iOS中Protocol Buffer的集成和使用

一、protocol buffer介绍

protocol buffer全称 Google Protocol Buffer，是 Google 公司内部的混合语言数据标准，目前已经正在使用的有超过 48,162 种报文格式定义和超过 12,183 个 .proto 文件。他们用于 RPC 系统和持续数据存储系统。

Protocol Buffers 是一种轻便高效的结构化数据存储格式，**可以用于结构化数据串行化，或者说序列化，它很适合做数据存储或 RPC 数据交换格式**。可用于通讯协议、数据存储等领域的语言无关、平台无关、可扩展的序列化结构数据格式。目前官方提供了 C++、Java、Python 三种语言的 API,也可以支持其他语言，实现它的派生方法即可，目前支持objc的版本已有(下文会讲到)。

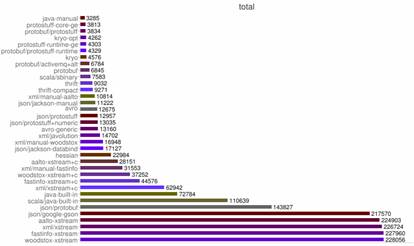
二、ios中protocol buffer 的应用

那么在ios中我们是如何应用的呢？举个例子：客户端与服务器通信之间进行IM通信的时候，先通过socket与服务器建立长连接，服务器与客户端之间发送、接受的原始数据都是二进制的，我们需要拿到这个数据进行解包，就是我们所说说的反序列化，或者将上层model进行封包（序列化），传统我们都是使用json完成，拿到一个json数据之后我们还需要对json数据进行数据解析以此得到我们想要的model，此时我们的protocolbuf就能完成json序列化和数据解析两个过程，这2步被它完美的合二为一了。

那么我们为什么要使用protocol buffer呢，它有什么优点呢？

简单说来 **Protobuf 的主要优点就是：简单，快**。有人做了一个时间测试，包括创建对象，将对象序列化为内存中的字节序列，然后再反序列化的整个过程，从下图中我们可以看到protocol buffer成绩很好。

图 1. 性能测试结果



三、protocol buffer(简称pb)集成

1.安装homebrew

集成pb需要先安装[homebrew](https://brew.sh/index_zh-cn.html), **homebrew是**macOS 缺失的软件包管理器，我们需要通过brew安装protobuf.下面我们打开终端，依次输入的命令如下:

1.1先检查有没有安装brew：

**wupengdeMacBook-Pro:~ wupeng$ brew -v**

1.2如果没有安装，采用如下命令安装：

**wupengdeMacBook-Pro:~ wupeng$ ruby -e "$(curl -fsSL**[**https://raw.githubusercontent.com/Homebrew/install/master/install**](https://link.jianshu.com/?t=https%3A%2F%2Fraw.githubusercontent.com%2FHomebrew%2Finstall%2Fmaster%2Finstall)**) "**

1.3安装brew需要的工具

**wupengdeMacBook-Pro:~ wupeng$ brew install automake**

**wupengdeMacBook-Pro:~ wupeng$ brew install libtool**

**wupengdeMacBook-Pro:~ wupeng$ brew install protobuf**

至此，protobuf已经安装到我们的电脑中了，刚才说了，pb目前官方支持的api语言只有C++、Java、Python，想要支持objc就需要安装派生插件。

2.安装pb的oc插件

到[Releases·google/protobuf](https://link.jianshu.com/?t=https%3A%2F%2Fgithub.com%2Fgoogle%2Fprotobuf%2Freleases" \t "_blank)下载最新版压缩包  
[protobuf-objectivec-3.2.0.tar.gz](https://link.jianshu.com/?t=https%3A%2F%2Fgithub.com%2Fgoogle%2Fprotobuf%2Freleases%2Fdownload%2Fv3.2.0%2Fprotobuf-objectivec-3.2.0.tar.gz)或者[protobuf-objectivec-3.2.0.zip](https://link.jianshu.com/?t=https%3A%2F%2Fgithub.com%2Fgoogle%2Fprotobuf%2Freleases%2Fdownload%2Fv3.2.0%2Fprotobuf-objectivec-3.2.0.zip" \t "_blank)  
注：这里需要翻墙，可以使用[Lantern](https://link.jianshu.com/?t=https%3A%2F%2Fwww.getlantern.org%2F" \t "_blank)

下载之后解压，然后终端进入到解压完的文件夹下，命令如下：

**wupengdeMacBook-Pro:~ wupeng$ cd /Users/wupeng/Downloads/protobuf-3.2.0**

**wupengdeMacBook-Pro:~ wupeng$ ./configure**

**wupengdeMacBook-Pro:~ wupeng$ make (这个过程会需要几分钟)**

**wupengdeMacBook-Pro:~ wupeng$ make check (这个过程会需要几分钟)**

**wupengdeMacBook-Pro:~ wupeng$ sudo make install (这个过程会需要几分钟)**

至此，我们的oc插件就装好了，就可以通过命令对.proto源文件（.proto文件下文会介绍）进行编译转换成我们想要的.h和.m文件了。

3.pods集成

到这里相信各位的cocoapods环境已经安装了，附[cocoaspods安装教程](https://www.jianshu.com/p/9e4e36ba8574)。

Podfile文件写入：

platform :ios, '7.0'

target 'haha' do

pod 'Protobuf', '~> 3.6.0'

end

说一个注意点，刚开始我是pod 'ProtocolBuffers', '~> 1.9.11' ，编译时各种报错，后来才发现是少了runtime相关的支持代码，说明pod错了，应该pod 'Protobuf', '~> 3.6.0'

**wupengdeMacBook-Pro:protoTest wupeng$ pod install**

至此，pb就完美的集成到我们的项目当中了。

四．Pb的使用

4.1编写proto文件

前面我们说到了一个.proto源文件，使用pb必须要有.proto文件。proto 文件是**用来定义我们程序中需要处理的结构化数据**，在 protobuf 的术语中，结构化数据被称为 Message。proto 文件非常类似 java 或者 C 语言的数据定义，在项目中我们可以理解为model的前身，用来定义model的格式，它决定model里有哪些属性。proto文件一般由后端人员编写，前端也可以按照[proto官方标准语法](https://developers.google.com/protocol-buffers/)编码。

4.2 proto源文件转换

终端输入以下命令：

**wupengdeMacBook-Pro:protobuf-3.2.0 wupeng$ cd /Users/wupeng/Desktop/protoTest (定位到.proto源文件目录)**

**wupengdeMacBook-Pro:~ wupeng$ protoc --objc\_out=./ Person.proto**  
注:./是生成文件的目的地址  Person.proto是proto源文件

或者

**wupengdeMacBook-Pro:~ wupeng$ protoc Person.proto --objc\_out=./**

**或者**

**wupengdeMacBook-Pro:~ wupeng$ protoc --plugin=/usr/local/bin/protoc-gen-objc Person.proto --objc\_out="./"**

执行以上操作之后，你的目标文件夹下就会出现以下两个文件了：

Person.pbobjc.h和Person.pbobjc.m

这两个文件就是我们需要使用的model类了，拖入项目中即可。需要注意的是，生成的这两个文件是mrc的，arc环境下我们需要对这两文件做混编处理：**需要在\*.pbobjc.m设置-fno-objc-arc。**

4.3 利用pb序列化、反序列化的代码实现

前面我们讲了关于pb整个集成过程，那么集成到项目中之后，如何调用呢？

涉及到公司后台协议属保密产品，我们还是用person.proto做测试。

[测试Demo地址](https://github.com/FuWees/protocolbufDemo)

参考文章：

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

<http://www.cnblogs.com/tandaxia/p/6181534.html>

<https://www.jianshu.com/p/751aa2b621d5>