

Protocol Buffers

Carlos, Jean, Mateus, Victor, Ruan



Agenda:

- 1 → O que é *protocol buffers*
- 2 → Criar arquivos .proto
- 3 → Compilador em ação
- 4 → gRPC para *remote procedure call*
- 5 → Exemplos, bibliografia e material do curso

1

O que é *protocol buffers*?



Protocol Buffers

- Representação externa de dados estruturados.
- Criado pelo *Google*
- Formato de arquivos **.proto** e compilados para linguagens específicas.



Protocol Buffers

- Usado como IDL para *remote procedure Call* (RPC).
- Compilador protoc gera código C++, java e python.
- Salva apenas os dados em formato binário – Não interpretável por humanos.

2

Criação de arquivos .proto



Criar arquivos .proto

- Tipos de dados: bool, int32, float, double, string.
- Valores padrão: falso para bool, 0 para int e vazio para string.
- É possível aninhar tipos de dados criados.



Criar arquivos .proto

```
syntax = "proto3";
```

```
import "/home/tutorial1.proto";
```

Linha 1: Define a sintaxe do arquivo. Se não for definida a sintaxe, o compilador adota proto2 por padrão.



Criar arquivos .proto

```
syntax = "proto3";
```

```
import "/home/tutorial1.proto";
```

É possível reutilizar código de um arquivo .proto existente. Realize a importação com **import**.



Criar arquivos .proto

```
message SearchRequest {  
    string query = 1;  
    int32 page_number = 2;  
    int32 result_per_page = 3;  
}
```

```
message Foo {  
    reserved 2, 15, 9 to 11;  
    reserved "foo", "bar";  
}
```

```
message SearchRequest {  
    string query = 1;  
    int32 page_number = 2;  
    int32 result_per_page = 3;  
    enum Corpus {  
        UNIVERSAL = 0;  
        WEB = 1;  
        IMAGES = 2;  
        LOCAL = 3;  
        NEWS = 4;  
        PRODUCTS = 5;  
        VIDEO = 6;  
    }  
    Corpus corpus = 4;  
}
```



Criar arquivos .proto

```
message SearchRequest {  
    string query = 1;  
    int32 page_number = 2;  
    int32 result_per_page = 3;  
}
```

```
message Foo {  
    reserved 2, 15, 9 to 11;  
    reserved "foo", "bar";  
}
```

Message: Cada *message* será mapeada para uma classe que poderá ser instanciada.



Criar arquivos .proto

```
message SearchRequest {  
    string query = 1;  
    int32 page_number = 2;  
    int32 result_per_page = 3;  
}
```

```
message Foo {  
    reserved 2, 15, 9 to 11;  
    reserved "foo", "bar";  
}
```

Tag: Cada campo tem uma tag associada. É usada na codificação para binário.



Criar arquivos .proto

```
message SearchRequest {  
    string query = 1;  
    int32 page_number = 2;  
    int32 result_per_page = 3;  
}
```

```
message Foo {  
    reserved 2, 15, 9 to 11;  
    reserved "foo", "bar";  
}
```

Valores reservados: Indicam ao compilador que os valores de tag não podem ser usados, nem os nomes de campos especificados.

Motivo: Atualizei um arquivo.proto que tinha um campo **foo**. Se um usuário utilizar esse valor, posso ter incompatibilidade com a versão anterior, causando possíveis inconsistências.



Criar arquivos .proto

```
message SearchRequest {  
  string query = 1;  
  int32 page_number = 2;  
  int32 result_per_page = 3;  
  enum Corpus {  
    UNIVERSAL = 0;  
    WEB = 1;  
    IMAGES = 2;  
    LOCAL = 3;  
    NEWS = 4;  
    PRODUCTS = 5;  
    VIDEO = 6;  
  }  
  Corpus corpus = 4;  
}
```

Enumeração: Define os valores possíveis para o campo. Somente serão aceitos valores contidos no corpo do **enum**.

O primeiro campo deve ter tag 0. A tag 0 indica o valor *default* e compatibiliza com **proto2**.



Criar arquivos .proto

```
message SearchRequest {  
  string query = 1;  
  int32 page_number = 2;  
  int32 result_per_page = 3;  
  enum Corpus {  
    UNIVERSAL = 0;  
    WEB = 1;  
    IMAGES = 2;  
    LOCAL = 3;  
    NEWS = 4;  
    PRODUCTS = 5;  
    VIDEO = 6;  
  }  
  Corpus corpus = 4;  
}
```

Comentários: Possuem sintaxe similar a C e C++.

//Define o tipo de conteúdo do corpo da página



Criar arquivos .proto

```
message SearchResponse {  
  message Result {  
    string url = 1;  
    string title = 2;  
    repeated string snippets = 3;  
  }  
  repeated Result results = 1;  
}
```

repeated: Indica que esse campo é dinâmico. Você pode salvar 0 ou vários valores daquele tipo no campo.

Similar a uma lista dinâmica.



Criar arquivos .proto

```
message SampleMessage {  
  oneof test_oneof {  
    string name = 4;  
    SubMessage sub_message = 9;  
  }  
}
```

oneof: Indica que apenas um dos campos pode ser setado. Não podem haver campos **repeated** dentro de uma declaração **oneof**.

Métodos auxiliares para verificação se há um valor setado, e se houver, qual valor:

Case()
WichOneOf()



Tipos de dados aceitos – Arquivos .proto

Tipo .proto	Tipo C++	Tipo Java	Tipo Python
double	double	double	float
float	float	float	float
int32	int32	int	int
int64	int64	long	int/long
uint32	uint32	int	int/long
uint64	uint64	long	int/long
sint32	int32	int	int
sint64	int64	long	int/long
fixed32	uint32	int	int
fixed64	uint64	long	int/long
bool	bool	boolean	bool
string	string	String	str/unicode
bytes	string	ByteString	str

Adaptado de <https://developers.google.com/protocol-buffers/docs/proto#scalar>

3

Geração de código com protoc



.proto → Classe

- O compilador aceita muitos parâmetros para personalizar a saída. De forma simplificada:
- **Parâmetro 1:** protoc
- **Parâmetro 2:** arquivo.proto
- **Parâmetro 3:** linguagem de saída
- **Parâmetro 4:** Diretório de saída (pasta atual =.)



.proto → Classe

■ Executando...

```
$ protoc arquivo.proto --java_out=.
```

```
$ protoc arquivo.proto --cpp_out=.
```

```
$ protoc arquivo.proto --python_out=.
```

4

Framework gRPC



O que é o gRPC

- *Framework* para chamada de procedimentos remotos aberto, universal e de alto desempenho.
- Usa o protocolo HTTP/2 para transporte.
- Usa *protocol buffers* como linguagem de descrição da interface (IDL).



Recursos do gRPC

- ▬ Autenticação;
- ▬ Transmissão bidirecional;
- ▬ Controle de fluxo;
- ▬ Tempo limite (temporização);



Tipos de serviço providos

- ▀ RPC simples
- ▀ RPC de transimissão do lado do servidor
- ▀ RPC de transmissão do lado do cliente
- ▀ RPC de transmissão bidirecional



Tipos de serviço providos

RPC simples

O cliente envia uma solicitação ao servidor usando o stub e aguarda uma resposta de retorno, assim como uma chamada de função normal.

RPC de transmissão do lado do servidor

O cliente envia uma solicitação ao servidor e recebe um fluxo para ler uma seqüência de mensagens de volta.



Tipos de serviço providos

RPC de transmissão do lado do cliente

Onde o cliente escreve uma seqüência de mensagens e as envia para o servidor, novamente usando um fluxo fornecido. Em seguida ele aguarda resposta do servidor. O método de transmissão do lado do cliente têm a palavra chave ***stream*** antes do tipo da solicitação.

RPC de transmissão bidirecional

Onde ambos os lados enviam uma seqüência de mensagens usando um fluxo de leitura e gravação. Não há ordem prevista nessa troca de mensagens. Você especifica esse tipo de método colocando a palavra-chave ***stream*** antes da solicitação e da resposta.

5

Prática, materiais e referências



Documentação oficial *Protocol Buffers*:

- <https://developers.google.com/protocol-buffers/>

Documentação oficial gRPC

- <https://grpc.io>

Outros materiais:

- <https://spline.de/static/talks/protobuf.pdf>

Mini curso disponível em:

<https://github.com/Carlos-Henreis/COM242>



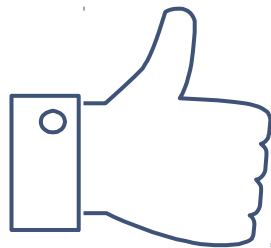
Tutoriais de instalação e
compilação, slides



Vídeo de instalação do
compilador



Arquivos.proto , código
fonte



OBRIGADO!

Dúvidas? Sugestões?