# **Lua的for in和pairs**

Lua中的for in迭代语法格式如下：

for k,v in pairs(t) do

print(k,v)

End

查看以下实例：

array = {"Lua", "Tutorial"}

for key,value in pairs(array) do

print(key, value)

End

输出结果为：

1 Lua

2 Tutorial

**pairs和ipairs**

上面的例子中，将pairs换成ipairs结果是一样的。然而它们的实现却有些不同，官方手册的描述：

ipairs (t) Returns three values: an iterator function, the table t, and 0, so that the construction for i,v in ipairs(t) do body end will iterate over the pairs (1,t[1]), (2,t[2]), ···, up to the first integer key absent from the table. pairs (t) Returns three values: the next function, the table t, and nil. next (table [,index]) Allows a program to traverse all fields of a table. Its first argument is a table and its second argument is an index in this table. next returns the next index of the table and its associated value. When called with nil as its second argument, next returns an initial index and its associated value. When called with the last index, or with nil in an empty table, next returns nil. If the second argument is absent, then it is interpreted as nil. In particular, you can use next(t) to check whether a table is empty.

ipairs适用于数组（i估计是integer的意思），pairs适用于对象，因为数组也是对象，所以pairs用于数组也没问题。

next(t)或next(t,nil)得到t的第一个key,value，将取出的key作为next的第二个参数又可以得到t的 第二个key,value，直到t的末尾。

**for in**

for k,v in pairs(t) do

print(k,v)

end

实际上是：

for k,v in next,t,nil do

print(k,v)

End

for in在自己内部保存三个值：迭代函数、状态常量、控制变量。更通用的for in语法应该是：

for k,v in iter,tab,variable do

body

End

我认为for in是一种语法糖，等价的代码是：

k,v = iter(tab,variable)

if(k) then

Repeat

Body

k,v = iter(tab,k)

until(not k)

End

《Programming in Lua》给出的代码是：

Do

local \_f,\_s,\_var = iter,tab,var

while true do

local \_var,value = \_f(\_s, \_var)

if not \_var then break end

Body

End

End

**Ipairs**

在手册里，pairs的返回值能明确看到，但ipairs里的an iterator function却没有说明。ipairs可以这样实现：

function iter(t,k)

k = k+1

if(t[k])then

return k,t[k]

End

End

function ipairs(a)

return iter,a,0

End

**for in**

array = {"Lua", "Tutorial"}

function elementIterator(collection)

local index = 0

local count = #collection

return function()

index = index + 1

if index <= count then

return collection[index]

End

End

End

for element in elementIterator(array) do

print(element)

End

elementIterator(array)返回一个匿名函数作为迭代函数，该迭代函数会忽略掉传给它的参数array和nil，array和控制变量已被保存在迭代函数中。 将for in展开来看会更明显：

iterator = elementIterator(array)

element,value = iterator(nil,nil)--忽略参数,value置为nil

if(element) then

Repeat

print(element)

element,value = iterator(nil,element)--忽略参数

until(not element)

End

原文链接：<http://blog.csdn.net/liuyuan185442111/article/details/54144348>