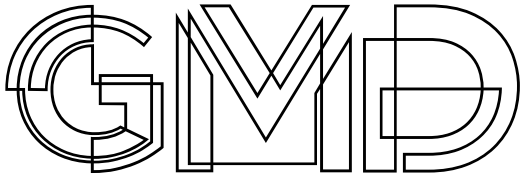


박성호(3417)의 활동 일지

ver. 2015.7 학습동아리 발표대회

2015 년 5 월 9 일

node.js 공부 시작 및 작업 환경 구축

윈도우 7 에서 node.js 개발 환경을 구축했다. IDE 는 Eclipse 를 사용하기로 했다. Eclipse 의 plug-in 인 nodeclipse 를 통해 Exprss 와 node.js 의 binary 를 Eclipse 에 연결했다. 원래 nodeclipse 를 설치하면 기본적으로 딸려오는 node.js binary 를 Eclipse 에서 사용하게 되는데 이렇게 하면 node.js 업데이트 등 보수 작업을 할 때 불편한 점이 많을 듯 해서 node.js 공식 웹 사이트에서 배포하는 node.js 를 사용하기 위해 위와 같은 작업을 한 것이다. 이런 작업 과정을 인터넷에서 찾지 못했기에 이런 방법을 직접 찾았다.

2015 년 5 월 10 일

fs, os, process, url, querystring 모듈 사용법을 익힘

node.js 에는 기본적으로 제공해주는 global module 이 있는데 이들의 사용법을 익혀야지 프로그래밍 할 때 유용하다고 한다. 열심히 익혔다.

2015 년 5 월 11 일

Event driven programming model 의 개념을 알게 됨

node.js 는 event driven programming model 을 차용했다. 이는 내가 그 동안 익숙했던 프로그래밍 방식인 procedural programming model 과 차이가 있다. procedural programming model 은 말 그대로 소스 파일에 쓰여진 코드의 순서대로 프로그램이 진행되는 것이다. 하지만 event driven programming model 는 특정 이벤트가 발생하면 특정 이벤트에 맵핑된 함수가 실행되는 방식으로 프로그램이 진행된다.

또한 node.js 는 비동기식 I/O 를 사용하는데 이는 프로그램에 I/O 요청을 보내고 응답이 올 때까지 프로그램이 멈추는 기존의 방식이 아닌, I/O 요청을 보내고 바로 다음 코드로 프로그램을 진행시킨다. I/O 응답은 응답에 맵핑해 놓은 함수가 실행된다. (이 부분에서 event driven programming model 의 장점이 나타나는 것이다). 이렇게 맵핑해 놓고 실행되는 함수를 callback 함수라고 한다.

2015 년 5 월 12 일

socket.io 모듈을 이용해 실시간 채팅 구현

내 블로그에 올린 socket.io 에 대해 정리한 글을 인용하겠다.

“

웹 소켓의 등장 배경

초창기 웹은 콘텐츠를 전달하는 역할만을 수행했다. 그래서 초창기 CERN 과 같은 연구기관에서 사용했던 것과 같은 전형적인 브라우저 랜덩링 방식인 HTTP 요청에 대한 HTTP 응답을 받아서 브라우저의 화면을 모두 지우고 받은 내용을 새로 표시하는 방식을 사용해도 문제가 없었다.

하지만 Ajax 와 같이 사용자와 긴밀히 상호작용하는 RIA(Rich Internet Application)와 같은 웹 서비스가 발달하면서 웹 소켓이 등장하게 되었다. 이러한 방식은 숨겨진 프레임(Hidden Frame)을 이용하거나 Long Polling, Stream 등의 방법을 사용했다. 이는 브라우저가 HTTP 요청을 보내고 웹 서버가 이 요청에 대해서 웹 서버가 HTTP 응답을 보내는 단방향의 메시지 교환 방식을 유지하는 선에서 구현되는 방식이다. 이것은 기존의 방법에 일종의 트릭을 사용한 방법이기 때문에 기존의 웹 기술을 이용하여 실시간 웹 서비스를 만드는 일은 복잡하고 어려웠다.

이러한 불편함을 해소하고 사용자와 서버가 긴밀히 상호작용하는 웹 페이지를 만들기 위해 자유로운 양방향 메시지 송수신 방법으로서 HTML5 표준안의 일부인 웹 소켓 API 가 등장했다. 하지만 아직까지 웹 소켓 프로토콜은 확정된 상태가 아니다.

”

socket.io 모듈은 WebSocket 을 지원함과 동시에 WebSocket 을 지원하지 않는 브라우저에 대해서도 AJAX Long Polling, MultiPart Streaming, Iframe 을 이용한 푸쉬와 JSON Polling, Flash Socket 등의 방법을 사용해 푸쉬 메시지를 일관되게 보내는 것을 지원한다.

간단히 이야기하면 socket.io 모듈을 통해 실시간 통신을 구현할 수 있게 된 것이다.

socket.io 를 이용해 다음과 같이 간단하게 채팅이 구현된 서버와 웹 사이트를 제작했다. 코드는 핵심 부분만 옮겨놨다.

app.js

```
var io = require('socket.io').listen(server);

io.sockets.on('connection', function(socket){
    socket.emit('toclient', {msg:'Welcome!!'});
    socket.on('fromclient', function(data){
        socket.braodcast.emit('toclient', data);
        socket.emit('toclient', data);
        console.log('Message from client : ' + data.msg);
    });
});
```

test.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>Send message</b><p>

    Message <input type="text" id="msgbox"/>

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

    <script type="text/javascript">

        var socket = io.connect('http://200.200.100.149:3000');

        $("#msgbox").keyup(function(event) {
            //press enter
            if (event.which == 13) {
                socket.emit('fromclient', {msg : $('#msgbox').val()});
                $('#msgbox').val('');
            }
        });

        socket.on('toclient', function(data){
            console.log(data.msg);
            //$('#msgs').append(data.msg + '<BR>');
            $('#msgs').prepend(data.msg + '<BR>');
        });

    </script>
</body>
```

2015 년 5 월 13 일 ~ 5 월 14 일

귓속말 채팅 구현

socket.io 는 각 클라이언트마다 아이디를 부여해주는데 이를 통해 클라이언트를 구별할 수 있다. 이를 이용해 위에서 구현한 채팅 프로그램에서 아이디를 구별해 메시지를 송신하는 기능을 추가하면 귓속말을 구현할 수 있다.

```
socket.on('send_msg', function(data){
	var toSendNickname = data.toSend;
	//data.msg = toSendNickname + ' : ' + data.msg;

	//전체에게 메시지 전달
	if(data.toSend == 'ALL')
		io.sockets.emit('get_msg_all', data);

	//특정 사용자에게만 메시지 전달
	else{
		var toSendId = socket_ids[data.toSend];
		if(toSendId != undefined){
			io.to(toSendId).emit('get_msg', data);
			socket.emit('get_msg', data);
		}
	}
});
```

2015 년 5 월 15 일

마피아 게임에 대한 계획 구상

우연찮게 친구들과 이야기하는 중에 마피아 게임 이야기가 나왔고 채팅 서버 구현이 막 끝난 타이밍이라 '심정현', '강석훈'과 마피아 게임을 구현해 보기로 하였다. 게임의 구현은 다음과 같이 하기로 하였다. 아래 사항은 개발 중에 계속 변경하기로 했다. '심정현', '강석훈'은 Silver Fish SNS 개발 및 논문 집필 때문에 7 월 모의고사 이후부터 참여하기로 하고 그 때까지는 내가 혼자 하기로 했다.

- 마피아, 경찰, 시민, 의사가 기본적인 직업이다. (후에 많은 직업들을 추가할 예정)
- 회원가입을 구현한다. 이를 통해 사용자를 구분한다.
- 게임을 하기 전 사용자들이 머무는 로비를 구현한다.
- 게임이 이루어지는 공간인 '방'을 구현한다.
- 게임은 '방'에서 채팅으로 진행되며 게임이 끝나는 조건은 다음과 같다.
  - ✓ 마피아가 모두 죽는다. → 시민 승리
  - ✓ 시민이 모두 죽는다. → 마피아 승리
  - ✓ 두 명이 남았는데 이들은 마피아와 마피아가 아닌 사람이다. → 마피아 승리

- 그많데 SNS 인 Silver Fish 와 연동한다. Silver Fish 가입자는 마피아 게임에 로그인이 가능하며 자신의 정보를 그대로 마피아 게임 서버가 사용할 수 있게 한다.
- (5 월 30 일 추가) 게임은 '도시'라는 공간에서 진행된다. '도시'는 그래픽을 통해 구현되며 사용자들은 '도시'를 자유롭게 돌아다닐 수 있다. 그래픽은 게임의 모든 요소가 개발된 후에 추가하기로 함.
- (5 월 30 일 추가) '뉴스'가 사용자들에게 보이도록 한다. '뉴스'는 '도시'에서 이루어지는 사건, 사소한 일들을 보도해준다. 보도되는 내용은 쓸모 있는 내용과 쓸모 없는 내용으로 구분한다. 사용자들은 뉴스의 쓸모 있는 내용들에서 유용한 정보를 추리할 수 있게 하며 이를 통해 마피아의 정체와 같은 게임의 요소들을 추리할 수 있게 한다.

2015 년 5 월 16 일 ~ 5 월 23 일

마피아 게임의 기본적인 틀 구현

UI, Database 를 제외하고 단순하게 게임의 진행이 이루어지는 웹 페이지와 서버 프로그램을 제작했다.

##
서버의 코드는 모두 합해 1232 줄 이므로 활동 일지에 옮겨놓기에는 너무 많아 활동일지에 옮겨 놓지 않았습니다. 첨부된 파일을 확인해주시길 바랍니다.
##

2015 년 5 월 24 일

Bootstrap 공부

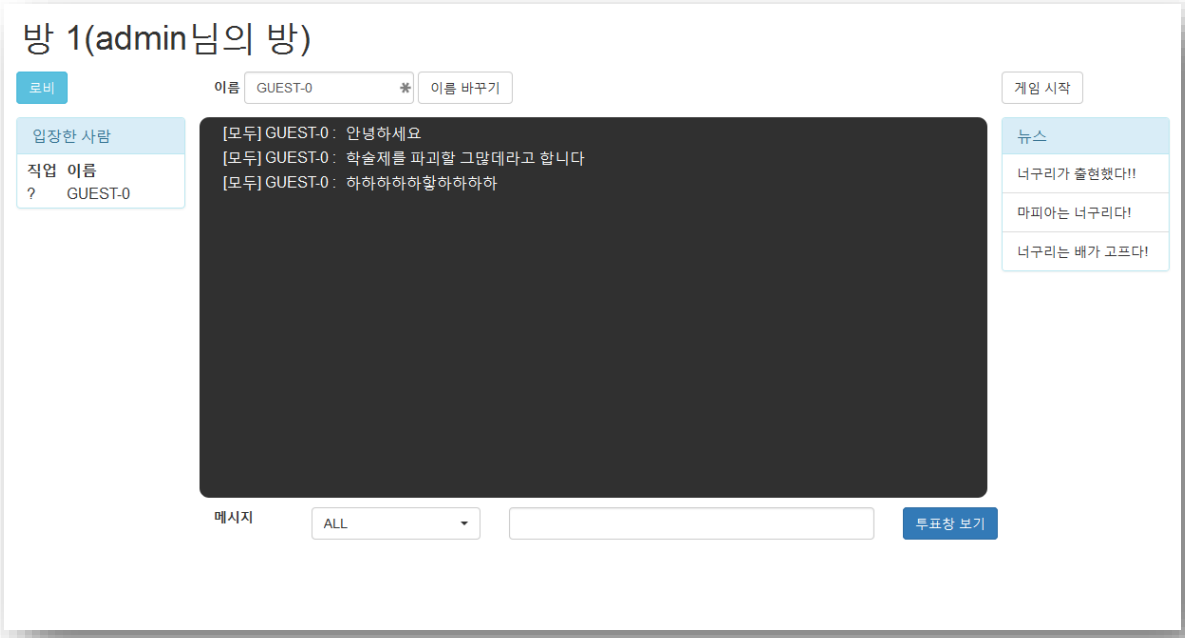
마피아 게임의 핵심적인 부분의 개발은 끝났지만 UI(User Interface)가 너무 촌스러웠다. 이쁘게 만들 방법을 찾던 중 Bootstrap 이라는 HTML/CSS/JS Framework 를 알게 되었다. 트위터도 사용하는 것이라 하길래 냉큼 '부트스트랩으로 디자인하라(앙용석)'이라는 책을 통해 공부를 시작했다.

2015 년 5 월 25 일 ~ 5 월 28 일

Bootstrap 을 통해 웹 페이지 디자인

Bootstrap 을 통해 가독성 있는 웹 페이지 디자인을 대략적으로 완료했다.

##
바뀌게 된 웹 페이지의 코드 또한 첨부된 파일을 확인해주시기 바랍니다.
##

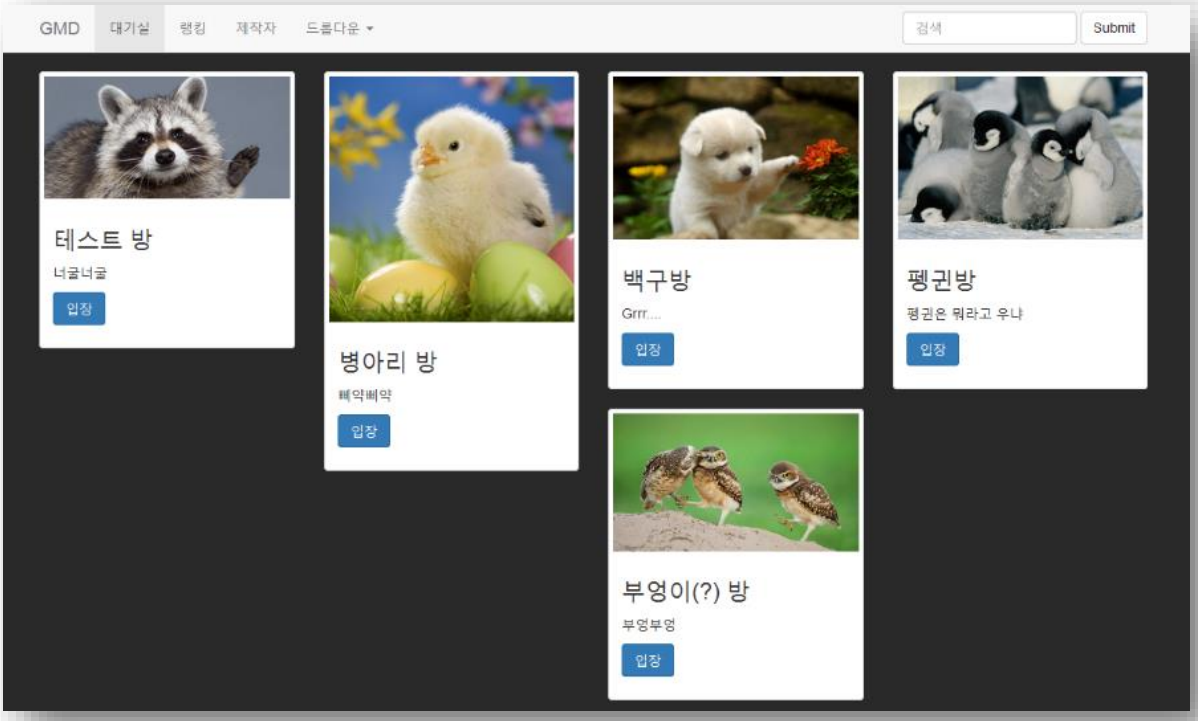


2015 년 6 월 4 일

로그인 페이지, 로비 페이지 구현 및 페이지 라우팅

로그인 페이지와 로비 페이지를 구현할 때는 처음부터 Bootstarp 을 사용해서 상당히 편했다.

##
로그인, 로비 페이지 코드는 첨부된 파일을 확인해주시기 바랍니다.
##



2015 년 6 월 5 일 ~ 7 월 9 일

시험 기간임을 감안해 활동 중지.

끝.