05. gRPC 실용사례 ruby

본 문서 에서는 ruby 에서 gRPC 를 사용하여 Client 프로그램을 작성 하는 방법을 소개 한다. ruby 에서 grpc 설치는 여기 의 `4. Ruby 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. protoc 를 이용한 Interface 파일 생성

ruby 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_ruby_src.sh`

```
# ruby
grpc_tools_ruby_protoc -I ../proto --ruby_out=./lib --grpc_out=./lib
xdb_grpc.proto
```

위 명령을 실행 하면 lib directory 에 `xdb_grpc_pb.rb`, `xdb_grpc_services_pb.rb` 두개의 파일이 생성 된다.

3. Client 코드 작성

ruby 를 이용한 client 는 ruby_xdb_client.rb 로 작성 하였으며 source code 는 다음과 같다.

```
#!/usr/bin/env ruby

this_dir = File.expand_path(File.dirname(__FILE__))
lib_dir = File.join(this_dir, 'lib')
$LOAD_PATH.unshift(lib_dir) unless $LOAD_PATH.include?(lib_dir)

require 'grpc'
require 'xdb_grpc_services_pb'

def main
  stub = Xdbproto::Xecuredb::Stub.new('localhost:50052',
:this_channel_is_insecure)
  strAlias = 'normal'
  strSource = '1234567890'
  algo = 6
  strEnc = stub.encrypt(Xdbproto::EncRequest.new(alias: strAlias, plain:
strSource)).cipher
```

```
  strDec = stub.decrypt(Xdbproto::DecRequest.new(alias: strAlias, cipher:
strEnc)).plain
  strHash = stub.crypto_hash(Xdbproto::HashRequest.new(id: algo, plain:
strSource)).cipher
  p "Encrypt: #{strEnc}"
  p "Decrypt: #{strDec}"
  p "Hash   : #{strHash}"
end

main
```

ruby 에서 gRPC 를 사용 할 경우 stub function 이 생성 될 때는 함수명이 변경되 므로 주의가 필요하다. 위의 `xdb_grpc.proto` 를 보면 Interface 함수가 Encrypt, Decrypt,CryptoHash 로 되어 있는데, ruby 의 stub function 은 encrypt, decrypt, crypto_hash 로 변경 해서 사용 하는 부분을 주의깊게 보아야 한다.

4. ruby client 프로그램 실행

프로그램의 실행은 다음 명령을 이용한다.

```
ruby ruby_xdb_client.rb
```

`실행 결과`

```
"Encrypt: AAG7OTjmmCbk96JOdMmv6jCM"
"Decrypt: 1234567890"
"Hash   : x3Xnt1ft5jDNCqERO9ECZhqziCnKUqZCKreChi8mhkY="
```

5. Remark

위 프로그램은 gRPC examples 에 있는 Helloworld 를 참조 하여 작성 하였다.

- 끝 -