第六章

数据存储全方案--详解持久化技术

主讲：王海

本章目标

- 了解Android数据存储方式
- 能够使用I/O流操作文件
- 能够读写SD卡文件
- 能够使用SharedPreferences存储
- 能够熟练使用SQLite进行数据的增删改查

数据存储

- Android提供了以下 三种数据存储方式：
  - ✓ 文件存储：少量数据需要保存，且数据格式无需结构化
  - ✓ SharedPreferences存储：数据是以“key-value”键值对的方式进行组织和管理，并保存到XML文件中
  - ✓ SQLite数据库存储：用于数据量较多，且需要进行结构化存储

I/O流操作文件

- 通过Context对象提供的openFileInput()和openFileOuput()两个方法分别来获得文件的输入流和输出流：
  - ✓ FileInputStream openFileInput(String name)：用于获取应用程序的数据文件夹下指定“name”文件名的标准文件输入流，以便读取设备中的文件
  - ✓ FileOutputStream openFileOuput(String name,int mode)：用于获取应用程序的数据文件夹下指定“name”文件名的标准文件输出流，以便将数据写入设备的文件中

I/O流操作文件

- Context类中提供4个静态常量用于表示不同的输出模式

模式	功能描述
Context.MODE_PRIVATE	私有模式所创建的文件都是私有文件，只能被应用本身所访问
Context.MODE_APPEND	附加模式首先会检查文件是否存在，若文件不存在，则创建新文件；若文件存在，则在原文件的末尾追加内容
Context.MODE_WORLD_READABLE	可读模式，该模式的文件可以被其他应用程序读取
Context.MODE_WORLD_WRITABLE	可写模式，该模式的文件可以被其他应用程序读写

从Android 4.2开始，不推荐使用Context.MODE_WORLD_WRITABLE可读模式和Context.MODE_WORLD_READABLE可写模式，由于这两种模式允许其他应用程序操作本应用程序所创建的文件数据，很容易会引起安全漏洞，因此在高版本的Android系统中尽量不要采用这两种模式。

I/O流操作文件

- Context上下文对象还提供了一些方法来访问应用程序的数据文件夹

方法	功能描述
File getDir(String name,int mode)	在应用程序的数据文件夹下获取或创建name对应的子目录
File getFilesDir()	获取应用程序的数据文件夹的绝对路径
String[] fileList()	返回应用程序的数据文件夹下的所有文件
boolean deleteFile(String name)	删除应用程序的数据文件夹下的指定文件

I/O流操作文件

- 获取文件输入流进行读取文件

```
//定义文件名
String file = "qst.txt";
//获取指定文件的文件输入流
FileInputStream fileInputStream = openFileInput(file);
//定义一个字节缓存数组
byte[] buffer=new byte[fileInputStream.available()];
//将数据读到缓存区
fileInputStream.read(buffer);
//关闭文件输入流
fileInputStream.close();
```

- 获取文件输出流进行写文件

```
//获取文件输出流，操作模式是私有
FileOutputStream fileOutputStream = openFileOutput(file,Context.MODE_PRIVATE);
String strContent = "QST青软实训";
//将内容写入文件
fileOutputStream.write(strContent.getBytes());
fileOutputStream.close();
```

读写SD卡步骤

① 使用Environment.getExternalStorageState()方法判断是否插入SD卡，且应用程序具有读写SD卡的权限

② 使用Environment.getExternalStorageDirectory()方法获取SD卡的目录

③ 使用文件输入流（FileInputStream、FileReader）或输出流（FileOutputStream、FileWriter）来读写SD卡中的文件

SD卡（Secure Digital Memory Card）是一种基于半导体快闪记忆器的多功能存储卡，具有大容量、高性能、安全高等多种特点，被广泛地用于便携式移动设备，例如手机、数码相机、PDA等。SD卡极大地扩充了手机的存储能力。

读SD卡上的文件

```
// 1、如果手机插入了SD卡，而且应用程序具有访问SD的权限
if (Environment.getExternalStorageState().equals(Environment.MEDIA_MOUNTED)){
    // 2、获取SD卡对应的存储目录
    File sdCardDir = Environment.getExternalStorageDirectory();
    Log.d("FileIO",""+ sdCardDir);
    //3、 获取指定文件对应的输入流
    FileInputStream fis = new FileInputStream(sdCardDir.getCanonicalPath()
        + FILE_NAME);
        ......//读文件
}
```

- Android应用程序读写SD卡中的文件时，需要注意以下两点：
  - ✓ 确保已插入SD卡
  - ✓ 在AndroidManifest.xml程序清单文件中配置SD卡的读写权限

配置SD卡的读写权限

```
<!-- 在SD卡中创建与删除文件权限 -->
<uses-permission android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS"/>
<!-- 向SD卡写入数据权限 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

注意

除了使用Environment.getExternalStorageDirectory()方法来获取SD卡的路径外，还可以直接判断SD卡所对应的路径是否存在，这样也可以知道手机是否插入了SD卡

文件浏览器

- 用于查看SD卡中的文件信息
- 使用ListView组件来显示指定目录中的全部文件和文件夹

SharedPreferences接口

- SharedPreferences常用方法

方法	功能描述
boolean contains (String key)	判断SharedPreferences是否包含指定key的数据
SharedPreferences.Editor edit()	返回SharedPreferences.Editor编辑对象
Map<String,?> getAll()	获取SharedPreferences中所有key-value对，返回值的类型为Map类型
xxx getXxx(String key,xxx defValue)	返回SharedPreferences中指定key的数据值

SharedPreferences.Editor接口

- SharedPreferences.Editor接口常用方法

方法	功能描述
SharedPreferences.Editor clear()	清除SharedPreferences中所有数据
SharedPreferences.Editor putXxx(String key,xxx value)	将指定key所对应的数据保存到SharedPreferences中
SharedPreferences.Editor remove(String key)	删除SharedPreferences中指定key所对应的数据
boolean commit()	当Editor编辑完成后，使用该方法提交内容，以便数据保存到SharedPreferences中

注意

SharedPreferences和SharedPreferences.Editor需要组合使用，SharedPreferences负责读取数据，而SharedPreferences.Editor负责保存数据。

SharedPreferences.Editor接口

- getSharedPreferences(String name,int mode)方法的参数说
  - 参数name用于指定存储数据的XML文件名，该文件名无须后缀（.xml），系统会自动添加.xml后缀，并在“/data/data/包名/shared_prefs/”目录中创建该文件
  - 参数mode用于设定文件的操作模式，取值可以是三种：
    - ✓ sanzhContext.MODE_WORLD_READABLE（可读）
    - ✓ Context.MODE_WORLD_WRITEABLE（可写）
    - ✓ Context.MODE_PRIVATE(私有)

注意

从Android 4.2开始不再推荐MODE_WORLD_READABLE（可读）和MODE_WORLD_WRITEABLE（可写）这两种模式

SharedPreferences操作步骤

① 使用getSharedPreferences()方法获取一个SharedPreferences实例对象

② 使用SharedPreferences实例对象的edit()方法，获取SharedPreferences.Editor编辑对象

③ 使用SharedPreferences.Editorr编辑对象的putXxx()方法来保存数据

④ 使用SharedPreferences.Editor编辑对象的commit()方法将数据提交到XML文件中

⑤ 使用SharedPreferences对象的getXxx()方法来读取数据

SQLite特征

- 轻量级
- 独立
- 操作简单
- 便于管理和维护
- 可移植性
- 语言无关
- 事务性

SQLiteDatabase类

- 打开数据库的方法：
  - openDatabase(String path, SQLiteDatabase.CursorFactory factory, int flags)：打开path所指定的SQLite数据库
  - openOrCreateDatabase(String path,SQLiteDatabase.CursorFactory factory)：打开或创建（如果文件不存在）path所指定的SQLite数据库
  - openOrCreateDatabase(File file,SQLiteDatabase.CursorFactory factory)：打开或创建（如果文件不存在）file所指定的SQLite数据库

SQLiteDatabase常用操作方法

方法	功能描述
insert(String table,String nullColumnHack,ContentValues values)	插入一条记录
delete(String table,String whereClause,String[] whereArgs)	删除一条记录
query (boolean distinct, String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy, String limit)	查询记录
update(String table,ContentValues value,String whereClause, String[] whereArgs)	修改记录
execSQL(String sql)	执行一条SQL语句
rawQuery(String sql,String[] selectionArgs)	执行带占位符的SQL查询
beginTransaction()	开始事务
endTransaction()	结束事务
close()	关闭数据库

SQLite数据库的创建和删除

- **创建或打开数据库**
  - 使用openDatabase()方法打开指定的数据库时，需要三个参数：
    - ✓ path用于指定数据库的路径，若指定的数据库不存在，则抛出FileNotFoundException异常
    - ✓ factory用于构造查询时的游标，若factory为null，则表示使用默认的factory构造游标
    - ✓ flags指定了数据库打开的模式，SQLite定义了4种数据库打开模式：
      - OPEN_READONLY（只读）
      - OPEN_READWRITE（可读可写）
      - CREATE_IF_NECESSARY（若数据库不存在先创建数据库）
      - NO_LOCALIZED_COLLATORS（不按照本地化语言对数据进行排序）

- 使用openOrCreateDatabase()方法打开或创建指定的数据库

```
SQLiteDatabase sqliteDatabase = SQLiteDatabase
        .openOrCreateDatabase ("qst_Student.db", null);
```

- 使用openDatabase()方法打开指定的数据库

```
SQLiteDatabase sqliteDatabase = SQLiteDatabase
        .openDatabase("qst_Student.db", null, NO_LOCALIZED_COLLATORS);
```

SQLite数据库的创建和删除

- 删除数据库
  - 使用deleteDatabase()方法删除数据库

```
deleteDatabase("qst_Student.db");   //删除数据库qst_Student.db
```

- 关闭数据库
  - 使用close()方法关闭数据库

```
sqliteDatabase.close(); //关闭数据库，sqliteDatabase是一个实例对象
```

表的创建和删除

- 创建表
  - 使用execSQL()方法创建表

```
//创建表的SQL语句
String sql= "CREATE TABLE student(ID INTEGER PRIMARY KEY,
age INTEGER,name TEXT)";
//执行该SQL语句创建表
sqliteDatabase.execSQL(sql);
```

- 删除表
  - 使用execSQL()方法删除表

```
//创建表的SQL语句
String sql= "CREATE TABLE student(ID INTEGER PRIMARY KEY,
age INTEGER,name TEXT)";
//执行该SQL语句创建表
sqliteDatabase.execSQL(sql);
```

记录的插入、修改和删除

- 插入记录
  - 【语法】`insert(String table,String nullColumnHack,ContentValues values)`
    - 第1个参数table是需要插入数据的表名称
    - 第2个参数nullColumnHack是空列的默认值
    - 第3个参数values是ContentValues类型的对象，用于封装列名和列值的Map集合，代表一条记录信息
  - 使用insert()方法插入记录

```
ContentValues contentValues = new ContentValues();
contentValues.put("ID", 1);
contentValues.put("age", 26);
contentValues.put("name", "StudentA");
//调用insert()方法将contentValues对象封装的数据插入到student表中
sqliteDatabase.insert("student" , null, contentValues);
```

SQLite数据库的创建和删除

使用execSQL()方法插入记录

```
//定义插入SQL语句
String sql= "INSERT INTO student (ID,age,name) values (1, 26,'StudentA')";
//调用execSQL()方法执行SQL语句，将数据插入到student表中
sqliteDatabase.execSQL(sql);
```

插入记录

- **【语法】**

```
delete(String table,String whereClause,String[] whereArgs)
```

  - **第1个参数table是需要删除数据的表名称**
  - **第2个参数whereClause是删除条件**
  - **第3个参数whereArgs是删除条件所需的参数数组**

SQLite数据库的创建和删除

- 使用delete()方法删除记录

```
sqliteDatabase.delete("student","name=?",new
String[]{"StudentA"});
```

- 使用execSQL()方法删除记录

```
//定义更新SQL语句
String sql= "DELETE FORM student where name='StudentA'";
//调用execSQL()方法执行SQL语句更新student表中的记录
sqliteDatabase.execSQL(sql);
```

数据查询与Cursor接口

- 使用SQLiteDatabase的query()方法可以查询记录
  - 【语法】

```
public Cursor query (boolean distinct, String table, String[] columns,
    String selection, String[] selectionArgs, String groupBy, String having,
    String orderBy, String limit);
```

- ✓ distinct是一个可选的布尔值，用来说明返回的值是否只包含唯一的值
- ✓ table是表名称
- ✓ columns是由列名称构成的数组
- ✓ selection是条件where子句，可以包含“?”通配符，在子句中用作占位符
- ✓ selectionArgs是参数数组，替换where子句中的“?”占位符
- ✓ groupBy表示分组列
- ✓ having是分组条件
- ✓ orderBy是排序列
- ✓ limit是一个可选的字符串，用来对返回的行数进行限制

Cursor游标常用方法

方法	功能描述
move(int offset)	以当前的位置为基准，将Cursor移动到偏移量为offset的位置
moveToPosition(int position)	将Cursor移动到绝对位置position处
moveToNext()	将Cursor向前移动一个位置
moveToLast()	将Cursor移动到最后一条记录
moveToFisrt()	将Cursor移动到第一条记录
isBeforeFirst()	判断Cursor是否指向第一项数据之前
isAfterLast()	判断Cursor是否指向最后一项数据之后
isClosed()	判断Cursor是否关闭
isFirst()	判断Cursor是否指向第一项记录
isLast()	判断Cursor是否指向最后一条记录
isNull(int columnIndex)	判断指定的位置columnIndex的记录是否存在
getCount()	获取当前表的行数（即记录总数）
getInt(int columnIndex)	获取指定列索引的int类型值
getString(int columnIndex)	获取指定列索引的String类型值

使用query()方法查询记录

```
//创建隐式Intent来启动新的Activity
Cursor cursor=sqliteDatabase.query(true, "student", null, "name=StudentA",
                                                          null,null,
null,null,null);
//将游标移动到第一条记录，并判断
if(cursor.moveToFirst()){
    //获得列信息
     int id=cursor.getInt(0);
     int age=cursor.getInt(1);
     String name=cursor.getString(3);
     //输出
     Log.debug("id+":"+age+":"+name ");
}
```

事务处理

- SQLiteDatabase提供以下几个方法来控制事务：
  - beginTransaction()方法用于开始事务
  - endTransaction()方法用于结束事务
  - inTransaction()方法用于判断当前上下文是否处于事务环境中
  - setTransactionSuccessful()方法来设置事务成功标志
- 事务处理过程

```
//开始事务
sqliteDatabase.beginTransaction();
try{
    ......//执行DML语句
    //调用setTransactionSuccessful()方法设置事务成功，
    //否则endTransaction()方法将回滚事务
    sqliteDatabase.setTransactionSuccessful();
}finally{
    //由事务标志决定是提交事务，还是回滚事务
    sqliteDatabase.endTransaction();
}
```

SQLiteOpenHelper类

方法	功能描述
SQLiteOpenHelper(Context context,String name, SQLiteDatabase.CursorFactory,int version)	构造函数，第二个参数是数据库名称
onCreate(SQLiteDatabase db)	创建数据库时调用
onUpgrade(SQLiteDatabase db,int oldVersion,int newVersion)	版本更新时调用
getReadableDatabase()	创建或打开一个只读数据库
getWritableDatabase()	创建或打开一个读写数据库

本章总结

- Android提供了多种数据存储方式，包括：文件存储、SharedPreferences和SQLite
- 文件存储方式不受类型限制，可以将一些数据直接以文件的形式保存在设备中
- Android支持使用I/O流方式来访问手机等移动设备上存储的文件
- 在Android应用程序中，可以通过Context上下文环境提供的openFileInput()和openFileOuput()方法分别来获得文件的输入流和输出流
- SD卡是一种基于半导体快闪记忆器的多功能存储卡，扩充了手机的存储能力
- SharedPreferences保存的数据都以“key-value”键值对的方式存储在XML文件中
- 使用SharedPreferences中的getXxx()方法获取数据，使用SharedPreferences.Editor的putXxx()方法保存数据
- SQLite是一种免费开源、支持很多语言的数据库
- SQLiteDatabase代表一个数据库对象，提供了操作数据库的一些方法
- SQLiteDatabase的query()方法的返回值是一个Cursor游标对象，可以查询记录
- SQLiteOpenHelper是SQLiteDatabase的一个帮助类，用来管理数据库的创建和版本更新
- 使用ListView控件可以实现滑动分页

本章总结

- 使用SharedPreferences中的getXxx()方法获取数据，使用SharedPreferences.Editor的putXxx()方法保存数据
- SQLite是一种免费开源、支持很多语言的数据库
- SQLiteDatabase代表一个数据库对象，提供了操作数据库的一些方法
- SQLiteDatabase的query()方法的返回值是一个Cursor游标对象，可以查询记录
- SQLiteOpenHelper是SQLiteDatabase的一个帮助类，用来管理数据库的创建和版本更新
- 使用ListView控件可以实现滑动分页