关于视频课

- 目的：在实际项目中，我们学习的知识是如何被使用的
- 涉及到代码的部分，建议在 PC 上收看

极客时间

23.HTTP 客户端：lua-resty-requests

学习目的

- 如何找到高质量的 lua-resty-* 库
- lua-resty-requests 库的使用
- 如何阅读 lua-resty-*库

如何找到高质量的 HTTP 客户端?

- 官方包中没有 HTTP 客户端库
- 从 awesome-resty 中寻找，至少有 8 个第三方库
- 考虑因素：作者、测试覆盖、star 数、项目活跃、接口封装

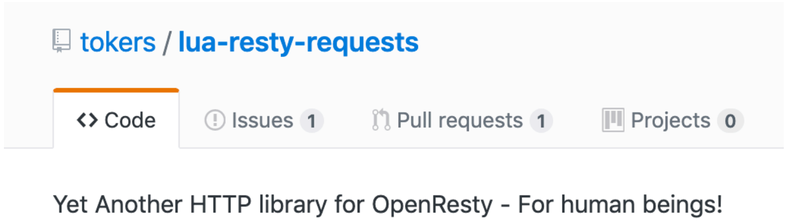
Networking

- lua-resty-http by @pintsized — Lua HTTP client cosocket driver for OpenResty / ngx_lua
- lua-resty-http by @liseen — Lua http client driver for the ngx_lua based on the cosocket API
- lua-resty-http by @DorianGray — Lua HTTP client driver for ngx_lua based on the cosocket API
- lua-resty-http-simple — Simple Lua HTTP client driver for ngx_lua
- lua-resty-httpipe — Lua HTTP client cosocket driver for OpenResty / ngx_lua
- lua-resty-httpclient — Nonblocking Lua HTTP Client library for aLiLua & ngx_lua
- lua-httpcli-resty — Lua HTTP client module for OpenResty
- lua-resty-requests — Yet Another HTTP Library for OpenResty

我为什么选择它?

极客时间

- 作者 tokers，OpenResty 的积极贡献者，代码有保证
- 喜欢的接口风格，人类友好，类似 Python 的 Requests 库
- 社区活跃



1. 安装并运行示例代码

- 通过 LuaRocks、OPM 或者手工安装，建议手工安装
- 使用 resty 运行

```
resty -e 'print(require "resty.requests".get{ url = "https://github.com", stream = false }.content)'
```

- GET 请求的三种写法：

```
opts = {stream = false}

requests.request("GET", url, opts)

requests.get(url, opts)

requests.get {url = url, stream = false}
```

缩写的形式，省略了opt以及括号。
为了可读性，最好还是加上括号

2. 项目结构

.ci	配合 travis CI，安装 OpenResty、luarocks 等
lib/resty	lua-resty-*库的统一代码目录
t	测试案例目录
util	OpenResty 官方提供的 luareleng 代码测试工具
.gitattributes	
.gitignore	
.luacheckrc	代码检测工具 luacheck 配置文件
.travis.yml	集成 github 内置的 CI
CHANGELOG.markdown	
LICENSE	
Makefile	调用 luareleng、luacheck 代码检测工具，执行测试案例等
README.markdown	
dist.ini	OPM 包描述文件
lua-resty-requests-0.7.2-1.rockspec	LuaRocks 包描述文件

3. 结合测试案例看文档

- 以 get 函数为例，它的测试案例集在“可用性测试”中

get

syntax: *local r, err = requests.get(url, opts?)*
syntax: *local r, err = requests.get { url = url, ... }*

Sends a HTTP GET request. This is identical with

```
requests.request("GET", url, opts)
```

Branch: **master** ▾ **lua-resty-requests** / t / **00-sanity.t**

```
250 === TEST 3: normal GET request with body
251
252 --- http_config eval: $::http_config
253 --- config
254 location /t1 {
255     content_by_lua_block {
256         local req_data = "你好吗? Hello?"
257         local requests = require "resty.requests"
258         local url = "http://127.0.0.1:10088/t3?usebody=true&af=b"
259         local headers = {
260             ["content-length"] = #req_data
261         }
262
263         local opts = {
264             headers = headers,
265             body = req_data
266         }
267
268         local r, err = requests.get(url, opts)
269         if not r then
270             ngx.log(ngx.ERR, err)
271         end
272
273         ngx.print(r:body())
274     }
275 }
```

先看标题

body 支持 string、table 和 function，
都可以在测试案例中找到

再找 get 函数，然后梳理

get 的实现

```lua
25  local function request(method, url, opts)
26      if not url and is_tab(method) then
27          -- shortcut type
28          return request_shortcut(nil, method)
29      end
30
31      local s = session.new()
32      return s:request(method, url, opts)
33  end
34
35
36  local function get(url, opts)
37      if not opts and is_tab(url) then
38          -- shortcut type
39          return request_shortcut("GET", url)
40      end
41
42      return request("GET", url, opts)
43  end
```

最终的发送请求函数

花括号调用的处理

```lua
requests.get {url = url, stream = false}
```

工整的 _M 导出

左侧是 lua-resty-lru 的 _M 导出，
散布在代码之间，比较凌乱

```
function _M.new(size)

function _M.get(self, key)
```

此项目在文件的最后集中导出，很直观

```
_M.request = request
_M.get = get
_M.head = head
_M.post = post
_M.put = put
_M.delete = delete
_M.options = options
_M.patch = patch
_M.state = state
_M.session = session.new

local STATE = util.STATE
for k, v in pairs(STATE) do
    _M[k] = v
end

return _M
```

get 函数的导出

总结

- 从作者、活跃度、测试案例、文档等方面选择开源项目
- 看项目不要陷入细节，先看整体结构，再从代码脉络着手
- 和其他项目作对比

极客时间

Q&A