河北师范大学软件学院
Software College of Hebei Normal University

Android 基础开发

第八章 第二讲
Android的SQLite

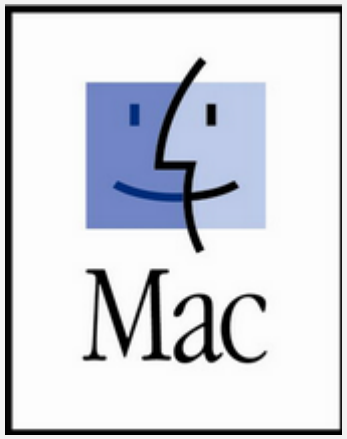
教学目标

- 掌握Android中SQLite数据库的使用

目录

SQLite的简介

- SQLite，是一款轻型的数据库，它的设计目标是嵌入式产品，它占用资源非常的低，在嵌入式设备中，可能只需要几百K的内存就够了。



SQLite的简介

- SQLite也有对应的操作工具：
  - Android SDK的platform-tools下sqlite3.exe。
  - Sqlite Developer。
  - SQLite Expert。
  - Sqliteadmin Administrator：可创建、设计和管理SQLite 数据库文件。

目录

SQLite的操作

- Android中使用SQLiteDatabase代表数据库，并且提供一些列的静态方法来操作数据：

方法名称	方法表示含义
openOrCreateDatabase(String path, SQLiteDatabase.CursorFactory factory)	打开或创建数据库
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(String table, String[] columns, String selection, String[] selectionArgs, String groupBy, String having, String orderBy)	查询一条记录
update(String table, ContentValues values, String whereClause, String[] whereArgs)	修改记录
execSQL(String sql)	执行一条SQL语句
close()	关闭数据库

打开或者创建数据库

- 使用openOrCreateDatabase()打开或者创建一个数据库，如果存在则打开，不存在则创建一个数据库；创建成功则返回SQLiteDatabase对象，否则抛出异常FileNotFoundException。

```
SQLiteDatabase db = SQLiteDatabase
        .openOrCreateDatabase("/data/data/com.edu2act.db/databases/stu.db", null);
// 如果路径不存在会会抛出异常Could not open database
```

  - 参数一：数据库路径。
  - 参数二：游标，一般使用null。

创建表

- 表的创建可通过调用SQLiteDatabase.execSQL()方法来执行SQL建表语句。

```
// 创建表SQL语句
String stu_table ="CREATE TABLE COMPANY("
         + "ID INTEGER PRIMARY KEY AUTOINCREMENT,"
         + "NAME TEXT NOT NULL, AGE INT NOT NULL,"
         + "ADDRESS CHAR(50),   SALARY REAL)";
// 执行SQL语句
db.execSQL(stu_table);
```

插入数据

- 调用SQLiteDatabase.insert()方法完成数据插入。

```
ContentValues cValue = new ContentValues();
cValue.put("NAME", "XiaoMing");
cValue.put("AGE", 19);
cValue.put("ADDRESS", "Hebei Normal University");
cValue.put("SALARY", 1500.0);
db.insert("COMPANY", null, cValue);
```

删除数据

- 调用SQLiteDatabase. delete()方法完成数据删除。

```
// 删除条件
String whereClause = "id=?";
// 删除条件参数
String[] whereArgs = {String.valueOf(2)};
// 执行删除
db.delete("COMPANY", whereClause, whereArgs);
```

  - 参数一：表名称。
  - 参数二：删除条件。
  - 参数三：删除条件值数组。

修改数据

- 调用SQLiteDatabase. update()方法完成数据删除。

```
ContentValues values = new ContentValues();
values.put("SALARY", 2000.0);
// 修改条件
String whereClause = "ID=?";
// 修改添加参数
String[] whereArgs={String.valueOf(1)};
db.update("COMPANY", values, whereClause, whereArgs);
```

  - 参数一：表名称。
  - 参数二：ContentValues类型的键值对Key-Value。
  - 参数三：更新条件（where字句）。
  - 参数四：更新条件数组。

通过SQL语句插入、删除、修改数据

- 通过SQL插入语句完成数据插入、删除、修改操作。

```
String sql = "INSERT INTO COMPANY(NAME, AGE,"
          + "ADDRESS, SALARY) VALUES ('XiaoLi', "
          + "20, 'Hebei Normal University', 1400.0)";
db.execSQL(sql);
```

查询数据

- 在Android中查询数据是通过Cursor类来实现的，当我们使用SQLiteDatabase.query()方法时，会得到一个Cursor对象，Cursor指向的就是每一条数据。

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

– table : 表名称
– columns : 列名称数组
– selection : 条件字句，相当于where
– selectionArgs : 条件字句，参数数组
– groupBy : 分组列
– having : 分组条件
– orderBy : 排序列
– limit : 分页查询限制

查询数据

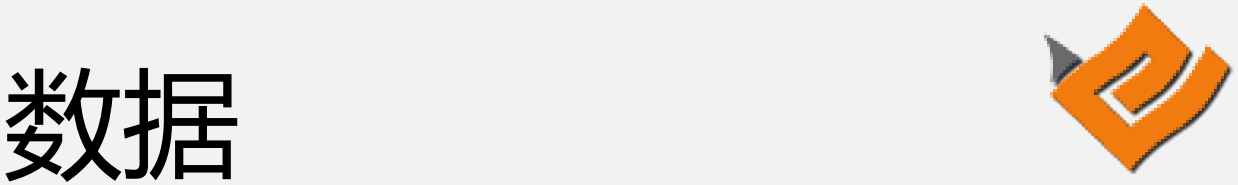
- Cursor游标常用方法：

方法名称	方法描述
getCount()	获得总的数据项数
isFirst()	判断是否第一条记录
isLast()	判断是否最后一条记录
moveToFirst()	移动到第一条记录
moveToLast()	移动到最后一条记录
move(int offset)	移动到指定记录
moveToNext()	移动到下一条记录
moveToPrevious()	移动到上一条记录
getColumnIndexOrThrow(String columnName)	根据列名称获得列索引
getInt(int columnIndex)	获得指定列索引的int类型值
getString(int columnIndex)	获得指定列缩影的String类型值

查询数据

```
// 查询获得游标
Cursor cursor = db.query ("COMPANY",null,null,null,null,null,null);
// 判断游标是否为空
if(cursor.moveToFirst()) {
  // 遍历游标
  do {
    int id = cursor.getInt(cursor.getColumnIndex("ID"));
    String name=cursor.getString(cursor.getColumnIndex("NAME"));
    int age=cursor.getInt(cursor.getColumnIndex("AGE"));
    String address = cursor.getString(cursor.getColumnIndex("ADDRESS"));
    String salary = cursor.getString(cursor.getColumnIndex("SALARY"));
    Log.e("mydb", id + "|"+name+"|"+age+"|"
                  +address+"|"+salary);
  } while (cursor.moveToNext());
}
```

查询数据



- 同样可以通过SQL查询语句完成数据的查询。

```
Cursor cursor
     = db.rawQuery("SELECT * FROM COMPANY", null);
```

SQLite中的数据库操作

- 常见的操作SQLite数据库的方法
  - 执行SQL语句方法的优缺点：
    - 适合了解SQL语言的人使用。
    - 可以进行所有数据库操作，不可进行查询操作。
    - execSQL没有返回值，不能进行结果的判断。
  - 特殊方法操作数据库的优缺点：
    - 适合初学SQL的人使用。
    - 可以进行基本所有操作，除了创建表操作。
    - 可以通过返回值得到操作的结果。

SQLiteDatabase操作步骤

- 获取SQLiteDatabase对象，它代表与数据库的连接。
- 调用SQLiteDatabase的方法来执行SQL语句。
- 操作SQL语句的执行结果。
- 关闭SQLiteDatabase，回收资源。

目录

SQLite中事务的处理

- 数据库事务(Database Transaction)，是指作为单个逻辑工作单元执行的一系列操作。 事务处理可以确保除非事务性单元内的所有操作都成功完成，否则不会永久更新面向数据的资源。通过将一组相关操作组合为一个要么全部成功要么全部失败的单元，可以简化错误恢复并使应用程序更加可靠。一个逻辑工作单元要成为事务，必须满足所谓的ACID(原子性、一致性、隔离性和持久性)属性。

SQLite中事务的处理

- SQLiteDatabase采用如下方法来控制事务：
  - beginTransaction（ ）
  - endTransaction（ ）
  - inTransaction（ ）
  - setTransactionSuccessful（ ）

```
db.beginTransaction();
try {
      // 执行事务中的很多sql语句
      db.setTransactionSuccessful();
} finally {
      db.endTransaction();
}
```

目录

SQLiteOpenHelper类

- SQLiteDatabase中判断表是否存在的话使用try-catch方式判断是否有异常。
- Android提供SQLiteOpenHelper类来管理数据库。主要负责数据库的创建、版本更新，一般情况通过创建它的子类并扩展onCreate()和onUpgrade()方法来实现。

SQLiteOpenHelper类

- SQLiteOpenHelper类常用方法：
  - SQLiteDatabase getReadableDatabase()。
  - SQLiteDatabase getWritableDatabase()。
  - abstract void onCreate()：第一次创建数据库的回调。
  - abstract void onUpgrade()：数据库版本更新的回调。
  - void close()：关闭所有打开的SQLiteDatabase。

SQLiteOpenHelper类

- 这样的话大家使用SQLiteOpenHelper来操作数据，就无需使用SQLiteDatabase的静态方法创建数据库实例了。
- 注意：
  - 用getReadableDatabase， getWriteableDatabase打开数据库时，如果数据库的磁盘空间满了就会打开失败。

课程回顾

1. SQLite的简介
2. SQLite的操作
3. SQLite中事务的处理
4. SQLiteOpenHelper类

Thank you!