

## 第 6 讲      数据存储

### 使用说明

【源代码根目录】： 工程目录 DataStorage

【记号】：

(@编程练习)：表明该实验是需要编将工程文件和实验报告一起提交。

(@团队编程练习)：表明该实验是以小组为单位完成的，每个小组完成一份程序和报告即可，报告和工程和其他的实验报告和工程最后要分开交。

本次实验的个人编程部分，请建立在名为 KH6 的 project 中。团队编程练习建立 KHTD6 的工程。

上传方式：<http://disk.lehu.shu.edu.cn/index.aspx>

上传码：

android-sybk (本科)

android-syyjs (研究生)

【实验报告的要求】

- 1、文件名规范： 学号+姓名+实验名称.doc
- 2、内容格式见 实验报告格式.doc

## 目录

第 6 讲	数据存储 .....	1
【课堂实验 KT6.1】	:SharedPreference.....	4
【课堂实验 KT6.2】	:文件的读写 .....	6
【课堂实验 KT6.3】	: 数据库操作 .....	9
【课堂实验 KT6.4】	: Content Provider .....	10
【课堂实验 KT6.5】	:ListView 和 GridView 的用法.....	12
【课后练习 KH6.1】	:文件管理器 .....	13
【课后实验 KH6.2】	比赛管理应用.....	13

### 【前言】

Android 提供了 5 种数据存储的方式

- 1、键值对方式用于存储少量的私有基本类型的数据 Shared Preferences
- 2、文件的方式保存到设备自带存储器中 Internal Storage
- 3、文件的方式保存到扩展存储器中，比如 sdcardExternal Storage
- 4、数据库方式存储结构化的数据 SQLite Databases
- 5、通过网络将数据存储到远程服务器上 Network Connection

另外，xml 是一种特殊的文件，现在的程序设计中大量地运用 xml 来保存数据  
应该根据对数据安全性的要求不同来选择以上合适的存储方式，android 还提供了一种将 app 私有数据进行安全共享的机制--content provider。

手机通讯录就是 content provider 最好的实例。

网络存储是远程存储，这部分主要介绍和本地（这里特指设备带的存储器包括扩展卡）存储有关的 api，因此，关于数据存储的部分，需要了解的就是：

- 1、shared preferences
- 2、文件存储到内部和外部存储器
- 3、数据库
- 4、content provider
- 5、系统 provider 实例 通信录

以上几个部分，我们除了掌握其 **api** 的用法外，还需要了解如何选择合适的存储方式，也就是要知道他们的访问安全性，还有读写的一致性如何保证等。

## 【课堂实验 KT6.1】:SharedPreference

【工程模块】: SharedPreferencesSample

【工程描述】: 通过一个保存短信草稿的例子，展示了如何用 xml 文件保存键值对形式的数据。

【知识点注释】:

### 一、概述

Shared preferences 相关 API 用于在设备自带存储中持久保存 app 的数据。这些数据以键值对（key-value）的形式进行读写。可以保存的数据类型包括常用的基本类型：

- Boolean
- Float
- Int
- Long
- String
- String [Set](#)

在 android 内部，数据是以一个由 android 系统在 app 相关路径下生成的 xml 文件来保存并管理的。默认情况下，这些数据仅仅能在 app 内部被访问，其他 app 是不能访问的（也可以选择允许其他 app 访问这些数据）。而且通常也随着 app 卸载而被清除。访问 shared preferences 无需权限。

**Note:** The SharedPreferences 不要和 Preference APIs 混淆了,后者用来为 app 的配置提供开发接口，尽管其实现机制也是利用了 SharedPreferences 来存储配置信息。下例将展示如何使用 Preference APIs。

### 二、基本用法

#### 1、获得 SharedPreferences 对象

使用如下两个方法来获得 SharedPreferences 对象，该对象代表一个保存了键值对数据的 xml 文件。

**public abstract SharedPreferences getSharedPreferences (String name, int mode)**

- Context 类的方法，该方法需要提供文件名来指定创建或者打开的 xml 文件。
- mode 参数用来指定 xml 文件的访问性。共有 4 种取值
  - MODE\_PRIVATE 0 默认值，只允许 app 内部访问，创建和装载标识
  - MODE\_WORLD\_READABLE 1，允许其他 app 读，创建和装载标识
  - MODE\_WORLD\_WRITEABLE 2，允许其他 app 写，创建和装载标识
  - MODE\_MULTI\_PROCESS 4，允许多个进程读写该对象，装载标识，注意 2.3 以前不需要设置这种特性，系统默认允许多进程读写特性，2.3 以后需要明确设置该标识才能支持多进程读写。

## public SharedPreferences getPreferences (int mode)

- Activity 类的方法，和上面方法不同的是，不需要一个文件名参数，系统会用调用它的 activity 类名作为参数来打开 xml 文件。
- 也可以如下写法：

```
SharedPreferences sharedPref =  
PreferenceManager.getDefaultSharedPreferences(this);
```

- mode 参数：
  - MODE\_PRIVATE 0 默认值，只允许 app 内部访问，创建和装载标识
  - MODE\_WORLD\_READABLE 1，允许其他 app 读，创建和装载标识
  - MODE\_WORLD\_WRITEABLE 2，允许其他 app 写，创建和装载标识

## 2、读写示例

-----写-----

代码举例：

```
SharedPreferences pre = getSharedPreferences(TEMP_SMS, MODE_WORLD_READABLE);  
String content = pre.getString("sms_content", "");  
myEditText.setText(content);
```

-----读-----

```
SharedPreferences.Editor editor = getSharedPreferences(TEMP_SMS,  
MODE_WORLD_WRITEABLE).edit();  
editor.putString("sms_content", myEditText.getText().toString());  
editor.commit();
```

**【问题思考】：**sharedpreferences 对应着系统为每个 app 在其上下文路径中建立的一个 xml 文件。请在 Android Device Monitor（点击工具栏最后一个图标）窗口中，选中当前运行的设备。在右边窗口的 File Explorer 中查看 data/data/{你的 app} 路径下是否有这个 xml，它的名字叫什么。

## 【课堂实验 KT6.2】:文件的读写

【工程模块】: FileIOSample

【工程描述】: 展示对设备的闪存进行文件读写的基本操作

【实验步骤】: 本例子有两个 activity，请分别设置为 launcher，测试其结果，并分析代码。

【知识点注释】:

内部存储器的读写:

1. 用 `openFileOutput()` 获得一个 `FileOutputStream`. (也可以用 `getFilesDir()` 获得一个 app 内部存储的根路径 `File` 对象, java 中 `File` 对象可以理解为一个文件名(含路径)的封装。 `File file = new File(context.getFilesDir(), filename);` `File` 对象也可以构造 `FileOutputStream`. 如: `new FileOutputStream(file)`)
2. Write to the file with `write()`.
3. Close the stream with `close()`.

`openFileOutput()`函数的参数

- `MODE_PRIVATE` will create the file (or replace a file of the same name) and make it private to `MODE_APPEND`
- `MODE_WORLD_READABLE`
- `MODE_WORLD_WRITEABLE`.

```
String FILENAME = "hello_file";
```

```
String string = "hello world!";
```

```
FileOutputStream fos = openFileOutput(FILENAME, Context.MODE_PRIVATE);
```

```
fos.write(string.getBytes());
```

```
fos.close();
```

- 1、内部文件读写基本使用了 `java.io` 的相关函数来完成
- 2、大型文件的读写，如果费时的话，因该写在线程中。不能像本例这样写在主线程中，因为会导致 android 出现“未响应”保护。
- 3、文件操作应该使用 `try catch` 结构。读写文件的操作完全和 `java` 类似，不熟悉的请参考 `java` 关于 I/O 操作的章节。
- 4、`public InputStream openRawResource (int id)` 可以以只读的方式打开一个放在 `res/raw` 下的文件。
- 5、更多的文件操作：
- 6、[`getFilesDir\(\)`](#)

Gets the absolute path to the filesystem directory where your internal files are saved.

[`getDir\(\)`](#)

Creates (or opens an existing) directory within your internal storage space.

[`deleteFile\(\)`](#)

Deletes a file saved on the internal storage.

[`fileList\(\)`](#)

Returns an array of files currently saved by your application.

- 7、如果在 `app` 中要使用临时文件，比如 `cache`，应该使用 `createTempFile` 方法，这样创建的文件系统就可以根据情况自动进行清除：

```
public File getTempFile(Context context, String url) {
    File file;
    try {
        String fileName = Uri.parse(url).getLastPathSegment();
        file = File.createTempFile(fileName, null, context.getCacheDir());
    } catch (IOException e) {
        // Error while creating file
    }
    return file;
}
```

## 二、外部存储区文件的读写

android 将文件系统分成两个区域，所谓 `internal` 和 `external`。前者总是存在的，而后者则因为用户将它作为 `usb` 存储器访问或者拔除等原因而不可访问。注意 `external` 不一定是 `removable` 的 `sdcard`，有些设备将系统自带的固定存储器也分成 `internal` 和 `external` 两个区。

2、若 `app` 的文件不希望被用户通过资源管理器软件或者其他 `app` 访问，则应该将文件存储在 `inetrantl` 区域中，这些文件将随着 `app` 卸载而被删除。

3、存储在 `external` 中的文件可以被任何 `app` 访问，如果存储路径是通过 `getExternalFilesDir()` 来获得的则文件可以跟随 `app` 的卸载被系统删除。存储在 `external` 中的文件分为两种，一种为 `public` 不随 `app` 卸载而删除。一种是 `private`。

Public 文件：被任何 `app` 访问，不会随着创建它的 `app` 卸载而删除。因此应该

通过 [getExternalStoragePublicDirectory\(\)](#) 来获得这个文件。

```
public File getAlbumStorageDir(String albumName) {
    // Get the directory for the user's public pictures directory.
    File file = new File(Environment.getExternalStoragePublicDirectory(
        Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```

Private 文件：随着创建它的 app 卸载而删除。通过 [getExternalFilesDir\(\)](#) 来获得这个文件。

```
public File getAlbumStorageDir(Context context, String albumName) {
    // Get the directory for the app's private pictures directory.
    File file = new File(context.getExternalFilesDir(
        Environment.DIRECTORY_PICTURES), albumName);
    if (!file.mkdirs()) {
        Log.e(LOG_TAG, "Directory not created");
    }
    return file;
}
```



1、访问外部存储设备需要权限

```
<manifest ...>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    ...
</manifest>
<manifest ...>
    <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
    ...
</manifest>
```

定义了写权限，就默认也有了读权限。

2、由于 sd 卡随时不可用因此，需要在读写前检查可用性：

```
/* Checks if external storage is available for read and write */
public boolean isExternalStorageWritable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state)) {
        return true;
    }
    return false;
}

/* Checks if external storage is available to at least read */
public boolean isExternalStorageReadable() {
    String state = Environment.getExternalStorageState();
    if (Environment.MEDIA_MOUNTED.equals(state) ||
        Environment.MEDIA_MOUNTED_READ_ONLY.equals(state)) {
        return true;
    }
    return false;
}
```

## 【课堂实验 KT6.3】：数据库操作

【工程模块】：SQLiteSample

【工程描述】：该程序展示了如何使用 sqlite 数据库的基本方法。

【实验步骤】：

1、运行程序，观察现象。

2、分析理解代码

【知识点注释】：

Android 全面支持嵌入式数据库 SQLite。用户可以在自己的应用中创建数据库并访问，而其他应用不会访问到它。系统会将 app 的数据库文件存放在内部存储器中。

sql 数据库首先要定义其框架 **schema**，**schema** 被映射成 sql 语句来创建数据库。通常的做法是创建一个辅助类 **Contract**。该类定义数据库的静态结构。好的做法是全局变量定义在类的 **root** 级，每个表和其列用内部类定义。

通常还定义一个 **helper** 类继承自 **SQLiteOpenHelper**，执行数据库创建和版本变化后执行的动作，通过 **helper** 类就可以获得该数据实例。

```
MatchDbHelper(Context c, int version) {
    super(c, MatchDbContract.DB_NAME, null, version);
}
```

**Helper** 类的构造函数，当 **helper** 第一次运行时会创建数据库，该数据库文件通常在 `/data/data/your packagename/database/` 下。同时，回调函数 **onCreate** 也会被执行，在该函数中通常执行创建表的工作。**Version** 是版本号，若 **new helper** 是传入比上一次大的 **version**，则 **onUpgrade** 回调函数会被执行，通常会删除原来的表格，并重新创建新的表格。

```
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
    db.execSQL(SQL_DELETE_MATCHSTBL);
    db.execSQL(SQL_DELETE_PLAYERSTBL);
    onCreate(db);
}

@Override
public void onCreate(SQLiteDatabase db) {
    this.mDb = db;

    db.execSQL(CREATE_MATCHSTBL);
    db.execSQL(CREATE_PLAYERSTBL);
}
```

请注意增删改查等语法的实现，根据需要可以重载这些函数。

Sqlite 数据库可以通过 **cmd** 下的 **adb shell** 命令进行调试。方法如下：

将 **sdk** 的 **platform-tools** 目录放到系统 **path** 变量中，然后运行虚拟机和 **app**，打开 **CMD** 命令窗口，输入：

**Adb shell** 默认会进入到当前虚拟机的文件系统。

**cd /data/data/your package/database**

**ls** 查看数据库文件

**sqlite3 yourdatabase**

**sqlite3>.help**

查看帮助

**sqlite3>.table** 查看全部表

注意不要忘记 **table** 前的.

**sqlite3>select \* from xtable;**

注意不要忘记 **sql** 语句后的;

## 【课堂实验 KT6.4】：Content Provider

【工程模块】：ContentProviderSample

【工程描述】：展示了如何访问系统 **provider(Contact Provider)**，和自定义 **Content Provider**

的例子。

【实验步骤】请分别将设置为主 Activity,运行程序并测试

【知识点注释】

通过 Content Provider 获取系统的联系人数据。

注意,系统自带的 contact 和 calendar 两个 provider 的访问 api 和文档等随版本变化均有变化,请参考最新手册。

<http://developer.android.com/guide/topics/providers/calendar-provider.html>

<http://developer.android.com/guide/topics/providers/contacts-provider.html>

Contentproviders 用来管理对结构化数据集的链接访问,不仅封装了数据而且定义了数据访问安全的机制。其提供了跨进程访问数据的标准接口。

如果不打算共享数据,app 不需要提供 provider。但如果要提供对本 app 某种定制化的查询请求或者允许其他 app 拷贝自己 app 中复杂的数据或者文件的话就需要定义 provider。

和数据库对象不同的是,contentprovider 是为了共享而封装出的数据库对象。或者你可以这样理解,SQLiteDatabase 是 app 私有数据库的,而 provider 是公有数据库,是别的 app 可以访问的。要让 app 之外能访问到就需要有一个地址。这个地址被封装为 Uri 对象。其实质是一个固定格式的字符串,以“content: //”开头。

`content://com.example.transportationprovider/trains/122`

```
Uri singleUri = ContentUri.withAppendedId(UserDictionary.Words.  
CONTENT_URI, 4);
```

通常,构造一个 contentprovider 首先在本地需要先构造一个 SQLiteDatabase 作为数据持久化的数据源(原则上,provider 可以封装任何数据,但它封装后的行为效果类似于关系数据库),然后对它进行一个封装,这个封装的重点在于,提供一个外界可以访问的 Uri 地址和实现增删改查函数。

- 通常类似于 sqllitedatabase 的构造,也为 provider 构造一个 contract 类,定义其静态结构,包括 Uri。
- 在实现增删改查函数时,和 sqllitedatabase 不同的是,通常需要先检查 Uri 是否匹配。UriMatcher 类是用来做匹配检查的

```
sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);  
sUriMatcher.addURI(Diary.AUTHORITY, "diaries", DIARIES);  
sUriMatcher.addURI(Diary.AUTHORITY, "diaries/#", DIARY_ID);
```

ContentProvider 是四大构件之一,和 activity 一样,要使用它就必须要在 manifest 中声明。

```
<provider  
android:name="cn.edu.shu.cs.android.experiment05.data_contentprovider.Diary  
ContentProvider"  
    android:authorities="cn.edu.shu.cs.android.experiment05.data_contentpro  
vider" />
```

基本用法：

### 1、如何访问 ContentProvider

- 通过 `getContentResolver()` 得到 `ContentResolver` 对象，这是来操控访问 `Provider` 的 `client`。如果仅仅为了查询，在 `activity` 中可以使用 `managedQuery()`。`ContentResolver` 是访问操控 `Provider` 的客户端，其要求传递一个 `Uri` 参数，并且拥有如 `query()`, `insert()`, `delete()`, `update()` 等方法。这些方法实质是调用了 `Provider` 中实现的相应方法。
- 如果要访问非本 `app` 内定义的 `provider` 则要声明权限 `<uses-permission>`
- `provider` 通过一个或多个 `table` 的方式来组织数据呈现给外部访问者，因此查询数据方式近似于访问关系数据库，如下：

```
mCursor = getContentResolver().query(
    UserDictionary.Words.CONTENT_URI, // The content URI of the words table
    mProjection, // The columns to return for each row

    mSelectionClause, // Selection criteria

    mSelectionArgs, // Selection criteria
    mSortOrder
);
```

1. `Diary` 是一个 `contract` 类，定义了 `Uri`
2. `DiaryContentProvider` 实现了主要的增删改查函数
3. 本例中，将使用 `sqlitedatabase` 作为数据源，因此数据库的实现也包含在这个库中。而数据库的 `contract` 类则部分是现在 `Diary` 中。在实际开发中，开发者应该设计更好的类结构。

## 【课堂实验 KT6.5】:ListView 和 GridView 的用法

【工程模块】: `ListViewSample`

【工程描述】: 展示了 `ListView` 和 `GridView` 的基本用法。

【知识点注释】:

本例中使用了不同的 `adapter`，它们的数据源类型往往不同，请注意代码中注释掉的语句。

*//使用 simpleAdapter 的例子*

*//ListAdapter adapter=getSimpleAdapter();*

*//使用 ArrayAdapter 类的例子*

*//ListAdapter adapter=getArrayAdapter();*

*/\* 将 adapter 添加到 m\_ListView 中 \*/*

```
madapter=getCkAdapter();  
m_ListView.setAdapter(madapter);  
setListViewEvent();
```

## 【课后练习 KH6.1】:文件管理器

【功能要求】: ([@编程练习](#))参照有关 listview 和文件读写例子,编写一个小型的 file explore。可以根据给定的路径,用 listview 列出其下的文件,单击 add 按钮增加一个文件,选择文件可以删除或读取文件。请尝试为 Listview 定制一个 layout。

## 【课后实验 KH6.2】 比赛管理应用

【目标要求】: ([@编程练习](#))

完成一个用于比赛现场管理的 app, 首先建立一个比赛, 然后设置比赛小组数和每组的选手数。选择小组进行小组选手的录入。然后长按小组条目进入比赛结果记录界面记录选手对阵比分。这些比赛数据的存储采用 xml 或者文件或者数据库的方式均可。设置中可以预置默认的比分, 和小组人数, 设置信息使用 sharepreferenced 来保存。

【参考设计】

界面参考示意图如下, 在能完成功能情况下界面可以自行进行优化调整。



主界面

点击“新比赛”进入赛事管理界面



### 赛事管理

点击 条目右边图标进入该比赛的设计界面，设置比赛的小组数和小组人数



比赛设置

设置完成返回赛事管理界面，长按赛事条目进入该赛事分组人员管理界面





分组人员管理

点击小组条目右边图标，显示该小组人员，点击添加图标加入新选手。长按小组条目，进入比分记录界面



比分记录

在该界面进行比分登记。

提示：使用 Listview 和两个 listview 嵌套。

**【提交要求】：**1、实验报告：类极其作用的列表。运行结果的截图和功能说明。需要说明的问题。

3、请尝试设计一个比赛成绩列表显示界面，类似如下效果