1. 实现步骤：

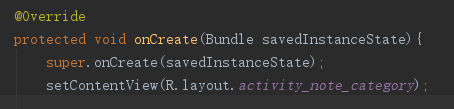
1、新建Note\_category\_main类

### 1.1、在下新建java class，命名为Note\_category\_main

### 1.2、继承AppCompatActivity

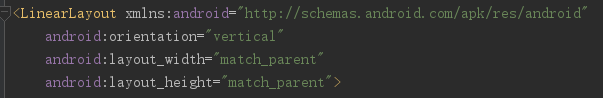


### 1.3、定义onCreate()



2、在layout目录下新建activity\_note\_category.xml

### 2.1、LinearLayout布局



### 2.2、在res/drawable-xhdpi目录下添加图片

### 2.3、在res/drawable-xxhdpi目录下添加图片

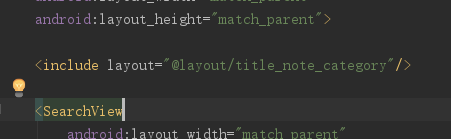


### 2.4、在layout目录下新建title\_note\_category.xml

代码：

<?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="wrap\_content"  
 android:background="#FFFFFF">  
  
 <ImageView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/image\_back"  
 android:layout\_alignParentLeft="true"  
 android:layout\_margin="10dp"  
 android:background="@drawable/back\_1"  
 android:visibility="gone"  
 />  
  
 <ImageView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/image\_arrange\_or\_select\_all"  
 android:layout\_alignParentRight="true"  
 android:layout\_margin="10dp"  
 android:src="@drawable/arrange"  
 />  
  
 <ImageView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/image\_change\_or\_add"  
 android:layout\_toLeftOf="@+id/image\_arrange\_or\_select\_all"  
 android:layout\_margin="10dp"  
 android:src="@drawable/edit\_change"/>  
  
</RelativeLayout>

### 2.5、在activity\_note\_category.xml中引用title\_note\_category.xml



### 2.6、在res/drawable目录下新建jasmine\_yellow\_shape.xml

代码：

<?xml version="1.0" encoding="utf-8"?>  
<shape  
 xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shape="rectangle">  
 <!-- 填充的颜色 -->  
 <solid android:color="#e6c35c" />  
 <!-- 设置按钮的四个角为弧形 -->  
 <!-- android:radius 弧形的半径 -->  
 <corners android:radius="6dp" />  
  
</shape>

### 2.7、在res/drawable目录下新建while\_shape.xml

代码：

<?xml version="1.0" encoding="utf-8"?>  
<shape xmlns:android="http://schemas.android.com/apk/res/android"  
 android:shape="rectangle">  
 <!-- 填充的颜色 -->  
 <solid android:color="#FFFFFF" />  
 <!-- 设置按钮的四个角为弧形 -->  
 <!-- android:radius 弧形的半径 -->  
 <corners android:radius="6dp" />  
  
</shape>

### 2.8、定义布局

代码：

<SearchView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/searchview\_page"  
 android:iconifiedByDefault="false"  
 android:queryHint="搜索页号/标题"  
 />  
  
<LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:orientation="horizontal"  
 android:id="@+id/linear\_open"  
 android:background="#fff">  
  
 <Spinner  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:id="@+id/spinner\_category"  
 android:layout\_margin="5dp"  
  
 >  
 </Spinner>  
  
 <Button  
 android:layout\_width="0dp"  
 android:layout\_height="match\_parent"  
 android:layout\_weight="1"  
 android:background="@drawable/jasmine\_yellow\_shape"  
 android:layout\_margin="5dp"  
 android:text="写笔记"  
 android:textSize="20sp"  
 android:id="@+id/btn\_open"/>  
  
</LinearLayout>  
  
<View  
 android:layout\_width="match\_parent"  
 android:layout\_height="1px"  
 android:background="#606060"/>  
  
<android.support.v7.widget.RecyclerView  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/recycler\_search"  
 android:visibility="gone"  
 android:layout\_weight="1">  
  
</android.support.v7.widget.RecyclerView>  
  
<ExpandableListView  
 android:id="@+id/expandable\_category"  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:childDivider="#193ca3"  
 android:layout\_weight="1"  
 android:focusable="false"  
 android:focusableInTouchMode="false"/>  
  
<LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:gravity="center"  
 android:id="@+id/linear\_search"  
 android:layout\_weight="1"  
 android:visibility="gone">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/search\_no\_text"  
 android:text="抱歉，没有找到该目标"  
 android:textColor="#808080"  
 android:textSize="30dp"/>  
  
</LinearLayout>  
  
<LinearLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/linear\_arrange\_dialog"  
 android:layout\_gravity="bottom"  
 android:background="#fff"  
 android:orientation="vertical"  
 android:visibility="gone"  
 >  
  
 <View  
 android:layout\_width="match\_parent"  
 android:layout\_height="1px"  
 android:background="#606060"/>  
  
 <Button  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/btn\_edit\_change"  
 android:background="@drawable/jasmine\_yellow\_shape"  
 android:text="更改"  
 android:visibility="gone"  
 android:layout\_margin="10dp"  
 android:textSize="20sp"/>  
  
 <RelativeLayout  
 android:layout\_width="match\_parent"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/relative\_arrange\_dialog"  
 android:background="#fff">  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/tv\_chosen"  
 android:layout\_centerVertical="true"  
 android:layout\_marginLeft="15dp"  
 android:text="已选择"  
 android:textSize="15sp"/>  
  
 <TextView  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/tv\_chosen\_num"  
 android:layout\_centerVertical="true"  
 android:layout\_marginLeft="9dp"  
 android:layout\_toRightOf="@+id/tv\_chosen"  
 android:text="0"  
 android:textSize="15sp"/>  
  
 <Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/btn\_delete"  
 android:layout\_alignParentRight="true"  
 android:layout\_centerVertical="true"  
 android:layout\_marginRight="15dp"  
 android:layout\_marginTop="10dp"  
 android:layout\_marginBottom="10dp"  
 android:background="@drawable/jasmine\_yellow\_shape"  
 android:gravity="center"  
 android:text="删除"  
 android:textSize="15sp"/>  
  
 <Button  
 android:layout\_width="wrap\_content"  
 android:layout\_height="wrap\_content"  
 android:id="@+id/btn\_move"  
 android:layout\_centerVertical="true"  
 android:layout\_marginRight="15dp"  
 android:layout\_toLeftOf="@+id/btn\_delete"  
 android:background="@drawable/while\_shape"  
 android:gravity="center"  
 android:text="移动"  
 android:textSize="15sp"/>  
  
 </RelativeLayout>  
  
</LinearLayout>

## 3、定义数据库操作类

### 3.1、在下新建一个package，命名为DB

### 3.2、在DB中新建java class，命名为DatabaseHelper

### 3.3、继承SQLiteOpenHelper，并初始化

代码：

public class DatabaseHelper extends SQLiteOpenHelper {  
 //记录分类名  
 private static final String *CREATE\_CATEGORY* = "create table if not exists tb\_category(id integer " +  
 "primary key autoincrement, category\_name text)";  
 //记录笔记本的最后修改时间  
 private static final String *CREATE\_BOOK\_TIME* = "create table if not exists tb\_book\_time(book\_id integer " +  
 "primary key,book\_time timestamp)";  
  
 */\*\*  
 \*  
 \** ***@param*** *context  
 \** ***@param*** *name  
 \** ***@param*** *factory 用于查询时返回Cursor的子类对象；或者传入null使用默认的factory构造  
 \** ***@param*** *version 数据库的版本号  
 \*/* public DatabaseHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version){  
 super(context,name,factory,version);  
 }  
  
 @Override  
 public void onCreate(SQLiteDatabase db) {  
 db.execSQL(*CREATE\_CATEGORY*);  
 db.execSQL(*CREATE\_BOOK\_TIME*);  
 ContentValues values = new ContentValues();  
 values.put("category\_name","最近");  
 db.insert("tb\_category",null,values);  
  
 values = new ContentValues();  
 values.put("category\_name","默认");  
 db.insert("tb\_category",null,values);  
 }  
  
 @Override  
 public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
 db.execSQL("drop table if exists category");  
 onCreate(db);  
 }  
}

## 定义Category类

### 4.1、在下新建package，命名位category

### 4.2、在category包下，新建java class，命名为Category

代码：

public class Category {  
 private String category\_name;  
 private boolean chosen\_category;  
 private List<Page> pageList = new ArrayList<Page>();  
  
 public Category(String category\_name){  
 this.category\_name = category\_name;  
 }  
  
 public Category() {  
  
 }  
  
 //使用final修饰，方法不重载，编译器内联提高性能  
 final public List<Page> getPageList() {  
 return pageList;  
 }  
  
 final public void setPageList(List<Page> pageList) {  
 this.pageList = pageList;  
 }  
  
 final public String getCategory\_name() {  
 return category\_name;  
 }  
  
 final public void setCategory\_name(String category\_name) {  
 this.category\_name = category\_name;  
 }  
  
 final public boolean isChosen\_category() {  
 return chosen\_category;  
 }  
  
 final public void setChosen\_category(boolean chosen\_category) {  
 this.chosen\_category = chosen\_category;  
 }  
  
 public static class Page implements Cloneable{  
 private String book\_id;  
 private String page\_id;  
 private String page\_name;  
 private String page\_image;  
 private boolean chosen\_page;  
  
 public Page(String book\_id,String page\_id,String page\_name,String page\_image){  
 this.book\_id = book\_id;  
 this.page\_id = page\_id;  
 this.page\_name = page\_name;  
 this.page\_image = page\_image;  
 }  
  
 public Page clone(){  
 Page page = null;  
 try{  
 page = (Page) super.clone();  
 }catch (CloneNotSupportedException e){  
 e.printStackTrace();  
 }  
 return page;  
 }  
  
 final public String getBook\_id() {  
 return book\_id;  
 }  
  
 final public String getPage\_id() {  
 return page\_id;  
 }  
  
 final public String getPage\_name() {  
 return page\_name;  
 }  
  
 final public void setPage\_name(String page\_name) {  
 this.page\_name = page\_name;  
 }  
  
 final public String getPage\_image() {  
 return page\_image;  
 }  
  
 final public boolean isChosen\_page() {  
 return chosen\_page;  
 }  
  
 final public void setChosen\_page(boolean chosen\_page) {  
 this.chosen\_page = chosen\_page;  
 }  
 }  
}

## 完善Note\_category\_main类

### 5.1、声明全局变量与对象

代码：

private static final String *TAG* = "Note\_category\_main";  
private List<Category> categoryList = new ArrayList<Category>();  
private DatabaseHelper database\_helper;  
private Spinner category\_spinner;  
private Button open\_button;  
private ExpandableListView category\_expandable;  
private ImageView back\_image;  
private TextView chosen\_num\_tv;  
private LinearLayout open\_linear;  
private LinearLayout arrange\_dialog\_linear;  
private ImageView arrange\_or\_select\_all\_image;  
private ImageView change\_or\_add\_image;  
private Button edit\_change\_btn;  
private RelativeLayout arrange\_dialog\_relative;  
private SearchView page\_searchview;  
private LinearLayout search\_linear;  
private PageAdapter pageAdapter;  
private RecyclerView search\_recycler;  
private Button move\_page\_btn;  
private Button delete\_page\_btn;  
  
public static final int *OPEN\_NOTEBOOK* = 1;  
private String spinner\_select\_name = "默认";  
private int screenHeight; //屏幕高度值  
private MyExpandable myExpandable;  
private List<Integer> chosen\_num\_list = new ArrayList<Integer>(); //各类别被选中数  
private boolean arrange\_select\_show; //选择管理启动标志  
private boolean edit\_change; //更名启动标志  
private boolean chosen\_all;  
private int chosen\_count;  
private int radio\_group;  
private int radio\_children;

### 5.2、在onCreate()下控件对象和类别数据初始化

代码：

//隐藏原标题  
ActionBar actionBar = getSupportActionBar();  
if (actionBar != null){  
 actionBar.hide();  
}  
  
database\_helper = new DatabaseHelper(this,"note\_book.db",null,1);  
initCategory\_list();

chosen\_num\_list.clear();  
int categoryCount = categoryList.size();  
for (int i = 0;i < categoryCount;i++){  
 chosen\_num\_list.add(0);  
}

category\_spinner = (Spinner)findViewById(R.id.*spinner\_category*);  
open\_button = (Button)findViewById(R.id.*btn\_open*);  
  
category\_expandable = (ExpandableListView)findViewById(R.id.*expandable\_category*);  
chosen\_num\_tv = (TextView)findViewById(R.id.*tv\_chosen\_num*);  
open\_linear = (LinearLayout)findViewById(R.id.*linear\_open*);  
arrange\_dialog\_linear = (LinearLayout)findViewById(R.id.*linear\_arrange\_dialog*);  
edit\_change\_btn = (Button)findViewById(R.id.*btn\_edit\_change*);  
arrange\_dialog\_relative = (RelativeLayout)findViewById(R.id.*relative\_arrange\_dialog*);  
  
back\_image = (ImageView)findViewById(R.id.*image\_back*);  
arrange\_or\_select\_all\_image = (ImageView)findViewById(R.id.*image\_arrange\_or\_select\_all*);  
change\_or\_add\_image = (ImageView)findViewById(R.id.*image\_change\_or\_add*);  
  
move\_page\_btn = (Button)findViewById(R.id.*btn\_move*);  
delete\_page\_btn = (Button)findViewById(R.id.*btn\_delete*);  
  
page\_searchview = (SearchView)findViewById(R.id.*searchview\_page*);  
search\_linear = (LinearLayout)findViewById(R.id.*linear\_search*);  
search\_recycler = (RecyclerView) findViewById(R.id.*recycler\_search*);  
  
//实现每行两个控件布局  
GridLayoutManager layoutManager = new GridLayoutManager(this, 2);  
search\_recycler.setLayoutManager(layoutManager);

### 5.3、定义initCategory\_list()类别数据初始化函数

代码：

private void initCategory\_list() {

//返回读写数据库的对象  
 SQLiteDatabase db = database\_helper.getWritableDatabase();  
 try{  
 categoryList.clear();  
 Cursor cursor\_category = db.query("tb\_category",new String[] { "category\_name" },null,null  
 ,null,null,null);  
  
 if (cursor\_category.moveToFirst()){  
 do {  
 String category\_name = cursor\_category.getString(0);  
 Category category = new Category(category\_name);  
  
 if (category\_name.equals("最近")) {  
 Cursor cursor\_book\_time = db.query("tb\_book\_time", new String[]{"book\_id"}, null  
 , null, null, null, "book\_time", "3");  
  
 if (cursor\_book\_time.moveToFirst()) {  
  
 List<Category.Page> pageList = new ArrayList<Category.Page>();  
 List<Date> dateList = new ArrayList<Date>();  
 do {  
 int book\_id = cursor\_book\_time.getInt(0);  
 Cursor cursor\_book = db.query("tb\_" + String.*valueOf*(book\_id), new String[]  
 {"page\_id", "page\_time", "page\_name", "page\_image"}, null  
 , null, null, null, "page\_time desc", "3");  
  
 if (cursor\_book.moveToFirst()) {  
 do {  
 String time = cursor\_book.getString(1);  
 Log.*d*(*TAG*, "initCategory\_list: time:" + time);  
 SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss:SSS");  
 Date date = format.parse(time);  
 int num = pageList.size();  
 if (num < 3 || dateList.get(num - 1).getTime() < date.getTime()) {  
 String page\_id = cursor\_book.getString(0);  
 String page\_name = cursor\_book.getString(2);  
 String page\_image = cursor\_book.getString(3);  
 Category.Page page = new Category.Page(String.*valueOf*(book\_id),page\_id, page\_name  
 , page\_image);  
 if (num < 3) {  
 pageList.add(page);  
 dateList.add(date);  
  
 } else {  
 int count = 1;  
 while (count < 4) {  
 if (count != 1) {  
 if (dateList.get(num - count).getTime() > date.getTime()) {  
 break;  
 }  
 Category.Page new\_page = (Category.Page) pageList.get(num - count).clone();  
 pageList.set(num - count - 1, new\_page);  
 Date new\_date = (Date) dateList.get(num - count).clone();  
 dateList.set(num - count - 1, new\_date);  
 }  
 pageList.set(num - count, page);  
 dateList.set(num - count, date);  
 count++;  
 }  
 }  
 }  
 } while (cursor\_book.moveToNext());  
  
 }  
 cursor\_book.close();  
 } while (cursor\_book\_time.moveToNext());  
  
 category.setPageList(pageList);  
 }  
 cursor\_book\_time.close();  
 } else {  
 //初始化非“最近”类别数据  
 Cursor cursor\_book\_time = db.query("tb\_book\_time", new String[]{"book\_id"}, null  
 , null, null, null, "book\_time", null);  
 if (cursor\_book\_time.moveToFirst()) {  
 do {  
 int book\_id = cursor\_book\_time.getInt(0);  
 Cursor cursor\_book = db.query("tb\_" + String.*valueOf*(book\_id), new String[]  
 {"page\_id", "page\_name", "page\_image"}, "category\_name = ?"  
 , new String[]{category\_name}, null, null, null);  
 if (cursor\_book.moveToFirst()){  
 do {  
 String page\_id = cursor\_book.getString(0);  
 String page\_name = cursor\_book.getString(1);  
 String page\_image = cursor\_book.getString(2);  
 Category.Page page = new Category.Page(String.*valueOf*(book\_id),page\_id,page\_name,page\_image);  
 category.getPageList().add(page);  
 }while (cursor\_book.moveToNext());  
  
 }  
 cursor\_book.close();  
 } while (cursor\_book\_time.moveToNext());  
 }  
 cursor\_book\_time.close();  
 }  
  
 categoryList.add(category);  
 } while (cursor\_category.moveToNext());  
  
 }  
 cursor\_category.close();  
 } catch (Exception e){  
 e.printStackTrace();  
 } finally {  
 db.close();  
 }  
  
 }

### 5.4、在onCreate()中实现下拉列表功能

代码：

ArrayAdapter<String> arrayAdapter\_spinner = new ArrayAdapter<String>(this,R.layout.*support\_simple\_spinner\_dropdown\_item*,getCategory\_name());  
 category\_spinner.setAdapter(arrayAdapter\_spinner);  
 category\_spinner.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {  
 @Override  
 public void onItemSelected(AdapterView<?> adapterView, View view, int i, long l) {  
 spinner\_select\_name = categoryList.get(i + 1).getCategory\_name();  
 }  
  
 @Override  
 public void onNothingSelected(AdapterView<?> adapterView) {  
  
 }  
 });

### 5.5、定义getCategory\_name\_list()类别名集合获取函数

代码：

public ArrayList<String> getCategory\_name\_list() {  
 ArrayList<String> list = new ArrayList<String>();  
 int categoryCount = categoryList.size();  
 for (int i = 1;i < categoryCount;i++){  
 list.add(categoryList.get(i).getCategory\_name());  
 }  
 return list;  
}

### 5.6、在onCreate()中实现笔记本常规打开功能

代码：

open\_button.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View view) {  
 Intent intent = new Intent(Note\_category\_main.this,OidActivity.class);  
 intent.putExtra("DESIGNATION",false);  
 intent.putExtra("CATEGORY\_NAME",spinner\_select\_name);  
 intent.putExtra("ALLCATEGORY\_NAME",getCategory\_name\_array());  
 }  
});

### 5.7、定义getCategory\_name\_array()获取类别名数组函数

代码：

private String[] getCategory\_name\_array() {  
 int category\_count = categoryList.size();  
 StringBuilder stringBuilder;  
 */\*\*  
 \* StringBuilder 是一个可以动态增加自身数据长度的类，其默认长度（capacity属性）为16。  
 \* 长度超过16，StringBuilder每次自动申请一个新的两倍的长度空间，并将原存储空间的数据复制到新申请空间，性能降低。  
 \*  
 \* 假设每个类名长度为2，类名字数当超过16的3/4时，使用自定义长度。  
 \*/* if (category\_count > 6){  
 stringBuilder = new StringBuilder(category\_count << 1);  
 } else {  
 stringBuilder = new StringBuilder();  
 }  
  
 for (int i = 1; i < category\_count; i++) {  
 stringBuilder.append(categoryList.get(i).getCategory\_name() + " ");  
 }  
 return new String(stringBuilder).split(" ");  
}

### 5.8、在onCreate()中实现各分类笔记初始化和显示

代码：

DisplayMetrics displayMetrics = new DisplayMetrics();  
getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);  
screenHeight = displayMetrics.heightPixels;  
  
myExpandable = new MyExpandable(categoryList,Note\_category\_main.this,screenHeight);  
category\_expandable.setAdapter(myExpandable);  
category\_expandable.expandGroup(0);

### 5.9、定义分类显示的ExpandableList适配器

#### **5.9.1、在category包下新建MyExpandable类**

代码：

public class MyExpandable extends BaseExpandableListAdapter {  
 private Context context;  
 private List<Category> categoryList = new ArrayList<Category>();  
 private int screenHeight;  
 private boolean show\_select;  
 private boolean edit\_change;  
  
 public void setCategoryList(List<Category> categoryList){  
 this.categoryList = categoryList;  
 }  
  
 public void setIsEdit\_change(boolean edit\_change){  
 this.edit\_change = edit\_change;  
 }  
  
 public void setIsShow\_select(boolean show\_select){  
 this.show\_select = show\_select;  
 }  
  
 public MyExpandable(List<Category> categoryList,Context context,int screenHeight){  
 this.categoryList = categoryList;  
 this.context = context;  
 this.screenHeight = screenHeight;  
 }  
  
 public interface onItemClickListener{  
 boolean onChildrenLongClick(View view,int groupPosition,int pos);  
 void onChildrenClick(View view, int groupPosition, int pos, PageAdapter pageAdapter, List<Category.Page> pageList);  
 void onGroupClick(View view, int pos, Category category);  
 }  
  
 private onItemClickListener onItemClickListener;  
  
 public void setOnItemClickListener(MyExpandable.onItemClickListener onItemClickListener){  
 this.onItemClickListener = onItemClickListener;  
 }  
  
 private static class GroupViewHolder{  
 private TextView group\_name;  
 private ImageView group\_select\_image;  
 }  
  
 private static class ChildViewHolder{  
 private RecyclerView child\_recycler;  
 }  
  
 @Override  
 public int getGroupCount() {  
 return categoryList.size();  
 }  
  
 @Override  
 public int getChildrenCount(int groupPosition) {  
 if (categoryList.get(groupPosition).getPageList() == null){  
 return 0;  
 }else {  
 return 1;  
 }  
  
 }  
  
 @Override  
 public Object getGroup(int groupPosition) {  
 return categoryList.get(groupPosition);  
 }  
  
 @Override  
 public Object getChild(int groupPosition, int childPosition) {  
 return categoryList.get(groupPosition).getPageList();  
 }  
  
 @Override  
 public long getGroupId(int groupPosition) {  
 return groupPosition;  
 }  
  
 @Override  
 public long getChildId(int groupPosition, int childPosition) {  
 return childPosition;  
 }  
  
 */\*\*  
 \* 用来判断ExpandableListView内容id是否有效的(返回true or false)  
 \* ，系统会跟据id来确定当前显示哪条内容，也就是firstVisibleChild的位置。  
 \*  
 \* 分组和子选项是否持有稳定的ID, 就是说底层数据的改变会不会影响到它们  
 \*/*  
 @Override  
 public boolean hasStableIds() {  
 return false;  
 }  
  
 @Override  
 public View getGroupView(final int groupPosition, boolean isExpanded, View convertView, ViewGroup parent) {  
 final GroupViewHolder groupHolder;  
 if(convertView == null){  
 convertView = LayoutInflater.*from*(context).inflate(  
 R.layout.*item\_group*, parent, false);  
 groupHolder = new GroupViewHolder();  
 groupHolder.group\_name = (TextView) convertView.findViewById(R.id.*text\_group\_name*);  
 groupHolder.group\_select\_image = (ImageView)convertView.findViewById(R.id.*image\_group*);  
  
 //进行缓存  
 convertView.setTag(groupHolder);  
 }else{  
 groupHolder = (GroupViewHolder)convertView.getTag();  
 }  
  
 final Category category = categoryList.get(groupPosition);  
 groupHolder.group\_name.setText(category.getCategory\_name());  
  
 if (show\_select){  
 groupHolder.group\_select\_image.setVisibility(View.*VISIBLE*);  
 if(!edit\_change) {

//判断是否出现全选  
 for (int i = 0; i < category.getPageList().size(); i++) {  
 Category.Page page = category.getPageList().get(i);  
 boolean isSelelct = page.isChosen\_page();  
 if (isSelelct) {  
 category.setChosen\_category(true);  
 } else {  
 category.setChosen\_category(false);  
 break;  
 }  
 }  
 }  
  
 boolean group\_select = category.isChosen\_category();  
  
 if (group\_select){  
 groupHolder.group\_select\_image.setImageResource(R.drawable.*select*);  
 }  
 else {  
 groupHolder.group\_select\_image.setImageResource(R.drawable.*no\_select*);  
 }  
  
 groupHolder.group\_select\_image.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View view){  
  
 onItemClickListener.onGroupClick(view,groupPosition,category);  
 }  
 });  
 }else {  
 groupHolder.group\_select\_image.setVisibility(View.*GONE*);  
 }  
  
 return convertView;  
 }  
  
 @Override  
 public View getChildView(final int groupPosition, int childPosition, boolean isLastChild, View convertView, ViewGroup parent) {  
 ChildViewHolder childViewHolder;  
 if(convertView == null){  
 convertView = LayoutInflater.*from*(context).inflate(R.layout.*item\_child*,parent,false);  
 childViewHolder = new ChildViewHolder();  
 childViewHolder.child\_recycler = (RecyclerView)convertView.findViewById(R.id.*recycler\_child*);  
 GridLayoutManager layoutManager = new GridLayoutManager(context, 2);  
 childViewHolder.child\_recycler.setLayoutManager(layoutManager);  
  
 convertView.setTag(childViewHolder);  
 }else {  
 childViewHolder = (ChildViewHolder)convertView.getTag();  
 }  
  
 final List<Category.Page> list\_page = categoryList.get(groupPosition).getPageList();  
 final PageAdapter adapter = new PageAdapter(list\_page,screenHeight,show\_select);  
 childViewHolder.child\_recycler.setAdapter(adapter);  
  
 adapter.setOnItemClickListener(new PageAdapter.onItemClickListener() {  
 @Override  
 public void onClick(View view, int pos, List<Category.Page> pageList) {  
  
 onItemClickListener.onChildrenClick(view,groupPosition,pos,adapter,pageList);  
 }  
  
 @Override  
 public boolean onLongClick(View view, int pos, List<Category.Page> pageList) {  
  
 return onItemClickListener.onChildrenLongClick(view,groupPosition,pos);  
 }  
 });  
  
 return convertView;  
 }  
  
 //用来判断某Group某个child是否可可选。  
 @Override  
 public boolean isChildSelectable(int groupPosition, int childPosition) {  
 return true;  
 }  
}

#### 5.9.2、在category包下新建PageAdapter类

代码：

public class PageAdapter extends RecyclerView.Adapter<PageAdapter.MyViewHolder> {  
 private List<Category.Page> pageList = new ArrayList<Category.Page>();  
 private int screenHeight;  
 private Context context;  
 private boolean show\_select;  
// private static final String TAG = "PageAdapter";  
  
 final public void setPageList(List<Category.Page> pageList){  
 this.pageList = pageList;  
 }  
  
 final public void setIsShow\_select(boolean show\_select){  
 this.show\_select = show\_select;  
 }  
  
 public PageAdapter(List<Category.Page> pageList,int screenHeight,boolean show\_select){  
 this.pageList = pageList;  
 this.screenHeight = screenHeight;  
 this.show\_select = show\_select;  
 }  
  
 @Override  
 public MyViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {  
 if (context == null){  
 context = parent.getContext();  
 }  
 View view = LayoutInflater.*from*(context).inflate(R.layout.*item\_page*,parent,false);  
 final MyViewHolder holder = new MyViewHolder(view);  
 return holder;  
 }  
  
 @Override  
 public void onBindViewHolder(MyViewHolder holder, final int position) {  
 Category.Page page = pageList.get(position);  
 String page\_id = String.*valueOf*(Integer.*parseInt*(page.getPage\_id()) + 1);  
 String page\_name = page.getPage\_name();  
 //页名显示处理  
 if (page\_name == null){  
 page\_name = "";  
 } else {  
 page\_name = "," + page\_name;  
 }  
  
 holder.page\_name.setText(page\_id + page\_name);  
 //drawable文件下对应默认图片  
 String page\_image = "p" + page.getPage\_id();  
 final int resources\_id = context.getResources().getIdentifier(page\_image,"drawable",context.getPackageName());  
  
 if (resources\_id != 0) {  
// Log.d(TAG, "onBindViewHolder: ");  
 */\*\*  
 \* with()方法可以接收Context、Activity或者Fragment类型的参数，可用范围非常广。  
 \* load()方法用于指定待加载的图片资源。Glide支持加载各种各样的图片资源，包括网络图片、本地图片、应用资源、二进制流、Uri对象等等。  
 \* apply()自定义  
 \* error()请求图片加载错误占位图。  
 \* into()方法希望让图片显示在哪个ImageView上，把这个ImageView的实例传进去就可以了。  
 \*/* Glide.*with*(context).load(page.getPage\_image()).apply(new RequestOptions().error(resources\_id)).into(holder.page\_image);  
 }  
 holder.page\_image.setMaxHeight(screenHeight / 3);  
  
 if (show\_select){  
 holder.select\_image.setVisibility(View.*VISIBLE*);  
  
 final boolean page\_select = page.isChosen\_page();  
  
 if (page\_select){  
 holder.select\_image.setImageResource(R.drawable.*select*);  
 }  
 else {  
 holder.select\_image.setImageResource(R.drawable.*no\_select*);  
 }  
 }  
 else {  
 holder.select\_image.setVisibility(View.*GONE*);  
 }  
  
 holder.cardView.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View v){  
  
 onItemClickListener.onClick(v,position,pageList);  
 }  
 });  
  
 holder.cardView.setOnLongClickListener(new View.OnLongClickListener() {  
 @Override  
 public boolean onLongClick(View v) {  
 return onItemClickListener.onLongClick(v,position,pageList);  
 }  
 });  
 }  
  
 public interface onItemClickListener{  
 void onClick(View view, int pos, List<Category.Page> pageList);  
  
 boolean onLongClick(View view, int pos, List<Category.Page> pageList);  
 }  
  
 private onItemClickListener onItemClickListener;  
  
 public void setOnItemClickListener(PageAdapter.onItemClickListener onItemClickListener){  
 this.onItemClickListener = onItemClickListener;  
 }  
  
 @Override  
 public int getItemCount() {  
  
 return pageList.size();  
 }  
  
 static class MyViewHolder extends RecyclerView.ViewHolder{  
 CardView cardView;  
 ImageView page\_image;  
 TextView page\_name;  
 ImageView select\_image;  
 public MyViewHolder(View view){  
 super(view);  
 cardView = (CardView)view;  
 page\_image = (ImageView)view.findViewById(R.id.*image\_page*);  
 page\_name = (TextView)view.findViewById(R.id.*tv\_page\_name*);  
 select\_image = (ImageView)view.findViewById(R.id.*image\_select\_page*);  
 }  
 }  
}

### 5.10、在onCreate()中对ExpandableList中的三个分类显示点击事件接口进行实现

代码：

myExpandable.setOnItemClickListener(new MyExpandable.onItemClickListener() {  
 @Override  
 public boolean onChildrenLongClick(View view, int groupPosition, int pos) {  
 if (!arrange\_select\_show){  
 arrange\_or\_change\_show();  
 categoryList.get(groupPosition).getPageList().get(pos).setChosen\_page(true);  
 int num = chosen\_num\_list.get(groupPosition) + 1;  
 chosen\_num\_list.set(groupPosition,num);  
 addChosen\_count(1);  
 } else {  
 arrange\_quit();  
 back\_image.setVisibility(View.*GONE*);  
 }  
 return false;  
 }  
  
 @Override  
 public void onChildrenClick(View view, int groupPosition, int pos, PageAdapter adapter, List<Category.Page> adapter\_pageList) {  
 if (arrange\_select\_show) {  
 List<Category.Page> pageList = categoryList.get(groupPosition).getPageList();  
 boolean isChild\_select = pageList.get(pos).isChosen\_page();  
 pageList.get(pos).setChosen\_page(!isChild\_select);  
 if (edit\_change) {  
 boolean one\_group = false;  
  
 if (groupPosition == radio\_group && radio\_children != -1){  
 one\_group = true;  
 }  
 radio\_change(groupPosition,pos);  
 if(one\_group){  
 //重绘当前可见区域  
 adapter.notifyDataSetChanged();  
 } else {  
 myExpandable.notifyDataSetChanged();  
 }  
 } else {  
  
 int num = chosen\_num\_list.get(groupPosition);  
 if (!isChild\_select) {  
 addChosen\_count(1);  
 num++;  
 if (pageList.size() == num) {  
 myExpandable.notifyDataSetChanged();  
 } else {  
 adapter.notifyDataSetChanged();  
 }  
  
 } else {  
  
 reduceChosen\_count(1);  
 if (pageList.size() == num) {  
 num--;  
 myExpandable.notifyDataSetChanged();  
 } else {  
 num--;  
 adapter.notifyDataSetChanged();  
 }  
 }  
 chosen\_num\_list.set(groupPosition, num);  
 }  
 } else {  
 Category.Page page = adapter\_pageList.get(pos);  
 Intent intent = new Intent(Note\_category\_main.this, OidActivity.class);  
 intent.putExtra("DESIGNATION", true);  
 intent.putExtra("BOOKID", 100);  
 intent.putExtra("PAGEID", Integer.*parseInt*(page.getPage\_id()));  
 startActivityForResult(intent,*OPEN\_NOTEBOOK*);  
  
 }  
 }  
  
 @Override  
 public void onGroupClick(View view, int pos, Category group\_category) {  
 boolean group\_select\_onclick = group\_category.isChosen\_category();  
 group\_category.setChosen\_category(!group\_select\_onclick);  
 if(edit\_change) {

//打开指定页面  
 radio\_change(pos,-1);  
  
 } else {  
 List<Category.Page> pageList = group\_category.getPageList();  
 for (int i = 0; i < pageList.size(); i++) {  
 pageList.get(i).setChosen\_page(!group\_select\_onclick);  
 }  
 if (!group\_select\_onclick) {  
 addChosen\_count(pageList.size());  
 chosen\_num\_list.set(pos, pageList.size());  
  
 } else {  
 reduceChosen\_count(pageList.size());  
 chosen\_num\_list.set(pos, 0);  
 }  
 }  
 myExpandable.notifyDataSetChanged();  
 }  
});

### 5.11、在onCreate()中实现查询搜索功能

代码：

page\_searchview.setFocusable(false);  
page\_searchview.setOnQueryTextListener(new SearchView.OnQueryTextListener() {  
 @Override  
 public boolean onQueryTextSubmit(String query) {  
 return false;  
 }  
  
 @Override  
 public boolean onQueryTextChange(String newText) {  
 if (!TextUtils.*isEmpty*(newText)){  
 category\_expandable.setVisibility(View.*GONE*);  
 final List<Category.Page> pageList = search\_page(newText);  
  
 if (pageList.size() > 0){  
 search\_result\_view(pageList);  
 } else {  
 search\_recycler.setVisibility(View.*GONE*);  
 search\_linear.setVisibility(View.*VISIBLE*);  
 }  
 } else {  
 search\_linear.setVisibility(View.*GONE*);  
 search\_recycler.setVisibility(View.*GONE*);  
 category\_expandable.setVisibility(View.*VISIBLE*);  
 if (edit\_change){  
 search\_reset\_radio();  
 }  
 pageAdapter = null;  
 }  
 return false;  
 }  
});

### 5.12、在onCreate()中实现更名和类别添加点击监听

代码：

change\_or\_add\_image.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (arrange\_select\_show){  
 change\_or\_add\_image.setImageResource(R.drawable.*add\_2*);  
 show\_add\_or\_change\_dialog();  
 } else {  
 edit\_change = true;  
 */\*\*  
 \* radio\_children = -1;radio\_group = -1 : 初始化状态  
 \* radio\_children != -1;radio\_group = -1 : 搜索功能下页面单选选中  
 \* radio\_children != -1;radio\_group != -1 : 页面单选选中  
 \* radio\_children = -1;radio\_group != -1 : 类别单选选中  
 \*/* radio\_children = -1;  
 radio\_group = -1;  
 arrange\_or\_change\_show();  
 }  
 }  
});

### 5.13、定义arrange\_or\_change\_show()整理和更名功能初始化函数

代码：

private void arrange\_or\_change\_show() {  
 arrange\_select\_show = true;  
 if (pageAdapter != null){  
 pageAdapter.setIsShow\_select(true);  
 pageAdapter.notifyDataSetChanged();  
 }  
 myExpandable.setIsShow\_select(true);  
 back\_image.setVisibility(View.*VISIBLE*);  
 back\_image.setImageResource(R.drawable.*back\_1*);  
 chosen\_num\_tv.setText(String.*valueOf*(0));  
 open\_linear.setVisibility(View.*GONE*);  
 arrange\_dialog\_linear.setVisibility(View.*VISIBLE*);  
 if (edit\_change){  
 arrange\_or\_select\_all\_image.setVisibility(View.*GONE*);  
 change\_or\_add\_image.setVisibility(View.*GONE*);  
 myExpandable.setIsEdit\_change(true);  
 arrange\_dialog\_relative.setVisibility(View.*GONE*);  
 edit\_change\_btn.setVisibility(View.*VISIBLE*);  
 myExpandable.notifyDataSetChanged();  
  
 edit\_change\_btn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if(edit\_change && (radio\_group != -1 || radio\_children != -1)){  
 if (radio\_group == -1){  
 reset\_search\_radio();  
 }  
 if (radio\_group < 2 && radio\_children == -1){  
 Toast.*makeText*(Note\_category\_main.this,"“最近”和“默认”类别不能更改",Toast.*LENGTH\_SHORT*).show();  
 }else{  
 show\_add\_or\_change\_dialog();  
 }  
  
 } else {  
 Toast.*makeText*(Note\_category\_main.this,"请选择要更改的笔记",Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
 } else {  
 //进入笔记分类整理功能  
 chosen\_count = 0;  
 chosen\_all = false;  
 arrange\_or\_select\_all\_image.setImageResource(R.drawable.*select\_all*);  
  
 change\_or\_add\_image.setImageResource(R.drawable.*add\_1*);  
  
 edit\_change\_btn.setVisibility(View.*GONE*);  
 arrange\_dialog\_relative.setVisibility(View.*VISIBLE*);  
  
 move\_page\_btn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if(arrange\_select\_show && chosen\_count > 0) {  
   
 show\_move\_dialog();  
 } else {  
 Toast.*makeText*(Note\_category\_main.this,"请选择要移动的笔记",Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 });  
  
 delete\_page\_btn.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 if (arrange\_select\_show){  
 boolean hasChosen = false;  
 if (chosen\_count > 0){  
 show\_delete\_dialog();  
 hasChosen = true;  
 }else {  
 for (Category category : categoryList){  
 if (category.isChosen\_category()){  
 show\_delete\_dialog();  
 hasChosen = true;  
 break;  
 }  
 }  
 }  
 if (!hasChosen){  
 Toast.*makeText*(Note\_category\_main.this, "请选择要删除的选项", Toast.*LENGTH\_SHORT*).show();  
 }  
 }  
 }  
 });  
 }  
  
 back\_image.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View view){  
 if (arrange\_select\_show){  
 back\_image.setImageResource(R.drawable.*back\_2*);  
 new Thread() {  
 public void run(){  
 SystemClock.*sleep*(150);  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 back\_image.setVisibility(View.*GONE*);  
 }  
 });  
 }  
 }.start();  
  
 arrange\_or\_change\_quit();  
 }  
  
 }  
 });  
 }

### 5.14、定义reset\_search\_radio()重置查询功能下所需更改页面的单选标记组号和子号函数

代码：

private void reset\_search\_radio() {  
 if(radio\_group == -1 && radio\_children != -1){  
 for (int i = 1;i < categoryList.size();i++){  
 List<Category.Page> pageList = categoryList.get(i).getPageList();  
 for (int j = 0;j < pageList.size();j++){  
 if (pageList.get(j).isChosen\_page()){  
 radio\_group = i;  
 radio\_children = j;  
 myExpandable.notifyDataSetChanged();  
 break;  
 }  
 }  
 }  
 }  
}

### 5.15、定义show\_add\_or\_change\_dialog()类别添加和更名对话框功能函数

private void show\_add\_or\_change\_dialog() {  
 View view = View.*inflate*(this,R.layout.*edit\_two\_btn\_dialog*,null);  
 TextView edit\_message = (TextView)view.findViewById(R.id.*tv\_edit\_message*);  
 final EditText name\_edit = (EditText)view.findViewById(R.id.*edit\_name*);  
 name\_edit.setText("");  
 final TextView confirm\_edit\_tv = (TextView)view.findViewById(R.id.*tv\_edit\_confirm*);  
 final TextView cancel\_edit\_tv = (TextView)view.findViewById(R.id.*tv\_edit\_cancel*);  
  
 final AlertDialog.Builder builder = new AlertDialog.Builder(Note\_category\_main.this);  
  
 //true:类别被选中；false:页面被选中  
 boolean group\_or\_children = false;  
 if (edit\_change){  
 String name = null;  
  
 if (radio\_children == -1){  
 if(radio\_group != -1){  
 name = categoryList.get(radio\_group).getCategory\_name();  
 group\_or\_children = true;  
 }  
 } else if(radio\_group != -1){  
 name = categoryList.get(radio\_group).getPageList().get(radio\_children).getPage\_name();  
 group\_or\_children = false;  
  
 }  
 edit\_message.setText("将“" + name + "”目标更改为：");  
 } else {  
 edit\_message.setText("添加分类类别");  
 }  
  
 builder.setView(view);  
 //按返回键不能退出  
 builder.setCancelable(false);  
 builder.setOnKeyListener(keyListener);  
  
 final AlertDialog dialog = builder.create();  
 final boolean finalGroup\_or\_children = group\_or\_children;  
  
 confirm\_edit\_tv.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View view){  
 confirm\_edit\_tv.setBackgroundColor(Color.*parseColor*("#C6B36E"));  
 confirm\_edit\_tv.setTextColor(Color.*parseColor*("#ffffff"));  
 String content = name\_edit.getText().toString();  
  
 boolean noConform = false;  
 if (content.contains(" ")){  
 noConform = true;  
 Toast.*makeText*(Note\_category\_main.this, "内容不可包含空格", Toast.*LENGTH\_SHORT*).show();  
 } else {  
 for (int i = 0;i < chosen\_num\_list.size();i++){  
 if (categoryList.get(i).getCategory\_name().equals(content)){  
 noConform = true;  
 Toast.*makeText*(Note\_category\_main.this, "该类别名称已存在", Toast.*LENGTH\_SHORT*).show();  
 break;  
 }  
 }  
 }  
  
 if (noConform){  
 name\_edit.setText("");  
 new Thread() {  
 public void run() {  
 SystemClock.*sleep*(150);  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 confirm\_edit\_tv.setBackgroundColor(Color.*parseColor*("#ffffff"));  
 confirm\_edit\_tv.setTextColor(Color.*parseColor*("#C6B36E"));  
 }  
 });  
 }  
 }.start();  
 } else {  
 SQLiteDatabase db = database\_helper.getWritableDatabase();  
 if (edit\_change) {  
 if (finalGroup\_or\_children) {  
 */\*\*  
 \* 使用SQLiteDatabase的beginTransaction()方法可以开启一个事务，程序执行到endTransaction() 方法时会检查事务的标志是否为成功  
 \* ，如果程序执行到endTransaction()之前调用了setTransactionSuccessful() 方法设置事务的标志为成功则提交事务  
 \* ，如果没有调用setTransactionSuccessful() 方法则回滚事务。  
 \*/* db.beginTransaction();  
 try {  
 ContentValues values = new ContentValues();  
 values.put("category\_name", content);  
 db.update("tb\_category", values, "category\_name = ?"  
 , new String[]{ categoryList.get(radio\_group).getCategory\_name() });  
 Log.*d*(*TAG*, "onClick: id =:category\_name::::::::" + radio\_group + "===" + categoryList.get(radio\_group).getCategory\_name());  
  
 Cursor cursor\_book\_time = db.query("tb\_book\_time", new String[]{ "book\_id" }, null  
 , null, null, null, "book\_time", null);  
 if (cursor\_book\_time.moveToFirst()) {  
 do {  
 int book\_id = cursor\_book\_time.getInt(0);  
  
 db.update("tb\_" + String.*valueOf*(book\_id), values, "category\_name = ?"  
 , new String[]{ categoryList.get(radio\_group).getCategory\_name() });  
 } while (cursor\_book\_time.moveToNext());  
 }  
 cursor\_book\_time.close();  
 db.setTransactionSuccessful();  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 db.endTransaction();  
 }  
  
 categoryList.get(radio\_group).setCategory\_name(content);  
 categoryList.get(radio\_group).setChosen\_category(false);  
 } else {  
 //页面更名  
 Category.Page page = categoryList.get(radio\_group).getPageList().get(radio\_children);  
  
 ContentValues values = new ContentValues();  
 values.put("page\_name", content);  
 db.update("tb\_" + page.getBook\_id(), values, "page\_id = ?", new String[]{ page.getPage\_id() });  
  
 page.setPage\_name(content);  
 page.setChosen\_page(false);

String pageId = page.getPage\_id();  
 String pageBookId = page.getBook\_id();  
  
 update\_page:  
 for (int i = 1;i < categoryList.size();i++){  
 List<Category.Page> pageList = categoryList.get(i).getPageList();  
 Log.*d*(*TAG*, "onClick: update\_page:=========" + i);  
 for (int j = 0;j < pageList.size();j++){  
 if (pageList.get(j).getPage\_id().equals(pageId) && pageList.get(j).getBook\_id().equals(pageBookId)){

pageList.get(j).setPage\_name(content);  
 break update\_page;  
 }  
   
 }  
 }  
 }  
  
 radio\_group = -1;  
 radio\_children = -1;  
  
 if (pageAdapter != null) {  
 //以当前搜索栏输入值重新检索页面  
 pageAdapter.setPageList(search\_page(page\_searchview.getQuery().toString()));  
 pageAdapter.notifyDataSetChanged();  
 }  
 categoryList.get(0).setPageList(resetRecent\_list(categoryList));  
  
 } else {  
 //类别添加  
 Category category = new Category(content);  
 categoryList.add(category);  
 chosen\_num\_list.add(0);  
 ContentValues values = new ContentValues();  
 values.put("category\_name", content);  
 db.insert("tb\_category", null, values);  
 }  
 db.close();  
  
 myExpandable.notifyDataSetChanged();  
  
 ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(Note\_category\_main.this,  
 R.layout.*support\_simple\_spinner\_dropdown\_item*, getCategory\_name\_list());  
 category\_spinner.setAdapter(arrayAdapter);  
  
 new Thread() {  
 public void run() {  
 SystemClock.*sleep*(300);  
 if (!edit\_change) {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 change\_or\_add\_image.setImageResource(R.drawable.*add\_1*);  
 }  
 });  
 }  
 dialog.dismiss();  
 }  
 }.start();  
 }  
 }  
 });  
  
 cancel\_edit\_tv.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View view){  
 cancel\_edit\_tv.setBackgroundColor(Color.*parseColor*("#C6B36E"));  
 cancel\_edit\_tv.setTextColor(Color.*parseColor*("#ffffff"));  
  
 new Thread() {  
 public void run(){  
 SystemClock.*sleep*(300);  
 if (!edit\_change) {  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 change\_or\_add\_image.setImageResource(R.drawable.*add\_1*);  
 }  
 });  
 }  
 dialog.dismiss();  
 }  
 }.start();  
 }  
 });  
  
 dialog.show();  
}  
  
//按键监听  
DialogInterface.OnKeyListener keyListener = new DialogInterface.OnKeyListener() {  
 @Override  
 public boolean onKey(DialogInterface dialog, int keyCode, KeyEvent event) {  
 if (keyCode == KeyEvent.*KEYCODE\_BACK* && event.getRepeatCount() == 0){  
 return true;  
 } else {  
 return false;  
 }  
 }  
};

### 5.16、定义search\_page()页面检索功能函数

代码：

private List<Category.Page> search\_page(String newText) {  
 final List<Category.Page> pageList = new ArrayList<Category.Page>();  
 for (int i = 1;i < categoryList.size();i++){  
 if (categoryList.get(i).getPageList() != null) {  
 int page\_count = categoryList.get(i).getPageList().size();  
 for (int j = 0;j < page\_count; j++) {  
 Category.Page page = categoryList.get(i).getPageList().get(j);  
  
 String page\_id = String.*valueOf*(Integer.*parseInt*(page.getPage\_id()) + 1);  
 if (page\_id.contains(newText) || (page.getPage\_name() != null && page.getPage\_name().contains(newText))) {  
  
 pageList.add(page);  
 if (edit\_change && page.isChosen\_page()) {  
 radio\_group = -1;  
 radio\_children = pageList.size() - 1;  
 }  
 }  
 }  
 }  
 }  
 return pageList;  
 }

### 5.17、定义resetRecent\_list()“最近”类别重置函数

代码：

private List<Category.Page> resetRecent\_list(List<Category> new\_categoryList) {  
 List<Category.Page> recent\_pageList = new\_categoryList.get(0).getPageList();  
 List<Category.Page> new\_recent\_pageList = new ArrayList<Category.Page>();  
 for (int i = 0;i < recent\_pageList.size();i++){  
 for (int j = 1;j < new\_categoryList.size();j++){  
 List<Category.Page> pageList = new\_categoryList.get(j).getPageList();  
 for (int k = 0;k < pageList.size();k++){  
 if(pageList.get(k).getPage\_id().equals(recent\_pageList.get(i).getPage\_id()) && pageList.get(k).getBook\_id()  
 .equals(recent\_pageList.get(i).getBook\_id())){  
 new\_recent\_pageList.add(pageList.get(k).clone());  
 }  
 }  
 }  
 }  
 return new\_recent\_pageList;  
}

### 5.18、定义addChosen\_count()页面计数增长函数

代码：

public void addChosen\_count(int num){  
 chosen\_count += num;  
 chosen\_num\_tv.setText(String.*valueOf*(chosen\_count));  
}

### 5.19、定义arrange\_or\_change\_quit()整理和更名功能退出函数

代码：

private void arrange\_or\_change\_quit() {  
 arrange\_select\_show = false;  
 if (pageAdapter != null){  
 pageAdapter.setIsShow\_select(false);  
 pageAdapter.notifyDataSetChanged();  
 }  
 myExpandable.setIsShow\_select(arrange\_select\_show);  
 if (edit\_change){  
 edit\_change = false;  
 myExpandable.setIsEdit\_change(false);  
 arrange\_or\_select\_all\_image.setVisibility(View.*VISIBLE*);  
 change\_or\_add\_image.setVisibility(View.*VISIBLE*);  
 if (radio\_group != -1){  
 if (radio\_children != -1){  
 categoryList.get(radio\_group).getPageList().get(radio\_children).setChosen\_page(false);  
 } else {  
 categoryList.get(radio\_group).setChosen\_category(false);  
 }  
 } else if (radio\_children != -1){  
 reset\_search\_radio();  
 categoryList.get(radio\_group).getPageList().get(radio\_children).setChosen\_page(false);  
 }  
 } else {  
 chosen\_count = 0;  
 for(int i = 0;i < categoryList.size();i++){  
 categoryList.get(i).setChosen\_category(false);  
  
 if (chosen\_num\_list.get(i) > 0){  
 List<Category.Page> pageList = categoryList.get(i).getPageList();  
 for (int j = 0;j < pageList.size();j++){  
 pageList.get(j).setChosen\_page(false);  
 }  
 }  
 chosen\_num\_list.set(i,0);  
 }  
 }  
  
 arrange\_or\_select\_all\_image.setImageResource(R.drawable.*arrange*);  
 change\_or\_add\_image.setImageResource(R.drawable.*edit\_change*);  
 open\_linear.setVisibility(View.*VISIBLE*);  
  
 arrange\_dialog\_linear.setVisibility(View.*GONE*);  
}

### 5.20、定义reduceChosen\_count()页面计数递减函数

代码：

public void reduceChosen\_count(int num){  
 chosen\_count -= num;  
 chosen\_num\_tv.setText(String.*valueOf*(chosen\_count));  
}

### 5.21、定义radio\_change()单选标记组号和子号更改函数

代码：

private void radio\_change(int group, int children) {  
 if(radio\_group != group || radio\_children != children) {  
 if (radio\_group != -1 && radio\_children == -1) {  
 categoryList.get(radio\_group).setChosen\_category(false);  
 } else if (radio\_group != -1) {  
 categoryList.get(radio\_group).getPageList().get(radio\_children).setChosen\_page(false);  
 }  
 radio\_group = group;  
 radio\_children = children;  
 }  
}

### 5.22、定义search\_result\_view()页面查询结果显示功能函数

代码：

private void search\_result\_view(final List<Category.Page> result\_pageList) {  
  
 pageAdapter = new PageAdapter(result\_pageList,screenHeight,arrange\_select\_show);  
 search\_linear.setVisibility(View.*GONE*);  
 search\_recycler.setVisibility(View.*VISIBLE*);  
 search\_recycler.setAdapter(pageAdapter);  
 //取消焦点，避免自动滚动  
 search\_recycler.setFocusableInTouchMode(false);  
 pageAdapter.setOnItemClickListener(new PageAdapter.onItemClickListener() {  
 @Override  
 public void onClick(View view, int pos, List<Category.Page> pageList) {  
 if(arrange\_select\_show){  
 boolean chosen = pageList.get(pos).isChosen\_page();  
  
 pageList.get(pos).setChosen\_page(!chosen);  
  
 if (edit\_change){  
 if(radio\_children != pos || radio\_group != -1) {  
// Log.d(TAG, "onClick: radio\_children != pos || radio\_group != -1 " + radio\_group + "=" + radio\_children);  
 if(radio\_children != -1) {  
 if (radio\_group == -1) {  
 pageList.get(radio\_children).setChosen\_page(false);  
 pageAdapter.notifyItemChanged(radio\_children);  
 } else {  
 categoryList.get(radio\_group).getPageList().get(radio\_children).setChosen\_page(false);  
 radio\_group = -1;  
 myExpandable.notifyDataSetChanged();  
 }  
 } else if(radio\_group != -1) {  
 categoryList.get(radio\_group).setChosen\_category(false);  
 Log.*d*(*TAG*, "onClick: radio\_group: " + radio\_group);  
 myExpandable.notifyDataSetChanged();  
  
 radio\_group = -1;  
 }  
 radio\_children = pos;  
 }  
 }else {  
 if (!chosen) {  
 addChosen\_count(1);  
 } else {  
 reduceChosen\_count(1);  
 }  
 }  
  
 pageAdapter.notifyItemChanged(pos);  
 } else {  
 Category.Page page = pageList.get(pos);  
 Intent intent = new Intent(Note\_category\_main.this, OidActivity.class);  
 //打开指定页  
 intent.putExtra("DESIGNATION", true);  
 intent.putExtra("BOOKID", 100);  
 intent.putExtra("PAGEID", Integer.*parseInt*(page.getPage\_id()));  
 startActivityForResult(intent,*OPEN\_NOTEBOOK*);  
 }  
 }  
  
 @Override  
 public boolean onLongClick(View view, int pos, List<Category.Page> pageList) {  
 if (!edit\_change) {  
 arrange\_or\_change\_show();  
 if (arrange\_select\_show) {  
 boolean chosen = pageList.get(pos).isChosen\_page();  
 pageList.get(pos).setChosen\_page(!chosen);  
 addChosen\_count(1);  
 } else {  
 arrange\_or\_change\_quit();  
 }  
 pageAdapter.setIsShow\_select(arrange\_select\_show);  
 pageAdapter.notifyDataSetChanged();  
 }  
 return false;  
 }  
 });  
 }

### 5.23、定义show\_move\_dialog()页面移动对话框函数

代码：

private void show\_move\_dialog() {  
 View view = View.*inflate*(this,R.layout.*choice\_two\_btn\_dialog*,null);  
 final TextView confirm\_move\_tv = (TextView)view.findViewById(R.id.*tv\_move\_confirm*);  
 final TextView cancel\_move\_tv = (TextView)view.findViewById(R.id.*tv\_move\_cancel*);  
 final int[] pos = {0};  
 String[] names = getCategory\_name\_array();  
  
 final AlertDialog.Builder builder = new AlertDialog.Builder(Note\_category\_main.this);  
 builder.setTitle("选择目标类别");  
 builder.setView(view);  
 builder.setCancelable(false);  
 builder.setOnKeyListener(keyListener);  
 //移动类别单选  
 builder.setSingleChoiceItems(names, -1, new DialogInterface.OnClickListener() {  
 @Override  
 public void onClick(DialogInterface dialog, int which) {  
 pos[0] = which + 1;  
 }  
 });  
  
 final AlertDialog dialog = builder.create();  
  
 confirm\_move\_tv.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View view){  
 confirm\_move\_tv.setBackgroundColor(Color.*parseColor*("#C6B36E"));  
 confirm\_move\_tv.setTextColor(Color.*parseColor*("#FFFFFF"));  
 if (pos[0] == 0){  
 Toast.*makeText*(Note\_category\_main.this,"请选择类别",Toast.*LENGTH\_SHORT*).show();  
 new Thread() {  
 public void run() {  
 SystemClock.*sleep*(150);  
 runOnUiThread(new Runnable() {  
 @Override  
 public void run() {  
 confirm\_move\_tv.setBackgroundColor(Color.*parseColor*("#ffffff"));  
 confirm\_move\_tv.setTextColor(Color.*parseColor*("#C6B36E"));  
 }  
 });  
 }  
 }.start();  
 } else {  
 SQLiteDatabase db = database\_helper.getWritableDatabase();  
 db.beginTransaction();  
 try {  
 for (int i = 0; i < chosen\_num\_list.size(); i++) {  
 int num = chosen\_num\_list.get(i);  
 if (i == 0 && num > 0) {  
 Toast.*makeText*(Note\_category\_main.this, "“最近”类别不能更改", Toast.*LENGTH\_SHORT*).show();  
 } else if (num > 0 && pos[0] != i) {  
  
 List<Category.Page> pageList = categoryList.get(i).getPageList();  
  
 if (categoryList.get(i).isChosen\_category()) {  
 categoryList.get(i).setChosen\_category(false);  
 }  
 int page\_count = pageList.size();  
 reduceChosen\_count(num);  
  
 ContentValues values = new ContentValues();  
 values.put("category\_name", categoryList.get(pos[0]).getCategory\_name());  
 int page\_move\_num = 0;  
 for (int j = 0; j < page\_count; j++) {  
 Category.Page page = pageList.get(j - page\_move\_num);  
 if (page.isChosen\_page()) {  
 Category.Page newPage = page.clone();  
 categoryList.get(i).getPageList().remove(page);  
 newPage.setChosen\_page(false);  
 categoryList.get(pos[0]).getPageList().add(newPage);  
 page\_move\_num++;  
 db.update("tb\_" + newPage.getBook\_id(), values, "page\_id = ?", new String[]{newPage.getPage\_id()});  
 }  
 }  
 chosen\_num\_list.set(i, 0);  
 }  
 }  
 db.setTransactionSuccessful();  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 db.endTransaction();  
 db.close();  
 }  
  
 new Thread() {  
 public void run() {  
 SystemClock.*sleep*(300);  
 dialog.dismiss();  
 }  
 }.start();  
  
 myExpandable.notifyDataSetChanged();  
 }  
 }  
 });  
  
 cancel\_move\_tv.setOnClickListener(new View.OnClickListener() {  
 @Override  
 public void onClick(View v) {  
 cancel\_move\_tv.setBackgroundColor(Color.*parseColor*("#C6B36E"));  
 cancel\_move\_tv.setTextColor(Color.*parseColor*("#FFFFFF"));  
 new Thread() {  
 public void run() {  
 SystemClock.*sleep*(300);  
 dialog.dismiss();  
 }  
 }.start();  
 }  
 });  
  
 dialog.show();  
 }

### 5.24、定义show\_delete\_dialog()删除对话框函数

代码：

private void show\_delete\_dialog() {  
 View view = View.*inflate*(this,R.layout.*two\_btn\_dialog*,null);  
 TextView message\_tv = (TextView)view.findViewById(R.id.*tv\_delete\_message*);  
 final TextView confirm\_tv = (TextView)view.findViewById(R.id.*tv\_delete\_confirm*);  
 final TextView cancel\_tv = (TextView)view.findViewById(R.id.*tv\_delete\_cancel*);  
 final AlertDialog.Builder builder = new AlertDialog.Builder(Note\_category\_main.this);  
 message\_tv.setText("删除所选项");  
 builder.setView(view);  
 builder.setCancelable(false);  
 builder.setOnKeyListener(keyListener);  
  
 final AlertDialog dialog = builder.create();  
  
 confirm\_tv.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View view){  
 confirm\_tv.setBackgroundColor(Color.*parseColor*("#C6B36E"));  
 confirm\_tv.setTextColor(Color.*parseColor*("#ffffff"));  
 List<Category> new\_categoryList = new ArrayList<Category>();  
 int delete\_category\_num = 0;  
  
 SQLiteDatabase db = database\_helper.getWritableDatabase();  
  
 for (int i = 0;i < chosen\_num\_list.size();i++){  
 int num = chosen\_num\_list.get(i);  
  
 if (categoryList.get(i).isChosen\_category()) {  
 db.beginTransaction();  
 if (i < 2) {  
 Category category = new Category();  
 category.setCategory\_name(categoryList.get(i).getCategory\_name());  
 new\_categoryList.add(category);  
 }else {  
 delete\_category\_num++;  
 db.delete("tb\_category","category\_name = ?"  
 ,new String[]{ categoryList.get(i).getCategory\_name() });  
 }  
  
 try {  
 for (Category.Page page : categoryList.get(i).getPageList()) {  
  
 db.delete("tb\_" + page.getBook\_id(),"page\_id = ?",new String[]{ page.getPage\_id() });  
 db.delete("tb\_" + page.getBook\_id() + "\_" + page.getPage\_id(),null,null);  
  
 }  
 db.setTransactionSuccessful();  
 } catch (Exception e){  
 e.printStackTrace();  
 } finally {  
 db.endTransaction();  
 }  
  
 reduceChosen\_count(categoryList.get(i).getPageList().size());  
 chosen\_num\_list.set(i,0);  
 } else {  
 //类别单个页面删除  
 List<Category.Page> pageList = categoryList.get(i).getPageList();  
 Category category = new Category();  
 category.setCategory\_name(categoryList.get(i).getCategory\_name());  
  
 if (chosen\_count == 0 && num == 0){  
 List<Category.Page> newPage\_list = new ArrayList<Category.Page>(pageList);  
 category.setPageList(newPage\_list);  
 } else {  
 db.beginTransaction();  
 try {  
 for (Category.Page page : pageList) {  
 if (!page.isChosen\_page()) {  
 Category.Page newPage = page.clone();  
 category.getPageList().add(newPage);  
  
 }else {  
 reduceChosen\_count(1);  
 Log.*d*(*TAG*, "onClick: i===" + i);  
  
 db.delete("tb\_" + page.getBook\_id(),"page\_id = ?",new String[]{ page.getPage\_id() });  
 db.delete("tb\_" + page.getBook\_id() + "\_" + page.getPage\_id(),null,null);  
 }  
 }  
 db.setTransactionSuccessful();  
 } catch (Exception e) {  
 e.printStackTrace();  
 } finally {  
 db.endTransaction();  
 }  
 chosen\_num\_list.set(i,0);  
 }  
 new\_categoryList.add(category);  
 }  
 }  
 db.close();  
  
 while (delete\_category\_num > 0){  
 chosen\_num\_list.remove(0);  
 delete\_category\_num--;  
 }  
  
 new\_categoryList.get(0).setPageList(resetRecent\_list(new\_categoryList));  
 categoryList.clear();  
 categoryList = new\_categoryList;  
 if (pageAdapter != null){  
 pageAdapter.setPageList(search\_page(page\_searchview.getQuery().toString()));  
 pageAdapter.notifyDataSetChanged();  
 }  
 myExpandable.setCategoryList(categoryList);  
 myExpandable.notifyDataSetChanged();  
  
 ArrayAdapter<String> arrayAdapter = new ArrayAdapter<String>(Note\_category\_main.this,  
 R.layout.*support\_simple\_spinner\_dropdown\_item*,getCategory\_name\_list());  
 category\_spinner.setAdapter(arrayAdapter);  
  
 new Thread() {  
 public void run() {  
 SystemClock.*sleep*(300);  
  
 dialog.dismiss();  
 }  
 }.start();  
 }  
 });  
  
 cancel\_tv.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View view){  
 cancel\_tv.setBackgroundColor(Color.*parseColor*("#C6B36E"));  
 cancel\_tv.setTextColor(Color.*parseColor*("#ffffff"));  
 new Thread() {  
 public void run() {  
 SystemClock.*sleep*(300);  
 dialog.dismiss();  
 }  
 }.start();  
 }  
 });  
  
 dialog.show();  
 }

### 5.25、在onCreate()中实现开启整理和全选点击监听

代码：

arrange\_or\_select\_all\_image.setOnClickListener(new View.OnClickListener(){  
 @Override  
 public void onClick(View view){  
 if (!arrange\_select\_show) {  
 arrange\_or\_change\_show();  
 }  
 else {  
 chosen\_all = !chosen\_all;  
 List<Category.Page> pageList;  
 for (int i = 0;i < categoryList.size();i++){  
 categoryList.get(i).setChosen\_category(chosen\_all);  
 pageList = categoryList.get(i).getPageList();  
 for (int j = 0;j < pageList.size();j++){  
 pageList.get(j).setChosen\_page(chosen\_all);  
 }  
 if (chosen\_all){  
 chosen\_num\_list.set(i,pageList.size());  
 } else {  
 chosen\_num\_list.set(i,0);  
 }  
 }  
  
 if (chosen\_all) {  
 arrange\_or\_select\_all\_image.setImageResource(R.drawable.*no\_select\_all*);  
 }  
 else {  
 arrange\_or\_select\_all\_image.setImageResource(R.drawable.*select\_all*);  
 }  
 chosen\_count = 0;  
 for (Category category : categoryList){  
 if (category.isChosen\_category()){  
 chosen\_count += category.getPageList().size();  
 }else {  
 for (Category.Page page : category.getPageList()){  
 if (page.isChosen\_page()){  
 chosen\_count++;  
 }  
 }  
 }  
 }  
 chosen\_num\_tv.setText(String.*valueOf*(chosen\_count));  
  
 }  
 if (pageAdapter != null) {  
 pageAdapter.setIsShow\_select(arrange\_select\_show);  
 pageAdapter.notifyDataSetChanged();  
 }  
 else {  
 myExpandable.notifyDataSetChanged();  
 }  
  
 }  
});

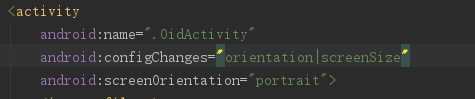
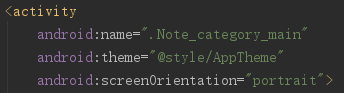
### 5.26、添加onActivityResult()界面跳转回调函数

代码：

@Override  
 protected void onActivityResult(int requestCode, int resultCode, Intent data) {  
 switch (requestCode){  
 case *OPEN\_NOTEBOOK*:  
 initCategory\_list();  
 myExpandable.notifyDataSetChanged();  
 if (pageAdapter != null){  
 pageAdapter.notifyDataSetChanged();  
 }  
 break;  
 default:  
 break;  
 }  
 }

## 配置AndroidManifest.xml

### 6.1、修改AndroidManifest.xml

将，更改为

### 6.2、添加activity

代码：

<activity  
 android:name=".OidActivity"  
 android:configChanges="orientation|screenSize"  
 android:screenOrientation="portrait" />

### 6.3、在res/valus目录下，新建styles.xml

代码：

<resources>

<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">  
 <!-- Customize your theme here. -->  
 <item name="colorPrimary">#FFFFFF</item>  
  
 </style>  
  
</resources>

二、参考资料：

1、Android 购物车的最佳实现（附源码）（通过ExpandableListView实现分店铺）：<https://blog.csdn.net/qq941263013/article/details/80901277>

2、关于Recyclerview 实现多选,单选,全选,反选,批量删除的功能的实现：<https://blog.csdn.net/guohaosir/article/details/72403724>

3、Android长按图片多选效果（Recyclerview+Checkbox）的实现：<https://www.jianshu.com/p/4c03c89d5a0f>

4、RecyclerViewSelectableAdapterDemo【封装BaseSelectableAdapter用于多选、单选，以及切换选中状态等功能】：https://www.cnblogs.com/whycxb/p/9318829.html

三、改进建议：

可添加按时间划分的全页面展示选项；页面属性查看功能。