GoFrame

https://oldme.net/article/47

GoFrame

`err""""err SQL err`

`GoFrame gerror err`

HTTP

```
    {
        "code": 10001,
        "message": "",
        "data": null
}

{
        "code": 10002,
        "message": "",
        "data": null
}
```

HTTP 200

/api/exception/v1/exception.go

```
    //
type BusinessReq struct {
        g.Meta `path:"/business" method:"get"`
}

type BusinessRes struct {
        Name string
        Age  int
}

//
type SystemReq struct {
        g.Meta `path:"/system" method:"get"`
}

type SystemRes struct {
        Name string
        Age  int
}
```

/internal/controller/exception/exception_v1_*.go

```
    func (c *ControllerV1) Business(ctx context.Context, req *v1.BusinessReq) (res *v1.BusinessRes, err error) {
        err = service.Exception().Business()
        if err != nil {
                return nil, err
        }
        return &v1.BusinessRes{
                Name: "business",
                Age:  1,
        }, nil
}

func (c *ControllerV1) System(ctx context.Context, req *v1.SystemReq) (res *v1.SystemRes, err error) {
        err = service.Exception().System()
        if err != nil {
                return nil, err
        }
        return &v1.SystemRes{
                Name: "system",
                Age:  1,
        }, nil
}
```

/internal/logic/exception/exception.go

```
    func (s *sException) Business() error {
        return gerror.New("")
}

// System  gjson.Decode() err
func (s *sException) System() error {
        _, err := gjson.Decode("")
        if err != nil {
                return err
        }
        return nil
}
```

api controller logic

Business

curl http://127.0.0.1:8000/business

blocked URL

```
    {
        "code": 50,
        "message": "",
        "data": null
}
```

System

curl http://127.0.0.1:8000/system

blocked URL

```
    {
        "code": 50,
        "message": "json Decode failed: EOF",
        "data": null
}
```

1.
2. `message` "json Decode failed: EOF"
3. `code` 50

err

GoFrame `/internal/utility` `err`

`/internal/utility/err.go`

```
    type pErr struct {
        maps map[int]string
}

var Err = &pErr{
        maps: map[int]string{
                0:     "",
                10001: "",
                10002: "",
                99999: "",
        },
}

// GetMsg codemsg
func (c *pErr) GetMsg(code int) string {
        return c.maps[code]
}

// Skip
func (c *pErr) Skip(code int, msg ...string) (err error) {
        var msgStr string
        if len(msg) == 0 {
                msgStr = c.GetMsg(code)
        } else {
                msg = append([]string{c.GetMsg(code)}, msg...)
                msgStr = strings.Join(msg, ", ")
        }
        // NewWithOption  gf  NewOption
        return gerror.NewWithOption(gerror.Option{
                Stack: false,
                Text:  msgStr,
                Code:  gcode.New(code, "", nil),
        })
}

// Sys code99999
// !!!
// !!!  code 99999
func (c *pErr) Sys(err error) error {
        return gerror.NewCode(gcode.New(CodeErrSys, "", nil), err.Error())
}
```

HTTP

/internal/logic/middleware/response.go

```
    type sMiddleware struct {
}

func init() {
        service.RegisterMiddleware(New())
}

func New() *sMiddleware {
        return &sMiddleware{}
}

type Response struct {
        Code    int         `json:"code"    dc:""`
        Message string      `json:"message" dc:""`
        Data    interface{} `json:"data"    dc:""`
}

func (s *sMiddleware) Response(r *ghttp.Request) {
        r.Middleware.Next()

        if r.Response.BufferLength() > 0 {
                return
        }

        //
        if r.Response.Status >= http.StatusInternalServerError {
                //
                r.Response.ClearBuffer()
                r.Response.Writeln("")
        }

    var (
                res     = r.GetHandlerResponse()
                err     = r.GetError()
                code    = gerror.Code(err)
                codeInt = code.Code()
                msg     string
        )

        if err != nil {
                //
                if codeInt == utility.CodeErrSys {
                        msg = utility.Err.GetSysMsg()
                } else {
                        msg = err.Error()
                }
        } else {
                code = gcode.CodeOK
                msg = utility.Err.GetMsg(code.Code())
        }

        r.Response.WriteJson(Response{
                Code:    code.Code(),
                Message: msg,
                Data:    res,
        })
}
```

/internal/cmd/cmd.go

```
    s.Group("/", func(group *ghttp.RouterGroup) {
        group.Middleware(service.Middleware().Response)
        group.Bind(
                exception.NewV1(),
        )
})
```

`utility/err`

`/internal/logic/exception/exception.go`

```
    func (s *sException) Business() error {
        return utility.Err.Skip(10001)
}

// System  gjson.Decode() err
func (s *sException) System() error {
        _, err := gjson.Decode("")
        if err != nil {
                return utility.Err.Sys(err)
        }
        return nil
}
```

```
    Business
{
        "code": 10001,
        "message": "",
        "data": null
}

System
{
        "code": 99999,
        "message": "",
        "data": null
}
```

blocked URL

Github

`err` Github/oldme-api