07. gRPC 실용사례 java

본 문서 에서는 java 에서 gRPC 를 사용하여 Client 프로그램을 작성 하는 방법을 소개 한다. java 에서 grpc 설치는 여기 의 `8. Java 에서 gRPC 사용하기` 를 참조 한다.

1. Interface 를 위한 .proto 파일

실용사례 에서 공통으로 사용할 .proto 파일로 이름은 `xdb_grpc.proto` 로 작성 하였다.
`01. gRPC 실용사례 CPP` 에 있는 내용과 동일 하다.

```
syntax = "proto3";

// Option for Java.
option java_multiple_files = true;
option java_package = "com.hancomsechre.xecuredb.grpc";
option java_outer_classname = "XecuredbProto";

// Option for Ovjective-C.
option objc_class_prefix = "XDB";

package xdbproto;

// The Xecuredb service definition.
service Xecuredb {
  rpc Encrypt (EncRequest) returns (EncReply) {}
  rpc Decrypt (DecRequest) returns (DecReply) {}
  rpc CryptoHash (HashRequest) returns (HashReply) {}
}

// The request message containing the alias and plain text.
message EncRequest {
  string alias = 1;
  string plain = 2;
}

// The response message containing the err code and cipher text.
message EncReply {
  int32 err = 1;
  string msg = 2;
  string cipher = 3;
}

// The request message containing the alias and cipher text.
message DecRequest {
  string alias = 1;
  string cipher = 2;
}

// The response message containing the err code and plain text.
message DecReply {
  int32 err = 1;
  string msg = 2;
```

```
  string plain = 3;
}

// The request message containing the alogorithm id and plain text.
message HashRequest {
  int32 id = 1;
  string plain = 2;
}

// The response message containing the err code and hash string.
message HashReply {
  int32 err = 1;
  string msg = 2;
  string cipher = 3;
}
```

2. 사전 준비

2.1 ptotoc 에서 grpc 모듈 생성을 위한 plugin 컴파일

protoc 에서 java 용 gRPC 모듈을 생성 하기 위해서는 plugin 을 별도로 컴파일 하여야 한다. C/CPP, PHP, PYTHON,RUBY,NODE 용 plugin 은 grpc 설치 시 함께 설치가 되지만 Java 용 plugin 은 별도의 gitgub 저장소에 의해 관리가 되고 있어서 별도의 컴파일이 필요 하다. 다음 명령을 수행 하여 plugin 을 컴파일 한다, 컴파일을 하기 위해서는 C 컴파일러가 설치되어 있어야 한다.

```
cd  ~/grpc-java/compiler

../gradlew java_pluginExecutable

cd ~/grpc-java/compiler/build/exe/java_plugin

sudo cp protoc-gen-grpc-java /usr/local/bin/

cd

which protoc-gen-grpc-java
```

2.2 java lib 파일 복사

java 에서 gRPC 사용시 참조되는 jar 파일 들을 현재 directory 로 복사 한다.

```
cp -r ~/grpc-java/examples/build/install/examples/lib .
```

3. protoc 를 이용한 Interface 파일 생성

java 에서 gRPC 를 사용하기 위해서는 protoc 를 이용하여 interface 파일을 생성 하여야 한다. 다음은 interface 파일을 생성 하기 위한 명령 이다. 작업을 수행 하기 전에 작업 현재 경로에 `lib` directory 를 만든 후 스크립트를 실행 하여야 한다. `create_java_src.sh`

```
# java
protoc -I ../proto --java_out=. xdb_grpc.proto
protoc -I ../proto --grpc-java_out=. --plugin=protoc-gen-grpc-java=`which protoc-gen-grpc-java` xdb_grpc.proto
```

위 명령을 실행 하면 작업 directory 에 `com/hancomsechre/xecuredb/grpc` directory 가 생성 된다. `com.hancomsechre.xecuredb.grpc` 는 `xdb_grpc.proto` 에서 정의한 `option java_package` 에 따라 다른 위치가 될 수 있다. 다음은 `com/hancomsechre/xecuredb/grpc` 에 있는 파일 목록 이다.

`./lib` 파일 목록

```
DecReply.java
DecReplyOrBuilder.java
DecRequest.java
DecRequestOrBuilder.java
EncReply.java
EncReplyOrBuilder.java
EncRequest.java
EncRequestOrBuilder.java
HashReply.java
HashReplyOrBuilder.java
HashRequest.java
HashRequestOrBuilder.java
XecuredbGrpc.java
XecuredbProto.java
```

4. Client 코드 작성

java 를 이용한 client 는 java_xdb_client.java 로 작성 하였으며 source code 는 다음과 같다.

```
import io.grpc.ManagedChannel;
import io.grpc.ManagedChannelBuilder;
import io.grpc.StatusRuntimeException;
import java.util.concurrent.TimeUnit;
import java.util.logging.Level;
import java.util.logging.Logger;

// protoc 에 의해 생성된 package
import com.hancomsechre.xecuredb.grpc.*;

public class java_xdb_client {
  private final ManagedChannel channel;
  private final XecuredbGrpc.XecuredbBlockingStub blockingStub;

  public java_xdb_client(String host, int port) {
    this(ManagedChannelBuilder.forAddress(host, port)
        .usePlaintext()
        .build());
  }

  java_xdb_client(ManagedChannel channel) {
    this.channel = channel;
    blockingStub = XecuredbGrpc.newBlockingStub(channel);
  }
```

```
  public void shutdown() throws InterruptedException {
    channel.shutdown().awaitTermination(5, TimeUnit.SECONDS);
  }

  public String Encrypy(String Alias, String Plain) {
    EncRequest request =
EncRequest.newBuilder().setAlias(Alias).setPlain(Plain).build();
    EncReply response;
    try {
      response = blockingStub.encrypt(request);
    } catch (StatusRuntimeException e) {
      return "RPC failed: " + e.getStatus();
    }

    String cipher = response.getCipher();

    return cipher;
  }

  public String Decrypy(String Alias, String Cipher) {

  DecRequest request =
DecRequest.newBuilder().setAlias(Alias).setCipher(Cipher).build();
    DecReply response;
    try {
      response = blockingStub.decrypt(request);
    } catch (StatusRuntimeException e) {
      return "RPC failed: " + e.getStatus();
    }

    String plain = response.getPlain();

    return plain;
  }

  public String Hash(int algo, String Plain) {
    HashRequest request =
HashRequest.newBuilder().setId(algo).setPlain(Plain).build();
    HashReply response;
    try {
      response = blockingStub.cryptoHash(request);
    } catch (StatusRuntimeException e) {
      return "RPC failed: " + e.getStatus();
    }

    String cipher = response.getCipher();

    return cipher;
  }

  public static void main(String[] args) throws Exception {
    java_xdb_client client = new java_xdb_client("localhost", 50052);
    try {
      String Alias = "normal";
      String Plain = "1234567890123";
      String strEnc, strDec, strHash;
```

```
      strEnc  = client.Encrypy(Alias, Plain );
      strDec  = client.Decrypy(Alias, strEnc );
      strHash = client.Hash(6, Plain );

      System.out.println("Encrypt client received: " + strEnc );
      System.out.println("Decrypt client received: " + strDec );
      System.out.println("Hash    client received: " + strHash );
    } finally {
      client.shutdown();
    }
  }
}
```

5. Generate 된 gRPC Package 컴파일

protoc 에 의해 generate 된 java 파일들 을 컴파일 하여 jar 파일을 만들기 위한 과정은 다음 명령을 수행 하여 생성 할 수 있다. 작업을 수행 하기 전에 class directory 는 만들어져 있어야 한다.

```
mkdir class

find ./com/ -name "*.java" > file_name

javac -cp "lib/*" -d ./class -encoding UTF-8 @file_name

jar -cfv ./lib/xdb_grpc.jar -C ./class com
```

위 명령을 실행 하면 `./lib` 경로에 `xdb_grpc.jar` 파일이 추가 된다, `./lib` 에는 현재 다음과 같은 파일들이 존재 한다.

```
animal-sniffer-annotations-1.17.jar
checker-compat-qual-2.5.2.jar
commons-codec-1.3.jar
commons-lang3-3.5.jar
commons-logging-1.1.1.jar
error_prone_annotations-2.2.0.jar
google-auth-library-credentials-0.9.0.jar
google-auth-library-oauth2-http-0.9.0.jar
google-http-client-1.19.0.jar
google-http-client-jackson2-1.19.0.jar
grpc-alts-1.16.1.jar
grpc-auth-1.16.1.jar
grpc-context-1.16.1.jar
grpc-core-1.16.1.jar
grpc-grpclb-1.16.1.jar
grpc-netty-1.16.1.jar
grpc-netty-shaded-1.16.1.jar
grpc-protobuf-1.16.1.jar
grpc-protobuf-lite-1.16.1.jar
grpc-stub-1.16.1.jar
gson-2.7.jar
guava-26.0-android.jar
httpclient-4.0.1.jar
httpcore-4.0.1.jar
j2objc-annotations-1.1.jar
```

```
jackson-core-2.1.3.jar
jsr305-3.0.2.jar
netty-buffer-4.1.30.Final.jar
netty-codec-4.1.30.Final.jar
netty-codec-http2-4.1.30.Final.jar
netty-codec-http-4.1.30.Final.jar
netty-codec-socks-4.1.30.Final.jar
netty-common-4.1.30.Final.jar
netty-handler-4.1.30.Final.jar
netty-handler-proxy-4.1.30.Final.jar
netty-resolver-4.1.30.Final.jar
netty-tcnative-boringssl-static-2.0.7.Final.jar
netty-transport-4.1.30.Final.jar
opencensus-api-0.12.3.jar
opencensus-contrib-grpc-metrics-0.12.3.jar
protobuf-java-3.5.1.jar
protobuf-java-util-3.5.1.jar
proto-google-common-protos-1.0.0.jar
xdb_grpc.jar
```

6. Client 모듈 컴파일

위의 `4. Client 코드 작성` 에서 작성한 source 를 이용하여 컴파일을 실행 한다. 컴파일 명령은 다음과 같다.

```
javac -cp "lib/*" -encoding UTF-8 java_xdb_client.java
```

7. java client 프로그램 실행

프로그램의 실행은 다음 명령을 이용한다.

```
java -cp .:"lib/*" java_xdb_client
```

`실행 결과`

```
Encrypt client received: AAGo8vet8aHoqsGorhx1CsXL
Decrypt client received: 1234567890123
Hash    client received: vKKOGisl4TfIP+4Oave9Hg9SvVYFg8oHobQvmUTFxQs=
```

- 특정 경로에 있는 *.jar 파일들을 classpath 에 추가하여 javac 또는 java 를 실행 할 경우 -cp "lib/*" 처럼 지정 하면 된다. windows os 인 경우는 -cp "lib*" 로 지정 할 수 있다.

8. Remark

위 프로그램은 gRPC examples 에 있는 Helloworld 를 참조 하여 작성 하였다. example 에 있는 Helloworld 는 `gradlew` 을 이용한 sample 로 전체적인 구조를 파악 하기 쉽지 않게 되어 있다. 하지만 `protoc-gen-grpc-java` 생성 및 `protoc` 을 위한 Source Generation 만 완료 되면 나머지는 다른 언어와 비슷하게 사용 할 수 있다.관련 내용은 모두 위에 있으니 참조 하면 된다. 그리고 java plugin 관련 컴파일 방법은 https://github.com/grpc/grpc-java/tree/master/compiler 를 참조 하였다.

- 끝 -