Road to Mercari

Gopher Dojo module 03

Summary: This document is the subject for the Gopher Dojo module 03 of the Road to Mercari @ 42 Tokyo.

Contents

Chapter I

Instructions

- Only this page will serve as reference; do not trust rumors.

- Watch out! This document could potentially change up to an hour before submission.

- These exercises are carefully laid out by order of difficulty - from easiest to hardest. We `will not` take into account a successfully completed harder exercise if an easier one is not perfectly functional.

- Make sure you have the appropriate permissions on your files and directories.

- You have to follow the submission procedures for every exercise.

- Your exercises will be checked and graded by your fellow classmates.

- You cannot leave any additional file in your directory than those specified in the subject.

- Got a question? Ask your peer on the right. Otherwise, try your peer on the left.

- Your reference guide is called `Google / man / the Internet / ...`.

- Examine the examples thoroughly. They could very well call for details that are not explicitly mentioned in the subject...

- If no other explicit information is displayed, you must use the latest versions of Go.

Chapter II

Foreword

https://gopherdojo.org/

Chapter III

Exercise 00 : Omikuji API

Exercise 00
HTTP API and JSON
Turn-in directory : *ex00/*
Files to turn in : *

Create an omikuji API that follows the following specifications.

- Your API endpoint should be hosted on port given by a command line argument.

- When the endpoint is called, your API should return a random omikuji and an HTTP 200 status code.

- The API should return the omikuji in JSON format.

- The `fortune` property is required, and must contain one of the following values: `"Dai-kichi"`, `"Kichi"`, `"Chuu-kichi"`, `"Sho-kichi"`, `"Sue-kichi"`, `"Kyo"`, `"Dai-kyo"`

- Aside from `fortune`, the returned JSON must contain at least one other property. All other properties and their values are optional.

- Example of json:

```
{
  "fortune": "Dai-kichi",
  "health": "You will fully recover, but stay attentive after you do.",
  "residence": "You will have good fortune with a new house.",
  "travel": "When traveling, you may find something to treasure.",
  "study": "Things will be better. It may be worth aiming for a school in a different area.",
  "love": "The person you are looking for is very close to you."
}
```

- Your omikuji can only return Dai-kichi during Shogatsu (1/1 - 1/3). (use the servers real time to implement this feature.)

- When you retrieve the actual time, it should be implemented so that the test is mockable.

- Your omikuji program should handle all the errors without crashing.

- Example of execution:

```
?> ./omikuji 4242 &
[1] 81634
?> curl localhost:4242
{"fortune":"Kyo","health":"You will partially recover, but stay attentive after you do.","residence
    ": "You will have good fortune with a new house.","travel":"When traveling, you may find
    something to treasure.","study":"Things will be better. It may be worth aiming for a school in
     a different area.","love":"The person you are looking for is very close to you."}
```

- Write a test for your http handler and also for self implemented package.

- Example of test execution:

```
?> go test -cover ./...
ok     _/XX/XX/ex00   0.385s coverage: 58.3% of statements
ok     _/XX/XX/ex00/omikuzi  0.239s coverage: 44.0% of statements
```

https://pkg.go.dev/testing

https://pkg.go.dev/net/http/httptest