ProtoBuf

Dongdong Deng <LibFetion@gmail.com>

OverView

- 1、简介
- 2、示例
- 3、.proto语法

1、ProtoBuf

Google开源的，用于序列结构化数据的协议

http://code.google.com/apis/protocolbuffers

- 跨编程语言
- 跨系统平台
- 可扩展

1、ProtoBuf vs XML

- 简单
- 比XML小3-10倍
- 比XML快20-100倍
- 避免歧义
- 自动产生数据操作类，让开发者更容易使用

1、ProtoBuf 缺点

- 目前使用不够广泛

ProtoBuf目前没有广泛使用，如果系统需要提供若干对外的接口给第三方系统调用，XML目前是更好的选择

- 二进制格式导致可读性差

二进制格式进行编码，通信协议出现问题不好跟踪

- 缺乏自描述

如果不配合相应的proto文件，ProtoBuf基本是看不懂，在配置文件方面,XML还是很有优势的

2、ProtoBuf 示例

1: Write .proto 文件

2: protoc --cpp_out --java_out --python_out

3: *.pb.h *.pb.cc

4: using above *.pb.h in your program

2、.proto

```
message Test
{
  required int32 id = 1;
  required string pwd = 2;
  optional string others = 3;
}
```

2、protoc

```
protoc --cpp_out=./test test.proto

$ls ./test
test.pb.h test.pb.cc
```

2、progam-产生方

产生方:

```
Test test;
test.set_id(344300);
test.set_pwd(123);
test.set_others("othersothers");

string sTest;
test.SerailzeToString(&sTest);
```

2、progam-读取方

读取方:

```
string sTest; //得到协议数据，存放到sTest
Test test;
if(test.ParseFromString(sTest))
{  cout << "ID:" << test.id() << endl
        << "PWD:" << test.pwd() << endl
        << "others:" << test.others() << endl;
} else {
  cerr << "parse error!" << endl;
}
```

3、proto 语法

```
package tutorial;

message Person {
  required string name = 1;
  required int32 id = 2;        // Unique ID number for this person.
  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;
}

// Our address book file is just one of these.
message AddressBook {
  repeated Person person = 1;
}
```