## Tags»

- 조대협
- cloud
- 구글
- 클라우드
- 클라우드 컴퓨팅
- 초보
- 머신러닝
- 쿠버네티스
- 튜토리얼
- 강좌

more tags »

2020.07.31


평범하게 살고 싶은 월급쟁이 기술적인 토론 환영합니다.같이 이야기 하고 싶으시면 부담 말고 연락주세요:이메일-bwcho75골뱅이지메일 닷컴. 조대협

Category»

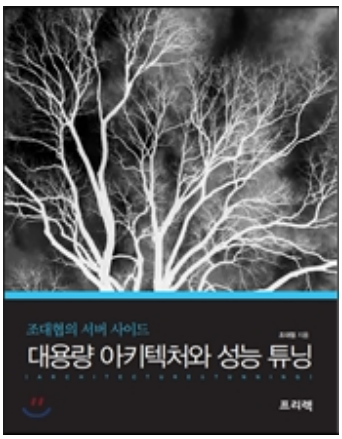
        - 모바일
    - 성능과 튜닝
        - JVM
        - APM (AP 성능 측정)
        - 자바 성능팁
        - WAS 튜닝
    - ALM
        - 애자일
        - 배포(Deployment)
        - JIRA
        - 에세이
        - SCM/VCS
        - Build Automation (빌..
        - Test Automation
        - Build Automation(이..
        - Task Management
    - 프로그래밍
        - 알고리즘
        - 안드로이드
        - Ruby
        - JavaScript
        - Python
        - Spring & Maven
        - LIBS
        - Hibernate(하이버네이..
        - 프로그래밍팁
        - MVC
        - XML 관련
        - J2EE
        - JSF & Oracle ADF Faces
        - Groovy
        - Visual Studio
        - C# & .NET
        - ASP.NET
        - Windows Phone7
        - 아두이노
    - 엔터프라이즈 솔루션
        - Wiki
        - 우분투
        - 포탈
        - Oracle BPEL
        - Oracle Service Bus (..
        - BEA Tuxedo
        - MS-SQL
        - SharePoint
        - BEA WebLogic
        - 빅데이타



## Contact»

페이스북 : https://www.facebook.com/terry.cho.7 Linked in:https://www.linkedin.com/profile/view?id=36267891

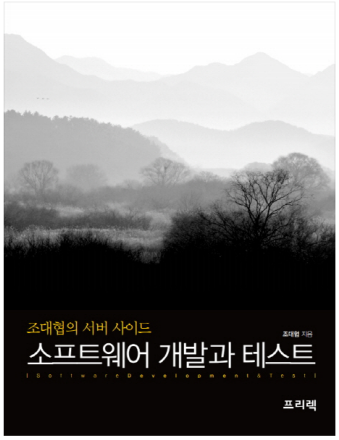
## Books




조대협의 서버사이드 #2

**대용량 아키텍처와 성능 튜닝**

아키텍처 설계 프로세스, 최신 레퍼런스 아키텍처 (SOA,MSA,대용량 실시간 분석 람다 아키텍처) REST API 디자인 가이드, 대용량 시스템 아키텍처, 성능 튜닝 및 병목 발견 방법




조대협의 서버사이드 #1

**소프트웨어 개발과 테스트**

애자일,소프트웨어 개발 조직,ALM과 테스트에 대한 이야기 JIRA,Git,Docker등에 대한 이야기와 함께.


빠르게 훑어보는 구글 클라우드 플랫폼

구글 컴퓨트엔진에 대한 소개와 설명 (무료 이북)

## Recent Post»

- 영어 발음 연습 방법.
- Prometheus 를 스케일링...
- 오픈소스 모니터링 툴 - P...
- 오픈소스 모니터링 툴 - P...
- 오픈소스 모니터링툴 - Pr... (1)

## Recent Comment»

- [승인대기].
- [승인대기].
- [승인대기].
- [승인대기].
- [승인대기].

## Recent Trackback»

- 2013년 오픈소스 슈퍼루키...
- [우분투 12.04] MongoDB...
- NoSQL과 트렌드..
- 코드 커버러지란??.
- 제가 만든 자바 트위터 포...

## Archive»

- 2020/02 (2)
- 2020/01 (3)
- 2019/12 (3)
- 2019/11 (7)
- 2019/10 (4)

## My Link»

- 서버사이드 아키텍트 그룹.
- Dzone.
- InfoQ.
- 마틴파울러 옹.
- Craig Larman 홈페이지.
- 강대명님(Redis) 블로그.
- 수학공부닷컴(중학교수준).
- Udacity.
- 커니의 안드로이드.
- 코드 스쿨.
- 랭귀지 튜토리얼.
- Code Academy.
- Coursera.
- 온라인강좌-Udemy.
- 데이타 과학 놀이터.
- 데이타 관련 튜토리얼.

# 빠르게 훑어 보는 node.js - #12 Socket.IO 4/4 - 채팅방 기능 추가하기

클라우드 컴퓨팅 & NoSQL/Vert.x & Node.js | 2014. 4. 24. 23:23 | Posted by 조대협

# 빠르게 훑어보는 node.js

#12 - Socket.IO (4/4)

조대협 (http://bcho.tistory.com)

### 채팅 프로그램에 방(room/그룹)의 기능을 추가하기

다음은 앞에서 만든 1:1 귓속말이 가능한 채팅에 "채팅방" 기능을 추가한 버전이다.

```
var express = require('express');
var routes = require('./routes');
var http = require('http');
var path = require('path');

var app = express();
```

```
app.use(express.bodyParser());
app.use(express.cookieParser('your secret here'));
app.use(express.session());
app.use(express.static(path.join(__dirname, 'public')));
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'ejs');
app.use(express.favicon());
app.use(express.logger('dev'));
app.use(express.json());
app.use(express.urlencoded());
app.use(express.methodOverride());
app.use(app.router);

var httpServer =http.createServer(app).listen(3000, function(req,res){
    console.log('Socket IO server has been started');
});
// upgrade http server to socket.io server
var io = require('socket.io').listen(httpServer);

var count = 0;
var rooms = [];

app.get('/:room',function(req,res){
    console.log('room name is :'+req.params.room);
    res.render('index',{room:req.params.room});
});



io.sockets.on('connection',function(socket){

    socket.on('joinroom',function(data){
        socket.join(data.room);

        socket.set('room', data.room,function() {
            var room = data.room;
            var nickname = '손님-'+count;
            socket.set('nickname',nickname,function(){
                socket.emit('changename', {nickname: nickname});

                // Create Room
                if (rooms[room] == undefined) {
                    console.log('room create :' + room);
                    rooms[room] = new Object();
                    rooms[room].socket_ids = new Object();
                }
                // Store current user's nickname and socket.id to MAP
                rooms[room].socket_ids[nickname] = socket.id

                // broad cast join message
                data = {msg: nickname + ' 님이 입장하셨습니다.'};
                io.sockets.in(room).emit('broadcast_msg', data);

                // broadcast changed user list in the room
                io.sockets.in(room).emit('userlist', {users:
Object.keys(rooms[room].socket_ids)});
                count++;
            });
        });

    });

    socket.on('changename',function(data){
        socket.get('room',function(err,room){
            socket.get('nickname',function(err,pre_nick) {
                var nickname = data.nickname;
                // if user changes name get previous nickname from nicknames MAP
                if (pre_nick != undefined) {
                    delete rooms[room].socket_ids[pre_nick];
                }
```

```
                rooms[room].socket_ids[nickname] = socket.id
                socket.set('nickname',nickname,function() {
                    data = {msg: pre_nick + ' 님이 ' + nickname + '으로 대화명을 변경
하셨습니다.'};
                    io.sockets.in(room).emit('broadcast_msg', data);

                    // send changed user nickname lists to clients
                    io.sockets.in(room).emit('userlist', {users:
Object.keys(rooms[room].socket_ids)});
                });
            });

        });
    });


    socket.on('disconnect',function(data){
        socket.get('room',function(err,room) {
            if(err) throw err;
            if(room != undefined
                && rooms[room] != undefined){

                socket.get('nickname',function(err,nickname) {
                    console.log('nickname ' + nickname + ' has been disconnected');
                    // 여기에 방을 나갔다는 메세지를 broad cast 하기
                    if (nickname != undefined) {
                        if (rooms[room].socket_ids != undefined
                            && rooms[room].socket_ids[nickname] != undefined)
                            delete rooms[room].socket_ids[nickname];
                    }// if
                    data = {msg: nickname + ' 님이 나가셨습니다.'};

                    io.sockets.in(room).emit('broadcast_msg', data);
                    io.sockets.in(room).emit('userlist', {users:
Object.keys(rooms[room].socket_ids)});
                });
            }
        }); //get
    });

    socket.on('send_msg',function(data){
        socket.get('room',function(err,room) {
            socket.get('nickname',function(err,nickname) {
                console.log('in send msg room is ' + room);
                data.msg = nickname + ' : ' + data.msg;
                if (data.to == 'ALL')
socket.broadcast.to(room).emit('broadcast_msg', data); // 자신을 제외하고 다른 클라이
언트에게 보냄
                else {
                    // 귓속말
                    socket_id = rooms[room].socket_ids[data.to];
                    if (socket_id != undefined) {

                        data.msg = '귓속말 :' + data.msg;
                        io.sockets.socket(socket_id).emit('broadcast_msg', data);
                    }// if
                }
                socket.emit('broadcast_msg', data);
            });
        });
    })
});
```

코드를 살펴보자

처음에 입장은 http://localhos:3000/{방이름} 으로 하게 된다.

```
app.get('/:room',function(req,res){
    console.log('room name is :'+req.params.room);
    res.render('index',{room:req.params.room});
});
```

그러면 URL에 있는 방이름을 받아서, index.ejs에 있는 UI로 채팅창을 띄워주고 방이름을 parameter로 index.ejs에 넘겨준다.

```
    socket.on('joinroom',function(data){
        socket.join(data.room);
```

클라이언트가 서버에 접속되면 맨 먼저 클라이언트가 join 이벤트를 보내는데, 이 join 이벤트를 받으면, 이때 같이 온 room 이름으로 현재 소켓을 room 이름의 room에 join한다.

```
        socket.set('room', data.room,function() {
```

다음으로, 해당 소켓이 어느 룸에 있는지 를 set 명령을 이용하여 socket에 저장해놓는다.

```
            var room = data.room;
            var nickname = '손님-'+count;
            socket.set('nickname',nickname,function(){
                socket.emit('changename', {nickname: nickname});

                // Create Room
                if (rooms[room] == undefined) {
                    console.log('room create :' + room);
                    rooms[room] = new Object();
                    rooms[room].socket_ids = new Object();
                }
```

윗부분이 room 데이터 객체를 생성하는 것인데, 앞의 예제와는 달리, 현재 연결된 클라이언트의 socket.id를 이제는 room 단위로 관리를 해야 한다. 그래서 rooms라는 객체를 이용하여, 해당 room에 대해서 rooms.room이라는 객체로 만들고, 그리고, 이 room에 현재 연결된 클라이언트 의 socket.id를 저장하는 socket_ids 객체를 생성한다.

```
                // Store current user's nickname and socket.id to MAP
                rooms[room].socket_ids[nickname] = socket.id
```

그리고 나서, socket_ids에 귓속말 채팅방 예제와 같이 nickname to socket.id 에 대한 맵핑 정보를 저장한다.

```
                // broad cast join message
                data = {msg: nickname + ' 님이 입장하셨습니다.'};
                io.sockets.in(room).emit('broadcast_msg', data);
// broadcast changed user list in the room
                io.sockets.in(room).emit('userlist', {users:
                      Object.keys(rooms[room].socket_ids)});
                count++;
            });
```

그리고 위와 같이 현재 room에 들어 있는 클라이언트들에게만 , 새로운 사용자가 입장했음을 알리고, 사용자 리스트를 업데이트하는 이벤트를 보낸다.

disconnect에 대한 부분도 크게 달라진 것이 없다. Socket_ids 객체가 rooms 아래로 들어갔고, 메시지를 보낼때, 귓속말 채팅방 예제가 io.sockets.emit 대신에, room의 범위를 지정하는 in() 메서드를 써서 io.sockets.in(room).emit와 같이 보내게 된다. Sendmsg 이벤트 부분도, broadcast하는 부분에서 to를 이용하여 다음과 같이socket.broadcast.to(room).emit 특정 room에 있는 클라이언트에게만 메시지를 보내는 것으로만 변경되었다

아래는 클라이언트쪽의 코드이다. 앞의 예제에서 나온 귓속말이 가능한 대화방과 코드가 거의 동일하다. 단

```
<script type="text/javascript">
        var socket = io.connect('http://localhost');
        socket.emit('joinroom',{room:'<%=room%>'});
```

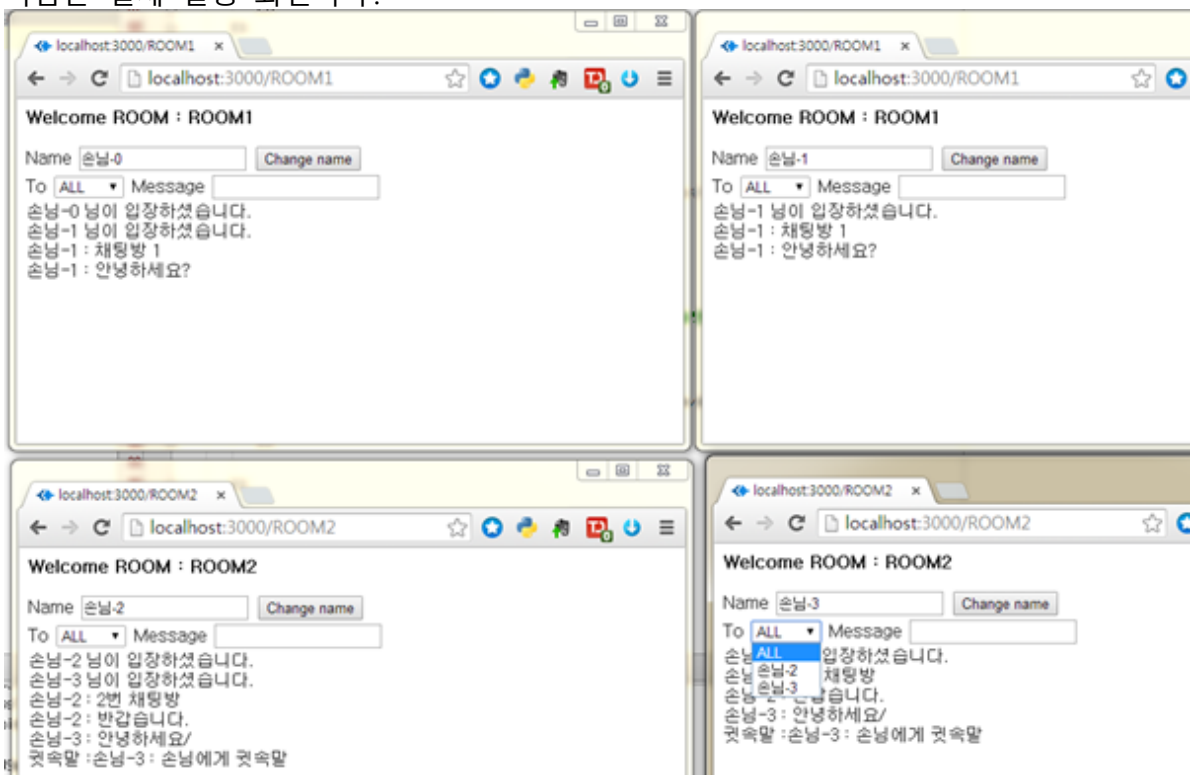
처음에 접속하였을 때, 서버 코드에서 방이름을 URL로부터 읽어서, 그 방이름으로 join 하는 이벤트를 보낸다.

/vies/index.ejs

```
 <html>
 <head>

     <title></title>
     <script src="/socket.io/socket.io.js"></script>
     <script src="//code.jquery.com/jquery-1.11.0.min.js"></script>

 </head>
 <body>

 <b>Welcome ROOM : <%= room%></b><p>
     Name <input type="text" id="nickname" /> <input type="button" id="changename"
 value="Change name"/><br>
     To
     <select id="to">
         <option value="ALL">ALL</option>
     </select>
     Message  <input type="text" id="msgbox"/>
     <br>
```

```
    <span id="msgs"></span>

    <script type="text/javascript">
        var socket = io.connect('http://localhost');
        socket.emit('joinroom',{room:'<%=room%>'});

        $('#changename').click(function(){
            socket.emit('changename',{nickname:$('#nickname').val()});
        });
        $("#msgbox").keyup(function(event) {
            if (event.which == 13) {
                socket.emit('send_msg',{to:$('#to').val(),msg:$('#msgbox').val()});
                $('#msgbox').val('');
            }
        });
        socket.on('new',function(data){
            console.log(data.nickname);
            $('#nickname').val(data.nickname);
        });

        // 새로운 사용자가 들어오거나, 사용자가 이름을 바꿨을때 "To" 리스트를 변경함
        socket.on('userlist',function(data){
            var users = data.users;
            console.log(users);
            console.log(data.users.length);
            $('#to').empty().append('<option value="ALL">ALL</option>');
            for(var i=0;i<data.users.length;i++){
                $('#to').append('<option value="'+users[i]+'">'+users[i]+"
</option>");
            }
        });

        socket.on('broadcast_msg',function(data){
            console.log(data.msg);
            $('#msgs').append(data.msg+'<BR>');
        });
    </script>
</body>
</html>
```

다음은 실제 실행 화면이다.



2 구독하기

'클라우드 컴퓨팅 & NoSQL > Vert.x & Node.js' 카테고리의 다른 글

| | |
|---|---|
| 빠르게 훑어 보는 node.js - #14 - Passport를 이용한 사용자 인증 구축 (5) | 2014.06.30 |
| 빠르게 훑어 보는 node.js - #13 Socket.IO 클러스터링 (1) | 2014.05.06 |
| 빠르게 훑어 보는 node.js - #12 Socket.IO 4/4 - 채팅방 기능 추가하기 (4) | 2014.04.24 |
| 빠르게 훑어 보는 node.js - #11 Socket.IO 3/4 (1:1 귓속말 구현) (3) | 2014.04.24 |
| 빠르게 훑어 보는 node.js - #10 Socket.IO 2/4 API 요약 (8) | 2014.04.24 |
| 빠르게 훑어 보는 node.js - #9 Socket.IO 1/4 - socket.io 기본 및 채팅 만들기 (16) | 2014.04.22 |



«이전 1 … 412 413 414 415 416 417 418 419 420 … 1221 다음»