# HTTPClient-

GoFrameHTTP

> ⚠ ghttp.ServerClientMaxBodySizehttps://pkg.go.dev/github.com/gogf/gf/v2/net/ghttp#ServerConfig 8MB

Request

Content Menu

```
package main

import (
        "github.com/gogf/gf/v2/frame/g"
        "github.com/gogf/gf/v2/net/ghttp"
)

// Upload uploads files to /tmp .
func Upload(r *ghttp.Request) {
        files := r.GetUploadFiles("upload-file")
    names, err := files.Save("/tmp/")
    if err != nil {
                r.Response.WriteExit(err)
        }
        r.Response.WriteExit("upload successfully: ", names)
}

// UploadShow shows uploading simgle file page.
func UploadShow(r *ghttp.Request) {
        r.Response.Write(`
    <html>
    <head>
        <title>GF Upload File Demo</title>
    </head>
        <body>
            <form enctype="multipart/form-data" action="/upload" method="
post">
                <input type="file" name="upload-file" />
                <input type="submit" value="upload" />
            </form>
        </body>
    </html>
    `)
}

// UploadShowBatch shows uploading multiple files page.
func UploadShowBatch(r *ghttp.Request) {
        r.Response.Write(`
    <html>
    <head>
        <title>GF Upload Files Demo</title>
    </head>
        <body>
            <form enctype="multipart/form-data" action="/upload" method="
post">
                <input type="file" name="upload-file" />
                <input type="file" name="upload-file" />
                <input type="submit" value="upload" />
            </form>
        </body>
    </html>
    `)
}

func main() {
        s := g.Server()
        s.Group("/upload", func(group *ghttp.RouterGroup) {
                group.POST("/", Upload)
                group.ALL("/show", UploadShow)
                group.ALL("/batch", UploadShowBatch)
        })
        s.SetPort(8199)
        s.Run()
}
```

3

1. http://127.0.0.1:8199/upload/show H5
2. http://127.0.0.1:8199/upload/batch H5

3. http://127.0.0.1:8199/upload

http://127.0.0.1:8199/upload/show

1. r.GetUploadFilesr.GetUploadFile
2. r.GetUploadFiles("upload-file")"upload-file"
3. files.SaveSave
4. group.POST("/", Upload)POST

```
package main

import (
        "fmt"

        "github.com/gogf/gf/v2/frame/g"
        "github.com/gogf/gf/v2/os/gctx"
        "github.com/gogf/gf/v2/os/glog"
)

func main() {
        var (
                ctx  = gctx.New()
                path = "/home/john/Workspace/Go/github.com/gogf/gf/v2
/version.go"
        )
        result, err := g.Client().Post(ctx, "http://127.0.0.1:8199
/upload", "upload-file=@file:"+path)
        if err != nil {
                glog.Fatalf(ctx, `%+v`, err)
        }
        defer result.Close()
        fmt.Println(result.ReadAllString())
}
```

=@file: HTTPgf @file:+

```
package main

import (
        "fmt"

        "github.com/gogf/gf/v2/frame/g"
        "github.com/gogf/gf/v2/os/gctx"
        "github.com/gogf/gf/v2/os/glog"
)

func main() {
        var (
                ctx   = gctx.New()
                path1 = "/Users/john/Pictures/logo1.png"
                path2 = "/Users/john/Pictures/logo2.png"
        )
        result, err := g.Client().Post(
                ctx,
                "http://127.0.0.1:8199/upload",
                fmt.Sprintf(`upload-file=@file:%s&upload-file=@file:%s`,
path1, path2),
        )
        if err != nil {
                glog.Fatalf(ctx, `%+v`, err)
        }
        defer result.Close()
        fmt.Println(result.ReadAllString())
}
```

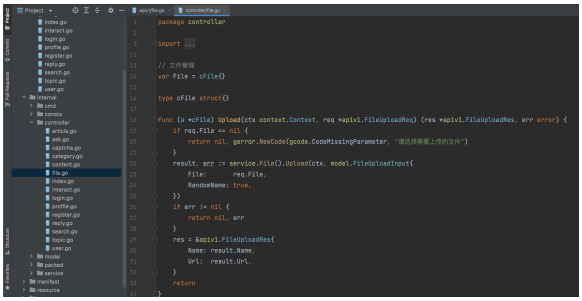
`=@file:xxx&=@file:xxx...[]=@file:xxx&[]=@file:xxx...`

`FileName`

```
s := g.Server()
s.BindHandler("/upload", func(r *ghttp.Request) {
    file := r.GetUploadFile("TestFile")
    if file == nil {
        r.Response.Write("empty file")
        return
    }
    file.Filename = "MyCustomFileName.txt"
    fileName, err := file.Save(gfile.TempDir())
    if err != nil {
        r.Response.Write(err)
        return
    }
    r.Response.Write(fileName)
})
s.SetPort(8999)
s.Run()
```

- `*ghttp.UploadFile`
- `typefile`


Project
focus-single ~/Workspace/GoGOI
api/v1
api.go
article.go
ask.go
captcha.go
category.go
content.go
file.go
index.go
interact.go
login.go
profile.go
register.go
reply.go
search.go
topic.go
user.go
file.go
package apiv1
import ...
type FileUploadReq struct {
g.Meta `path:"/file" method:"post" mime:"multipart/form-data" tags:"工具" summary:"上传文件"`
File *ghttp.UploadFile `json:"file" type:"file" dc:"选择上传文件"`
}
type FileUploadRes struct {
Name string `json:"name" dc:"文件名称"`
Url string `json:"url" dc:"访问URL，可能只是URI"`
}


Project
index.go
interact.go
login.go
profile.go
register.go
reply.go
search.go
topic.go
user.go
internal
cmd
consts
controller
article.go
ask.go
captcha.go
category.go
content.go
file.go
index.go
interact.go
login.go
profile.go
register.go
reply.go
search.go
topic.go
user.go
model
packed
service
manifest
resource
apiv1/file.go
controller/file.go
package controller
import ...
// 文件管理
var File = cFile{}
type cFile struct{}
func (a *cFile) Upload(ctx context.Context, req *apiv1.FileUploadReq) (res *apiv1.FileUploadRes, err error) {
if req.File == nil {
return nil, gerror.NewCode(gcode.CodeMissingParameter, "请选择要上传的文件")
}
result, err := service.File().Upload(ctx, model.FileUploadInput{
File: req.File,
RandomName: true,
})
if err != nil {
return nil, err
}
res = &apiv1.FileUploadRes{
Name: result.Name,
Url: result.Url,
}
return
}