服务化最佳实践

**分包**

* 建议将服务接口，服务模型，服务异常等均放在API包中，因为服务模型及异常也是API的一部分，  
  同时，这样做也符合分包原则：重用发布等价原则(REP)，共同重用原则(CRP)
* 如果需要，也可以考虑在API包中放置一份spring的引用配置，这样使用方，只需在Spring加载过程中引用此配置即可，  
  配置建议放在模块的包目录下，以免冲突，如：com/alibaba/china/xxx/dubbo-reference.xml

**粒度**

* 服务接口尽可能大粒度，每个服务方法应代表一个功能，而不是某功能的一个步骤，否则将面临分布式事务问题，Dubbo暂未提供分布式事务支持。
* 服务接口建议以业务场景为单位划分，并对相近业务做抽象，防止接口数量爆炸
* 不建议使用过于抽象的通用接口，如：Map query(Map)，这样的接口没有明确语义，会给后期维护带来不便。

**版本**

* 每个接口都应定义版本号，为后续不兼容升级提供可能，如：<dubbo:service interface="com.xxx.XxxService" version="1.0" />
* 建议使用两位版本号，因为第三位版本号通常表示兼容升级，只有不兼容时才需要变更服务版本。
* 当不兼容时，先升级一半提供者为新版本，再将消费者全部升为新版本，然后将剩下的一半提供者升为新版本。

**兼容性**

* 服务接口增加方法，或服务模型增加字段，可向后兼容，删除方法或删除字段，将不兼容，枚举类型新增字段也不兼容，需通过变更版本号升级。

**枚举值**

* 如果是完备集，可以用Enum，比如：ENABLE, DISABLE。
* 如果是业务种类，以后明显会有类型增加，不建议用Enum，可以用String代替。
* 如果是在返回值中用了Enum，并新增了Enum值，建议先升级服务消费方，这样服务提供方不会返回新值。
* 如果是在传入参数中用了Enum，并新增了Enum值，建议先升级服务提供方，这样服务消费方不会传入新值。

**序列化**

* 服务参数及返回值建议使用POJO对象，即通过set,get方法表示属性的对象。
* 服务参数及返回值不建议使用接口，因为数据模型抽象的意义不大，并且序列化需要接口实现类的元信息，并不能起到隐藏实现的意图。
* 服务参数及返回值都必需是byValue的，而不能是byRef的，消费方和提供方的参数或返回值引用并不是同一个，只是值相同，Dubbo不支持引用远程对象。

**异常**

* 建议使用异常汇报错误，而不是返回错误码，异常信息能携带更多信息，以及语义更友好，
* 如果担心性能问题，在必要时，可以通过override掉异常类的fillInStackTrace()方法为空方法，使其不拷贝栈信息，
* 查询方法不建议抛出checked异常，否则调用方在查询时将过多的try...catch，并且不能进行有效处理，
* 服务提供方不应将DAO或SQL等异常抛给消费方，应在服务实现中对消费方不关心的异常进行包装，否则可能出现消费方无法反序列化相应异常。

**调用**

* 不要只是因为是Dubbo调用，而把调用Try-Catch起来。Try-Catch应该加上合适的回滚边界上。
* 对于输入参数的校验逻辑在Provider端要有。如有性能上的考虑，服务实现者可以考虑在API包上加上服务Stub类来完成检验。

**推荐用法**

**在Provider上尽量多配置Consumer端属性**

原因如下：

1. 作服务的提供者，比服务使用方更清楚服务性能参数，如调用的超时时间，合理的重试次数，等等
2. 在Provider配置后，Consumer不配置则会使用Provider的配置值，即Provider配置可以作为Consumer的缺省值。  
   否则，Consumer会使用Consumer端的全局设置，这对于Provider**不可控的**，并且往往是**不合理的**。

PS: 配置的覆盖规则：1) 方法级配置别优于接口级别，即小Scope优先 2) Consumer端配置 优于 Provider配置 优于 全局配置，最后是Dubbo Hard Code的配置值（见配置文档）

Provider上尽量多配置Consumer端的属性，让Provider实现者一开始就思考Provider服务特点、服务质量的问题。

示例：



|  |
| --- |
|  |

在Provider可以配置的Consumer端属性有：

1. **timeout**，方法调用超时
2. **retries**，失败重试次数，缺省是2（表示加上第一次调用，会调用3次）
3. **loadbalance**，负载均衡算法（有多个Provider时，如何挑选Provider调用），缺省是随机（random）。  
   还可以有轮训(roundrobin)、最不活跃优先（leastactive，指从Consumer端并发调用最好的Provider，可以减少的反应慢的Provider的调用，因为反应更容易累积并发的调用）
4. **actives**，消费者端，最大并发调用限制，即当Consumer对一个服务的并发调用到上限后，新调用会Wait直到超时。  
   在方法上配置（dubbo:method ）则并发限制针对方法，在接口上配置（dubbo:service），则并发限制针对服务。

详细配置说明参见：[Dubbo配置参考手册](http://code.alibabatech.com/wiki/display/dubbo/User+Guide-zh#UserGuide-zh-ConfigurationReference)

**Provider上配置合理的Provider端属性**

|  |
| --- |
|  |

Provider上可以配置的Provider端属性有：

1. **threads**，服务线程池大小
2. **executes**，一个服务提供者并行执行请求上限，即当Provider对一个服务的并发调用到上限后，新调用会Wait（Consumer可能到超时）。在方法上配置（dubbo:method ）则并发限制针对方法，在接口上配置（dubbo:service），则并发限制针对服务。