

구글's Protocol Buffer

Knight76 at gmail.com

김용환

개념

1. RPC와 쉽게 연동, Not RPC
2. 하나의 개념으로 다양한 언어에서 쓸 수 있게
구글에서 다양한 언어별로 개발 하다 보니. 통신을 위한 단일 표준이 필요



References

- **Protocol Buffers: A new open source release**
 - <http://www.youtube.com/watch?v=K-e8DDRwVUg>
- **Home Page**
 - <http://code.google.com/p/protobuf/>

Protocol Buffer 이야기

- Not socket
- 구글 에서 2008년 7월 발표
- 이슈 제시
 - XML 문제
 - Parsing, serialization (debugging)
 - Portable : IDL처럼 사용
 - Heavy Optimization
 - Language 지원
- 짧은 데이터의 송수신 용도/긴 데이터 송수신 이 목표가 아님

XML보다 좋은 장점

- Simple
- 3~10배 작음
- 20~100배 속도 빠름
- 모호하지 않음
- 바로 프로그램에 사용하기 쉬움

<http://code.google.com/intl/ko-KR/apis/protocolbuffers/docs/overview.html>

어디서 쓰고 있나?

- 구글
 - 원래 index server request/response protocol로 사용했었음
 - 48,162 different message types
 - 12,183 .proto files
- 다양한 회사
- 국내/외 게임 회사의 통신

장점

- 쓰기 편함
- Stub 코드 자동 생성
 - 통신에서 가져야 할 보편적 특성을 다 추가
 - Serializing, Parsing 지원
- 코드 일치
 - 클라이언트/서버 코드 동일
- IDL 형태로 정의가 단순
 - Portable
 - 클래스 또는 struct 디자인
- 언어 지원
 - java, c++, python
 - 3rd party lib (많은 언어 지원.
http://code.google.com/p/protobuf/wiki/ThirdPartyAddOns#RPC_Implementations)
- 배우기 쉬움
- 이클립스 플러그인 존재
- Lite 버전 개발 가능
- Good Document
- BSD license
- 언어마다 특화되고 쓰기 편한 특징을 제공

단점?

- Output이 binary만 존재
 - PB의 Reflection을 이용해서 json으로 전달 가능
- Map, set 지원 없음

개발자 가이드

[My favorites](#) ▼ | 1

Google code
e.g. "adwords" or "open source"

Protocol Buffers

[Home](#) [Docs](#) [FAQ](#) [Forum](#) [Download](#) [+1](#)

Developer Guide

[Overview](#)

[Language Guide](#)

[Style Guide](#)

[Encoding](#)

[Tutorials](#)

[Techniques](#)

[Add-ons](#)

API Reference

[Reference Overview](#)

[C++ Generated Code](#)

[C++ API](#)

[Java Generated Code](#)

[Java API \(javadoc\)](#)

[Python Generated Code](#)

[Python API \(epydoc\)](#)

[Other Languages and
Plugins](#)

Developer Guide

Welcome to the developer documentation for protocol buffers – a language-neutral, platform-neutral, extensible way of serializing structured data for use in communications protocols, data storage, and more.

This documentation is aimed at Java, C++, or Python developers who want to use protocol buffers in their applications. This overview introduces protocol buffers and tells you what you need to do to get started – you can then go on to follow the [tutorials](#) or delve deeper into [protocol buffer encoding](#). [API reference documentation](#) is also provided for all three languages, as well as [language](#) and [style](#) guides for writing `.proto` files.

What are protocol buffers?

Protocol buffers are a flexible, efficient, automated 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. You can even update your data structure without breaking deployed programs that are compiled against the "old" format.

How do they work?

You specify how you want the information you're serializing to be structured by defining protocol buffer message types in `.proto` files. Each protocol buffer message is a small logical record of information, containing a series of name-value pairs. Here's a very basic example of a `.proto` file that defines a message containing information about a person:

```
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];
  }
}
```

<http://code.google.com/intl/ko-KR/apis/protocolbuffers/docs/overview.html>









개발환경 구성



- Download proto compiler in google code
- (Option) eclipse plugin

설치

- Proto compiler 설치

- <http://code.google.com/p/protobuf/downloads/list>

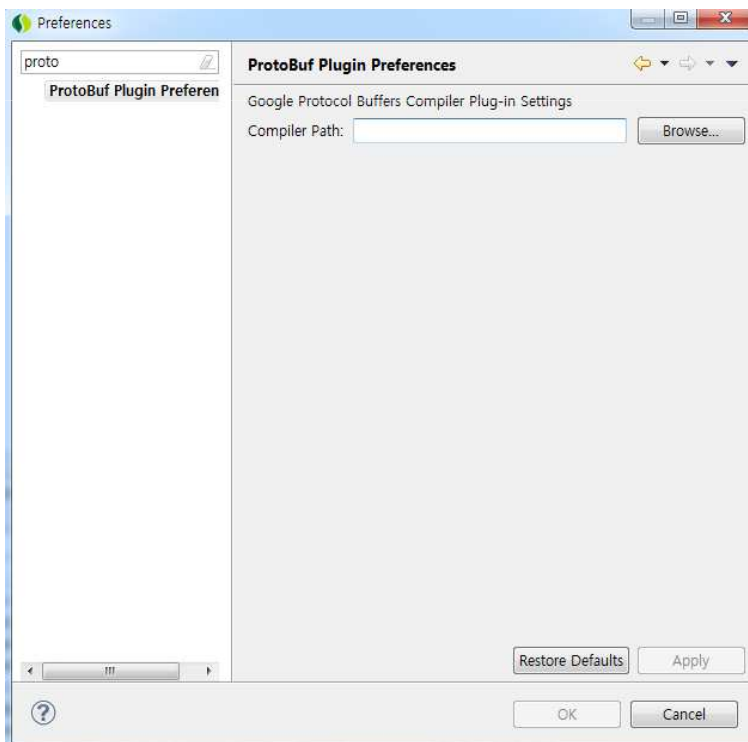
Filename ▼	Summary + Labels ▼
 protobuf-2.4.1.tar.bz2	Protocol Buffers 2.4.1 full source -- C++, Java, Python Featured
 protobuf-2.4.1.tar.gz	Protocol Buffers 2.4.1 full source -- C++, Java, Python Featured
 protobuf-2.4.1.zip	Protocol Buffers 2.4.1 full source -- C++, Java, Python Featured
 protoc-2.4.1-win32.zip	Protocol Buffers 2.4.1 compiler -- Windows binary Featured
 protobuf-2.3.0.tar.bz2	Protocol Buffers 2.3.0 full source -- C++, Java, Python
 protobuf-2.3.0.tar.gz	Protocol Buffers 2.3.0 full source -- C++, Java, Python
 protobuf-2.3.0.zip	Protocol Buffers 2.3.0 full source -- C++, Java, Python
 protoc-2.3.0-win32.zip	Protocol Buffers 2.3.0 compiler -- Windows binary

 protoc.exe
 readme.txt

이클립스 환경 셋팅

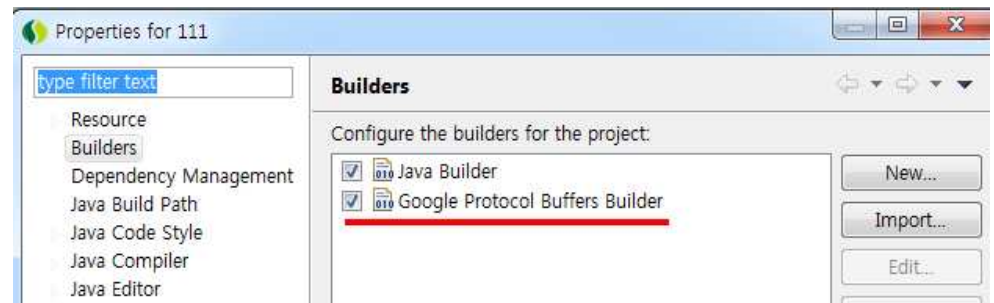
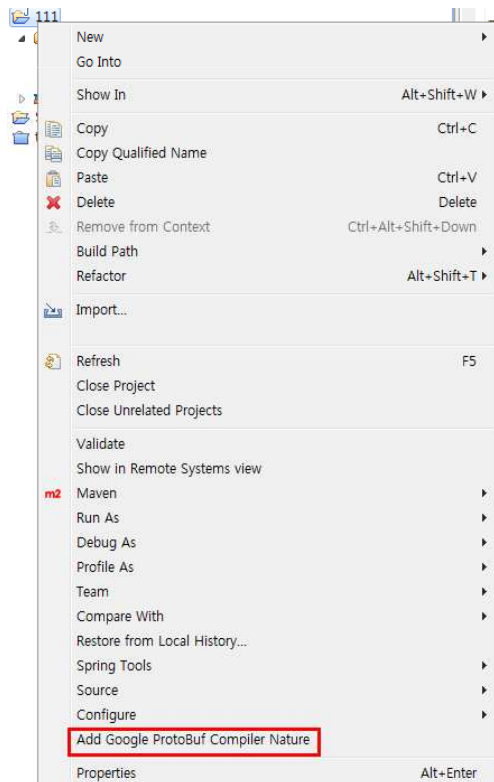
Eclipse 플러그인 사용 #1

- Protoclips 업데이트 주소
<http://protoclipse.googlecode.com/svn/trunk/site/>
- Windows-Preference-ProtoBuf Plugin Preferences 화면에서 protoc.exe compiler가 설치된 path를 넣는다.



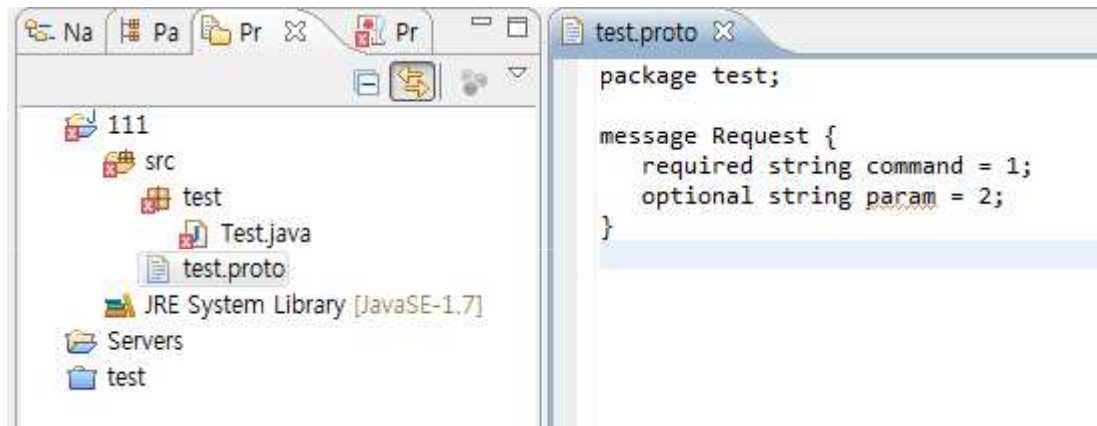
Eclipse 플러그인 사용 #2

- PB 컴파일 환경 구축
 - Project 오른쪽 버튼 -> Add Google ProtoBuf Compiler Nature 선택



Eclipse 플러그인 사용 #3

- src/test.proto 파일 생성



- test/Test.java 파일이 자동으로 생성
 - But, PB의 lib가 있어야 컴파일이 될 것임
 - 직접 다운받거나 maven 이용!

<http://repo2.maven.org/maven2/com/google/protobuf/protobuf-java/>

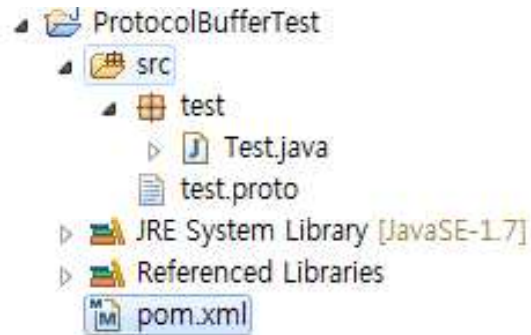
Eclipse 플러그인 사용 #4

- Pom.xml

```
<dependencies>  
<dependency>  
<groupId>com.google.protobuf</groupId>  
<artifactId>protobuf-java</artifactId>  
<version>2.4.1</version>  
</dependency>  
</dependencies>
```


Eclipse 플러그인 사용 #4

- 컴파일 완료된 것 확인



실습

PB Format->소스 전환

```
// text.txt
package test;

message Request {
  required string command = 1;
  optional string param = 2;
}
```

```
protoc -I=$SRC_DIR --java_out=$DST_DIR $SRC_DIR/addressbook.proto
```

```
/cygdrive/c/protocolbuffer
$ ./protoc.exe test.txt --java_out=.
```



Test폴더 생기고 TestTxt.java 생성됨
(17KB, Protocol buffer 내부 클래스 사용)

```
// c++ out
test.txt.pb.cc (12K)
test.txt.pb.h (8K)
```

```
// python out
/test/txt_pb2.py (2K)
```

naming

생성된 자바 코드

```
test
├── Test
│   ├── RequestOrBuilder
│   │   ├── hasCommand() : boolean
│   │   ├── getCommand() : String
│   │   ├── hasParam() : boolean
│   │   └── getParam() : String
│   └── Request
│       ├── getDefaultInstanceForType() : Request
│       ├── hasCommand() : boolean
│       ├── getCommand() : String
│       ├── hasParam() : boolean
│       ├── getParam() : String
│       ├── FisInitialized() : boolean
│       ├── writeTo(CodedOutputStream) : void
│       ├── getSerializedSize() : int
│       ├── newBuilderForType() : Builder
│       ├── toBuilder() : Builder
│       └── Builder
```

```
Builder
├── clear() : Builder
├── clone() : Builder
├── getDescriptorForType() : Descriptor
├── getDefaultInstanceForType() : Request
├── build() : Request
├── buildPartial() : Request
├── mergeFrom(Message) : Builder
├── mergeFrom(Request) : Builder
├── FisInitialized() : boolean
├── mergeFrom(CodedInputStream, ExtensionRegistryLite) : Builder
├── hasCommand() : boolean
├── getCommand() : String
├── setCommand(String) : Builder
├── clearCommand() : Builder
├── hasParam() : boolean
├── getParam() : String
├── setParam(String) : Builder
├── clearParam() : Builder
```

생성된 자바 코드

- Descriptor 지원
 - internal_static_test_Request_descriptor
- Reflection 지원
 - Message / Message.Builder interfaces.
 - Json 처럼 프로토콜로 변경 가능 (ajax 가능)
- 메시지 수정시 하위 호환 보장, 새로운 메시지로 변경되면 기존 코드에 대한 필드만 처리
- 파일이름을 디폴트로 해서 소스를 생성하지만, 내가 원하는 클래지 이름과 클래스 이름의 개별 지정이 가능
 - option java_package = "com.example.foo.bar";
 - option java_outer_classname = "ProtocolData";
- PB의 Enum은 java의 enum으로 변경

크기 제한

- 디폴트로 크기 제한
 - 64 MB
- 속도를 최적화 또는 악의를 가진 사용자로부터 보호하기 위해서 크기를 제한할 수 있음
 - CodedInputStream/CodedOutputStream
(ZeroCopyInputStream/ZeroCopyOutputStream)
 - SetTotalBytesLimit 메소드

유의할 점

- package 선언
- 클래스 파일 이름
- 운영을 위한 파일명 .proto
- message 등록
 - Protocol buffer language guide
- Protoc.exe(컴파일러)에 의해 만들어진 java, python, c++ 코드는 고치지 말아야 한다.(immutable)
- protoc에서 컴파일 되면, 자동으로 accessor가 붙는다.
 - get/set/has...

message

- package
- Type
 - Bool, int32, uint32, float, double, string, bytes,
 - Enum
- Nested type
- Default value
- importing
- Modifier
 - required : 반드시 사용해야 할 필드. 미초기화된 상태
미초기화된 메시지를 빌드하면 RuntimeException,
초기화되지 않은 메시지를 파싱하면서 에러나면 IOException발
생
 - optional : option의 개념
 - hasXXX() 로 확인
 - Repeated : 0을 포함하는 개수를 계속 넣을 수 있음
 - List객체로 구현됨

message

- 번호를 반드시 주는 이유
 - 번호를 주지 않으면 protoc 에러 발생
 - Write/Read 할 때, serialization 순서를 주기 위함
 - 필드 정보가 set되었는지 쉽게 알기 위함(내부적으로 bit 연산함)

Versioning 정보가 없는 이유

- 새로운 필드를 언제든지 추가 될 수 있음
- 모든 정보를 볼 필요 없이 필요한 정보만 파싱할 수 있도록 함
- But, java는 기본으로 존재하지 않지만, c++ 은 존재한다. 링킹 이슈.
(GOOGLE_PROTOBUF_VERIFY_VERSION 매크로)
 - Incompatible한 버전 때문에 문제가 없도록 해야 함

데모

DEMO #1 - 0

- PB 코드

```
// text.txt
package test;

message Request {
  required string command = 1;
  optional string param = 2;
}
```

DEMO #1 -1

```
package com.example.test;

import java.io.FileOutputStream;
import test.Test.Request;

public class Writer {
    public static void main(String[] args) throws Exception {
        Request request = Request.newBuilder()
            .setCommand("commit")
            .setParam("every files")
            .build();
        FileOutputStream output = new FileOutputStream("r.os");
        request.writeTo(output);
        output.close();
    }
}
```

DEMO #1 -2

```
package com.example.test;

import java.io.FileInputStream;
import test.Test.Request;

public class Reader {
    public static void main(String[] args) throws Exception {
        Request request = Request.parseFrom(new FileInputStream("r.os"));
        System.out.println("command : " + request.getCommand());
        if (request.hasParam()) {
            System.out.println("params : " + request.getParam());
        }
    }
}
```

DEMO #1 -3

command : commit
params : every files

```
package tutorial;

option java_package = "com.example.tutorial";
option java_outer_classname = "AddressBookProtos";

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;
}

message AddressBook {
  repeated Person person = 1;
}
```



DEMO #2

DEMO #3 – 1

생성된 c++ 코드를 이용해서 코딩

```
#include <iostream>
#include <fstream>
#include <string>
#include "test.pb.h"

using namespace std;

int main(int argc, char* argv[]) {
    Request request;
    request.set_command("init");
    request.set_param("0");

    fstream out("streams", ios::out | ios::binary | ios::trunc);
    request.SerializeToOstream(&out);
    out.close();

    return 1;
}
```

DEMO #3 – 1

생성된 c++ 코드를 이용해서 코딩

```
#include <iostream>
#include <fstream>
#include <string>
#include "test.pb.h"

using namespace std;

int main(int argc, char* argv[]) {
    Request request;
    fstream in("streams", ios::in | ios::binary);
    if (!request.ParseFromIstream(&in)) {
        cerr << "Failed to parse streams." << endl;
        exit(1);
    }
    cout << "command: " << request.command() << endl;
    if (request.has_param()) {
        cout << "param: " << request.param() << endl;
    }
}
```

기타 정보

Protocol Buffer Language Guide

- <http://code.google.com/intl/ko-KR/apis/protocolbuffers/docs/proto.html>

The screenshot shows the 'Protocol Buffers' website. At the top, there are navigation links: Home, Docs, FAQ, Forum, and Download. The main content is divided into two columns. The left column is a 'Developer Guide' with links for Overview, Language Guide, Style Guide, Encoding, Tutorials, Techniques, Add-ons, API Reference, Reference Overview, C++ Generated Code, C++ API, Java Generated Code, Java API (javadoc), Python Generated Code, Python API (epydoc), and Other Languages and Plugins. The right column is the 'Language Guide' with links for Defining A Message Type, Scalar Value Types, Optional And Default Values, Enumerations, Using Other Message Types, Nested Types, Updating A Message Type, Extensions, Packages, Defining Services, Options, and Generating Your Classes. Below these links, there is a paragraph explaining the guide's purpose and a reference to a tutorial. A section titled 'Defining A Message Type' follows, with a paragraph explaining a simple example and a code block showing a message definition for 'SearchRequest' with three fields: 'query' (required string), 'page_number' (optional int32), and 'result_per_page' (optional int32). A final paragraph explains the message definition and a section titled 'Specifying Field Types' is partially visible at the bottom.

Protocol Buffers Home Docs FAQ Forum Download

Developer Guide

- Overview
- Language Guide
- Style Guide
- Encoding
- Tutorials
- Techniques
- Add-ons

API Reference

- Reference Overview
- C++ Generated Code
- C++ API
- Java Generated Code
- Java API (javadoc)
- Python Generated Code
- Python API (epydoc)
- Other Languages and Plugins

Language Guide

- Defining A Message Type
- Scalar Value Types
- Optional And Default Values
- Enumerations
- Using Other Message Types
- Nested Types
- Updating A Message Type
- Extensions
- Packages
- Defining Services
- Options
- Generating Your Classes

This guide describes how to use the protocol buffer language to structure your protocol buffer data, including `.proto` file syntax and how to generate data access classes from your `.proto` files.

This is a reference guide – for a step by step example that uses many of the features described in this document, see the [tutorial](#) for your chosen language.

Defining A Message Type

First let's look at a very simple example. Let's say you want to define a search request message format, where each search request has a query string, the particular page of results you are interested in, and a number of results per page. Here's the `.proto` file you use to define the message type.

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

The `SearchRequest` message definition specifies three fields (name/value pairs), one for each piece of data that you want to include in this type of message. Each field has a name and a type.

Specifying Field Types

API

- <http://code.google.com/intl/ko-KR/apis/protocolbuffers/docs/reference/java/index.html>

The screenshot shows a web browser window displaying the API reference for `com.google.protobuf`. The browser's address bar shows the URL `code.google.com/intl/ko-KR/apis/protocolbuffers/docs/reference/java/index.html`. On the left side, there is a sidebar titled "All Classes" listing various classes and interfaces, including `AbstractMessage`, `BlockingRpcChannel`, and various `DescriptorProtos` classes. The main content area is titled "Package com.google.protobuf" and features an "Interface Summary" table. The table lists several interfaces and their descriptions:

Interface	Description
BlockingRpcChannel	Abstract interface for a blocking RPC channel.
BlockingService	Blocking equivalent to Service .
DescriptorProtos.DescriptorProto.ExtensionRangeOrBuilder	
DescriptorProtos.DescriptorProtoOrBuilder	
DescriptorProtos.EnumDescriptorProtoOrBuilder	
DescriptorProtos.EnumOptionsOrBuilder	
DescriptorProtos.EnumValueDescriptorProtoOrBuilder	
DescriptorProtos.EnumValueOptionsOrBuilder	
DescriptorProtos.FieldDescriptorProtoOrBuilder	
DescriptorProtos.FieldOptionsOrBuilder	
DescriptorProtos.FileDescriptorProtoOrBuilder	
DescriptorProtos.FileDescriptorSetOrBuilder	
DescriptorProtos.FileOptionsOrBuilder	
DescriptorProtos.MessageOptionsOrBuilder	
DescriptorProtos.MethodDescriptorProtoOrBuilder	
DescriptorProtos.MethodOptionsOrBuilder	
DescriptorProtos.ServiceDescriptorProtoOrBuilder	
DescriptorProtos.ServiceOptionsOrBuilder	
DescriptorProtos.SourceCodeInfo.LocationOrBuilder	
DescriptorProtos.SourceCodeInfoOrBuilder	
DescriptorProtos.UninterpretedOption.NamePartOrBuilder	
DescriptorProtos.UninterpretedOptionOrBuilder	

PB Format

- http://wiki.openstreetmap.org/wiki/PBF_Format

```
00000000 00 00 00 0d - length in bytes of the BlobHeader in network-byte order
00000000 -- -- -- -- 0a - S 1 'type'
00000000 -- -- -- -- 09 - length 9 bytes
00000000 -- -- -- -- 4f 53 4d 48 65 61 64 65 72 - "OSMHeader"
00000000 -- -- -- -- -- -- -- -- -- -- 18 - V 3 'datasize'
00000010 7c - 124 bytes long
00000010 -- 10 - V 2 'raw_size'
00000010 -- -- 71 - 113 bytes long
00000010 -- -- -- 1a - S 3 'zlib_data'
00000010 -- -- -- -- 78 - length 120 bytes

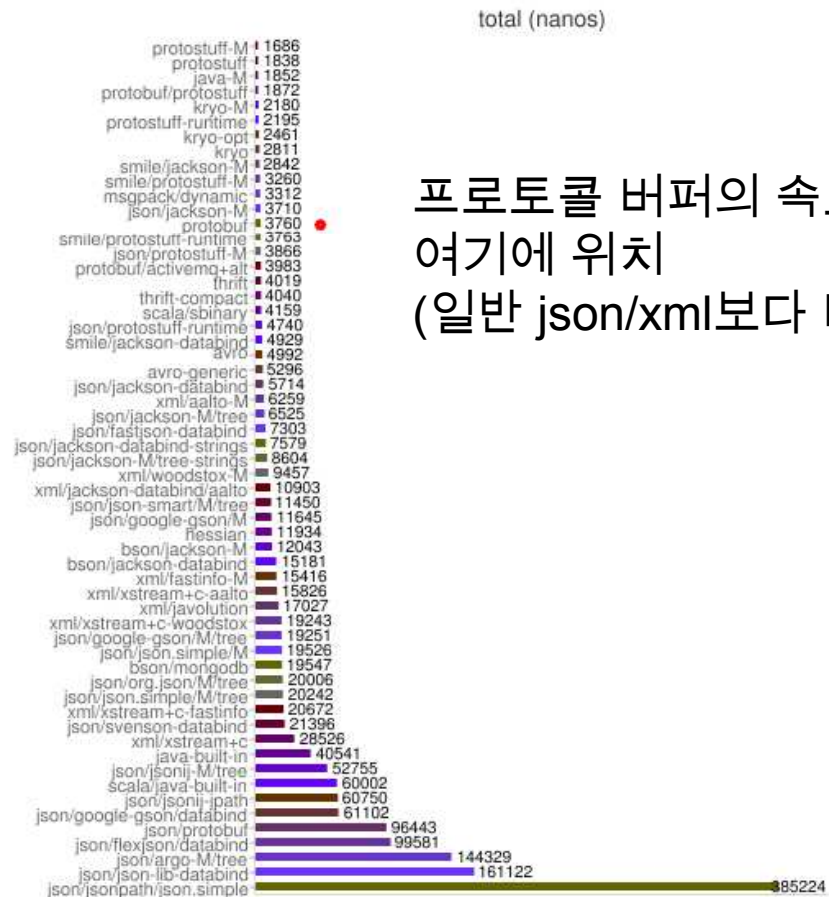
--- compressed section:
00000010 -- -- -- -- 78 9c e3 92 e2 b8 70 eb da 0c 7b ||.q.xx....p...{|
00000020 81 0b 7b 7a ff 39 49 34 3c 5c bb bd 9f 59 a1 61 |..{z.9l4<@...Y.a|
00000030 ce a2 df 5d cc 4a 7c fe c5 b9 c1 c9 19 a9 b9 89 |...].J|.....|
00000040 ba 61 06 7a 66 4a 5c 2e a9 79 c5 a9 7e f9 29 a9 |.a.zfJ#.y...|.|
00000050 c5 4d 8c fc c1 7e 8e 01 c1 1e fe 21 ba 45 46 26 |.M...~.....!EF&|
00000060 96 16 26 5d 8c 2a 19 25 25 05 56 fa fa e5 e5 e5 |..&].*.%V.....|
00000070 7a f9 05 40 a5 25 45 a9 a9 25 b9 89 05 7a f9 45 |z..@.xE...z.E|
00000080 e9 fa 89 05 99 fa 40 43 00 c0 94 29 0c

--- decompressed --->
00000000 0a - S 1 'type'
00000000 -- 1a - length 26 bytes
00000000 -- -- 08 d0 da d6 98 3f 10 d0 bc 8d fe 42 18 80
00000010 e1 ad b7 8f 03 20 80 9c a2 fb 8a 03 - BBOX (4*Varint)
00000010 -- -- -- -- -- -- -- -- -- -- 22 - S 4 'required_features'
00000010 -- -- -- -- -- -- -- -- -- -- 0e - length 14 bytes
00000010 -- -- -- -- -- -- -- -- -- -- 4f 73
00000020 6d 53 63 68 65 6d 61 2d 56 30 2e 36 - "OsmSchema-V0.6"
00000020 -- -- -- -- -- -- -- -- -- -- 22 - S 4 'required_features'
00000020 -- -- -- -- -- -- -- -- -- -- 0a - length 10 bytes
00000020 -- -- -- -- -- -- -- -- -- -- 44 65
00000030 6e 73 65 4e 6f 64 65 73 - "DenseNodes"
00000030 -- -- -- -- -- -- -- -- -- -- 82 01 - S 16 'writingprogram'
00000030 -- -- -- -- -- -- -- -- -- -- 0f - length 15 bytes
00000030 -- -- -- -- -- -- -- -- -- -- 53 4e 41 50 53
00000040 48 4f 54 2d 72 32 34 39 38 34 - "SNAPSHOT-r24984"
00000040 -- -- -- -- -- -- -- -- -- -- 8a 01 - S 17 'source'
00000040 -- -- -- -- -- -- -- -- -- -- 24 - length 36 bytes
```

다른 솔루션과 속도 비교

- <https://github.com/eishay/jvm-serializers/wiki>

The screenshot shows the GitHub Wiki page for the repository 'eishay/jvm-serializers'. The page title is 'Home' and it includes a 'Page History' button. A warning message states: 'WARNING: Benchmarks can be misleading'. Below this, there are several bullet points explaining the limitations of the benchmarks, such as using specific data values and different hardware/software environments. A 'Setup' section provides details on the hardware (Intel i7 2600k), software (Sun JRE 1.6.0_26), and JVM options used for testing. It also mentions the version of the benchmarking code and the methodology, which involves warming up the test and running it 500 times to get the best result.



프로토콜 버퍼의 속도는 여기에 위치 (일반 json/xml보다 빠름)

End of Document