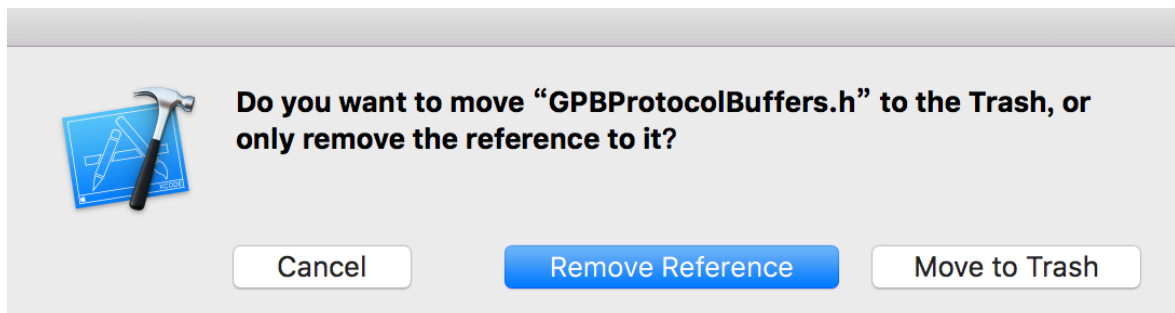


protobuf 配置说明:

1. 下载protobuf第三方库,在github上下载; 地址: <https://github.com/google/protobuf> , 记得下载objectivec这个库;

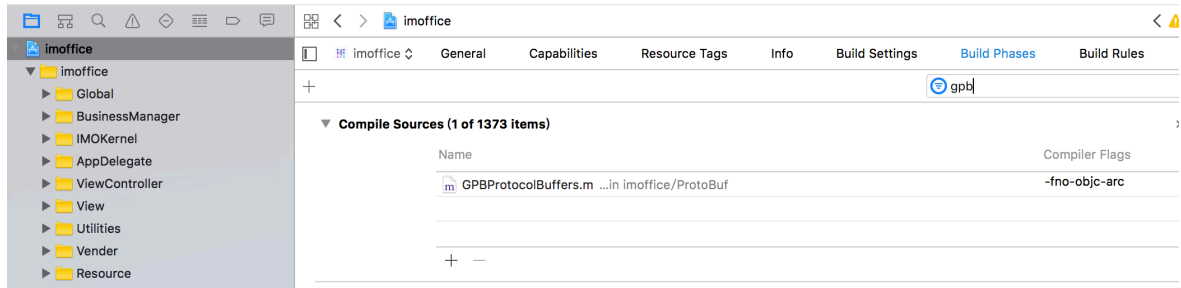
jenkins	Restore jenkins files.
js	Added back in binary serialization round
m4	Merge pull request #789 from motahan,
more_tests	Add makefile for extended tests to be r
objectivec	Update README.md
php	Rename Empty to GPBEmpty in php ger
protoc-artifacts	Update version number.

2. 导入protobuf库到工程中,将库中除了GPBProtocolBuffers.m这个之外的所有.m文件都删除掉,是从目录中删除,不要从工程中删除,如下图中点击蓝色按钮(记得是所有.m文件,包括google文件夹里的);

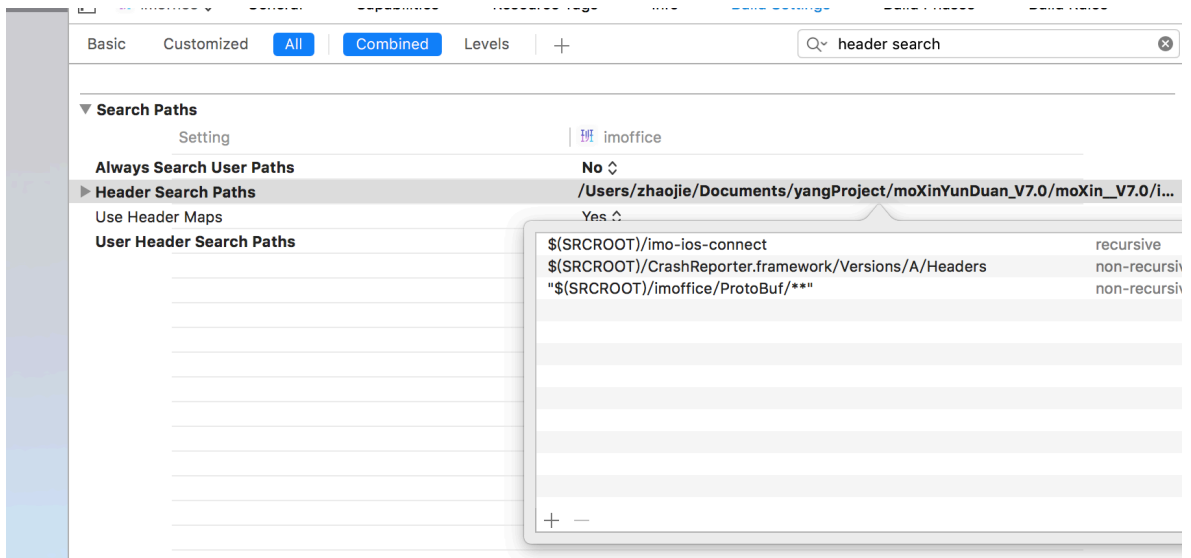




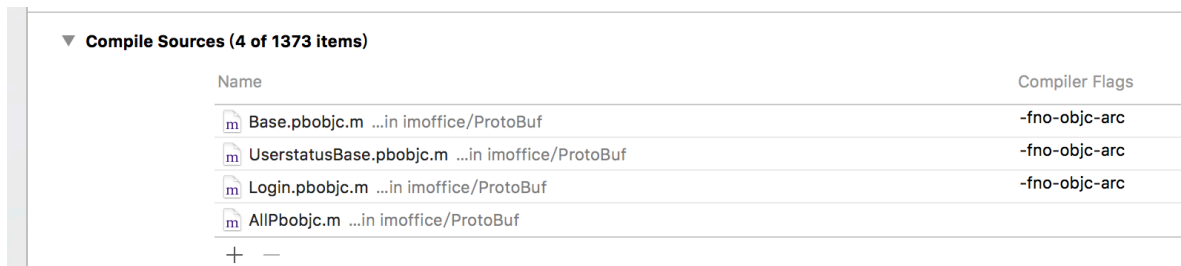
3. 在工程的build phases里,将 compile source下的GPBProtocolBuffers.m 添加 -fno-objc-arc;转为支持ARC;













4. 将导入工程的protobuf位置添加到头文件搜索中; 在工程的build setting里搜索header search paths ,记得后面要加 `/**` ,否则会报错,找不到下面的文件;



5. 生成的pbobjc.m文件都要转换成支持ARC;



6. 在地址 <https://github.com/google/protobuf/releases> 中下载 protoc-3.1.0-osx-x86_64 库

 [protobuf-php-3.1.0.zip](#)
 [protobuf-python-3.1.0.tar.gz](#)
 [protobuf-python-3.1.0.zip](#)
 [protobuf-ruby-3.1.0.tar.gz](#)
 [protobuf-ruby-3.1.0.zip](#)
 [protoc-3.1.0-linux-x86_32.zip](#)
 [protoc-3.1.0-linux-x86_64.zip](#)
 [protoc-3.1.0-osx-x86_32.zip](#)
 [protoc-3.1.0-osx-x86_64.zip](#)
 [protoc-3.1.0-win32.zip](#)

7. 配置protobuf编译器,在终端中输入 `export PATH=/Users/machaojie/Library/protoc-3.1.0-osx-x86_64/bin:$PATH` (export PATH= 红色部分是protoc-3.1.0-osx-x86_64文件地址, 是你库在电脑中的位置);

8. 编译器装好后,在终端就可以生成相应的累啦; 输入 `protoc --help` 可以看到相应的输出命令, 例如 要生成 objc 需要的类; 首先在终端切换到你的.proto文件目录下, 然后执行 `protoc --objc_out=. login.proto` 命令, 就会生成.h .m 文件啦;

	PROTO_FILES should be given when using this flag.
-oFILE,	Writes a FileDescriptorSet (a protocol buffer, defined in descriptor.proto) containing all of the input files to FILE.
--descriptor_set_out=FILE	
--include_imports	When using --descriptor_set_out, also include all dependencies of the input files in the set, so that the set is self-contained.
--include_source_info	When using --descriptor_set_out, do not strip SourceCodeInfo from the FileDescriptorProto. This results in vastly larger descriptors that include information about the original location of each decl in the source file as well as surrounding comments.
--dependency_out=FILE	Write a dependency output file in the format expected by make. This writes the transitive set of input file paths to FILE
--error_format=FORMAT	Set the format in which to print errors. FORMAT may be 'gcc' (the default) or 'msvs' (Microsoft Visual Studio format).
--print_free_field_numbers	Print the free field numbers of the messages defined in the given proto files. Groups share the same field number space with the parent message. Extension ranges are counted as occupied fields numbers.
--plugin=EXECUTABLE	Specifies a plugin executable to use. Normally, protoc searches the PATH for plugins, but you may specify additional executables not in the path using this flag. Additionally, EXECUTABLE may be of the form NAME=PATH, in which case the given plugin name is mapped to the given executable even if the executable's own name differs.
--cpp_out=OUT_DIR	Generate C++ header and source.
--csharp_out=OUT_DIR	Generate C# source file.
--java_out=OUT_DIR	Generate Java source file.
--javanano_out=OUT_DIR	Generate Java Nano source file.
--js_out=OUT_DIR	Generate JavaScript source.
--objc_out=OUT_DIR	Generate Objective C header and source.
--php_out=OUT_DIR	Generate PHP source file.
--python_out=OUT_DIR	Generate Python source file.
--ruby_out=OUT_DIR	Generate Ruby source file.

<http://www.2cto.com/kf/201503/382440.html> 或者 <http://www.jianshu.com/p/d5642a7d1e10>

oc安装protobuf 将里面克隆地址替换: "https://github.com/google/protobuf.git"

<http://www.jianshu.com/p/751aa2b621d5> swift安装protobuf