6 - Golang Programming with Protocol Buffers

Code generation in Golang

/src/PROTOCOL-EXAMPLE-GO/generate.sh

```
protoc -I src/ --go_out=src src/simple/simple.proto
```

Simple Proto Struct in Golang

/src/PROTOCOL-EXAMPLE-GO/main.go

```
package main

import (
    "fmt"
    "strconv"

    simplepb "github.com/PROTOCOL-EXAMPLE-GO/src/simple"
)

func main() {
    doSimple()
}

func doSimple() {
    sm := simplepb.SimpleMessage{
        Id:         12345,
        IsSimple:   true,
        Name:       "My Simple Message",
        SampleList: []int32{1, 4, 7, 8},
    }

    sm.Name = "I renamed you"

    fmt.Println(sm)
    fmt.Println("The ID is:" + strconv.FormatInt(int64(sm.GetId()), 10))
}
```

Reading and Writing to Disk

/src/PROTOCOL-EXAMPLE-GO/main.go

```
package main
```

```
2
3  import (
4      "fmt"
5      "io/ioutil"
6      "log"
7      "strconv"
8
9      simplepb "github.com/PROTOCOL-EXAMPLE-GO/src/simple"
10     "github.com/golang/protobuf/proto"
11 )
12
13 func main() {
14     sm := doSimple()
15     readAndWriteDemo(sm)
16 }
17
18 func readAndWriteDemo(pb proto.Message) {
19     // writeToFile()
20     writeToFile("simple.bin", pb)
21     sm := &simplepb.SimpleMessage{}
22     // readFromFile()
23     readFromFile("simple.bin", sm)
24     fmt.Println(sm)
25 }
26
27 func doSimple() *simplepb.SimpleMessage {
28     sm := simplepb.SimpleMessage{
29         Id:         12345,
30         IsSimple:   true,
31         Name:       "My Simple Message",
32         SampleList: []int32{1, 4, 7, 8},
33     }
34
35     sm.Name = "I renamed you"
36
37     fmt.Println(sm)
38     fmt.Println("The ID is:" + strconv.FormatInt(int64(sm.GetId()), 10))
39
40     return &sm
41 }
42
43 func writeToFile(fname string, pb proto.Message) error {
44     out, err := proto.Marshal(pb)
45     if err != nil {
46         log.Fatalln("Can't serialize to bytes", err)
47         return err
48     }
49     if err := ioutil.WriteFile(fname, out, 0644); err != nil {
50         log.Fatalln("Can't write to file", err)
51         return err
52     }
53     fmt.Println("Data has been written!")
54     return nil
55 }
56
57 func readFromFile(fname string, pb proto.Message) error {
```

```
58      in, err := ioutil.ReadFile(fname)
59      if err != nil {
60          log.Fatalln("Something went wrong when reading the file", err)
61          return err
62      }
63
64      err2 := proto.Unmarshal(in, pb)
65      if err != nil {
66          log.Fatalln("Couldn't put the bytes into the protocol buffers struct", err)
67          return err2
68      }
69
70      return nil
71  }
```

Reading and Writing to JSON