Урок Спринт 1.1

Интерфейсы Reader и Writer

`Reader` и `Writer` (а точнее — `io.Reader` и `io.Writer`) — это интерфейсы в Go для чтения и записи данных.

`Reader` определяет метод `Read`, который позволяет читать данные из источника (например, файла или сетевого соединения) в буфер. Метод `Read` возвращает количество прочитанных байтов или ошибку, если чтение не удалось.

Интерфейс `Writer` работает в обратном направлении: с его помощью метод `Write` записывает данные в источник из буфера, возвращает количество записанных байтов или ошибку, если запись не удалась.

`io.Reader` и `io.Writer` находятся в стандартной библиотеке Go, у них сложные фиксированные контракты. Все желающие могут написать свои `Reader` и `Writer` согласно этим контрактам, а разработчики знают, чего ожидать от этих интерфейсов.

Оба интерфейса — гибкие и удобные. Например, вы можете использовать `Reader`, чтобы читать данные из файла, а затем — `Writer`, чтобы записать их в сетевое соединение.

Пакеты `bufio.Reader` и `os.File` реализуют интерфейсы `Reader` и `Writer`. Мы позднее воспользуемся ими.

Структура json.Декодер

`json.Decoder` в Go — это структура, которая позволяет декодировать (парсить) JSON-данные из `Reader` (например, из файла, сетевого соединения или буфера). С её помощью можно читать JSON-данные и преобразовывать их в объекты Go — например, структуры или слайсы. `Decoder` может читать большие объемы JSON-данных, а также обрабатывать ошибки при чтении и декодировании. При этом одновременно загружать все данные в память этой структуре не требуется.

Декодинг JSON с помощью Reader

Для чтения JSON из буфера в Go можно использовать интерфейс `Reader` и `Decoder`. Вот пример:

```go
package main

import (
    "encoding/json"
    "fmt"
    "strings"
)

type Person struct {
    Name string `json:"name"`
    Age  int    `json:"age"`
}

func main() {
    jsonStr := `{"name": "John", "age": 30}`

    // Создаём буфер с JSON-данными
    reader := strings.NewReader(jsonStr)

    // Создаём Decoder для чтения JSON из буфера
    decoder := json.NewDecoder(reader)

    // Создаём переменную для хранения декодированных данных
    var person Person

    // Читаем JSON из буфера и записываем в переменную person
    err := decoder.Decode(&person)
    if err != nil {
        fmt.Println("Ошибка чтения JSON:", err)
        return
    }

    fmt.Printf("Имя: %s, Возраст: %d\n", person.Name, person.Age)
}
```

В этом примере мы создаём буфер с JSON-данными, затем — Reader на основе этого буфера и передаём его в Decoder. Затем с помощью метода Decode мы читаем JSON из буфера, записываем данные в переменную типа Person и выводим значения полей Name и Age на экран.

Структура json.Кодировщик

JSON Encoder в Go — это структура, которая позволяет кодировать данные в JSON и записывать их в writer — например, в файл или буфер. С её помощью можно кодировать и записывать значения любого типа данных в JSON — то есть заниматься сериализацией. JSON Encoder включён в стандартную библиотеку Go.

Справка