Get these slides ⇨

Introduction to the protobuf ecosystem

FOR: IBANFIRST R&D PAYMENT TEAM

Théophile Roos

orness.com

ORNESS

Contents

1 What is protobuf ?

Short for Protocol Buffers: an efficient, language-neutral data serialization format developed by Google

- Much smaller and faster than formats like JSON or XML
- Strong Typing and Schema: strict structure of the data defined in `.proto` files, enabling forward and backward compatibility
- Used for many use-cases: synchronous communication between network services, async communication through brokers systems (Kafka, RabbitMQ...), WebAssembly interfaces...

1.1 Syntax

Protobuf syntax versions

- `proto2`: legacy should not be used
- `proto3`: most used version currently
- `revision <year>`: new version with very limited support

First line of the `.proto` file

```proto
syntax = "proto3";
// syntax = "proto2";
// edition = "2023";
```

1.1.1 Message definition

Message definition

```proto
message User {
  string name = 1 [json_name = "name"];
  optional int32 age = 2 [default = 18];

  // a comment
  repeated string hobbies = 3;

  map<string, int32> ibanToBalance = 4;
}
```

Fields are defined with:

- a FieldType: int32, string, bool, bytes, etc.
- a FieldName: name, age, etc.
- a Field Cardinality: optional, required, repeated, etc.

notes:

- *maps can't be repeated*

1.1.2 Enum definition

```proto
enum USER_TYPE {
  // a comment
  USER_TYPE_UNSPECIFIED = 0;
  USER_TYPE_REGULAR = 1;
  USER_TYPE_GUEST = 2;
  USER_TYPE_ADMIN = 3;
}

message User {
  // ...
  USER_TYPE user_type = 3;
}
```

The enum has a default value, which is the first value defined in the enum and with the FieldNumber `0`.

1.1.3 Enum definition (Aliases)

```proto
enum Corpus {
  option allow_alias = true;

  CORPUS_UNSPECIFIED = 0;
  CORPUS_UNIVERSAL = 1;
  CORPUS_WEB = 2;
  CORPUS_WWW = 2;
}
```

Message and enums can have options defined with the option keyword.

1.1.4 Message Composition

```proto
// user.proto
import "user_type.proto";

message User {
  // ...
  USER_TYPE user_type = 3;
}

message Organisation {
  string name = 1;
  repeated User users = 2;
}
```

```proto
// user_type.proto
enum USER_TYPE {
  // a comment
  USER_TYPE_UNSPECIFIED = 0;
  USER_TYPE_REGULAR = 1;
  // ...
}
```

An *Organisation* contains a list of *User* wich contain a field of type *USER_TYPE*

1.1.5 Nested Types

```proto
message User {
  enum USER_TYPE {
    USER_TYPE_UNSPECIFIED = 0;
    USER_TYPE_REGULAR = 1;
    // ...
  }
  // ...
  USER_TYPE user_type = 3;
}
```

Using nested type outside of the parent message definition

```proto
message UserProfile {
  User.USER_TYPE type = 1;
}
```

1.1.6 oneof

```proto
enum USER_TYPE {
  // ...
}
enum ADMIN_TYPE {
  // ...
}

message User {
  // ...
  oneof type {
    USER_TYPE user_type = 3;
    ADMIN_TYPE admin_type = 4;
  }
}
```

Using oneof to define a field that can be either a user or an admin.

Only one field of the oneof can be set at a time.

=> avoid using oneof if possible because of restrictions on the backward compatibility.

1.1.7 Packages

file structure:

```
./foo/bar/baz.proto
./fizz/buzz.proto
```

```proto
// foo/bar/baz.proto
package foo.bar;

message Baz {
  // ...
}
```

```proto
// fizz/buzz.proto
package fizz;

import "foo/bar/baz.proto";

message Buzz {
  foo.bar.Baz baz = 1;
}
```

1.1.8 Service definition

```proto
message GetUserRequest {
  string username = 1;
}
message GetUserResponse {
  User user = 1;
}

service UserService {
  rpc GetUser(GetUserRequest) returns (GetUserResponse);
  // rpc DeleteUser(DeleteUserRequest) returns (DeleteUserResponse);
}
```

A service *UserService* is defined with an RPC method *GetUser* that takes a *GetUserRequest* and returns a *GetUserResponse*.

1.1.9 TODO extension and annotations

1.2 Usefull links

- Official documentation: https://protobuf.dev/overview/
- Un-official (but very good) documentation by the Buf team: https://protobuf.com/docs/introduction
- Third party protoc code generation plugins: https://github.com/protocolbuffers/protobuf/blob/main/docs/third_party.md
- Registed protobuf extension: https://github.com/protocolbuffers/protobuf/blob/main/docs/options.md#existing-registered-extensions
- somewhat standard google protobuf definitions: https://github.com/googleapis/googleapis

2 RPCs

Remote Procedure Calls: a way to call a function on a remote server

RPC based communication is pattern with many different implementations [1]:

- NFS: Network File System
- JSON-RPC: an RPC protocol that uses JSON-encoded messages.
- gRPC is a modern open source RPC framework that uses HTTP/2 and Protocol Buffers.
- IBM AIX: use of RPC in the AIX operating system. [2]

2.1 Protobuf based RP

some commonly used RPC framework

Framework	Transport Layer	Code Generation	Notable Features
gRPC	HTTP/2	Yes	Streaming, TLS, strong ecosystem
ConnectRPC	HTTP	Yes	Streaming, TLS, wide platform support
Ranger RPC	HTTP (any version)	Yes	Simple, fast, Go-centric
Apache Thrift	Any	Yes	Simple, fast, C++-centric
trpc	Any	No	Pure TypeScript

Bibliography

[1] Contributors to Wikimedia projects, “Remote procedure call - Wikipedia.” [Online]. Available: https://en.wikipedia.org/w/index.php?title=Remote_procedure_call&oldid=1292581498

[2] “AIX.” [Online]. Available: https://www.ibm.com/docs/en/aix/7.3.0?topic=call-rpc-features

3 That's it!

⇧Get these slides

Get in touch 👋

🌐 https://orness.com