Android数据库高手秘籍(2)：创建表和LitePal的基本用法

2015-09-06 安卓应用频道

(点击上方蓝字，可快速关注我们)

> 来源：郭霖
> 网址：http://blog.csdn.net/guolin_blog/article/details/38556989

上一篇文章中我们学习了一些Android数据库相关的基础知识，和几个颇为有用的SQLite命令，都是直接在命令行操作的。但是我们都知道，数据库是要和程序结合在一起使用的，单独对一个数据库去进行增删改查操作并没有什么意义，因此今天我们就来学习一下如何在Android程序当中去操作SQLite数据库，还没看过前一篇文章的朋友可以先去参考 Android数据库高手秘籍(1)：SQLite命令 。

操作数据库的第一步当然是创建表了，传统创建表的方法相信大多数人都知道，那么今天我除了会展示传统的建表方法之外，还会讲解LitePal这个框架的基本用法，并使用它来完成同样的建表操作，让大家体会到使用框架来操作数据库的魅力。

那么先来简单介绍一下吧，LitePal是一款开源的Android数据库框架，它采用了对象关系映射(ORM)的模式，并将我们平时开发时最常用到的一些数据库功能进行了封装，使得不用编写一行SQL语句就可以完成各种建表、增删改查的操作。并且LitePal很“轻”，jar包只有100k不到，而且近乎零配置，这一点和Hibernate这类的框架有很大区别。目前LitePal的源码已经托管到了GitHub上，地址是 https://github.com/LitePalFramework/LitePal 。

OK，简单介绍完了LitePal，我们还是先来看一下，在传统的Android开发中，需要怎么去创建表。

传统的建表方式

其实为了方便我们对数据库表进行管理，Android本身就提供了一个帮助类：SQLiteOpenHelper。这个类集创建和升级数据库于一身，并且自动管理了数据库版本，算是一个非常好用的工具。

那我们现在就来试试SQLiteOpenHelper的用法吧。首先你要知道SQLiteOpenHelper是一个抽象类，这意味着如果我们想要使用它的话，就需要创建一个自己的帮助类去继承它。SQLiteOpenHelper中有两个抽象方法，分别是onCreate()和onUpgrade()，我们必须在自己的帮助类里面重写这两个方法，然后分别在这两个方法中去实现创建、升级数据库的逻辑。

本篇文章只需要把注意力放在创建数据库这里就行了，升级数据库我们会在下一篇文章中去讨论。

新建一个MySQLiteHelper类并让它继承SQLiteOpenHelper，这样一个最基本的数据库帮助类的代码如下所示：

```
public class MySQLiteHelper extends SQLiteOpenHelper {

    public MySQLiteHelper(Context context, String name, CursorFactory factory,
            int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    }

}
```

其中，当数据库创建的时候会调用onCreate()方法，在这里去执行建表操作就可以了。比如说我们想新建一张news表，其中有title，content，publishdate，commentcount这几列，分别代表着新闻标题、新闻内容、发布时间和评论数，那么代码就可以这样写：

```
public class MySQLiteHelper extends SQLiteOpenHelper {

    public static final String CREATE_NEWS = "create table news ("
            + "id integer primary key autoincrement, "
            + "title text, "
            + "content text, "
            + "publishdate integer,"
            + "commentcount integer)";

    public MySQLiteHelper(Context context, String name, CursorFactory factory,
            int version) {
        super(context, name, factory, version);
    }
```

```
    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_NEWS);
    }
    ...
}
```

可以看到，我们把建表语句定义成了一个常量，然后在onCreate()方法中去执行了这条建表语句，news表也就创建成功了。这条建表语句虽然简单，但是里面还是包含了一些小的细节，我来解释一下。首先，根据数据库的范式要求，任何一张表都应该是有主键的，所以这里我们添加了一个自增长的id列，并把它设为主键。然后title列和content列都是字符串类型的，commentcount列是整型的，这都很好理解，但是publishdate列该怎么设计呢？由于SQLite中并不支持存储日期这种数据类型，因此我们需要将日期先转换成UTC时间(自1970年1月1号零点)的毫秒数，然后再存储到数据库中，因此publishdate列也应该是整型的。

现在，我们只需要获取到SQLiteDatabase的实例，数据库表就会自动创建了，如下所示：

```
SQLiteOpenHelper dbHelper = new MySQLiteHelper(this, "demo.db", null, 1);
SQLiteDatabase db = dbHelper.getWritableDatabase();
```

感觉很简单很方便是吗？那你就太容易满足了，下面我们就来学习一下LitePal的基本用法，看一看使用这个框架是如何实现同样的功能的。

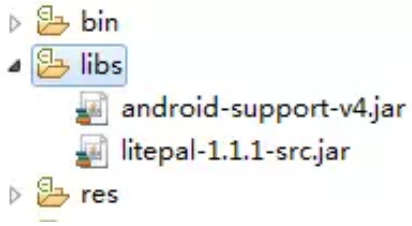
LitePal的基本用法

虽说LitePal宣称是近乎零配置，但也只是“近乎”而已，它还是需要进行一些简单配置才可以使用的，那么我们第一步就先快速学习一下LitePal的配置方法。

快速配置

1. 引入Jar包或源码

首先我们需要将LitePal的jar包引入到项目当中，可以点击这里查看LitePal的最新版本，选择你需要的下载即可。下载好了jar包之后，把它复制到项目的libs目录中就算是引入成功了，如下图所示：



如果你不想用jar包的话，也可以把LitePal的源码下载下来，然后作为一个library库导入到Eclipse当中，再让我们的项目去引用这个library库就可以了。

2. 配置litepal.xml

接着在项目的assets目录下面新建一个litepal.xml文件，并将以下代码拷贝进去：

```
<?xml version="1.0" encoding="utf-8"?>
<litepal>
    <dbname value="demo" ></dbname>

    <version value="1" ></version>

    <list>
    </list>
</litepal>
```

配置文件相当简单，<dbname>用于设定数据库的名字，<version>用于设定数据库的版本号，<list>用于设定所有的映射模型，我们稍后就会用到。

3. 配置LitePalApplication

由于操作数据库时需要用到Context，而我们显然不希望在每个接口中都去传一遍这个参数，那样操作数据库就显得太繁琐了。因此，LitePal使用了一个方法来简化掉Context这个参数，只需要在AndroidManifest.xml中配置一下LitePalApplication，所有的数据库操作就都不用再传Context了，如下所示：

```
<mainfest>
    <application
        android:name="org.litepal.LitePalApplication"
        ...
    >
    ...
    </application>
```

```
</manifest>
```

当然，有些程序可能会有自己的Application，并在这里配置过了。比如说有一个MyApplication，如下所示：

```
<manifest>
  <application
    android:name="com.example.MyApplication"
    ...
  >
  ...
  </application>
</manifest>
```

没有关系，这时只需要修改一下MyApplication的继承结构，让它不要直接继承Application类，而是继承LitePalApplication类，就可以使用一切都能正常工作了，代码如下所示：

```
public class MyApplication extends LitePalApplication {
  ...
}
```

但是，有些程序可能会遇到一些更加极端的情况，比如说MyApplication需要继承另外一个AnotherApplication，并且这个AnotherApplication还是在jar包当中的，不能修改它的代码。这种情况应该算是比较少见了，但是如果你遇到了的话也不用急，仍然是有解释方案的。你可以把LitePal的源码下载下来，然后把src目录下的所有代码直接拷贝到你项目的src目录下面，接着打开LitePalApplication类，将它的继承结构改成继承自AnotherApplication，再让MyApplication继承自LitePalApplication，这样所有的Application就都可以在一起正常工作了。

仅仅三步，我们就将所有的配置工作全部完成了，并且这是一件一本万利的事情，自此以后，你就可以开心地体验LitePal提供的各种便利了，就让我们从建表开始吧。

开始建表

前面在介绍的时候已经说了，LitePal采取的是对象关系映射(ORM)的模式，那么什么是对象关系映射呢？简单点说，我们使用的编程语言是面向对象语言，而我们使用的数据库则是关系型数据库，那么将面向对象的语言和面向关系的数据库之间建立一种映射关系，这就是对象关系映射了。

但是我们为什么要使用对象关系映射模式呢？这主要是因为大多数的程序员都很擅长面向对象编程，但其中只有少部分的人才比较精通关系型数据库。而且数据库的SQL语言晦涩难懂，就算你很精通它，恐怕也不喜欢经常在代码中去写它吧？而对象关系映射模式则很好地解决了这个问题，它允许我们使用面向对象的方式来操作数据库，从而可以从晦涩难懂的SQL语言中解脱出来。

那么接下来我们就看一看LitePal中是如何建表的吧。根据对象关系映射模式的理念，每一张表都应该对应一个模型(Model)，也就是说，如果我们想要建一张news表，就应该有一个对应的News模型类。新建一个News类，如下所示：

```
package com.example.databasetest.model;

public class News {
}
```

然后，表中的每一列其实就是对应了模型类中的一个字段，比如news表中有id、title、content、publishdate、commentcount这几个列，那么在News类中就也应该有这几个字段，代码如下所示：

```
public class News {

    private int id;

    private String title;

    private String content;

    private Date publishDate;

    private int commentCount;

    // 自动生成get、set方法
    ...
}
```

其中id这个字段可写可不写，因为即使不写这个字段，LitePal也会在表中自动生成一个id列，毕竟每张表都一定要有主键的嘛。

这里我要特别说明一下，LitePal的映射规则是非常轻量级的，不像一些其它的数据库框架，需要为每个模型类单独配置一个映射关系的XML，LitePal的所有映射都是自动完成的。根据LitePal的数据类型支持，可以进行对象关系映射的数据类型一共有8种，int、short、long、

float、double、boolean、String和Date。只要是声明成这8种数据类型的字段都会被自动映射到数据库表中，并不需要进行任何额外的配置。

那么有的朋友可能会问了，既然是自动映射的话，如果News类中有一个字符串字段我并不想让它映射到数据库表中，这该怎么办呢？对此，LitePal同样采用了一种极为轻量的解决方案，只有声明成private修饰符的字段才会被映射到数据库表中，如果你有某一个字段不想映射的话，只需要将它改成public、protected或default修饰符就可以了。

现在模型类已经建好了，我们还差最后一步，就是将它配置到映射列表当中。编辑assets目录下的litepal.xml文件，在<list>标签中加入News模型类的声明：

```
<?xml version="1.0" encoding="utf-8"?>
<litepal>
    <dbname value="demo" ></dbname>

    <version value="1" ></version>

    <list>
        <mapping class="com.example.databasetest.model.News"></mapping>
    </list>
</litepal>
```

注意这里一定要填入News类的完整类名。

OK，这样所有的工作就都已经完成了，现在只要你对数据库有任何的操作，news表就会被自动创建出来。比如说LitePal提供了一个便捷的方法来获取到SQLiteDatabase的实例，如下所示：

```
SQLiteDatabase db = Connector.getDatabase();
```

调用一下上述代码，news表就应该已经创建成功了。我们使用在上一篇文章中学到的SQLite命令来查看一下，打开demo.db数据库，输入.table命令，结果如下图所示：

可以看到，news表已经存在了。另外两张android_metadata和table_schema表是自动生成的，我们不用理。接下来我们还可以再查询一下news表的建表语句，如下图所示：

```
sqlite> select * from sqlite_master where name='news';
select * from sqlite_master where name='news';
    type = table
    name = news
tbl_name = news
rootpage = 6
     sql = CREATE TABLE news (id integer primary key autoincrement,content text,
 title text, commentcount integer, publishdate integer)
sqlite>
```

这就是LitePal根据News类中的字段自动帮我们生成的建表语句，由此也说明，建表操作已经成功完成了。

好了，到目前为止你已经算是对LitePal的用法有点入门了，那么本篇文章的内容就到这里，下篇文章当中我们将学习使用LitePal进行升级表的操作。感兴趣的朋友请继续阅读 Android数据库高手秘籍(三)——使用LitePal升级表 。