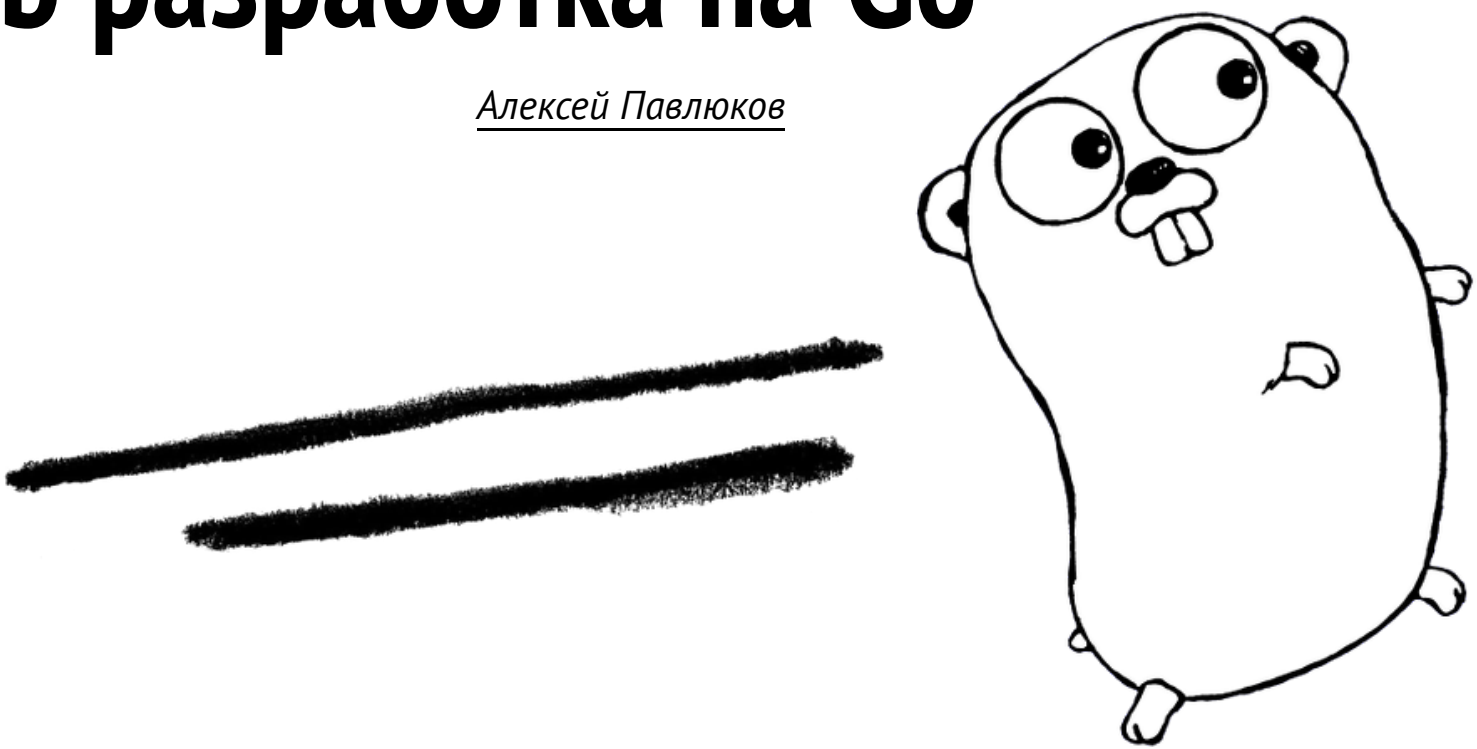
Web разработка на Go

Алексей Павлюков



Web в данной презентации

Взаимодействие браузер - сервер HTTP 1 и 2

1. Загрузка ресурсов в браузер (рендеринг и статический контент)
2. Загрузка ресурсов на сервер - FORMs
3. Взаимодействие в реальном времени - WebSocket
4. Динамический контент - REST/~~GraphQL~~

Чем хорош Go в web

1. Статически типизированный язык
2. Обширная стандартная библиотека
3. Корутины - горутины
4. Обратная совместимость
5. Один бинарный файл
6. Кросс-компиляция

Для начала

1. Установить Go с сайта golang.org
2. IDE для удобства
    - Visual Studio Code
    - Goland
    - Другая на Ваш выбор
3. Установка пакетов `go get ...`

HTTP сервер из коробки

```
package main
import (
    "fmt"
    "net/http"
    "time"
)
func main() {
    http.ListenAndServe("127.0.0.1:8001", http.HandlerFunc(
        func(w http.ResponseWriter, r *http.Request) {
            fmt.Fprintf(w, "Welcome to web at %s!", time.Now().Format("03:04:05"))
        }))
}
```

Test it

```
Welcome to web at 05:14:38!
```

Роутинг и шаблоны

```
func main() {
    tmpl := template.Must(template.New("index").Parse(`<html>
    <head><link rel="stylesheet" href="/static/awsm.min.css"></head>
    <body><h1>Welcome to web at {{.Format "03:04:05"}}!</h1></body>
    </html>`))

    router := http.NewServeMux()
    router.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
        w.Header().Add("Content-Type", "text/html")
        w.WriteHeader(http.StatusOK)
        tmpl.Execute(w, time.Now())
    })

    fs := http.FileServer(http.Dir("static/"))
    router.Handle("/static/", http.StripPrefix("/static/", fs))
    http.ListenAndServe("127.0.0.1:8002", router)
}
```

Test it

Welcome to web at 05:14:20!

Подключение к SQL БД

```
import _ "github.com/mattn/go-sqlite3"

func main() {
    db, err := sql.Open("sqlite3", "file:db.sqlite?cache=shared")
    if err != nil {
        log.Fatalln(err)
    }
    defer db.Close()
    _, err = db.Exec("CREATE TABLE demo (id INTEGER PRIMARY KEY AUTOINCREMENT, url TEXT)")
    if err != nil {
        log.Fatalln(err)
    }
}
```

Middleware

```
func main() {
    // ...
    router := http.NewServeMux()
    middleware := func(w http.ResponseWriter, r *http.Request) {
        path, now := r.URL.String(), time.Now()
        if _, err := db.Exec("INSERT INTO demo (url, time) VALUES (?,?)", path, now); err != nil {
            log.Println(err)
        }
        router.ServeHTTP(w, r)
    }
    // ...
    http.ListenAndServe("127.0.0.1:8003", http.HandlerFunc(middleware))
}
```

Структурируем параметры шаблона

```
import (...)
type templateDataRecord struct {
    ID   int
    Time time.Time
    URL  string
}
type templateData struct {
    Now     time.Time
    Error   string
    Records []templateDataRecord
}

func main() {
```

Добавим в шаблон вывод лога запросов

```
func main() {
        // ...
        tmpl := template.Must(template.New("index").Parse(`<html>
            <head><link rel="stylesheet" href="/static/awsm.min.css"></head> <body>
                <h1>Welcome to web at {{.Now.Format "03:04:05"}}!</h1>
                {{if .Error}}<h2 style="color:red;">Error: {{.Error}}</h2>{{- end}}
                <table>
                    {{range .Records}}<tr><td>{{.ID}}</td><td>{{.URL}}</td>
                    <td>{{.Time.Format "2006.01.02 03:04:05"}}</td></tr>{{- end}}
                </table>
            </body> </html>`))
        // ...
    }
```

Добавим данных

```
router.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) { //...
    data := templateData{Now: time.Now()}
    if rows, err := db.Query("SELECT id, url, time FROM demo ORDER BY time DESC LIMIT 15"); err != nil {
        data.Error = err.Error()
    } else {
        defer rows.Close()
        for rows.Next() {
            var record templateDataRecord
            if err := rows.Scan(&record.ID, &record.URL, &record.Time); err != nil {
                data.Error = err.Error()
                break
            }
            data.Records = append(data.Records, record)
        }
    }
    tmpl.Execute(w, data)
})
```

Test it

Welcome to web at 05:13:51!

13	/index.html	2018.08.20 05:13:51
12	/index.html	2018.08.20 05:13:42
11	/index.html	2018.08.20 05:13:24
10	/static/awsm.min.css	2018.08.20 05:07:31
9	/index.html	2018.08.20 05:07:30

Как сделать проще?

Используем библиотеки

1. `go get -u github.com/jmoiron/sqlx`
2. `go get -u github.com/labstack/echo/...`

Подключаем SQL БД

```
import (
    // ...
    "github.com/jmoiron/sqlx"
)
func main() {
    db, err := sqlx.Open("sqlite3", "file:db.sqlite?cache=shared")
    if err != nil {
        log.Fatalln(err)
    }
    defer db.Close()
    if _, err := db.Exec("CREATE TABLE IF NOT EXISTS demo (id INTEGER PRIMARY KEY AUTOINCREMENT, url TE
    // ...
}
```

Сервер

```
func main() { //...
    e := echo.New()
    //...
    e.Static("/static", "static")
    e.GET("/", func(c echo.Context) error {
        data := templateData{Now: time.Now()}
        if err := db.Select(&data.Records, "SELECT * FROM demo ORDER BY time DESC LIMIT 15"); err != ni
            data.Error = err.Error()
        }
        return c.Render(http.StatusOK, "index", data)
    })
    e.Logger.Fatal(e.Start("127.0.0.1:8004"))
}
```

Middleware

```
func main() {
    //...
    e.Use(func(next echo.HandlerFunc) echo.HandlerFunc {
        return func(c echo.Context) error {
            path, now := c.Request().URL.String(), time.Now()
            if _, err := db.Exec("INSERT INTO demo (url, time) VALUES (?,?)", path, now); err != nil {
                c.Logger().Error(err)
            }
            return next(c)
        }
    })
    // ...
}
```

Шаблоны

```
const indexTemplate = `...`
type templateRenderer struct {
    templates *template.Template
}
func (t *templateRenderer) Render(w io.Writer, name string, data interface{}, c echo.Context) error {
    return t.templates.ExecuteTemplate(w, name, data)
}
func main() {
    //...
    e := echo.New()
    e.Renderer = &templateRenderer{template.Must(template.New("index").Parse(indexTemplate))}
    // ...
}
```

Работа с JSON/XML

```
package main
import (
    "encoding/json"
    "fmt"
    "os"
)
func main() {
    type ColorGroup struct {
        ID     int
        Name   string
        Colors []string
    }
    group := ColorGroup{ID: 1, Name: "Reds",
        Colors: []string{"Crimson", "Red", "Ruby", "Maroon"},
    }
    b, err := json.Marshal(group)
    if err != nil {
        fmt.Println("error:", err)
    }
    os.Stdout.Write(b)
}
```

```
package main
import (
    "encoding/json"
    "fmt"
)
func main() {
    var jsonBlob = []byte(`[
    {"Name": "Platypus", "Order": "Monotremata"},
    {"Name": "Quoll",    "Order": "Dasyuromorphia"}
]`)
    type Animal struct {
        Name  string
        Order string
    }
    var animals []Animal
    err := json.Unmarshal(jsonBlob, &animals)
    if err != nil {
        fmt.Println("error:", err)
    }
    fmt.Printf("%+v", animals)
}
```

Работа в echo с JSON/XML

```
package main
import ( // ...
    "sync/atomic"
    "github.com/labstack/echo"
)
type response struct {
    Number int64
    Time   time.Time
}
var responseCounter int64
func newResponse() response {
    return response{atomic.AddInt64(&responseCounter, 1), time.Now()}
}
```

Работа в echo с JSON/XML

```
func main() {
    e := echo.New()
    e.GET("/json", func(c echo.Context) error {
        return c.JSONPretty(http.StatusOK, newResponse(), "  ")
    })
    e.GET("/xml", func(c echo.Context) error {
        return c.XMLPretty(http.StatusOK, newResponse(), "  ")
    })
    e.Logger.Fatal(e.Start("127.0.0.1:8005"))
}
```

Test it

```
{
  "Number": 3,
  "Time": "2018-08-20T17:13:51.9691136+03:00"
}
```

```
4 2018-08-20T17:13:51.9700984+03:00
```

echo и JSON/XML/FORM

```
type User struct {
    Name  string `json:"name" form:"name" query:"name"`
    Email string `json:"email" form:"email" query:"email"`
}
func Handler(c echo.Context) (err error) {
  u := new(User)
  if err = c.Bind(u); err != nil {
    return
  }
  return c.JSON(http.StatusOK, u)
}
```

Сборка проекта и deploy

1. `go build cmd/main.go`
2. В Go работает кросс-компиляция, включается переменными окружения GOOS и GOARCH
3. Инструменты автоматизирующие сборку
    - Makefile
    - Task github.com/go-task/task
    - Mage magefile.org
    - Releaser goreleaser.com - пакеты для разных OS, Docker образы

Пишем REST сервис

1. Используем библиотеки роутинга gorilla/mux, chi, fasthttprouter
2. Используем легкие фреймворки echo, gin, macaron
3. Используем фреймворки beego, buffalo
4. Используем кодогенераторы goa, go-swagger
5. Используем библиотеки для построения микросервисов go-kit, gizmo

Тестирование

```
// файл main_test.go
package main
import (
    "encoding/xml"
    "net/http"
    "net/http/httptest"
    "testing"
    "time"
)

type testResponse struct {
    Number int64
    Time   time.Time
}
// в main.go func createServer() *echo.Echo { /*...*/ }
```

```
func TestServer(t *testing.T) {
    server := httptest.NewServer(createServer())
    defer server.Close()
    now := time.Now()
    resp, err := http.Get(server.URL + "/xml")
    if err != nil {
        t.Errorf("%s", err)
        t.FailNow()
    }
    var r response
    if err := xml.NewDecoder(resp.Body).Decode(&r); err != nil {
        t.Errorf("%s", err)
    }
    if r.Time.Before(now) {
        t.Errorf("response time %v is before %v", r.Number, now)
    }
}
```

Тестирование - фреймворки

1. testify - общего назначения
2. ginkgo - TDD фреймворк
3. httpexpect - тестирование REST API
4. GoConvey - тестирование в browser

WebSocket

Варианты работы с

WebScoket

1. golang.org/x/net/websocket
2. github.com/gorilla/websocket

```
var upgrader = websocket.Upgrader{}
func (s *Service) WebSocket(c echo.Context) error {
    conn, err := upgrader.Upgrade(c.Response(), c.Request(), nil)
    if err != nil {
        return err
    }
    defer conn.Close()
    // ...
    for {
        mt, message, err := c.ReadMessage()
        //...
        if err := c.WriteMessage(mt, message); err != nil {
            //...
        }
    }
    return nil
}
```

Что почитать

1. Effective Go
2. Go by Example
3. Go database/sql tutorial

Спасибо

Алексей Павлюков i@aleksei.co