后端开发笔试题

在这个笔试题中，应聘者需要使用Golang / C / C ++（若都不熟悉上述语言，允许使用自己最擅长的语言）编写一个无需考虑实用性的SimpleHTTPServer（没有特别要求的框架），以及相应的“README文档”（文件格式可以是.txt | .md | 等）。

在提交你的答案之前，请使用你熟悉的HttpClient进行测试，例如：Postman工具，或者你自定义的CLI HttpClient脚本（推荐）。

笔试题最长提交时间为拿到题目后一周，越早提交则会获得更高的分数。笔试题答案&简历投递邮箱：dev@lokcol.com。

以下为题目正文部分，为了保证题目意思表述的正确性，题目和解释将以英语进行呈现。

----------Assignment description start----------

The SimpleHTTPServer should serve the following http/1.0 APIs.

----------#1 API----------

Request

method	GET
Parameterized Http Route	/v{:version}/test-api
Requested Http Route	/v3/test-api
query in http URL	N/A
Content-Type	application/x-www-form-urlencoded
body	N/A

Response

Content-Type	application/json
body	{ret: 1000}

----------#2 API----------

Request

method	GET
Parameterized Http Route	/v{:version}/arithmetic/{:action}
Requested Http Route	/v3/plus?a=3&b=4
query in http URL	N/A
Content-Type	application/x-www-form-urlencoded
body	N/A

Response

Content-Type	application/json
body	{ret: 1000, version: 3, action: plus, result: 7}

----------#3 API----------

Request

method	POST
Parameterized Http Route	/v{:version}/arithmetic/{:action}
Requested Http Route	/v3/plus
query in http URL	N/A
Content-Type	application/x-www-form-urlencoded
body	{a: 3, b: 4}

Response

Content-Type	application/json
body	{ret: 1000, version: 3, action: plus, result: 7}

----------#4 API----------

Request

method	GET
Parameterized Http Route	/{:namespace}/{:resource}/{:action}
Requested Http Route	/tutorial/student/list

query in http URL	N/A
Content-Type	application/x-www-form-urlencoded
body	N/A

Response

Content-Type	text/html
body	<html> <head> <meta>...</meta> </head> <body> ... </body> </html>

Your response body should be translated from a "Html Template Engine", and look like the following which merely provides a list of brief introduction to students from whatever storage, but preferrably a MySQL/PostgresSQL server.

```
<html>
  <head>
    <title>Student List</title>
    <meta>...</meta>
  </head>
  <body>
    <table style="width:100%">
      <tr>
        <th>Firstname</th>
        <th>Lastname</th>
        <th>Age</th>
      </tr>
      <tr>
        <!-- Brief intro of student #1. -->
        ...
      </tr>
      <tr>
        <!-- Brief intro of student #2. -->
        ...
      </tr>
      ...
      <tr>
        <!-- Brief intro of student #N. -->
        ...
      </tr>
  </body>
</html>

```

In this last item, you'll be required to use a SQL compliant server to hold some persistent storage data for implementing an authentication middleware by the so-called "AccessToken". Prepare 2 tables with the data specified as follows, and choose the proper data types by yourself.

```
user
==
{
  id: sp100029,
  name: peter,
  age: 25
}
{
  id: sp100030,
  name: tom,
  age: 50
}
{
  id: sp100031,
  name: karl,
  age: 34
}
{
  id: sp100032,
  name: mary,
  age: 18
}
```

(we understand that the following "table" should sometimes be put into a "VolatileStorage", e.g. Redis or Memcache, for speeding, but it's not the requirement here)

```
int_auth_token_cache
==
{
  int_auth_token: xxxyyyzzz,
  id: sp100029,
  device: "iPhone6",
  ip: "192.168.1.88"
}
{
  int_auth_token: aaabbbccc,
  id: sp100029,
  device: "Samsung Galaxy S3",
  ip: "177.15.33.8"
}
{
  int_auth_token: pppqqqsss,
  id: sp100030,
```

```
  device: "Samsung Note Bomb",
  ip: "192.168.7.55"
}
{
  int_auth_token: dddeeefff,
  id: sp100031,
  device: "Xiaomi 5X",
  ip: "192.168.7.58"
}
{
  int_auth_token: eeefffggg,
  id: sp100031,
  device: "Xiaomi 4",
  ip: "111.20.3.7"
}
{
  intAuthToken: yuqbajnnr,
  id: sp100032,
  device: "iPhone SE",
  ip: "121.2.88.137"
}
```

Then implement an API aligned with the following request/response specs.

Request (Not Authenticated)

method	POST
Parameterized Http Route	/user/{:userId}/wallet/self/detail
Requested Http Route	/user/sp100032/wallet/self/detail
query in http URL	N/A
Content-Type	application/x-www-form-urlencoded

body	{
	
	}

Response (Not Authenticated)

Content-Type	application/json
body	{ret: 1001}

Request (Authenticated)

method	POST
Parameterized Http Route	/user/{:userId}/wallet/self/detail

Requested Http Route	/user/sp100032/wallet/self/detail
query in http URL	N/A
Content-Type	application/x-www-form-urlencoded

body	{
	  intAuthToken: yuqbajnnr
	}

Response (Authenticated)

Content-Type	text/html

body	<html>
	  <head>
	  mary
	  </head>
	  <body>
	  Welcome "iPhone SE" user from "121.2.88.137"!
	  </body>
	</html>

Please submit your codes and documents in single ".zip" file whose top "README" file tells us at least how to start/check/stop your server process(es), e.g. what bash commands or scripts to execute with what params. The email address is dev@lokcol.com.

- Demonstrating the technique for appropriate separation of production/test environments is a plus, e.g. by injecting runtime variables into your server process(es).
- Demonstrating the techinque for appropriate handling of RaceCondition/ThreadSafety issues is a plus, e.g. by using mutex locks or SQL ordered intra-transaction statements "correctly".
- Demonstrating the techinque for appropriate use of SQL schema sync is a plus, e.g. by simply mysql dump/source scripts, or dedicated tools like gh-ost, skeema or SchemaSync.
- Demonstrating the techinque for appropriate use of recursive code style formatting tools is a plus, e.g. by tools that help the automation like "gofmt" for Golang.
- Demonstrating the techinque for appropriate use of version control software such as git, mercurial or svn is a plus, e.g. by adding corresponding ".gitignore" or ".hgignore" files.
- Consider that we need several other APIs similar to "/user/{:userId}/wallet/self/detail" which requries a preprocessing of "IntAuthToken" and makes use of the corresponding "userId" to proceed (with later database CRUD access), e.g. "/user/{:userId}/mailbox/list/read", is your code ready for easily adding/removing such APIs?

----------Assignment description end----------

若你看完题目后希望能得到进一步的提示，可以看一个例题示范，但以下示范只是提供其中一个实例来帮助你理解题目，我们并不希望你要严格按照下述的实现方法来完成上述的题目。

例题示范

Request

method	GET
Parameterized Http Route	/user/:name
Requested Http Route	/user/test
query in http URL	N/A
Content-Type	application/x-www-form-urlencoded
body	N/A

Response

Content-Type	test/plain
body	Hello, test

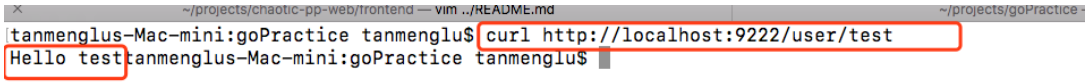
1. 在本地搭建Http Server。本题以Gin Web Framework(written in GO) 为例，在本地搭建了一个监听9992端口的Http Server

```
tanmenglus-Mac-mini:goPractice tanmenglu$ go run main.go
[GIN-debug] [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

[GIN-debug] [WARNING] Running in "debug" mode. Switch to "release" mode in production.
 - using env:   export GIN_MODE=release
 - using code:  gin.SetMode(gin.ReleaseMode)

[GIN-debug] GET    /user/:name               --> main.setupRouter.func1 (3 handlers)
[GIN-debug] GET    /v:version/test-api       --> main.setupRouter.func2 (3 handlers)
[GIN-debug] GET    /v:version/arithmetic/:action --> main.setupRouter.func3 (3 handlers)
[GIN-debug] Listening and serving HTTP on :9222
[GIN] 2018/11/30 - 18:17:39 | 200 |     531.239µs |             ::1 | GET      /user/test
[GIN] 2018/11/30 - 18:17:48 | 200 |      49.576µs |             ::1 | GET      /user/test
```

2. 根据题目需求配置对应的HttpRouter。

```go
// This handler will match /user/john but will not match /user/ or /user
router.GET("/user/:name", func(c *gin.Context) {
        name := c.Param("name")
        c.String(http.StatusOK, "Hello %s", name)
})
```

3. 成功运行服务器后，应能使用局域网或本地请求(in GET method) http://localhost:port/RequestedHttpRoute



洛克互娱公司简介

深圳市洛克互娱科技有限公司是一家为社交而生的技术研发商与创新实验团队，专注开发基于移动端的社交游戏以及社交工具。我们以探造社交乐趣为所有产品的价值理念核心，并为之提倡“创造极致，尊重真实”的人文工程师团队文化。

我们既提倡“解决问题”的技术思维，又鼓励“想多一点”的设计想象，希望通过优秀的玩法设计，融合社交场景中的有效需求，最终为每一位用户创造简单、纯粹、快乐的社交体验。我们将以，用户能够感知到我们对社交本质的思考、热爱与追求，为自身最大的成就感。

笔试题&简历投递邮箱：dev@lokcol.com

公司地址：深圳市南山区留学生创业大厦1期2305