**第六章 详解持久化技术**

**1. 持久化简介**

**数据持久化就是指将那些内存中的瞬时数据保存到存储设备中, 保证即使在手机或电脑关机的情况下, 这些数据仍然不会丢失。久化技术则提供了一种机制可以让数据在瞬时状态和持久状态之间进行转換。Android系统中主要提供了3种方式用于简単地实现数据持久化功能, 即文件存储、 sharedPreference存储以及数据库存储。**

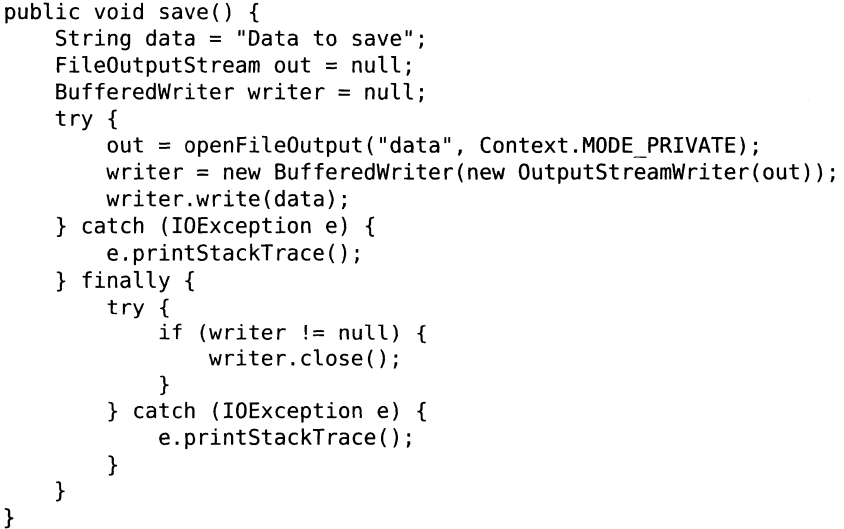
**2. 文件存储**

**（1）将数据存储到文件中**

**Context类中提供了一个 openFileOutput()方法,可以用于将数据存储到指定的文件中。这个方法接收两个参数,第一个参数是文件名，第二个参数是文件的操作模式, 主要有两种模式可选, MODE\_PRIVATE和 MODE\_APPEND。**

**其中 MODE\_PRIVATE是默认的操作模式,表示当指定同样文件名的时候,所写入的内容将会覆盖原文件中的内容,而 MODE\_APPEND则表示如果该文件已存在, 就往文件里面追加内容,不存在就创建新文件。**

**openFileOutput()方法返回的是一个 FileOutputStream对象,得到了这个对象之后就可以使用 Java流的方式将数据写入到文件中。将一段文本文件保存到文件中，代码如下所示：**

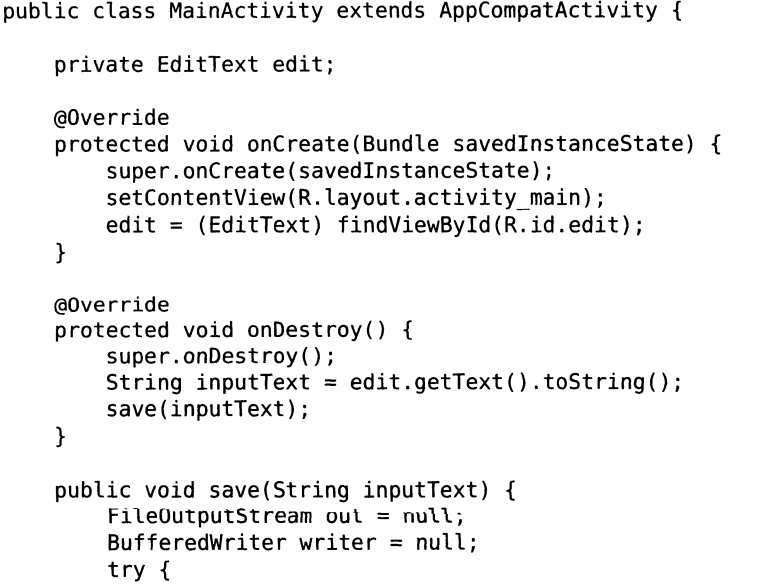


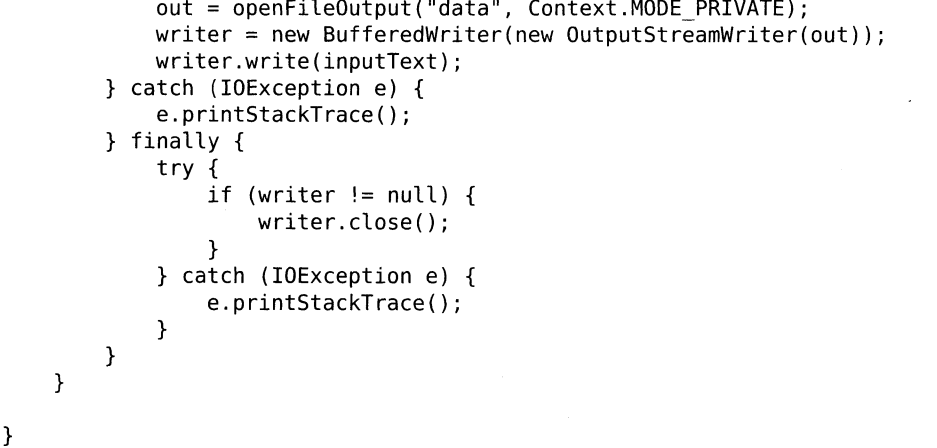
**这里通过 openFileOutput( )方法能够得到一个 FileOutputStream 对象, 然后再借助它构建出一个 OutputStreamWriter对象,接着再使用 OutputStreamWriter构建出一个 BufferedWriter对象,这样就可以通过 BufferedWriter来将文本内容写入到文件中了。**

**下面是一个完整的在 Android项目中使用文件存储例子,** **首先创建一个 FilePersistenceTest项目, 并修改 activity\_main.xm1中的代码, 如下所示:**



**这里在布局中加入了一个EditText。修改MainActivity中的代码，如下所示：**

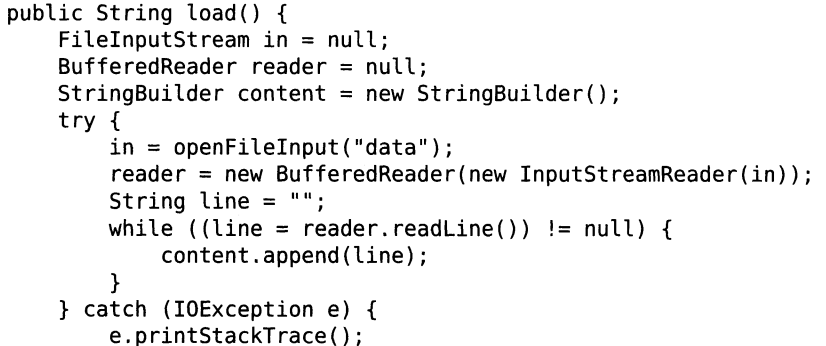


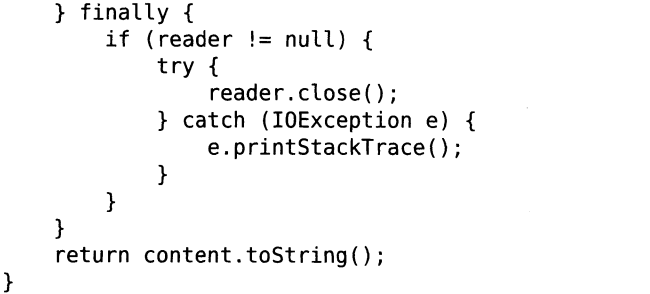


**这里首先在 onCreate()方法中获取了 EditText的实例,然后重写了 onDestroy()** **方法, 这样就可以保证在活动销毁之前一定会调用这个方法 。 在 onDestroy()方法中获取EditText中输入的内容,并调用 save()方法把输入的内容存储到文件中,文件命名为 data。**

**（2）从文件中读取数据**

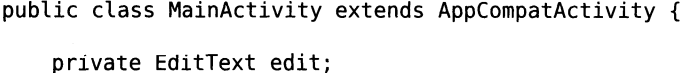
**context类中还提供了一个 openFileInput()方法,用于从文件中读取数据。它只接收一个参数，即要读取的文件名，并返回一个FileInputStream对象，在通过java流的方式就可以将数据读取出来了。从文件中读取文本数据，代码如下所示:**





**首先通过 openFileInput()方法获取到了一个 FileInputstream对象,然后借助它又构建出了一个 InputStreamReader对象, 接着再使用 InputstreamReader构建出一个 BufferedReader对象,这样就可以通过 BufferedReader进行一行行地读取,把文件中所有的文本内容全部读取出来,并存放在一个 stringBunder对象中,最后将读取到的内容返回。**

**修改（1）中的例子, 使得重新启动程序时 EditText中能够保留上次输入的内容。修改 MainActivity 中的代码,如下所示：**





**在 oncreate()方法中调用load()方法来读取文件中存储的文本内容,如果读到的内容不为 null,就调用 EditText的 setText()方法将内容填充到 EditText里,并调用 setselection()方法将输入光标移动到文本的末尾位置以便于继续输入,然后弹出一句还原成功的提示。**

**3. SharedPreferences存储**

**SharedPreferences使用键值得方式来存储数据，给保存的数据提供一个对应的键，在读取数据时通过相应的键值把数据读取出来。SharedPreferences还支持多种不同数据类型的存储。**

1. **将数据存储到SharedPreferences中**

**要想使用 SharedPreferences来存储数据,首先需要获取到 SharedPreferences对象。Android中主要提供了3种方法用于得到 sharedPreferences对象。**

1. **Context类中的 getSharedPreferences()方法**

**此方法接收两个参数,第一个参数用于指定 SharedPreferences文件的名称,** **第二个参数用于指定操作模式, 目前只有 MODE\_PRIVATE这一种模式可选。**

1. **Activity类中的 getPreferences()方法**

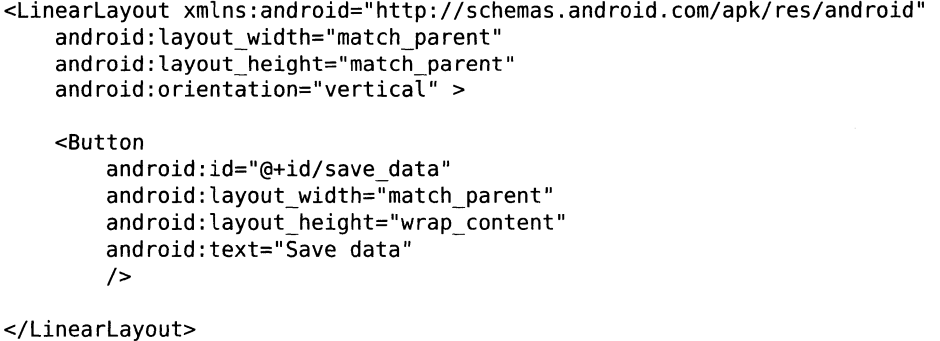
**只接收一个操作模式参数,使用这个方法时会自动将当前活动的类名作为 SharedPreferences的文件名。**

1. **PreferenceManager类中的 getDefaultSharedPreferences()方法**

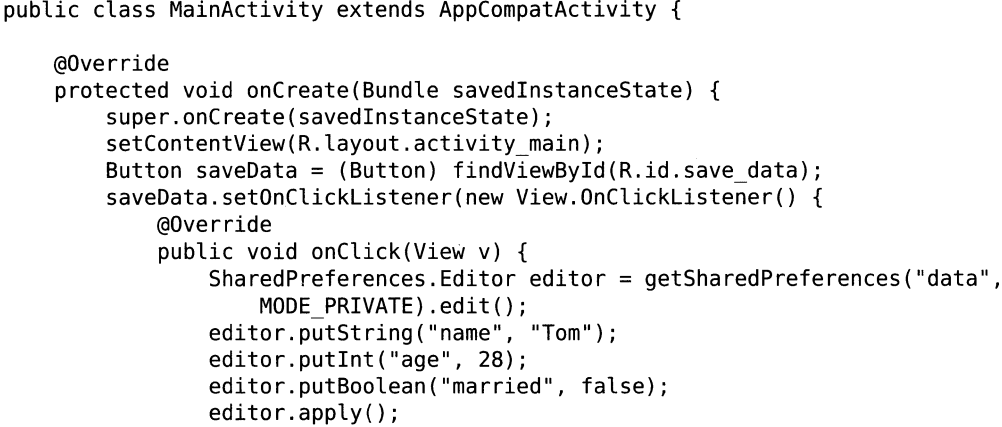
**这是一个静态方法,它接收一个 context参数,并自动使用当前应用程序的包名作为前缀来命名 SharedPreferences文件。向 SharedPreferences文件中存储数据, 主要可以分为3步实现:**

1. **调用 SharedPreferences对象的 edit()方法来获取一个 SharedPreferences.Editor对象。**
2. **向 SharedPreferences.Editor对象中添加数据,添加一个布尔型数据就使用putBoolean()方法, 添加一个字符串则使用 putString()方法, 以此类推。**
3. **调用 apply()方法将添加的数据提交, 从而完成数据存储操作。**

**新建一个 SharedPreferencesTest项目,然后修改 activity\_main.xm1中的代码, 如下所示:**



**这里放置了一个按钮用于将一些数据存储到Sharedpreferences文件当中。然后修改 MainActivity中的代码, 如下所示:**



**}**

**});**

**}**

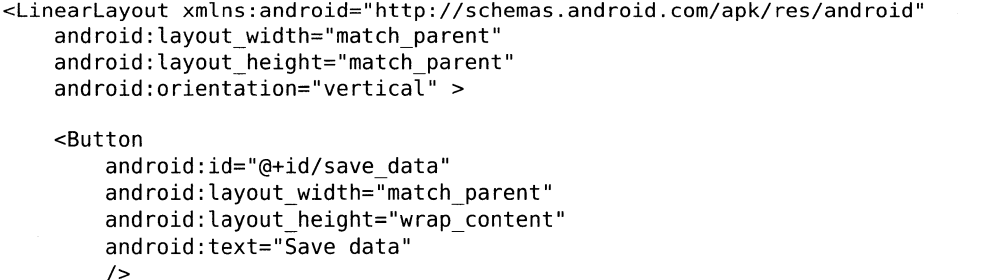
**}**

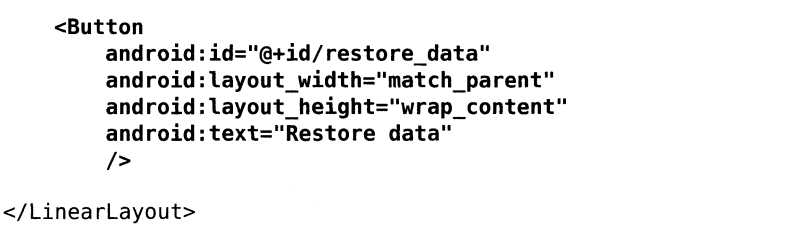
**这里首先给按钮注册了一个点击事件, 然后在点击事件中通过 getsharedPreferences()方法指定 SharedPreferences的文件名为 data,并得到了SharedPreferences.Editor对象。接着向这个对象中添加了3条不同类型的数据,最后调用 apply()方法进行提交,从而完成了数据存储的操作。**

1. **从SharedPreference中读取数据**

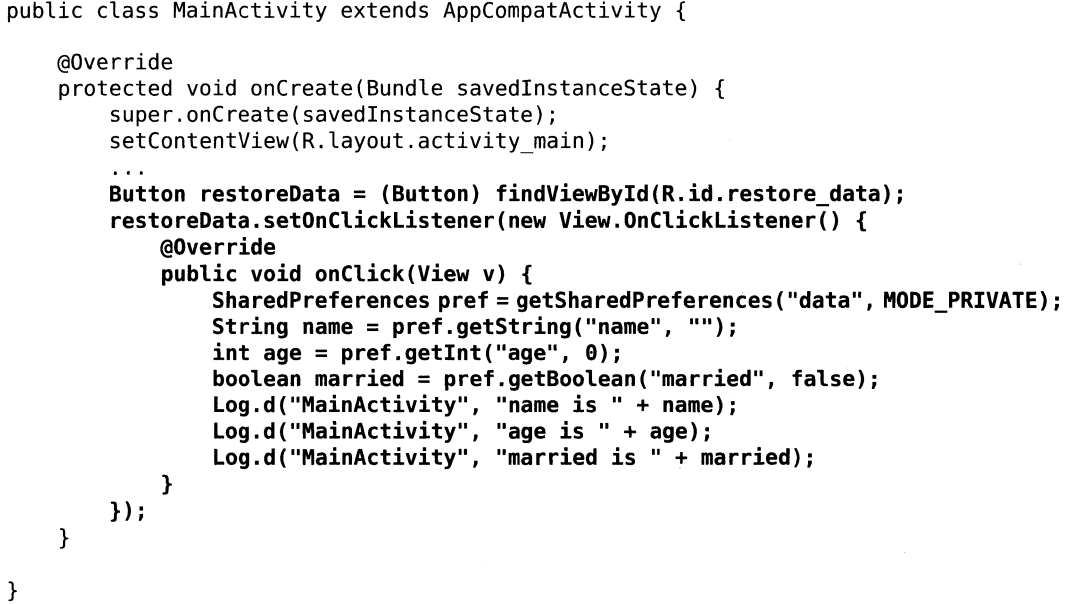
**SharedPreferences对象中提供了一系列的get方法,用于对存储的数据进行读取,每种 get方法都对应了 sharedPreferences.Editor中的一种 put方法。这些get方法都接收两个参数，第一个参数是键，第二个参数是默认值，即当传入的键找不到对应的值时会以什么样的默认値进行返回。**

**在 SharedPreferencesTest项目的基础上继续开发,修改 activity\_main.xm1中的代码,如下所示：**





**这里增加了一个还原数据的按钮,通过点击这个按钮来从 SharedPreferences文件读取数据。修改 MainActivity中的代码,如下所示:**



**首先通过getSharedPreferences()方法得到了 SharedPreferences对象,然后分别调用它的 getString()、 getInt()和 getBoolean()方法,去获取前面所存储的姓名、年龄和是否已婚,如果没有找到相应的值,就会使用方法中传入的默认值来代替,最后通过 Log将这些值打印出来。**

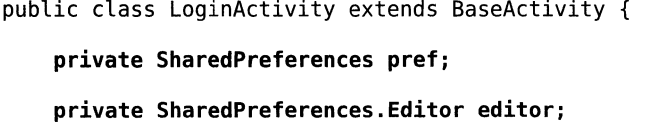
1. **实现记住密码功能**

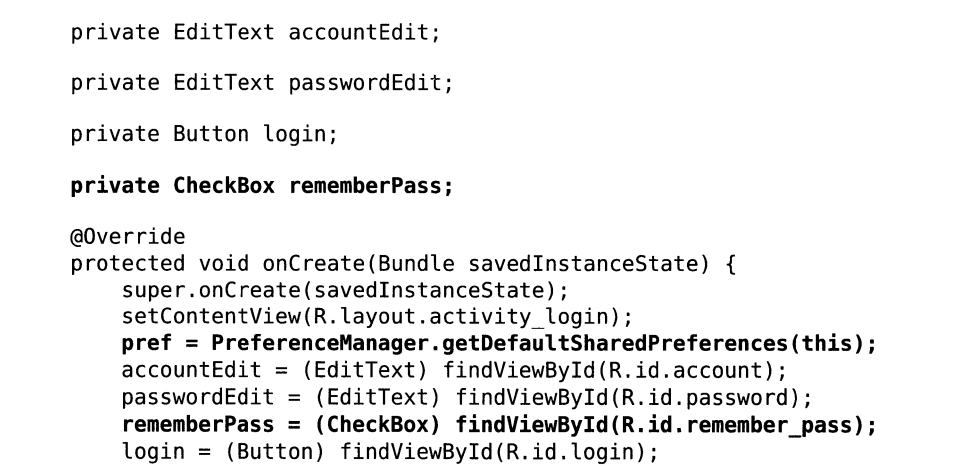
**打开 BroadcastBestPractice项目,编辑登录界面的布局。修改 activity\_1ogin.xm1中的代码,如下所示：**



**这里使用了一个新控件 checkBox。这是一个复选框控件,用户可以通过点击的方式来进行选中和取消, 这里使用这个控件来表示用户是否需要记住密码。**

**然后修改 LoginActivity中的代码, 如下所示:**







**}**

**}**

**这里首先在 onCreate()方法中获取到了sharedPreferences对象,然后调用它的 getBoolean()方法去获取 remember\_password这个键对应的值。由于不存在对应值，所以使用默认值false。接着在登录成功之后,会调用checkBox的isChecked()方法来检查复选框是否被选中,被选中则表示用户想要记住密码，这时将remember\_password设置为 true,然后把 account和 password对应的值都存入到 SharedPreferences文件当中并提交。如果没有被选中,就简単地调用一下 clear( )方法, 将SharedPreferences文件中的数据全部清除掉。**

**\*\*\*\*\*\*\*\*\*\***

**将密码以明文的形式存储在 SharedPreferences文件中是非常不安全的,很容易就会被别人盗取, 因此在正式的项目里还需要结合一定的加密算法来对密码进行保护。**

**\*\*\*\*\*\*\*\*\*\*\*\*\***

1. **SQLite数据库存**

**Android系统内置了SQLite数据库，它是一款轻量级的关系型数据库。SQLite不仅支持标准的 SQL语法,还遵循了数据库的 ACID事务。**

1. **创建数据库**

**Android专门提供了一个 SQLiteOpenHelper帮助类,其中有两个抽象方法，分别是onCreate()和onUpgrade()方法。需要在自己的带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。**

**SQLiteOpenHelper中还有两个非常重要的实例方法: getReadableDatabase()和 getWritableData-**

**base用来创建或打开一个现有的数据库。并返回一个可对数据库进行读写操作的对象。不同的是,** **当数据库不可写入的时候, getReadableDatabase()方法返回的对象将以只读的方式去打开数据库, 而 getWritableDatabase( )方法则将出现异常。**

**首先新建一个 DatabaseTest项目，这里要创建一个名为 BookStore.db的数据库,然后在这个数据库中新建一张 Book表,** **表中有 id(主键)、作者、价格、页数和书名等列。建表语句如下所示：**

**Create table Book(**

**Id integer primary key autoincrement,**

**author text,**

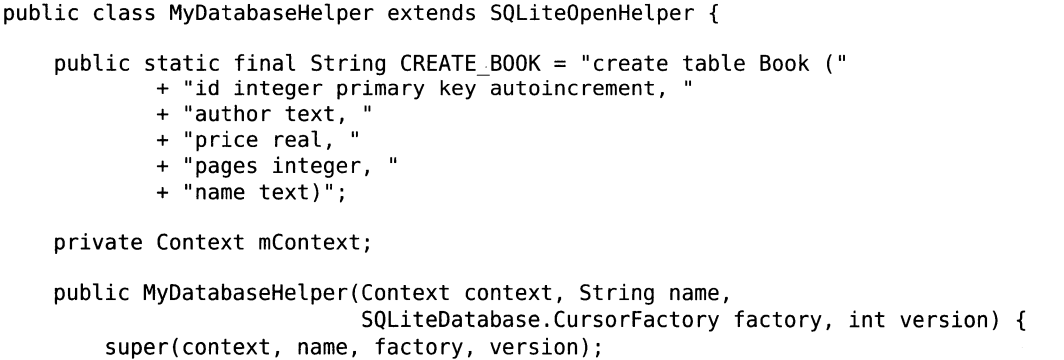
**price real,**

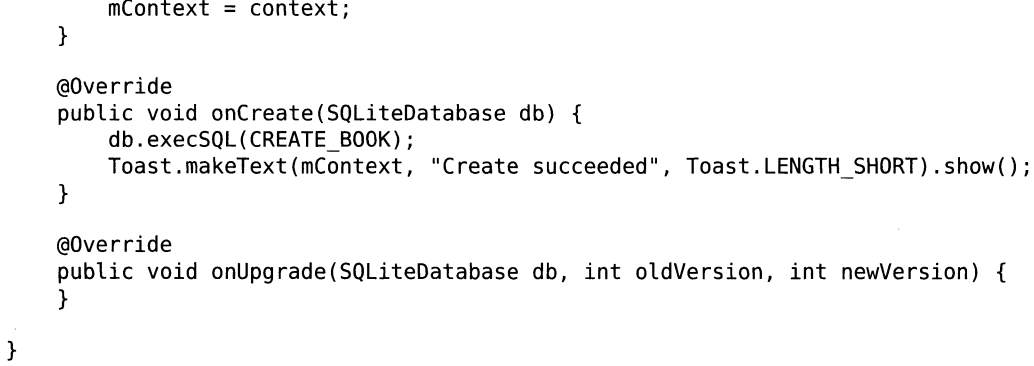
**pages integer,**

**name text)**

**SQLite的数据类型很简単, integer表示整型, real表示浮点型, text表示文本类型, blob表示二进制类型。另外,上述建表语句中还使用了primarykey将 id列设为主键,并用 autoincrement关键字表示 id列是自增长的。**

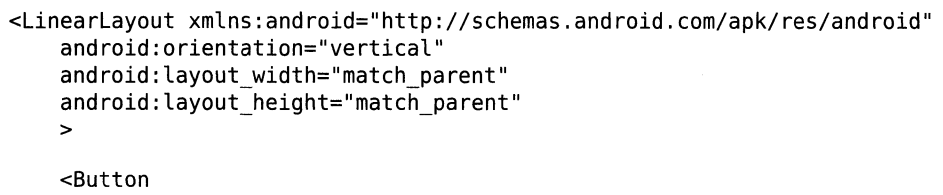
**新建 MyDatabaseHelper类继承自 SQLiteOpenHelper,代码如下所示:**

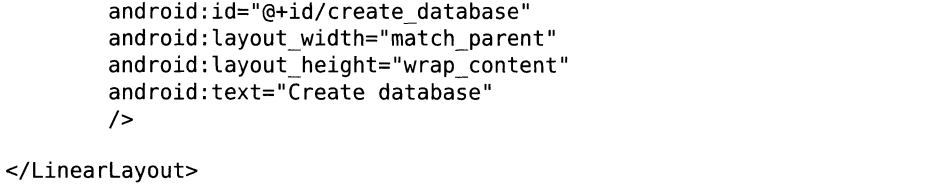




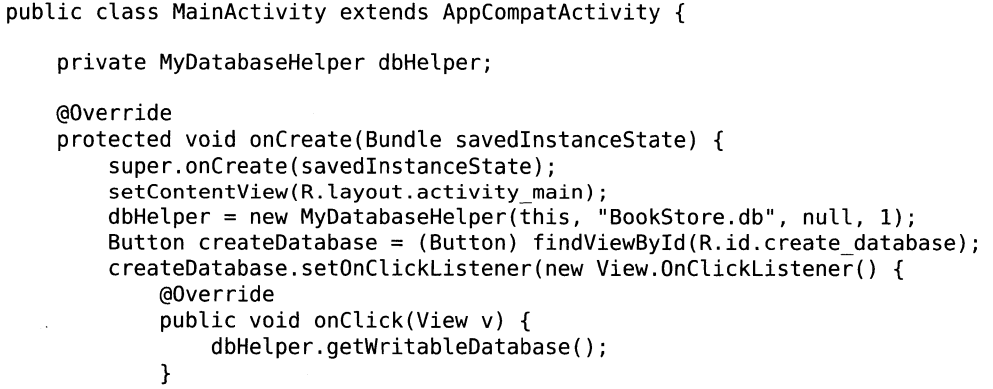
**这里把建表语句定义成了一个字符串常量, 然后在 onCreate()方法中又调了SQLiteDatabase的 execSQL()方法去执行这条建表语句, 并弹出一-个 Toast提示创建成功,这样就可以保证在数据库创建完成的同时还能成功创建 Book表 。**

**修改 activity\_main.xm1中的代码,如下所示:**





**这里就加入了一个按钮用于创建数据库，最后修改MainActivity中的代码如下所示:**

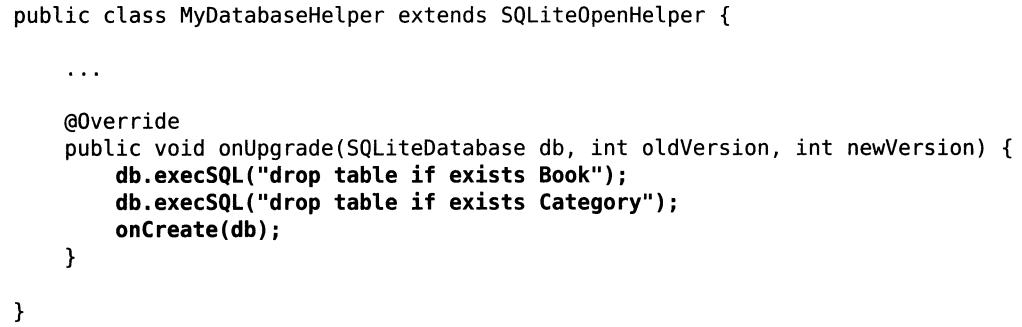




**这里在 onCreate()方法中构建了一个 MyDatabaseHelper对象,并且通过构造函数的参数将数据库名指定为 BookStore.db,版本号指定为1,然后在Createdatabase按钮的点击事件里调用了 getWritableDatabase()方法。**

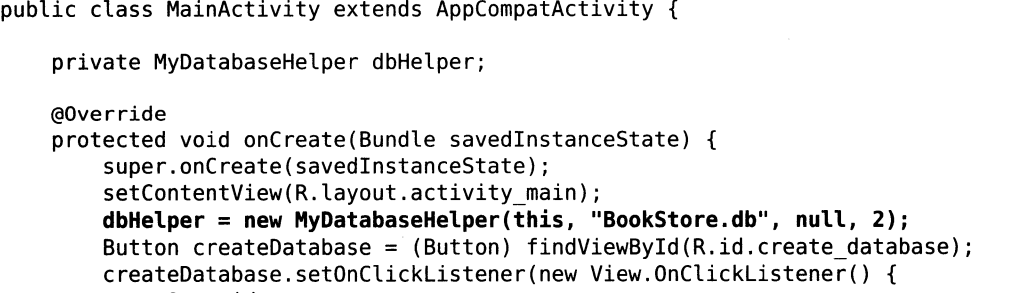
1. **升级数据库**

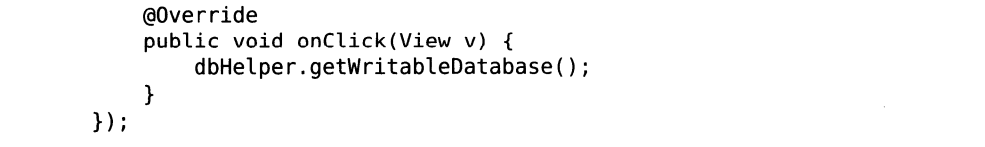
**修改 MyDatabaseHelper中的代码, 如下所示:**



**这里在 onUpgrade()方法中执行了两条 DROP语句,如果发现数据库中已经存在Book表或 category 表了,就将这两张表删除掉,然后再调用 onCreate()方法重新创建。**

**再传入一个版本号比一大的数，就可以让 onUpgrade()方法得到执行了。修改 MainActivity 中的代码,如下所示：**





**}**

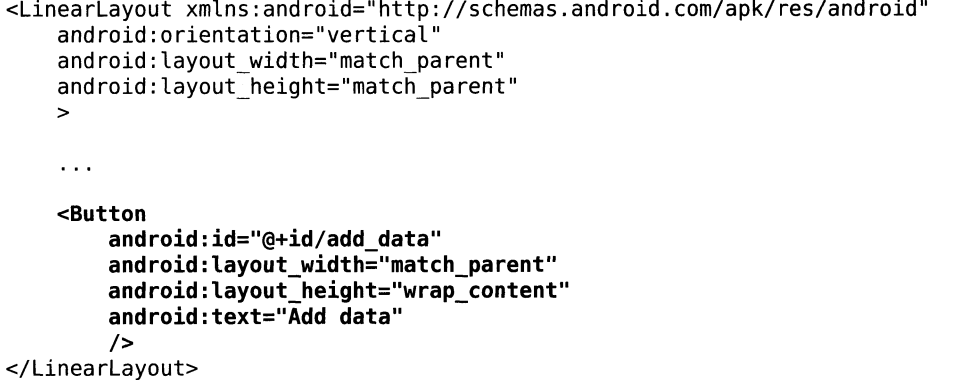
**}**

**这里将数据库版本号指定为2,表示我们对数据库进行升级了。**

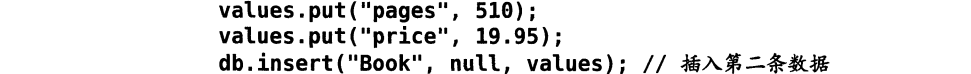
1. **添加数据**

**SQLiteDatabase中提供了一个 insert()方法,这个方法就是专门用于添加数据的。它接收3个参数,第一个参数是表名,** **第二个参数用于在未指定添加数据的情况下给某些可为空的列自动赋值 NULL, 一般用不到这个功能,直接传入NULL即可。第三个参数是一个ContentValues对象,它提供了一系列的 put()方法重载,用于向 ContentValues中添加数据, 只需要将表中的每个列名以及相应的待添加数据传入即可。**

**修改改 activity\_main.xm1中的代码, 如下所示:**



**接着修改 MainActivity 中的代码, 如下所示:**



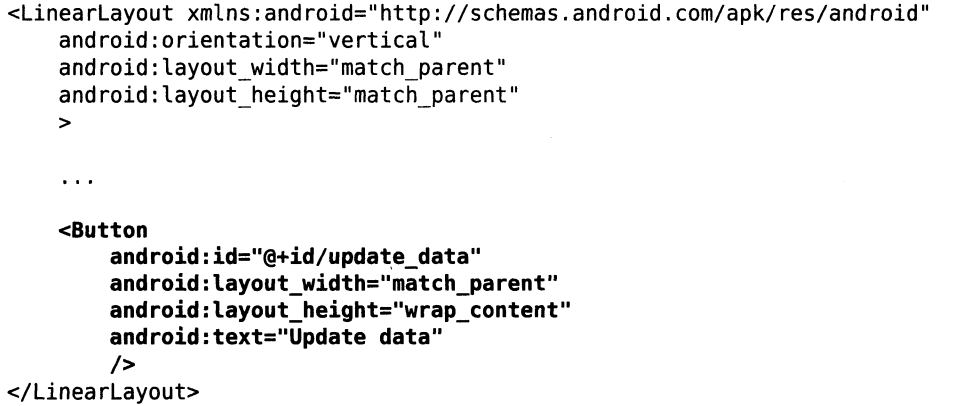


**在添加数据按钮的点击事件里面,先获取到了 sQLiteDatabase对象,然后使用Contentvalues来对要添加的数据进行组装。接下来调用了 insert()方法将数据添加到表当中。**

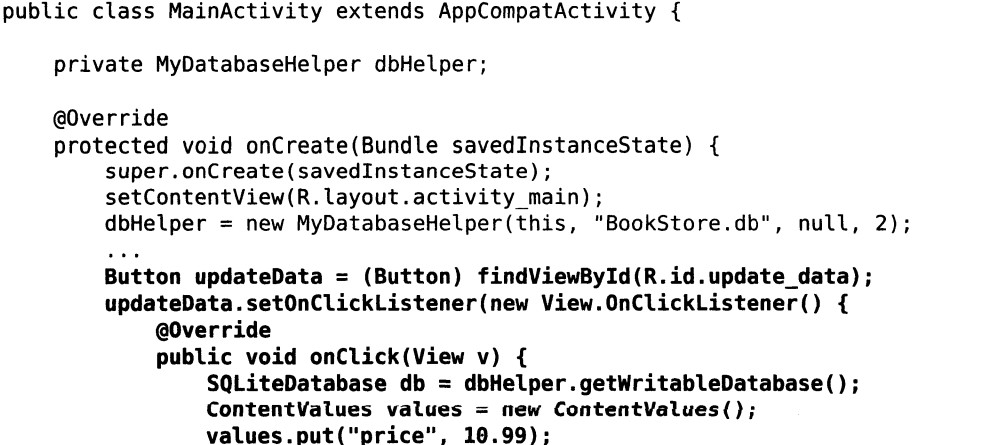
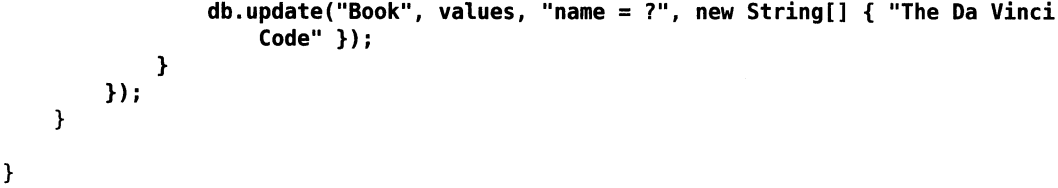
1. **更新数据**

**sQLiteDatabase中也提供了一个非常好用的 update()方法,用于对数据进行更新,这个方法接收4个参数，第一个参数是表名，第二个参数是 ContentValues对象,要把更新数据在这里组装进去。第三、第四个参数用于约束更新某一行或某几行中的数据, 不指定的话默认就是更新所有行。**

**首先修改 activity\_main.xml中的代码,如下所示：**



**然后修改MainActivity 中的代码,如下所示:**



**书的价格被改为10.99。**

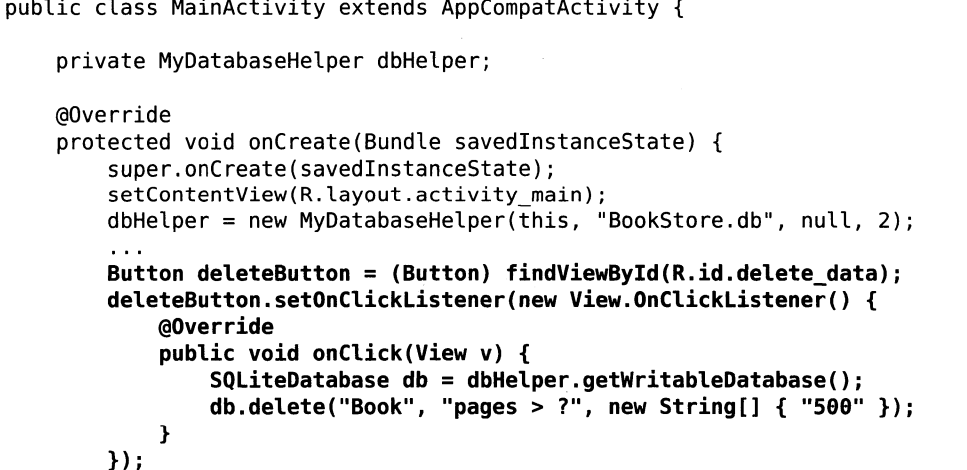
1. **删除数据**

**SQLiteDatabase中提供了一个 deLete()方法,专门用于删除数据,这个方法接收3个参数,** **第一个参数仍然是表名，第二、第三个参数又是用于约束删除某一行或某几行的数据,不指定的话默认就是删除所有行。**

**修改 activity\_main.xml中的代码,如下所示：**



**修改MainActivity 中的代码,如下所示:**



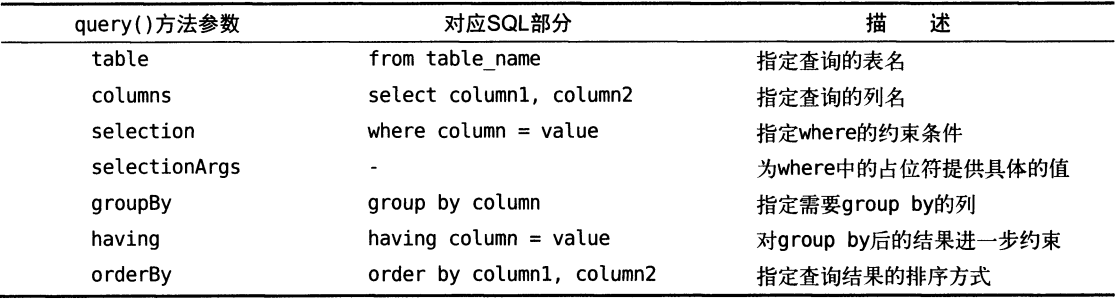
**}**

**}**

**这里在删除按钮的点击事件里指明去删除 Book表中的数据, 并且通过第二、 第三个参数来指定仅删除那些页数超过500页的书。**

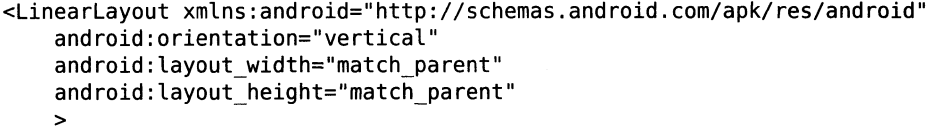
1. **查询数据**

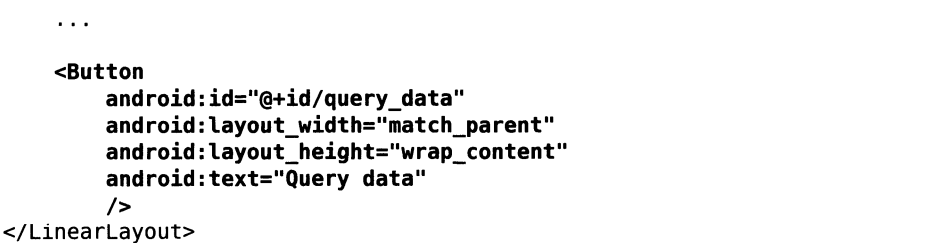
**SQLiteDatabase中提供了一个 query( )方法用于对数据进行查询：**



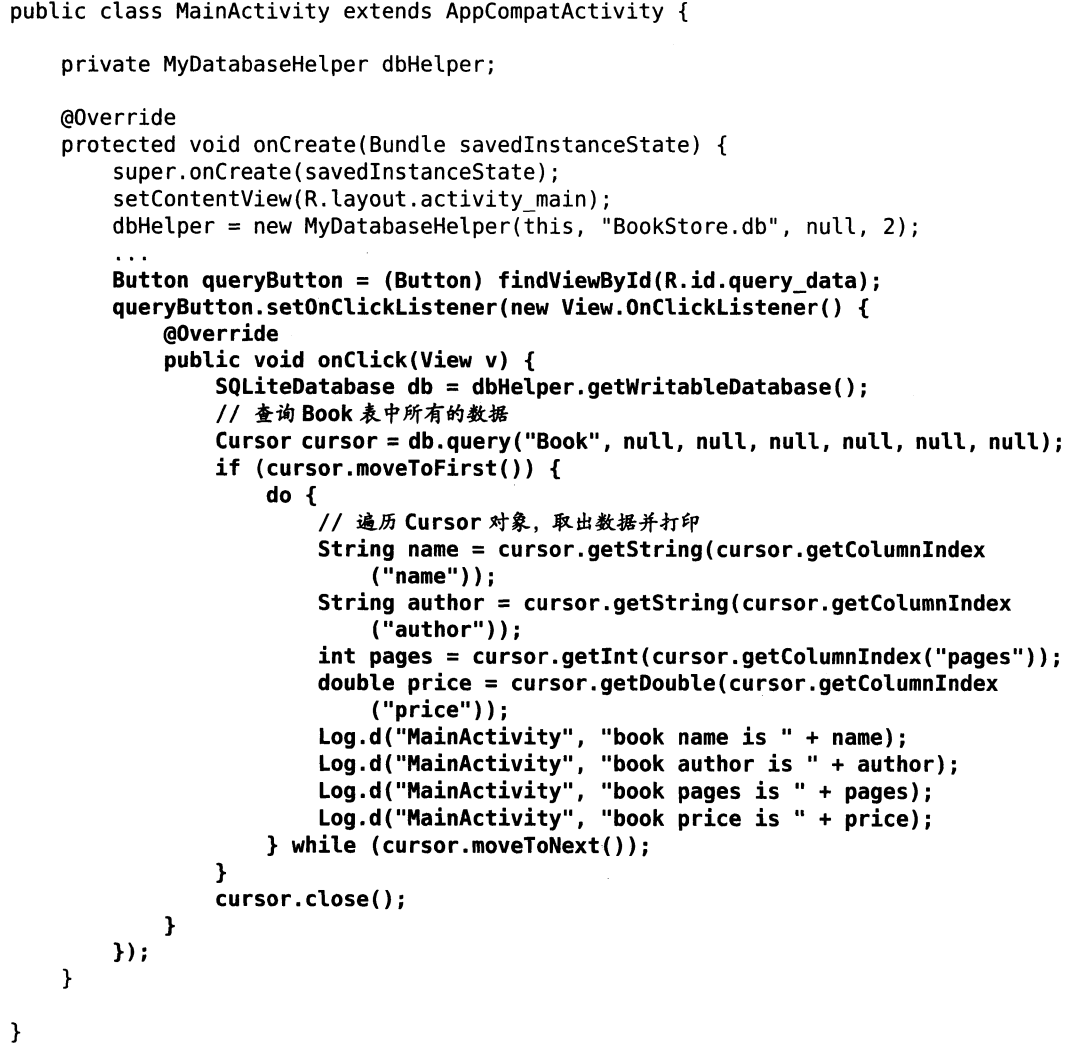
**调用 query()方法后会返回一个 Cursor对象, 査询到的所有数据都将从这个对象中取出 。**

**修改 activity\_main.xml中的代码,如下所示：**





**修改MainActivity 中的代码,如下所示:**



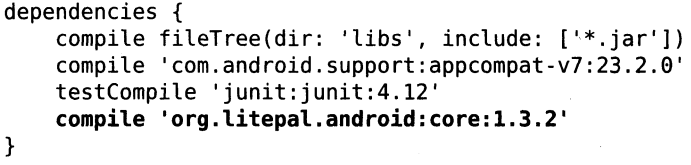
**首先在査询按钮的点击事件里面调用了 SQLiteDatabase的 query()方法去查询数据。 这里的 query( )方法表示希望査询这张表中的所有数据。査询完之后就得到了一个 Cursor对象,接着调用它的 moveToFirst()方法将数据的指针移动到第一行的位置, 然后进入了一个循环当中, 去遍历査询到的每一行数据。接着使用 Log的方式取出的数据打印出来,** **最后调用 close( )方法来关闭 cursor。**

1. **使用 LitePal操作数据**
2. **LitePal简介**

**LitePa1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,并将平时开发最常用到的一些数据库功能进行了封装。**

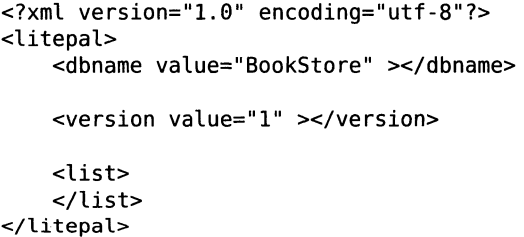
1. **配置LitePal**

**用 LitePa1的第一步,就是编辑 app/build.gradle文件,在 dependencies闭包中添加如下内容：**



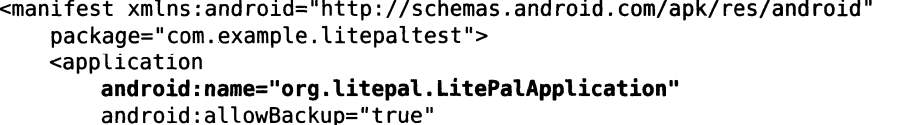
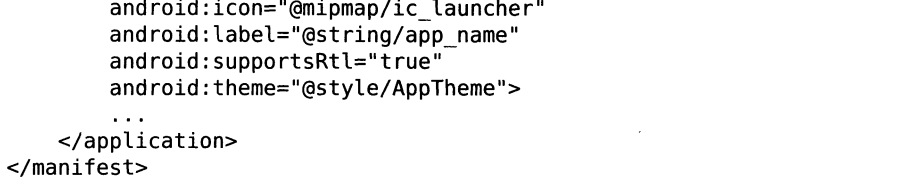
**添加的这一行声明中,前面部分是固定的, 最后的1.3.2是版本号的意思。**

**接下来需要配置1itepa1.xm1文件。右击app/src/main目录→New→Directory,创建一个 assets目录,然后在assets目录下再新建一个1itepa1.xml文件, 接着编辑ntepa1.xm1文件中的内容, 如下所示:**



**其中, <dbname>标签用于指定数据库名, <version>标签用于指定数据库版本号, <list>标签用于指定所有的映射模型。**

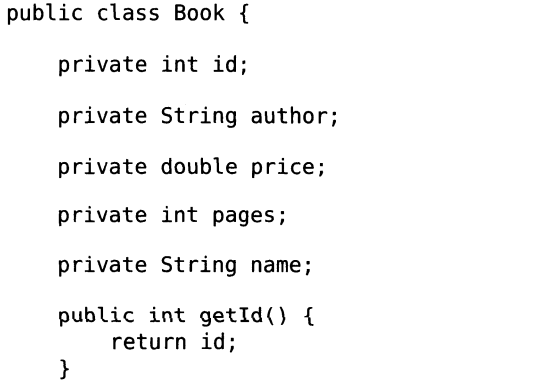
**最后还 需要再配置一下 LitePalApplication,修改AndroidManifest.xml 中的代码,如下所示:**

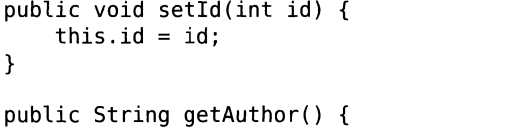


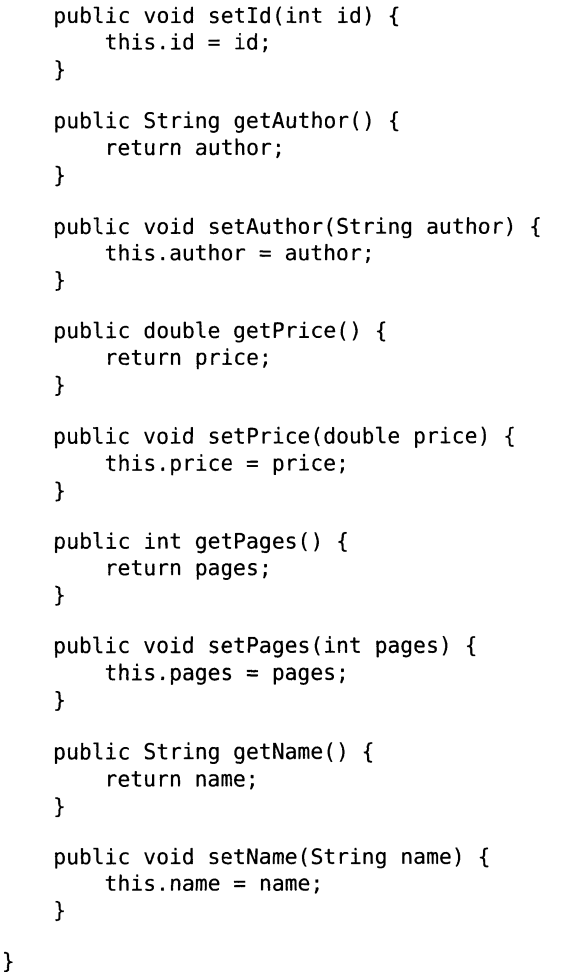
**这里将项目的 application配置为 org.1itepal.LitePalApplication, 这样才能让LitePa1的所有功能都可以正常工作。**

1. **创建和升级数据库**

**使用 LitePa1可以用面向对象的思维来创建一张Book表。定义一个 Book类,代码如下所示:**



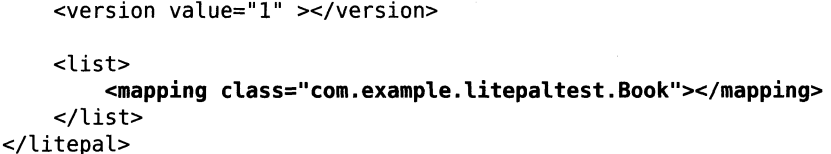




**这里在 Book类中定义了 id、 author、 price、 pages、 name这几个字段,并生成了相应的 getter和 setter方法。Book类对应数据库中的Book表，且类中的每一个字段分别对应了表中的每一个列, 这就是对象关系映射。**

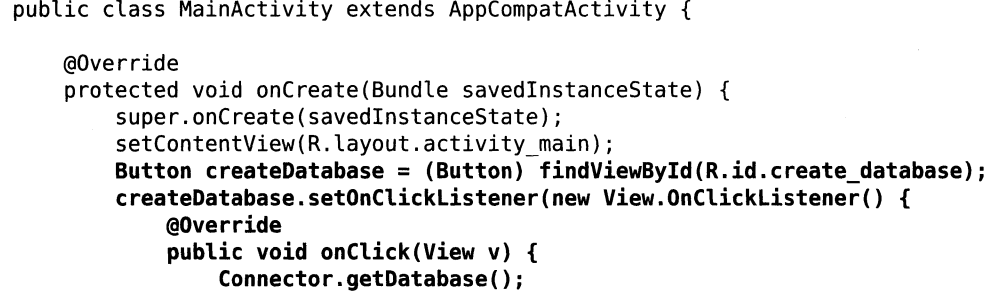
**将Book类添加到映射模型列表当中，修改Litepal.xml中的代码如下所示：**





**这里使用<mapping>标签来声明要配置的映射模型类, 一定要使用完整的类名 。 不管有多少模型类需要映射, 都使用同样的方式配置在<list>标签下即可。**

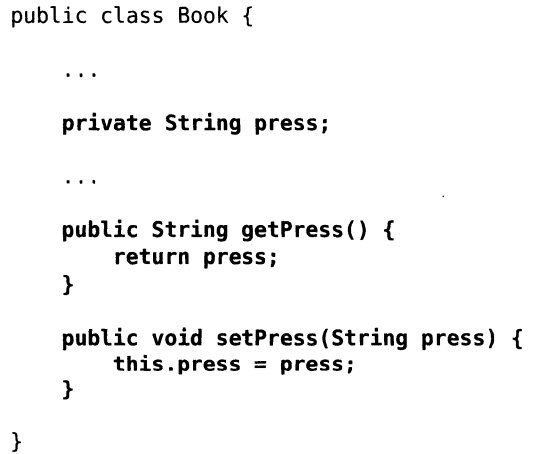
**修改MainActivity中的代码.如下所示:**



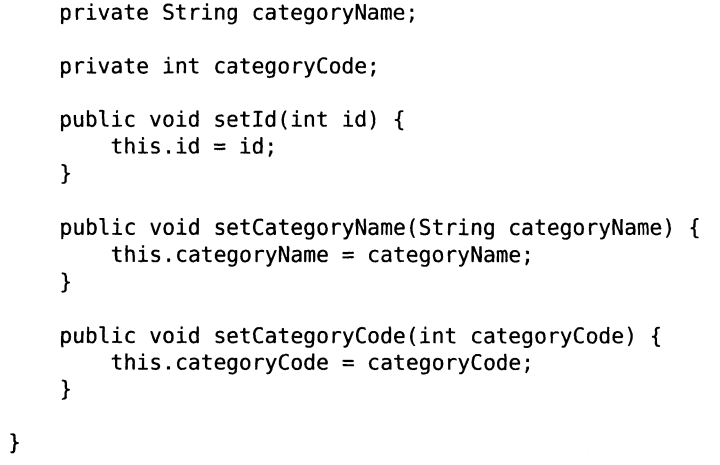


**其中, 调用 Connector.getDatabase()方法就是一次最简単的数据库操作,只要点击一下按钮,数据库就会自动创建完成了。**

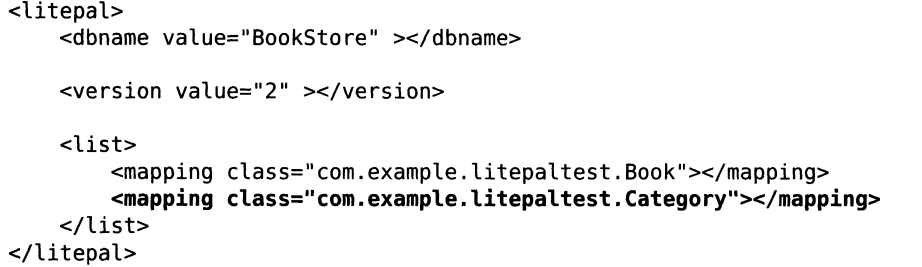
**要向 Book表中添加一个 press(出版社)列,直接修改 Book类中的代码,添加一个 press字段即可,如下所示:**



**想再添加一张 Category表, 那么只需要新建一个 category类就可以了,代码如下所示：**



**改完了想改的东西, 只需要记得将版本号加1即可。由于这里还添加了一个新的模型类, 因此也需要将它添加到映射模型列表中。修改1itepa1.xm1中的代码,如下所示:**



1. **小结**

**本章主要是对 Android常用的数据持久化方式进行了详细的讲解, 包括文件存储、SharedPreferences存储以及数据库存储。**

**\*文件存储适用于存储一些简单的文本数据或者二进制数据。**

**\*SharedPreferences适用于存储一些键值对。**

**\*数据库则适用于存储那些复杂的关系型数据。**