Lua元表（Metatable)(原创)

Frank_何益明 +关注
2018.04.19 15:39* 字数 1278 阅读 360 评论 0 喜欢 1

元表这个词听起来就觉得抽象，我开始接触Lua的时候就是这种感觉。其实不要被表面吓到。

元表就是，如果一个tableB 调用setmetatable方法设置另外一个tableA作为它的元表，那么当你操作tableB进行如设置键值，查找键值，table间使用运算符号、或者直接传个参数调用等操作时就会变得不一样或者原本不可行操作的现在可行了。而tableA之所以能作为元表，是因为里面包含了特殊的键代表着特殊的含义，代码在运行时会做特殊的处理，下面会一一做介绍。

首先记住两个函数：

> ***setmetatable(table,metatable)***

对指定table设置元表(metatable)，如果元表(metatable)中存在__metatable键值，setmetatable会失败 。

> **getmetatable(table)**

返回对象的元表(metatable)。如元表中存在__metatable键值，返回元表会失败

__metatable键值是用于安全考虑，可以防止获取和修改元表中的内容。

下面实例演示如何设置元表

```
metatable= {}    ---- 元表

table= {}    ----普通表

setmetatable(table,metatable)    ----将metatable设置为table的元表
```

也可以直接这么写

```
tableB = setmetable({},{})
```

返回元表

```
getmetatable(tableB )     -----返回tableA
```

1、访问与设值

__index键

当你通过一个键访问table的时候，如果这个table中没有这个键值对，但它设置了元表，且元 表中有__index这个键,如果__index的值为一个表，那么程序就会到这个表中去查找，如果__index的值为一个函数，那么程序就会 调用这个函数。看代码

```
local tableA = {k2 = "Hello"}

 local tableB = {k1 = "Hi"}

 setmetatable(tableB,{__index = tableA})

 print(tableB.k2)

打印：

Hello
```

__index 值为函数

```
function fn (table,key)

   print(table,"\n",key)

   return "Hello"

 end

 local tableB = {k1 = "Hi"}

 setmetatable(tableB,{__index = fn})      ----会将table 和 key作为参数传给fn

 print(tableB.k2)

打印：

table: 000000000067a690

k2

Hello
```

__newindex键

__index用于访问，__newindex用于设置新值。当一个table设置了元表，为这个table设置键值的时候，如果这个键存在，那么直接修改当前键值；如果这个键不存在，那么会操作元表中__newindex相应的值。__newindex的值为一个table或者函数。看如下代码

```
  local tableA = {}

  local tableB = setmetatable({k1 = "Hi"},{__newindex = tableA})

  tableB.k1 = "HiHi"

  tableB.k2 = "Hello"

  print(tableB.k1,tableB.k2,tableA.k2)

打印结果：

HiHi   nil   Hello      -------tableA 被设置了一个新值
```

__newindex的值为函数：

```
function fn (table,key,value)

   print(table,"\n",key,"\n",value)

 end

 local tableB = {k1 = "Hi"}

 setmetatable(tableB,{__newindex = fn})    ---会将table、key、value传给fn作为参数

 tableB.k2 = "Good"

打印结果：

table: 000000000256a650

k2

Good
```

> **温馨提示：以上例子都是通过一个key访问具体值，如果这个key为一个方法也是同样试用的，自己没有会到元表中去操作。**

2、运算符

当我们想table与table之间使用"+、-、*、/、==、<="等运算时，就得通过元表来实现。以"+"为例，对应的元方法为**__add**,看如下代码，实现table1+table2 输出一个newtable,newtable中的元素为table1与table2的和， 看如下代码 ：

```
local table1 = {1,2,3}

local table2 = {4,5,6}

local fn = function (t1,t2)         -----table会被传入作为参数

  local newtable = {}

  for i=1,3 do

    newtable[i] = t1[i] + t2[i]

  end

  return newtable

end

setmetatable(table2,{__add = fn})

local newtable = table1+table2

for k,v in pairs(newtable) do

  print("newtable --",k,v)

end

打印结果：

newtable -- 1    5

newtable -- 2    7

newtable -- 3    9
```

两表相加，必须至少其中一个表设置了带__add键的元表，否则会报错（其他运算符同理），程序会执行__add对应的函数。**如果两个表都设置了有__add键的元表，程序会去执行"+"左侧的表中的元表的__add对应的函数。**

其他运算符：

模式	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 '<='.

3、调用

__call键

我们可以直接像调用普通方法一样传入参数直接调用table，但前提是table设置了元表，并且元表中有__call键。看代码：

```
local table = {1,2,3}

 local sum = 0

 local fn = function (t,v)       ----table会作为第一个参数，其他的为传入的参数

  for i=1,#t do

     sum =sum+ t[i] *v       ----将table中的每一个元素*5 再相加求和

  end

  return sum

 end

 setmetatable(table,{__call = fn})

 print(table(5))      -----调用table，传入一个参数5（可以传入多个参数）

打印结果：

30
```

4、自定义打印

__tostring键

可以自定义table的输出行为，看如下代码：

```
local table = {key1 = "a" ,ke2 = "b" ,ke3 = "c"}

 local  fn = function (t)          ----table会作为参数传入

     return "Hello here"

 end

 setmetatable(table,{__tostring = fn})

 print(table)

打印结果：

Hello here
```

以上就是元表的相关介绍。祝你每天都有收获

Frank_何益明 +关注
写了 5992 字，被 8 人关注，获得了 5 个喜欢
目前专业：iOS开发 游戏开发 lua脚本学习原则：不是理解带来熟练，而是熟练带来理解。刻意练习很重要。