[**https://developer.android.com/training/basics/data-storage/index.html**](https://developer.android.com/training/basics/data-storage/index.html) **保存数据**



1. 保存键值集

如果您有想要保存的相对较小键值集合，您应使用 [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html)

[SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html) 对象指向包含键值对的文件并提供读写这些文件的简单方法。 每个[SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html) 文件由框架进行管理并且可以专用或共享。

**注**：[SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html) API 仅用于读写键值对，您不得将其与 [Preference](https://developer.android.com/reference/android/preference/Preference.html) API 混淆，后者帮助您为您的应用设置构建用户界面（尽管它们使用 [SharedPreferences](https://developer.android.com/reference/android/content/SharedPreferences.html) 作为其实现以保存应用设置）。 有关使用 [Preference](https://developer.android.com/reference/android/preference/Preference.html) API 的信息，请参阅[设置](https://developer.android.com/guide/topics/ui/settings.html)指南。





1. 保存文件https://developer.android.com/training/basics/data-storage/files.html

Android 使用与其他平台上基于磁盘的文件系统类似的文件系统。 本课程讲述如何使用 Android 文件系统通过 [File](https://developer.android.com/reference/java/io/File.html) API 读取和写入文件。

[File](https://developer.android.com/reference/java/io/File.html) 对象适合按照**从开始到结束的顺序不跳过地读取或写入大量数据**。 例如，它适合于图片文件或通过网络交换的任何内容。



**提示**：尽管应用默认安装在内部存储中，但您可在您的清单文件中指定 [android:installLocation](https://developer.android.com/guide/topics/manifest/manifest-element.html#install) 属性，这样您的应用便可安装在在外部存储中。当 APK 非常大且它们的外部存储空间大于内部存储时，用户更青睐这个选择。 如需了解详细信息，请参阅[应用安装位置](https://developer.android.com/guide/topics/data/install-location.html)。

## 获取外部存储的权限

<manifest ...>  
    <uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE" />  
    ...  
</manifest>

**注意：**目前，所有应用都可以读取外部存储，而无需特别的权限。 但这在将来版本中会进行更改。如果您的应用需要读取外部存储（但不向其写入信息），那么您将需要声明 [READ\_EXTERNAL\_STORAGE](https://developer.android.com/reference/android/Manifest.permission.html#READ_EXTERNAL_STORAGE) 权限。要确保您的应用继续正常工作，您应在更改生效前声明此权限。

<manifest ...>  
    <uses-permission android:name="android.permission.READ\_EXTERNAL\_STORAGE" />  
    ...  
</manifest>

但是，如果您的应用使用 [WRITE\_EXTERNAL\_STORAGE](https://developer.android.com/reference/android/Manifest.permission.html#WRITE_EXTERNAL_STORAGE) 权限，那么它也隐含读取外部存储的权限。

在内部存储中保存文件时，您可以通过调用以下两种方法之一获取作为 [File](https://developer.android.com/reference/java/io/File.html) 的相应目录：

[getFilesDir()](https://developer.android.com/reference/android/content/Context.html#getFilesDir())

返回表示您的应用的内部目录的 [File](https://developer.android.com/reference/java/io/File.html) 。

[getCacheDir()](https://developer.android.com/reference/android/content/Context.html#getCacheDir())

返回表示您的应用临时缓存文件的内部目录的 [File](https://developer.android.com/reference/java/io/File.html)。 务必删除所有不再需要的文件并对在指定时间您使用的内存量实现合理大小限制，比如，1MB。 如果在系统即将耗尽存储，它会在不进行警告的情况下删除您的缓存文件。

**要在这些目录之一中新建文件，您可以使用**[**File()**](https://developer.android.com/reference/java/io/File.html#File(java.io.File, java.lang.String))**构造函数，传递指定您的内部存储目录的上述方法之一所提供的**[**File**](https://developer.android.com/reference/java/io/File.html)**。例如：**

File file = new File(context.getFilesDir(), filename);

或者，您可以调用 [openFileOutput()](https://developer.android.com/reference/android/content/Context.html#openFileOutput(java.lang.String, int)) 获取写入到内部目录中的文件的 [FileOutputStream](https://developer.android.com/reference/java/io/FileOutputStream.html)。例如，下面显示如何向文件写入一些文本：

String filename = "myfile";  
String string = "Hello world!";  
FileOutputStream outputStream;  
  
try {  
  outputStream = openFileOutput(filename, Context.MODE\_PRIVATE);  
  outputStream.write(string.getBytes());  
  outputStream.close();  
} catch (Exception e) {  
  e.printStackTrace();  
}

**记得每次读写文件都要加异常检测**

或者，如果您需要缓存某些文件，您应改用 [createTempFile()](https://developer.android.com/reference/java/io/File.html#createTempFile(java.lang.String, java.lang.String))。例如，以下方法从 [URL](https://developer.android.com/reference/java/net/URL.html) 提取文件名并正在您的应用的内部缓存目录中以该名称创建文件：

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;  
}

**注：**您的应用的内部存储设备目录由您的应用在 Android 文件系统特定位置中的软件包名称指定。从技术上讲，如果您将文件模式设置为可读，那么，另一应用也可以读取您的内部文件。 但是，此应用也需要知道您的应用的软件包名称和文件名。 其他应用无法浏览您的内部目录并且没有读写权限，除非您明确将文件设置为可读或可写。 只要您为内部存储上的文件使用 [MODE\_PRIVATE](https://developer.android.com/reference/android/content/Context.html#MODE_PRIVATE)，其他应用便从不会访问它们。

## 将文件保存在外部存储中

由于外部存储可能不可用—比如，当用户已将存储装载到电脑或已移除提供外部存储的 SD 卡时—因此，在访问它之前，您应始终确认其容量。 您可以通过调用 [getExternalStorageState()](https://developer.android.com/reference/android/os/Environment.html#getExternalStorageState()) 查询外部存储状态。 如果返回的状态为 [MEDIA\_MOUNTED](https://developer.android.com/reference/android/os/Environment.html#MEDIA_MOUNTED)，那么您可以对您的文件进行读写。 例如，以下方法对于确定存储可用性非常有用：

## 将文件保存在外部存储中

**尽管java.io定义的大多数类是实行流式操作的，File类不是。它直接处理文件和文件系统。也就是说，File类没有指定信息怎样从文件读取或向文件存储；它描述了文件本身的属性。**File对象用来获取或处理与磁盘文件相关的信息，例如权限，时间，日期和目录路径。此外，File还浏览子目录层次结构。 下面的构造函数可以用来生成File 对象：

**File(String directoryPath)**

**File(String directoryPath, String filename)**

**File(File dirObj, String filename)**

这里，directoryPath是文件的路径名，filename 是文件名，dirObj 一个指定目录的File 对象。下面的例子创建了三个文件：f1，f2，和f3。第一个File对象是由仅有一个目录路径参数的构造函数生成的。第二个对象有两个参数——路径和文件名。第三个File对象的参数包括指向f1文件的路径及文件名。f3和f2指向相同的文件。

File f1 = new File("/");

File f2 = new File("/","autoexec.bat");

File f3 = new File(f1,"autoexec.bat");

注意：Java 能正确处理UNIX和Windows/DOS约定路径分隔符。如果在Windows版本的Java下用斜线（/），路径处理依然正确。记住，如果你用Windows/DOS使用反斜线（/）的约定，你需要在字符串内使用它的转义序列（//）。Java约定是用UNIX和URL风格的斜线来作路径分隔符。

**总结一下吧，File的实例如果没有文件名，就是路径，有文件名，就是文件**

由于外部存储可能不可用—比如，当用户已将存储装载到电脑或已移除提供外部存储的 SD 卡时—因此，在访问它之前，您应始终确认其容量。 您可以通过调用 [getExternalStorageState()](https://developer.android.com/reference/android/os/Environment.html#getExternalStorageState()) 查询外部存储状态。 如果返回的状态为 [MEDIA\_MOUNTED](https://developer.android.com/reference/android/os/Environment.html#MEDIA_MOUNTED)，那么您可以对您的文件进行读写。 例如，以下方法对于确定存储可用性非常有用：

**/\* 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;  
}

公共文件

应供其他应用和用户自由使用的文件。 当用户卸载您的应用时，用户应仍可以使用这些文件。

例如，您的应用拍摄的照片或其他已下载的文件。

私有文件

属于您的应用且在用户卸载您的应用时应予删除的文件。 尽管这些文件在技术上可被用户和其他应用访问（因为它们存储在外部存储中）， 但它们实际上不向您的应用之外的用户提供任何输出值。 当用户卸载您的应用时，系统会删除应用外部私有目录中的所有文件。

如果您要将公共文件保存在外部存储设备上，请使用 [getExternalStoragePublicDirectory()](https://developer.android.com/reference/android/os/Environment.html#getExternalStoragePublicDirectory(java.lang.String)) 方法获取表示外部存储设备上相应目录的 [File](https://developer.android.com/reference/java/io/File.html)。 该方法使用指定您想要保存以便它们可以与其他公共文件在逻辑上组织在一起的文件类型的参数，比如 [DIRECTORY\_MUSIC](https://developer.android.com/reference/android/os/Environment.html#DIRECTORY_MUSIC) 或 [DIRECTORY\_PICTURES](https://developer.android.com/reference/android/os/Environment.html#DIRECTORY_PICTURES)。例如：

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;  
}

如果您要保存您的应用专用文件，您可以通过调用 [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html#getExternalFilesDir(java.lang.String)) 并向其传递指示您想要的目录类型的名称，从而获取相应的目录。通过这种方法创建的各个目录将添加至封装您的应用的所有外部存储文件的父目录，当用户卸载您的应用时，系统会删除这些文件。

例如，您可以使用以下方法来创建个人相册的目录：

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;  
}

如果没有适合您文件的预定义子目录名称，您可以改为调用 [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html#getExternalFilesDir(java.lang.String)) 并传递 null。这将返回外部存储上您的应用的专用目录的根目录。

切记，[getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html#getExternalFilesDir(java.lang.String)) 在用户卸载您的应用时删除的目录内创建目录。如果您正保存的文件应在用户卸载您的应用后仍然可用—比如，当您的应用是照相机并且用户要保留照片时—您应改用 [getExternalStoragePublicDirectory()](https://developer.android.com/reference/android/os/Environment.html#getExternalStoragePublicDirectory(java.lang.String))。

**无论您对于共享的文件使用**[**getExternalStoragePublicDirectory()**](https://developer.android.com/reference/android/os/Environment.html#getExternalStoragePublicDirectory(java.lang.String))**还是对您的应用专用文件使用**[**getExternalFilesDir()**](https://developer.android.com/reference/android/content/Context.html#getExternalFilesDir(java.lang.String))**，您使用诸如**[**DIRECTORY\_PICTURES**](https://developer.android.com/reference/android/os/Environment.html#DIRECTORY_PICTURES)**的 API 常数提供的目录名称非常重要。这些目录名称可确保系统正确处理文件。 例如，保存在**[**DIRECTORY\_RINGTONES**](https://developer.android.com/reference/android/os/Environment.html#DIRECTORY_RINGTONES)**中的文件由系统媒体扫描程序归类为铃声，而不是音乐**

## 查询可用空间

如果您事先知道您将保存的数据量，您可以查出是否有足够的可用空间，而无需调用 [getFreeSpace()](https://developer.android.com/reference/java/io/File.html#getFreeSpace()) 或 [getTotalSpace()](https://developer.android.com/reference/java/io/File.html#getTotalSpace()) 引起 [IOException](https://developer.android.com/reference/java/io/IOException.html)。这些方法分别提供目前的可用空间和存储卷中的总空间。 此信息也可用来避免填充存储卷以致超出特定阈值。

但是，系统并不保证您可以写入与 [getFreeSpace()](https://developer.android.com/reference/java/io/File.html#getFreeSpace()) 指示的一样多的字节。如果返回的数字比您要保存的数据大小大出几 MB，或如果文件系统所占空间不到 90%，则可安全继续操作。否则，您可能不应写入存储。

**注：**保存您的文件之前，您无需检查可用空间量。 您可以尝试立刻写入文件，然后在 [IOException](https://developer.android.com/reference/java/io/IOException.html) 出现时将其捕获。 如果您不知道所需的确切空间量，您可能需要这样做。 例如，如果在保存文件之前通过将 PNG 图像转换成 JPEG 更改了文件的编码，您事先将不知道文件的大小。

## 删除文件

您应始终删除不再需要的文件。删除文件最直接的方法是让打开的文件参考自行调用 [delete()](https://developer.android.com/reference/java/io/File.html#delete())。

myFile.delete();

如果文件保存在内部存储中，您还可以请求 [Context](https://developer.android.com/reference/android/content/Context.html) 通过调用 [deleteFile()](https://developer.android.com/reference/android/content/Context.html#deleteFile(java.lang.String)) 来定位和删除文件：

myContext.deleteFile(fileName);

**注**：当用户卸载您的应用时，Android 系统会删除以下各项：

* 您保存在内部存储中的所有文件
* 您使用 [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html#getExternalFilesDir(java.lang.String)) 保存在外部存储中的所有文件。

但是，您应手动删除使用 [getCacheDir()](https://developer.android.com/reference/android/content/Context.html#getCacheDir()) 定期创建的所有缓存文件并且定期删除不再需要的其他文件。

# 在 SQL 数据库中保存数据

**注**：通过实现 [BaseColumns](https://developer.android.com/reference/android/provider/BaseColumns.html) 接口，您的内部类可继承名为 \_ID 的主键字段，某些 Android 类（比如光标适配器）将需要内部类拥有该字段。 这并非必需项，但可帮助您的数据库与 Android 框架协调工作。

例如，该代码段定义了单个表格的表格名称和列名称：

public final class FeedReaderContract {  
    // To prevent someone from accidentally instantiating the contract class,  
    // make the constructor private.  
    private FeedReaderContract() {}  
  
    /\* Inner class that defines the table contents \*/  
    public static class FeedEntry implements BaseColumns {  
        public static final String TABLE\_NAME = "entry";  
        public static final String COLUMN\_NAME\_TITLE = "title";  
        public static final String COLUMN\_NAME\_SUBTITLE = "subtitle";  
    }  
}

## 使用 SQL 辅助工具创建数据库

在您定义了数据库的外观后，您应实现创建和维护数据库和表格的方法。 这里有一些典型的表格创建和删除语句：

private static final String TEXT\_TYPE = " TEXT";  
private static final String COMMA\_SEP = ",";  
private static final String SQL\_CREATE\_ENTRIES =  
    "CREATE TABLE " + FeedEntry.TABLE\_NAME + " (" +  
    FeedEntry.\_ID + " INTEGER PRIMARY KEY," +  
    FeedEntry.COLUMN\_NAME\_TITLE + TEXT\_TYPE + COMMA\_SEP +  
    FeedEntry.COLUMN\_NAME\_SUBTITLE + TEXT\_TYPE + " )";  
  
private static final String SQL\_DELETE\_ENTRIES =  
    "DROP TABLE IF EXISTS " + FeedEntry.TABLE\_NAME;

就像您在设备的[内部存储](https://developer.android.com/guide/topics/data/data-storage.html#filesInternal)中保存文件那样，Android 将您的数据库保存在私人磁盘空间，即关联的应用。 您的数据是安全的，因为在默认情况下，其他应用无法访问此区域。

[SQLiteOpenHelper](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html) 类中有一组有用的 API。当您使用此类获取对您数据库的引用时，系统将只在需要之时而不是应用启动过程中执行可能长期运行的操作：创建和更新数据库。 您仅需调用 [getWritableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#getWritableDatabase()) 或 [getReadableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#getReadableDatabase()) 即可。

**注**：由于它们可能长期运行，因此请确保您在后台线程中调用 [getWritableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#getWritableDatabase()) 或 [getReadableDatabase()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#getReadableDatabase())，比如使用 [AsyncTask](https://developer.android.com/reference/android/os/AsyncTask.html) 或 [IntentService](https://developer.android.com/reference/android/app/IntentService.html)。

要使用 [SQLiteOpenHelper](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html)，请创建一个替换 [onCreate()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#onCreate(android.database.sqlite.SQLiteDatabase))、[onUpgrade()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#onUpgrade(android.database.sqlite.SQLiteDatabase, int, int)) 和[onOpen()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#onOpen(android.database.sqlite.SQLiteDatabase)) 回调方法的子类。您可能还希望实现 [onDowngrade()](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html#onDowngrade(android.database.sqlite.SQLiteDatabase, int, int))，但这并非必需操作。

例如，下面是一个使用如上所示一些命令的 [SQLiteOpenHelper](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html) 的实现：

public class FeedReaderDbHelper extends SQLiteOpenHelper {  
    // If you change the database schema, you must increment the database version.  
    public static final int DATABASE\_VERSION = 1;  
    public static final String DATABASE\_NAME = "FeedReader.db";  
  
    public FeedReaderDbHelper(Context context) {  
        super(context, DATABASE\_NAME, null, DATABASE\_VERSION);  
    }  
    public void onCreate(SQLiteDatabase db) {  
        db.execSQL(SQL\_CREATE\_ENTRIES);  
    }  
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
        // This database is only a cache for online data, so its upgrade policy is  
        // to simply to discard the data and start over  
        db.execSQL(SQL\_DELETE\_ENTRIES);  
        onCreate(db);  
    }  
    public void onDowngrade(SQLiteDatabase db, int oldVersion, int newVersion) {  
        onUpgrade(db, oldVersion, newVersion);  
    }  
}

要访问您的数据库，请实例化 [SQLiteOpenHelper](https://developer.android.com/reference/android/database/sqlite/SQLiteOpenHelper.html) 的子类：

FeedReaderDbHelper mDbHelper = new FeedReaderDbHelper(getContext());

## 将信息输入到数据库

通过将一个 [ContentValues](https://developer.android.com/reference/android/content/ContentValues.html) 对象传递至 [insert()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html#insert(java.lang.String, java.lang.String, android.content.ContentValues)) 方法将数据插入数据库：

// Gets the data repository in write mode  
SQLiteDatabase db = mDbHelper.getWritableDatabase();  
  
// Create a new map of values, where column names are the keys  
ContentValues values = new ContentValues();  
values.put(FeedEntry.COLUMN\_NAME\_TITLE, title);  
values.put(FeedEntry.COLUMN\_NAME\_SUBTITLE, subtitle);  
  
// Insert the new row, returning the primary key value of the new row  
long newRowId = db.insert(FeedEntry.TABLE\_NAME, null, values);

[insert()](https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.html#insert(java.lang.String, java.lang.String, android.content.ContentValues)) 的第一个参数即为表格名称。

第二个参数将指示框架在 [ContentValues](https://developer.android.com/reference/android/content/ContentValues.html) 为空（即，您没有 [put](https://developer.android.com/reference/android/content/ContentValues.html#put(java.lang.String, byte[])) 任何值）时执行哪些操作。如果指定列名称，则框架将插入一行并将该列的值设置为 null。如果指定 null（就像此代码示例中一样），则框架不会在没有值时插入行。