ozontech

ozon{ech

Лекция 5. Go + HTTP & JSON

Юсипов Гаяз

Разработчик информационных систем в команде пользовательского контента

OZON

Москва, 2021

Содержание

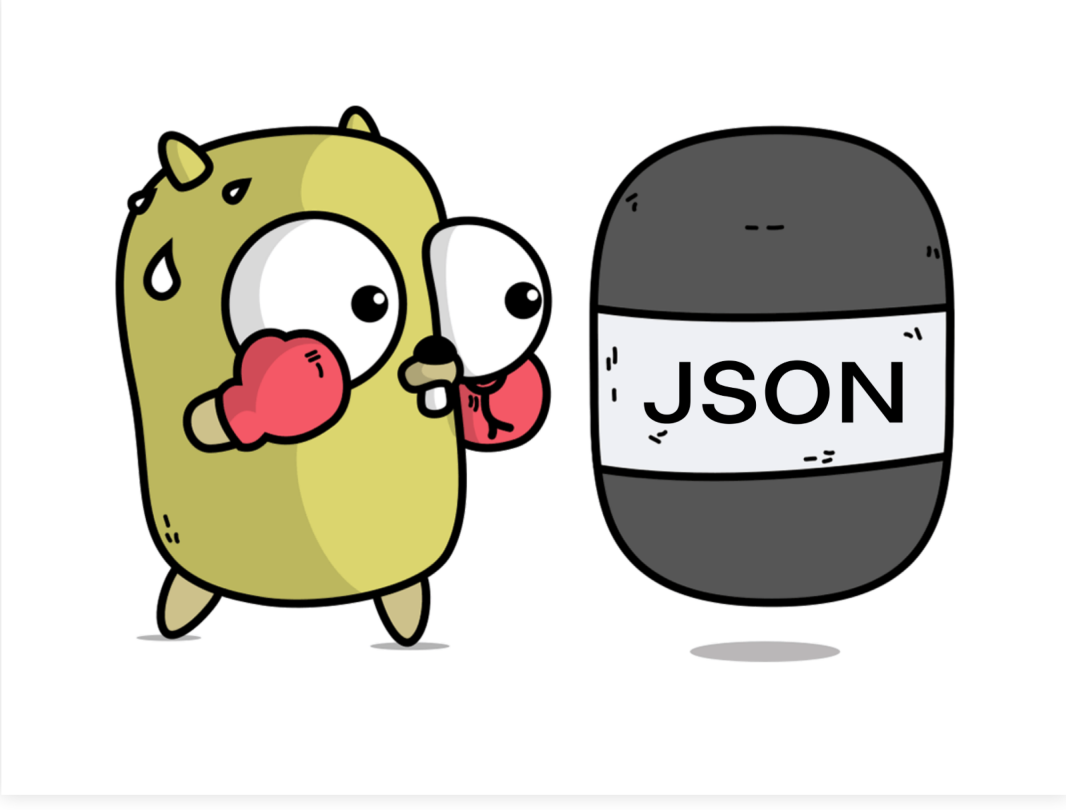
- работа с JSON
- HTTP клиент
- HTTP сервер
- unit тестирование HTTP сервера
- graceful shutdown

ozon tech

Глоссарий

- Так как gRPC работает поверх HTTP/2, что под HTTP я буду подразумевать просто HTTP, то есть без gRPC
- HTTP или gRPC «ручка» — это endpoint или handler

Работа с JSON



encoding/json

- встроенный пакет
- подойдет для большинства кейсов

Теги у полей структур

```
type myStruct struct {
    // Будет использовать название и тип поля
    Field1 int
    // Изменено название
    Field2 int `json:"renamedField2"`
    // Вырезается zero-value и изменено название
    Field3 int `json:"renamedField3,omitempty"`
    // Вырезается zero-value
    Field4 int `json:",omitempty"`
    // Поле игнорируется
    Field5 int `json:"-"`
    // Поле будет называться `-`
    Field6 int `json:"-,"`
    // Примет строковое представление (слишком большое число для JS)
    Int64String int64 `json:",string"`
}
```

Интерфейсы

```
type Marshaler interface {
    MarshalJSON() ([]byte, error)
}
```

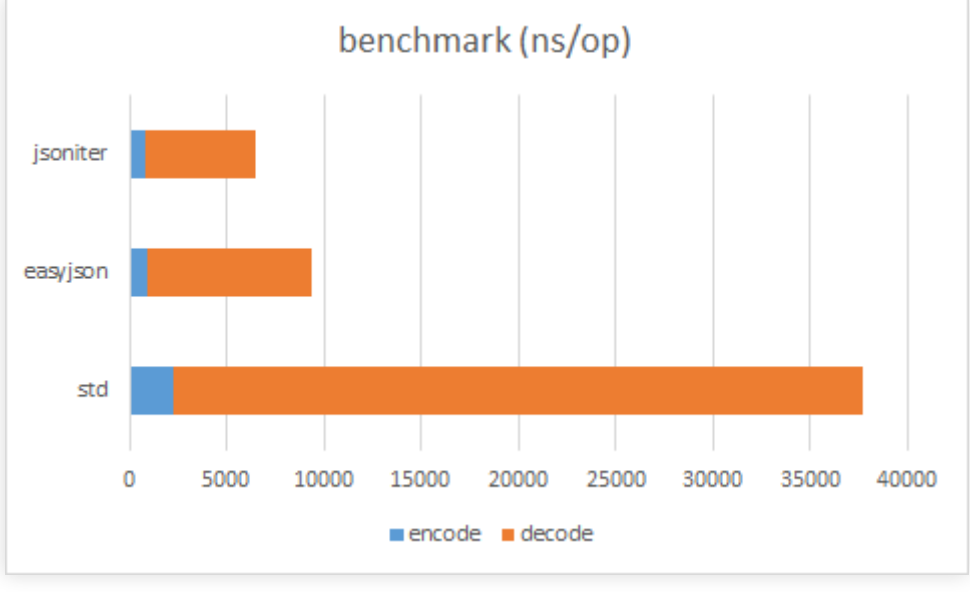
```
type Unmarshaler interface {
    UnmarshalJSON([]byte) error
}
```

Нужны чтобы переопределить стандартное поведение

json-iterator/go

```
go get github.com/json-iterator/go
```

- быстрее чем `encoding/json` и многие аналоги
- полностью совместимая с `encoding/json`
- чтобы перейти с `encoding/json` достаточно поменять пару строк



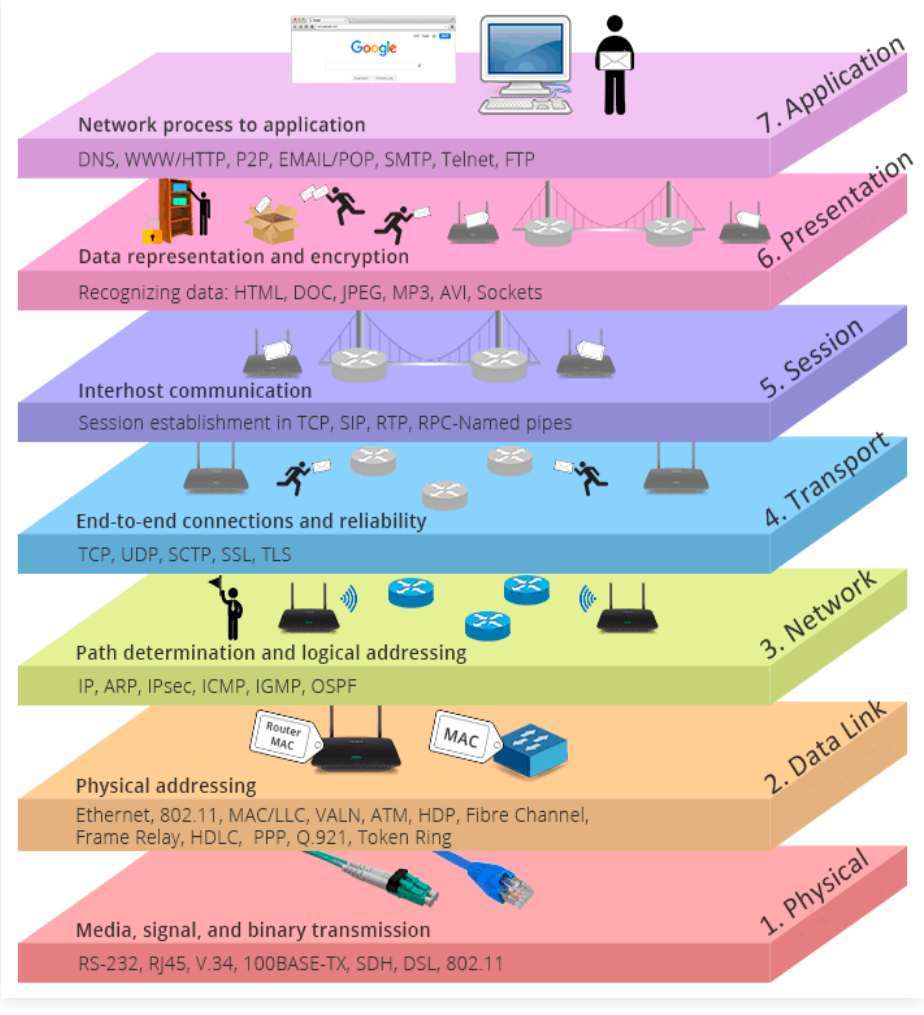
Работа с JSON

- встроенного пакета достаточно для большинства случаев
- можно гибко настроить через реализацию интерфейсов
- есть библиотеки быстрее стандартной, но нужны они редко

Вопросы?

Сетевая модель OSI

The Open Systems Interconnection model



7. Прикладной: Доступ к сетевым службам
6. Представления: Представление и шифрование данных
5. Сеансовый: Управление сеансом связи
4. Транспортный: Прямая связь между конечными пунктами и надёжность
3. Сетевой: Определение маршрута и логическая адресация
2. Канальный: Физическая адресация
1. Физический: Работа со средой передачи, сигналами и двоичными данными<

Немного тулинга

- Postman
- Insomnia
- `*.http` файлы в JetBrains IDE
- curl + jq — процессор JSON командной строки

```
echo '{"foo": 0}' | jq
{
  "foo": 0
}
```

HTTP клиент

Имеется встроенный пакет net для работы с сетью и подпакет net/http для работы с HTTP

http.Client

```
import "net/http"

http.Client{
    Transport     // Как каждый запрос будет сделан, RoundTripper интерф
    CheckRedirect // Функция, определяющая политику работы редиректа
    Jar:          // Общее хранилище cookie
    Timeout:      // Лимит времени выполнения запросов
}
```

HTTP клиент

- не используйте `http.DefaultClient` в продакшене
- всегда используйте контекст с ограничением по времени
- не читайте из `io.Reader` (или `io.ReadCloser`) более одного раза

Вопросы?

HTTP Сервер

Функция `http.ListenAndServe`

```go
http.ListenAndServe("127.0.0.1:8080", nil)
```

- функция блокирующая
- “под капотом” эта функция вызывает функцию `net.Listen`, а затем `http.Serve`
- аргумент `addr` принимает IP адресс и порт `<ip-address>:<port>`, но можно указать только порт `:<port>` и тогда сервер будет доступен по любому IP машины
- всегда возвращает ошибку:
  - сразу, если сервер не получилось запустить (например занят порт)
  - специальную ошибку `http.ErrServerClosed` при вызове метода `Shutdown`

В Ozon мы не так часто делаем HTTP сервера, так как gRPC более производительный

Интерфейс `http.Handler`

```
type Handler interface {
    ServeHTTP(ResponseWriter, *Request)
}
```

Обработчик `handler func(ResponseWriter, *Request)`

- Интерфейс `http.ResponseWriter` позволяет записать заголовки и тело ответа
- Структура `*http.Request` содержит:
  - контекст запроса
  - информацию о запросе
  - payload запроса

HTTP Сервер

- если `http.Handler` передан как nil, то будет использваться `http.DefaultServeMux`
- в стандартном мультиплексере паттерн `/` работает как fallback для всех запросов.
- скорее всего придется подключать мультиплексер как библиотеку, потому как стандартного очень быстро станет недостаточно
- для тестов есть пакет `net/http/httptest` и структура `httptest.ResponseRecorder`
- `Content-Length` выставляется автоматически, можно переопределить
- нужно самостоятельно указывать `Content-Type`, иначе под капотом будет вызываться “дорогая” функция `http.DetectContentType`

Вопросы?

Спасибо за внимание

- Официальная документация пакета `encoding/json` https://pkg.go.dev/encoding/json
- Официальная документация пакета `het/http` https://pkg.go.dev/net/http
- Go: десериализация JSON с неправильной типизацией, или как обходить ошибки разработчиков API https://habr.com/ru/post/502176/
- Don’t use Go’s default HTTP client (in production) https://medium.com/@nate510/don-t-use-go-s-default-http-client-4804cb19f779

Gaiaz Iusipov g.iusipov@gmail.com 2021