Практические задачи решаем функционально

Сериализуем Protobuf на F#

Марк Шевченко

Программирует с 1989 года. Писал на C, C++, Delphi, Perl, PHP, C#, Ruby, F# и Rust. Организует встречи Московского клуба программистов. Помогает коллегам делать интересные доклады и писать хорошие статьи.

Проблема: тестирование бекенда

REST API

1. Swagger
2. Postman
3. curl

gRPC

1. ???

Проблема: тестирование бекенда

req.json

```
{
  "from":
  {
    "latitude": 55.75124,
    "longitude": 37.61842
  },
  "to":
  {
    "latitude": 59.93863,
    "longitude": 30.31413
  }
}
```

Проблема: тестирование бекенда

```
> curl -X POST -H "Content-Type: application/json" -d @req.json http://svc.geo/distance
{
  "result": 634.6292282187935
}
```

Protobuf

Схема (.proto)

```
message DistanceRequest {
  Point from = 1;
  Point to = 2;
  optional CalculationMethod method = 3;
}

message Point {
  double latitude = 1;
  double longitude = 2;
}

enum CalculationMethod {
  COSINE = 0;
  HAVERSINE = 1;
}
```

Запрос (.textproto)

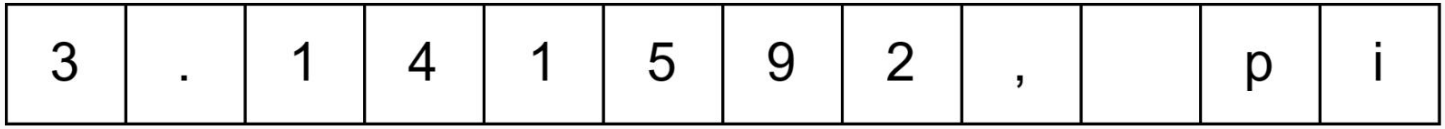
```
from: {
  latitude: 55.75124
  longitude": 37.61842
}
to: {
  latitude: 59.93863
  longitude: 30.31413
}
```

Ответ (.textproto)

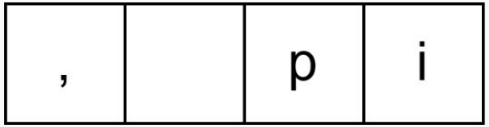
```
result: 634.6292282187935
```

https://github.com/binateq/protos

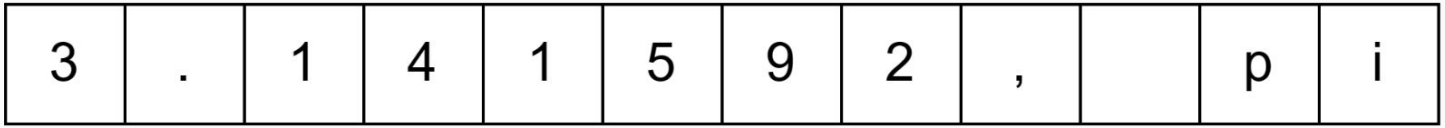
FParsec



pfloat → success: 3.141592



identifier → failure: { line: 1; position; 1, expectations: '_' or letter }



Результат

```
digit → '1'
```

```
many1Chars digit → "100500"
```

```
many1Chars digit |>> int32 → 100500
```

Парсер-комбинаторы

```
let digits: Parser<int32, unit> = many1Chars digit ||> int32
```

```
let pint32: Parser<int32, unit> = (pchar '-' >>. digits ||> (~-)) <|> digits
```

.proto

https://protobuf.dev/reference/protobuf/proto3-spec/

Переводим BNF в F#: идентификатор

```
ident = letter { letter | decimalDigit | "_" }
```

```
let identifier = many1Chars2 asciiLetter (asciiLetter <|> digit <|> pchar '_')
```

Переводим BNF в F#: полный идентификатор

```
fullIdent = ident { "." ident }
```

```
let fullIdentifier = stringsSepBy1 identifier (pstring ".")
```

Пакет

```
package = "package" fullIdent ";"
```

```
let package =
    skipString "package" >>. spaces1 >>.
    fullIdentifier .>> spaces .>>
    skipChar ';' .>> spaces
    |>> Package
```

Abstract Syntax Tree

Пакет

```
type Package = Package of string
```

Импорт

```
import = "import" [ "weak" | "public" ] strLit ";"
```

```
type Import =
    | Import of string
    | WeakImport of string
    | PublicImport of string

let import =
   let path =
       (skipString "weak" >>. spaces1 >>. stringLiteral |>> WeakImport) <|>
       (skipString "public" >>. spaces1 >>. stringLiteral |>> PublicImport) <|>
       (stringLiteral |>> Import)

   skipString "import" >>. spaces1 >>. path .>> spaces
   .>> skipChar ';' .>> spaces
```

Опции и перечисления

```
enum EnumAllowingAlias {
  option allow_alias = true;
  EAA_UNSPECIFIED = 0;
  EAA_STARTED = 1;
  EAA_RUNNING = 2 [(custom_option) = "hello world"];
}
```

Опции и перечисления

```
enum = "enum" enumName enumBody
enumBody = "{" { option | enumField | emptyStatement | reserved } "}"
enumField = ident "=" [ "-" ] intLit
            [ "[" enumValueOption { ","  enumValueOption } "]" ]";"
enumValueOption = optionName "=" constant

option = "option" optionName  "=" constant ";"
optionName = ( ident | "(" ["."] fullIdent ")" )

constant = fullIdent | ( [ "-" | "+" ] intLit ) | ( [ "-" | "+" ] floatLit ) |
           strLit | boolLit | MessageValue
```

.textproto

https://developers.google.com/protocol-buffers/docs/text-format-spec

Просто список полей

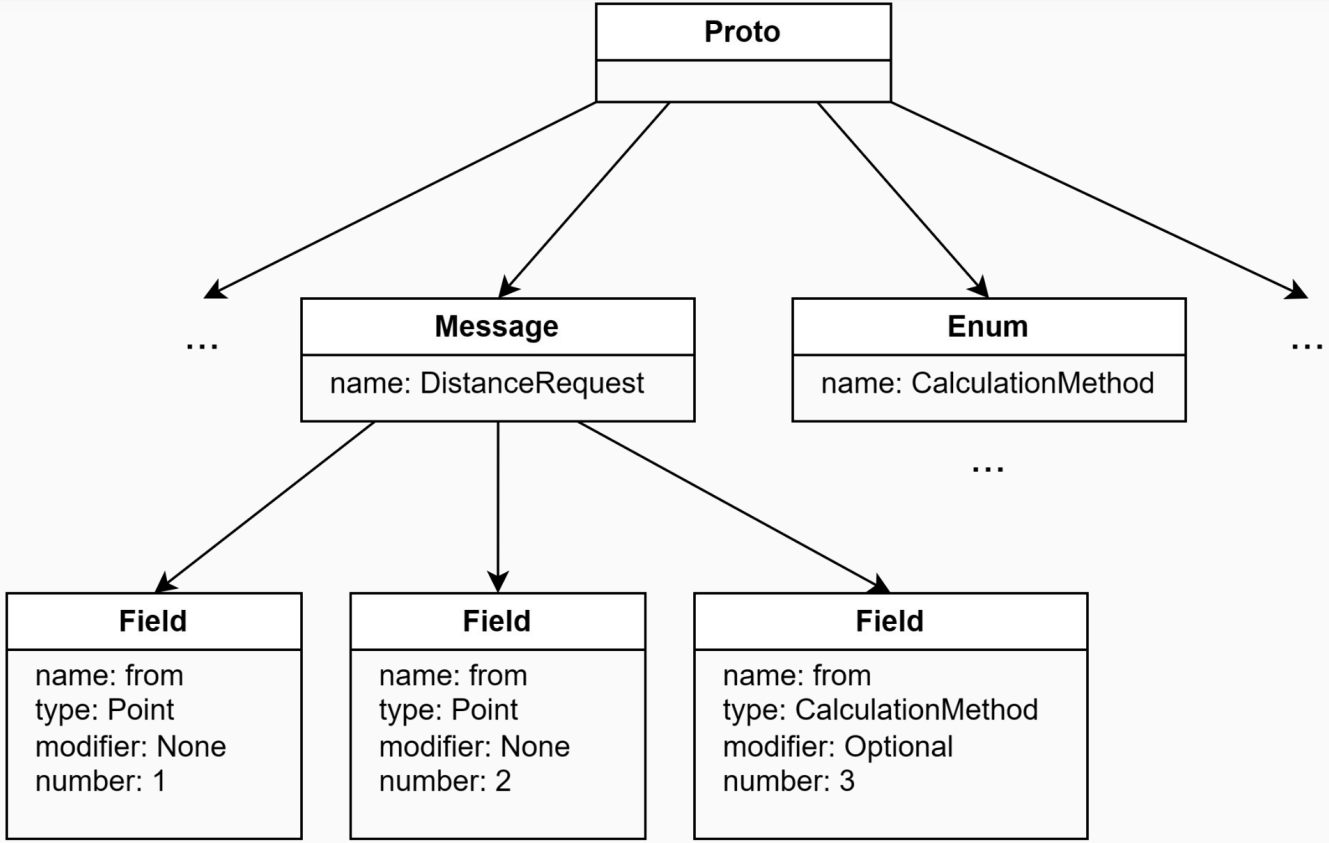
```
from: {
  latitude: 55.75124
  longitude": 37.61842
}

to: {
  latitude: 59.93863
  longitude: 30.31413
}
```

Вопросы по первой части

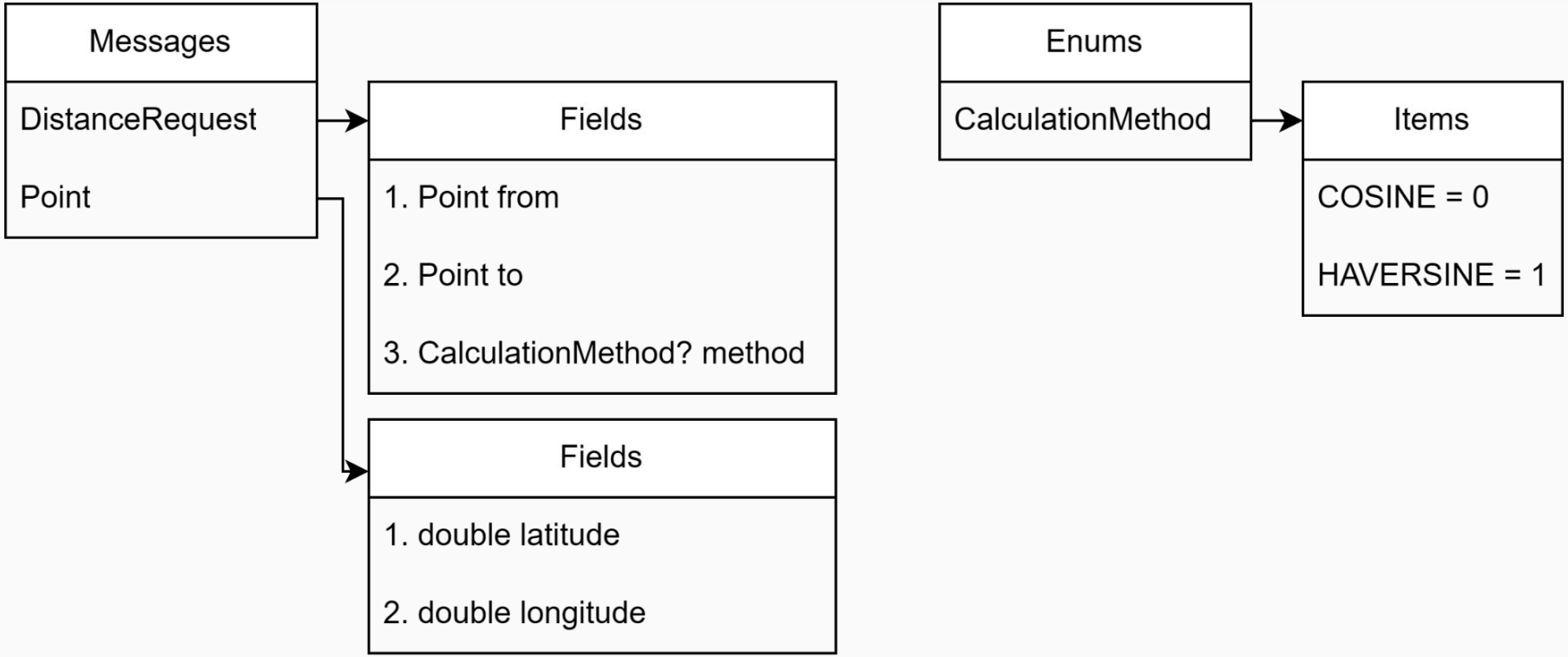
- gRPC
- Схема (.proto)
- Запрос и Ответ (.textproto)
- Парсеры — это функции
- У парсеров есть результат
- Парсер-комбинаторы
- Переводим BNF в F#
- Abstract Syntax Tree
- Алгебраические типы данных
- Тип-сумма discriminated unions
- Single case discriminated unions
- Пакет (package)
- Импорт (import)
- Опции и перечисления (option, enum)
- Просто список полей (.textproto)

Компиляция

AST


Proto
...
Message
name: DistanceRequest
Enum
name: CalculationMethod
...
...
Field
name: from
type: Point
modifier: None
number: 1
Field
name: from
type: Point
modifier: None
number: 2
Field
name: from
type: CalculationMethod
modifier: Optional
number: 3

Схема


Messages
DistanceRequest
Point
Fields
1. Point from
2. Point to
3. CalculationMethod? method
Fields
1. double latitude
2. double longitude
Enums
CalculationMethod
Items
COSINE = 0
HAVERSINE = 1

Сериализация

Бинарные данные

```
> protoc --encode="geo.DistanceRequest" geo.proto↵  <
example.textproto > example.bin
```

```
%HOMEDRIVE%%HOMEPATH%\.nuget\packages\grpc.tools\↵
2.40.0\tools\windows_x64\
```

https://protobuf.dev/programming-guides/encoding/

Бинарные данные

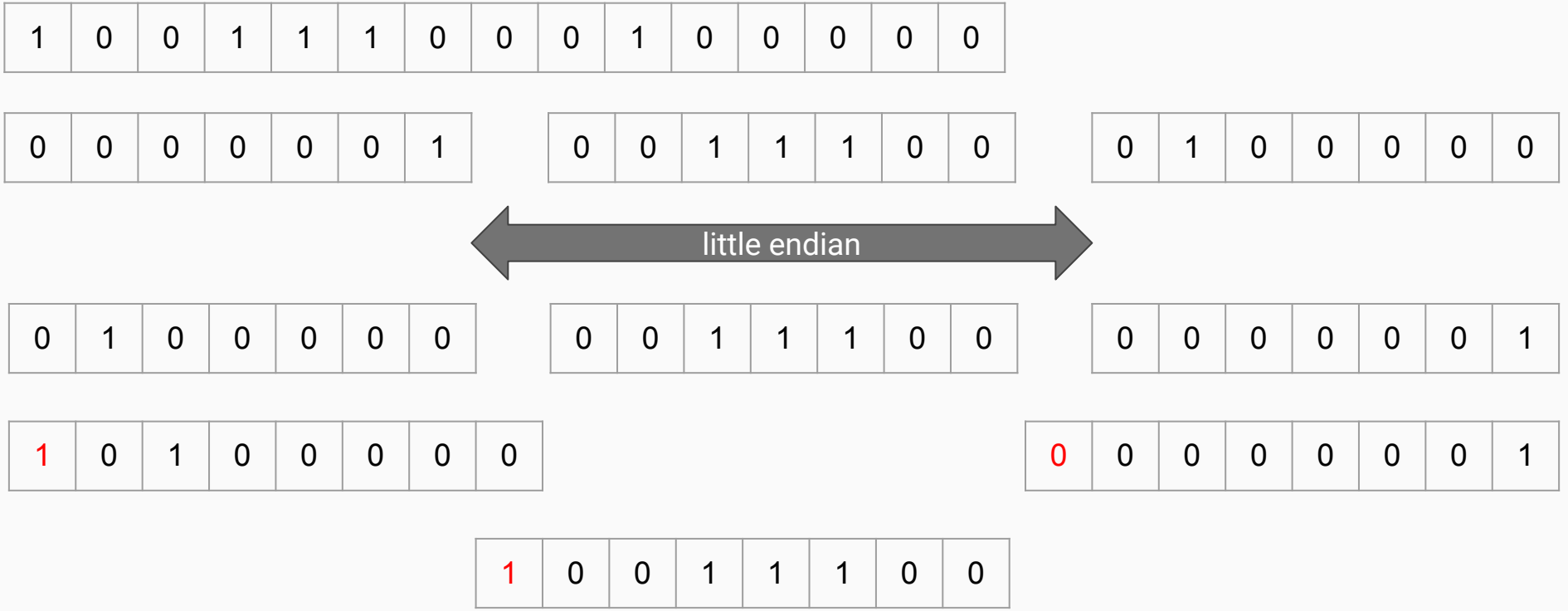
```
0000000000: 0A 12 09 39 B9 DF A1 28 | E0 4B 40 11 9E 98 F5 62
0000000010: 28 CF 42 40 12 12 09 B2 | 85 20 07 25 F8 4D 40 11
0000000020: 46 B1 DC D2 6A 50 3E 40 |
```

Типы сериализации

```
type WireType =
    | Varint = 0u
    | I64 = 1u
    | Len = 2u
    | [<Obsolete>] SGroup = 3u
    | [<Obsolete>] EGroup = 4u
    | I32 = 5u
```

Varint

20000



Varint

```
let serializeVarint n (stream: Stream) =
    let generator (isStop, remainder) =
        if isStop
        then None
        else
            let nextRemainder = remainder / 128uL
            let nextByte = byte (remainder % 128uL)
            if nextRemainder = 0uL
            then Some (nextByte, (true, nextRemainder))
            else Some (0x80uy ||| nextByte, (false, nextRemainder))
    let bytes = Array.unfold generator (false, n)
    stream.Write(bytes)
```

Отрицательные числа

$-2_{10}=11111110111111111111111111111111111111111111111111111111111111111111111100000001_2$

Целое со знаком	Кодируется как
0	0
-1	1
1	2
-2	3
2	4
…	…

Теги

```
tag = fieldNumber <<< 3 + wireType
```

Field Number	Wire Type	Wire Type (int)	Tag
1	Varing	0	8 (0x08)
3	I32	5	29 (0x1D)

Десериализация

Не забываем про тесты!

```
let deserializeVarint (stream: Stream) =
    0uL


let deserializeTag (stream: Stream) =
    (0u, WireType.Varint)


let deserializeBool (stream: Stream) =
    false


let deserializeSInt32 stream =
    0
```

Вопросы по второй части

- Компиляция
- AST
- Схема
- protoc
- Типы сериализации (wire types)
- Varint
- Отрицательные числа
- Теги
- Тесты
- Десериализация