Coworking App
-Backend-
gin.Context Get & Set

Ivan <<ossan>> Pesenti
Software Developer

HACKERSGEN
POWERED BY
SORINT.lab

RAGGI X

HACKERSGEN

POWERED BY

SORINT.lab

gin.Context

La struct **gin.Context**, oltre a trasportare la richiesta e la risposta HTTP, contiene anche un campo chiamato **Keys** di tipo **map[string]any**.
Questo non è altro che un insieme di coppie chiave/valore che sono valide per il contesto dell'esecuzione corrente. In altre parole, le coppie aggiunte, durante la gestione di una richiesta HTTP, vengono distrutte appena inviata la risposta.
Questo campo **Keys** estende l'interfaccia **Context** definita nel package **context** della standard library di Go.

Set

Per aggiungere una nuova coppia chiave-valore, bisogna invocare il metodo **c.Set(key string, value any)**. Quest'ultimo accetta una chiave di tipo **string** e un valore di qualsiasi tipo (**any** può essere usato come sinonimo di **interface{}**). Nel caso in cui la mappa **Keys** non sia stata ancora inizializzata, la inizializza questo metodo.

```go
// Set is used to store a new key/value pair exclusively for this context.
// It also lazy initializes  c.Keys if it was not used previously.
func (c *Context) Set(key string, value any) {
    c.mu.Lock()
    defer c.mu.Unlock()
    if c.Keys == nil {
        c.Keys = make(map[string]any)
    }

    c.Keys[key] = value
}
```

Get: Safe

Per leggere una coppia dal campo **Keys**, bisogna runnare il metodo con la seguente firma: **c.Get(string) (any, bool)**. Quest'ultimo ritornerà una coppia di valori:

1. **value** è il valore racchiuso in un'interfaccia
2. **exists** è il valore di tipo **bool** che indica se la chiave era presente tra quelle del campo **Keys**

```go
// Get returns the value for the given key, ie: (value, true).
// If the value does not exist it returns (nil, false)
func (c *Context) Get(key string) (value any, exists bool) {
    c.mu.RLock()
    defer c.mu.RUnlock()
    value, exists = c.Keys[key]
    return
}
```

Get: Hard

Nel caso la mancata presenza di una chiave sia una condizione “fatale” per il nostro programma, possiamo utilizzare il metodo di convenienza con la firma “**c.MustGet(string) any**”. Nel caso in cui la chiave non sia presente, fa panicare la goroutine.

```go
// MustGet returns the value for the given key if it exists, otherwise it panics.
func (c *Context) MustGet(key string) any {
    if value, exists := c.Get(key); exists {
        return value
    }
    panic("Key \"" + key + "\" does not exist")
}
```

SPERIMENTIAMO CON IL CONTEXT

HACKERSGEN

POWERED BY SORINT.lab

THANKS!

@ossan

HACKERSGEN

POWERED BY

SORINT.lab