

gRPC

gRPC egy modern, nyílt forráskódú távoli eljárásívást (Remote Procedure Call - RPC) megvalósító általános keretrendszer, amelyet a Google fejlesztett. Lehetővé teszi, hogy különböző platformokon és nyelveken írt alkalmazások egyszerűen kommunikáljanak egymással. gRPC protokollja alapértelmezetten Protocol Buffers-t (protobuf) használ.

Alapvető koncepciók:

- Szolgáltatások és metódusok: Egy .proto fájlban határozzuk meg a szolgáltatásokat, illetve azok távoli hívható metódusait.
- Kliens és szerver oldali kódgenerálás: Protobuf alapján automatikusan létrejönnek az interfészek és adattípusok.

Mintapélda

Hozzuk létre egy virtuális környezetet:

```
python -m virtualenv ./venv
```

Majd aktiváljuk:

```
./venv/Scripts/activate
```

Telepítsük a függőségeket:

```
python -m pip install grpcio  
python -m pip install grpcio-tools
```

Hozzuk létre a ./proto könyvtárat és benne a IDL fájlt helloworld.proto néven:

```
syntax = "proto3";  
  
// The greeting service definition.  
service Greeter {  
  // Sends a greeting  
  rpc SayHello (HelloRequest) returns (HelloReply) {}  
  // Sends another greeting  
  rpc SayHelloAgain (HelloRequest) returns (HelloReply) {}  
}  
  
// The request message containing the user's name.  
message HelloRequest {  
  string name = 1;  
}  
  
// The response message containing the greetings
```

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

Futtassuk a stub generátort az alábbi paranccsal:

```
python -m grpc_tools.protoc -I ./protos/ --grpc_python_out=. --python_out=. .\protos\helloworld.proto
```

A helloworld_pb2.py és helloworld_pb2_grpc.py fájlok létrejönnek a gyökérben.

A kliens és a szerver kódja az alábbi lesz:

greeter_client.py

```
import grpc
import helloworld_pb2
import helloworld_pb2_grpc

def run():
    print("Will try to greet world ...")
    with grpc.insecure_channel("localhost:50051") as channel:
        stub = helloworld_pb2_grpc.GreeterStub(channel)
        response = stub.SayHello(helloworld_pb2>HelloRequest(name="you"))
    print("Greeter client received: " + response.message)
```

greeter_server.py

```
from concurrent import futures

import grpc
import helloworld_pb2
import helloworld_pb2_grpc

class Greeter(helloworld_pb2_grpc.GreeterServicer):
    def SayHello(self, request, context):
        return helloworld_pb2>HelloReply(message="Hello, %s!" % request.name)

def serve():
    port = "50051"
    server = grpc.server(futures.ThreadPoolExecutor(max_workers=10))
    helloworld_pb2_grpc.add_GreeterServicer_to_server(Greeter(), server)
    server.add_insecure_port("[::]:" + port)
    server.start()
    print("Server started, listening on " + port)
```

```
server.wait_for_termination()

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

Indítsuk el a szervert és klienst.

From:

<https://edu.iit.uni-miskolc.hu/> - Institute of Information Science - University of Miskolc

Permanent link:

https://edu.iit.uni-miskolc.hu/tanszek:oktatas:informacios_rendszerek_integralasa:grpc?rev=1742586634

Last update: **2025/03/21 19:50**

