

# 基于 Android 系统的数据存储访问机制研究

倪红军

(南京师范大学泰州学院 信息工程学院, 江苏 泰州 225300)

**摘要:**数据存储是开发应用程序时需要解决的最基本的问题,数据必须以某种方式保存,并且能够有效、方便的使用和更新处理。Android 系统是基于 Google 推出的能在智能终端设备上运行的操作系统,随着应用范围和开发需求的增大,对软件的开发效率、性能及数据的存储访问机制受到普遍关注。文中从 SharedPreferences、文件、SQLite、ContentProvider、网络五个方面深入阐述了 Android 系统的数据存储访问机制原理,并结合它们的内在原理,给出了具体实现方法。最后根据它们各自的优缺点,分析了各类存储访问机制的适用范围。

**关键词:**应用开发;数据存储;访问;实现方法;适用范围

中图分类号:TN929.53

文献标识码:A

文章编号:1673-629X(2013)06-0090-04

doi:10.3969/j.issn.1673-629X.2013.06.023

## Study of Data Storage Access Mechanism Based on Android System

NI Hong-jun

(College of Information Engineering, Taizhou College of Nanjing Normal University, Taizhou 225300, China)

**Abstract:** The data storage is the most basic issues that need to be addressed when developing applications, the data must be saved in some way, and to be effective, easy to use and update processing. The Android system is introduced based on Google in the intelligent terminal equipment operation of the operating system, along with the increase of application and development demand, the software development efficiency performance and data storage access mechanism has been paid more attention. From the SharedPreferences, files, SQLite, ContentProvider, network five aspects discussed deeply the data memory access mechanism principle of Android system, and combined with their intrinsic principle, the particular method of implementation was introduced. Finally based on their respective advantages and disadvantages, analyze the applicable scope of the various memory access mechanism.

**Key words:** application development; data storage; access; realization method; application scope

## 0 引言

随着 Android 系统的智能终端市场占有率逐渐走高,用户对该类设备的应用软件需求也逐渐增多。同时软件的功能也越来越多,数据存储量也逐渐增大,对数据的存储访问更加频繁。而 Android 系统自有一套存储访问机制,开发者对该机制的深入理解和合理使用会直接影响软件的开发效率和性能。因此对 Android 系统的数据存储访问机制研究有很重要的意义。

## 1 Android 简介

Android 是 Google 开发的基于 Linux 开放性内核的开源手机操作系统,目前已广泛应用于手机或平板电脑等智能终端。Android 系统从软件层次上来说,其

结构包含操作系统、中间件及一些关键的平台应用程序<sup>[1]</sup>。它采用软件堆层(Software Stack)的架构,主要分为四个部分:①应用层;②应用框架层;③系统运行库层;④Linux 内核层<sup>[2,3]</sup>。

## 2 Android 存储访问机制

在进行各类应用开发时,涉及的数据存储访问方式有三类:文件、数据库和网络。其中文件和数据库存储方式多用于离线应用开发中,文件使用较方便,不需要数据库管理系统的支持就可以进行存储访问;而数据库用起来较为复杂,但它有其强大的优点,比如在海量数据时性能优越、能方便进行数据的增删改查、可以跨平台等等;网络存储方式则用于比较重要的项目事

收稿日期:2012-09-10

修回日期:2012-12-16

网络出版时间:2013-03-05

基金项目:江苏省泰州市 2011 年计划性课题(201153);Google 中国大学合作部 2012 年 Android 创新(开发)重点资助项目(SOW12-11/64005799)

作者简介:倪红军(1975-),男,江苏靖江人,硕士,讲师,研究方向为管理信息系统、Android 应用开发。

网络出版地址: <http://www.cnki.net/kcms/detail/61.1450.TP.20130305.0816.020.html>

务,比如在线售票、天气预报等实时数据需要通过网络传输到数据处理中心进行存储、处理。

Android 应用开发也会涉及数据存储访问,结合 Android 系统的特点其存储方式总体也分为上面三类,但从开发者角度来讲,它包含五种存储访问机制。

## 2.1 SharedPreferences 存储访问机制

### 2.1.1 原理

SharedPreferences 是一种用来存储简单配置信息的机制,也是 Android 平台上一个轻量级的存储类。只能存储 Long、Float、Integer、String 和 Boolean 五种基本类型。它是通过“key—value 对”的机制将数据存储存储在 XML 文件中,存储位置默认在/data/data/<包名>/shared\_prefs 目录下。

### 2.1.2 实现方法

#### (1) SharedPreferences 存储数据方法。

SharedPreferences 对象本身只能获取数据而不支持存储和修改,存储修改是通过 Editor 对象实现的。代码如下:

```
Context context = this.getApplicationContext();
SharedPreferences sharedPreferences = context.getSharedPreferences("nnutinfo", Context.MODE_PRIVATE); // 根据 Context 获取 SharedPreferences 对象
Editor editor = sharedPreferences.edit(); // 获取 Editor 对象
editor.putString("schoolName", "泰州学院"); // 学校名称;
通过 Editor 对象存储 key-value 键值对数据。
editor.putInt("schoolCount", 635); // 职工人数
editor.commit(); // 提交数据
```

执行上述代码后,生成的 nnutinfo.xml 文件,内容如下:

```
<? xml version='1.0' encoding='utf-8' standalone='yes'? >
<map>
<string name="schoolName">泰州学院</string>
<int name="schoolCount" value="635" />
</map>
```

#### (2) SharedPreferences 读取数据方法。

SharedPreferences 对象读取数据比较简单,使用对象.get \* \* ("键名", "缺省值")格式就可方便地读出数据,其中第二个参数为缺省值,如果 SharedPreferences 中不存在该键名(key),将返回缺省值。代码如下:

```
String schoolName = sharedPreferences.getString("schoolName", "");
int schoolCount = sharedPreferences.getInt("schoolCount", 1);
```

## 2.2 文件数据存储访问机制

### 2.2.1 原理

Android 系统采用 java.io.\* 库所提供的 I/O 接口来实现文件读写。同时引入了资源文件,用于存储应用程序所需的一些资源,如图片、字符串等。每一种资

源文件的语法、格式、保存位置取决于资源类型,在进行开发时,需要在 res/目录下的适当子目录下创建和存储资源文件。如 res/layout、res/drawable、res/raw 等目录。可以通过 R.resource\_type.resource\_name 语句来直接引用这些资源。

### 2.2.2 实现方法

#### (1) 流文件的读写。

可以使用 FileOutputStream 类中 openFileOutput() 方法把数据输出到文件中,该文件默认保存在/data/data/<包名>/files 目录下,也可以通过修改 SDCard 的访问权限将该文件存放在 SDCard 卡上。使用 FileInputStream 类中 openFileInput() 方法从文件中读出数据。此方法与 J2SE 环境基本相同,这里不再赘述。

#### (2) XML 文件的读写。

XML 即可扩展标记语言,可以用来标记数据、定义数据类型,允许用户对自己的标记语言进行定义的源语言。由于 XML 文件易于与 Windows、Mac OS、Linux 等平台下产生的信息结合,现在已经成为数据交换的公共语言。所以在进行 Android 应用开发时也可以采用 XML 文件存储数据;可以使用 Simple API for XML(SAX)、Document Object Model(DOM)及自带的 pull 解析器解析 XML 文件。

SAX<sup>[4]</sup>是一个解析速度快并且占用内存少的解析器,非常适合 Android 等移动设备。它采用事件驱动,不需要解析完整文档,在按内容顺序解析文档的过程中,SAX 会判断当前读到的字符是否符合 XML 语法中的某部分,如果符合就会触发事件。这类事件定义在 ContentHandler 接口<sup>[5]</sup>,包括 startDocument()、endDocument()、startElement()、endElement()、characters()等。SAX 解析代码如下:

```
public List<Person> getPerson(InputStream instream) throws Throwable {
    SAXParserFactory factory = SAXParserFactory.newInstance();
    SAXParser parser = factory.newSAXParser(); // 创建一个 SAX 解析器
    PersonPaser personpaser = new PersonPaser();
    parser.parse(instream, personpaser);
    return PersonPaser.getPersons();
}
```

通过 SAX 解析器对 XML 文件进行解析,使用 parse(InputStream, DefaultHandler)方法;第一个参数表示输入流传入的 XML 文件,第二个参数表示根据读入的 XML 文件的标记来调用 DefaultHandler 类中的方法,该类中的这些方法都需要进行重写,上面代码中的 PersonPaser 是一个内部类,在该类中根据 XML 语法中的标记编写了方法来实现对 XML 文档的解析。

DOM 解析 XML 文件时,会将 XML 文件的所有内容以文档树方式存放在内存中,然后使用 DOM API 遍历 XML 树、检索所需的数据。使用 DOM<sup>[6]</sup> 操作 XML 的代码比较直观,编码比 SAX 的实现简单,但是,DOM 需要将 XML 文件的所有内容存放到内存中,所以内存的消耗较大,对于运行 Android 的移动设备来说,设备资源比较宝贵,不适合使用。

Pull 解析器是 Android 系统内置用来解析 XML 文件的。也提供了事件,可以使用 `parser.next()` 进入下一个元素并触发相应事件<sup>[6,7]</sup>。事件将作为数值代码被发送。Pull 解析代码如下:

XmlPullParser parser = Xml.newPullParser(); // 得到一个 Pull 解析器

```
parser.setInput(instream, "UTF-8");
int eventType = parser.getEventType();
while(eventType != XmlPullParser.END_DOCUMENT) {
    switch(eventType) {
        case XmlPullParser.START_DOCUMENT: // XML 文档开始
            .....
            break;
        case XmlPullParser.START_TAG: // 元素开始标记
            .....
            break;
        case XmlPullParser.END_TAG: // 元素结束标记
            .....
            break;
    }
    eventType = parser.next();
}
```

另外有时也需要生成一个 XML 文件,生成 XML 文件的方法有很多。如:可以只使用一个 `StringBuilder` 组建 XML 内容,然后把内容写入到文件中;也可以使用 DOM API 生成 XML 文件,或者使用 Pull 解析器生成 XML 文件,根据编码的直观性和执行效率,使用 Pull 解析器较适合。

## 2.3 SQLite 数据库数据存储访问机制

### 2.3.1 原理

SQLite 是一个轻量级嵌入式数据库引擎,它支持 SQL 语言,并且只占用很少的内存。它由 SQL 编译器、内核、后端以及附件四个部分组成。SQLite 在创建表时,可以把任何数据类型放入任何列中;在插入数据时,如果该类型与关联的列不匹配,则 SQLite 会尝试将该值转换成该列的类型,如果不能转换,则该值将作为其本身具有的类型存储。

对于熟悉 Java EE 的开发人员,在 Android 开发中使用 SQLite 比较简单。但是,由于 JDBC 消耗资源太多,所以 JDBC 对于智能终端类内存受限的设备来说并不合适。因此,Android 系统提供了一些新的 API 来

使用 SQLite 数据库。

### 2.3.2 实现方法

Android 系统提供了 `SQLiteOpenHelper` 抽象类,通过继承该 `SQLiteOpenHelper` 类并重写 `onCreate()`、`onUpgrade()` 这两个方法来实现数据库的创建及升级更新。数据库默认存储在 `data/<包名>/databases/` 下,也可通过 `android.os.Environment.getExternalStorageDirectory().getAbsolutePath()` 方法获取 SDCard 卡目录并将数据库存储在 SDCard 卡上。代码如下:

```
public class DBOpenHelper extends SQLiteOpenHelper {
    private static final String DBName = "nnutc.db";
    private static final int DBVersion = 1;
    public DBOpenHelper(Context context, int DBVersion) {
        super(context, DBName, null, DBVersion);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL("CREATE TABLE IF NOT EXISTS person (personid integer primary key autoincrement, " + "name varchar(20), age INTEGER)");
    }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL("ALTER TABLE person ADD COLUMN sex varchar(2)");
    }
}
```

为了实现数据库内容的增删改查,Android 系统提供了一个 `SQLiteDatabase` 的类,使用该类封装的方法可以完成 `Insert`、`Select`、`Update` 和 `Delete` 等操作。执行时用 `SQLiteDatabase` 类中的 `execSQL()` 和 `rawQuery()` 方法。代码如下:

```
SQLiteDatabase db = ....;
db.execSQL("insert into person (name, age) values('张三', 25)");
Cursor cursor = db.rawQuery("select * from person", null);
while (cursor.moveToNext()) {
    String name = cursor.getString(1); // 获取第二列的值,第一列的索引从 0 开始
}
```

查询语句执行后会返回一个 `Cursor`,它是 Android 系统的数据库游标,可以使用循环语句遍历所有记录。

## 2.4 ContentProvider 共享数据机制

### 2.4.1 原理

Android 系统中的文件数据和数据库数据都是私有的。一般情况下多个应用程序之间不能进行数据交换。为了解决这个问题,它提供了 `ContentProvider` 类,该类实现了一组标准的方法接口,能够让其他的应用保存或读取此 `ContentProvider` 的各种数据。另外 An-

droid 系统也提供了一些已经在系统中实现的标准 ContentProvider, 比如联系人信息、图片库等, 开发者可以用这些 ContentProvider 来访问设备上存储的信息。

#### 2.4.2 实现方法

一个应用程序通过实现一个 ContentProvider 的抽象接口将数据暴露出去。常见的接口有 query(通过 Uri 进行查询, 返回一个 Cursor)、insert(将一组数据插入到 Uri 指定的地方)、update(更新 Uri 指定位置的数据)、delete(删除指定 Uri 并符合一定条件的数据), 每个 ContentProvider 定义一个唯一的公开的 Uri, 用于指定它的数据集。Uri 由 3 个部分组成, “content://”, 代表数据的路径和一个可选的标识数据的 ID。

当外部应用程序需要对 ContentProvider 中的数据进行添加、删除、修改和查询操作时, 使用 ContentResolver 类来完成<sup>[8,9]</sup>。代码如下:

```
Content Resolver resolver=getContentResolver();
Uri uri=Uri.parse("content://<包名>/person");
ContentValue scontentValues = new ContentValues();
contentValues.put("name", "李小乐");
resolver.insert(uri, contentValues); //添加一条记录
Cursor cursor = resolver.query(uri, null, null, null, "person-
id desc");//获取 person 表中所有记录
Uri updateIdUri = ContentUris.withAppendedId(uri, 2);
resolver.update(updateIdUri, contentValues, null, null); //
把 id 为 2 的记录的 name 字段值更改为李小乐
Uri deleteIdUri = ContentUris.withAppendedId(uri, 1);
resolver.delete(deleteIdUri, null, null); //删除 id 为 1 的记
录
```

### 2.5 网络数据存储访问机制

#### 2.5.1 原理

在实际开发时, 有时也需要将数据以文件的方式上传到服务器或者从服务器读取。Android 系统的网络存储使用 HTTP 协议, 由于 Android 是使用 Java 来开发的, 所以网络开发也使用 J2SE 的包<sup>[10]</sup>。

#### 2.5.2 实现方法

Android 系统智能终端要实现与服务器交互数据可以建立一个 Web 应用, 在 Web 应用的相关请求处理中接收 Android 提交的数据、返回 XML 数据或 JSON 数据<sup>[11,12]</sup>。Android 系统终端发送相应的请求并接收服务器端传送的请求数据。代码如下:

```
URL url = new URL("http://www.nnutc.edu.cn/ie/ie.
gif");
HttpURLConnection conn = (HttpURLConnection) url.open-
Connection();
conn.setConnectTimeout(6 * 1000);
InputStream inStream = conn.getInputStream();
获得 InputStream 后对数据的操作与 J2SE 类似。
```

### 3 适用范围分析

上述五种方式各有其优缺点, 在进行应用开发时要根据不同的实际情况来选择, 下面结合各种方式的优缺点谈谈最合适的使用情况。

SharedPreferences 是用来存储一些 key—value 对的基本数据类型, 最适合使用 SharedPreferences 的地方就是保存配置信息。当然由于 SharedPreferences 使用十分方便, 所以能用它就尽量不要用文件或是数据库。对于数据量大的应用, 就需要选择文件或数据库存储。由于智能终端的网络稳定性, 以及所产生的流量等因素, 对使用网络存储有很大影响。但若是非常重要的实时数据, 或是需要发送给远端服务器处理的, 就必须使用网络时进行实时发送。


### 4 结束语

文中对 Android 系统的五种存储访问机制进行研究, 在阐述其实现原理的同时, 展示了各种方式的实现代码, 最后通过对各种方式优缺点的分析, 给出了一般适用范围, 为理解 Android 存储访问机制的原理, 合理使用相应接口, 应用程序的开发颇有好处。

#### 参考文献:

- [1] 公磊, 周聪. 基于 Android 的移动终端应用程序开发与研究[J]. 计算机与现代化, 2008(8): 85-89.
- [2] 代敏, 张晶. 基于 Android 平台的嵌入式软件“混合”定位控制策略研究[J]. 科学技术与工程, 2012, 12(5): 1161-1163.
- [3] 姚昱昱, 刘卫国. Android 的架构与应用开发研究[J]. 计算机系统应用, 2008, 17(11): 110-112.
- [4] 朱珊珊, 李书琴, 安福定, 等. XML 文档到关系数据库的转换研究[J]. 计算机工程与设计, 2008, 29(21): 5507-5509.
- [5] Katysovas T. A first look at Google Android[D]. Bohano: Free University of Bohano, 2008.
- [6] 靳岩, 姚尚朗. Google Android 开发入门与实践[M]. 北京: 人民邮电出版社, 2009.
- [7] 郭宏志. Android 应用开发详解[M]. 北京: 电子工业出版社, 2010: 190-192.
- [8] Google. Androidofficialwebsite[EB/OL]. [2009-11-10]. <http://www.android.com>.
- [9] 王向辉, 张国印, 沈洁. Android 应用程序开发[M]. 北京: 清华大学出版社, 2010.
- [10] 杨丰盛. Android 应用开发揭秘[M]. 北京: 机械工业出版社, 2010.
- [11] 3GPPTS34. 171. Termini Conformance Specification: Assisted Global Positioning System[S]. 2004.
- [12] Deitel P J, Deitel H M. Java for programmers[M]. 张君施, 译. 北京: 电子工业出版社, 2010.

## 基于Android系统的数据存储访问机制研究

作者: [倪红军, NI Hong-jun](#)  
作者单位: [南京师范大学泰州学院信息工程学院, 江苏泰州, 225300](#)  
刊名: [计算机技术与发展](#)   
英文刊名: [Computer Technology and Development](#)  
年, 卷(期): 2013, 23(6)

本文链接: [http://d.g.wanfangdata.com.cn/Periodical\\_wjfz201306023.aspx](http://d.g.wanfangdata.com.cn/Periodical_wjfz201306023.aspx)