

Pomelo

Node.js

node.js npm

. [Node.js](#)

. npm

```
$ npm install pomelo -g
```

git

Hello World

Hello World

```
$ pomelo init ./HelloWorld
```

```
$ mkdir HelloWorld
$ cd HelloWorld
$ pomelo init
```

HelloWorld

```
$ sh npm-install.sh
```

```
$ cd game-server
$ pomelo start --daemon
```

-daemon

command

가.

...

```
$ cd web-server
$ node app
```

node가 forever

<http://localhost:3001>

. 가

Test Game Server

'game server is ok'

Gate

Frontend load balancing connector
가 gate gate connector
gate.gateHandler load balancing .

```
// gateHandler.js
var connectors = this.app.getServersByType('connector');

// select connector, because more than one connector existed.
var dispatcher = require('/util/dispatcher');
var res = dispatcher.dispatch(uid, connectors);
// ...

// dispatcher.js
var crc = require('crc');
module.exports.dispatch = function(key, list) {
  var index = Math.abs(crc.crc32(key)) % list.length;
  return list[index];
};
```

- [Server scalability](#)

Gate connector servers.json port
. clientPort .

Connector

Client connection (session.bind()) 가
routing client backend backend
push connector client .
Connector client listen clientPort , backend server port

Gate , Connector .

- [Getting source code & installation](#)

Application

Application logic 가 client service gate,
connector frontend client . application backend

client

Backend

clientPort

.

frontend

rpc

Port

.

client

service

client

.

args

```
"game": [
  {
    "id": "game-server-1",
    "host": "127.0.0.1",
    "port": 3250,
    "args": " --debug=32315 "
  }
]
```

32315

Master

RPC invocation

Interprocess communication

backend

,

backend

RPC

.

Interprocess communication

frontend

```
app.rpc.game.gameRemote.request( routeParam, args, cb );
```

game

app/servers

gameRemote.js

,

gameRemote

game

remote

(

,

request

)

game/remote

.

pomelo

.

args

cb

rpc call

.

2

request

1

cb

.(cb

.)

2

1

cb

.)



args

...

... Converting circular structure to JSON

가

routeParams

route

.

session

user

id

,

route

.

((

)

Route

```
// app.js
var router = function(routeParam, msg, context, cb) {
  var gameServers = app.getServersByType('game');
```

```
var id = gameServers[routeParams% gameServers.length].id;
cb(null, id);
}

app.configure('production|development', function() {
  app.route('game', router); // router
});
```

servers.json game 가

```
"game": [
  {"id": "game-server-1", "host": "127.0.0.1", "port": 7000},
  {"id": "game-server-2", "host": "127.0.0.1", "port": 7001}
]time
```

app.route()

- [RPC Invocation](#)
- [Rpc framework](#)

Route, router

Route 가
. Javascript 가

```
window.pomelo.request( 'game.gameHandler.request', {
  protocol: ...
}, function( result ) {
  ...
} );
```

game , gameHandler handler , request . pomelo
app/servers game handler gameHandler.js
request

```
channel.pushMessage('onChat', param);
```

channel pomelo app channelService channel
param param

```
pomelo.on('onChat', function (msg) {
});
```

Session

```
{
  id : <session id> // readonly
  frontendId : <frontend server id> // readonly
  uid : <bound uid> // readonly
  settings : <key-value map> // read and write
  __socket__ : <raw_socket>
  __state__ : <session state>

  // ...
}
```

uid session.bind(value, callback) bind value

Channel

()
server-local , Server A Server B

Channel app.get('channelService').channels["channel "]
channels "channel " :

```
app.get('channelService').getChannel(roomname, true);
```

getChannel

```
ChannelService.prototype.getChannel = function(name, create) {
  var channel = this.channels[name];
  if(!channel && !!create) {
    channel = this.channels[name] = new Channel(name, this);
  }
  return channel;
};
```

Channel

Request, response, notify, push

Filter

Handler

Handler client logic . 가 .

```
handler.methodName = function(msg, session, next) {  
  // ...  
}
```

handler remote 가 remote rpc invocation

Error handler

Component

Protocol

Pomelo socket.io 0.3 TCP websocket binary
Connector 가 .
socket.io sioconnector TCP websocket hybridconnector가 . (
connector game-server/node_modules/pomelo/lib/connectors)

```
// app.js  
app.configure('production|development', 'connector', function(){  
  app.set('connectorConfig', {  
    connector: pomelo.connectors.hybridconnector,  
    heartbeat: 3,  
    useDict: true,  
    useProtobuf: true,  
    checkClient: function(type, version) {  
      // check the client type and version then return true or false  
    },  
    handshake: function(msg, cb){  
      cb(null, { /* message pass to client in handshake phase */ });  
    }  
  });  
});
```

- [Pomelo 0.3 new features](#)

Pomelo protocol

- Pomelo communications protocol

servers.json

cpu

0.5 () cpu .

```
{
  "development":{
    "connector":[
      {"id":"connector-server-1", "host":"127.0.0.1", "port":4050,
"clientPort": 3050, "frontend": true, "cpu": 2}
    ]
    "chat":[
      {"id":"chat-server-1", "host":"127.0.0.1", "port":6050, "cpu":
1}
    ]
    "gate":[
      {"id": "gate-server-1", "host": "127.0.0.1", "clientPort": 3014,
"frontend": true, "cpu": 3}
    ]
  }
}
```

, .

0.6 hybridconnector pomelo.init
encrypt 가 .

```
pomelo.init({
  host:'127.0.0.1',
  port:3014,
  encrypt:true
}, function() {
  // do something connected
});
```

app.js connectorConfig useCrypto 가 .

```
app.set('connectorConfig', {  
  connector: pomelo.connectors.hybridconnector,  
  heartbeat: 3,  
  useDict: true,  
  useProtobuf: true,  
  useCrypto: true  
});
```

Pomelo

1. Gate handler connector host, port
 2. Gate handler session connector
 3. connector host, port connector handler
 4. Connector handler session bind
- router connector handler session
route

-
- [Pomelo Wiki](#)
- [Promelo Wiki](#) ([Tutorial](#) 가 ?)

From:
<http://www.obg.co.kr/doku/> - **OBG WiKi**

Permanent link:
<http://www.obg.co.kr/doku/doku.php?id=programming:javascript:nodejs:pomelo>

Last update: **2020/11/29 14:09**

