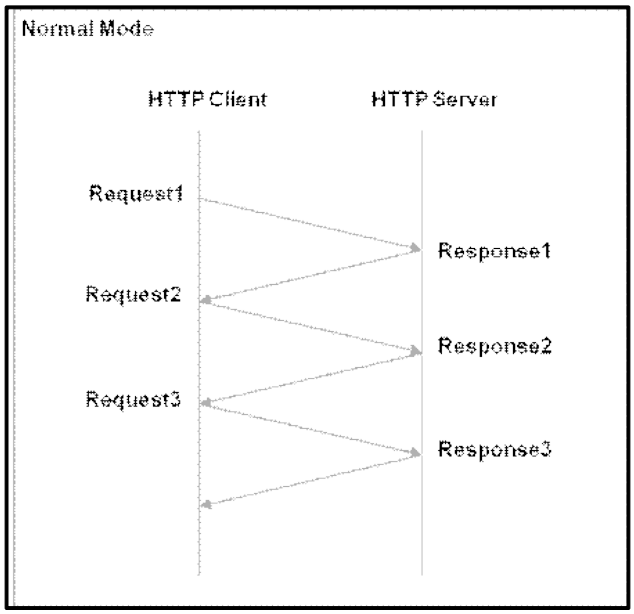
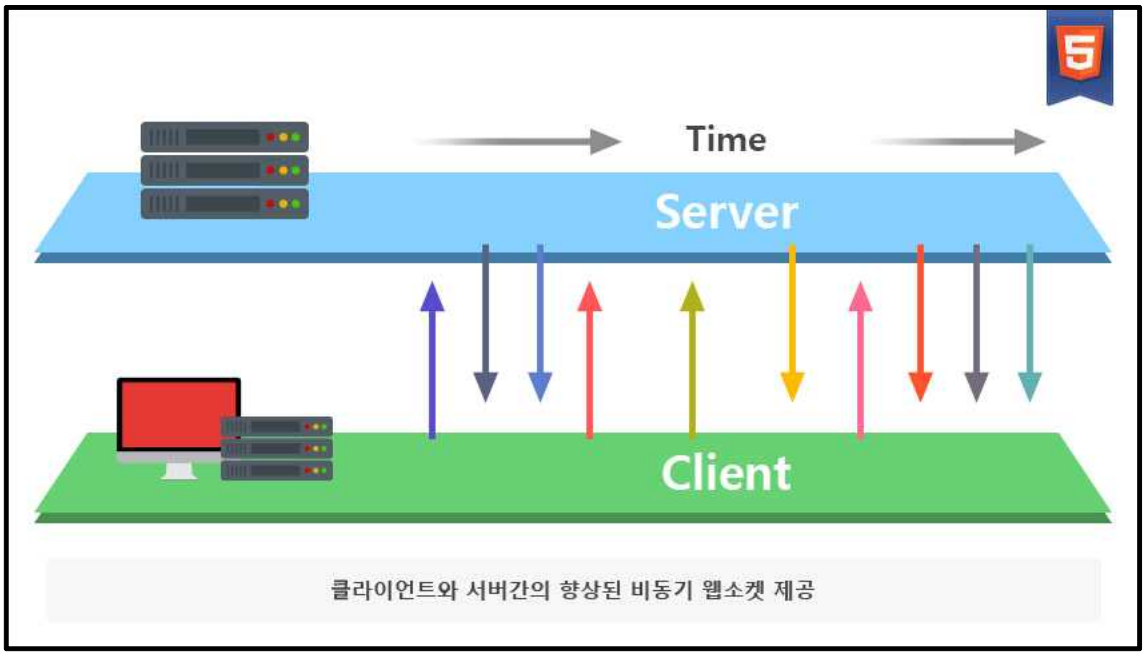
■ 웹소켓의 개념

1) 기존의 HTTP프로토콜



- 단방향 : 클라이언트가 요청하면 서버가 응답하는 방식

2)웹소켓 프로토콜



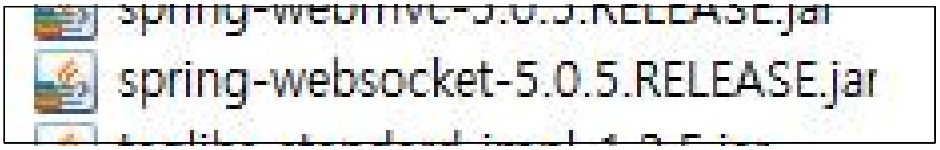
- 서버와 클라이언트가 양방향으로 통신 가능

■ 프론트 엔드(HTML / CSS / javascript) 개발

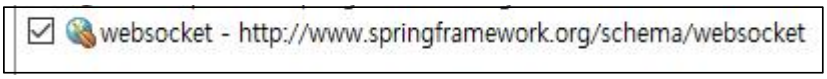
1) 오늘 날짜로 프로젝트 생성

- spring nature 변경

- 라이브러리 등록



- xxx-servlet.xml에 'websocket' 네임스페이스 등록



2) '/index' 만들기

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
        pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>웹소켓을 이용한 채팅 구현</title>
<link rel="stylesheet" href="css/reset.css" />
<link           href="https://fonts.googleapis.com/css?family=Do+Hyeon¦Yeon+Sung&amp;subset=korean"
rel="stylesheet">
<style>

html {
        overflow-y: scroll;
}

body {
        font:17px 'Do Hyeon', sans-serif;
        background:#F8F8F8;
```

```
}

#wrap {
        width:1200px;
        border:1px solid #4527a0;
        margin:auto;
        overflow: hidden;
}

#header {
        width:1200px;
        height:100px;
        border-bottom:1px solid #4527a0;
        background:#f8f8f8;
        top:0;
        z-index: 9;
}

#header h1 {
        font-size:60px;
        font-weight:900;
        line-height:100px;
        text-align: center;
        color:#4527a0;
}

#content {
        width:1200px;
        min-height:400px;
        position: relative;
}

#footer {
        border-top:1px solid #4527a0;
        width:1200px;
        height:100px;
        text-align: center;
        line-height:100px;
        font-weight:500;
        color:#4527a0;
        clear:both;

}
.btn {
        display:inline-block;
        border: 0;
        margin: 0;
```

```
        padding: 8px 14px;
        background: #ffc107;
        color: #424242;
        text-decoration: none;
        font-weight: bold;
        font: 500 17px 'Do Hyeon', sans-serif;
}

button.btn {
        cursor: pointer;
        font: bold 17px 'Do Hyeon', sans-serif;
}

.btn:hover {
        background: #c79100;
        box-shadow: 0 0 2px #333;
}



#nickname {
width: 300px;
font: 20px 'Do Hyeon', sans-serif;
}


#msg {
        width: 700px;
        font: 20px  'Do Hyeon', sans-serif;
        padding: 6px;
}

#chattingContentBox {
        width:950px;
        height: 600px;
        border-left: 1px solid #463A84;
        border-right: 1px solid #463A84;
        margin:auto;
        position: relative;
}

#userList {
        width:119px;
        border-right: 1px solid #463A84;
        height: 600px;
        position:absolute;
        left:0;
```

```
        top:0;
        overflow-y:scroll;
}
.user {
        width:89px;
        text-align: center;
        padding:5px;
}

.user.mine {
        background:#D1C4E9;
}

.user h3 {
        width:89px;
        overflow: hidden;
        text-overflow: ellipsis;
        white-space: nowrap;
        color:#5E35B1;
}

#chattingBox {
        width: 830px;
        height: 600px;
        position: absolute;
        overflow: hidden;
        right:0;
        top:0;
}

.box_form {
        width: 828px;
        height: 50px;
        position: absolute;
        bottom: 0;
        border-top: 1px solid #4527a0;
        line-height: 50px;
        text-align: center;
}

#chatList {
        width: 817px;
        height: 538px;
        overflow-y: scroll;
        background: #fff;
        box-shadow: 0 0 2px #333;
        margin: 6px;
```

```
}

#chatList ul li {
	padding: 8px;
}



#chatList .msg strong {
	font-weight: 500;
	color: #4527a0;
	font-size: 22px;
}

#chatList .card_user {
	left:10px;
	top:10px;
	position: absolute;
	text-align: center;
}

#chatList .mine .card_user {
	left:auto;
	right:10px;
	top:10px;
}

.card_user img {
	display: block;
	margin:auto;
}

#chatList li {
	padding: 10px;
	min-height: 100px;
	overflow: hidden;
	position: relative;
	padding-bottom: 20px;
}

#chatList .box_reply {
	width: 680px;
	min-height: 80px;
	/* 포지셔닝 컨텍스트 지정 */
	position: relative;
	padding-left: 20px;
	padding-top: 50px;
	left:90px;
```

```
        top:0;
}
#chatList .mine .box_reply {
        padding-right: 20px;
        padding-top: 50px;
        left:-20px;
        top:0;
        text-align: right;
}

#chatList .comments {
font-family: 'Yeon Sung', cursive;
        font-size:22px;
        max-width: 660px;
        min-height: 25px;
        background: #b5f4f2;
        border-radius: 0 12px 12px 12px;
        padding: 6px 10px;
        position: relative;
        color: #424242;
        white-space: pre-wrap;
        /* 단어를 브레이크 */
        word-wrap: break-word;
        display: inline-block;
}

#chatList .mine .comments {
        border-radius: 12px 0px 12px 12px;
}

#chatList .box_reply .time {
        top: 26px;
        left: 20px;
        position: absolute;
        font-size: 15px;
}
#chatList .mine .comments::before {
        right:-10px;
        left:inherit;
        border:none;
        border-bottom: 10px solid transparent;
        border-left: 10px solid #b5f4f2;
}

#chatList .comments::before {
        content: "";
        display: block;
```

```
        position: absolute;
        left: -10px;
        top: 0;
        width: 0;
        height: 0;
        border-bottom: 10px solid transparent;
        border-right: 10px solid #b5f4f2;
}

.ip {
        font-size:15px;
        color:#9575CD;
}

.msg span {
        color:#F44336;
}

img {
        width: 80px;
        height: 80px;
        border-radius: 40px;
        box-shadow: 0 0 4px #333;
}

#loginBox {
        width: 1200px;
        height: 600px;
        background: #4527a0;
        position: absolute;
        left: 0;
        top: 0;
        text-align: center;
        transition: .3s ease;
        z-index: 1;
}

#loginBox h2 {
        color: #fff;
}

#loginBox .box_img {
        position: absolute;
        width: 1200px;
        top: 150px;
        overflow: hidden;
}
```

```
#loginBox .box_img img {
	margin: 10px;
	cursor: pointer;
}

#loginBox .box_img img.off {
	opacity: .3;
}

#loginBox .box_btn {
	position: absolute;
	width: 1200px;
	height: 100px;
	bottom: 100px;
}
#loginBtn {
	font-size:20px;
}

#loader {
	width: 100%;
	height: 100%;
	position: fixed;
	left: 0;
	top: 0;
	background: url(../img/loader.gif) no-repeat center;
	z-index: 100;
	background-color: rgba(255, 255, 255, .6);
	display: none;
}

#closeBtn {
	position:absolute;
	right:10px;
	bottom:10px;
}

</style>
</head>
<body>
	<div id="wrap">
		<div id="header">
			<h1>웹소켓을 이용한 채팅</h1>
		</div><!-- //header -->
		<div id="content">
			<div id="chattingContentBox">
```

```
                                <ul id="userList">
                                </ul>
                                <div id="chattingBox">
                                        <div id="chatList">
                                                <ul>
                                                </ul>
                                        </div><!-- //chatList -->
                                        <div class="box_form">
                                                <input type="text" name="msg" id="msg" />
                                                <button id="sendBtn" class="btn">보내기</button>
                                        </div><!-- //box_form -->
                                </div><!-- //chattingBox -->
                        </div><!-- //chattingContentBox -->
                        <div id="loginBox">
                                <div class="box_img">
                                        <h2>프로필 이미지 선택</h2>
                                        <img src="img/1.png" />
                                        <img src="img/2.png" />
                                        <img src="img/3.png" />
                                        <img src="img/4.png" />
                                        <img src="img/5.png" />
                                        <img src="img/6.png" />
                                        <img src="img/7.png" />
                                </div><!--// "box_img" -->
                                <div class="box_btn">
                                        <input     type="text"     name="nickname"     id="nickname"
placeholder="닉네임 입력"/>
                                        <button class="btn" id="loginBtn">채팅시작</button>
                                </div><!--// box_btn -->
                        </div><!--// "loginBox" -->
                        <button class="btn" id="closeBtn">채팅 종료</button>
                </div><!--// "content" -->
                <div id="footer">&copy;2018 jbm.com</div>
                <div id="loader"></div>
        </div>
        <script src="js/jquery.js"></script>
        <script src="js/underscore-min.js"></script>
        <script>
        _.templateSettings = {
                    interpolate: /\<\@\=(.+?)\@\>/gim,
                    evaluate: /\<\@(.+?)\@\>/gim,
                    escape: /\<\@\-(.+?)\@\>/gim
                };

        var $loader = $("#loader");

        var $profileImg = $("#loginBox img");
```

```
var $loginBox = $("#loginBox");

var $userList = $("#userList");

var $list = $("#chatList ul");

var img = "";

var nickname = "";

var ws = null;

var users = [];

var myId = "";
var ip = "";

var $msg = $("#msg");

var $sendBtn = $("#sendBtn");

var $chatList = $("#chatList");


var $nickname = $("#nickname");

var $loginBtn = $("#loginBtn");

var $closeBtn = $("#closeBtn");


$closeBtn.click(function() {
        ws.close();
});

$msg.keyup(function(e) {
        if(e.keyCode==13) {
                send();
        }
});

$sendBtn.click(send);

function send() {
        var msg = $msg.val();
        $msg.val("").focus();
```

```
		sendMsg(new Protocol(myId,5,nickname,img,ip,msg));
	}

	$profileImg.click(function() {
		var $this = $(this);

		if($this.hasClass("on")) {
			$profileImg.attr("class","");
			img = "";
		}else {
			img = $this.attr("class","on").attr("src");
			console.log(img);

			$profileImg.not(this).attr("class","off");
		}
	});

	$loginBtn.click(function() {
		nickname = $nickname.val().trim();

		if(nickname.length==0) {
			alert("닉네임을 입력해주세요~");
			$nickname.val("").focus();
			return;
		}

		if(img=="") {
			alert("이미지를 선택해주세요~");
			return;
		}

		$loader.show();

		//$loader.hide();
		//$loginBox.css("left",-1200);

		ws = new WebSocket("ws://192.168.0.103/chat");

		$(ws).bind("open",function() {

			//alert("open");

			$loader.hide();
			$loginBox.css("left",-1200);
			$nickname.val("");
			$msg.focus();
```

```
	}).bind("close",closeChatting)
	.bind("message",function(e) {

		var evt = e.originalEvent;

		var protocol = JSON.parse(evt.data);

		console.log(protocol);

		//alert(protocol);

		switch(protocol.code) {

		case 1:

			ip = protocol.ip;
			myId = protocol.id;

			var p = new Protocol(myId,3,nickname,img,ip);

			console.log(p.toString());

			sendMsg(p);
			return;
		case 2:
			sendMsg(new Protocol(myId,4,nickname,img,ip)); return;
		case 3:
			joinUser(protocol); return;
		case 4:
			removeUser(protocol); return;
		case 5:
			showMsg(protocol); return;
		}


	}).bind("error",function(error) {
		alert(error);
		$loader.hide();
	});

});

function closeChatting() {
	$nickname.val("");
	ws = null;
	$list.empty();
```

```
                $profileImg.attr("class","");
                $loginBox.css("left",0);
        }

        function showMsg(protocol) {
                $list.append(msgTmp({user: protocol}));
                var height= $list.height();
                //alert(height);
                $chatList.animate({scrollTop:height},100);
        }

        function removeUser(protocol) {
                protocol.id;

                users=JSON.parse(protocol.msg);

                console.log(users);

                $userList.html(usersTmp({users: users}));

                $list.append(leaveTmp({user: protocol}));
        }

        function joinUser(protocol) {

                users=JSON.parse(protocol.msg);

                console.log("myId:" + myId)

                console.log(users);

                $userList.html(usersTmp({users:users}));

                $list.append(joinTmp({user: protocol}));

        }

        function sendMsg(protocol) {
                ws.send(protocol);
        }

        function User(id,nickname,img,ip) {
                this.id=id||null;
                this.nickname = nickname||null;
                this.img = img||null;
                this.ip = ip||null;
        }
```

```
        function Protocol(id,code,nickname,img,ip,msg) {
                this.code = code||0;
                this.msg = msg||null;
                this.id=id||null;
                this.nickname = nickname||null;
                this.img = img||null;
                this.ip=ip||null;

                this.toString= function() {
                        return "{\"id\":\""+
                        this.id + "\",\"code\":"+
                        this.code+",\"nickname\":\""+
                        this.nickname+"\",\"msg\":\""+
                        this.msg + "\",\"ip\":\""+
                        this.ip+"\",\"img\":\""+
                        this.img + "\"}";
                }
        }

        </script>

        <script type="text/template" id="joinTmp">
                <li class="msg">
                        <img src="<@=user.img@>" alt="<@=user.nickname@>" />
                        <span><strong><@=user.nickname@></strong>님이 들어왔습니다.</span>
                </li>
        </script>
        <script type="text/template" id="leaveTmp">
                <li class="msg">
                        <img src="<@-user.img@>" alt="<@-user.nickname@>" />
                        <span><strong><@-user.nickname@></strong>님이 나갔습니다.</span>
                </li>
        </script>
        <script type="text/template" id="msgTmp">
                <li
<@ if(user.id==myId) {@>
        class='mine'
<@ } @>
>
                        <div class="card_user">
                                <img src="<@-user.img@>" alt="<@-user.nickname@>"/>
                                <strong><@-user.nickname@></strong>
                        </div><!-- //card_user -->
                        <div class="box_reply">
                                <div class="comments"><@-user.msg@></div>
<@ if(user.id!=myId) {@>
```

```
        <span class="ip"><@-user.ip@></span>
<@ } @>

                        </div><!--//box_reply-->
                </li>
        </script>

        <script type="text/template" id="userListTmp">
                <@ _.each(users,function(user){ @>
                <li class="user
<@ if(user.id==myId) {@>
        mine
<@ } @>
                ">
                        <img src="<@=user.img@>"/>
                        <h3><@=user.nickname@></h3>
                </li>
                <@})@>
        </script>

        <script>
        var usersTmp = _.template($("#userListTmp").html());
        var joinTmp = _.template($("#joinTmp").html());
        var msgTmp = _.template($("#msgTmp").html());
        var leaveTmp = _.template($("#leaveTmp").html());
        </script>
</body>
</html>
```

■ 서버 사이드(HTML) 개발

```
<bean id="webSocketHandler"
p:controller-ref="WSController"
 class="com.jbm.ws.websocket.WebSocketHandler"/>

<bean id="objectMapper" class="com.fasterxml.jackson.databind.ObjectMapper"/>

<bean id="WSController"
 class="com.jbm.ws.controller.WSController" p:mapper-ref="objectMapper">
       <property name="sessions">
       <list></list>
       </property>
       <property name="userMap">
       <map></map>
       </property>
</bean>

<bean class="com.jbm.ws.controller.IndexController"/>
```

■ WebsocketHandler

```
package com.jbm.ws.websocket;

import org.springframework.web.socket.CloseStatus;
import org.springframework.web.socket.WebSocketMessage;
import org.springframework.web.socket.WebSocketSession;
import org.springframework.web.socket.handler.TextWebSocketHandler;

import com.jbm.ws.controller.WSController;

public class WebSocketHandler extends TextWebSocketHandler {

        private WSController controller;



        public void setController(WSController controller) {
                this.controller = controller;
        }
```

```
	@Override
	public void afterConnectionClosed(WebSocketSession session,
			CloseStatus status) throws Exception {
		controller.remove(session);
	}

	@Override
	public void afterConnectionEstablished(WebSocketSession session)
			throws Exception {
		System.out.println("설립");
		controller.add(session);
	}

	@Override
	public void handleMessage(WebSocketSession session,
			WebSocketMessage<?> message) throws Exception {
		System.out.println("메세지 왔음");
		controller.execute(session, (String)message.getPayload());
	}


}
```

■ WSController

```
package com.jbm.ws.controller;

import java.util.Collection;
import java.util.Map;
import java.util.Vector;

import org.springframework.web.socket.TextMessage;
import org.springframework.web.socket.WebSocketSession;

import com.fasterxml.jackson.databind.ObjectMapper;
import com.jbm.ws.vo.Protocol;
import com.jbm.ws.vo.User;

public class WSController {

private ObjectMapper mapper;

	public void setMapper(ObjectMapper mapper) {
		this.mapper = mapper;
	}
```

```
	private Map<String,User> userMap;


	public void setUserMap(Map<String, User> userMap) {
		this.userMap = userMap;
	}

	private Vector<WebSocketSession> sessions;

	public void setSessions(Vector<WebSocketSession> sessions) {
		this.sessions = sessions;
	}

	public void add(WebSocketSession session) throws Exception {

		System.out.println("새로운 사용자 나타남");

		sessions.add(session);

		Protocol protocol = new Protocol(Protocol.ASK_JOIN,session.getId(),
session.getRemoteAddress().getHostString());


		System.out.println(protocol);

		this.sendMessage(protocol,session);

		System.out.println("사용자들:"+userMap);
	}


	public void sendMessage(Protocol protocol,WebSocketSession session) throws Exception{
		String msg = mapper.writeValueAsString(protocol);

		System.out.println("sendMessage:"+msg);

		session.sendMessage(new TextMessage(msg));
	}

	private void broadcast(Protocol protocol) throws Exception{

		String msg = mapper.writeValueAsString(protocol);

		for(WebSocketSession session: sessions) {
			session.sendMessage(new TextMessage(msg));
		}
```

```
	}

	public void execute(WebSocketSession session, String msg) throws Exception{

		System.out.println(msg);

		Protocol protocol = mapper.readValue(msg, Protocol.class);

		System.out.println(protocol.getCode());

		switch(protocol.getCode()) {
		case 3:
			protocol.setId(session.getId());
			this.setUser(protocol, session);
			return;
		case 4:;
		case 5:;
		case 6:;
		case 7:;
		case 8:this.broadcast(protocol);

		}
	}

	private void setUser(Protocol protocol,WebSocketSession session) throws Exception{

		User user = new User(protocol);
		userMap.put(session.getId(), user);

		changeUsers(protocol);
	}

	private void changeUsers(Protocol protocol) throws Exception{
		Collection<User> userList = userMap.values();

		System.out.println(protocol);

		String json = mapper.writeValueAsString(userList);

		System.out.println(userList);

		protocol.setMsg(json);

		broadcast(protocol);
	}

	public void remove(WebSocketSession session) throws Exception {
```

```
                sessions.remove(session);

                User user = userMap.get(session.getId());

                userMap.remove(session.getId());
                System.out.println("사용자 나감");
                System.out.println(userMap);

                Protocol protocol = new Protocol(Protocol.LEAVE);

                protocol.setNickname(user.getNickname());
                protocol.setImg(user.getImg());
                protocol.setId(session.getId());

                changeUsers(protocol);

        }

}
```

■ Protocol

```
package com.jbm.ws.vo;

public class Protocol {

        private int code;
        private String id,msg,nickname, img, ip;



        public static final int ASK_JOIN = 1;
        public static final int ASK_LEAVE = 2;

        public static final int JOIN = 3;//실제 join
        public static final int LEAVE = 4;//실제 leave
        public static final int MESSAGE = 5;
        public static final int CHANGE_NICKNAME= 6;
        public static final int CHANGE_IMG = 7;

        public Protocol() {
                // TODO Auto-generated constructor stub
        }

        public Protocol(int code) {
                this.code = code;
```

```
	}

	public Protocol(int code, String ip) {
		this.code = code;
		this.ip = ip;
	}

	public Protocol(int code,String id, String ip) {
		this.code = code;
		this.id = id;
		this.ip = ip;
	}

	public Protocol(int code, String msg, String nickname,
			String img, String ip) {
		this.code = code;
		this.msg = msg;
		this.nickname = nickname;
		this.img = img;
		this.ip = ip;
	}

	public String getIp() {
		return ip;
	}

	public void setIp(String ip) {
		this.ip = ip;
	}

	public String getId() {
		return id;
	}

	public void setId(String id) {
		this.id = id;
	}

	public int getCode() {
		return code;
	}
	public void setCode(int code) {
		this.code = code;
	}

	public String getMsg() {
		return msg;
```

```
	}
	public void setMsg(String msg) {
		this.msg = msg;
	}
	public String getNickname() {
		return nickname;
	}
	public void setNickname(String nickname) {
		this.nickname = nickname;
	}
	public String getImg() {
		return img;
	}
	public void setImg(String img) {
		this.img = img;
	}

}
```

■ User

```
package com.jbm.ws.vo;

public class User {
	private String id,nickname,img,ip;

	public User(Protocol protocol) {
		this.id = protocol.getId();
		this.nickname = protocol.getNickname();
		this.img = protocol.getImg();
		this.ip = protocol.getIp();
	}
	public User() {
		// TODO Auto-generated constructor stub
	}



	public String getIp() {
		return ip;
	}

	public void setIp(String ip) {
		this.ip = ip;
	}
```

```
        public String getId() {
                return id;
        }

        public void setId(String id) {
                this.id = id;
        }

        public String getNickname() {
                return nickname;
        }
        public void setNickname(String nickname) {
                this.nickname = nickname;
        }
        public String getImg() {
                return img;
        }
        public void setImg(String img) {
                this.img = img;
        }

}
```

■ index.jsp에서

```
function User(id,nickname,img,ip) {
                this.id=id||null;
                this.nickname = nickname||null;
                this.img = img||null;
                this.ip = ip||null;
}

function Protocol(id,code,nickname,img,ip,msg) {
        this.code = code||0;
        this.msg = msg||null;
        this.id=id||null;
        this.nickname = nickname||null;
        this.img = img||null;
        this.ip=ip||null;

        this.toString= function() {
                return "{\"id\":\""+
                this.id + "\",\"code\":"+
                this.code+",\"nickname\":\""+
                this.nickname+"\",\"msg\":\""+
```

```
                this.msg + "\",\"ip\":\""+
                this.ip+"\",\"img\":\""+
                this.img + "\"}";
        }
}
```