关于怎么实现承载更多用户量的系统，一直是我重点关注的一个技术方向。改造架构提高承载能力，通常来将分为两个大方向，互相配合实现。

**硬件架构改进**，主要是使用阿里云这种多组件的云环境；通过负载均衡SLB，模板克隆的云服务器ECS，云数据库RDS，共享对象存储OSS等不同职责的云产品组合实现。

**软件架构优化**，主要是软件代码开发的规范；业务解耦合，架构微服务，单击无状态化，文件存储共享等。

今天要说的就是软件开发时候对于接口服务的“幂等性”实现！

#### 幂等性

效果：系统对某接口的多次请求，都应该返回同样的结果！（网络访问失败的场景除外）

目的：避免因为各种原因，重复请求导致的业务重复处理。

重复请求场景案例：

1. 客户端第一次请求后，网络异常导致收到请求执行逻辑但是没有返回给客户端，客户端的重新发起请求。
2. 客户端迅速点击按钮提交，导致同一逻辑被多次发送到服务器。

简单来划分，业务逻辑无非都可以归纳为增删改查！

对于查询，内部不包含其他操作，属于只读性质的那种业务必然符合幂等性要求的。

对于删除，重复做测试请求至少不会造成数据杂乱，不过也有些场景更希望重复点击提示的是删除成功，而不是目标不存在的提示。

对于新增和修改，这里是重点关注的部分：新增，需要避免重复插入；修改，避免进行无效的重复修改；

#### 幂等性的实现方式

实现方法：客户单做某一请求的时候带上识别参数标识，服务端对此标识进行识别，重复请求则重复返回第一次的结果即可。

举个例子：比如添加请求的表单里，在打开添加表单页面的时候，就生成一个AddId标识，这个AddId跟着表单一起提交到后台接口。

后台接口根据这个AddId，服务端可以进行缓存标记并进行过滤，缓存值可以是AddId作为缓存key，返回内容作为缓存Value，这样即使添加按钮被多次点下也可以识别出来。

这个AddId什么时候更新呢？只有在保存成功并且清空表单之后，才变更这个AddId标识,从而实现新数据的表单提交。