Advanced Networks — Laboratory 3

Juliusz Chroboczek

12 March 2025

In this lab, we will be modifying the chat client of the last lab. I recommend that you either make frequent backups of your code for later reference, or use a revision control system (for example *git*). Do not leave commented-out code in your source.

Exercice 1. Before you start, make sure there are at least a few messages in the chat.

1. Let *id* be the identifier of a chat message. Using `curl`, examine the contents of the URL `/chat/`*id*`.json`.
2. Modify your chat client to use `json.Unmarshal` to parse the data structure into a variable of type `any` and then display the chat message. Since Go is a strongly typed language, you will need to analyse the data structure using either checked type assertions:

   ```
   var a any
   ...
   d, ok := a.(map[string]any)
   ```

   or by using typecase statements:

   ```
   var a any
   ...
   switch d := a.(type) {
   case map[string]any:
       ...
   }
   ```

3. Define the following structure:

   ```
   type chatMessage struct {
           Id   string    `json:"id,omitempty"`
           Time time.Time `json:"time,omitempty"`
           Body string    `json:"body"`
   }
   ```

   Note that all the fields are public (they start with a capital letter) and are annotated with information for the `json` package. Modify your code to unmarshal into a structure of type `chatMessage` and display the result.

4. Using the high-level functionality of the `json` package is obviously more convienient than manually analysing an untyped data structure. Give one reason why it might be necessary to do the analysis manually in some cases.
5. In this server, I have decided to put the plain-text data and the JSON data at different URLs (`/chat/` and `/chat.json` respectively). Another possible approach would be to put them at the same URL and use the `Accept` request header (RFC 9110 Section 12.5.1) to communicate the client's preferences. What are the advantages of each approach?

Exercice 2.

1. Using `curl`, examine the URLs `/chat.json` and `/chat.json?count=4`.
2. Modify your program to display the last 50 messages using the JSON API. How many RTTs are required?
3. What are the advantages of using the JSON API? What are its drawbacks (advantages of the REST-like API)?
4. The ETag returned by the server does not depend on the value of the `count` parameter. Is that correct?
5. Modify your program so that it keeps displaying messages as they are published. Compare with the REST-like approach.

Exercice 3 (If there's time left)**.** Write a chat system with clients for the web and for common mobile systems. Make *Whatsapp* and *Signal* obsolete. Become a millionaire. Share your fortune with your lecturer.