2 登录模块

从这一章开始我们进入应用的设计开发阶段，首先设计登录模块。 需要注意的是，小编的开发环境从Mac切换到了Windows，以后发布的Lab Manual也基于Windows平台的程序制作。

> “ 题外话：在切换到Windows平台后，小编发现AS（Android Studio，下同）文本编辑器中的字体很不友好，如数字0和大写字母O仅凭肉眼无法区分。 小编随手搜了一些“适合编程字体”，参照下面的“编程字体选择标准”，换了一个字体，看起来好了很多，有相同情况的朋友可以试一下。
>
> - 非常清晰的字体
> - 支持扩展字符集，否则会出现乱码或者方格
> - 对空白把握的很好（包括但不限于空格、非中断空格、制表符...）
> - 可轻易区分'l', 'ı' 和 'i'
> - 可轻易区分'0', 'o' 和 'O'
> - 可轻易区分左右引号 - 最好是对称的
> - 清晰的标点符号，特别是大括号小括号和中括号

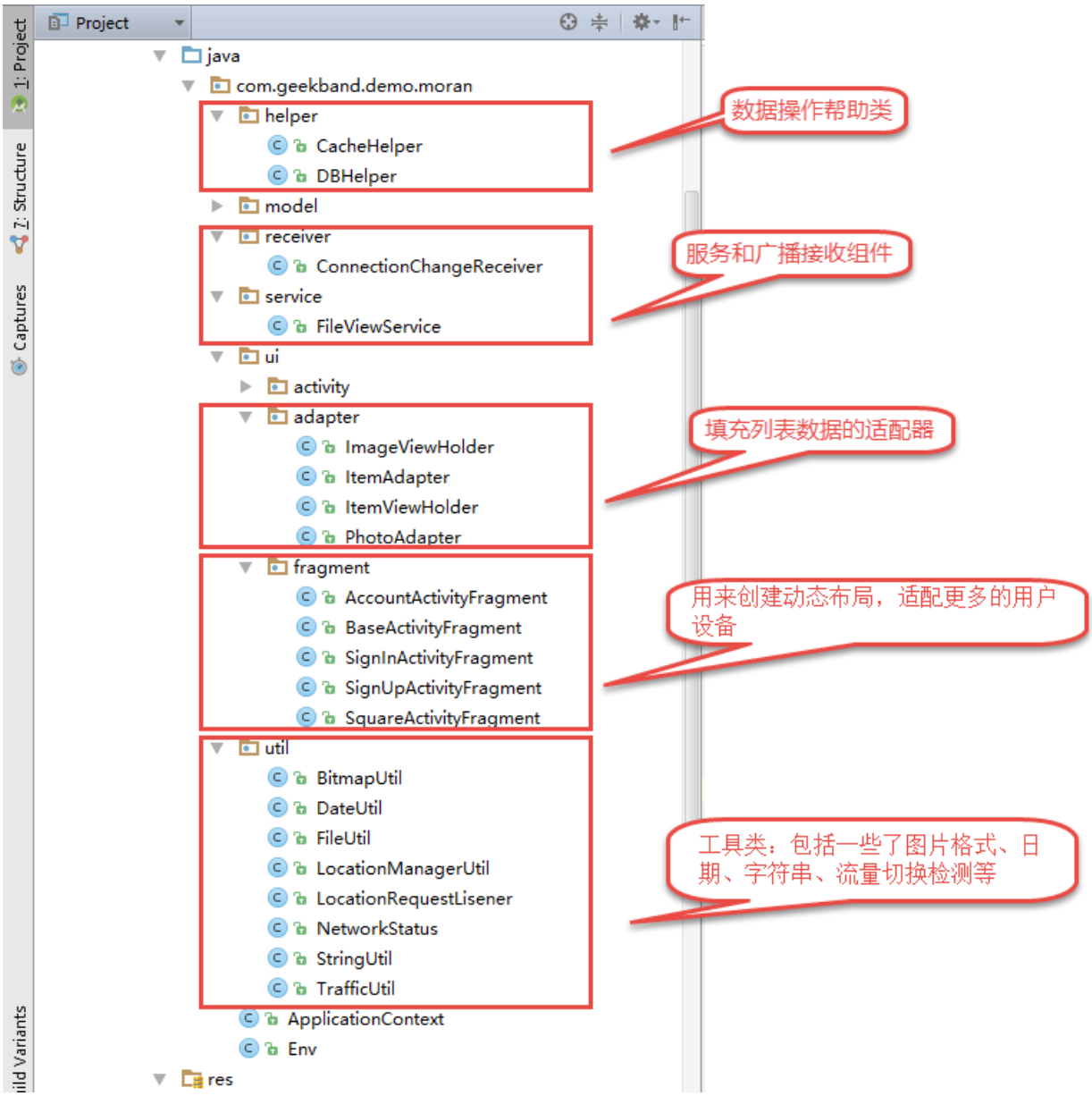
开发环境

操作系统：Windows 10

开发工具：Android Studio 1.4

操作步骤

2.1 框架搭建



如图所示，在上一章基础上，加入其他组件，把代码框架搭建起来。

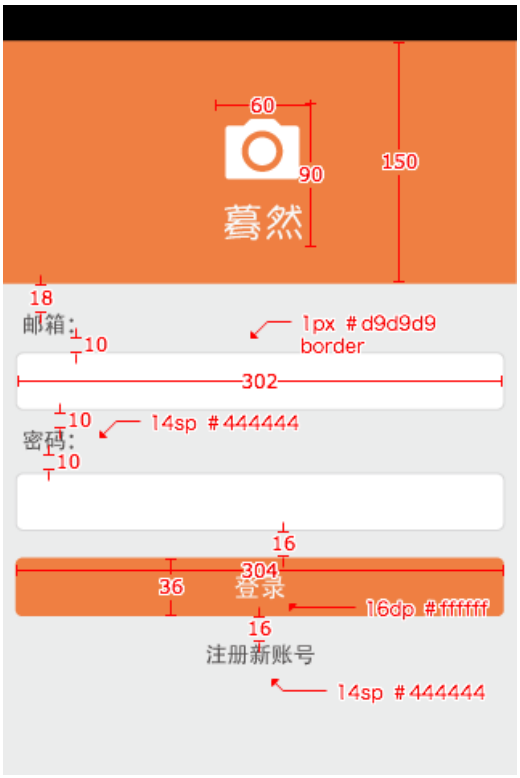
这里加入了一些常见的工具类，以后需要用到的时候再详细介绍，大家也可以自行到网上搜索相关资源。

用户界面进行了重构，加入fragment进行布局，忘了如何重构的复习下项目初始化lab manual。

细心的朋友可能已经发现，上图中还少了一个内容提供者（ContentProvider）组件，大家有需要可自行扩展程序。

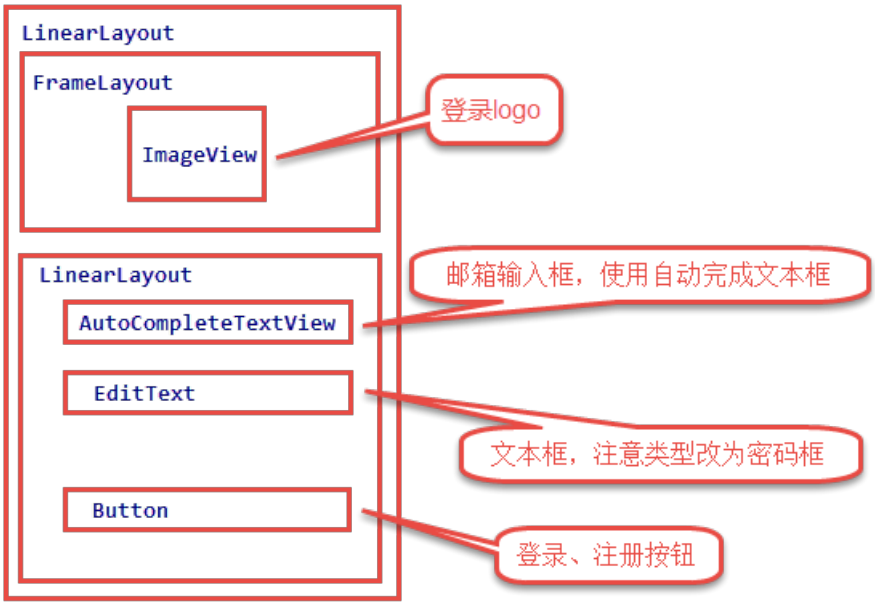
2.2 界面设计

（1）界面设计



先观察下登录原型，设计师已经帮我们做好了一些标注（如果还有其他需要可以自己找一个标注软件）。

最上方的黑色矩形是预留的系统状态栏部分，我们只需要关注下方的部分。简单的画一下设计布局：



我们把界面分成上下两个部分，图片logo放在层布局的中间，下面的表单放在垂直方向的线性布局里，在最外层嵌套一个线性布局。

（2）字符串资源组织

我们把程序需要的字符串资源统一组织起来，方便复用和国际化。 打开res/values/strings.xml，添加我们需要的字符串：

```
<resources>
    <string name="app_name">蓦然</string>
    <!--登录、注册-->
    <string name="prompt_email">邮箱：</string>
    <string name="prompt_password">密码：</string>
    <string name="prompt_comfirm_password">重复密码：</string>
    <string name="action_sign_in">登录</string>
    <string name="action_sign_up">注册</string>
    <string name="sign_up">注册新账号</string>
    <string name="sign_in">登录账号</string>
</resources>
```

默认会有一个名为app_name的string标签，这就是在手机上显示的应用名称，我们可以把它的值改成中文“蓦然”。

然后大家就可以自己添加需要的字符串了，name可按照喜欢的规则自己定义，可以把同一个模块的资源定义在一起，加上注释。

这里小编把登录和注册的资源放在一起，复用的资源取名时就应该取一个通用的名字了，比如这里的邮箱、密码标签等。

（3）尺寸资源组织

我们同样可以把内外边距等尺度资源组织在一起。打开res/values/dimens.xml，定义相应资源：

```
<resources>
    <!-- Default screen margins, per the Android Design guidelines. -->
    <dimen name="activity_horizontal_margin">16dp</dimen>
    <dimen name="activity_vertical_margin">16dp</dimen>
    <!--登录、注册-->
    <dimen name="logo_container_height">150dp</dimen>
    <dimen name="logo_width">60dp</dimen>
    <dimen name="logo_height">90dp</dimen>
    <dimen name="sign_vertical_padding">10dp</dimen>
    <dimen name="sign_vertical_margin">10dp</dimen>
    <dimen name="sign_view_height">36dp</dimen>
    <dimen name="sign_text_size">14sp</dimen>
    <dimen name="sign_button_size">16sp</dimen>
</resources>
```

登录和注册的设计有很多相同的地方，我们可以把这些相同的尺度资源整合在一起。这样做有利有弊，管理起来规范，但设计时可能来回切换，究竟哪些资源应该复用，哪些不应该，大家自己把握了。

（4）配色方案设计

同样，我们把颜色资源也整合在一起，这是非常有必要的，因为一般应用中的颜色就那么几种，但使用的地方却非常多。

打开res/values/colors.xml，添加应用中几种常见颜色：

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="colorPrimary">#3F51B5</color>
    <color name="colorPrimaryDark">#303F9F</color>
    <color name="colorAccent">#FF4081</color>
    <!--以下为添加的应用配色-->
    <color name="colorBackground">#ebecec</color>
    <color name="colorOrange">#ef7f42</color>
    <color name="colorGrey">#444</color>
    <color name="colorWhite">#fff</color>
    <color name="colorBorder">#d9d9d9</color>
</resources>
```

上半部分为手机主题的配色资源，保留就好，不做改动。背景和字体有用到相同颜色，所以为使用方便这样命名了，大家可取自己喜欢的资源名。

（5）形状资源设计

原型中文本框和按钮是圆角矩形，而且颜色不一样，我们可以绘制两个圆角矩形，作为登录注册界面文本框和按钮

的背景。

在res/drawable目录下新建两个文件：text_shape.xml和button_shape.xml，分别定义如下：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <!--填充颜色-->
    <solid android:color="@color/colorWhite" />
    <!--边框的宽度、颜色-->
    <stroke android:width="1dp" android:color="@color/colorBorder" />
    <!--圆角半径-->
    <corners android:radius="5dp"/>
</shape>
```

上面是text_shape，下面是button_shape，可以看到填充颜色用到了上一小节定义的颜色资源，下同：

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
    android:shape="rectangle">
    <solid android:color="@color/colorOrange" />
    <stroke android:width="1dp" android:color="@color/colorBorder" />
    <corners android:radius="5dp"/>
</shape>
```

2.3 界面实现

接下来实现登录的用户界面，新建SignInActivity(代码资源在初始化文档基础上重构了，这里是建在java/包名/ui/activity/下)，布局文件为activity_sign_in.xml（小编使用的1.4版的AS多了一个content为前缀名的布局文件，为了叙述方便，行文中还是称activity前缀的xml为布局控件）。

打开布局文件（activity_sign_in.xml/content_sign_in.xml），最外层改为线性布局，方向为垂直（在编辑器左下角有“Design”和“Text”标签进行设计模式的切换，对于初学者强烈建议使用“Text”，手敲代码）。

用户界面的参考代码如下，初学者建议理解代码后自己敲一遍，即使对照着再敲一遍也比复制粘贴要好，一定要习惯多敲代码！

```
<?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:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="@color/colorBackground"
    tools:context="com.geekband.demo.moran.ui.activity.SignInActivity">

    <ScrollView
        android:id="@+id/signin"
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <LinearLayout
            android:id="@+id/signin_form"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <FrameLayout
                android:layout_width="match_parent"
                android:layout_height="@dimen/logo_container_height"
                android:background="@color/colorOrange">

                <ImageView
                    android:layout_width="@dimen/logo_width"
                    android:layout_height="@dimen/logo_height"
                    android:src="@drawable/sign_logo"
                    android:layout_gravity="center|center_horizontal|center_vertical"/>
            </FrameLayout>
```

```
            <LinearLayout
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:orientation="vertical"
                android:paddingLeft="@dimen/sign_vertical_padding"
                android:paddingRight="@dimen/sign_vertical_padding">


                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="18dp"
                    android:layout_marginBottom="@dimen/sign_vertical_margin"
                    android:paddingLeft="5dp"
                    android:textSize="@dimen/sign_text_size"
                    android:textColor="@color/colorGrey"
                    android:text="@string/prompt_email"/>

                <AutoCompleteTextView
                    android:id="@+id/email"
                    android:inputType="textEmailAddress"
                    android:layout_width="match_parent"
                    android:layout_height="@dimen/sign_view_height"
                    android:paddingLeft="5dp"
                    android:maxLines="1"
                    android:singleLine="true"
                    android:background="@drawable/text_shape"/>
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_marginTop="@dimen/sign_vertical_margin"
                    android:layout_marginBottom="@dimen/sign_vertical_margin"
                    android:layout_marginLeft="5dp"
                    android:textSize="@dimen/sign_text_size"
                    android:textColor="@color/colorGrey"
                    android:text="@string/prompt_password"
                    />

                <EditText
                    android:id="@+id/password"
                    android:inputType="textPassword"
                    android:layout_width="match_parent"
                    android:layout_height="@dimen/sign_view_height"
                    android:paddingLeft="5dp"
                    android:maxLines="1"
                    android:singleLine="true"
                    android:background="@drawable/text_shape"/>


                <Button
                    android:id="@+id/sign_in_button"
                    android:layout_width="match_parent"
                    android:layout_height="@dimen/sign_view_height"
                    android:layout_marginTop="16dp"
                    android:text="@string/action_sign_in"
                    android:textStyle="bold"
                    android:textColor="@color/colorWhite"
                    android:background="@drawable/button_shape"/>
                <Button
                    android:id="@+id/sign_up_button"
                    android:layout_width="match_parent"
                    android:layout_marginTop="16dp"
                    android:layout_height="@dimen/sign_view_height"
                    android:layout_gravity="center"
                    android:gravity="center"
                    android:text="@string/sign_up"
                    android:textSize="@dimen/sign_text_size"
                    android:textColor="@color/colorGrey"
                    style="?android:attr/borderlessButtonStyle"
                    />
            </LinearLayout>
        </LinearLayout>

    </ScrollView>
</LinearLayout>
```

实现方法和上面画的图稍有不同，我们嵌套了一个ScrollView，用来支持界面滚动，由于它只能包含一个子对象，所以外面再嵌套一个垂直方向的线性布局。

2.4 添加代码逻辑

（1）获取成员对象。打开java/包名/ui/activity/SignInActivity.java文件，在SignInActivity类中添加如下代码：

```
public class SignInActivity extends AppCompatActivity {

    //定义成员变量
    private AutoCompleteTextView mEmail;
    private TextView mPassword;
    private Button mSignIn;
    private Button mSignUp;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sign_in);

        //获取对UI小组件的引用
        mEmail = (AutoCompleteTextView) findViewById(R.id.email);
        mPassword = (TextView) findViewById(R.id.password);
        mSignIn = (Button) findViewById(R.id.sign_in_button);
        mSignUp = (Button) findViewById(R.id.sign_up_button);
    }
}
```

成员变量命名时在前面加上“m”，这样在想使用时敲一个“m”，开发工具就能把所有成员列出来。（添加上述代码时需要添加import语句，以包含提供这些视图的库，可以在AS中设置自动导入。）

（2）给登录按钮注册点击事件。在onCreate方法中添加如下代码：

```
……
mSignUp = (Button) findViewById(R.id.sign_up_button);
//注册点击事件
mSignIn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

    }
});
```

（3）添加验证逻辑。几个简单的验证规则：

- 邮箱和密码不能为空
- 邮箱格式正确
- 密码长度不小于6

现在我们在onClick事件中添加上述验证逻辑：

```
//注册点击事件
mSignIn.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {

        String email = mEmail.getText().toString().trim();
        String password = mPassword.getText().toString().trim();

        if (TextUtils.isEmpty(email) || TextUtils.isEmpty(password)) {
            Toast.makeText(getApplicationContext(), "邮箱和密码不能为空", Toast.LENGTH_SHORT).show();
        } else if (email.contains("@") == false) {
            Toast.makeText(getApplicationContext(), "不是正确的邮箱格式", Toast.LENGTH_SHORT).show();
        } else if (password.length() < 6){
            Toast.makeText(getApplicationContext(),"密码长度不少于6位",Toast.LENGTH_SHORT).show();
        } else {
            Toast.makeText(getApplicationContext(),"登录成功",Toast.LENGTH_SHORT).show();
        }
    }
});
```

现在把程序部署到模拟器上运行一下吧。

在运行之前记得在AndroidManifest.xml里application节点中将这个activity设置为启动项（注意：只能有一个启动项），参考代码如下：

```
<application>
    ……
    <activity
        android:name=".ui.activity.SignInActivity"
        android:label="@string/title_activity_sign_in"
        android:theme="@style/AppTheme.NoActionBar" >
        <intent-filter>
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
        </intent-filter>
    </activity>
    ……
</application>
```

（4）尝试封装。上面的邮箱格式及密码长度验证是直接写在逻辑条件里，现在要像上面的isEmpty方法一样，把代码封装起来，修改如下：

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    ……
    //注册点击事件
    mSignIn.setOnClickListener(new View.OnClickListener() {
    @Override
        public void onClick(View v) {
            ……
            if (TextUtils.isEmpty(email) || TextUtils.isEmpty(password)) {
                Toast.makeText(getApplicationContext(), "邮箱和密码不能为空", Toast.LENGTH_SHORT).show();
            } else if (isEmailValid(email) == false) {
                Toast.makeText(getApplicationContext(), "不是正确的邮箱格式", Toast.LENGTH_SHORT).show();
            } else if (isPasswordValid(password)==false) {
                Toast.makeText(getApplicationContext(), "密码长度不少于6位", Toast.LENGTH_SHORT).show();
            } else {
                Toast.makeText(getApplicationContext(), "登录成功", Toast.LENGTH_SHORT).show();
            }
        }
    });
}

    //验证邮箱格式，通过验证则返回true
    private boolean isEmailValid(String email) {
        return email.contains("@");
    }
    //通过验证返回true
    private boolean isPasswordValid(String password) {
        return password.length() >= 6;
    }
```

这里把原来的判断逻辑封装在了isEmailValid和isPasswordValid中，邮箱格式的判断稍后使用工具类改写。

（5）改进提示信息。做如下改进：有错误的文本框里显示错误提示，并获取焦点，如果两个都出现错误，则第一个文本框获取焦点。为了方便文档阅读，将验证逻辑封装起来：

```
protected void onCreate(Bundle savedInstanceState) {
    ……
}
public void SignIn() {
    //重置错误提示
    mEmail.setError(null);
    mPassword.setError(null);

    //获取登录数据
    String email = mEmail.getText().toString().trim();
    String password = mPassword.getText().toString().trim();

    //初始化获取焦点的视图
    boolean isValid = true;
    View focusView = null;

    //邮箱验证
    if(TextUtils.isEmpty(email)){
        mEmail.setError(getString(R.string.error_empty_email));
        focusView=mEmail;
        isValid=false;
    }else if(isEmailValid(email)==false){
        mEmail.setError(getString(R.string.error_pattern_email));
        focusView=mEmail;
        isValid=false;
    }

    //密码验证
    if(TextUtils.isEmpty(password)){
        mPassword.setError(getString(R.string.error_empty_password));
        focusView=mPassword;
        isValid=false;
    }else if (isPasswordValid(password)==false){
        mPassword.setError(getString(R.string.error_length_password));
        focusView=mPassword;
        isValid=false;
    }

    if (isValid == false) {
        focusView.requestFocus();
    } else if (email.equals("test@moran.com") && password.equals("123456")) {
        Toast.makeText(getApplicationContext(),
            getString(R.string.success_signin), Toast.LENGTH_SHORT).show();
            //这里换行是为了导出的pdf文档能显示全，下同
    } else {
        Toast.makeText(getApplicationContext(),
            getString(R.string.error_invalid), Toast.LENGTH_SHORT).show();
    }
}
```

在onClick事件中调用：

```
public void onClick(View v) {
    SignIn();
}
```

其中，字符串资源添加在res/values/strings.xml中：

```
    <string name="error_empty_email">输入邮箱</string>
    <string name="error_pattern_email">邮箱格式错误</string>
    <string name="error_empty_password">密码不能为空</string>
    <string name="error_length_password">密码长度小于6</string>
    <string name="error_invalid">邮箱或密码错误</string>
    <string name="success_signin">登录成功</string>
```

重新部署运行，观察一下效果，如果两个输入框都有错误，那么实际上焦点是在第二个输入框里，怎么才能让第一个输入框获的焦点呢？

这里有一个很简单的办法，把上面的邮箱验证和密码验证的顺序交换一下，是不是就可以了呢？

这里的邮箱和密码是写死在代码里的，我们等后期完成注册模块、连上真实数据库后再进一步完善。

（6）工具类的使用。验证是否为邮箱格式常用的做法是使用正则表达式，我们这里使用已经封装好的工具类，使用时就不用自己写了，直接调用就好了。

原来的isEmailValid方法可以删掉了，不删也可以，因为我们使用的是工具类里的方法，方法名也不一样，把调用的方法改为StringUtil.isEmail：

```
        //邮箱验证
        if(TextUtils.isEmpty(email)){
            mEmail.setError(getString(R.string.error_empty_email));
            focusView=mEmail;
            isValid=false;
        }else if(StringUtil.isEmail(email)==false){
            mEmail.setError(getString(R.string.error_pattern_email));
            focusView=mEmail;
            isValid=false;
        }
```

这个工具类就放在文档的框架搭建部分，列在util目录下，打开或新建java/包名/util/StringUtil.java，代码如下：

```
public class StringUtil {

    // email
    private final static Pattern emailer
        = Pattern.compile("\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*");
    //上面换行是为了导出的pdf文档能完整显示
    /**
     * if string is empty
     *
     * @param input
     * @return boolean
     */
    public static boolean isEmpty(String input) {
        if (input == null || "".equals(input))
            return true;

        for (int i = 0; i < input.length(); i++) {
            char c = input.charAt(i);
            if (c != ' ' && c != '\t' && c != '\r' && c != '\n') {
                return false;
            }
        }
        return true;
    }

    /**
     * if email is valid
     *
     * @param email
     * @return
     */
    public static boolean isEmail(String email) {
        if (email == null || email.trim().length() == 0)
            return false;
        return emailer.matcher(email).matches();
    }

    /**
     * 半角转换为全角
     *
     * @param input
     * @return
     */
    public static String toDBC(String input) {
        char[] c = input.toCharArray();
        for (int i = 0; i < c.length; i++) {
            if (c[i] == 12288) {
                c[i] = (char) 32;
                continue;
            }
            if (c[i] > 65280 && c[i] < 65375)
                c[i] = (char) (c[i] - 65248);
```

```
        }
        return new String(c);
    }

    /**
     * encrypt phone number
     * @param phone
     * @return
     */
    public static String encryptPhoneNumber(String phone) {
        if(phone == null) {
            return "*****";
        }
        if(phone.length() < 11) {
            return phone;
        }
        return phone.substring(0, 3) + "*****" + phone.substring(8, 11);
    }

    /**
     * convert user credit log's address
     * @param address
     * @return
     */
    public static String convertUserDeliveryAddress(String address) {
        String[] info = address.split("#");
        StringBuffer text = new StringBuffer();
        text
                .append(info[1]) // province
                .append(info[2]) // city
                .append(info[3]) // zone
                .append(info[4]) // street
                .append("\n")
                .append(info[0]) // name
                .append(" ")
                .append(info[6]) // phone
        ;

        return text.toString();
    }
}
```

注意需要import依赖库。至此，登录模块就暂时告一段落了，其他功能待后期逐步完善。