HƯỚNG DẪN SỬ DỤNG
gRPC VERSION 1.0.3 TRÊN JAVA

Date	Change log	Author	Approver
05/04/2017	Khởi tạo tài liệu	DanhCC	TienHT

STT	Tài liệu đính kèm	Ghi chú
1	Code mẫu project Server	gRPCService.zip
2	Code mẫu project stub	gRPCStub.zip

"Lập trình và phụ nữ đều giống nhau ở chỗ càng tìm hiểu càng thấy khó hiểu"

Huỳnh Trọng Tiến

I.Build Protobuf Compiler (protoc)

1. Nguồn tải

Để có protoc ta nên build trực tiếp từ opensource. Ngoài ra, cũng có thể tải từ internet nhưng như thế không an toàn vì đó có thể là protoc của phiên bản trước, không phù hợp với phiên bản hiện tại.

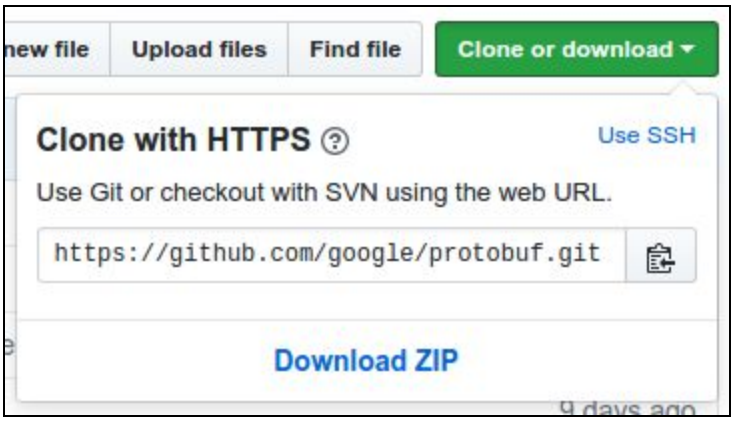
Đi từ trang chủ của Protobuf để tìm hướng dẫn tải source Protobuf:

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

Hiện tại, truy cập vào link trên sẽ được hướng dẫn đến link source Protobuf trên GitHub:

https://github.com/google/protobuf

Chọn Clone or Download để tải source zip.



Hình 1: Tải source Protobuf

Tương tự, trên trang hướng dẫn, ta tìm được link đến Maven để tải các file .jar:

https://mvnrepository.com/artifact/com.google.protobuf/protobuf-java

Lưu ý: để sử dụng gRPC thì version của protoc phải từ 3.0 trở lên do đó cần kiểm tra version protoc trước. Giả sử sau khi đã cài thành công protobuf, kiểm tra version của protoc bằng lệnh sau:

Command	protoc --version
Output	libprotoc **3.2.0**

Như vậy, version của protoc là từ 3.0.0 trở lên, nên ta có thể dùng được gRPC.

2. Build protoc

Bước 1: Giải nén source zip của Protobuf vào thư mục bất kì. Ví dụ:

~/dev/protobuf-master

Bước 2: Lần lượt chạy các lệnh sau:

sudo apt-get install autoconf automake libtool curl make g++ unzip
cd ~/dev/protobuf-master
./autogen.sh
./configure
make
make check
sudo make install
sudo ldconfig

Bước 3: Kiểm tra kết quả và version của protoc:

protoc --version

"Chuẩn bị và ngăn chặn, đừng sửa chữa và hối tiếc"

Khuyết danh

II.Chuẩn bị source và library

1. Download library và tool gen code Java

Đi từ trang chủ của gRPC để xem hướng dẫn tải library và plugin gen code Java: http://www.grpc.io/

Hiện tại, theo hướng dẫn, ta sẽ được dẫn đến trang tải library và plugin gen code Java: Link tải file .jar và tool gen code Java

Lưu ý:

+ Hướng dẫn này được viết cho **gRPC version 1.0.3**. Do đó, ta tải các file .jar và tool gen code Java cho version 1.0.3. Ngoài các file jar ở trên, ta còn cần dùng *guava*, *netty-all* và *netty-tcnative-boringssl-static* (nếu cần dùng SSL). **Đối với các phiên bản mới hơn của gRPC có thể yêu cầu thêm các jar mới**. Ví dụ: ở version 1.2.0 cần thêm instrumentation-api-0.3.0, error_prone_annotations-2.0.19, guava-20

+ Plugin download về sẽ có tên **protoc-gen-grpc-java-1.0.3-linux-x86_64.exe**, ta nên đổi tên thành **plugin-grpc-java** để tiện dùng.

2. Chuẩn bị thư mục build:

Giả sử thư mục build là **~/protobuf**.

Bước	Action
1	Tạo 3 thư mục đồng cấp: ~/protobuf/idl, ~/protobuf/java_out, ~/protobuf/tool
2	Copy plugin gRPC (**plugin-grpc-java**) vào thư mục tool
3	Set quyền plugin: **chmod 755 ~/protobuf/tool/plugin-grpc-java**

3. Build protobuf và gRPC từ file .proto

Giả sử ta có file HelloWorld.proto với nội dung như sau: định nghĩa một service có tên Greeter, sở hữu phương thức sayHello. Service này nhận message HelloRequest từ client, sau khi xử lý sẽ response trở lại bằng message HelloReply. Nhiệm vụ của chúng ta chỉ duy nhất là phải override phương thức sayHello của server. Còn các vấn đề khác gRPC đã có sẵn, chỉ việc dùng lại.

```
syntax = "proto3";
option java_multiple_files = true;
option java_package = "io.grpc.examples.helloworld";
option java_outer_classname = "HelloWorldProto";

package helloworld;

// The greeting service definition.
service Greeter {
  // Sends a greeting
  rpc SayHello (HelloRequest) returns (HelloReply) {}
}

// The request message containing the user's name.
message HelloRequest {
  string name = 1;
}

// The response message containing the greetings
message HelloReply {
  string message = 1;
}
```

Copy file helloworld.proto vào thư mục **~/protobuf/idl** và tiến hành build bằng các lệnh sau:

```
cd ~/protobuf
```

```
protoc -I=idl --java_out=java_out idl/helloworld.proto
```

```
protoc -I=idl
--plugin=protoc-gen-grpc-java=~/protobuf/tool/plugin-grpc-java
```

```
--grpc-java_out=java_out idl/helloworld.proto
```

Sau khi build, ta được các file sau:

+ GreeterGrpc.java
+ HelloReply.java
+ HelloReplyOrBuilder.java
+ HelloRequest.java
+ HelloRequestOrBuilder.java
+ HelloWorldProto.java

"Nếu anh không thể giải thích đơn giản thì anh chưa hiểu đủ rõ"

Albert Einstein

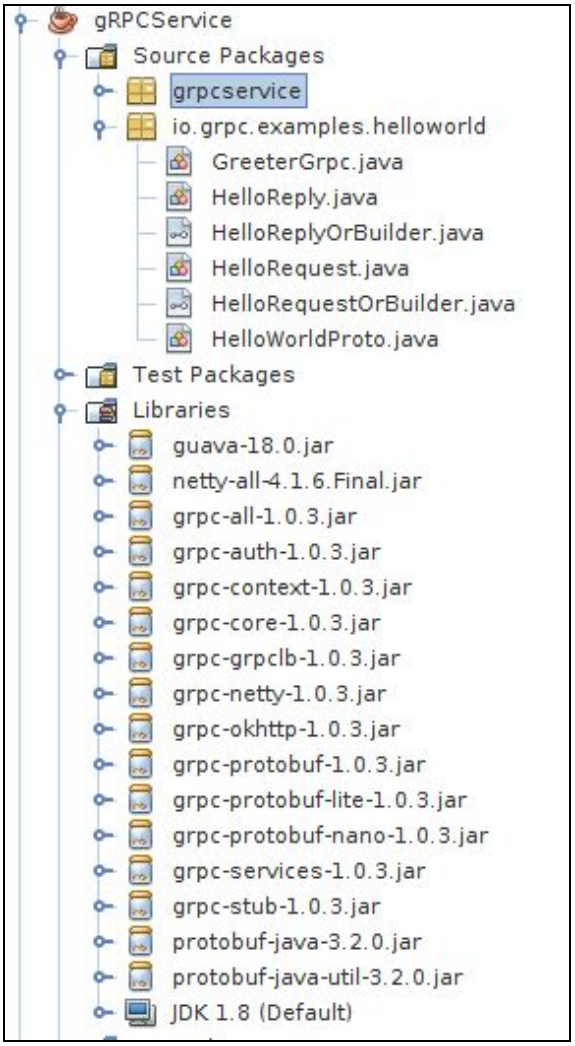
III.gRPC Server

1. Tạo project Server

+ Tạo một project Java trên Netbeans, giả sử đặt tên là **gRPCService**.

+ Import các file .java đã build được từ file .proto vào project. Lưu ý: thêm các file này vào đúng package như đã khai báo option **java_package** trong file proto.

+ Import các file .jar cần thiết của gRPC vào project. Nếu cần sử dụng kết nối SSL thì thêm netty-tcnative-boringssl-static.



2. Implement handler

Trong HelloWorld.proto ta có service là Greeter, implement các phương thức cần thiết để handle các rpc bằng cách extends class GreeterImpleBase và Override các phương thức.

Cụ thể, ở đây ta khai báo class GreeterGrpcHandler kế thừa từ GreeterImplBase. và tiến hành implement sayHello.

```
public class GreeterGrpcHandler extends GreeterGrpc.GreeterImplBase {

    @Override
    final public void sayHello(HelloRequest request,
            StreamObserver<HelloReply> responseObserver) {
        HelloReply reply = this.handleRequest(request);
        responseObserver.onNext(reply);
        responseObserver.onCompleted();
    }

    private HelloReply handleRequest(HelloRequest request) {
        HelloReply reply = HelloReply.newBuilder()
                .setMessage(request.getName() + " is processed")
                .build();
        return reply;
    }
}
```

3. Server

Sử dụng ServerBuilder để build và start service. Chỉ định rõ port listen và service mà server sẽ xử lý.

```
private Server server;
private final GreeterGrpcHandler greeterHandler = new GreeterGrpcHandler();

private void start(int port) throws IOException {
    server = ServerBuilder.forPort(port)
            .addService(greeterHandler)
            .build()
            .start();

    Runtime.getRuntime().addShutdownHook(new Thread() {
        @Override
        public void run() {
            ZServer.this.stop();
        }
    });
}
```

"Sự việc không thay đổi, chúng ta thay đổi"

Henry David Thoreau

IV.gRPC Stub

1. Tạo project

Tạo mới project và import các file .java và các file .jar tương tự như server. Giả sử đặt tên project là gRPCStub.

2. Implement

Tạo channel và stub để gọi các rpc của server. Chỉ định rõ host, port của server để kết nối. Mặc định gRPC sử dụng kết nối SSL, do đó nếu cần nhận dữ liệu thông thường thì thêm chỉ định usePlainText(true).

- Stub không sử dụng SSL

```
private final ManagedChannel channel;
private final GreeterGrpc.GreeterBlockingStub blockingStub;

public ZStub(String host, int port) throws Exception {
    channel = ManagedChannelBuilder
            .forAddress(host, port)
            .usePlaintext(true)
            .build();
    blockingStub = GreeterGrpc.newBlockingStub(channel);
}
```

- Stub sử dụng SSL

```
private final ManagedChannel channel;
private final GreeterGrpc.GreeterBlockingStub blockingStub;

public ZStub(String host, int port) throws Exception {
    channel = ManagedChannelBuilder
            .forAddress(host, port)
            .build();
    blockingStub = GreeterGrpc.newBlockingStub(channel);
}
```

3. Gọi RPC

Dùng stub đã tạo để gọi các rpc và nhận kết quả:

```
public HelloReply sayHello() {
    if (channel.isShutdown() || channel.isTerminated()) {
        System.err.println("Connection is closed. Reconnecting ...");
        return null;
    }

    HelloRequest request = HelloRequest.newBuilder()
            .setName("grpcStub").build();

    try {
        HelloReply response = blockingStub.sayHello(request);
        if (response == null) {
            System.err.println("get response null");
        }
        return response;
    } catch (StatusRuntimeException e) {
        System.err.println("RPC throw exception: " + e.getStatus());
    } catch (Exception ex) {
        System.err.println("process response string throw exception: "
                + ex.getMessage());
    }
    return null;
}
```