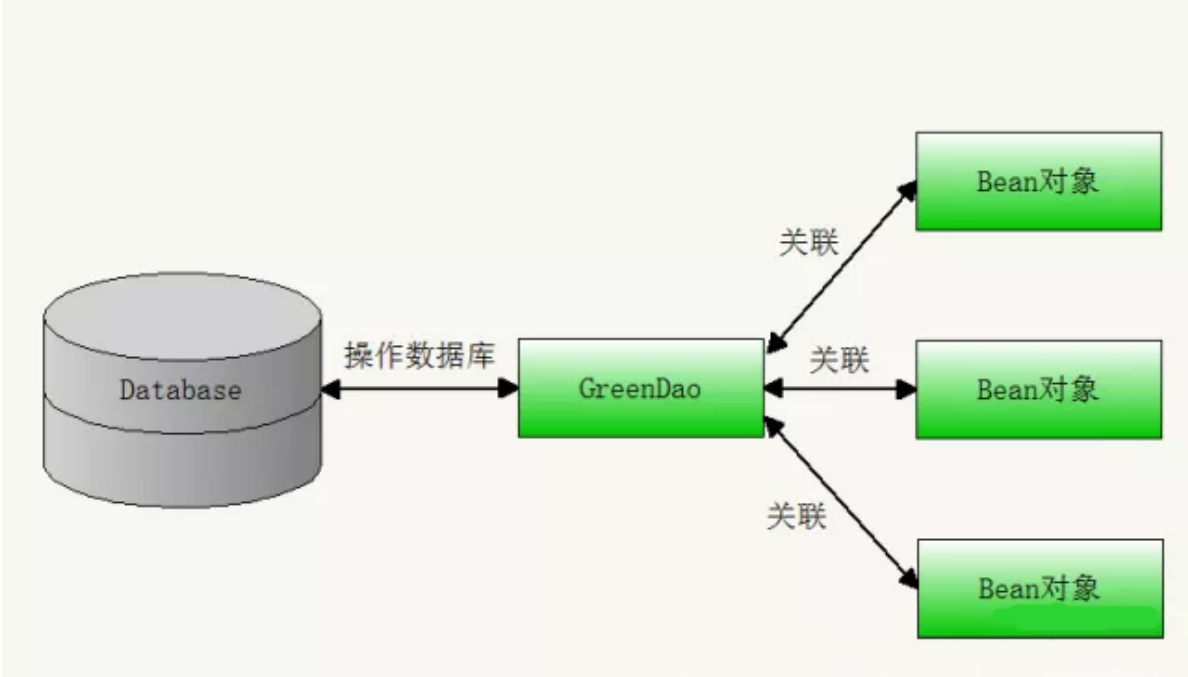
# 一、前言

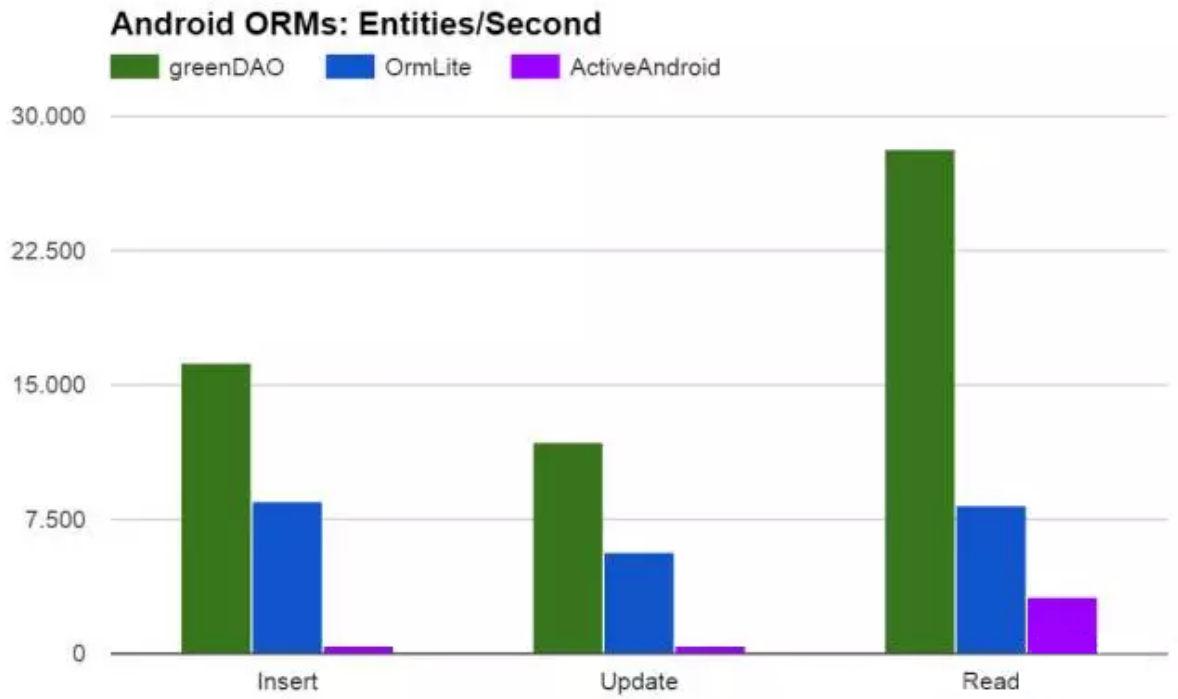
GreenDao是一款操作数据库的神器，经过了2.0版本的升级后，已经被广泛的开发者使用。确实是很好用，入门简单，可以剩去了数据库的建表操作和数据库SQL的编写，博主用了一次之后爱不释手，和以前的数据库操作一大堆的代码将它缩成了一句话，舒服

# 二、GreenDao3.2的简介

认识GreenDao之前必须知道ORM（Object Relation Mapping对象关系映射），其表现形式就是通过GreenDao将数据库和Bean对象关联起来，其表现形式如下图




GreenDao之所以很流行，跟它的优点是息息相关的，从官网中可以看到这样一张图，其表示了在主流的ORM第三方库中，其对数据库操作的速度是最快的



不仅如此，其优点还包括有以下几点

- 存取速度快
- 支持数据库加密
- 轻量级
- 激活实体
- 支持缓存
- 代码自动生成

# 三、GreenDao3.2的配置

> GreenDao的配置很简单，不过需要注意的是，有些人按照正确的配置后却频频出错，个人也经历过，最后的原因是网络有问题。因为校园网的DNS服务很差，所以解析不到GreenDao的依赖网站

**1、需要在工程（Project）的build.gradle中添加依赖**

```
buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.0.0'
        //GreenDao3依赖
        classpath 'org.greenrobot:greendao-gradle-plugin:3.2.1'
    }
}
```

**2、在项目（Module）的build.gradle中添加依赖**

```
apply plugin: 'com.android.application'
//使用greendao
apply plugin: 'org.greenrobot.greendao'
android {
    compileSdkVersion 23
    buildToolsVersion "23.0.2"
    defaultConfig {
        applicationId "com.handsome.didi"
        minSdkVersion 14
        targetSdkVersion 23
        versionCode 1
        versionName "1.0"
    }
    //greendao配置
    greendao {
        //版本号，升级时可配置
        schemaVersion 1
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
                    'proguard-rules.pro'
        }
    }
}
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    testCompile 'junit:junit:4.12'
    compile 'com.android.support:appcompat-v7:23.1.1'
    compile 'org.greenrobot:greendao:3.2.0' //greendao依赖
}
```

到这里就配置成功了

# 四、GreenDao3.2的使用

> 配置完成后，最重要的就是GreenDao的使用了，或许使用过Bmob第三方后端云的同学会知道，他们的API有些相像，都是通过API来拼装SQL语句的

下面就以购物车的实战来使用GreenDao，这里的购物车展示图如下


> 我们所知道的数据库操作需要：数据库名、表名、字段名，缺一不可，下面就是这三项的创建

## 1、创建Bean对象（表名和字段名）

GreenDao需要创建Bean对象之后，该Bean对象就是表名，而它的属性值就是字段名，其实现是通过注释的方式来实现的，下面是购物车的Bean对象（每个Bean对象对应一张表）

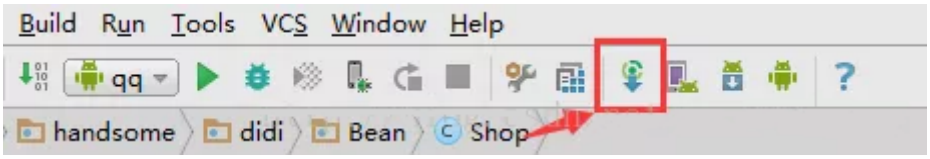
```java
@Entity
public class Shop{

    //表示为购物车列表
    public static final int TYPE_CART = 0x01;
    //表示为收藏列表
    public static final int TYPE_LOVE = 0x02;

    //不能用int
    @Id(autoincrement = true)
    private Long id;
    //商品名称
    @Unique
    private String name;
    //商品价格
    @Property(nameInDb = "price")
    private String price;
    //已售数量
    private int sell_num;
    //图标url
    private String image_url;
    //商家地址
    private String address;
    //商品列表类型
    private int type;
}
```

这里需要注意的是，创建完成之后，需要build gradle来完成我们的代码自动生成。自动生成的代码有

1. Bean实体的构造方法和get、set方法
2. DaoMaster、DaoSession、DAOS类




这里对Bean对象的注释进行解释

1. @Entity：告诉GreenDao该对象为实体，只有被@Entity注释的Bean类才能被dao类操作
2. @Id：对象的Id，使用Long类型作为EntityId，否则会报错。(autoincrement = true)表示主键会自增，如果false就会使用旧值
3. @Property：可以自定义字段名，注意外键不能使用该属性
4. @NotNull：属性不能为空
5. @Transient：使用该注释的属性不会被存入数据库的字段中
6. @Unique：该属性值必须在数据库中是唯一值
7. @Generated：编译后自动生成的构造函数、方法等的注释，提示构造函数、方法等不能被修改

## 2、创建数据库（数据库名）

数据库的表名和字段都建好了，下面差个数据库的创建，下面通过传统和GreenDao的比较来体验其优点

① 传统的数据库创建

```java
public class CommonOpenHelper extends SQLiteOpenHelper {

    private static CommonOpenHelper helper;

    public static CommonOpenHelper getInstance(Context context) {
        if (helper == null) {
            helper = new CommonOpenHelper(context, "common.db", null, 1);
        }
        return helper;
    }
    private CommonOpenHelper(Context context, String name,
          SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }
    @Override
    public void onCreate(SQLiteDatabase db) {
        //创建love表
        db.execSQL("create table love(" +
                "id integer primary key autoincrement, " +
                "name varchar, " +
                "price varchar, " +
                "sell_num integer, " +
                "image_url varchar, " +
                "address varchar" +
                ")");
    }
    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
    }
}
```

② GreenDao数据库创建

```java
public class BaseApplication extends Application {

    private static DaoSession daoSession;

    @Override
    public void onCreate() {
        super.onCreate();
        //配置数据库
        setupDatabase();
    }

    /**
     * 配置数据库
     */
    private void setupDatabase() {
        //创建数据库shop.db"
        DaoMaster.DevOpenHelper helper = new DaoMaster.DevOpenHelper(this,
                "shop.db", null);
        //获取可写数据库
        SQLiteDatabase db = helper.getWritableDatabase();
        //获取数据库对象
        DaoMaster daoMaster = new DaoMaster(db);
        //获取Dao对象管理者
        daoSession = daoMaster.newSession();
    }

    public static DaoSession getDaoInstant() {
        return daoSession;
    }
}
```

可以发现，GreenDao已经将我们的数据库创建缩成几句话，代码会自动将Bean对象创建成表，不再是传统的手写SQL语句。这里的数据库创建只需要在Application中执行一次即可，这里对几个类进行解释

- DevOpenHelper：创建SQLite数据库的SQLiteOpenHelper的具体实现
- DaoMaster：GreenDao的顶级对象，作为数据库对象、用于创建表和删除表

- DaoSession：管理所有的Dao对象，Dao对象中存在着增删改查等API

由于我们已经创建好了DaoSession和Shop的Bean对象，编译后会自动生成我们的ShopDao对象，可通过DaoSession获得

```
ShopDao dao = daoSession.getShopDao();
```

这里的Dao（Data Access Object）是指数据访问接口，即提供了数据库操作一些API接口，可通过dao进行增删改查操作

## 3、数据库的增删改查

数据库的表名、字段、数据库都建好了，下面就通过传统和GreenDao对数据库的操作来比较体验其优点

① 传统的增删改查

```
//采用ContentProvider进行增删改查
public class CartDao {
    //添加数据
    public static boolean insertCart(ContentResolver resolver, Shop shop) {
        ContentValues values = new ContentValues();
        values.put("name", shop.getName());
        values.put("price", shop.getPrice());
        values.put("sell_num", shop.getSell_num());
        values.put("image_url", shop.getImage_url());
        values.put("address", shop.getAddress());
        resolver.insert(MyCartProvider.URI.CODE_CART_INSERT, values);
        BaseApplication.getDaoInstant().getShopDao().insert(shop);
        return true;
    }
    //删除数据
    public static void deleteCart(ContentResolver resolver, int id) {
        resolver.delete(MyCartProvider.URI.CODE_CART_DELETE, "id = " + id, null);
    }
    //查询数据
    public static List<Shop> queryCart(ContentResolver resolver) {
        List<Shop> list = new ArrayList<Shop>();
        String[] projection = {"id", "name", "price", "sell_num",
                "image_url", "address"};
        Cursor cursor = resolver.query(MyCartProvider.URI.CODE_CART_QUERY,
                projection, null, null, null);
        while (cursor.moveToNext()) {
            Shop shop = new Shop();
            shop.setId(cursor.getLong(cursor.getColumnIndex("id")));
            shop.setName(cursor.getString(cursor.getColumnIndex("name")));
            shop.setPrice(cursor.getString(cursor.getColumnIndex("price")));
            shop.setSell_num(cursor.getInt(cursor.getColumnIndex("sell_num")));
            shop.setImage_url(cursor.getString(cursor.getColumnIndex("image_url")));
            shop.setAddress(cursor.getString(cursor.getColumnIndex("address")));
            list.add(shop);
        }
        return list;
    }
    /**
     * 省略更新数据
     */
}
```

② GreenDao增删改查

```
public class LoveDao {

    //添加数据，如果有重复则覆盖
    public static void insertLove(Shop shop) {
        BaseApplication.getDaoInstant().getShopDao().insertOrReplace(shop);
    }
    // 删除数据
    public static void deleteLove(long id) {
        BaseApplication.getDaoInstant().getShopDao().deleteByKey(id);
    }

    // 更新数据
    public static void updateLove(Shop shop) {
        BaseApplication.getDaoInstant().getShopDao().update(shop);
    }

    //查询条件为Type=TYPE_LOVE的数据
    public static List<Shop> queryLove() {
        return BaseApplication.getDaoInstant().getShopDao()
                .queryBuilder()
                .where(ShopDao.Properties.Type.eq(Shop.TYPE_LOVE)).list();
    }

    //查询全部数据
    public static List<Shop> queryAll() {
        return BaseApplication.getDaoInstant().getShopDao().loadAll();
    }

}
```

效果很明显，GreenDao的封装更加短小精悍，语义明朗，下面对GreenDao中

# 五、Dao对象其他API的介绍

- 增加单个数据
  - getShopDao().insert(shop);
  - getShopDao().insertOrReplace(shop);
- 增加多个数据
  - getShopDao().insertInTx(shopList);
  - getShopDao().insertOrReplaceInTx(shopList);
- 查询全部
  - List< Shop> list = getShopDao().loadAll();
  - List< Shop> list = getShopDao().queryBuilder().list();
- 查询附加单个条件
  - .where()
  - .whereOr()
- 查询附加多个条件
  - .where(, , ,)
  - .whereOr(, , ,)
- 查询附加排序
  - .orderDesc()
  - .orderAsc()
- 查询限制当页个数
  - .limit()
- 查询总个数
  - .count()
- 修改单个数据
  - getShopDao().update(shop);

- 修改多个数据
  - getShopDao().updateInTx(shopList);
- 删除单个数据
  - getTABUserDao().delete(user);
- 删除多个数据
  - getUserDao().deleteInTx(userList);
- 删除数据ByKey
  - getTABUserDao().deleteByKey();