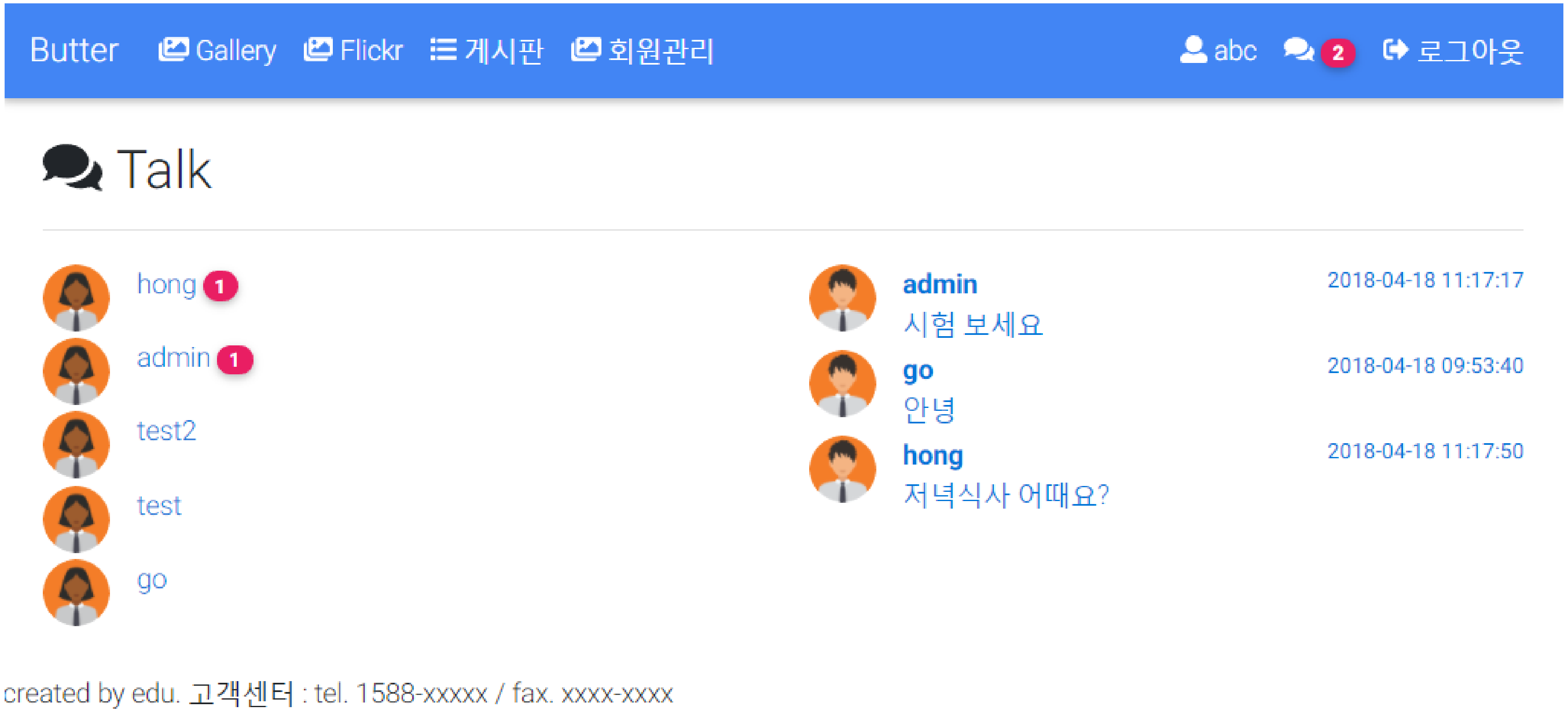
Talk

Talk



Talk

hong 1

admin 1

test2

test

go

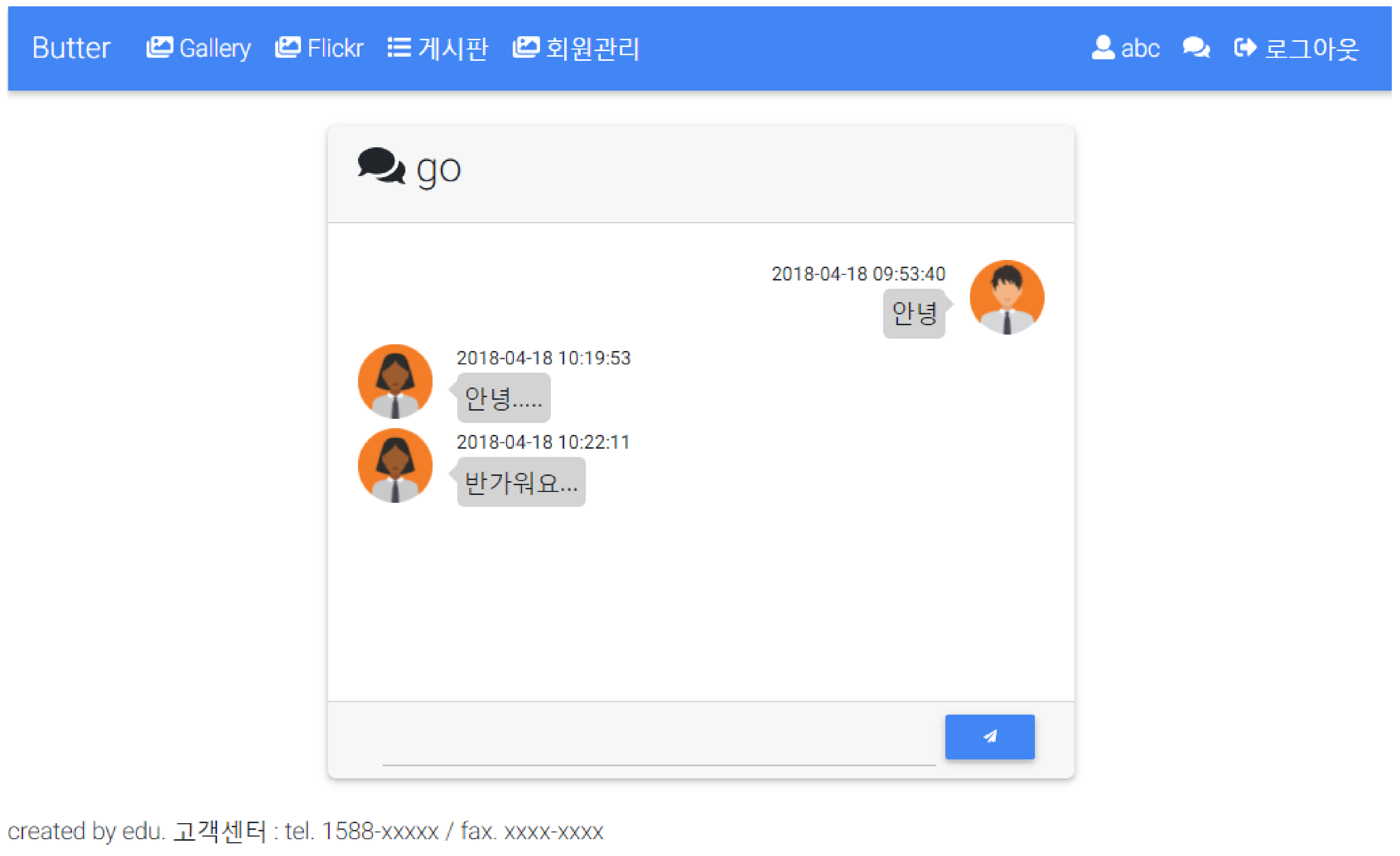
admin 2018-04-18 11:17:17
시험 보세요

go 2018-04-18 09:53:40
안녕

hong 2018-04-18 11:17:50
저녁식사 어때요?

created by edu. 고객센터 : tel. 1588-xxxxx / fax. xxxx-xxxx

Talk



2018-04-18 09:53:40
안녕

2018-04-18 10:19:53
안녕.....

2018-04-18 10:22:11
반가워요...

created by edu. 고객센터 : tel. 1588-xxxxx / fax. xxxx-xxxx

Talk

❖ Talks 테이블 정의

```
CREATE TABLE TALKS (
   TALK_ID     NUMBER PRIMARY KEY,
   USER_ID     VARCHAR2(20),
   WITH_TALK   VARCHAR2(20),
   RECEIVED NUMBER(1),
   CHECKED     NUMBER(1),
   MESSAGE  VARCHAR2(1024),
   REG_DATE DATE
);
```

❖ 시퀀스 정의

```
CREATE SEQUENCE TALKS_SEQ;
```

Talk

❖ **모델 : Talk**

```
@Data
@Builder
@NoArgsConstructor
@AllArgsConstructor
public class Talk {
    private  int       talkId;     // talk ID
    private String userId;     // 사용자 ID
    private String withTalk;   // 대상 사용자 ID
    private int    received;   // 수신 여부 0: 발신, 1: 수신
    private int    checked; // 확인 여부
    private String message; // 메시지
    private Date   regDate; // 등록일
}
```

Talk

❖ 매퍼 : resources/mapper/talk-mapper.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
            "http://mybatis.org/dtd/mybatis-3-mapper.dtd">

<mapper namespace="edu.iot.butter.dao.TalkDao">

   <select id="getCount" parameterType="Talk" resultType="int">
      SELECT
      count(*) FROM talks
      WHERE user_id=#{userId} and with_talk = #{withTalk}
   </select>

   <select id="selectOne" parameterType="int" resultType="Talk">
      SELECT *
      FROM talks
      WHERE talk_id=#{talkId}
   </select>
```

Talk

❖ 매퍼 : resources/mapper/talk-mapper.xml

```
<select id="selectList" parameterType="Talk" resultType="Talk">
   SELECT *
   FROM talks
   WHERE user_id=#{userId} AND
   with_talk=#{withTalk}
   ORDER BY talk_id
</select>

<select id="getNewTalks" parameterType="String" resultType="Talk">
   SELECT * FROM talks
   WHERE checked = 0 AND user_id = #{userId}
   ORDER BY talk_id DESC
</select>
```

Talk

❖ 매퍼 : resources/mapper/talk-mapper.xml

```
   <insert id="insert" parameterType="Talk">
      insert into talks ( TALK_ID, USER_ID, WITH_TALK, RECEIVED,
            CHECKED, MESSAGE, REG_DATE)
      values(TALKS_SEQ.NEXTVAL, #{userId}, #{withTalk}, #{received},
            #{checked}, #{message},sysdate)
   </insert>

   <update id="updateCheck" parameterType="Talk"><![CDATA[
      update talks set
         checked = 1
      where user_id = #{userId} and with_talk = #{withTalk}
            and checked=0
   ]]></update>

   <delete id="delete" parameterType="int">
      DELETE FROM talks
      WHERE
      talk_id=#{talkId}
   </delete>
</mapper>
```

Talk

❖ Dao : TalkDao

```
public interface TalkDao {
   int getCount(Talk talk) throws Exception;

   Talk selectOne(int talkId) throws Exception;

   List<Talk> selectList(Talk talk) throws Exception;

   List<Talk> getNewTalks(String userId) throws Exception;

   int insert(Talk talk) throws Exception;

   int delete(int talkId) throws Exception;

   int updateCheck(Talk talk) throws Exception;

}
```

Talk

❖ 서비스 : TalkService

```
public interface TalkService {

   List<Talk> getTalkList(Talk talk)  throws Exception;

   Talk getTalk(int talkId) throws Exception;

   boolean create(Talk talk) throws Exception;

   boolean send(Talk talk) throws Exception;

   boolean delete(int talkId) throws Exception;

   List<Talk> getNewTalks(String userId) throws Exception;

   boolean updateCheck(Talk talk) throws Exception;

}
```

Talk

❖ 서비스 : TalkServiceImpl

```
@Service
public class TalkServiceImpl implements TalkService{
   @Autowired
   TalkDao dao;

   @Override
   public List<Talk> getTalkList(Talk talk) throws Exception {
      System.out.println(talk);
      return dao.selectList(talk);
   }

   @Override
   public Talk getTalk(int talkId)  throws Exception{
      return dao.selectOne(talkId);
   }

   @Override
   public List<Talk> getNewTalks(String userId) throws Exception {
      return dao.getNewTalks(userId);
   }
```

Talk

❖ 서비스 : TalkServiceImpl

```
@Transactional
@Override
public boolean send(Talk talk)  throws Exception{
   return dao.insert(talk)==1;
}

@Transactional
@Override
public boolean delete(int talkId)  throws Exception{
   return dao.delete(talkId) ==1;
}


@Transactional
@Override
public boolean create(Talk talk) throws Exception {
   return dao.insert(talk)==1;
}
```

Talk

❖ 서비스 : TalkServiceImpl

```
   @Transactional
   @Override
   public boolean updateCheck(Talk talk) throws Exception {
      return dao.updateCheck(talk)>0;
   }
}
```

Talk

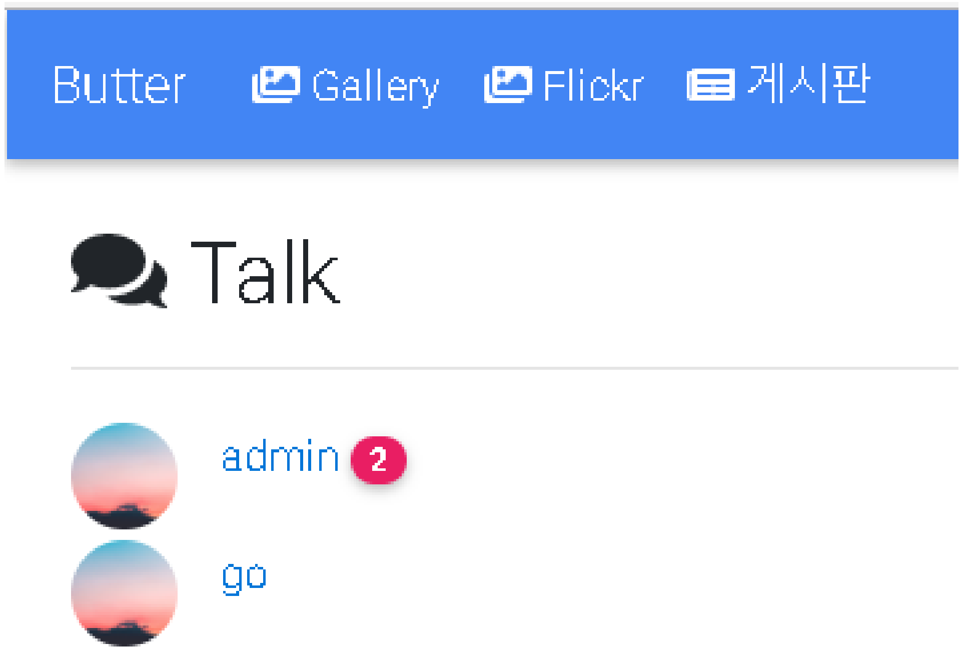
❖ 메뉴 : menu.jsp

```
<li class="nav-item">
   <a class="nav-link" href="${root}member/profile">

      <i class="fa fa-user"></i> ${USER.userId}
   </a>
</li>
<li class="nav-item">
   <a class="nav-link" href="${root}talk/home">
      <i class="fa fa-comments"></i>
   </a>
</li>
```

Talk

❖ 멤버 목록 리스트

- 신규 메시지가 있는 경우 배지로 표시



Talk

❖ 모델 수정 : Member

```
@Data
public class Member {
   @NotEmpty(message="사용자 ID는 필수 항목입니다.")
   private String   userId;
   @NotEmpty(message="이름은 필수 항목입니다.")
   private String   name;
   @NotEmpty(message="비밀번호는 필수 항목입니다.")
   private String   password;
   @NotEmpty(message="전화번호는 필수 항목입니다.")
   private String   cellPhone;
   private String   email;
   private String   address;
   private int    grade;
   private int    newMessages;   // 신규 메시지 개수
   private Date   regDate;
   private Date   updateDate;
}
```

Talk

❖ 맵퍼 수정 : member-mapper.xml

```
<select id="selectListWithMessages" resultType="Member"><![CDATA[
   select *
   from
      members m,
      ( SELECT with_talk, count(*) newMessages FROM talks
         WHERE checked = 0 and user_Id = #{userId}
         group by with_talk
      ) t
   where
      m.user_id <> #{userId} and
      m.user_id = t.with_talk(+)
]]></select>
```

❖ DAO 수정 : MemberDao 인터페이스

```
public interface MemberDao
   extends BaseDao<Member, String>{

   int changePassword(Password password) throws Exception;

   List<Member> selectListWithMessages(String userId);

   // 관리자 호출 메서드
   int updateByAdmin(Member member) throws Exception;
   int changePasswordByAdmin(Password password) throws Exception;
}
```

Talk

❖ 서비스 수정 : MemberService 인터페이스

```
public interface MemberService {
   :

   List<Member> getListWithMessages(String userId) throws Exception;

}
```

Talk

❖ 서비스 수정 : MemberServiceImpl

```
public interface MemberServiceImpl {
    :
    @Override
    public List<Member> getListWithMessages(String userId)
    throws Exception {
        return dao.selectListWithMessages(userId);
    }
}
```

Talk

❖ 컨트롤러 : TalkController

```
@Controller
@RequestMapping("/talk")
public class TalkController {

   @Autowired
   MemberService service;

   @Autowired
   TalkService talkService;
```

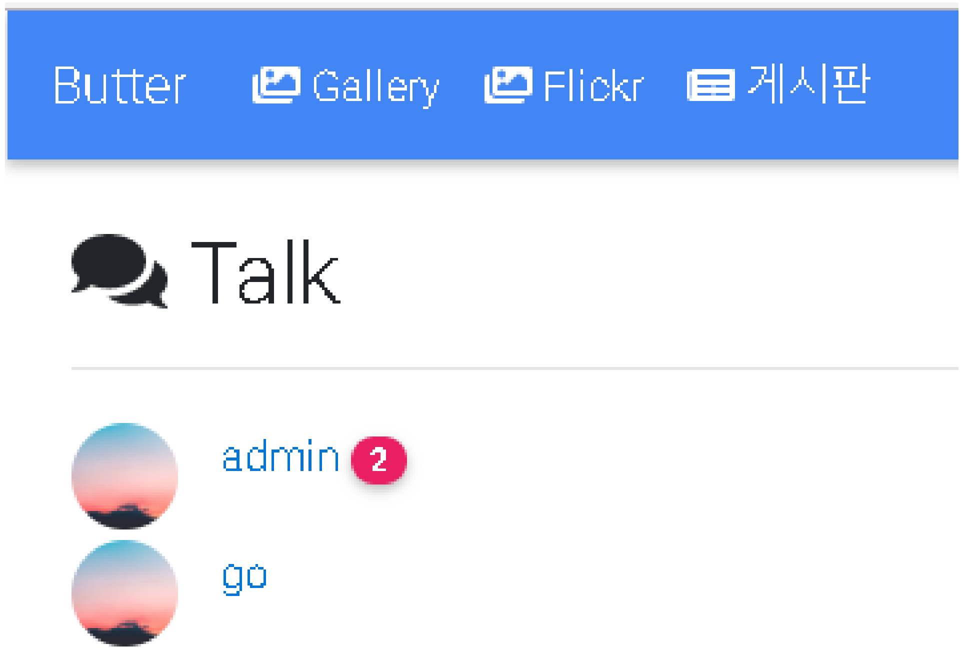
Talk

❖ 컨트롤러 : TalkController

```
    @RequestMapping(value="/home", method=RequestMethod.GET)
    public void talk(
          @RequestParam(value="page", defaultValue="1") int page,
          HttpSession session,
          Model model)  throws Exception {
       Member member = (Member)session.getAttribute("USER");
       String userId = member.getUserId();

       model.addAttribute("memberList",
                      service.getListWithMessages(userId));
    }

}
```

Talk

❖ 뷰 : talk/home.jsp



Talk

❖ 뷰 : talk/home.jsp

```
<div class="m-4">
   <h2><i class="fa fa-comments"></i> Talk</h2>
   <hr/>
   <div class="row" >
      <div class="col-md-6">
         <c:forEach var="member" items="${memberList}">
            <div class="media my-1">
                <img src="${root}member/avata?userId=${member.userId}“
                     class="d-flex mr-3 rounded-circle avata">
                <div class="media-body">
                  <a
href="with?userId=${USER.userId}&withTalk=${member.userId}">
                     ${member.userId}
                     <c:if test="${member.newMessages>0}">
                        <span class="badge badge-pill pink">
                           ${member.newMessages}
                        </span>
                     </c:if>
                  </a>
                </div>
```

Talk

❖ 뷰 : talk/home.jsp

```
                </div>
            </c:forEach>
        </div>
    </div>
</div>
```

Talk

❖ 웹소켓 핸들러 : TalkHandler

```
@Component
public class TalkHandler extends TextWebSocketHandler {
   // 동기화 맵 생성
   Map<WebSocketSession, String> sessionMap =
                     Collections.synchronizedMap(new HashMap<>());
   Map<String, WebSocketSession> memberMap =
                     Collections.synchronizedMap(new HashMap<>());

   @Autowired
   TalkService service;

   @Override
   public void afterConnectionClosed(WebSocketSession session,
                             CloseStatus status) throws Exception {
      String memberId = sessionMap.remove(session);
      memberMap.remove(memberId);

      super.afterConnectionClosed(session, status);
   }
```

Talk

❖ 웹소켓 핸들러 : TalkHandler

```
@Override
public void afterConnectionEstablished(WebSocketSession session)
throws Exception {
   Member member = (Member) session.getAttributes().get("USER");
   sessionMap.put(session, member.getUserId());

   memberMap.put(member.getUserId(), session);

   super.afterConnectionEstablished(session);
}
```

Talk

❖ 웹소켓 핸들러 : TalkHandler

```
@Override
protected void handleTextMessage(WebSocketSession session,
                               TextMessage message) throws Exception {
   // 메시지 수신자 식별 및 처리
   Gson gson = new Gson();
   Talk talk = gson.fromJson(message.getPayload(), Talk.class);

   // 전송자의 전송 talk 저장
   service.create(talk);

   send(talk, message);
   super.handleTextMessage(session, message);
}
```

Talk

❖ 웹소켓 핸들러 : TalkHandler

```
public void send(Talk talk, TextMessage message) throws Exception{

   WebSocketSession s = memberMap.get(talk.getWithTalk());
   if (s != null) { // 수신자 접속시  checked 1
      s.sendMessage(message);   // 메시지 전송
      saveTalk(talk, 1);
   } else {         // 수신자 미접속시 checked 0
      saveTalk(talk, 0);
   }
}
```

Talk

❖ **웹소켓 핸들러 : TalkHandler**

```
   public void saveTalk(Talk talk, int checked) throws Exception {
      // 수신자 측 수신 Talk 저장
      Talk talk2 = Talk.builder()
                 .userId(talk.getWithTalk())
                 .withTalk(talk.getUserId())
                 .checked(checked)
                 .received(1)
                 .message(talk.getMessage())
                 .build();
      service.create(talk2);
   }
}
```

Talk

❖ **servlet-context.xml**

```
   <websocket:handlers>
      <websocket:mapping handler="testHandler" path="/talk/echo" />
      <websocket:mapping handler="talkHandler" path="/talk/talk" />
      <websocket:handshake-interceptors>
         <beans:bean
   class="org.springframework.web.socket.server.support.HttpSessionHa
ndshakeInterceptor" />
      </websocket:handshake-interceptors>
      <websocket:sockjs />
   </websocket:handlers>
```

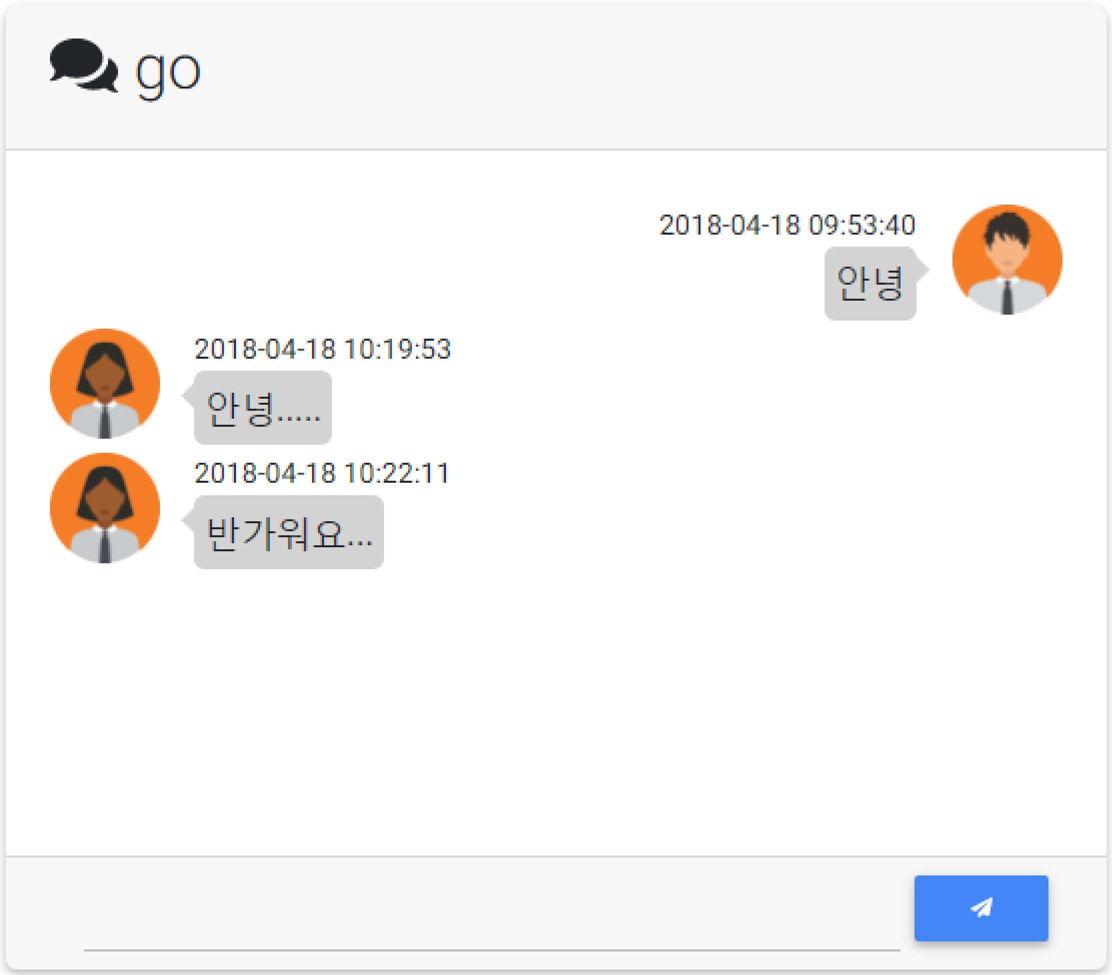
Talk

❖ 컨틀롤러 : TalkController

```
@Controller
@RequestMapping("/talk")
public class TalkController {
      :

   @RequestMapping(value="/with", method=RequestMethod.GET)
   public void with(Talk talk)  throws Exception {
      // 읽지 않은 메시지 checked를 1로 변경
      talkService.updateCheck(talk);
   }

   // 대화 상대의 talk 리스트 추출 후 리턴
   @ResponseBody
   @RequestMapping(value="/talkList", method=RequestMethod.GET)
   public List<Talk> talkList(Talk talk)  throws Exception {
      System.out.println(talk);
      return talkService.getTalkList(talk);
   }
}
```

Talk

❖ 뷰 : talk/with.jsp



go

2018-04-18 09:53:40

안녕

2018-04-18 10:19:53

안녕.....

2018-04-18 10:22:11

반가워요...

Talk

❖ 뷰 : talk/with.jsp

```
<div class="m-4 mx-auto" style="width:500px">
   <div class="card border-primary mb-3" >
      <div class="card-header">
         <h3>
            <i class="fa fa-comments primary"></i> ${param.withTalk}
         </h3>
      </div>

      <div class="card-body scroll" style="height:20rem;">
         <div id="messages"></div>
      </div>
```

스크롤 및 높이 지정

전송, 수신 메시지 출력 영역

Talk

❖ 뷰 : talk/with.jsp

```
        <div class="card-footer py-2">
            <div class="md-form input-group my-0">
                <input type="text" id="send-message"
                            class="form-control my-0 py-0 ">
                <span class="input-group-btn">
                    <button id="send-btn"
                            class="my-0 btn btn-primary btn-sm">
                        <i class="fa fa-paper-plane"></i>
                    </button>
                 </span>
            </div>
        </div>
    </div>
</div>
```

❖ 뷰 : talk/with.jsp

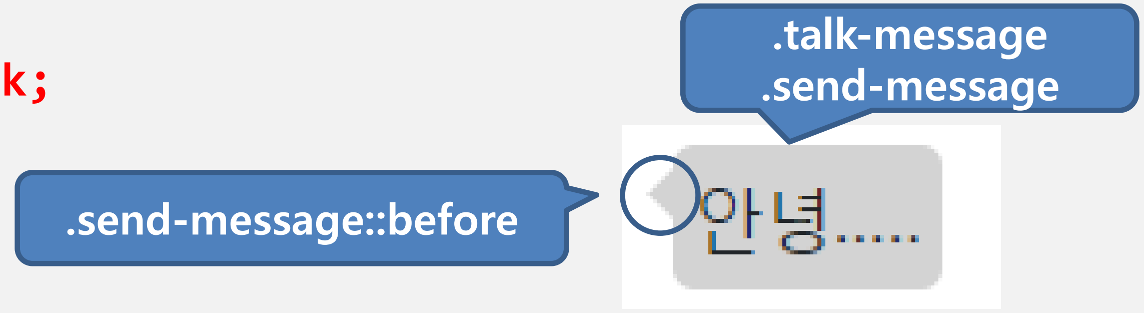
- talk 플러그인 설정

```
<link rel="stylesheet"
      href="<c:url value="/resources/css/talk.css"/>"/>
<script
   src="https://cdnjs.cloudflare.com/ajax/libs/sockjs-client/1.1.4/sockjs.js">
</script>
<script src="<c:url value="/resources/js/talk.js"/>"></script>
<script src="<c:url value="/resources/js/rest.js"/>"></script>
<script>
$(function(){
   $('#messages').talk({
      userId : '${USER.userId}',            // 사용자 ID
      withTalk : '${param.withTalk}',     // 대화 상대방 ID
      url : '/butter/talk',               // 웹 소켓 url
      sendBtn : $('#send-btn'),           // 전송 버튼
      sendMessage : $('#send-message'), // 전송 메시지 엘리먼트
   });
});
</script>
```

Talk

❖ 스타일 : resources/css/talk.css

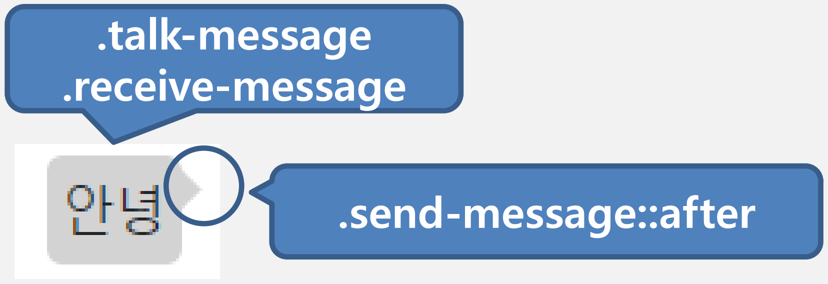
```
.talk-message {
   background-color : lightgray;
   border-radius : 5px;
   padding : 5px;
   display : inline-block;
   position : relative;
}

.send-message::before {
    content: " ";
    position: absolute;
    top: 15px;
    right: 100%;
    margin-top: -10px;
    border-width: 6px;
    border-style: solid;
    border-color: transparent lightgray transparent transparent;
}
```



Talk

❖ 스타일 : resources/css/talk.css

```
.receive-message::after {
    content: " ";
    position: absolute;
    top: 15px;
    left: 100%;
    margin-top: -10px;
    border-width: 6px;
    border-style: solid;
    border-color: transparent transparent transparent lightgray;
}
```



Talk

❖ **스크립트: resources/js/talk.js**

  ○ 메시지 템플릿 정의

```
var talkTempl = {
   sendTempl : function(msg) {
      return `
      <div class="media my-1">
          <img src="/butter/member/avata?userId=${msg.userId}“
               class="d-flex mr-3 rounded-circle avata">
          <div class="media-body text-left ml-3 mr-5">
            <div class="small">${msg.regDate.str()}</div>
            <div class="talk-message send-message">
               ${msg.message}</div>
          </div>
      </div>`;
   },
```

Talk

❖ **스크립트: resources/js/talk.js**

- 메시지 템플릿 정의

```
    receiveTempl : function(msg) {
        return `
        <div class="media my-1">
            <div class="media-body text-right mr-3 ml-5">
              <div class="small">${msg.regDate.str()}</div>
              <div  class="talk-message receive-message">
                 ${msg.message}</div>
            </div>
            <img src="/butter/member/avata?userId=${msg.withTalk}“
                  class="d-flex mr-3 rounded-circle avata" >

        </div>`;
    }
};
```

Talk

❖ **스크립트: resources/js/talk.js**

- Talk 클래스 정의

```
class Talk {
   constructor(opt){
      this.opt = opt;

      // 웹 소켓 접속
      this.socket = new SockJS(this.opt.url + '/talk');

      // 이벤트 핸들러를 Talk 객체의 프로토타입 메서드로 설정
      // 호출된 이벤트 핸들러내에 this를 현재 인스턴스로 바인딩
      this.socket.onopen = this.onopen.bind(this);
      this.socket.onmessage = this.onmessage.bind(this);
      this.socket.onclose = this.onclose.bind(this);
      this.socket.onerror = this.onerror.bind(this);
```

Talk

❖ **스크립트: resources/js/talk.js**

- Talk 클래스 정의

```
    // 이전 talk 리스트 받아 초기화
    $.get(opt.url + '/talkList', {
        userId : this.opt.userId,
        withTalk : this.opt.withTalk
    }, this.initMessage.bind(this));
}

initMessage(messages) {
    var self = this;
    messages.forEach( function(msg) {
        msg.regDate = new Date(msg.regDate)
        if(msg.received == 1) {   // 수신 데이터
            self.opt.panel.append(talkTempl.receiveTempl(msg));
        } else { // 전송 데이터
            self.opt.panel.append(talkTempl.sendTempl(msg));
        }
    });
    self.opt.panel.parent()
        .scrollTop(self.opt.panel.height());
}
```

Talk

❖ **스크립트: resources/js/talk.js**
- Talk 클래스 정의

```
// 접속 성공 이벤트 핸들러
onopen(){
   console.log('접속 성공');
}

// 접속 해제 이벤트 핸들러
onclose(){
   console.log('접속 해제');
}

// 에러 이벤트 핸들러
onerror(e) {
   console.log('에러', e);
}
```

Talk

❖ **스크립트: resources/js/talk.js**

○ Talk 클래스 정의

```
// 메시지 수신 이벤트 핸들러
onmessage(msg) {
   this.opt.panel.append(talkTempl.receiveTempl(msg));
   this.opt.panel.parent()
         .scrollTop(this.opt.panel.height());
}

// 전송 메시지 템플릿 추가
addSendTempl(msg) {
   this.opt.panel.append(talkTempl.sendTempl(msg));
   this.opt.panel.parent()
         .scrollTop(this.opt.panel.height());
}
```

Talk

❖ **스크립트: resources/js/talk.js**

- Talk 클래스 정의

```
   // 메시지 전송
   send(message) {
      var msg = {
            userId : this.opt.userId,
            withTalk : this.opt.withTalk,
            received : 0,
            message : message,
            checked : 1
      };
      // console.log(JSON.stringify(msg));
      this.socket.send(JSON.stringify(msg));
      msg.regDate = new Date();
      return msg;
   }
}
```

Talk

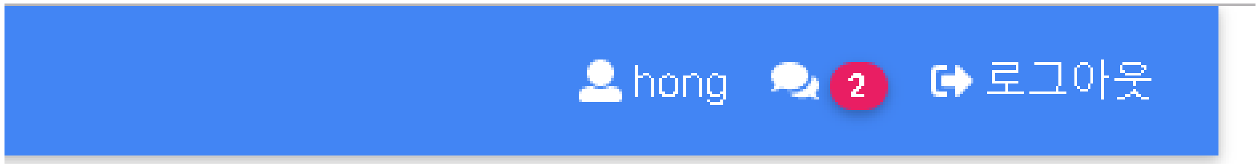
❖ **스크립트: resources/js/talk.js**

- talk 플러그인 정의

```
$.fn.talk = function(opt) {
   opt = $.extend(opt, {panel: this});  // 메시지 출력 엘리먼트 설정
   var talk = new Talk(opt);

   // 전송 버튼 이벤트 핸들러
   opt.sendBtn.click(function () {
      var message = opt.sendMessage.val();
      if(message.trim() == '') return;  // 메시지 없는 경우 취소

      var msg = talk.send(message);     // 메시지 전송
      talk.addSendTempl(msg);           // 전송 Talk 화면 출력
      opt.sendMessage.val('').focus();  // 전송 메시지 입력 창 지움
   });
}
```

Talk

❖ 신규 메시지 있음 알림



- 인터셉터로 체크
  - TalkInterceptor 정의

Talk

❖ 인터셉터 : TalkInterceptor

```
@Component
public class TalkInterceptor extends HandlerInterceptorAdapter {
   @Autowired
   TalkService service;

   @Override
   public boolean preHandle(HttpServletRequest request,
                  HttpServletResponse response, Object handler)
         throws Exception {

      Member member = (Member)request.getSession()
                               .getAttribute("USER");
      if(member != null) {
         List<Talk> talks = service.getNewTalks(member.getUserId());
         if(talks.size()>0) {
            request.setAttribute("newTalks", talks.size());
         }
      }
      return true;
   }
}
```

Talk

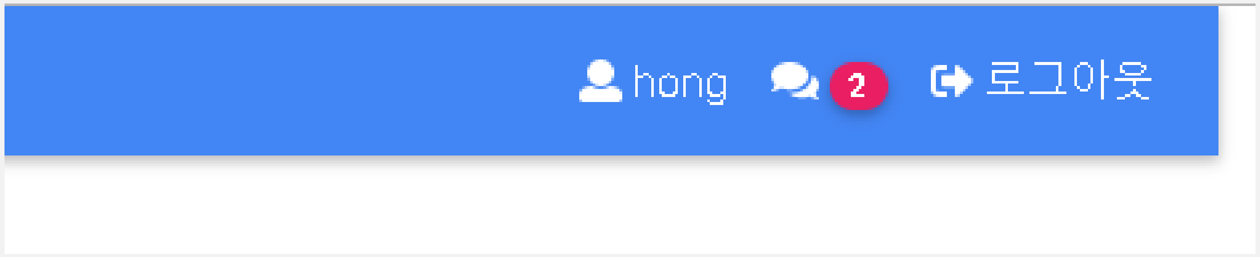
❖ servlet-context.xml

```
<interceptors>
   <interceptor>
      <mapping path="/**" />
      <beans:ref bean="talkInterceptor" />
   </interceptor>
   <interceptor>
      <mapping path="/admin/**" />
      <mapping path="/talk/*" />
      <mapping path="/gallery/*" />
      <mapping path="/member/*" />
      <mapping path="/board/create" />
      <beans:ref bean="loginInterceptor" />
   </interceptor>
   <interceptor>
      <mapping path="/admin/**" />
      <beans:ref bean="adminInterceptor" />
   </interceptor>
</interceptors>
```

Talk

❖ **menu.jsp**

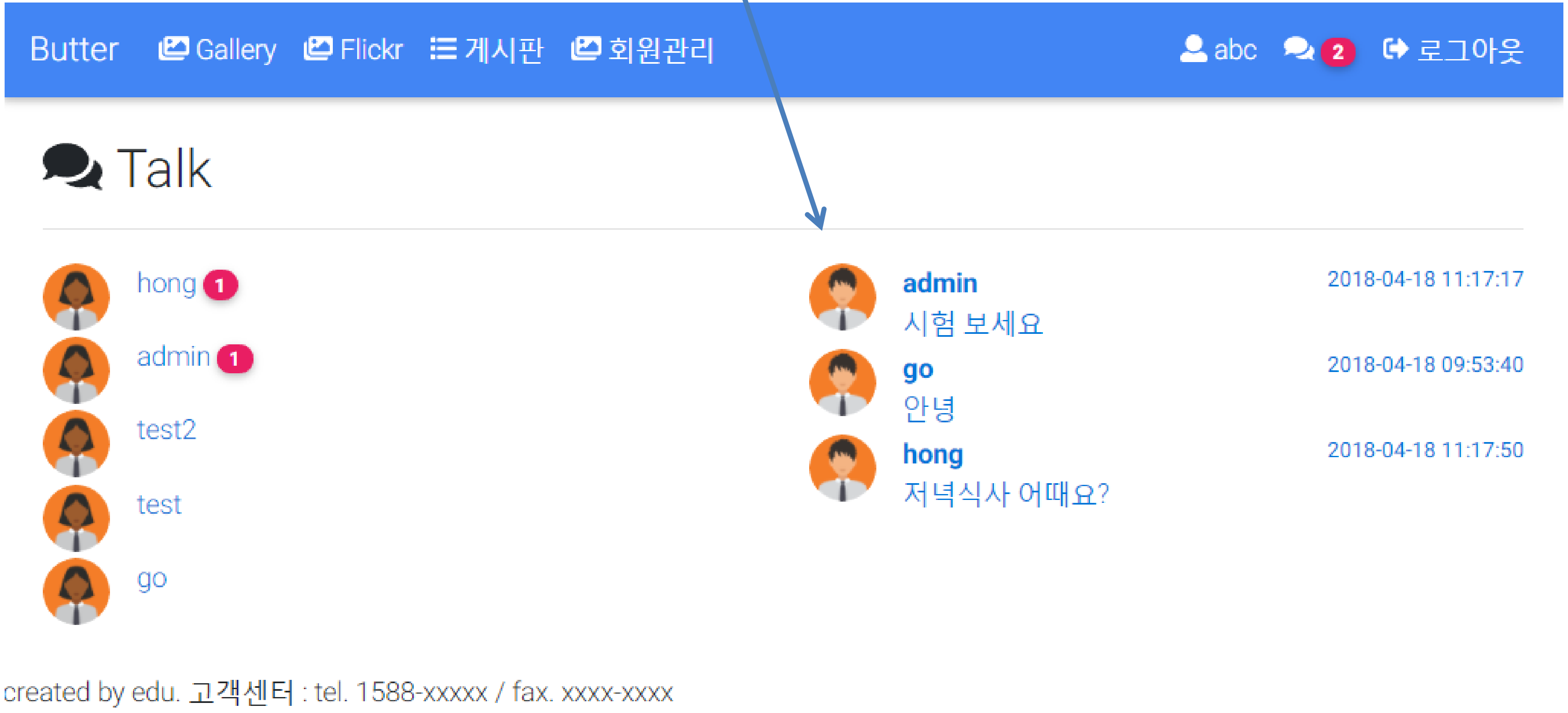
```
<li class="nav-item">
   <a class="nav-link" href="${root}member/profile">
      <i class="fa fa-user"></i> ${USER.userId}
   </a>
</li>
<li class="nav-item">
   <a class="nav-link" href="${root}talk/home">
      <i class="fa fa-comments"></i>
      <c:if test="${newTalks>0}">
         <span class="badge badge-pill pink">
            ${newTalks}
         </span>
      </c:if>
   </a>
</li>
```



Talk

❖ **도전 !**

- 문제 1: 전송 메시지 작성에서 엔터를 누르면 전송하기.
- 문제 2: 수신한 메시지를 회원별 최근 1건 표시하기
- 문제 3: 채팅



admin 2018-04-18 11:17:17
시험 보세요

go 2018-04-18 09:53:40
안녕

hong 2018-04-18 11:17:50
저녁식사 어때요?

created by edu. 고객센터 : tel. 1588-xxxxx / fax. xxxx-xxxx