元表

元表的定义

允许我们改变table的行为。

setmetatable(普通表，元表)

```
-- 元表
a = {"a","b","c"} -- 普通表
b = {} --元表

c = setmetatable(a,b)

print("------------------------")

f = {}

print("f:",f)
d = setmetatable({"c","d"},f)
print(d[1])

e = getmetatable(d)
print("e:",e)
```

index元方法

__index (两个下划线)

定义普通表 p。

给普通表p，设置元表y，而元表y中有__index，__index=一个表 i，i中有 我们访问的那个不存在的key。

__index=表

```
print("-------------测试__index---------------")
-- 普通表
tab1 = {"a","b","c"}
print(tab1[5])

-- 普通表，有一个元素 5="e"
newTab = {}
newTab[5] = "e"
```

```
metaTab1 = {__index=newTab}

setmetatable(tab1, metaTab1)
print(tab1[5])
```

--index=函数(表，key)

```
-- 普通表
tab1 = {"a","b","c"}
print(tab1[5])

print("原始的tab1:",tab1)

metaTab1 = {
        __index=function(tab, key )
                print("参数当中的tab:",tab)
                print("参数当中的key：",key)

                if(key == 5) then
                        return "index--5"
                end

        end

}


setmetatable(tab1, metaTab1)
print(tab1[5])
```

请求表中key的值：

先在普通表中找，有返回，没有，看元表。

如果元表有__index, 且 __index中有对应的key。

如果没有，继续找__index的function。

newindex元方法

对表进行更新时调用。

函数用法

```
print("-----------newindex--------------")
mytab2 = {"a","b"}
metatab2 = {
	__newindex = function(tab, key ,value)
		print("被调用")
		-- tab[key] = value
		rawset(tab,key,value)

	end
}
setmetatable(mytab2,metatab2)

mytab2[3]="c"
print(mytab2[3])
```

表

```
mytab2 = {"a","b"}
mytab21 = {}
metatab2 = {
	__newindex = mytab21
}
setmetatable(mytab2,metatab2)

mytab2[3]="c"
print(mytab2[3])
print(mytab2[3])
```

为表添加操作符

加法操作

```
print("-------------操作符------------")
tab3 = {"1","2"}
tab4 = {"a","b","c"}

metatab3 = {
	__add = function(tab1,tab2)
		local m = #tab1

		for k,v in pairs(tab2)
		do
			m = m+1
			tab1[m] = v
```

```
        end

        return tab1

    end

}

setmetatable(tab3,metatab3)

v = tab3 + tab4
print(v)
for k,v in pairs(v)
do
    print(k,v)
end
```

__add:+

__sub: -

_mul:*

__div: /

__mod: %

__concat: ..

__eq:==

__lt: <

_le: <=

call元方法

lua中，当表被当成函数调用时，会触发。

```
print("-----------call-------------")
tab_a1 = {"a","b"}
print("tab_a1原始值：",tab_a1)
tab_a2 = {"1","2"}

metatab_a = {
    __call = function(tab, ...)
```

```
        local a = {...}
        for k,v in pairs(a) do

            print(v)
        end


    end

}
setmetatable(tab_a1,metatab_a)



result = tab_a1(6,7,8)
```

tostring

用于修改表的输出行为。类似于java中的toString（ ）。

```
print("-----------call-------------")
tab_a1 = {"a","b","c","d"}
print("tab_a1原始值：",tab_a1)
tab_a2 = {"1","2"}

metatab_a = {

    __call = function(tab, ...)
        local a = {...}
        for k,v in pairs(a) do

            print(v)
        end


    end,



    __tostring = function(tab)
        local str = ""
        for k,v in pairs(tab) do

            str = str..v..","
```

```
            end

            return str
        end




}
setmetatable(tab_a1,metatab_a)

-- result = tab_a1(6,7,8)

print(tab_a1)
```

ps：每个元方法之间 用 ,