从所有教程的词条中查询

首页 › 慕课教程 › Go工程师体系课全新版 › 2. go控制grpc的metadata

bobby · 更新于 2022-11-08

上一节 1. protobuf官方... 3. grpc拦截器 -... 下一节

gRPC让我们可以像本地调用一样实现远程调用，对于每一次的RPC调用中，都可能会有一些有用的数据，而这些数据就可以通过metadata来传递。metadata是以key-value的形式存储数据的，其中key是 `string` 类型，而value是 `[]string`，即一个字符串数组类型。metadata使得client和server能够为对方提供关于本次调用的一些信息，就像一次http请求的RequestHeader和ResponseHeader一样。http中header的生命周期是一次http请求，那么metadata的生命周期就是一次RPC调用。

1. go中使用metadata

源码：源码
项目文档：文档

1. 新建metadata

MD 类型实际上是map，key是string，value是string类型的slice。

代码块

```
type MD map[string][]string
```

创建的时候可以像创建普通的map类型一样使用new关键字进行创建：

代码块

```
//第一种方式
md := metadata.New(map[string]string{"key1": "val1", "key2": "val2"})
//第二种方式 key不区分大小写，会被统一转成小写。
md := metadata.Pairs(
    "key1", "val1",
    "key1", "val1-2", // "key1" will have map value []string{"val1", "val1-2"}
    "key2", "val2",
)
```

2. 发送metadata

代码块

```
md := metadata.Pairs("key", "val")

// 新建一个有 metadata 的 context
ctx := metadata.NewOutgoingContext(context.Background(), md)

// 单向 RPC
response, err := client.SomeRPC(ctx, someRequest)
```


3. 接收metadata

代码块

```
func (s *server) SomeRPC(ctx context.Context, in *pb.SomeRequest) (*pb.SomeRespor
    md, ok := metadata.FromIncomingContext(ctx)
    // do something with metadata
}
```

索引目录

2. grpc中使用metadata

1. proto

代码块

```
syntax = "proto3";
option go_package=".;proto";

// The greeting service definition.
service Greeter {
    //   Sends a greeting
    rpc SayHello (HelloRequest) returns (HelloReply) {
    }
}

// The request message containing the user's name.
message HelloRequest {
    string name = 1;
}

// The response message containing the greetings
message HelloReply {
    string message = 1;
}
```

2. client

代码块

```
package main

import (
    "OldPackageTest/grpc_test/proto"
    "context"
    "fmt"
    "google.golang.org/grpc"
    "google.golang.org/grpc/metadata"
)

func main(){
    //stream
    conn, err := grpc.Dial("127.0.0.1:50051", grpc.WithInsecure())
    if err != nil {
        panic(err)
    }
    defer conn.Close()

    //md := metadata.Pairs("timestamp", time.Now().Format(timestampFormat))
    md := metadata.New(map[string]string{
        "name":"bobby",
        "pasword":"imooc",
    })
```

```
26      ctx := metadata.NewOutgoingContext(context.Background(), md)
27      r, err := c.SayHello(ctx, &proto.HelloRequest{Name:"bobby"})
28      if err != nil {
29          panic(err)
30      }
31      fmt.Println(r.Message)
32  }
33
```

3. server

代码块 预览 复制

```
1   package main
2
3   import (
4       "context"
5       "fmt"
6       "google.golang.org/grpc/metadata"
7       "net"
8
9       "google.golang.org/grpc"
10
11      "OldPackageTest/grpc_test/proto"
12  )
13
14  type Server struct{}
15
16  func (s *Server) SayHello(ctx context.Context, request *proto.HelloRequest) (*pro
17      error) {
18
19      md, ok := metadata.FromIncomingContext(ctx)
20      if ok {
21          fmt.Println("get metadata error")
22      }
23      if nameSlice, ok := md["name"]; ok {
24          fmt.Println(nameSlice)
25          for i, e := range nameSlice {
26              fmt.Println(i, e)
27          }
28      }
29      return &proto.HelloReply{
30          Message: "hello " + request.Name,
31      }, nil
32  }
33
34  func main() {
35      g := grpc.NewServer()
36      proto.RegisterGreeterServer(g, &Server{})
37      lis, err := net.Listen("tcp", "0.0.0.0:50051")
38      if err != nil {
39          panic("failed to listen:" + err.Error())
40      }
41      err = g.Serve(lis)
42      if err != nil {
43          panic("failed to start grpc:" + err.Error())
44      }
```