# Butterknife依赖注入

Module:app 中build.gradle

apply **plugin**: **'com.jakewharton.butterknife'**

compile **'com.jakewharton:butterknife:8.7.0'**annotationProcessor **'com.jakewharton:butterknife-compiler:8.7.0'**

Project:build.gradle中：

classpath **'com.jakewharton:butterknife-gradle-plugin:8.7.0'**

备注：查看版本号地址：<https://github.com/JakeWharton/butterknife>

# 2.GreenDao数据库ORM

## 基本介绍

greenDAO3开始使用注解的方式定义实体类（entity），并且是通过安装gradle插件来生成代码

## 导入相关的包（添加相关依赖）

Module:app 中build.gradle

apply **plugin**: **'org.greenrobot.greendao'** greendao {  
 schemaVersion 1  
 daoPackage **'com.a1900.android.study\_android.greendao'** targetGenDir **'src/main/java'**}

dependencies{

compile **'org.greenrobot:greendao:3.0.1'**compile **'org.greenrobot:greendao-generator:3.0.0'**

}

Project:build.gradle中：

classpath **'org.greenrobot:greendao-gradle-plugin:3.0.0'**

备注：

 schemaVersion---->指定数据库schema版本号，迁移等操作会用到

daoPackage-------->通过gradle插件生成的数据库相关文件的包名，默认为你的entity所在的包名

    targetGenDir-------->这就是我们上面说到的自定义生成数据库文件的目录了，可以将生成的文件放到我们的java目录中，而不是build中，这样就不用额外的设置资源目录了

## 编写实体类

1. @Entity
2. **public** **class** User {
3. @Id
4. **private** Long id;
5. **private** String name;
6. @Transient
7. **private** **int** tempUsageCount; // not persisted
8. }
9. @Entity：将我们的java普通类变为一个能够被greenDAO识别的数据库类型的实体类
10. @Id：通过这个注解标记的字段必须是Long类型的，这个字段在数据库中表示它就是主键，并且它默认就是自增的
11. @Transient：表明这个字段不会被写入数据库，只是作为一个普通的java类字段，用来临时存储数据的，不会被持久化

## Make Project (Build—>make project)

daoPackage **'com.a1900.android.study\_android.greendao'**

**就会在这个目录下生成相关的代码文件**

## 使用

UserDao userDao = GreenDaoManager.*getInstance*().getSession().getUserDao();

得到userDao就可以CURD了

## 外加浏览器测试db的AS依赖

debugCompile **'com.amitshekhar.android:debug-db:1.0.0'**

在运行项目的时候，测试机要和电脑在局域网下，logout里面有IP直接打开就可以在浏览器里面查看数据库了

## 增

User user = new User(null,”weiguo”);

userDao.insert(user);

insert方法返回long，可以判断是否成功

## 删(先查询出来对象再删除)

1. User findUser = userDao.queryBuilder().where(UserDao.Properties.Name.eq("wyk")).build().unique();
2. **if**(findUser != **null**){
3. userDao.deleteByKey(findUser.getId());
4. }

## 改

1. User findUser = userDao.queryBuilder().where(UserDao.Properties.Name.eq("wyk")).build().unique();
2. **if**(findUser != **null**) {
3. findUser.setName(newName);
4. userDao.update(findUser);
5. } **else** {
6. Toast.makeText(MyApplication.getContext(), "用户不存在", Toast.LENGTH\_SHORT).show();
7. }

## 查

userDao.queryBuilder().where(UserDao.Properties.Name.eq("wyk")).build().unique();

## 自定义sql

有的时候需要用到group by或者left join等复杂的语句，可以调用android原生的sqlite去进行查询。

1. ChatHistoryDao dao = GreenDaoManager.getInstance().getSession().getChatHistoryDao();
2. Cursor cursor = dao.getDatabase().rawQuery("select t.sales\_wx\_nick\_name,t.wx\_nick\_name,count(\*),t.talker\_id,t.sales\_wx\_account from chat\_history t group by t.talker\_id,t.sales\_wx\_account order by t.created\_at desc", **null**);
3. **while** (cursor.moveToNext()) {
4. String salesWxNickName = cursor.getString(0);
5. String clientWxNickName = cursor.getString(1);
6. **int** chatCount = cursor.getInt(2);
7. **int** talkerId = cursor.getInt(3);
8. String salesWxAccount = cursor.getString(4);
9. }

# 3.FlycoDialog万能(Dialog)

## Demo APK

## 效果截图

[E:\coding\github\mDialog\FlycoDialog\_Master-master\gif](file:///E:\coding\github\mDialog\FlycoDialog_Master-master\gif)

## 使用

<https://github.com/H07000223/FlycoDialog_Master>

Module:app 中build.gradle

After v1.3.0

dependencies{

compile 'com.flyco.dialog:FlycoDialog\_Lib:1.3.2'

}

# 4.动态权限申请

## 地址

<https://github.com/lovedise/PermissionGen>

## 使用

Module:app 中build.gradle

compile 'com.lovedise:permissiongen:0.1.1'

1. 当你需要动态申请有关权限时

PermissionGen.with(MainActivity.this)

.addRequestCode(100)

.permissions(

Manifest.permission.READ\_CONTACTS,

Manifest.permission.RECEIVE\_SMS,

Manifest.permission.WRITE\_CONTACTS)

.request();

或者（推荐使用）

PermissionGen.needPermission(ContactFragment.this, 100,

new String[] {

Manifest.permission.READ\_CONTACTS,

Manifest.permission.RECEIVE\_SMS,

Manifest.permission.WRITE\_CONTACTS

}

);

1. 在activity 或者 fragment 中重写

@Override public void onRequestPermissionsResult(int requestCode, String[] permissions,int[] grantResults) {

PermissionGen.onRequestPermissionsResult(this, requestCode, permissions, grantResults);

}

1. 权限申请成功了

@PermissionSuccess(requestCode = 100)

public void doSomething(){

Toast.makeText(this, "Contact permission is granted", Toast.LENGTH\_SHORT).show();

}

1. 权限申请失败了

@PermissionFail(requestCode = 100)

public void doFailSomething(){

Toast.makeText(this, "Contact permission is not granted", t.LENGTH\_SHORT).show();

}

# APK更新

# Glide-transformations图片加载缓存处理

## 地址

<https://github.com/wasabeef/glide-transformations>

## 效果图

[E:\coding\github\mLoadingImage\glide-transformations-master\art](file:///E:\coding\github\mLoadingImage\glide-transformations-master\art)

## 使用

**注意：加载网络图片的时候需要加网路权限**

<**uses-permission android:name="android.permission.INTERNET"** />

1. **加载网络图片**

Glide.*with*(**this**).load(**"http://pic6.huitu.com/res/20130116/84481\_20130116142820494200\_1.jpg"**).bitmapTransform(**new** CropCircleTransformation(**this**))  
 .into(**mTestImageBt**);

CropCircleTransformation : 圆形

1. **加载本地图片**

Glide.*with*(**this**).load(R.drawable.***ic\_demo\_org***).bitmapTransform(**new** CropCircleTransformation(**this**))  
 .into(**mTestImageBt**);