

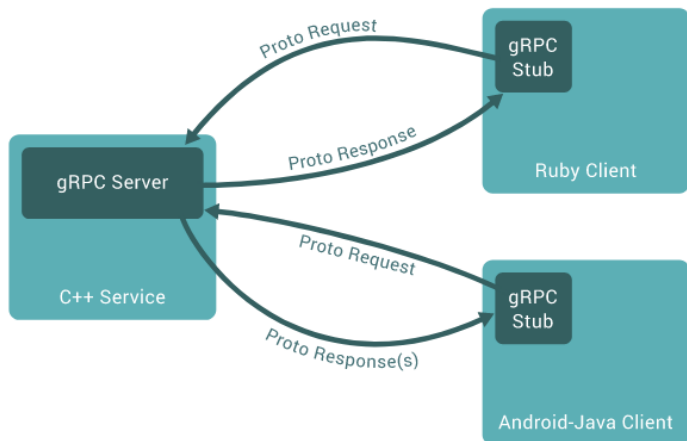


Workshop

Simple service definition

Define your service using Protocol Buffers, a powerful binary serialization toolset and language

[READ MORE](#)



Works across languages and platforms

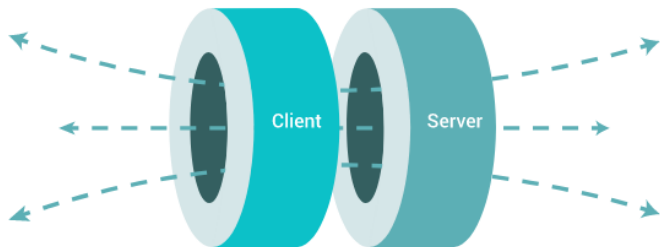
Automatically generate idiomatic client and server stubs for your service in a variety of languages and platforms

[READ MORE](#)

Start quickly and scale

Install runtime and dev environments with a single line and also scale to millions of RPCs per second with the framework

[READ MORE](#)



Bi-directional streaming and integrated auth

Bi-directional streaming and fully integrated pluggable authentication with http/2 based transport

[READ MORE](#)

Why Grpc

- Heeft de voordelen van SoapUI :
 - contract
 - security (verschillende modellen)
 - code generatie
- En heeft de voordelen Json/Rest
 - speed
 - eenvoud

Protobuf

Contract wordt vastgelegd in een .proto file, en van daaruit wordt code gegenereerd.

Syntax:

```
service ServiceNaam {  
    rpc MethodeAbc (msg) returns (msg) {}  
    rpc MethodeYYY(stream msg) returns (stream msg) {}  
}
```

```
message AbcMsg {  
    string Naam = 1;  
    int64 GeboorteDatum = 2;  
    bytes RuweData = 3;  
    repeated Adres = 4;  
    GeslachtMsg geslacht = 5;  
}
```

Protobuf vervolg ...

```
// 'inheritance' is supported with:
message GenericMsg {
    oneof value {
        Specific01Msg specific01 = 1;
        Specific02Msg specific02 = 2;
    }
    ..
}

//enumerations:
enum GeslachtMsg {
    MAN = 0;
    VROUW = 1;
    NEUTRAAL = 2;
}

// other proto file(s) can be used.
import "andere.proto";
```

Protobuf vervolg ...

Compileren kan met protoc, mvn en/of gradle :

```
mvn protobuf:compile
```

```
mvn protobuf:compile-custom
```

Deze genereerd code in ./target/generated-source/protobuf/java :

```
XxxMsg.java + XxxMsgOrBuilder.java
```

resp: ./target/generated-source/protobuf/java-grpc :

ServiceNaamGrpc.java , deze bevat:

```
newStub(channel)
```

```
newBlockingStub(channel)
```

```
ServiceNaamGrpc.ServiceNaamImplBase
```

Implement grpc code ..1

Starten van de server (zonder security)

```
server = ServerBuilder.forPort(int).addService(service-impl).build();  
server.start();
```

Starten van de client (zonder security):

```
channel = ManagedChannelBuilder.forAddress(host, port).usePlaintext(true));
```

```
//todo secure
```


Implement grpc code vervolg .. 2

Server, implement services:

Maak een class die: `ServiceNaamGrpc.ServiceNaamImplBase` extends, en implementeer de bijbehorende methodes.

Voorbeeld: `rpc GetFeature(Point) returns (Feature) {}`

`@Override`

```
public void getFeature(Point request,  
StreamObserver<Feature> responseObserver) {  
    responseObserver.onNext(Feature);  
    responseObserver.onCompleted();  
}
```

```
feature = blockingStub.getFeature(request);
```

Implement grpc code vervolg .. 3a(server)

Server, implement services:

Voorbeeld: rpc RecordRoute(stream Point) returns (RouteSummary) {}

```
public StreamObserver<Point>
recordRoute(StreamObserver<RouteSummary> responseObserver) {
    return new StreamObserver<Point>() {
        @Override
        public void onNext(Point point) {
            ...
        }
        @Override
        public void onError(Throwable t) {
            ...
        }
        @Override
        public void onCompleted() {
            responseObserver.onNext(RouteSummary.newBuilder().build());
            responseObserver.onCompleted();
        }
    };
};
```

Implement grpc code vervolg .. 3b (client)

Server, implement services:

Voorbeeld: `rpc RecordRoute(stream Point) returns (RouteSummary) {}`

```
StreamObserver<Point> reqObserver = asyncStub.recordRoute(respObserver);
```

```
StreamObserver<RouteSummary> responseObserver = new StreamObserver<RouteSummary>()  
{  
    @Override  
    public void onNext(RouteSummary summary) {  
        //doe iets met summary  
    }  
  
    @Override  
    public void onError(Throwable t) {  
    }  
  
    @Override  
    public void onCompleted() {  
        info("Finished RecordRoute");  
        finishLatch.countDown();  
    }  
};
```

Implement grpc code vervolg .. 4

Server, implement services:

Voorbeeld: rpc RouteChat(stream RouteNote) returns (stream RouteNote) {}

```
public StreamObserver<RouteNote> routeChat(StreamObserver<RouteNote>
    responseObserver)
{
    return new StreamObserver<RouteNote>() {
        @Override
        public void onNext(RouteNote note) {
            ...
            responseObserver.onNext(RouteNote..build());
        }

        @Override
        public void onError(Throwable t) {
        }

        @Override
        public void onCompleted() {
            responseObserver.onCompleted();
        }
    };
};
```

Hands-on

Volg de aanwijzing in :

doc\workshop-hanson.pdf

Met hints in doc\hints.txt (en deze presentatie)
(en werkende code in: \doc\working-code)

Zie verder : <http://www.grpc.io/>