全部开发者教程

1. 有哪些规范我们应该遵循
2. git的简单规范
3. go代码规范
4. go项目目录规范
5. 微服务该采用multi-repo还是mono-repo?
6. 微服务的目录结构（mono-repo）
7. go代码的检测工具
8. go中常见的错误

第22周 设计模式和单元测试

1. go最常用的设计模式 – 函数选项
2. 单例模式和懒加载
3. 测试金字塔

bobby · 更新于 2022-11-16

上一节 4. 商品分类接口 6. 品牌分类 下一节

1. form

代码块

```
    package forms

    type BannerForm struct {
        Image      string `form:"image" json:"image" binding:"url"`
        Index      int    `form:"index" json:"index" binding:"required"`
        Url      string `form:"url" json:"url" binding:"url"`
    }
```

2. handler

代码块

```
package banners

import (
    "context"
    "github.com/gin-gonic/gin"
    "github.com/go-playground/validator/v10"
    "github.com/golang/protobuf/ptypes/empty"
    "google.golang.org/grpc/codes"
    "google.golang.org/grpc/status"
    "mxshop-api/goods-web/forms"
    "mxshop-api/goods-web/global"
    "mxshop-api/goods-web/proto"
    "net/http"
    "strconv"
    "strings"
)

func removeTopStruct(fields map[string]string) map[string]string{
    rsp := map[string]string{}
    for field, err := range fields {
        rsp[field[strings.Index(field, ".")+1:]] = err
    }
    return rsp
}


func HandleGrpcErrorToHttp(err error, c *gin.Context) {
    //将grpc的code转换成http的状态码
    if err != nil {
        if e, ok := status.FromError(err); ok {
```

```
32              case codes.NotFound:
33                  c.JSON(http.StatusNotFound, gin.H{
34                      "msg": e.Message(),
35                  })
36              case codes.Internal:
37                  c.JSON(http.StatusInternalServerError, gin.H{
38                      "msg:": "内部错误",
39                  })
40              case codes.InvalidArgument:
41                  c.JSON(http.StatusBadRequest, gin.H{
42                      "msg": "参数错误",
43                  })
44              case codes.Unavailable:
45                  c.JSON(http.StatusInternalServerError, gin.H{
46                      "msg": "用户服务不可用",
47                  })
48              default:
49                  c.JSON(http.StatusInternalServerError, gin.H{
50                      "msg": e.Code(),
51                  })
52              }
53              return
54          }
55      }
56  }
57
58  func HandleValidatorError(c *gin.Context, err error){
59      errs, ok := err.(validator.ValidationErrors)
60      if !ok {
61          c.JSON(http.StatusOK, gin.H{
62              "msg":err.Error(),
63          })
64      }
65      c.JSON(http.StatusBadRequest, gin.H{
66          "error": removeTopStruct(errs.Translate(global.Trans)),
67      })
68      return
69  }
70
71
72  func BannerList(ctx *gin.Context) {
73      rsp, err := global.GoodsSrvClient.BannerList(context.Background(), &empty.Emp
74      if err != nil {
75          HandleGrpcErrorToHttp(err, ctx)
76          return
77      }
78
79      result := make([]interface{}, 0)
80      for _, value := range rsp.Data {
81          reMap := make(map[string]interface{})
82          reMap["id"] = value.Id
83          reMap["index"] = value.Index
84          reMap["image"] = value.Image
85          reMap["url"] = value.Url
86
87          result = append(result, reMap)
88      }
89
90      ctx.JSON(http.StatusOK, result)
91  }
92
93  func NewBanner(ctx *gin.Context) {
94      bannerForm := forms.BannerForm{}
```

```
97          return
98      }
99
100     rsp, err := global.GoodsSrvClient.CreateBanner(context.Background(), &proto.B
101         Index:      int32(bannerForm.Index),
102         Url: bannerForm.Url,
103         Image: bannerForm.Image,
104     })
105
106     if err != nil {
107         HandleGrpcErrorToHttp(err, ctx)
108         return
109     }
110
111     response := make(map[string]interface{})
112     response["id"] = rsp.Id
113     response["index"] = rsp.Index
114     response["url"] = rsp.Url
115     response["image"] = rsp.Image
116
117     ctx.JSON(http.StatusOK, response)
118 }
119
120 func UpdateBanner(ctx *gin.Context) {
121     bannerForm := forms.BannerForm{}
122     if err := ctx.ShouldBindJSON(&bannerForm); err != nil {
123         HandleValidatorError(ctx, err)
124         return
125     }
126
127     id := ctx.Param("id")
128     i, err := strconv.ParseInt(id, 10, 32)
129     if err != nil {
130         ctx.Status(http.StatusNotFound)
131         return
132     }
133
134     _, err = global.GoodsSrvClient.UpdateBanner(context.Background(), &proto.Banr
135         Id:         int32(i),
136         Index:      int32(bannerForm.Index),
137         Url:       bannerForm.Url,
138     })
139     if err != nil {
140         HandleGrpcErrorToHttp(err, ctx)
141         return
142     }
143
144     ctx.Status(http.StatusOK)
145 }
146
147
148 func DeleteBanner(ctx *gin.Context) {
149     id := ctx.Param("id")
150     i, err := strconv.ParseInt(id, 10, 32)
151     if err != nil {
152         ctx.Status(http.StatusNotFound)
153         return
154     }
155     _, err = global.GoodsSrvClient.DeleteBanner(context.Background(), &proto.Banr
156     if err != nil {
157         HandleGrpcErrorToHttp(err, ctx)
158         return
159     }
```

意见反馈 收藏教程 标记书签

```
162     ctx.JSON(http.StatusOK, "")
    }
```

3. router

代码块

```
1   package router
2
3   import (
4       "github.com/gin-gonic/gin"
5       "mxshop-api/goods-web/api/banners"
6       "mxshop-api/goods-web/middlewares"
7   )
8
9   func InitBannerRouter(Router *gin.RouterGroup) {
10      BannerRouter := Router.Group("banners")
11      {
12          BannerRouter.GET("", banners.BannerList)          // 轮播图列表页
13          BannerRouter.DELETE("/:id", middlewares.JWTAuth(), middlewares.IsAdminAut
14          BannerRouter.POST("",  middlewares.JWTAuth(), middlewares.IsAdminAuth(),
15          BannerRouter.PUT("/:id", middlewares.JWTAuth(), middlewares.IsAdminAuth()
16      }
17  }
```

3. 在初始化中调用router

4. 商品分类接口 上一节 下一节 6. 品牌分类

我要提出意见反馈