gRPC

Юрий Литвинов
yurii.litvinov@gmail.com

29.04.2019г

Protocol buffers

protobuf

- Механизм сериализации-десериализации данных
- Компактное бинарное представление
- Декларативное описание формата данных, генерация кода для языка программирования
  - Поддерживается Java, C#, Python, C++, Objective-C, Go, Ruby, ...
- Бывает v2 и v3, с некоторыми синтаксическими отличиями
- Хитрый протокол передачи, https://developers.google.com/protocol-buffers/docs/encoding
  - Base 128 varint-ы
  - До 10 раз компактнее XML

Пример

Файл .proto:

```
message Person {
  required string name = 1;
  required int32 id = 2;
  optional string email = 3;
}
```

Файл .java:

```
Person john = Person.newBuilder()
    .setId(1234)
    .setName("John Doe")
    .setEmail("jdoe@example.com")
    .build();
output = new FileOutputStream(args[0]);
john.writeTo(output);
```

Технические подробности

Для Java:

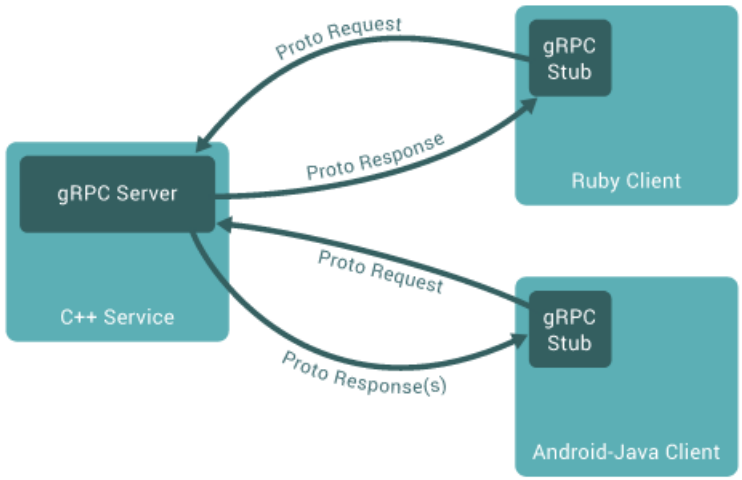
- .proto-файлы — в папку src/main/proto
- gradle-плагин com.google.protobuf
- https://github.com/google/protobuf-gradle-plugin/blob/master/examples/exampleProject/build.gradle

Для остального: скачать и поставить protoc

- https://github.com/google/protobuf

gRPC

- Средство для удалённого вызова (RPC)
- Работает поверх protobuf
- Разрабатывается Google
- Поддерживает Java, C#, C++, Python, Objective-C, Go, Ruby, ...



Технические подробности

- Сервисы описываются в том же .proto-файле, что и протокол protobuf-a
- В качестве типов параметров и результатов — message-и protobuf-a

```
service RouteGuide {
  rpc GetFeature(Point) returns (Feature) {}
  rpc ListFeatures(Rectangle) returns (stream Feature) {}
  rpc RecordRoute(stream Point) returns (RouteSummary) {}
  rpc RouteChat(stream RouteNote) returns (stream RouteNote) {}
}
```

- Сборка — плагином grpc к protoc

Реализация сервиса на Java

```
private static class RouteGuideService extends RouteGuideGrpc.RouteGuideImplBase {
  ...
  @Override
  public void getFeature(Point request, StreamObserver<Feature> responseObserver) {
   responseObserver.onNext(checkFeature(request));
   responseObserver.onCompleted();
  }

  @Override
  public void listFeatures(Rectangle request, StreamObserver<Feature> responseObserver) {
   for (Feature feature : features) {
     ...
     int lat = feature.getLocation().getLatitude();
     int lon = feature.getLocation().getLongitude();
     if (lon >= left && lon <= right && lat >= bottom && lat <= top) {
       responseObserver.onNext(feature);
     }
   }
   responseObserver.onCompleted();
  }
```

Реализация сервиса на Java (2)

```
@Override
public StreamObserver<RouteNote> routeChat(
      final StreamObserver<RouteNote> responseObserver) {
 return new StreamObserver<RouteNote>() {
  @Override
  public void onNext(RouteNote note) {
   List<RouteNote> notes = getOrCreateNotes(note.getLocation());
   for (RouteNote prevNote : notes.toArray(new RouteNote[0])) {
    responseObserver.onNext(prevNote);
   }
   notes.add(note);
  }
  @Override
  public void onError(Throwable t) {
   logger.log(Level.WARNING, "routeChat cancelled");
  }
  @Override
  public void onCompleted() {
   responseObserver.onCompleted();
  }
 };
}
```

Что написать в build.gradle

```
protobuf {
    protoc {
        artifact = "com.google.protobuf:protoc:3.5.1-1"
    }
    plugins {
        grpc {
            artifact = 'io.grpc:protoc-gen-grpc-java:1.13.0-SNAPSHOT'
        }
    }
    generateProtoTasks {
        all()*.plugins {
            grpc {
                outputSubDir = 'java'
            }
        }
    }
    generatedFilesBaseDir = "$projectDir/src"
}
```

Запуск сервиса

```
private void start() throws IOException {
    int port = 50051;
    server = ServerBuilder.forPort(port)
        .addService(new GreeterImpl())
        .build()
        .start();
    logger.info("Server started, listening on " + port);
    Runtime.getRuntime().addShutdownHook(new Thread() {
        @Override
        public void run() {
            System.err.println("*** shutting down gRPC server since JVM is shutting down");
            HelloWorldServer.this.stop();
            System.err.println("*** server shut down");
        }
    });
}
```

Инициализация клиента

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

https://github.com/grpc/grpc-java/blob/master/examples/src/main/java/io/grpc/examples/helloworld/HelloWorldClient.java

Задание на пару

В командах по два человека разработать сетевой чат (наподобие Telegram) с помощью gRPC

- peer-to-peer, то есть соединение напрямую
- Графический пользовательский интерфейс
  - Отображение имени отправителя, даты и текста сообщения
- При запуске указываются:
  - Адрес peer-а и порт, если хотим подключиться
    - Должно быть можно не указывать, тогда работаем в режиме сервера
  - Своё имя пользователя