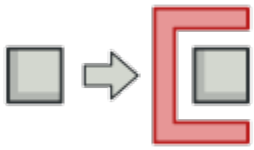
refactoring.guru

Proxy em Go / Padrões de Projeto

4–5 minutes



O **Proxy** é um padrão de projeto estrutural que fornece um objeto que atua como um substituto para um objeto de serviço real usado por um cliente. Um proxy recebe solicitações do cliente, realiza alguma tarefa (controle de acesso, armazenamento em cache etc.) e passa a solicitação para um objeto de serviço.

O objeto proxy tem a mesma interface que um serviço, o que o torna intercambiável com um objeto real quando passado para um cliente.

Exemplo conceitual

Um servidor da web como o Nginx pode atuar como proxy para seu servidor de aplicativos:

- Ele fornece acesso controlado ao seu servidor de aplicativos.
- Pode limitar a taxa.
- Pode fazer solicitação de cache.

server.go: Alvo

```
package main

type server interface {
    handleRequest(string, string) (int, string)
}
```

nginx.go: Proxy

```
package main

type Nginx struct {
    application       *Application
    maxAllowedRequest int
    rateLimiter       map[string]int
}

func newNginxServer() *Nginx {
    return &Nginx{
        application:       &Application{},
        maxAllowedRequest: 2,
        rateLimiter:       make(map[string]int),
    }
}

func (n *Nginx) handleRequest(url, method string) (int, string) {
    allowed := n.checkRateLimiting(url)
    if !allowed {
        return 403, "Not Allowed"
    }
    return n.application.handleRequest(url, method)
```

```
}

func (n *Nginx) checkRateLimiting(url string) bool {
    if n.rateLimiter[url] == 0 {
        n.rateLimiter[url] = 1
    }
    if n.rateLimiter[url] > n.maxAllowedRequest {
        return false
    }
    n.rateLimiter[url] = n.rateLimiter[url] + 1
    return true
}
```

application.go: Alvo real

```
package main

type Application struct {
}

func (a *Application) handleRequest(url, method string) (int, string)
{
    if url == "/app/status" && method == "GET" {
        return 200, "Ok"
    }

    if url == "/create/user" && method == "POST" {
        return 201, "User Created"
    }
    return 404, "Not Ok"
```

```
}
```

main.go: Código cliente

```
package main

import "fmt"

func main() {

    nginxServer := newNginxServer()
    appStatusURL := "/app/status"
    createuserURL := "/create/user"

    httpCode, body := nginxServer.handleRequest(appStatusURL,
"GET")
    fmt.Printf("\nUrl: %s\nHttpCode: %d\nBody: %s\n",
appStatusURL, httpCode, body)

    httpCode, body = nginxServer.handleRequest(appStatusURL,
"GET")
    fmt.Printf("\nUrl: %s\nHttpCode: %d\nBody: %s\n",
appStatusURL, httpCode, body)

    httpCode, body = nginxServer.handleRequest(appStatusURL,
"GET")
    fmt.Printf("\nUrl: %s\nHttpCode: %d\nBody: %s\n",
appStatusURL, httpCode, body)

    httpCode, body = nginxServer.handleRequest(createuserURL,
```

```
"POST")
    fmt.Printf("\nUrl: %s\nHttpCode: %d\nBody: %s\n",
appStatusURL, httpCode, body)

    httpCode, body = nginxServer.handleRequest(createuserURL,
"GET")
    fmt.Printf("\nUrl: %s\nHttpCode: %d\nBody: %s\n",
appStatusURL, httpCode, body)
}
```

output.txt: Resultados da execução

```
Url: /app/status
HttpCode: 200
Body: Ok

Url: /app/status
HttpCode: 200
Body: Ok

Url: /app/status
HttpCode: 403
Body: Not Allowed

Url: /app/status
HttpCode: 201
Body: User Created

Url: /app/status
HttpCode: 404
```

Body: Not Ok

Proxy em outras linguagens