openresty / **lua-resty-upload**

Streaming reader and parser for http file uploading based on ngx_lua cosocket

59 commits **1** branch **9** releases **3** contributors

branch: **master** **lua-resty-upload** / +

suppressed a false positive in libdl.

agentzh authored on Apr 1 latest commit eaf2ec39c2

lib/resty	fixed the version number in the code. it should be 0.09.	8 months ago
t	use Test::Nginx::Socket::Lua instead of Test::Nginx::Socket in our te…	9 months ago
.gitignore	initial checkin.	3 years ago
Makefile	updated Makefile.	3 years ago
README.markdown	"compute" edit	10 months ago
valgrind.suppress	suppressed a false positive in libdl.	5 months ago

README.markdown

Name

lua-resty-upload - Streaming reader and parser for HTTP file uploading based on ngx_lua cosocket

Table of Contents

Status

This library is considered production ready.

Description

This Lua library is a streaming file uploading API for the ngx_lua nginx module:

http://wiki.nginx.org/HttpLuaModule

The multipart/form-data MIME type is supported.

The API of this library just returns tokens one by one. The user just needs to call the `read` method repeatedly until a nil token type is returned. For each token returned from the `read` method, just check the first return value for the current token type. The token type can be `header`, `body`, and `part end`. Each `multipart/form-data` form field parsed consists of several `header` tokens holding each field header, several `body` tokens holding each body data chunk, and a `part end` flag indicating the field end.

This is how streaming reading works. Even for giga bytes of file data input, the memory used in the lua land can be small and constant, as long as the user does not accumulate the input data chunks herself.

This Lua library takes advantage of ngx_lua's cosocket API, which ensures 100% nonblocking behavior.

Note that at least ngx_lua 0.7.9 or ngx_openresty 1.2.4.14 is required.

Synopsis

```
lua_package_path "/path/to/lua-resty-redis/lib/?.lua;;";

server {
    location /test {
        content_by_lua '
            local upload = require "resty.upload"
            local cjson = require "cjson"

            local chunk_size = 5 -- should be set to 4096 or 8192
                                 -- for real-world settings

            local form, err = upload:new(chunk_size)
            if not form then
                ngx.log(ngx.ERR, "failed to new upload: ", err)
                ngx.exit(500)
            end

            form:set_timeout(1000) -- 1 sec

            while true do
                local typ, res, err = form:read()
                if not typ then
                    ngx.say("failed to read: ", err)
                    return
                end

                ngx.say("read: ", cjson.encode({typ, res}))

                if typ == "eof" then
                    break
                end
            end

            local typ, res, err = form:read()
            ngx.say("read: ", cjson.encode({typ, res}))
        ';
    }
}
```

A typical output of the /test location defined above is:

```
read: ["header",["Content-Disposition","form-data; name=\"file1\"; filename=\"a.txt\"",
```

```
read: ["header",["Content-Type","text\/plain","Content-Type: text\/plain"]]
read: ["body","Hello"]
read: ["body",", wor"]
read: ["body","ld"]
read: ["part_end"]
read: ["header",["Content-Disposition","form-data; name=\"test\"","Content-Disposition:
read: ["body","value"]
read: ["body","\r\n"]
read: ["part_end"]
read: ["eof"]
read: ["eof"]
```

You can use the lua-resty-string library to compute SHA-1 and MD5 digest of the file data incrementally. Here is such an example:

```
local resty_sha1 = require "resty.sha1"
local upload = require "resty.upload"

local chunk_size = 4096
local form = upload:new(chunk_size)
local sha1 = resty_sha1:new()
local file
while true do
    local typ, res, err = form:read()

    if not typ then
         ngx.say("failed to read: ", err)
         return
    end

    if typ == "header" then
        local file_name = my_get_file_name(res)
        if file_name then
            file = io.open(file_name, "w+")
            if not file then
                ngx.say("failed to open file ", file_name)
                return
            end
        end

     elseif typ == "body" then
        if file then
            file:write(res)
            sha1:update(res)
        end

    elseif typ == "part_end" then
        file:close()
        file = nil
        local sha1_sum = sha1:final()
        sha1:reset()
        my_save_sha1_sum(sha1_sum)

    elseif typ == "eof" then
        break

    else
        -- do nothing
    end
end
```

If you want to compute MD5 sums for the uploaded files, just use the resty.md5 module shipped by the lua-resty-string library. It has a similar API as resty.sha1.

For big file uploading, it is important not to buffer all the data in memory. That is, you should never accumulate data chunks either in a huge Lua string or in a huge Lua table. You must write the data chunk into files as soon as possible and throw away the data chunk immediately (to let the Lua GC free it up).

Instead of writing the data chunk into files (as shown in the example above), you can also write the data chunks to upstream cosocket connections if you do not want to save the data on local file systems.

Back to TOC

Author

Yichun "agentzh" Zhang (章亦春) agentzh@gmail.com, CloudFlare Inc.

Back to TOC

Copyright and License

Back to TOC

See Also

- the ngx_lua module
- the lua-resty-string library
- the lua-resty-memcached library

- the lua-resty-redis library
- the lua-resty-mysql library

Back to TOC