**1、在app/build.gradle中添加以下依赖**

implementation 'android.arch.persistence.room:runtime:1.0.0'

annotationProcessor 'android.arch.persistence.room:compiler:1.0.0'

**最新版本号：https://developer.android.com/topic/libraries/architecture/adding-components**

**2、创建JavaBean**

@Entity

public class User {

@PrimaryKey(autoGenerate = true)*//主键是否自动增长，默认为false*

private int id;

private String name;

private int age;

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getAge() {

return age;

}

public void setAge(int age) {

this.age = age;

}

*//这里的getter/setter方法是必须的*

*//这里的getter/setter方法是必须的*

*//这里的getter/setter方法是必须的*

*//重要的事说三遍*

}

* 这里需要使用@Entity来注解该类
* 至少要有一个主键@PrimaryKey

**3、创建Dao**  
接下来，需要为我们的实体创建DAO。 DAO代表数据访问对象，所以它是告诉我们的数据库如何操作数据的一种方式：

@Dao

public interface UserDao {

@Query("SELECT \* FROM user")

List<User> getAllUsers();

@Insert

void insert(User... users);

@Update

void update(User... users);

@Delete

void delete(User... users);

}

* 使用@Dao注解该接口
* @Insert, @Update, @Delete,@Query代表我们常用的插入、更新、删除、查询数据库操作

@Insert, @Update, @Delete可以传入多种不同的参数。例如：

@Insert

void insert(User... users);

@Insert

void insert(User user);

@Insert

void insert(List<User> userLists);

同理，@Query也可以返回多种不同的类型。

@Query("SELECT \* FROM user")

List<User> getAllUsers();

@Query("SELECT \* FROM user WHERE id=:id")

User getUser(int id);

@Query("SELECT \* FROM user")

Cursor getUserCursor();

当然，除了这些我们还可以传入一些限制符进去。例如，

@Query("SELECT \* FROM user WHERE age=:age")

List<User> getUsersByAge(int age);

@Query("SELECT \* FROM user WHERE age=:age LIMIT :max")

List<User> getUsersByAge(int max, int... age);

**4、创建数据库**

@Database(entities = { User.class }, version = 1,exportSchema = false)

public abstract class UserDatabase extends RoomDatabase {

private static final String DB\_NAME = "UserDatabase.db";

private static volatile UserDatabase instance;

static synchronized UserDatabase getInstance(Context context) {

if (instance == null) {

instance = create(context);

}

return instance;

}

private static UserDatabase create(final Context context) {

return Room.databaseBuilder(

context,

UserDatabase.class,

DB\_NAME).build();

}

public abstract UserDao getUserDao();

}

这里使用@Database注解该类并添加了**表名**、**数据库版本**（每当我们改变数据库中的内容时它都会增加），所以这里使用exportSchema = false

注意：除了添加表映射的类以及和数据库版本外，还要添加exportSchema = false否则会报警告。  
Error:(10, 17) 警告: Schema export directory is not provided to the annotation processor so we cannot export the schema. You can either provide room.schemaLocation annotation processor argument OR set exportSchema to false.

我们抽象了getUserDao()方法，这是必要的。

**5、使用数据库**  
我们终于能够操作我们的数据库了。但是所有的操作必须在后台线程中完成。你可以通过使用AsyncTask，Thread，Handler，RxJava或其它方式来完成。  
如果没有在后台线程执行，并且也没有说明可以在主线程操作的话，就会报以下错误。

Caused by: java.lang.IllegalStateException: Cannot access database on the main thread since it may potentially lock the UI for a long period of time.

例如，我们可以像下面这样插入数据

User user=new User();

user.setName("name1");

user.setAge(18);

UserDatabase

.getInstance(context)

.getUserDao()

.insert(user);

或者这样

List<User> allUsers = UserDatabase

.getInstance(RoomActivity.this)

.getUserDao()

.getAllUsers();