2016/2/24 ldy

数据库操作标准

# 总共分为3层

## dao层，包括数据库的创建，及单表的增删改查的简单操作

### DBCreator继承SQLiteOpenHelper，负责和系统底层交互，创建和更新数据库。

### Dao类负责具体表的增删改查等，没有复杂的组合操作，只对其对应的表的进行操作，即有几个表就要有几个Dao类，它们都需要继承BaseDao，BaseDao实现了EntityDao接口，并在其初始化时获取SQLiteDatabase实例，全局使用这一个实例，一般不需要关闭它。

### beans包底下放着这一层需要使用的实体类，其与对应表的列一一对应，dao层负责把数据库的内容和实体对象之间进行来回转换

## DatabaseHelper，对dao层的操作进行组合，从而完成最上层的任务，最上层不直接与dao层交互

### 这一层主要是对单个表或多个表的多个基础Dao操作进行组合，从而完成上层的数据库相关的业务

### 这层可以不复用dao层的bean类，可以定义一些单独的bean类（如：代码中的personBean），直接供上层使用

## 展示层，与界面相关的操作（代码中并没有实现这一层）

### 这一层直接与DatabaseHelper交互从而完成数据库相关的操作，不能直接访问dao层

# 测试，示例代码中直接使用的junit，建议采用Robolectric来做测试，速度比较快

## 需要对dao层的每一个方法进行单独及组合的测试

## 示例代码没有写UI，可以使用对应测试做验证

# 事务

所有多次的写操作必须显式的开启事务，不然系统会为每一次操作单独开启事务，导致写入缓慢且出错时无法回滚

## 嵌套事务

事务在进行嵌套时，可以认为只有最外层的事务有效，里层的事务失效，所以

### 在批量的插入和修改时在dao层显式的使用了事务

### 在DatabaseHelper对dao层进行组合操作时又显式的开启事务，避免只写入部分数据导致程序出错

# 其它：

## 数据库的相关操作类基本都是单例的，所以在传入参数时自动获取applicationContext避免出现内存泄露。

## SQLiteDatabase实例只会在dao类创建时获取一个，所以如果没有必要不需要释放，但是cursor使用完之后一定要及时释放

## 在DBHelper中创建表时，将所用的常量定义在自己的类中，如果较多可以将这些定义在单独的类中，不要将这些定义在dao类中，因为理论上来说dao类属于他的上一层，避免出现互相引用。

## dao层和databaseHelper其实都属于cleanarchitecture的data层，其实上面还有一层domain层，是用来将数据库操作和其他操作（如网络操作）组装成一个完整的UseCase的，如果感兴趣可以自行搜索android cleanarchitecture。

## 每一层并不把异常抛给上层，而是自己处理（主要是写操作相关的回滚）并打印，只是把操作结果告诉上层

## 关于一些查询操作可以调用一些sqlite的原生方法，直接在数据库中筛选，而不是获取出来数据后在筛选，可以显著提高性能，经常需要查询而又不经常改动的部分，可以通过建立索引来显著减少查询时间

# 注：

基本上所有东西都在示例代码里，时间紧急，可能有很多问题没有考虑到，可能代码中还存在着其它的没有测试出来的bug或者不合适的地方，欢迎一起探讨，邮箱：ldy@xdja.com