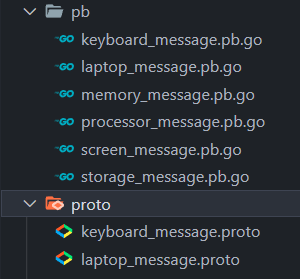
proto

1. 学习gRPC需要先学会protocol buffer来定义消息和服务。.proto文件适用于十几种开发语言并且它支持你使用同一种框架来支持每秒百万级以上的RPC调用

.proto文件包括gRPC服务的定义和服务端和客户端之间传递的消息

1. .proto文件生成.pb.go文件的命令

例：



PS C:\Users\黄保定\Desktop\grpc格式化> protoc --proto\_path=./proto ./proto/\*.proto --go\_out=plugins=grpc:./

注释：protoc 编译的路径是当前文件夹下的proto文件 编译的文件是当前文件夹下的proto文件夹下的所有.proto文件 使用plugins=grpc插件进行输出，输出的路径是当前文件夹下自动生成的pb文件夹

1. 开头

syntax = "proto3";

option go\_package="/包名"; //包名也是文件夹名，会在同名文件夹下生产.proto文件

1. 命名规范
   1. message名字：UpperNameCase
   2. 字段名字：lower\_snake\_case (转为Go代码后开头是大写)
2. 定义message

message Screen {

    message Resolution { //嵌套message只能在此Screen message内使用

        uint32 width = 1; //不能为负数

        uint32 height = 2;

    }

    enum Panel{                                 //枚举

        UNKNOWN = 0;

        IPS = 1;

        OLED = 2;

    }

    float size\_inch = 1;                        //屏幕的尺寸。

    Resolution resolution = 2;

    Panel panel = 3;

bool multitouch = 4;                        //是否是多点触控屏幕

}



在Protocol Buffers里，字段的tag比名字更重要

1<tag<2^29-1（536870911），但是不可以使用19000-19999

1-15的tag数只占用1个字节的空间，所以说它们应该被用在更频繁使用的字段上。16-2047则占用2个字节，它们可以用在不那么频繁使用的字段上

1. 字段的默认值

字符和数组为空，bool为false，数值为0。我们需注意区分字段是实际值还是恰好等于默认值

1. 字段规则
   1. 单数字段（Singular）：只能出现0次或一次
   2. 重复字段（Repeated）：一般用于list或数组等
2. 枚举、

枚举的第一个值必须从0开始

1. 删除字段后需标记为reserved，防止之后的消息使用老版本protocol时出错
2. proto文件的变更
   1. 变更规则
      1. 向前兼容变更：使用新的.proto文件来写数据，从旧的.proto文件读数据
      2. 向后兼容变更：使用旧的.proto文件来写数据，从新的.proto文件读数据
3. .proto和.go文件的转化

message Memory{ //type Memory struct { ... }

enum Unit{          //type Memory\_Unit int32

        UNKNOWN = 0; //const Memory\_UNKNOWN Memory\_Unit = 0

        BIT = 1; //const Memory\_BIT Memory\_Unit = 1

        BYTE = 2;

        KYLOBYTE = 3;

        MEGABYTE = 4;

        GIGABYTE = 5;

        TERABYTE = 6;

    }

     uint64 value = 1; //Value uint64 (结构体Memory里的字段）

     Unit uint = 2; //Uint Memory\_Unit

}

message GPU { //type GPU struct{...}

    string brand = 1;

    string name = 2;

    double min\_ghz = 3;

    double max\_ghz = 4;

    Memory memory = 5; //Memory \*Memory (GPU.Memory=&Memory)

}

message Screen {

    message Resolution { //type Screen\_Resolution struct{...}(.go文件中在Screen外定义）

        uint32 width = 1;

        uint32 height = 2;

    }

    enum Panel{

        UNKNOWN = 0;

        IPS = 1;

        OLED = 2;

    }

    float size\_inch = 1; //SizeInch float32

    Resolution resolution = 2;//Resolution \*Screen\_Resolution

    Panel panel = 3;

    bool multitouch = 4;

}

message Laptop {

    string id = 1;

    string brand = 2;

    string name = 3;

    CPU cpu = 4;

    Memory ram = 5;

    repeated GPU gpus = 6; //[]\*GPU

    repeated Storage storages = 7;//[]\*Storage

    Screen screen = 8;

    Keyboard keyboard = 9;

    oneof weight {                                  //只有最后一个设置字段的值有效。

        double weight\_kg = 10;

        double weight\_lb = 11;

    }

    double price\_usd = 12;                          //电脑价格

    uint32 release\_year = 13;                       //发布年份

    google.protobuf.Timestamp update\_at = 14;       //记录存储系统中最后更新的时间。

}