

Android 应用软件设计

E 2 Intent and Intent Filter

学号：SA18225402

姓名：吴文韬

报告撰写时间：2018/09/18

一、 主要目标

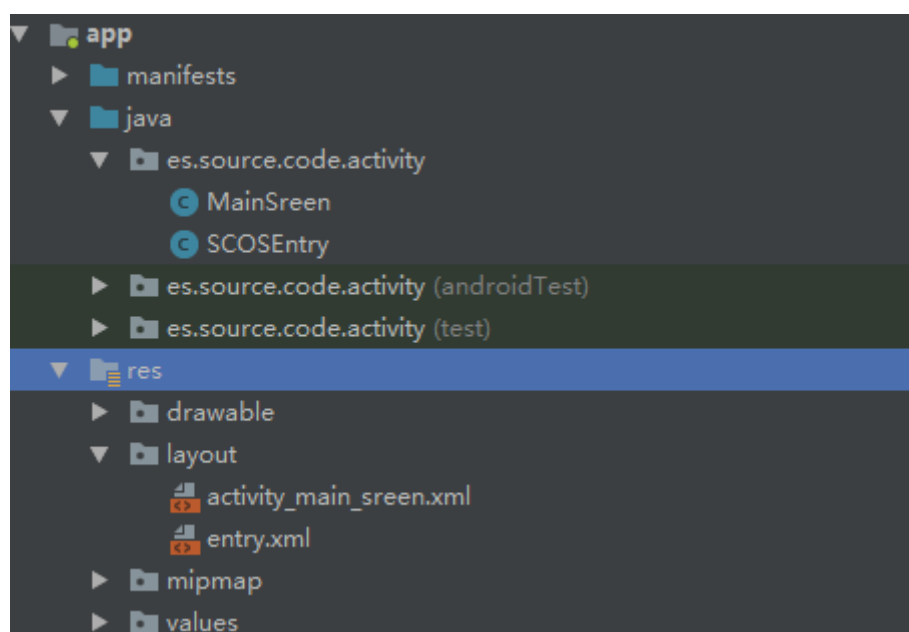
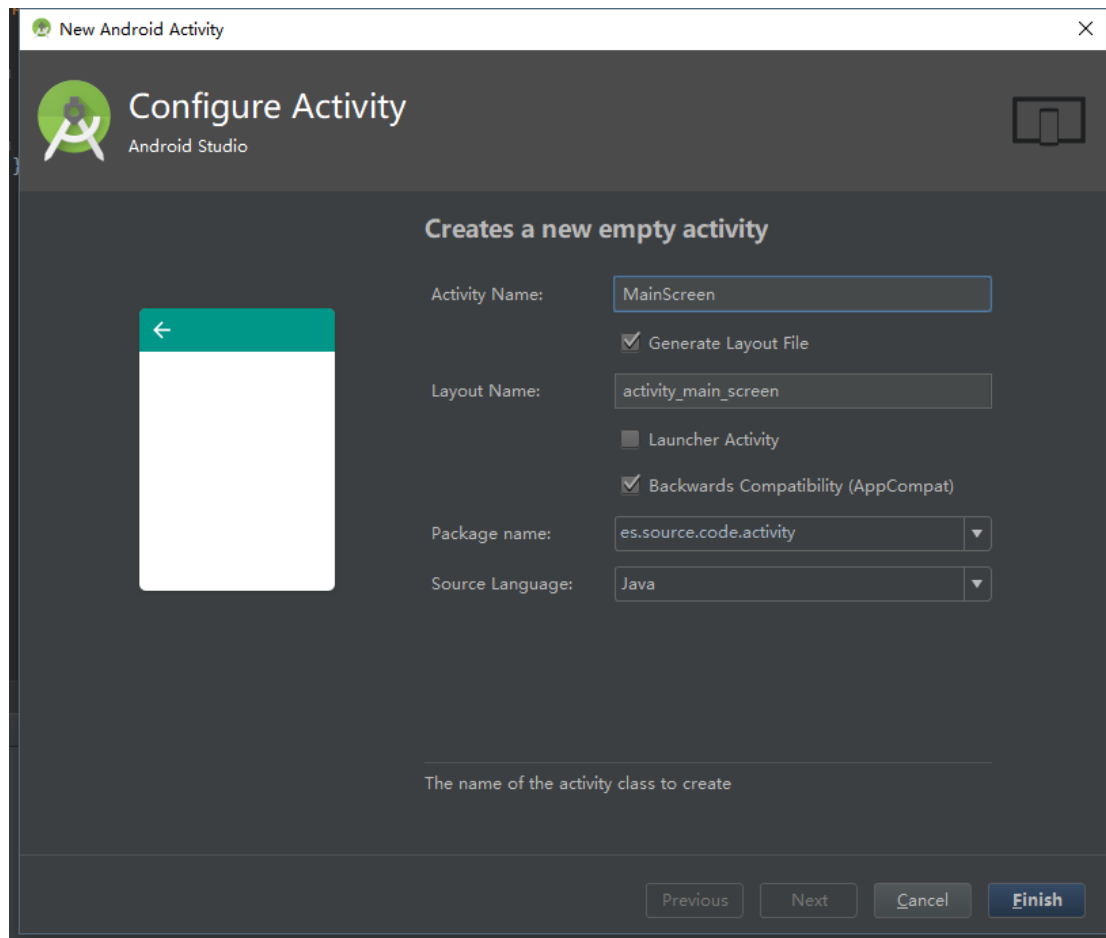
1. 在 E1 的 SCOS 工程包 `es.source.code.activity` 下定义一个新 Activity 类 `MainScreen.java`
2. 在 `MainScreen` 的屏幕布局中设计 SCOS 系统的主功能导航，导航项含：点菜、查看订单、登录/注册、系统帮助。要求导航项有图标和文字，并且可点击。
3. 在 `AndroidManifest.xml` 中添加 `MainScreen` 并定义其 `IntentFilter` 属性，添加一个 `Action` 值为 `"scos.intent.action.SCOSMAIN"` 和一个 `Category` 值为 `"scos.intent.category.SCOSLAUNCHER"`。
4. 修改 E1 中 `SCOSEntry.java` 类的代码，实现当用户在 `SCOSEntry` 屏幕上水平向左滑动时，从 `SCOSEntry` 屏幕跳转到 `MainScreen` 屏幕，并使用 `Intent` 向 `MainScreen` 类传递一个 `String` 数据值 `"FromEntry"`。
5. 修改 `MainScreen.java` 类的代码，当屏幕跳转至 `MainScreen` 时，获取 `SCOSEntry` 通过 `Intent` 传递过来的 `String` 值，并作判断是否和 `"FromEntry"` 相等；如果相等，则正常显示当前屏幕；如果不相等，则隐藏导航项：点菜，查看订单。测试该功能是否正确运行，如不能，请根据调试信息，修改代码使之能正确运行。
6. 在 SCOS 工程 `AndroidManifest.xml` 中定义 `Permission` 值为 `"scos.permission.ACCESSSCOS"`，并将该 `Permission` 属性 `protection level` 设置为 `"dangerous"`。
7. 修改 `AndroidManifest.xml` 中 `MainScreen` 的属性 `android:permission`，并将值设为 `"scos.permission.ACCESSSCOS"`。再次测试从 `SCOSEntry` 屏幕跳转到 `MainScreen` 屏幕，观察是否能正确运行。如不能，请根据调试信息，修改代码使之能正确运行。
8. 在 SCOS 工程包 `es.source.code.activity` 下定义一个新 Activity 类 `LoginOrRegister.java`
9. 在 `LoginOrRegister` 布局中添加“登录名”输入框和“登录密码”输入框，将“登录密码”输入框类型设置为 `Password`，添加按钮“登录”、

“返回”。

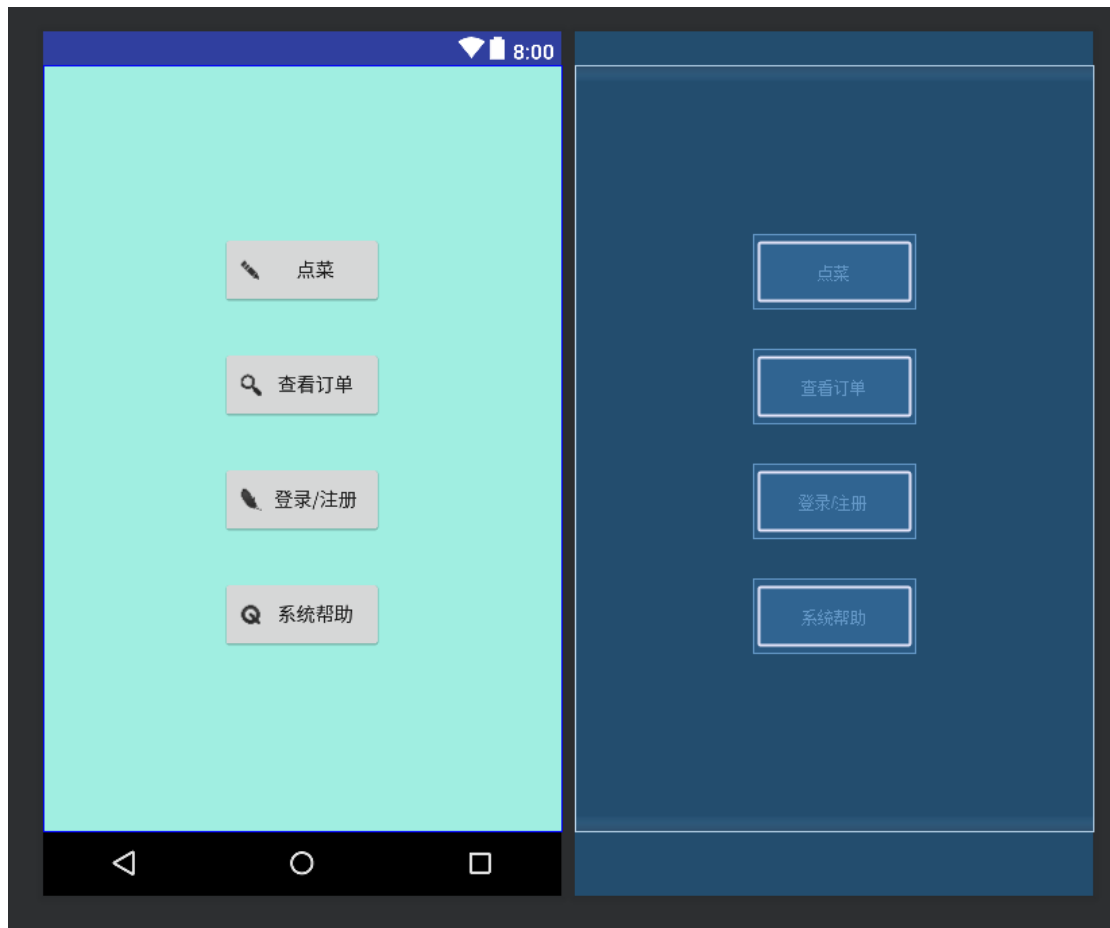
10. 当用户在 LoginOrRegister 屏幕中，点击登录按钮时，使用 ProgressBar 显示登录进度，2 秒钟后进度条消失；同时，使用正则表达式验证“登录名”和“登录密码”是否符合以下规则：不为空，只包含英文大小写和数字。当输入内容不符合规则时，则使用 setError 方法在当前输入框出提示错误信息“输入内容不符合规则”。当符合规则后，屏幕跳转至 MainScreen，并向 MainScreen 类传递一个数据 String 值为“LoginSuccess”。
11. 当用户在 LoginOrRegister 屏幕中，点击返回按钮时，屏幕跳转至 MainScreen，并向 MainScreen 类传递一个数据 String 值为“Return”。
12. 修改 MainScreen.java 代码，判断由 LoginOrRegister 传回的 String 数据值。如果返回数据为“LoginSuccess”，则检查“点菜”和“查看订单”是否为隐藏状态，如果为隐藏，则设为显示。
13. 新建一个工程为 TestSCOS，在该工程中添加 Activity 类 TestMain.java，在 TestMain 屏幕中添加一个按钮“SCOS”，当点击此按钮时，屏幕跳转至 SCOS 的 MainScreen 屏幕。请根据调试信息，解决程序错误，达到正确运行此功能。
14. 请查阅资料总结 android:protectionlevel 的不同类型，并说明如何使用。
15. 请查阅资料总结 IntentFilter 如何测试 Action、Category、Data，并说明 IntentFilter 用法。

二、 实现和证明

1. 新建 MainScreen. java, 创建结果如下所示



2. 在 activity_main_screen.xml 中设置为 LinearLayout 布局，并且添加 4 个按钮，并且相关的 string,color,dimen 属性分别存放在资源文件 colors.xml,dimen.xml,strings.xml 文件中。设置后界面如下图所示。



3. 前往 AndroidManifest.xml 文件修改，添加如下内容

```
<activity android:name=".MainScreen"
    android:exported="false">
    <intent-filter>
        <action android:name="scos.intent.action.SCOSMAIN" />
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="scos.intent.category.SCOSLAUNCHER" />
    </intent-filter>
</activity>
```

4. 将类 SCOSEntry 实现接口 View.OnTouchListener 和接口 GestureDetector.OnGestureListener，在类中添加当前布局的私有成员和

GestureDetector 的私有成员。并且在重写的类方法中重写 OnFling 方法来判断左滑，在左滑判定生效以后新建一个 Intent 对象实现将 Activity: MainScreen 启动并传递字符串 FromEntry(其中 MAINSCREEN_ACTION 为存储在 MainScreen 中的 final static 字符串即为 MainScreen 的 Action 属性 scos.intent.action.MAIN); onTouch 方法将 event 转移到更高级的 GestureDetector 中处理。并且把布局监听设置为 this，具体代码如下所示。

```
@Override
public boolean onFling(MotionEvent e1, MotionEvent e2, float velocityX, float velocityY) {
    //检测到在屏幕上的滑动，根据相对位置来判断滑动的方向
    float offsetX = e1.getX() - e2.getX();
    float offsetY = e1.getY() - e2.getY();
    if(Math.abs(offsetX) > Math.abs(offsetY)) { //在X上的偏移大于Y上的偏移时，为左右滑动
        if(offsetX > 0) { //此时为向右滑动
            Intent intent = new Intent(MainScreen.MAINSCREEN_ACTION);
            intent.putExtra( name: "data", value: "FromEntry");
            startActivity(intent);
        }
    }

    return true;
}

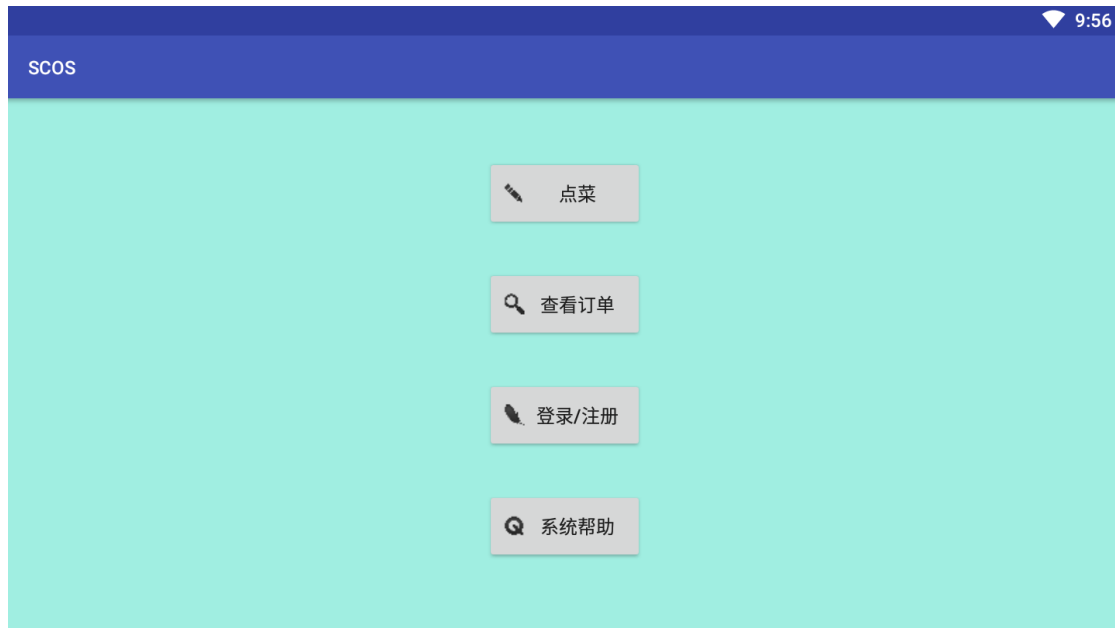
@Override
public boolean onTouch(View v, MotionEvent event) {
    return mGestureDetector.onTouchEvent(event);
}
```

5. 在 MainScreen.java 中使用 getIntent() 来获取传入的字符串，判断传入的字符串是否为我们想要的字符串 FromEntry, 如果时则使用 setVisibility() 方法将按钮设置为 View.GONE。代码如下：

```
button_order = findViewById(R.id.order);
button_watch_order = findViewById(R.id.watch_order);
button_login = findViewById(R.id.login);
button_help = findViewById(R.id.help);

Intent intent = getIntent();
String get_from_intent = intent.getStringExtra( name: "data");
if(!get_from_intent.equals("FromEntry"))
{
    button_order.setVisibility(View.GONE);
    button_watch_order.setVisibility(View.GONE);
}
```

当传入的时我们想要的对象时结果如下图：



当不是我们想要的字符串时，点菜和查看订单不可见：

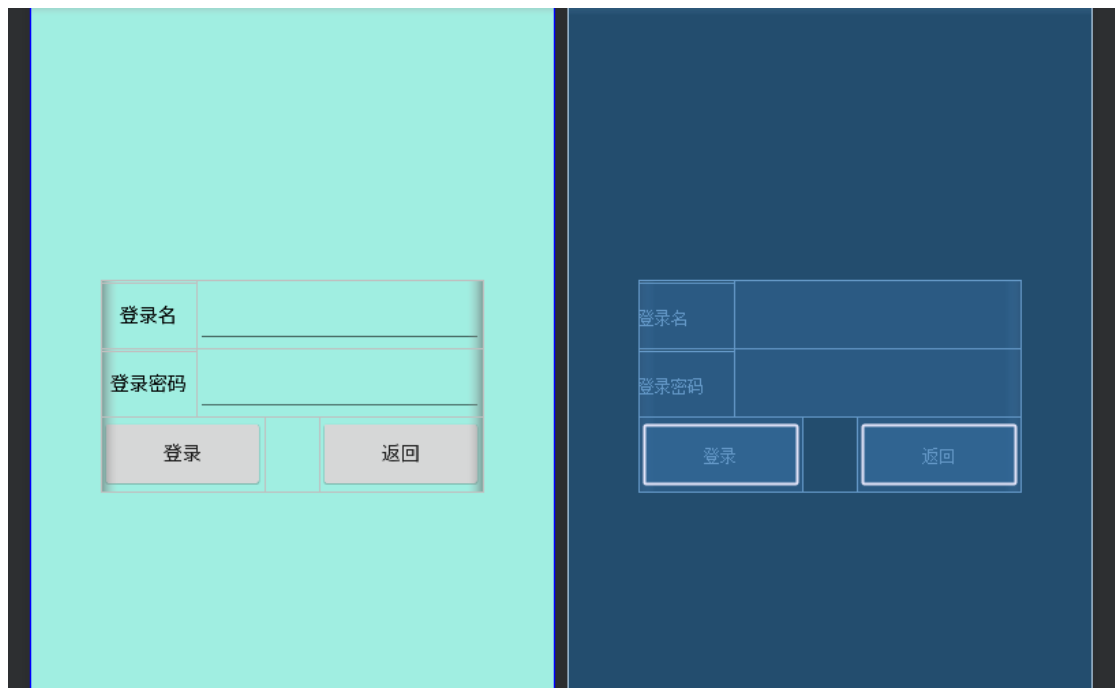
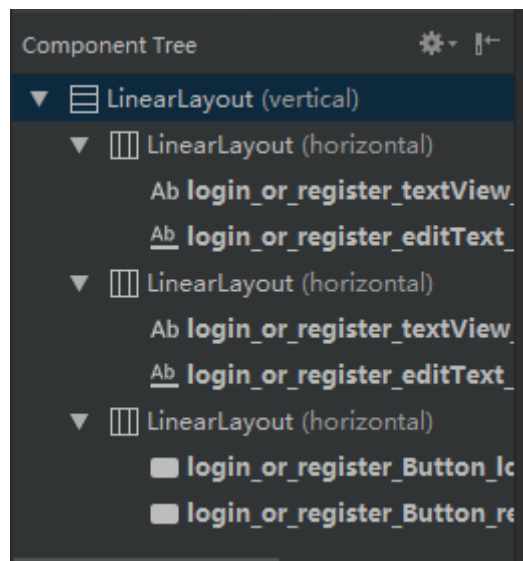


6、7. 在 AndroidManifest.xml 中依照要求添加 permission 条目，如下图所
示。在 MainScreen 中添加 permission 为我们第六步添加的权限名，此时测试
程序仍然可以正常运行。

```
<permission android:name="scos.permission.ACCESSSCOS"  
            android:protectionLevel="dangerous"/>
```

```
<permission android:name="scos.permission.ACCESSSCOS"
    android:protectionLevel="dangerous"
/>
```

8、9. 在 es.source.code.activity 下定义一个叫 LoginOrRegister 的新的 Activity 类。在 xml 设计中使用了 LinearLayout 布局，添加了两个 TextView、两个 EditText 和两个 Button。如下图所示。



10. 为了实现登录进度条，此处使用了 ProgressDialog 来实现，并且使用了 Handler 来实现延迟 2s。代码如下所示。

```
@Override
public void onClick(View v) {
    if(v.getId() == login.getId()){ //当登录按钮被按下
        //圆形转动的进度条
        mProgressDialog.setProgressStyle(ProgressDialog.STYLE_SPINNER);

        //设置是否可以通过点击Back键取消
        mProgressDialog.setCancelable(false);

        //设置是否可以点击Dialog外取消进度条
        mProgressDialog.setCanceledOnTouchOutside(false);

        //设置title的图标
        mProgressDialog.setIcon(R.drawable.launch);

        //设置标题
        mProgressDialog.setTitle("登录中");

        //dismiss监听
        mProgressDialog.setOnDismissListener((dialog) -> {
            Toast.makeText(context, LoginOrRegister.this, text: "登录成功", Toast.LENGTH_SHORT).show();
        });

        //监听cancel实践
        mProgressDialog.setOnCancelListener((dialog) -> {
            Toast.makeText(context, LoginOrRegister.this, text: "取消登录", Toast.LENGTH_SHORT).show();
        });
        mProgressDialog.setMessage("登录中，请等待");
        mProgressDialog.show();
        mHandler.postDelayed(mDialog_key, delayMillis: 2000);
    }else if(v.getId() == back.getId()){
        //当返回按钮被按下时
        mIntent.putExtra(name: "data", value: "Return");
        setResult(Activity.RESULT_CANCELED, mIntent);
        finish();
    }
}
```

```
//使用Handler延迟处理ProgressDialog
private Handler mHandler = new Handler();
private Runnable mDialog_key = () -> {
    mProgressDialog.dismiss();

    //登录成功 返回LoginSuccess字符串
    mIntent.putExtra(name: "data", value: "LoginSuccess");
    setResult(Activity.RESULT_OK, mIntent);
    finish();
};
```

而在登录界面实现输入时，使用了正则表达式来判断用户名是否符合要求(只包含数字和字母且不为空)，正则表达式为“^[0-9A-Za-z]+\$”。判断是否满足另写成一个方法，如下所示。

```
public boolean judgeRegex(String str){
    Pattern p = Pattern.compile("^[0-9A-Za-z]+$"); //正则表达式

    Matcher matcher = p.matcher(str);

    return matcher.matches();
}
```

在实现对 EditText 的监听上，使用了 addTextChangedListener 方法为两个编辑栏添加了 TextWatcher 监听器。在编辑栏发生变化时对编辑栏中的内容进行正则表达式的判断，根据判断结果来使能登录按钮。

```
user_name.addTextChangedListener(new TextWatcher() { //为username添加一个监听器
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        if(!judgeRegex(user_name.getText().toString())){ //未通过正则表达式判断
            login.setEnabled(false);
            password.setEnabled(false);
            user_name.setError("输入内容不符合规则");
        }else if(user_name.getText().toString().trim().length() == 0){ //用户名为空
            login.setEnabled(false);
            password.setEnabled(false);
        }else if(password.getText().toString().length() !=0){ //当用户名满足条件且密码不为空
            password.setEnabled(true);
            login.setEnabled(true);
        }else{ //用户名满足要求且密码为空
            password.setEnabled(true);
        }
    }


    @Override
    public void afterTextChanged(Editable s) {
    }
});
```

```
password.addTextChangedListener(new TextWatcher() { //给密码设置监听
    @Override
    public void beforeTextChanged(CharSequence s, int start, int count, int after) {
    }

    @Override
    public void onTextChanged(CharSequence s, int start, int before, int count) {
        if(password.getText().toString().length() <=0){ //密码为空
            login.setEnabled(false);
        }else{ //密码不为空
            login.setEnabled(true);
        }
    }

    @Override
    public void afterTextChanged(Editable s) {
    }
});
```

对结果进行测试时，当用户名为空时密码栏和登录按钮都无法正常使用，且当用户名输入不和规则时会提示输入内容不和规则。当用户名输入正确时，才可以对密码栏进行输入，在密码和用户名都输入完毕后，登录按钮才会亮起。

登录名 -2 

登录密码

输入内容不符合规则

登录 返回

登录名 123

登录密码 ...

登录 返回

此时点击登录按钮进入登录界面。登录完成后提示登录成功。

 登录中

 登录中，请等待

登录 返回



11. 当用户在 LoginOrRegister 屏幕上点击返回按钮时, 屏幕返回值 MainScreen。

```
}else if(v.getId() == back.getId()){  
    //当返回按钮被按下时  
    mIntent.putExtra( name: "data", value: "Return");  
    setResult(Activity.RESULT_CANCELED, mIntent);  
    finish();  
}
```



12. 此时在 MainScreen 的代码中添加判断返回的数据是否为 LoginSuccess, 根据返回的数据来显示点菜和查看订单。

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode == 0){
        if(resultCode == Activity.RESULT_OK){
            String fromLogin = (data != null ? data.getStringExtra( name: "data") : "null");
            if(fromLogin.equals("LoginSuccess")){
                checkAndSetIsVisible();
            }
        }
    }
}

private void checkAndSetIsVisible() //检查点菜和查看订单按钮时否被隐藏, 如果被隐藏则显示
{
    if(button_order.getVisibility() != View.VISIBLE || button_watch_order.getVisibility() != View.VISIBLE)
    {
        button_order.setVisibility(View.VISIBLE);
        button_watch_order.setVisibility(View.VISIBLE);
    }
}
```

13. 当在新创建的工程中调用至 mainScreen 屏幕时，提示错误。

```
requires scos.permission.ACCESSSCOS
```

此时需要在 AndroidManifest.xml 文件中添加需要权限的声明。

```
<uses-permission android:name="scos.permission.ACCESSSCOS" />
```

这样就可以在 TestSCOS 中实现跳转至 SCOS 的 mainScreen 了。

```
@Override
public void onClick(View v) {
    if(v.getId() == TurnToSCOS.getId()){
        Intent intent = new Intent( action: "scos.intent.action.SCOSMAIN");
        startActivity(intent);
    }
}
```

14. Android protection level 分为四个级别：

Normal, Dangerous, Signature, SignatureOrSystem. 其中：

Normal:这是最低风险的权限，如果应用声明了此权限，系统会在应用安装时直接默认该应用拥有此权限，而不会要求用户明确批准。

Dangerous:这是风险较高的权限，这种权限可以使应用程序访问私有用户的数据或者对用户产生负面影响。这种权限的引入会存在一定的风险，所以系统可能不会自动将其授予请求的应用程序。例如可能会在用户继续前需要进行确认。

Signature: 这种权限仅当请求的应用程序和声明权限的应用程序具有相同的证书签名时，系统才会自动授予或者要求用户批准。

SignatureOrSystem:应当避免使用此选项，因为这是在供应商将应用程序内置到系统影响中需要使用的系统应用权限。

15. Intent-filter 用来注册 Activity、Service、Broadcast Receiver 具有在某种数据上处理一个动作的能力。

Action 过滤：在 intent-filter 元素中可以包括子元素 action，一个 intent-filter 元素至少应当包含一个 action，否则任何 intent 请求都不能和该 intent-filter 匹配，如果 intent 请求的 action 和 intent-filter 中的某一个 action 匹配，那么该 Intent 就通过了 intent-filter 的动作测试。

Category 过滤：只有当 Intent 请求总所有 category 与组件中某一个 Intent-filter 的 category 王权匹配是，才会让该 intent 请求通过测试。如果 intent-filter 中有多余的 category 并不会导致匹配失败。一个没有指定任何类别测试的 intent-filter 仅仅只会匹配没有设置类别的 intent 请求。

Data 过滤：data 元素指定了希望接收的 intent 请求的数据 URI 和数据类型，URI 被分成三部分来进行匹配：scheme, authority 和 path。其中用 setData() 设定的 Intent 请求的 URI 数据类型和 schema 必须与 intent-filter 中所指定的一直，若 Intent-filter 中海指定了 authority 或 path，他们也要相匹配才会通过测试。

三、 结论

本次作业主要实现了主界面和登录界面的设计以及通过其他应用来进入本软件的主界面。

本次作业的主要问题在于对各个监听器和软件接口 API 的了解程度不够例如：如何通过 intent 传送数据和接受返回的数据、如何使用 ProgressBar 等问题。通过查阅资料和 android 的 API 文档，对本次作业所需要使用的一些软件接口有了一定的了解。还有就是在通过其他软件进行调用时，需要在 AndroidManifest.xml 文件中声明需要的权限，不然时无法访问 SCOS 的界面的。

四、 参考资料

1. [android 中 button 上如何显示图片和文字](#)
2. [Android Studio 之多个 Activity 的滑动切换](#)
3. [AndroidManifest.xml 文件详解 \(permission\)](#)
4. [Android 学习笔记之 AndroidManifest.xml 文件解析](#)
5. [Android 控件--ProgressBar](#)
6. [android 自定义 permission android:protectionLevel 说明](#)
7. [Android 开发--Intent-filter 属性详解](#)