Documentation for Project Framework

Kacper Gawroński

October 21, 2018

1 Introduction

Source files can be found on https://github.com/KacperGawronski/project_framework
.
Project started as example usage of socket.h.
Next step was implementing handling of http request using Lua scripting language. When it was done, I noticed that it can script page generation structure and handle all requests. Project is on GNU GPL license, however, some external libraries might be on other licenses.

2 Dependencies

Basic project require:

- POSIX compiliant system Basically project is targeted on GNU/Linux OS - Debian.
- Lua5.3 libraries and headers

Example worker also require:

- MariaDB dev files (C connector)
- jansson library and header files for database api

To run example project it is also required to have MariaDB installed, with example database from:
https://github.com/datacharmer/test_db
installation process can be found on:
https://www.ibm.com/developerworks/library/l-lpic1-105-3/index.html

3 Structure

3.1 webserver.c

webserver.c contains code that allow handling connections. It makes listening socket (by default on port 9090), and forwards accepted connections to threads based on worker function. It passes struct stack_element, taken from initialized stack, containing all required data to process connection. When connection is done, structure is pushed back on the stack by thread for further use.
By default, threre is limitation 100 for number of threads, it's controlled by semaphore.

3.2 worker.c

function worker() handles connections. It reads sent data (by default only 10240 bytes), initializes Lua library as interpreter, and pushes on it additional C functions.
Example of adding C function to Lua scripting level:

```
lua_pushcfunction(L,generate_menu);
lua_setglobal(L,"generate_menu");
```

Function should be defined as:

```
int name(lua_State * L){ (...) return number_of_returned_values }
```

Example functions are placed in menu.c and mariadb_connector.c files.

3.3 Page structure

Example page is described in following files:

- example.lua
  Placed in app/pages directory. File contains page description - header definitions, javascript files to use and body
- example.js
  Placed in app/javascript directory. File contains javascript which is run on page load.
- style.css
  Global css file placed in app/css directory.
- api.lua
  File describing json api, as for api.json?[smth] requests.
- page_template.lua File placed in app directory. It is used for processing every lua file in app/pages directory. It contains page structure definition.

Generally

- worker directory contains basic files for processing and handling requests - especially forwarding them to Lua interpreter.
- app directory contains definitions of what should be done with request.
- app/GET.lua Main file for processing GET request. It describes actions taken on specific GET requests. In this project, for example, it does file app/pages/filename.lua for request like "/page?filename"
- app/page_template.lua returns function which should be done by every page using definied table as argument.
- app/pages directory contains files describing pages, in format definied by page_template.lua code.
- app/pages/pagename.lua is file describing pagename site. It should define table as used by page_template.lua. Link to it will be generated by generate_menu() function as page?pagename . To add link in html you should simple write page?filename without extension (it will be added in code, prevents a little from hacking).
- app/javascript is main directory for javascript .js files. It won't automatically include js for every page, it needs to be definied in page file.
- app/css directory contains css files, and description of files included in every page in requires.txt.
- json.api contains file api.lua, which is called on GET /api.json?[smth] request. It is example of additional aplication.