



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 сети»
«Клиент и сервер HTTP»

Студент группы ИУ9-32Б Лавров Р. Д.

Преподаватель Посевин Д. П.

Moskva 2025

1 Задание

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

2 Результаты

Получение курса указанной пользователем валюты за указанную дату с [https://www.sberometer.ru/cbr/ ??](https://www.sberometer.ru/cbr/).

Листинг 1: Пример реализации класса

```
1 package main
2
3 import (
4     "fmt"
5     "html/template"
6     "net/http"
7     "strings"
8
9     log "github.com/mgutz/logxi/v1"
10    "golang.org/x/net/html"
11 )
12
13 var ITEMS []*Item
14
15 func getAttr(node *html.Node, key string) string {
16     for _, attr := range node.Attr {
17         if attr.Key == key {
18             return attr.Val
19         }
20     }
21     return ""
22 }
23
24 func isElem(node *html.Node, tag string) bool {
25     return node != nil && node.Type == html.ElementNode && node.Data == tag
26 }
27
28 type Item struct {
29     Date, USD, EUR, CNY string
30 }
31
32 func getText(node *html.Node) string {
```

```

33  if node == nil {
34      return ""
35  }
36  if node.Type == html.TextNode {
37      return strings.TrimSpace(node.Data)
38  }
39  for c := node.FirstChild; c != nil; c = c.NextSibling {
40      if txt := getText(c); txt != "" {
41          return txt
42      }
43  }
44  return ""
45 }

46
47 func readItem(tr *html.Node) *Item {
48     var cols []string
49     for c := tr.FirstChild; c != nil; c = c.NextSibling {
50         if isElem(c, "td") {
51             cols = append(cols, getText(c))
52         }
53     }
54     if len(cols) < 4 {
55         return nil
56     }
57     return &Item{
58         Date: cols[0],
59         USD: cols[1],
60         EUR: cols[2],
61         CNY: cols[3],
62     }
63 }

64
65 func search(node *html.Node) []*Item {
66     if isElem(node, "tbody") {
67         var items []*Item
68         for c := node.FirstChild; c != nil; c = c.NextSibling {
69             if isElem(c, "tr") {
70                 if item := readItem(c); item != nil {
71                     items = append(items, item)
72                 }
73             }
74         }
75         ITEMS = items
76         return items
77     }
78     for c := node.FirstChild; c != nil; c = c.NextSibling {

```

```

79     if items := search(c); items != nil {
80         return items
81     }
82 }
83 return nil
84 }

85

86 func downloadRates() []*Item {
87     log.Info("sending request to sberometer.ru")
88     if response, err := http.Get("https://www.sberometer.ru/cbr/"); err != nil {
89         log.Error("request failed", "error", err)
90     } else {
91         defer response.Body.Close()
92         status := response.StatusCode
93         log.Info("got response", "status", status)
94         if status == http.StatusOK {
95             if doc, err := html.Parse(response.Body); err != nil {
96                 log.Error("invalid HTML", "error", err)
97             } else {
98                 log.Info("HTML parsed successfully")
99                 return search(doc)
100            }
101        }
102    }
103    return nil
104 }

105

106 const INDEX_HTML = `
107 <!doctype html>
108 <html lang="ru">
109 <head>
110     <meta charset="utf-8">
111     <titleКурсы> валюта ЦБ РФ</title>
112     <style>
113         body { font-family: sans-serif; margin: 20px; }
114         table { border-collapse: collapse; width: 60%; }
115         th, td { border: 1px solid #ccc; padding: 6px 12px; text-align: center; }
116         th { background-color: #f0f0f0; }
117     </style>
118 </head>
119 <body>
120     <h2Курсы> валюта ЦБ РФ</h2>
121     {{if .}}
122     <table>
123         <tr>
124             <thДата></th>

```

```

125      <th>USD</th>
126      <th>EUR</th>
127      <th>CNY</th>
128    </tr>
129    {{range .}}
130    <tr>
131      <td>{{.Date}}</td>
132      <td>{{.USD}}</td>
133      <td>{{.EUR}}</td>
134      <td>{{.CNY}}</td>
135    </tr>
136    {{end}}
137  </table>
138  {{else}}
139    <p>Не удалось загрузить данные!</p>
140  {{end}}
141</body>
142</html>
143‘
144
145const USD_HTML = ‘
146<!doctype html>
147<html lang="ru">
148<head>
149  <meta charset="utf-8">
150  <title>Курсы валют ЦБ РФ</title>
151  <style>
152    body { font-family: sans-serif; margin: 20px; }
153    table { border-collapse: collapse; width: 60%; }
154    th, td { border: 1px solid #ccc; padding: 6px 12px; text-align: center; }
155    th { background-color: #f0f0f0; }
156  </style>
157</head>
158<body>
159  <h2>Курсы валют ЦБ РФ</h2>
160  {{if .}}
161  <table>
162    <tr>
163      <th>Дата</th>
164      <th>USD</th>
165    </tr>
166    {{range .}}
167    <tr>
168      <td>{{.Date}}</td>
169      <td>{{.USD}}</td>
170    </tr>

```

```

171     {{end}}
172   </table>
173 {{else}}
174   <p>Не удалось загрузить данные!</p>
175 {{end}}
176 </body>
177 </html>
178 '
179
180 const EUR_HTML = '
181 <!doctype html>
182 <html lang="ru">
183 <head>
184   <meta charset="utf-8">
185   <title>Курсы валют ЦБ РФ</title>
186   <style>
187     body { font-family: sans-serif; margin: 20px; }
188     table { border-collapse: collapse; width: 60%; }
189     th, td { border: 1px solid #ccc; padding: 6px 12px; text-align: center; }
190     th { background-color: #f0f0f0; }
191   </style>
192 </head>
193 <body>
194   <h2>Курсы валют ЦБ РФ</h2>
195 {{if .}}
196   <table>
197     <tr>
198       <th>Дата</th>
199       <th>EUR</th>
200     </tr>
201   {{range .}}
202     <tr>
203       <td>{{.Date}}</td>
204       <td>{{.EUR}}</td>
205     </tr>
206   {{end}}
207 </table>
208 {{else}}
209   <p>Не удалось загрузить данные!</p>
210 {{end}}
211 </body>
212 </html>
213 '
214
215 const CNY_HTML = '
216 <!doctype html>

```

```

217 <html lang="ru">
218 <head>
219   <meta charset="utf-8">
220   <titleКурсы валют ЦБ РФ</title>
221   <style>
222     body { font-family: sans-serif; margin: 20px; }
223     table { border-collapse: collapse; width: 60%; }
224     th, td { border: 1px solid #ccc; padding: 6px 12px; text-align: center; }
225     th { background-color: #f0f0f0; }
226   </style>
227 </head>
228 <body>
229   <h2Курсы валют ЦБ РФ</h2>
230   {{if .}}
231   <table>
232     <tr>
233       <th>Дата</th>
234       <th>CNY</th>
235     </tr>
236     {{range .}}
237     <tr>
238       <td>{{.Date}}</td>
239       <td>{{.CNY}}</td>
240     </tr>
241     {{end}}
242   </table>
243   {{else}}
244     <p>Не удалось загрузить данные!</p>
245   {{end}}
246 </body>
247 </html>
248 '
249
250 var indexHtml = template.Must(template.New("index").Parse(INDEX_HTML))
251
252 func serveClient(response http.ResponseWriter, request *http.Request) {
253   path := request.URL.Path
254   log.Info("got request", "Method", request.Method, "Path", path)
255   if path != "/" && path != "/index.html" {
256     response.WriteHeader(http.StatusNotFound)
257     return
258   }
259   items := downloadRates()
260   if err := indexHtml.Execute(response, items); err != nil {
261     log.Error("HTML creation failed", "error", err)
262   } else {

```

```

263     log.Info("response sent successfully")
264 }
265 }
266
267 func serveUSD(response http.ResponseWriter, request *http.Request) {
268     var indexHtml2 = template.Must(template.New("index").Parse(USD_HTML))
269     if err := indexHtml2.Execute(response, ITEMS); err != nil {
270         log.Error("HTML creation failed", "error", err)
271     } else {
272         log.Info("response sent successfully")
273     }
274 }
275
276 func serveEUR(response http.ResponseWriter, request *http.Request) {
277     var indexHtml2 = template.Must(template.New("index").Parse(EUR_HTML))
278     if err := indexHtml2.Execute(response, ITEMS); err != nil {
279         log.Error("HTML creation failed", "error", err)
280     } else {
281         log.Info("response sent successfully")
282     }
283 }
284
285 func serveCNY(response http.ResponseWriter, request *http.Request) {
286     var indexHtml2 = template.Must(template.New("index").Parse(CNY_HTML))
287     if err := indexHtml2.Execute(response, ITEMS); err != nil {
288         log.Error("HTML creation failed", "error", err)
289     } else {
290         log.Info("response sent successfully")
291     }
292 }
293
294 func main() {
295     http.HandleFunc("/", serveClient)
296     http.HandleFunc("/usd", serveUSD)
297     http.HandleFunc("/eur", serveEUR)
298     http.HandleFunc("/cny", serveCNY)
299     log.Info("starting server on :6060")
300     fmt.Println("listener failed", "error", http.ListenAndServe(":6060", nil))
301 }
```

The screenshot shows a web browser window with a dark theme. The title bar reads "Programming" and "Небезопасно — 185.102.139.161". The address bar shows the URL "Курсы валют ЦБ РФ". The main content is a table titled "Курсы валют ЦБ РФ" with four columns: "Дата" (Date), "USD", "EUR", and "CNY". The table lists daily exchange rates from October 18, 2025, to September 17, 2025. The last row shows the total values for the month.

Дата	USD	EUR	CNY
18.10.2025	80.9834	94.5835	11.3435
17.10.2025	79.0839	92.0834	11.0623
16.10.2025	78.839	91.7258	11.0216
15.10.2025	79.9589	92.6816	11.119
14.10.2025	80.8548	93.9181	11.2816
11.10.2025	81.1898	94.0491	11.3626
10.10.2025	81.4103	94.703	11.3642
09.10.2025	81.5478	94.9351	11.3476
08.10.2025	81.9349	95.6722	11.3685
07.10.2025	83	96.8345	11.5581
04.10.2025	81.8969	96.0525	11.4165
03.10.2025	81.0085	95.3411	11.3342
02.10.2025	81.4967	95.6382	11.392
01.10.2025	82.6084	96.8644	11.5548
30.09.2025	82.8676	97.141	11.5978
27.09.2025	83.6118	97.6823	11.6751
26.09.2025	83.6069	98.1542	11.7064
25.09.2025	83.9914	98.6015	11.7231
24.09.2025	83.3506	99.0363	11.7014
23.09.2025	84.0186	98.9638	11.7292
20.09.2025	83.5904	98.8845	11.706
19.09.2025	83.1725	98.9773	11.6661
18.09.2025	82.9987	98.2994	11.6623
17.09.2025	82.8359	97.8903	11.6052

Рис. 1 — Результат работы

3 Вывод

Я научился собирать данные с сайтов с помощью функционала языка Golang