Android 应用软件设计

E 2　Intent and Intent Filter

学号：SA17225268
姓名：彭明

报告撰写时间：2017/10/02

1.主题概述

本次主题是围绕 activity 之间启动方式 Intent 来展开的，内容包括了 Intent 启动的两种方式，显示 Intent 和隐式 Intent。同时还包括了通过 Intent 传值，获取被启动的 activity 返回的值，不同应用间通过 Intent filter 来过滤启动另一个应用中的 activity。

2.假设

在同一个应用中不同 activity 之间的跳转，我们需要指定跳转的 activity 类名，这是一种显示 Intent 启动方式。不同应用之间，我们需要使用同一个 action 和 category，来达到 Intent filter 隐式启动的目的。

3.实现或证明

为了实现当用户在 SCOSEntry 屏幕上水平向左滑动跳转到 MainScreen 并传值的功能，在 SCOSEntry.java 中添加了如下代码：

```
    int lastX = 0;
    public final static String EXTRA_MESSAGE =
"es.source.code.activity.MESSAGE";
    @Override
    public boolean onTouchEvent(MotionEvent event) {
        int x = (int) event.getX();
        switch (event.getAction()) {
            case MotionEvent.ACTION_DOWN:
                lastX = x;
                break;
            case MotionEvent.ACTION_UP:
                int offsetX = x - lastX;
                if (offsetX < -5) {
                    Intent intent = new Intent(this, MainScreen.class);
                    intent.putExtra(EXTRA_MESSAGE, "FromEntry");
                    startActivity(intent);
                }
                break;
        }
        return super.onTouchEvent(event);
    }
```

AndroidManifest.xml 中添加的 Permission 和 MainScreen 的定义如下：

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

```
        <activity
            android:name=".MainScreen"
            android:label="@string/title_activity_main_screen"
            android:permission="scos.permission.ACCESSSCOS">
            <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>
```

MainScreen 的布局如下：

MainScreen.java 中采用了 BottomNavigationView：

```
        navigation.setOnNavigationItemSelectedListener(mOnNavigationI
temSelectedListener);
```

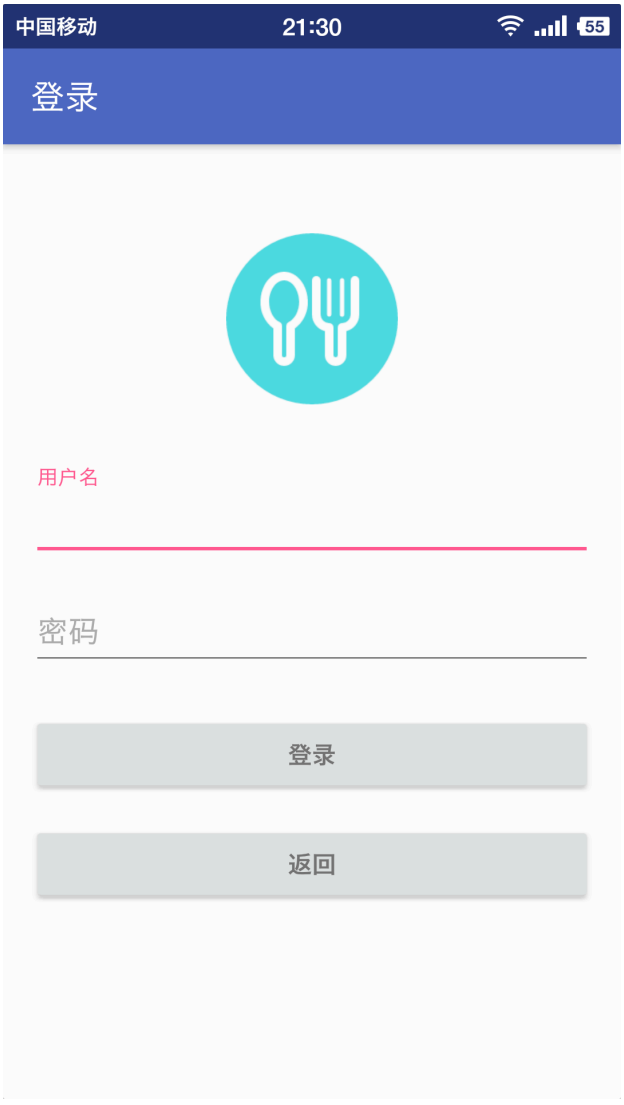
```
        private BottomNavigationView.OnNavigationItemSelectedListener
mOnNavigationItemSelectedListener
        = new BottomNavigationView.OnNavigationItemSelectedListener() {
        @Override
        public boolean onNavigationItemSelected(@NonNull MenuItem item){
            switch (item.getItemId()) {
                case R.id.navigation_ordering:
                    mTextMessage.setText(R.string.title_ordering);
                    return true;
                case R.id.navigation_myorder:
                    mTextMessage.setText(R.string.title_myorder);
                    return true;
                case R.id.navigation_login:
                    //跳转到登录界面
                    Intent loginIntent = new Intent(MainScreen.this,
LoginOrRegister.class);
                    startActivityForResult(loginIntent, REQUESTCODE);
                    return true;
                case R.id.navigation_help:
                    mTextMessage.setText(R.string.title_help);
                    return true;
            }
            return false;
        }
    };
```

在 MainScreen.java 的 onCreate 方法中添加了对 SCOSEntry 传过来值的处理：

```
        try{
            String message =
getIntent().getStringExtra(SCOSEntry.EXTRA_MESSAGE);
```

```
        if (!message.equals("FromEntry")) {
         navigation.getMenu().removeItem(R.id.navigation_ordering);
         navigation.getMenu().removeItem(R.id.navigation_myorder);
        }else{
            //disableShiftMode 函数是为了消除当 menuitem>3 时产生的动画效果
            disableShiftMode(navigation);
        }
    }
    catch (Exception e){
        disableShiftMode(navigation);
    }
```

LoginOrRegister 布局：



在 LoginOrRegister.java 中添加了对登录和返回的处理：

```
        mEmailSignInButton.setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View view) {
                attemptLogin();
            }
        });
```

```
    private void attemptLogin() {
        // Reset errors.
        mEmailView.setError(null);
        mPasswordView.setError(null);
        // Store values at the time of the login attempt.
        String username = mEmailView.getText().toString();
        String password = mPasswordView.getText().toString();

        boolean cancel = false;
        View focusView = null;
        // Check for a valid password.
        if (TextUtils.isEmpty(password))
{           mPasswordView.setError(getString(R.string.error_password_re
quired));
            focusView = mPasswordView;
            cancel = true;
        } else if (!isValid(password)) {
            mPasswordView.setError(getString(R.string.error_invalid));
            focusView = mPasswordView;
            cancel = true;
        }

        // Check for a valid username.
        if (TextUtils.isEmpty(username))
{           mEmailView.setError(getString(R.string.error_username_requi
red));
            focusView = mEmailView;
```

```
            cancel = true;
        } else if (!isValid(username)) {
            mEmailView.setError(getString(R.string.error_invalid));
            focusView = mEmailView;
            cancel = true;
        }

        if (cancel) {
            focusView.requestFocus();
        } else {
            mProgressView.setVisibility(View.VISIBLE);
            //2 秒钟后进度条消失
            mHandler.postDelayed(new Runnable() {
                @Override
                public void run() {
                    mProgressView.setVisibility(View.GONE);
                    returnIntent.putExtra(RETURN_TAG, "LoginSuccess");
                    setResult(2, returnIntent);
                    finish();
                }
            }, 2000);
        }
    }
```

```
    private boolean isValid(String word) {
        Pattern p = Pattern.compile("^[A-Za-z0-9]+",
Pattern.CASE_INSENSITIVE);
        Matcher m = p.matcher(word);
        return m.matches();
    }
```

对返回的处理：

```
        mReturnButton.setOnClickListener(new OnClickListener() {
            @Override
```

```
            public void onClick(View v) {
                returnIntent.putExtra(RETURN_TAG, "Return");
                setResult(2, returnIntent);
                finish();
            }
        });
```

在 MainScreen.java 中实现 onActivityResult 方法来对登录页面返回的数据进行处理：

```
    @Override
    protected void onActivityResult(int requestCode, int resultCode,
Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(resultCode == 2){
            if(requestCode == REQUESTCODE){
                String returnMessage =
data.getStringExtra(LoginOrRegister.RETURN_TAG);
                if(returnMessage.equals("LoginSuccess")){
                    if(navigation.getMenu().size() == 2){
                        navigation.getMenu().add(0,
R.id.navigation_ordering, 0, R.string.title_ordering)
                                .setIcon(R.drawable.ic_restaurant_black_24
dp);
                        navigation.getMenu().add(0,
R.id.navigation_myorder, 1, R.string.title_myorder)
                                .setIcon(R.drawable.ic_menu_black_24dp);
                        disableShiftMode(navigation);
                    }
                }
            }
        }
    }
```

在工程 TestSCOS 的 AndroidManifest.xml 中添加 use-permission：

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

在 TestMain.java 中实现隐式 Intent 跳转到 SCOS 的 MainScreen 界面：

```
mButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent();
        intent.setAction("scos.intent.action.SCOSMAIN");
        intent.addCategory("scos.intent.category.SCOSLAUNCHER");
        startActivity(intent);
    }
});
```

4.结论

android:protectionLevel 有四种类型：

(1)Normal

权限被声明为 Normal 级别，任何应用都可以申请，在安装应用时，不会直接提示给用户，点击全部才会展示。

(2)Dangerous

权限被声明为 Dangerous 级别，任何应用都可以申请，在安装应用时，会直接提示给用户。

(3)Signature

权限被声明为 Signature 级别，只有和该 apk(定义了这个权限的 apk)用相同的私钥签名的应用才可以申请该权限。

frameworks/base/core/res/AndroidManifest.xml 声明的权限为 Signature 级别，那么只有 Android 官方使用相同私钥签名的应用才可以申请该权限。

(4)SignatureOrSystem

权限被声明为 SignatureOrSystem 级别，有两种应用可以申请该权限。

1)和该 apk(定义了这个权限的 apk)用相同的私钥签名的应用

2)在/system/app 目录下的应用

IntentFilter 是用来匹配隐式 Intent 的，当对象被 IntentFilter 进行匹配测试时，会考虑到 Action、Category、Data 三个方面。

Action 的匹配规则：

Intent 中的 Action 必须能够和 Activity 过滤规则中的 Action 匹配.(这里的匹配是完全相等).一个过滤规则中有多个 action,那么只要 Intent 中的 action 能够和 Activity 过滤规则中的任何一个 action 相同即可匹配成功。

Category 的匹配规则：

如果 Intent 中的存在 category 那么所有的 category 都必须和 Activity 过滤规则中的 category 相同。才能和这个 Activity 匹配。Intent 中的 category 数量可能少于 Activity 中配置的 category 数量，但是 Intent 中的这 category 必须和 Activity 中配置的 category 相同才能匹配。

Data 的匹配规则：

data 由两部分组成，mineType 和 URI。mineType 指媒体类型，URI 是统一资源标识符，当两个都匹配时才完成匹配。

5.参考文献

1. Android 开发文档中关于 BottomNavigationView 的详细介绍 https://developer.android.google.cn/reference/android/support/design/widget/BottomNavigationView.html

2. startActivityForResult 用法 http://www.jianshu.com/p/75eccd29c229

3. Android 开发文档中介绍的在 AndroidManifest.xml 里使用系统权限 https://developer.android.google.cn/guide/topics/security/permissions.html#custom-recommendations

4. 关闭 BottomNavigationView 的滑动动画 http://www.jianshu.com/p/eada0f16afd9

5. Intent Filter 在 Action 、 Category 、 Data 三个方面上的匹配规则 http://blog.csdn.net/mynameishuangshuai/article/details/51673273