GL.iNet Software development

process and specification

2018.08.14

Author: Lancer Luo

Table of Contents

[1 Create a code repository 3](#_Toc522028516)

[2 Create your own project 3](#_Toc522028517)

[2.1 Reference 3](#_Toc522028518)

[2.1 API mapping and make entry 3](#_Toc522028519)

[2.3 Generating dynamic libraries 3](#_Toc522028520)

[3 Code comment specification 4](#_Toc522028521)

[4 Debug 5](#_Toc522028522)

[4.1 Unit test 5](#_Toc522028523)

[4.2 CGI debugging 5](#_Toc522028524)

[5 Generate apiDoc 6](#_Toc522028525)

# Create a code repository

Please contact **AIfie** to create a code repository in GL Dev groups.

# Create your own project

## 2.1 Reference

Please refer to the demo [git@gitlab.com:luoyejiang/gl-module-demo.git](mailto:git@gitlab.com:luoyejiang/gl-module-demo.git) to do it.

## 2.1 API mapping and make entry

|  |
| --- |
| int demo\_hello(json\_object\* input, json\_object\* output) /\* Function definition \*/  {  gjson\_add\_string(output, "msg", "Hello World");  return 0;  }  /\*\* The implementation of the GetAPIFunctions function \*\*/  **#include <gl/glapibase.h>**  static api\_info\_t gl\_lstCgiApiFuctionInfo[] = { /\* api mapping \*/  map("/demo/hello", "get", demo\_hello),  };  api\_info\_t\* get\_api\_entity(int\* pLen) /\* api entry \*/  {  (\*pLen) = sizeof(gl\_lstCgiApiFuctionInfo) / sizeof(gl\_lstCgiApiFuctionInfo[0]);  return gl\_lstCgiApiFuctionInfo;  } |

This is a template. You just need to add new api mapping and implementation function definition.

## 2.3 Generating dynamic libraries

Write a Makefile to compile your source code. Please refer to gl-module-demo/src/Makefile.

|  |
| --- |
| CC=gcc  CFLAGS+=  LDFLAGS+= -lglutil -ljson-c  all: libraries demotest  @echo "compile done"  **libraries:**  **$(CC) $(CFLAGS) -Wall demo.c -shared -fPIC -o libgldemo.so $(LDFLAGS)**  demotest: unittest.o  $(CC) $^ -o $@ $(LDFLAGS) -L. -lgldemo  clean:  rm $(PROC) $(OBJS)  .PHONY: all clean |

After compile, you can find a dynamic library in

*build\_dir/target-mips\_24kc\_musl/gl-module-demo/libgldemo.so*,

and a ipk package

*bin/packages/mips\_24kc/base/gl-module-demo\_1\_mips\_24kc.ipk*

# 3 Code comment specification

|  |
| --- |
| /\*\***demo\_hello**  \* @api {GET} /demo/hello /demo/hello  \* @apiGroup demo  \* @apiVersion 1.0.0  \* @apiDescription **Demo for comment.**  \* @apiParam {String} msg **Show message.**  \* @apiParamExample {json} Request  \* $ curl --cookie "QSESSIONID=9177564b1203c30000" 192.168.8.1/api/demo/hello  \* @apiSuccess (200) {int} code 0 is success and occur error when code < 0. <a href="#api-ErrorCode-DescriptionCode">Detailed...</a>  \* @apiSuccess (200) {int} -1 Invalid user, permission denied or not logged in!  \* @apiSuccess (200) {int} -4 Invalid parameter, value or format!  \* @apiSuccess (200) {int} -5 No parameter found!  \* @apiSuccessExample {json} Response  \* {"code":0,"msg":"Hello World","http\_cookie":"QSESSIONID=9177564b1203c30000"}  \*/  int **demo\_hello**(json\_object\* input, json\_object\* output)  {  gjson\_add\_string(output, "msg", "Hello World");  return 0;  } |

Red coloured words：Function name.

Brown coloured words：First level path as apiGroup.

Blue coloured words：Possible error code and description.

Warning：Please use uppercase for initials in description.

# 4 Debug

## 4.1 Unit test

To make sure your program is viable. You need to do unit testing yourself.

Below is a simple test code:

|  |
| --- |
| #include <stdio.h>  #include <string.h>  #include <stdlib.h>  #include <gl/gjson.h>  #include "demo.h"  int main(int argc , char \*argv[])  {  json\_object\* input = json\_object\_new\_object();  json\_object\* output = json\_object\_new\_object();    demo\_hello(input, output);  printf("result:\n%s\n", json\_object\_to\_json\_string(output));    json\_object\_put(input);  json\_object\_put(output);  return 0;  } |

Upload *gl-module-demo\_1\_mips\_24kc.ipk* to the router */tmp* directory via SSH,

|  |
| --- |
| root@GL-AR750S:/tmp# ls -l gl-module-demo\_1\_mips\_24kc.ipk  -rw-r--r-- 1 root root 2953 Aug 14 2018 gl-module-demo\_1\_mips\_24kc.ipk |

Install,

|  |
| --- |
| root@GL-AR750S:/tmp# opkg install gl-module-demo\_1\_mips\_24kc.ipk  Installing gl-module-demo (1) to root...  Configuring gl-module-demo. |

Execution unit test program,

|  |
| --- |
| root@GL-AR750S:/tmp# demotest  result:  { "msg": "Hello World" } |

OK.

## 4.2 CGI debugging

Login if needed token,

|  |
| --- |
| $ curl --cookie "QSESSIONID=9177564b1203c30000" 192.168.8.1/api/router/login -d "pwd=goodlife1" -v  \* Trying 192.168.8.1...  \* Connected to 192.168.8.1 (192.168.8.1) port 80 (#0)  > POST /api/router/login HTTP/1.1  > Host: 192.168.8.1  > User-Agent: curl/7.47.0  > Accept: \*/\*  > Cookie: QSESSIONID=9177564b1203c30000  > Content-Length: 13  > Content-Type: application/x-www-form-urlencoded  >  \* upload completely sent off: 13 out of 13 bytes  < HTTP/1.1 200 OK  < Set-Cookie: QSESSIONID=33bf61b82b04660000; path=/  < Content-Type: application/json  < Content-Length: 99  < Date: Mon, 13 Aug 2018 16:56:11 GMT  < Server: lighttpd/1.4.48  <  \* Connection #0 to host 192.168.8.1 left intact  {"code":0,"token":"u2bnHJhT6aMWbTYrlBmgMgcchj7QZUB5","http\_cookie":"QSESSIONID=9177564b1203c30000"} |

Request,

|  |
| --- |
| $ curl -H "Authorization: u2bnHJhT6aMWbTYrlBmgMgcchj7QZUB5" \  --cookie "QSESSIONID=33bf61b82b04660000" 192.168.8.1/api/demo/hello |

Response,

|  |
| --- |
| {  "code": 0,  "msg": "Hello World",  "http\_cookie": "QSESSIONID=33bf61b82b04660000"  } |

OK.

# 5 Generate apiDoc

Copy *gl-module-demo/apitest.py* to your project root directory, and excuting *python apitest.py*.

A folder named *apiDoc* is automatically created in your project root directory.

|  |
| --- |
| $ tree -L 1  .  ├── apiDoc  ├── apiDoc.js  ├── apidoc.json  ├── apitest.py  ├── Error.txt  ├── Makefile  └── src |

You can open *apiDoc/index.html* via browser.

The effect is shown below,

