2015.1.4

Android SQLite

徐　丙鉄

SQLite　データ型

INTEGER	符号付整数
REAL	浮動小数点
TEXT	テキスト
BLOB	バイナリデータ
NULL	NULL

java.lang.Object

↳ android.database.sqlite.**SQLiteOpenHelper**

データベース作成，オープン，アップグレード

public Constructors

SQLiteOpenHelper(Context context, String **name**, SQLiteDatabase.CursorFactory factory, int version)

Called when the database is created for the first time.

Parameters

context　to use to open or create the database

name　of the database file, or null for an in-memory database
ファイル名を指定した場合は /data/data/パッケージ名/database/ファイル名 に、データベースファイルが作成される。

factory　to use for creating cursor objects, or null for the default

version　number of the database (starting at 1);

public　SQLiteDatabase　**getReadableDatabase()**　Create and/or open a database.

---

java.lang.Object

↳ android.database.sqlite.SQLiteClosable

↳ android.database.sqlite.**SQLiteDatabase**

データーベースの操作：追加 insert，削除 delete，検索 query

Cursor　**query**(String table, String[] columns, String selection,
String[] selectionArgs, String groupBy, String having,
String orderBy, String limit)
Query the given table, returning a Cursor over the result set.

void　**execSQL**(String sql)
Execute a single SQL statement that is NOT a SELECT or any other SQL tatement that returns data.

long　**insert**(String table, String nullColumnHack, ContentValues values)
Returns: the row ID of the newly inserted row, or -1 if an error occurred
Convenience method for inserting a row into the database.

---

public interface

Cursor

This interface provides random read-write access to the **result set returned by a database query.**

abstract String **getString**(int columnIndex)
Returns the value of the requested column as a String
abstract int **getInt**(int columnIndex)
Returns the value of the requested column as an int.
abstract boolean **moveToNext**()
Move the cursor to the next row. Returns whether the move succeeded.

参考 http://ichitcltk.hustle.ne.jp/gudon2/index.php?pageType=file&id=Android032_SQLite

sample1 ----------

```
static final String DB = "sqlite_sample.db";
static final int DB_VERSION = 1;
static final String CREATE_TABLE = "create table mytable ( _id integer primary key
autoincrement, data integer not null );";
static final String DROP_TABLE = "drop table mytable;";

private static class MySQLiteOpenHelper extends SQLiteOpenHelper {
    public MySQLiteOpenHelper(Context c) {
        super(c, DB, null, DB_VERSION);
    }
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(CREATE_TABLE);
    }
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        db.execSQL(DROP_TABLE);
        onCreate(db);
    }
}
```

注：Android の一部のクラスが「_id」という名前のユニークキーの存在を前提に作られている。例えば、Android の画面部品の中に、データベースの検索結果を表示しやすくするための仕組みが備わっているものがある。この仕組みの中で「_id」という名前のカラムが利用されている。そのため、“必須”ではないが、特に理由がない限り、「_id」というカラムを定義するのが定石になっている。

sample2----------

```
import android.app.Activity;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.widget.LinearLayout;
import android.widget.TextView;

public class SQLiteSample extends Activity {
```

```
        private SQLiteDatabase studyDB;
        private String studyDataTableName;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

    StudeyDBOpenHelper helper = new StudeyDBOpenHelper (this);
                studyDB = helper.getReadableDatabase();
                studyDataTableName = helper. getStudyDataTableName ();
                //テーブルから全データ抽出
              Cursor c = studyDB.query(   studyDataTableName, new String[] { "name", "time" },
                                                                  null, null, null, null, null);
            //後ろ５つの引数  String[] selectionArgs, String groupBy, String having, String orderBy, String limit

        boolean isNext = c.moveToFirst();
        while (isNext) {
            TextView tv = new TextView(this);
            tv.setText(String.format("%s：%d  分", c.getString(0), c.getInt(1) ));
            isNext = c.moveToNext();
            layout.addView(tv);
        }
        c.close();      studyDB.close();
    }
    // 1 レコード挿入
    public void insertData( String name, String time ) {
                ContentValues cv = new ContentValues();
                cv.put( "name", name );    cv.put("time", time);
                studyDB.insert( studyDataTableName, null, val );
    }

}

-----------------------------------------------------------------------------------------------------------------------
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

public class StudeyDBOpenHelper extends SQLiteOpenHelper {
    final static private int DB_VERSION = 1;
    final private String dbFileName =”study.db”;
    final static String studyDataTableName = "studyData_table";

    public StudeyDBOpenHelper (Context context) {
        super(context, dbFileName, null, DB_VERSION);
    }

    public void onCreate(SQLiteDatabase db) {
        String CREATE_TABLE = "create table" + studyDataTableName +
                                                "( name text not null, time integer not null ); ";
```

```
        String insertSql = "insert into " + studyDataTableName + " (name,time) values";
     // table create
        db.execSQL(    CREATE_TABLE );
     //  初期データ入力 table row insert
         /* ContentValues val = new ContentValues();
                val.put( "name", caption );
                 db.insert(studyDataTableName, null, val);    */

        db.execSQL( insertSql + " ('近藤', 30);"    );
        db.execSQL( insertSql + " ('遠藤', 60);"    );
        db.execSQL( insertSql + " ('松井', 50);"    );
    }

    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        // データベースの変更が生じた場合は、ここに処理を記述する。
    }
    //追加のメソッド
    public String getStudyDataTableName(){ return studyDataTableName; }

}
```