



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 КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Компьютерные системы и сети

О Т Ч Е Т

по лабораторной работе № 9

Название: Back-End разработка с использованием фреймворка Echo

Дисциплина: языки интернет-программирования

Студент

ИУ6-32Б

(Группа)

(Подпись, дата)

И.В.Порохницкий

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

В.Д.Шульман

(И.О. Фамилия)

Москва, 2024

Цель работы - получение первичных навыков использования веб-фреймворков в BackEnd-разработке на Golang.

Задание

1. Ознакомиться с теоретическими сведениями
2. Сделать форк данного репозитория в GitHub, клонировать получившуюся копию локально, создать от мастера ветку dev и переключиться на неё
3. Перекопировать код сервисов, полученный в ходе выполнения 8-й лабораторной работы, в соответствующие поддиректории в директории cmd
4. Доработать сервисы таким образом, чтобы роутинг, обработка запросов, парсинг json, обработка ошибок и логирование осуществлялись на базе фреймворка Echo
5. Проверить свой код линтерами с помощью команды make lint
6. Сделать отчёт и поместить его в директорию docs
7. Зафиксировать изменения, сделать коммит и отправить получившееся состояние ветки dev в личный форк данного репозитория в GitHub
8. Через интерфейс GitHub создать Pull Request dev --> master

Ход работы

1. Ознакомился с теоретическими сведениями
2. Сделал форк данного репозитория в GitHub, клонировал получившуюся копию локально, создал от мастера ветку dev и переключился на нее
3. Выполнил 3 задания:

Задание №1 “count”:

Необходимо написать веб сервер, аналогичный данному из 8-ой лабораторной работы на echo (рис. 1):

```

1 package main
2
3 import (
4     "database/sql"
5     "fmt"
6     "log"
7     "net/http"
8     "strconv"
9
10    "github.com/labstack/echo"
11    "github.com/labstack/echo/middleware"
12    "github.com/lib/pq"
13 )
14
15 const (
16     host      = "localhost"
17     port      = 5432
18     user      = "postgres"
19     password  = "80WZ6e1248882"
20     dbname    = "sandbox"
21 )
22
23 type Handlers struct {
24     dbProvider DatabaseProvider
25 }
26
27 type DatabaseProvider struct {
28     db *sql.DB
29 }
30
31 func (h *Handlers) PostCount(c echo.Context) error {
32     countStr := c.FormValue("count")
33     if countStr == "" {
34         return echo.NewHTTPError(http.StatusBadRequest, "count is empty")
35     }
36
37     count, err := strconv.Atoi(countStr)
38     if err != nil {
39         return echo.NewHTTPError(http.StatusBadRequest, fmt.Sprintf("Invalid count value: %v", err))
40     }
41
42     if err := h.dbProvider.IncrementCount(count); err != nil {
43         return echo.NewHTTPError(http.StatusInternalServerError, "Database error").SetInternal(err)
44     }
45
46     return c.String(http.StatusOK, fmt.Sprintf("Value %d has been added to count", count))
47 }
48
49 func (h *Handlers) GetCount(c echo.Context) error {
50     val, err := h.dbProvider.SelectCount()
51     if err != nil {
52         return echo.NewHTTPError(http.StatusInternalServerError, "Database error").SetInternal(err)
53     }
54
55     return c.String(http.StatusOK, fmt.Sprintf("Current count value: %d", val))
56 }
57
58 func (dp *DatabaseProvider) SelectCount() (int, error) {
59     var val int
60     err := dp.db.QueryRow("SELECT value FROM count WHERE id = 1").Scan(&val)
61     if err != nil {
62         return 0, err
63     }
64     return val, nil
65 }
66
67 func (dp *DatabaseProvider) IncrementCount(n int) error {
68     _, err := dp.db.Exec("UPDATE count SET value = value + ($1) WHERE id = 1", n)
69     return err
70 }
71
72 func main() {
73     pqInfo := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s sslmode=disable", host, port, user, password, dbname)
74     db, err := sql.Open("postgres", pqInfo)
75     if err != nil {
76         log.Fatalf("Failed to connect to database: %v", err)
77     }
78     defer db.Close()
79
80     _, err = db.Exec(
81         CREATE TABLE IF NOT EXISTS count (
82             id SERIAL PRIMARY KEY,
83             value INTEGER NOT NULL DEFAULT 0
84         )
85     )
86     if err != nil {
87         log.Fatalf("Failed to create table: %v", err)
88     }
89
90     _, err = db.Exec("INSERT INTO count (id, value) VALUES (1, 0) ON CONFLICT (id) DO NOTHING")
91     if err != nil {
92         log.Fatalf("Failed to insert initial data: %v", err)
93     }
94
95     e := echo.New()
96
97     e.Use(middleware.Logger())
98     e.Use(middleware.Recover())
99
100    dp := DatabaseProvider{db: db}
101    h := Handlers{dbProvider: dp}
102
103    e.GET("/count", h.GetCount)
104    e.POST("/count", h.PostCount)
105
106    e.Logger.Fatal(e.Start(":3333"))
107 }
108

```


Командная строка

Microsoft Windows [Version 10.0.19045.5247]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\asset>curl -X GET http://localhost:3333/count
Current count value: 12
C:\Users\asset>curl -X POST -d "count=5" http://localhost:3333/count
Value 5 has been added to count
C:\Users\asset>curl -X GET http://localhost:3333/count
Current count value: 17
C:\Users\asset>

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

[Running] go run "c:\Users\asset\web-9\cmd\query\main.go"

 04.13.2
High performance, minimalist Go web framework
<https://echo.labstack.com>

1

SELECT *

2

FROM count

Data Output

Messages

Notifications

	id [PK] integer	value integer
1	1	17

Рис.1

В данной и последующих заданиях были использованы echo роутинг, методы обработки JSON и Form параметров, унифицированный обработчик ошибок, автоматическое логирование запросов

Задание №2 “hello”(Рис. 2):

```
cmd > hello > @ mango > ...
1 package main
2
3 import (
4     "database/sql"
5     "encoding/json"
6     "fmt"
7     "log"
8     "net/http"
9
10    _ "github.com/lib/pq"
11 )
12
13 const (
14     host = "localhost"
15     port = 5432
16     user = "postgres"
17     password = "8KM2661248882"
18     dbname = "Sandbox"
19 )
20
21 type Handlers struct {
22     dbProvider DatabaseProvider
23 }
24
25 type DatabaseProvider struct {
26     db *sql.DB
27 }
28
29 // Обработчики HTTP-запросов
30 func (h *Handlers) GetHello(w http.ResponseWriter, r *http.Request) {
31     msg, err := h.dbProvider.SelectHello()
32     if err != nil {
33         w.WriteHeader(http.StatusInternalServerError)
34         w.Write([]byte(err.Error()))
35     }
36
37     w.WriteHeader(http.StatusOK)
38     w.Write([]byte(msg))
39 }
40
41 func (h *Handlers) PostHello(w http.ResponseWriter, r *http.Request) {
42     input := struct {
43         Msg string `json:"msg"`
44     }{}
45
46     decoder := json.NewDecoder(r.Body)
47     err := decoder.Decode(&input)
48     if err != nil {
49         if err != nil {
50             w.WriteHeader(http.StatusBadRequest)
51             w.Write([]byte(err.Error()))
52         }
53     }
54
55     err = h.dbProvider.InsertHello(input.Msg)
56     if err != nil {
57         w.WriteHeader(http.StatusInternalServerError)
58         w.Write([]byte(err.Error()))
59     }
60
61     w.WriteHeader(http.StatusCreated)
62 }
63
64 // Методы для работы с базой данных
65 func (dp *DatabaseProvider) SelectHello() (string, error) {
66     var msg string
67
68     // Получаем одно сообщение из таблицы hello, отсортированной в случайном порядке
69     row := dp.db.QueryRow("SELECT message FROM hello ORDER BY RANDOM() LIMIT 1")
70     err := row.Scan(&msg)
71     if err != nil {
72         return "", err
73     }
74
75     return msg, nil
76 }
77
78 func (dp *DatabaseProvider) InsertHello(msg string) error {
79     _, err := dp.db.Exec("INSERT INTO hello (message) VALUES ($1)", msg)
80     if err != nil {
81         return err
82     }
83
84     return nil
85 }
86
87 func main() {
88     // Формирование строки подключения для postgres
89     psqlInfo := fmt.Sprintf("host=%s port=%d user=%s "+
90         "password=%s dbname=%s sslmode=disable",
91         host, port, user, password, dbname)
92
93     // Создание соединения с сервером postgres
94     db, err := sql.Open("postgres", psqlInfo)
95     if err != nil {
96         log.Fatal(err)
97     }
98     defer db.Close()
99
100    // Создаем провайдер для БД с набором методов
101    dp := DatabaseProvider(db)
102    // Создаем экземпляр структуры с набором обработчиков
103    h := Handlers{dbProvider: dp}
104
105    // Регистрируем обработчики
106    http.HandleFunc("/get", h.GetHello)
107    http.HandleFunc("/post", h.PostHello)
108
109    // Запускаем веб-сервер на указанном адресе
110    err = http.ListenAndServe(":8081", nil)
111    if err != nil {
112        log.Fatal(err)
113    }
114 }
```

Рис.2

Результат работы (рис. 3):

```
mbi — -zsh — 79x24
Last login: Tue Dec 17 00:51:57 on ttys010
[mbi@MacBook-Air-ila ~ % curl -X POST -H "Content-Type: application/json" -d '{"msg":"Привет, Go!"}' http://localhost:8081/post
[mbi@MacBook-Air-ila ~ % curl -X GET http://localhost:8081/get
Привет, Go!%
mbi@MacBook-Air-ila ~ %
```

1	SELECT *
2	FROM hello












Data Output	Messages	Notifications
        		
	id [PK] integer 	message character varying (100) 
1	1	Привет, мир!
2	2	Привет, мир!
3	3	Привет, Go!

рис. 3

Задание №3 “query”(Рис. 4):

```
cmd /c go run -v message.go
1 package main
2
3 import (
4     "database/sql"
5     "fmt"
6     "log"
7     "net/http"
8 )
9
10 "github.com/libstack/echo/v4"
11 "github.com/libstack/echo/v4/middleware"
12 "github.com/libstack/echo/v4/middleware"
13
14 const (
15     host = "localhost"
16     port = 8080
17     user = "postgres"
18     password = "postgres"
19     dbname = "postgres"
20 )
21
22 type Handler struct {
23     dbProvider database.Provider
24 }
25
26 type DatabaseProvider struct {
27     db *sql.DB
28 }
29
30 func (h *Handler) PostQuery(c echo.Context) error {
31     name := c.QueryParam("name")
32     if name == "" {
33         return echo.NewHTTPError(http.StatusBadRequest, "name is empty")
34     }
35
36     if err := h.dbProvider.InsertName(name); err != nil {
37         return echo.NewHTTPError(http.StatusInternalServerError, "Database error: "+err.Error())
38     }
39
40     return c.String(http.StatusOK, "name is posted")
41 }
42
43 func (h *Handler) GetQuery(c echo.Context) error {
44     name := c.QueryParam("name")
45     if name == "" {
46         return c.String(http.StatusBadRequest, "name is empty")
47     }
48
49     if err := h.dbProvider.SelectName(name); err != nil {
50         return c.String(http.StatusInternalServerError, "name does not exist")
51     }
52
53     return c.String(http.StatusOK, "hello, "+name+"!")
54 }
55
56 func (h *DatabaseProvider) InsertName(name string) error {
57     _, err := h.db.Exec("INSERT INTO query (name) VALUES ($1)", name)
58     return err
59 }
60
61 func (h *DatabaseProvider) SelectName(name string) error {
62     var result string
63     err := h.db.QueryRow("SELECT name FROM query WHERE name = $1", name).Scan(&result)
64     if err != nil {
65         return err
66     }
67     return nil
68 }
69
70 func main() {
71     // Create DB
72     pgURL := fmt.Sprintf("host=%s port=%d user=%s password=%s dbname=%s sslmode=disable", host, port, user, password, dbname)
73     db, err := sql.Open("postgres", pgURL)
74     if err != nil {
75         log.Fatalf("Database connection error: %v", err)
76     }
77     defer db.Close()
78
79     // Create table
80     _, err = db.Exec("CREATE TABLE IF NOT EXISTS query (name VARCHAR(50))")
81     if err != nil {
82         log.Fatalf("Table creation error: %v", err)
83     }
84
85     // Create handler
86     h := &Handler{
87         dbProvider: &DatabaseProvider{
88             db: db,
89         },
90     }
91
92     // Create echo instance
93     e := echo.New()
94     e.Use(middleware.Logger())
95     e.Use(middleware.Recover())
96
97     // Create routes
98     e.GET("/get", h.GetQuery)
99     e.POST("/post", h.PostQuery)
100
101     if err := e.Start(":8080"); err != nil {
102         log.Fatalf("Server error: %v", err)
103     }
104 }
105
106 207
```

```
Командная строка
Microsoft Windows [Version 10.0.19045.5247]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\asset\curl -X GET "http://localhost:8080/api/user/get?name=Ivan"
User does not exist!
C:\Users\asset\curl -X POST "http://localhost:8080/api/user/post?name=Ivan"
Name is posted
C:\Users\asset\curl -X GET "http://localhost:8080/api/user/get?name=Ivan"
Hello, Ivan!
C:\Users\asset>
```

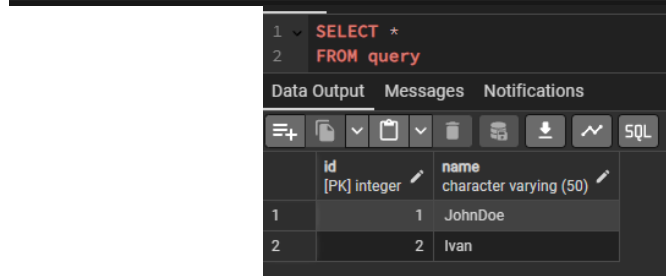


рис. 6

рис. 7

4. Сделал отчёт и поместил его в директорию docs
Зафиксировал изменения, сделал коммит и отправил полученное состояние ветки дев в удаленный репозиторий GitHub

Вывод

Я получил первичные навыки использования веб-фреймворков в BackEnd-разработке на Golang.