

프리랜서 아웃소싱 서비스

[핵심 기술 소개서]

노마드 (NOMAD)

[팀장 : 하지원]

구 성 원	김재원, 박시인, 조원빈, 최영진, 최지훈
개발일정	2017.07.18 ~ 2017.08.31
발 표 일	2017.09.01 (금요일)
작 성 일	2017.09.04 ~ 2017.09.07

목차

1	개요	3p
2	채팅방 목록 생성	4p
3	채팅방 입장	7p
4	채팅 전송, 출력	10p
5	자격시험 리스트	16p
6	자격시험 입력	19p
7	자격시험 페이지	22p
8	자격시험 테스트	25p
9	세부기술 선택	31p
10	프로젝트 등록	34p
11	결제	39p
12	아이디 찾기	42p
13	비밀번호 찾기	45p
14	공지사항 수정이동	50p
15	공지사항 수정	53p
16	신고페이지 이동	56p
17	신고 작성	59p
18	댓글 작성	63p
19	댓글 삭제	67p
20	포트폴리오 다중업로드	70p
21	포트폴리오 출력	75p
22	포트폴리오 상세보기	78p

PROJECT NAME : STEEP(프리랜서 아웃소싱 서비스)

■ 개요

1. 고객이 원하는 프로젝트를(개발, 디자인) 합리적으로 진행할 수 있는 서비스
2. 개발자(프리랜서)가 편리하게 구직을 할 수 있는 서비스

■ 사용기술

1. **Language** : Java, JSP, HTML, CSS, JavaScript
2. **Framework** : Spring, Mybatis, JQuery, Bootstrap
3. **DB** : ORACLE
4. **Server** : Tomcat v9.0
5. **Design Pattern** : MVC (Model - View – Controller)

■ 규모

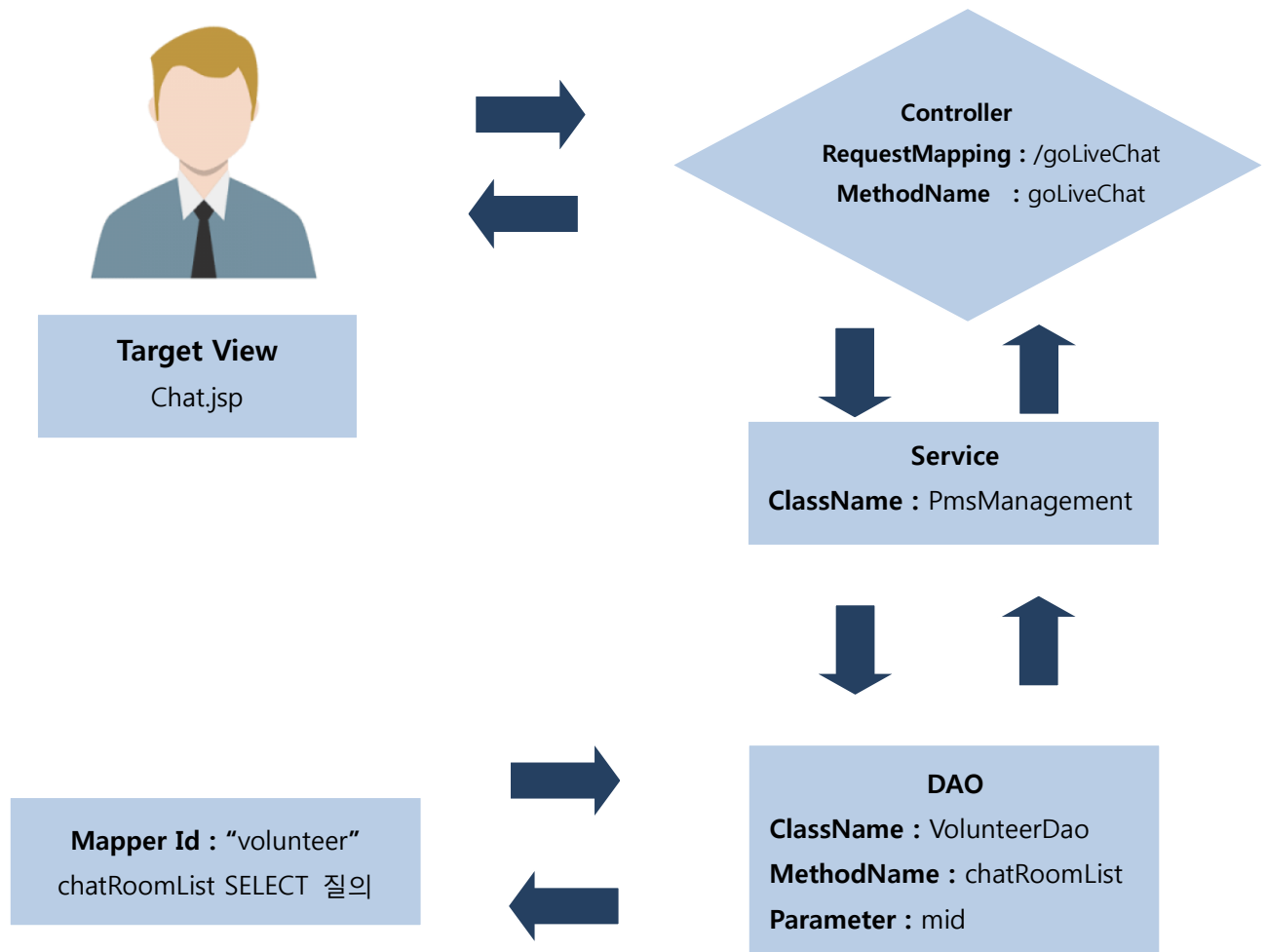
1. DB

- 1) DB Table : 17개 (총 Column : 121개)
- 2) 사용된 Query : 193개

2. Source

- 1) Class
 - Controller : 9개 (권한별, 기능별 분류) - 총 URL - 128개
 - Bean : 26개
 - Service : 8개
 - DAO : 12개
 - 기타 Class 포함 총 수량 : 57개
- 2) View (JSP File) : 58개

JOB CODE	A01	JOB NAME	채팅방 목록 생성	CLASS	PmsManagement
PROCESS FLOW					



■ 개요

1. Client와 Freelancer가 채팅을 할 방을 선택
2. chatList SELECT
3. TargetView : chat.jsp

■ CONTROLLER

REQUEST : /goLiveChat

```

@RequestMapping(value = "/goLiveChat")
public ModelAndView goLiveChat(){
    mav = pmm.execute(5);
    return mav;
}

```

1. ServiceClass인 PmsManagement 클래스의 execute(5) 함수를 실행
2. PmsManagement의 goLiveChat 함수를 실행
3. ModelAndView를 return

■ SERVICE

PARAMETER :

```

//채팅방리스트
private void goLiveChat() {
    //1. ModelAndView 생성
    mav = new ModelAndView();
    String m_id = session.getAttribute("m_id").toString();
    if(session.getAttribute("m_id")!=null && session != null){
        //2. chatRoomList에서 vDao의 chatRoomList 함수의 결과를 담는다.(팀이 등록 된 목록을 불러옴)
        List<Volunteer> chatRoomList = vDao.chatRoomList(m_id);
        String chatRoomListMake = chatRoomListMake(chatRoomList);
        //4. Append한 HTML 코드를 ModelAndView에 addObject 한다.
        mav.addObject("makeList", chatRoomListMake);
    }
    //5. TargetView(chat.jsp)를 ModelAndView에 setViewName 한다.
    mav.setViewName("chat");
}
private String chatRoomListMake(List<Volunteer> chatRoomList) {
    //3. chatRoomListMake(list)에 StringBuilder()를 이용하여 HTML 코드를 append 한다.
    StringBuilder sb = new StringBuilder();
    Volunteer volunteer = null;
    for(int i=0; i<chatRoomList.size(); i++){
        volunteer = chatRoomList.get(i);
        sb.append("<div class='panel panel-default' style='width:250px; float: left; margin: 5px'>");
        sb.append("<h4 class='panel-heading' style='text-align:center;'>"+volunteer.getP_title()+"</h4><br/>");
        sb.append("<div style='text-align:center;'>"
            + "<input type='button' class='btn btn-default' "
            + "onclick='chatStart(\""+volunteer.getV_pnum()+"\")' value='채팅하기' /></div>");
        sb.append("</div>");
    }
    return sb.toString();
}
}

```

1. ModelAndView 생성
2. chatRoomList에서 vDao의 chatRoomList 함수의 결과를 담는다.(팀이 등록 된 목록을 불러옴)
3. chatRoomListMake(list)에 StringBuilder()를 이용하여 HTML 코드를 append 한다.
4. Append한 HTML 코드를 ModelAndView에 addObject 한다.
5. TargetView(chat.jsp)를 ModelAndView에 setViewName 한다.

■ DAO & MAPPER PARAMETER : mid

```
public List<Volunteer> chatRoomList(String mid) {
    return sqlSession.selectList("volunteer.chatRoomList",mid);
}
```

```
<select id="chatRoomList" parameterType="String" resultType="Volunteer">
    SELECT p.p_title, v.v_pnum, p.p_mid FROM project p INNER JOIN volunteer v ON p.p_num = v.v_pnum
    WHERE v.v_ptteam = 1 AND v.v_mid = #{mid} OR p.p_mid = #{mid}
</select>
```

1. Mapper id "volunteer"의 쿼리 실행 결과를 List<Volunteer>에 담는다.
2. Id "Volunteer"는 채팅방 목록을 불러오는 select 질의

■ TARGET VIEW MODEL chat.jsp

PMS Project Management Service

❗작업량 확인 팀원간 채팅 멤버관리를 해주세요.

안드로이드 앱 제작

채팅하기

웹 홈페이지 제작

채팅하기

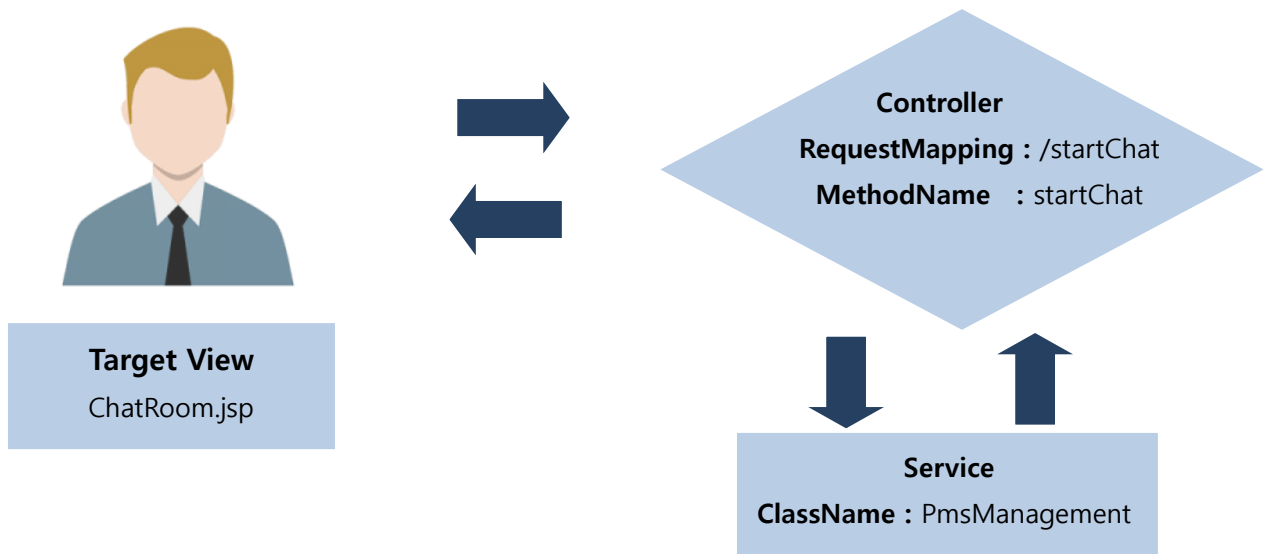
JAVA를 이용한 프로젝트
제작

채팅하기

→ 실시간 채팅

Client가 프로젝트를 등록한 채팅방 목록을 불러온 결과

JOB CODE	A02	JOB NAME	채팅방 입장하기	CLASS	PmsManagement
PROCESS FLOW					



■ 개요

1. Client와 Freelancer가 등록되어 있는 채팅방에 입장
2. TargetView : chatRoom.jsp

■ CONTROLLER

REQUEST : /startChat

```

@RequestMapping(value = "/startChat")
public ModelAndView startChat(){
    mav = pmm.execute(6);
    return mav;
}
  
```

1. ServiceClass인 PmsManagement 클래스의 execute(6) 함수를 실행
2. PmsManagement의 startChat 함수를 실행
3. ModelAndView를 return

```

private void startChat() {
    //1. ModelAndView 생성
    mav = new ModelAndView();
    int pnum = Integer.parseInt(request.getParameter("pnum"));
    //2. chatRoomMake() 메서드에 프로젝트 번호(pnum)를 매개변수로 넘긴다.
    String chatRoom = chatRoomMake(pnum);
    //4. Append한 HTML 코드를 ModelAndView에 addObject 한다.
    mav.addObject("chatRoomHtml", chatRoom);
    //5. TargetView(chatRoom.jsp)를 ModelAndView에 setViewName 한다.
    mav.setViewName("chatRoom");
}

private String chatRoomMake(int pnum) {
    //3. StringBuilder()를 이용하여 HTML코드를 append 한다.
    StringBuilder sb = new StringBuilder();
    sb.append("<div id='chat' class='panel-collapse collapse in'>");
    sb.append("<div id='chatList' class='portlet-body chat-widget' style='overflow-y:auto; overflow-x:hidden; width:auto; height:400px;'>");
    sb.append("</div></div>");
    sb.append("<div class='portlet-footer'>");
    sb.append("<div class='row' style='height: 90px;'>");
    sb.append("<div class='form-group col-xs-10'>");
    sb.append("<textarea style='height:80px;' id='chatContent' class='form-control' placeholder='메시지를 입력하세요' maxlength='100'></textarea>");
    sb.append("</div>");
    sb.append("<div class='form-group col-xs-2'>");
    sb.append("<button type='button' class='btn btn-default pull-right' onclick='submitFunction(\"+pnum+")' style='width: 150px; height:80px;'>전송</button>");
    sb.append("<input type='hidden' id='pnum' value=\""+pnum+"\">");
    sb.append("</div>");
    sb.append("</div>");
    sb.append("</div>");
    return sb.toString();
}

```

1. ModelAndView 생성
2. chatRoomList() 메서드에 프로젝트 번호(pnum)을 매개변수로 넘긴다.
3. StringBuilder()를 이용하여 HTML코드를 append 한다.
4. Append한 HTML 코드를 ModelAndView에 addObject 한다.
5. TargetView(chatRoom.jsp)를 ModelAndView에 setViewName 한다.

■ TARGET VIEW

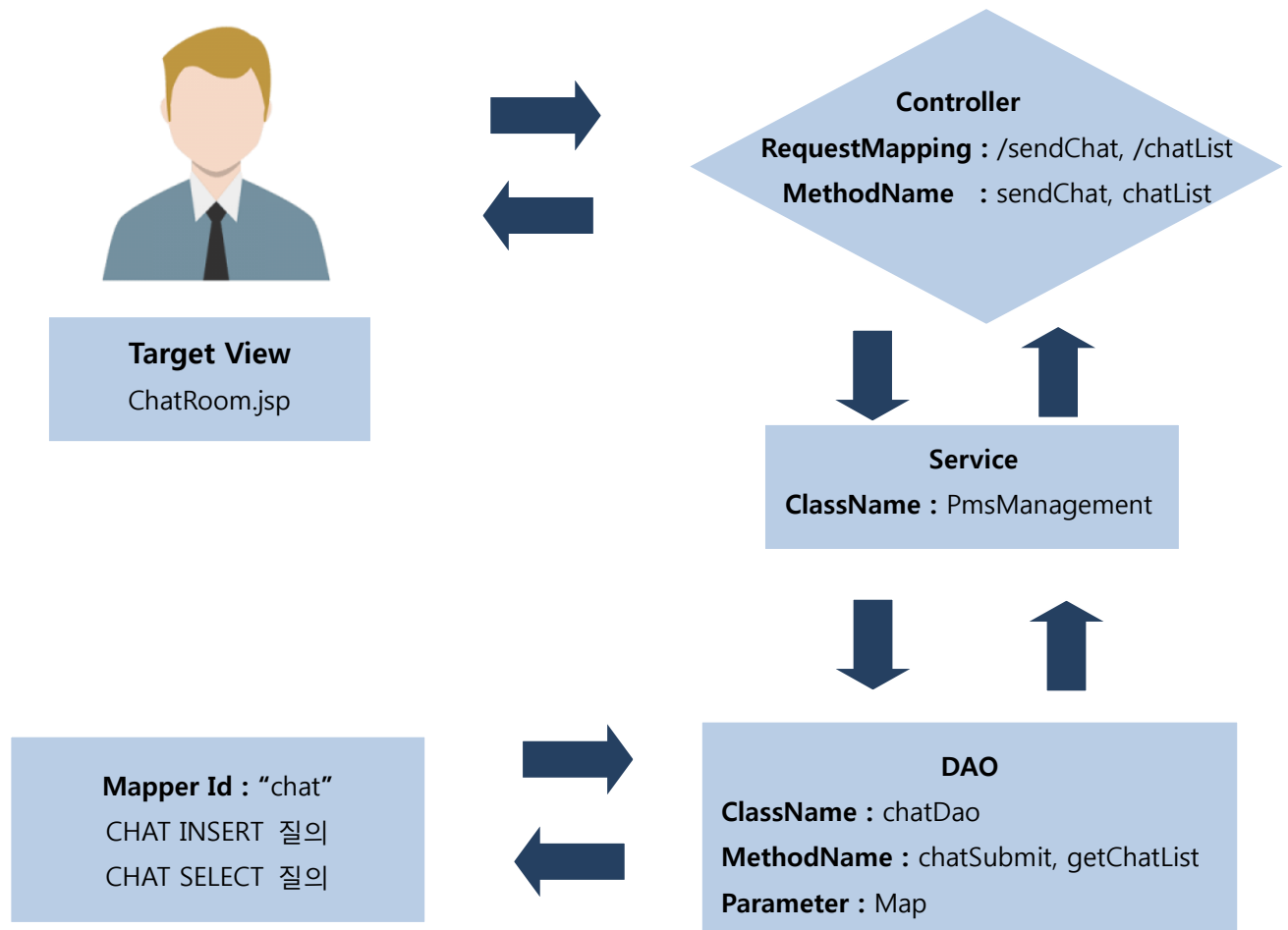
→ 실시간 채팅

메시지를 입력하세요

전송

Client와 Freelancer가 채팅 할 수 있는 공간 생성

JOB CODE	A03	JOB NAME	채팅 전송, 출력	CLASS	PmsManagement
PROCESS FLOW					



■ 개요

1. Client와 Freelancer가 text 입력 후 전송버튼 클릭
2. CHAT INSERT, CHAT SELECT
3. TargetView : chatRoom.jsp

■ CONTROLLER

REQUEST : /sendChat,
/chatList

```
@RequestMapping(value = "/sendChat")
public @ResponseBody String sendChat(){
    String jsonStr = pmm.executeAjax(2);
    return jsonStr;
}
```

1. ServiceClass인 PmsManagement 클래스의 executeAjax(2) 함수를 실행
2. PmsManagement의 sendChat 함수 실행
3. String Type으로 Json 형태로 받아서 return

```
@RequestMapping(value = "/chatList", produces = "application/text; charset=utf8")
public @ResponseBody String chatList(){
    String jsonStr = pmm.executeAjax(3);
    return jsonStr;
}
```

1. ServiceClass인 PmsManagement 클래스의 executeAjax(3) 함수를 실행
2. PmsMangement의 chatList 함수 실행
3. String Type으로 Json 형태로 받아서 return

■ SERVICE PARAMETER :

```
private void sendChat() {
    //1. chatRoom.jsp으로부터 받아온 채팅내용(chatContent), 프로젝트 번호(pnum)를 변수에 담는다.
    String chatContent = request.getParameter("chatContent");
    System.out.println("chatContent:"+chatContent);
    String pnum = request.getParameter("pnum");
    System.out.println("pnum:"+pnum);
    String mid = session.getAttribute("m_id").toString();
    Map<String, String> map = new HashMap<String, String>();
    int chatNum = 0;
    //2. Mid(session ID), 채팅내용(chatContent), 프로젝트 번호(pnum) 중 null값이 있다면 json에 0을 return
    if(mid == null || mid.equals("") || chatContent == null || chatContent.equals("")
        || pnum == null || pnum.equals("")){
        jsonStr = "0";
    }else{
        if(chatDao.chatCount()!=0){
            chatNum = chatDao.chatMaxNum()+1;
        }else{
            chatNum = 1;
        }
        System.out.println("chatNum:"+chatNum);
        //3. 채팅내용(chatContent), 프로젝트 번호(pnum)을 HashMap에 담는다.
        try {
            map.put("pnum", pnum);
            map.put("chatContent", URLDecoder.decode(chatContent,"UTF-8"));
            map.put("mid", mid);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
        map.put("chatNum", String.valueOf(chatNum));
        //4. HashMap에 담은 변수를 chatDao의 chatSubmit 함수에 매개변수로 넘긴다.
        //5. Chat table에 INSERT 질의 후 반환 된 return 값을 jsonStr로 return
        jsonStr=String.valueOf(chatDao.chatSubmit(map));
    }
}
```

1. chatRoom.jsp으로부터 받아온 채팅내용(chatContent), 프로젝트 번호(pnum)를 변수에 담는다.
2. Mid(session ID), 채팅내용(chatContent), 프로젝트 번호(pnum) 중 null값이 있다면 json에 0을 return
3. 채팅내용(chatContent), 프로젝트 번호(pnum)을 HashMap에 담는다.
4. HashMap에 담은 변수를 chatDao의 chatSubmit 함수에 매개변수로 넘긴다.
5. Chat table에 INSERT 질의 후 반환 된 return 값을 jsonStr로 return

```

private void chatList(){
    //1. chatRoom.jsp으로부터 받아온 프로젝트 번호(pnum), 채팅 구분(listType)을 변수로 저장
    String pnum = request.getParameter("pnum");
    String mid = session.getAttribute("m_id").toString();
    String listType = request.getParameter("listType");
    //2. HashMap에 변수를 저장한다.
    Map<String, String> map = new HashMap<String, String>();
    try {
        map.put("mid", URLDecoder.decode(mid, "UTF-8"));
        map.put("pnum", URLDecoder.decode(pnum, "UTF-8"));
    } catch (UnsupportedEncodingException e) {
        e.printStackTrace();
    }
    if(mid == null || mid.equals("") || pnum == null || pnum.equals("")
        || listType == null || listType.equals("")){
        jsonStr = "";
    }else if(listType.equals("ten")){
        try {
            map.put("listType", URLDecoder.decode(String.valueOf(10), "UTF-8"));
            List<Chat> tenList = chatDao.getTenList(map);
            jsonStr = chatTenList(tenList);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        }
    }else{
        map.put("listType", listType);
        //3. chatDao의 getChatList 함수에 매개변수로 map을 넘긴다.
        List<Chat> chatList = chatDao.getChatList(map);
        jsonStr = chatList(chatList);
    }
}

```

1. chatRoom.jsp으로부터 받아온 프로젝트 번호(pnum), 채팅 구분(listType)을 변수로 저장
2. HashMap에 변수를 저장한다.
3. chatDao의 getChatList 함수에 매개변수로 map을 넘긴다.

```

private String inputChatList(List<Chat> chatList){
    //1. StringBuilder를 이용하여 json형태로 append 한다.
    StringBuilder sb = new StringBuilder();
    sb.append("{\"result\":["");
    if(chatList.size() == 0){return "";}
    for(int i=0; i<chatList.size(); i++){
        //2. 매개변수로 받은 값을 replaceAll 함수를 사용하여 변환한다.
        sb.append("[{\"value\": \""+chatList.get(i).getC_mid().replaceAll(" ", "&nbsp;")
            .replaceAll("<", "&lt;").replaceAll(">", "&gt;").replaceAll("\n", "<br>")+\""},");
        sb.append("{\"value\": \""+chatList.get(i).getC_content().replaceAll(" ", "&nbsp;")
            .replaceAll("<", "&lt;").replaceAll(">", "&gt;").replaceAll("\n", "<br>")+\""},");
        sb.append("{\"value\": \""+chatList.get(i).getC_date()+\""}]");
        if(i != chatList.size()-1){
            sb.append(",");
        }
    }
    sb.append("], \"last\": \""+chatList.get(chatList.size()-1).getC_id()+\"");
    //3. append한 코드를 return 한다.
    return sb.toString();
}

```

1. StringBuilder를 이용하여 json형태로 append 한다.
2. 매개변수로 받은 값을 replaceAll 함수를 사용하여 변환한다.
3. append한 코드를 return 한다.

■ DAO & MAPPER PARAMETER : Map

```

public int chatSubmit(Map<String, String> map){
    return sqlSession.insert("chat.chatSubmit", map);
}

```

```

<insert id="chatSubmit" parameterType="Map">
    INSERT INTO chat VALUES(#{chatNum},#{pnum},#{mid},#{chatContent},default)
</insert>

```

1. Mapper id "chatSubmit"의 쿼리 실행 결과를 int로 return
2. INSERT 질의에 성공하면 1을 return 아니면 0을 return

```

public List<Chat> getChatList(Map<String, String> map) {
    return sqlSession.selectList("chat.getChatList", map);
}

```

```

<select id="getChatList" parameterType="Map" resultType="Chat">
    <![CDATA[
        SELECT * FROM chat WHERE c_id > #{listType} AND c_pnum = #{pnum} ORDER BY c_date
    ]]>
</select>

```

1. Mapper id "getChatList"의 쿼리 실행 결과를 List<Chat>에 담는다.
2. Id "getChatList" 는 최신 채팅 내용을 불러오는 select 질의

■ TARGET VIEW

→ 실시간 채팅

client

안녕하세요^^

2017-09-07 11:37:19

siin

네 반갑습니다 ^^

2017-09-07 11:38:22

client

안드로이드 경력이 어떻게 되시나요?

2017-09-07 11:40:27

siin

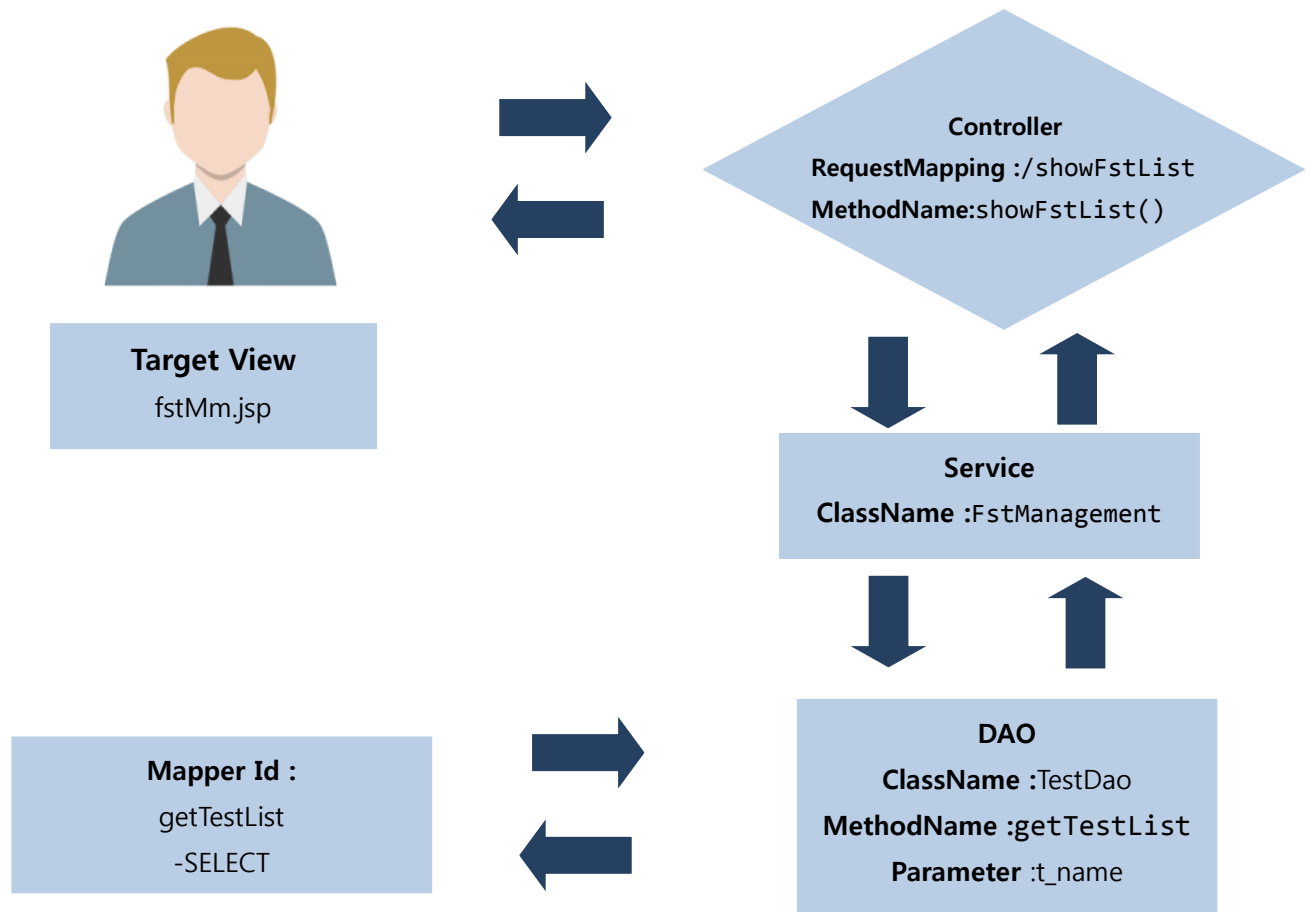
4~5년 되었습니다!!

2017-09-07 11:40:58

메시지를 입력하세요

전송

JOB CODE	B01	JOB NAME	자격시험 리스트	CLASS	TestController
PROCESS FLOW					



■ 개요

등록 되어있는 문제들의 리스트를 불러온다

■ CONTROLLER REQUEST: / showFstList

```

@RequestMapping(value = "/showFstList")
public ModelAndView showFstList() {
    System.out.println("시험 리스트");
    mav=new ModelAndView();
    mav=fm.execute(6);
    return mav;
}
  
```

1. serviceClass인 FstManagement 클래스의 excute의 (6)번 함수 : showFstList() 를 실행

■ SERVICE PARAMETER t_name

```
private void showFstList() { //관리자 시험 문제 리스트 보기
    mav=new ModelAndView();
    String view = null;
    List<Test> tlist = null;
    //선택된 카테고리의 값을 가져와 변수 t_name에 담아준다
    String t_name = req.getParameter("FstList");
    System.out.println(t_name);
    //t_name에 일치하는 값의 결과들을 select해주어 tlist에 담아준다
    tlist = tDao.getTestList(t_name);
    System.out.println(tlist);
    //세션이 남아있고 세션의 값이 admin일 경우
    if(ss!=null && ss.getAttribute("m_id").equals("admin")){
        if(tlist!=null){
            StringBuilder sb = new StringBuilder();
            //가져온 값의 리스트를 for문으로 돌리며 StringBuilder로 append 해준다
            for(int i=0; i<tlist.size(); i++){
                Test t=tlist.get(i);
                sb.append("<tr><td class = 't1'>" + t.getT_num() + "</td>");
                sb.append("<td class = 't2'>"
                    + "<a class='a2' href='showFstDetail?t_num="
                    + t.getT_num()+"'>" + t.getT_content() + "</a></td>");
                sb.append("<td class = 't3'>" + t.getT_answer() + "</td>");
                sb.append("<td class = 't4'>"
                    + "<a class='a4' href='goUpdateFst?t_num="
                    + t.getT_num()+"'>수정</a></td></tr>");
            }
            //StringBuilder에 담은 정보들을 addObject를 이용해 fstMm에 출력하여 보여준다.
            mav.addObject("tlist", sb.toString());
            mav.addObject("tname", t_name);
        }
        view="fstMm";
    }else{//세션의 값이 admin이 아닐경우
        //관리자가 아니라는 메시지를 얼럿창으로 띄워준다.
        mav.addObject("msg", "관리자가 아닙니다.");
        //home.jsp로 이동한다.
        view="home";
    }
    mav.setViewName(view);
}
```

1. Session의 저장된 아이디의 값이 admin일 경우
tlist의 내용을 StringBuilder를 이용하여 fstMm.jsp에 리스트를 찍어준다.
2. Session의 저장된 아이디의 값이 admin이 아닐경우
Home.jsp로 이동후 "관리자가 아닙니다." 라는 alert창을 띄어준다.

■ BEAN

```
public class Test {
    private int t_num;
    private String t_name;
    private String t_content;
    private String t_no1;
    private String t_no2;
    private String t_no3;
    private String t_no4;
    private int t_answer;
```

■ DAO & MAPPER PARAMETER :t_name

```
public List<Test> getTestList(String t_name) {
    return sqlSession.selectList("test.getTestList", t_name);
}
```

```
<select id = "getTestList" parameterType = "String" resultType = "test">
    SELECT * FROM TEST WHERE T_NAME = #{t_name} ORDER BY T_NUM
</select>
```

1. Mapper id "getTestList"의 쿼리 실행 결과를 (bean)test에 담는다.
2. 선택된 t_name과 일치하는 값의 결과를 select한후 t_num순으로 정렬하여 보여준다.

■ TARGET VIEW MODEL :fstMm.jsp

JSP

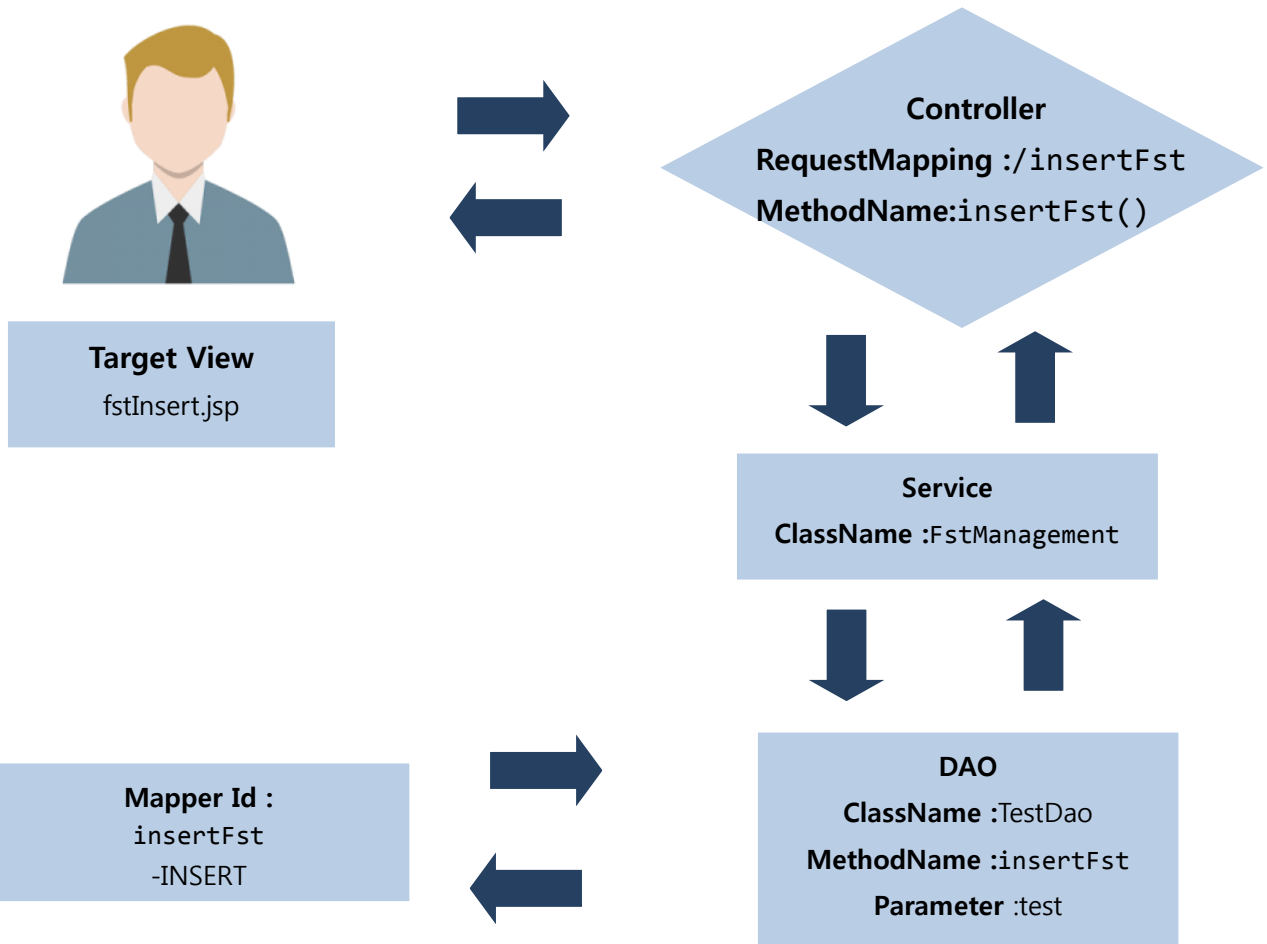
확인

jsp

문제 번호	문제 내용	답	수정 & 삭제
41	웹 또는 웹 서비스와 직접적인 관련이 없는 것은 무엇인가?	4	수정
42	동적인 웹 문서를 작성하는 기술과 관련이 없는 것은 무엇인가?	2	수정
43	HTTP 프로토콜에서 클라이언트의 요청에 서버가 응답하면 접속이 해제된다. 이것을 무엇이라 하는가?	4	수정
44	JSP 언어에 관한 설명으로 잘못된 것은?	4	수정
45	원하는 방식으로 인코딩 된 데이터를 메시지 몸체에 포함하여 전송하면서 파일을 요청하고자 하는 경우 사용된다	3	수정
46	Java 언어에서 제공되는 기본 자료형이 아닌 것은?	1	수정
47	키워드 final의 사용 예를 설명한 것이다. 잘못된 것은?	3	수정
48	System.out.print("\$+1000+0)와 System.out.print("\$+1000+0)의 출력 결과는 각각 무엇인가?	2	수정
49	어떤 클래스에서 protected 접근성을 가지는 필드와 메소드가 선언되어 있다. 이러한 필드나 메소드를 사용할 수 있는 위치가 아닌 것은?	3	수정
50	Interface A { int method(); }를 구현하는 클래스 B의 가장 간단한 정의는 무엇인가?	1	수정
51	Java에서 패키지 사용에 관한 설명이다. 잘못된 것은?	1	수정
52	Java 프로그램의 실행 중 보기와 같은 유형의 예외가 발생하였다고 가정하자. 예외 처리가 반드시 필요한 것은 무엇인가?	3	수정
53	JSP 기술을 사용하여 웹 어플리케이션을 개발하고자 할 때 설치해야 하는 것이 아닌 것은?	4	수정
54	WAR 파일에 대한 설명으로 잘못된 것은?	2	수정
55	Tomcat에 대한 설명으로 정확한 것은?	1	수정
56	JSP 태그의 종류와 형태가 잘못 짝지어진 것은?	2	수정
57	무엇일까?<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>	3	수정
58	다음은 스크립팅을 사용하지 않고 변수나 수식의 값을 표현하는 것이다. 무엇이라 하는가? <%= expr %>	1	수정
59	액션은 요청을 처리할 때 특정 기능을 수행하는 것이다. 모든 JSP 컨테이너가 구현해야 하는 표준 액션을 표현할 때, 액션 태그의 접두어로 사용되는 단어는 무엇인가?	4	수정
60	JSP 기술에 기초하여 웹 어플리케이션을 구축할 때, 구성 요소로 보기 힘든 것은?	4	수정

문제 추가

JOB CODE	B02	JOB NAME	자격시험 입력	CLASS	TestController
PROCESS FLOW					



■ 개요

1. DB에 시험문제를 추가한다.

■ CONTROLLER

REQUEST:/insertFst

```

@RequestMapping(value = "/goInsertFst")
public ModelAndView goInsertFst() {
    System.out.println("문제추가 페이지");
    mav=new ModelAndView();
    mav.setViewName("fstInsert");
    return mav;
}

@RequestMapping(value = "/insertFst")
public ModelAndView insertFst() {
    System.out.println("시험문제 추가");
    mav=new ModelAndView();
    mav=fm.execute(2);
    return mav;
}
  
```

1. serviceClass인 FstManagement 클래스의 excute의 (2)번 함수: insertFst() 를 실행

```

private void insertFst() { //관리자 시험 문제 추가
    String view = null;
    mav = new ModelAndView();
    Test test = new Test();
    //세션이 남아있고 세션의 값이 admin일 경우
    if(ss!=null && ss.getAttribute("m_id").equals("admin")){
        //가져온 정답의 번호를 t_answer에 담아준다.
        int t_answer = Integer.parseInt(req.getParameter("t_answer"));
        //DB에 저장된 t_num의 최대값에 +1 한후 test에 담아준다
        test.setT_num(tDao.getTestMaxNum()+1);
        //선택한 카테고리의 값을 test에 담아준다
        test.setT_name(req.getParameter("t_name"));
        //입력한 문제내용을 test에 담아준다
        test.setT_content(req.getParameter("t_content"));
        //위에서 받은 답의 숫자를 test에 담아준다
        test.setT_answer(t_answer);
        //보기 1번의 값을 입력받아 test에 담아준다
        test.setT_no1(req.getParameter("t_no1"));
        //보기 2번의 값을 입력받아 test에 담아준다
        test.setT_no2(req.getParameter("t_no2"));
        //보기 3번의 값을 입력받아 test에 담아준다
        test.setT_no3(req.getParameter("t_no3"));
        //보기 4번의 값을 입력받아 test에 담아준다
        test.setT_no4(req.getParameter("t_no4"));
        System.out.println(test);
        if(tDao.insertFst(test)!=0){
            System.out.println("문제추가성공");
            mav.addObject("msg", "문제추가성공");
        }else{
            System.out.println("문제추가실패");
            mav.addObject("msg", "문제추가실패");
        }
        view = "fstMm";
    }else{
        mav.addObject("msg", "관리자가 아닙니다.");
        System.out.println("관리자가 아닙니다.");
        view = "home";
    }
    mav.setViewName(view);
}

```

1. Session의 저장된 아이디의 값이 admin일 경우
(bean)test에 fstinsert.jsp에서 가져온 값을 담는다
2. Session의 저장된 아이디의 값이 admin이 아닐경우
Home.jsp로이동후 "관리자가 아닙니다." 라는 alert창을 띄어준다.

■ BEAN

```
public class Test {
    private int t_num;
    private String t_name;
    private String t_content;
    private String t_no1;
    private String t_no2;
    private String t_no3;
    private String t_no4;
    private int t_answer;
}
```

문제를 출제하는 Test 테이블의 bean

■ DAO & MAPPER PARAMETER :test

```
public int insertFst(Test test) {
    return sqlSession.insert("test.insertFst", test);
}
```

```
<insert id = "insertFst" parameterType = "test">
    INSERT INTO TEST VALUES(#{t_num},#{t_name},#{t_content},#{t_answer},
                             #{t_no1},#{t_no2},#{t_no3},#{t_no4})
</insert>
```

1. Mapper id "insertFst"의 쿼리 실행 결과를 (bean)test에 담는다.
2. 입력 받은 값을 insert한다.

■ TARGET VIEW MODEL :fstMm.jsp

JAVA ▼

문제내용

문제를 입력하시오.

1번

2번

3번

4번

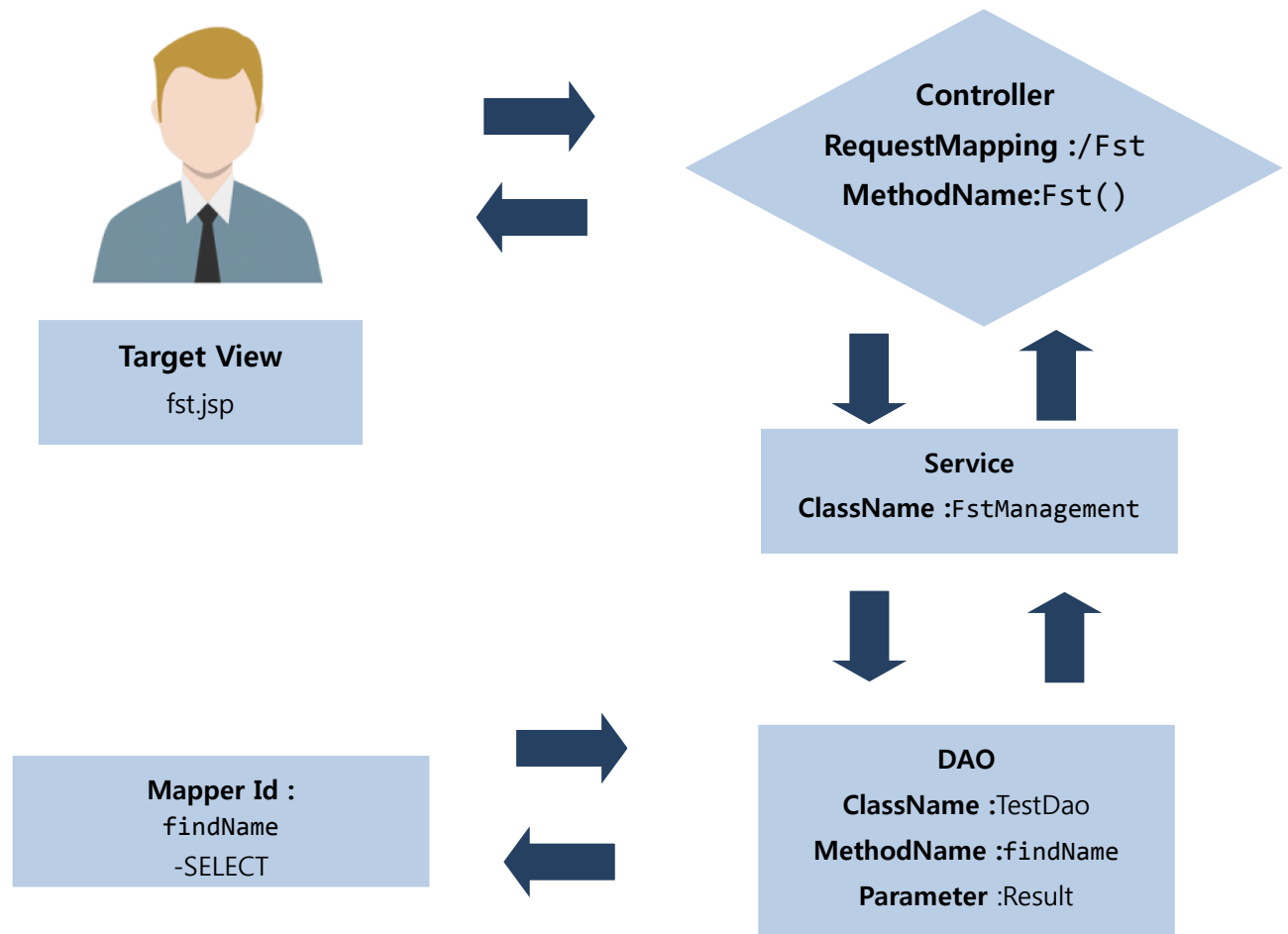
정답번호

1 ▼

확인

뒤로가기

JOB CODE	B03	JOB NAME	자격시험 페이지	CLASS	TestController
PROCESS FLOW					



■ 개요

1. 자격시험 합격 여부와 시험 볼 목록을 선택

■ CONTROLLER

REQUEST:/fst

```

@RequestMapping(value = "/fst")
public ModelAndView fst() {
    System.out.println("자격시험 페이지");
    mav=new ModelAndView();
    mav=fm.userExecute(4);
    return mav;
}

```

1. serviceClass인 FstManagement 클래스의 user.Excute의 (4)번 함수 : fst() 를 실행

■ SERVICE PARAMETER :Result

```
private void fst() {
    String view = null;
    mav = new ModelAndView();
    String a_mid = (String) ss.getAttribute("m_id");
    List<Result> rlist = null;
    Result res = new Result();
    StringBuilder sb = new StringBuilder();
    res.setRs_mid(a_mid);
    rlist = tDao.findName(res);
    if(ss!=null && ss.getAttribute("m_id")!=null){
    if(rlist!=null){
        for(int i=0; i<rlist.size();i++){
            Result rs = rlist.get(i);
            System.out.println(rs.getRs_tname());
            System.out.println(rs.getRs_pc());
            if(rs.getRs_pc()>50){
                sb.append("<tr><td>" + rs.getRs_tname() + "</td>"
                    + "<td>" + rs.getRs_pc() + "%</td><td>합격입니다.</td></tr>");
            }
        }
        mav.addObject("rlist", sb.toString());
        view="fst";
    }else{
        mav.addObject("msg", "로그인을 해주세요");
        view = "home";
    }
    mav.setViewName(view);
}
}
```

1. ModelAndView를 생성한다..
2. m_id의 시험 결과 값들을 DB에서 불러와 합격 여부를 표시해준다.
3. Session의 저장된 아이디가 Null이 아닐 경우
시험 볼수 있는 목록을 클릭할 시 해당하는 카테고리의 시험문제로 이동한다.
4. Session의 저장된 아이디가 Null일 경우
Home.jsp로 이동 후 "로그인을 해주세요." 라는 alert창을 띄어준다

■ BEAN

```
public class Result {
    private int rs_num;
    private String rs_mid;
    private String rs_tname;
    private int rs_pc;
}
```

문제를 풀고 난 뒤 결과를 알려주는 Result 테이블의 bean

■ DAO & MAPPER PARAMETER :Result

```
public List<Result> findName(Result res) {
    return sqlSession.selectList("test.findName",res);
}
```

```
<select id = "findName" parameterType = "String" resultType = "result">
    SELECT * FROM RESULT WHERE RS_MID = #{rs_mid}
</select>
```

1. Mapper id "fineName"의 쿼리 실행 결과를 result에 담는다.
2. 출력된 결과를 fst.jsp에 띄워준다.

■ TARGET VIEW MODEL :fst.jsp

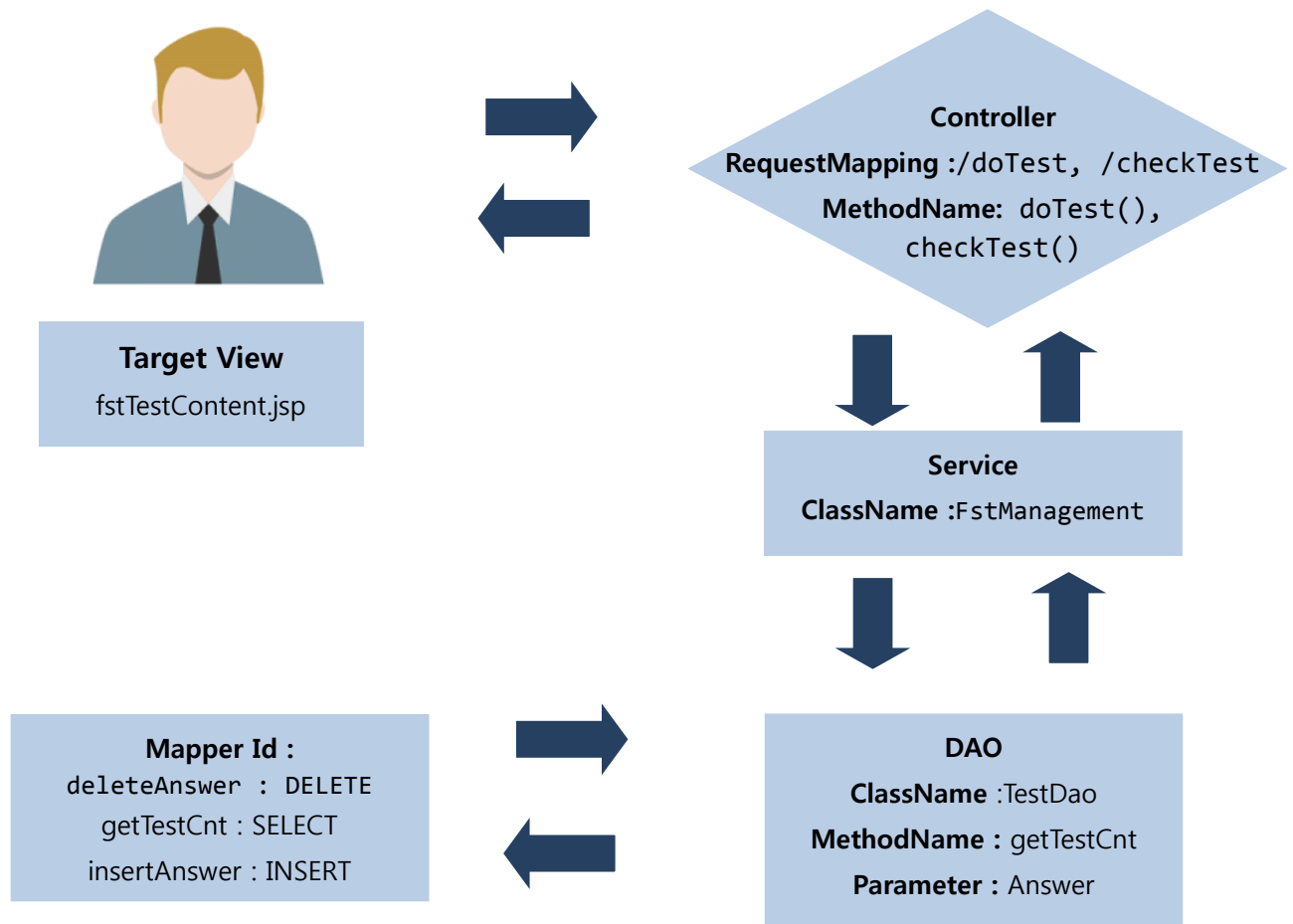
합격한 시험

과목	퍼센트	합격여부
java	70%	합격입니다.

자격시험 목록

JAVA	보러가기
HTML	보러가기
JSP	보러가기
C	보러가기
Android	보러가기

JOB CODE	B04	JOB NAME	자격시험 테스트	CLASS	TestController
PROCESS FLOW					



■ 개요

1. 이전 문제 답 입력 값 delete
2. 문제 답 카운트 select
3. 입력한 답 입력 값 insert

■ CONTROLLER

REQUEST:/doTest ,/checkTest

```

@RequestMapping(value = "/doTest", method = RequestMethod.POST)
public ModelAndView doTest() {
    System.out.println("시험 테스트 시작");
    mav=new ModelAndView();
    mav=fm.userExecute(2);
    return mav;
}

@RequestMapping(value = "/checkTest", method = RequestMethod.POST)
public ModelAndView checkTest() {
    System.out.println("시험 문제 푸는중");
    mav=new ModelAndView();
    mav=fm.userExecute(3);
    return mav;
}

```

■ BEAN

```

public class Answer {
    private int a_num;
    private int a_insertnum;
    private int a_check;
    private int a_tnum;
    private String a_mid;
    private String a_tname;
}

```

1. 문제의 답을 입력 받고 비교하는 Answer테이블의 bean

```

public class Test {
    private int t_num;
    private String t_name;
    private String t_content;
    private String t_no1;
    private String t_no2;
    private String t_no3;
    private String t_no4;
    private int t_answer;
}

```

2. 문제를 출제하는 Test 테이블의 bean

```

public class Result {
    private int rs_num;
    private String rs_mid;
    private String rs_tname;
    private int rs_pc;
}

```

3. 문제를 풀고 난 뒤 결과를 알려주는 Result 테이블의 bean

■ SERVICE

PARAMETER :Answer,Test,Result

```

private void doTest() { //문제 풀기 시작
    String view = null;
    mav = new ModelAndView();
    String t_name = req.getParameter("t_name");
    String a_mid = (String) ss.getAttribute("m_id");
    Answer ans = new Answer();
    ans.setA_tname(t_name);
    ans.setA_mid(a_mid);
    tDao.deleteAnswer(ans); //받은 카테고리 와 세션아이디와 일치하는 값들을 DB에서 지워준다.
    int tCnt = tDao.getTestCnt(ans); //DB answer 테이블에 입력된 값을 카운트 해준다.
    List<Test> tlist = null;
    int a[] = new int[10];
    Random r = new Random();
    StringBuilder sb = new StringBuilder();
    if(ss!=null && ss.getAttribute("m_id")!=null){ //세션이 남아있을경우
        tlist = tDao.getTestList(t_name); //문제 테이블에서 선택한 카테고리의 문제들을 가져와 tlist에 담아준다
        if(ss.getAttribute("No"+tCnt+"a")==null){ //answer에 입력된 값이 없을경우
            for(int i=0; i<10; i++){ //tlist 중에서 랜덤으로 값을 10개 가져온다
                a[i] = r.nextInt(tlist.size());
                for(int j=0; j<i; j++){ //중복되게 가져온 값들을 지우고 다시 뽑아준다.
                    if(a[i] == a[j]){
                        System.out.println("중복 제거");
                        i--;
                    }
                }
            }
            for(int k=0; k<10; k++){ //가져온 값들을 각각 번호를붙인 세션에 담아준다
                ss.setAttribute("No"+k+"a", a[k]);
                Test t = tlist.get(a[k]);
            }
            ss.setAttribute("No"+10+"a", 10);
        }
    }
    else{
        mav.addObject("msg", "로그인을 해주세요");
        view = "home";
    }
    int No = (Integer) ss.getAttribute("No"+tCnt+"a"); //카운터 수와 일치하는 번호의 세션의 번호를 가져온다
    Test t = tlist.get(No); //가져온 세션 번호에 일치하는 문제를 가져와 출력하며 보여준다.
    System.out.println("번호 : "+t.getT_num());
    if(tCnt<10){
        sb.append("<table class='tableFst'><tr class = 'tr01'><td class='tdName'>"
            + "<input type = 'text' id = 'a_tname' name = 'a_tname' value="+t.getT_name()+" readonly='readonly' />"
            + "</td><td colspan = '3' class='fstContent'><h3>"+t.getT_content()+"</h3></td></tr>");
        sb.append("<tr><td rowspan='4' class='td00'>문제"+(tCnt+1)+"</td><td class='td01'>1번 : </td>"
            + "<td class='td02'>"+t.getT_no1()+"</td><td class='td03'>"
            + "<input type='radio' name='answer' id='answer1' value='1' /></td></tr>");
        sb.append("<tr class = 'tr01'><td class='td01'>2번 : </td>"
            + "<td class='td02'>"+t.getT_no2()+"</td><td class='td03'>"
            + "<input type='radio' name='answer' id='answer2' value='2' /></td></tr>");
        sb.append("<tr><td class='td01'>3번 : </td>"
            + "<td class='td02'>"+t.getT_no3()+"</td><td class='td03'>"
            + "<input type='radio' name='answer' id='answer3' value='3' /></td></tr>");
        sb.append("<tr class = 'tr01'><td class='td01'>4번 : </td>"
            + "<td class='td02'>"+t.getT_no4()+"</td><td class='td03'>"
            + "<input type='radio' name='answer' id='answer4' value='4' /></td></tr>");
        sb.append("<tr><td colspan = '4' style='text-align:right;'>"
            + "<input id = 'a_tnum' 'type='hidden' name='a_tnum' value="+t.getT_num()+" readonly='readonly' />"
            + "<input type = 'button' value = '입력' id='check' /></td></tr></table>");
        sb.append("<div id='ViewTimer'></div>");
        ss.removeAttribute("No"+tCnt+"a"); //가져온 문제의 세션 번호를 비워 준다.
        view = "fstTestContent";
    }
    mav.addObject("tlist", sb.toString()); //문제를 tlist에 담아
    mav.setViewName(view); //fstTestContent에 띄어준다
}

```

1. Answer테이블에서 아이디와 카테고리가 일치하는 데이터를 다 지워준다 (카운트)
2. Test테이블에서 맞는 카테고리의 문제들을 가져와 랜덤으로 10문제를 뽑아준다 (중복제거)
3. 10번의 포문으로 세션에 번호를 주어 각 번호마다 랜덤으로 뽑은 문제번호를 넣는다.
4. 1번의 Answer테이블의 카운트와 일치하는 세션 번호의 문제를 fstTestContent.jsp에 출력

```

private void checkTest() {
    String view = null;
    mav = new ModelAndView();
    Answer ans = new Answer();
    List<Test> tlist = null;
    Test test = new Test();
    int a_insernum = 0;
    StringBuilder sb = new StringBuilder();
    if(req.getParameter("answer")==null){//타이머 기능에서 시간초가 다되었을경우
        a_insernum = 5;//5번 을 답으로 입력해준다. 정답은 1~4 까지 따라서 5는 오답처리
    }else{
        a_insernum = Integer.parseInt(req.getParameter("answer")); //타이머전에 값을 입력하면 입력된 값을 받는다.
    }
    int a_tnum = Integer.parseInt(req.getParameter("a_tnum"));
    String a_tname = req.getParameter("a_tname");
    test.setT_name(a_tname); //카테고리 이름
    test.setT_num(a_tnum); //문제 번호
    int t_answer = tDao.getT_answer(test); //문제의 정답을 가져온다.
    String a_mid = (String) ss.getAttribute("m_id");
    ans.setA_num(tDao.getAnswerMaxNum()+1); //Answer테이블의 문제 번호 +1
    ans.setA_mid(a_mid); //Answer테이블에 문제 준사람 아이디를 입력
    ans.setA_tnum(a_tnum); //Answer테이블에 문제번호 입력
    ans.setA_insernum(a_insernum); //Answer테이블에 입력한 값을 받아온다
    ans.setA_tname(a_tname); //Answer테이블에 문제 카테고리 값을 가져온다
    if(a_insernum == t_answer){ //문제의 정답과 준정답의 값을 비교하여
        int a_check = 1; //정답일 경우 Answer테이블에 1값을 입력
        ans.setA_check(a_check);
    }else{ int a_check = 0; //정답이 아닐 경우 Answer테이블에 0값을 입력
        ans.setA_check(a_check);}
    if(tDao.insertAnswer(ans)!=0){ //위의 값들을 Answer테이블에 insert 해준다.
        int iCnt = tDao.getTestCnt(ans); //Answer테이블에 아이디와 카테고리가 일치하는 입력된 답을 카운트한다.
        tlist = tDao.getTestList(a_tname); //tlist에 카테고리 및 일치하는 문제들을 담아둔다.
        int No = (Integer) ss.getAttribute("No"+tCnt+"a"); //카운터 번호와 일치하는 세션 번호의 값을 담아
        if(tCnt<=9){ //카운터가 9보다 작거나 같을경우
            ss.removeAttribute("No"+tCnt+"a"); //불러온 문제 세션번호를 지워주고
            Test t = tlist.get(No); //세션 번호에 일치하는 문제를 출력해준다
            sb.append("<table class='tableFst'><tr class = 'tr01'><td class='tdName'>"
                + "<input type = 'text' id = 'a_tname' name = 'a_tname' value="+t.getT_name()+" readonly='readonly' />"
                + "</td><td colspan = '3' class='fstContent'><h3>"+t.getT_content()+"</h3></td></tr>");
            sb.append("<tr><td rowspan='4' class='td00'>문제"+(tCnt+1)+"</td><td class='td01'>1번 : </td>"
                + "<td class='td02'>"+t.getT_no1()+"</td><td class='td03'>"
                + "<input type='radio' name='answer' id='answer1' value='1' /></td></tr>");

            sb.append("<tr class = 'tr01'><td class='td01'>2번 : </td>"
                + "<td class='td02'>"+t.getT_no2()+"</td><td class='td03'>"
                + "<input type='radio' name='answer' id='answer2' value='2' /></td></tr>");
            sb.append("<tr><td class='td01'>3번 : </td>"
                + "<td class='td02'>"+t.getT_no3()+"</td><td class='td03'>"
                + "<input type='radio' name='answer' id='answer3' value='3' /></td></tr>");
            sb.append("<tr class = 'tr01'><td class='td01'>4번 : </td>"
                + "<td class='td02'>"+t.getT_no4()+"</td><td class='td03'>"
                + "<input type='radio' name='answer' id='answer4' value='4' /></td></tr>");
            sb.append("<tr><td colspan = '4' style='text-align:right;'>"
                + "<input id = 'a_tnum' 'type='hidden' name='a_tnum' value="+t.getT_num()+" readonly='readonly' />"
                + "<input type = 'button' value = '입력' id='check' /></td></tr></table>");
            sb.append("<div id='ViewTimer'></div>");
        }
        if(tCnt>=10){ //카운터가 10개보다 크거나 같으면
            ss.removeAttribute("No"+tCnt+"a"); //세션을 지우고
            int sum = tDao.getSum(ans); //a_check가 1인 정보의 개수를 세어 정답 개수를 가져온다
            Result res = new Result();
            res.setRs_num(tDao.getResultMaxNum()+1);
            res.setRs_mid(a_mid); //Result 테이블에 준사람 아이디입력
            res.setRs_tname(a_tname); //Result 테이블에 준문제 카테고리 값 입력
            int rs_pc = (sum*10);
            res.setRs_pc(rs_pc); //Result 테이블에 정답개수 x 10 정답 비율 입력
            if(sum>5){ //답개수가 6개 이상일경우
                int r_num = tDao.selectResult(res); //Result 테이블에 저장된 rs_pc 입력값을 확인한다.
                if(r_num==0){
                    tDao.insertResult(res); //입력된 값이 없을경우 Result 테이블에 insert해준다.
                }
                if(r_num<rs_pc){ //저장된 값보다 새로입력 받는 값이 클경우
                    tDao.updateResult(res); //새로운 정보를 update해준다.
                }
            }
            sb.append("<table class='tableFst'><tr><td class='lastTd'>시험이 끝났습니다. 정답 비율 "+rs_pc+"% 입니다.</td></tr>");
            sb.append("<tr><td colspan = '3' class='lastA'><button class='endButton'>"
                + "<a href='fst' class='AA'>시험종료</a></button></td></tr></table>");
        } //시험이 끝나면 결과를 보여준다.
    }
    view = "fstTestContent";
    mav.addObject("tlist", sb.toString());
    mav.setViewName(view);
}

```

1. 문제를 풀 때 마다 값을 insert해주고 insert된 값의 카운트가 10개일 때 결과 창으로 이동
2. 정답이 6개이상일 때 Result 테이블에 insert 해주고 재시험결과가 더 좋으면 update

```

<script>
    function ajax(url) {
        var FormData = $('#checkTest').serialize();
        $.ajax({
            url : 'checkTest',
            type : 'post',
            data : FormData,
            success : function(data) {
                $('#Test').html(data);
            },
            error : function(error) {
                console.log(error);
            }
        }); //ajax End
    }
    $("#check").click(function(){
        if ( $("input[type=radio]:checked").length < 1 ) {
            alert('정답을 입력하십시오');
        } else if ( $("input[type=radio]:checked").length = 1 ) {
            $(ajax(checkTest)).submit;
        }
    });
    $("#goFst").click(function(){
        $("goFst").submit;
    });
</script>
<script>
    var SetTime = 10; // 최초 설정 시간(기본 : 초)
    function msg_time() { // 1초씩 카운트
        m = Math.floor(SetTime % 60) + "초"; // 남은 시간 계산
        var msg = "현재 남은 시간은 <font color='red'>" + m + "</font> 입니다.";
        document.all.ViewTimer.innerHTML = msg; // div 영역에 보여줌
        SetTime--; // 1초씩 감소
        if (SetTime < 0) { // 시간이 종료 되었으면..
            $(ajax(checkTest)).submit;
            alert("시간초과");
        }
    }
    window.onload = function TimerStart(){ tid=setInterval('msg_time()',1000) };
</script>

```

1. ajax로 문제를 풀 때 마다 그 페이지에 그대로 새로운 문제 출력
2. 문제를 푸는 제한시간을 두어 10초를 초과 할 시 타임오버 오답처리

■ DAO & MAPPER PARAMETER :Answer

```

public int insertAnswer(Answer ans) {
    return sqlSession.insert("test.insertAnswer", ans);
}

```

```

<insert id = "insertAnswer" parameterType = "answer">
    INSERT INTO ANSWER VALUES(#{a_num},#{a_mid},#{a_tnum},#{a_insertnum},#{a_check},#{a_tname})
</insert>

```

```

public int getSum(Answer ans) {
    return sqlSession.selectOne("test.getSum", ans);
}

```

```

<select id = "getSum" parameterType = "answer" resultType = "Integer">
    SELECT COUNT(*) FROM ANSWER WHERE A_TNAME = #{a_tname} AND A_MID = #{a_mid} AND A_CHECK = 1
</select>

```

java

시험을 치루기에 앞서 꼭 읽어주세요

1. 모든문제는 객관식 입니다.
2. 모든문제는 단수 정답입니다.
3. 이전 문제로 되돌아갈수 없습니다.
4. 합격은 60%이상의 정답을 맞춰야 합니다.
5. 문제를 다 풀면 맞은 %만을 알려드립니다.

시험 시작

jsp

원하는 방식으로 인코딩 된 데이터를 메시지 몸체에 포함하여 전송하면서 파일을 요청하고자 하는 경우 사용된다

1번 : GET ☐

2번 : HEAD ☐

3번 : POST ☐

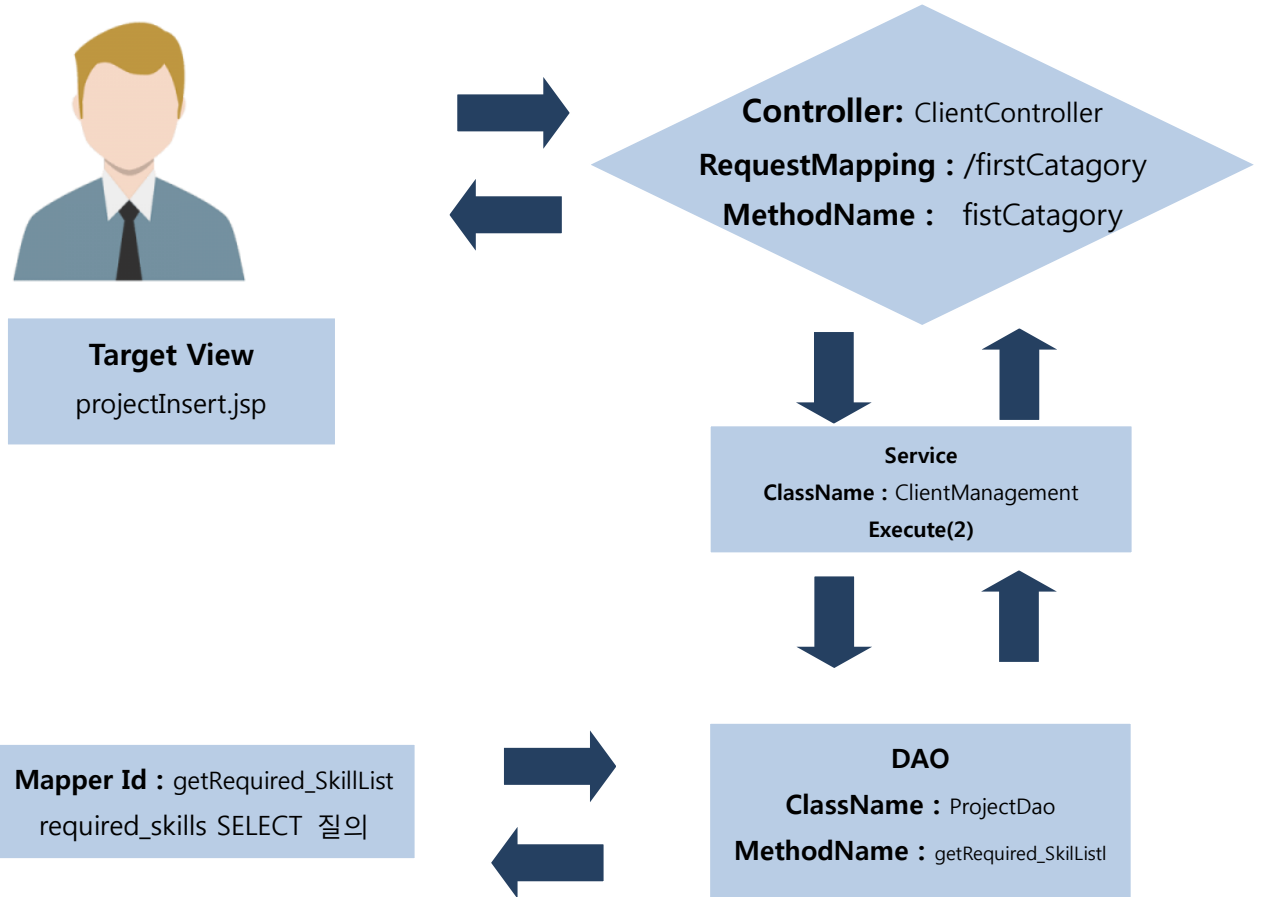
4번 : OPTIONS ☐

문제1

입력

현재 남은 시간은 6초 입니다.

JOB CODE	C01	JOB NAME	세부 기술 선택	CLASS	ClientManagement
PROCESS FLOW					



■ 개요

1. 세부기술을 입력하기 위한 선택
2. required_skills SELECT
3. TargetView: projectInsert.jsp

■ CONTROLLER

REQUEST : /firstCategory

```

@RequestMapping(value="/firstCategory", method = RequestMethod.GET)
public ModelAndView firstCategory(){
    mav = new ModelAndView();
    mav=cm.execute(2);//ClientManagement 클래스의 execute(2)(setRequired_Skill() 메서드) 함수를 실행
    return mav;//ModelAndView를 return
}

```

1. Serviceclass인 **ClientManagement** 클래스의 execute(2)(setRequired_Skill() 메서드) 함수를 실행
2. ModelAndView를 return

■ SERVICE

```

private void setRequired_Skill() { //세부기술 받아오는 메서드
    String view=null;
    mav=new ModelAndView();//ModelAndView를 생성
    List<Required_Skill> slist=null;//Required_Skill Bean을 담을하는 slist 생성
    slist=pDao.getRequired_SkillList();//slist에 pDao의 getRequired_SkillList() 함수의 결과값을 담음
    System.out.println(slist);

    if(slist!=null){
        StringBuilder sb = new StringBuilder();
        sb.append("<input type='hidden' value='' name='p_plnum[]' onClick='CountChecked(this)' id='inter' />");//제프리스트의 복수값을 배열로 받아 저장하기 위한 숨겨진 input
        for(int i=0; i<slist.size(); i++){//slist.size()만큼 StringBuilder에 HTML코드를 append
            Required_Skill rs=slist.get(i);//slist에 저장된 값을 Required_skill Bean을 이용하기 위한 객체 생성
            System.out.println(rs.getRs_plnum());
            sb.append("<input type='checkbox' value='"+rs.getRs_plnum()+"' name='p_plnum[]' id='inter' onClick='CountChecked(this)' />"+rs.getRs_plnum());
            //제프리스트 생성
            sb.append("/");
        }
        mav.addObject("slist", sb.toString());//append한 HTML코드를 ModelAndView에 addObject
    }
    view="projectInsert";
    mav.setViewName(view);//Target View를 ModelAndView에 setViewName
}

```

1. Required_Skill Bean을 활용하는 **slist**에 pDao의 getRequired_SkillList() 함수의 결과값을 담는다.
2. slist에 결과값이 담기면 slist.size()만큼 **StringBuilder**에 HTML 코드를 **append**
3. append한 HTML 코드를 ModelAndView에 **addObject**
4. Target View(projectInsert.jsp)를 ModelAndView에 **setViewName**

■ Bean

```

package com.steppe.nomad.bean;

public class Required_Skill {
    private int rs_id;//세부언어 번호
    private String rs_plnum;//세부언어 품목

    public int getRs_id() {
        return rs_id;
    }
    public void setRs_id(int rs_id) {
        this.rs_id = rs_id;
    }
    public String getRs_plnum() {
        return rs_plnum;
    }
    public void setRs_plnum(String rs_plnum) {
        this.rs_plnum = rs_plnum;
    }
}

```


■ DAO & MAPPER

```
public int getProjectMaxNum() { //mapper id "getRequired_SkillList"의 실행 결과를 List<required_Skill>의 값을 담음
    return sqlSession.selectOne("project.getProjectMaxNum");
}
```

```
<select id="getRequired_SkillList" resultType="required_skill">
    select * from required_skills
</select><!-- Required_Skill레이블의 세부기술을 불러오는 SELECT -->
```

1. mapper id **"getRequired_SkillList"**의 쿼리 실행 결과를 **List<Required_Skill>**에 담는다.
2. Id **"getRequired_SkillList"**는 세부기술을 불러오는 **select** 질의

■ TARGET VIEW

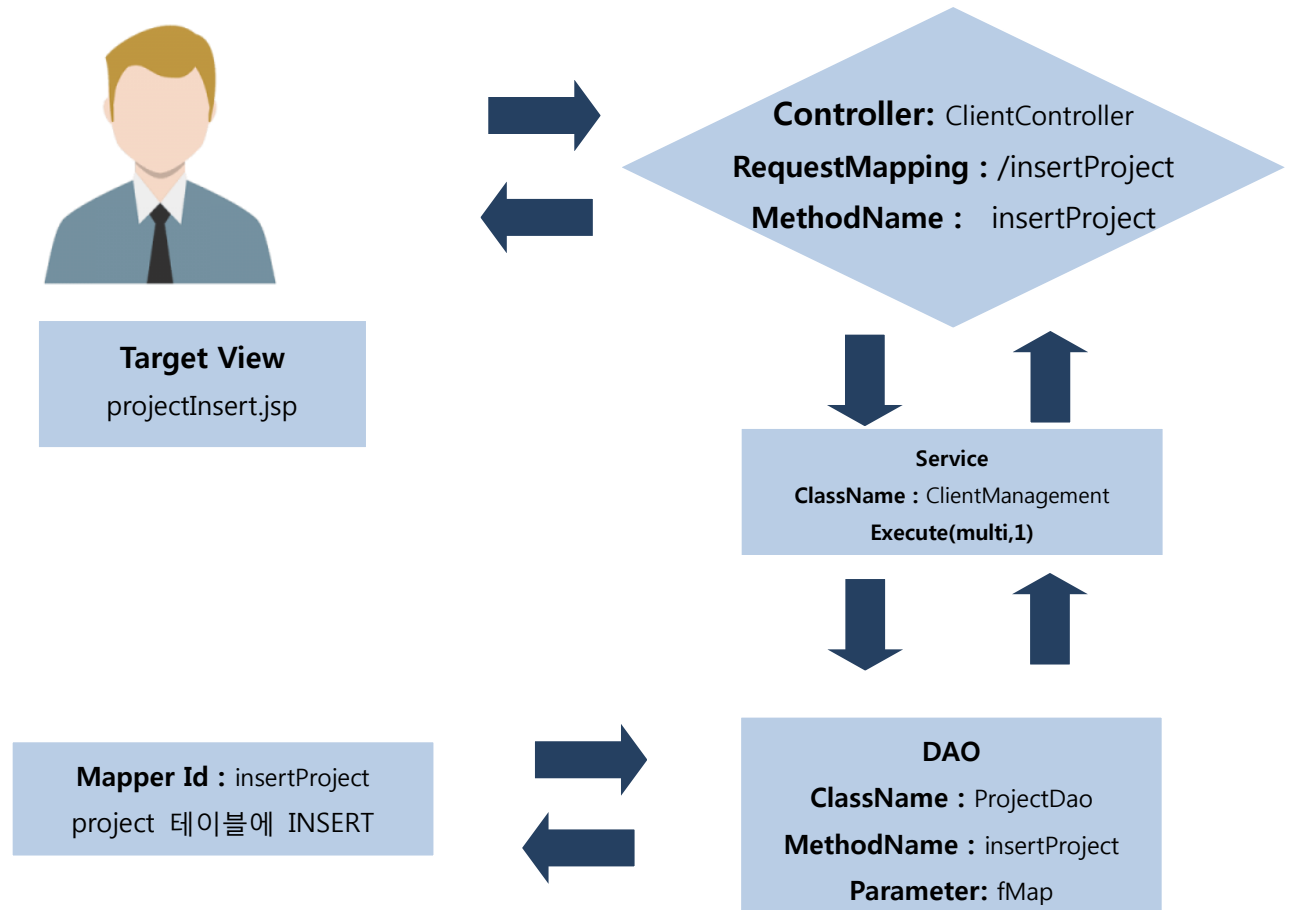
세부기술

세부기술 선택(최대3개)

☐파워포인트 ☐CAD/ ☐JAVA/ ☐C/ ☐C++/ ☐자바스크립트/ ☐SQL/ ☐CSS/ ☐PHP/ ☐IONIC/ ☐PYTHON/ ☐WPF/

required_skills에서 받아온 값을 체크박스 형태로 출력

JOB CODE	C02	JOB NAME	프로젝트 등록	CLASS	ClientManagement
PROCESS FLOW					



■ 개요

1. 프로젝트 등록
2. project INSERT
3. TargetView: projectInsert.jsp

■ CONTROLLER

REQUEST : /insertProject

```

@RequestMapping(value="/insertProject", method = RequestMethod.POST)//프로젝트 등록
public ModelAndView insertProject(MultipartHttpServletRequest multi){
    System.out.println("insertProject");
    mav = new ModelAndView();
    mav=cm.execute(multi,1);//ClientManagement 클래스의 execute(multi,1)호출(insertProject(MultipartHttpServletRequest multi) 메서드)를 실행
    return mav;//ModelAndView를 return
}
  
```

1. Serviceclass인 **ClientManagement** 클래스의 execute(multi,1) 함수
(insertProject(MultipartHttpServletRequest multi) 메서드)를 실행
2. ModelAndView를 return

■ SERVICE PARAMETER : multi

```
private void insertProject(MultipartHttpServletRequest multi) { //프로젝트 등록 메서드
    String mid = session.getAttribute("m_id").toString(); //세션에 저장되어있던 m_id를 받아와 mid에 저장
    String pc1_name=multi.getParameter("pc1_name");//1차 카테고리
    String pc2_name=multi.getParameter("pc2_name");//2차 카테고리

    int p_budget=Integer.parseInt(multi.getParameter("p_budget")); //예산
    String p_term=multi.getParameter("p_term");//프로젝트 기간
    String p_title=multi.getParameter("p_title");//제목
    String p_content=multi.getParameter("p_content");//내용
    int check=Integer.parseInt(multi.getParameter("fileCheck")); //파일등록 확인
    String p_deadline=multi.getParameter("p_deadline");//프로젝트 입출 줄을 시간
    String p_plnum0=multi.getParameter("p_plnum0");//첫번째 세부기술
    String p_plnum1=multi.getParameter("p_plnum1");//두번째 세부기술
    String p_plnum2=multi.getParameter("p_plnum2");//세번째 세부기술
    int p_person=Integer.parseInt(multi.getParameter("p_person")); //프로젝트 인원
    System.out.println("check="+check); //1이면 첨부됨
    Map<String, Object> fMap=new HashMap<String, Object>();
    //UploadFile.java에 있는 파일 업로드 메서드인 public Map<String,Object> fileUp(MultipartHttpServletRequest multi)를 사용위한 객체 생성
    Map<String, String> bmMap = new HashMap<String, String>();
    int bookmarkNum = 0;
    if(pbDao.bookmarkCount()!=0){
        bookmarkNum = pbDao.bookmarkMaxNum()+1;
    }else{
        bookmarkNum = 1;
    }
    if(check==1){
        UploadFile upload=new UploadFile();
        //서버에 파일을 업로드 한 뒤,
        //첨부 부여
        //크리치널 파일업, 시스템 파일업을 리턴 * Map에 저장

        fMap=upload.fileUp(multi);

        System.out.println(fMap);
    }
    Project project=new Project();//Project Bean들을 위한 객체 생성
    if(pDao.getProjectCount() != 0){ //프로젝트 존재여부 판별
        project.setP_num(pDao.getProjectMaxNum()+1); //등록된 프로젝트 번호를 부여하기 위해 DB에 저장된 프로젝트 번호에 1을 더함
    }else{
        project.setP_num(1); //최초 프로젝트 이면 등록번호에 1을 부여
    }
}
```

```

//project bean에 multi로 받아온 파라미터를 저장
project.setP_pc1name(pc1_name);
project.setP_pc2name(pc2_name);
project.setP_mid(session.getAttribute("m_id").toString());
project.setP_budget(p_budget);
project.setP_term(p_term);
project.setP_title(p_title);
project.setP_content(p_content);
project.setP_deadline(p_deadline);
project.setP_plnum0(p_plnum0);
project.setP_plnum1(p_plnum1);
project.setP_plnum2(p_plnum2);
project.setP_person(p_person);
project.setP_status(1);
//fMap에 파라미터가 저장된 Bean값을 저장
fMap.put("p_num", project.getP_num());
fMap.put("pc1_name", project.getP_pc1name());
fMap.put("pc2_name", project.getP_pc2name());
fMap.put("p_mid", project.getP_mid());
fMap.put("p_budget", project.getP_budget());
fMap.put("p_term", project.getP_term());
fMap.put("p_title", project.getP_title());
fMap.put("p_content", project.getP_content());
fMap.put("p_deadline", project.getP_deadline());
fMap.put("p_plnum0", project.getP_plnum0());
fMap.put("p_plnum1", project.getP_plnum1());
fMap.put("p_plnum2", project.getP_plnum2());
fMap.put("p_person", project.getP_person());
fMap.put("p_status", project.getP_status());

bmMap.put("pb_num", String.valueOf(bookmarkNum));
bmMap.put("pb_pnum", String.valueOf(project.getP_num()));
bmMap.put("pb_mid", mid);

mav=new ModelAndView();
String view=null;

System.out.println(fMap);

if(pDao.insertProject(fMap)!=0){//fMap에 저장된 값을 project 테이블에 Insert
    System.out.println("북마크 db삽입1");
    pbDao.bookmarkInsert(bmMap);
    System.out.println("북마크 db삽입2");
    view="redirect:goMyPageCI";//insert성공시 goMyPageCI url 생성
}else{
    view="redirect:goAddProject";//insert실패시 goAddProject url 생성
}
mav.setViewName(view);
}

```

1. mid에 session 값 m_id를 받아 String mid에 저장
2. 파일업로드를 가능하게 하는 정적메소드인 **MultipartHttpServletRequest**를 이용해 파라미터를 저장
3. **UploadFile.java**에 있는 파일 업로드 메서드인 **public Map<String,Object> fileUp(MultipartHttpServletRequest multi)**를 사용 위해 객체 생성
4. **fMap**에 **MultipartHttpServletRequest**를 활용한 파라미터를 업로드 할 수 있는 권한 부여
5. **project** 테이블에 번호를 받아오는 **pDao.getProjectMaxnum()**실행 후 번호가 0이 아니라면 프로젝트 번호는 최대 프로젝트 번호에 1을 더하고 0이라면 1을 준다.
6. 입력 받은 값을 **Project Bean**에 저장
7. **fMap**에 **Project Bean**에 저장된 값을 입력
8. **ModelAndView**객체 생성
9. **fMap**를 파라미터로 하는 **pDao의 insertProject**를 실행
10. 입력이 성공하면 **goMyPageCI URL** 생성 실패하면 **goAddProject URL** 생성

■ Bean

```
package com.steppe.nomad.bean;

public class Project {
    private int p_num; //프로젝트 번호
    private String p_pc1name; //1차 카테고리 아이디
    private String p_pc2name; //2차 카테고리 아이디
    private String p_mid; //항목 아이디
    private int p_budget; //프로젝트 예산
    private String p_term; //프로젝트 기간
    private String p_title; //프로젝트 제목
    private String p_content; //프로젝트 내용
    private String p_filename; //프로젝트 첨부파일
    private String p_deadline; //프로젝트 마감일
    private String p_plnum; //프로젝트 필요 언어
    private String p_plnum0; //초반과 세부기술
    private String p_plnum1; //중반과 세부기술
    private String p_plnum2; //세반과 세부기술
    private int p_person; //프로젝트 인원
    private int p_status; //프로젝트 상태
    private String p_status2;
    private int p_vol; //프로젝트 인원수
    private int p_bookmark; //
```

■ DAO & MAPPER PARAMETER : fMap

```
public int getProjectCount() { //프로젝트 존재여부 확인
    return sqlSession.selectOne("project.getProjectCount");//mapper id "getProjectCount"의 쿼리 실행 결과를 getProjectCount에 담음
}

public int getProjectMaxNum() { //mapper id "getRequired_SkillList"의 실행 결과를 List<required_Skill>의 결과를 담음
    return sqlSession.selectOne("project.getProjectMaxNum");
}

public int insertProject(Map<String, Object> fMap) { //mapper id "insertProject"에 fMap로 받아온 파라미터를 넘김
    return sqlSession.insert("project.insertProject", fMap);
}

<select id="getProjectCount" resultType="Integer">
    SELECT COUNT(*) FROM PLIST
</select><!-- project 테이블에 프로젝트가 존재하는지 확인 -->

<select id="getProjectMaxNum" resultType="Integer">
    select NVL(MAX(p_num),0) FROM project
</select><!-- 프로젝트 번호의 최대값을 확인// 번호가 null이면 0으로 대체 -->

<insert id="insertProject" parameterType = "Map">
    INSERT INTO project VALUES(#{p_num},#{pc1_name},#{pc2_name},#{p_mid},#{p_budget},#{p_term},#{p_title},#{p_content}
    ,#{sysFileName},#{p_deadline},#{p_plnum0},#{p_plnum1}, #{p_plnum2}, #{p_person}, #{p_status}, default, default)
</insert><!-- 프로젝트 테이블에 Map으로 받아온 값을 Insert -->
```

1. DAO

- 1) mapper id "getProjectCount"의 쿼리 실행 결과를 **getProjectCount()**에 담는다.
- 2) mapper id "getProjectMaxNum"의 쿼리 실행 결과를 **getProjectMaxNum()**에 담는다.
- 3) mapper id "insertProject"에 **fMap**로 받아온 파라미터를 넘김

2. Mapeer

- 1) Id "getProjectCount"는 **project 테이블의** 테이블 존재여부를 **SELECT**
- 2) Id "getProjectMaxNum"은 **project 테이블의** 테이블 번호를 **SELECT**
- 3) Id "insertProject"는 **Map**으로 받은 파라미터를 project 테이블에 **INSERT**

■ TARGET VIEW

1차 카테고리 ▼			기간(월)
2차 카테고리 ▼			0
제목	제목을 입력하여 주세요		
내용			
파일 첨부	파일 선택 선택된 파일 없음		
입찰 마감	연도-월-일		
예산비용	0만원		

프로젝트 등록하는 페이지

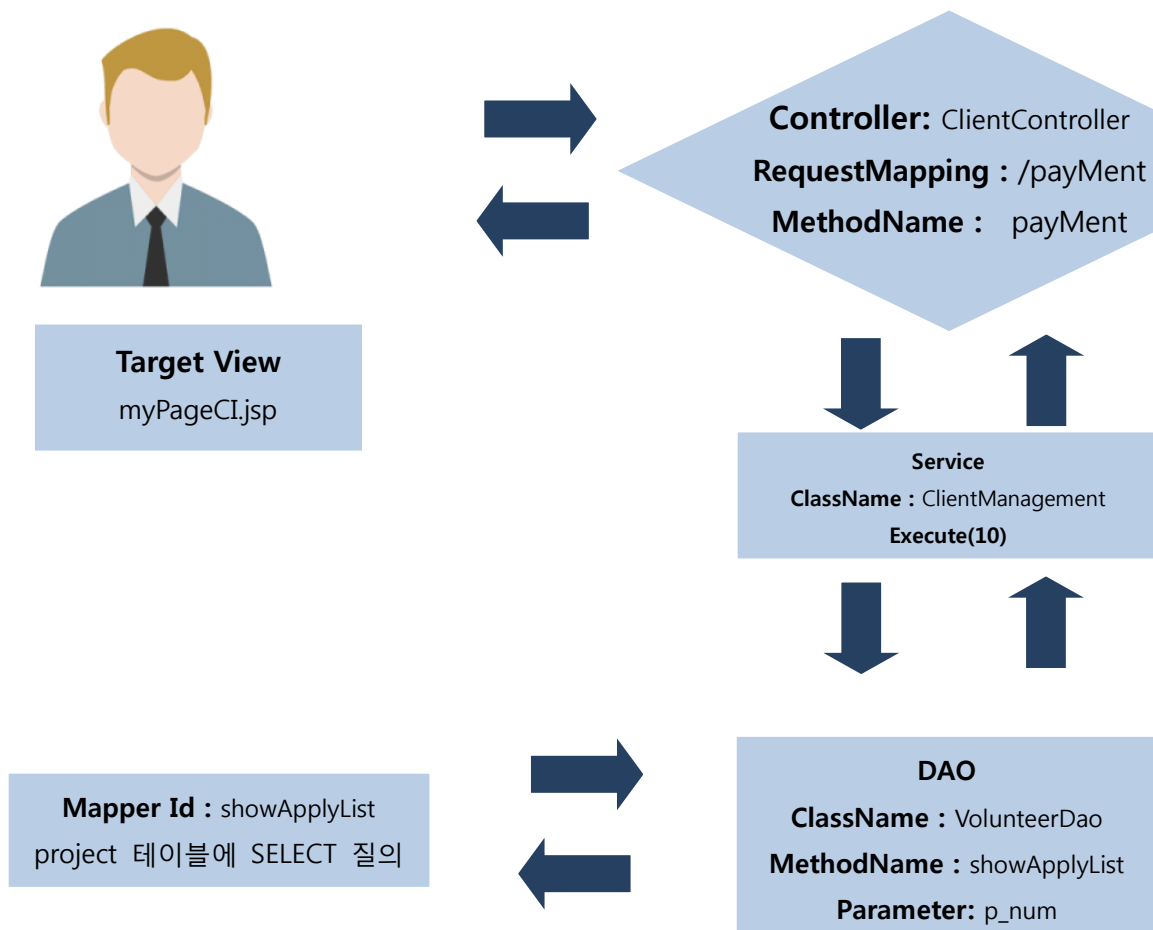
프로젝트 목록

번호	제목	지원자 수	상태
1	간단한 교통앱을 만들고 싶습니다.	0	대기중

삭제

등록에 성공한 본인의 프로젝트 목록을 보여주는 페이지

JOB CODE	C03	JOB NAME	결제	CLASS	ClientManagement
PROCESS FLOW					



■ 개요

1. 입찰가와 입찰시간을 오름차순으로 정렬해 프로젝트 인원에 맞는 지원자만 Select
2. Volunteer Select
3. Target View: myPageCI

■ CONTROLLER

REQUEST : /payMent

```

@RequestMapping(value="/payMent")//결제하기
public ModelAndView payMent(){
    System.out.println("payMent메서드 실행");
    mav = new ModelAndView();
    mav=cm.execute(10);//ClientManagement 클래스의 execute(10) 함수(payMent 메서드를 실행)실행
    return mav;
}
  
```

1. Serviceclass인 **ClientManagement** 클래스의 execute(10) 함수를 실행
2. ModelAndView를 return

■ SERVICE

PARAMETER : p_num

```
private void payMent() { //결제하기
    mav=new ModelAndView();
    String view=null;
    int p_num=Integer.parseInt(req.getParameter("p_num")); //프로젝트 번호 파라미터를 받아옴
    if(session!=null && session.getAttribute("m_id")!=null){
        List<Volunteer> vList=null; //Volunteer Bean을 활용하는 vList객체 생성
        System.out.println(p_num);
        vList=vDao.showApplyList(p_num); //p_num을 매개값으로 해당프로젝트에 등록된 자원자 리스트를 받아옴
        System.out.println(vList);

        if(vList!=null){
            StringBuilder sb = new StringBuilder();
            for(int i=0; i<vList.size(); i++){ //프로젝트 인원수 를 받아오기 위해
                Volunteer v=vList.get(i); //vList에 들어있는 값 중 p_person은 Volunteer Bean의 p_person에 값이 들어있음
                System.out.println(v.getP_person());
                for(int a=0; a<v.getP_person(); a++){ //프로젝트 인원수 만큼 자원자를 받아오기 위해 //인원 수 만큼 StringBuilder에 HTML코드를 append
                    Volunteer vl=vList.get(a);
                    System.out.println("a");
                    sb.append("<input type='hidden' value=''+v.getId()+"' name='v_mid' onClick='CountChecked(this)' id='vmid' />");
                    sb.append("<tr><td><input type='hidden' value=''+vl.getV_pnum()+"' name='v_pnum' />"+vl.getV_pnum()+"</td>");
                    sb.append("<td>"+vl.getV_num()+"</td>");
                    sb.append("<td>"+vl.getV_mid()+"</td>");
                    System.out.println(vl.getV_mid());
                    sb.append("<td>"+vl.getV_bid()+"</td>");
                    sb.append("<td><input type='checkbox' value=''+vl.getV_mid()+"' name='v_mid' id='vmid' "
                        + " onClick='CountChecked(this)' /></td></tr>");
                }
            }
            mav.addObject("vList", sb.toString()); //append한 HTML코드를 ModelAndView에 addObject
        }
        view="applyList"; //성공하면 applyList.jsp로 이동
    }else{
        view="home"; //실패하면 home.jsp로 이동
    }
    mav.setViewName(view);
}
```

1. ModelAndView 객체 생성
2. 파라미터로 프로젝트 번호인 p_num을 받아와 int p_num에 저장
3. Volunteer Bean을 활용하는 List인 vList에 vDao의 showApplyList(p_num) 메서드의 결과값을 받음
4. vList에 결과값이 담기면 **StringBuilder**객체 생성
5. vDao에서 받아온 값 중 프로젝트 인원인 p_person을 활용하기 위해 Volunteer v=vList.get(i)생성
6. **p_person**값 만큼 **StringBuilder**에 HTML 코드를 **append**
7. append한 HTML 코드를 ModelAndView에 **addObject**
8. **applyList.jsp**를 ModelAndView에 **setViewName**

■ Bean

```
public class Volunteer {
    private int rownum;
    private int v_num; //자원자 번호
    private int v_ptteam; //팀 결성 여부
    private int v_pnum; //프로젝트 번호
    private String v_mid; //자원자 아이디
    private int v_bid; //입찰액
    private String v_time; //입찰 시간
    private String p_title; //프로젝트명
    private String p_mid; //프로젝트 아이디
    private int p_person; //인원수
    private String v_mid0; //프로젝트 자원자1
    private String v_mid1; //자원자2
    private String v_mid2; //자원자3
    private String v_mid3; //자원자4
    private String v_mid4; //자원자5
}
```


■ DAO & MAPPER PARAMETER : p_num

```
public List<Volunteer> showApplyList(int p_num) { //mapper id "showApplyList"의 쿼리 실행 결과를 List<Volunteer>에 담음
    return sqlSession.selectList("volunteer.showApplyList",p_num);
}
```

```
<![CDATA[
select rownum as 작업자, v.v_num, v.v_mid, v.v_bid, v.v_pnum, v.v_ptteam, p.p_person from volunteer v inner join project p on v.v_pnum = p.p_num
where v.v_pnum=#{p_num} order by v.v_bid, v.v_time
]]>
</select><!-- p_num과 일치하는 프로젝트에 지원한 프리랜서를 입찰가와 입찰시간을 기준으로 오름차순으로 정렬해 Select -->
```

1. Dao

1) mapper id "showApplyList"의 쿼리 실행 결과를 **List<Volunteer>**에 담는다.

2. Mapper

1) 파라미터인 p_num과 일치하는 프로젝트에 지원한 프리랜서를 입찰가와 입찰시간을 기준으로 오름차순으로 정렬해 Select

■ TARGET VIEW

프로젝트 목록

번호	제목	지원자 수	상태
1	간단한 교통앱 만들고 싶습니다.	1	대기중

삭제

결제

해당 프로젝트 지원자가 해당 프로젝트 인원 이상이 되면 결제를 시작할 수 있는 페이지

지원자 리스트

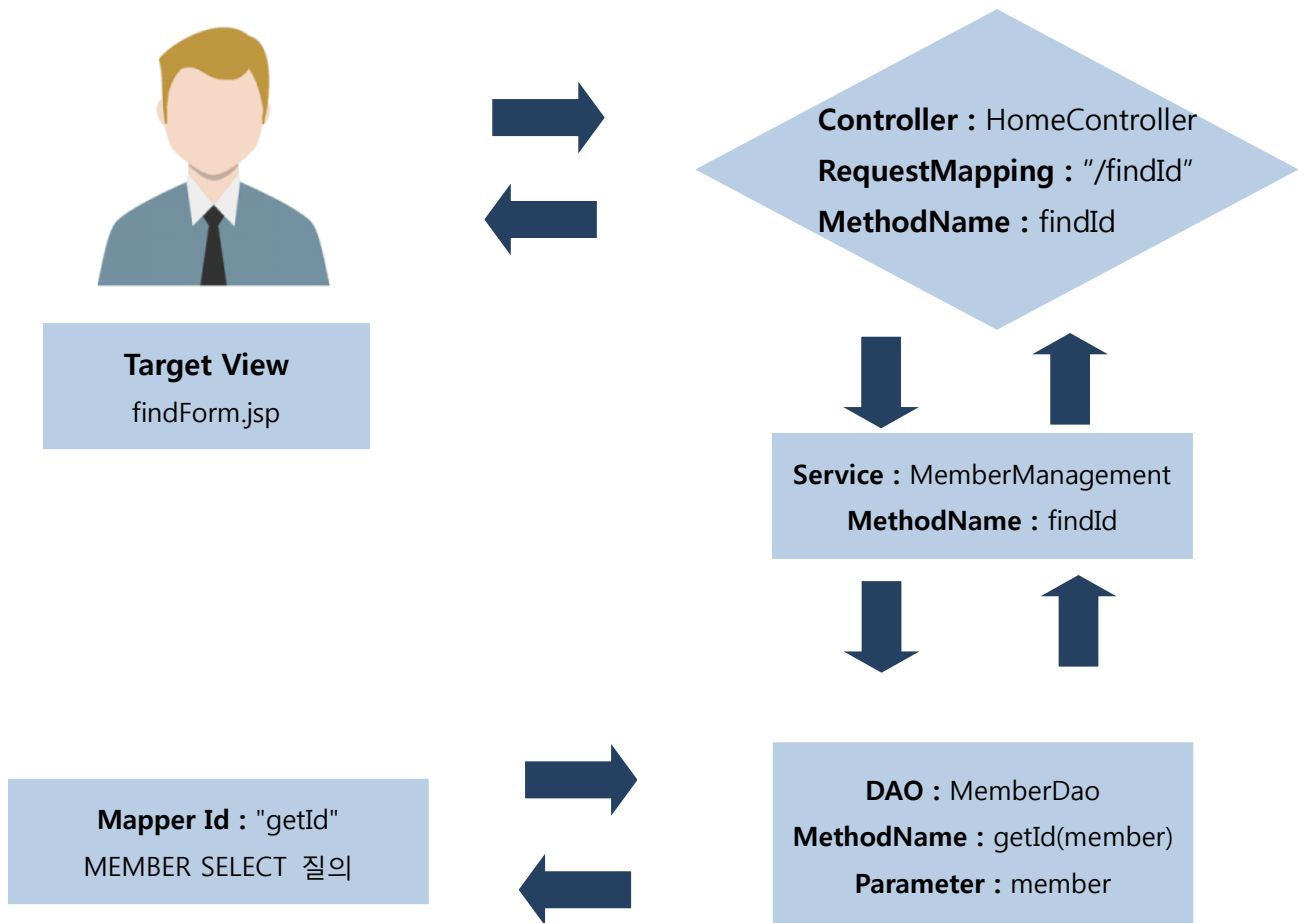
프로젝트 번호	지원자 번호	지원자	입찰액	선택
1	1	siin	190	<input checked="" type="checkbox"/>

결제하기

뒤로가기

입찰가와 입찰시간에 따라 오름차순으로 프로젝트 인원만큼 출력되어 선정하고 결제 페이지

JOB CODE	D01	JOB NAME	아이디 찾기	CLASS	MemberManagement
PROCESS FLOW					



■ 개요

1. 회원의 이름과 이메일 확인하여 회원의 아이디를 이메일로 전송.
2. MEMBER SELECT

■ CONTROLLER

REQUEST : /findId

```
//아이디 찾기
@RequestMapping(value = "/findId", method = RequestMethod.POST)
public ModelAndView findId(String m_name, String m_email) {
    System.out.println("아이디 찾기");
    mav = mm.execute(m_name, m_email, 1);
    return mav;
}
```

1. 회원이 입력한 이름(m_name)과 이메일(m_email)값을 모델에 담아 넘겨준다.
2. ServiceClass인 MemberManagement 클래스의 excute(m_name, m_email, 1) 함수를 실행.
3. ServiceClass에서 findId(m_name, m_email) 함수를 실행
4. ModelAndView로 return.

■ SERVICE PARAMETER : m_name, m_email

```
private void findId(String m_name, String m_email) {
    mav = new ModelAndView();

    String name = request.getParameter("m_name");
    String email = request.getParameter("m_email");

    System.out.println("name="+name);
    System.out.println("email="+email);

    Member member = new Member();
    member.setM_name(name);
    member.setM_email(email);

    // 메일 보내기
    String id = mDao.getId(member);

    //일반 텍스트메일
    SimpleMailMessage simpleMailMessage = new SimpleMailMessage();
    simpleMailMessage.setFrom("emailtest20170314@gmail.com");
    simpleMailMessage.setTo(email);
    simpleMailMessage.setSubject("Steppe 아이디 찾기 입니다.");
    simpleMailMessage.setText("귀하의 아이디는 "+id+"입니다.");

    javaMailSenderImpl.send(simpleMailMessage);

    mav.setViewName("findForm");
}
```

root-context.xml

```
<!-- 메일보내기 -->
<bean id="javaMailSenderImpl" class="org.springframework.mail.javamail.JavaMailSenderImpl">
    <property name="host" value="smtp.gmail.com" />
    <property name="port" value="587" />
    <property name="username" value="emailtest20170314@gmail.com" />
    <property name="password" value="test20170314" />

    <property name="javaMailProperties">
        <props>
            <prop key="mail.transport.protocol">smtp</prop>
            <prop key="mail.smtp.auth">true</prop>
            <prop key="mail.smtp.starttls.enable">true</prop>
            <prop key="mail.debug">true</prop>
        </props>
    </property>
</bean>
```

1. ModelAndView를 생성 한다.
2. 변수 name과 email에 parameter값을 저장 한다.
3. 두 변수를 Member 빈에 담는다.
4. 변수 id에 값이 담긴 빈을 사용하여 mDao.getId(member) 실행한 값을 저장한다.
5. 만약에 정상적으로 DAO가 수행이 되면 돌아온 id값으로 이메일을 보낸다.
6. TargetView findForm.jsp를 ModelAndView에 setViewName 한다.

■ DAO & MAPPER PARAMETER : member

```
public String getId(Member member) {
    return sqlSession.selectOne("member.getId", member);
}

<select id="getId" parameterType = "member" resultType="String">
    SELECT M_ID FROM MEMBER WHERE M_NAME =#{m_name} AND M_EMAIL = #{m_email}
</select>
```

1. Member 빈에 담긴 값으로 mapper id : getId 실행 SELECT 질의 한다.
2. 질의 결과를 String에 담아서 넘긴다.

■ TARGET VIEW MODEL : findForm.jsp / 이메일

STEPPE

[STEPPE?](#)
[공지사항](#)
[프로젝트](#)
[프리랜서](#)
[이용방법](#)
[로그인](#)
[회원가입](#)

아이디 찾기

아이디 찾기

비밀번호 찾기

인증번호 발송

비밀번호 찾기

© Day Theme. All Rights Reserved.
Free Bootstrap Themes by BootstrapMade
[공지사항](#) [faq](#) [이용약관](#)

내 정보 찾기 페이지



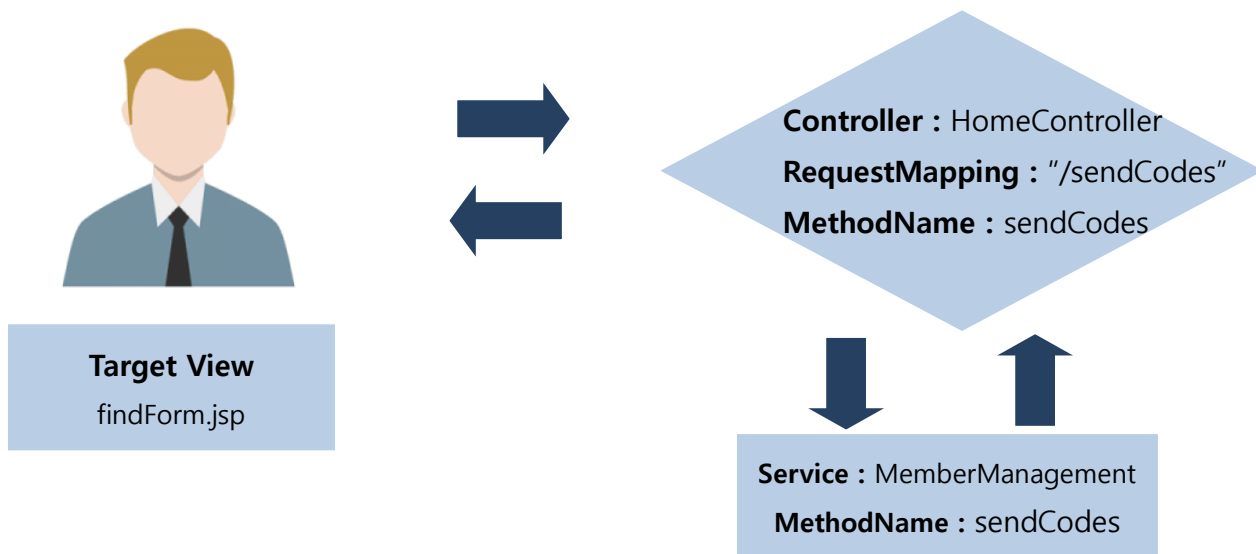
emailtest20170314@gmail.com

나에게 ▾

귀하의 아이디는 c입니다.

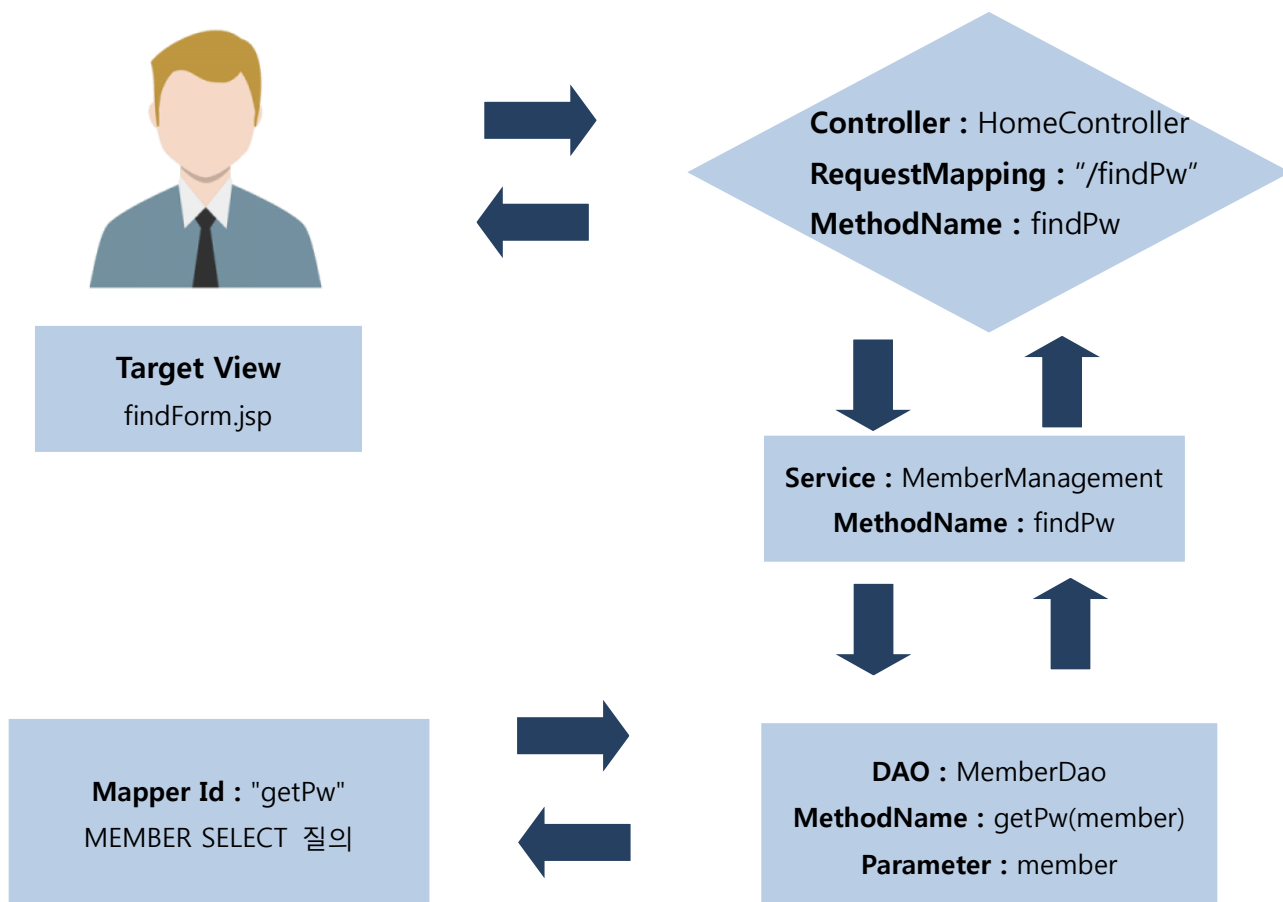
회원의 아이디를 안내하는 이메일

JOB CODE	D02	JOB NAME	비밀번호 찾기	CLASS	MemberManagement
PROCESS FLOW					



■ 개요

1. 회원이 입력한 이메일로 인증번호(랜덤 4자리 숫자) 전송.



■ 개요

1. 회원의 아이디와 이메일 확인하여 회원의 비밀번호를 이메일로 전송.
2. MEMBER SELECT

■ CONTROLLER

REQUEST : /sendCodes , /findPw

```
//이메일 인증 내정보 찾기에서
@RequestMapping(value = "/sendCodes")
public ModelAndView sendCodes() throws Exception {
    System.out.println("ModelAndView sendCodes");
    mav = new ModelAndView();
    mm.sendCodes();
    return mav;
}
```

1. 인증번호를 발송하는 sendCodes() 함수를 실행한다.

```
//비밀번호 찾기
@RequestMapping(value = "/findPw", method = RequestMethod.POST)
public ModelAndView findPw(String m_id, String m_email) {
    System.out.println("비밀번호 찾기");
    mav = mm.executes(m_id,m_email,1);
    return mav;
}
```

1. 회원이 입력한 아이디(m_id)와 이메일(m_email)을 모델에 담아 excutes(m_id, m_email, 1) 함수를 실행한다.
2. 넘겨 받은 값으로 함수 findPw(m_id,m_email)를 실행 한다.

■ SERVICE

PARAMETER : m_id, m_email

```

<script>
/* 이메일 인증 */
function sendCodes(){
    var flag = 1;
    var email = $('#userEmail').val();

    if(email!=''){
        $.ajax({
            type: 'POST',
            url: './sendCodes',
            data: {flag:flag, userEmail:email},

            success:function(data){
                var msg = data.split(",");
                console.log(data);
                $('#notiEmail').html(msg[0]);
                $('#notiCode').html(msg[1]);
                console.log('flag');
            },

            error:function(error){
                console.log(error);
            }
        });
    }
    else
    {
        $('#notiEmail').html("이메일을 입력해주세요");
    }
}
</script>

```

1. findForm.jsp에서 이메일이 입력 되었을 경우 flag값을 1로 지정한다.
2. Ajax로 지정된 flag값과 회원이 입력한 userEmail 값을 넘긴다.

```

//이메일 인증 내정보 찾기에서
public String sendCodes() throws Exception{

    int flag = Integer.valueOf(request.getParameter("flag"));
    System.out.println("flag: "+flag);
    String email = request.getParameter("userEmail");
    System.out.println("userEmail="+ email);

    if(flag == 1){
        // 인증번호 생성

        Random random = new Random();
        code = random.nextInt(10000) + 1000;

        if(code > 10000)
            code = code - 1000;

        // 메일 보내기
        HttpSession session = request.getSession();
        session.setAttribute("code", code);

        //일반 텍스트메일
        SimpleMailMessage simpleMailMessage = new SimpleMailMessage();
        simpleMailMessage.setFrom("emailtest20170314@gmail.com");
        simpleMailMessage.setTo(email);
        simpleMailMessage.setSubject("Steppe 이메일 인증 코드 입니다.");
        simpleMailMessage.setText("인증 코드는 "+code+"입니다.");

        javaMailSenderImpl.send(simpleMailMessage);
    }
    return null;
}

```

root-context.xml

```

<!-- 메일보내기 -->
<bean id="javaMailSenderImpl" class="org.springframework.mail.javamail.JavaMailSenderImpl">
    <property name="host" value="smtp.gmail.com" />
    <property name="port" value="587" />
    <property name="username" value="emailtest20170314@gmail.com" />
    <property name="password" value="test20170314" />

    <property name="javaMailProperties">
        <props>
            <prop key="mail.transport.protocol">smtp</prop>
            <prop key="mail.smtp.auth">true</prop>
            <prop key="mail.smtp.starttls.enable">true</prop>
            <prop key="mail.debug">true</prop>
        </props>
    </property>
</bean>

```

1. ajax로 넘겨 받은 flag 값과 userEmail값을 변수 flag와 email에 저장한다.
2. if구문에서 flag의 값이 1일 때 인증번호를 생성한다.
3. Random함수를 이용하여 4자리 숫자의 인증번호를 생성하여 변수 code에 저장한다.
4. 이때, code를 session Attribute에 저장하여 회원이 이메일로 전송 받은 code와 세션에 저장된 code를 비교 할 수 있게 한다.

```

private void findPw(String m_id, String m_email) {
    mav = new ModelAndView();

    String id = request.getParameter("m_id");
    String email = request.getParameter("m_email");

    Member member = new Member();
    member.setM_id(id);
    member.setM_email(email);

    // 메일 보내기
    String password = mDao.getPwds(member);

    //일반 텍스트메일
    SimpleMailMessage simpleMailMessage = new SimpleMailMessage();
    simpleMailMessage.setFrom("emailtest20170314@gmail.com");
    simpleMailMessage.setTo(email);
    simpleMailMessage.setSubject("Steppe 비밀번호 찾기 입니다.");
    simpleMailMessage.setText("현재 비밀번호는 "+password+"입니다.");

    javaMailSenderImpl.send(simpleMailMessage);

    mav.setViewName("findForm");
}

```

1. ModelAndView를 생성 한다.
2. 회원이 입력한 아이디(m_id)와 이메일(m_email)을 변수 id와 email에 저장한다.
3. 저장한 변수를 Member 빈에 담는다.
4. password 변수에 빈을 이용하여 mDao.getPwds(member)를 실행한 값을 저장한다..
5. password 값을 이메일로 전송한다.

■ DAO & MAPPER PARAMETER : member

```
public String getPwds(Member member) {
    return sqlSession.selectOne("member.getPwds", member);
}

<select id="getPwds" parameterType = "member" resultType="String">
    SELECT M_PW FROM MEMBER WHERE M_ID=#{m_id} AND M_EMAIL = #{m_email}
</select>
```

1. Member 빈에 담긴 값으로 mapper id : getPwds 실행 SELECT 질의 한다.
2. 질의 결과를 String에 담아서 넘긴다.

■ TARGET VIEW MODEL : findForm.jsp / 이메일

STEPPE

[STEPPE?](#)
[공지사항](#)
[프로젝트](#)
[프리랜서](#)
[이용방법](#)
[로그인](#)
[회원가입](#)

아이디 찾기

아이디 찾기

비밀번호 찾기

인증번호 발송


비밀번호 찾기

© Day Theme. All Rights Reserved.

Free Bootstrap Themes by BootstrapMade

[공지사항](#)
[faq](#)
[이용약관](#)

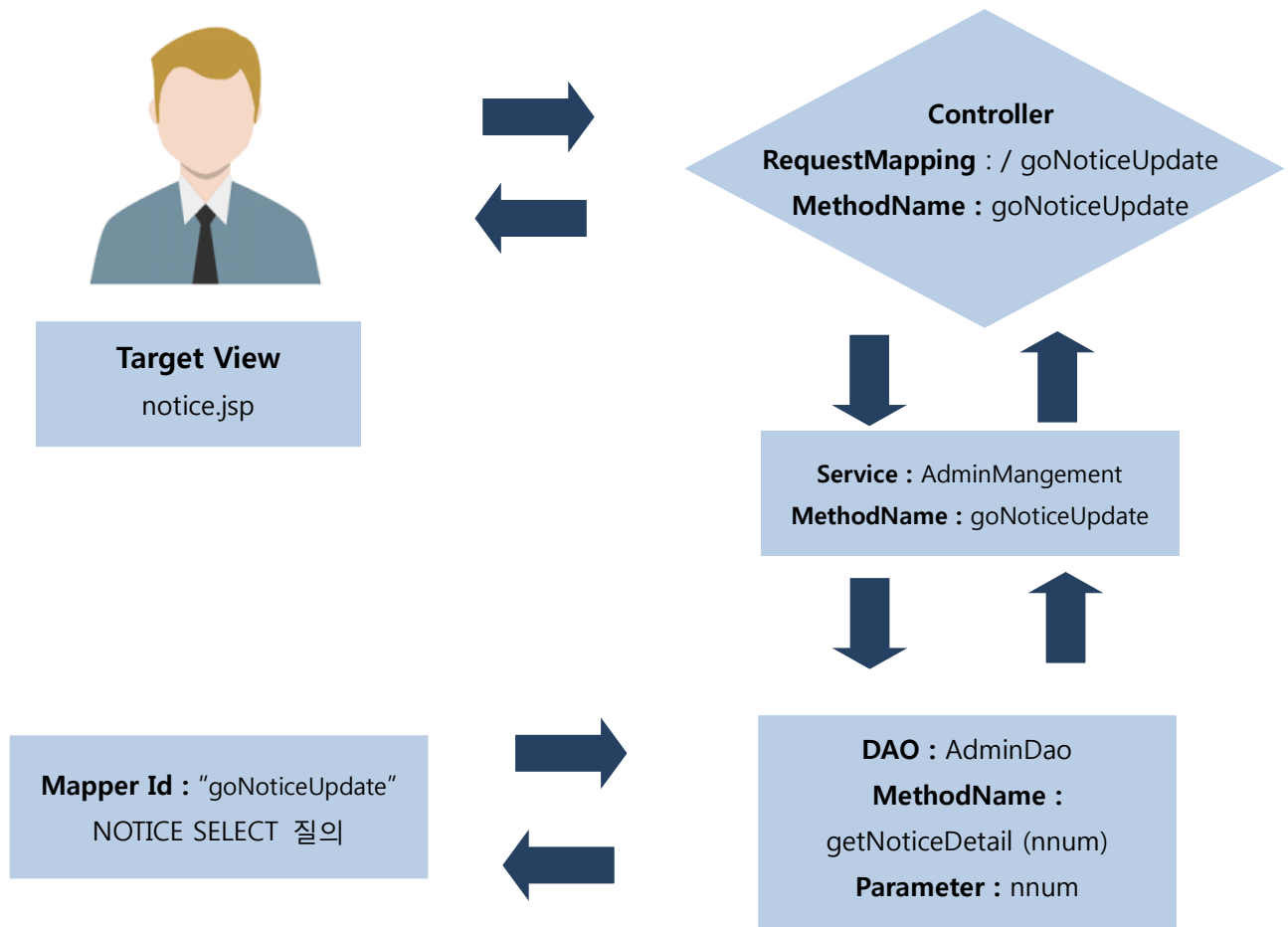
내 정보 찾기 페이지



emailtest20170314@gmail.com
나에게 ▼
현재 비밀번호는 c입니다.

현재 비밀번호를 안내하는 이메일

JOB CODE	D03	JOB NAME	공지사항 수정 이동	CLASS	AdminManagement
PROCESS FLOW					



■ 개요

1. 공지사항 상세 페이지로 이동한다
2. 상세페이지에서 출력되는 값을 빈에 저장하여 넘긴다

■ CONTROLLER

REQUEST : /noticeUpdate

```
//공지사항 수정 페이지로 이동
@RequestMapping(value = "/goNoticeUpdate")
public ModelAndView goNoticeUpdate(int nnum) {
    mav = am.execute(nnum, 2);
    return mav;
}
```

1. 공지사항 리스트 페이지에 출력된 글 번호(n_num)를 넘겨준다.
2. ServiceClass인 AdminManagement 클래스의 execute(nnum, 2) 함수를 실행 한다.
3. ServiceClass에서 goNoticeUpdate(nnum) 함수를 실행한다.
4. ModelAndView로 return

■ SERVICE

PARAMETER : int nnum

//공지사항 수정 페이지로 이동

```
private String goNoticeUpdate(int nnum) {
    mav = new ModelAndView();
    List<Notice> ndlist = null;

    nnum = Integer.parseInt(request.getParameter("nnum"));
    ndlist = aDao.getNoticeDetail(nnum);
    nt = ndlist.get(0);

    int nnums = nt.getN_num();
    System.out.println("nnums="+nnums);
    String ntitle = nt.getN_title();
    System.out.println("ntitle = "+ntitle);
    String ncontent = nt.getN_content();
    System.out.println("ncontent = "+ncontent);

    mav.addObject("ndlist",nt);

    mav.setViewName("noticeUpdate");
    return null;
}
```

1. ModelAndView 생성 한다.
2. SELECT 질의를 통해 받아올 값들을 리스트에 저장하기 위해 ndlist를 선언 한다.
3. 변수 nnum에 parameter값을 저장 한다.
4. nnum 이용하여 AdminDao의 getNoticeDetail(nnum)을 실행한 후 넘겨 받은 값을 ndlist에 담는다.
5. Notice 빈 nt에 리스트값을 담아준후 리스트에 넣어준다.
6. TargetView noticeUpdate.jsp를 ModelAndView에 setViewName 한다.

■ DAO & MAPPER

PARAMETER : Notice

```
public List<Notice> getNoticeDetail(int nnum) {
    return sqlSession.selectList("notice.getNoticeDetail",nnum);
}
```

```
<select id="getNoticeDetail" parameterType="Integer" resultType="notice">
    SELECT * FROM NOTICE WHERE N_NUM = #{n_num}
</select>
```

1. nnum 값으로 mapper id: getNoticeDetail 실행 SELECT 질의 한다.

공지사항

글번호	제목	작성자	작성날짜	수정	삭제
1	공지사항 입니다.	관리자	2017-08-31 13:21:14	수정	삭제

[1]

글쓰기

글번호 1

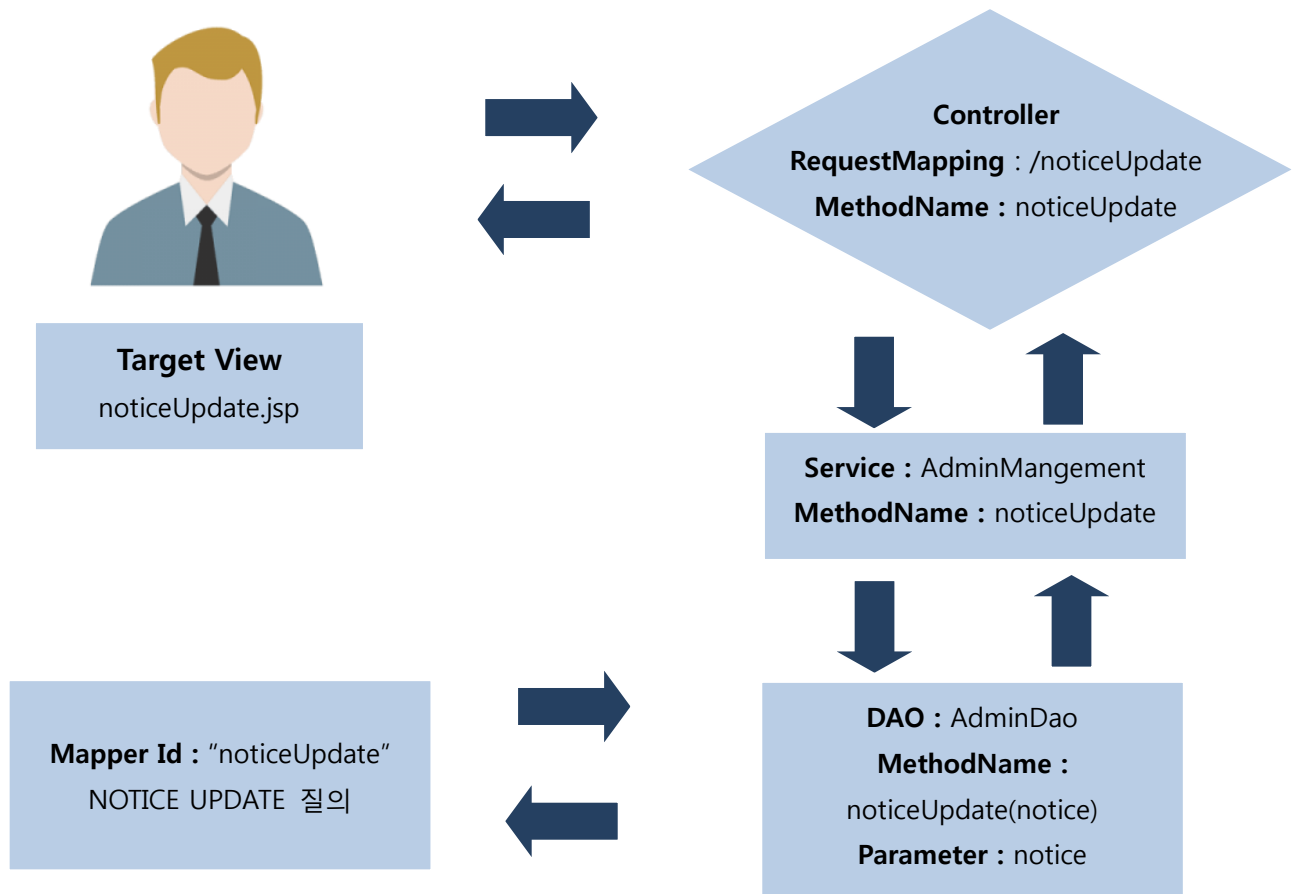
제목 공지사항 입니다.

공지사항 내용 입니다.

수정

공지사항에 원래 입력되어있는 내용을 getNoticeDetail을 통해 공지사항 수정 페이지에 출력해준다.

JOB CODE	D04	JOB NAME	공지사항 수정	CLASS	AdminManagement
PROCESS FLOW					



■ 개요

1. 공지사항 상세 페이지의 내용을 수정 한다.
2. 회원이 입력한 값을 빈에 저장하여 넘긴 후 수정

■ CONTROLLER

REQUEST : /noticeUpdate

```

//공지사항 수정
@RequestMapping(value = "/noticeUpdate")
public ModelAndView noticeUpdate(Notice notice) {
    mav = am.execute(notice, 2);
    return mav;
}

```

1. 공지사항 상세 페이지에 출력된 글 번호(n_num), 글 제목(n_title), 글 내용(n_content) 와 회원이 새로 입력한 값들을 Notice 빈에 담아 넘겨준다.
2. ServiceClass인 AdminManagement 클래스의 execute(notice, 2) 함수를 실행
3. ModelAndView로 return

■ SERVICE

PARAMETER : Notice

```
//공지사항 수정
private String noticeUpdate(Notice notice) {
    System.out.println("공지사항 수정");
    mav = new ModelAndView();

    int nnum = nt.getN_num();
    System.out.println("nnum="+nnum);
    String ntitle = notice.getN_title();
    System.out.println("ntitle="+ntitle);
    String ncontent = notice.getN_content();
    System.out.println("ncontent="+ncontent);

    notice.setN_num(nnum);
    notice.setN_title(ntitle);
    notice.setN_content(ncontent);

    aDao.noticeUpdate(notice);

    mav.setViewName("notice");
    getNoticeList();
    return null;
}
```

1. ModelAndView 생성 한다.
2. 변수 nnum, ntitle, ncontent에 parameter값을 저장 한다.
3. 변수들을 Notice 빈에 담는다.
4. 빈에 담긴 값을 이용하여 공지사항을 수정한다.
5. ModelAndView로 setViewName(notice)을 한 후에 다시 공지사항 리스트 출력 메서드를 실행한다.

■ DAO & MAPPER

PARAMETER : Notice

```
public int noticeUpdate(Notice notice) {
    return sqlSession.update("notice.noticeUpdate",notice);
}
```

```
<update id="noticeUpdate" parameterType="notice">
    UPDATE NOTICE SET n_title = #{n_title}, n_content = #{n_content} WHERE n_num = #{n_num}
</update>
```

1. Notice 빈에 담긴 값으로 mapper id: noticeUpdate 실행 UPDATE 질의 한다.

공지사항

글번호	제목	작성자	작성날짜	수정	삭제
1	공지사항 입니다.	관리자	2017-08-31 13:21:14	수정	삭제

[1]

글쓰기

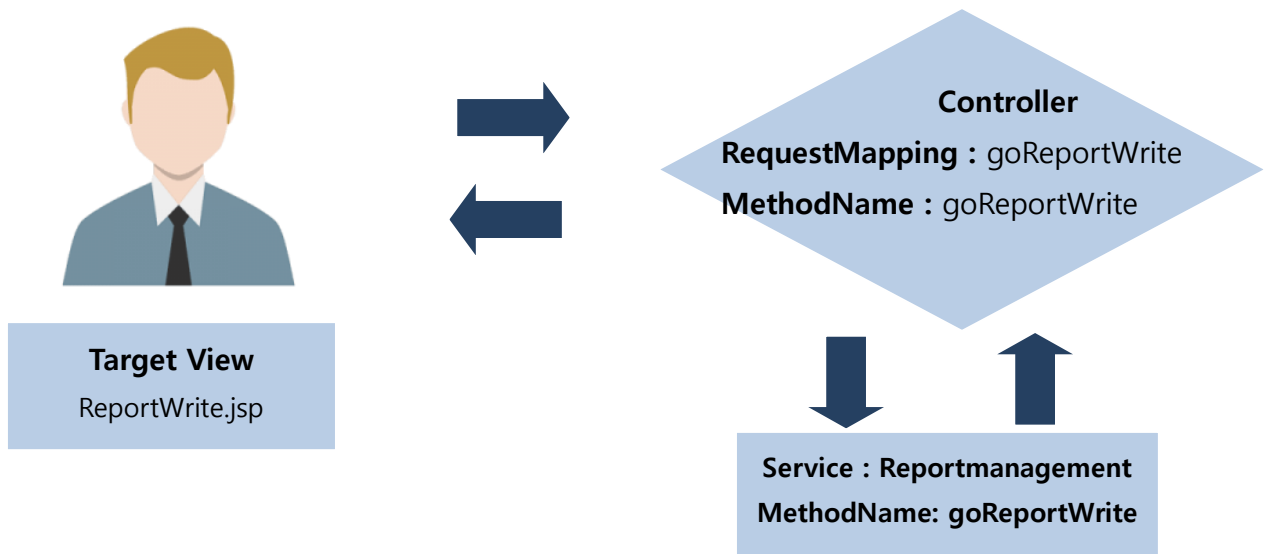
글번호 1

제목 공지사항 입니다.

공지사항 내용 입니다.

수정

JOB CODE	E01	JOB NAME	신고페이지 이동	CLASS	
PROCESS FLOW					



■ 개요

1. 프리랜서 상세보기 내 페이지에 존재하는 신고하기 버튼 클릭
2. 신고자의 아이디와 신고대상이 되는 URL 값을 저장하여 신고페이지로 이동

■ CONTROLLER

REQUEST : goReportWrite

```
//신고작성 페이지로 이동
@RequestMapping(value = "/goReportWrite")
public ModelAndView goReportWrite(){
    mav=rm.goReportWrite();
    return mav;
}
```

1. 프리랜서 상세보기 페이지 내부의 신고하기 버튼 클릭
2. 서비스 Reportmanagement의 goReportWrite 메소드 실행
3. 신고 항목의 URL과 신고자의 ID값을 View페이지에 출력


```

//신고 작성 페이지로 이동
public ModelAndView goReportWrite() {
    mav = new ModelAndView();
    String view=null;
    //세션에 저장된 아이디 값 호출
    String m_id=(String) session.getAttribute("m_id");
    //세션에 저장된 아이디의 값이 없을때 로그인 페이지로 이동
    if(m_id==null){
        mav.addObject("msg","로그인이 필요한 서비스입니다.");
        view="login";
        mav.setViewName(view);
    }
    //세션에 저장된 아이디의 값이 존재할때
    if(m_id!=null){
        //작성자의 아이디 값을 호출
        String user=request.getParameter("m_id");
        //레퍼러 값을 호출
        String returnUrl=request.getHeader("REFERER");
        //페이지단에 작성자 아이디 값을 object로 담아 view단에 호출
        mav.addObject("user2",user);
        //페이지단에 레퍼러 값을 object로 담아 view단에 호출
        mav.addObject("reportUrl",returnUrl);
        //view페이지를 신고작성 페이지로 정의
        view="reportWrite";
        mav.setViewName(view);
    }
    return mav;
}

```

1. ModelAndView 객체생성
2. 현재 세션에 저장된 ID의 값을 호출하여 저장 저장된 값이 존재하지 않을 때 로그인 페이지로 이동
3. 세션에 저장된 값이 존재할 때 세션에 담긴ID와 REFERER를 이용하여 신고대상 페이지의 URL값을 가져와 변수에 저장하여 오브젝트에 담아 View페이지로 전달

STEPPE

STEPPE?

공지사항

프로젝트

프리랜서

이용방법

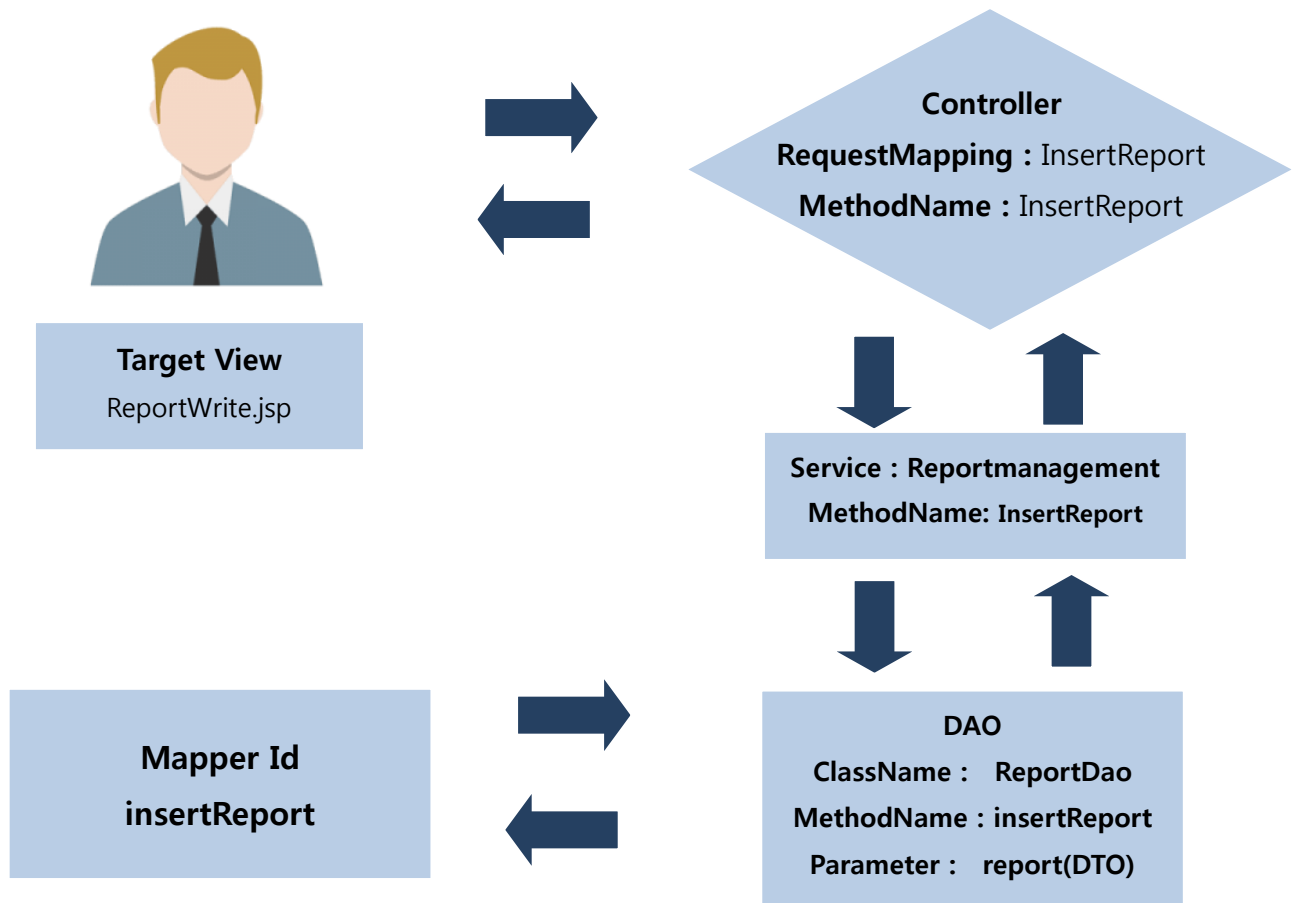
프리랜서님 아서오세요.

신고자	lsw
신고 주소	http://localhost:85/nomad/goFreelancerDetail?m_id=free
신고사유	신고 사유를 선택해 주십시오. ▼
신고 내용	

작성
돌아가기

1. 신고자 사항에 세션에 현재 저장된 ID의 값 표출
2. 신고 주소에 REFERER를 이용한 신고대상의 URL값 표출

JOB CODE	E02	JOB NAME	신고 작성	CLASS	ReportManagement
PROCESS FLOW					



■ 개요

1. 신고 사유와 신고 내용을 받아 작성

■ CONTROLLER

REQUEST :InsertReport

```
//신고 작성
@RequestMapping(value="/InsertReport")
public ModelAndView InsertReport(Report report){
    mav=rm.InsertReport(report);
    return mav;
}
```

1. 신고(Report) 정보를 파라미터로 받음
2. ReportManagement의 InsertReport 실행

■ Bean(Report) Data Transfer Object

```
public class Report {
    private String r_mid;
    private int r_num;
    private String r_kind;
    private String r_content;
    private String r_url;
}
```

■ SERVICE PARAMETER : Report

```
//신고작성
public ModelAndView InsertReport(Report report) {
    //ModelAndView 객체 생성
    mav = new ModelAndView();
    String view=null;
    //세션에 저장된 ID값을 가져옴
    String m_id=(String) session.getAttribute("m_id");
    //ID값을 Set
    report.setR_mid(m_id);
    //Referer를 이용하여 view에 담아준 URL을 가져옴
    String r_url=request.getParameter("r_url");
    //URL Set
    report.setR_url(r_url);
    //신고사유 값을 가져옴
    String r_kind=request.getParameter("r_kind");
    //신고사유 값 Set
    report.setR_kind(r_kind);
    //신고내용 값 가져옴
    String r_content=request.getParameter("r_contents");
    //신고내용 값 Set
    report.setR_content(r_content);
    System.out.println(m_id);
    //신고 번호 값 데이터베이스에서 계산후 최대값에 +1을 더하여 변수에 담아줌(Sequence 대체)
    int r_num = rDao.getMaxNum()+1;
    //번호 값 Set
    report.setR_num(r_num);
    //작성된 값을 parameter로 사용하며 DAO에서 활용
    rDao.insertReport(report);
    //작성완료시 프리랜서목록 페이지로 이동
    view="redirect:/goFreelancer";
    mav.setViewName(view);
    return mav;
}
```

1. ModelAndView 객체 생성
2. Report Bean(DTO : Data Transfer Object)에 Parameter저장
3. 저장한 Parameter를 이용하여 Dao(Data Access Object) ReportDao 의 insertReport를실행한다.

■ DAO & MAPPER PARAMETER : Report

```
public int insertReport(Report report) {

    return sqlSession.insert("report.insertReport",report);
}

public int getMaxNum() {
    return sqlSession.selectOne("report.selectMaxNum");
}

<insert id="insertReport" parameterType="report">
    INSERT INTO REPORT VALUES({r_num},{r_mid},{r_kind},{r_content},{r_url})
</insert>
<select id="selectMaxNum" resultType="int">
    select NVL(MAX(r_num),0) FROM report
</select>
```

1. DAO(Data Access Object)
 - 1) Report_mapper에 있는 insertReport 에 Parameter Report(Data Transfer Object)를 보냄
 - 2) Report_mapper에 있는 SelectMaxNum으로 보내어 DataBase에 존재하는 신고 글의 개수 파악
2. Mapper
 - 1) insertReport : 넘어온 Parameter 값을 사용하여 신고 글의 정보를 Insert(삽입) 한다.
 - 2) selectMaxNum : 신고 테이블 내의 Column의 마지막 수를 표출한다.

■ TARGET VIEW MODEL :reportWrite.jsp

STEPPE

[STEPPE?](#)
[공지사항](#)
[프로젝트](#)
[프리랜서](#)
[이용방법](#)
[프리랜서님 어서오세요.](#)

신고자	lsw
신고 주소	http://localhost:85/nomad/goFreelancerDetail?m_id=free
신고사유	<input type="text" value="신고 사유를 선택해 주십시오."/>
신고 내용	<div></div>

© Day Theme. All Rights Reserved.
Free Bootstrap Themes by BootstrapMade

[공지사항](#)
[faq](#)
[이용약관](#)

신고 작성 페이지

STEPPE

localhost:85 내용:
입력 사항을 작성하지 않았습니다.
확인

프리랜서
이용방법
프리랜서님 어서오세요.

신고자lsw
신고 주소http://localhost:85/nomad/goFreelancerDetail?m_id=lsw
신고사유신고 사유를 선택해 주십시오.

신고 내용

작성
돌아가기

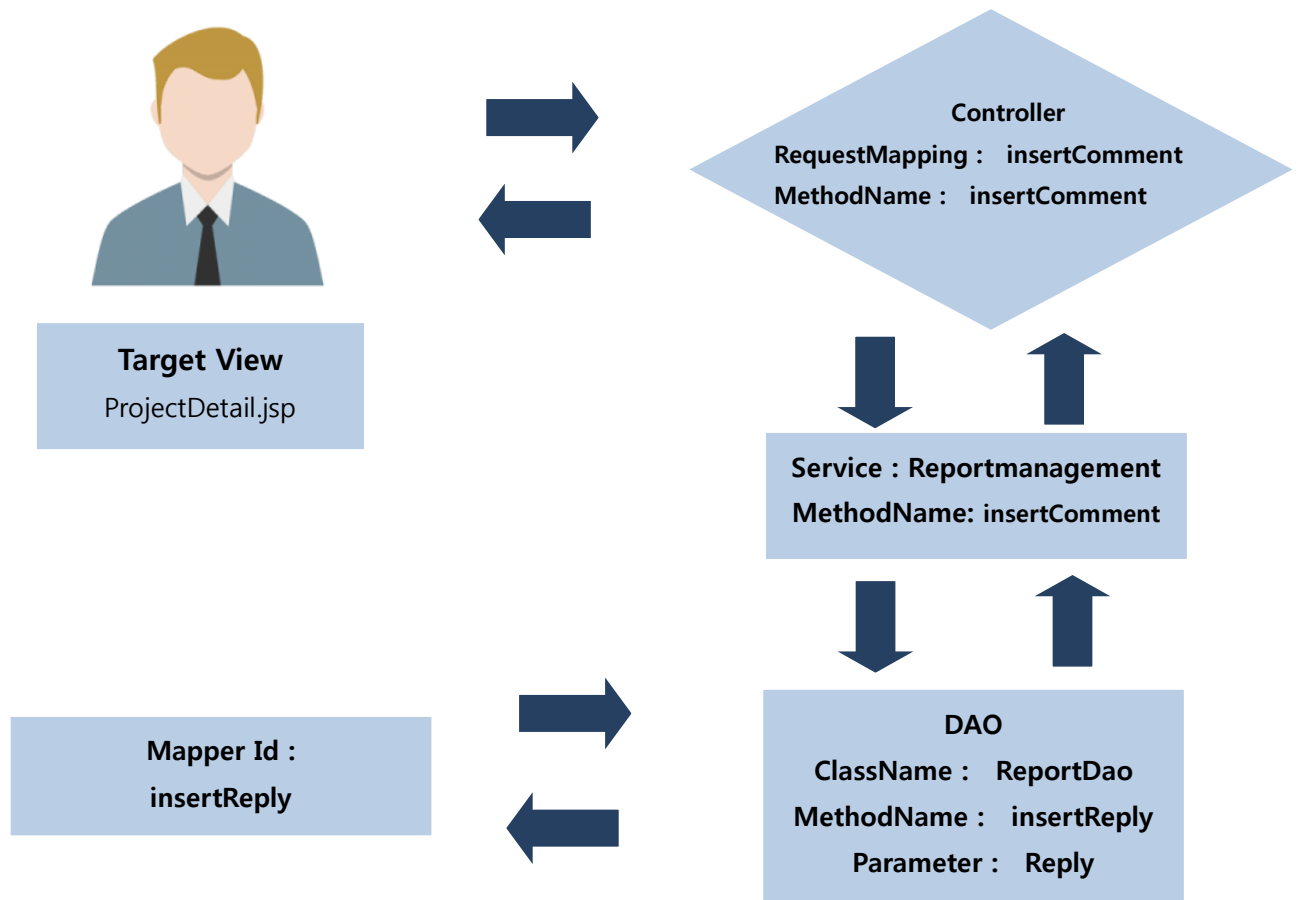
© Day Theme. All Rights Reserved.
Free Bootstrap Themes by BootstrapMade
공지사항 faq 이용약관

작성 사항을 입력하지 않았을 시(작성 실패)

글 번호	신고자	신고종류	신고내용	신고대상 주소	삭제
1	siin1992	기타	너무 멧쟁이라	페이지 이동	삭제
2	lsw	기타	신고합니다.	페이지 이동	삭제

관리자 계정으로 로그인 했을 때 신고 관리 페이지에서 신고 테이블에 존재하는 Data를 추출하여 볼 수 있다. (3번 view의 경우 report.jsp에 해당한다.)

JOB CODE	E03	JOB NAME	댓글 작성	CLASS	ReportManagement
PROCESS FLOW					



■ 개요

1. 현재 세션 내 저장된 ID 정보와 댓글의 정보를 받아 댓글 작성

■ CONTROLLER

REQUEST :insertComment

```
//프로젝트 댓글 삽입
@RequestMapping(value="/insertComment")
public ModelAndView insertComment(Reply reply){
    mav=rm.insertComment(reply);
    return mav;
}
```

1. 댓글 정보 Reply(DTO:Data Transfer Object)를 Parameter로 받는다.
2. Service 클래스 ReportMangement의 insertComment를 실행한다.

■ Bean(Report)	Data Transfer Object
<pre> public class Report { private String r_mid; private int r_num; private String r_kind; private String r_content; private String r_url; </pre>	
■ SERVICE	PARAMETER : Reply

//프로젝트 댓글삽입

```

public ModelAndView insertComment(Reply reply) {
    //ModelAndView 객체 생성
    mav=new ModelAndView();
    //세션에 저장된 아이디의 정보를 가져온다.
    String r_mid=(String) session.getAttribute("m_id");
    //세션에서 불러온 아이디 값을 Set
    reply.setR_mid(r_mid);
    //프로젝트 번호의 값을 가져온다.
    int r_pnum=Integer.parseInt(request.getParameter("p_num"));
    //System.out.println("프로젝트 번호:"+r_pnum);
    //가져온 프로젝트 번호의 값을 set
    reply.setR_pnum(r_pnum);
    //댓글의 내용값을 가져온다.
    String r_content=request.getParameter("r_content");
    //System.out.println("댓글내용:"+r_content);
    //댓글의 내용 Set
    reply.setR_content(r_content);
    //댓글 테이블 내 댓글의 갯수를 세어 최대값에서 1을 더한다.
    int r_num=rDao.getReplyMaxNum()+1;
    System.out.println("댓글번호:"+r_num);
    //댓글 번호를 Set
    reply.setR_num(r_num);
    //set한 Parameter를 DAO(Data Access Object)로 보내어 실행
    rDao.insertReply(reply);
    //댓글이 작성되는 원글의 페이지로 이동
    mav.setViewName("redirect:goProjectDetail?p_num="+r_pnum);
    return mav;
}

```

1. ModelAndView 객체 생성
2. 현재 세션에 저장된 ID와 댓글 테이블에 저장될 데이터의 정보를 Reply DTO(Data Transfer Object)에 담아 Parameter로 활용해 DAO(Data Access Object)인 ReportDao에서 활용하여 실행한다.

■ DAO & MAPPER PARAMETER : Report

//테이블내 댓글 최대수 계산

```
public int getReplyMaxNum() {
    return sqlSession.selectOne("report.selectReplyMaxNum");
}
```

//댓글삽입

```
public int insertReply(Reply reply) {
    return sqlSession.insert("report.insertReply", reply);
}
```

//댓글 표시

```
public List<Reply> showReply(int p_num) {

    return sqlSession.selectList("report.showReply", p_num);

}
```

<!--댓글 테이블 내 컬럼 최대수 계산-->

```
<select id="selectReplyMaxNum" resultType="int">
    select NVL(MAX(r_num),0) FROM REPLY
</select>
```

<!--댓글 삽입-->

```
<insert id="insertReply" parameterType="reply">
    INSERT INTO REPLY VALUES("#{r_num}","#{r_mid}","#{r_pnum}","#{r_content}",default)
</insert>
```

<!--댓글 표시-->

```
<select id="showReply" parameterType="Integer" resultType="reply">
    SELECT * FROM REPLY WHERE R_PNUM=#{r_pnum} ORDER BY R_NUM ASC
</select>
```

1. DAO(Data Access Object)

- 1) Report_mapper에 있는 insertReply 에 Parameter Reply(Data Transfer Object, bean)를 보냄
- 2) Report_mapper에 있는 selectReplyMaxNum으로 DataBase에 존재하는 댓글의 개수 파악
- 3) Report_mapper에 있는 showReply에 원 글의 번호 값을 담은 parameter를 보냄

2. Mapper

- 1) insertReply : 넘어온 Parameter 값을 사용하여 댓글의 정보를 Insert(삽입) 한다.
- 2) selectReplyMaxNum: 댓글 테이블 내의 Column의 마지막 수를 표시한다.
- 3) showReply : 댓글 테이블에 있는 Data 중 원 글의 번호와 일치하는 댓글 리스트를 표시토
록 한다.

제목		간단한 교통앱을 만들고 싶습니다.	
작성자		client	
프로젝트 기간	지원자 (필요인원)	예산 금액	마감일
30일	0명 (1명)	400만원	2017-09-16
필요언어	JAVA SQL	입찰가 (만원단위 ex)200	<div>결정</div> <div>목록보기</div>

입찰 마감 시간이 9일 6시간 23분 6초 남았습니다.

프로젝트 설명

성실하고 약속을 잘 지키시는 자바 개발자 한 분을 모집합니다.

댓글

작성자	작성내용	작성날짜
<div></div>		
<div>댓글작성</div>		

댓글 삽입 전 view 페이지

제목		간단한 교통업을 만들고 싶습니다.	
작성자		client	
프로젝트 기간	지원자 (필요인원)	예산 금액	마감일
30일	0명 (1명)	400만원	2017-09-16
필요언어	JAVA SQL	입찰가 (만원단위 ex)200	<div>결정</div> <div>목록보기</div>

입찰 마감 시간이 9일 6시간 23분 49초 남았습니다.

프로젝트 설명

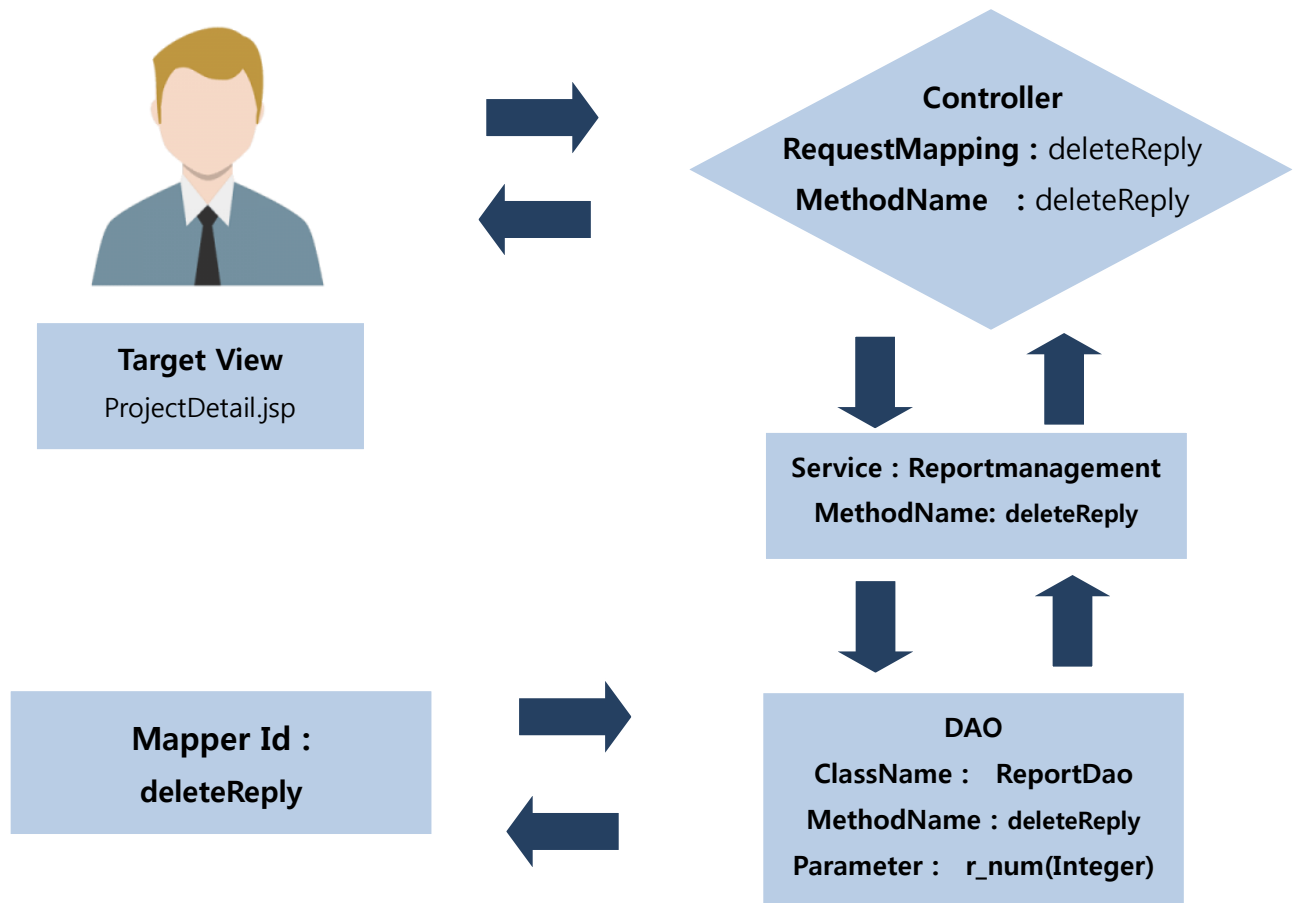
성실하고 약속을 잘 지키시는 자바 개발자 한 분을 모집합니다.

댓글

작성자	작성내용	작성날짜
lsw	댓글 테스트	17/09/07 <div>삭제</div>
lsw	댓글 테스트 2	17/09/07 <div>삭제</div>
<div></div>		
<div>댓글작성</div>		

댓글 삽입 후 view페이지

JOB CODE	E04	JOB NAME	댓글 삭제	CLASS	ReportManagement
PROCESS FLOW					



■ 개요

1. 현재 세션 내 저장된 ID 정보와 댓글의 ID 정보의 값을 비교 후 일치할 경우 삭제

■ CONTROLLER

REQUEST : deleteReply

```
//프로젝트에 포함된 댓글 삭제
@RequestMapping(value="/deleteReply")
public ModelAndView deleteReply(){
    mav=rm.deleteReply();
    return mav;
}
```

1. Service 클래스인 ReportManagement에 존재하는 deleteReply Method실행

■ SERVICE

PARAMETER : r_num(Integer)

```
//댓글 삭제
public ModelAndView deleteReply() {
    //세션에서 현재 접속 ID의 정보를 가져온다.
    String m_id=(String) session.getAttribute("m_id");
    //댓글을 작성한 작성자의 ID의 정보를 가져온다.
    String r_mid=request.getParameter("r_mid");
    System.out.println("현재 접속:"+m_id);
    System.out.println("작성자:"+r_mid);
    //프로젝트의 번호를 가져온다.
    int p_num=Integer.parseInt(request.getParameter("p_num"));
    //정보를 가져오는지 확인을 위해 콘솔창 출력체크
    System.out.println(p_num);
    //댓글의 번호값을 가져온다.
    int r_num=Integer.parseInt(request.getParameter("r_num"));
    //댓글 번호 값을 가져오는지 확인을 위해 콘솔창 출력체크
    System.out.println("리플 번호:"+r_num);
    //현재 세션에 저장된 ID와 삭제에 요청하는 댓글의 작성자 ID를 비교하여 두 변수의 값이 일치할때 댓글삭제 실행
    //관리자의 계정이 로그인 되어있는 경우에도 댓글 삭제가능.
    if(m_id.equals(r_mid)||m_id.equals("admin")){
        rDao.deleteReply(r_num);
        showReplyList();
        System.out.println("삭제성공");
    }
    //만약 일치하지 않는다면 alert창 발생
    else{
        System.out.println("삭제실패");
        mav.addObject("msg","작성자의 ID와 현재 접속한 ID가 다릅니다.");
    }
    mav.setViewName("projectDetail");

    return mav;
}
```

1. 현재 세션에 저장되어 있는 ID의 정보를 불러온 후 삭제 시도하는 댓글의 정보를 불러와 ID의 정보 일치여부 확인
2. 비교한 변수들의 값이 일치할 시 ReportDao(Data Access Object)로 Parameter를 전송한다

■ DAO & MAPPER

PARAMETER : r_num

```
//댓글 삭제
public int deleteReply(int r_num) {

    return sqlSession.delete("report.deleteReply", r_num);
}

<!--댓글 삭제-->
<delete id="deleteReply" parameterType="Integer">
    DELETE FROM REPLY WHERE R_NUM=#{r_num}
</delete>
```

1. DAO(Data Access Object)
 - 1) Service클래스에 받은 Parameter를 Mapper에 존재하는 deleteReply에 보냄.
2. Mapper
 - 1) DAO(Data Access Object)에서 받은 Parameter를 활용하여 삭제 실행

■ TARGET VIEW MODEL : ProjectDetail.jsp

제목	간단한 교통앱 만들고 싶습니다.		
작성자	client		
프로젝트 기간	지원자 (필요인원)	예산 금액	마감일
20일	1명 (1명)	200만원	2017-09-16
필요언어	JAVA 자바스크립트	입찰가 (만원단위 ex)200	<div>결정</div> <div>목록보기</div>

입찰 마감 시간이 8일 9시간 38분 46초 남았습니다.

프로젝트 설명

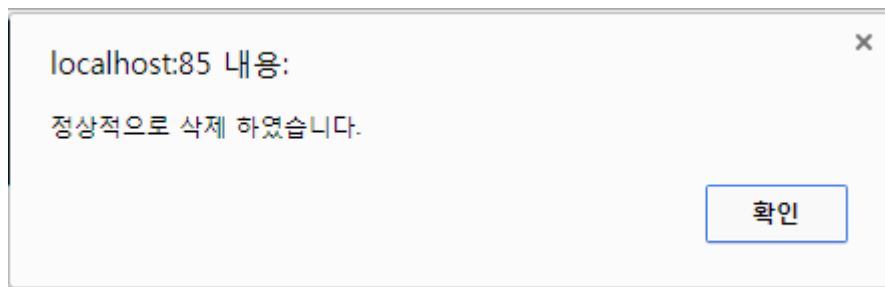
성실하고 약속을 잘 지키시는 자바 개발자 모십니다.

댓글

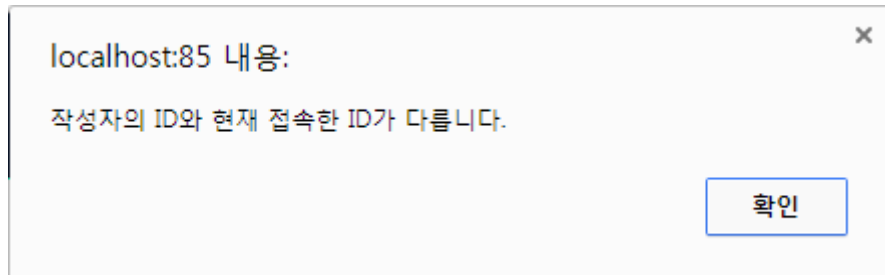
작성자	작성내용	작성날짜	
lsw	댓글 테스트 1번	17/09/08	<div>삭제</div>
lsw	댓글 테스트 2번	17/09/08	<div>삭제</div>

댓글작성

삭제 실행 전 TargetView(ProjectDetail.jsp)

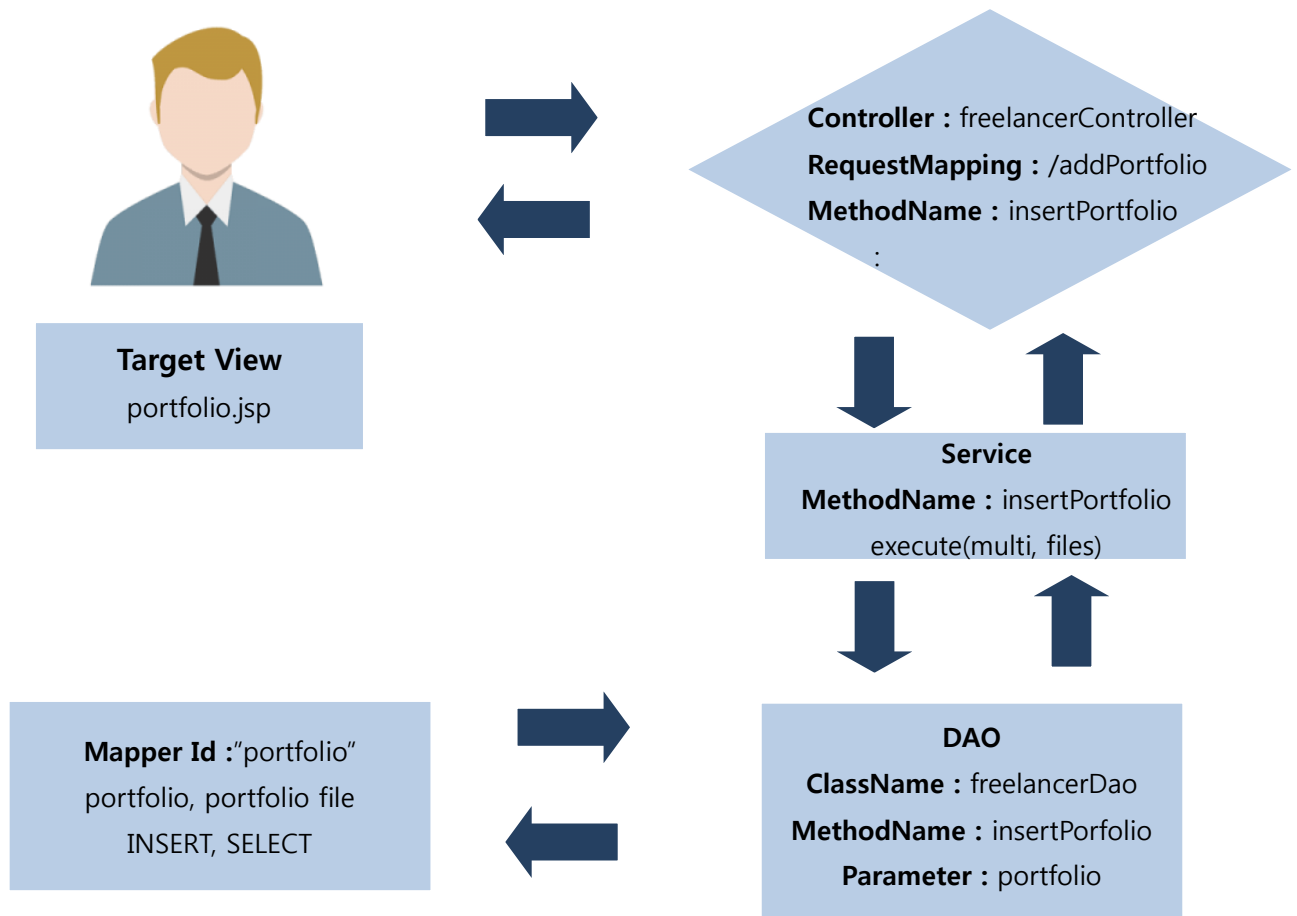


정상적으로 댓글의 삭제가 실행되었을 경우 View에 출력되는 Alert창



비교결과 정보가 일치하지 않았을경우 View에 출력되는 Alert 창

JOB CODE	F01	JOB NAME	포트폴리오 다중업로드	CLASS	freelancerManagement
PROCESS FLOW					



■ 개요

1. 프리랜서의 포트폴리오 등록(다중 업로드 사용)
2. PORTFOLIO , PORTFOLIOFILE INSERT, SELECT
3. targetView : portfolio.jsp

■ CONTROLLER

REQUEST : /addPortfolio

```

@RequestMapping(value = "/addPortfolio")
public ModelAndView portFolioInsert(MultipartHttpServletRequest multi, @RequestParam("pf_image[]") MultipartFile[] files){
    mav = fm.execute(1, multi, files);
    return mav;
}
  
```

1. MultipartHttpServletRequest를 사용하여 파일을 List형태로 받을 수 있다.
2. @RequestParam("pf_image[]")는 jsp에서 pf_image 라는 파라미터를 전달 받을 때 쓰인다.
3. MultipartFile[] files은 스프링에서 업로드 한 파일을 표현할 때 쓰인다.
4. fm(freelancerManagement 추후 fm) 클래스의 execute(1, multi, files) insertPortfolio를 실행한다.
5. ModelAndView 를 return 한다.

■ SERVICE PARAMETER :multi,files

```
private void insertPortfolio(MultipartHttpServletRequest multi, MultipartFile[] files) {
    mav = new ModelAndView();
    String view = null;
    if(ss!=null && ss.getAttribute("m_id")!=null){
        String pf_mid = (String) ss.getAttribute("m_id");
        String pf_title = multi.getParameter("pf_title");
        String pf_term = multi.getParameter("pf_term");
        String pf_contribute = multi.getParameter("pf_contribute");
        String pf_content = multi.getParameter("pf_content");
        UploadFile upload = new UploadFile();
        List<String> mapList = upload.fileUp(multi, files);
        Portfolio portfolio = new Portfolio();
        portfolio.setPf_mid(pf_mid);
        if(fDao.getPortfolioCount()!=0){
            portfolio.setPf_num(fDao.getPortfolioMaxNum()+1);
        }else{
            portfolio.setPf_num(1);
        }
        portfolio.setPf_title(pf_title);
        portfolio.setPf_term(pf_term);
        portfolio.setPf_contribute(pf_contribute);
        portfolio.setPf_content(pf_content);

        if(fDao.insertPortfolio(portfolio)!=0){
            List<Portfolio> pflist = fDao.getPortfolioList(pf_mid);
            Gson jsonObj=new Gson();
            jsonStr = jsonObj.toJson(pflist);
            System.out.println("jsonStr : "+jsonStr);
            mav.addObject("pflist",jsonStr);
            view = "portfolio";
        }
        int pfNum = fDao.getPortfolioMaxNum();
        fDao.portfolioFileInsert(mapList,pfNum);
        System.out.println(mapList);
        mav.setViewName("portfolio");
    }
}
```

1. ModelAndView를 생성한다.
2. 세션값이 null이 아닐 때, 포트폴리오 항목들을 multi.getParameter 로 받아와서 저장한다.
3. uploadfile 메소드를 생성한다.
 - 1) MapList를만들어 multi.getParameter로 받아온 값을 리스트에 저장한다.
 - 2) 포트폴리오 빈을 새로 생성한다.
 - 3) fDao.getPortfolioCount메소드를 실행해서, 카운팅된 숫자가 0이 아니면, 포트폴리오 빈에 fDao.getPortfolioMaxNum()+1(최대값) 함수를 이용하여 기본포트폴리오 번호를 set해준다. 나머지 항목들도 portfolio 빈에 set해준다.
4. fDao.insertPortfolio메소드로 포트폴리오 인서를 한후, 그 값이 0이 아닐 때, 리스트에 담아 Json에 담는다. 추후 Ajax 비동기통신에 쓰임.
5. portfolioFile에도 MaxNum 함수와, 아까 저장한 MapList를 담아 insert한다.
6. setViewName을 portfolio.jsp로 설정한다.

```

public List<String> fileUp(MultipartHttpServletRequest multi, MultipartFile[] files){
    //1.저장경로 찾기
    String root=multi.getSession().getServletContext().getRealPath("/"); //물리적이 주소 찾을/위의 Spring-board까지 찾을
    String path=root+"/resources/upload/";
    //2.폴더 생성을 꼭 할것...
    File dir=new File(path);
    if(!dir.isDirectory()){ //폴더 없다면
        dir.mkdir(); //upload폴더 생성
    }
    MultipartFile mf;
    List<String> listStr = new ArrayList<String>();
    for(MultipartFile file : files){
        byte[] bytes;
        try {
            bytes = file.getBytes();
            String oriFileName=file.getOriginalFilename();
            System.out.println("filename:"+file);
            mf = multi.getFile(oriFileName);
            String sysFileName=System.currentTimeMillis()+"."+
                +oriFileName.substring(oriFileName.lastIndexOf(".")+1);
            System.out.println("sysFileName:"+sysFileName+"path:"+path);
            File serverFile = new File(path+sysFileName);
            BufferedOutputStream stream = new BufferedOutputStream(
                new FileOutputStream(serverFile));
            stream.write(bytes);
            stream.close();
            listStr.add(sysFileName);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    return listStr;
}

```

1. root 라는 물리적인 주소를 설정한다.
2. path 는 따로 /resource/upload/ 라는 업로드폴더를 따로 만들어 지정해준다.
3. file.getBytes를 이용하여 파일의 용량을 구한다.
4. 기본용량은 서블릿에 1메가 정도로 설정했다.
5. oriFileName / sysFileName 은 업로드한 파일의 오리지날 파일명과 / DB에 들어갈 파일명으로 실질적으로 sysFileName이 주로 쓰인다.
6. path+sysFileName으로 view파일인 portfolio.jsp에서 출력한다.

■ DAO & MAPPER PARAMETER :portfolio,Map

```

public int getPortfolioMaxNum() {
    return sqlSession.selectOne("Portfolio.getPortfolioMaxNum");
}

public int insertPortfolio(Portfolio portfolio) {
    return sqlSession.insert("Portfolio.insertPortfolio", portfolio);
}

public int getPortfolioCount() {
    return sqlSession.selectOne("Portfolio.getPortfolioCount");
}

```

```

<select id="getPortfolioMaxNum" resultType="Integer">
    SELECT NVL(MAX(PF_NUM),0) FROM PORTFOLIO
</select>

<select id="getPortfolioCount" resultType="Integer">
    SELECT COUNT(*) FROM PORTFOLIO
</select>

<insert id="insertPortfolio" parameterType="portfolio">
    INSERT INTO PORTFOLIO VALUES(#{pf_num},#{pf_mid},#{pf_title},#{pf_term},#{pf_contribute},#{pf_content})
</insert>

<insert id="portfolioFileInsert" parameterType="Map">
    INSERT INTO PORTFOLIOFILE VALUES(portfolioFile_SEQ.nextval, #{pfNum}, #{sysFileName})
</insert>

```

1. PortfolioMaxNum : Pf_num의 최대값을 선택한다.
2. insertPortfolio : 포트폴리오를 insert한다. (parameterType: portfolio)
3. PortfolioCount : 포트폴리오 개수를 카운팅한다.
4. PortfolioFileInsert: 포트폴리오파일을 insert한다. (parameterType: Map)

■ TARGET VIEW MODEL :portfolio.jsp

```

$("#complete").click(function() {
    var formdata = new FormData(document.getElementById('portfolio'));
    var pflist='';
    $('#portfolio').ajaxForm({
        type : 'post',
        url : 'addPortfolio',
        enctype: "multipart/form-data",
        dataType : 'json',
        success : function(data) { //json형태 반환
            console.log(data); //json 구조파악
            for(var i=0;i<data.length;i++){
                pflist+="|
|  |

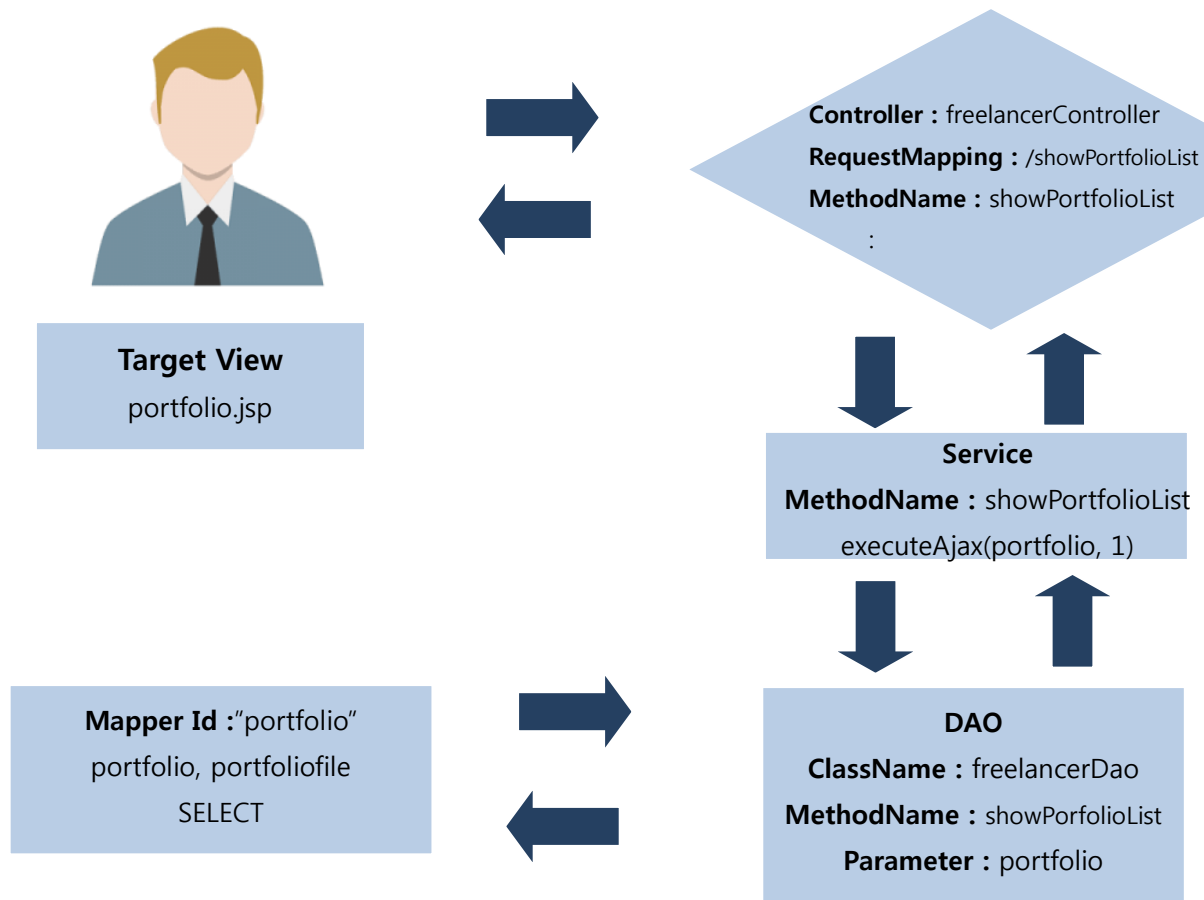
```

1. Button에 id를 complete로 줘서 ajaxForm 사용
2. Json 형태로 반환

제목	<input type="text" value="포트폴리오 입니다."/>
기간	<input type="text" value="6개월"/>
기여도	<input type="text" value="100%"/>
내용	<input type="text" value="프리랜서의 포트폴리오 입니다."/>
포트폴리오 첨부	<input type="button" value="파일 선택"/> 파일 3개
<input type="button" value="작성완료"/>	

다중업로드 사용하여 파일 다중 선택

JOB CODE	F02	JOB NAME	포트폴리오 출력	CLASS	freelancerManagement
PROCESS FLOW					



■ 개요

1. 위의 포트폴리오 등록한 것 출력.
2. Ajax 사용하여 비 동기통신으로 traffic을 줄일 수 있다.
3. target View : portfolio.jsp

■ CONTROLLER REQUEST :/showPortfolioList

```

@RequestMapping(value = "/showPortfolioList")
public @ResponseBody String showPortfolioList(Portfolio portfolio) {
    String jsonStr=fm.executeAjax(portfolio, 1);
    return jsonStr;
}

```

1. fm.executeAjax(portfolio, 1) showPortfolioList 메소드를 실행한다.
2. jsonStr 을 리턴한다.

■ SERVICE PARAMETER :portfolio(bean)

```
private void showPortfolioList(Portfolio portfolio) {
    List<Portfolio> pflist = null;
    if(ss!=null && ss.getAttribute("m_id")!=null){
        String m_id = (String) ss.getAttribute("m_id");
        System.out.println(m_id);
        pflist = fDao.getPortfolioList(m_id);
        System.out.println(pflist);
        if(pflist!=null){
            Gson jsonObj=new Gson();
            jsonStr = jsonObj.toJson(pflist);
            System.out.println("jsonStr : "+jsonStr);
        }
    }
}
```

- 빈 List를 불러와 freelancer ID에 맞춰서 리스트를 출력하여 json 형태로 반환한다.

■ DAO & MAPPER PARAMETER :pf_mid(String)

```
public List<Portfolio> getPortfolioList(String pf_mid) {
    return sqlSession.selectList("Portfolio.getPortfolioList", pf_mid);
}

<select id="getPortfolioList" parameterType="String" resultType="portfolio">
    SELECT * FROM PORTFOLIO INNER JOIN PORTFOLIOFILE ON PF_NUM = PT_PFNUM AND PT_NUM =
    (SELECT MIN(PT_NUM) FROM PORTFOLIOFILE WHERE PT_PFNUM=PF_NUM) WHERE PF_MID = #{pf_mid}
</select>
```

- selectList를 이용하여 pf_mid 를 파라미터 값으로 리스트를 출력한다.
- 출력할 때, 포트폴리오 파일중 한가지의 대표이미지가 보여야 하므로 테이블간 INNER JOIN과 서브쿼리 사용하여, 포트폴리오 번호와 포트폴리오 파일 번호가 같고, DB에 저장된 다중파일 중 제일 낮은번호의 파일을 선택하여서 출력한다.

■ TARGET VIEW MODEL :portfolio.jsp

```
$(document).ready(function(){
    var pflist='';
    $.ajax({
        type : 'get',
        url : 'showPortfolioList',
        data : $('#portfolio').serialize(),
        dataType : 'json',
        success : function(data) {
            console.log(data);
            for(var i=0;i<data.length;i++){
                pflist+<tr height="25" align="center">
                    +<td>"+data[i].pf_mid+"님의 포트폴리오입니다+"</td></tr>
                    +<tr><td style="width:400px; height:400px;"><a href='showPortfolioDetail?pfnum="+data[i].pf_num+"'>
                    +<img src='resources/upload/'+data[i].pt_sysname+" style='width:400px; height:400px;' /></a></td></tr>
            }
            $('#pfTable').html(pflist);
        },
        error : function(error) {
            console.log(error);
        }
    });
});
```

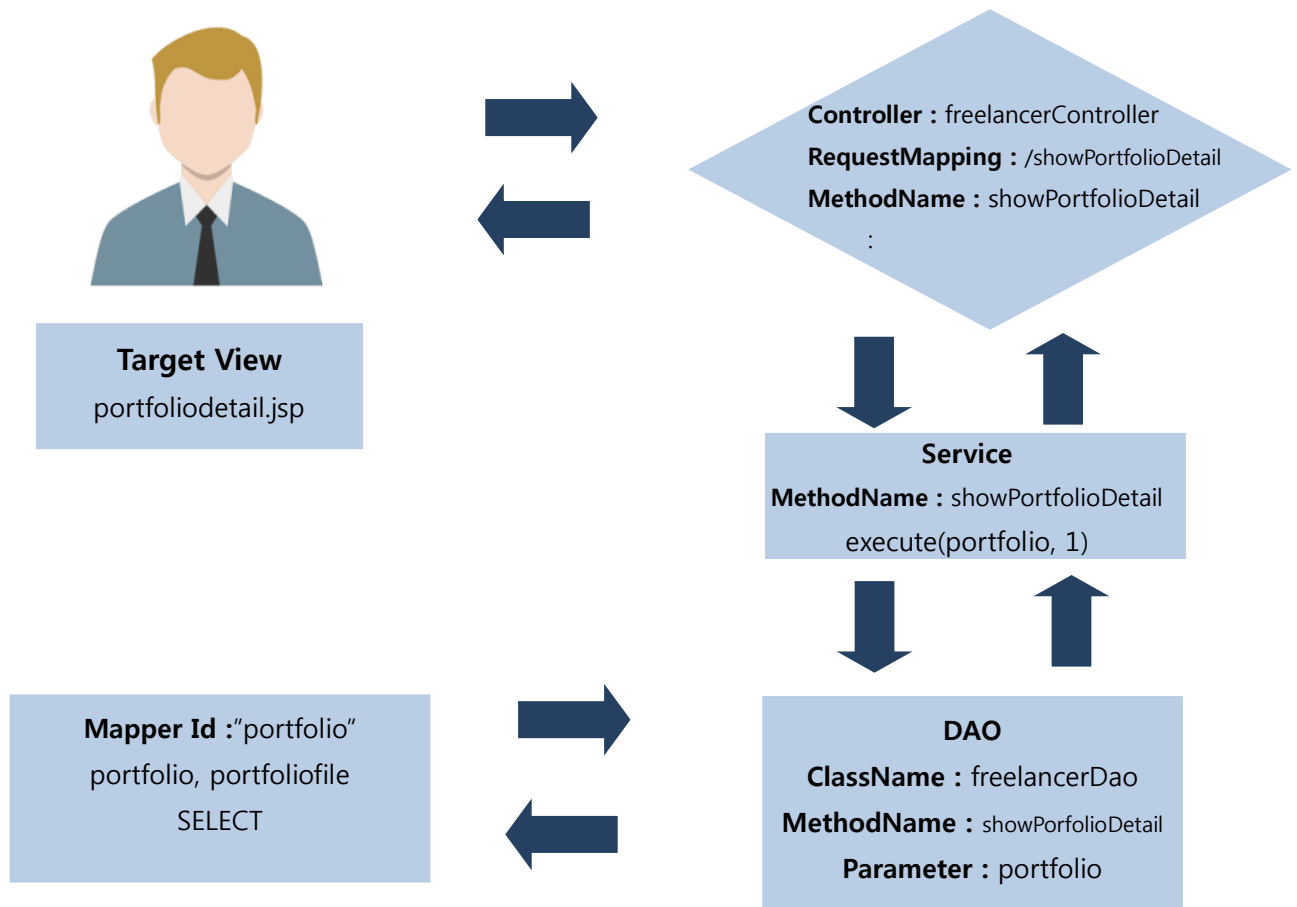
1. ready 함수사용 insert하기 전 이미 저장되어있는 포트폴리오 리스트 출력하기 위해
2. data에 serialize()함수 사용, 폼전체를 전송하는 ajax 전송법
3. datatype : json , 성공시 pfTable에 포트폴리오 번호와 파일출력

siin님의 포트폴리오입니다



위의 포트폴리오 등록에서 작성완료를 누르게되면 Ajax 사용하여 바로 포트폴리오 등록됨.

JOB CODE	F03	JOB NAME	포트폴리오 상세보기	CLASS	freelancerManagement
PROCESS FLOW					



■ 개요

1. 위의 포트폴리오 리스트 클릭시 상세보기로 전환
2. Target View : portfolioDetail.jsp

■ CONTROLLER

REQUEST : /showPortfolioDetail

```

@RequestMapping(value = "/showPortfolioDetail")
public ModelAndView showPortfolioDetail(Portfolio portfolio) {
    mav = fm.execute(portfolio, 1);
    return mav;
}
  
```

1. fm.execute(portfolio, 1) showPortfolioDetail 메소드를 실행한다.
2. mav를 리턴한다.

■ SERVICE PARAMETER :portfolio(bean)

```
private ModelAndView showPortfolioDetail(Portfolio portfolio) {
    mav=new ModelAndView();
    String view = null;
    if(ss!=null && ss.getAttribute("m_id")!=null){
        int pf_num = Integer.parseInt(req.getParameter("pfnum"));
        String pf_mid = (String) ss.getAttribute("m_id");
        List<Portfolio> pf=fDao.getPortfolioDetailList(pf_num);
        if(pf!=null){
            StringBuilder sb=new StringBuilder();
            for(int i=0; i<pf.size(); i++){
                Portfolio pf1=pf.get(i);
                if(i<1){
                    sb.append("<table class='table table-striped' style='text-align:center; color:black;'");
                    sb.append("<tr><td style='width:10%; text-align:center;'>"+<td>"+</td>");
                    sb.append("<td>"+pf1.getPf_title()+"</td></tr>");
                    sb.append("<tr><td style='text-align:center;'>"+<td>"+</td>");
                    sb.append("<td>"+pf1.getPf_term()+"</td></tr>");
                    sb.append("<tr><td style='text-align:center;'>"+<td>"+</td>");
                    sb.append("<td colspan='3'>"+pf1.getPf_contribute()+"</td></tr>");
                    sb.append("<tr><td style='text-align:center;'>"+<td>"+</td>");
                    sb.append("<td colspan='3'>"+pf1.getPf_content()+"</td></tr>");
                    sb.append("<tr rowspan='4'><td style='vertical-align:middle; text-align:center;'>"+<td>"+</td>");
                    sb.append("<td colspan='4'><img style='width:100%; height:100%;' src='resources/upload/'"+pf1.getPt_sysname()+"</td></tr>");
                    sb.append("</table>");
                }else{
                    sb.append("<table class='table table-responsive' style='text-align:center; color:black;'");
                    sb.append("<tr><td><img style='width:100%; height:100%;' src='resources/upload/'"+pf1.getPt_sysname()+"</td></tr>");
                    sb.append("</table>");
                }

                sb.append("<form action='deletePortfolio' method='post' id='portfolio'>");
                sb.append("<input type='hidden' name='pfnum' value='"+pf_num+" />");
                mav.addObject("portfolio",sb.toString());
                mav.addObject("pf_num",pf_num);
            }
            view="portfolioDetail";
            mav.setViewName(view);
        }
        return mav;
    }
}
```

1. pf_num 을 파라미터값으로 fDao.getPortfolioDetailList 가져옴
2. 스트링빌더로 sb.append로 상세내용을 찍는다.
3. for 문으로 List 사이즈만큼 돌게 찍어준다. Pt_sysname은 포트폴리오 파일명인데, 다중업로드에
서 포트폴리오컬럼은 하나인데, 파일은 여러 개일경우를 대비하여 for문 안에 if문을 줘서 이미
지가 한 개이던, 여러 개이던 정상적으로 출력된다.
4. mav.addObject 를 이용하여 mav에 포트폴리오 번호와 상세내용을 담아서 리턴한다.

■ DAO & MAPPER PARAMETER : pf_num

```
public List<Portfolio> getPortfolioDetailList(int pf_num) {
    return sqlSession.selectList("Portfolio.getPortfolioDetailList", pf_num);
}
```

```
<select id="getPortfolioDetailList" parameterType="Integer" resultType="portfolio">
    SELECT * FROM PORTFOLIO INNER JOIN PORTFOLIOFILE ON PF_NUM = PT_PFNUM WHERE PF_NUM = #{pf_num}
</select>
```

1. selectList를 이용하여 pf_num를 파라미터 값으로 출력한다.
2. INNER JOIN 사용하여 포트폴리오 파일 이미지와 포트폴리오 상세내용을 출력한다.

■ TARGET VIEW MODEL :portfolioDetail.jsp

제목	포트폴리오 입니다.
기간	6개월
참여율	100%
내용	프리랜서의 포트폴리오 입니다.

포트폴리오

2. 벤치 마킹

실시간 예약 기능

5. 기능 구현

포트폴리오 수정

portfolioDetail.jsp에서 el로 찍으면 위와 같이 포트폴리오 상세내용이 나오게 된다.