7. Веб-приложения

net/http

- net/http https://pkg.go.dev/net/http
  - виды методов: https://pkg.go.dev/net/http#pkg-constants
  - виды запросов: https://pkg.go.dev/net/http#pkg-constants
  - запрос: https://pkg.go.dev/net/http#Request
  - ответ: https://pkg.go.dev/net/http#Response
- https://gobyexample.com/http-server
- https://gobyexample.com/http-client
- а где клиент и сервер?

Что такое http.Get

- https://pkg.go.dev/net/http#Get
  - просто конструктор структуры Response
  - а что внутри?
- код http.Get https://cs.opensource.google/go/go/+/master:src/net/http/client.go;l=109?q=DefaultClient%20struct%20&sq=&ss=go%2Fgo
- клиент https://pkg.go.dev/net/http#Client
- https://go.dev/play/p/gxrV3Uw0ajw

Что такое HandleFunc

- https://pkg.go.dev/net/http#HandleFunc
  - Вызывает функцию при обращении на ресурс
  - *HandleFunc registers the handler function for the given pattern in DefaultServeMux*
- ServeMux
  - https://pkg.go.dev/net/http#ServeMux
  - https://pkg.go.dev/net/http#ServeMux.Handle
  - HTTP мультиплексор
  - сопоставляет входящие URL с запросами на ресурсы на функции, которые должны для них вызваться
  - есть дефолтный DefaultServeMux
  - лучше создавать новый
    - потому что DefaultServeMux может использоваться еще в другом коде
    - можно сделать несколько мультиплексоров для разных обработчиков

Где сервер?

- https://gobyexample.com/http-server
- http.ListenAndServe
  - https://pkg.go.dev/net/http#ListenAndServe
  - https://cs.opensource.google/go/go/+/refs/tags/go1.24.2:src/net/http/server.go;l=3664
- сервер – управляет жизнью обработчика запросов
  - https://pkg.go.dev/net/http#Server
    - есть адрес, на котором слушать
    - есть Handler https://pkg.go.dev/net/http#Handler
      - который обрабатывает запросы → serveMux
  - https://pkg.go.dev/net/http#Server.ListenAndServe
  - https://pkg.go.dev/net/http#Server.Shutdown
  - https://pkg.go.dev/net/http#Server.Close

Сервер может отдавать файлы

- https://pkg.go.dev/net/http#FileServer
- ServeMux → FileServer
  - указываем директорию, в которой смотреть файлы
  - обращаемся к файлам в соответствии с тем, как они лежат в директории
- пример – в материалах к лекции
  - https://go.dev/play/p/vUo6wKni3_A
  - https://go.dev/play/p/BTiLyxD1t6K

net/http → gin-gonic, gorilla/mux

- net/http – есть много альтернатив:
  - https://github.com/gin-gonic/gin - в 40 раз быстрее, самый популярный
  - https://github.com/gorilla/mux
  - https://github.com/go-chi/chi
  - https://www.reddit.com/r/golang/comments/dojyv2/gorillamux_or_gingonicgin/
  - все альтернативы совместимы с net/http и дополняют
  - у всех разный набор всего :)

net/http → gin-gonic

```
func handler(w http.ResponseWriter, r *http.Request) {
    w.Header().Set(
            "Content-Type", "application/json")
    id := strings.TrimPrefix(r.URL.Path, "/users/")

    user, err := db.GetUser(id)
    if err != nil {
        w.WriteHeader(404)
        m := map[string]string{"error": "Not found"}
        json.NewEncoder(w).Encode(m)
        return
    }
    json.NewEncoder(w).Encode(user)
}
```

```
func handler(c *gin.Context) {
    user, err := db.GetUser(c.Param("id"))
    if err != nil {
        c.AbortWithStatusJSON(404, gin.H{"error": "Not
found"})
        return
    }
    c.JSON(200, user)
}
```

Postman

- https://www.postman.com/
- curl – не совсем удобно
  - надо печатать в консоли
  - надо аккуратно указывать запрос: тело запроса, адрес, заголовки