一个精简的库

性能最大化

内存开销最小化

易于使用的 APIs

对 Android 进行高度优化

GREENDAO 设计的主要特点

greenDAO 性能远远高于同类的 ORMLite，具体测试结果可见官网

greenDAO 支持 protocol buffer(protobuf) 协议数据的直接存储，如果你通过 protobuf 协议与服务器交互，将不需要任何的映射。

与 ORMLite 等使用注解方式的 ORM 框架不同，greenDAO 使用「Code generation」的方式，这也是其性能能大幅提升的原因。

Greendao的使用：

greendao中的注解

(一) @Entity 定义实体

@nameInDb 在数据库中的名字，如不写则为实体中类名

@indexes 索引

@createInDb 是否创建表，默认为true,false时不创建

@schema 指定架构名称为实体

@active 无论是更新生成都刷新

(二) @Id

(三) @NotNull 不为null

(四) @Unique 唯一约束

(五) @ToMany 一对多

(六) @OrderBy 排序

(七) @ToOne 一对一

(八) @Transient 不存储在数据库中

(九) @generated 由greendao产生的构造函数或方法

1.）实体@Entity注解

schema：告知GreenDao当前实体属于哪个schema

active：标记一个实体处于活动状态，活动实体有更新、删除和刷新方法

nameInDb：在数据中使用的别名，默认使用的是实体的类名

indexes：定义索引，可以跨越多个列

createInDb：标记创建数据库表\*\*

2.）基础属性注解

@Id :主键 Long型，可以通过@Id(autoincrement = true)设置自增长

@Property：设置一个非默认关系映射所对应的列名，默认是的使用字段名举例：@Property (nameInDb="name")

@NotNul：设置数据库表当前列不能为空

@Transient：添加次标记之后不会生成数据库表的列

3.)索引注解

@Index：使用@Index作为一个属性来创建一个索引，通过name设置索引别名，也可以通过unique给索引添加约束

@Unique：向数据库列添加了一个唯一的约束

4.）关系注解

@ToOne：定义与另一个实体（一个实体对象）的关系

@ToMany：定义与多个实体对象的关系

在建立数据库时，每一个实体类会建立一张数据表，代表一个关系，而不同实体之间必然存在一定的关系，反映到数据表上也需要建立关系。

比如一个用户的账户都有对应的头像picture，且假设每个用户的头像Picture对象都不同，因此每个picture也对应一个用户，这就是一对一的关系，而在网上购物可以下订单，每个用户可以有多个订单Oder，而每个Oder都对应一个用户User，这是一对多的关系。在数据表的关系中可以通过外键表示这种一对一和一对多的关系。除此以外还有多对多的关系，在对象的结构中每个Oder包含多个物品Item，而每个Item也可能包含在多个Oder中，因此这是多对多的关系，在数据表中需要额外的表表示对应关系，如Oder\_Item表，表示一个Oder与一个Item具有关系。

在greenDao中，使用@ToOne表示一对一的关系，使用@ToMany表示一对多的关系，而多对多的关系支持度不够好，目前还不完善，使用较为复杂。

@ToOne

比如每个用户都有一个对应Picture属性，需要一个pictureId代表这个Picture属性，通过@ToOne(joinProperty = "XXXX")指定pictureId，在数据表中则会有pictureId这一列作为外键，与Picture数据表建立联系，如果你没有指定pictureId, greenDAO也会在数据表中生成一列属性其作用与指定的pictureId相同，而实体类中则可以使用User的Picture属性，代码如下：

@Entity

public class User {

@Id

private Long id;

private String name;

private Long pictureId;

@ToOne(joinProperty = "pictureId")

private Picture picture;

....

}

当然greenDAO会为我们生成其他的代码，如构造器，getter和setter等，在代码中可以直接使用user.getPicture()和setPicture，而在数据表中则是pictureId是指向Picture数据表的外键。Picture的实体类如下：

@Entity

public class Picture {

@Id

private long pictureId;

...

}

@ToMany

一对多的关系，定义了一个实体对象对应着多个实体对象，比如一个用户对应多个Order, 在建立数据表示会在目标实体（即一对多的那个多的实体类）的数据表中建立外键，指向源实体类（一对多中的一那个实体类）的数据表。目标数据表中的外键属性由@ToMany(referencedJoinProperty = "XXXX")指定。

如User中添加List<Order> orders属性：

@Entity

public class User {

@Id

private Long id;

private String name;

private Long pictureId;

@ToOne(joinProperty = "pictureId")

private Picture picture;

@ToOne

private Picture thumbnailPicture;

@ToMany(referencedJoinProperty = "ownerId")

private List<Order> oders;

...

}

Order的实例代码为：

@Entity

public class Order {

@Id

private long id;

private long ownerId;

...

}

在build->Make Project之后会生成对应代码，此时在代码中可以使用User#getOrders()获取user的所有对应的order的List. 在这种情况下， ownerId是在Order中的外键，指向User的主键Id。

5.）排序算法

[java] view plain copy

List<User> userList = userDao.queryBuilder()

.where(UserDao.Properties.Id.notEq(999))

.orderAsc(UserDao.Properties.Id)

.limit(5)

.build().list();

[java] view plain copy

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();

当然，也可以通过纯SQL语句去查询，但不推荐。

先通过queryBuilder生成查找构造器，然后写相应条件进行查询

where 条件判断

orderAsc 排序

limit 查询条数

list() 查询结果为一个集合

unique()或uniqueOrThrow()，返回单个结果，如果没有满足条件的结果，前者返回null， 后者抛出异常

方法介绍：

查询