Protobuf

What？

Google

独立于平台

独立于预言

可扩展的

序列化

结构化数据

的机制

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.

 the protocol buffer format supports the idea of extending the format over time in such a way that the code can still read data encoded with the old format

## Why Use Protocol Buffers

Pickling

but it doesn't deal well with schema evolution, and also doesn't work very well if you need to share data with applications written in C++ or Java.

invent

require writing one-off encoding and parsing code, and the parsing imposes a small run-time cost. This works best for encoding very simple data.

XML

However, XML is notoriously space intensive, and encoding/decoding it can impose a huge performance penalty on applications. Also, navigating an XML DOM tree is considerably more complicated than navigating simple fields in a class normally would be

as an optimization you can decide to use those tags for the commonly used or repeated elements, leaving tags 16 and higher for less-commonly used optional elements

和marshal之间的联系，为啥和客户的通信部直接用marshal，可以把他的效率和marshal对比，考虑对比下几种序列化工具的效率marshal c\_marshal pickl protbuf

和rpc之间的联系，是否可以作为固定存储

主要是学习protbuf是如何设计，已经有使用duonet的经验，看看人家是怎样设计的，本身不是一个特别复杂的东西，主要是学设计的，我们自己设计能否设计到这么好了

由于是跨语言的，那么其实C层也是可以使用的，python层也是可以使用的

尽可能了解全和深，然后你了解了很多东西，怎么取舍，怎么讲的明白

XML文档被看作是文档的数据库化和数据的文档化

# Development Warning

The Python implementation of Protocol Buffers is not as mature as the C++ and Java implementations. It may be more buggy, and it is known to be pretty slow at this time. If you would like to help fix these issues, join the Protocol Buffers discussion list and let us know!

<http://www.jianshu.com/p/0c563b2c0fdb> win下如何使用的方法

<https://developers.google.com/protocol-buffers/docs/reference/python-generated> 如何自动生成代码

**修饰符**

Required：没有就报错 慎用可能不兼容 Some engineers at Google have come to the conclusion that using **required** does more harm than good; they prefer to use only **optional** and **repeated**. However, this view is not universal.

Optional：可以指定[default = HOME] 不指定就根据类型，

Repeated：类似动态数组

**Compiling**

protoc -I=$SRC\_DIR --python\_out=$DST\_DIR $SRC\_DIR/addressbook.proto ：proto路径、代码路径

..\protobuf-3.3.0\src\protoc.exe -I=.\ --python\_out=.\ addressbook.proto

**The Protocol Buffer API**

The important line in each class is \_\_metaclass\_\_

the GeneratedProtocolMessageType metaclass uses the specified descriptors to create all the Python methods you need to work with each message type and adds them to the relevant classes.

not just adding arbitrary new fields to a generic Python object

person.no\_such\_field = 1 # raises AttributeError

person.id = "1234" # raises TypeError

**Enums**

Enums are expanded by the metaclass into a set of symbolic constants with integer values

**Standard Message Methods**

Each message class also contains a number of other methods that let you check or manipulate the entire message, including:

IsInitialized(): checks if all the required fields have been set.

\_\_str\_\_(): returns a human-readable representation of the message, particularly useful for debugging. (Usually invoked as str(message) or print message.)

CopyFrom(other\_msg): overwrites the message with the given message's values.

Clear(): clears all the elements back to the empty state.

**Parsing and Serialization**

hjc另外研究的几个

可以从这篇开始，这一篇比较全，里面还有一些外部连接，也可以看看

<http://mikewang.blog.51cto.com/3826268/1432136/>

这个是官方文档

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

这个是源代码地址

<https://github.com/google/protobuf/>

其他一些链接

文档翻译？

http://blog.csdn.net/menuconfig/article/details/12837173

http://andrewliu.in/2016/06/05/Google-protobuf-C-学习笔记/?hmsr=toutiao.io&utm\_medium=toutiao.io&utm\_source=toutiao.io

http://swiftcafe.io/2017/02/26/protobuffer/?hmsr=toutiao.io&utm\_medium=toutiao.io&utm\_source=toutiao.io

<http://www.oschina.net/p/pyrobuf?fromerr=5bXFrvBj>

<https://www.ibm.com/developerworks/cn/linux/l-cn-gpb/>