bobby · 更新于 2022-11-16

上一节 5. service代码 1. 什么是elastic... 下一节

1. router相关代码

1. message

代码块 预览 复制

```
1   package router
2
3   import (
4       "github.com/gin-gonic/gin"
5       "mxshop-api/userop-web/api/message"
6       "mxshop-api/userop-web/middlewares"
7   )
8
9   func InitMessageRouter(Router *gin.RouterGroup) {
10      MessageRouter := Router.Group("message")
11      {
12          MessageRouter.GET("", middlewares.JWTAuth(), message.List)          // 轮
13          MessageRouter.POST("", middlewares.JWTAuth(), message.New)       //新建轮
14      }
15  }
```

2. address

代码块

```
1   package router
2
3   import (
4       "github.com/gin-gonic/gin"
5       "mxshop-api/userop-web/api/address"
6       "mxshop-api/userop-web/middlewares"
7   )
8
9   func InitAddressRouter(Router *gin.RouterGroup) {
10      AddressRouter := Router.Group("address")
11      {
12          AddressRouter.GET("", middlewares.JWTAuth(), address.List)          // 轮
13          AddressRouter.DELETE("/:id", middlewares.JWTAuth(), address.Delete) // 删
14          AddressRouter.POST("", middlewares.JWTAuth(), address.New)       //新建轮
15          AddressRouter.PUT("/:id",middlewares.JWTAuth(), address.Update) //修改轮播
16      }
17  }
```

3. user_fav

代码块

```
package router

import (
    "github.com/gin-gonic/gin"
    "mxshop-api/userop-web/api/user_fav"
    "mxshop-api/userop-web/middlewares"
)

func InitUserFavRouter(Router *gin.RouterGroup) {
    UserFavRouter := Router.Group("userfavs")
    {
        UserFavRouter.DELETE("/:id", middlewares.JWTAuth(), user_fav.Delete) // 
        UserFavRouter.GET("/:id", middlewares.JWTAuth(), user_fav.Detail) // 获取
        UserFavRouter.POST("", middlewares.JWTAuth(), user_fav.New)       //新建收
        UserFavRouter.GET("", middlewares.JWTAuth(), user_fav.List)       //获取当

    }
}
```

2. handler相关代码

1. message

代码块

```
package message

import (
    "context"
    "github.com/gin-gonic/gin"
    "go.uber.org/zap"
    "mxshop-api/userop-web/api"
    "mxshop-api/userop-web/forms"
    "mxshop-api/userop-web/global"
    "mxshop-api/userop-web/proto"
    "net/http"
)

func List(ctx *gin.Context) {
    rsp, err := global.MessageClient.MessageList(context.Background(), &proto.Mes
    if err != nil {
        zap.S().Errorw("获取留言失败")
        api.HandleGrpcErrorToHttp(err, ctx)
        return
    }

    reMap := map[string]interface{}{
        "total": rsp.Total,
    }
    result := make([]interface{}, 0)
    for _, value := range rsp.Data {
        reMap := make(map[string]interface{})
        reMap["id"] = value.Id
        reMap["user_id"] = value.UserId
        reMap["type"] = value.MessageType
        reMap["subject"] = value.Subject
```

```
34
35          result = append(result, reMap)
36      }
37      reMap["data"] = result
38
39      ctx.JSON(http.StatusOK, reMap)
40  }
41
42  func New(ctx *gin.Context) {
43      messageForm := forms.MessageForm{}
44      if err := ctx.ShouldBindJSON(&messageForm); err != nil {
45          api.HandleValidatorError(ctx, err)
46          return
47      }
48
49      rsp, err := global.MessageClient.CreateMessage(context.Background(), &proto.M
50          MessageType: messageForm.MessageType,
51          Subject: messageForm.Subject,
52          Message: messageForm.Message,
53          File: messageForm.File,
54      })
55
56      if err != nil {
57          zap.S().Errorw("添加留言失败")
58          api.HandleGrpcErrorToHttp(err, ctx)
59          return
60      }
61
62      request := make(map[string]interface{})
63      request["id"] = rsp.Id
64
65      ctx.JSON(http.StatusOK, request)
66  }
67
68
```

2. address

代码块

```
1  package address
2
3  import (
4      "context"
5      "github.com/gin-gonic/gin"
6      "go.uber.org/zap"
7      "mxshop-api/userop-web/api"
8      "mxshop-api/userop-web/forms"
9      "mxshop-api/userop-web/global"
10     "mxshop-api/userop-web/models"
11     "mxshop-api/userop-web/proto"
12     "net/http"
13     "strconv"
14 )
15
16 func List(ctx *gin.Context) {
17     request := &proto.AddressRequest{}
18
19     claims, _ := ctx.Get("claims")
20     currentUser := claims.(*models.CustomClaims)
21
```

意见反馈 收藏教程 标记书签

```
24          request.UserId = int32(userId.(uint))
25      }
26
27      rsp, err := global.AddressClient.GetAddressList(context.Background(), request
28      if err != nil {
29          zap.S().Errorw("获取地址列表失败")
30          api.HandleGrpcErrorToHttp(err, ctx)
31          return
32      }
33
34      reMap := map[string]interface{}{
35          "total": rsp.Total,
36      }
37
38      result := make([]interface{}, 0)
39      for _, value := range rsp.Data {
40          reMap := make(map[string]interface{})
41          reMap["id"] = value.Id
42          reMap["user_id"] = value.UserId
43          reMap["province"] = value.Province
44          reMap["city"] = value.City
45          reMap["district"] = value.District
46          reMap["address"] = value.Address
47          reMap["signer_name"] = value.SignerName
48          reMap["signer_mobile"] = value.SignerMobile
49
50          result = append(result, reMap)
51      }
52
53      reMap["data"] = result
54
55      ctx.JSON(http.StatusOK, reMap)
56  }
57
58  func New(ctx *gin.Context) {
59      addressForm := forms.AddressForm{}
60      if err := ctx.ShouldBindJSON(&addressForm); err != nil {
61          api.HandleValidatorError(ctx, err)
62          return
63      }
64
65      rsp, err := global.AddressClient.CreateAddress(context.Background(), &proto.A
66          Province: addressForm.Province,
67          City: addressForm.City,
68          District: addressForm.District,
69          Address: addressForm.Address,
70          SignerName: addressForm.SignerName,
71          SignerMobile: addressForm.SignerMobile,
72      })
73
74      if err != nil {
75          zap.S().Errorw("新建地址失败")
76          api.HandleGrpcErrorToHttp(err, ctx)
77          return
78      }
79
80      request := make(map[string]interface{})
81      request["id"] = rsp.Id
82
83      ctx.JSON(http.StatusOK, request)
84  }
85
86  func Delete(ctx *gin.Context) {
```

```
    if err != nil {
        ctx.Status(http.StatusNotFound)
        return
    }
    _, err = global.AddressClient.DeleteAddress(context.Background(), &proto.Addr
    if err != nil {
        zap.S().Errorw("删除地址失败")
        api.HandleGrpcErrorToHttp(err, ctx)
        return
    }

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

func Update(ctx *gin.Context) {
    addressForm := forms.AddressForm{}
    if err := ctx.ShouldBindJSON(&addressForm); err != nil {
        api.HandleValidatorError(ctx, err)
        return
    }

    id := ctx.Param("id")
    i, err := strconv.ParseInt(id, 10, 32)
    if err != nil {
        ctx.Status(http.StatusNotFound)
        return
    }

    _, err = global.AddressClient.UpdateAddress(context.Background(), &proto.Addr
        Id:   int32(i),
        Province: addressForm.Province,
        City: addressForm.City,
        District: addressForm.District,
        Address: addressForm.Address,
        SignerName: addressForm.SignerName,
        SignerMobile: addressForm.SignerMobile,
    })
    if err != nil {
        zap.S().Errorw("更新地址失败")
        api.HandleGrpcErrorToHttp(err, ctx)
        return
    }
    request := make(map[string]interface{})
    ctx.JSON(http.StatusOK, request)
}

```

3. userfav

代码块

```
package user_fav

import (
    "context"
    "github.com/gin-gonic/gin"
    "go.uber.org/zap"
    "mxshop-api/userop-web/api"
    "mxshop-api/userop-web/forms"
    "mxshop-api/userop-web/global"
    "mxshop-api/userop-web/proto"
```

意见反馈 收藏教程 标记书签

```
)

func List(ctx *gin.Context) {
    userId, _ := ctx.Get("userId")
    userFavRsp, err := global.UserFavClient.GetFavList(context.Background(), &pro
        UserId:  int32(userId.(uint)),
    })
    if err != nil {
        zap.S().Errorw("获取收藏列表失败")
        api.HandleGrpcErrorToHttp(err, ctx)
        return
    }

    ids := make([]int32, 0)
    for _, item := range userFavRsp.Data{
        ids = append(ids, item.GoodsId)
    }

    if len(ids) == 0 {
        ctx.JSON(http.StatusOK, gin.H{
            "total":0,
        })
        return
    }

    //请求商品服务
    goods, err := global.GoodsSrvClient.BatchGetGoods(context.Background(), &prot
        Id: ids,
    })
    if err != nil {
        zap.S().Errorw("[List] 批量查询【商品列表】失败")
        api.HandleGrpcErrorToHttp(err, ctx)
        return
    }

    reMap := map[string]interface{}{
        "total": userFavRsp.Total,
    }

    goodsList := make([]interface{}, 0)
    for _, item := range userFavRsp.Data{
        data := gin.H{
            "id":item.GoodsId,
        }

        for _, good := range goods.Data {
            if item.GoodsId == good.Id {
                data["name"] = good.Name
                data["shop_price"] = good.ShopPrice
            }
        }

        goodsList = append(goodsList, data)
    }
    reMap["data"] = goodsList
    ctx.JSON(http.StatusOK, reMap)
}

func New(ctx *gin.Context) {
    userFavForm := forms.UserFavForm{}
    if err := ctx.ShouldBindJSON(&userFavForm); err != nil {
        api.HandleValidatorError(ctx, err)
        return
```

```
78      //缺少一步，  这个时候应该去商品服务查询一下这个是否存在
79      userId, _ := ctx.Get("userId")
80      _, err := global.UserFavClient.AddUserFav(context.Background(), &proto.UserFa
81          UserId:  int32(userId.(uint)),
82          GoodsId: userFavForm.GoodsId,
83      })
84
85      if err != nil {
86          zap.S().Errorw("添加收藏记录失败")
87          api.HandleGrpcErrorToHttp(err, ctx)
88          return
89      }
90
91      ctx.JSON(http.StatusOK, gin.H{})
92  }
93
94  func Delete(ctx *gin.Context) {
95      id := ctx.Param("id")
96      i, err := strconv.ParseInt(id, 10, 32)
97      if err != nil {
98          ctx.Status(http.StatusNotFound)
99          return
100     }
101
102     userId, _ := ctx.Get("userId")
103     _, err = global.UserFavClient.DeleteUserFav(context.Background(), &proto.User
104         UserId:  int32(userId.(uint)),
105         GoodsId: int32(i),
106     })
107     if err != nil {
108         zap.S().Errorw("删除收藏记录失败")
109         api.HandleGrpcErrorToHttp(err, ctx)
110         return
111     }
112
113     ctx.JSON(http.StatusOK, gin.H{
114         "msg":"删除成功",
115     })
116 }
117
118 func Detail(ctx *gin.Context) {
119     goodsId := ctx.Param("id")
120     goodsIdInt, err := strconv.ParseInt(goodsId, 10, 32)
121     if err != nil {
122         ctx.Status(http.StatusNotFound)
123         return
124     }
125     userId, _ := ctx.Get("userId")
126     _, err = global.UserFavClient.GetUserFavDetail(context.Background(), &proto.U
127         UserId:  int32(userId.(uint)),
128         GoodsId: int32(goodsIdInt),
129     })
130     if err != nil {
131         zap.S().Errorw("查询收藏状态失败")
132         api.HandleGrpcErrorToHttp(err, ctx)
133         return
134     }
135
136     ctx.Status(http.StatusOK)
137 }
```

意见反馈 收藏教程 标记书签

1. message

代码块

```
package forms

type MessageForm struct {
    MessageType    int32 `form:"type" json:"type" binding:"required,oneof=1 2 3 4
    Subject string `form:"subject" json:"subject" binding:"required"`
    Message string `form:"message" json:"message" binding:"required"`
    File string `form:"file" json:"file" binding:"required"`
}
```

2. address

代码块

```
package forms

type AddressForm struct {
    Province    string `form:"province" json:"province" binding:"required"`
    City string `form:"city" json:"city" binding:"required"`
    District string `form:"district" json:"district" binding:"required"`
    Address string `form:"address" json:"address" binding:"required"`
    SignerName string `form:"signer_name" json:"signer_name" binding:"required"`
    SignerMobile string `form:"signer_mobile" json:"signer_mobile" binding:"requi
}
```

2. userfav

代码块

```
package forms

type UserFavForm struct {
    GoodsId    int32 `form:"goods" json:"goods" binding:"required"`
}
```

5. service代码 上一节 下一节 1. 什么是elasticsearch

我要提出意见反馈

企业服务 网站地图 网站首页 关于我们 联系我们 讲师招募 帮助中心 意见反馈 代码托管