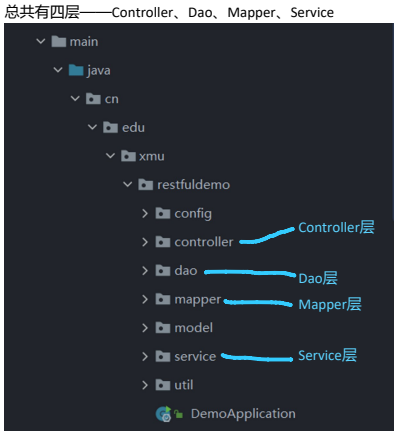
层次结构及对象定义

2021年10月6日 20:46

以RestfulDemoMyBatis为例

层次结构

总共有四层——Controller、Dao、Mapper、Service



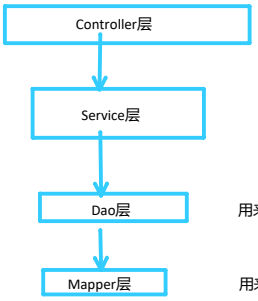
model包不能称之为一层，model包中的对象时要用在这四层中间的

config包用来放一些需要用java代码来做配置的类，如跨域访问、Swagger

util包是实用工具包，内部是工具类

四层的职责

这四层是严格分层的，**只能向下调用，不可跨层调用**



负责接收前端的数据；把数据传给后端；封装要返回给前端的数据；处理一些错误码、异常

主要用来写业务代码

用来访问数据，或者说从数据库中拿到数据，然后把数据封装成我们想要的对象

用来实现对象模型和关系数据库的转换

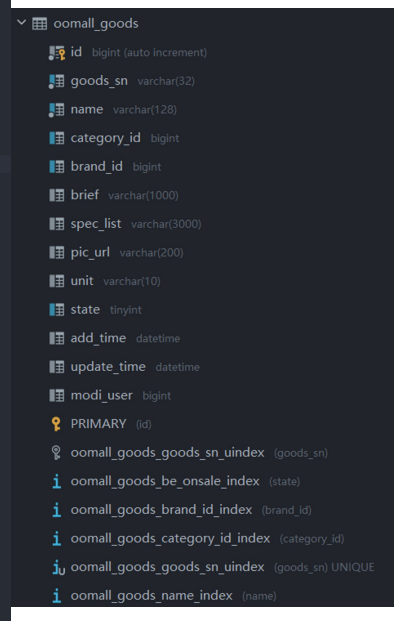
Model包

model包内包含了三类对象

1. VO对象（值对象，value object）
   VO对象是用在Controller层的，因为Controller层会对前端过来的数据进行合法性检查，像GoodsVo、ProductVo这些VO的类中是有合法性检查的
   这些VO是用来从前端接收数据的，是完全按照前端传过来的数据的格式来定义的，与对象模型、数据库存储没有关系
   a. RetVO对象（返回VO对象，return value object）
      这一类VO对象是用来封装返回给前端的数据的，如GoodsRetVo
      在这种VO对象的类中是没有合法性检查的
2. 业务对象（有时称之为BO，是没有PO、VO等后缀的类，如Goods、Product等）
   这一类对象**是我们构成对象模型的重要组成部分**，对象模型其实就是由业务对象组建出来的
   我们需要把对象模型组装出来，在上面去实现我们的业务逻辑
3. PO对象（持久对象，persistent object）
   PO对象是用来对数据库进行访问的，PO对象的类基本是是与数据库的表一一对应的
   基本上每一张数据库中的表，都会有一个对应的PO，例：

   PO对象并不能体现出面向对象的模型，DAO层的作用就是从PO对象中去构建BO对象

```java
@Data
@Alias("GoodsPo")
public class GoodsPo {

    private Integer id;

    private String goodsSn;

    private String name;

    private Integer categoryId;

    private Integer brandId;

    private String brief;

    private String picUrl;

    private String unit;

    private Integer state;

    private String specList;

    private LocalDateTime addTime;

    private LocalDateTime updateTime;

    private Integer modiUser;

    private List<ProductPo> productList;
}
```



util包

util包内是一些通用的工具，包中代码很多为静态方法实现

VO、PO、BO的关系

1. VO有两类，一类用来传对象的，一种是用来返回值的
   用来传对象的VO中都会有一个方法，**可以让他去创建对应的BO对象**
   为了节省内存空间，我们在这个例子中把PO对象放在了BO对象中

2. BO对象并不是真正存在的对象，其实是把PO对象用BO对象的方式展现出来

这个我们称之为**代理设计模式**/装饰器设计模式，它其实是个PO，但展现出来是BO
所以我们会把PO中一些不需要看的属性，以及在PO中没有的属性，在BO上实现

3. 对于每一个BO，是可以产生一个RetVO对象，用于返回给前端
   所以BO中都实现了VoObject的接口，用于产生RetVO对象