Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Теоретическая информатика и компьютерные технологии»

Лабораторная работа № 2

по курсу «Компьютерные системы и сети»

«Клиент и сервер HTTP»

Студент группы ИУ9-32Б Федуков А. А.

Преподаватель Посевин Д. П.

17 сентября 2024 г.

Цель работы

Целью данной работы является создание HTTP-клиента и HTTP-сервера на языке Go.

Задание

В ходе выполнения лабораторной работы нужно разработать на языке Go HTTP-сервер, который формирует динамические HTML-страницы на основе данных, получаемых с указанного web-сервера. Детали задания приведены в таблицах ниже.

Основные требования к HTTP-серверу: 1. полная проверка данных, получаемых из сети (как от клиента, так и от web-сервера); 2. устойчивость к обрыву соединения; 3. возможность одновременного подключения нескольких клиентов к одному серверу; 4. сервер должен вести подробный лог всех ошибок, а также других важных событий (установка и завершение соединения с клиентом, приём и передача сообщений, и т. п.); 5. в зависимости от варианта список новостей должен быть стилизован и содержать, в случае наличия на сайте-доноре, ссылки на первоисточник открывающейся в новом окне, в случае наличия превью изображения к новости на сайте доноре, то это изображение должно отображаться на реализовываемом HTTP-сервере.

Реализация

Я создал три файла: download.go, который реализовывал парсинг сайта, и server.go, который реализовывал логическую часть сервера, а также start.sh, который собирал код, а потом запускал его.

Код

Листинг 1: Файл download.go

```
1 package main
2
3 import (
```

```
4   "net/http"
5   log "github.com/mgutz/logxi/v1"
6   "golang.org/x/net/html"
7 )
8
9 func getAttr(node *html.Node, key string) string {
10  for _, attr := range node.Attr {
11    if attr.Key == key {
12      return attr.Val
13    }
14  }
15  return ""
16 }
17
18 func getChildren(node *html.Node) []*html.Node {
19  var children []*html.Node
20  for c := node.FirstChild; c != nil; c = c.NextSibling {
21    children = append(children, c)
22  }
23  return children
24 }
25
26 func isElem(node *html.Node, tag string) bool {
27  return node != nil && node.Type == html.ElementNode && node.Data ==
      tag
28 }
29
30 func isDiv(node *html.Node, class string) bool {
31  return isElem(node, "div") && getAttr(node, "class") == class
32 }
33
34 type Link struct {
35  Ref, Text string
36 }
37
38 func getLink(node *html.Node, link *Link) {
39  if node != nil {
40    for c := node.FirstChild; c != nil; c = c.NextSibling {
41      if isElem(c, "a") {
42        link.Ref = getAttr(c, "href")
43        if link.Ref[0] == '/' {
44          link.Ref = "https:" + link.Ref
45        }
46        if isDiv(c.FirstChild, "related-body") {
47          link.Text = "WATCH LIVE: " + c.FirstChild.LastChild.Data
48
```

```
49          } else {
50            link.Text = c.FirstChild.Data
51          }
52
53        } else {
54          getLink(c, link)
55        }
56      }
57    }
58  }
59
60  type Pic struct {
61    Src, W, H, Alt string
62  }
63
64  func getPic(node *html.Node, pic *Pic) {
65    if node != nil {
66      for c := node.FirstChild; c != nil; c = c.NextSibling {
67
68        if isElem(c, "img") {
69          pic.Src = "https:" + getAttr(c, "src")
70          pic.W = getAttr(c, "width")
71          pic.H = getAttr(c, "height")
72          pic.Alt = getAttr(c, "alt")
73        } else if isElem(c, "video") {
74
75          cc := c.FirstChild.NextSibling
76
77          pic.Alt = "VIDEO"
78          pic.Src = getAttr(cc, "src")
79          pic.W = getAttr(cc, "type")
80
81        } else {
82          getPic(c, pic)
83        }
84      }
85    }
86  }
87
88  type Item struct {
89    Pic  Pic
90    Link Link
91  }
92
93  var ITEMS []*Item
94
```

```
95 func search(node *html.Node) {
96   if isElem(node, "article") {
97     var pic Pic
98     var link Link
99     for c := node.FirstChild; c != nil; c = c.NextSibling {
100      // pic
101      if isDiv(c, "m") || isDiv(c, "m ") {
102        if pic.Src != "" {
103          panic("PIC")
104        }
105        getPic(c, &pic)
106      }
107      // link
108      if isDiv(c, "info ") || isDiv(c, "info") {
109        if link.Text != "" {
110          panic("LINK")
111        }
112        getLink(c, &link)
113      }
114    }
115    if pic != (Pic{}) && link != (Link{}) {
116      ITEMS = append(ITEMS, &Item{pic, link})
117    }
118  }
119
120  for c := node.FirstChild; c != nil; c = c.NextSibling {
121    search(c)
122  }
123 }
124
125 func downloadNews() []*Item {
126   log.Info("sending request to foxnews.com")
127   if response, err := http.Get("https://www.foxnews.com/"); err != nil {
128     log.Error("request to foxnews.com failed", "error", err)
129   } else {
130     defer response.Body.Close()
131     status := response.StatusCode
132     log.Info("got response from foxnews.com", "status", status)
133     if status == http.StatusOK {
134       if doc, err := html.Parse(response.Body); err != nil {
135         log.Error("invalid HTML from foxnews.com", "error", err)
136       } else {
137         log.Info("HTML from foxnews.com parsed successfully")
138         search(doc)
139         return ITEMS
140       }
```

```
141     }
142   }
143   return nil
144 }
```

Листинг 2: Файл server.go

```
1  package main
2
3  import (
4    "html/template"
5    "net/http"
6    "github.com/mgutz/logxi/v1"
7  )
8
9  const INDEX_HTML = `
10     <!doctype html>
11     <html lang="ru">
12         <head>
13             <meta charset="utf-8">
14             <title>Last foxnews</title>
15         </head>
16         <body>
17             {{range .}}
18         <div class="article">
19           <h3 class="title">
20             <a href={{ .Link.Ref }}> {{ .Link.Text }} </a>
21           </h3>
22
23
24           <div class="pic">
25             {{ if eq .Pic.Src "https://static.foxnews.com/static/orion/
       img/clear-16x9.gif"}}
26               <p>
27                 <i>Can't load the pic</i> <br>
28                 {{.Pic.Alt}}
29               </p>
30             {{ else }}
31               {{if eq .Pic.Alt "VIDEO"}}
32                 <video playsinline="" autoplay="" muted="" loop="">
33                   <source src={{.Pic.Src}} type={{.Pic.W}}>
34                 </video>
35               {{else}}
36                 <img src={{ .Pic.Src }} width={{ .Pic.W }} height={{ .
       Pic.H }}  alt={{ .Pic.Alt }}>
37               {{end}}
38             {{ end }}
```

```
            </div>
          </div>
          <hr>
        {{end}}
          </body>
      </html>
      `
var indexHtml = template.Must(template.New("index").Parse(INDEX_HTML))

func serveClient(response http.ResponseWriter, request *http.Request) {
  path := request.URL.Path
  log.Info("got request", "Method", request.Method, "Path", path)
  if path != "/" && path != "/index.html" {
    log.Error("invalid path", "Path", path)
    response.WriteHeader(http.StatusNotFound)
  } else if err := indexHtml.Execute(response, downloadNews()); err !=
    nil {
    log.Error("HTML creation failed", "error", err)
  } else {
    log.Info("response sent to client successfully")
  }
}

func main() {
  http.HandleFunc("/", serveClient)
  log.Info("starting listener")
  log.Error("listener failed", "error",
        http.ListenAndServe("185.102.139.169:1414", nil))

}
```

Листинг 3: Файл start.sh

```
#!/bin/bash
export LOGXI=*
export LOGXI_FORMAT=pretty,happy
go build
./main
```

Вывод программы

После загрузки программы на серверной части 185.102.139.169 и ее запуска, я перешел по адресу 185.102.139.169:1414.

Я увидел такую картину

FBI agent exposes 'clear component' he claims motivated would-be assassin to take aim at Trump

Harris presser drought hits 61 days — but she makes time for Hollywood heavyweight

---

5 insane Biden-Harris appliance regulations heading your way

На серверной части же вывелось:

Листинг 4: Лог сервера

```
root@net4:~/Fedukov/lab2# ./start
15:20:03.649321 INF ~ starting listener
15:20:16.190285 INF ~ got request
    Method: GET
    Path: /
15:20:16.191988 INF ~ sending request to foxnews.com
15:20:16.646569 INF ~ got response from foxnews.com
    status: 200
15:20:16.694418 INF ~ HTML from foxnews.com parsed successfully
15:20:17.115671 INF ~ response sent to client successfully
15:20:21.742236 INF ~ got request
```

Вывод

Я научился использовать http протокол на Go, а кроме того парсить сайты и пользоваться net/html шаблонами.