gRPC 구현

Quote >

https://holjjack.tistory.com/252

Service

gRPC Service 프로젝트 추가

gRPC Service 프로젝트가 보이지 않는다면

NET Core 3.1 SDK 다운로드 : https://dotnet.microsoft.com/download/dotnet-core/3.1

- Services : 함수를 정의
- Protos : service에 정의한 함수와 message에 서버와 클라이언트 간에 주고받을 데이터 포맷을 정의
- csharp_namespace에서 정의한 네임스페이스를 아래 서비스 구현부(GreeterService.cs)의 namespace와 매칭이 되어야 한다.

API 추가

1. proto file에 service(API)와 message(argument, return)를 추가한다.

```
service Greeter {
  rpc TestAPI (TestMsg) returns (TestRet);
}

message TestMsg {
        string param = 1;
}

message TestRet {
        string param = 1;
}
```

2. service file에 구현부를 추가한다.

```
public override Task<TestRet> TestAPI(TestMsg massage, ServerCallContext context)
{
    return Task.FromResult(new TestRet
    {
        Param = "Service received " + massage.Param
```

```
    });
}
```

Client

1. 콘솔 애플리케이션 프로젝트 추가
2. NuGet 패키지 추가
    - Grpc.Net.Client
    - Grpc.Tools
    - Google.Protobuf
3. Service의 Protos 폴더를 복사
4. .proto file의 csharp_namespace를 Client에 맞게 변경
5. 프로젝트 파일 편집 - Protobuf 부분 추가

```
<ItemGroup>
        <Protobuf Include="Protos\greet.proto" GrpcServices="Client" />
</ItemGroup>
```

6. Main부를 하기와 같이 수정

```
static async Task Main(string[] args)
{
    using var channel = GrpcChannel.ForAddress("https://localhost:5001");
    var client = new Greeter.GreeterClient(channel);

    var reply = await client.SayHelloAsync(
                        new HelloRequest { Name = "GreeterClient" });
    Console.WriteLine("Greeting: " + reply.Message);

    var ret = await client.TestAPIAsync(
                            new TestMsg { Param = "TestParameter" });
    Console.WriteLine("Greeting: " + ret.Param);

    Console.WriteLine("Press any key to exit...");
    Console.ReadKey();
}
```