Г

Go_Day04-1

Развилка из недоступного проекта

Обновление с 5
Администратор создал 5 месяцев назад

Имя	Последний коммит	Последнее обновление
образцы кода	Инициировать фиксацию	1 год назад
образцы данных	Инициировать фиксацию	1 год назад
наборы данных	Инициировать фиксацию	1 год назад
материалы	Обновление с 5	5 месяцев назад
разное	Инициировать фиксацию	1 год назад
источник	Инициировать фиксацию	1 год назад
.gitignore	Обновление с 3	1 год назад
ИЗМЕНЕНИЙ	Инициировать фиксацию	1 год назад
ЛИЦЕНЗИЯ	Инициировать фиксацию	1 год назад
README.md	Обновление с 4	6 месяцев назад

README.md

День 04. Учебный лагерь

Конфеты!

Содержание

Chapter I

General rules

- Your programs should not quit unexpectedly (giving an error on a valid input). If this happens, your project will be considered non functional and will receive a 0 during the evaluation.
- We encourage you to create test programs for your project even though this work won't have to be submitted and won't be graded. It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.

- Submit your work to your assigned git repository. Only the work in the git repository will be graded.
- If your code is using external dependencies, it should use Go Modules for managing them

Chapter II

Rules of the day

- You should only turn in `*.go` files and (in case of external dependencies) `go.mod + go.sum`
- Your code for this task should be buildable with just `go build`
- Even though it is required to not modify the C code, you'll still have to comment out `main()` function in it, otherwise the program won't compile (two entry points)

Chapter III

Intro

Mister Rogers is very sad. He's sitting at your office and mumbles "My whole business...How am I supposed to make people happy now?".

The story is as old as the world itself. This new client of yours started a new business selling candy all across this muddy town. At first, everything was perfect - several vending machines, 5 delicious kinds of candy and lines of children begging their parents to buy some for them. And then it was like a thunder, when someone broke into a data center and stole a server responsible for handling candy orders. Not only that, an old developer has gone missing, too! Coincidence? You don't think so.

You pour mister Rogers a glass of good old bourbon and start asking questions trying to get more details.

"Well, I don't know much. All our vending machines were selling the same set of candy, you know, here they are on the brochure" - he gives you the piece of colorful paper advertising five new amazing tastes:

```
Cool Eskimo: 10 cents
Apricot Aardvark: 15 cents
Natural Tiger: 17 cents
Dazzling        Elderberry: 21 cents
Yellow Rambutan: 23 cents
```

"That's some weird sounding names" - you say - "How do people even remember these things?" "Oh, that one's easy" - said Rogers - "We've been using abbreviations everywhere, including our source code, so it's CE, AA, NT and so on"

He sobs.

"But does this even matter now? My business is ruined anyway, all this is just nonsense now!"

"Please focus, mister Rogers" - you've seen guys behaving like this many times, this place isn't called "Gopher PI" for nothing - "Is there any detail you didn't mention yet?"

"You're right! I've almost forgot!" - he pulls a piece of paper out of the pocket and hands it over. - "The thief left a note!"

You look at the text written with marker on one side: "I can't eat any more candy!". This doesn't give you much. Then you turn over the sheet and...

"Okay, mister Rogers. The good news is, I now know for sure it was your ex-employee who stole the server. But not only that! Something tells me I can help you restore your business, too."

Chapter IV

Exercise 00: Catching the Fortune

Выяснилось, что вор использовал первый лист бумаги, который оказался у него на столе, и по счастливой случайности это была спецификация протокола между торговым автоматом и сервером. Это выглядело так:

```yaml
---
swagger: '2.0'
info:
  version: 1.0.0
  title: Candy Server
paths:
  /buy_candy:
    post:
      consumes:
        - application/json
      produces:
        - application/json
```

```
    parameters:
      - in: body
        name: order
        description: summary of the candy order
        schema:
          type: object
          required:
            - money
            - candyType
            - candyCount
          properties:
            money:
              description: amount of money put into vending machine
              type: integer
            candyType:
              description: kind of candy
              type: string
            candyCount:
              description: number of candy
              type: integer
    operationId: buyCandy
    responses:
      201:
        description: purchase succesful
        schema:
            type: object
            properties:
              thanks:
                type: string
              change:
                type: integer
      400:
        description: some error in input data
        schema:
            type: object
            properties:
              error:
                type: string
      402:
        description: not enough money
        schema:
            type: object
            properties:
              error:
                type: string
```

В ближайшие часы мистер Роджерс рассказал вам все подробности. Чтобы воссоздать сервер, вы должны использовать эту спецификацию для создания набора кода Go, который фактически реализует внутреннюю часть. Можно переписать все вручную, но в этом случае вор может уйти раньше, чем вы это сделаете, поэтому вам нужно сгенерировать код как можно скорее.

Каждый покупатель конфет вкладывает деньги, выбирает, какие конфеты купить и в каком количестве. Эти данные отправляются на сервер через HTTP и JSON, а затем:

1. Если сумма цен на конфеты (см. главу 1) меньше или равна сумме денег, которую покупатель дал машине, сервер отвечает HTTP 201 и возвращает JSON с двумя полями — «спасибо», говоря «Спасибо!» а «сдача» - это сумма сдачи, которую машина должна вернуть покупателю.
2. Если сумма больше предоставленной суммы денег, сервер отвечает HTTP 402 и сообщением об ошибке в формате JSON, в котором говорится: «Вам нужно больше денег на {сумма}!», где {сумма} — разница между предоставленной и ожидаемой.
3. Если клиент предоставил отрицательный candyCount или неверный candyType (помните - все пять типов конфет закодированы двумя буквами, поэтому это один из "CE", "AA", "NT", "DE" или "YR", во всех остальных случаях считаются недействительными), тогда сервер должен ответить 400 и ошибкой внутри JSON, описывающей, что пошло не так. На самом деле вы можете сделать это двумя разными способами — либо написать код вручную с этими проверками, либо изменить спецификацию Swagger выше, чтобы она охватывала эти случаи.

Помните — все данные как с клиента, так и с сервера должны быть в формате JSON, поэтому вы можете протестировать свой сервер, например, так:

```
curl -XPOST -H "Content-Type: application/json" -d '{"money": 20, "candyType": "AA", "candyCount": 1}' http://12

{"change":5,"thanks":"Thank you!"}
```

или

```
curl -XPOST -H "Content-Type: application/json" -d '{"money": 46, "candyType": "YR", "candyCount": 2}' http://12

{"change":0,"thanks":"Thank you!"}
```

Кроме того, вам не нужно самостоятельно следить за запасами различных видов конфет, просто считайте, что это делают сами машины. Просто подтвердите ввод пользователя и рассчитайте изменение.

Глава V

Упражнение 01: Закон и порядок

Ты лежишь и улыбаешься, чувствуя что-то, что казалось хорошо приготовленным. Мистер Роджерс, похоже, тоже немного расслабился. Но затем его лицо снова меняется.

«Я знаю, что мы уже заплатили за повышенную безопасность в нашем дата-центре», — сказал он немного задумчиво. - "...но что, если этот преступник решит провернуть какую-нибудь аферу с посредником ? Мой бизнес снова будет разрушен! Люди потеряют работу, а я разорюсь!"

«Полегче, добрый сэр», — говорите вы с ухмылкой. - "Думаю, у меня есть именно то, что тебе нужно."

So, you need to implement a certificate authentication for the server as well as a test client which will be able to query your API using a self-signed certificate and a local security authority to "verify" it on both sides.

You already have a server which supports TLS, but it is possible that you'll have to re-generate the code specifying an additional parameter, so it will be using use secure URLs.

Also, you'll need a local "certificate authority" to manage certificates. For our task minica seems like a good enough solution. There is a link to a really helpful video in last Chapter if you want to know more details about how Go works with secure connections.

So, because we're talking a full-blown mutual TLS authentication, you'll have to generate two cert/key pairs - one for the server and one for the client. Minica will also generate a CA file called `minica.pem` for you which you'll need to plug into your client somehow (your auto-generated server should already support specifying CA file as well as `key.pem` and `cert.pem` through command line parameters). Also, generating certificate may require you to use a domain instead of an IP address, so in examples below we will use "candy.tld". For it to work on a local machine you can put it into '/etc/hosts' file.

Keep in mind, that because you're using a custom local CA you likely won't be able to query your API using cURL, web browser or tool like Postman anymore without tuning.

Your test client should support flags '-k' (accepts two-letter abbreviation for the candy type), '-c' (count of candy to buy) and '-m' (amount of money you "gave to machine"). So, the "buying request" should look like this:

```
~$ ./candy-client -k AA -c 2 -m 50
Thank you! Your change is 20
```

Chapter VI

Exercise 02: Old Cow

In a few days mister Rogers finally calls you with some great news - the thief was apprehended and immediately confessed! But candy businessman also had a small request.

"You seem like you really do know your way around machines, don't ya? There is one last thing I'd ask you to do, basically nothing. Our customers prefer something funny instead of just plain 'thank you', so my nephew Patrick wrote a program that generates some weird animal saying things. I think it's written in C, but that's not a problem for you, isn't it? Please don't change the code, Patrick is still improving it!"

Oh boy. You look through your emails and notice one from mister Rogers with a code attached to it:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

unsigned int i;
unsigned int argscharcount = 0;

char *ask_cow(char phrase[]) {
  int phrase_len = strlen(phrase);
  char *buf = (char *)malloc(sizeof(char) * (160 + (phrase_len + 2) * 3));
  strcpy(buf, " ");

  for (i = 0; i < phrase_len + 2; ++i) {
```

```
    strcat(buf, "_");
  }

  strcat(buf, "\n< ");
  strcat(buf, phrase);
  strcat(buf, " ");
  strcat(buf, ">\n ");

  for (i = 0; i < phrase_len + 2; ++i) {
    strcat(buf, "-");
  }
  strcat(buf, "\n");
  strcat(buf, "        \\   ^__^\n");
  strcat(buf, "         \\  (oo)\\_______\n");
  strcat(buf, "            (__)\\       )\\/\\\n");
  strcat(buf, "                ||----w |\n");
  strcat(buf, "                ||     ||\n");
  return buf;
}

int main(int argc, char *argv[]) {
  for (i = 1; i < argc; ++i) {
    argscharcount += (strlen(argv[i]) + 1);
  }
  argscharcount = argscharcount + 1;

  char *phrase = (char *)malloc(sizeof(char) * argscharcount);
  strcpy(phrase, argv[1]);

  for (i = 2; i < argc; ++i) {
    strcat(phrase, " ");
    strcat(phrase, argv[i]);
  }
  char *cow = ask_cow(phrase);
  printf("%s", cow);
  free(phrase);
  free(cow);
  return 0;
}
```

Looks like you'll have to return an ASCII-powered cow as a text in "thanks" field in response. When querying by cURL it will look like this:

```
~$ curl -s --key cert/client/key.pem --cert cert/client/cert.pem --cacert cert/minica.pem -XPOST -H "Content-Typ
{"change":0,"thanks":" _____________\n< Thank you! >\n ------------\n        \\   ^__^\n         \\  (oo)\\_______
```

Apparently, all you need is to reuse this `ask_cow()` C function without rewriting it in your Go code.

«Иногда мне кажется, что я должен бросить эту детективную работу и просто пойти работать старшим инженером», — ворчите вы.

По крайней мере, вы, вероятно, должны получить столько конфет, сколько захотите взамен. Мол, на всю оставшуюся жизнь.

Глава VII

Чтение

Использование спецификации Безопасные соединения Оригинальное предложение