# 中山大学移动信息工程学院本科生实验报告

（2017年秋季学期）

课程名称：移动应用开发 任课教师：郑贵锋

|  |  |  |  |
| --- | --- | --- | --- |
| 年级 | 15M1 | 专业（方向） | 移动互联网 |
| 学号 | 15352008 | 姓名 | 蔡荣裕 |
| 电话 | 13727021990 | Email | 897389207@qq.com |
| 开始日期 | 2017/9/23 | 完成日期 | 2017/9/25 |

## 实验题目

事件处理

## 实验目的

1. 了解 Android 编程基础
2. 熟悉ImageView、Button、RadioButton 等基本控件，能够处理这些控件的基本事件
3. 学会弹出基本的对话框，能够定制对话框中的内容，能对确定和取消按钮的事件做处理

## 实验内容

实现一个Android应用，要求如下：

|  |  |  |
| --- | --- | --- |
| 1. 该界面为应用启动后看到的第一个界面   (如右图1)   1. 输入学号和密码的控件要求用 TextInputLayout 实现 2. 点击图片，弹出对话框如图2  * 点击“拍摄”选项，弹出 Toast 信息“您选择了[拍摄]”； * 点击“从相册选择”选项，弹出 Toast 信息“您选择了[从相册选择]； * 点击“取消”按钮，弹出 Toast 信息“您选择了[取消]”。  1. 切换 RadioButton 的选项，弹出 Snackbar 提示“您选择了 xx”；例如从选项“学生”切换到选项“教职工”，则提示“您选择了教职工”；点击 Snackbar 上的“确定”按钮，则弹出 Toast 信息“Snackbar 的确定按钮被点击了” 2. 点击登录按钮   依次判断学号是否为空，密码是否为空，用户名和密码是否正确（正确的学号和密码分别为“123456”，“6666”）；不正确则给出错误信息，如学号和密码都正确则提示“登陆成功”，如右图：   1. 点击注册按钮   如果切换选项时，RadioButton 选中的是“学生”，那么弹出 Snackbar 信息“学生注册功能尚未启用”，如果选中的是“教职工”，那么弹出 Toast 信息“教职工注册功能尚未启用”。如右图： |  |  |
| 图1 | 图2 |



## 课堂实验结果

1. 实验截图：
   * 1. 界面效果如下：



* + 1. 虚拟机上效果如下：



|  |  |  |
| --- | --- | --- |
| 1. 点击图片，弹出对话框，此处的拍摄和从相册选择都是额外实现的功能。 | | |
|  |  |  |
|  |  |  |
| 1. 切换 RadioButton 的选项，弹出 Snackbar 提示“您选择了 xx” | | |
|  |  |  |
| 1. 点击按钮效果 | | |
|  |  |  |
|  |  |  |

1. 关键步骤：
2. 本次实验需要使用到TextInputLayout所以需要在修改先对应的 build.gradle 文件中的配置

|  |
| --- |
| apply plugin: 'com.android.application'  android {  compileSdkVersion 26 //版本对应  buildToolsVersion '26.0.0' //版本对应  defaultConfig {  applicationId "cn.chonor.lab2"  minSdkVersion 19  targetSdkVersion 26 //版本对应  versionCode 1  versionName "1.0"  testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"  }  buildTypes {  release {  minifyEnabled false  proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'  }  } }  dependencies {  compile fileTree(dir: 'libs', include: ['\*.jar'])  androidTestCompile('com.android.support.test.espresso:espresso-core:2.2.2', {  exclude group: 'com.android.support', module: 'support-annotations'  })  compile 'com.android.support:appcompat-v7:26.0.0-alpha1' //版本对应  compile 'com.android.support:design:26.0.0-alpha1' //版本对应  compile 'com.android.support.constraint:constraint-layout:1.0.2' //TextInputLayout控件  testCompile 'junit:junit:4.12' } |

1. 删掉之前的EidtText控件和对应的TextView控件，替换为TextInputLayout控件，代码修改部分如下(一个TextInputLayout控件)：

|  |
| --- |
| <android.support.design.widget.TextInputLayout  android:id="@+id/textInputLayout"  android:layout\_width="0dp"  android:layout\_height="wrap\_content"  android:layout\_marginEnd="20dp"  android:layout\_marginLeft="20dp"  android:layout\_marginRight="20dp"  android:layout\_marginStart="20dp"  android:layout\_marginTop="20dp"  app:layout\_constraintBottom\_toBottomOf="parent"  app:layout\_constraintHorizontal\_bias="0.0"  app:layout\_constraintLeft\_toLeftOf="parent"  app:layout\_constraintRight\_toRightOf="parent"  app:layout\_constraintTop\_toBottomOf="@+id/imageView"  app:layout\_constraintVertical\_bias="0.0"  >   <EditText  android:id="@+id/textid"  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:hint="@string/user\_name\_hit"  android:inputType="number" /> </android.support.design.widget.TextInputLayout> |

1. 界面修改完毕之后开始添加事件处理代码。
   1. ImageView点击事件

* 首先先引入控件，此处我们需要引入ImageView控件

|  |
| --- |
| mImage = (ImageView) findViewById(R.id.*imageView*); |

* 接着设置对于点击事件的监听

|  |
| --- |
| mImage.setOnClickListener(new View.OnClickListener() {  @Override  public void onClick(View view) {  AlertDialog dialog=builder.create();//完成创建AlertDialog并显示  dialog.show();  } }); |

* 此处因为点击图片之后我们需要弹出一个AlertDialog对话框，所需要新建一个AlertDialog并设置好其选择内容，AlertDialog对话框创建如下

|  |
| --- |
| public void alertdialog\_build(){  //alertdialog 初始化  builder.setTitle("上传头像");  final String[] Items={"拍摄","从相册选择"};  builder.setItems(Items,new DialogInterface.OnClickListener(){  @Override //使用setItems构建选择列表 并增加点击检测  public void onClick(DialogInterface dialogInterface, int i) {  Toast.*makeText*(getApplicationContext(), "您选择了["+Items[i]+"]", Toast.*LENGTH\_SHORT*).show();  }  });  builder.setNegativeButton("取消", new DialogInterface.OnClickListener() {  @Override //设置取消按钮动作  public void onClick(DialogInterface dialogInterface, int i) {  Toast.*makeText*(getApplicationContext(),"您选择了[取消]", Toast.*LENGTH\_SHORT*).show();  }  });  builder.setCancelable(true); //允许取消 } |

* 1. RadioGroup切换的提示
* 考虑到我们在RadioGroup所选择的部分在之后button点击事件中还要使用，所以设置一个全局的String变量用来记录RadioGroup点击内容

|  |
| --- |
| String rb\_string="学生"; //初始化为默认选项学生 |

* 之后引入RadioGroup控件和RadioButton控件

|  |
| --- |
| mRB = (RadioGroup) findViewById(R.id.*radioGroup*);  rb\_student = (RadioButton) findViewById(R.id.*radioButton2*); rb\_teacher = (RadioButton) findViewById(R.id.*radioButton1*); |

* 然后设置RadioGroup改变选择的事件监听，此处弹出Snackbar，选择Snackbar的取消时还需要弹出Toast

|  |
| --- |
| mRB.setOnCheckedChangeListener(new RadioGroup.OnCheckedChangeListener() {  @Override  public void onCheckedChanged(RadioGroup group, int checkedId) {  if (rb\_student.getId() == checkedId) { //保存当前选择  rb\_string = rb\_student.getText().toString();  }  if (rb\_teacher.getId() == checkedId) {  rb\_string = rb\_teacher.getText().toString();  }  Snackbar.*make*(button\_init, "您选择了"+rb\_string, Snackbar.*LENGTH\_SHORT*)  .setAction("确定", new View.OnClickListener() {  @Override  public void onClick(View view) {  Toast.*makeText*(getApplicationContext(),"Snackbar的确定按钮被点击了",Toast.*LENGTH\_SHORT*).show();  }  })  .setActionTextColor(getResources().getColor(R.color.*colorPrimary*))  .show();  } }); |

* 1. Button点击事件的监听
* 引入button 控件

|  |
| --- |
| button\_login = (Button) findViewById(R.id.*button*);  button\_init = (Button) findViewById(R.id.*button2*); |

* 登录按钮事件监听,此处需要使用的例如TextInputLayout等已经引入只需要根据相应情况弹出对应提示

|  |
| --- |
| button\_login.setOnClickListener(new View.OnClickListener(){  @Override  public void onClick(View view){  if(et\_id.getText().toString().length()==0){ //先判断 学号  til\_id.setError("学号不能为空");  til\_id.setErrorEnabled(true);  }  else if(et\_password.getText().toString().length()==0) { //再判断 密码  til\_password.setError("密码不能为空");  til\_password.setErrorEnabled(true);  }  else if(et\_id.getText().toString().equals("123456") && et\_password.getText().toString().equals("6666")) { //如果都不为空 判断是否中输入正确  Snackbar.*make*(button\_init, "登录成功", Snackbar.*LENGTH\_SHORT*)  .setAction("确定", new View.OnClickListener() {  @Override  public void onClick(View view) {  Toast.*makeText*(getApplicationContext(), "Snackbar的确定按钮被点击了", Toast.*LENGTH\_SHORT*).show();  }  })  .setActionTextColor(getResources().getColor(R.color.*colorPrimary*))  .show();  }  else if(et\_id.getText().toString().length()!=0 && et\_password.getText().toString().length()!=0) { //输入不正确  Snackbar.*make*(button\_init, "学号或密码错误", Snackbar.*LENGTH\_SHORT*)  .setAction("确定", new View.OnClickListener() {  @Override  public void onClick(View view) {  Toast.*makeText*(getApplicationContext(), "Snackbar的确定按钮被点击了", Toast.*LENGTH\_SHORT*).show();  }  })  .setActionTextColor(getResources().getColor(R.color.*colorPrimary*))  .show();  }  } }); |

1. 实验中遇到的困难和解决思路：
2. 最开始写java文件的时候发现想用很多控件名字打出来都是cannot resolve symbol，然后百度半天import，然后发现可以直接使用alt+enter自动补全
3. 一开始修改好界面之后，在虚拟机中运行发现只要一开启app，屏幕焦点就自动集中在密码输入框上，然后输入密码的提示消失。于是百度之，最终解决方法是在xml文件中使得一开始把焦点移走，此处选择的是把焦点移到完全没有影响的地方上，在xml文件中增加如下两句:

|  |
| --- |
| android:focusable="true"  android:focusableInTouchMode="true" |

增加之后显示效果正常。

## 课后实验结果

* + 1. 实验中要求点击照片弹出拍照和从相册选择这两个选项，然而并不要求实现，只要求输出提示，于是百度了一下如何调出相机和相册，此处只需要在AlertDialog的Items，点击中多增加几句即可，使用Intent 打开相机或者相册，然后传入参数调用处理函数。

|  |
| --- |
| if(i==0){//额外 打开相机  Intent intent = new Intent(MediaStore.*ACTION\_IMAGE\_CAPTURE*);  startActivityForResult(intent, *CAMERA*); }else {//额外 打开相册  Intent intent = new Intent(Intent.*ACTION\_PICK*);  intent.setType("image/\*");  startActivityForResult(intent, *IMAGE*); } |

处理函数针对相机和相册分别处理，此处的相机处理比较简单只需要一句话丢给ImageView显示就可以了，此处因为我们不储存所以很简单，从相册获取，这个代码是像百度学习的，因为不同的系统需要不用的调用处理函数，所以此处也是不太清楚，于是就学习了适合API版本26的做法，使用输入流处理。

|  |
| --- |
| protected void onActivityResult(int requestCode, int resultCode, Intent data) {  if (requestCode == *CAMERA* && resultCode == *RESULT\_OK*) {  Bitmap photo = (Bitmap) data.getExtras().get("data");  mImage.setImageBitmap(photo); //设置图片  para = mImage.getLayoutParams(); // 设置自动宽高   para.height = height;  para.width = width;  mImage.setLayoutParams(para);   }else if(requestCode == *IMAGE* && resultCode == *RESULT\_OK*){  try { //此处提示需要抛出异常 所以加了  InputStream inputStream = getContentResolver().openInputStream(data.getData());  Bitmap photo = BitmapFactory.*decodeStream*(inputStream); //使用输入流转化图片  mImage.setImageBitmap(photo);  para = mImage.getLayoutParams();// 设置自动宽高   para.height = height;  para.width = width;  mImage.setLayoutParams(para);  }catch (FileNotFoundException e) {  e.printStackTrace();  }  }  super.onActivityResult(requestCode, resultCode, data); } |

此处需要设置固定宽高，所以要先获取图片宽高，测试了百度上的两种方式最后发现用View.MeasureSpec测量宽高，之后设置图片的时候LayoutParams固定照片大小比较方便。

* + 1. TextInputLayout控件在输入结束时直接判断是不是输了空账号密码，而不是需要经过button
* TextInputLayout和EditText控件

|  |
| --- |
| et\_id = (EditText) findViewById(R.id.*textid*);  et\_password = (EditText) findViewById(R.id.*textpassword*); til\_id = (TextInputLayout) findViewById(R.id.*textInputLayout*); til\_password = (TextInputLayout) findViewById(R.id.*textInputLayout1*); |

* 增加输入框输入的事件监听(此处展示输入学号密码同理)，此处同时使得提示更加友好，选择教职工的时候提示从学号xxxx变成教职工号xxxx

|  |
| --- |
| et\_id.addTextChangedListener(new TextWatcher() {  @Override  public void beforeTextChanged(CharSequence charSequence, int i, int i1, int i2) {  }  @Override  public void onTextChanged(CharSequence charSequence, int i, int i1, int i2) {  }  @Override  public void afterTextChanged(Editable editable) { //额外//输入完毕自主判断  if (editable.length() == 0) {  til\_id.setErrorEnabled(true);  til\_id.setError(rb\_string\_hit+"不能为空");  } else {  til\_id.setErrorEnabled(false);  }  } }); |

## 实验思考及感想

这次实验比较简单，只需要把各种控件拿出来组UI就可以了，主要就是注意下要求，没有什么过于复杂的地方，最多就是对齐比较麻烦，其他的东西实验文档已经很清楚了，照着实验文档进行实验即可，直接在可视界面对控件属性进行修改速度比使用代码修改会方便很多，同时在可视界面直接组控件，比去找代码也会方便，当然有的属性要翻一下的时候还是用代码比较快比如那个radiogroup的水平排布。