

gRPC

Юрий Литвинов
yurii.litvinov@gmail.com

17.05.2017г

Protocol buffers

protobuf

- ▶ Механизм сериализации-десериализации данных
- ▶ Компактное бинарное представление
- ▶ Декларативное описание формата данных, генерация кода для языка программирования
 - ▶ Поддерживается Java, Python, Objective-C, C++, Go, JavaNano, Ruby, C#
- ▶ Бывает v2 и v3, с некоторыми синтаксическими отличиями
- ▶ Хитрый протокол передачи,
<https://developers.google.com/protocol-buffers/docs/encoding>
 - ▶ 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:

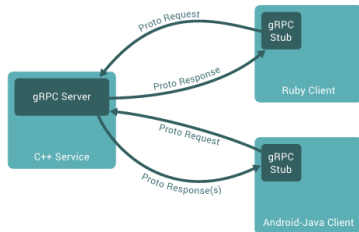
- ▶ .proto-файлы — в папку `src/main/proto`
- ▶ gradle-плагин `com.google.protobuf`
- ▶ <https://github.com/google/protobuf-gradle-plugin/blob/master/exampleProject/build.gradle>

Для остального: скачать и поставить `protoc`

- ▶ <https://github.com/google/protobuf>

gRPC

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



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();
        }
    };
}

```


Задание на пару

Разработать сетевой чат (наподобие Jabber) с помощью gRPC

- ▶ peer-to-peer, то есть соединение напрямую
- ▶ Консольный пользовательский интерфейс
 - ▶ Отображение имени отправителя, даты и текста сообщения
- ▶ Адрес peer-а и порт — параметры
- ▶ Указание своего имени — параметром или в конфигурационном файле
- ▶ Логирование