Giới thiệu GRPC

Contents

I. Tổng quan

a. Khái niệm

- gRPC là một framework RPC mã nguồn mở, hiện đại và hiệu năng cao mà có thể chạy trên bất kỳ môi trường nào. Framework này được Google khởi công phát triển vào năm 2015, đến 08/2016 thì được phát hành chính thức. Đây được cho là một thế hệ tiếp theo của RPC (Remote Procedure Calls) đặc biệt là trong mô hình Microservices.
- Sử dụng giao thức HTTP2 mã hoá các dữ liệu thành nhị phân
- Giao tiếp theo dạng client server cả phía client hay server phải đầy đủ các class của service cần dùng
- Sử dụng cho các mô hình phân tán có hiệu suất cao
- Sử dụng các file .proto để general các class

II. Cách cài đặt và sử dụng:

a. Cài đặt môi trường:

- Java, các thư viện như maven, protoc, protoc-gen-grpc-java
  - Java: sau khi cài đặt cấu hình JAVA_HOME trong path
  - Maven: Cấu hình MAVEN_HOME trong path environment
  - Protoc thêm đường dẫn vào trong path
  - Protoc-gen-grpc-java: giúp general code sử dụng trong câu lệnh protoc.

b. Cấu hình project:

- Cấu hình maven:
  - Các dependentcy: Spring-boot-started-test, mysql-connector-java, io.grpc:grpc-netty, io.grpc:grpc-protobuf, io.grpc:grpc-stub …
  - Cấu hình file build và plugin của java

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-parent</artifactId>
       <version>2.7.11</version>
       <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo</name>
    <description>Demo project for Spring Boot</description>
    <properties>
       <java.version>1.8</java.version>
    </properties>
    <dependencies>
       <dependency>
```

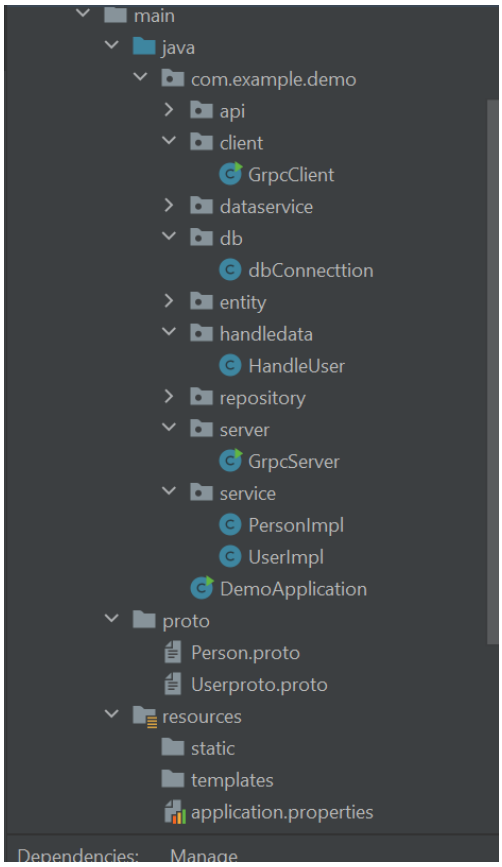
```
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
<!--connect database Mysql-->
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-jdbc</artifactId>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.27</version>
        </dependency>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>5.3.14</version>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
<!--end connect database-->

        <dependency>
            <groupId>io.grpc</groupId>
            <artifactId>grpc-netty</artifactId>
            <version>1.16.1</version>
        </dependency>
        <dependency>
            <groupId>io.grpc</groupId>
            <artifactId>grpc-protobuf</artifactId>
            <version>1.16.1</version>
        </dependency>
        <dependency>
            <groupId>io.grpc</groupId>
            <artifactId>grpc-stub</artifactId>
            <version>1.16.1</version>
        </dependency>

    </dependencies>
    <build>
        <extensions>
            <extension>
                <groupId>kr.motd.maven</groupId>
                <artifactId>os-maven-plugin</artifactId>
                <version>1.6.1</version>
            </extension>
        </extensions>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
```

```
          <plugin>
             <groupId>org.xolstice.maven.plugins</groupId>
             <artifactId>protobuf-maven-plugin</artifactId>
             <version>0.6.1</version>
             <configuration>
                <protocArtifact>
         com.google.protobuf:protoc:3.3.0:exe:${os.detected.classifier}
                </protocArtifact>
                <pluginId>grpc-java</pluginId>
                <pluginArtifact>
                   io.grpc:protoc-gen-grpc-
java:1.4.0:exe:${os.detected.classifier}
                </pluginArtifact>
             </configuration>
             <executions>
                <execution>
                   <goals>
                      <goal>compile</goal>
                      <goal>compile-custom</goal>
                   </goals>
                </execution>
             </executions>
          </plugin>
          <plugin>
             <groupId>org.codehaus.mojo</groupId>
             <artifactId>build-helper-maven-plugin</artifactId>
             <version>3.2.0</version>
             <executions>
                <execution>
                   <phase>generate-sources</phase>
                   <goals>
                      <goal>add-source</goal>
                   </goals>
                   <configuration>
                      <sources>
                       <source>target/generated-sources/protobuf/grpc-
java</source>
                      </sources>
                   </configuration>
                </execution>
             </executions>
          </plugin>
       </plugins>
    </build>
</project>
```

File cấu hình pom của maven.



Cấu trúc project đơn giản.

III Tạo các file proto trong src/main/proto

Mặc định khi genneral code thì đường dẫn src/main/proto được sử dụng để tạo code có thể sửa đường dẫn bằng cách sửa

```
<protoFile>
    <directory>${basedir}/src/main/proto</directory>
    <includes>
```

Cấu trúc các file proto:

```
syntax = "proto3";

option  java_multiple_files = true;
package com.example.demo;


message PersonRequest{
    string firstName = 1;
    string lastName = 2;
}
message PersonResponse{
    string fullName = 1;
}
service PersonService{
    rpc getFullName(PersonRequest) returns (PersonResponse);
}
```

Sau khi tạo các file proto xong ta có 2 cách để sử dụng

Cách 1 sử dụng protoc để tạo các class grpc:

```
protoc --plugin=protoc-gen-grpc-java=$PATH_TO_PLUGIN -I=$SRC_DIR
  --java_out=$DST_DIR --grpc-java_out=$DST_DIR $SRC_DIR/HelloService.proto
```

Trong đó $PATH_TO_PLUGIN là đường đẫn đến file exe của protoc-gen-grpc.exe sau cuối là tên thư mục .proto

Cách 2 sử dụng maven install để cài đặt code

Sau khi dùng maven install ở cmd ta cần phải cấu hình thêm plugin để sử dụng code trong target/generated-source

```
<plugin>
    <groupId>org.codehaus.mojo</groupId>
    <artifactId>build-helper-maven-plugin</artifactId>
    <version>3.2.0</version>
    <executions>
       <execution>
          <phase>generate-sources</phase>
          <goals>
             <goal>add-source</goal>
          </goals>
          <configuration>
             <sources>
                <source>target/generated-sources/protobuf/grpc-
java</source>
             </sources>
          </configuration>
       </execution>
    </executions>
</plugin>
```

Tạo các class service kế thừa grpcImplBase

Override các service có trong file .proto

```
public class PersonImpl extends PersonServiceGrpc.PersonServiceImplBase {

    @Override
    public void getPerson(PersonRequest request, StreamObserver<Person>
responseObserver) {
        String greeting = new StringBuilder()
                .append("Hello, ")
                .append(request.getFirstName())
                .append("Alex")
                .append(request.getLastName())
                .toString();

        PersonResponse response = PersonResponse.newBuilder()
                .setFullName(greeting)
                .build();
    }
    @Override
```

```
    public void getFullName(PersonRequest request,
StreamObserver<PersonResponse> responseObserver) {
        // Lấy thông tin từ PersonRequest
        String firstName = request.getFirstName();
        String lastName = request.getLastName();

        // Tạo tên đầy đủ từ các thông tin trên
        String fullName = firstName;
        fullName += " " + lastName;

        // Tạo đối tượng Person để trả về
        PersonResponse person = PersonResponse.newBuilder()
                .setFullName(fullName)
                .build();

        // Trả về đối tượng Person
        responseObserver.onNext(person);
        responseObserver.onCompleted();
    }
}
```

Tạo server với post và đăng ký service

```
public class GrpcServer {
    public static void main(String[] args) throws IOException,
InterruptedException {
        Server server = ServerBuilder.forPort(8081)
                .addService(new PersonImpl())
                .addService(new UserImpl())
                .build();

        server.start();
        server.awaitTermination();
    }

}
```

Tạo client để gọi grpcserver

```
public class GrpcClient {
    public static void main(String[] args) {
        ManagedChannel channel =
ManagedChannelBuilder.forAddress("localhost", 8081)
                .usePlaintext()
                .build();

        PersonServiceGrpc.PersonServiceBlockingStub stub
                = PersonServiceGrpc.newBlockingStub(channel);

        PersonResponse helloResponse =
stub.getFullName(PersonRequest.newBuilder()
                .setFirstName("Nguyen")
                .setLastName("Hello")
                .build());
        System.out.println("Full name: " + helloResponse.getFullName());
      channel.shutdown();
      }
}
```

Connect database

Tạo class db connect

```
2 usages
public class dbConnecttion {
    1 usage
    public  static Connection mysqlConnecttion(){
        Connection connection = null;
        String jdbcUrl = "jdbc:mysql://localhost:3306/testgrpc";
        String userName = "root";
        String password = "";
        try {
            connection = DriverManager.getConnection(jdbcUrl, userName, password);
        } catch (SQLException e) {
            throw new RuntimeException(e);
        }
        return connection;
    }
}
```

Sử dụng dbconnet trong service

```
@Override
public void getListUser(UserRequest request, StreamObserver<UserResponse>
responseStreamObserver) {
    int limit = request.getLimit();
    String sortBy = request.getSortBy();

    // TODO: Thực hiện lấy danh sách người dùng từ database và chuyển đổi
sang đối tượng UserEntity
    String sql = "SELECT * FROM users";
    Statement statement = null;
    List<UserEntity> userEntityList = new ArrayList<>();
    try {
        statement = connection.createStatement();
        ResultSet rs = statement.executeQuery(sql);
        userEntityList = HandleUser.handle(rs);
        statement.close();
    } catch (SQLException e) {
        throw new RuntimeException(e);
    }

    UserResponse userResponse = UserResponse.newBuilder()
            .addAllUserEntity(userEntityList)
            .build();

    responseStreamObserver.onNext(userResponse);
    responseStreamObserver.onCompleted();
}
```