

Lua模块与元表(Metatable)

凉笙

0.1 2018.03.22 00:07 字数 818 阅读 818 评论 0 喜欢 1

一、模块 (module)

模块类似于一个封装库，可以把一些公用的代码放在一个文件里，以 API 接口的形式在其他地方调用，有利于代码的重用和降低代码耦合度。

Lua 的模块是由变量、函数等已知元素组成的 table，因此创建一个模块很简单，就是创建一个 table，然后把需要导出的常量、函数放入其中，最后返回这个 table 就行。以下为创建自定义模块 `module.lua`。

```
--文件名为module.lua
--定义一个module的模块
module={}
--定义一个常量
module.var="Lian"

--局部函数，相当于c#中的私有函数，不能在外界调用
local function func1()
    print("这是一个局部私有函数")
end

--定义全局一个函数
function module.func2()
    func1()
    print("这是一个全局公有函数")
end

return module
```

由上可知，模块的结构就是一个 table 的结构，因此可以像操作调用 table 里的元素那样来操作调用模块里的常量或函数。

上面的 func1 声明为程序块的局部变量，即表示一个私有函数，因此是不能从外部访问模块里的这个私有函数，必须通过模块里的公有函数来调用。

下面再去创建usermodule.lua文件去调用模块中的函数和变量,使用require 函数

Lua提供了一个名为**require**的函数用来加载模块。要加载一个模块，只需要简单地调用就可以了。

```
--require "模块名"
m= require "module"

print(m.var)--调用模块中的变量
m.func2()--调用模块中的全局函数
```

最后运行可以看到输出结果调用了module模块中的函数和变量

```
运行结果：
Lian
这是一个局部私有函数
这是一个全局公有函数
```

二、元表 (Metatable)

在 Lua table 中我们可以访问对应的key来得到value值，但是却无法对两个 table 进行操作。

因此 Lua 提供了元表(Metatable)，允许我们改变table的行为，每个行为关联了对应的元方法。

例如，使用元表我们可以定义Lua如何计算两个table的相加操作a+b。

有两个很重要的函数来处理元表：

```
setmetatable(table,metatable):
对指定table设置元表(metatable)。如果元表(metatable)中存在__metatable键值，setmetatable会返回该元表。

getmetatable(table): 返回对象的元表(metatable)。
```

如何对指定的表设置元表：

```
mytable={"Lua","Java","C","C#","C++"}--普通表
--使用__metatable可以保护元表，禁止用户访问元表中的成员或者修改
mymetatable={__metatable="lock"}--元表 元表扩展了普通表的行为

mytable=setmetatable(mytable,mymetatable)--给指定的表设定元表
print(getmetatable(mytable))--返回普通表的元表
```

输出结果：lock

__index 元方法

是 metatable 最常用的键。

当你通过键来访问 table 的时候，如果这个键没有值，那么Lua就会寻找该table的 metatable（假定有metatable）中的__index 键。如果__index包含一个表格，Lua会在表格中查找相应的键。

如果__index包含一个函数的话，Lua就会调用那个函数，table和键会作为参数传递给函数。

__index 元方法查看表中元素是否存在，如果不存在，返回结果为 nil；如果存在则由__index 返回结果。

```
--__index用来处理访问到的索引不存在的时候怎么办
mytable={"Lua","Java","C","C#","C++"}--普通表
mymetatable={
    __index=function (tab,key)
        return "JavaScript"
    end
}--元表 元表扩展了普通表的行为

mytable=setmetatable(mytable,mymetatable)--给指定的表设定元表
print(mytable[10])--访问表中不存在的键时会执行元表中的匿名函数
```

输出结果 JavaScript

__newindex 元方法

__newindex 元方法用来对表更新，__index则用来对表访问。

当你给表的一个缺少的索引赋值，解释器就会查找__newindex 元方法；如果存在则调用这个函数而不进行赋值操作。

```
--__newindex当我们对表的数据进行修改的时候，当我们给表添加新的键值对的时候会起作用
mytable={"Lua","Java","C","C#","C++"}--普通表
mymetatable={
    __newindex=function (tab,key,value)
        print("我们要修改的key为:"..key.."把这个key值修改为:"..value)

        rawset(tab,key,value)--设置键值对，添加输入进来的键值对
    end
}--元表 元表扩展了普通表的行为

mytable=setmetatable(mytable,mymetatable)--给指定的表设定元表
mytable[6]="JavaScript"
print(mytable[6])
```

输出结果:

```
我们要修改的key为:6把这个key值修改为:JavaScript
JavaScript
```

为表添加操作符

__add元方法 当两个表相加时实现相应的函数方法

```
mytable={"Lua","Java","C","C#","C++"}--普通表
mymetatable={
    --两表相加将表中的元素添加到mytable中去
    __add=function (tab,newtab)
        local mi=0
        for k,v in pairs(tab) do
            if (k~=1)then
                mi=k
            end

            for k,v in pairs(newtab) do
                mi=mi+1
                table.insert(tab,mi,v)
            end
            return tab
        end
    }--元表 元表扩展了普通表的行为

mytable=setmetatable(mytable,mymetatable)--给指定的表设定元表

newtab={"PHP","Python"}

--其中只要有一个表设置了元表都可以实现加法
v=mytable+newtab

for k,v in pairs(v) do
    print(k,v)
end
```

输出结果:

```
1 Lua
2 Java
3 C
4 C#
5 C++
6 PHP
7 Python
```

其他操作符:

操作符	描述
__add	对应的运算符 '+'.
__sub	对应的运算符 '-'.
__mul	对应的运算符 '*'.
__div	对应的运算符 '/'.
__mod	对应的运算符 '%'.
__unm	对应的运算符 '-'.
__concat	对应的运算符 '..'.
__eq	对应的运算符 '=='.
__lt	对应的运算符 '<'.
__le	对应的运算符 '<='.

image.png

__call元方法

__call 元方法在 Lua 调用一个值时调用。也就是说当我们把表当作函数来使用的时候调用，例如：mytable(25)

```
mytable={"Lua","Java","C","C#","C++"}
mymetatable={
    __call=function (tab,arg)
        print("Call元方法_"..arg)
    end
}

mytable=setmetatable(mytable,mymetatable)
mytable("京")
```

输出结果 Call元方法_京

__tostring 元方法

__tostring 元方法用于修改表的输出行为。表示在使用输出print(table)的时候会调用元表中的函数,如下

```
mytable={"Lua","Java","C","C#","C++"}
mymetatable={
    __tostring=function (tab)
        local str=""
        for k,v in pairs(tab) do
            str=str..v.." "
        end
        return str
    end
}

mytable=setmetatable(mytable,mymetatable)

print(mytable)
```

输出结果:

```
Lua,Java,C,C#,C++
```

小礼物走一走，来简书关注我

赞赏支持

Lua

原创文章 0 著作权归作者所有

凉笙

写了 70313 字，被 202 人关注，获得了 197 个喜欢

时间最强大的一件事就是，它会改变未来

喜欢


1



更多分享



下载简书 App 随时随地发现和创作内容





登录后发表评论

评论



智慧如你，不想发表一点想法吗~

推荐阅读

更多精彩内容 >

全套抖音玩法，一步接入！注册了解>>

短视频SDK，广电级视频编解码算法，架构灵活接口丰富。一小时快速接入。

广告

提供智能搜索音频整体解决方案

Lua 5.1 参考手册

Lua 5.1 参考手册 by Roberto Ierusalimsky, Luiz Henrique de Figueiredo, Waldemar Celes 云风译

www.codingnow.com Copyright © 2006 Lua.org, PUC-Rio...

苏黎九歌

lua

Lua 是一种轻量小巧的脚本语言，用标准C语言编写并以源代码形式开放， 其设计目的是为了嵌入应用程序中，从而为应用程序提供灵活的扩展和定制功能。 ...

萧非子



Lua笔记

Lua 是一种轻量小巧的脚本语言，用标准C语言编写并以源代码形式开放， 其设计目的是为了嵌入应用程序中，从而为应用程序提供灵活的扩展和定制功能。 ...

、沙工



[code.openresty] Openresty Nginx API for Lua -上

Nginx API for Lua Introduction ngx_arg ngx_var VARIABLE Core constants HTTP method constants HTTP status constants Nginx log level consta...

吃瓜的东

源码分享最新教程 - 草稿

¥启动¥ [iAPP实现进入界面执行第一屏] 【2017-08-25 15:22:14】 《/首先开一个线程，因为你这么多的stop要执行，其次必须使用ufusui（）,否则可能不会显示你的字体。切记:文字必须设置'text'，否则像端...

小翼c

你是否保持最初的模样

人之初，性本善。 这是我最喜欢的一句话。我们每个人都是赤裸裸的来到这个世界，在一声啼哭中，告诉人们一个纯粹干净的人生就此开始。当我们第一次睁开双眼时，世界是那般干净明亮，不知什么时候开始， ...

倩文湘云

那些愿意为你花钱的人，都是用心对你的人

前两天一个朋友发朋友圈在吐槽说她一个朋友平时没事都不找她，可是今年过年就找了她。而简单又现实的原因是爱花钱，每次借完了钱就还，连简单的时候零嘴都没有。让朋友感觉到有点心疼。 后来她就在下...

潘点潘点赵赵

希腊移民排期周期情况介绍

希腊被誉为是西方文明的发源地，拥有悠久的历史。希腊旅游业极为发达，美丽的爱琴海，圣托里尼岛等构成了希腊蓝白与白的世界。近年，很多人被希腊所吸引...

chenderen



无绪

静夜思 思 思不出

水笔仔o12530

开始

我甚至都记不清自己有多久没有写过东西了。以前的我有个怪癖，必须在纸上写写画画，然后摩挲了一遍又一遍纸上的字后，再誊写到电脑上。微博说很久没看到我写的东西了，是啊，像微信这种怎么玩不腻的工...

虞实的现实