Go http package

Nurali Virani
Gojek, Bangalore

Demo1 - Basic Go web server and Go Client (homeHandler)

- Client-side
  - http.Get()
  - http.DefaultClient
    - Get()
  - Response{}
    - Body()
- Server-side
  - http.HandleFunc()
  - http.ListenAndServe()

You will be developing web server to server your web application.

You will be developing web client to call APIs exposed by other web server. These APIs could be Third-party APIs (for ex, Paytm calling HDFC Third-party APIs) OR it can be Internal APIs (for ex, "*Order History*" service calling "*Customer*" service APIs) in case of Microservice architecture.

Demo2 - HTTP Get Request / Response (itemsHandler)

- Client-side
  - http.NewRequest()
  - http.MethodGet
  - Client{}
    - Do()
  - Response{}
    - StatusCode
    - Status

- Server-side
  - Request.URL.Query().Get()
  - http.Error()

Demo3 - http.Server and http.Client

- Client-side
  - Transport{}
  - Client{}
    - Timeout

- Server-side
  - http.NewServeMux()
  - ServeMux{}
    - HandleFunc()
  - Server
    - ListenAndServe()

Never use http.DefaultClient as there is NO request timeout set so your client will hang on request if server takes lot of time to respond.

A Transport is a struct used by clients to manage the underlying TCP connection (set certificate, TLS handshake timeout etc) and it’s Dialer is a struct that manages the establishment of the connection (timeout etc).

Demo code will cap the TCP connect and TLS handshake timeouts, as well as establishing an end-to-end request timeout.

Demo4 - Session management using Cookie (buyHandler + loginHandler)

- Client-side
  - Request{}
    - SetBasicAuth()
    - AddCookie()
  - Response{}
    - Cookies()

- Server-side
  - Request.Header.Get()
  - Request{}
    - Cookie(“cookie-name”)
  - Cookie{}

Session management using Cookie is popular in Monolithic web application.

Microservice architecture would not prefer to use Cookie rather in each request authentication details (like JWT token) is sent.

Cookie is still useful to set user preferences. For example, we want to store whether user like “Dark mode” or “Light mode” theme.

Go http package covered functionalities

- **Client-side**
  - http.Get()
  - http.DefaultClient
    - Get()
  - Response{}
    - Body()
  - http.NewRequest()
  - http.MethodGet
  - Client{}
    - Do()
  - Response{}
    - StatusCode / Status
  - Transport{}
  - Client{}
    - Timeout
  - Request{}
    - SetBasicAuth()
    - AddCookie()
  - Response{}
    - Cookies()

- **Server-side**
  - http.HandleFunc()
  - http.ListenAndServe()
  - Request.URL.Query().Get()
  - http.Error()
  - http.NewServeMux()
  - ServeMux{}
    - HandleFunc()
  - Server
    - ListenAndServe()
  - Request.Header.Get()
  - Request{}
    - Cookie(“cookie-name”)
  - Cookie{}

Summary

- We have covered different Methods, Types, Constants available under Go “http” package. Here is “http” package doc which has full list - https://pkg.go.dev/net/http?tab=doc
- Most of useful functionalities are covered which is used for developing Web Server and Web Client in Go
- You can find all codes and this slide at my github - go-http
- You can find my past talks here - https://github.com/nurali-techie/meetup-golang/

- Please share your Anonymous Feedback at - link

Thank you

Nurali Virani
https://linkedin.com/in/nurali-techie/
https://github.com/nurali-techie/