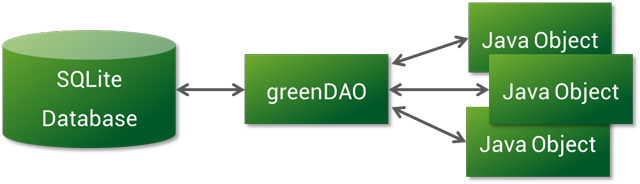
# 简介

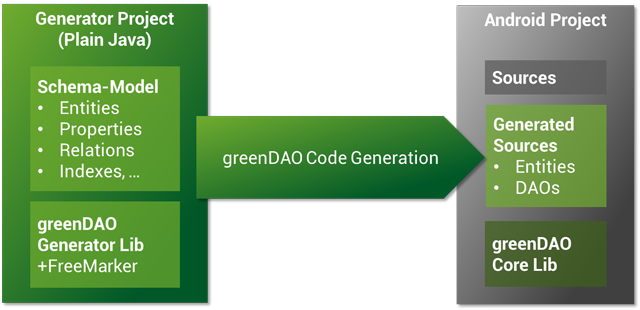
greenDAO是一个用于帮助安卓开发者处理sqlite数据库相关操作的开源工具项目，这个项目里面实现了一套ORM（对象关系映射）方案，将数据库中的表抽象成Java data objects (实体) 和 DAOs (data access object)，使得开发者不需要花精力处理sqlite中的细节问题。



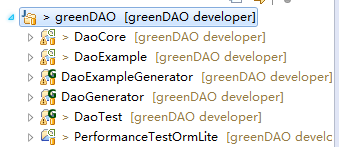
greenDAO具有以下特点：

1. 使用方便，项目提供了代码生成工具，可以自动生成实体和对应的DAO，DAO中提供了大量的api来完成增删改查等操作；
2. 功能强大，DAO中除了提供增删改查的api，还提供缓存功能以加快查询速度，同时，还支持实体对象之间的各种关系映射（一对一，一对多）；
3. 性能优异，insert、update和load数据的速度可以达到几千条每秒；
4. 集成方便，生成的代码直接加入项目中就能使用，二次开发工作少；
5. Jar包较小，只有几十K。

# 整体结构



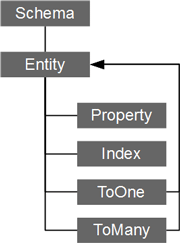
GreenDAO主要分两大模块，一个是上图左边的代码生成模块，另一个是上图右下方的greenDAO Core Lib。



上图是git上greenDAO的目录结构，DaoCore是核心lib项目，DaoGenerator是代码生成器，DaoExample和DaoExampleGenerator分别是安卓使用示例和代码生成示例项目，剩下两个是测试项目。

# 具体功能

## 实体和DAO的生成



如图，Schema对象的结构如上：

Schema是主对象，每张表我们都会定义一个Entity，然后添加到Schema里面；

Property对应表中的列，定义Entity时，我们只需给每一列生成一个对应的Property，然后添加到Entity对象中就行；

Index是索引，我们可以添加索引到Property上，生成建表语句时就会自动给对应的列加上索引；

ToOne和ToMany是关系属性，可以通过他们定义Entity之间的关系。

如何定义一张表，并生成相应的实体类和DAO呢，我们只需要定义DaoGenerator中的一个schema对象，然后再调用DaoGenerator中GenerateAll方法就行：

Schema schema = new Schema(1, "de.greenrobot.daoexample");  
  
Entity note= schema.addEntity("Note");  
note.addIdProperty();  
note.addStringProperty("text").notNull();  
note.addStringProperty("comment");  
note.addDateProperty("date");  
  
new DaoGenerator().generateAll(schema, "../DaoExample/src-gen");

索引例子：

Entity his = schema.addEntity("PlayHistory");

his.addIdProperty().autoincrement();

his.addStringProperty("playId");

his.addStringProperty("subjectId");

his.addStringProperty("videoTitle");

his.addStringProperty("playedTime");

his.addStringProperty("clientType");

his.addStringProperty("definition");

his.addStringProperty("episode");

his.addStringProperty("picPath");

his.addStringProperty("categoryId");

his.addIntProperty("tvLength");

his.addIntProperty("isSynchronized");

his.addStringProperty("albumName");

his.addIntProperty("nextplayId");

his.addStringProperty("lastWatchTime");

his.addStringProperty("passport");

his.addStringProperty("localUrl");

his.addIntProperty("tvisfee");

his.addIntProperty("real\_playorder");

his.addIntProperty("site");

List<Property> pros = his.getProperties();

**if** (pros != **null** && pros.size() > 0) {

**for** (Property p : pros) {

**if** ("playId".equals(p.getPropertyName())) {

Index index = **new** Index();

index.addProperty(p);

his.addIndex(index);

**break**;

}

}

}

ToOne、ToMany例子：

Entity customer = schema.addEntity("Customer");

customer.addIdProperty();

customer.addStringProperty("name").notNull();

Entity order = schema.addEntity("Order");

order.setTableName("ORDERS"); // "ORDER" is a reserved keyword

order.addIdProperty();

Property orderDate = order.addDateProperty("date").getProperty();

Property customerId = order.addLongProperty("customerId").notNull().getProperty();

order.addToOne(customer, customerId);

ToMany customerToOrders = customer.addToMany(order, customerId);

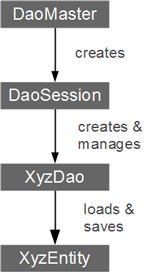
customerToOrders.setName("orders");

customerToOrders.orderAsc(orderDate);

生成的DAO中有建表语句和删表语句，在数据库初始化时会执行对应的语句去sqlite中建立对应的表。

## 在安卓项目中的使用

#### 核心类



greenDAO的核心类包括上面四个：

DaoMaster是greenDAO的入口类，DaoMaster中存了SQLiteDatabase对象；DaoMaster有两个内部类，OpenHelper和DevOpenHelper，这两个类都继承了

SQLiteOpenHelper，初始化时，会通过这两个类来实现数据库的创建和升级；

DaoSession负责管理所有的DAO对象，通过对应的getter方法就能获取某个实体对应的DAO对象，同时DaoSession提供了一些基本的增删改查方法，而且负责管理实体的缓存；

DAO主要负责实体数据的持久化和查询，每一个实体都有一个对应的DAO，DAO中具有比DaoSession更多的方法，如count, loadAll, and insertInTx等等；

http://greendao-orm.com/javadoc/greendao/

Entity是实体对象，其属性与数据库中对应表的属性一一对应。

#### 初始化

DevOpenHelperhelper = new DaoMaster.DevOpenHelper( this, "notes-db", null);  
db = helper.getWritableDatabase();  
daoMaster = new DaoMaster(db);  
daoSession = daoMaster.newSession();  
noteDao = daoSession.getNoteDao();

如上，DevOpenHelper继承了SQLiteOpenHelper，通过DevOpenHelper拿到SQLiteDatabase对象，然后初始化DaoMaster对象，从DaoMaster对象中可以拿到DaoSession对象，然后通过DaoSession对象可以拿到各个实体对应的DAO对象，从而可以对相应的表进行操作。如果要实现自己的新建和升级数据库的策略，可以重写DevOpenHelper，每次DaoGenerator生成代码时，都会重新生成对应的类，如果要想保留我们自己的代码，可以在要保留的代码前后分别加上

*// KEEP INCLUDES - put your custom includes here*  
*// KEEP INCLUDES END*

#### 增删改查

DAO对象中提供了一系列的api，我们可以直接调用：

|  |  |
| --- | --- |
| **Method Summary** | |
| long | [**count**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#count())() |
| void | [**delete**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#delete(T))([T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html) entity)            Deletes the given entity from the database. |
| void | [**deleteAll**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#deleteAll())() |
| void | [**deleteByKey**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#deleteByKey(K))([K](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html) key)            Deletes an entity with the given PK from the database. |
| boolean | [**detach**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#detach(T))([T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html) entity)            Detaches an entity from the identity scope (session). |
| java.lang.String[] | [**getAllColumns**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#getAllColumns())() |
| java.lang.String[] | [**getNonPkColumns**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#getNonPkColumns())() |
| java.lang.String[] | [**getPkColumns**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#getPkColumns())() |
| [Property](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/Property.html) | [**getPkProperty**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#getPkProperty())() |
| [Property](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/Property.html)[] | [**getProperties**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#getProperties())() |
| [AbstractDaoSession](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDaoSession.html) | [**getSession**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#getSession())() |
| java.lang.String | [**getTablename**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#getTablename())() |
| long | [**insert**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#insert(T))([T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html) entity)            Insert an entity into the table associated with a concrete DAO. |
| void | [**insertInTx**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#insertInTx(java.lang.Iterable))(java.lang.Iterable<[T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html)> entities)            Inserts the given entities in the database using a transaction. |
| void | [**insertInTx**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#insertInTx(java.lang.Iterable, boolean))(java.lang.Iterable<[T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html)> entities, boolean setPrimaryKey)            Inserts the given entities in the database using a transaction. |
| void | [**insertInTx**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#insertInTx(T...))([T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html)... entities)            Inserts the given entities in the database using a transaction. |
| long | [**insertOrReplace**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#insertOrReplace(T))([T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html) entity)            Insert an entity into the table associated with a concrete DAO. |
| void | [**insertWithoutSettingPk**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#insertWithoutSettingPk(T))([T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html) entity)            Insert an entity into the table associated with a concrete DAO. |
| [T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html) | [**load**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#load(K))([K](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html) key)            Loads and entity for the given PK. |
| java.util.List<[T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html)> | [**loadAll**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#loadAll())()            Loads all available entities from the database. |
| [T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html) | [**loadByRowId**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#loadByRowId(long))(long rowId) |
| java.util.List<[T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html)> | [**query**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#query(java.lang.String, java.lang.String[], java.lang.String, java.lang.String, java.lang.String))(java.lang.String selection, java.lang.String[] selectionArgs, java.lang.String groupBy, java.lang.String having, java.lang.String orderby)            Performs a standard Android-style query for entities. |
| [QueryBuilder](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/QueryBuilder.html)<[T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html)> | [**queryBuilder**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#queryBuilder())() |
| java.util.List<[T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html)> | [**queryRaw**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#queryRaw(java.lang.String, java.lang.String...))(java.lang.String where, java.lang.String... selectionArg)            A raw-style query where you can pass any WHERE clause and arguments. |
| void | [**refresh**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#refresh(T))([T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html) entity)            Resets all locally changed properties of the entity by reloading the values from the database. |
| void | [**update**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#update(T))([T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html) entity) |
| void | [**updateInTx**](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html#updateInTx(java.lang.Iterable))(java.lang.Iterable<[T](http://greendao-orm.com/javadoc/greendao/de/greenrobot/dao/AbstractDao.html)> entities)            Inserts the given entities in the database using a transaction. |

#### 复杂查询

如果要实现复杂一点的查询，可以通过QueryBuilder来实现：

QueryBuilder qb = userDao.queryBuilder();  
qb.where(Properties.FirstName.eq("Joe"),  
qb.or(Properties.YearOfBirth.gt(1970),  
qb.and(Properties.YearOfBirth.eq(1970), Properties.MonthOfBirth.ge(10))));  
List youngJoes = qb.list();

如果我们想自己写sql语句查询，可以通过QueryBuilder的StringCondition来实现：

Query query = userDao.queryBuilder().where(  
new StringCondition("\_ID IN " +  
"(SELECT USER\_ID FROM USER\_MESSAGE WHERE READ\_FLAG = 0)").build();

QueryBuilder还提供多种查询方式，支持延迟加载：

|  |  |
| --- | --- |
| **list()** | 直接加载 |
| **listLazy()** | List中的对象被用到时，目标对象才从数据库中取出，可以实现延迟加载，不过用完后需要关闭对应的对象，被用到的数据会加入到缓存中 |
| **listLazyUncached()** | List中有对象被用到时，整个list才从数据库中取出，用完后需要关闭. |
| **listIterator()** | Iterateor中的对象被用到时，目标对象才从数据库中取出，用完后需要关闭对应的对象，被用到的数据不会加入到缓存中 |

#### 数据缓存

为了加快查询速度，DaoSession中会默认以id为key对查询结果进行缓存（可配置为不缓存），下次根据id查询时，会先到缓存中去查询，如果查询到了，就不会去数据库中重新取了。如果启用缓存，查询速度会加快，但是要注意的是，如果我们用其它方式（不是用greenDAO提供的api进行操作）进行了update操作的话，缓存中的数据会不同步。

#### 对象关系映射

ORM是greenDAO的一大特色，通过对象关系映射，可以把业务上有联系的不同实体连接起来，通过查询A实体，方便的关联到与之相关的B、C等等实体。

目前greenDAO支持一对一和一对多两种关系：

一对一：

Entity customer = schema.addEntity("Customer");  
customer.addIdProperty();  
customer.addStringProperty("name").notNull();  
  
Entity order = schema.addEntity("Order");  
order.setTableName("ORDERS"); *// "ORDER" is a reserved keyword*  
order.addIdProperty();

Property customerId = order.addLongProperty("customerId").notNull().getProperty();  
order.addToOne(customer, customerId);

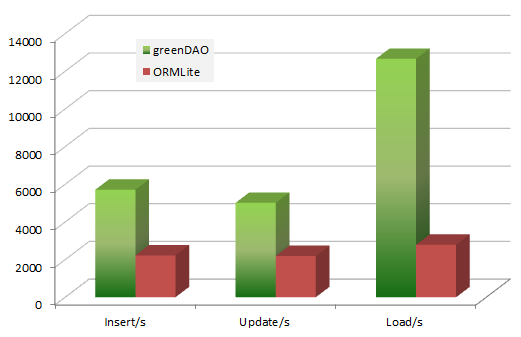
一对多：

Entity customer = schema.addEntity("Customer");  
customer.addIdProperty();  
customer.addStringProperty("name").notNull();  
  
Entity order = schema.addEntity("Order");  
order.setTableName("ORDERS"); *// "ORDER" is a reserved keyword*  
order.addIdProperty();

Property customerId = order.addLongProperty("customerId").notNull().getProperty();  
  
ToMany customerToOrders = customer.addToMany(order, customerId);

查询时，我们可以通过Customer 对象的getOrders方法，获取所有与之关联的order，同时可以通过Order对象的getCustomer方法,获取order对应的customer。

### 性能



上图是greenDAO官方进行的一个对比表测试，可以看到insert、update、load速度分别到了5000/s、4000/s、12000/s，并且远超对比的另一个ORM工具ORMLite。

我在本地也做了一个测试，测试的是有索引和没索引时的insert和load速度：

在没有索引时，insert速度为10000/s，在600万条数据中根据id查询一条记录的时间为4秒；

在有索引时，insert的速度随着数据量的增加而下降，在100万条数据以内时，inser速度平均为2000/s，100万条数据以上insert就非常慢了，从第100万条数据开始，插入500万条，总共花了30分钟左右，但是查询速度非常快，从600万条数据里面根据id查询一条记录的时间为0.15秒。

待解决问题：

1,Model里面有自定义字段时，生成代码时能否保留；

2,自定义升级策略；

3,Model里面有复杂对象时如何存储。