链滴

protobuf v3 中的字段默认值和空字段区分？

作者：iTrace

原文链接：https://ld246.com/article/1599317161812

来源网站：链滴

简介

protobuf是google开源的序列化工具，在微服务架构中是常见的dto构建工具。

protobuf v3

在protobuf3中，字段的的默认规则都是optional的，正是因为如此，我们在marshal的时候字段是选的，但是在unmarshal的时候，所有的字段都是会被设置值的，如在marshal未设置字段在unmarshal的时候会被默认值填充，这时我们是没有办法区分字段是nullState还是defaulteValue。

如下：

```
syntax="proto3"

package dto
message Request {
    string name = 1;
    int32 age = 2;
    string sex = 3;
}

message Response {

}

service Transport {
    rpc Send(Request) returns(Response)
}

import "dto"
```

原文链接：protobuf v3 中的字段默认值和空字段区分?

```
func TestNullState() {
    req := &dto.Request {
        Name: "joe",
        Age: 23,
    }
    //假设拿到了grpc的client
    resp, err := client.Send(req)
}

package service

import "dto"

type TransportService{}

func (transport *TransportService) Send(ctx context.Context, req *dto.Request) (resp *dto.Res
onse, err error) {
    //在这里我们拿到了dto.Requset,
    //那么，如何判断req.Sex是未设置，还是设置了""呢?
    //protobuf是不提共判断的，从v3开始。
}
```

判断nullState和defaultValue

方案1：使用特殊值判断

使用特殊值替代nullState，如年龄字段，age显然不能为负，所以负数都可以替代nullState，这是表层的（represent）的nullState。 但是如果是description，这样的字段，如何表示未设置状态呢?

其次就是特殊值的耦合很深，编码不灵活。

方案二：显式定义 boolean 字段(不建议)

显式定义bool字段，那么message的字段数加倍，并且排版难看。

```
message Request {
    string name = 1;
    in32 age = 2;
    string sex = 3;
    bool has_name = 4;
    bool has_age = 5
    bool has_sex = 6;
}
```

方案三：使用 oneof 黑科技

```
message SampleMessage {
  oneof test_oneof {
```

```
    string name = 4;
    SubMessage sub_message = 9;
  }
}
```

oneof关键字和c语言中的union是一个意思，就是oneof中的所有字段只能同时set一个，set一个前个将被抹去。

方案四：使用wrapper类型

使用wrapper类型来传递，是一个很好的方法，很nice。

```go
message String {//wrapper string.
    string value = 1;
    bool flag = 2;
}
```

这样虽然和显式增加判断字段相似，但是好看不是吗？且可维护性和扩展性更好。

方案五：使用bitset来判断字段是否set

添加一个bitset类型字段来实现存储字段是否存储。至于bitset如何实现，这不重要（使用一个int64本就已经够用了）。

```
message Request {
    int64 fields_state = 1;
    string name = 2;
    int32 age = 3;
    string sex = 4;
}

//通过位移运算来判断字段是否被设置。
```

以字段索引为索引，通过位移运算来判断字段是否被设置。

方案六：使用json传递数据，在protobuf中开洞

在protobuf里面打个洞，传递json数据，也就是string。

```
message Request {
    string content = 1;
}
```

在grpc后端，我们再对content字段进行json.Unmarshal来解析请求。

这样做是没有任何问题的，但是效率有点低。首先grpc是基于htpp2的，所以就需要经过http编解码protobuf编解码，就是两次，再加上json的话就是3次，此时效率会降低。

Reference

[1] https://www.cnblogs.com/tohxyblog/p/8974763.html

[2] https://zhuanlan.zhihu.com/p/46603988

[3] https://stackoverflow.com/questions/42622015/how-to-define-an-optional-field-in-protobuf-3