Assignment – Erol Kazancli 09.09.2018

Installation

Unzip the bidding folder into the Home directory. Remove the outer folder. The topmost folder should be bidding in the home directory.

Set the gopath (to the project or a common place) with the command:
export GOPATH=$HOME/bidding/

Get the necessary packages into /bidding/src folder by running the following commands:

```
go get -u github.com/gorilla/mux
go get github.com/mattn/go-sqlite3
```

Set working directory as $HOME/bidding/src/bidtracking and run the following:
go run *.go

This should expose the port 8000 to the API.

Alternatively I prepared a dockerfile:

Set working directory as $HOME/bidding
Run the following commands:
docker build -t bidding .
docker run --rm -p 8000:8000 bidding

This will run the server on the port 8000

Database

I used sqlite package for golang. A database named bidtracking.db is created if it doesn't exist when the server is started. All the bids are recorded into a table named "bid" for simplicity purposes. This table has five columns: id, item, user, value, unit. The unit is the currency of the value but is currently not being used for comparisons of the bid.

HTTP Router

I used gorilla/mux package for http routing.

How to use API

Send a bid for a user:
http://localhost:8000/api/v1/sendBidForItem

example body:

```
{
	"item": "{itemname}",
	"user": "{username}",
	"value": {valueofthebid},
	"unit": "{unitofthebid}"
}
```

Get all the bids for a user:
http://localhost:8000/api/v1/getBidsOfUser/{username}

example output:

[{"item":"chair","user":"erol","value":5.5,"unit":"dollar"},{"item":"chair","user":"erol","value":7,"unit":"dollar"},{"item":"chair","user":"erol","value":8,"unit":"dollar"},{"item":"table","user":"erol","value":10,"unit":"dollar"}]

Get all the bids for an item:
http://localhost:8000/api/v1/getBidsForItem/{itemname}

example output:

[{"item":"chair","user":"erol","value":5.5,"unit":"dollar"},{"item":"chair","user":"erol","value":7,"unit":"dollar"},{"item":"chair","user":"joe","value":6,"unit":"dollar"},{"item":"chair","user":"erol","value":8,"unit":"dollar"},{"item":"chair","user":"erol","value":10,"unit":"dollar"}]

Get winning bid for an item:
http://localhost:8000/api/v1/getWinningBidForItem/{itemname}

example output:

{"item":"chair","user":"erol","value":10,"unit":"dollar"}

I also prepared a collection for postman included in the zip file.

Data Structures

Json format is used for http requests and response. Go array data structure is used for the get requests.

Concurrency

The package I used for http routing is gorilla/mux which handles the concurrency using goroutines. The database object is opened once and is used globally by all the requests.

Performance

I ran one get request 100 times and it took 3.333 seconds. (the table bid has around 500 records)

time (for i in `seq 1 100`; do
curl http://localhost:8000/api/v1/getBidsOfUser/erol
done)

0.33s user 0.28s system 18% cpu 3.333 total

I ran all the three get requests 100 times and it took 7,170 seconds

time (for i in `seq 1 100`; do
curl http://localhost:8000/api/v1/getBidsForItem/chair;curl
http://localhost:8000/api/v1/getBidsOfUser/erol; curl
http://localhost:8000/api/v1/getWinningBidForItem/chair
done)

1.02s user 0.83s system 25% cpu 7.170 total

I ran all 3 get requests and one post request 100 times and it took 7.217 seconds

time (for i in `seq 1 100`; do
curl http://localhost:8000/api/v1/getBidsForItem/chair;curl
http://localhost:8000/api/v1/getBidsOfUser/erol; curl
http://localhost:8000/api/v1/getWinningBidForItem/chair\;curl -d '{ "item": "chair", "user": "erol", "value": 10, "unit": "dollar"}' -H "Content-Type: application/json" -X POST http://localhost:8000/api/v1/sendBidForItem
done)

1.05s user 0.84s system 26% cpu 7.217 total

The performance does not look bad but of course this judgment will also depend in the real case on how much data we have on the database, our search algorithm, the data structures, the

network speed, and what type of load we are expecting to have, etc. There might be other packages handling the concurrency better or we can also write our own http router with goroutines.