一、数据库结构

1、分类名表

tb\_category{

id integer primary key autoincrement;

category\_name text;

};

2、笔记本最后修改时间表

tb\_book\_time{

book\_id integer primary key;

book\_time timestamp;

};

3、笔记本属性表

tb\_bookid{ //”bookid”在表创建时，取笔所获取到的bookid值

page\_id integer primary key;

page\_time timestamp;

category\_name text;

page\_name text;

pen\_name text;

page\_image text;

};

4、页面笔迹点属性表

tb\_bookid\_pageid{ /\* ”bookid”和”pageid”在表创建时，取笔所获取到 的bookid和pageid值 \*/

id integer primary key autoincrement;

book\_id integer;

page\_id integer;

counter integer;

pointX float;

pointY float;

force integer;

type integer;

pen\_width integer;

color integer;

angle integer;

};

二、实现步骤（OidActivity类）：

1、定义全局变量

代码：

private DatabaseHelper database\_helper;  
 private SQLiteDatabase db;  
 private Map<Integer,Boolean> newDot = new HashMap<Integer,Boolean>(); //新点标志  
 private Map<Integer,Integer> save\_sql\_dot\_num = new HashMap<Integer,Integer>(); //各页数据库点存储数

private Map<Integer,Boolean> newPage\_image = new HashMap<Integer,Boolean>(); //新页标志

private String category\_name = "默认";  
 private Handler save\_sql\_handler;  
 private Runnable save\_sql\_runnable;  
 private String[] allCategory\_name;

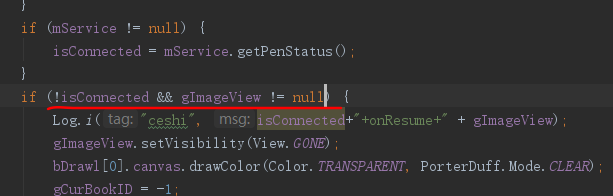
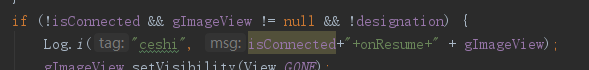
1. 在onCreate()中接收Note\_category\_main类传送的信息

代码：

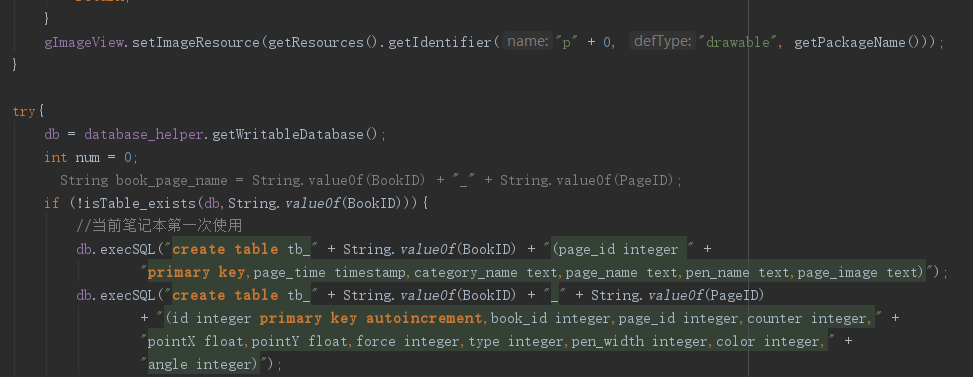
database\_helper = new DatabaseHelper(this,"note\_book.db",null,1);  
 Intent intent = getIntent();  
 designation = intent.getBooleanExtra("DESIGNATION",false);  
 allCategory\_name = intent.getStringArrayExtra("ALLCATEGORY\_NAME");  
 if (designation){  
 *dot\_number*.clear();  
 int go\_book\_id = intent.getIntExtra("BOOKID",0);  
 int go\_page\_id = intent.getIntExtra("PAGEID",0);

SetBackgroundImage(go\_book\_id, go\_page\_id);  
 gImageView.setVisibility(View.*VISIBLE*); //视图可见  
 gCurBookID = go\_book\_id;  
 gCurPageID = go\_page\_id;  
 drawInit();  
 DrawExistingStroke(gCurBookID, gCurPageID);  
 }else {  
 category\_name = intent.getStringExtra("CATEGORY\_NAME");  
 }

1. 修改onResume()

将更改为：

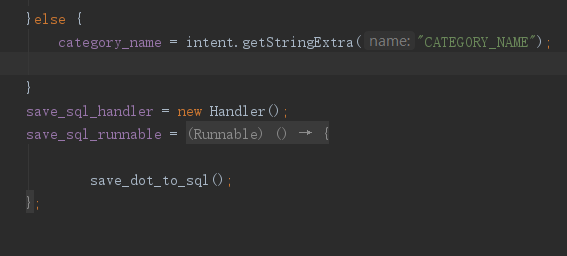
1. 在SetBackgroundImage()中实现当前页数据初始化



代码：

try{  
 db = database\_helper.getWritableDatabase();  
 int num = 0;  
 if (!isTable\_exists(db,String.*valueOf*(BookID))){  
 //当前笔记本第一次使用  
 db.execSQL("create table tb\_" + String.*valueOf*(BookID) + "(page\_id integer " +  
 "primary key,page\_time timestamp,category\_name text,page\_name text,pen\_name text,page\_image text)");  
 db.execSQL("create table tb\_" + String.*valueOf*(BookID) + "\_" + String.*valueOf*(PageID)  
 + "(id integer primary key autoincrement,book\_id integer,page\_id integer,counter integer," +  
 "pointX float,pointY float,force integer,type integer,pen\_width integer,color integer," +  
 "angle integer)");  
 ContentValues values = new ContentValues();  
 values.put("page\_id",PageID);  
 values.put("category\_name",category\_name);  
 db.insert("tb\_" + String.*valueOf*(BookID),null,values);  
  
 values.clear();  
 values.put("book\_id",BookID);  
 db.insert("tb\_book\_time",null,values);  
 save\_sql\_dot\_num.put(PageID,0);  
 Log.*d*(*TAG*, "SetBackgroundImage: PageID:" + PageID);  
 } else if (!isTable\_exists(db,String.*valueOf*(BookID) + "\_" + String.*valueOf*(PageID))){  
 //当前页第一次使用  
 db.execSQL("create table tb\_" + String.*valueOf*(BookID) + "\_" + String.*valueOf*(PageID)  
 + " (id integer primary key autoincrement,book\_id integer,page\_id integer,counter integer," +  
 "pointX float,pointY float,force integer,type integer,pen\_width integer,color integer," +  
 "angle integer)");  
 ContentValues values = new ContentValues();  
 values.put("page\_id",PageID);  
 values.put("category\_name",category\_name);  
 db.insert("tb\_" + String.*valueOf*(BookID),null,values);  
 save\_sql\_dot\_num.put(PageID,0);  
 } else {  
 try {  
 Cursor cursor = db.query("tb\_" + String.*valueOf*(BookID) + "\_" + String.*valueOf*(PageID),  
 new String[]{"book\_id", "page\_id", "counter", "pointX", "pointY", "force",  
 "type", "pen\_width", "color", "angle"}, "book\_id = ? and page\_id = ?",  
 new String[]{String.*valueOf*(BookID), String.*valueOf*(PageID)}, null, null,  
 null);  
 if (cursor.moveToFirst()) {  
 //读取当前页笔迹点数据  
 do {  
 int book\_id = cursor.getInt(0);  
 int page\_id = cursor.getInt(1);  
 int counter = cursor.getInt(2);  
 float pointX = cursor.getFloat(3);  
 float pointY = cursor.getFloat(4);  
 int force = cursor.getInt(5);  
 int type = cursor.getInt(6);  
 int pen\_width = cursor.getInt(7);  
 int color = cursor.getInt(8);  
 int angle = cursor.getInt(9);  
  
 Dots dots = new Dots(book\_id, page\_id, pointX, pointY, force, type, pen\_width  
 , color, counter, angle);  
 *dot\_number*.put(page\_id, dots);  
 num++;  
  
 } while (cursor.moveToNext());  
  
 cursor.close();  
 } else {  
 //当前页数据被清空过  
 ContentValues values = new ContentValues();  
 values.put("page\_id",PageID);  
 values.put("category\_name",category\_name);  
 db.insert("tb\_" + String.*valueOf*(BookID),null,values);  
 }  
 } catch (Exception e){  
 e.printStackTrace();  
 }  
 save\_sql\_dot\_num.put(PageID, num);  
 }  
  
 } catch (Exception e){  
 e.printStackTrace();  
 }  
 Log.*d*(*TAG*, "SetBackgroundImage: save\_sql\_dot\_num:" + save\_sql\_dot\_num);  
  
 newDot.put(PageID,false);

1. 在onCreate()中实现当前页笔迹点数据本地存储



代码：

save\_sql\_handler = new Handler();  
save\_sql\_runnable = new Runnable() {  
 @Override  
 public void run() {  
  
 save\_dot\_to\_sql();  
 }  
};

1. 定义save\_dot\_to\_sql()笔迹点数据存储函数

代码：

private void save\_dot\_to\_sql() {  
 int page\_id = gCurPageID;  
 int book\_id = gCurBookID;  
 List<Dots> dotsList = *dot\_number*.get(page\_id);  
  
 if (!newDot.get(page\_id)){  
 return;  
 }  
 ContentValues values = new ContentValues();  
 String save\_time = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss:SSS").format(new Date());  
  
 values.put("page\_time",save\_time);  
 values.put("pen\_name",ApplicationResources.*mPenName*);  
 db.update("tb\_" + String.*valueOf*(book\_id),values,"page\_id = ?",  
 new String[]{ String.*valueOf*(page\_id) });  
 values.clear();  
  
 values.put("book\_time",save\_time);  
 db.update("tb\_book\_time",values,"book\_id = ?",new String[]{ String.*valueOf*(book\_id) });  
 int num = save\_sql\_dot\_num.get(page\_id);  
 int dot\_num = num;  
 int count = dotsList.size();  
 while (dot\_num < count){  
 values.clear();  
 Dots dots = dotsList.get(dot\_num);  
 values.put("book\_id",dots.bookID);  
 values.put("page\_id",dots.pageID);  
 values.put("counter",dots.ncounter);  
 values.put("pointX",dots.pointX);  
 values.put("pointY",dots.pointY);  
 values.put("force",dots.force);  
 values.put("type",dots.ntype);  
 values.put("pen\_width",dots.penWidth);  
 values.put("color",dots.ncolor);  
 values.put("angle",dots.nangle);  
 long flag = db.insert("tb\_" + String.*valueOf*(book\_id) + "\_" + String.*valueOf*(page\_id),  
 null,values);  
 if (flag != -1){  
 num++;  
 }  
 dot\_num++;  
  
 }  
 db.setTransactionSuccessful();  
 save\_sql\_dot\_num.put(page\_id,num);  
 save\_page\_image();  
 newDot.put(page\_id,false);  
 db.endTransaction();  
 }

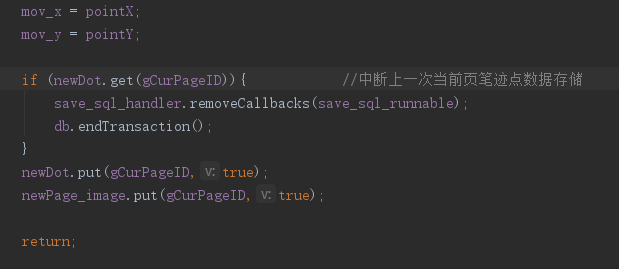
1. 定义save\_page\_image()保存函数

代码：

private void save\_page\_image() {  
 */\*\*  
 \* decorView是window中的最顶层view，可以从window中获取到decorView  
 \* ，然后decorView有个getWindowVisibleDisplayFrame方法可以获取到程序显示的区域，包括标题栏  
 \* ，但不包括状态栏。  
 \*/* View windowView = getWindow().getDecorView();  
 windowView.setDrawingCacheEnabled(true);  
 windowView.buildDrawingCache();  
 Bitmap bitmap = windowView.getDrawingCache();  
  
 try {  
 //“外部缓存”路径  
 String cachePath = getExternalCacheDir().getPath();  
 String filePath = cachePath + File.*separator* + String.*valueOf*(gCurBookID) + "\_"  
 + String.*valueOf*(gCurPageID) + ".png";  
 File file = new File(filePath);  
 if (file.exists()) {  
 file.delete();  
 //通知媒体数据库删除该图片的缩略图  
 getContentResolver().delete(MediaStore.Images.Media.*EXTERNAL\_CONTENT\_URI* , MediaStore.Images.Media.*DATA* + "=?", new String[]{filePath});  
  
 } else {  
 ContentValues values = new ContentValues();  
 values.put("page\_image", filePath);  
 db.update("tb\_" + String.*valueOf*(gCurBookID), values, "page\_id = ?"  
 , new String[]{String.*valueOf*(gCurPageID)});  
 }  
   
 file = null;  
 FileOutputStream outputStream = new FileOutputStream(filePath);  
 bitmap.compress(Bitmap.CompressFormat.*PNG*, 100, outputStream);  
 outputStream.flush();  
 outputStream.close();  
 newPage\_image.put(gCurPageID,false);  
 //刷新系统图库  
 MediaScannerConnection.*scanFile*(this, new String[]{filePath}, null, null);  
   
 } catch (Exception e) {  
 e.printStackTrace();  
 }  
 //释放缓存  
 windowView.setDrawingCacheEnabled(false);  
 windowView.buildDrawingCache(false);  
}

1. 在ProcessEachDot()中对笔迹点数据存储进行起始和中断操作

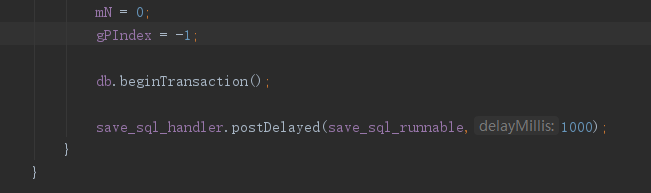
### 8.1、在笔下落阶段



代码：

if (newDot.get(gCurPageID)){ //中断上一次当前页笔迹点数据存储  
 save\_sql\_handler.removeCallbacks(save\_sql\_runnable);  
 db.endTransaction();  
}  
newDot.put(gCurPageID,true);  
newPage\_image.put(gCurPageID,true);

### 8.2、在笔拿起阶段

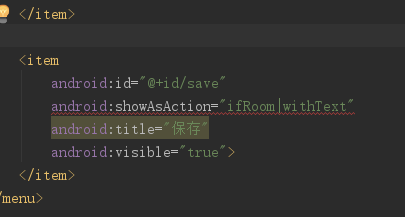


代码：

db.beginTransaction();  
  
save\_sql\_handler.postDelayed(save\_sql\_runnable,1000);

1. 实现当前页面指定保存

### 9.1、修改res/menu/option\_menu.xml



代码：

<item  
 android:id="@+id/save"  
 android:showAsAction="ifRoom|withText"  
 android:title="保存"  
 android:visible="true">  
</item>

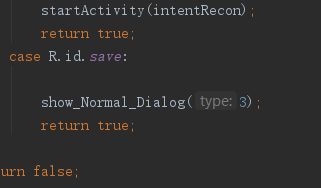
### 9.2、更改onCreateOptionsMenu()



代码：

if (!isConnected && !designation) {  
 menu.findItem(R.id.*flag1*).setVisible(false);  
 menu.findItem(R.id.*flag2*).setVisible(true);  
 menu.findItem(R.id.*clear*).setVisible(false);  
 menu.findItem(R.id.*color*).setVisible(false);  
 menu.findItem(R.id.*width*).setVisible(false);  
 menu.findItem(R.id.*replay*).setVisible(false);  
 menu.findItem(R.id.*setting*).setVisible(false);  
 menu.findItem(R.id.*offline*).setVisible(false);  
 menu.findItem(R.id.*LED*).setVisible(false);  
 menu.findItem(R.id.*OTA*).setVisible(false);  
 menu.findItem(R.id.*check\_txt*).setVisible(false);  
 menu.findItem(R.id.*recognition*).setVisible(false);  
 menu.findItem(R.id.*save*).setVisible(false);  
} else if (designation) {  
 menu.findItem(R.id.*flag1*).setVisible(false);  
 menu.findItem(R.id.*flag2*).setVisible(true);  
 menu.findItem(R.id.*clear*).setVisible(true);;  
 menu.findItem(R.id.*replay*).setVisible(true);  
 menu.findItem(R.id.*recognition*).setVisible(true);  
 menu.findItem(R.id.*save*).setVisible(true);  
 menu.findItem(R.id.*color*).setVisible(false);  
 menu.findItem(R.id.*width*).setVisible(false);  
 menu.findItem(R.id.*setting*).setVisible(false);  
 menu.findItem(R.id.*offline*).setVisible(false);  
 menu.findItem(R.id.*LED*).setVisible(false);  
 menu.findItem(R.id.*OTA*).setVisible(false);  
 menu.findItem(R.id.*check\_txt*).setVisible(false);  
} else {  
 menu.findItem(R.id.*flag1*).setVisible(true);  
 menu.findItem(R.id.*flag2*).setVisible(false);  
 menu.findItem(R.id.*clear*).setVisible(true);  
 menu.findItem(R.id.*color*).setVisible(true);  
 menu.findItem(R.id.*width*).setVisible(true);  
 menu.findItem(R.id.*replay*).setVisible(true);  
 menu.findItem(R.id.*setting*).setVisible(true);  
 menu.findItem(R.id.*offline*).setVisible(true);  
 menu.findItem(R.id.*LED*).setVisible(false);  
 menu.findItem(R.id.*OTA*).setVisible(false);  
 menu.findItem(R.id.*check\_txt*).setVisible(true);  
 menu.findItem(R.id.*recognition*).setVisible(true);  
 menu.findItem(R.id.*save*).setVisible(true);  
}

### 9.3、更改onOptionsItemSelected()

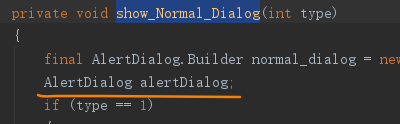


代码：

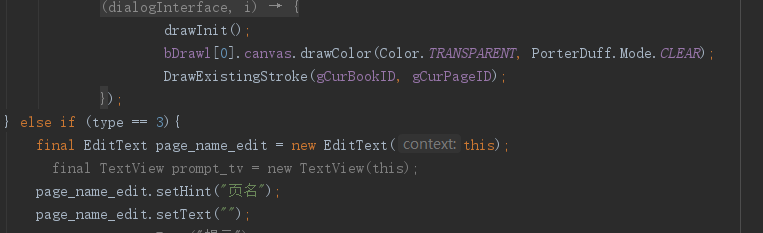
case R.id.*save*:  
  
 show\_Normal\_Dialog(3);  
 return true;

### 9.4、更改show\_Normal\_Dialog()

#### 9.4.1、初始



#### **9.4.2、定义当前页指定保持对话框**

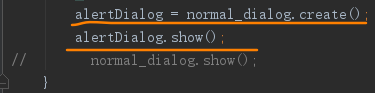


代码：

} else if (type == 3){  
 final EditText page\_name\_edit = new EditText(this);  
 page\_name\_edit.setHint("当前页页名");  
 page\_name\_edit.setText("");  
// prompt\_tv.setText("提示");  
 final int[] pos = {-1};  
  
 normal\_dialog.setTitle("请为当前页选择类别和输入页名");

normal\_dialog.setSingleChoiceItems(allCategory\_name, -1, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialogInterface, int which) {  
 pos[0] = which;  
 }  
 });  
 normal\_dialog.setView(page\_name\_edit);  
 normal\_dialog.setPositiveButton("确定", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialogInterface, int i) {  
 String page\_name = page\_name\_edit.getText().toString();  
 if (pos[0] == -1 || page\_name.isEmpty()){  
 Toast.*makeText*(OidActivity.this,"请选择类别或填充页名",Toast.*LENGTH\_SHORT*).show();  
 } else {  
 String chosen\_category\_name = allCategory\_name[pos[0]];  
 ContentValues values = new ContentValues();  
 values.put("category\_name",chosen\_category\_name);  
 values.put("page\_name",page\_name);  
 db.update("tb\_" + String.*valueOf*(gCurBookID),values,"page\_id = ?",new String[] { String.*valueOf*(gCurPageID) });  
 }  
 }  
 });  
 normal\_dialog.setNegativeButton("取消", new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialogInterface, int i) {  
  
 }  
 });  
   
 }

#### **9.4.3、修改show()方法**



二、参考资料：

定时任务执行：https://www.cnblogs.com/fozero/p/5876328.html

三、改进建议：

在OidActivity()整体布局界面风格统一以后，可将对话框重新自定义，并添加默认保存类别修改功能。