

Opcjonalne pola struktur danych i argumenty wywołania middleware

Wstęp:

Wykorzystane przeze mnie technologie middleware:

- ICE
- gRPC

W obu przypadkach część serwerowa została napisana w języku Java, a część kliencka w języku Python.

Omówię w jaki sposób w interfacach podobnych technologii definiuje się pola opcjonalne, jak wygląda ich implementacja, oraz serializacja danych. Omówię dane zagadnienie na przykładzie interfacu zawierającego przykładowe dane dotyczące dostawy jedzenia, takie jak adres dostawy (nazwa ulicy, numer domu, oraz opcjonalny numer mieszkania), numer telefonu klienta, opcjonalne napiwki, jeżeli zostanie podana tablica napiwków będę w stanie obliczyć z nich średnią.

Definiowanie pól opcjonalnych w interface'ach:

ICE:

```
3
4  module DeliveryService {
5      sequence<double> DeliveryTips;
6
7      class Delivery {
8          string phoneNumber;
9          string streetName;
10         int houseNumber;
11         optional(1) int apartmentNumber;
12         optional(2) DeliveryTips deliveryTips;
13     };
14
15     interface Promotion {
16         double CountTips(Delivery delivery);
17     };
18 };
19
```

W ICE wystarczy oznaczyć poprzez pole "optional()" a jako argument podać liczbę naturalną która się wcześniej nie powtórzyło.

gRPC:

```
7
8  message Delivery {
9      string phoneNumber = 1;
10     string streetName = 2;
11     int32 houseNumber = 3;
12     optional int32 apartmentNumber = 4;
13     message DeliveryTips {
14         repeated double tip = 1;
15     }
16     optional DeliveryTips deliveryTips = 5;
17 }
18
19 message DeliveryTip {
20     double deliveryAverageTips = 1;
21 }
22
23 service Promotion {
24     rpc CountTips(Delivery) returns (DeliveryTip) {}
25 }
26
```

W gRPC wystarczy oznaczyć poprzez pole "optional".

Implementacja:

ICE:

```
2 usages
9      public class PromotionI implements Promotion {
10          1 usage
11          @Override
12          public double CountTips(Delivery delivery, Current current) {
13              try {
14                  double[] tipsArray = delivery.getDeliveryTips();
15
16                  if (tipsArray != null && tipsArray.length > 0) {
17                      double sum = 0;
18                      for (double tip : tipsArray) {
19                          sum += tip;
20                      }
21                      return sum / tipsArray.length;
22                  } else {
23                      return 0.0;
24                  }
25              } catch (java.util.NoSuchElementException e) {
26                  System.err.println("Lista deliveryTips nie została podana.");
27                  e.printStackTrace();
28                  return 0.0;
29              }
30          }
31      }
```

Podczas implementacji metody do obliczania średniej próbę pobrania listy napiwków należy opakować w try/catch, ponieważ zwracany jest nam niżej zaprezentowany exception.

```
1 usage
104      public double[] getDeliveryTips()
105      {
106          if(!_deliveryTips)
107          {
108              throw new java.util.NoSuchElementException("deliveryTips is not set");
109          }
110          return deliveryTips;
111      }
112  }
```

gRPC:

```
10 public class PromotionImpl extends PromotionGrpc.PromotionImplBase {
11     1 usage
12     @Override
13     public void countTips(Delivery request,
14                           io.grpc.stub.StreamObserver<sr.grpc.gen.DeliveryTip> responseObserver){
15         List<Double> tipsList = request.getDeliveryTips().getTipList();
16
17         double averageTip;
18         if (tipsList.isEmpty()) {
19             averageTip = Double.NaN;
20         } else {
21             double sum = 0;
22             for (Double tip : tipsList) {
23                 sum += tip;
24             }
25             averageTip = sum / tipsList.size();
26         }
27
28         DeliveryTip deliveryTipResponse = DeliveryTip.newBuilder()
29             .setDeliveryAverageTips(averageTip)
30             .build();
31
32         responseObserver.onNext(deliveryTipResponse);
33         responseObserver.onCompleted();
34     }
35 }
```

W gRPC w przeciwieństwie do ICE'a nie musimy opakowywać próby pobrania danych oznaczonych poprzez optional w try/catch, ponieważ nie zostanie nam rzucony wyjątek. Jednak wypada samemu zadbać o odpowiednie obwarunkowanie przypadku gdy opcjonalne pola nie zostaną podane.

Serializacja danych:

ICE:

W przypadku gdy nie podamy listę napiwków:

Zapytanie na serwer:

> Frame 16: 184 bytes on wire (1472 bits), 184 bytes captured (1472 bits) on interface \Device\NPF_{Loopback}	0000 02 00 00 00 45 00 00 b4 80 59 40 00 80 06 00 00E...Y@.....
> Null/Loopback	0010 7f 00 00 01 7f 00 00 02 8b ac 27 10 b4 85 92 64f...f...d
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.2	0020 92 b3 23 a7 50 18 7f fc 2c 93 00 00 49 63 65 50#...P...IceP
> Transmission Control Protocol, Src Port: 35756, Dst Port: 10000, Seq: 1, Ack: 15, Len: 140	0030 01 00 01 00 00 02 8c 00 00 00 73 00 00 00 42 5as...s...BZ
> Internet Communications Engine Protocol	0040 68 31 31 41 59 26 53 59 ce 9d 31 97 00 00 06 7f h11AY&SY...1....
Magic Number: IceP	0050 80 70 a0 80 08 60 08 06 50 5c 00 0c 00 2e 2f df "P"...P...J...
Protocol Major: 1	0060 20 20 00 48 6a 69 a4 00 f5 18 80 1e a0 01 a5 3dHji.....=
Protocol Minor: 0	0070 20 1e a6 86 80 c8 03 4f 51 a8 66 ce 88 11 00 d10...Q...f...
Encoding Major: 1	0080 82 1a 97 c8 e0 f9 ac b9 c1 97 31 fa c2 22 b8 441...".D
Encoding Minor: 0	0090 c2 78 af 5f 57 61 c2 95 23 af 02 38 6d 12 45 68 ...x...Wa...#...8m...Eh
Message Type: Request (0)	00a0 77 d1 36 4e b5 12 9d 98 cc c5 3a 02 01 f8 bb 92 w...6N...:.....
Compression Status: Compressed, sender can accept a compressed reply (2)	00b0 29 c2 84 86 74 e9 8c b8}...t...
Message Size: 140	
> Request Message Body	
Request Identifier: 115	
Object Identity Name: Zh11AY&SY	
Object Identity Content:	
> [[Expert Info (Warning/Protocol): facet can be max one element]	
[facet can be max one element]	
[Severity level: Warning]	
[Group: Protocol]	

Odpowiedź z serwera:

> Frame 18: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface \Device\NPF_{Loopback}	0000 02 00 00 00 45 00 00 49 f8 67 40 00 80 06 00 00E...I...g@.....
> Null/Loopback	0010 7f 00 00 02 7f 00 00 01 27 10 8b ac 92 b3 23 a7f...f...e...
> Internet Protocol Version 4, Src: 127.0.0.2, Dst: 127.0.0.1	0020 b4 85 92 f0 50 18 7f dd ab 79 00 00 49 63 65 50P...y...IceP
> Transmission Control Protocol, Src Port: 10000, Dst Port: 35756, Seq: 15, Ack: 141, Len: 33	0030 01 00 01 00 02 01 21 00 00 00 01 00 00 00 0e1...e...s...
> Internet Communications Engine Protocol	0040 00 00 00 01 01 00 00 00 00 00 00 00 00 00s...s...s...
Magic Number: IceP	
Protocol Major: 1	
Protocol Minor: 0	
Encoding Major: 1	
Encoding Minor: 0	
Message Type: Reply (2)	
Compression Status: Uncompressed, sender can accept a compressed reply (1)	
Message Size: 33	
> Reply Message Body	
Request Identifier: 1	
Reply Status: Success (0)	
Reported reply data: 0e00000001010000000000000000	

W przypadku gdy podamy listę napiwków:

Zapytanie na serwer:

> Frame 1071: 208 bytes on wire (1664 bits), 208 bytes captured (1664 bits) on interface \Device\NPF_{Loopback}	0000 02 00 00 00 45 00 00 cc 80 69 40 00 80 06 00 00E...i...g@.....
> Null/Loopback	0010 7f 00 00 01 7f 00 00 02 8c 66 27 10 53 64 cf 41f...f...Sd-A
> Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.2	0020 b1 d4 05 f1 50 18 7f fc d4 6e 00 00 49 63 65 50P...n...IceP
> Transmission Control Protocol, Src Port: 35942, Dst Port: 10000, Seq: 1, Ack: 15, Len: 164	0030 01 00 01 00 00 02 a4 00 00 00 04 00 00 00 42 5as...s...BZ
> Internet Communications Engine Protocol	0040 68 31 31 41 59 26 53 59 f2 05 18 d7 00 00 09 7f h11AY&SY...1....
Magic Number: IceP	0050 80 f8 b0 82 2a 42 08 0e 50 5c 00 0c 00 2e 2f dfH...P...J...
Protocol Major: 1	0060 20 02 00 a0 00 75 0d 28 00 00 68 34 d0 03 41 88s...u...h4...A
Protocol Minor: 0	0070 35 c2 26 49 b4 46 d4 da 69 18 46 10 0d ea 91 bc 528I-F...i:F...
Encoding Major: 1	0080 46 c3 00 fd 18 22 01 10 d8 1b 11 c9 92 2f 24 10F...xT...#...t...
Encoding Minor: 0	0090 9f ce 46 d1 d5 f1 78 54 21 d1 c0 b0 23 f0 74 0a "A...x...V...<...Q...t...
Message Type: Request (0)	00a0 22 41 f1 78 19 56 a6 f4 ad 3c 8e 98 51 0e fd 12 ...n...n...:...W...
Compression Status: Compressed, sender can accept a compressed reply (2)	00b0 98 6e a2 91 93 0a d6 a6 dc 7d 9c 14 87 4d 93 2d&.....N...\$...F\$...
Message Size: 164	00c0 0d c1 e0 26 bf c5 dc 91 4e 14 24 3c 81 46 35 c0
> Request Message Body	
Request Identifier: 148	
Object Identity Name: Zh11AY&SY	
> [[Expert Info (Error/Malformed): missing or truncated string]	
[missing or truncated string]	
[Severity level: Error]	
[Group: Malformed]	

Odpowiedź z serwera:

> Frame 1073: 77 bytes on wire (616 bits), 77 bytes captured (616 bits) on interface \Device\NPF_{Loopback}	0000 02 00 00 00 45 00 00 49 f8 75 40 00 80 06 00 00E...I...u@.....
> Null/Loopback	0010 7f 00 00 02 7f 00 00 01 27 10 8c 66 b1 d4 05 f1f...f...f...
> Internet Protocol Version 4, Src: 127.0.0.2, Dst: 127.0.0.1	0020 53 64 cf e5 50 18 7f d7 27 0d 00 00 49 63 65 50 Sd...P...n...IceP
> Transmission Control Protocol, Src Port: 10000, Dst Port: 35942, Seq: 15, Ack: 165, Len: 33	0030 01 00 01 00 02 01 21 00 00 00 01 00 00 00 0e1...e...s...
> Internet Communications Engine Protocol	0040 00 00 00 01 01 77 77 77 77 77 77 77 13 40s...s...s...
Magic Number: IceP	
Protocol Major: 1	
Protocol Minor: 0	
Encoding Major: 1	
Encoding Minor: 0	
Message Type: Reply (2)	
Compression Status: Uncompressed, sender can accept a compressed reply (1)	
Message Size: 33	
> Reply Message Body	
Request Identifier: 1	
Reply Status: Success (0)	
Reported reply data: 0e000000010177777777777777771340	

W obu przypadkach wszystkie dane są kompresowane.

gRPC:

W przypadku gdy nie podamy listę napiwków:

Zapytanie na serwer:

The image shows a Wireshark packet capture of a gRPC request. The left pane displays the packet details tree, and the right pane shows the packet bytes and their hexadecimal representation.

Packet Details:

- ...1 = End Stream: True
- 0... = Reserved: 0x0
- ...0000 0000 0000 0000 0000 0011 = Stream Identifier: 3
- [Pad Length: 0]
- DATA payload (33 bytes)
- [Connection window size (before): 1048543]
- [Connection window size (after): 1048510]
- [Stream window size (before): 4194304]
- [Stream window size (after): 4194271]
- GRPC Message: /Promotion/CountTips, Request
- 0... = Frame Type: Data (0)
- ...0 = Compressed Flag: Not Compressed (0)
- Message Length: 28
- Message Data: 28 bytes
- Protocol Buffers: /Promotion/CountTips,request
- Message: <UNKNOWN> Message Type
- Field(1):
 - [Field Name: <UNKNOWN>]
 - ...000 1... = Field Number: 1
 - ...010 = Wire Type: Length-delimited (2)
 - Value Length: 15
 - Value: 2b34382035353520353535203535353535
- Field(2):
 - [Field Name: <UNKNOWN>]
 - ...001 0... = Field Number: 2
 - ...010 = Wire Type: Length-delimited (2)
 - Value Length: 7
 - Value: 42756472796b61
- Field(3): 2 (uint32)
 - [Field Name: <UNKNOWN>]
 - ...001 1... = Field Number: 3
 - ...000 = Wire Type: varint (0)
 - Value: 02

Packet Bytes:

0000 02 00 00 00 45 00 00 7d 04 2e 40 00 80 06 00 00 ...E...>@...
0010 7f 00 00 01 7f 00 00 05 8c b7 c3 83 25 a6 29 6f ...k...o
0020 9b a1 a7 4d 50 18 27 f8 7a 66 00 00 00 00 08 01 ...NP...zf...
0030 04 00 00 00 03 c2 83 86 c1 c0 bf be 00 00 04 ...
0040 00 00 00 00 03 00 00 00 05 00 00 21 00 01 00 ...
0050 00 00 03 00 00 00 00 1c 0a 0f 2b 34 38 20 35 35 ...448 55
0060 35 20 35 35 35 20 35 35 35 12 07 42 75 64 72 79 5 555 55 5--Budny
0070 6b 61 18 02 00 00 04 08 00 00 00 00 00 00 00 00 ka...
0080 05

Odpowiedź z serwera:

The image shows a Wireshark packet capture of a gRPC response. The left pane displays the packet details tree, and the right pane shows the packet bytes and their hexadecimal representation.

Packet Details:

- [Connection window size (after): 4194276]
- [Stream window size (before): 4194309]
- [Stream window size (after): 4194295]
- GRPC Message: /Promotion/CountTips, Response
- 0... = Frame Type: Data (0)
- ...0 = Compressed Flag: Not Compressed (0)
- Message Length: 9
- Message Data: 9 bytes
- Protocol Buffers: /Promotion/CountTips,response
- Message: <UNKNOWN> Message Type
- Field(1): 9221120237041090560 (uint64)
 - [Field Name: <UNKNOWN>]
 - ...000 1... = Field Number: 1
 - ...001 = Wire Type: 64-bit (1)
 - Value: 0000000000000f87f

HyperText Transfer Protocol 2

- Stream: HEADERS, Stream ID: 3, Length 1
- Length: 1
- Type: HEADERS (1)
- Flags: 0x05, End Headers, End Stream
- 00.0 ... = Unused: 0x00
- ..0 ... = Priority: False
- ...0 ... = Padded: False
- ...1 ... = End Headers: True
- ...1 ... = End Stream: True
- 0... = Reserved: 0x0
- ...0000 0000 0000 0000 0000 0011 = Stream Identifier: 3
- [Pad Length: 0]
- Header Block Fragment: be
- [Header Length: 20]
- [Header Count: 1]
- Header: grpc-status: 0
- [Time since request: 0.007331000 seconds]
- [Request in frame: 1169]

Packet Bytes:

0000 02 00 00 00 45 00 00 56 18 3e 40 00 80 06 00 00 ...E...>@...
0010 7f 00 00 05 7f 00 00 01 c3 83 8c ff 6b 06 44 b4 ...k...D
0020 63 66 c7 9a 50 18 27 f7 c6 be 00 00 00 00 04 01 cf...P...
0030 04 00 00 00 03 88 c1 c0 bf 00 00 00 00 00 00 ...
0040 00 03 00 00 00 00 09 09 00 00 00 00 00 f8 7f ...
0050 00 00 01 01 05 00 00 00 03 be

W przypadku gdy podamy listę napiwków:
Zapytanie na serwer:

The image shows a Wireshark packet capture of a gRPC request. The left pane displays the protocol tree and details. The right pane shows the raw packet data in hexadecimal and ASCII.

Protocol Tree:

- ▼ Protocol Buffers: /Promotion/CountTips,request
 - ▼ Message: <UNKNOWN> Message Type
 - ▼ Field(1):
 - [Field Name: <UNKNOWN>]
 - .000 1... = Field Number: 1
 -010 = Wire Type: Length-delimited (2)
 - Value Length: 15
 - Value: 2b343820353535203535203535203535
 - ▼ Field(2):
 - [Field Name: <UNKNOWN>]
 - .001 0... = Field Number: 2
 -010 = Wire Type: Length-delimited (2)
 - Value Length: 7
 - Value: 42756472796b61
 - ▼ Field(3): 2 (uint32)
 - [Field Name: <UNKNOWN>]
 - .001 1... = Field Number: 3
 -000 = Wire Type: varint (0)
 - > Value: 02
 - ▼ Field(4): 501 (uint32)
 - [Field Name: <UNKNOWN>]
 - .010 0... = Field Number: 4
 -000 = Wire Type: varint (0)
 - > Value: f503
 - ▼ Field(5):
 - [Field Name: <UNKNOWN>]
 - .010 1... = Field Number: 5
 -010 = Wire Type: Length-delimited (2)
 - Value Length: 26
 - Value: 0a1800000000000044000000000000001640000000000000c40
- ▼ HyperText Transfer Protocol 2

Raw Packet Data (Hex/ASCII):

```
0000 02 00 00 00 45 00 00 9c 04 46 40 00 80 06 00 00 .....E...F@....
0010 7f 00 00 01 7f 00 00 05 8c ff c3 83 63 66 c7 ab .....k-D-P...|.....
0020 6b 06 44 f3 50 18 27 f8 f6 7c 00 00 00 00 00 01 k-D-P...|.....
0030 04 00 00 00 05 c3 c2 83 86 c1 c0 bf be 00 00 04 .....@.....
0040 08 00 00 00 05 00 00 00 05 00 00 00 40 00 01 00 .....@.....
0050 00 00 05 00 00 00 3b 0a 0f 2b 34 38 20 35 35 .....+48 55
0060 35 20 35 35 35 20 35 35 35 12 07 42 75 64 72 79 5 555 55 S-Budry
0070 6b 61 18 02 20 f5 83 2a 1a 0a 18 00 00 00 00 ka...*.....
0080 00 04 40 00 00 00 00 00 00 16 40 00 00 00 00 00 .....@.....@.....
0090 00 0c 40 00 00 04 08 00 00 00 00 00 00 00 0e .....@.....
```

Odpowiedź z serwera:

The image shows a Wireshark packet capture of a gRPC response. The left pane displays the protocol tree and details. The right pane shows the raw packet data in hexadecimal and ASCII.

Protocol Tree:

- ▼ GRPC Message: /Promotion/CountTips, Response
 - 0... .. = Frame Type: Data (0)
 -0 = Compressed Flag: Not Compressed (0)
 - Message Length: 9
 - Message Data: 9 bytes
- ▼ Protocol Buffers: /Promotion/CountTips,response
 - ▼ Message: <UNKNOWN> Message Type
 - ▼ Field(1): 4615814318085810859 (uint64)
 - [Field Name: <UNKNOWN>]
 - .000 1... = Field Number: 1
 -001 = Wire Type: 64-bit (1)
 - > Value: abaaaaaaaaa0e40
- ▼ HyperText Transfer Protocol 2
 - ▼ Stream: HEADERS, Stream ID: 5, Length 1
 - Length: 1
 - Type: HEADERS (1)
 - ▼ Flags: 0x05, End Headers, End Stream
 - 00.0 .0. = Unused: 0x00
 - .0. = Priority: False
 - 0... = Padded: False
 -1. = End Headers: True
 -1 = End Stream: True
 - 0... .. = Reserved: 0x0
 - .000 0000 0000 0000 0000 0000 0101 = Stream Identifier: 5
 - [Pad Length: 0]
 - Header Block Fragment: be
 - [Header Length: 20]
 - [Header Count: 1]
 - > Header: grpc-status: 0
 - [Time since request: 0.042214000 seconds]
 - [\[Request in frame 1417\]](#)

Raw Packet Data (Hex/ASCII):

```
0000 02 00 00 00 45 00 00 56 18 42 40 00 80 06 00 00 .....E..V.B@....
0010 7f 00 00 05 7f 00 00 01 c3 83 8c ff 6b 06 44 f3 .....k-D-
0020 63 66 c8 1f 50 18 27 f7 ab 38 00 00 00 04 01 cf..P...8.....
0030 04 00 00 00 05 88 c1 c0 bf 00 00 0e 00 00 00 00 .....@.....
0040 00 05 00 00 00 00 09 09 ab aa aa aa aa 0e 40 .....@.....
0050 00 00 01 01 05 00 00 00 05 be .....@.....
```

Jak możemy zauważyć gRPC używa http2 wartości które są stringami nie są kodowane, natomiast cała reszta już tak.