

数据交换格式

作者：少林之巅

目录

1. 数据交换格式简介
2. JSON数据格式
3. XML数据格式
4. MSGPack数据格式
5. Protobuf数据格式

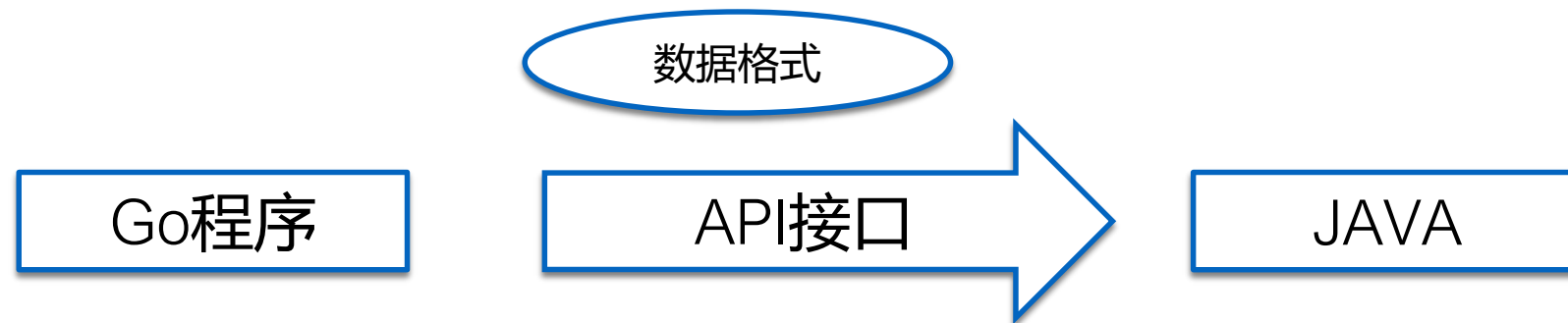
数据交换格式简介

1. 简介

A. 分布式系统

B. 打包和解包操作

C. 传输模式：1. 网络传输，API接口。2. 文件传输。



JSON数据格式

2. JSON数据格式

a. 对象, key-value形式。 {}

b. 数组, []

```
{
  "dates": {
    "date": [
      {
        "id": "1",
        "name": "JSON",
        "abb": "JavaScript Object Notation"
      },
      {
        "id": "2",
        "name": "XML",
        "abb": "eXtensible Markup Language"
      },
      {
        "id": "3",
        "name": "YAML",
        "abb": "Yet Another Markup Language"
      }
    ]
  }
}
```

JSON数据格式

3. 实例代码

A. Marshal=>序列化

B. UnMarshal=>反序列化

C. 应用场景：Api接口

XML数据交换格式

4. 简介

A. 可扩展标记语言。

B. 声明、根标签、子元素、属性

```
<?xml version="1.0" encoding="UTF-8" ?>
<dates>
  <date>
    <id>1</id>
    <name>JSON</name>
    <abb>JavaScript Object Notation</abb>
  </date>
  <date>
    <id>2</id>
    <name>XML</name>
    <abb>eXtensible Markup Language</abb>
  </date>
  <date>
    <id>3</id>
    <name>YAML</name>
    <abb>Yet Another Markup Language</abb>
  </date>
</dates>
```

XML数据交换格式

5. 实例代码

A. Marshal=>序列化

B. UnMarshal=>反序列化

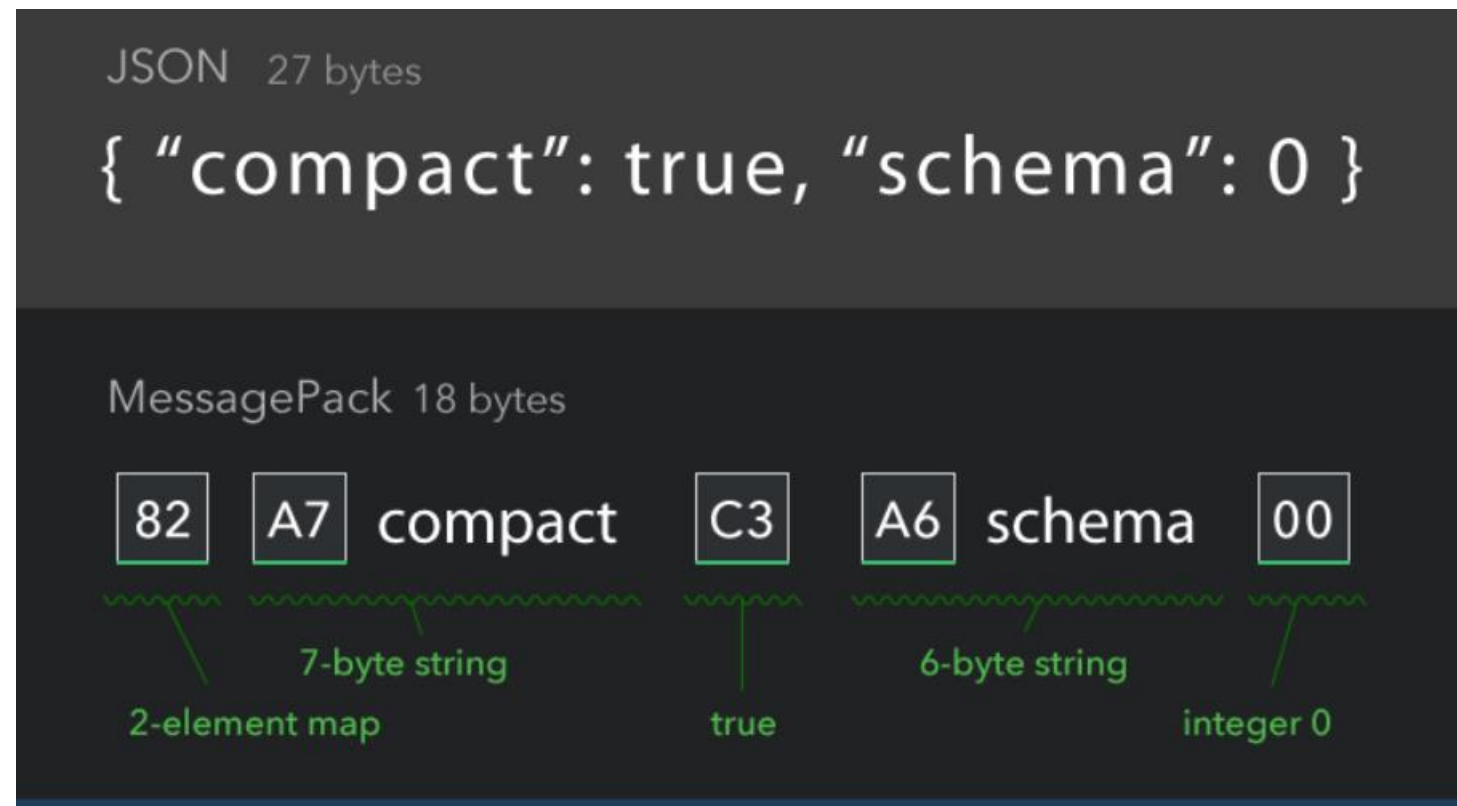
C. 应用场景：配置文件以及webservice， 比如soap协议

msgpack数据交换格式

6. 二进制的json协议

A. 性能更快。

B. 更省空间。



msgpack数据交换格式

7. 应用实战

A. Marshal=>序列化

B. UnMarshal=>反序列化

C. 应用场景：Api通讯

8. Protobuf协议简介

- A. Google推出的数据交换格式。
- B. 二进制的。
- C. 基于代码自动生成。

protobuf数据交换格式

9. Protobuf开发流程

A. IDL编写

B. 生成只定语言的代码

C. 序列化和反序列化

protobuf数据交换格式

10.IDL编写

.proto Type	Notes	C++ Type	Java Type	Python Type[2]	Go Type
double		double	double	float	float64
float		float	float	float	float32
int32	使用变长编码，对于负值的效率很低，如果你的域有可能有负值，请使用sint64替代	int32	int	int	int32
uint32	使用变长编码	uint32	int	int/long	uint32
uint64	使用变长编码	uint64	long	int/long	uint64
sint32	使用变长编码，这些编码在负值时比int32高效的多	int32	int	int	int32
sint64	使用变长编码，有符号的整型值。编码时比通常的int64高效。	int64	long	int/long	int64
fixed32	总是4个字节，如果数值总是比总是比228大的话，这个类型会比uint32高效。	uint32	int	int	uint32
fixed64	总是8个字节，如果数值总是比总是比256大的话，这个类型会比uint64高效。	uint64	long	int/long	uint64
sfixed32	总是4个字节	int32	int	int	int32
sfixed64	总是8个字节	int64	long	int/long	int64
bool		bool	boolean	bool	bool
string	一个字符串必须是UTF-8编码或者7-bit ASCII编码的文本。	string	String	str/unicode	string
bytes	可能包含任意顺序的字节数据。	string	ByteString	str	[]byte

protobuf数据交换格式

11.IDL编写

A. 枚举类型

```
enum EnumAllowingAlias {  
    UNKNOWN = 0;  
    STARTED = 1;  
    RUNNING = 2;  
}
```

12.IDL编写

A. 结构体类型

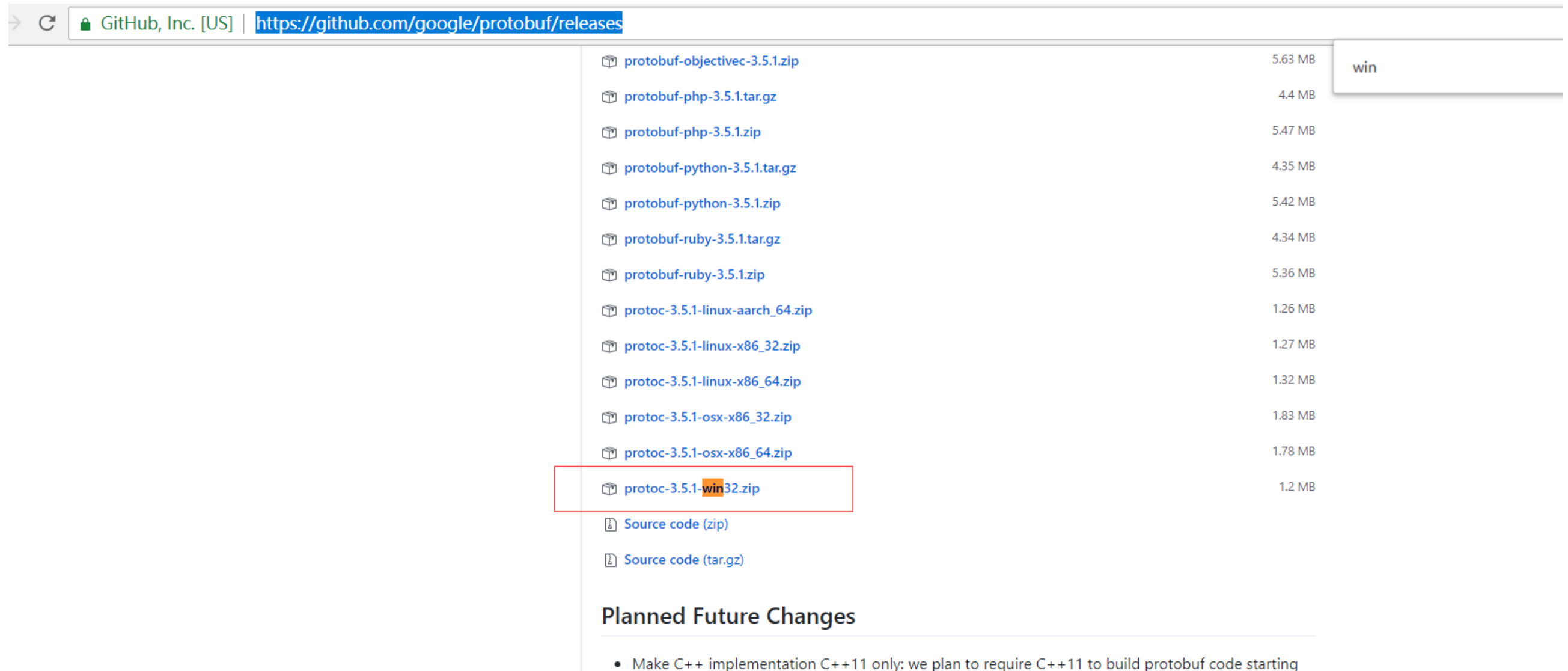
```
message Person {  
    //后面的数字表示标识号  
    int32 id = 1;  
    string name = 2;  
    //repeated表示可重复  
    //可以有多个手机  
    repeated Phone phones = 3;  
}
```

protobuf数据交换格式

13. Golang应用

A. 安装protoc编译器，解压后拷贝到GOPATH/bin目录下

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



Asset Name	Size
protobuf-objectivec-3.5.1.zip	5.63 MB
protobuf-php-3.5.1.tar.gz	4.4 MB
protobuf-php-3.5.1.zip	5.47 MB
protobuf-python-3.5.1.tar.gz	4.35 MB
protobuf-python-3.5.1.zip	5.42 MB
protobuf-ruby-3.5.1.tar.gz	4.34 MB
protobuf-ruby-3.5.1.zip	5.36 MB
protoc-3.5.1-linux-aarch_64.zip	1.26 MB
protoc-3.5.1-linux-x86_32.zip	1.27 MB
protoc-3.5.1-linux-x86_64.zip	1.32 MB
protoc-3.5.1-osx-x86_32.zip	1.83 MB
protoc-3.5.1-osx-x86_64.zip	1.78 MB
protoc-3.5.1-win32.zip	1.2 MB
Source code (zip)	
Source code (tar.gz)	

Planned Future Changes

- Make C++ implementation C++11 only: we plan to require C++11 to build protobuf code starting

protobuf数据交换格式

13. Golang应用

C. 安装golang代码插件, `go get -u github.com/golang/protobuf/protoc-gen-go`

protobuf数据交换格式

14. Golang实战

A. 生成代码, protoc --go_out=. *.proto