

Lua 教程
Lua 环境安装
Lua 基本语法
Lua 数据类型
Lua 变量
Lua 循环
Lua 流程控制
Lua 函数
Lua 运算符
Lua 字符串
Lua 数组
Lua 迭代器
Lua table(表)
Lua 模块与包
Lua 元表(Metatable)
Lua 协同程序(coroutine)
Lua 文件 I/O
Lua 错误处理
Lua 调试(Debug)
Lua 垃圾回收
Lua 面向对象
Lua 数据库访问
Lua 5.3 参考手册

← Lua 迭代器

Lua 模块与包 →

Lua table(表)

table 是 Lua 的一种数据结构用来帮助我们创建不同的数据类型，如：数组、字典等。Lua table 使用关联型数组，你可以用任意类型的值来作数组的索引，但这个值不能是 nil。Lua table 是不固定大小的，你可以根据自己需要进行扩容。

Lua 通过 table 来解决模块 (module)、包 (package) 和对象 (Object) 的问题。例如 string.format 表示使用 "format" 来索引 table string。

table(表)的构造

构造器是创建和初始化表的表达式。表是 Lua 特有的功能强大的东西。最简单的构造函数是 {}，用来创建一个空表。可以直接初始化数组。

```
-- 初始化表
mytable = {}

-- 指定值
mytable[1] = "Lua"

-- 移除引用
mytable = nil
-- Lua 释放回收会释放内存
```

当我们为 table a 并设置元素，然后将 a 赋值给 b，则 a 与 b 都指向同一个内存。如果 a 设置为 nil，则 b 同样能访问 table 的元素。如果没有指定的变量指向 a，Lua 的垃圾回收机制会清理相对应的内存。

以下实例演示了以上的描述情况：

实例

```
-- 简单的 table
mytable = {}
print("mytable 的类型是 ", type(mytable))

mytable[1] = "Lua"
mytable["wow"] = "修改前"
print("mytable 索引为 1 的元素是 ", mytable[1])
print("mytable 索引为 wow 的元素是 ", mytable["wow"])

-- alternatetable #!mytable的是指同一个 table
alternatetable = mytable

print("alternatetable 索引为 1 的元素是 ", alternatetable[1])
print("mytable 索引为 wow 的元素是 ", alternatetable["wow"])

-- 改变变量
alternatetable = nil
print("alternatetable 是 ", alternatetable)

-- mytable 仍然可以访问
print("mytable 索引为 wow 的元素是 ", mytable["wow"])

mytable = nil
print("mytable 是 ", mytable)
```

以上代码执行结果为：

```
mytable 的类型是 table
mytable 索引为 1 的元素是 Lua
mytable 索引为 wow 的元素是 修改前
alternatetable 索引为 1 的元素是 Lua
mytable 索引为 wow 的元素是 修改前
mytable 索引为 wow 的元素是 修改后
alternatetable 是 nil
mytable 索引为 wow 的元素是 修改后
mytable 是 nil
```

Table 操作

以下列出了 Table 操作常用的方法：

序号	方法 & 用途
1	table.concat (table [, sep [, start [, end]]]): concat 是 concatenate(连接) 的缩写。table.concat() 函数列出参数中指定 table 的数组部分从 start 位置到 end 位置的所有元素，元素间以指定的分隔符 (sep) 隔开。
2	table.insert (table, [pos], value): 在 table 的数组部分指定位置 (pos) 插入值为 value 的一个元素。pos 参数可选，默认为数组部分末尾。
3	table.maxn (table) 指定 table 中所有正数 key 值中最大的 key 值。如果不存在 key 值为正数的元素，则返回 0。 (Lua 5.2 之后该方法已经不存在了，本文使用了自定义函数实现)
4	table.remove (table [, pos]): 返回 table 数组部分位于 pos 位置的元素。其后的元素会被前移。pos 参数可选，默认为 table 长度，即从最后一个元素删起。
5	table.sort (table [, comp]): 对给定的 table 进行升序排序。

接下来我们来看下这几个方法的实例。

Table 连接

我们可以使用 concat() 输出一个列表中元素连接成的字符串：

```
fruits = {"banana", "orange", "apple"}
-- 返回 table 连接后的字符串
print("连接后的字符串 ", table.concat(fruits))

-- 指定连接字符串
print("连接后的字符串 ", table.concat(fruits, ","))

-- 指定索引来连接 table
print("连接后的字符串 ", table.concat(fruits, "", 2, 3))
```

执行以上代码输出结果为：

```
连接后的字符串 bananaorangeapple
连接后的字符串 banana, orange, apple
连接后的字符串 orange, apple
```

插入和移除

以下实例演示了 table 的插入和移除操作：

```
fruits = {"banana", "orange", "apple"}

-- 在末尾插入
table.insert(fruits, "mango")
print("索引为 4 的元素为 ", fruits[4])

-- 在索引为 2 的键处插入
table.insert(fruits, 2, "grapes")
print("索引为 2 的元素为 ", fruits[2])

print("最后一个元素为 ", fruits[5])
table.remove(fruits)
print("移除后最后一个元素为 ", fruits[5])
```

执行以上代码输出结果为：

```
索引为 4 的元素为 mango
索引为 2 的元素为 grapes
最后一个元素为 mango
移除后最后一个元素为 nil
```

Table 排序

以下实例演示了 sort() 方法的使用，用于对 Table 进行排序：

```
fruits = {"banana", "orange", "apple", "grapes"}
print("排序前")
for k, v in pairs(fruits) do
    print(k, v)
end

table.sort(fruits)
print("排序后")
for k, v in pairs(fruits) do
    print(k, v)
end
```

执行以上代码输出结果为：

```
排序前
1  banana
2  orange
3  apple
4  grapes
排序后
1  apple
2  banana
3  grapes
4  orange
```

Table 最大值

table.maxn 在 Lua 5.2 之后该方法已经不存在了，我们定义了 table_maxn 方法来实现。

以下实例演示了如何获取 table 中的最大值：

```
function table_maxn(t)
    local mn=nil
    for k, v in pairs(t) do
        if mn==nil then
            mn=v
        else
            if mn < v then
                mn = v
            end
        end
    return mn
end
```

```
tbl = {[1] = 2, [2] = 6, [3] = 34, [26] = 5}
print("tbl 最大值: ", table_maxn(tbl))
print("tbl 长度 ", #tbl)
```

执行以上代码输出结果为：

```
tbl 最大值: 34
tbl 长度 3
```

注意：

当我们获取 table 的长度的时候无论是使用 # 还是 table.getn，其都会在索引中断的地方停止计数，而导致无法正确取得 table 的长度。

可以使用以下方法来代替：

```
function table_leng(t)
    local leng=0
    for k, v in pairs(t) do
        leng=leng+1
    end
    return leng
end
```

执行以上代码输出结果为：

```
tbl 最大值: 34
tbl 长度 3
```

分类导航

HTML / CSS

JavaScript

服务端

数据库

移动端

XML 教程

ASP.NET

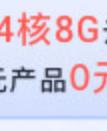
Web Service

开发工具

网站建设

Advertisement

广告 X



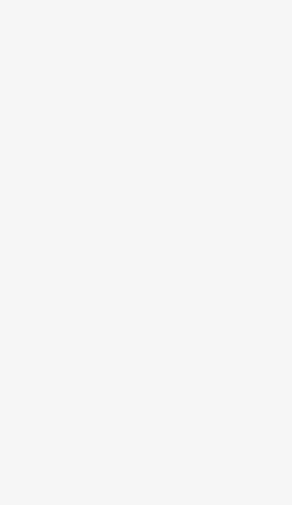
华为云

免费试用

高达4核8G云服务器

千元产品0元体验

立即领取



← Lua 迭代器

Lua 模块与包 →

6 篇笔记

写笔记

1M独立带宽云服务器，100元/年

特价云服务器促销，1M独立带宽，自建BGP，独立IP，CPU负载无限制，20G SSD数据盘

立即申请

UCloud

在线实例

HTML 实例

CSS 实例

JavaScript 实例

Ajax 实例

jQuery 实例

XML 实例

Java 实例

字符集&工具

HTML 字符集设置

HTML ASCII 字符集

HTML ISO-8859-1

HTML 实体符号

HTML 换行器

JSON 格式化工具

最新更新

PostgreSQL 删除...

PostgreSQL 选择...

PostgreSQL 创建...

PostgreSQL 数据...

C 语言整数与字...

Vue.js Ajax(x...

PostgreSQL 语法

站点信息

PostgreSQL 备份...

PostgreSQL 优化...

PostgreSQL 语义...

PostgreSQL 语句...

PostgreSQL 视图...

PostgreSQL 触发...

PostgreSQL 索引...

PostgreSQL 视图...

PostgreSQL 触发...

PostgreSQL 视图...