Android软件设计课程实验

Android软件设计课程实验报告

学号 SA20225506

姓名 谢雪松

日期 2020.10.19

|  |  |
| --- | --- |
| **实验名称** | 实验一：熟练使用Activity，设置控件 |
| **实验内容** | 要求：  1. 页面上至少包含三种不同的控件；  2. 要求点击控件会产生控件外观的变化(例如颜色、文字等)；  完成要求的基本功能；  UI设计美观可加分；  除控件本身变化，实现其余附加功能可另外加分； |
| **实验完成情况**  （包括完成的实验内容及每个实验的完成程度。注意要贴出每个实验的  核心代码） | 完成的实验内容：   1. 页面中含有Button、TextView、EditText、ImageView三种不同控件。 2. 点击EditText会产生hint文字的变化；点击Button会产生颜色变化。 3. 点击用户名EditText在输入完成后，会查询数据库里该用户名对应的头像，并显示在头像的ImagaView里；点击登录Button会获取两个Edit Text中的用户名和密码信息，并查询数据库，登录结果用Toast展示给用户。   完成程度：   1. 控件从login\_layout.xml文件里加载，里面定义了Button组件，其中通过 android:background="@drawable/buttonselector"来设定按钮的背景 2. 定义了两个EditText控件，用于输入用户名和密码。其中通过android:hint="@string/name"可以指定输入框的提示文字。 3. 定义了一个ImageView控件用于显示头像信息。通过android:src="@drawable/head"可以指定该控件的初始图像，这里我们设为了用户的默认头像图片。 4. 定义了两个TextView对象，分别显示为“忘记密码”、“用户注册”，可以提示用户点击后进行找回密码或注册新用户等操作。但这次实验并未实现上述功能，后续迭代开发将会实现。 5. 通过DBHelper类，实现对用户对象（User）的持久化操作。其中，用户对象包括用户名（name），密码（password），头像（avatar）三个主要字段，通过调用DBHelper类的addUserWithAvatar（）方法可以向sqlite数据库里写入一个带有头像的用户信息。通过haveUser（）方法可以查找数据库中是否存在某个用户；通过getAvatarByName（）方法，可以根据用户名查找该用户的图片；通过loginWith（）方法可以使用用户名和密码等参数在数据库中查找，进而完成登录的过程。 |
| **实验中的问题**  （包括在实验中遇到的问题，以及解决问题的方法） | 1. EditText在默认情况下，点击后开始输入时，hint不会消失，影响用户体验。   解决方法：让LoginActivity类实现View.OnFocusChangeListener接口，并重写onFocusChange（）方法，在该方法里，判断某个输入框是否有焦点，如果有就让hint值为空，如果没有就让hint值为对应的值。 最后，在onCreate( )将输入框对象的通过setOnFocusChangeListener(this)绑定监听器。   1. 按钮在点击时，默认情况下是不会有明显的颜色变化。   解决办法：  android:background="@drawable/buttonselector"来设定按钮的背景  并在buttonselector.xml文件中加入<item android:drawable="@drawable/buttonpress" android:state\_pressed="true"/> 来改变按钮点击时的颜色。   1. 登录界面不能全屏显示，影响用户体验。   解决办法：  在Manifest.xml文件中 application节点的属性里添加  android:theme="@style/Theme.AppCompat.Light.NoActionBar.FullScreen" |
| **实验结果**  **及成果展示**  （包括实验完成后的源码和打包文件的说明） | **实验结果展示**  打开应用后显示登录界面，点击输入框时，hint会消失    当输完用户名，开始输入密码时，如果数据库中该用户有头像，则会显示在上面；如果没有该用户，或该用户没有头像，则显示默认头像图片。    输入密码，点击登录时按钮会色，当按钮释放时恢复原来的颜色。之后查询数据库进行登录验证。如果登录成功，会跳转到下一个页面。    如果密码错误，会用Toast提示“密码错误”。  如果该用户不存在，会提示“该用户不存在！”。    **实验代码展示**  **登录界面的布局文件login\_layout.xml**  主要是对登录界面布局的定义，整体采用相对布局的方式。定义的控件分别有头像图片显示、账号及密码的编辑框、登录按钮。底部在相对布局基础上左右文字显示采用水平线性布局。  <?xml version="1.0" encoding="utf-8"?> <RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"  android:layout\_width="match\_parent"  android:layout\_height="match\_parent"  android:focusableInTouchMode="true"  >   <!-- 整个页面使用相对布局 -->   <!-- 实现头像的ImageView -->  <ImageView  android:id="@+id/pic"  android:layout\_width="wrap\_content"  android:layout\_height="100dp"  android:layout\_centerHorizontal="true"  android:layout\_margin="40dp"  android:src="@drawable/head" />   <!-- 作输入框的EditText -->  <EditText  android:id="@+id/Acount"  android:layout\_width="match\_parent"  android:layout\_height="65dp"  android:layout\_below="@+id/pic"  android:layout\_marginLeft="30dp"  android:layout\_marginRight="30dp"  android:layout\_weight="1"  android:background="@drawable/edittextselector"  android:hint="@string/name"  android:selectAllOnFocus="false"  android:singleLine="true"  android:textAlignment="center"  android:visibility="visible" />   <EditText  android:id="@+id/passwd"  android:layout\_width="match\_parent"  android:layout\_height="65dp"  android:layout\_below="@+id/Acount"  android:layout\_marginLeft="30dp"  android:layout\_marginTop="10dp"  android:layout\_marginRight="30dp"  android:layout\_marginBottom="35dp"  android:layout\_weight="1"  android:background="@drawable/edittextselector"  android:hint="@string/passwd"  android:inputType="textPassword"  android:selectAllOnFocus="false"  android:singleLine="true"  android:textAlignment="center"  android:visibility="visible" />   <Button  android:id="@+id/loginbutton"  android:layout\_width="80dp"  android:layout\_height="50dp"  android:layout\_below="@+id/passwd"  android:layout\_centerHorizontal="true"  android:background="@drawable/buttonselector"  android:text="@string/login" />   <!-- 下半部分使用线性布局 -->   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_alignParentBottom="true"  android:layout\_marginBottom="20dp"  android:orientation="horizontal">   <!-- 分左右两个线性布局实现两个TextView居中对称 -->   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:orientation="vertical">   <TextView  android:id="@+id/forget"  android:layout\_width="wrap\_content"  android:layout\_height="match\_parent"  android:layout\_gravity="center\_horizontal"  android:hint="@string/forget" />   </LinearLayout>   <LinearLayout  android:layout\_width="match\_parent"  android:layout\_height="wrap\_content"  android:layout\_weight="1"  android:orientation="vertical">   <TextView  android:id="@+id/signin"  android:layout\_width="wrap\_content"  android:layout\_height="match\_parent"  android:layout\_gravity="center\_horizontal"  android:hint="@string/signin" />  </LinearLayout>  </LinearLayout>   </RelativeLayout>  **LoginActivity.java**  登陆界面对应的Activity代码，除了上面布局文件所需要的控件包，还引入数据库包来进行数据库查询操作，细分功能列举如下：   1. 未输入账号密码，文字框内有提示；点击文字框提示消失 2. 账号不对，头像不变，点击登录Toast提示“该用户不存在！”；账号正确，自动显示对应头像 3. 账号正确而密码错误，则Toast提示“密码错误”；账号密码均正确则登录并intent跳转到聊天页面。   package com.rom471.main; import android.content.Intent; import android.graphics.drawable.Drawable; import android.os.Bundle; import android.view.Gravity; import android.view.View; import android.widget.Button; import android.widget.EditText; import android.widget.ImageView; import android.widget.Toast; import androidx.appcompat.app.AppCompatActivity; import com.rom471.lab1.R; import com.rom471.userdb.DBHelper;  public class LoginActivity extends AppCompatActivity implements View.OnClickListener, View.OnFocusChangeListener {  EditText name\_et;//用户名输入框  EditText password\_et;//密码输入框  DBHelper db; //数据库建立工具  ImageView avatar\_img; //头像显示图片  Button login\_bt;//登录按钮  @Override  protected void onCreate(Bundle savedInstanceState) {  super.onCreate(savedInstanceState);  setContentView(R.layout.*login\_layout*);  //绑定资源  avatar\_img = findViewById(R.id.*pic*);  name\_et = findViewById(R.id.*Acount*);  password\_et = findViewById(R.id.*passwd*);  login\_bt = findViewById(R.id.*loginbutton*);  //设置登陆按钮监听  login\_bt.setOnClickListener(this);  //设置文本框焦点变化监听  name\_et.setOnFocusChangeListener(this);  password\_et.setOnFocusChangeListener(this);  //初始化数据库  db = new DBHelper(this);  db.InitialWithTestData();//使用测试数据初始化  }  //按键处理  @Override  public void onClick(View v) {  switch (v.getId()) {  case R.id.*loginbutton*:  login\_action();  break;  }  }  //点击登录按钮执行的操作  private void login\_action() {  String name = name\_et.getText().toString();  String password = password\_et.getText().toString();  if (!db.haveUser(name)) {  toast("该用户不存在！");  return;  }  if (db.loginWith(name, password)) {  // 登录成功，跳转到Home界面  Intent intent = new Intent(LoginActivity.this, HomeActivity.class);  startActivity(intent);  } else {  toast("密码错误");  }  }  //输入框焦点变化时，会使hint消失或出现；如果是用户名输入框，还会更新头像信息  @Override  public void onFocusChange(View v, boolean hasFocus) {  switch (v.getId()) {  case R.id.*Acount*:  if (hasFocus)  ((EditText) v).setHint("");  else {  ((EditText) v).setHint(R.string.*name*);  changeAvatarImage();  }  break;  case R.id.*passwd*:  if (hasFocus) {  ((EditText) v).setHint("");  } else  ((EditText) v).setHint(R.string.*passwd*);  break;  }  }  //从输入框获取用户名，查询数据库，改变头像  private void changeAvatarImage() {  String name = name\_et.getText().toString();//获取输入的用户名  Drawable avatar\_drawable = db.getAvatarByName(name);  if (avatar\_drawable != null) //数据库里有头像  avatar\_img.setImageDrawable(avatar\_drawable);  else //数据库里不存在该用户，或该用户没有头像，设置默认头像  avatar\_img.setImageResource(R.drawable.*head*);  }  //在合理位置显示Toast  private void toast(String text) {  Toast toast = Toast.*makeText*(this, text, Toast.*LENGTH\_SHORT*);  toast.setGravity(Gravity.*CENTER*, 0, 0);  toast.show();  } }  **User.java**  用户的账号密码头像的设置定义文件，定义用户User类，具体在数据库定义的类中被实例化。  package com.rom471.userdb; import android.graphics.drawable.Drawable; //用户的JavaBean public class User {  private int id;//用户id，由数据库指定  private String name;//用户姓名  private String password;//用户密码  private Drawable avatar;//用户头像  public User() {  }  @Override  public String toString() {  return "User{" +  "id=" + id +  ", name='" + name + '\'' +  ", password='" + password + '\'' +  '}';  }  public User(String name, String password, Drawable avatar) {  this.name = name;  this.password = password;  this.avatar = avatar;  }  public void setAvatar(Drawable avatar) {  this.avatar = avatar;  }  public Drawable getAvatar() {  return avatar;  }  public int getId() {  return id;  }  public void setId(int id) {  this.id = id;  }  public String getName() {  return name;  }  public String getPassword() {  return password;  }  public void setPassword(String password) {  this.password = password;  }  public void setName(String name) {  this.name = name;  }   }  **DBHelper.java**  数据库基本操作是在此文件下定义，包括创建table、初始化表（这里初始化的同时写入了我们小组三个默认用户）、表插入新用户（用于注册新用户）、索引添加头像、用户查询等功能。  package com.rom471.userdb;  import android.content.ContentValues; import android.content.Context; import android.database.Cursor; import android.database.sqlite.SQLiteDatabase; import android.database.sqlite.SQLiteOpenHelper; import android.graphics.BitmapFactory; import android.graphics.drawable.BitmapDrawable; import android.graphics.drawable.Drawable; import com.rom471.lab1.R; import java.util.ArrayList; import java.util.List;  public class DBHelper extends SQLiteOpenHelper {  //数据库名字  public static final String *DB\_NAME*="users.db";  //存放用户信息的表名  public static final String *TABLE\_NAME*="User";  public static final int *DB\_VERSION*=1;  //内部Context引用  private Context context;  //内部数据库引用  private SQLiteDatabase db;   public DBHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {  super(context, name, factory, version);  }   @Override  public void onCreate(SQLiteDatabase db) {  }  @Override  public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {   }  //初始化内部引用  public DBHelper(Context context){  super(context,*DB\_NAME*,null,*DB\_VERSION*);  db=super.getWritableDatabase();  this.context=context;  }  //创建用户信息表  public void createUserTable(){  final String CREATE\_USER="create table User ("  + "id integer primary key autoincrement,"  + "name text ,"  + "password text,"  + "avatar blob)";  db.execSQL(CREATE\_USER);  }   public void dropTable(String table\_name){  db.execSQL("drop table if exists "+table\_name);  }  //判断是否存在表  public boolean havingTable(String table\_name){  Cursor cursor;  boolean a=false;  cursor = db.rawQuery("select name from sqlite\_master where type='table' ", null);  while(cursor.moveToNext()) {  //遍历出表名  String name = cursor.getString(0);  if (name.equals(table\_name)) { //存在表  cursor = db.query(table\_name, null, null, null, null, null, null);  //检查是不是空表  if (cursor.getCount() > 0) {  return true;  } else //空表  return false;  }  }  return false;  }   //用测试数据初始化数据库  public void InitialWithTestData(){  if(havingTable(*TABLE\_NAME*))  return;  createUserTable();  //添加3个测试用户  addUserWithAvatar("xxs","xxs",R.drawable.*avatar\_xxs*);  addUserWithAvatar("wjf","wjf",R.drawable.*avatar\_wjf*);  addUserWithAvatar("wkq","wkq",R.drawable.*avatar\_wkq*);  }  //添加带头像的用户  private void addUserWithAvatar(String name,String password,int rid){  Drawable avatar=MyUtils.*getDrawableFromResource*(context,rid);  User user=new User(name,password,avatar);  this.insertUsers(user);  }  //向表中插入一个用户  public void insertUsers(User user){  ContentValues contentValues=new ContentValues();  contentValues.put("name",user.getName());  contentValues.put("password",user.getPassword());  Drawable avatar = user.getAvatar();  contentValues.put("avatar",MyUtils.*getBytesFromDrawable*(avatar));  db.insert(*TABLE\_NAME*,null,contentValues);  }  //查询所有用户并返回  public List<User> queryAll(){  List<User> users=new ArrayList<>();  Cursor cursor=db.query(*TABLE\_NAME*,new String[]{"id","name","password","avatar"},null,null,null,null,null);  while(cursor.moveToNext()){  int id=cursor.getInt(cursor.getColumnIndex("id"));  String name=cursor.getString(cursor.getColumnIndex("name"));  String password=cursor.getString(cursor.getColumnIndex("password"));  Drawable avatar=MyUtils.*getDrawableFormCursor*(cursor,"avatar");//直接从数据库获取头像Drawable  User user=new User(name,password,avatar);  user.setId(id);  users.add(user);  }  cursor.close();  return users;  }  //判断用户名是否存在  public boolean haveUser(String name){  String sql=" select \* from "+*TABLE\_NAME*+ " where name = ?";  Cursor cursor=db.rawQuery(sql,new String[]{name});  if(cursor.moveToFirst()==true){  cursor.close();  return true;  }  return false;  }  //使用用户名和密码登录  public boolean loginWith(String name,String password){  String sql=" select \* from "+*TABLE\_NAME*+ " where name = ? and password=?";  Cursor cursor=db.rawQuery(sql,new String[]{name,password});  if(cursor.moveToFirst()==true){  cursor.close();  return true;  }  return false;  }  //由用户名获取头像  public Drawable getAvatarByName(String name){  String sql = "SELECT avatar FROM "+*TABLE\_NAME* +" WHERE name = ?";  Drawable avatar=null;  byte[] bytes;  Cursor cursor=db.rawQuery(sql,new String[]{name});  if(cursor.moveToFirst()){  if((bytes=cursor.getBlob(cursor.getColumnIndex("avatar")))!=null){  avatar= new BitmapDrawable(BitmapFactory.*decodeByteArray*(bytes, 0, bytes.length));  }  }  cursor.close();  return avatar;  } }  MyUtils.java  资源定义文件，主要是将账户id和用户头像图片进行绑定。  package com.rom471.userdb;  import android.content.Context; import android.content.res.Resources; import android.database.Cursor; import android.graphics.Bitmap; import android.graphics.BitmapFactory; import android.graphics.drawable.BitmapDrawable; import android.graphics.drawable.Drawable; import java.io.ByteArrayOutputStream;  //工具类 public class MyUtils {  //指定资源id，来设置用户的头像  public static Drawable getDrawableFromResource(Context context, int rid) {  Resources resources = context.getResources();  Bitmap bmp = BitmapFactory.*decodeResource*(resources, rid);  ByteArrayOutputStream baos = new ByteArrayOutputStream();  bmp.compress(Bitmap.CompressFormat.*PNG*, 100, baos);//压缩图片  byte[] bytes = baos.toByteArray();  return *getDrawableFromBytes*(bytes);  }  //将Drawable对象转换为bytes好存入数据库  public static byte[] getBytesFromDrawable(Drawable d) {   if (d != null) {  Bitmap imageBitmap = ((BitmapDrawable) d).getBitmap();  ByteArrayOutputStream baos = new ByteArrayOutputStream();  imageBitmap.compress(Bitmap.CompressFormat.*PNG*, 100, baos);  byte[] byteData = baos.toByteArray();  return byteData;  } else  return null;  }  public static Drawable getDrawableFromBytes(byte[] data) {  if (data == null)  return null;  else  return new BitmapDrawable(BitmapFactory.*decodeByteArray*(data, 0, data.length));  }  public static Drawable getDrawableFormCursor(Cursor cursor, String index) {  byte[] avatar\_bytes = cursor.getBlob(cursor.getColumnIndex("avatar"));  return *getDrawableFromBytes*(avatar\_bytes);  } } |