



《手写OS操作系统》小班二期招生，全程直播授课，大牛带你掌握硬核技术！

点此查

慕课网首页 免费课 实战课 体系课 慕课教程 专栏 手记 企业服务

Q 购物车 通知 我的

从所有教程的词条中查询...

首页 > 慕课教程 > Go工程师体系课全新版 > 8. 品牌分类接口

全部开发者教程 三

第20周 api网关、部署

1. api网关对比

2. kong的安装和配置

3. 配置jwt

4. 什么是敏捷开发

5. jenkins的安装和配置

6. jenkins的插件管理

7. 部署到远程服务器并运行

8. jenkins的pipeline参数详解

9. 定时构建的语法

10. shell脚本启动go服务

第21周 开发规范和go基础扩展

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



bobby · 更新于 2022-11-08

← 上一节 7. 品牌相关的接口 1. 商品列表接口 下一节 →

1. CategoryBrandList

<> 代码块

预览 复制

```
1 func (s *GoodsServer) CategoryBrandList(ctx context.Context, req *proto.CategoryBrandListRequest) (*proto.CategoryBrandListResponse, error) {
2     var categoryBrands []model.GoodsCategoryBrand
3     categoryBrandListResponse := proto.CategoryBrandListResponse{}
4
5     var total int64
6     global.DB.Model(&model.GoodsCategoryBrand{}).Count(&total)
7     categoryBrandListResponse.Total = int32(total)
8
9     global.DB.Scopes(Paginate(int(req.Pages), int(req.PagePerNums))).Find(&categoryBrands)
10
11     var categoryResponses []*proto.CategoryBrandResponse
12     for _, categoryBrand := range categoryBrands {
13         categoryResponses = append(categoryResponses, &proto.CategoryBrandResponse{
14             Category: &proto.CategoryInfoResponse{
15                 Id: categoryBrand.Category.ID,
16                 Name: categoryBrand.Category.Name,
17                 Level: categoryBrand.Category.Level,
18                 IsTab: categoryBrand.Category.IsTab,
19                 ParentCategory: categoryBrand.Category.ParentCategoryID,
20             },
21             Brand: &proto.BrandInfoResponse{
22                 Id: categoryBrand.Brands.ID,
23                 Name: categoryBrand.Brands.Name,
24                 Logo: categoryBrand.Brands.Logo,
25             },
26         })
27     }
28
29     categoryBrandListResponse.Data = categoryResponses
30     return &categoryBrandListResponse, nil
31 }
```

2. GetCategoryBrandList

<> 代码块

```
1 func (s *GoodsServer) GetCategoryBrandList(ctx context.Context, req *proto.GetCategoryBrandListRequest) (*proto.GetCategoryBrandListResponse, error) {
2     brandListResponse := proto.BrandListResponse{}
3
4     var category model.Category
5     global.DB.Find(&category, req.CategoryID)
6     if category.ID == 0 {
7         return &proto.GetCategoryBrandListResponse{
8             Result: Rows,
9         }, errors.New("category not found")
10     }
11     global.DB.Find(&categoryBrands, req.CategoryID)
12     if len(categoryBrands) == 0 {
13         return &proto.GetCategoryBrandListResponse{
14             Result: Rows,
15         }, errors.New("category brands not found")
16     }
17     return &proto.GetCategoryBrandListResponse{
18         Result: Rows,
19     }, nil
20 }
```

意见反馈

收藏教程

标记书签



```
7     }
8
9     var categoryBrands []model.GoodsCategoryBrand
10    if result := global.DB.Where(&model.GoodsCategoryBrand{CategoryID: category.},
11        brandListResponse.Total = int32(result.RowsAffected)
12    }
13
14    var brandInfoResponses []*proto.BrandInfoResponse
15    for _, categoryBrand := range categoryBrands {
16        brandInfoResponses = append(brandInfoResponses, &proto.BrandInfoResponse{
17            Id: int32(categoryBrand.Brand.ID),
18            Name: categoryBrand.Brand.Name,
19            Logo: categoryBrand.Brand.Logo,
20        })
21    }
22
23    brandListResponse.Data = brandInfoResponses
24
25    return &brandListResponse, nil
26 }
```

3. CreateCategoryBrand

<> 代码块

```
1 func (s *GoodsServer) CreateCategoryBrand(ctx context.Context, req *proto.CategoryBrandRequest) (*proto.CategoryBrandResponse, error) {
2     var category model.Category
3     if result := global.DB.First(&category, req.CategoryId); result.RowsAffected == 0 {
4         return nil, status.Errorf(codes.InvalidArgument, "商品分类不存在")
5     }
6
7     var brand model.Brands
8     if result := global.DB.First(&brand, req.BrandId); result.RowsAffected == 0 {
9         return nil, status.Errorf(codes.InvalidArgument, "品牌不存在")
10    }
11
12    categoryBrand := model.GoodsCategoryBrand{
13        CategoryID: req.CategoryId,
14        BrandID: req.BrandId,
15    }
16
17    global.DB.Save(&categoryBrand)
18    return &proto.CategoryBrandResponse{Id: int32(categoryBrand.ID)}, nil
19 }
```

4. DeleteCategoryBrand

<> 代码块

```
1 func (s *GoodsServer) DeleteCategoryBrand(ctx context.Context, req *proto.CategoryBrandRequest) (*proto.CategoryBrandResponse, error) {
2     if result := global.DB.Delete(&model.GoodsCategoryBrand{}, req.Id); result.RowsAffected == 0 {
3         return nil, status.Errorf(codes.NotFound, "品牌分类不存在")
4     }
5     return &emptypb.Empty{}, nil
6 }
```

5. UpdateCategoryBrand

<> 代码块

```
1 func (s *GoodsServer) UpdateCategoryBrand(ctx context.Context, req *proto.CategoryBrand) error {
2     var categoryBrand model.GoodsCategoryBrand
3
4     if result := global.DB.First(&categoryBrand, req.Id); result.RowsAffected == 0 {
5         return nil, status.Errorf(codes.InvalidArgument, "品牌分类不存在")
6     }
7
8     var category model.Category
9     if result := global.DB.First(&category, req.CategoryId); result.RowsAffected == 0 {
10        return nil, status.Errorf(codes.InvalidArgument, "商品分类不存在")
11    }
12
13    var brand model.Brands
14    if result := global.DB.First(&brand, req.BrandId); result.RowsAffected == 0 {
15        return nil, status.Errorf(codes.InvalidArgument, "品牌不存在")
16    }
17
18    categoryBrand.CategoryID = req.CategoryId
19    categoryBrand.BrandID = req.BrandId
20
21    global.DB.Save(&categoryBrand)
22
23    return &emptypb.Empty{}, nil
24 }
```

7. 品牌相关的接口 ◀ 上一节 下一节 ▶ 1. 商品列表接口

✎ 我要提出意见反馈