aardio 范例: sqlite 库 - 参数化进阶

```
//sqlite 库 - 参数化进阶

/*
一、说明
aardio 中所有数据库的支持库都支持用法与规则类似的参数化语法，
请不要用字符串拼接的方法生成 SQL 语句，这会导致不当字符注入 SQL 语句的意外或风险。

参数化查询可以避免注入问题。
aardio 在预处理命令使用了 sqlite 原生的参数化查询功能，
也使用 sqlite.format 改进和增强了 SQL 参数化查询的语法。

sqlite 中执行SQL的函数 sql 语句后面如果是一个表，
就会调用 sqlite.format 使用参数化规则格式化并生成新的 SQL 语句。
如果参数不是一个表，则简单的调用 string.format 函数格式化。
string.format 并不具有参数化、避免注入等功能，应当尽量避免使用。
*/

import console;
import util.table;
import sqlite;

sqlite.format2 = sqlite.format;
sqlite.format = function(sql,...){
    console.log('\r\n格式化代码：')
    console.log("-------------------------------------------------")
    console.log(`sqlite.format("`+sql + `",` + util.table.stringify(...) + `);` );
    console.log('\r\n格式化结果：')
    console.log("-------------------------------------------------")

    sql = sqlite.format2(sql,...)
    console.log(sql);

    console.more(1,true);
    return sql;
}

console.log( "SQL语句中 @ 字符开始的命名参数使用参数表的名值对元素格式化" );
sqlite.format("SELECT *  FROM `tab-name` WHERE name = @name",{name = "参数值"})

console.log( "SQL语句中 ? 或 ?? 占位符使用参数表的数组元素格式化" );
sqlite.format("SELECT *  FROM `tab-name` WHERE name = ? AND text = ?",{"参数值1","参数值2"});

console.log( "其中 ?? 格式化为标识符( 放入反引号 ),其他占位符格式化为参数值" );
sqlite.format("SELECT *  FROM ?? WHERE name = ? AND text = ?",{"`format-tab`","参数值1","参数值2"});

console.log( "字符串转为 SQL 安全转义字符串,buffer转为X'4D7953514C'格式" );
sqlite.format("SELECT *  FROM ``format-tab`` WHERE str = @str AND buffer = @buffer",{str="a'a";buffer=raw.buffer("sqlite")});

console.log( "数组则自动展开为列表,例如{'a', 'b'}格式化为'a', 'b'" );
sqlite.format("INSERT INTO `format-tab`(??) VALUES(?)",{{'name1', 'name2'},{'value1', 'value2'}});

console.log( "嵌套数组则格式化为分组列表" );
sqlite.format("INSERT INTO `format-tab`(text,number) VALUES @values",{ values = { {'text1', 123}, {'text2', 456} } });

console.log( "非数组的命名表，则格式化为 SQL 键值对,默认以逗号为分隔符" );
sqlite.format("UPDATE `format-tab` SET ?",{ {text="text2";number=123} });

console.log( "??占位符用于格式化 SQL 键值对则以 AND 为分隔符,数组值转换为IN语句" );
sqlite.format("UPDATE `format-tab` SET text=? WHERE ??",{ "text2+",{text="text2";number={1,2,456}} });

sqlite.format = sqlite.format2;
var db = sqlite("/test-sqlite.db");
if( not db.existsTable("format-tab") ){
    db.exec( "CREATE TABLE `format-tab`(text, number);");
}

//执行参数化 SQL 插入数据
db.exec("INSERT INTO `format-tab`(text,number) VALUES @values",{ values = { {'text1', 123}, {'text2', 456} } });

//查询数据
console.dumpJson(db.getTable("SELECT * FROM `format-tab` "))

//删除测试表
db.exec("DROP TABLE [format-tab]" );
console.pause();
```

Markdown 格式