In the name of God

RMI Exercise Report

Distributed Systems

Reza Sajedi

Fall 1400

GitHub

- This project has been published on GitHub.
- Instructions to use as a library or run the example is available.
- Anyway, the instructions have been also included in this report.

https://github.com/geraked/js-rmi

Design

- Based on HTTP protocol.
- Local and remote objects communicate using JSON messages.
- The developer doesn’t realize the background communication and feels like it’s a local object.
- Local objects POST three types of messages to the server to call methods, get properties and set properties of remote objects.
- The server puts the return value in a message and replies.
- When we say local object, we mean a proxy of actual remote object.

Design

- Multiple clients can connect to remote object.
- Inheritance and consistency has been considered.
- The binding of client and server is dynamic.
- No external library is used.
- The usage is very similar to Java RMI.
- The implementation consists of two parts: ServerStub, ClientStub

Design

- ServerStub is implemented in *lib/server.js* :

https://github.com/geraked/js-rmi/blob/master/lib/server.js

Design

- ClientStub is implemented in *lib/client.js* :

https://github.com/geraked/js-rmi/blob/master/lib/client.js

Design Examples of messages

```
human.looseWeight(5)
```

```
{
    "type": "method",
    "name": "looseWeight",
    "args": [5]
}
```

Design Examples of messages

```
human.age
```

```
{
    "type": "get",
    "name": "age"
}
```

Design Examples of messages

```
human.height = 1.80
```

```
{
    "type": "set",
    "name": "height",
    "value": 1.80
}
```

Strength

- Generalized to use as a library.
- Compiler works is not needed, thanks to JS.
- All types in JS are supported: string, number (float|integer), boolean, array, object, etc.
- Best practices in programming are followed.

Weaknesses

- It’s a simple library and maybe doesn’t provide advance features of complex libraries such as Java RMI, Pyro, etc.

Run the example application

- Make sure Node.js has been installed on your machine.
- Download the repository as ZIP or use the following command:

```
git clone https://github.com/geraked/js-rmi.git
```

- Go to the directory where the file *package.json* exists and execute the command:

```
npm run example
```

Client code

main.js

example > client > main.js > ...

```
1  import { IHuman } from "../shared/IHuman.js";
2  import { ClientStub } from "../../index.js";
3
4  let stub = new ClientStub('localhost', 3000);
5  let human = stub.lookup('/human', IHuman);
6
7  // Before manipulation
8  console.log('\x1b[44m%s\x1b[0m', '*** Before Manipulation ***');
9  console.log('toJSON:', await human.toJSON());
10 console.log('BMI:', await human.bmi());
11 console.log('\n');
12
13 // Manipulate
14 await (human.name = 'Reza');
15 await (human.height = 1.80);
16 await (human.age = await human.age + 3);
17 await human.looseWeight(5);
18
19 // After manipulation
20 console.log('\x1b[44m%s\x1b[0m', '*** After Manipulation ***');
21 console.log('toJSON:', await human.toJSON());
22 console.log('BMI:', await human.bmi());
23 console.log('\n');
24
25
26
```

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
Client

*** Before Manipulation ***
toJSON: { name: 'Amir', age: 17, weight: 67, height: 1.73 }
BMI: 22.386314277122523


*** After Manipulation ***
toJSON: { name: 'Reza', age: 20, weight: 62, height: 1.8 }
BMI: 19.1358024691358


PS G:\amir\ds\js-rmi>
```

Server code

main.js

example > server > main.js > ...

```
1  import { ServerStub } from '../../index.js'
2  import { Human } from './Human.js';
3
4  let stub = new ServerStub('localhost', 3000, true);
5  let human = new Human('Amir', 17, 67, 1.73);
6
7  stub.bind('/human', human);
8
9
10
```

Windows PowerShell

```
PS G:\amir\ds\js-rmi> npm run example
```

C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe

```
Server

Server running at http://localhost:3000/

 127.0.0.1 : 1409
{ type: 'method', name: 'toJSON', args: [] }
 127.0.0.1 : 1410
{ type: 'method', name: 'bmi', args: [] }
 127.0.0.1 : 1411
{ type: 'set', name: 'name', value: 'Reza' }
 127.0.0.1 : 1412
{ type: 'set', name: 'height', value: 1.8 }
 127.0.0.1 : 1413
{ type: 'get', name: 'age' }
 127.0.0.1 : 1415
{ type: 'method', name: 'looseWeight', args: [ 5 ] }
 127.0.0.1 : 1414
{ type: 'set', name: 'age', value: 20 }
 127.0.0.1 : 1416
{ type: 'method', name: 'toJSON', args: [] }
 127.0.0.1 : 1417
{ type: 'method', name: 'bmi', args: [] }
```

Use as a library The example application

- Install the library.
- Define your interface and share it between the server and clients.
- Implement the interface on the server.
- Create an object of that type on the server and bind it.
- Lookup the remote object from the client and use it.

Use as a library Install

- Execute the following command in the root of your npm project to get the library:

```
npm i https://github.com/geraked/js-rmi.git
```

Use as a library Define an interface

- Define your desired interface and share it between the server and clients.

https://github.com/geraked/js-rmi/blob/master/example/shared/IHuman.js

Use as a library Implement the interface

- Implement the interface on the server.

https://github.com/geraked/js-rmi/blob/master/example/server/Human.js

Use as a library Create remote object & bind

- Create an object of that type on the server and bind it.

```
import { ServerStub } from "rmi";
import { Human } from "./Human.js";

let stub = new ServerStub("localhost", 3000);
let human = new Human("Amir", 17, 67, 1.73);

stub.bind("/human", human);
```

Use as a library Lookup the remote object

- Lookup the remote object from the client and use it.

```
import { IHuman } from "../shared/IHuman.js";
import { ClientStub } from "rmi";

let stub = new ClientStub("localhost", 3000);
let human = stub.lookup("/human", IHuman);

console.log("toJSON:", await human.toJSON());
console.log("BMI:", await human.bmi());

// Manipulate
await (human.name = "Reza");
await (human.height = 1.8);
await (human.age = (await human.age) + 3);
await human.looseWeight(5);
```