Protocol Buffers

Between JSON and XML

by
Rafał Hryciuk (rhr@touk.pl)

What Are Protocol Buffers?

Protocol buffers are Google's language-neutral, platform-neutral, extensible mechanism for serializing structured data – think XML, but smaller, faster, and simpler. You define how you want your data to be structured once, then you can use special generated source code to easily write and read your structured data to and from a variety of data streams and using a variety of languages – Java, C++, or Python.

```
package pl.touk.workshop.protobuf.messages;

option java_package = "pl.touk.workshop.protobuf.messages";
option java_outer_classname = "PersonMessage";

message Person {
    required string name = 1;
    required int32 id = 2;
    optional string email = 3;

    enum PhoneType {
        MOBILE = 0;
        HOME = 1;
        WORK = 2;
    }

    message PhoneNumber {
        required string number = 1;
        optional PhoneType type = 2 [default = HOME];
    }

    repeated PhoneNumber phone = 4;
}
```

Field Rules

- required: exactly one (required is forever)
- optional: zero or one ([default = 10])
- repeated: any number of times (including zero) [packed=true]

Data Types

- scalar types - double, float, bool, string, bytes, int32, int64, uint32 ...
- enumerations
- composite types

Features

- Importing definitions
- Nested types
- Extensions

```
message Foo {
    optional int32 foo = 1;
    extensions 100 to 199;
}

extend Foo {
    optional int32 bar = 126;
}
```

Extending a Protocol Buffer

- you must not change the tag numbers of any existing fields.
- you must not add or delete any required fields.
- you may delete optional or repeated fields.
- you may add new optional or repeated fields

If you follow these rules, old code will happily read new messages and simply ignore any new fields

Why not just use XML?

Protocol buffers have many advantages over XML for serializing structured data. Protocol buffers:

- are simpler
- are 3 to 10 times smaller
- are 20 to 100 times faster
- are less ambiguous
- generate data access classes that are easier to use programmatically

Advantages over JSON?

- Defined and validatable message structure
- JSON natively doesn't support binary data

Efficiency

Comparison for Person with 1000 phone numbers

	Object Creation	Serialization	Serialization Size	Deserialization
Protobuf	32	17	15022	10
JSON	2	73	38061	54

Times in ms, size in bytes.

Third-Party Add-ons for Protocol Buffers

- Action Script, C, C++, C#, Clojure, Common Lisp, D, Erlang, Go, Haskell, Java, Java ME, Javascript, Lua, Matlab, Mercury, Objective C, OCaml, Perl, Perl/XS, PHP, Python, R, Ruby, Scala, Visual Basic
- RPC Implementations
- IDE plugins

Interesting details

- BSD license
- "Protocol buffers are now Google's lingua franca for data – at time of writing, there are 48,162 different message types defined in the Google code tree across 12,183 .proto files. They're used both in RPC systems and for persistent storage of data in a variety of storage systems."

Name comes from early days where there was just a class ProtocolBuffer with ability to push pairs <key, value>

- First public version of Protocol Buffers was 2. Version 1 developed since 2001 was just inside Google and too messy to publish.

Reference

- https://developers.google.com/protocol-buffers/?hl=pl
- git://github.com/rhryciuk/protobuf-example.git