A3_RPC : Internals of Application Servers

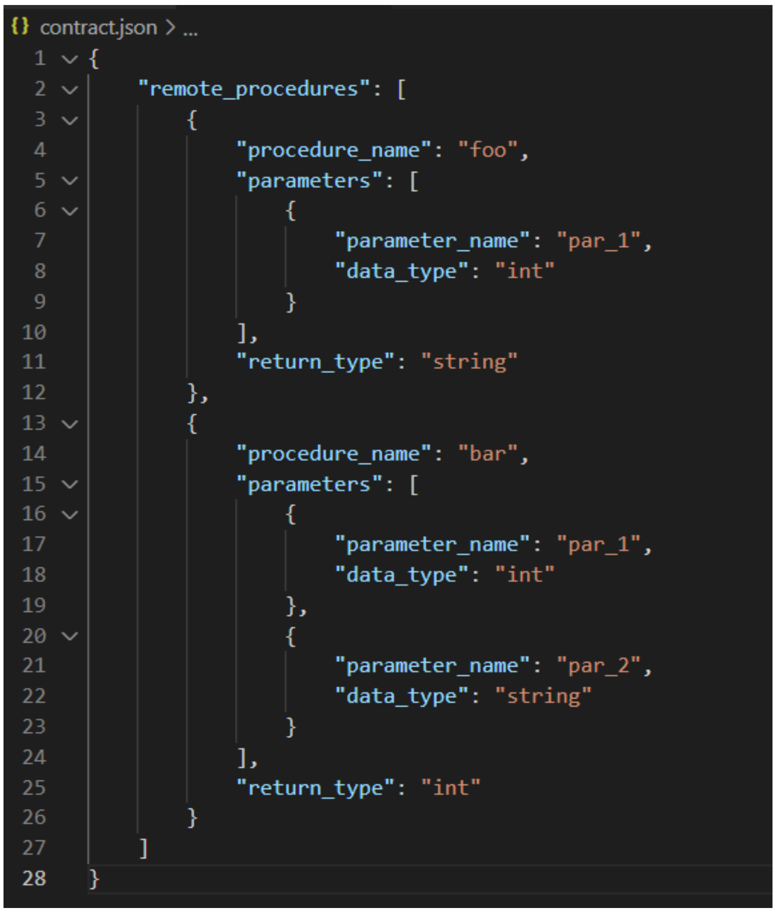
Remote Procedure Call

- Build a Simple Middleware System. That allows dynamically adding a service into the system and allows accessing the service from a client program.
- Design a RPC protocol consisting of Server Stub and Client Stub, responsible to run any given function present in server machine as shared library on client side. Communication protocol shold be socket only
- **Flask or Any other library are not allowed.**
- Protocol should be able to adapt any generic function and these functions should be usable from client machine using import functions.

A Simple walk through:

You will be given following files

- `Client.py`
- `server_precodeures.py`
- `contract.json`


contract.json > ...

```
{
    "remote_procedures": [
        {
            "procedure_name": "foo",
            "parameters": [
                {
                    "parameter_name": "par_1",
                    "data_type": "int"
                }
            ],
            "return_type": "string"
        },
        {
            "procedure_name": "bar",
            "parameters": [
                {
                    "parameter_name": "par_1",
                    "data_type": "int"
                },
                {
                    "parameter_name": "par_2",
                    "data_type": "string"
                }
            ],
            "return_type": "int"
        }
    ]
}
```

You have to make following files

- `code_generartor_client.py`
    - Will generate `rpc_client.py` based on contract provided to you
- `rpc_client.py`
    - Should contains client-side stub for every procedure defined in `contract.json`
- `code_generator_server.py`
    - will generate `rpc_server.py` based on contract provided to you
- `rpc_server.py`
    - Should contains server-side stub for every procedure defined in `contract.json`
    - Also generic listener which will listen the client stub request
    - Also function caller to call procedure residing in `server_precodeures.py`

Testing

- TA will run `code_generator_client.py` using following command

  ```
  python code_generator_client.py contract.json
  ```

  - This will generate the `rpc_client.py` which will have client side stubs for every procedures defined in `contract.json`

- TA will run `code_generator_server.py` using following command

  ```
  python code_generator_server.py contract.json
  ```

  - This will generate the `rpc_server.py` which will have server side stubs for every procedures defined in `contract.json`

- TA will run following commands after that

  ```
  Terminal 1
  > python rpc_server.py

  Terminal 2
  > python client.py
  ```

NOTE

Directory Structure

- All files will be put in same directory
- All files should be generated in same directory

client.py

- Will include `rpc_client.py` generated from `code_generator_client.py`

server.py

- it contains only procedure and its implementation
- which you will include in your `rpc_server.py` file auto generated from `code_generator_server.py`
- This will generate the `rpc_server.py` which will have client side stubs for every procedures defined in `contract.json`
- Also generic listener which will listen the client stub request
- Also function caller to call procedure residing in `server_precodeures.py`