GreenDao介绍
android开发的本地数据库存储是sqlite。greenDAO应该算是当前最火的数据库开源框架了吧，它是一个移动开发的ORM（object / relational mapping）框架，是对sqlite数据库访问的对象化封装。以对象的形式去访问数据库，数据库表里面的字段就相当于对象的属性了。可以直接obj.data的形式访问了。如果觉得效率不够高，你也可以自己ORM的框架。据我所致GreeDao是android开发性能最好的数据库开源框架。

总之，一句话，greenDAO就是实现Java对象和SQLite Datebase的一个媒介人，简化了SQLite的操作。

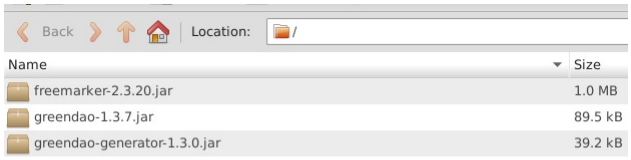
一.下载GreenDao

要使用肯定要先下载他的软件包了，官网上有它的连接，对于marven和gradle环境直接到serarch.maven.org上下载jar包就好了。

下载的jar导入到工程里面就可以了，通常都是/libs目录下。

我这里也提供一个csdn的下载地址：http://download.csdn.net/detail/csm_qz/8569031

解压后有三个文件如下图



greendao-generator-1.3.0.jar 和 freemarker-2.3.20.jar是我们创建java **generator**工程时生成Dao文件需要用到的（什么是我说的Dao文件，往下看就会知道）
greendao-1.3.7.jar是Android开发中需要用到的

二.创建generator工程（用来生成GreenDao开发过程中需要的java文件）
（1）创建Java工程（非Android工程）
（2）导入greenDao-generator.jar和freemarker.jar两个包。
freemarker是一个用java写的模板引擎，它能够基于模板来生成文本输出。应该就是用来自动生成DAO文件的。eclipse下面就是在properties --> Java build path --> libraries下面导入jar包。
（3）创建一个包javagreendao
（4）创建一个类，类名为ExampleDaoGenerator，类的定义如下：

```
package javagreendao;

import de.greenrobot.daogenerator.DaoGenerator;
import de.greenrobot.daogenerator.Entity;
import de.greenrobot.daogenerator.Property;
import de.greenrobot.daogenerator.Schema;
import de.greenrobot.daogenerator.ToMany;
/**
 * Generates entities and DAOs for the example project DaoExample.
 *
 * Run it as a Java application (not Android).
 *
 * @author Markus
 */
public class ExampleDaoGenerator
{

    public static void main(String[] args) throws Exception
    {
        Schema schema = new Schema(3, "com.cn.speedchat.greendao");    // 参数3是数据库版本号，“com.cn.speedchat.greendao”是包名，也就是说生成的Dao文件会在这个包下，可以将Schema理解为数据库上下文吧
        addNote(schema);        //addNote() addSession() addReplay()这三个函数相当于建立了三个表，表名你都可以不用管了
        addSession(schema);
        addReplay(schema);
        addCustomerOrder(schema);
        new DaoGenerator().generateAll(schema, "../javagreendao/src-gen");    //这个是生成Dao文件的路径的位置，这个代表当前工程的上一级目录的javagreendao的src-gen文件夹里面，其实就是跟src同一级目录，所以你自己要在src同一级目录下新建一个src-ge
    }

    private static void addNote(Schema schema)          //这个是一个Note表,然后后面的node.add***是表的字段名以及属性
    {
        Entity note = schema.addEntity("MqttChatEntity");    //"MqttChatEntity"相当于是表的类名，用MqttChatEntity生成对象就可以访问这个表属性了，也就是这个表对应了这个类，待会使用你就会明白了
        note.addIdProperty().autoincrement();
        note.addIntProperty("mode").notNull();
        note.addStringProperty("sessionid").notNull();
        note.addStringProperty("from").notNull();
        note.addStringProperty("to").notNull();
        note.addStringProperty("v_code");
        note.addStringProperty("timestamp").notNull();
        note.addStringProperty("platform");
        note.addStringProperty("message");
        note.addBooleanProperty("isread").notNull();
        note.addLongProperty("gossipid");
        note.addStringProperty("gossip");
        note.addIntProperty("chattype").notNull();
        note.addStringProperty("imagepath");
        note.addStringProperty("base64image");
    }
```

```
    private static void addSession(Schema schema)     //这个是一个Session表,然后后面的node.add***是表的字段名以及属性（这是我写的会话的一个表）
    {
        Entity note = schema.addEntity("SessionEntity");
        note.addIdProperty().autoincrement();
        note.addStringProperty("sessionid").notNull().unique();
        note.addStringProperty("from").notNull();
        note.addStringProperty("to").notNull();
        note.addLongProperty("gossipid").notNull();
        note.addStringProperty("gossip");
        note.addIntProperty("sessiontype").notNull();
        note.addBooleanProperty("asdasd").notNull();
    }

    private static void addReplay(Schema schema)  //这个是一个Replay表,然后后面的node.add***是表的字段名以及属性（这是我写的回复的一个表）
    {
        Entity note = schema.addEntity("ReplayEntity");     //ReplayEntity对应的类名
        note.addIdProperty().autoincrement();
        note.addIntProperty("mode").notNull();
        note.addStringProperty("from").notNull();
        note.addStringProperty("to").notNull();
        note.addStringProperty("v_code");
        note.addStringProperty("timestamp").notNull();
        note.addStringProperty("platform");
        note.addStringProperty("message");
        note.addIntProperty("msgtype").notNull();
        note.addBooleanProperty("isread").notNull();
    }


    private static void addCustomerOrder(Schema schema)    //这个不用管了，你照抄吧
    {
        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);
    }

}
```

（1）增加表

//如果你自己想加一些其他的表的话，你可以自己按照addSession，addNote ,addReplay三个函数的方式加，类名、字段名可以自己随便取

比如说，比我我要加一个用户表，字段包括name,age,sex三个，我可以这样做

```
    private static void addUser(Schema schema)  //这个是一个Replay表,然后后面的node.add***是表的字段名以及属性（这是我写的回复的一个表）
    {
        Entity note = schema.addEntity("UserEntity");     //ReplayEntity对应的类名
        note.addIdProperty().autoincrement();
        note.addStringProperty("name").notNull();
        note.addIntProperty("age").notNull();
        note.addBooleanProperty("sex").notNull();     //true代表男，false代表女
    }
```

然后在main函数里面做如下调用

```
    public static void main(String[] args) throws Exception
    {
        Schema schema = new Schema(3, "com.cn.speedchat.greendao");     // 参数3是数据库版本号，“com.cn.speedchat.greendao”是包名，也就是说生成的Dao文件会在这个包下，可以将Schema理解为数据库上下文吧
        addNote(schema);         //addNote() addSession() addReplay()这三个函数相当于建立了三个表，表名你都可以不用管了
        addSession(schema);
        addReplay(schema);
        addUser(schema);
        addCustomerOrder(schema);
        new DaoGenerator().generateAll(schema, "../javagreendao/src-gen");
    }
```

（2）删除表

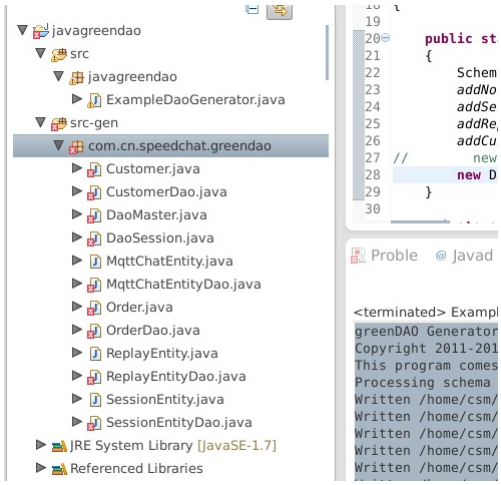
当然一些不需要的表你可以不用，删掉就行，比如说你不须要addReplay，你就在main函数里面别调用 addReplay(schema)就行

总之呢，这就是一个基于greenDao-generator.jar和freemarker.jar两个包的java工程

然后运行该工程，控制台打印出如下结果：

```
greenDAO Generator
Copyright 2011-2013 Markus Junginger, greenrobot.de. Licensed under GPL V3.
This program comes with ABSOLUTELY NO WARRANTY
Processing schema version 3...
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/MqttChatEntityDao.java
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/MqttChatEntity.java
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/SessionEntityDao.java
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/SessionEntity.java
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/ReplayEntityDao.java
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/ReplayEntity.java
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/CustomerDao.java
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/Customer.java
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/OrderDao.java
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/Order.java
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/DaoMaster.java
Written /home/csm/workspace/javagreendao/src-gen/com/cn/speedchat/greendao/DaoSession.java
Processed 5 entities in 189ms
```

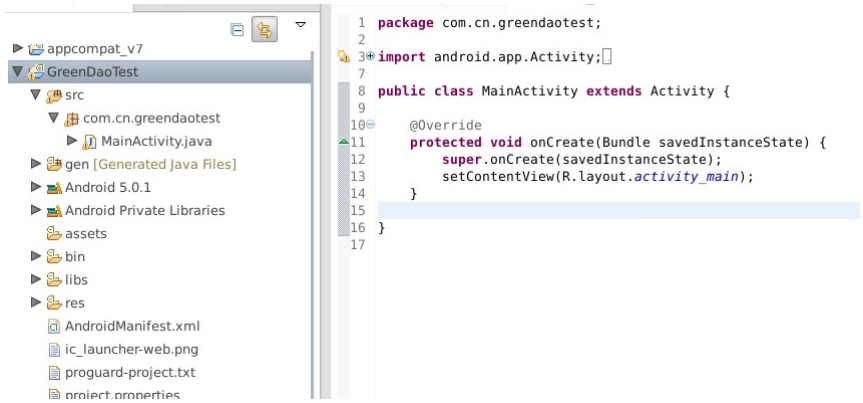
这代表成功的生成了Dao文件，然后我们按F5刷新该工程，在查看src-gen目录文件，自动生成了很多java文件，这就是我们要的，我这里截图给大家看



但是有很多错误是不是，没关系，这个工程识别不了这些文件，这些文件是基于greendao-1.3.7.jar包的，是Android工程里面要用到的。先不管这个java工程了。接下来网下走

三.创建Android工程

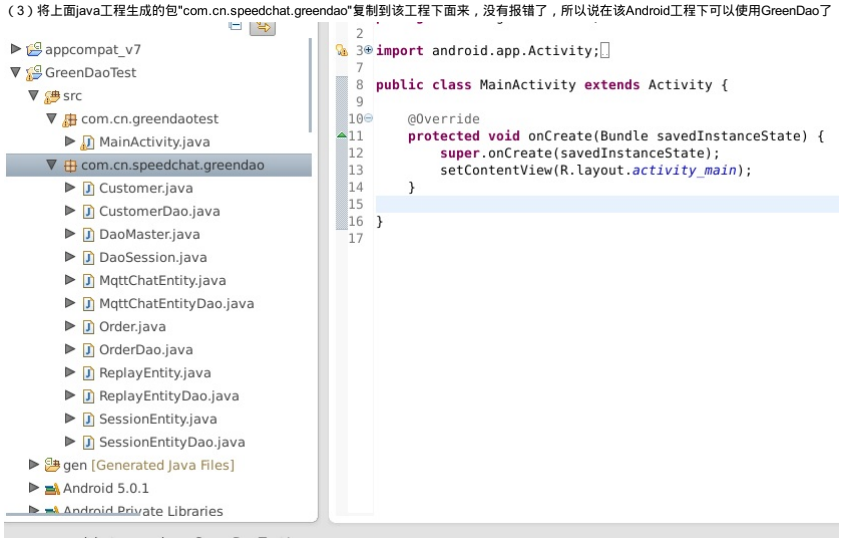
（1）创建一个Android工程



（2）导入greendao-1.3.7.jar

方法是将greendao-1.3.7.jar文件放在libs文件夹下，然后右键该jar包=>add to build path

（3）将上面java工程生成的包"com.cn.speedchat.greendao"复制到该工程下面来，没有报错了，所以说在该Android工程下可以使用GreenDao了



我们拿一个文件来分析，比如MqttChatEntity.java文件，内容如下：

```
package com.cn.speedchat.greendao;

// THIS CODE IS GENERATED BY greenDAO, DO NOT EDIT. Enable "keep" sections if you want to edit.
/**
 * Entity mapped to table SESSION_ENTITY.
 */
public class SessionEntity {

    private Long id;
    /** Not-null value. */
    private String sessionid;
    /** Not-null value. */
    private String from;
    /** Not-null value. */
    private String to;
    private long gossipid;
    private String gossip;
    private int sessiontype;
    private boolean asdasd;

    public SessionEntity() {
    }

    public SessionEntity(Long id) {
        this.id = id;
    }

    public SessionEntity(Long id, String sessionid, String from, String to, long gossipid, String gossip, int sessiontype, boolean asdasd) {
        this.id = id;
```

```
        this.sessionid = sessionid;
        this.from = from;
        this.to = to;
        this.gossipid = gossipid;
        this.gossip = gossip;
        this.sessiontype = sessiontype;
        this.asdasd = asdasd;
    }

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    /** Not-null value. */
    public String getSessionid() {
        return sessionid;
    }

    /** Not-null value; ensure this value is available before it is saved to the database. */
    public void setSessionid(String sessionid) {
        this.sessionid = sessionid;
    }

    /** Not-null value. */
    public String getFrom() {
        return from;
    }

    /** Not-null value; ensure this value is available before it is saved to the database. */
    public void setFrom(String from) {
        this.from = from;
    }

    /** Not-null value. */
    public String getTo() {
        return to;
    }

    /** Not-null value; ensure this value is available before it is saved to the database. */
    public void setTo(String to) {
        this.to = to;
    }

    public long getGossipid() {
        return gossipid;
    }

    public void setGossipid(long gossipid) {
        this.gossipid = gossipid;
    }

    public String getGossip() {
        return gossip;
    }

    public void setGossip(String gossip) {
        this.gossip = gossip;
    }

    public int getSessiontype() {
        return sessiontype;
    }

    public void setSessiontype(int sessiontype) {
        this.sessiontype = sessiontype;
    }

    public boolean getAsdasd() {
        return asdasd;
    }

    public void setAsdasd(boolean asdasd) {
        this.asdasd = asdasd;
    }

}
```

这就是我们平时写Android封装的一个普通的类，将一个表的字段封装到一个类里面了，然后我们就可以以对象的形式去访问，说白了就是数据库里面的一条数据对应一个对象了，下面看看怎么使用

（4）官方推荐将取得DaoMaster对象的方法放到Application层这样避免多次创建生成Session对象，

在Application实现得到DaoMaster和DaoSession的方法，实现如下，新建一个ControlApp.java文件继承自Application：

```
package com.cn.greendaotest;

import com.cn.speedchat.greendao.DaoMaster;
import com.cn.speedchat.greendao.DaoMaster.OpenHelper;
import com.cn.speedchat.greendao.DaoSession;

import android.app.Application;
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;

public class ControlApp extends Application{

 private static DaoMaster daoMaster;
 private static DaoSession daoSession;
 public static  SQLiteDatabase db;
 public static final String DB_NAME = "dbname.db";    //数据库名，表名是自动被创建的

  /**
   * 取得DaoMaster
   *
   * @param context
```

```
     * @return
     */
    public static DaoMaster getDaoMaster(Context context) {
        if (daoMaster == null) {
            OpenHelper helper = new DaoMaster.DevOpenHelper(context,DB_NAME, null);
            daoMaster = new DaoMaster(helper.getWritableDatabase());
        }
        return daoMaster;
    }

    /**
     * 取得DaoSession
     *
     * @param context
     * @return
     */
    public static DaoSession getDaoSession(Context context) {
        if (daoSession == null) {
            if (daoMaster == null) {
                daoMaster = getDaoMaster(context);
            }
            daoSession = daoMaster.newSession();
        }
        return daoSession;
    }
    /**
     * 得到Datebase
     *
     * @param context
     * @return
     */
    public static SQLiteDatabase getSQLDatebase(Context context) {
        if (daoSession == null) {
            if (daoMaster == null) {
                daoMaster = getDaoMaster(context);
            }
            db = daoMaster.getDatabase();
        }
        return db;
    }

    @Override
    public void onCreate() {

    }

}
```

（5）创建一个DBHelper的帮助类，这个帮助类写了自己封装的一些方法，内容如下

```
package com.cn.speedchat.greendao;

import java.util.ArrayList;
import java.util.List;

import com.cn.greendaotest.ControlApp;
import com.cn.speedchat.greendao.MqttChatEntityDao.Properties;

import de.greenrobot.dao.query.QueryBuilder;
import android.content.Context;
import android.util.Log;

public class DBHelper {
    private static final String TAG = DBHelper.class.getSimpleName();
    private static DBHelper instance;
    private static Context appContext;
    private DaoSession mDaoSession;
    private MqttChatEntityDao chatDao;
    private SessionEntityDao sessionDao;
    private DBHelper() {
    }
    //单例模式，DBHelper只初始化一次
    public static  DBHelper getInstance(Context context) {
        if (instance == null) {
            instance = new DBHelper();
                if (appContext == null){
                    appContext = context.getApplicationContext();
                }
                instance.mDaoSession = ControlApp.getDaoSession(context);
                instance.chatDao = instance.mDaoSession.getMqttChatEntityDao();
                instance.sessionDao = instance.mDaoSession.getSessionEntityDao();
            }
        return instance;
    }

    //删除Session表
    public  void dropSessionTable()
    {
     SessionEntityDao.dropTable(mDaoSession.getDatabase(), true);
    }
    //删除MqttChatEntity表
    public void dropChatTable()
    {
     MqttChatEntityDao.dropTable(mDaoSession.getDatabase(), true);
    }
    //删除所有表
    public void dropAllTable()
    {
     MqttChatEntityDao.dropTable(mDaoSession.getDatabase(), true);
     SessionEntityDao.dropTable(mDaoSession.getDatabase(), true);
     ReplayEntityDao.dropTable(mDaoSession.getDatabase(), true);
    }
    //创建所有表
    public void createAllTable()
    {
     MqttChatEntityDao.createTable(mDaoSession.getDatabase(), true);
     SessionEntityDao.createTable(mDaoSession.getDatabase(), true);
     ReplayEntityDao.createTable(mDaoSession.getDatabase(), true);
    }
```

```
    /**
     * insert or update note
     * @param note
     * @return insert or update note id
     */
    //插入或者删除session项
    public long saveSession(SessionEntity session){
        return sessionDao.insertOrReplace(session);
    }

    //获得所有的Session倒序排存到List列表里面
    public List<SessionEntity> loadAllSession() {
     List<SessionEntity> sessions = new ArrayList<SessionEntity>();
     List<SessionEntity> tmpSessions = sessionDao.loadAll();
     int len = tmpSessions.size();
     for (int i = len-1; i >=0; i--) {
    sessions.add(tmpSessions.get(i));
  }
     return sessions;
    }

    public void DeleteSession(SessionEntity entity) {
     sessionDao.delete(entity);
    }
    //删除某一项Session
    public void DeleteNoteBySession(SessionEntity entity) {
     QueryBuilder<MqttChatEntity> mqBuilder = chatDao.queryBuilder();
     mqBuilder.where(Properties.Sessionid.eq(entity.getSessionid()));
     List<MqttChatEntity> chatEntityList = mqBuilder.build().list();
     chatDao.deleteInTx(chatEntityList);
    }

    //根据id找到某一项
    public MqttChatEntity loadNote(long id) {
        return chatDao.load(id);
    }
    //获得所有的MqttChatEntity列表
    public List<MqttChatEntity> loadAllNote(){
        return chatDao.loadAll();
    }

    /**
     * query list with where clause
     * ex: begin_date_time >= ? AND end_date_time <= ?
     * @param where where clause, include 'where' word
     * @param params query parameters
     * @return
     */
      //查询满足params条件的列表
    public List<MqttChatEntity> queryNote(String where, String... params){
     ArrayList<MqttChatEntity> ad = new ArrayList<MqttChatEntity>();
     return chatDao.queryRaw(where, params);
    }

    //不一一介绍了，大家可以自己写，有些比较难的查询可以使用QueryBuilder来查询
    public List<MqttChatEntity> loadLastMsgBySessionid(String sessionid){
     QueryBuilder<MqttChatEntity> mqBuilder = chatDao.queryBuilder();
     mqBuilder.where(Properties.Sessionid.eq(sessionid))
     .orderDesc(Properties.Id)
     .limit(1);
     return mqBuilder.list();
    }

    public List<MqttChatEntity> loadMoreMsgById(String sessionid, Long id){
     QueryBuilder<MqttChatEntity> mqBuilder = chatDao.queryBuilder();
     mqBuilder.where(Properties.Id.lt(id))
     .where(Properties.Sessionid.eq(sessionid))
     .orderDesc(Properties.Id)
     .limit(20);
     return mqBuilder.list();
    }

    /**
     * delete all note
     */
    public void deleteAllNote(){
        chatDao.deleteAll();
    }

    /**
     * delete note by id
     * @param id
     */
    public void deleteNote(long id){
        chatDao.deleteByKey(id);
        Log.i(TAG, "delete");
    }

    public void deleteNote(MqttChatEntity note){
        chatDao.delete(note);
    }
}
```

四，使用

在Android工程下创建一个Activity使用如下（是一个启动Activity ps:launcher）：

```
package com.cn.greendaotest;

import java.util.List;

import com.cn.speedchat.greendao.DBHelper;
import com.cn.speedchat.greendao.SessionEntity;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

public class MainActivity extends Activity {
```

```
 private DBHelper dBManager;    //定义一个DBHelper对象，用他来对数据库进行增删改查

 @Override
 protected void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_main);
  dBManager = DBHelper.getInstance(this);    //得到DBHelper对象
  SessionEntity entity = new SessionEntity();    //创建一个SessionEntity实体对象，并赋值
  entity.setFrom("A");
  entity.setGossip("大家好吗？我来了...");
  entity.setGossipid(10);
  entity.setSessionid("abcdefg");
  entity.setSessiontype(1);
  entity.setTo("B");
  //下面这一行就把entity对象存数据库了，然后我们新建一个SessionEntity列表再读一下
  dBManager.saveSession(entity);    //保存到数据库

   //下面这个方法是查询Session表里面的所有数据返回一个数据列表，相当于select * from table，然后扫描打印出来
  List<SessionEntity> listentity = dBManager.loadAllSession();

  for(int i=0;i<listentity.size();i++)
  {
   SessionEntity tmpEntity = listentity.get(i);
   Log.v("tmpEntity.getFrom()",tmpEntity.getFrom());
   Log.v("tmpEntity.getGossip()",tmpEntity.getGossip());
   Log.v("tmpEntity.getGossipid()",tmpEntity.getGossipid()+"");
   Log.v("tmpEntity.getSessionid()",tmpEntity.getSessionid());
   Log.v("tmpEntity.getSessiontype()",tmpEntity.getSessiontype()+"");
   Log.v("tmpEntity.getTo()",tmpEntity.getTo());
  }
 }

}
```

启动app打印结果如下因为数据库里面的session表里面只有一条数据：

```
324    29324    com.cn.greendaotest    greenDAO                  Creating tables for schema version 3
324    29324    com.cn.greendaotest    tmpEntity.getFrom()       A
324    29324    com.cn.greendaotest    tmpEntity.getGossip()     大家好吗？我来了...
324    29324    com.cn.greendaotest    tmpEntity.getGossipid()   10
324    29324    com.cn.greendaotest    tmpEntity.getSessionid()  abcdefg
324    29324    com.cn.greendaotest    tmpEntity.getSessiontype() 1
324    29324    com.cn.greendaotest    tmpEntity.getTo()         B
324    29324    com.cn.greendaotest    libEGL                    loaded /system/lib/egl/libEGL_mali.so
```

其他表的增删改插参照DBHelper文件里面的实现吧，更复杂的查询请用QueryBuilder，DBHelper里面也有可以参考。下面我将java工程和Android工程的下载地址给大家

五.查看数据库

通过adb shell进入该程序的包路径下，查看确实创建了dbname.db数据库：

```
shell@android:/data/data/com.cn.greendaotest # cd databases/
shell@android:/data/data/com.cn.greendaotest/databases # l
dbname.db
dbname.db-journal
shell@android:/data/data/com.cn.greendaotest/databases #
```

将数据库dbname.db导出来用sqlite3在命令行查看

```
sqlite> .schema
CREATE TABLE android_metadata (locale TEXT);
CREATE TABLE 'MQTT_CHAT_ENTITY' ('_id' INTEGER PRIMARY KEY AUTOINCREMENT ,'MODE' INTEGER NOT NULL ,'SESSIONID' TEXT NOT NULL ,'FROM' TEX
T NOT NULL ,'TO' TEXT NOT NULL ,'V_CODE' TEXT,'TIMESTAMP' TEXT NOT NULL ,'PLATFORM' TEXT,'MESSAGE' TEXT,'ISREAD' INTEGER NOT NULL ,'GOSS
IPID' INTEGER,'GOSSIP' TEXT,'CHATTYPE' INTEGER NOT NULL ,'IMAGEPATH' TEXT,'BASE64IMAGE' TEXT);
CREATE TABLE 'SESSION_ENTITY' ('_id' INTEGER PRIMARY KEY AUTOINCREMENT ,'SESSIONID' TEXT NOT NULL UNIQUE ,'FROM' TEXT NOT NULL ,'TO' TEX
T NOT NULL ,'GOSSIPID' INTEGER NOT NULL ,'GOSSIP' TEXT,'SESSIONTYPE' INTEGER NOT NULL ,'ASDASD' INTEGER NOT NULL );
CREATE TABLE 'REPLAY_ENTITY' ('_id' INTEGER PRIMARY KEY AUTOINCREMENT ,'MODE' INTEGER NOT NULL ,'FROM' TEXT NOT NULL ,'TO' TEXT NOT NULL
 ,'V_CODE' TEXT,'TIMESTAMP' TEXT NOT NULL ,'PLATFORM' TEXT,'MESSAGE' TEXT,'MSGTYPE' INTEGER NOT NULL ,'ISREAD' INTEGER NOT NULL );
CREATE TABLE 'CUSTOMER' ('_id' INTEGER PRIMARY KEY ,'NAME' TEXT NOT NULL );
CREATE TABLE 'ORDERS' ('_id' INTEGER PRIMARY KEY ,'DATE' INTEGER,'CUSTOMER_ID' INTEGER NOT NULL );
sqlite>
```

```
sqlite> .table
CUSTOMER          ORDERS            SESSION_ENTITY
MQTT_CHAT_ENTITY  REPLAY_ENTITY     android_metadata
sqlite>
```

如上图，SESSION_ENTITY、MQTT_CHAT_ENTITY、REPLAY_ENTITY是我们创建的那三个表

```
sqlite> select * from SESSION_ENTITY;
3|abcdefg|A|B|10|大家好吗？我来了...|1|0
sqlite>
```

上面用select * from SESSION_ENTITY 可以查询到我们插入的那条数据

回过头来看最开始那张图：

Jave Object 相当于我们生成的MqttChatEntity、SessionEntity、ReplayEntity通过greenDAO对象话的去访问数据库，其实最重要的还是java工程和DBHelper的编写，因为其他都是自动生成的。

总结：

推荐大家一些链接：

官方提供的github地址，也有一个示例： https://github.com/greenrobot/greenDAO

DaoExample和DaoExampleGenerator。你可以clone到本地，运行或者直接在github上直接浏览。

如果你从git仓储中检出了DaoExample，可以直接像Android应用一样运行它。正如你所看到的，它就是一个简单的笔记本。可以添加新的note，

或者点击已存在的note进行删除。

其他关于GreenDao好的文章：

http://www.jcodecraeer.com/a/anzhuokaifa/androidkaifa/2014/1127/2069.html