Android 数据交互

本章概要

- file 偏好配置文件
- sqlite数据库与操作

清華大學 Tsinghua University

App偏好设置

保存偏好设置

- 持久性数据（Persisting Data）是应用程序执行时需要长期保存的一些数据，对于Android App是非常重要的部分，一般来说，我们可以使用 **SharedPreference 对象、文件或者 SQLite 数据库**来保存持久性数据

- 在 Android 中，提供的 SharedPreferences 对象来保存一些简单的应用程序数据，主要是App配置（字体大小，主题等）、或者一些简单的用户数据，它是使用XML格式的偏好配置文件来保存这些数据的

- 类似Windows目录下的ini文件 （例如，虚拟机的配置文件）

获取偏好设置

◆ 取得 SharedPreference 对象

我们可以使用继承自 Context 类的 getSharedPreferences() 方法来取得 SharedPreferences 对象

```
private SharedPreference prefs;
…
prefs = getSharedPreference("MyPref", MODE_PRIVATE);
```

方法的第1个参数是偏好配置文件的名称，第2个参数MODE_PRIVATE表示只允许建立偏好配置文件的App访问该文件（权限控制）。

然后，在活动中，我们可以直接使用prefs这个对象来访问偏好设置值。

MODE_APPEND：模式会检查文件是否存在，存在就往文件追加内容，否则就创建新文件
MODE_WORLD_READABLE：表示当前文件可以被其他应用读取；
MODE_WORLD_WRITEABLE：表示当前文件可以被其他应用写入

读取偏好设置

在Activity活动读取偏好设置，建议在onResume()这个方法里进行读取。可以使用SharedPreferences 的 getString()、getFloat()、getInt() 方法来取得之前保存的配置

```
String name = prefs.getString("name", "Harry");
```

方法的第1个参数是配置项的名称（可以理解为HashMap中的键值Key），第2个参数为默认值，表示当配置文件中没有这一项的时候，函数返回的值。

保存偏好设置

在Activity活动中需要保存偏好设置时，一般建议在onPause()方法里进行保存。我们可以使用SharedPreferences.Editor 对象来编辑存入的数据

```
SharedPreferences.Editor prefEditor = prefs.edit();
```

然后我们可以使用 putString()、putInt() 和 putFloat() 等方法存入字符串、整数和浮点数等数据

```
prefEditor.putString("name", "Tony")
```

方法的第1个参数是配置项的名称，第2个参数配置项的数据值。在编辑完成之后，记得使用commit() 方法写入到文件中。

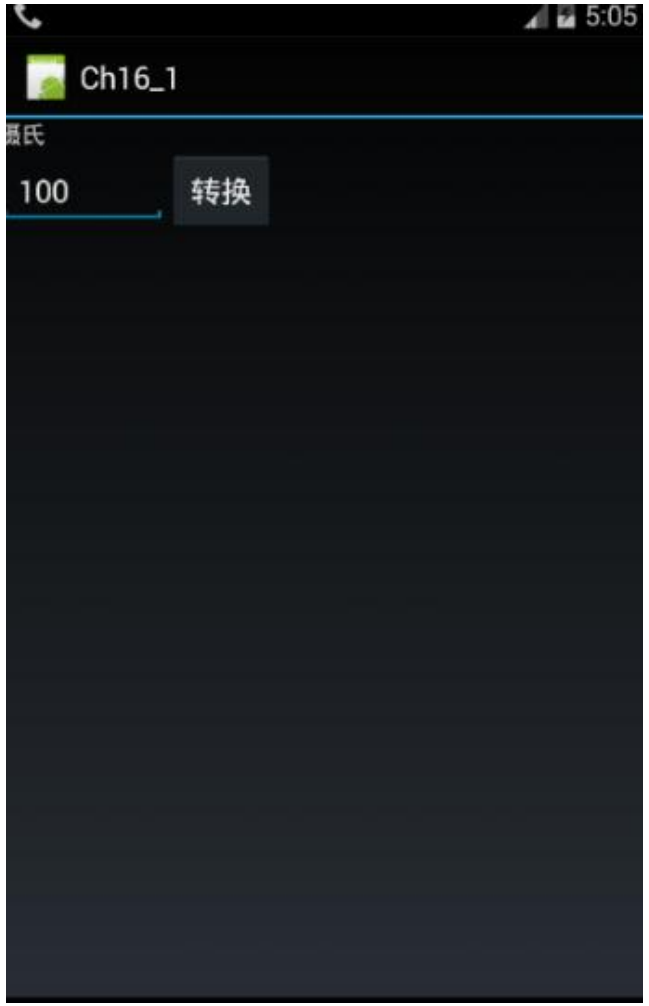
```
prefEditor.commit()
```

示例

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    // 取得SharedPreferences对象
    prefs = getSharedPreferences("MyPref", MODE_PRIVATE);
    // 取得EditText控件
    txtC = (EditText) findViewById(R.id.txtC);
}
@Override
protected void onResume() {
    super.onResume();
    // 取得偏好设置数据
    String tempc = prefs.getString("TEMPC" , "100");
    txtC.setText(tempc);
}
@Override
protected void onPause() {
    super.onPause();
    // 取得Editor对象
    SharedPreferences.Editor prefEdit = prefs.edit();
    // 存入偏好设置数据到Editor对象
    prefEdit.putString("TEMPC", txtC.getText().toString());
    prefEdit.commit(); // 确认写入文件镝
}
```

课堂练习(1)

第一次启动



第二次启动

清華大學 Tsinghua University

SQLite 数据库

SQLite 数据库与 SQL 语言

- SQLite 是目前世界上最广泛使用的免费数据库引擎。
  http://www.sqlite.org

  - Android系统中内置的数据库。
  - 一种轻量级关系型数据库。
  - 占用用资源小，大概只需几百K内存
  - 特别适合在嵌入式系统中使用
  - 支持标准的SQL语法，支持事务机制
  - Android专门提供了SQLiteOpenHelper类，对数据库进行行操作。

SQL 数据库语言

“SQL” (Structured Query Language) 为 ANSI 标准的数据库语言，可以访问和修改数据库的记录数据

◆ **SQL 语言的 DDL 指令**

1. CREATE TABLE 建立数据表

```
CREATE TABLE student (
  _id integer primary key,
  name text no null,
  age integer no null,
  score integer no null
)
```

建立了一个名为student的数据表，拥有4个字段 _id name age score，其中not null表示该字段不能为空；primary key 为主键，主键不可重复

2. DROP TABLE 删除数据表

```
DROP TABLE student
```

可以使用IF EXISTS 表示当存在数据表时才删除

```
DROP TABLE IF EXISTS student
```

_id	name	age	score
1	张三	23	80
2	李四	23	85

SQL 数据库语言

◆ SQL 语言的 DML 指令

DML 是数据表记录查询、插入、删除和修改相关的SQL指令。本节主要说明SELECT指令，它的基本语法为

```
SELECT 字段1，字段2 FROM 数据表
WHERE conditions
ORDER BY 字段列表
```

上述SELECT指令的字段1~2为记录的字段，conditions为查询条件，是用文字描述就是“从数据表取回符合WHERE子句条件的记录，显示字段1和2，并且以ORDER BY子句的字段来排序”

SQL 数据库语言

◆ SQL 语言的 DML 指令

例子：

```
SELECT * FROM student
```

```
SELECT * FROM student WHERE name='Joe Chen'
```

```
SELECT * FROM student WHERE score=90
```

```
SELECT * FROM student WHERE name LIKE '%Chen%'
```

```
SELECT name, age FROM student
```

```
SELECT name FROM student WHERE name='Joe Chen'
```

SQLite命令行

进入shell，输入sqlite3，出现
sqlite>

```
E:\Android>adb shell
root@generic_x86:/ # sqlite3
SQLite version 3.7.11 2012-03-20 11:35:50
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .help
.backup ?DB? FILE      Backup DB (default "main") to FILE
.bail ON|OFF           Stop after hitting an error.  Default OFF
.databases             List names and files of attached databases
.dump ?TABLE? ...      Dump the database in an SQL text format
                         If TABLE specified, only dump tables matching
                         LIKE pattern TABLE.
.echo ON|OFF           Turn command echo on or off
.exit                  Exit this program
.explain ?ON|OFF?      Turn output mode suitable for EXPLAIN on or off.
                         With no args, it turns EXPLAIN on.
.header(s) ON|OFF      Turn display of headers on or off
.help                  Show this message
.import FILE TABLE     Import data from FILE into TABLE
.indices ?TABLE?       Show names of all indices
                         If TABLE specified, only show indices for tables
                         matching LIKE pattern TABLE.
.log FILE|off          Turn logging on or off.  FILE can be stderr/stdout
.mode MODE ?TABLE?     Set output mode where MODE is one of:
                         csv      Comma-separated values
                         column   Left-aligned columns.  (See .width)
                         html     HTML <table> code
                         insert   SQL insert statements for TABLE
```

SQLite命令行

在shell中输入 sqlite3 student 打开/创建student数据库
（创建数据文件时需要注意当前目录是否有写文件的权限）
在sqlite>中输入
.databases （注意有一个.） 显示所有的数据库
.tables 查看当前数据库下的所有表
.schema [table_name] 查看表结构（主要看列信息）
.drop table [table_name] 删除表
.dump > filename.sql 导出数据到数据

```
root@generic_x86:/sdcard # sqlite3  student
SQLite version 3.7.11 2012-03-20 11:35:50
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .databases
seq  name             file
---  ---------------  ----------------------------------------------------
0    main             /storage/sdcard/student
sqlite> create table student(id integer,name text,age integer,score integer);
sqlite> .tables
student
sqlite> .schema
CREATE TABLE student(id integer,name text,age integer,score integer);
sqlite>
```

SQLite命令行

```
sqlite> insert into student values(1001,"zhangsan",23,80);
sqlite> insert into student values(1002,"lisi",23,85);
sqlite> insert into student values(1003,"wangwu",24,82);
sqlite> select * from student;
1001|zhangsan|23|80
1002|lisi|23|85
1003|wangwu|24|82
sqlite> select  idx,age from student;
1001|23
1002|23
1003|56
sqlite> .header on
sqlite> select * from student;
idx|name|age
1001|zhangsan|23
1002|lisi|23
1003|wangwu|56
sqlite> .mode column
sqlite> select * from student;
idx         name        age
----------  ----------  ----------
1001        zhangsan    23
1002        lisi        23
1003        wangwu      56
sqlite>
```

insert into <table_name> values (value1, value2,…);
向表中添加新记录
select filed1,field2··· from <table_name>;
查询表中某些字段的记录。
select * from <table_name>;
查询表中所有字段的记录
.header on 显示字段名称
.mode column 以列模式显示字段的记录。默认是list模式

SQLite命令行

```
select * from <table_name> order by field desc|asc;
```
可以用于表达排序，desc 表示降序，asc表示升序
```
update <table_name> set <field1=value1>,<fied2=value2>··where <expression>;
```
更新表中记录。若没有where子句，则会全部修改。
```
delete from <table_name> where <expression>;
```
删除表中记录。若没有where子句，则会删全表内容，但不同于drop。

```
sqlite> select * from student;
1001        zhangsan    23          80
1002        lisi        23          85
1003        wangwu      24          82
sqlite> select * from student order by score desc;
1002        lisi        23          85
1003        wangwu      24          82
1001        zhangsan    23          80
sqlite> select * from student order by score asc;
1001        zhangsan    23          80
1003        wangwu      24          82
1002        lisi        23          85
```

```
sqlite> select * from student order by score asc;
1001        zhangsan    23          80
1003        wangwu      24          82
1002        lisi        23          85
sqlite> update student set name="test" where id=1001;
sqlite> select * from student order by score asc;
1001        test        23          80
1003        wangwu      24          82
1002        lisi        23          85
sqlite> delete from student where id=1001;
sqlite> select * from student order by score asc;
1003        wangwu      24          82
1002        lisi        23          85
```

使用 SQLite 数据库

Android 应用程序通过 SQLiteOpenHelper 和 SQLiteDatabase 类来建立和访问数据库的记录数据，这两个类位于 android.database.sqlite 包下

◆ 使用 SQLiteOpenHelper 建立数据库

SQLiteOpenHelper是一个帮助我们访问SQLite数据库的帮助类(Helper Class)，我们需要继承此类，通过继承类进行SQLite数据表的建立。

```
public class StdDBHelper extends SQLiteOpenHelper{
  private static final String DATABASE_NAME = “Class”;
  private static final int DATABASE_VERSION = 1;
  public StdDBHelper(Context context){
    super(context, DATABASE_NAME, null, DATABASE_VERSION);
  }
```

/data/data/<包名称>/databases/

```
root@generic_x86:/data/data/androidEx.ch16 # cd databases/
root@generic_x86:/data/data/androidEx.ch16/databases # ls
Class
Class-journal
```

这个子类构造函数直接调用了父类的构造函数进行对象构造。四个参数分别为Context对象，数据库名字，Cursor对象（默认为null），数据库版本

使用 SQLite 数据库

接着我们重写onCreate()和onUpgrade()

```
@Override
public void onCreate(SQLiteDatabase db){
  db.execSQL(“CREATE TABLE student (” +
    “_id integer primary key, name text no null, ” +
    “age integer no null, score integer no null )”);
}

@Override
public void onUpgrade(SQLiteDatabase db,
                      int oldVersion, int newVersion) {
  db.exeSQL(“DROP TABLE IF EXISTS student”);
  onCreate(db);
}
```

SQLiteOpenHelper 的其他相关方法

- getReadableDatabase()
  - 创建或者打开（如果存在）一个只读数据库，成功打开则返回 SQLiteDatabase 对象
- getWritableDatabase()
  - 创建或者打开（如果存在）一个读写数据库，成功打开则返回 SQLiteDatabase 对象
- close()
  - 关闭打开的数据库

使用 SQL 数据库

◆ 使用 SQLiteDatabase 访问数据表

1. 打开可以擦写的数据库

   通常在Activity类的onCreate()方法中打开数据库：

```
dbHelper = new StdDBHelper(this);
Db = dbHelper.getWritableDatabase();
```

   建立StdDBHelper对象之后，调用getWritableDatabase() 方法来获取数据库**（SQLiteDatabase 对象）**

2. 关闭数据库

   通常在Activity类的onStop()方法中关闭数据库：

```
db.close();
```

使用 SQL 数据库

◆ **使用 SQLiteDatabase 访问数据表**

3. 添加记录

```
Db.execSql("insert into student (name, age, height) values('张三', 24, 170.0)");
```

4. 修改记录

```
Db.execSql("update student set age = 35 where name = '张三'");
```

```
Db.execSql("update student set age = 35 where height < 0.1");
```

5. 删除

```
Db.execSql("delete from student where name = '张三'");
```

```
Db.execSql("delete from student where age <= 10");
```

使用 SQL 数据库

◆ **使用 SQLiteDatabase 访问数据表**

6. 查询

```
Cursor c = db.rawQuery("select name, age,height from student where height > 1000.0",null);
colNames=c.getColumnNames();
int rows = c.getCount();
c.moveToFirst();
for(int I = 0;I < rows;i++){
        String name = c.getString(0);
        int age = c.getInt(1);
        float height = c.getFloat(2);
        // add your code ….
        c.moveToNext();
}
```

课堂练习(2)

课堂练习(3)

初始化数据接口

```
package com.example.abcd;

import android.content.Context;

public class SqlitedBInter extends SQLiteOpenHelper {

    private static final String DATABASE_NAME = "abcd";
    private static final int DATABASE_VERSION = 1;

    public SqlitedBInter(Context context){
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
      }

    @Override
    public void onCreate(SQLiteDatabase db) {
        // TODO Auto-generated method stub
        String sql = "create table student(name varchar2(64), age int, height float)";
        db.execSQL(sql);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // TODO Auto-generated method stub

    }
}
```

课堂练习(2)

初始化人机界面(onCreate)

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_std_info);

    Button btn_commit = (Button) findViewById(R.id.student_commit);
    btn_commit.setOnClickListener(this);

    Button btn_quit = (Button) findViewById(R.id.student_quit);
    btn_quit.setOnClickListener(this);

    Button btn_delete = (Button) findViewById(R.id.student_delete);
    btn_delete.setOnClickListener(this);

    ArrayAdapter adapter = new ArrayAdapter(this, android.R.layout.simple_list_item_1);
    ListView student_list = (ListView) findViewById(R.id.student_list);
    student_list.setAdapter(adapter);

    dbHelper = new SqlitedBInter(this);
    db = dbHelper.getWritableDatabase();
    update_list();

    EditText student_filter_name = (EditText) findViewById(R.id.student_filter_name);
    TextWatcher mTextWatcher = new TextWatcher() {
        public void beforeTextChanged(CharSequence s, int start, int count, int after) {
        }

        public void onTextChanged(CharSequence s, int start, int before, int count) {
        }

        public void afterTextChanged(Editable s) {
            update_list();
        }
    };
    student_filter_name.addTextChangedListener(mTextWatcher);
}
```

课堂练习(2)

更新listView

```
public void update_list() {
    try {
        // 更新结果
        String filter_name = ((EditText) findViewById(R.id.student_filter_name)).getText().toString();

        ListView student_list = (ListView) findViewById(R.id.student_list);
        ArrayAdapter<String> adapter = (ArrayAdapter<String>) student_list.getAdapter();

        String select_cmd = "select name,age,height from student";

        if (!filter_name.isEmpty()) {
            select_cmd += String.format(" where name like '%%%s%%'", filter_name);
        }

        Cursor c = db.rawQuery(select_cmd, null);

        adapter.clear();
        c.moveToFirst();
        for (int i = 0; i < c.getCount(); i++) {
            String name = c.getString(0);
            int age = c.getInt(1);
            float height = c.getFloat(2);
            adapter.add(name + " " + age + " " + height);
            c.moveToNext();
        }
        c.close();
    } catch (Exception e) {
        Log.d("StdInfo", "update_list:" + e.getMessage());
    }
}
```

课堂练习(2)

点击“添加”按钮

```
// 添加（或修改）一条记录
try {
    EditText edit_name = (EditText) findViewById(R.id.student_name_input);
    EditText edit_age = (EditText) findViewById(R.id.student_age_input);
    EditText edit_height = (EditText) findViewById(R.id.student_height_input);
    String name = edit_name.getText().toString();
    int age = Integer.parseInt(edit_age.getText().toString());
    float height = Float.parseFloat(edit_height.getText().toString());

    String select_cmd = String.format("select name from student where name  = '%s'", name);

    Cursor c = db.rawQuery(select_cmd, null);

    if (c.getCount() == 0) {
        String sqlCmd = String.format("insert into student (name, age,height) values('%s',%d,%f)", name,
                age, height);
        db.execSQL(sqlCmd);
    } else {
        String sqlCmd = String.format("update student set age = %d, height = %f where name = '%s'", age,
                height, name);
        db.execSQL(sqlCmd);
    }

    EditText filter_name = (EditText) findViewById(R.id.student_filter_name);
    update_list();
} catch (Exception e) {
    Log.d("StdInfo", e.getMessage());
    Toast.makeText(this, e.getMessage(), Toast.LENGTH_SHORT).show();
}
```

课堂练习(2)

点击“删除”按钮

```
} else if (v.getId() == R.id.student_delete) {
    // 删除记录
    try {
        EditText edit_name = (EditText) findViewById(R.id.student_name_input);
        String name = edit_name.getText().toString();
        String delete_cmd = String.format("delete from student where name  = '%s'", name);
        db.execSQL(delete_cmd);
        update_list();
    } catch (Exception e) {
        Log.d("StdInfo", e.getMessage());
        Toast.makeText(this, e.getMessage(), Toast.LENGTH_SHORT).show();
    }
```