



《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！

点此查看

首页 > 慕课教程 > Go工程师体系课全新版 > 3. 商品列表页功能

全部开发者教程

16. grpc的超时和重试

17. 常见的幂等性解决方案

第19周 链路追踪、限流、熔断、降级

1. 什么是链路追踪

2. 链路追踪技术选型

3. jaeger安装和架构

4. opentracing解析

5. grpc下添加jaeger

6. 熔断、限流技术选型

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



bobby · 更新于 2022-11-08

上一节 2. protobuf定义 4. 库存的微服务... 下一节

1. model转message

<> 代码块

```
1 goodsInfoResponse := proto.GoodsInfoResponse {
2     Id:      goods.ID,
3     CategoryId: goods.CategoryID,
4     Name: goods.Name,
5     GoodsSn: goods.GoodsSn,
6     ClickNum: goods.ClickNum,
7     SoldNum: goods.SoldNum,
8     FavNum: goods.FavNum,
9     MarketPrice: goods.MarketPrice,
10    ShopPrice: goods.ShopPrice,
11    GoodsBrief: goods.GoodsBrief,
12    ShipFree: goods.ShipFree,
13    GoodsFrontImage: goods.GoodsFrontImage,
14    IsNew: goods.IsNew,
15    IsHot: goods.IsHot,
16    OnSale: goods.OnSale,
17    DescImages: goods.DescImages,
18    Images: goods.Images,
19    Category: &proto.CategoryBriefInfoResponse{
20        Id:      goods.Category.ID,
21        Name: goods.Category.Name,
22    },
23    Brand: &proto.BrandInfoResponse{
24        Id:      goods.Brands.ID,
25        Name: goods.Brands.Name,
26        Logo: goods.Brands.Logo,
27    },
28 }
```

2. goodslist

<> 代码块

```
1 func (s *GoodsServer) GoodsList(ctx context.Context, req *proto.GoodsFilterRequest) {
2     //关键词搜索、查询新品、查询热门商品、通过价格区间筛选, 通过商品分类筛选
3     goodsListResponse := &proto.GoodsListResponse{}
4
5     var goods []model.Goods
6     localDB := global.DB.Model(model.Goods{})
7     if req.KeyWords != "" {
8         //搜索
9         localDB = localDB.Where("name LIKE ?", "%"+req.KeyWords+"%")
10    }
```

意见反馈

收藏教程

标记书签

```
13     }
14     if req.IsNew {
15         localDB = localDB.Where(model.Goods{IsNew: true})
16     }
17
18     if req.PriceMin > 0 {
19         localDB = localDB.Where("shop_price>=?", req.PriceMin)
20     }
21     if req.PriceMax > 0 {
22         localDB = localDB.Where("shop_price<=?", req.PriceMax)
23     }
24
25     if req.Brand > 0 {
26         localDB = localDB.Where("brand_id=?", req.Brand)
27     }
28
29     //通过category去查询商品
30     var subQuery string
31     if req.TopCategory > 0 {
32         var category model.Category
33         if result := global.DB.First(&category, req.TopCategory); result.RowsAffected == 0 {
34             return nil, status.Errorf(codes.NotFound, "商品分类不存在")
35         }
36
37         if category.Level == 1 {
38             subQuery = fmt.Sprintf("select id from category where parent_category_id=0")
39         } else if category.Level == 2 {
40             subQuery = fmt.Sprintf("select id from category WHERE parent_category_id=%d", req.TopCategory)
41         } else if category.Level == 3 {
42             subQuery = fmt.Sprintf("select id from category WHERE id=%d", req.TopCategory)
43         }
44
45         localDB = localDB.Where(fmt.Sprintf("category_id in (%s)", subQuery))
46     }
47
48     var count int64
49     localDB.Count(&count)
50     goodsListResponse.Total = int32(count)
51
52     result := localDB.Preload("Category").Preload("Brands").Scopes(Paginate(int(req.Page), int(req.PageSize))).Find(&goods)
53     if result.Error != nil {
54         return nil, result.Error
55     }
56
57     for _, good := range goods {
58         goodsInfoResponse := ModelToResponse(good)
59         goodsListResponse.Data = append(goodsListResponse.Data, &goodsInfoResponse)
60     }
61
62     return goodsListResponse, nil
63 }
```

3. 批量获取商品信息

<> 代码块

```
1 func (s *GoodsServer) BatchGetGoods(ctx context.Context, req *proto.BatchGoodsIdRequest) (*proto.BatchGoodsResponse, error) {
2     goodsListResponse := &proto.BatchGoodsResponse{}
3     var goods []model.Goods
4
5     //调用where并不会真正执行sql 只是用来生成sql的 当调用find, first才会去执行sql,
6     //这里我们使用find, first来生成sql, 然后使用find, first来执行sql, 最后使用find, first来返回结果
```

意见反馈

收藏教程

标记书签



```
8         goodsInfoResponse := ModelToResponse(good)
9         goodsListResponse.Data = append(goodsListResponse.Data, &goodsInfoResponse)
10    }
11    goodsListResponse.Total = int32(result.RowsAffected)
12    return goodsListResponse, nil
13 }
```

4. 获取商品的详情

<> 代码块

```
1 func (s *GoodsServer) GetGoodsDetail(ctx context.Context, req *proto.GoodInfoRequest) (*proto.GoodInfoResponse, error) {
2     var goods model.Goods
3
4     if result := global.DB.First(&goods, req.Id); result.RowsAffected == 0 {
5         return nil, status.Errorf(codes.NotFound, "商品不存在")
6     }
7     goodsInfoResponse := ModelToResponse(goods)
8     return &goodsInfoResponse, nil
9 }
10
```

5. 添加商品

<> 代码块

```
1 func (s *GoodsServer) CreateGoods(ctx context.Context, req *proto.CreateGoodsInfoRequest) (*proto.GoodInfoResponse, 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    //这里没有看到图片文件是如何上传， 在微服务中 普通的文件上传已经不再使用
12    goods := model.Goods{
13        Brands: brand,
14        BrandsID: brand.ID,
15        Category: category,
16        CategoryID: category.ID,
17        Name: req.Name,
18        GoodsSn: req.GoodsSn,
19        MarketPrice: req.MarketPrice,
20        ShopPrice: req.ShopPrice,
21        GoodsBrief: req.GoodsBrief,
22        ShipFree: req.ShipFree,
23        Images: req.Images,
24        DescImages: req.DescImages,
25        GoodsFrontImage: req.GoodsFrontImage,
26        IsNew: req.IsNew,
27        IsHot: req.IsHot,
28        OnSale: req.OnSale,
29    }
30
31    global.DB.Save(&goods)
32    return &proto.GoodInfoResponse{
33        Id: goods.ID,
34    }, nil
35 }
```



意见反馈

收藏教程

标记书签

6. 删除商品

<> 代码块

```
1 func (s *GoodsServer) DeleteGoods(ctx context.Context, req *proto.DeleteGoodsInfo) error {
2     if result := global.DB.Delete(&model.Goods{}, req.Id); result.RowsAffected == 0 {
3         return nil, status.Errorf(codes.NotFound, "商品不存在")
4     }
5     return &emptypb.Empty{}, nil
6 }
```

7. 更新商品

<> 代码块

```
1 func (s *GoodsServer) UpdateGoods(ctx context.Context, req *proto.CreateGoodsInfo) error {
2     var goods model.Goods
3
4     if result := global.DB.First(&goods, req.Id); result.RowsAffected == 0 {
5         return nil, status.Errorf(codes.InvalidArgument, "商品不存在")
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     goods.Brands = brand
19     goods.Category = category
20     goods.Name = req.Name
21     goods.GoodsSn = req.GoodsSn
22     goods.MarketPrice = req.MarketPrice
23     goods.ShopPrice = req.ShopPrice
24     goods.GoodsBrief = req.GoodsBrief
25     goods.ShipFree = req.ShipFree
26     goods.Images = req.Images
27     goods.DescImages = req.DescImages
28     goods.GoodsFrontImage = req.GoodsFrontImage
29     goods.IsNew = req.IsNew
30     goods.IsHot = req.IsHot
31     goods.OnSale = req.OnSale
32
33     global.DB.Save(&goods)
34     return &emptypb.Empty{}, nil
35 }
```

2. protobuf定义 ◀ 上一节

下一节 ▶ 4. 库存的微服务设计

✎ 我要提出意见反馈

✎ 意见反馈

♡ 收藏教程

🔖 标记书签

