll"
5      android:layout_width="fill_parent"
6      android:layout_height="fill_parent"
7      android:orientation="vertical"
8      android:fadingEdge="vertical"
9      android:gravity="center"
10     tools:context=".EditProfileActivity">
11
12     <LinearLayout ...>
446   </ScrollView>
```

`id` 为 `edit_profile_background` 的 `LinearLayout` 的属性设置如下：

```
<LinearLayout
    android:id="@+id/edit_profile_background"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:gravity="center"
    android:divider="@drawable/divider"
    android:showDividers="middle"
    android:padding="10dip">
```

由属性 `android:orientation="vertical"` 可以看出这个 `LinearLayout` 是垂直方向的，设置属性 `android:padding="10dip"` 的原因是设置直接子控件或直接子布局距离这个 `LinearLayout` 的上下左右边的距离，这样就为子控件和子布局提供了一个内边框，以免它们贴着屏幕边缘显示而导致显示效果不美观。属性 `android:divider="@drawable/divider"` 来设置分割线样式，属性 `android:showDividers="middle"` 是为了能够在 `LinearLayout` 中的每两个直接子控件或直接子布局之间显示分割线（分割线的效果在上面的界面运行的实际效果图中）。分割线所对应的文件 `res/drawable/divider.xml` 内容如下：

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <shape xmlns:android="http://schemas.android.com/apk/res/android">
3   <solid android:color="@color/gray"/>
4   <size android:height="1dp"/>
5 </shape>
```

其中标签 `<solid>` 用于设置分割线的填充颜色，`<size>` 设置尺寸（即分割线的粗细）。

### 3.1.2 子布局及控件的设计

#### 第1个 `RelativeLayout`

第1个 `RelativeLayout` 如下所示，这个是容纳进度条及相关控件的子布局，用于指示个人资料填写的完整度：

```
<RelativeLayout>
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="5dip">

    <TextView
        android:id="@+id/profile_progress_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingBottom="5dip"
        android:text="当前资料完成度: ">
    </TextView>
    <ProgressBar
        android:id="@+id/profile_progress"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:paddingTop="5dip"
        android:paddingBottom="5dip"
        android:layout_below="@+id/profile_progress_text"
        style="?android:attr/progressBarStyleHorizontal"
        android:max="100">
    </ProgressBar>

</RelativeLayout>
```

其中，`TextView`用于指示进度条的具体进度值。`ProgressBar`的属性  
`style="?android:attr/progressBarStyleHorizontal"`用于设置进度条的  
样式为水平长条形（默认是圆形且一直在转的进度条），属性  
`android:max="100"`用于设置进度条的最大进度所对应的值。  
`android:layout_below="@+id/profile_progress_text"`属性表明这个进度  
条的相对位置在这个`TextView`的下方。  
在`EditProfileActivity.java`的类`EditProfileActivity`中，  
`setEditProgress()`是与进度条控制相关的函数，如下图所示：

```

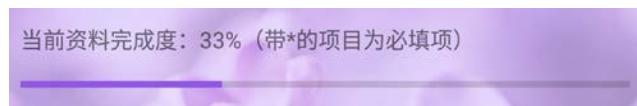
private void setEditProgress(int index) {
    final ProgressBar progress = (ProgressBar) findViewById(R.id.profile_progress);
    final TextView progressText = (TextView) findViewById(R.id.profile_progress_text);
    sum += index;
    if (sum < 0 || sum > 100) { // 设置进度条的值不能超出范围
        throw new IndexOutOfBoundsException("Progress out of bounds!");
    }
    progress.setProgress(sum);
    progressText.setText(String.format("当前资料完成度: %d%% (%带*的项目为必填项)", sum));
    if (sum == 100) {
        Drawable drawable = getResources().getDrawable(R.drawable.happy);
        drawable.setBounds( left: 0, top: 0, right: 55, bottom: 55 );
        progressText.setCompoundDrawables( left: null, top: null, drawable, bottom: null );
    }
    else {
        progressText.setCompoundDrawables( left: null, top: null, right: null, bottom: null );
    }
}

```

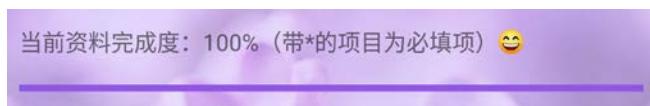
其中，参数 `index` 设置进度条值的增量，如果小于 `0`，进度条的值将减小。

进度条显示的效果如下：

这是进度还没到 100% 时的效果：



如果进度达到了 100%，则后面会显示一个笑脸：



## 第2个RelativeLayout

第 2 个 `RelativeLayout` 如下图所示，是容纳昵称填写框及相关控件的子布局，对这一项设置的进度值为 11%：

The image shows the Android Studio layout editor. On the left, there is an XML code snippet for a `RelativeLayout` containing three child views: `<EditText...>`, `<EditText...>`, and `<EditText...>`. On the right, the corresponding visual representation of the layout is shown in the preview pane, featuring three text input fields labeled `nickname_text`, `nickname_edit`, and `nickname_tips`.

```

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="5dp">

    <EditText...>
    <EditText...>
    <EditText...>

</RelativeLayout>

```

其中第一个 id 为 nickname\_text 对应的 TextView 的属性设置如下：

```
<TextView  
    android:id="@+id/nickname_text"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:paddingTop="15dip"  
    android:maxWidth="75dip"  
    android:minWidth="75dip"  
    android:text="昵称 *">
```

用于指示右边的 EditText 是一个昵称填写框。

第二个 id 为 nickname\_tips 的 TextView 用于指示昵称是否为空，还可以再输入多少个字符，属性设置如下：（其中初始文本 android:text 是我随便设置的，因为这个 TextView 的文本内容是会动态变化的）

```
<TextView  
    android:id="@+id/nickname_tips"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@id/nickname_edit"  
    android:layout_alignStart="@id/nickname_edit"  
    android:layout_alignLeft="@id/nickname_edit"  
    android:paddingTop="5dip"  
    android:paddingBottom="5dip"  
    android:text="昵称 *"  
</TextView>
```

android:layout\_below="@id/nickname\_edit" 属性表明这个 TextView 的相对位置在这个 EditText 的下方。属性 android:layout\_alignStart 和 android:layout\_alignLeft 表明该控件的左边缘和这个 EditText 左边缘对齐，这两个属性的含义也是一样的，同时设置也是为了兼容 Android 系统的高低版本。

id 为 nickname\_edit 对应的 EditText 是昵称填写框，属性设置如下：

```
<EditText  
    android:id="@+id/nickname_edit"  
    android:layout_width="wrap_content"  
    android:layout_height="40dip"  
    android:hint="不超过20个字符，必填"  
    android:layout_toRightOf="@id/nickname_text"  
    android:layout_toEndOf="@id/nickname_text"  
    android:minWidth="315dip"  
    android:maxWidth="315dip"  
    android:layout_marginTop="5dip"  
    android:padding="5dip"  
    android:background="@drawable/motto_edit_text"  
    android:inputType="text"  
    android:autofillHints="">  
</EditText>
```

其中属性 `android:layout_toRightOf` 和 `android:layout_toEndOf` 表明这个 `EditText` 的相对位置在上面介绍的 `TextView` 的右边，这两个属性的含义是一样的，同时设置的原因是为了兼容 Android 系统的高低版本。属性 `android:inputType="text"` 用于设置 `EditText` 接受的输入数据类型，这里是普通文本类型。`android:background="@drawable/motto_edit_text"` 是来设置 `EditText` 的背景样式的（效果在上面的界面运行的实际效果图中，形状设置成了圆角矩形，默认的样式是底部有一条横线），圆角矩形框对所对应的文件 `res/drawable/motto_edit_text.xml` 如下：

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <shape xmlns:android="http://schemas.android.com/apk/res/android">
3      <corners android:radius="10dip" />
4      <stroke android:color="@color/gray" android:width="1dip" />
5  </shape>
```

其中 `<shape>` 标签用于设置布局的形状，`<corners>` 标签中的属性 `android:radius` 设置圆角的半径大小，`<stroke>` 标签用于设置布局形状边框的颜色和粗细。

因此上面的 `EditText` 设置属性 `android:padding="5dip"` 的原因也是为了不要让 `EditText` 里面的文字挨住圆角矩形的边框。

在 `EditProfileActivity.java` 的 `EditProfileActivity` 类的 `onCreate()` 方法中，对这三个控件的相关设置如下。这里设置昵称的长度不超过 20 个字符。

```
1  /* 昵称设置 */
2  // 在设置监听器之前先根据 tempProfile 的昵称内容对界面显示的内容进行初始化
3  final EditText nicknameEdit = (EditText)
4      findViewById(R.id.nickname_edit);
5  final TextView nicknameText = (TextView)
6      findViewById(R.id.nickname_text);
7  final TextView nicknameTips = (TextView)
8      findViewById(R.id.nickname_tips);
9  nicknameEdit.setText(tempProfile.getNickname());
10 nicknameEdit.setSelection(tempProfile.getNickname().length()); // 设置默认光标位置
11 final int nicknameTipsColor = nicknameTips.getCurrentTextColor(); // 获取原来的文字颜色
12 final int nicknameTextColor = nicknameText.getCurrentTextColor(); // 获取原来的文字颜色
```

```
10     nicknameTips.setText(String.format(getString(R.string.nickname_tips_
content), // R.string.nickname_tips_content: 还能输入%d个字符
11             nicknameCharMaxNum - nicknameEdit.getText().length())));
12     if (nicknameEdit.getText().toString().length() > 0) {
13         setEditProgress(11);
14     }
15     else {
16
17         nicknameTips.setTextColor(getResources().getColor(R.color.errorColor
)); // R.color.errorColor: #810000, 红色
18         nicknameText.setTextColor(getResources().getColor(R.color.errorColor
));
19         hasError++;
20     }
21     // 设置控件的监听事件
22     nicknameEdit.addTextChangedListener(new TextWatcher() {
23         private CharSequence temp;// 监听前的文本
24         private int editStart;// 光标开始位置
25         private int editEnd;// 光标结束位置
26
27         @Override
28         public void beforeTextChanged(CharSequence s, int start, int count,
29             int after) {
30             temp = s;
31             if (s.length() == 0) { // 如果字数是从0到1, 则设置进度条, 恢
32                 nicknameTips.setOriginalColor(nicknameTipsColor);
33                 nicknameText.setOriginalColor(nicknameTextColor);
34                 hasError--; // 减去这个编辑错误
35             }
36
37         @Override
38         public void onTextChanged(CharSequence s, int start, int before,
39             int count) {
40             nicknameTips.setText(String.format(getString(R.string.nickname_tips_
content),
41                     nicknameCharMaxNum - s.length())));
42         }
43     });
44 }
```

```

43     @Override
44     public void afterTextChanged(Editable s) {
45         editStart = nicknameEdit.getSelectionStart();
46         editEnd = nicknameEdit.getSelectionEnd();
47         if (temp.length() == 0) {
48
49             nicknameTips.setText(getString(R.string.nickname_cannot_empty)); // R.string.nickname_cannot_empty: 昵称不能为空!
50
51             nicknameTips.setTextColor(getResources().getColor(R.color.errorColor)); // 如果输入之后的昵称是空的，则设置 nicknameTips 的颜色为红色
52
53             nicknameText.setTextColor(getResources().getColor(R.color.errorColor));
54             setEditProgress(-11); // 如果输入之后的昵称是空的，则不能算上填写昵称的进度
55             hasError++; // 同时增加一个编辑错误
56         }
57         else if (temp.length() > nicknameCharMaxNum) { // 如果昵称字数达到字数限制，新的字符将被丢弃，之后不能再输入新的字符
58             s.delete(editStart - 1, editEnd); // 删去新输入的超出字数限制的那部分字符
59             int tempSelection = editStart;
60             nicknameEdit.setText(s); // 将 nicknameEdit 的文本恢复输入前的文本
61             nicknameEdit.setSelection(tempSelection); // 重新设置光标位置在输入前光标的位置
62         } // 注意由于达到字数限制之后，字数并没有超出限制，所以不改变进度条
63
64         tempProfile.setNickname(nicknameEdit.getText().toString());
65     }
66 });
67 nicknameText.setOnClickListener(rand);
68 nicknameTips.setOnClickListener(rand);

```

实现的效果如下所示：

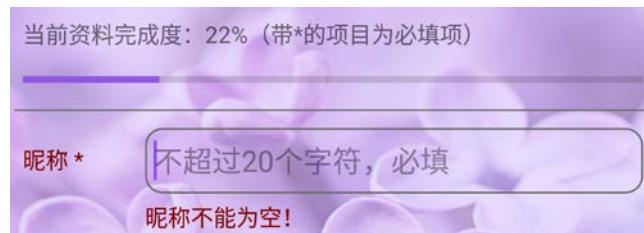
显示的初始状态如下：



添加了一些字符之后的状态如下，当输入字数达到限制后无法再输入新的字符：



删除了所有字符以后，两个 `TextView` 内容变红，表示有错误（昵称不能为空），进度条的值也相应的减去了 11：



重新输入，进度条的值恢复：



### 第3个 `RelativeLayout`

第3个 `RelativeLayout` 如下图所示，是容纳手机号和邮箱填写框及相关控件的子布局，对手机号和邮箱设置的进度值各为 11%：

```
<RelativeLayout  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:padding="5dip">  
    <TextView...>  
    <TextView...>  
    <EditText...>  
    <TextView...>  
    <TextView...>  
    <EditText...>  
    <EditText...>  
    <TextView...>  
</RelativeLayout>
```

▼ ■■■ RelativeLayout

- Ab must\_hint\_text "@string/must\_hint"
- Ab phone\_number\_text "@string/phone\_number"
- Ab phone\_number\_edit (Phone)
- Ab phone\_number\_tips "@string/phone\_number"
- Ab email\_text "@string/email"
- Ab email\_edit (E-mail)
- Ab email\_tips "@string/email"

第 1 个 `id` 为 `must_hint_text` 对应的 `TextView`, 用于提示手机号和邮箱只做身份验证 (在这个程序里面没有相关的验证), 不会作为个人信息展示, 如果手机号和邮箱的其中一项被修改, 则该文字的颜色变为绿色。属性设置如下:

```
<TextView  
    android:id="@+id/must_hint_text"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_alignLeft="@+id/phone_number_text"  
    android:layout_alignStart="@+id/phone_number_text"  
    android:layout_marginTop="5dip"  
    android:layout_marginBottom="5dip"  
    android:text="以下两项仅用于身份验证, 不会被展示在个人主页哦~">
```

第 2 个 `id` 为 `phone_number_text` 对应的 `TextView` 属性设置如下:

```
<TextView  
    android:id="@+id/phone_number_text"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/must_hint_text"  
    android:paddingTop="15dip"  
    android:maxWidth="75dip"  
    android:minWidth="75dip"  
    android:text="手机号">
```

第 3 个 `id` 为 `phone_number_edit` 对应的 `EditText` 为手机号编辑框, 属性设置如下:

```
<EditText
    android:id="@+id/phone_number_edit"
    android:layout_width="wrap_content"
    android:layout_height="40dip"
    android:layout_below="@+id/must_hint_text"
    android:layout_marginTop="5dip"
    android:hint="手机号码，3~15位"
    android:layout_toRightOf="@+id/phone_number_text"
    android:layout_toEndOf="@+id/phone_number_text"
    android:minWidth="315dip"
    android:maxWidth="315dip"
    android:padding="5dip"
    android:background="@drawable/motto_edit_text"
    android:inputType="phone"
    android:autofillHints="">
</EditText>
```

注意属性 `android:inputType="phone"` 将输入文本的属性设置为电话号码，此时弹出的虚拟键盘将是拨号键盘。

第 4 个 `id` 为 `phone_number_tips` 对应的 `TextView` 手机号输入的提示，属性设置如下：

```
<TextView
    android:id="@+id/phone_number_tips"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/phone_number_edit"
    android:layout_alignStart="@+id/phone_number_edit"
    android:layout_alignLeft="@+id/phone_number_edit"
    android:paddingTop="5dip"
    android:paddingBottom="10dip"
    android:text="手机号">
</TextView>
```

第 5 个 `id` 为 `email_text` 对应的 `TextView` 属性设置如下

```
<TextView
    android:id="@+id/email_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="57dip"
    android:layout_below="@+id/phone_number_text"
    android:layout_alignLeft="@+id/phone_number_text"
    android:layout_alignStart="@+id/phone_number_text"
    android:maxLength="75dip"
    android:minWidth="75dip"
    android:text="邮箱">
</TextView>
```

第 6 个 `id` 为 `email_edit` 对应的 `EditText` 为邮箱编辑框，属性设置如下：

```
<EditText  
    android:id="@+id/email_edit"  
    android:layout_width="wrap_content"  
    android:layout_height="40dip"  
    android:hint="example@example.com"  
    android:layout_toRightOf="@+id/email_text"  
    android:layout_toEndOf="@+id/email_text"  
    android:layout_below="@+id/phone_number_tips"  
    android:minWidth="315dip"  
    android:maxWidth="315dip"  
    android:padding="5dip"  
    android:background="@drawable/motto_edit_text"  
    android:inputType="textEmailAddress"  
    android:autofillHints="">  
</EditText>
```

属性 `android:inputType="textEmailAddress"` 将输入文本的属性设置为邮箱，`@` 的位置将会出现在虚拟键盘上显眼的位置。

第 7 个 `id` 为 `email_tips` 对应的 `TextView` 手机号输入的提示，属性设置如下：

```
<TextView  
    android:id="@+id/email_tips"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/email_edit"  
    android:layout_alignStart="@+id/email_edit"  
    android:layout_alignLeft="@+id/email_edit"  
    android:paddingTop="5dip"  
    android:paddingBottom="5dip"  
    android:text="邮箱">  
</TextView>
```

在 `EditProfileActivity.java` 的 `onCreate()` 方法中，对这些控件的相关设置如下。

```
1     /* 手机号设置 */  
2     // 先根据 tempProfile 的内容对手机号显示的内容初始化  
3     final EditText phoneEdit = (EditText)  
        findViewById(R.id.phone_number_edit);  
4     final TextView phoneText = (TextView)  
        findViewById(R.id.phone_number_text);  
5     final TextView phoneTips = (TextView)  
        findViewById(R.id.phone_number_tips);  
6     phoneEdit.setText(tempProfile.getPhone());
```

```
7     phoneEdit.setSelection(tempProfile.getPhone().length()); // 设置默认光标位置
8     final int phoneTipsColor = phoneTips.getCurrentTextColor(); // 获取原来的文字颜色
9     final int phoneTextColor = phoneText.getCurrentTextColor(); // 获取原来的文字颜色
10    if (tempProfile.getPhone().length() > 0 &&
11        tempProfile.getPhone().length() < phoneCharMinNum) {
12
13        phoneTips.setText(String.format(getString(R.string.phone_tips_content_less),
14                                         phoneCharMaxNum - phoneEdit.getText().length())); // 还能输入%d个字符，不得少于3个字符
15
16        phoneTips.setTextColor(getResources().getColor(R.color.errorColor));
17
18        phoneText.setTextColor(getResources().getColor(R.color.errorColor));
19        hasError++;
20    }
21
22    else if (tempProfile.getPhone().length() == 0 ||
23        tempProfile.getPhone().length() >= phoneCharMinNum) {
24
25        phoneTips.setText(String.format(getString(R.string.phone_tips_content_more),
26                                         phoneCharMaxNum - phoneEdit.getText().length())); // 还能输入%d个字符
27
28        if(tempProfile.getPhone().length() != 0) {
29            setEditProgress(11);
30
31        }
32
33        // 设置控件的监听事件
34        phoneEdit.addTextChangedListener(new TextWatcher() {
35
36            private CharSequence temp;// 监听前的文本
37            private int editStart;// 光标开始位置
38            private int editEnd;// 光标结束位置
39            private int tempNum;// 文本框内容改变前的文字长度
40
41            @Override
42            public void beforeTextChanged(CharSequence s, int start, int
43 count, int after) {
44                temp = s;
45                tempNum = s.length();
46            }
47
48        }
```

```
37     @Override
38     public void onTextChanged(CharSequence s, int start, int before,
39                             int count) {
40         // 当有内容输入时，根据内容长度改变提示信息
41         if(phoneEdit.getText().length() > 0 &&
42             phoneEdit.getText().length() < phoneCharMinNum) {
43             phoneTips.setText(String.format(getString(R.string.phone_tips_content_
44                                         less),
45                                         phoneCharMaxNum -
46                                         phoneEdit.getText().length()));
47         }
48         // 只要手机号和邮箱的其中一个被修改，就修改mustText 的内容为绿色
49         mustText.setTextColor(getResources().getColor(R.color.yesColor));
50     }
51
52     @Override
53     public void afterTextChanged(Editable s) {
54         editStart = phoneEdit.getSelectionStart();
55         editEnd = phoneEdit.getSelectionEnd();
56         // 输入内容的长度从 0 到 i (0 < i < phoneCharMinNum) , 增加错误
57         // 标记，减掉进度值
58         if (phoneEdit.getText().length() > 0 &&
59             phoneEdit.getText().length() < phoneCharMinNum) {
60             phoneTips.setTextColor(getResources().getColor(R.color.errorColor));
61             phoneText.setTextColor(getResources().getColor(R.color.errorColor));
62             if(phoneEdit.getText().length() == phoneCharMinNum - 1 &&
63                 tempNum == phoneCharMinNum) {
64                 setEditProgress(-11);
65                 hasError++;
66             }
67             if (tempNum == 0) {
68                 hasError++;
69             }
70         }
71     }
72 }
```

```
67     }
68     // 输入内容长度大于最大字数限制，则无法继续输入，删除新输入的字
符
69     else if (temp.length() > phoneCharMaxNum) {
70         s.delete(editStart - 1, editEnd);
71         int tempSelection = editStart;
72         phoneEdit.setText(s);
73         phoneEdit.setSelection(tempSelection);
74     }
75     // 输入内容清空
76     else if (temp.length() == 0) {
77         phoneTips.setTextColor(phoneTipsColor);
78         phoneText.setTextColor(phoneTextColor);
79         hasError--;
80     }
81     // 输入内容的长度由i(1 < i < phoneCharMinNum)到phoneCharMinNum
82     // 则输入的内容正确，算上进度值，去掉错误标记
83     else if (temp.length() == phoneCharMinNum) {
84         if(tempNum == phoneCharMinNum - 1) {
85             setEditProgress(11);
86             phoneTips.setTextColor(phoneTipsColor);
87             phoneText.setTextColor(phoneTextColor);
88             hasError--;
89         }
90     }
91     tempProfile.setPhone(phoneEdit.getText().toString());
92 }
93 );
94 phoneText.setOnClickListener(rand);
95 phoneTips.setOnClickListener(rand);
96
97 /* 邮箱设置 */
98 // 先根据tempProfile 的内容对邮箱显示的内容初始化
99 final EditText emailEdit = (EditText) findViewById(R.id.email_edit);
100 final TextView emailText = (TextView)
findViewById(R.id.email_text);
101 final TextView emailTips = (TextView)
findViewById(R.id.email_tips);
102 emailEdit.setText(tempProfile.getEmail());
103 emailEdit.setSelection(tempProfile.getEmail().length()); // 设置默
认光标位置
104 final int emailTipsColor = emailTips.getCurrentTextColor(); // 获
取原来的文字颜色
```

```
105     final int emailTextColor = emailText.getCurrentTextColor(); // 获
        取原来的文字颜色
106     if (tempProfile.getEmail().length() == 0) {
107
108         emailTips.setText(getString(R.string.email_tips_content_empty)); // 邮
        箱格式见提示
109     } else {
110         if (tempProfile.getEmail().matches(emailRegex)) {
111
112             emailTips.setText(getString(R.string.email_tips_content_matched)); // 邮
        箱格式正确
113             setEditProgress(11);
114         } else {
115
116             emailTips.setText(getString(R.string.email_tips_content_unmatched));
        // 邮箱格式不正确
117
118             emailText.setTextColor(getResources().getColor(R.color.errorColor));
119             hasError++;
120         }
121         // 设置控件的监听事件
122         emailEdit.addTextChangedListener(new TextWatcher() {
123             private CharSequence temp;// 监听前的文本
124             private boolean tempMatched;// 文本框内容改变前是否匹配
        emailRegex
125             private int tempNum;// 文本框内容改变前的文字长度
126
127             @Override
128             public void beforeTextChanged(CharSequence s, int start, int
        count, int after) {
129                 temp = s;
130                 tempNum = s.toString().length();
131                 tempMatched = s.toString().matches(emailRegex);
132             }
133
134             @Override
135             public void onTextChanged(CharSequence s, int start, int before,
        int count) {
```

```

136         // 当有内容输入时，判断内容是否匹配，改变提示信息和两个 TextView 内
137         if (emailEdit.getText().toString().length() == 0) {
138             emailTips.setText(getString(R.string.email_tips_content_empty));
139             emailTips.setTextColor(emailTipsColor);
140             emailText.setTextColor(emailTextColor);
141         }
142         else {
143             if (emailEdit.getText().toString().matches(emailRegex))
144             {
145                 emailTips.setText(getString(R.string.email_tips_content_matched));
146                 emailTips.setTextColor(emailTipsColor);
147                 emailText.setTextColor(emailTextColor);
148             }
149             else { // 不匹配则将两个 TextView 内容标红
150                 emailTips.setText(getString(R.string.email_tips_content_unmatched));
151                 emailTips.setTextColor(getResources().getColor(R.color.errorColor));
152                 emailText.setTextColor(getResources().getColor(R.color.errorColor));
153             }
154             // 只要手机号和邮箱的其中一个被修改，就修改mustText 的内容为绿
155             // 色
156             mustText.setTextColor(getResources().getColor(R.color.yesColor));
157         }
158     @Override
159     public void afterTextChanged(Editable s) {
160         if (emailEdit.getText().toString().length() != 0) { // 修
改后内容长度>0
161             // 如果修改前内容为空
162             if (tempNum == 0) {
163                 // 修改后不匹配，则标记错误
164                 if
165                     (!emailEdit.getText().toString().matches(emailRegex)) {
166                         hasError++;
167                     }
168                     else { // 匹配则算上进度值
169                         setEditProgress(11);

```

```

169             }
170         }
171         // 如果修改前匹配，修改后不匹配，则减去进度值，标记错误
172         else if
173             (!emailEdit.getText().toString().matches(emailRegex) && tempMatched) {
174                 setEditProgress(-11);
175                 hasError++;
176             }
177             // 如果修改前匹配，修改后不匹配，则算上进度值，取消错误标记
178             else
179                 if(emailEdit.getText().toString().matches(emailRegex) && !tempMatched)
180                 {
181                     setEditProgress(11);
182                     hasError--;
183                 }
184             }
185             else {
186                 // 如果修改后内容为空，修改前不匹配，则取消错误标记
187                 if (tempNum > 0 && !tempMatched) {
188                     hasError--;
189                 }
190             });
191         emailText.setOnClickListener(rand);
192         emailTips.setOnClickListener(rand);

```

这里设置手机号的长度为 3~15 个字符。

其中 `EditProfileActivity` 类变量 `emailRegex` 的字符串值为：

`"^[\u00a1-\u00d7\u00d8-\u00d9\u00d0-\u00d1_\u00d2]+@[a-zA-Z0-9_\u00d2]+\u00d3+(\.\u00d3[a-zA-Z0-9_\u00d2]+\u00d3)+$"`，为邮箱对应的正则表达式，通过查看 `String` 类的 `matches()` 函数的返回值（参数传入 `emailRegex`）来判断格式是否正确。

实现的效果如下：

初始状态如下图所示：

当前资料完成度：33%（带\*的项目为必填项）

---

昵称 \*  还能输入15个字符

以下两项仅用于身份验证，不会被展示在个人主页哦~

手机号  还能输入15个字符

邮箱  邮箱格式见提示

只输入手机号之后的状态如下图所示：

当前资料完成度：33%（带*的项目为必填项）	当前资料完成度：44%（带*的项目为必填项）
昵称 * <input type="text" value="Betty"/> 还能输入15个字符	昵称 * <input type="text" value="Betty"/> 还能输入15个字符
以下两项仅用于身份验证，不会被展示在个人主页哦~	以下两项仅用于身份验证，不会被展示在个人主页哦~
手机号 <input type="text" value="11"/> 还能输入13个字符，不得少于3个字符	手机号 <input type="text" value="111111"/> 还能输入9个字符

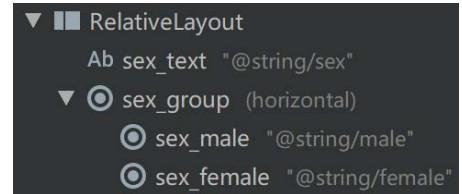
只输入邮箱之后的状态如下图所示：

当前资料完成度：33%（带*的项目为必填项）	当前资料完成度：44%（带*的项目为必填项）
昵称 * <input type="text" value="Betty"/> 还能输入15个字符	昵称 * <input type="text" value="Betty"/> 还能输入15个字符
以下两项仅用于身份验证，不会被展示在个人主页哦~	以下两项仅用于身份验证，不会被展示在个人主页哦~
手机号 <input type="text" value="手机号码，3~15位"/> 还能输入15个字符	手机号 <input type="text" value="手机号码，3~15位"/> 还能输入15个字符
邮箱 <input type="text" value="17"/> 邮箱格式不正确	邮箱 <input type="text" value="1711425@nankai.edu"/> 邮箱格式正确

<p>当前资料完成度：33%（带*的项目为必填项）</p> <hr/> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <span>昵称 *</span> <input type="text" value="Betty"/> 还能输入15个字符         </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;">           以下两项仅用于身份验证，不会被展示在个人主页哦～         </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <span>手机号</span> <input type="text" value="手机号码，3~15位"/> 还能输入15个字符         </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <span>邮箱</span> <input type="text" value="1711425@nankai.edu."/> 邮箱格式不正确         </div>	<p>当前资料完成度：44%（带*的项目为必填项）</p> <hr/> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <span>昵称 *</span> <input type="text" value="Betty"/> 还能输入15个字符         </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;">           以下两项仅用于身份验证，不会被展示在个人主页哦～         </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <span>手机号</span> <input type="text" value="手机号码，3~15位"/> 还能输入15个字符         </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 10px;"> <span>邮箱</span> <input type="text" value="1711425@nankai.edu.com.cn"/> 邮箱格式正确         </div>
---	--

## 第4个RelativeLayout

```
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="5dip">
    <TextView...>
    <RadioGroup>
        android:id="@+id/sex_group"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:layout_marginTop="3dip"
        android:layout_marginBottom="3dip"
        android:layout_toRightOf="@+id/sex_text"
        android:layout_toEndOf="@+id/sex_text">
            <RadioButton...>
            <RadioButton...>
        </RadioGroup>
    </RelativeLayout>
```



第4个 `RelativeLayout` 用于设置性别，有一个 `TextView` 和一个 `RadioGroup`，`RadioGroup` 中有 2 个用于选择性别的 `RadioButton`，对这一项设置的进度值为 11。

`id` 为 `sex_text` 的 `TextView` 的属性设置如下：

```
<TextView  
    android:id="@+id/sex_text"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:paddingTop="9dip"  
    android:maxWidth="75dip"  
    android:minWidth="75dip"  
    android:text="性别 *">  
</TextView>
```

两个 RadioButton 的属性设置如下：

```
<RadioButton  
    android:id="@+id/sex_male"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:text="男">  
</RadioButton> <RadioButton  
    android:id="@+id/sex_female"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginStart="50dip"  
    android:layout_marginLeft="50dip"  
    android:text="女">  
</RadioButton>
```

在 `EditProfileActivity.java` 的 `onCreate()` 函数中，对这些控件的相关设置如下：

```
1  /* 性别设置 */  
2  RadioGroup sex = (RadioGroup) findViewById(R.id.sex_group);  
3  final TextView sexText = (TextView) findViewById(R.id.sex_text);  
4  // 在 RadioButton 后面（右边）加上♂♀符号的Drawable  
5  Drawable femaleDrawable =  
6      getResources().getDrawable(R.drawable.femaleicon);  
7  Drawable maleDrawable =  
8      getResources().getDrawable(R.drawable.maleicon);  
9  femaleDrawable.setBounds(5, 0, 55, 50); // 设置Drawable显示的大小  
10 maleDrawable.setBounds(5, 0, 55, 50);  
11 // 在设置监听器之前先根据tempProfile的内容先进行初始化设置  
12 RadioButton female = (RadioButton) findViewById(R.id.sex_female);  
13 RadioButton male = (RadioButton) findViewById(R.id.sex_male);  
14 female.setCompoundDrawables(null, null, femaleDrawable, null); // 将  
15 Drawable只显示在RadioButton右侧  
16 male.setCompoundDrawables(null, null, maleDrawable, null);  
17 setEditProgress(11);  
18 if (tempProfile.getSex() == Sex.FEMALE) {  
19     female.setChecked(true);  
20     sexText.setTextColor(getResources().getColor(R.color.femaleColor));  
21 }  
22 else {  
23     male.setChecked(true);  
24 }
```

```

21     sexText.setTextColor(getResources().getColor(R.color.maleColor));
22 }
23 // 设置监听事件
24 sex.setOnCheckedChangeListener(new
RadioGroup.OnCheckedChangeListener() {
25     @Override
26     public void onCheckedChanged(RadioGroup group, int checkedId) {
27         if (checkedId == R.id.sex_female) {
28             // 哪个性别选中了，就改变sex_text 的颜色并改变tempProfile 中
29             // 对应的性别
30             tempProfile.setSex(Sex.FEMALE);
31         }
32         else {
33             tempProfile.setSex(Sex.MALE);
34
35             sexText.setTextColor(getResources().getColor(R.color.femaleColor));
36         }
37     });
38     sexText.setOnClickListener(rand);

```

实现的效果如下，左边的 `TextView` 颜色随着性别选择的变化而变化：



## 第5个 `RelativeLayout`

```

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="5dip">
    <TextView...>
    <!-- 去掉Button默认灰色边框: style="?android:buttonStyle" -->
    <Button...>
</RelativeLayout>

```

第5个 `RelativeLayout` 用于设置性别，有一个 `TextView` 和一个 `Button`，点

点击 Button 会弹出 DatePickerDialog 来设置生日，对这一项设置的进度值为 11%。

id 为 birthday\_text 的 TextView 的属性设置如下：

```
<TextView  
    android:id="@+id/birthday_text"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:paddingTop="15dip"  
    android:maxWidth="75dip"  
    android:minWidth="75dip"  
    android:text="生日">  
</TextView>
```

id 为 birthday\_set 的 Button 的属性设置如下，点击该 Button 将会弹出一个 DatePickerDialog 用于设置生日：

```
<Button  
    android:id="@+id/birthday_set"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_toRightOf="@+id/birthday_text"  
    android:layout_toEndOf="@+id/birthday_text"  
    android:paddingStart="0dip"  
    android:paddingLeft="0dip"  
    android:paddingEnd="30dip"  
    android:paddingRight="30dip"  
    android:background="#00000000"  
    style="?android:attr/borderlessButtonStyle"  
    android:text="button">  
</Button>
```

其中属性 android:background="#00000000" 是为了设置 Button 背景透明，但是背景透明了之后还能看得见 Button 的默认边框，因此属性设置 style="?android:attr/borderlessButtonStyle" 就是用来去掉 Button 默认边框的。

在 EditProfileActivity.java 的 onCreate() 方法中，对这些控件的相关设置如下：

```
1  /* 生日设置 */  
2  // 根据 tempProfile 的内容对 Button 上显示的文字进行初始化（显示  
3  tempProfile 设置的生日）  
4  final Button birthday = (Button) findViewById(R.id.birthday_set);  
5  final TextView birthdayText = (TextView)  
    findViewById(R.id.birthday_text);  
6  final int birthdayColor = birthdayText.getCurrentTextColor();
```

```

6     Calendar c = Calendar.getInstance();
7     Date date = new Date();
8     if (tempProfile.hasBirthday()) {
9         birthday.setText(tempProfile.getBirthdayString());
10        c.set(tempProfile.getBirthYear(),
11             tempProfile.getBirthMonth() - 1,
12             tempProfile.getBirthDay());
13        setEditProgress(11);
14    }
15    else {
16        birthday.setText(dateFormat.format(date));
17    }
18    // DatePickerDialog 相关设置
19    final DatePickerDialog pickerDialog = new DatePickerDialog(this,
20                  DatePickerDialog.THEME_DEVICE_DEFAULT_DARK, null, // 不设置
21                  DatePickerDialog.OnDateSetListener
22                      c.get(Calendar.YEAR), c.get(Calendar.MONTH),
23                      c.get(Calendar.DAY_OF_MONTH));
24    final DatePicker picker = pickerDialog.getDatePicker(); // 获取
25    // DatePickerDialog 对应的DatePicker
26    picker.setMaxDate(date.getTime()); // 设置最大日期, 避免用户在设置生日的
27    // 时候超过当前日期
28    /* 注意 Date.getTime() 和 Calendar.getTimeInMillis() 返回的都是从
29    // 1970-01-01 开始的毫秒数
30    // 通过 setMinDate() 能设置的最小日期也只能是 1970-01-01 */
31    // 设置 DatePickerDialog 两个按钮的响应
32    pickerDialog.setButton(DialogInterface.BUTTON_NEGATIVE,
33        getString(R.string.cancel),
34        new DialogInterface.OnClickListener() {
35            @Override
36            public void onClick(DialogInterface dialog, int which) {
37                // 如果点击了“取消”, 则将 TextView 的文字内容还原成原来的颜
38                // 色
39                birthdayText.setTextColor(birthdayColor);
40            }
41        });
42    pickerDialog.setButton(DialogInterface.BUTTON_POSITIVE,
43        getString(R.string.ok),
44        new DialogInterface.OnClickListener() {
45            @Override
46            public void onClick(DialogInterface dialog, int which) {
47                // 通过 picker 获取当前选中的日期
48                int year = picker.getYear();
49                int monthOfYear = picker.getMonth();

```

```

42         int dayOfMonth = picker.getDayOfMonth();
43         Calendar cal = Calendar.getInstance();
44         cal.set(year, monthOfYear, dayOfMonth);
45         if (!tempProfile.hasBirthday()) {
46             setEditProgress(11);
47         }
48         tempProfile.setBirthday(year, monthOfYear + 1,
49             dayOfMonth);
50         Date dat = cal.getTime();
51         birthday.setText(dateFormat.format(dat)); // 设置
Button 上面应该显示什么日期
52         // 如果点击了“确定”，则将 TextView 的文字内容设为绿色
53         birthdayText.setTextColor(getResources().getColor(R.color.yesColor));
54     }
55     // 设置 Button 的监听事件
56     birthday.setOnClickListener(new View.OnClickListener() {
57         @Override
58         public void onClick(View v) {
59             pickerDialog.show(); // 点击直接弹出 pickerDialog
60         }
61     });
62     birthdayText.setOnClickListener(rand);

```

我没有在 *DatePickerDialog* 的构造函数中直接设置

*DatePickerDialog.OnDateSetListener*, 因为设置了之后无论点击确定按钮还是取消按钮都会改变设置的日期, 而我想实现点击取消按钮就不改变上次的设置。因此直接通过 *setButton()* 函数单独定义对两个按钮的事件响应。在定义确定按钮的事件时, 使用了 *DatePicker* 的 *getYear()*、*getMonth()*、*getDayOfMonth()* 获取当前设置的日期。

实现的效果如下:

这是设置前的日期:



点击这个按钮, 弹出 *DatePickerDialog* 的初始状态如下:



要设置的日期如下：



点击确定可以看到按钮上面的日期已经改成了刚刚设置好的日期，并且 TextView 内容的颜色变成了绿色，表明刚刚点击了确定：



再次单击按钮，重新设置日期如下：



点击取消按钮，则按钮上的日期没有改变，`TextView` 的颜色恢复了原来的颜色，表明刚刚点击了取消按钮：



### 第6个和第7个`RelativeLayout`

第6个`RelativeLayout`是用来设置所在地（省份）的，第7个`RelativeLayout`是用来设置职业身份的，进度条的进度值都是11。这两个`RelativeLayout`都只有一个`TextView`和`Spinner`，设置的方式几乎是一样的（包括位置属性），所以我就只贴图了。

```
<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="5dip">
    <TextView
        android:id="@+id/province_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="5dip"
        android:maxWidth="75dip"
        android:minWidth="75dip"
        android:text="所在地">
    </TextView>
    <Spinner
        android:id="@+id/select_province"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/province_text"
        android:layout_toEndOf="@+id/province_text"
        android:layout_marginTop="3dip"
        android:layout_marginBottom="5dip"
        android:spinnerMode="dialog"
        android:prompt="所在地"
        android:entries="@array/provinces">
    </Spinner>
</RelativeLayout>

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="5dip">
    <TextView
        android:id="@+id/job_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="5dip"
        android:maxWidth="75dip"
        android:minWidth="75dip"
        android:text="职业身份">
    </TextView>
    <Spinner
        android:id="@+id/select_job"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_toRightOf="@+id/job_text"
        android:layout_toEndOf="@+id/job_text"
        android:layout_marginTop="3dip"
        android:layout_marginBottom="5dip"
        android:spinnerMode="dialog"
        android:prompt="职业身份"
        android:entries="@array/jobs">
    </Spinner>
</RelativeLayout>
```

其中，`Spinner`的`android:spinnerMode="dialog"`属性指定了`Spinner`的选

择项以对话框的形式弹出，`android:prompt` 设置弹出对话框的标题，`android:entries` 指定了 `Spinner` 绑定的数据源（`Spinner` 可以选择什么样的数据），数据源在 `res/values/arrays.xml` 文件中，如下图所示：



在 `EditProfileActivity.java` 的 `onCreate()` 方法中，相关设置如下。对 `Spinner` 使用了 `AdapterView.OnItemSelectedListener`。

```
1  /* 所在地设置 */
2  Spinner location = (Spinner) findViewById(R.id.select_province);
3  final TextView locationText = (TextView) findViewById(R.id.province_text);
4  final int locationColor = locationText.getCurrentTextColor();
5  // 根据tempProfile 的内容设置Spinner 初始选中的项
6  location.setSelection(tempProfile.getLocation() + 1);
7  if (tempProfile.getLocation() != -1) {
8      setEditProgress(11);
9
10     locationText.setTextColor(getResources().getColor(R.color.yesColor));
11 }
12 // 设置监听事件
13 location.setOnItemSelectedListener(new
14     AdapterView.OnItemSelectedListener() {
15         @Override
16         public void onItemSelected(AdapterView<?> parent, View view, int
17             position, long id) {
18             if (position != 0) { // position 参数为选中的项的索引，在 tempProfile
19                 中的索引为position-1
20                 if (tempProfile.getLocation() == -1) { // 如果设置了，就改变
21                     location_text 的颜色为绿色，增加进度值
22                     setEditProgress(11);
23                 }
24             }
25         }
26     }
27 }
```

```
18     locationText.setTextColor(getResources().getColor(R.color.yesColor));
// 设置 location_text 的颜色为绿色
19 }
20 }
21 else { // 选中了“请选择”则相当于没设置，不增加进度条的进度值
22     if (tempProfile.getLocation() != -1) {
23         setEditProgress(-11);
24         locationText.setTextColor(locationColor); // 恢复原来的颜色
25     }
26 }
27 tempProfile.setLocation(position - 1); // 同步 tempProfile 的设置
28 }
29
30 @Override
31 public void onNothingSelected(AdapterView<?> parent) {
32
33 }
34 });
35 locationText.setOnClickListener(rand);
36
37 /* 职业设置 */
38 Spinner job = (Spinner) findViewById(R.id.select_job);
39 final TextView jobText = (TextView) findViewById(R.id.job_text);
40 final int jobColor = jobText.getCurrentTextColor();
41 // 根据 tempProfile 的内容设置 Spinner 初始选中的项
42 job.setSelection(tempProfile.getJob() + 1);
43 if (tempProfile.getJob() != -1) {
44     setEditProgress(11);
45     jobText.setTextColor(getResources().getColor(R.color.yesColor));
46 }
47 // 设置监听事件
48 job.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
49     @Override
50     public void onItemSelected(AdapterView<?> parent, View view, int
position, long id) {
51         if (position != 0) {
52             if (tempProfile.getJob() == -1) { // position 参数为选中的项的
索引，在 tempProfile 中的索引为 position-1
53                 setEditProgress(11);
54
jobText.setTextColor(getResources().getColor(R.color.yesColor)); // 设置 location_text 的颜色为绿色
55         }
```

```

56     }
57     else { // 选中了“请选择”则相当于没设置，不增加进度条的进度值
58         if (tempProfile.getJob() != -1) {
59             setEditProgress(-11);
60             jobText.setTextColor(jobColor); // 恢复原来的颜色
61         }
62     }
63     tempProfile.setJob(position - 1); // 同步tempProfile 的设置
64 }
65
66 @Override
67 public void onNothingSelected(AdapterView<?> parent) {
68
69 }
70 });
71 jobText.setOnClickListener(rand);

```

实现的效果展示如下：

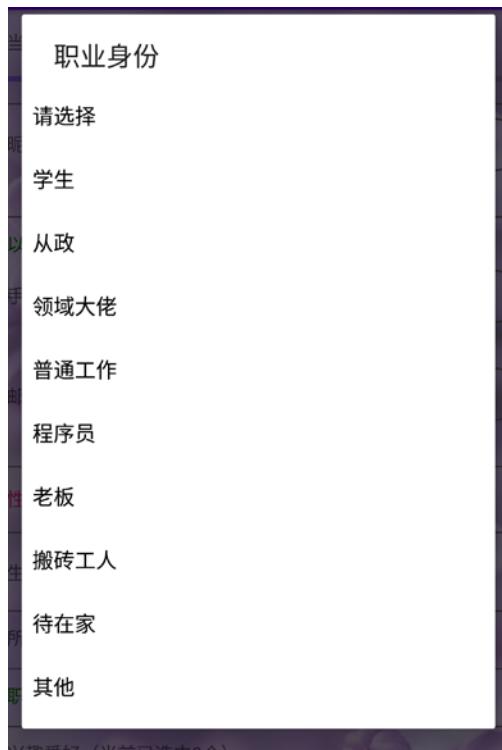
设置前 vs 设置后，两个 **Spinner** 都设置以后进度值增加 22%：



取消选择其中一个 **Spinner**，可以发现进度值减了 11%，并且对应的 **TextView** 恢复成了原来的颜色：



点击其中一个 Spinner 弹出对话框的效果：



## 第8个RelativeLayout

第8个 RelativeLayout 用于设置兴趣爱好，有一个 TextView 和 10 个多选的 CheckBox，TextView 除了指示这些 CheckBox 设置兴趣爱好之外还指示选中了多少个 CheckBox。进度值为 11%。

```

<RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:padding="5dip">
    <TextView...>
    <CheckBox...>
    </RelativeLayout>

```

▼ ■■■ RelativeLayout  
✓ hobby\_text "@string/hobby"  
✓ travel\_button "@string/travel"  
✓ art\_button "@string/art"  
✓ sports\_button "@string/sports"  
✓ food\_button "@string/food"  
✓ movie\_music\_button "@string/movie\_music"  
✓ play\_button "@string/play"  
✓ chat\_button "@string/chat"  
✓ study\_button "@string/study"  
✓ stay\_home\_button "@string/stay\_home"  
✓ sleep\_button "@string/sleep"

TextView 和其中一个 CheckBox 的属性设置如下。由于这些 CheckBox 的设置除了位置属性之外大致相同，所以只展示其中一个 CheckBox。

```

<TextView
    android:id="@+id/hobby_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:paddingTop="5dip"
    android:paddingBottom="10dip"
    android:text="兴趣爱好">
</TextView>

```

```

<CheckBox
    android:id="@+id/travel_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_below="@+id/hobby_text"
    android:layout_marginLeft="65dip"
    android:layout_marginStart="65dip"
    android:text="旅行和摄影">
</CheckBox>

```

在 `EditProfileActivity.java` 的 `EditProfileActivity` 类的 `onCreate()` 方法中，相关设置如下。对 CheckBox 使用了 `CompoundButton.OnCheckedChangeListener`。

```

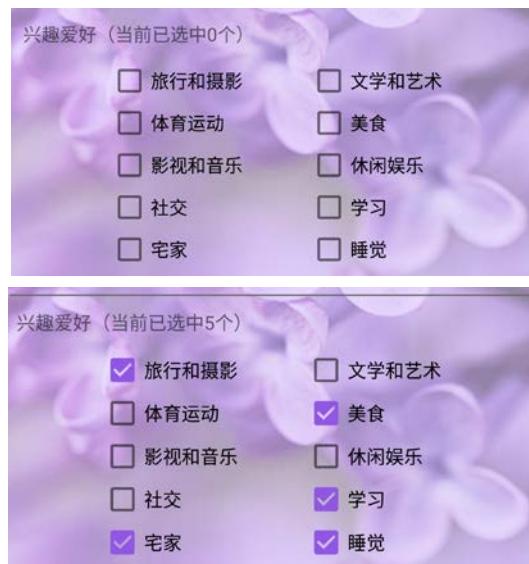
1  /* 兴趣爱好设置 */
2  final TextView hobbyDis = (TextView) findViewById(R.id.hobby_text);
3  final CheckBox[] hobbies = new CheckBox[Profile.hobbies.length];
4  hobbies[0] = (CheckBox) findViewById(R.id.travel_button);
5  hobbies[1] = (CheckBox) findViewById(R.id.art_button);
6  hobbies[2] = (CheckBox) findViewById(R.id.sports_button);
7  hobbies[3] = (CheckBox) findViewById(R.id.food_button);
8  hobbies[4] = (CheckBox) findViewById(R.id.movie_music_button);
9  hobbies[5] = (CheckBox) findViewById(R.id.play_button);
10 hobbies[6] = (CheckBox) findViewById(R.id.chat_button);
11 hobbies[7] = (CheckBox) findViewById(R.id.study_button);
12 hobbies[8] = (CheckBox) findViewById(R.id.stay_home_button);
13 hobbies[9] = (CheckBox) findViewById(R.id.sleep_button);
14 // 根据 tempProfile 的兴趣爱好的内容对 CheckBox 的选中状态进行初始化

```

```
15 boolean[] profileHobby = tempProfile.getHobby(); // 获取 hobby 所对应的
   boolean 数组, true 为选中
16 for (int i = 0; i < profileHobby.length; i++) {
17     if (profileHobby[i])
18         hobbies[i].setChecked(true);
19     else
20         hobbies[i].setChecked(false);
21 }
22 final int hobbyCount = tempProfile.getHobbyCount();
23 if (hobbyCount != 0)
24     setEditProgress(11);
25 // R.string.hobby_count_display: 兴趣爱好 (当前已选中%d 个)
26 hobbyDis.setText(String.format(getString(R.string.hobby_count_display),
   hobbyCount));
27 CompoundButton.OnCheckedChangeListener hobbyListener = new
   CompoundButton.OnCheckedChangeListener() {
28     private int prevHobbyCount = hobbyCount; // 记录上次选中的数目
29     @Override
30     public void onCheckedChanged(CompoundButton buttonView, boolean
   isChecked) {
31         for (int i = 0; i < hobbies.length; i++) {
32             if (hobbies[i].isChecked())
33                 tempProfile.addHobby(i);
34             else
35                 tempProfile.deleteHobby(i);
36         }
37         int currHobbyCount = tempProfile.getHobbyCount();
38         if (currHobbyCount == 0) { // 如果一个也没选中, 则不算上进度
39             if (prevHobbyCount > 0)
40                 setEditProgress(-11);
41         }
42         else {
43             if (prevHobbyCount == 0)
44                 setEditProgress(11);
45         }
46
47         hobbyDis.setText(String.format(getString(R.string.hobby_count_displa
   y), currHobbyCount));
48     }
49 };
50 // 为每个 CheckBox 设置同一个监听事件
51 for (CheckBox hobby : hobbies) {
52     hobby.setOnCheckedChangeListener(hobbyListener);
```

```
53 }  
54 hobbyDis.setOnClickListener(rand);
```

实现的效果展示如下：



## 第9个RelativeLayout

最后一个 RelativeLayout 用于设置个性签名，有一个 TextView 和一个 EditText。进度值占 12%。

```
<RelativeLayout  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:padding="5dip">  
    <TextView ...>  
    <EditText ...>  
</RelativeLayout>
```

▼ ■■■ RelativeLayout  
 Ab motto\_text "@string/motto"  
 Ab motto\_edit (Multiline Text)

id 为 motto\_text 的 TextView 的属性设置如下：

```
<TextView  
    android:id="@+id/motto_text"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:paddingTop="5dip"  
    android:text="个性签名">  
</TextView>
```

id 为 motto\_edit 的 EditText 的属性设置如下：

```
<EditText  
    android:id="@+id/motto_edit"  
    android:layout_width="fill_parent"  
    android:layout_height="90dip"  
    android:layout_below="@id/motto_text"  
    android:layout_marginTop="10dip"  
    android:minLines="5"  
    android:maxLines="5"  
    android:gravity="top"  
    android:padding="5dip"  
    android:background="@drawable/motto_edit_text"  
    android:textSize="14dip"  
    android:autofillHints=""  
    android:scrollbars="vertical"  
    android:inputType="textMultiLine" />
```

其中属性设置 `android:inputType="textMultiLine"` 为设置这个 `EditText` 是多行的。

`android:minLines="5"` 和 `android:maxLines="5"` 分别说明这个 `EditText` 一次最少或最多显示多少行，这里设置的都是 5，也即一次显示 5 行。

属性 `android:scrollbars="vertical"` 表示在 `EditText` 里面加一个垂直滚动条，这样这 5 行显示不下的内容就可以滚动查看了。

在 `EditProfileActivity.java` 的 `EditProfileActivity` 类的 `onCreate()` 方法中，相关设置如下，这次对 `EditText` 直接使用了 `OnKeyListener()`：

```
1  /* 个性签名设置 */  
2  // 根据 tempProfile 的内容对个性签名的显示进行初始化  
3  final TextView mottoTips = (TextView) findViewById(R.id.motto_text);  
4  final EditText mottoEdit = (EditText) findViewById(R.id.motto_edit);  
5  mottoEdit.setText(tempProfile.getMotto());  
6  mottoEdit.setSelection(tempProfile.getMotto().length()); // 设置默  
认光标位置  
7  final int mottoTipsColor = mottoTips.getCurrentTextColor(); // 获取  
原来的文字颜色  
8  if (tempProfile.getMotto().length() <= mottoCharMaxNum) {  
9      if (tempProfile.getMotto().length() > 0) {  
10         setEditProgress(12);  
11     }  
12  
13     mottoTips.setText(String.format(getString(R.string.motto_display_less),  
14                             tempProfile.getMotto().length()));  
15  }  
16  else {  
17  
18     mottoTips.setText(String.format(getString(R.string.motto_display_more),  
19                             tempProfile.getMotto().length()));  
20 }
```

```

17         tempProfile.getMotto().length())));
18
19     mottoTips.setTextColor(getResources().getColor(R.color.errorColor));
20     hasError++;
21 }
22 mottoEdit.setOnKeyListener(new View.OnKeyListener() {
23     int prevCount = tempProfile.getMotto().length(); // 记录上次编辑
24     的文本长度
25     @Override
26     public boolean onKey(View v, int keyCode, KeyEvent event) {
27         String str = mottoEdit.getText().toString();
28         int currCount = str.length();
29         if (str.length() == 0) {
30             if (prevCount > 0) { // 输入前长度为0, 输入后长度>0, 则减
31                 去进度值
32                     setEditProgress(-12);
33             }
34         }
35         // 输入前字数不在限定范围内, 输入后在, 则要算上进度值并去掉错误标记
36         else if (str.length() <= mottoCharMaxNum) {
37             if (prevCount == 0 || prevCount > mottoCharMaxNum) {
38                 setEditProgress(12);
39                 if (prevCount > mottoCharMaxNum) {
40                     hasError--;
41                 }
42             }
43         }
44         mottoTips.setText(String.format(getString(R.string.motto_display_less),
45                                         currCount));
46         mottoTips.setTextColor(mottoTipsColor);
47     }
48     // 输入前字数在限定范围, 输入后不在, 则减去进度值, 增加错误标记
49     else {
50         if (prevCount <= mottoCharMaxNum) {
51             setEditProgress(-12);
52             hasError++;
53         }
54         mottoTips.setText(String.format(getString(R.string.motto_display_more),
55                                         currCount));

```

```
55     mottoTips.setTextColor(getResources().getColor(R.color.errorColor));
56     }
57     tempProfile.setMotto(str);
58     prevCount = currCount;// 记录上次编辑的文本长度
59     return false;
60   }
61 });
62 mottoTips.setOnClickListener(rand);
```

实现的效果如下：



### 3.1.3 菜单和返回键的设计

我首先对菜单内容定义在 res/menu/edit\_profile\_menu.xml 中

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/save_button"
        android:title="保存"
        app:showAsAction="always"/>
</menu>
```

其中属性 `app:showAsAction="always"` 表示菜单项的标题文字（或图标）直接显示在菜单栏上，这样就实现了右上角保存按钮的效果。

在 `EditProfileActivity.java` 的 `onCreateOptionsMenu()` 方法设置菜单要绑定的菜单项源文件为 `res/menu/edit_profile_menu.xml`:

```
@Override  
public boolean onCreateOptionsMenu(Menu menu) {  
    getMenuInflater().inflate(R.menu.edit_profile_menu, menu);  
    return super.onCreateOptionsMenu(menu);  
}
```

左上角的返回按钮也算是菜单项，但是不需要绑定菜单项，需要在 `EditProfileActivity.java` 的 `onCreate()` 方法中启用左上角返回按钮（默认是不启用的）：

```
/* 左上角返回按钮设置 */
getSupportActionBar().setDisplayHomeAsUpEnabled(true); // 设置显示返回按钮
getSupportActionBar().setHomeButtonEnabled(true); // 设置返回按钮允许按下
```

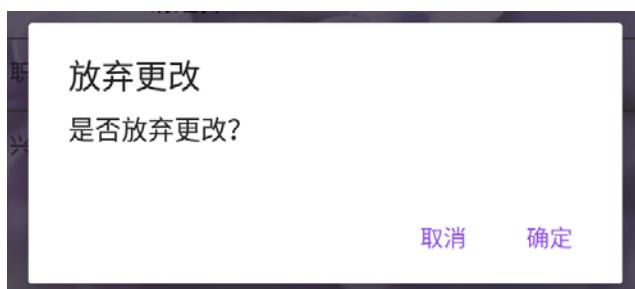
在 `EditProfileActivity.java` 的 `onOptionsItemSelected()` 方法中设置左上角返回按钮和右上角保存按钮的响应事件：

```
@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case android.R.id.home: // 注意是 android.R.id.home 不是 R.id.home
            showSaveDialog();
            break;
        case R.id.save_button:
            save();
            break;
        default:
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

其中 `case android.R.id.home` 设置的是左上角返回按钮的事件，注意一定是 `android.R` 不是 `R`，前者会取用 Android 定义好的资源，而我在这里是没有定义 `home` 这个 `id` 的。设置的响应事件在 `showSaveDialog()` 方法中：

```
private void showSaveDialog() {
    AlertDialog.Builder builder = new AlertDialog.Builder(context: EditProfileActivity.this);
    builder.setTitle("放弃更改");
    builder.setMessage("是否放弃更改？");
    builder.setPositiveButton("确定", new DialogInterface.OnClickListener() {
        @Override
        public void onClick(DialogInterface dialog, int which) {
            SwitchActivity.switchActivity(srcActivity: EditProfileActivity.this,
                ProfileActivity.class, finish: true);
        }
    });
    builder.setNegativeButton("取消", listener: null);
    builder.create().show();
}
```

这里设置了一个 `AlertDialog`，弹出的效果如下：



`setPositiveButton()`函数设置了监听器`DialogInterface.OnClickListener`, 行为是切换`Activity`到显示个人信息的`Activity`中。`setNegativeButton()`函数没有设置监听器, 因此点击“取消”对话框只会消失。

`case R.id.save_button` 设置的是右上角保存按钮的事件, 设置的响应事件在`save()`函数中:

```
private void save() {
    if (hasError == 0) {
        ShareProfile.profile.copyProfile(tempProfile);
        SwitchActivity.switchActivity( srcActivity: EditProfileActivity.this,
            ProfileActivity.class, finish: true);
        Toast.makeText( context: EditProfileActivity.this, "已保存", Toast.LENGTH_SHORT).show();
    }
    else {
        Toast.makeText( context: EditProfileActivity.this, "保存失败, 请检查并修改红色提示项",
            Toast.LENGTH_SHORT).show();
    }
}
```

如果编辑的个人信息存在错误(`hasError>0`), 则无法保存。如果保存就将这个`Activity`的临时资料`tempProfile`拷贝到全局使用的资料`ShareProfile.profile`中, 并切换`Activity`到显示个人信息的`Activity`中。

在`onKeyDown()`方法中, 设置点击左下角返回键的行为:

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_BACK) {
        showSaveDialog();
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```

和左上角返回按钮的行为一致。

在这里使用到的函数`switchActivity()`作用是使用`Intent`来切换`Activity`, 是我自定义的工具函数, 在包`com.example.uidesign`的`SwitchActivity`类中:

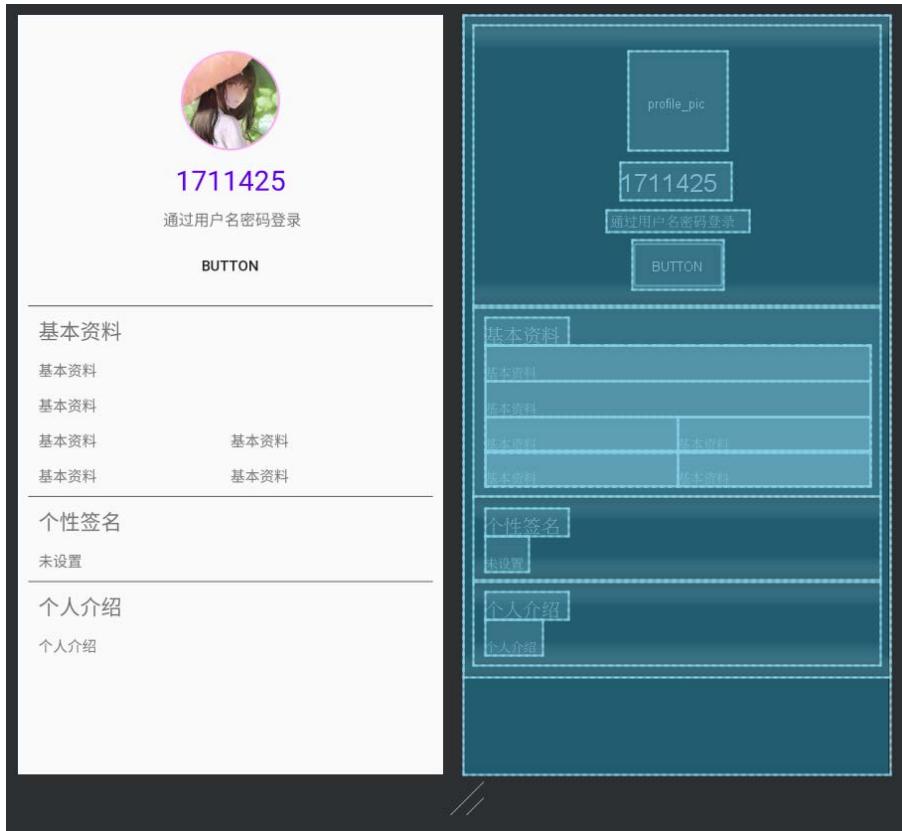
```
public class SwitchActivity {  
    public static void switchActivity(AppCompatActivity srcActivity, Class datActivity, boolean finish) {  
        Intent intent = new Intent(srcActivity, datActivity);  
        srcActivity.startActivity(intent);  
        if (finish) {  
            srcActivity.finish();  
        }  
    }  
}
```

## 3.2 个人信息展示界面的实现

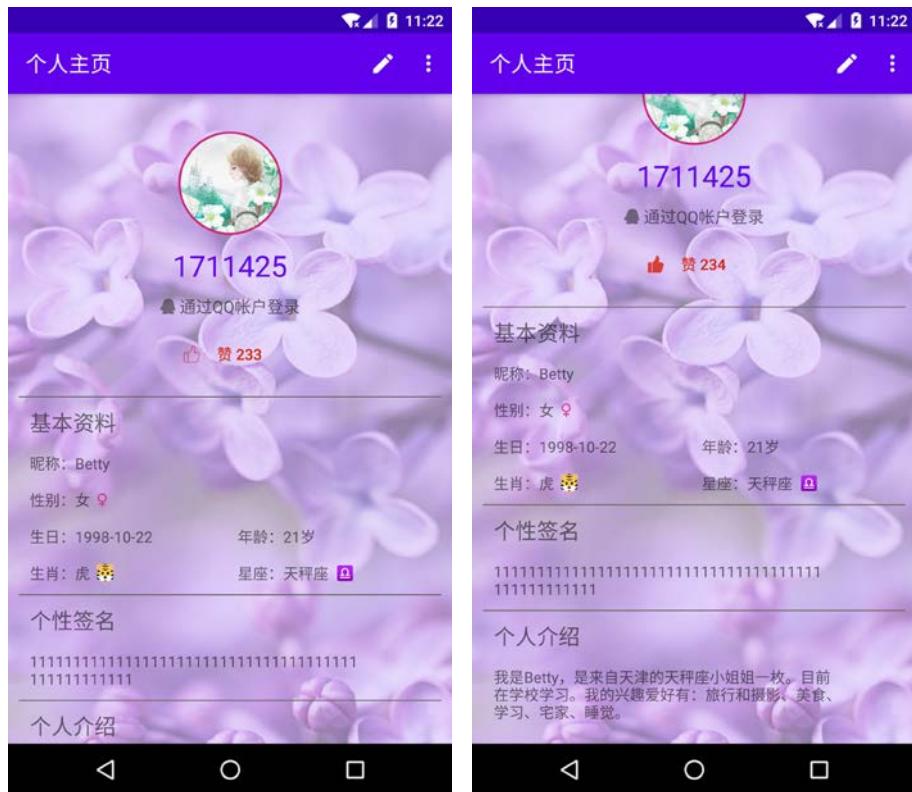
这个界面仅仅用来展示 `EditProfileActivity` 中设置的个人信息(除了手机号和邮箱)，由于不在要求范围内，除了菜单和左下角返回键之外，其他控件没有设置任何监听事件。

### 3.2.1 布局设计

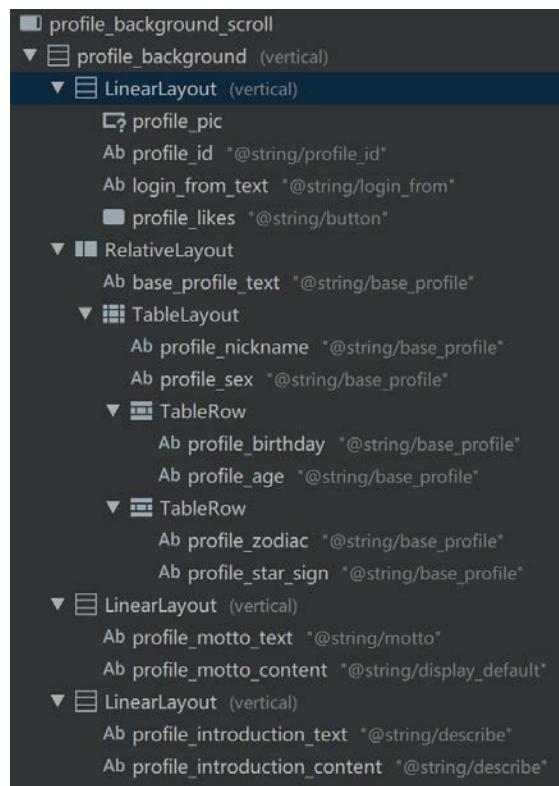
从 `activity_profile.xml` 文件的设计视图查看界面的大致布局情况：



运行程序之后，编辑个人信息并保存之后这个界面的实际效果如下：



切换视图，`activity_profile.xml` 整体布局结构如下：



```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/profile_background_scroll"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:fadingEdge="vertical"
    android:gravity="center"
    tools:context=".ProfileActivity">

    <LinearLayout
        android:id="@+id/profile_background"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:gravity="center"
        android:divider="@drawable/divider"
        android:showDividers="middle"
        android:padding="10dip">
```

可以看到最外层也是一个 `ScrollView`, 里面是一个垂直的 `LinearLayout`, 设置的参数和 `activity_edit_profile.xml` 几乎是一样的。`LinearLayout` 里面又嵌套了 3 个 `LinearLayout` 和 1 个 `RelativeLayout`, `RelativeLayout` 里面又嵌套了一个 `TableLayout`。

最里层的第一个 `LinearLayout` 用于显示头像、用户名、登录方式和点赞按钮。

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="15dip"
    android:gravity="center">

    <de.hdodenhof.circleimageview.CircleImageView...>

    <TextView...>
    <TextView...>
    <Button...>
</LinearLayout>
```

第一个控件的属性设置和在 `ProfileActivity.java` 的 `onCreate()` 方法的设置代码:

```
<de.hdodenhof.circleimageview.CircleImageView
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/profile_pic"
    android:layout_width="96dip"
    android:layout_height="96dip"
    android:src="@drawable/me"
    app:civ_border_width="2dip"
    android:layout_marginTop="10dip"
    android:layout_gravity="center"
    app:civ_border_color="@color/aboutColor"/>
```

```
CircleImageView pic = (CircleImageView) findViewById(R.id.profile_pic);
String sex_temp = "";
switch (ShareProfile.profile.getSex()) {
    case MALE:
        sex_temp = "男";
        pic.setImageResource(R.drawable.male);
        pic.setBorderColor(getResources().getColor(R.color.maleColor));
        sexDrawable = getResources().getDrawable(R.drawable.maleicon);
        sexDrawable.setBounds( left: -1040, top: 0, right: -990, bottom: 50 );
        sex.setCompoundDrawables( left: null, top: null, sexDrawable, bottom: null );
        break;
    case FEMALE:
        sex_temp = "女";
        pic.setImageResource(R.drawable.female);
        pic.setBorderColor(getResources().getColor(R.color.femaleColor));
        sexDrawable = getResources().getDrawable(R.drawable.femaleicon);
        sexDrawable.setBounds( left: -1040, top: 0, right: -990, bottom: 50 );
        sex.setCompoundDrawables( left: null, top: null, sexDrawable, bottom: null );
        break;
    default:
        break;
}
sex.setText(String.format("性别: %s", sex_temp));
```

这样就可以根据设置的性别在 `CircleImageView` 里面显示不同的头像了：



其余三个控件的属性设置如下，分别是显示用户名、登录方式和点赞的控件：

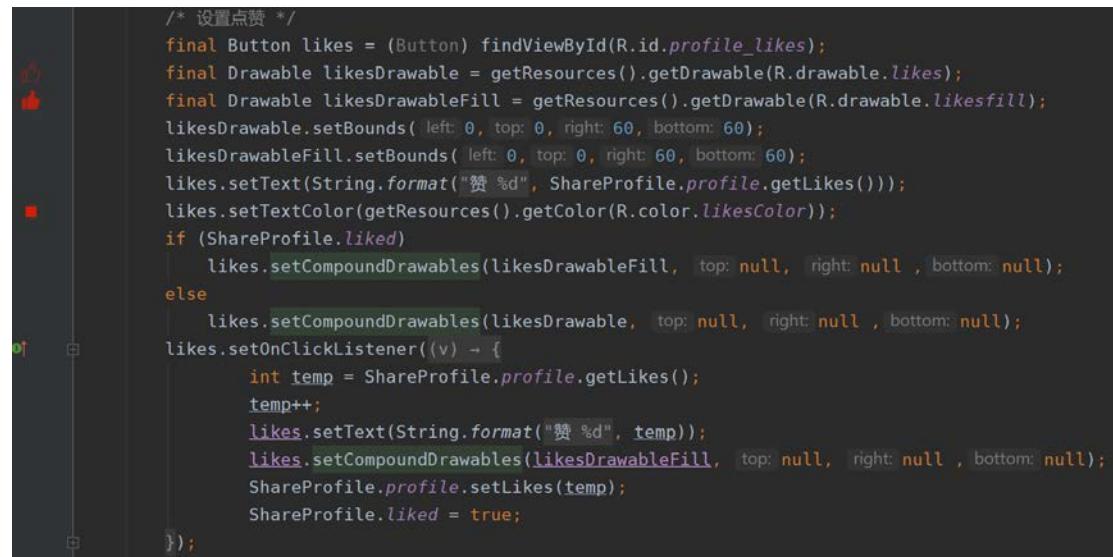
```
<TextView
    android:id="@+id/profile_id"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dip"
    android:text="1711425"
    android:textColor="@color/colorPrimary"
    android:textSize="27dip">
</TextView>
<TextView
    android:id="@+id/login_from_text"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dip"
    android:text="通过用户名密码登录"
    android:textSize="15dip">
</TextView>
<Button|
    android:id="@+id/profile_likes"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginTop="10dip"
    android:background="#00000000"
    style="?android:attr/borderlessButtonStyle"
    android:text="button">
</Button>
```

显示登录方式的代码如下：

```
TextView loginFrom = (TextView) findViewById(R.id.login_from_text);
Drawable loginDrawable;
switch (ShareProfile.profile.getFrom()) {
    case NAME:
        loginFrom.setText("通过用户名密码登录");
        loginDrawable = getResources().getDrawable(R.drawable.account);
        loginDrawable.setBounds( left: 0, top: 0, right: 60, bottom: 60 );
        loginFrom.setCompoundDrawables(loginDrawable, top: null, right: null, bottom: null);
        break;
    case WEIXIN:
        loginFrom.setText("通过微信帐户登录");
        loginDrawable = getResources().getDrawable(R.drawable.loginweixin);
        loginDrawable.setBounds( left: 0, top: 0, right: 50, bottom: 50 );
        loginFrom.setCompoundDrawables(loginDrawable, top: null, right: null, bottom: null);
        break;
    case WEIBO:
        loginFrom.setText(getText(("通过微博帐户登录")));
        loginDrawable = getResources().getDrawable(R.drawable.loginweibo);
        loginDrawable.setBounds( left: 0, top: 0, right: 50, bottom: 50 );
        loginFrom.setCompoundDrawables(loginDrawable, top: null, right: null, bottom: null);
        break;
    case QQ:
        loginFrom.setText("通过QQ帐户登录");
        loginDrawable = getResources().getDrawable(R.drawable.loginqq);
        loginDrawable.setBounds( left: 0, top: 0, right: 50, bottom: 50 );
        loginFrom.setCompoundDrawables(loginDrawable, top: null, right: null, bottom: null);
        break;
}
```

通过 `getResources().getDrawable()` 获取要显示的 `Drawable` 文件，通过 `setBounds()` 设置 `Drawable` 在界面中显示的大小，`setCompoundDrawables()` 用于将 `Drawable` 显示在 `TextView` 的上下左右侧，这里只显示在左侧。

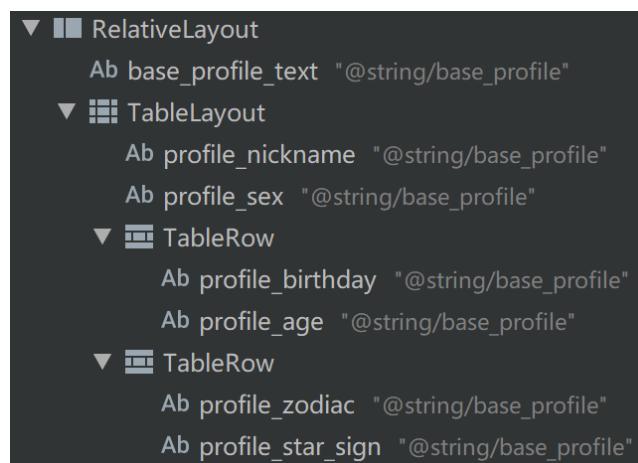
关于点赞的设置代码如下：



```
/* 设置点赞 */
final Button likes = (Button) findViewById(R.id.profile_likes);
final Drawable likesDrawable = getResources().getDrawable(R.drawable.likes);
final Drawable likesDrawableFill = getResources().getDrawable(R.drawable.likesfill);
likesDrawable.setBounds( left: 0, top: 0, right: 60, bottom: 60 );
likesDrawableFill.setBounds( left: 0, top: 0, right: 60, bottom: 60 );
likes.setText(String.format("赞 %d", ShareProfile.profile.getLikes()));
likes.setTextColor(getResources().getColor(R.color.likesColor));
if (ShareProfile.liked)
    likes.setCompoundDrawables(likesDrawableFill, top: null, right: null, bottom: null);
else
    likes.setCompoundDrawables(likesDrawable, top: null, right: null, bottom: null);
likes.setOnClickListener((v) ->
{
    int temp = ShareProfile.profile.getLikes();
    temp++;
    likes.setText(String.format("赞 %d", temp));
    likes.setCompoundDrawables(likesDrawableFill, top: null, right: null, bottom: null);
    ShareProfile.profile.setLikes(temp);
    ShareProfile.liked = true;
});
```

效果就是每点击一次按钮，按钮上的数字就+1，如果没有点过赞（`ShareProfile.liked==false`），则设置一个没有点赞的图标，否则设置已经点了赞的图标。

最里层的第 2 个 `RelativeLayout` 用于显示基本资料：昵称、性别、生日及其相关信息。



里面是一个 `TextView` 和一个 `TableLayout`。

```
<RelativeLayout  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:padding="10dip">  
    <TextView  
        android:id="@+id/base_profile_text"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="基本资料"  
        android:textSize="20dip">  
    </TextView>  
    <TableLayout...>  
</RelativeLayout>
```

其中，`TableLayout` 里面有两个单独的 `TextView` 各自独占一行；还有两个 `TableRow`，各自有两个 `TextView` 各占一列。`android:stretchColumns="0,1"` 表明这两个列允许伸展，如果 `TableRow` 只有两个控件则两个控件在一行当中各占一半的空间。

```
<TableLayout  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:layout_below="@+id/base_profile_text"  
    android:stretchColumns="0,1">  
    <TextView...>  
    <TextView...>  
    <TableRow>  
        <TextView...>  
        <TextView...>  
    </TableRow>  
    <TableRow>  
        <TextView...>  
        <TextView...>  
    </TableRow>  
</TableLayout>
```

其余的两个垂直 `LinearLayout` 在布局上没有什么区别, 分别用于个性签名和个人介绍的设置。



```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="10dip">
    <TextView
        android:id="@+id/profile_motto_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="个性签名"
        android:textSize="20dip">
    </TextView>
    <TextView
        android:id="@+id/profile_motto_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="15dip"
        android:text="未设置">
    </TextView>
</LinearLayout>
```

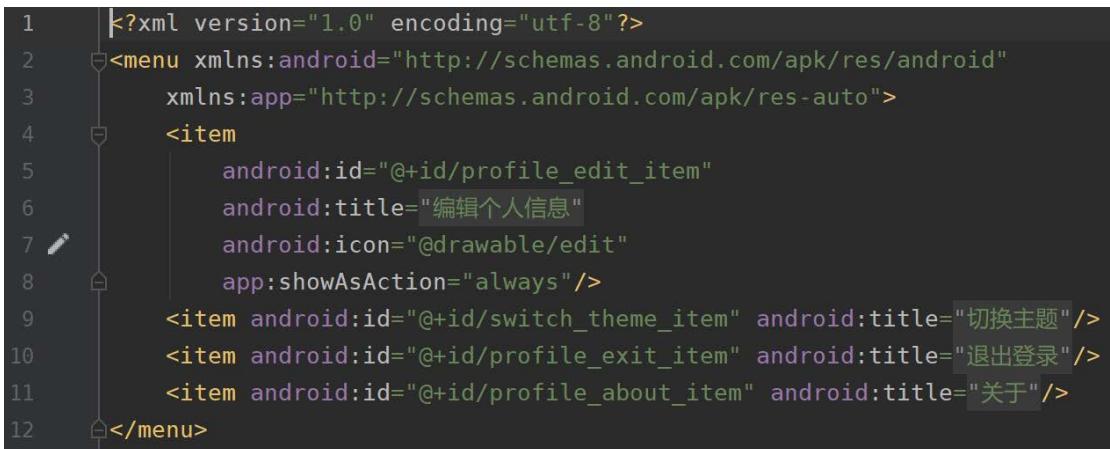
  

```
<LinearLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:padding="10dip">
    <TextView
        android:id="@+id/profile_introduction_text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="个人介绍"
        android:textSize="20dip">
    </TextView>
    <TextView
        android:id="@+id/profile_introduction_content"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:paddingTop="15dip"
        android:text="个人介绍">
    </TextView>
</LinearLayout>
```

与个人信息设置的相关代码这里就不列出了, 可以在项目文件夹中查阅相关的源代码。

### 3.2.2 菜单和返回键的设计

`ProfileActivity` 使用的菜单文件 `res/menu/profile_menu.xml` 如下:



```
1  |?xml version="1.0" encoding="utf-8"?>
2  |<menu xmlns:android="http://schemas.android.com/apk/res/android"
3  |    xmlns:app="http://schemas.android.com/apk/res-auto">
4  |        <item
5  |            android:id="@+id/profile_edit_item"
6  |            android:title="编辑个人信息"
7  |            android:icon="@drawable/edit"
8  |            app:showAsAction="always"/>
9  |        <item android:id="@+id/switch_theme_item" android:title="切换主题"/>
10 |        <item android:id="@+id/profile_exit_item" android:title="退出登录"/>
11 |        <item android:id="@+id/profile_about_item" android:title="关于"/>
12 |    </menu>
```

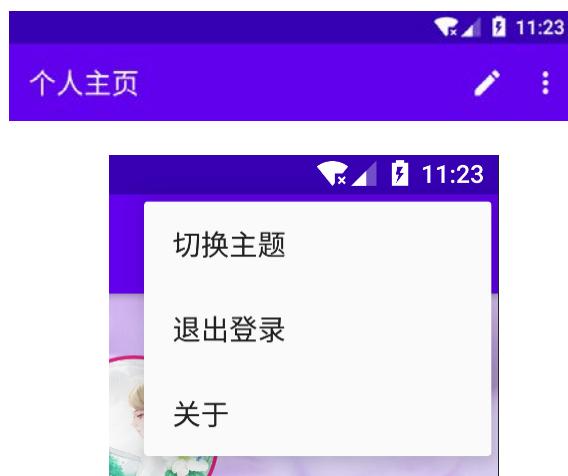
在标题栏右侧 (整个界面的右上角的画笔图标) 对应的菜单项的标题是“编辑个人信息”, 这里通过 `android:icon` 属性设置了显示的图标, 点击会切换到 `EditProfileActivity` 中。其他三项都隐藏在三个竖向排列的点的图标当中, 点击一下才能弹出。

以下是在 `ProfileActivity.java` 中设置菜单项源文件和响应事件的代码:

```
public boolean onCreateOptionsMenu(Menu menu) {
    getMenuInflater().inflate(R.menu.profile_menu, menu);
    return super.onCreateOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
    switch (item.getItemId()) {
        case R.id.profile_edit_item:
            SwitchActivity.switchActivity(srcActivity: ProfileActivity.this,
                EditProfileActivity.class, finish: true);
            break;
        case R.id.profile_exit_item:
            SwitchActivity.switchActivity(srcActivity: ProfileActivity.this,
                LoginActivity.class, finish: true);
            Toast.makeText(context: ProfileActivity.this, "已退出登录", Toast.LENGTH_SHORT).show();
            break;
        case R.id.switch_theme_item:
            switchTheme();
            break;
        case R.id.profile_about_item:
            new AboutDialog().showAboutDialog(srcActivity: ProfileActivity.this); // 弹出自定义的版权对话框
            break;
        default:
            break;
    }
    return super.onOptionsItemSelected(item);
}
```

菜单的显示效果如下：



**ProfileActivity.java** 的 `onKeyDown()` 方法设置了监听左下角返回键的代码，通过检测两次按下返回键的时间差来实现“再按一次退出程序”的功能，如果时间差小于 2s 则退出程序。

```
private long exitTime;
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if (keyCode == KeyEvent.KEYCODE_BACK) {
        if ((System.currentTimeMillis() - exitTime) > 2000) {
            Toast.makeText(context: ProfileActivity.this, "再按一次退出程序",
                           Toast.LENGTH_SHORT).show();
            exitTime = System.currentTimeMillis();
        } else {
            System.exit(status: 0);
        }
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```

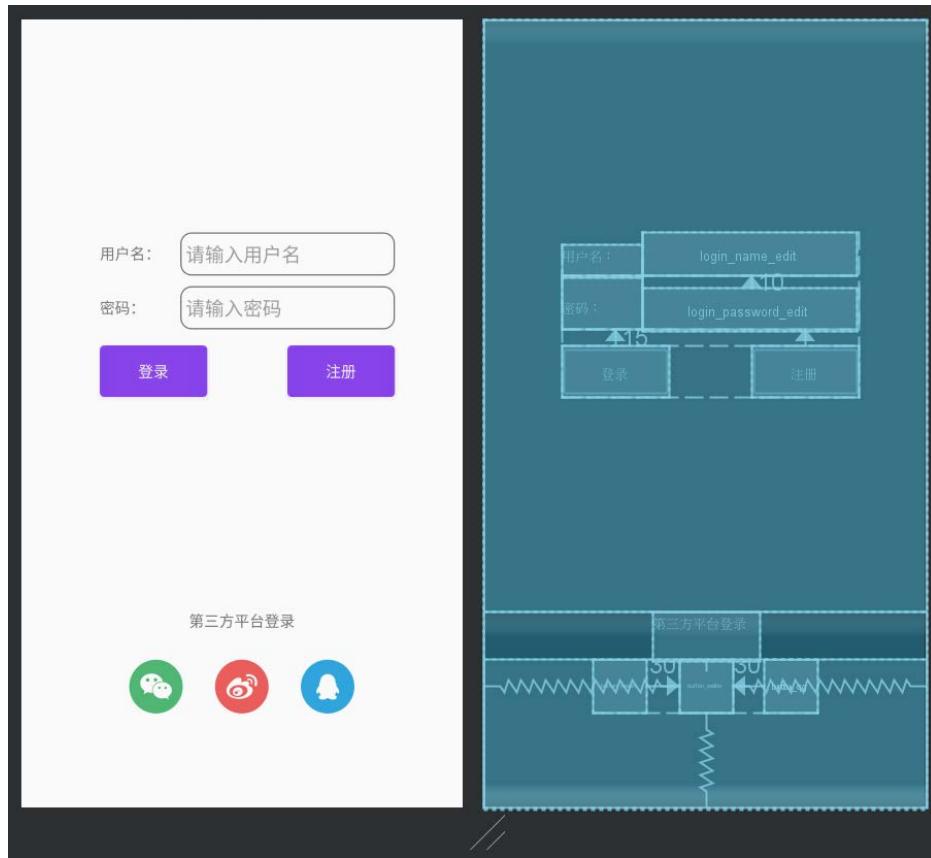
点击一次返回键的效果如下：



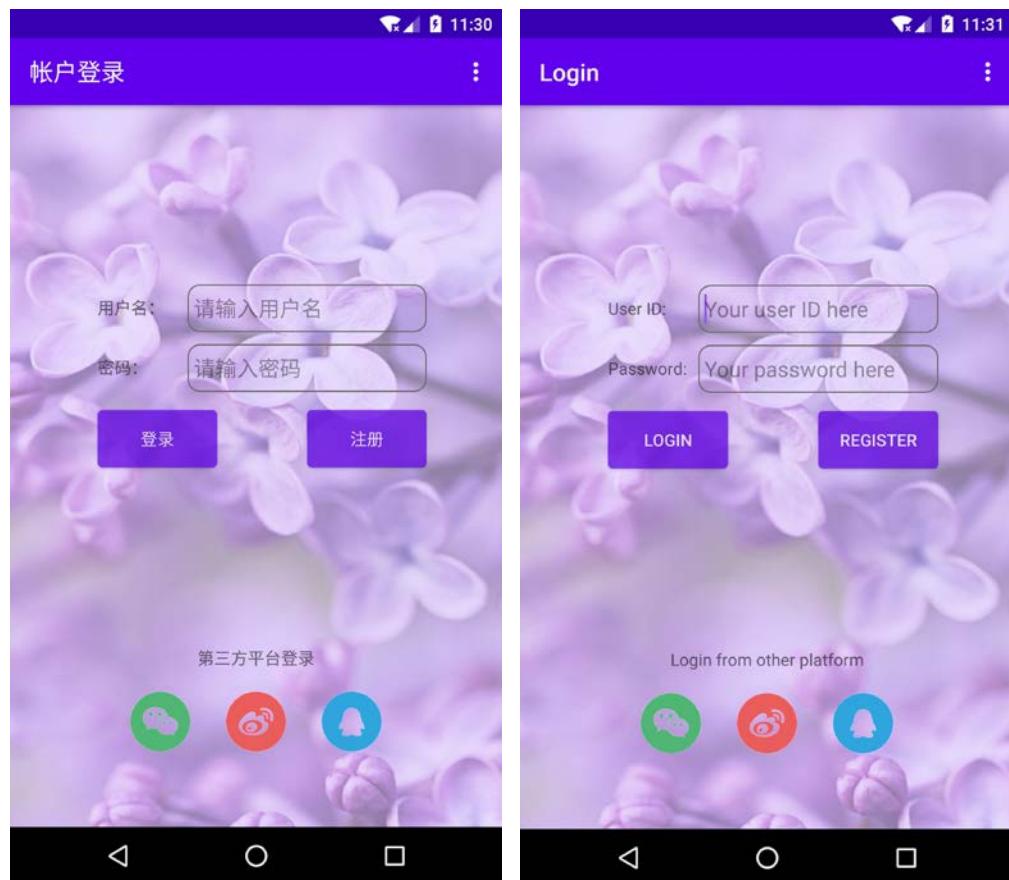
### 3.3 登录界面的实现

这个界面用于产生一个登录的效果，比前两个 **Activity** 的布局都简单很多。

从 `activity_login.xml` 文件的设计视图查看界面的大致布局情况：

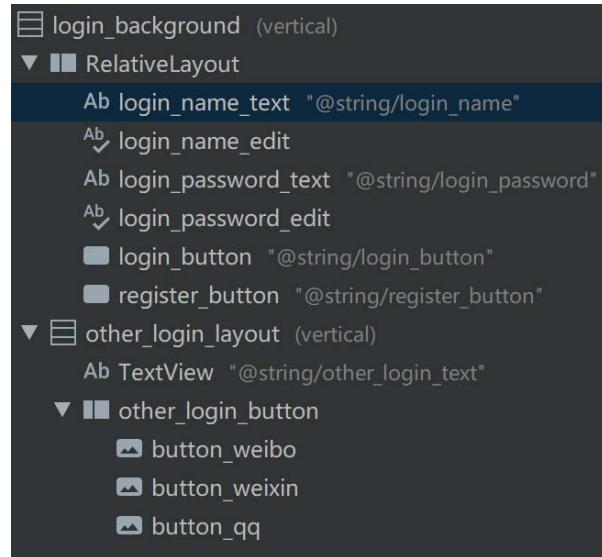


程序启动后界面的实际效果如下，左边的是在简体中文（中国）的系统语言环境下，右边的是在英语（美国）的系统语言环境下：



### 3.3.1 布局设计

切换视图查看整体布局代码：



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/login_background"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    android:gravity="center"
    tools:context=".LoginActivity">

    <RelativeLayout...>

    <LinearLayout...>

</LinearLayout>
```

最外层是垂直的 `LinearLayout`, 里面嵌套了一个 `RelativeLayout` 和一个垂直的 `LinearLayout`。

两个内层的布局属性设置如下：

```
<RelativeLayout  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:gravity="center"  
    android:layout_weight="1">  
  
    <TextView...>  
    <AutoCompleteTextView...>  
    <TextView...>  
    <AutoCompleteTextView...>  
    <Button...>  
    <Button...>  
  
</RelativeLayout>  
  
<LinearLayout  
    android:id="@+id/other_login_layout"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
    android:layout_weight="3"  
    android:layout_gravity="center"  
    android:orientation="vertical">
```

属性 `android:layout_weight` 设置子控件或子布局所在的区域占外层 `LinearLayout` 的比重，这个比重在这里是成反比的（在其他地方可能成正比）。上图同时列出了 `RelativeLayout` 里面的控件。其中用户名输入框和密码输入框使用了 `AutoCompleteTextView`，实现自动补全输入的用户名的密码，属性设置如下：

```
<AutoCompleteTextView  
    android:id="@+id/login_name_edit"  
    android:layout_width="wrap_content"  
    android:layout_height="40dip"  
    android:hint="请输入用户名"  
    android:layout_toRightOf="@+id/login_name_text"  
    android:layout_toEndOf="@+id/login_name_text"  
    android:minWidth="200dip"  
    android:maxWidth="200dip"  
    android:padding="5dip"  
    android:background="@drawable/motto_edit_text"  
    android:inputType="text"  
    android:completionThreshold="1"  
    android:autofillHints="">  
</AutoCompleteTextView>
```

```
<AutoCompleteTextView  
    android:id="@+id/login_password_edit"  
    android:layout_width="wrap_content"  
    android:layout_height="40dip"  
    android:hint="请输入密码"  
    android:layout_toRightOf="@id/login_password_text"  
    android:layout_toEndOf="@id/login_password_text"  
    android:layout_below="@id/login_name_edit"  
    android:layout_alignLeft="@id/login_name_edit"  
    android:layout_alignStart="@id/login_name_edit"  
    android:layout_alignRight="@id/login_name_edit"  
    android:layout_alignEnd="@id/login_name_edit"  
    android:background="@drawable/motto_edit_text"  
    android:layout_marginTop="10dip"  
    android:padding="5dip"  
    android:inputType="textPassword"  
    android:completionThreshold="1"  
    android:autofillHints="">  
</AutoCompleteTextView>
```

其中属性 `android:inputType="textPassword"` 设置输入文本的属性为密码  
(输入的文本会自动用 • 隐藏)。属性设置

`android:completionThreshold="1"` 表明输入框内只有一个字符就开始匹配  
补全的候选词。

在 `LoginActivity.java` 的 `LoginActivity` 类中, 使用 `Adapter` 可以将 `String`  
数组的内容导入到 `AutoCompleteTextView` 的候选词当中:

```
/* 登录按钮设置 */  
String[] names = { ShareProfile.profile.USERID };  
String[] passwords = { ShareProfile.profile.getPassword() };  
final Button login = (Button) findViewById(R.id.login_button);  
final AutoCompleteTextView password = (AutoCompleteTextView) findViewById(R.id.login_password_edit);  
final AutoCompleteTextView name = (AutoCompleteTextView) findViewById(R.id.login_name_edit);  
ArrayAdapter<String> passwordAdapter = new ArrayAdapter<~>( context: this,  
    android.R.layout.simple_list_item_1, passwords);  
password.setAdapter(passwordAdapter);  
ArrayAdapter<String> nameAdapter = new ArrayAdapter<~>( context: this,  
    android.R.layout.simple_list_item_1, names);  
name.setAdapter(nameAdapter);
```

`AutoCompleteTextView` 的效果如下:



除此之外还有登录和注册两个按钮，这里只展示登录按钮的属性设置：

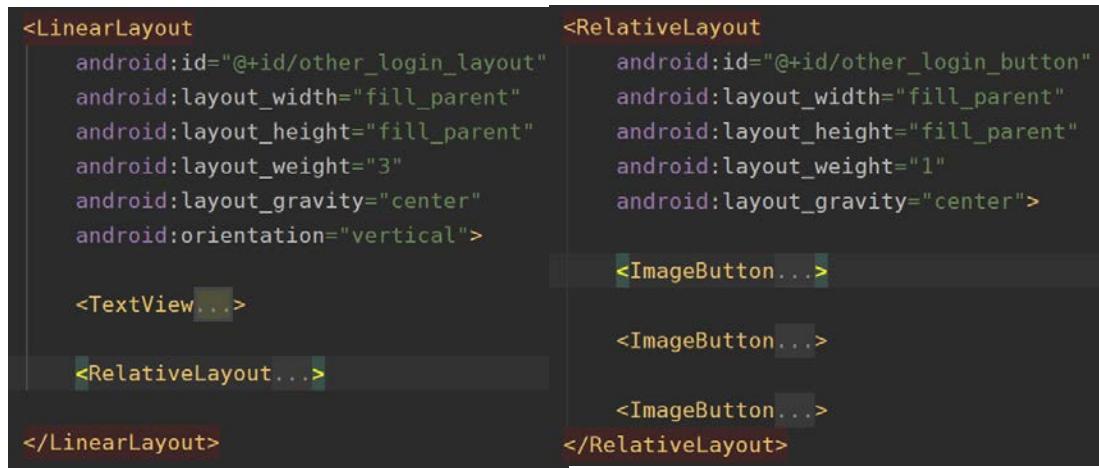
```
<Button  
    android:id="@+id/login_button"  
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_below="@+id/login_password_text"  
    android:layout_alignLeft="@+id/login_password_text"  
    android:layout_alignStart="@+id/login_password_text"  
    android:text="登录"  
    android:textColor="#ffffff"  
    android:alpha="0.7"  
    android:layout_marginTop="15dip"  
    android:minWidth="100dip"  
    android:maxWidth="100dip"  
    android:background="@drawable/button_rectangle">  
</Button>
```

其中属性 `android:alpha="0.7"` 用来设置按钮背景的透明度，值在 0 和 1 之间，值越小越透明。同时设置 `android:minWidth="100dip"` 和 `android:maxWidth="100dip"` 是为了限制死控件的宽度。属性 `android:background="@drawable/button_rectangle"` 设置了这个按钮的自定义背景样式，这个背景样式文件 `res/drawable/button_rectangle.xml` 内容如下所示：

```
1  <?xml version="1.0" encoding="utf-8"?>  
2  <selector xmlns:android="http://schemas.android.com/apk/res/android">  
3      <item android:state_pressed="true">  
4          <shape android:shape="rectangle">  
5              <solid android:color="@color/colorPrimaryDark"/>  
6              <corners android:radius="4dip"/>  
7          </shape>  
8      </item>  
9      <item>  
10         <shape android:shape="rectangle">  
11             <solid android:color="@color/colorPrimary"/>  
12             <corners android:radius="4dip"/>  
13         </shape>  
14     </item>  
15 </selector>
```

`<selector>` 标签表明这是一个选择器。其中 `<item>` 标签的属性 `android:state_pressed="true"` 表示设置按钮按下时的样子。`<shape>` 标签的属性 `android:shape="rectangle"` 为设置按钮形状为矩形。

在内层的垂直 `LinearLayout` 当中，有一个 `TextView` 和一个嵌套的 `RelativeLayout`。嵌套的 `RelativeLayout` 里面有三个 `ImageButton`，分别是微信、微博、QQ 登录按钮。



其中一个 `ImageButton` 的属性设置如下：



### 3.3.2 登录效果的实现与展示

这里虽然设置了从微信、微博、QQ 登录的按钮，但是都只是简单地做了一个从微信、微博、QQ 登录的效果，并在个人信息展示界面显示“登录的来源”。

在 `LoginActivity.java` 的 `LoginActivity` 类中，`login()` 函数实现了简单的登录效果：

```
1  public void login(LoginFrom from) {  
2      dialog = new ProgressDialog(LoginActivity.this); // ProgressDialog  
3      String msg = "";  
4      switch (from) { // 设置在 ProgressDialog 中显示的登录信息  
5          case QQ:  
6              msg = getString(R.string.qq_login);  
7              break;  
8          case WEIBO:
```

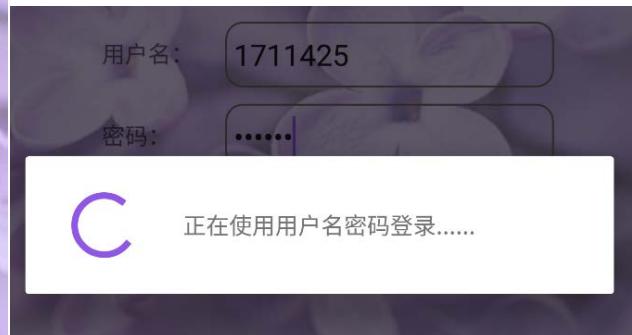
```
9         msg = getString(R.string.weibo_login);
10        break;
11    case WEIXIN:
12        msg = getString(R.string.weixin_login);
13        break;
14    case NAME:
15        msg = getString(R.string.name_login);
16        break;
17    default:
18        return;
19    }
20    ShareProfile.profile.setFrom(from); // 设置要在个人信息展示界面显示的
21    登录信息
22    dialog.setMessage(msg);
23    // 设置点击对话框之外的区域，对话框不消失
24    dialog.setCanceledOnTouchOutside(false);
25    dialog.show();
26    // 切换到个人信息展示界面
27    SwitchActivity.switchActivity(LoginActivity.this,
ProfileActivity.class, true);
28 }
```

以下是一些尝试登录的效果图：

点击“注册”按钮，由于我不在这里实现注册功能，因此就简单提示不能注册：



尝试点击“登录”按钮：



点击 QQ 按钮（之后马上会切换到个人信息展示界面）：



## 3.4 欢迎界面的实现

这是布局最简单的一个界面了，直接使用了 Android Studio 默认生成的 **ConstraintLayout**，里面只通过拖拽的方式加了两个 **TextView**：对应的布局文件 `activity_main.xml` 如下所示

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main_background"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <TextView...>

    <TextView...>

</androidx.constraintlayout.widget.ConstraintLayout>
```



The screenshot shows the Android Studio XML layout editor. A constraint set named "main\_background" is selected, indicated by a blue border. Inside this set, there are two `TextView` elements. The first `TextView` has the ID `@+id/textView2` and contains the text "Hello World!". The second `TextView` has the ID `@+id/textView` and contains the text "@string/welcome". Both `TextView` elements have their `textColor` set to `#ffffffff`.

其中两个 `TextView` 的属性设置如下：

```
<TextView
    android:id="@+id/textView2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Hello World!"
    android:textColor="#ffffffff"
    android:textSize="20sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.387" />
```

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Welcome"
    android:textColor="#ffffffff"
    android:textSize="40sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent"
    app:layout_constraintVertical_bias="0.119" />
```

在 `MainActivity.java` 中，`onCreate()` 函数定义如下，实现了 3 秒后以淡出淡入的动画效果自动切换到登录界面：

```
@Override  
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    // 3秒后自动切换到登录界面  
    new Handler().postDelayed(new Runnable() {  
        @Override  
        public void run() {  
            SwitchActivity.switchActivity(srcActivity: MainActivity.this, LoginActivity.class, finish: true);  
            overridePendingTransition(R.anim.fade_out, R.anim.fade_in); // 淡出淡入效果  
        }  
    }, delayMillis: 3000);  
}
```

其中 `postDelayed()` 方法通过 `Handler` 调用，第一个参数是一个 `Runnable` 对象（表示新开了一线程），`run()` 方法里面的语句会被新开的线程延迟执行，第二个参数指定了延迟执行的时间（ms）。

`overridePendingTransition()` 函数用来设置切换 `Activity` 的动画效果。动画效果的文件在 `res/anim` 文件夹中，`fade_in.xml` 和 `fade_out.xml` 的内容分别如下，参考了网上的代码：

```
<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android">
```

```
    <alpha  
        android:duration="500"  
        android:fromAlpha="1.0"  
        android:interpolator="@android:anim/accelerate_interpolator"  
        android:toAlpha="0.0" />
```

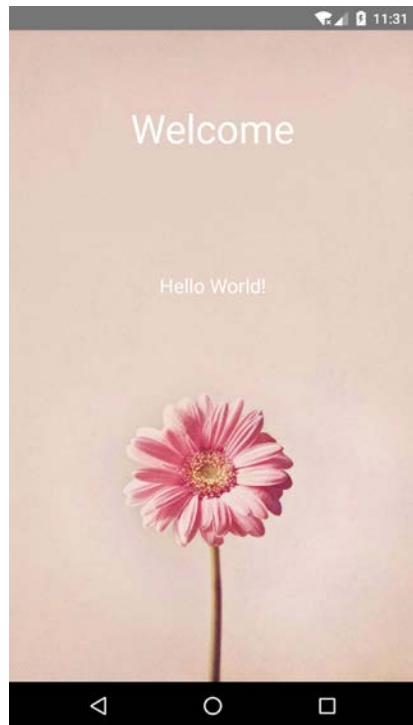
```
</set>
```

```
<?xml version="1.0" encoding="utf-8"?>  
<set xmlns:android="http://schemas.android.com/apk/res/android">
```

```
    <alpha  
        android:duration="500"  
        android:fromAlpha="0.0"  
        android:interpolator="@android:anim/accelerate_interpolator"  
        android:toAlpha="1.0" />
```

```
</set>
```

欢迎界面的效果如下（之后会以淡出淡入的形式跳转到登录界面）：



### 3.5 自定义对话框的实现

这里的自定义对话框用来显示一些版权信息，包括学号、GitHub、完成时间。布局文件 `activity_about_dialog.xml` 用来自定义对话框的布局。从文件名就知道原来我把它定义成了 Activity，然后在 `AndroidManifest.xml` 中通过属性 `android:theme="@style/Theme.AppCompat.DayNight.Dialog"` 将这个 Activity 的设置成对话框主题，但是这样的效果感觉不好看，于是就使用 `AlertDialog.Builder` 类将布局文件通过 `setView()` 方法加载到 `AlertDialog` 中。

以下是 `activity_about_dialog.xml` 的基本布局情况，最外层是一个垂直的 `LinearLayout`，里面有一个 `CircleImageView`、3 个 `TextView` 还有一个不会显示出来的 `Button`（原来是想让这个 `Button` 按钮充当确定按钮）。

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/dialog_about"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical"
    tools:context=".AboutDialog">

    <de.hdodenhof.circleimageview.CircleImageView...>
    <TextView...>
    <TextView...>
    <TextView...>
    <Button
        android:id="@+id/about_yes_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="10dip"
        android:layout_marginLeft="23dip"
        android:layout_marginStart="23dip"
        android:layout_marginRight="10dip"
        android:layout_marginEnd="10dip"
        android:textSize="15sp"
        android:text="确定"
        android:visibility="gone">
    </Button>
</LinearLayout>
```

注意 **Button** 的属性设置 `android:visibility="gone"` 表示不仅设置控件不可见而且界面不保留该控件所占的空间。

**CircleImageView** 也设置了一个属性 `android:layout_gravity="center"`，表示控件将在父布局中间对齐。

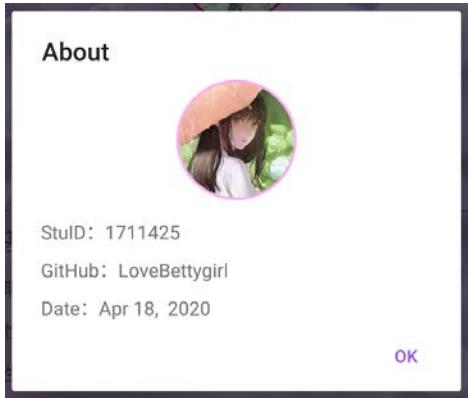
在 `AboutDialog.java` 的 `showAboutDialog()` 方法中，实现了自定义对话框的功能，这样通过 `AboutDialog` 类的对象调用这个方法就可以直接显示对话框了。

```
public void showAboutDialog(AppCompatActivity srcActivity) {
    AlertDialog.Builder aboutDialog =
        new AlertDialog.Builder(srcActivity);
    final View dialogView = LayoutInflater.from(srcActivity)
        .inflate(R.layout.activity_about_dialog, root: null);
    aboutDialog.setTitle("About");
    aboutDialog.setView(dialogView); // 向对话框加载自定义布局文件
    aboutDialog.setPositiveButton( text: "OK",
        new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                dialog.dismiss();
            }
        });
    aboutDialog.show();
}
```

自定义对话框只设定了一个确定按钮，设置监听器

`DialogInterface.OnClickListener` 监听确定按钮是否被单击，如果被单击则关闭对话框。同时点击对话框外侧的部分也将关闭对话框。

自定义对话框的显示效果如下：



## 3.6 切换主题的实现

在个人信息展示界面的菜单项“切换主题”中，点击这个菜单项会弹出一个对话框，显示了当前的主题和其他可切换的主题。

实现的主体思路是：`ProfileActivity.java` 的 `ProfileActivity` 类的 `switchActivity()` 函数实现弹出这样的对话框并设置全局变量 `ShareProfile.theme` 的值，然后重启当前的 `Activity`，在这个 `Activity` 的 `onCreate()` 函数中，调用自己定义的 `setDiyTheme()` 函数（下图只截取部分），里面又根据 `ShareProfile.theme` 的值调用了 `setTheme()` 函数加载对应的主题，真正实现动态切换主题的功能。然后我还定义了其他一些函数实现根据主题设置切换控件的颜色，这个就不列出了。当然，这只是一个简单思路的实现，事实上 app 切换主题的方式会比这个过渡更自然且性能较好。

```
1 void switchTheme() {  
2     AlertDialog.Builder builder = new  
3         AlertDialog.Builder(ProfileActivity.this);  
4     builder.setTitle(getString(R.string.switch_theme_text));  
5     final String[] items = { getString(R.string.purple_theme),  
6         getString(R.string.red_theme),  
7         getString(R.string.yellow_theme),  
8         getString(R.string.green_theme),  
9         getString(R.string.pink_theme),  
10        getString(R.string.blue_theme) };  
11    final boolean[] checkedItems = new boolean[items.length];  
12    // 向AlertDialog 加载并设置选中的主题列表项
```

```

12         builder.setSingleChoiceItems(items, ShareProfile.theme, new
13             DialogInterface.OnClickListener() {
14                 @Override
15                 public void onClick(DialogInterface dialog, int which) {
16                     for (int i = 0; i < checkedItems.length; i++) {
17                         checkedItems[i] = false;
18                     }
19                 }
20             });
21         builder.setNegativeButton(getString(R.string.cancel), new
22             DialogInterface.OnClickListener() {
23                 @Override
24                 public void onClick(DialogInterface dialog, int which) {
25                     dialog.dismiss();
26                 }
27             });
28         builder.setPositiveButton(getString(R.string.ok), new
29             DialogInterface.OnClickListener() {
30                 @Override
31                 public void onClick(DialogInterface dialog, int which) {
32                     for (int i = 0; i < checkedItems.length; i++) {
33                         if (checkedItems[i]) {
34                             ShareProfile.theme = i;
35                             break;
36                         }
37                     }
38                     dialog.dismiss();
39                     // 重启当前界面才能使重新设置的主题生效
40                     finish();
41                     startActivity(getIntent());
42                     Toast.makeText(ProfileActivity.this,
43                         getString(R.string.set_theme_success),
44                         Toast.LENGTH_SHORT).show();
45                 }
46             });
47         builder.create().show();
48     }

```

```

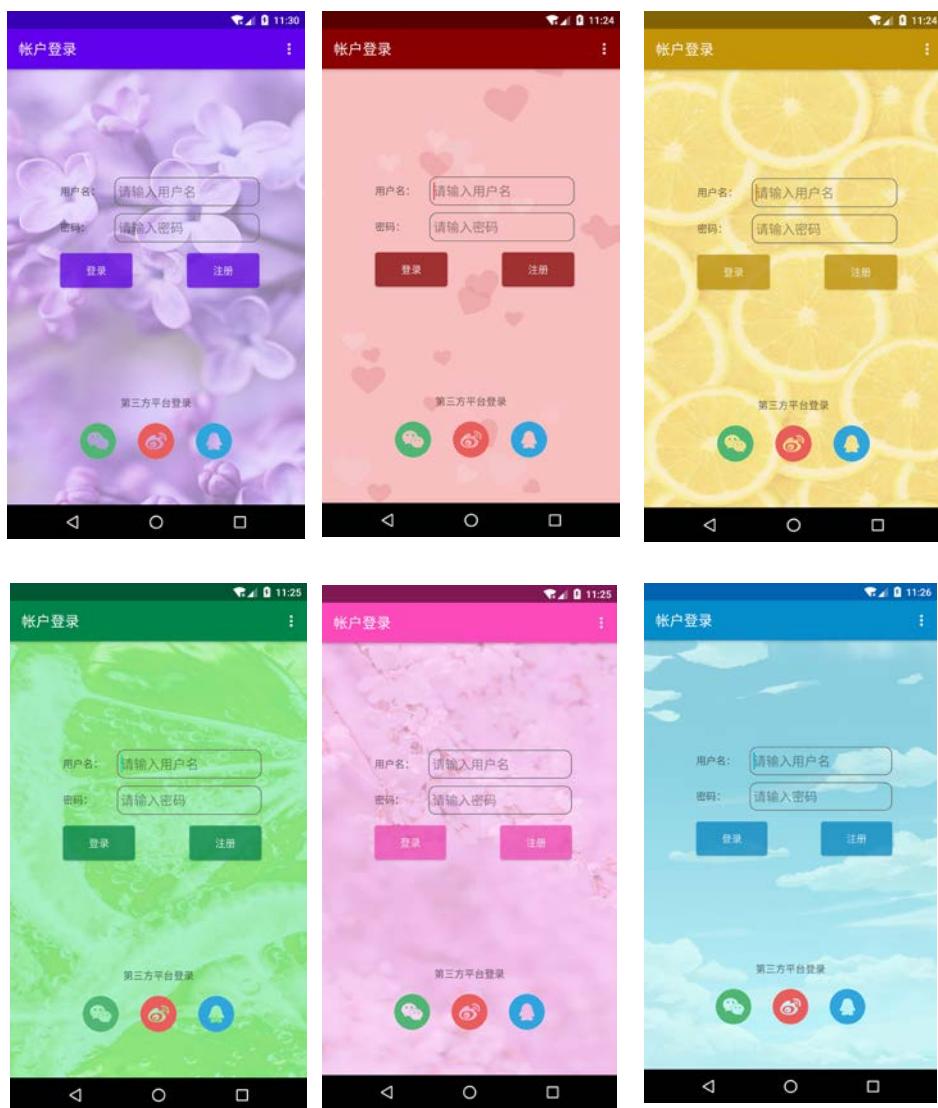
private void setDiyTheme() {
    switch (ShareProfile.theme) {
        case 0:
            setTheme(R.style.PurpleTheme);
            break;
        case 1:
            setTheme(R.style.RedTheme);
            break;
    }
}

```

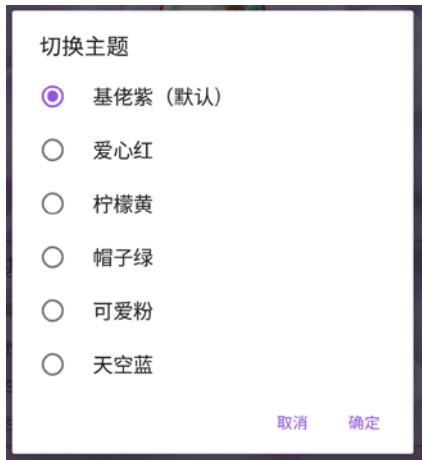
自己定义好的主题在 `res/values/styles.xml` 中, 这里只展示其中一个自定义主题的设置:

```
17 <style name="RedTheme" parent="Theme.AppCompat.Light.DarkActionBar">
18     <!-- Customize your theme here. -->
19     <item name="colorPrimary">@color/RedColorPrimary</item>
20     <item name="colorPrimaryDark">@color/RedColorPrimaryDark</item>
21     <item name="colorAccent">@color/RedColorAccent</item>
22     <item name="android:windowBackground">@color/RedBackground</item>
23 </style>
```

6 个自定义主题的显示效果如下 (仅展示登录界面):



主题选择的对话框如下:



### 3.7 国际化的简单实现

参考了上课 ppt 的内容。新建了文件夹 `res/values-en-rUS`, 加入了 `strings.xml` 文件。`res/values/strings.xml` (默认显示的) 和 `res/values-en-rUS/strings.xml` 对比如下:

```
<resources>
    <string name="app_name">UIDesign</string>
    <string name="welcome">Welcome</string>
    <string name="login_name">用户名: </string>
    <string name="login_name_hint">请输入用户名</string>
    <string name="login_password">密码: </string>
    <string name="login_password_hint">请输入密码</string>
    <string name="login_button">登录</string>
    <string name="register_button">注册</string>
    <string name="other_login_text">第三方平台登录</string>
    <string name="about">关于</string>
```

```
<resources>
    <string name="app_name">UIDesign</string>
    <string name="welcome">Welcome</string>
    <string name="login_name">User ID: </string>
    <string name="login_name_hint">Your user ID here</string>
    <string name="login_password">Password: </string>
    <string name="login_password_hint">Your password here</string>
    <string name="login_button">Login</string>
    <string name="register_button">Register</string>
    <string name="other_login_text">Login from other platform</string>
    <string name="about">About</string>
```

其实我只翻译了很小一部分字符串，其他很多字符串都没翻译。

## 四、实验遇到的问题及其解决方法

### 4.1 自定义界面背景程序会崩溃

我直接在 xml 中以 android:background 属性设置背景图片 (drawable)，结果程序每次运行都崩溃，百度了以下说一定要用.png 格式的图片。然后把格式换成.png 也不行，之后只能参考网上的代码，重写 **Activity** 的 **onResume()** 函数，在这里加载背景，然后就成功了。代码如下（对应 **MainActivity** 类的 **onResume()** 函数）：

```
@Override  
protected void onResume() {  
    super.onResume();  
    ConstraintLayout layout = (ConstraintLayout) findViewById(R.id.main_background);  
    InputStream is;  
    BitmapFactory.Options opt = new BitmapFactory.Options();  
    opt.inPreferredConfig = Bitmap.Config.ARGB_8888;  
    opt.inPurgeable = true;  
    opt.inInputShareable = true;  
    opt.inSampleSize = 2;  
    is = getResources().openRawResource(+R.drawable.welcome);  
    Bitmap bm = BitmapFactory.decodeStream(is, null, opt);  
    BitmapDrawable bd = new BitmapDrawable(getResources(), bm);  
    layout.setBackgroundDrawable(bd);  
}
```

### 4.2 切换界面出现 android.view.WindowLeaked 异常

最终发现是我没有关闭 **ProgressDialog** 就启动了新的 **Activity**，导致了 **ProgressDialog** 窗口泄露。解决的方式就是延迟（使用 **postDelayed()** 函数）关闭 **ProgressDialog** 的时间，关闭的时候判断一下创建它的上下文（**Activity**）的状态，由这个决定是否关闭对话框。代码已经在前面列出。

2020.4.27 修改：

交了作业之后，我才拿到真机，真机运行了一遍发现闪退了，然后我又放在另一个模拟器里面就发现显示对话框的方法 **show()** 又引起了这个问题，于是就在 **LoginActivity** 的 **login()** 函数中把延时的代码去掉（**login()** 函数的代码在 3.3.2 节已列出），重写 **LoginActivity** 的 **onDestroy()** 函数，在销毁 **Activity** 时关闭 **ProgressDialog** (**dismiss()** 方法) 就真的没问题了。

```
@Override  
protected void onDestroy() {  
    if(dialog != null) {  
        dialog.dismiss();  
    }  
    super.onDestroy();  
}
```

### 4.3 如何实现每个控件都要有处理函数？

我上面没有提到个人信息编辑界面的 `TextView` 和 `ProgressBar` 设置监听事件的问题。事实上我在 `com.example.uidesign` 包中定义了 `RandomListener` 类：

```
public class RandomListener implements View.OnClickListener {  
    private static String[] str = {  
        "? ? ? ",  
        "别点我呀~",  
        "干什么? ",  
        "有事吗? "  
    };  
    private AppCompatActivity activity;  
    public RandomListener(AppCompatActivity activity) {  
        this.activity = activity;  
    }  
  
    @Override  
    public void onClick(View v) {  
        Random random = new Random();  
        Toast.makeText(activity, str[random.nextInt(bound: 4)], Toast.LENGTH_SHORT).show();  
    }  
}
```

用于告知用户误触了并没有功能实现的控件。效果如下：



### 4.4 模拟器分辨率问题

用 Android SDK 自带的模拟器，启动之后手机框显示非常大，屏幕显示只占了 1/4。原因是我的电脑屏幕分辨率太高了。然后我就更改系统显示的 dpi，然后就没这个问题了，但是这样电脑屏幕看起来很不舒服。然后就在不改系统 dpi 的情况下更改启动模拟器程序显示的分辨率，也没有这个问题了，但是手机屏幕非常模糊。然后就只能用第三方模拟器 Genymotion 了，也可以连接到 ADB 进行调

试。

## 4.5 其他问题

在实验的过程中，不可避免会出现一些错误导致程序崩溃，然后起初我不懂，每次一崩溃就查百度，后来才知道应该要先查阅 Logcat，先尝试从这里发现问题。

## 五、实验结论

完成了这次实验之后，我感觉我对 Android UI 设计和控件的使用、界面布局和事件处理有了一些入门。在实验过程中，感到最费时的还是界面布局的设计，基本上每一个控件的摆放位置都调整了很久，才能达到现在所看到的效果。由此我发现做前端也是一门技术活。

Android Studio 的设计视图（控件的拖拽）也许只有 **ConstraintLayout** 才会更方便一些。布局的代码与处理函数的代码分离让我想到了以前自学的 MFC 框架也是这样的。

对于以前从未接触过移动平台开发的我来说，对于 Android 开发更是新手中的新手，很多我想实现的东西都要借助上课的 ppt、参考书、API 和网络博客才能完成。因此想要在大作业中真正实现一个完整功能的 app，还有很长的路要走。