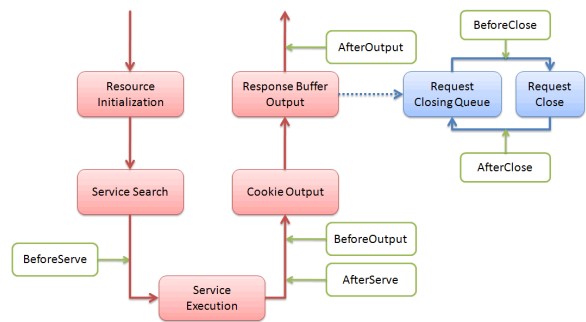


# HOOK



## Content Menu

- - 
  -
- [ExitHook](#)
- - [Request.URL](#)
  - [Request.Router](#)
- - 1
  - 2
  - 3
  - 4
  - 5

```
ghttp.Server

ghttp.Serverpattern(pattern)ghttp.ServerHOOK

func (s *Server) BindHookHandler(pattern string, hook string, handler
HandlerFunc) error
func (s *Server) BindHookHandlerByMap(pattern string, hookmap map[string]
HandlerFunc) error
```

```
func (d *Domain) BindHookHandler(pattern string, hook string, handler
HandlerFunc) error
func (d *Domain) BindHookHandlerByMap(pattern string, hookmap map[string]
HandlerFunc) error
```

### Hook

1. ghttp.HOOK\_BEFORE\_SERVE
2. ghttp.HOOK\_AFTER\_SERVE
3. ghttp.HOOK\_BEFORE\_OUTPUT
4. ghttp.HOOK\_AFTER\_OUTPUT

```
/*HOOKHOOK/*
HOOK(/user/*)HOOK
```

```
() (: ABC)HOOK(: A-B-C)
HOOK
```

## ExitHook

```
HOOKHOOKHOOKRequest.ExitHookHOOKHOOK
```

```

/HOOK_BEFORE_SERVE(/*)r.ExitAll()()

r.Redirect*r.ExitAll()http multiple response writeheader callsr.Redirect*
headerLocationheaderContent-Type/Content-Length

```

MiddlewareHOOKGF

1. ""Server
2. ""
- 3.

## Request.URLRequest.Router

Request.Router Request.URLURLhttp.RequestURLRequest.URL.PathURI

Request.RouternilRequest.URLURLRequest.Router

*APInginx*

gfWebServer

```

/*/static/js/index.js/upload/images/thumb.jpg

Request.IsFileRequest()

```

## 1

package main

```

import (
    "github.com/gogf/gf/frame/g"
    "github.com/gogf/gf/os/glog"
    "github.com/gogf/gf/net/ghttp"
)

func main() {
    //
    p := "/:name/info/{uid}"
    s := g.Server()
    s.BindHookHandlerByMap(p, map[string]ghttp.HandlerFunc{
        ghttp.HOOK_BEFORE_SERVE : func(r *ghttp.Request){ glog.Println(ghttp.
HOOK_BEFORE_SERVE) },
        ghttp.HOOK_AFTER_SERVE   : func(r *ghttp.Request){ glog.Println(ghttp.
HOOK_AFTER_SERVE) },
        ghttp.HOOK_BEFORE_OUTPUT : func(r *ghttp.Request){ glog.Println(ghttp.
HOOK_BEFORE_OUTPUT) },
        ghttp.HOOK_AFTER_OUTPUT  : func(r *ghttp.Request){ glog.Println(ghttp.
HOOK_AFTER_OUTPUT) },
    })
    s.BindHandler(p, func(r *ghttp.Request) {
        r.Response.Write(":", r.Get("name"), ", uid:", r.Get("uid"))
    })
    s.SetPort(8199)
    s.Run()
}

```

<http://127.0.0.1:8199/john/info/10000> WebServer

## 2

```

package main

import (
    "github.com/gogf/gf/frame/g"
    "github.com/gogf/gf/net/ghttp"
)

// HOOK
func beforeServeHook1(r *ghttp.Request) {
    r.SetParam("name", "GoFrame")
    r.Response.Writeln("set name")
}

// HOOK
func beforeServeHook2(r *ghttp.Request) {
    r.SetParam("site", "https://goframe.org")
    r.Response.Writeln("set site")
}

//
// HOOK
func main() {
    s := g.Server()
    s.BindHandler("/", func(r *ghttp.Request) {
        r.Response.Writeln(r.Get("name"))
        r.Response.Writeln(r.Get("site"))
    })
    s.BindHookHandler("/", ghttp.HOOK_BEFORE_SERVE, beforeServeHook1)
    s.BindHookHandler("/", ghttp.HOOK_BEFORE_SERVE, beforeServeHook2)
    s.SetPort(8199)
    s.Run()
}

```

SERVER	ADDRESS	DOMAIN	METHOD	P	ROUTE	HANDLER
MIDDLEWARE						
default	:8199	default	ALL	1	/	main.main.func1
default	:8199	default	ALL	2	/	main.beforeServeHook1
HOOK_BEFORE_SERVE						
default	:8199	default	ALL	1	/	main.beforeServeHook2
HOOK_BEFORE_SERVE						

<http://127.0.0.1:8199/>

```

set name
set site
GoFrame
https://goframe.org

```

```

package main

import (
    "fmt"
    "github.com/gogf/gf/frame/g"
    "github.com/gogf/gf/net/ghttp"
)

func main() {
    s := g.Server()
    // 1
    pattern1 := "/:name/info"
    s.BindHookHandlerByMap(pattern1, map[string]ghttp.HandlerFunc{
        ghttp.HOOK_BEFORE_SERVE: func(r *ghttp.Request) {
            r.SetParam("uid", 1000)
        },
    })
    s.BindHandler(pattern1, func(r *ghttp.Request) {
        r.Response.Write(":", r.Get("name"), ", uid:", r.Get("uid"))
    })

    // 2
    pattern2 :=("/{object}/list/{page}.java"
    s.BindHookHandlerByMap(pattern2, map[string]ghttp.HandlerFunc{
        ghttp.HOOK_BEFORE_OUTPUT: func(r *ghttp.Request) {
            r.Response.SetBuffer([]byte(
                fmt.Sprintf(", object:%s, page:%s", r.Get
("object"), r.GetRouterString("page"))),
            )
        },
    })
    s.BindHandler(pattern2, func(r *ghttp.Request) {
        r.Response.Write(r.Router.Uri)
    })
    s.SetPort(8199)
    s.Run()
}

```

1/:name/infoGET2/{object}/list/{page}.javaURL

- <http://127.0.0.1:8199/user/info>
- <http://127.0.0.1:8199/user/list/1.java>
- <http://127.0.0.1:8199/user/list/2.java>

## 4

```

package main

import (
    "github.com/gogf/gf/frame/g"
    "github.com/gogf/gf/net/ghttp"
)

func main() {
    s := g.Server()
    s.BindHandler("/priority/show", func(r *ghttp.Request) {
        r.Response.WriteLine("priority service")
    })

    s.BindHookHandlerByMap("/priority/:name", map[string]ghttp.HandlerFunc{
        {
            ghttp.HOOK_BEFORE_SERVE: func(r *ghttp.Request) {
                r.Response.WriteLine("/priority/:name")
            },
        })
    s.BindHookHandlerByMap("/priority/*any", map[string]ghttp.HandlerFunc{
        {
            ghttp.HOOK_BEFORE_SERVE: func(r *ghttp.Request) {
                r.Response.WriteLine("/priority/*any")
            },
        })
    s.BindHookHandlerByMap("/priority/show", map[string]ghttp.HandlerFunc{
        {
            ghttp.HOOK_BEFORE_SERVE: func(r *ghttp.Request) {
                r.Response.WriteLine("/priority/show")
            },
        })
    s.SetPort(8199)
    s.Run()
}

```

3/priority/show

<http://127.0.0.1:8199/priority/show>

```
/priority/show
/priority/:name
/priority/*any
priority service
```

5

/ CORS

HOOKHOOK

```
package main
```

```
import (
    "github.com/gogf/gf/frame/g"
    "github.com/gogf/gf/net/ghttp"
)

func Order(r *ghttp.Request) {
    r.Response.Write("GET")
}

func main() {
    s := g.Server()
    s.Group("/api.v1", func(group *ghttp.RouterGroup) {
        group.GET("/order", Order)
    })
    s.SetPort(8199)
    s.Run()
}
```

<http://localhost:8199/api.v1/order> (jQuery)F12consoleAJAX

```
$.get("http://localhost:8199/api.v1/order", function(result){
    console.log(result)
});
```



```
package main
```

```
import (
    "github.com/gogf/gf/frame/g"
    "github.com/gogf/gf/net/ghttp"
)

func Order(r *ghttp.Request) {
    r.Response.Write("GET")
}

func main() {
    s := g.Server()
    s.Group("/api.v1", func(group *ghttp.RouterGroup) {
        group.Hook("/*any", ghttp.HOOK_BEFORE_SERVE, func(r *ghttp.
Request) {
            r.Response.CORSDefault()
        })
        group.GET("/order", Order)
    })
    s.SetPort(8199)
    s.Run()
}
```

```
/api.v1/*anyghttp.HOOK_BEFORE_SERVECORSDefault/api.v1
```

## AJAX

