Coworking App
-Backend-
Gin Context

Ivan <<ossan>> Pesenti
Software Developer

HACKERSGEN
POWERED BY
SORINT.lab

GIN.CONTEXT

Premessa

Gli obiettivi primari di un handler HTTP sono:

1. Ricevere, validare e parsare una richiesta HTTP
2. Costruire ed inviare una risposta HTTP

In Go, questi concetti sono tradotti in due tipi, rispettivamente:

1. L'interfaccia **ResponseWriter**
2. La struct **http.Request**

Ogni handler HTTP deve quindi poter accedere a queste due "istanze" per assolvere alle sue funzioni.

La Soluzione del Framework Gin

type Context

```
type Context struct {
    Request *http.Request
    Writer  ResponseWriter

    Params Params

    // Keys is a key/value pair exclusively for the context of each request.
    Keys map[string]any

    // Errors is a list of errors attached to all the handlers/middlewares who used this context.
    Errors errorMsgs

    // Accepted defines a list of manually accepted formats for content negotiation.
    Accepted []string
    // contains filtered or unexported fields
}
```

Context is the most important part of gin. It allows us to pass variables between middleware, manage the flow, validate the JSON of a request and render a JSON response for example.

HACKERSGEN

POWERED BY SORINT.LAB

Perchè?

- Il **gin.Context** fornisce una serie di metodi e scorciatoie per manipolare la richiesta e la risposta HTTP
- Il **gin.Engine** è responsabile della creazione (e del riutilizzo) del **gin.Context**
- Il **gin.Engine** si occupa di passare questo **gin.Context** ai middleware e agli handlers
- Una qualsiasi funzione che accetta come unico argomento un ***gin.Context** può essere definita un **gin.HandlerFunc**

VEDIAMOLO NEL CODICE

HACKERSGEN

POWERED BY

SORINT.lab

THANKS!

@ossan

HACKERSGEN

POWERED BY

SORINT.lab