Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Московский государственный технический университет
имени Н.Э. Баумана»
(МГТУ им. Н.Э. Баумана)

Факультет: Информатика и системы управления
Кафедра: Теоретическая информатика и компьютерные технологии

Лабораторная работа №9
«Разработка web-консоли»
по курсу: «Компьютерные сети»

Выполнил:
Студент группы ИУ9-32Б
Гнатенко Т. А.

Проверил:
Посевин Д. П.

Москва, 2022

Задачи

Часть 1: https://dzen.ru/video/watch/63792c53dd8c881c51b74f2f
Часть 2: https://dzen.ru/video/watch/63793030be4db8003037e0b8

Решение

Исходный код

route.go

```go
package main

import (
    "bufio"
    "fmt"
    "io/ioutil"
    "net/http"
    "os"
    "os/exec"
    "os/user"
    "path/filepath"
    "strings"

    "github.com/gin-contrib/sessions"
    "github.com/gin-contrib/sessions/cookie"
    "github.com/gin-gonic/gin"
    "github.com/gorilla/websocket"
)

var wsupgrader = websocket.Upgrader{
    ReadBufferSize:  1024,
    WriteBufferSize: 1024,
}

func wshandler(w http.ResponseWriter, r *http.Request) {
    conn, err := wsupgrader.Upgrade(w, r, nil)
    if err != nil {
        fmt.Println("Failed to set websocket upgrade: %+v",
 ↪  err)
```

```go
            return
        }
        for {
            t, msg, err := conn.ReadMessage()
            if err != nil {
                break
            }
            m := strings.Fields(string(msg))
            println(string(msg))
            commName, params := m[0], m[1:]
            cmd := exec.Command(commName, params...)
            stdout, _ := cmd.StdoutPipe()
            err = cmd.Start()
            if err != nil {
                conn.WriteMessage(t, []byte(err.Error()))
            } else {
                scanner := bufio.NewScanner(stdout)
                scanner.Split(bufio.ScanWords)
                for scanner.Scan() {
                    m := scanner.Text()
                    err := conn.WriteMessage(t, []byte(m))
                    if err != nil {
                        println(err.Error())
                    }
                }
            }
        }
    }
func index(c *gin.Context) {
    c.HTML(http.StatusOK, "async.html", nil)
}
func auth(c *gin.Context) {
    c.HTML(http.StatusOK, "auth.html", nil)
}
func login(c *gin.Context) {
    session := sessions.Default(c)
    l := c.Query("login")
    _, err := user.Lookup(l)
    if err != nil {
        session.Set("login", "false")
        err2 := session.Save()
        if err2 != nil {
            println(err2.Error())
        }
        c.JSON(http.StatusBadRequest, gin.H{"err":
↪   err.Error()})
    } else {
        session.Set("login", "true")
        err := session.Save()
        if err != nil {
            println(err.Error())
        }
        println(session.Get("login") == "true")
        println(1)
        c.JSON(http.StatusOK, gin.H{"err": "None"})
    }
}
func sync(c *gin.Context) {
    shelForm := c.PostForm("command")
    commName, params := strings.Fields(shelForm)[0],
↪   strings.Fields(shelForm)[1:]
    cmd := exec.Command(commName, params...)
    out, err := cmd.Output()
    if err != nil {
        out = []byte(err.Error())
    }
    c.HTML(http.StatusOK, "index.html", gin.H{"out":
↪   string(out)})
}
func async(c *gin.Context) {
    session := sessions.Default(c)
    if session.Get("login") == "true" {
        wshandler(c.Writer, c.Request)
    }
}
func run(c *gin.Context) {
    params := strings.Fields(c.Query("path") + " " +
↪   c.Query("params"))
    cmd := exec.Command("python3", params...)
    //var buf bytes.Buffer
    //cmd.Stdout = &buf
    out, err := cmd.Output()
    if err != nil {
        println(err.Error())
        c.JSON(http.StatusBadRequest, gin.H{"err":
 ↪  err.Error()})
    }
    //println(buf.String())
    c.JSON(http.StatusOK, gin.H{
        "return": string(out),
    })
}
func download(c *gin.Context) {
    fileName := c.Query("file")
    _, err := os.Stat(fileName)
    if os.IsNotExist(err) {
        c.JSON(http.StatusBadRequest, gin.H{"err":
 ↪  err.Error()})
    }
    c.Header("Content-Description", "File Transfer")
    c.Header("Content-Transfer-Encoding", "binary")
    c.Header("Content-Disposition", "attachment;
 ↪  filename="+filepath.Base(fileName))
    c.Header("Content-Type", "application/octet-stream")
    c.File(fileName)
}
func upload(c *gin.Context) {
    file, _ := c.FormFile("file")
    fileName := c.PostForm("as")
    err := c.SaveUploadedFile(file, fileName)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"err":
 ↪  err.Error()})
    }
    c.JSON(http.StatusOK, gin.H{})
}
func runBf(c *gin.Context) {
    code := c.Query("code")
    b, err := ioutil.ReadFile(code)
    if err != nil {
        c.JSON(http.StatusBadRequest, gin.H{"err":
↪   err.Error()})
    }
    code = string(b)
    params := strings.Fields(c.Query("params"))
    brc := 0
    j := 0
    res := ""
    var acc [30000]byte
    i := 0
    for k := 0; k < len(code); k++ {
        switch code[k] {
        case '>':
            i++
            break
        case '<':
            i--
            break
        case '+':
            acc[i]++
            break
        case '-':
            acc[i]--
            break
        case '.':
            res += string(acc[i])
        case ',':
            acc[i] = params[j][0]
            j++
            break
        case '[':
            if acc[i] == 0 {
                brc++
                for brc != 0 {
                    k++
                    if code[k] == '[' {
                        brc++
                    }
                    if code[k] == ']' {
                        brc--
                    }
                }
            }
            break
        case ']':
            if acc[i] != 0 {
                if code[k] == ']' {
                    brc++
                }
                for brc != 0 {
                    k--
                    if code[k] == '[' {
                        brc--
                    }
                    if code[k] == ']' {
                        brc++
                    }
                }
                k--
            }
            break
        }
    }
    println(res)
    c.JSON(http.StatusOK, gin.H{"return": res})
}

func main() {
    router := gin.Default()
    router.LoadHTMLGlob("templates/*")
    store := cookie.NewStore([]byte("secret3652"))
    store.Options(sessions.Options{MaxAge: 3600 * 24})
    router.Use(sessions.Sessions("mysession", store))
    router.GET("/", index)
    router.GET("/auth", auth)
    router.GET("/login", login)
    router.GET("/shel-async", async)
    router.POST("/shel-sync", sync)
    router.GET("/shel-sync", index)
    err := router.Run("0.0.0.0:8000")
```

```go
    if err != nil {
        return
    }
}
```