代码粒度思考

作者：zwxbest
原文链接：https://ld246.com/article/1545461877416
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

为什么要划分粒度?

1. 减少用户等待时间，节省性能。
2. 提高安全性，减少不必要的信息暴露
3. 提高复用性，降低开发和维护成本
4. 利于单元测试

粗粒度和细粒度

信息数量超过业务需要的信息数量，为粗粒度

信息数量少于业务需要信息数量，为细粒度

划分太粗

1. 性能差
2. 安全性差

划分太细

1. 开发和维护性成本高
2. 单元测试成本高

粒度取舍

粗粒度和细粒度并不矛盾，两者可以同时存在。

细粒度可以提高复用性，粗粒度可以减少重复计算以及方便业务扩展。

粒度分类

Mapper的粒度

如果是按照主键查询某个商品，
针对频繁使用的字段和全部字段分别写两个Mapper。
写一个
只有id和商品名的Mapper和
全部信息都包括的Mapper

频繁的界限是按照业务以及经验来的。

服务的粒度

服务的方法粒度是根据业务逻辑来划分的。

原文链接：代码粒度思考

比如订单微服务中，订单服务，退款服务，对账服务，分账服务等等。

服务方法的粒度

假设我需要一个查询以及增加订单的功能，同时希望能在订单列表展示订单金额，在订单详情展示支方式。

提供细粒度的服务方法，对应Mapper的接口，因为Mapper接口不能写实现，所以此方法中提供业无关的内容，比如批量插入时的po集合，如果为空就打个log然后return，以及参数转po。

2. 提供粗粒度的方法，查询订单所有信息的方法。

3. 提供给Controller正常粒度的服务方法，根据需求，提供两个方法。1.从订单信息中获取订单价格信息。 2.从订单信息中获取订单支付方式以及其他信息，并组装其他服务的信息，比如商品名，支付户信息。

其他无关此无服务的方法，可以放到Util或者Converter中，比如调用其他微服务并处理内部调用异常，以及po装vo，vo转po，之类的。

参数的粒度

参数设计应该遵循迪米特法则。

在次引用一下

现在很多接口都是这样动不动就是一个DO或者DTO对象作为参数，但是到了实现一看，发现其实就用到DO/DTO的两三个字段而已。这种大对象作为参数实际上是一种非常不好的作法，特别是如果你用到了大数据对象的一小部分字段而已。举个例子大家就比较容易理解我为什么对它如此深恶痛绝了在com.alibaba.intl.biz.product.service.interfaces.ProductService获取产品独立detail页面URL的口定义如下：

com.alibaba.intl.biz.product.service.interfaces.ProductService
String getProductDetailUrl(URIBroker uriBroker, ProductSearchDTO product);

而在com.alibaba.intl.biz.product.service.impl.ProductServiceImpl中其实现只是用到了ProductSerchDTO的三个字段：getProductId()，getSubject()，getServiceType()。但是ProductSearchDT这个类呢有十几个属性，并且还关联了一些DO对象。你说这样的一个接口给用户，它怎么知道应该充这个DTO的那些字段呢？！这对内存空间也是一种浪费。 直接定义成这样的接口多简单： com.alibba.intl.biz.product.service.interfaces.ProductService

String getProductDetailUrl(URIBroker uriBroker, Integer productId, String subject, ServiceType serviceType);

返回值的粒度

尽量用返回值而不是修改接口的引用参数，同时接口引用参数大部分情况下也不推荐。

内部接口的粒度

内部接口应尽量和api接口隔离开，尽量调用上面业务无关的方法，因为内部接口大部分情况下只需根据外键来查询某个名字或者其他信息。

api接口的粒度

每个api接口都应该有各自的Request和Response类，Request和Respone类尽量不要复用，也不要虑组合或者继承关系，一旦组合或者继承加了个限制，比如金额限制，时间限制，拿着就隐形的bug。

单元测试粒度

controller层只测试参数限制以及返回的内容格式等

service只测试业务逻辑

mapper只测试sql正确性

无关的bean一律用mock傻对象注入，参考：https://aoyouzi.iteye.com/blog/2343725