NodeJS 5일차

몽고디비는 PK가 없어서 중복이 가능하다

\_id : ObjectId(" ") => \_는 스스로 만들어준다.

Socket.io

NodeJS 좋은점

1. MSA

=> Micro Service Architecture

Naver라는 포탈을 만든다.

카페 -> 패키지 -> 같은디비를 바라본다

메일 -> 패키지 -> 같은디비를 바라본다

웹툰 -> 패키지 -> 같은디비를 바라본다

=> 디비하나가 문제가 생기면 전체적인 시스템의 문제가 생겨버린다.

그것을 보안하기 위해 각각의 기능이 각각의 서비스를 할 수 있도록 하는방법

자기가 맡고 있는 기능말고는 다른 기능은 이용할 수 없다.

=> 다른기능을 이용하고 싶을 떄 api를 호출하여 기능을 동작시킨다.

회원 : Java spring => 회원만관리

카페 : php => 카페만 관리

메일 : Djongo => 메일만 관리

웹툰 : nodejs

=> 각각의 서비스를 다른 프로그램을 사용해서 제작할 수가 있다.

nosql을 잘 지원하는 것이 nodejs

2. Socket.io 사용하기 위해 node사용

자바8부터 웹소켓을 지원

spring-web-socket 모듈로 소켓사용 => 설정도 오래걸리고 성능이 그리 좋지않다.

소켓이란 ?

client 와 server가 통신하는 CS Programming

=> www

브라우저 HTTP 서버 1회성 통신

<- TCP/IP 연결하고->

-> 브라우저가 요청하면

<- 서버가 응답한다

<- TCP/IP 연결해제->

Sockjs Spring에서 지원하는 자바스크립트.

html5은 web socket을 공식적으로 지원하는 언어 => 자리잡는데 5~6년걸림

http(1.1) 는 web socket을 지원하지 않는다.

http(2.0)에서 web socket을 지원 => MEAN stack

M : 몽고디비, E : Express, A : anguler, N : Node

node혼자서는 소켓을 사용하지못하고 angular와 같이 사용해야 가능하다.

angular는 모든 통신을 소켓으로만 통신 => 페이지가 바뀌지않고 모든 서비스를 다 할 수 있다.

express에서는 socket.io 지원을 안한다.

소켓은 http에서 지원한다.

http는 express에서 지원한다. => 3개의 연결을 통해 소켓을 이용할수 있다.

구형브라우저는 WebSocket 대신 long polling 으로 사용

npm install --save socket.io

socket\_test 폴더생성

npm init 설치 후



pack.json 스크립트 수정

"scripts": {

"start" : "node index.js"

},

index.js 생성

const http = require("http");

const socketio = require("socket.io");

const fs = require("fs");

const express = require("express");

const app = express();

//서버 실행

const server = http.createServer(app);

//http에 서버를 올린다.

server.listen(3000,function(){

console.log("Server running at http://localhost:3000");

});

app.use("/js",express.static(\_\_dirname + "/resources/js"));

app.get("/", (req,res) => {//채팅페이지 보여주기

fs.readFile("view/chat.html","utf-8", (error,data) => {

//콜백함수

res.type("text.html");

res.send(data);

});

});

//socket 서버 실행하기

const io = socketio.listen(server);

//브라우저가 접속 했을 떄 실행될 이벤트

io.sockets.on("connection",(socket) => { //정의가 되어 있는 이벤트명 변경이 불가능

//socket : 방금 막 접속한 소켓

console.log(socket.id); //모든 브라우저는 자기만의 ID를 가지고 들어온다.

});

resources/js 폴더 생성후 자바스크립트 넣어주기'

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

<script src="/js/jquery-3.2.1.min.js" charset="utf-8"></script>

<script src="/socket.io/socket.io.js" charset="utf-8"> </script>

<script type="text/javascript">

//페이지가 로딩되어지면

$().ready(function(){

//웹소켓에 접속

var socket = io.connect();

});

</script>

</head>

<body>

<input type="text" id="message">

<input type="button" value="전송">

<div class="chat">

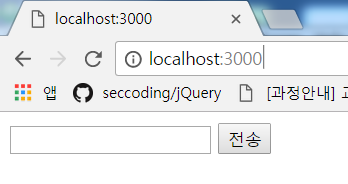
<!-- 텍스트박스에 데이터 입력 후 전송후하면 chat div에 보여주기 -->

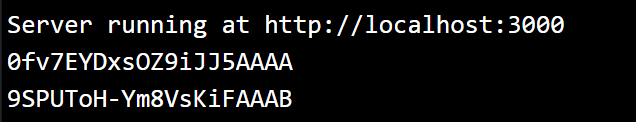
</div>

</body>

</html>

view 폴더생성 후 chat.html 생성





**혼자서 채팅하기**

index.js 수정

const http = require("http");

const socketio = require("socket.io");

const fs = require("fs");

const express = require("express");

const app = express();

//서버 실행

const server = http.createServer(app);

//http에 서버를 올린다.

server.listen(3000,function(){

console.log("Server running at http://localhost:3000");

});

app.use("/js",express.static(\_\_dirname + "/resources/js"));

app.get("/", (req,res) => {//채팅페이지 보여주기

fs.readFile("view/chat.html","utf-8", (error,data) => {

//콜백함수

res.type("text.html");

res.send(data);

});

});

//socket 서버 실행하기

const io = socketio.listen(server);

//브라우저가 접속 했을 떄 실행될 이벤트

io.sockets.on("connection",(socket) => { //정의가 되어 있는 이벤트명 변경이 불가능

//socket : 방금 막 접속한 소켓

console.log(socket.id); //모든 브라우저는 자기만의 ID를 가지고 들어온다.

socket.on("boardcast",(data) => {

console.log(data);

socket.emit("receive","message : " + data);

//소켓에 있는것을 실행시켜라. data를 전달

});

});

chat.html 수정

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

<script src="/js/jquery-3.2.1.min.js" charset="utf-8"></script>

<script src="/socket.io/socket.io.js" charset="utf-8"> </script>

<script type="text/javascript">

//페이지가 로딩되어지면

$().ready(function(){

//웹소켓에 접속

var socket = io.connect();

socket.on("receive",function(data){

var message = $("<p>" + data +"</p>"); //객체생성

$(".chat").prepend(message);

});

$("input[type=button]").click(function(){//버튼을 클릭하면

socket.emit("boardcast", $("input[type=text]").val() );

//emit 데이터를 전달

});

});

</script>

</head>

<body>

<input type="text" id="message">

<input type="button" value="전송">

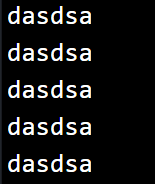
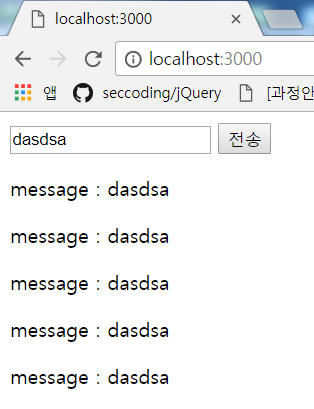
<div class="chat">

<!-- 텍스트박스에 데이터 입력 후 전송후하면 chat div에 보여주기 -->

</div>

</body>

</html>



**index.js**

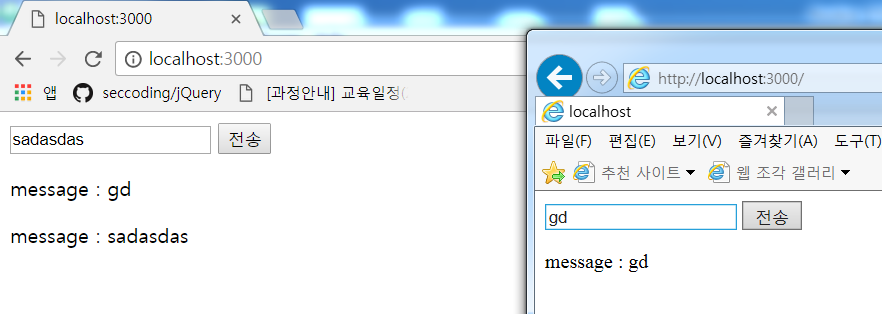
//socket.emit("receive","message : " + data);

//소켓에 있는것을 실행시켜라. data를 전달

io.sockets.emit("receive","message : " + data);

//소켓에 접속한 모두에게 데이터를 보낸다.

**=> socket.emit을 io.socket.emit로 변경**



=> 가장 심플하게 만들 수 있는 것이 캐치마인드

방을 만들어서 구분시켜서 데이터 보내기

index.js

const http = require("http");

const socketio = require("socket.io");

const fs = require("fs");

const express = require("express");

const app = express();

//서버 실행

const server = http.createServer(app);

//http에 서버를 올린다.

server.listen(3000,function(){

console.log("Server running at http://localhost:3000");

});

app.use("/js",express.static(\_\_dirname + "/resources/js"));

app.get("/", (req,res) => {//채팅페이지 보여주기

fs.readFile("view/chat.html","utf-8", (error,data) => {

//콜백함수

res.type("text.html");

res.send(data);

});

});

//socket 서버 실행하기

const io = socketio.listen(server);

//브라우저가 접속 했을 떄 실행될 이벤트

io.sockets.on("connection",(socket) => { //정의가 되어 있는 이벤트명 변경이 불가능

//socket : 방금 막 접속한 소켓

console.log(socket.id); //모든 브라우저는 자기만의 ID를 가지고 들어온다.

socket.on("join", (data) => { // 리터널 객체로 데이터를 받는다.

socket.join(data.room); //join은 원래 존재하는 function

//방에 들어가라

//방에 있는 모든사람에게 누가 입장햇는지 보여주기

io.sockets.in(data.room).emit("receive",{

"type" : "notice",

"message" : data.name + "님이 입장했습니다."

});//방에 있는 모두에게 데이터를 전달

});

socket.on("boardcast",(data) => {

console.log(data);

//socket.emit("receive","message : " + data);

//소켓에 있는것을 실행시켜라. data를 전달

io.sockets.emit("receive","message : " + data);

//소켓에 접속한 모두에게 데이터를 보낸다.

});

});

chat.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

<script src="/js/jquery-3.2.1.min.js" charset="utf-8"></script>

<script src="/socket.io/socket.io.js" charset="utf-8"> </script>

<script type="text/javascript">

//페이지가 로딩되어지면

$().ready(function(){

var room = prompt("접속할 방 이름을 입력하세요.");

var name = prompt("사용할 대화명을 입력하세요.");

//웹소켓에 접속

var socket = io.connect();

socket.emit("join",{

//index.js의 socket.on("join") 실행 join이벤트를 실행 리터널 데이터를 받는다

"room" : room,

"name" : name

});

socket.on("receive",function(data){

if(data.type == "notice"){

var message = $("<p>" + data.message +"</p>"); //객체생성

$(".chat").prepend(message);

}else if(data.type =="chat"){

var message = $("<p>" + data.name +" | " + data.message +"</p>"); //객체생성

$(".chat").prepend(message);

}

});

$("input[type=button]").click(function(){//버튼을 클릭하면

socket.emit("boardcast", $("input[type=text]").val() );

//emit 데이터를 전달

});

});

</script>

</head>

<body>

<input type="text" id="message">

<input type="button" value="전송">

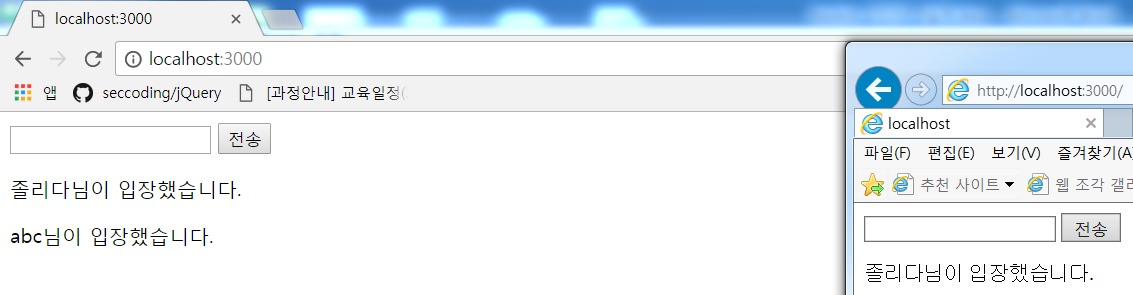
<div class="chat">

<!-- 텍스트박스에 데이터 입력 후 전송후하면 chat div에 보여주기 -->

</div>

</body>

</html>



데이터 불균형이 다르기 떔에 한쪽에서만 데이터를 받을 수 있기 때문에 데이터를

맞춰주어야한다.

index.js

const http = require("http");

const socketio = require("socket.io");

const fs = require("fs");

const express = require("express");

const app = express();

//서버 실행

const server = http.createServer(app);

//http에 서버를 올린다.

server.listen(3000,function(){

console.log("Server running at http://localhost:3000");

});

app.use("/js",express.static(\_\_dirname + "/resources/js"));

app.get("/", (req,res) => {//채팅페이지 보여주기

fs.readFile("view/chat.html","utf-8", (error,data) => {

//콜백함수

res.type("text.html");

res.send(data);

});

});

//socket 서버 실행하기

const io = socketio.listen(server);

//브라우저가 접속 했을 떄 실행될 이벤트

io.sockets.on("connection",(socket) => { //정의가 되어 있는 이벤트명 변경이 불가능

//socket : 방금 막 접속한 소켓

console.log(socket.id); //모든 브라우저는 자기만의 ID를 가지고 들어온다.

socket.on("join", (data) => { // 리터널 객체로 데이터를 받는다.

socket.join(data.room); //join은 원래 존재하는 function

//방에 들어가라

//방에 있는 모든사람에게 누가 입장햇는지 보여주기

io.sockets.in(data.room).emit("receive",{

"type" : "notice",

"message" : data.name + "님이 입장했습니다."

});//방에 있는 모두에게 데이터를 전달

});

socket.on("boardcast",(data) => {

console.log(data);

//socket.emit("receive","message : " + data);

//소켓에 있는것을 실행시켜라. data자체를 전달

io.sockets.in(data.room).emit("receive",data);

//소켓에 접속한 모두에게 데이터를 보낸다.

});

});

chat.html

<!DOCTYPE html>

<html>

<head>

<meta charset="utf-8">

<title></title>

<script src="/js/jquery-3.2.1.min.js" charset="utf-8"></script>

<script src="/socket.io/socket.io.js" charset="utf-8"> </script>

<script type="text/javascript">

//페이지가 로딩되어지면

$().ready(function(){

var room = prompt("접속할 방 이름을 입력하세요.");

var name = prompt("사용할 대화명을 입력하세요.");

//웹소켓에 접속

var socket = io.connect();

socket.emit("join",{

//index.js의 socket.on("join") 실행 join이벤트를 실행 리터널 데이터를 받는다

"room" : room,

"name" : name

});

socket.on("receive",function(data){

if(data.type == "notice"){

var message = $("<p>" + data.message +"</p>"); //객체생성

$(".chat").prepend(message);

}else if(data.type =="chat"){

var message = $("<p>" + data.name +" | " + data.message +"</p>"); //객체생성

$(".chat").prepend(message);

}

});

$("input[type=button]").click(function(){//버튼을 클릭하면

socket.emit("boardcast", {

"type": "chat",

"room" : room,

"name" : name,

"message" : $("input[type=text]").val()

});

//emit 데이터를 전달

});

});

</script>

</head>

<body>

<input type="text" id="message">

<input type="button" value="전송">

<div class="chat">

<!-- 텍스트박스에 데이터 입력 후 전송후하면 chat div에 보여주기 -->

</div>

</body>

</html>

