Maksym Kovacevic 4DHIT

Dokumentation gRPC Framework

Grundlagen

Fragen

1. What is gRPC and why does it work across languages?

gRPC is a framework that let programs talk to each other, and it works everywhere because it uses shared rules (proto files) that every language understands.

2. RPC lifecycle (client → server → client)

1. The client sent a request to the server.
2. The request is serialized using Protocol Buffers.
3. The server receives and deserializes the request.
4. The server executes the method.
5. The server sends a response back.
6. The client get the result.

3. Workflow of Protocol Buffers

1. You write a `.proto` file.
2. The compiler generates classes for your language.
3. Your program uses these classes to send/recieve data.

4. Benefits of Protocol Buffers

- Very fast
- Smaller than JSON
- Works across many languages

5. When NOT recommended?

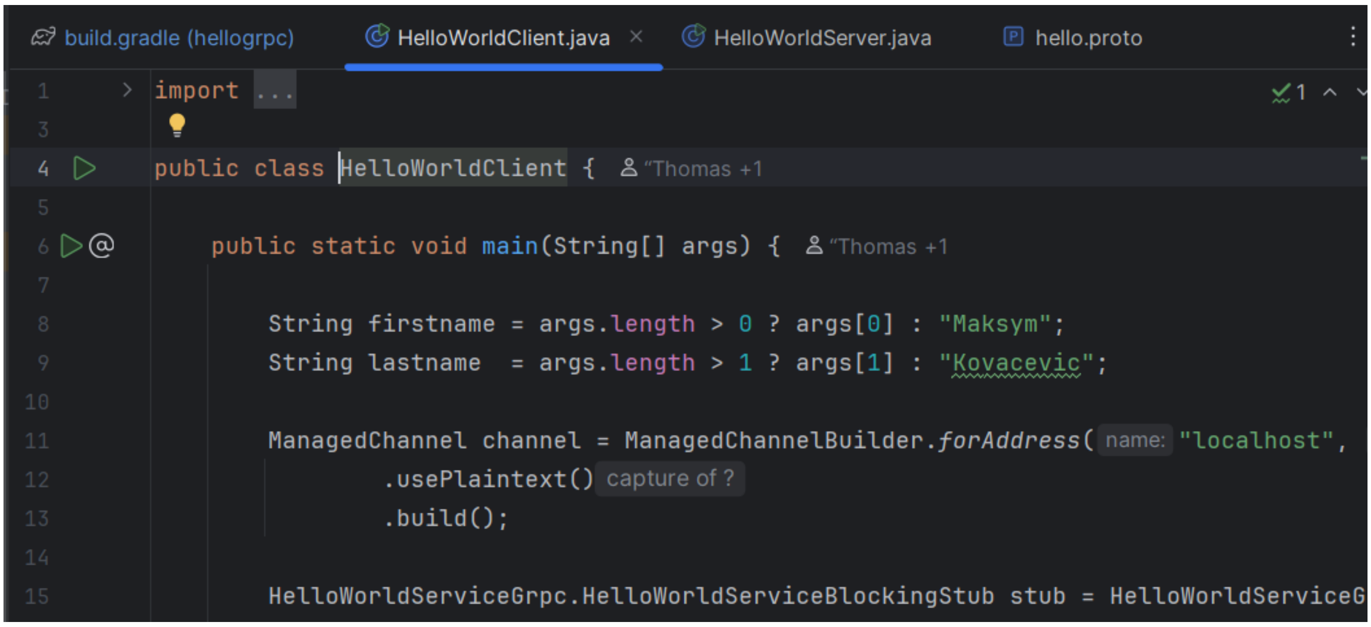
- If humans must read/edit the data

6. Three data types in Protobuf

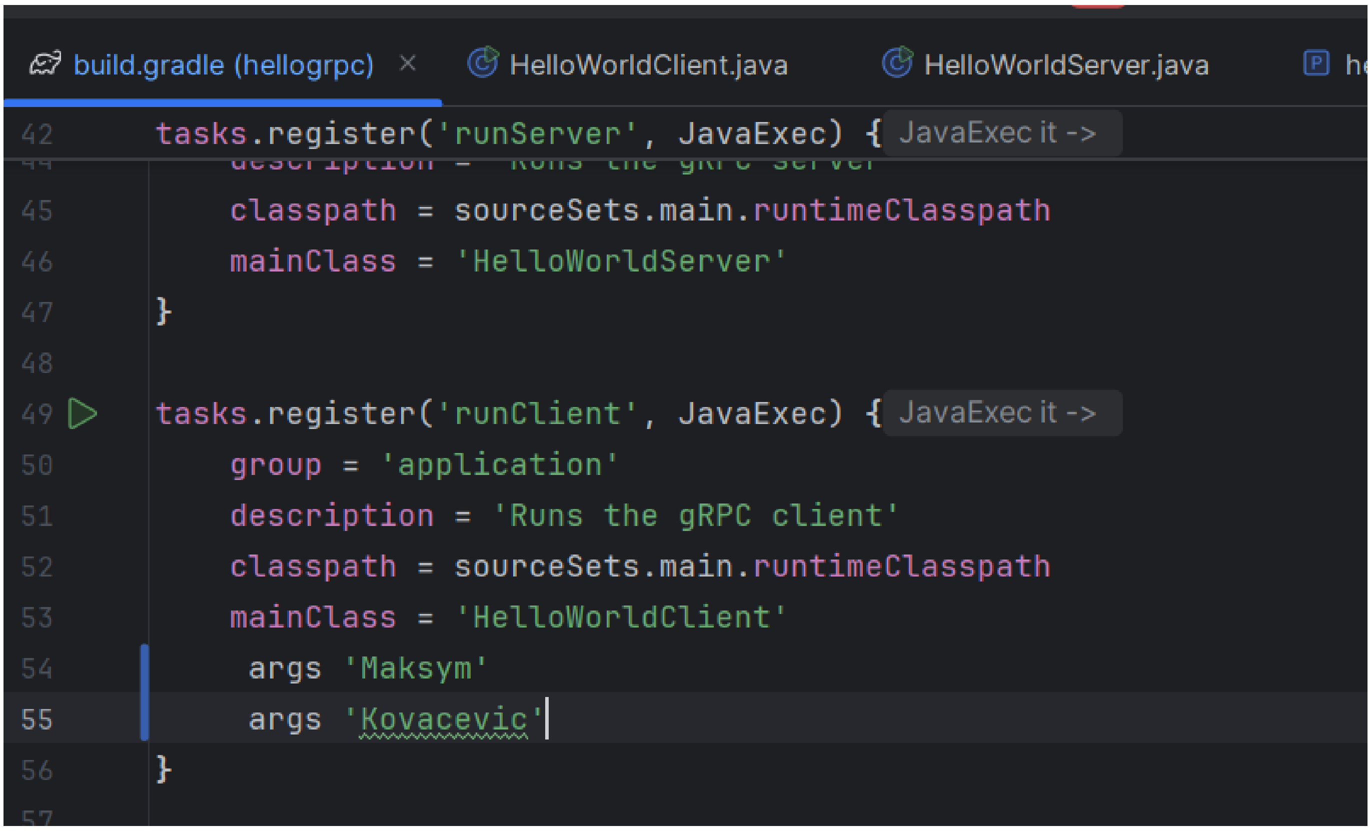
- `string`
- `int32`
- `bool`

Exercise

First i forked the Repository from the following Link from our teacher: https://github.com/ThomasMicheler/DEZSYS_GK_HELLOWORLD_GRPC.git Then i only need to change the Name in the build.gradle and in the HelloWorldClient class:

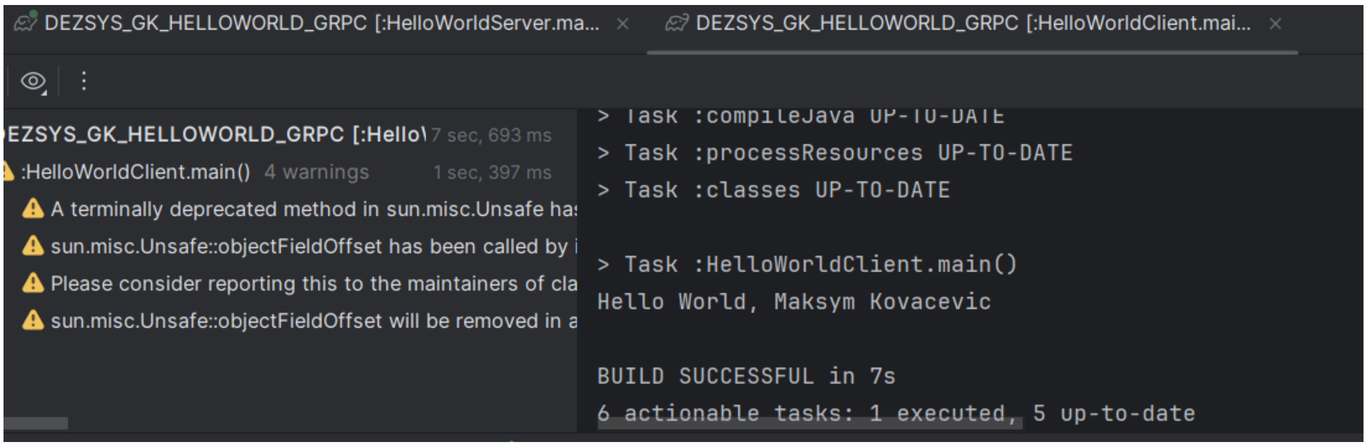


```java
import ...

public class HelloWorldClient {  "Thomas +1

    public static void main(String[] args) {  "Thomas +1

        String firstname = args.length > 0 ? args[0] : "Maksym";
        String lastname  = args.length > 1 ? args[1] : "Kovacevic";

        ManagedChannel channel = ManagedChannelBuilder.forAddress( name: "localhost",
                .usePlaintext() capture of ?
                .build();

        HelloWorldServiceGrpc.HelloWorldServiceBlockingStub stub = HelloWorldServiceG
```



```
tasks.register('runServer', JavaExec) {
    classpath = sourceSets.main.runtimeClasspath
    mainClass = 'HelloWorldServer'
}

tasks.register('runClient', JavaExec) {
    group = 'application'
    description = 'Runs the gRPC client'
    classpath = sourceSets.main.runtimeClasspath
    mainClass = 'HelloWorldClient'
     args 'Maksym'
     args 'Kovacevic'
}
```

Now when i run the Gradle Task i can see the following output:


DEZSYS_GK_HELLOWORLD_GRPC [:HelloWorldServer.ma...
DEZSYS_GK_HELLOWORLD_GRPC [:HelloWorldClient.mai...
EZSYS_GK_HELLOWORLD_GRPC [:Hello 7 sec, 693 ms
:HelloWorldClient.main() 4 warnings 1 sec, 397 ms

```
A terminally deprecated method in sun.misc.Unsafe has
sun.misc.Unsafe::objectFieldOffset has been called by
Please consider reporting this to the maintainers of cla
sun.misc.Unsafe::objectFieldOffset will be removed in a
```

```
> Task :compileJava UP-TO-DATE
> Task :processResources UP-TO-DATE
> Task :classes UP-TO-DATE

> Task :HelloWorldClient.main()
Hello World, Maksym Kovacevic

BUILD SUCCESSFUL in 7s
6 actionable tasks: 1 executed, 5 up-to-date
```

Erweiterte Grundlagen

After that i create a new `.proto` File:

```
syntax = "proto3";

service DataWarehouseService {
  rpc sendWarehouse (Warehouse) returns (Ack);
}

message Warehouse {
  string warehouseID = 1;
  string warehouseName = 2;
  string warehouseAddress = 3;
  string warehousePostalCode = 4;
  string warehouseCity = 5;
  string warehouseCountry = 6;
  string timestamp = 7;
  repeated ProductData productdata = 8;
}

message ProductData {
  string productID = 1;
  string productName = 2;
  string productCategory = 3;
  int32 productQuantity = 4;
  string productUnit = 5;
}

message Ack {
  string message = 1;
}
```

For the Server I also created a new file:

```
import io.grpc.Server;
import io.grpc.ServerBuilder;

public class DataWarehouseServer {
    public static void main(String[] args) throws Exception {

        Server server = ServerBuilder.forPort(50051)
                .addService(new DataWarehouseServiceImpl())
                .build();

        System.out.println("DataWarehouse gRPC Server running...");
        server.start();
        server.awaitTermination();
    }
}
```

Then I also created the Client as java language like this, which I copied the data from the last exercise:

```java
import io.grpc.ManagedChannel;
import io.grpc.ManagedChannelBuilder;

public class DataWarehouseClient {

    public static void main(String[] args) {

        ManagedChannel channel = ManagedChannelBuilder
                .forAddress("127.0.0.1", 50051)
                .usePlaintext()
                .build();

        DataWarehouseServiceGrpc.DataWarehouseServiceBlockingStub stub =
                DataWarehouseServiceGrpc.newBlockingStub(channel);

        Datawarehouse.ProductData product1 = Datawarehouse.ProductData.newBuilder()
                .setProductID("00-852374")
                .setProductName("Apfelsaft")
                .setProductCategory("Saft")
                .setProductQuantity(1245)
                .setProductUnit("Packung 1L")
                .build();

        Datawarehouse.ProductData product2 = Datawarehouse.ProductData.newBuilder()
                .setProductID("00-992100")
                .setProductName("Mineralwasser")
                .setProductCategory("Getränk")
                .setProductQuantity(500)
                .setProductUnit("Flasche 0.5L")
                .build();

        Datawarehouse.Warehouse warehouse = Datawarehouse.Warehouse.newBuilder()
                .setWarehouseID("001")
                .setWarehouseName("TGM Bahnhof")
                .setWarehouseAddress("Wexstraße")
                .setWarehousePostalCode("1210")
                .setWarehouseCity("Wien")
                .setWarehouseCountry("Österreich")
                .setTimestamp("2025-12-02 15:02:57.163")
                .addProductdata(product1)
                .addProductdata(product2)
                .build();

        stub.sendWarehouse(warehouse);

        System.out.println("=== Produktliste ===\n");

        for (Datawarehouse.ProductData p : warehouse.getProductdataList()) {
            System.out.println("ProduktID:      " + p.getProductID());
            System.out.println("Name:           " + p.getProductName());
            System.out.println("Kategorie:      " + p.getProductCategory());
            System.out.println("Menge:          " + p.getProductQuantity());
            System.out.println("Einheit:        " + p.getProductUnit());
            System.out.println("-----------------------------------");
        }

        channel.shutdown();
    }
}
```

Then i made a DataWarehouseServiceImpl Class like this:

```java
import io.grpc.stub.StreamObserver;

public class DataWarehouseServiceImpl
        extends DataWarehouseServiceGrpc.DataWarehouseServiceImplBase {

    @Override
    public void sendWarehouse(Datawarehouse.Warehouse request,
                              StreamObserver<Datawarehouse.Ack> responseObserver) {

        System.out.println("===== RECEIVED WAREHOUSE FROM PYTHON =====");

        System.out.println("warehouseID:       " + request.getWarehouseID());
        System.out.println("warehouseName:     " + request.getWarehouseName());
        System.out.println("warehouseAddress:  " + request.getWarehouseAddress());
        System.out.println("warehousePostCode: " + request.getWarehousePostalCode());
        System.out.println("warehouseCity:     " + request.getWarehouseCity());
        System.out.println("warehouseCountry:  " + request.getWarehouseCountry());
        System.out.println("timestamp:         " + request.getTimestamp());

        System.out.println("\nProducts:");

        for (Datawarehouse.ProductData p : request.getProductdataList()) {
            System.out.println("-----------------------------------");
            System.out.println("productID:       " + p.getProductID());
            System.out.println("productName:     " + p.getProductName());
            System.out.println("productCategory: " + p.getProductCategory());
            System.out.println("productQuantity: " + p.getProductQuantity());
            System.out.println("productUnit:     " + p.getProductUnit());
        }

        Datawarehouse.Ack ack = Datawarehouse.Ack.newBuilder()
                .setMessage("Warehouse & product data received successfully")
                .build();

        responseObserver.onNext(ack);
        responseObserver.onCompleted();
    }
}
```

Vertiefung

First i ran the 3 commands in the `README.md` Then i made this python File, the same as my Java Class:

```python
import grpc
import json
from datawarehouse_pb2 import Warehouse, ProductData
from datawarehouse_pb2_grpc import DataWarehouseServiceStub


def pretty_json(data):
    return json.dumps(data, indent=4, ensure_ascii=False)


def main():
    channel = grpc.insecure_channel("127.0.0.1:50051")
    stub = DataWarehouseServiceStub(channel)

    warehouse = Warehouse(
        warehouseID="001",
        warehouseName="TGM Bahnhof",
        warehouseAddress="Wexstraße",
        warehousePostalCode="1210",
        warehouseCity="Wien",
        warehouseCountry="Österreich",
        timestamp="2025-12-02 15:02:57.163",
        productdata=[
            ProductData(
                productID="00-852374",
                productName="Apfelsaft",
                productCategory="Saft",
                productQuantity=1245,
                productUnit="Packung 1L"
            ),
            ProductData(
                productID="00-992100",
                productName="Mineralwasser",
                productCategory="Getränk",
                productQuantity=500,
                productUnit="Flasche 0.5L"
            )
        ]
    )

    response = stub.sendWarehouse(warehouse)

    print("\n===== Warehouse Sent =====\n")

    output = {
        "warehouseID": warehouse.warehouseID,
        "warehouseName": warehouse.warehouseName,
        "warehouseAddress": warehouse.warehouseAddress,
        "warehousePostalCode": warehouse.warehousePostalCode,
        "warehouseCity": warehouse.warehouseCity,
            "warehouseCountry": warehouse.warehouseCountry,
            "timestamp": warehouse.timestamp,
            "productdata": [
                {
                    "productID": p.productID,
                    "productName": p.productName,
                    "productCategory": p.productCategory,
                    "productQuantity": p.productQuantity,
                    "productUnit": p.productUnit,
                }
                for p in warehouse.productdata
            ]
        }

    print(pretty_json(output))

    print("\n===== Server Response =====")
    print("Message:", response.message)


if __name__ == "__main__":
    main()
```