/controllerservice

1controller

```
// @summary 修改内容
// @tags    前台-内容
// @produce json
// @param   entity body define.ContentDoUpdateReq true "请求参数" required
// @router  /content/do-update [POST]
// @success 200 {object} response.JsonRes "请求结果"
func (a *contentApi) DoUpdate(r *ghttp.Request) {
    input := define.ContentUpdateInput{
        Id: r.GetUint("id"),
        ContentCreateUpdateBase: define.ContentCreateUpdateBase{
            Type:       r.GetString("type"),
            CategoryId: r.GetUint("category_id"),
            Title:      r.GetString("title"),
            Content:    r.GetString("content"),
            Brief:      r.GetString("brief"),
            Thumb:      r.GetString("thumb"),
            Referer:    r.GetString("referer"),
        },
    }
    if err := service.Content.Update(r.Context(), input); err != nil {
        response.JsonExit(r, 1, err.Error())
    } else {
        response.JsonExit(r, 0, "")
    }
}
```

- /
- HttpRequest/HttpContext/
-
-

2service

```
func (s *sAutoWork) UnbindRoute(
    ctx context.Context, appId int64, vip string, uniqVpcId string, uniqSubnetId string, instanceId string, vPort int, flow flowdriver.IFlowOperator,
) error {
    vpcLgc := NewVpcLogic(ctx)
    vpcId, subnetId, err := vpcLgc.UniqVpcToId(appId, uniqVpcId, uniqSubnetId)
    if err != nil {
        return flowdriver.PauseError(err)
    }
    flow.TFill("vpcId", vpcId)
    flow.TFill("subnetId", subnetId)
    flow.TFill("instanceId", instanceId)
    if IFlagDo("delTcpRoute", flow) {
        err := RetryFunc(5, 2, func() error {
            return vpcLgc.DeleteRoute(vpcId, vip, vPort)
        })
        if err != nil {
            return flowdriver.PauseError(err)
        }
        FlagDone("delTcpRoute", "done", flow)
    }
    if IFlagDo("delTcpService", flow) {
        err := RetryFunc(5, 2, func() error {
            return vpcLgc.DeleteService(vpcId, vip, vPort)
        })
        if err != nil {
            return flowdriver.PauseError(err)
        }
```

-
-
-

1controller

- /
-
- error
-

```go
type ContentUpdateReq struct {
    g.Meta      `path:"/content/update/{Id}" method:"post" tags:"内容" summary:"修改内容接口"`
    Id          uint     json:"id"       v:"min:1#请选择需要修改的内容" dc:"内容Id"
    Type        string   json:"type"     v:"required#内容模型不能为空" dc:"内容模型"
    CategoryId  uint     json:"cate"     v:"required#请选择栏目"      dc:"栏目ID"
    Title       string   json:"title"    v:"required#请输入标题"      dc:"标题"
    Content     string   json:"content"  v:"required#请输入内容"      dc:"内容"
    Brief       string   json:"brief"    dc:"摘要"
    Thumb       string   json:"thumb"    dc:"缩略图"
    Tags        []string json:"tags"     dc:"标签名称列表，以JSON存储"
    Referer     string   json:"referer"  dc:"内容来源，例如github/gitee"
}
type ContentUpdateRes struct{}
```

```go
func (a *hContent) Update(ctx context.Context, req *apiv1.ContentUpdateReq) (res *apiv1.ContentUpdateRes, err error) {
    err = service.Content().Update(ctx, model.ContentUpdateInput{
        Id: req.Id,
        ContentCreateUpdateBase: model.ContentCreateUpdateBase{
            Type:       req.Type,
            CategoryId: req.CategoryId,
            Title:      req.Title,
            Content:    req.Content,
            Brief:      req.Brief,
            Thumb:      req.Thumb,
            Tags:       req.Tags,
            Referer:    req.Referer,
        },
    })
    return
}
```

2service

-
-
-

```go
type UnbindRouteInput struct {
    AppId       int64               // Certain application key, necessary parameter for Network operations.
    Vip         string              // Custom virtual IP address, like: 192.168.1.1.
    UniqVpcId   string              // Unique VPC id.
    UniqSubnetId string             // Unique subnet id.
    InstanceId  string              // Unique business instance id.
    VPort       int                 // Virtual port that will be released.
    Flow        flowdriver.IFlowOperator // Context workflow object.
}

func (s *sNetwork) UnbindRoute(ctx context.Context, in UnbindRouteInput) error {
    vpcLgc := NewVpcLogic(ctx)
    vpcId, subnetId, err := vpcLgc.UniqVpcToId(in.AppId, in.UniqVpcId, in.UniqSubnetId)
    if err != nil {
        return flowdriver.PauseError(err)
    }
    in.Flow.TFill("vpcId", vpcId)
    in.Flow.TFill("subnetId", subnetId)
    in.Flow.TFill("instanceId", in.InstanceId)
    if !flagDo("deltegRoute", in.Flow) {
        err := RetryFunc(3, 2, func() error {
            return vpcLgc.DeleteRoute(vpcId, in.Vip, in.VPort)
        })
```

- service/