

大话企业级 Android 开发 · 第十三部分

本教程说明及版权声明

- 《大话企业级 Android 开发》是国土工作室为了方便中国 Android 开发者，推动 Android 企业级应用开发，特投入大量心血撰写的书籍，并在网络上免费发布，希望为移动互联网和智能手机时代贡献绵薄之力！所有相关文档版权均属国土工作室所有。
- 本教程是由国土工作室参考官方文档，综合市面相关书籍，经过充分的吸收消化，结合开发实践的一部原创作品，为了本教程及早与广大读者同仁见面、分享，特采用定稿一部分就发布一部分的连载方式发布。读者可以在本博客获取最新内容。
- 未经国土工作室授权，禁止将此文档及其衍生作品以标准（纸质）书籍形式发行。
- 本文档受有关法律的版权保护，对本文档内容的任何未经同意的复制和抄袭行为，将导致相应的法律责任。未经国土工作室同意，任何团体及个人不能用此教程牟利，违者必究。但是：在不收取其他人费用的前提下，您可以自由传播此文档，但必须保证版权信息、文档及其自带标示的完整性。
- 如果对该文档有任何疑问或者建议，请进入官方博客
<http://www.cnblogs.com/guoshiandroid/>留言或者直接与国土工作室联系（后附联系方式），我们会慎重参考您的建议并根据需要对本文档进行修改，以造福更多开发者！
- 《大话企业级 Android 开发》的最新及完整内容会在国土工作室官方博客定期更新，请访问国土工作室博客
<http://www.cnblogs.com/guoshiandroid/>获取更多更新内容。

关于国土工作室

我们(国土工作室)是一支专注于 Android 平台企业级应用开发的技术团队，对娱乐多媒体应用有着深刻的理解及研发能力，致力服务于企业用户。为音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络应用向移动互联网发展提供解决方案和技术支持，为企业提供 Android 培训服务等多种业务。

我们尤其擅长于提供从 Android 客户端到服务端的一站式解决方案和技术支持，服务端可以采用 Java EE，也可以采用轻量级流行的 LAMP 技术体系。目前，研发出了比 KU6、优酷更加强大和完善的 Android 视频网站娱乐多媒体客户端软件，并在持续升级中。

目前，我们正在务实而卓有成效的与音视频等娱乐多媒体网站、门户网站、SNS、论坛、电子商务等传统网络服务商合作，发展迅速，渴望有志之士的加入，和我们一起为成为世界最好的 Android 软件开发和咨询、培训公司而奋斗，为移动互联网和智能手机时代贡献力量！

联系我们

电话:15711060468

Email:guoshiandroid@gmail.com

博客: <http://www.cnblogs.com/guoshiandroid/>

小安：我们公司正在开发一款手机端的客户管理软件，在电脑上可以用 SQL-Server 或者 Oracle 来管理数据，但是在 android 手机上用什么呢？他赶快去请教大致博士。

大致：在 Android 系统中可以用 SQLite 数据库来存储应用的数据，在 Android 系统中的很多应用，比如：联系人、图库、音乐等都用 SQLite 数据库来存储数据，以后我们在开发应用的时候也经常會用到，这个知识点你必须掌握。之前我们已经讲过如何使用文件和 SharedPreferences 来存储数据，它们都比较适合存储数据量比较小而且被访问频率不是很高的数据，如果你要存储数据比较多，并且以后很可能还会去检索那些数据的话，那就要选择使用 SQLite 数据库存储数据。

1.1 SQLite 数据库简介

小安：您能详细介绍一下这个 SQLite 数据库吗？

大致：哈哈，不错，求知欲很强啊。

SQLite 是一个开源的嵌入式关系数据库，它在 2000 年由 D. Richard Hipp 发布，它可以减少应用程序管理数据的开销，SQLite 可移植性好、很容易使用、很小、高效而且可靠。

目前在 Android 系统中集成的是 SQLite3 版本，SQLite 不支持静态数据类型，而是使用列关系。这意味着它的数据类型不具有表列属性，而具有数据本身的属性。当某个值插入数据库时，SQLite 将检查它的类型。如果该类型与关联的列不匹配，则 SQLite 会尝试将该值转换成列类型。如果不能转换，则该值将作为其本身具有的类型存储。

SQLite 支持 NULL、INTEGER、REAL、TEXT 和 BLOB 数据类型。

例如：可以在 Integer 字段中存放字符串，或者在布尔型字段中存放浮点数，或者在字符型字段中存放日期型值。但是有一种例外，如果你的主键是 INTEGER，那么只能存储 64 位整数，当向这种字段中保存除整数以外的数据时，将会产生错误。另外，SQLite 在解析 CREATE TABLE 语句时，会忽略 CREATE TABLE 语句中跟在字段名后面的数据类型信息。

小安：会忽略字段名后面的数据类型信息？是不是像：

```
CREATE TABLE person (personid integer primary key autoincrement, name
varchar(20))
```

这个 SQLite 数据库在解析这个语句的时候，它会忽略掉跟在 name 字段后面的 varchar (20)，也就是说你可以往 name 字段填 SQLite 支持的那五种数据类型的任何一种，而且 name 字段可以存任意长度的字符，长度 20 不起作用。也就是说可以把 SQLite 数据库近似看作是一种无数据类型的数据库，你可以把任何类型的资料存放在非 Integer 类型的主键之外的其它字段上去，另外字段的长度也是没有限度的。

大致：对，你分析的很正确，还有一点你要注意，虽然 SQLite 数据库忽略字段后面的限制信息，但是建议你一定要在编写 SQL 语句的时候按照标准的 SQL 语法，因为这样在别人看你的代码的时候，可以明白你期望的 name 字段是长度为 20 的 varchar 类型，便于更好的去理解你的代码。

1.1.1 SQLite 的特点

小安：博士，SQLite 数据库有什么特点吗？

大致：SQLite 数据库总结起来有五大特点。

1. 零配置

SQLite3 不用安装、不用配置、不用启动、关闭或者配置数据库实例。当系统崩溃后不用做任何恢复操作，在下次使用数据库的时候自动恢复。

2. 可移植

它是运行在 Windows、Linux、BSD、Mac OS X 和一些商用 Unix 系统，比如 Sun 的 Solaris、IBM 的 AIX，同样，它也可以工作在许多嵌入式操作系统下，比如 Android、QNX、VxWorks、Palm OS、Symbian 和 Windows CE。

3. 紧凑

SQLite 是被设计成轻量级、自包含的。一个头文件、一个 lib 库，你就可以使用关系数据库了，不用任何启动任何系统进程。

4. 简单

SQLite 有着简单易用的 API 接口。

5. 可靠

SQLite 的源码达到 100%分支测试覆盖率。

1.1.2 SQLite 可以解析的 SQL 语句

小安：SQLite 数据库对于解析 SQL 有什么要求吗？像 MySQL 和 Oracle 有好多 SQL 语句都不一样。

大致：SQLite 可以解析大部分的标准 SQL 语句。

查询语句：select * from 表名 where 条件子句 group by 分组子句 having ... order by 排序子句。

分页语句：select * from Account limit 5 offset 3 或者 select * from Account limit 3,5。

插入语句：insert into 表名(字段列表) values(值列表)。

更新语句：update 表名 set 字段名=值 where 条件子句。

删除语句：delete from 表名 where 条件子句。

1.2 使用 SQLiteOpenHelper 抽象类建立数据库

小安：但是我们如何去执行这些 SQL 语句呢？

大致：在 Android 操作数据库的时候，会用到一个 SQLiteDatabase 类，是使用这个类的对象发送 SQL 语句去操作 SQLite 数据库，在学习用这个类操作数据库之前我们必须要先解决一些问题。首先，我们开发的是手机应用，如果你的应用使用到了 SQLite 数据库，用户第一次安装你的应用，你需要先做些什么呢？其次，如果用户已经安装过软件，当软件升级的时候，你又要做些什么？

小安：这个还真不知道。

库，如果是软件升级的话，那就要更改数据库的版本了。

小安：这些如何实现呢？

大致：Android 系统专门为我们提供了一个 SQLiteOpenHelper 抽象类，对数据库进行版本管理，去实现我们刚才要做的。这个类可以不使用，不过那样就失去了对数据库进行版本管理的功能。所以在开发项目的时候，建议最好使用这个类。

下面来看一下这个抽象类具体是如何解决我们的问题的。

为了实现对数据库版本进行管理，SQLiteOpenHelper 类提供了两个重要的方法，分别是 onCreate(SQLiteDatabase db) 和 onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)，前者用于初次使用软件时生成数据库表，后者用于升级软件时更新数据库表结构。

现在我们使用 SQLiteOpenHelper 这个类来完成生成数据库的操作，如果我们要开发使用数据库的软件，就必须先生成数据库。

创建项目：

下面我们新建一个名称为“db”的 Android 项目，如图 3-1

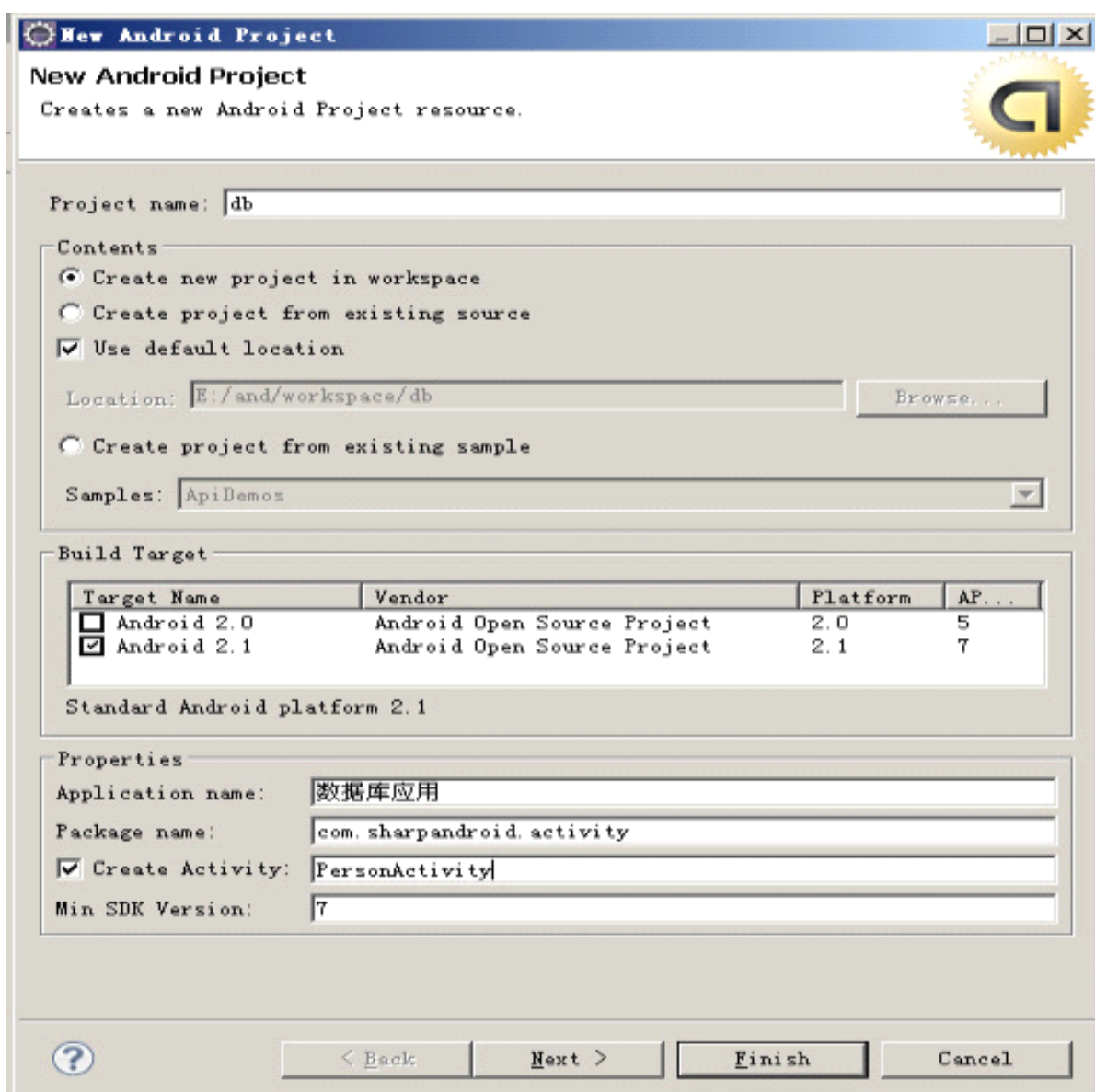


图 3-1

单击【Finish】完成。

com.sharpandroid.service 包下，如图 3-2，单击【Finish】完成。

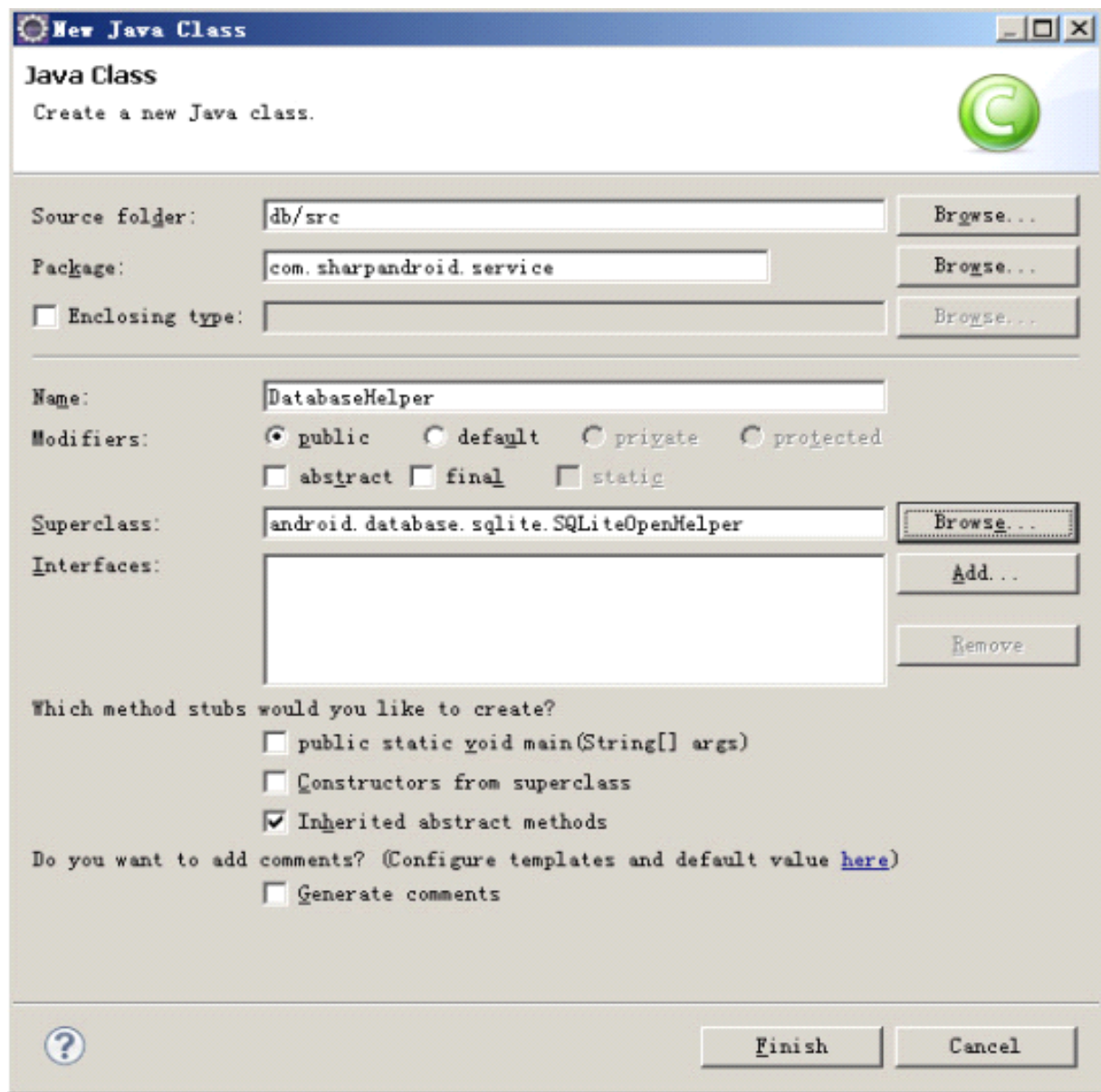


图 3-2

DatabaseHelper 类建成之后会有一个错误，如图 3-3

```
1 package com.sharpandroid.service;
2
3 import android.database.sqlite.SQLiteDatabase;
4
5
6 public class DatabaseHelper extends SQLiteOpenHelper {
7
8     @Override
9     public void onCreate(SQLiteDatabase db) {
10         // TODO Auto-generated method stub
11     }
12 }
13
```

图 3-3

这个错误的意思是说它的父类没有定义默认构造器，我们需要为它添加一个构造器，明确的

DatabaseHelper.java

```
public DatabaseHelper(Context context, String name, CursorFactory factory,  
    int version) {  
    super(context, name, factory, version);  
    // TODO Auto-generated constructor stub  
}
```

Context: 代表应用的上下文。

Name: 代表数据库的名称。

Factory:代表记录集游标工厂，是专门用来生成记录集游标，记录集游标是对查询结果进行迭代的，后面我们会继续介绍。

Version: 代表数据库的版本，如果以后升级软件的时候，需要更改Version版本号，那么onUpgrade(SQLiteDatabase db, **int** oldVersion, **int** newVersion)方法会被调用，在这个方法中比较适合实现软件更新时修改数据库表结构的工作。

在本应用中我们会把数据库名和版本号定义为一个常量，只用给构造器传一个Context就可以了，如以下代码：

```
private static final String NAME="sharp.db";//.db可有可无  
private static final int version=1;    //版本号不能为0  
  
public DatabaseHelper(Context context) {  
    super(context, NAME, null, version);  
}
```

在用户第一次使用软件数据库创建的时候会调用onCreate()方法，这个方法中特别适合生成数据库表的结构，它只会被调用一次，它的唯一一个参数是操作数据库的工具类，这个工具类提供了对数据的添、删、改、查等方法，用这个类实现对SQL语句的执行，如何执行SQL语句？如以下代码：

```
//在用户第一次使用软件时，会创建数据库，而该方法在数据库初次创建时被调用  
@Override  
public void onCreate(SQLiteDatabase db)  
{  
    db.execSQL("CREATE TABLE person (personid integer primary key autoincrement,  
name varchar(20), age integer)");  
}
```

在软件升级的时候会调用onUpgrade()方法，假设现在的数据库版本version为1，初次使用的时候执行onCreate()方法，生成刚才的数据库，假如一段时间之后，数据库版本version变为2或者其他的，同时person表比版本1多加了一个字段address，那就要调用onUpgrade()方法，去删除原有的表，然后调用onCreate()方法重建新的表，如以下代码：

```

public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL("DROP TABLE IF EXISTS person");
    onCreate(db);
}

```

上述做法在实际工作中是不能那么做的,有的软件在升级后,还要保存用户原来的数据,最好先对原有的数据进行备份,在新表建好之后把数据导入新表,这些在实际开发中会遇到。

1.2.1 测试建立数据库

完成上面的知识之后,我们就可以来测试一下数据库是否生成。

建一个单元测试,首先搭建测试环境,如图3-4

```

1<?xml version="1.0" encoding="utf-8"?>
2<manifest xmlns:android="http://schemas.android.com/apk/res/android"
3    package="com.sharpandroid.activity"
4    android:versionCode="1"
5    android:versionName="1.0">
6    <application android:icon="@drawable/icon" android:label="@string/app_name">
7        <uses-library android:name="android.test.runner" />
8        <activity android:name=".PersonActivity"
9            android:label="@string/app_name">
10            <intent-filter>
11                <action android:name="android.intent.action.MAIN" />
12                <category android:name="android.intent.category.LAUNCHER" />
13            </intent-filter>
14        </activity>
15    </application>
16    <uses-sdk android:minSdkVersion="7" />
17    <instrumentation android:name="android.test.InstrumentationTestRunner"
18        android:targetPackage="com.sharpandroid.activity" android:label="Tests for My App" />
19</manifest>

```

图3-4

在AndroidManifest.xml中加入下面代码:

```

<uses-library android:name="android.test.runner" />
<instrumentation android:name="android.test.InstrumentationTestRunner"
    android:targetPackage="com.sharpandroid.activity" android:label="Tests for
My App" />

```

然后建一个单元测试类DBTest继承AndroidTestCase类,如图3-5,单击【Finish】

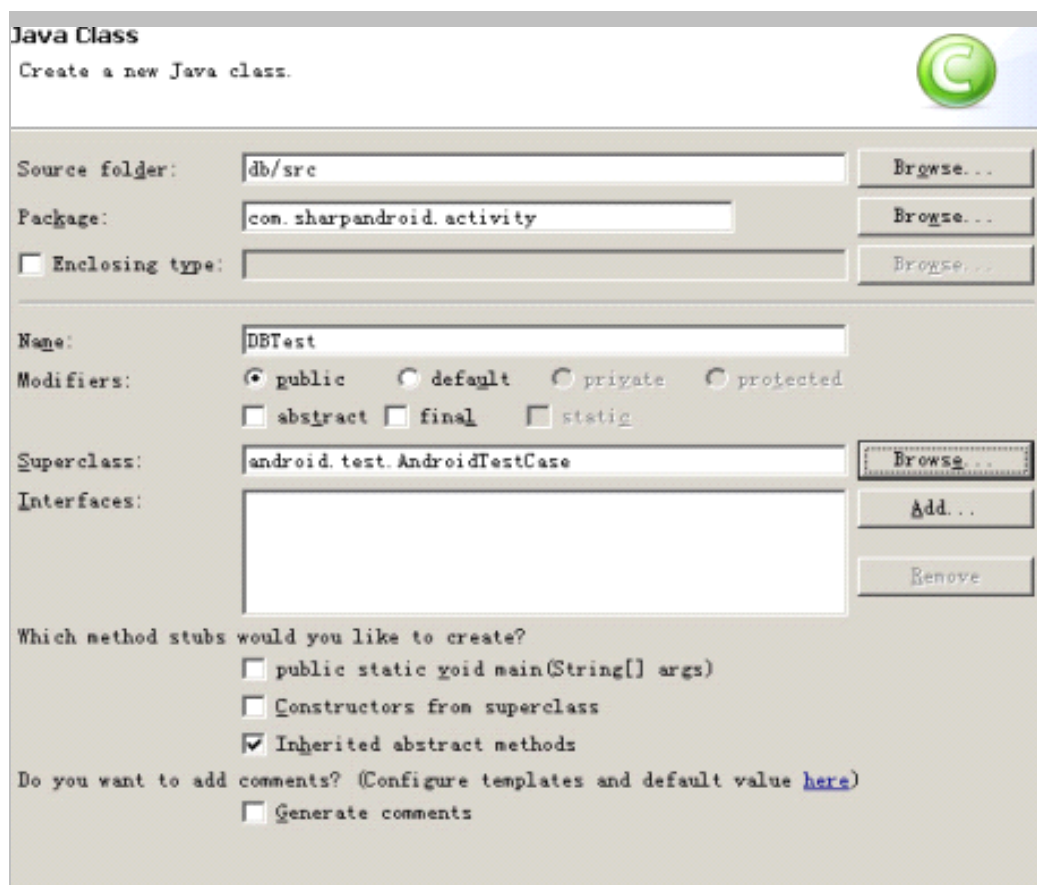


图3-5

在 DBTest 类中，当我们调用 DatabaseHelper 类的 `getWritableDatabase()` 或者 `getReadableDatabase()` 方法的时候，它都会导致数据库的生成，关于这两个方法的区别下面会给大家介绍，现在我们就先调用 `getWritableDatabase()` 方法，代码如下：

DBTest.java

```
public void testCreadDB() throws Throwable{  
    DatabaseHelper databaseHelper = new DatabaseHelper(this.getContext());  
    databaseHelper.getWritableDatabase();  
    //databaseHelper.getReadableDatabase();  
}
```

然后测试 `testCreadDB()` 方法，如图3-6


```

if (mDatabase != null && mDatabase.isOpen() && !mDatabase.isReadOnly()) {
    return mDatabase; // The database is already open for business
}

```

这段代码我们目前先用不关心，它是缓存SQLiteDatabase对象的，此次它并不执行。

```

if (mName == null) {
    db = SQLiteDatabase.create(null);
} else {
    db = mContext.openOrCreateDatabase(mName, 0, mFactory);
}

```

在数据库名称不为空的时候，调用mContext.openOrCreateDatabase()方法创建数据库，这个就是创建数据库底层的方法。但是，在使用这个方法的时候，它不一定创建数据库，如果数据库存在了，就是打开数据库；不存在，才创建数据库。

第一个参数mName：数据库名称。

第二个参数0：是表示操作模式为私有，其他应用不可以访问该数据库。

第三个参数mFactory：游标工厂，使用默认的就可以。

```

int version = db.getVersion();
if (version != mNewVersion) {
    db.beginTransaction();
    try {
        if (version == 0) {
            onCreate(db);
        } else {
            onUpgrade(db, version, mNewVersion);
        }
        db.setVersion(mNewVersion); //设置为新的版本号
        db.setTransactionSuccessful();
    } finally {
        db.endTransaction();
    }
}

```

版本号不为负上述代码的意思是：在第一次创建数据库的时候版本号db.getVersion()是0，它调用onCreate()创建表结构，当软件更新升级的时候，版本号改变version不为0，则调用onUpgrade()方法更新表结构，之后便把数据库版本号设置为新的版本号。

接下来，来看一下getReadableDatabase()的源代码比较一下和getWritableDatabase()有什么不同。

```

try {
    return getWritableDatabase();
} catch (SQLException e) {

```

```

        Log.e(TAG, "Couldn't open " + mName + " for writing (will try
read-only):", e);
    }

    SQLiteDatabase db = null;
    try {
        mIsInitializing = true;
        String path = mContext.getDatabasePath(mName).getPath();
        db = SQLiteDatabase.openDatabase(path, mFactory,
SQLiteDatabase.OPEN_READONLY);
        if (db.getVersion() != mNewVersion) {
            throw new SQLiteException("Can't upgrade read-only database from
version " +
                db.getVersion() + " to " + mNewVersion + ": " + path);
        }

        onOpen(db);
        Log.w(TAG, "Opened " + mName + " in read-only mode");
        mDatabase = db;
        return mDatabase;
    } finally {
        mIsInitializing = false;
        if (db != null && db != mDatabase) db.close();
    }
}

```

你会发现，它的内部调用getWritableDatabase()方法，在有异常被捕获的时候调用SQLiteDatabase.openDatabase()仅仅只是以只读方式打开数据库，而不像getWritableDatabase()是打开或者创建数据库。

总结getWritableDatabase()和getReadableDatabase()方法的异同，就是它们都可以获取一个用于操作数据库的SQLiteDatabase实例。但getWritableDatabase()方法以读写方式打开数据库，一旦数据库的磁盘空间满了，数据库就只能读而不能写，倘若使用的是getWritableDatabase()方法就会出错。getReadableDatabase()方法先以读写方式打开数据库，如果数据库的磁盘空间满了，就会打开失败，当打开失败后会继续尝试以只读方式打开数据库。

1.3 常用的数据库添删改查操作

小安：哇，博士您太牛了，源码您都有研究。数据库我们完成了，它的添、删、改、查是不是也很简单啊？

大致：呵呵，下面我就详细的告诉你，如何对数据库执行添、删、改、查操作，你可要认真听啊。

1.3.1 实现添删改查操作

首先，创建一个person类，放在com.sharpandroid.domain包下，如图3-8

Source folder: db/src Browse...

Package: com.sharpandroid.domain Browse...

☐ Enclosing type: Browse...

Name: Person

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

图3-8

然后为Person类添加字段和set、get方法，代码如下：

```
package com.sharpandroid.domain;

public class Person {
    private Integer id;
    private String name;
    private Integer age;

    public Person() {}

    public Person(String name, Integer age) {
        this.name = name;
        this.age = age;
    }

    public Integer getId() {
```



```

    }
    public void setId(Integer id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public Integer getAge() {
        return age;
    }
    public void setAge(Integer age) {
        this.age = age;
    }

    @Override
    public String toString() {
        return "Person [age=" + age + ", id=" + id + ", name=" + name + "]";
    }
}

```

创建一个PersonService类对Person进行添、删、改、查操作，如图3-9

Source folder: db/src Browse...

Package: com.sharpandroid.service Browse...

☐ Enclosing type: Browse...

Name: PersonService

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

图3-9

在PersonService类中，先实例化DatabaseHelper工具类，然后再由外部调用PersonService类传入上下文，代码如下：

```
private DatabaseHelper databaseHelper;
private Context context;

public PersonService(Context context) {
    this.context=context;
    databaseHelper = new DatabaseHelper(context);
}
```

添加方法

下面我们要实现保存操作，代码如下：

```
public void save(Person person) {
    SQLiteDatabase db = databaseHelper.getWritableDatabase();
    db.execSQL("insert into person(name, age) values( 'Tom' ,21)");
}
```

但是如果你这样做的话，你的程序很容易出错，比如用户在你的应用界面上面输入字符包含标点的时候，如图3-10

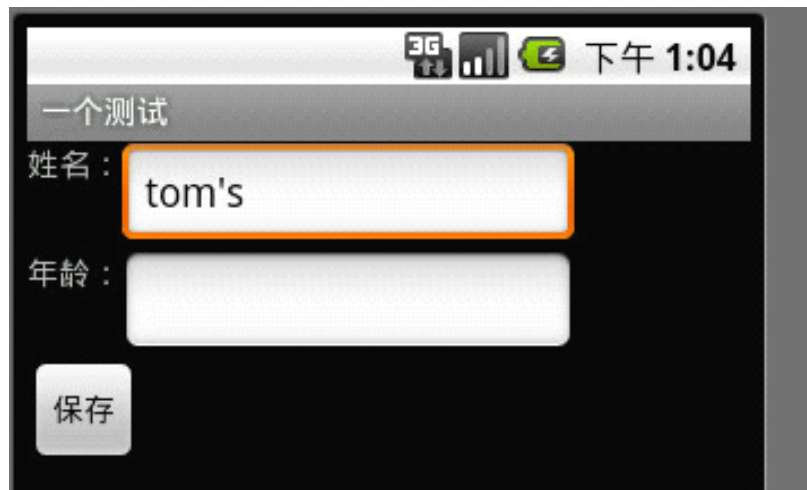


图3-10

如上图所示，用户输入了单引号，在程序组拼之后的SQL语句为“insert into person(name, age) values('tom's' ,21)”，那么程序就会出错，所以我们要这样做，代码如下：

```
public void save(Person person) {
    SQLiteDatabase db = databaseHelper.getWritableDatabase();
    db.execSQL("insert into person(name, age) values(?,?)",
        new Object[] {person.getName(), person.getAge()});
}
```

用占位符“？”，就可以解决上述错误。在save()方法中，是否关闭数据库都可以。

更新方法

我们继续实现更新操作，代码如下：

```
public void update(Person person) {
    SQLiteDatabase db = databaseHelper.getWritableDatabase();
    db.execSQL("update person set name=?,age=? where personid=?",
        new Object[] {person.getName(), person.getAge(),
person.getId()});
}
```

查询方法

实现查找操作，注意查找用到的是rawQuery()方法执行查找操作，代码如下：

```
public Person find(Integer id) {
    SQLiteDatabase db = databaseHelper.getReadableDatabase();
    Cursor cursor = db.rawQuery("select personid,name,age from person where
personid=?", new String[] {String.valueOf(id)});
    if(cursor.moveToNext()) { //迭代记录集
        Person person = new Person(); //实例化person

        person.setId(cursor.getInt(cursor.getColumnIndex("personid")));
        person.setName(cursor.getString(1));
        person.setAge(cursor.getInt(2)); //将查到的字段，放入person，
        return person;
    }
    cursor.close(); //游标关闭
    return null;
}
```

上述代码中的Cursor，跟JDBC中的ResultSet是一样的，专门用来对记录集进行迭代的，cursor.moveToNext()什么时候为false呢？在移过最后一条记录的时候就会返回false。

删除方法

删除操作大致和保存操作相似，代码如下：

```
public void delete(Integer id) {
    SQLiteDatabase db = databaseHelper.getWritableDatabase();
    db.execSQL("delete from person where personid=?", new Object[] {id});
}
```

分页方法

```

public List<Person> getScrollData(int firstResult, int maxResult){
    List<Person> persons = new ArrayList<Person>();
    SQLiteDatabase db = databaseHelper.getReadableDatabase();
    Cursor cursor = db.rawQuery("select personid,name,age from person
limit ?,?",
        new String[] {String.valueOf(firstResult),
String.valueOf(maxResult)});    //firstResult开始索引
    while(cursor.moveToNext()){    //maxResult每页获取的记录数

        Person person = new Person();
        person.setId(cursor.getInt(cursor.getColumnIndex("personid")));
        person.setName(cursor.getString(1));
        person.setAge(cursor.getInt(2));
        persons.add(person);
    }
    cursor.close();
    return persons;
}

```

获取记录总数方法

获取记录总数操作，代码如下：

```

public long getCount() {
    SQLiteDatabase db = databaseHelper.getReadableDatabase();
    Cursor cursor = db.rawQuery("select count(*) from person", null);
    //没有占位符参数的话，直接用 null
    cursor.moveToFirst();
    long count = cursor.getLong(0);
    cursor.close();
    return count;
}

```

1.3.2 测试业务

业务代码我们写完了，接下来就要测试是否通过了。

新建一个PersonServiceTest测试类，测试上述操作，如图3-11

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

图3-11

测试保存

然后，在PersonServiceTest类中编写testSave()测试方法,代码如下：

PersonServiceTest.java

```
public void testSave() throws Throwable{ //测试保存方法
    PersonService personService = new PersonService(this.getContext());
    //传入上下文
        Person person = new Person("Tom", 21);
        personService.save(person);
    }
```

然后，执行测试方法，如图3-12

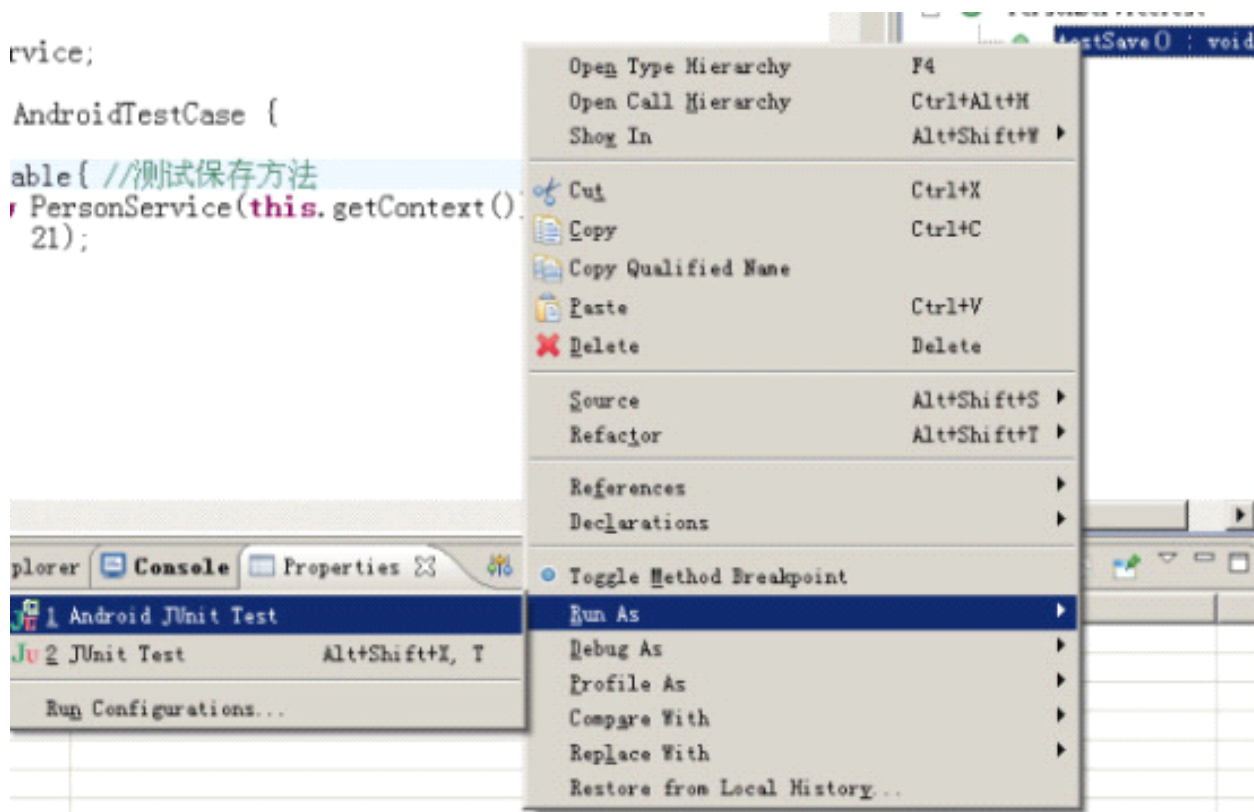


图3-12

当运行结构为图3-13时，表示保存成功。

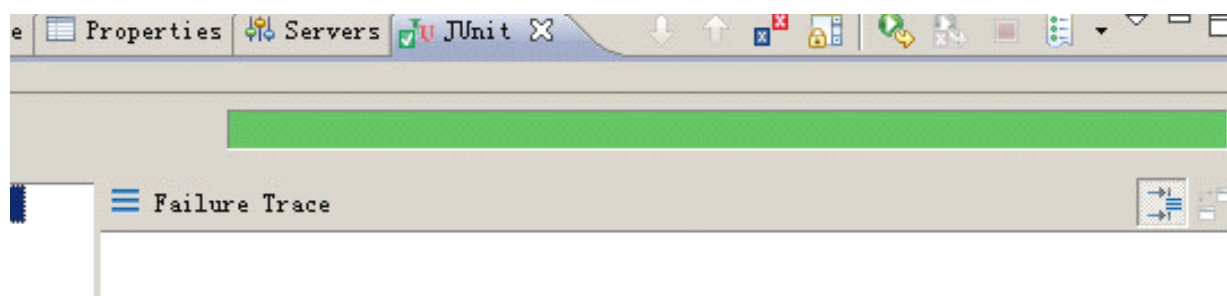


图3-13

如果要查看保存的数据。首先，在本应用软件所在包的databases文件夹下（图3-7），选中数据库文件，单击导出按钮（图3-14），把数据库文件导出到桌面。（图3-15）

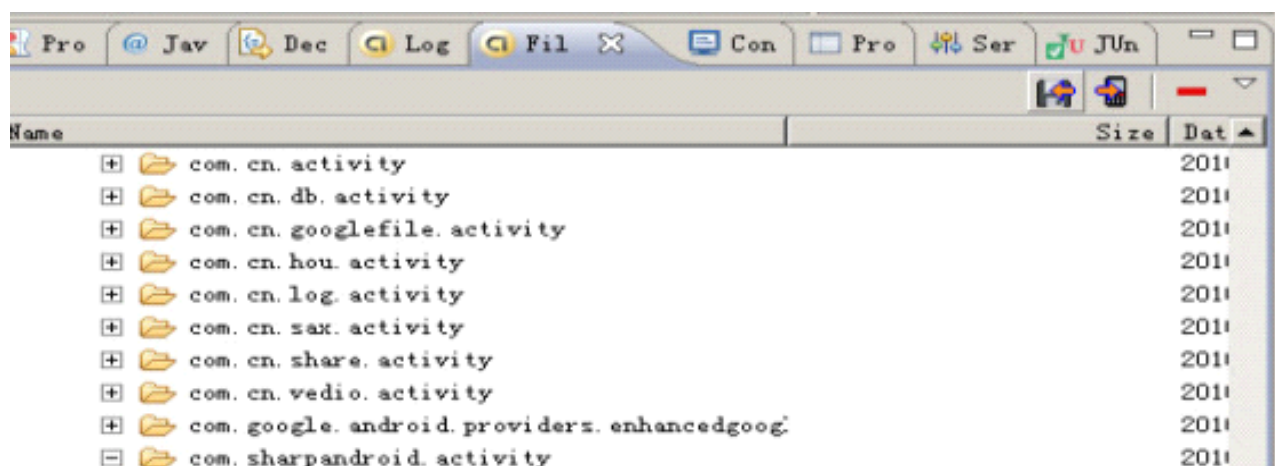


图3-14



图3-15

Sharp.db导出成功之后,可以用SQLite Developer软件查看数据库中的表,首先,启动SQLite Developer软件,添加刚才导出到桌面的sharp.db文件(图3-16),别名一定要填。

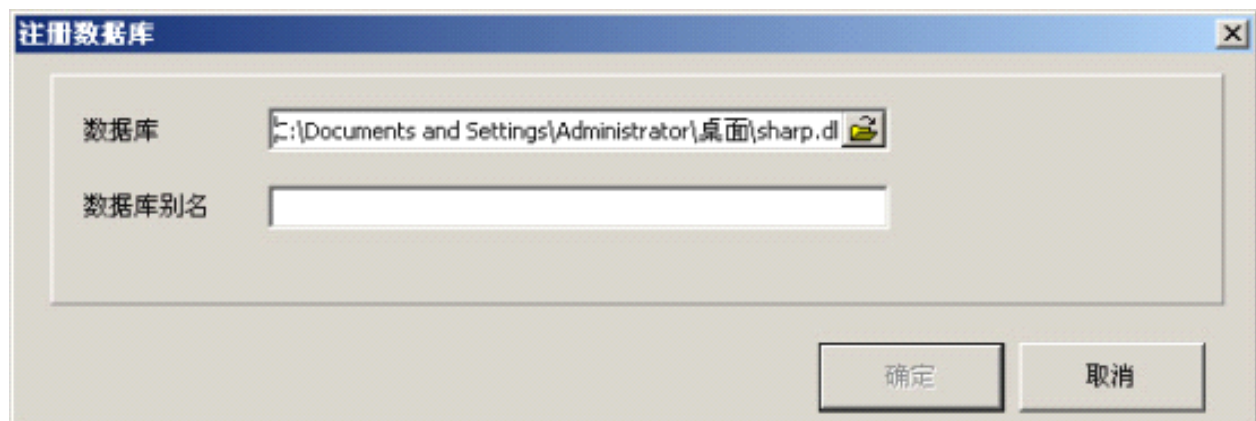


图3-16

找到并且打开刚才建立的person表,如图3-17



图3-17

然后查看添加的数据，是否成功，图3-18

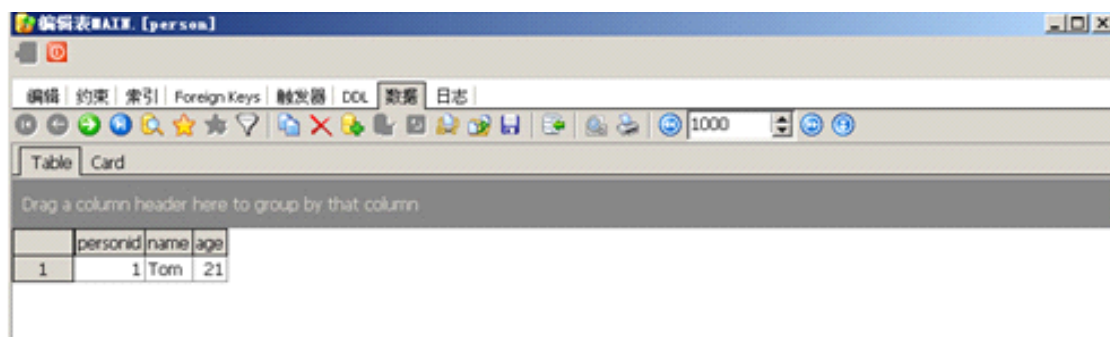


图3-18

我们发现数据已经添加成功。

测试查找

在PersonServiceTest类中编写testFind()测试方法,代码如下:

```
private static final String TAG = "PersonServiceTest";
public void testFind() throws Throwable{ //测试查找方法
    PersonService personService = new PersonService(this.getContext());
    Person person = personService.find(1);
    Log.i(TAG, person.toString());
}
```

执行testFind()测试,如图3-12和3-13,在logCat中加入过滤器PersonServiceTest如图3-19

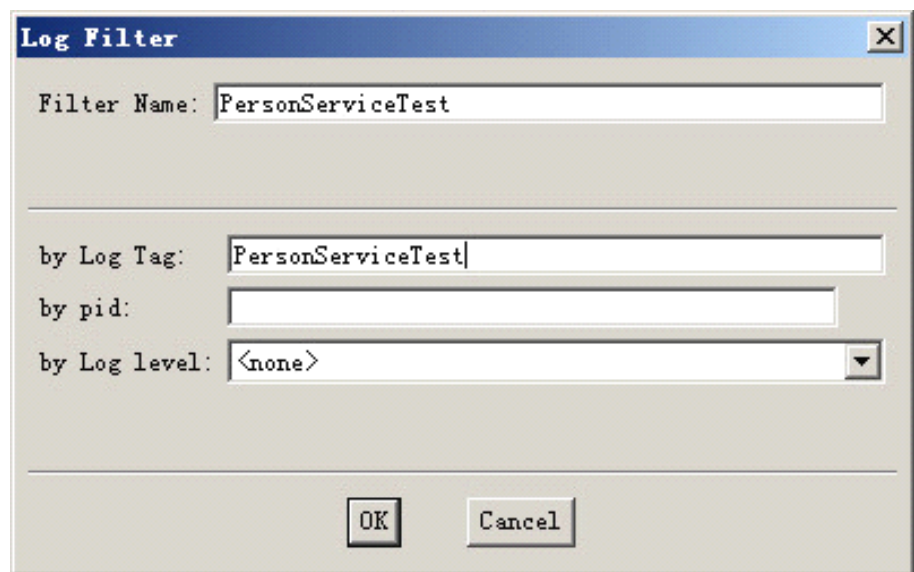
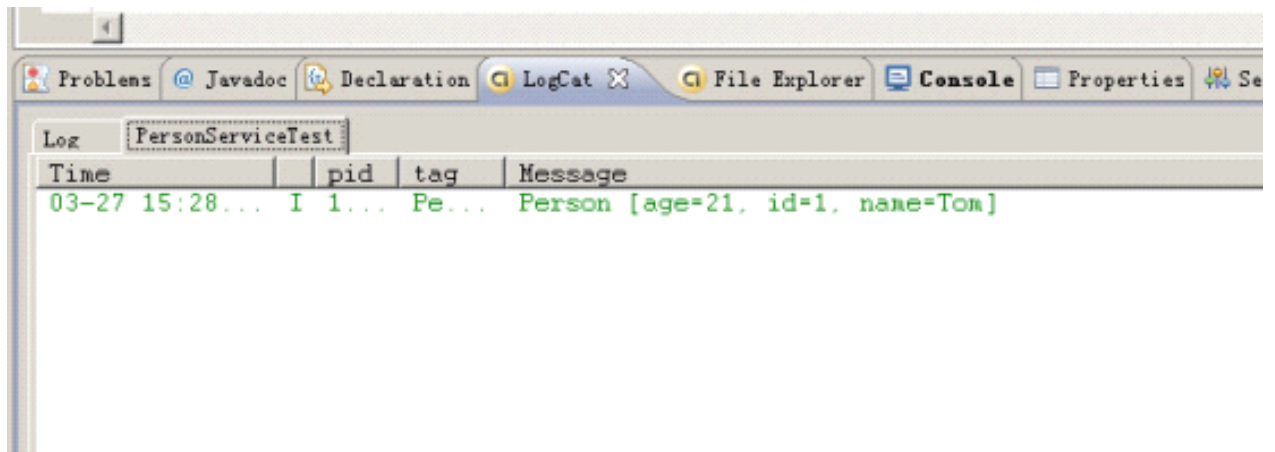


图3-19

查看结果，查找成功！如图3-20：



测试更新

在PersonServiceTest类中编写testUpdate()测试方法,代码如下：

```
public void testUpdate() throws Throwable{//测试更新方法
    PersonService personService = new PersonService(this.getContext());
    Person person = personService.find(1); //把第一条记录名字改为Jim
    person.setName("Jim");
    personService.update(person);
}
```

执行testUpdate()测试，然后再运行testFind() 查看是否更新成功，如图3-21

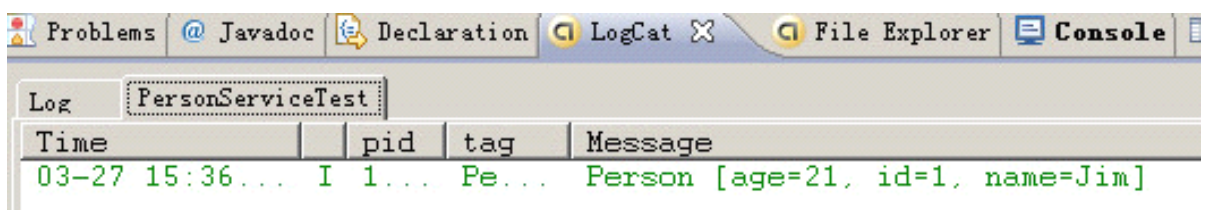


图3-21

更新测试成功。

测试记录总数

在PersonServiceTest类中编写testCount()测试方法,代码如下:

```
public void testCount() throws Throwable{//测试记录总数
    PersonService personService = new PersonService(this.getContext());
    Log.i(TAG, personService.getCount()+"");
}
```

执行testCount()测试,运行结果如图3-22

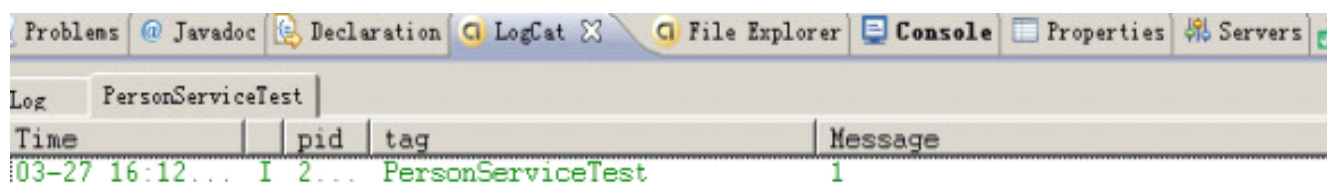


图3-22

测试成功。

测试分页

在PersonServiceTest类中编写testGetScrollData()测试方法,代码如下:

```
public void testGetScrollData() throws Throwable{ //测试分页
    PersonService personService = new PersonService(this.getContext());
    List<Person> persons = personService.getScrollData(0, 3);
    for(Person person : persons){
        Log.i(TAG, person.toString());
    }
}
```

我们只显示出前三条记录,分页测试之前在testSave()方法中用for语句添加十条记录,如下代码:

```
public void testSave() throws Throwable{ //测试保存方法
    PersonService personService = new PersonService(this.getContext());
    //传入上下文
    for(int i=0;i<10;i++){
        Person person = new Person("Tom"+i, 21); //i 用来标记
        personService.save(person); //添加十条记录
    }
}
```

添加十条记录成功之后,运行测试testGetScrollData(),运行结果如图3-23

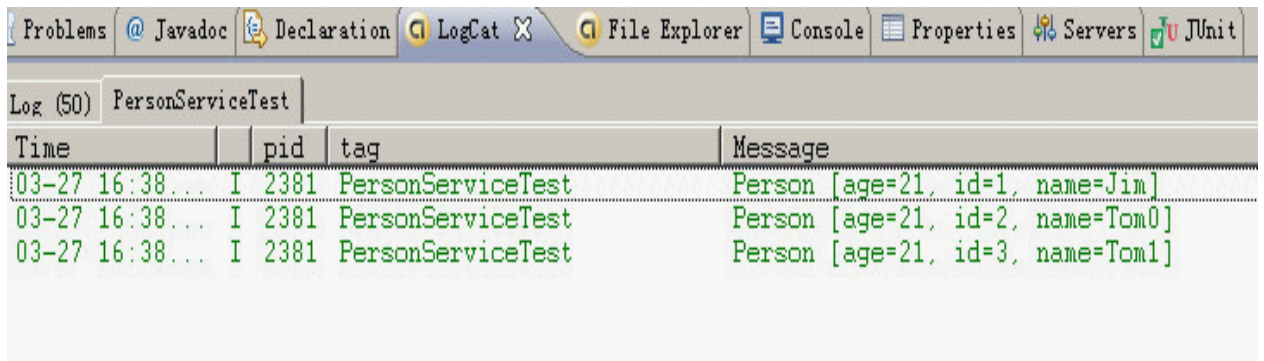


图3-23

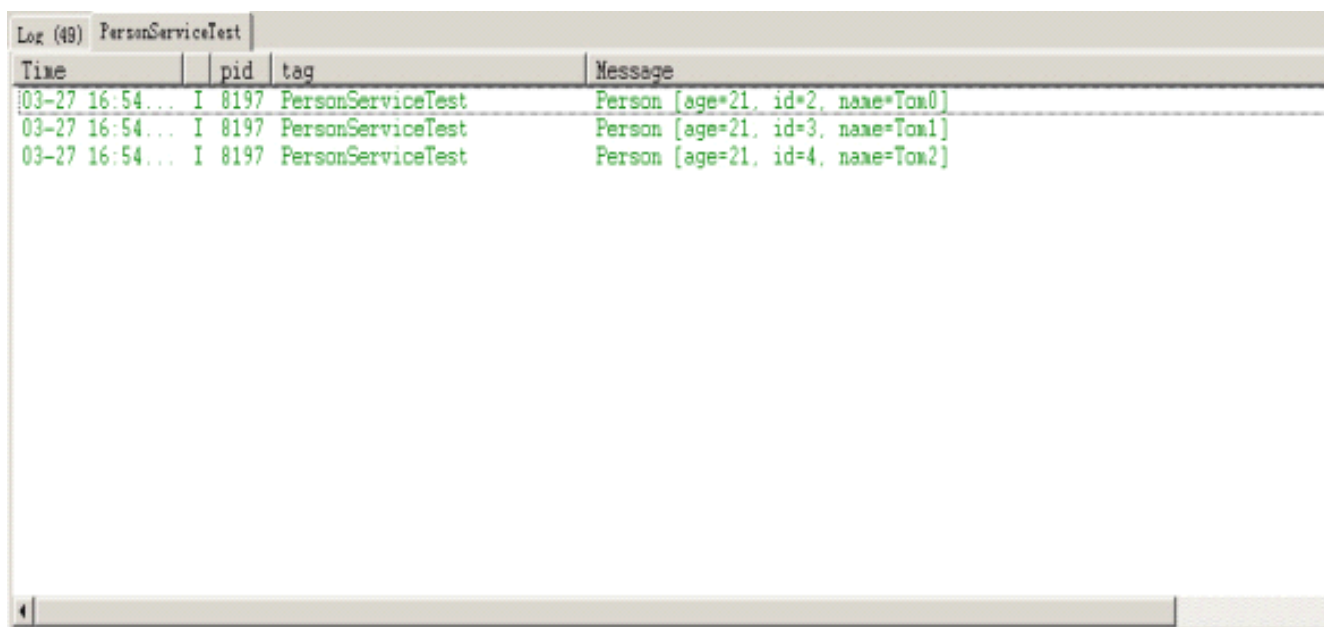
分页测试成功。

测试删除

在PersonServiceTest类中编写testDelete()测试方法,代码如下:

```
public void testDelete() throws Throwable{    //测试删除
    PersonService personService = new PersonService(this.getContext());
    personService.delete(1);
}
```

运行testDelete()测试通过之后,再执行testGetScrollData()方法查看是否还有id为1的记录,运行结果为图3-24



Id为1的记录已经成功删除。

1.4 另一种实现添删改查的方法

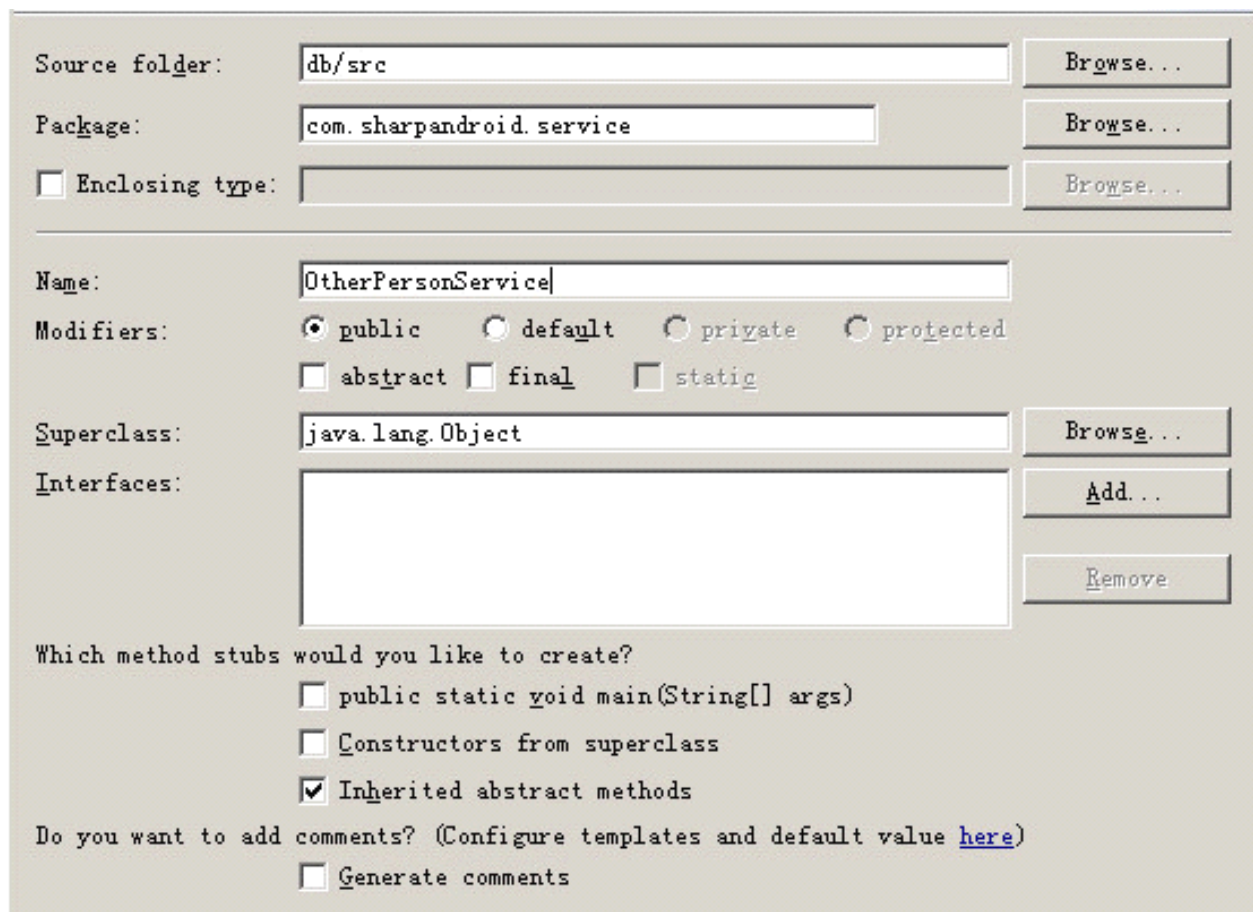
大致: 小安, 你知道吗, Android系统除了使用rawQuery()和execSQL()方法操作数据外, 还有另一种API用来实现对数据的操作。

小安: 那您快给我讲讲吧。

大致: 看如下代码。

1.4.1 实现添删改查操作

首先，我们还是新建OtherPersonService类，如图3-25



Source folder: db/src Browse...

Package: com.sharpandroid.service Browse...

☐ Enclosing type: Browse...

Name: OtherPersonService

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass: java.lang.Object Browse...

Interfaces: Add... Remove

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

图3-25

在 OtherPersonService 类中，和之前讲过的 PersonService 一样，先实例化 DatabaseHelper 工具类，然后由外部调用 OtherPersonService 类传入上下文，代码如下：

```
private DatabaseHelper databaseHelper;  
private Context context;  
public OtherPersonService(Context context) {  
    this.context=context;  
    databaseHelper = new DatabaseHelper(context);  
}
```

添加

我们要实现添加操作代码如下：

```
public void save(Person person) {  
    SQLiteDatabase database = databaseHelper.getWritableDatabase();  
    ContentValues values = new ContentValues();
```

```

        values.put("age", person.getAge());
        database.insert("person", "name", values);
        //database.close();
    }

```

上述代码insert()方法中：

第一个参数“person”：表示表名。

第二个参数“name”：表示当values为null时，用来拼接成完整的SQL语句，如

Insert into person (name) values (null)。

第三个参数“values”：表示添加的数据，其中ContentValues类是和map相似的一种结构。

注意：只要调用insert()方法，就要添加一条记录，不管values是否为null，用第二个参数拼凑成完整的SQL语句。

更新

实现更新操作，代码如下：

```

public void update(Person person) {
    SQLiteDatabase database = databaseHelper.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put("name", person.getName());
    values.put("age", person.getAge());
    database.update("person", values, "personid=?", new
String[] {String.valueOf(person.getId())});
}

```

上述update()方法中：

第一个参数“person”：表示表名。

第二个参数“values”：表示更新的数据。

第三个参数“personid=? ”：表示SQL语句中条件部分的语句。

第四个参数：表示占位符的值。

查找

实现查找操作，代码如下：

```

public Person find(Integer id) {
    SQLiteDatabase database = databaseHelper.getWritableDatabase();
    Cursor cursor = database.query("person", new
String[] {"personid", "name", "age"}, "personid=?", new String[] {String.valueOf(id)},
null, null, null);
    if(cursor.moveToNext()) {
        Person person= new Person();
        person.setId(cursor.getInt(0));
        person.setName(cursor.getString(1));
    }
}

```

```

        return person;
    }
    return null;
}

```

上述代码query()方法中：

第一个参数：表示表名。

第二个参数：表示查找需要返回的的字段。

第三个参数：表示SQL语句中的条件语句。

第四个参数：表示占位符的值。

第五个参数：表示分组，可设为null。

第六个参数：表示SQL语句中的having，可设为null。

第七个参数：表示结果的排序，可设为null。

删除

实现删除操作，代码如下：

```

public void delete(Integer id) {
    SQLiteDatabase database = databaseHelper.getWritableDatabase();
    database.delete("person", "personid=?", new String[] {String.valueOf(id)});
}

```

上述代码中delete()方法的参数和update()方法的一样。

分页

实现分页操作，代码如下：

```

public List<Person> getScrollData(int startResult, int maxResult) {
    List<Person> persons = new ArrayList<Person>();
    SQLiteDatabase database = databaseHelper.getWritableDatabase();
    Cursor cursor = database.query("person", new String[] {"personid", "name",
"age"}, null, null, null, null, "personid desc", startResult+ "", "+ maxResult);
    while(cursor.moveToNext()) { //迭代添加到persons
        Person person= new Person();
        person.setId(cursor.getInt(0));
        person.setName(cursor.getString(1));
        person.setAge(cursor.getInt(2));
        persons.add(person);
    }
    return persons;
}

```

上述代码query()方法中：

第二个参数：表示查找需要返回的的字段。

第三个参数：表示SQL语句中的条件语句。

第四个参数：表示占位符的值。

第五个参数：表示分组，可设为null。

第六个参数：表示SQL语句中的having，可设为null。

第七个参数：表示结果的排序，可设为null。

第八个参数：表示分页，组拼成完整的SQL语句。

获取记录总数

实现获取记录总数的操作，代码如下：

```
public long getCount() {  
    SQLiteDatabase database = databaseHelper.getWritableDatabase();  
    Cursor cursor = database.query("person", new String[] {"count(*)"}, null, null,  
    null, null, null);  
    if(cursor.moveToNext()) {  
        return cursor.getLong(0);  
    }  
    return 0;  
}
```

获取记录总数，Android系统中并没有提供对应的方法，所以我们用query()方法，组拼完整的SQL语句，实现获得记录总数。

1.4.2 测试业务

小安：我们是否可以用以前的方法进行测试？

大致：对，可以的。思路一样的，具体是这样的。

新建一个测试类OtherPersonServiceTest, 如图3-26

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

图3-26

这个类和之前的测试类PersonServiceTest内容几乎一样，只是把PersonService类名换成OtherPersonService类名，然后测试各个方法是否成功，OtherPersonServiceTest测试类代码如下：

```
package com.sharppandroid.activity;

import java.util.List;
import android.test.AndroidTestCase;
import android.util.Log;
import com.sharppandroid.domain.Person;
import com.sharppandroid.service.OtherPersonService;
import com.sharppandroid.service.PersonService;

public class OtherPersonServiceTest extends AndroidTestCase {

    private static final String TAG = "OtherPersonServiceTest";

    public void testSave() throws Throwable{ //测试保存方法
        OtherPersonService otherPersonService = new
        OtherPersonService(this.getContext()); //传入上下文
        for(int i=0;i<10;i++)
        {
            Person person = new Person("Tom"+i, 21);
```

```

    }

}

    public void testFind() throws Throwable{ //测试查找方法
        OtherPersonService otherPersonService = new
OtherPersonService(this.getContext());
        Person person = otherPersonService.find(1);
        Log.i(TAG, person.toString());
    }

    public void testUpdate() throws Throwable{//测试更新方法
        OtherPersonService otherPersonService = new
OtherPersonService(this.getContext());
        Person person = otherPersonService.find(1); //把第一条记录名字改为Jim
        person.setName("Jim");
        otherPersonService.update(person);
    }

    public void testCount() throws Throwable{//测试记录总数
        OtherPersonService otherPersonService = new
OtherPersonService(this.getContext());
        Log.i(TAG, otherPersonService.getCount()+"");
    }

    public void testGetScrollData() throws Throwable{ //测试分页
        OtherPersonService otherPersonService = new
OtherPersonService(this.getContext());
        List<Person> persons = otherPersonService.getScrollData(0, 3);
        for(Person person : persons){
            Log.i(TAG, person.toString());
        }
    }

    public void testDelete() throws Throwable{ //测试删除
        OtherPersonService otherPersonService = new
OtherPersonService(this.getContext());
        otherPersonService.delete(1);
    }
}

```

大致：经过测试可以实现和第一种方法一样的效果，说明这也是一种添、删、改、查操作数据库的方式，对于这两种方式推荐大家以后在开发中使用第一种，因为它比较容易理解，更容易让别人读懂你的代码。