OpenResty+LuaJIT 高并发

web 服务实践教程知识点

第四章：Lua 语言基础

第一节：LuaJit 环境搭建以及 LuaJit 的 HelloWorld

1. Lua 和 LuaJit 的区别：

   Lua 非常高效，它运行得比许多其它脚本(如 Perl、Python、Ruby)都快，这点在第三方的独立测评中得到了证实。尽管如此，仍然会有人不满足，他们总觉得“嗯，还不够快!”。LuaJIT 就是一个为了再榨出一点速度的尝试，它利用 JIT 编译技术把 Lua 代码编译成本地机器码后交由 CPU 直接执行。LuaJIT 测评报告表明，在浮点运算、循环和协程的切换等方面它的加速效果比较显著，但如果程序大量依赖 C 编写的函数，那么运行速度便不会有什么改进。目前 LuaJIT 只支持 X86 CPU。

   LuaJIT 是采用 C 语言写的 Lua 的解释器。LuaJIT 被设计成全兼容标准 Lua 5.1，因此 LuaJIT 代码的语法和标准 Lua 的语法没多大区别。LuaJIT 和 Lua 的一个区别是，LuaJIT 的运行速度比标准 Lua 快数十倍，可以说是一个 Lua 的高效率版本。

   若无特殊说明，我们接下来的章节都是基于 LuaJIT 进行介绍的。

2. 下载安装：

   下载地址：http://luajit.org/download.html

   安装步骤：make && make install (linux 所需环境：gcc 4.x+)

3. LuaJit 的 HelloWorld 程序

第二节：Lua 基础数据类型及 Table 库简介

1. 基础数据类型

- **nil（空）**

  nil 是一种类型，Lua 将 nil 用于表示“无效值”。一个变量在第一次赋值前的默认值是 nil，将 nil 赋予给一个全局变量就等同于删除它。

- **boolean（布尔）**

  布尔类型，可选值 true/false；Lua 中 nil 和 false 为“假”，其它所有值均为“真”。比如 0 和空字符串就是“真”

- **number（数字）**

  Number 类型用于表示实数，和 C/C++ 里面的 double 类型很类似。可以使用数学函数 math.floor（向下取整）和 math.ceil（向上取整）进行取整操作。

- **string（字符串）**

  在 Lua 实现中，Lua 字符串一般都会经历一个“内化”（intern）的过程，即两个完全一样的 Lua 字符串在 Lua 虚拟机中只会存储一份。每一个 Lua 字符串在创建时都会插入到 Lua 虚拟机内部的一个全局的哈希表中。

- **function（函数）**

  在 Lua 中，**函数** 也是一种数据类型，函数可以存储在变量中，可以通过参数传递给其他函数，还可以作为其他函数的返回值。

- **table（表）**

  在内部实现上，table 通常实现为一个哈希表、一个数组、或者两者的混合。具体的实现为何种形式，动态依赖于具体的 table 的键分布特点

2. table 库简介

- 下标从 1 开始
- table.getn 获取长度
- table.concat (table [, sep [, i [, j]]])
- table.insert (table, [pos ,] value)
- table.maxn (table)
- table.remove (table [, pos])
- table.sort (table [, comp])

第三节：Lua 表达式、控制结构和函数

1. Lua 表达式
   - 算数表达式：+， －， *， /， ^，%
   - 关系运算符：<,>,<=,>=,==,~=
   - 逻辑运算符：and（逻辑与），or（逻辑或），not（逻辑非）
   - 字符串相连：..(两个点)，还可以用 string.format
   - 优先级：

2. Lua 的控制结构
   - if-else
   - while

```
while 表达式 do
--body
end
```

   - repeat

```
repeat
      print(x)
until true
```

   - for

for 语句有两种形式：数字 for 和范型 for

```
for var = begin, finish, step do
      --body
end

for i, v in ipairs(a) do
   print("index:", i, " value:", v)
end
```

   - break,return

3. Lua 函数

- 函数的定义

```
function function_name (arc)   -- arc 表示参数列表，函数的参数列表可以为空
    -- body
end
function_name = function (arc)
   -- body
end
```

- 函数的参数

a) 按值传递
b) 按引用传递
c) 变长参数
d) 具名参数

- 函数的返回值

Lua 具有一项与众不同的特性，允许函数返回多个值。Lua 的库函数中，有一些就是返回多个值。

```
local function swap(a, b)    -- 定义函数 swap，实现两个变量交换值
    return b, a                    -- 按相反顺序返回变量的值
end
```

第四节：Lua 高级

1. Lua 数组大小判断

   table.getn(t) 等价于 #t 但计算的是数组元素，不包括 hash 键值。而且数组是以第一个 nil 元素来判断数组结束。

   注意：一定不要使用 # 操作符或 table.getn 来计算包含 nil 的数组长度，这是一个未定义的操作，不一定报错，但不能保证结果如你所想。

   如果你要删除一个数组中的元素，请使用 remove 函数，而不是用 nil 赋值。

2. Lua 非空判断

   - 简单类型的变量，用 *if (var == nil) then* 来判断
   - 对于 table 类型的对象，使用：

```
function isTableEmpty(t)
    if t == nil or next(t) == nil then
        return true
    else
        return false
    end
end
```

3. Lua 模块和自定义模块

   - Require 函数加载模块
   - 自定义模块

     lua 执行 require 命令时的顺序是这样的：

     a) 先看是否存在环境变量里面是否存在 LUA_PATH，如果存在，以 LUA_PATH 作为 package.path 加载，不存在，则继续下一步

     b) 以编译时指定的默认路径来作为 package.path 初始化