【Openresty】Nginx Lua的 執行階段

nginx 執行步驟

```
1、post-read
□取□求□容□段，nginx□取并解析完□求头之后就立即□始运行；例如模块 ngx_realip 就在 post-read □段注□了处理程序，它的功能是迫使 Nginx □□□前□求的□源地址是指定的某一个□求头的值。

2、server-rewrite
server□求地址重□□段；□ngx_rewrite模块的set配置指令直接□□在server配置块中□，基本上都是运行在server-rewrite □段

3、find-config
配置查找□段，□个□段并不支持Nginx模块注□处理程序，而是由Nginx核心□完成□前□求与location配置块之间的配□工作。


4、rewrite
location□求地址重□□段，□ngx_rewrite指令用于location中，就是再□个□段运行的；另外ngx_set_misc(设置md5、encode_base64等)模块的指令，□有ngx_lua模块的set_by_lua指令和rewrite_by_lua指令也在此□段。

5、post-rewrite
□求地址重□提交□段，□nginx完成rewrite□段所要求的□部跳□动作，如果rewrite□段有□个要求的□；

6、preaccess
□□□限□查准备□段，ngx_limit_req和ngx_limit_zone在□个□段运行，ngx_limit_req可以控制□求的□□□率，ngx_limit_zone可以控制□□的并发度；

7、access
□□□限□查□段，□准模块ngx_access、第三方模块ngx_auth_request以及第三方模块ngx_lua的access_by_lua指令就运行在□个□段。配置指令多是□行□□控制相□的任务，如□查用户的□□□限，□查用户的□源IP是否合法；

8、post-access
□□□限□查提交□段；主要用于配合access□段实□□准ngx_http_core模块提供的配置指令satisfy的功能。satisfy all(与□系),satisfy any(或□系)

9、try-files
配置□try_files处理□段；□门用于实□□准配置指令try_files的功能,如果前 N-1 个参□所□□的文件系统□象都不存在，try-files □段就会立即发起"□部跳□"到最后一个参□(即第 N 个参□)所指定的URI.

10、content
□容□生□段，是所有□求处理□段中最□重要的□段，因□□个□段的指令通常是用□生成HTTP响□□容并输出 HTTP 响□的使命；

11、log
日志模块处理□段；□□日志
```

```
#lua 模組使用 nginx 區塊
init_by_lua            http
set_by_lua             server, server if, location, location if
rewrite_by_lua         http, server, location, location if
access_by_lua          http, server, location, location if
content_by_lua         location, location if
header_filter_by_lua   http, server, location, location if
body_filter_by_lua     http, server, location, location if
log_by_lua             http, server, location, location if
timer
```

```
#config 區塊 基本配置
...                 # 全域性區塊

event{              # events 區塊
      ...
}
http{               # http 區塊

   init_by_lua_block{
```

```
    }

    init_by_lua_file /usr/local/nginx/conf/lua/init.lua

    #常見的功能是執行定時任務 or health_check
    #此階段一般用來進行權限檢查和黑白名單配置
    init_worker_by_lua_block{

   }

    #配置環境：**http，server，location，location if
   access_by_lua_block{}

   #配置環境**http，server，location，location if#
    #常用於header進行添加、刪除等操作
    header_filter_by_lua_block{}

    server{           # server 區塊
       location /aaa {     # location 區塊
        set $b '';
       #  設定變量
          set_by_lua_block $a {
             local t = 'test'
             ngx.var.b = 'test_b'
             return t
          }
          return 200 $a,$b; # test,test_b
          }

       location /bbb {
         set $b  '1';
        #rewrite_by_lua_block始終都在rewrite階段的後面執行
         rewrite_by_lua_block { ngx.var.b = '2'}
         set $b  '3';
           echo $b; #2
         }
       }

    location /ccc{
      #**配置環境：**location，location if
      #content_by_lua_block指令不可以和其他內容處理階段的模塊如echo、return、proxy_pass等放在一起
      content_by_lua_block{}
    }

       location /ddd{
          #content_by_lua_file可以直接□取URL中参□file_name的值
          content_by_lua_file conf/lua/$arg_file_name;
       }

       location /eee{
      #□响□体全部□□□大□
      #**配置環境：**http，server，location，location if
      body_filter_by_lua_block { ngx.arg[1] = string.upper(ngx. arg[1]) }
       }
    }
 }
```

https://blog.51cto.com/xikder/2331649

⏲修訂版本 #3
★由 treeman 建立於 4 ◆G◆◆ 2023 01:14:06
✎由 treeman 更新於 5 ◆◆◆◆ 2023 10:14:59