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

```
goodsInfoResponse := proto.GoodsInfoResponse {
        Id:       goods.ID,
        CategoryId: goods.CategoryID,
        Name: goods.Name,
        GoodsSn: goods.GoodsSn,
        ClickNum: goods.ClickNum,
        SoldNum: goods.SoldNum,
        FavNum: goods.FavNum,
        MarketPrice: goods.MarketPrice,
        ShopPrice: goods.ShopPrice,
        GoodsBrief: goods.GoodsBrief,
        ShipFree: goods.ShipFree,
        GoodsFrontImage: goods.GoodsFrontImage,
        IsNew: goods.IsNew,
        IsHot: goods.IsHot,
        OnSale: goods.OnSale,
        DescImages: goods.DescImages,
        Images: goods.Images,
        Category: &proto.CategoryBriefInfoResponse{
            Id:   goods.Category.ID,
            Name: goods.Category.Name,
        },
        Brand: &proto.BrandInfoResponse{
            Id:   goods.Brands.ID,
            Name: goods.Brands.Name,
            Logo: goods.Brands.Logo,
        },
    }
```

2. goodslist

<> 代码块

```
func (s *GoodsServer) GoodsList(ctx context.Context, req *proto.GoodsFilterReques
    //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
    goodsListResponse := &proto.GoodsListResponse{}

    var goods []model.Goods
    localDB := global.DB.Model(model.Goods{})
    if req.KeyWords != "" {
        //搜索
        localDB = localDB.Where("name LIKE ?", "%"+req.KeyWords+"%")
    }
    }
    if req.IsNew {
        localDB = localDB.Where(model.Goods{IsNew: true})
    }

    if req.PriceMin > 0 {
        localDB = localDB.Where("shop_price>=?", req.PriceMin)
    }
    if req.PriceMax > 0 {
        localDB = localDB.Where("shop_price<=?", req.PriceMax)
    }

    if req.Brand > 0 {
        localDB = localDB.Where("brand_id=?", req.Brand)
    }

    //通过category去查询商品
    var subQuery string
    if req.TopCategory > 0 {
        var category model.Category
        if result := global.DB.First(&category, req.TopCategory); result.RowsAffe
            return nil, status.Errorf(codes.NotFound, "商品分类不存在")
        }

        if category.Level == 1 {
            subQuery = fmt.Sprintf("select id from category where parent_category
        }else if category.Level == 2 {
            subQuery = fmt.Sprintf("select id from category WHERE parent_category
        }else if category.Level == 3 {
            subQuery = fmt.Sprintf("select id from category WHERE id=%d", req.Top
        }

        localDB = localDB.Where(fmt.Sprintf("category_id in (%s)", subQuery))
    }

    var count int64
    localDB.Count(&count)
    goodsListResponse.Total = int32(count)

    result := localDB.Preload("Category").Preload("Brands").Scopes(Paginate(int(
    if result.Error != nil {
        return nil, result.Error
    }

    for _, good := range goods {
        goodsInfoResponse := ModelToResponse(good)
        goodsListResponse.Data = append(goodsListResponse.Data, &goodsInfoRespons
    }

    return goodsListResponse, nil
}
```

3. 批量获取商品信息

<> 代码块

```
func (s *GoodsServer) BatchGetGoods(ctx context.Context, req *proto.BatchGoodsIdI
    goodsListResponse := &proto.GoodsListResponse{}
    var goods []model.Goods

    //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,
```

意见反馈 收藏教程 标记书签

```go
            goodsInfoResponse := ModelToResponse(good)
            goodsListResponse.Data = append(goodsListResponse.Data, &goodsInfoRespons
        }
        goodsListResponse.Total = int32(result.RowsAffected)
        return goodsListResponse, nil
    }
```

4. 获取商品的详情

<> 代码块

```go
    func (s *GoodsServer) GetGoodsDetail(ctx context.Context, req *proto.GoodInfoRequ
        var goods model.Goods

        if result := global.DB.First(&goods, req.Id); result.RowsAffected == 0 {
            return nil, status.Errorf(codes.NotFound, "商品不存在")
        }
        goodsInfoResponse := ModelToResponse(goods)
        return &goodsInfoResponse, nil
    }

```

5. 添加商品

<> 代码块

```go
    func (s *GoodsServer) CreateGoods(ctx context.Context, req *proto.CreateGoodsInfo
        var category model.Category
        if result := global.DB.First(&category, req.CategoryId); result.RowsAffected
            return nil, status.Errorf(codes.InvalidArgument, "商品分类不存在")
        }

        var brand model.Brands
        if result := global.DB.First(&brand, req.BrandId); result.RowsAffected == 0 {
            return nil, status.Errorf(codes.InvalidArgument, "品牌不存在")
        }
        //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
        goods := model.Goods{
            Brands: brand,
            BrandsID: brand.ID,
            Category: category,
            CategoryID: category.ID,
            Name: req.Name,
            GoodsSn: req.GoodsSn,
            MarketPrice: req.MarketPrice,
            ShopPrice: req.ShopPrice,
            GoodsBrief: req.GoodsBrief,
            ShipFree: req.ShipFree,
            Images: req.Images,
            DescImages: req.DescImages,
            GoodsFrontImage: req.GoodsFrontImage,
            IsNew: req.IsNew,
            IsHot: req.IsHot,
            OnSale: req.OnSale,
        }

        global.DB.Save(&goods)
        return &proto.GoodsInfoResponse{
            Id:  goods.ID,
```

意见反馈 收藏教程 标记书签

6. 删除商品

<> 代码块

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

7. 更新商品

<> 代码块

```
func (s *GoodsServer) UpdateGoods(ctx context.Context, req *proto.CreateGoodsInfo
    var goods model.Goods

    if result := global.DB.First(&goods, req.Id); result.RowsAffected == 0 {
        return nil, status.Errorf(codes.InvalidArgument, "商品不存在")
    }

    var category model.Category
    if result := global.DB.First(&category, req.CategoryId); result.RowsAffected
        return nil, status.Errorf(codes.InvalidArgument, "商品分类不存在")
    }

    var brand model.Brands
    if result := global.DB.First(&brand, req.BrandId); result.RowsAffected == 0 
        return nil, status.Errorf(codes.InvalidArgument, "品牌不存在")
    }

    goods.Brands = brand
    goods.Category = category
    goods.Name = req.Name
    goods.GoodsSn = req.GoodsSn
    goods.MarketPrice= req.MarketPrice
    goods.ShopPrice= req.ShopPrice
    goods.GoodsBrief= req.GoodsBrief
    goods.ShipFree= req.ShipFree
    goods.Images= req.Images
    goods.DescImages= req.DescImages
    goods.GoodsFrontImage= req.GoodsFrontImage
    goods.IsNew= req.IsNew
    goods.IsHot= req.IsHot
    goods.OnSale= req.OnSale

    global.DB.Save(&goods)
    return &emptypb.Empty{}, nil
}
```

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

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

意见反馈 收藏教程 标记书签