MVC Model2 게시판 구현 프로젝트

오금환

목차

1. 프로젝트 개요

a. 프로젝트 개요

b. 개발 환경

프로젝트 목적,

1. 게시판 구현을 위한 mvc2 모델 활용 기술 습득

- MVC2모델을 이용한 게시판 구현 연습을 통해 Servlet, JSP, Command 가 각각 어떻게 유기적으로 역할을 하는지 파악한다.

2. 게시판 구현을 통한 DB 기본기 숙달.

3. 게시판 구현을 통한 프런트 구성 연습.

1. 프로젝트 개요

a. 프로젝트 개요

b. 개발 환경

개발 툴

개발 언어

Oracle
SQL Developer

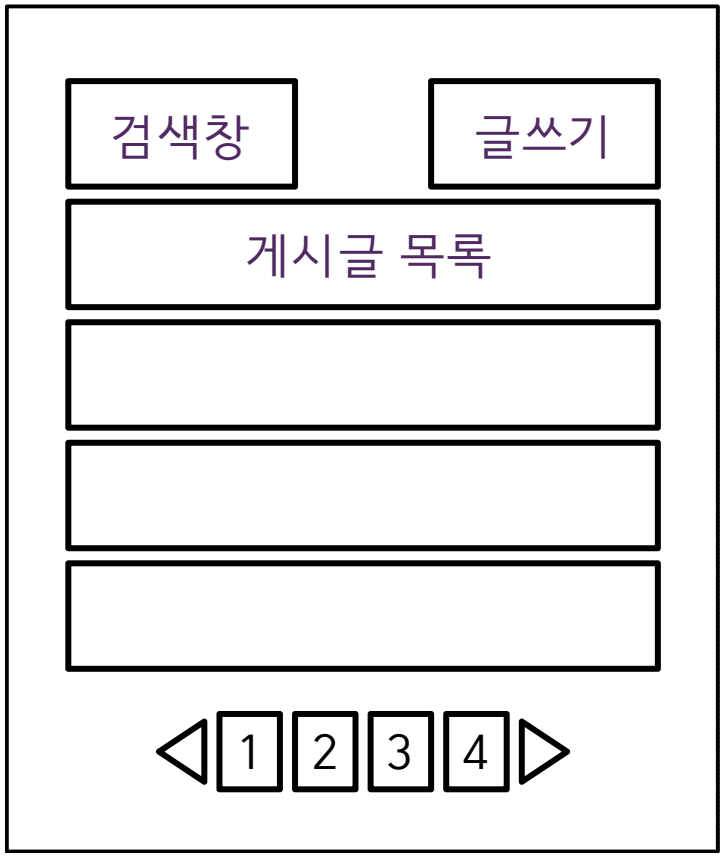
eSoftner

오픈 소스

2. 화면 설계도

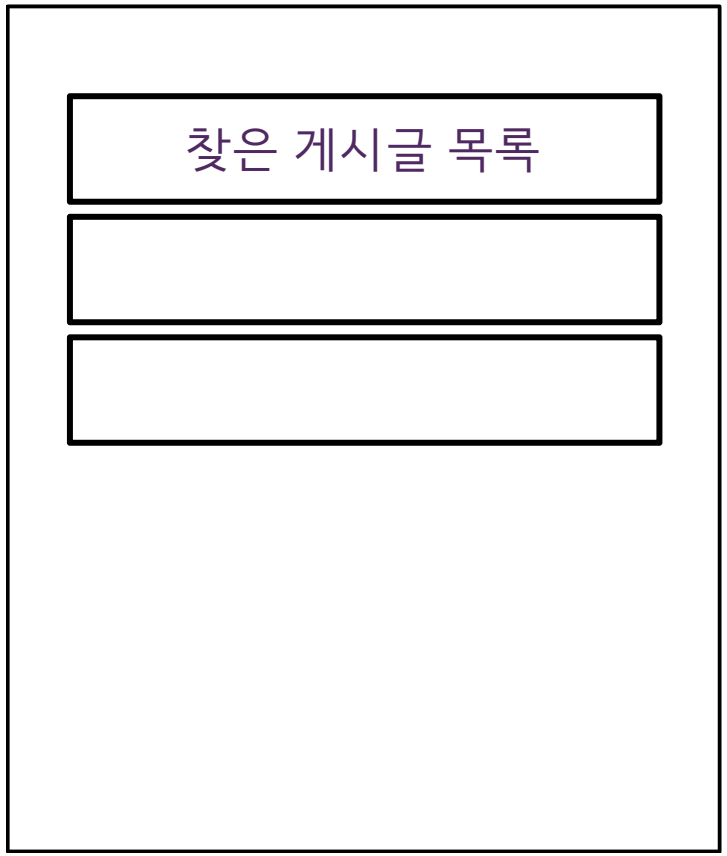
a. 화면 설계도

b. 실제 구현 화면



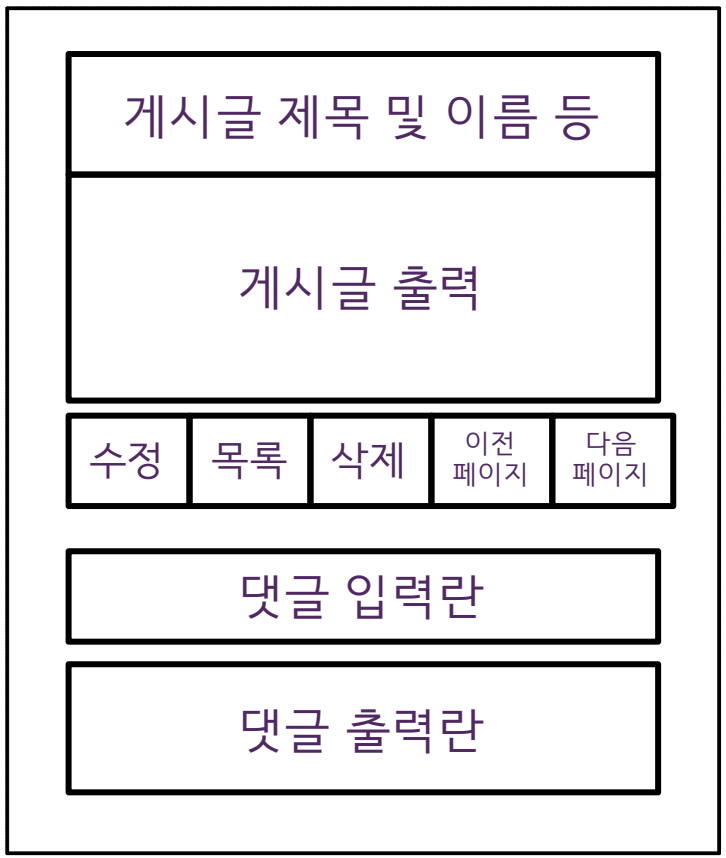
게시글을 출력하는 페이지

게시글의 번호, 제목, 날짜, 작성자를 한줄로해서 10줄씩 출력하고, 그 이상은 다음 페이지로 페이징 처리를 한다.



게시글 리스트 찾기 페이지

게시글 리스트에서 찾고자 하는 키워드를 검색하면 해당 키워드가 제목에 들어가 있는 게시글의 리스트를 띄워주는 페이지

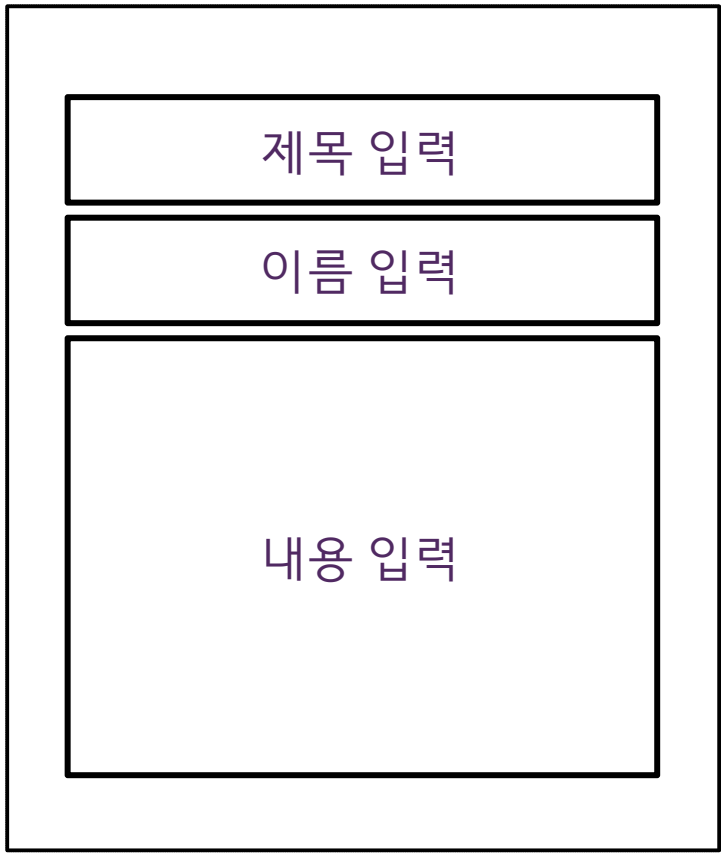


게시글을 출력하는 페이지

게시글의 제목, 작성자 이름, 작성 시간, 조회 수, 게시글 내용을 출력하는 페이지

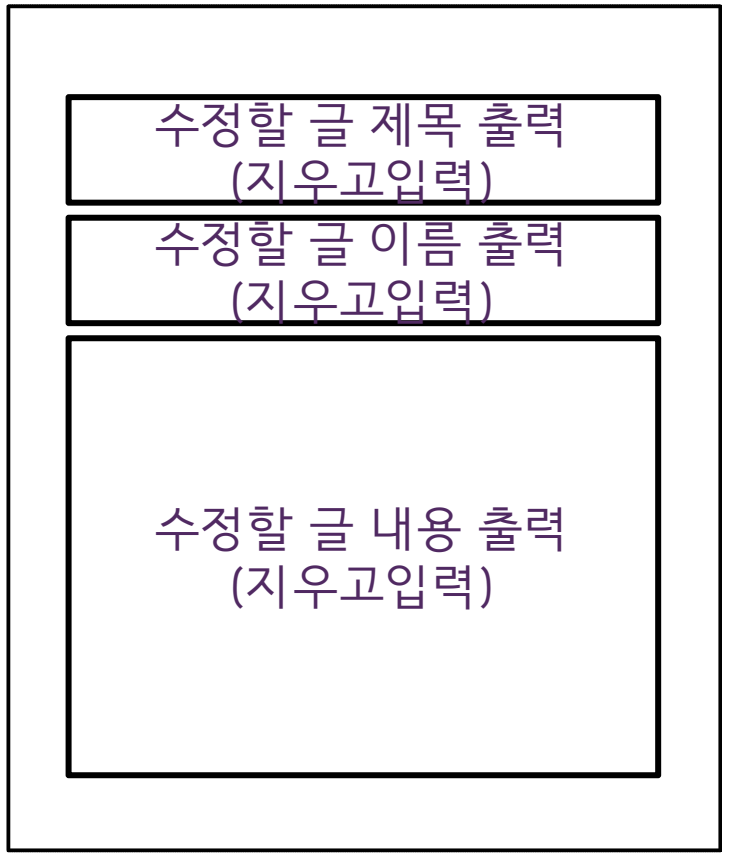
게시글의 댓글을 입,출력하는 페이지

게시글에 대한 댓글 입력이 가능하고, 입력한 댓글은 바로 아래에 출력하도록 한다.



게시글 작성 페이지

제목, 이름, 내용을 입력하는 페이지



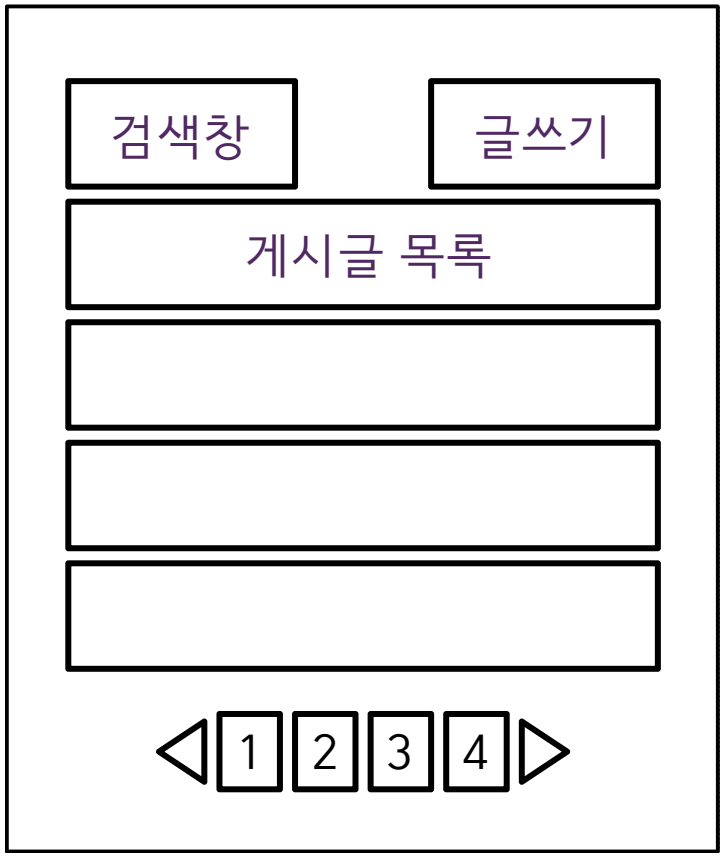
게시글 수정 페이지

출력 화면에는 기존의 글이 함께 뜨도록 하여, 보고서 수정이 가능하도록 구현한다.

2. 화면 설계도

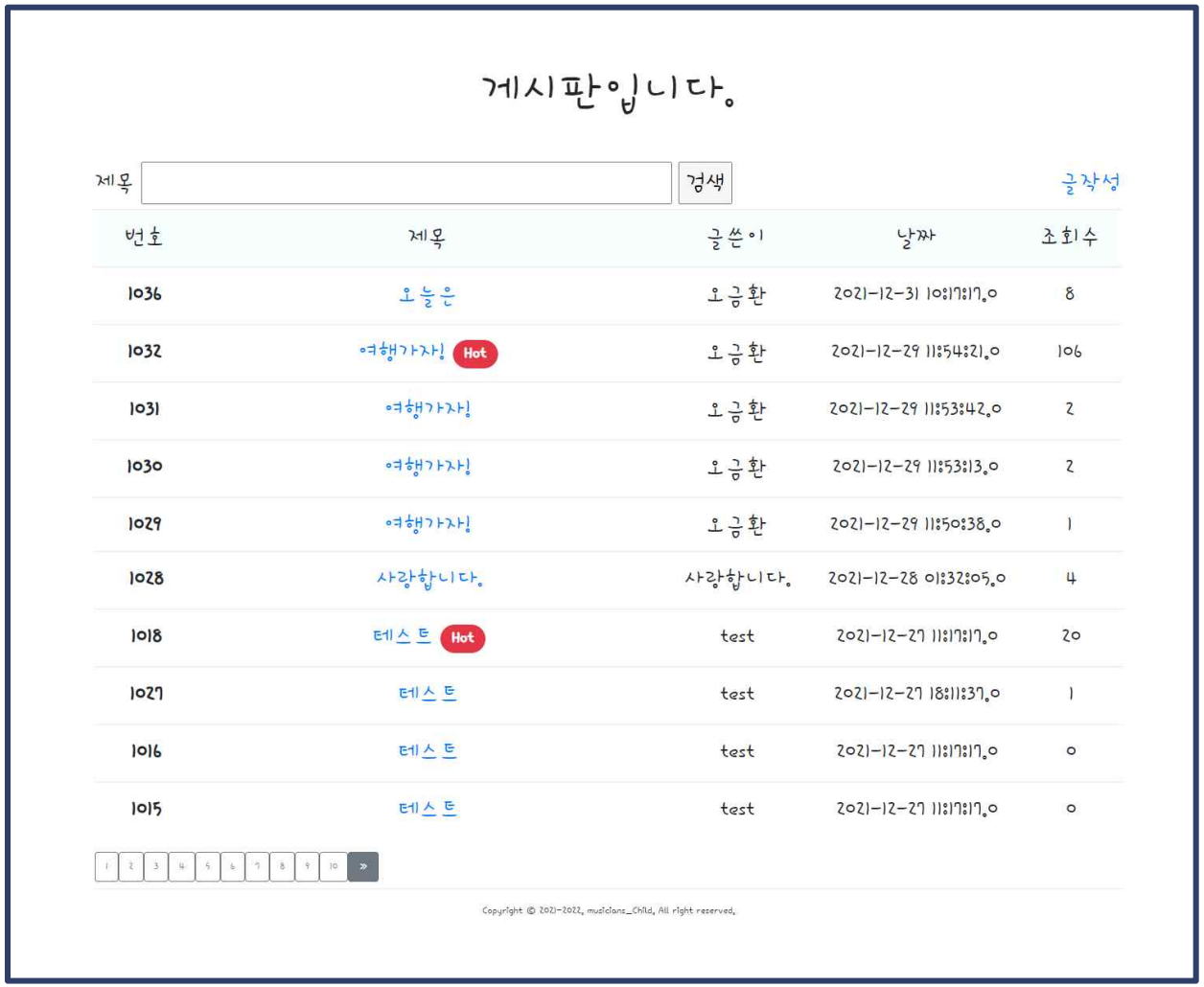
a. 화면 설계도

b. 실제 구현 화면



게시글을 출력하는 페이지

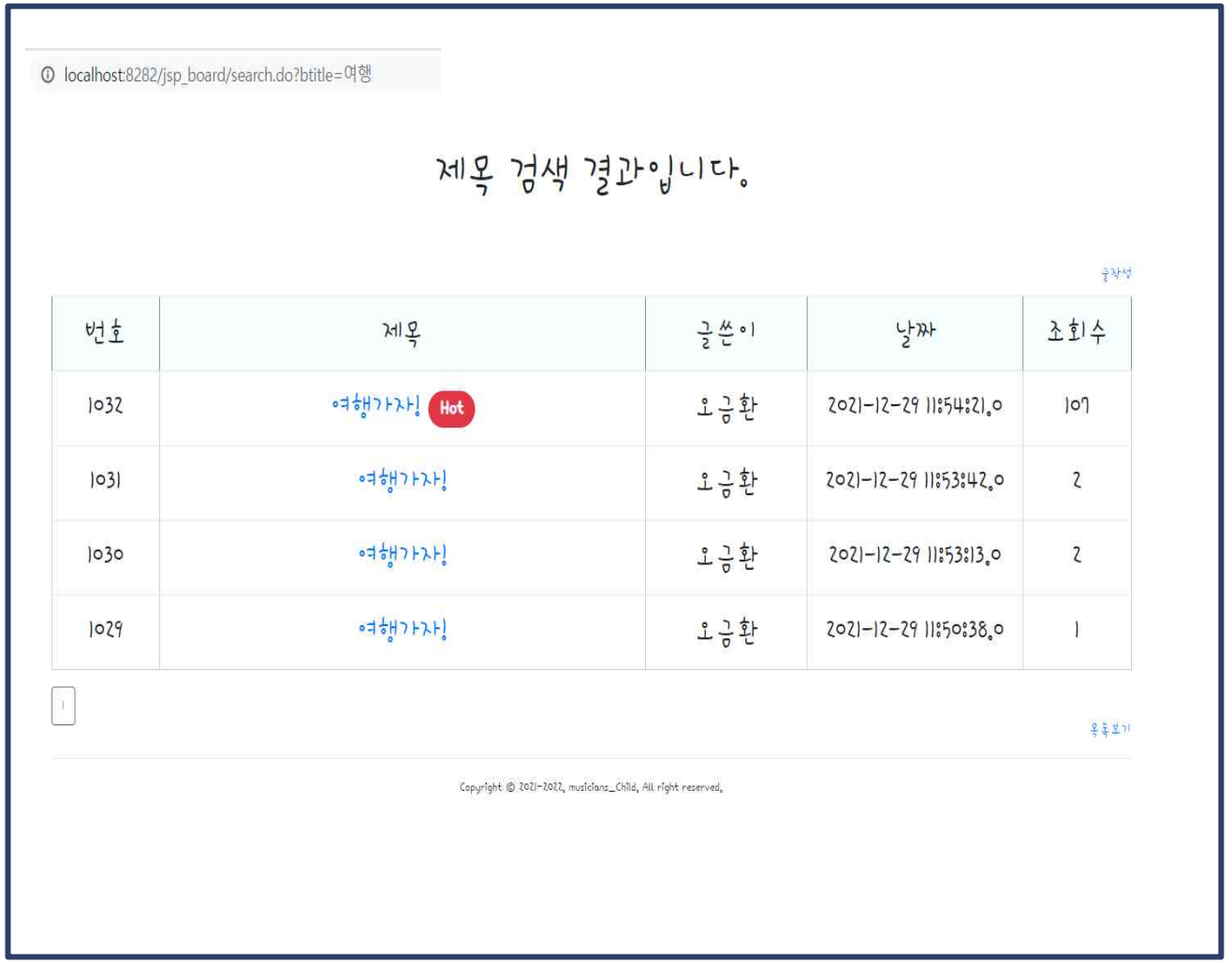
게시글의 번호, 제목, 날짜, 작성자를 한줄로해서 10줄씩 출력하고, 그 이상은 다음 페이지로 페이징 처리를 한다.

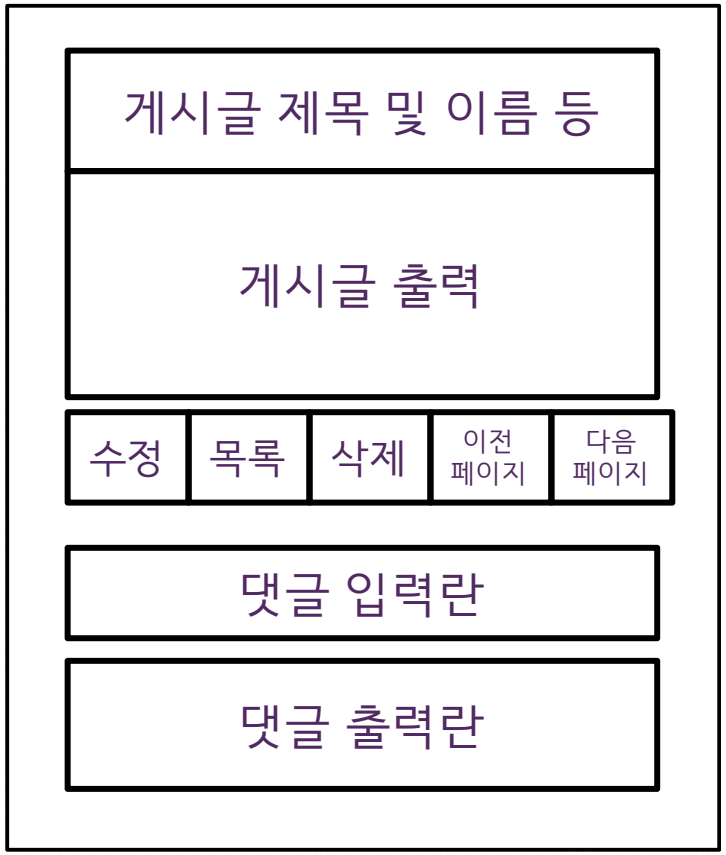


찾은 게시글 목록

게시글 리스트 찾기 페이지

게시글 리스트에서 찾고자 하는 키워드를 검색하면 해당 키워드가 제목에 들어가 있는 게시글의 리스트를 띄워주는 페이지





게시글을 출력하는 페이지

게시글의 제목, 작성자 이름, 작성 시간, 조회 수, 게시글 내용을 출력하는 페이지

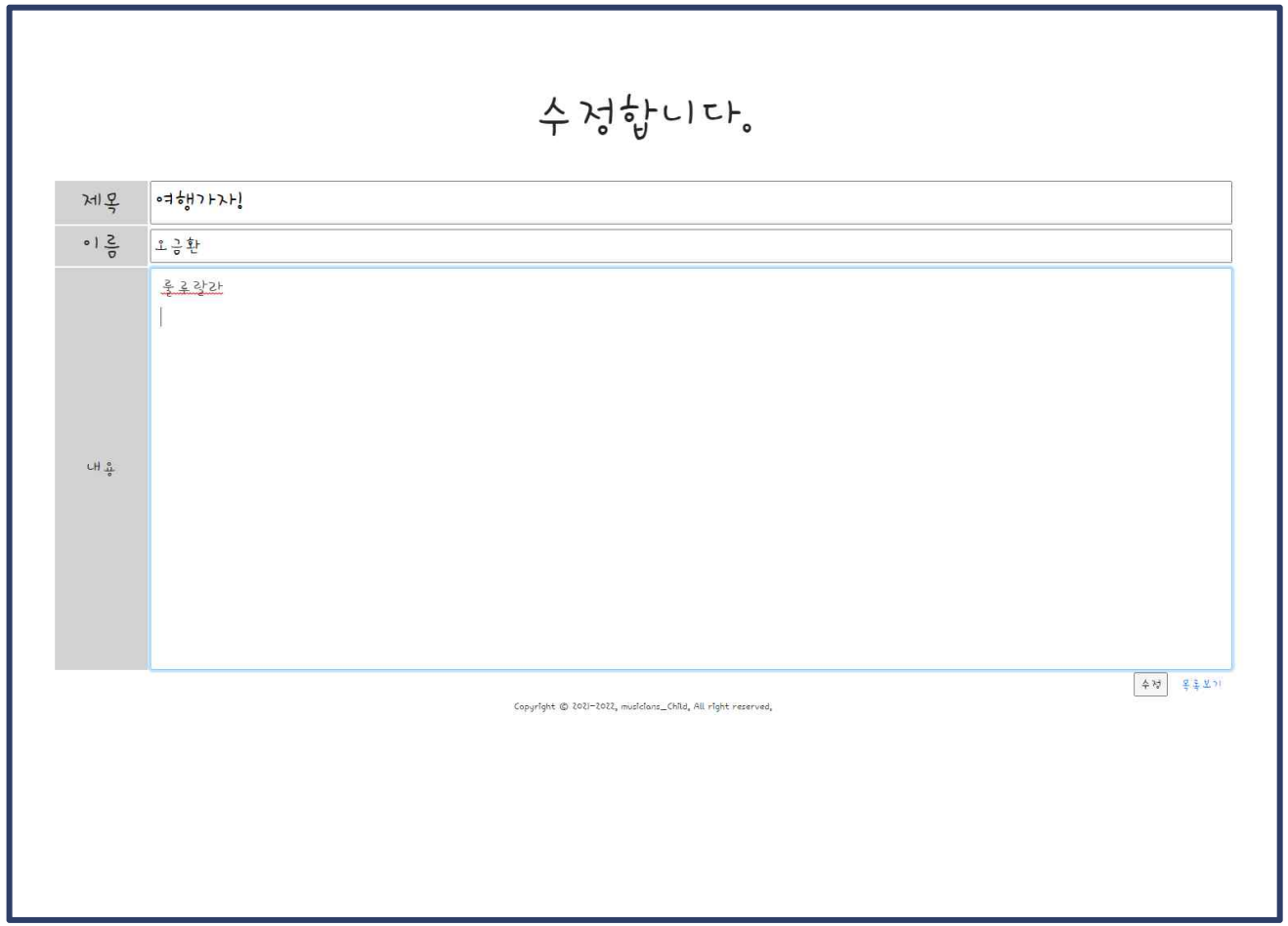
게시글의 댓글을 입,출력하는 페이지

게시글에 대한 댓글 입력이 가능하고, 입력한 댓글은 바로 아래에 출력하도록 한다.

수정할 글 제목 출력
(지우고입력)

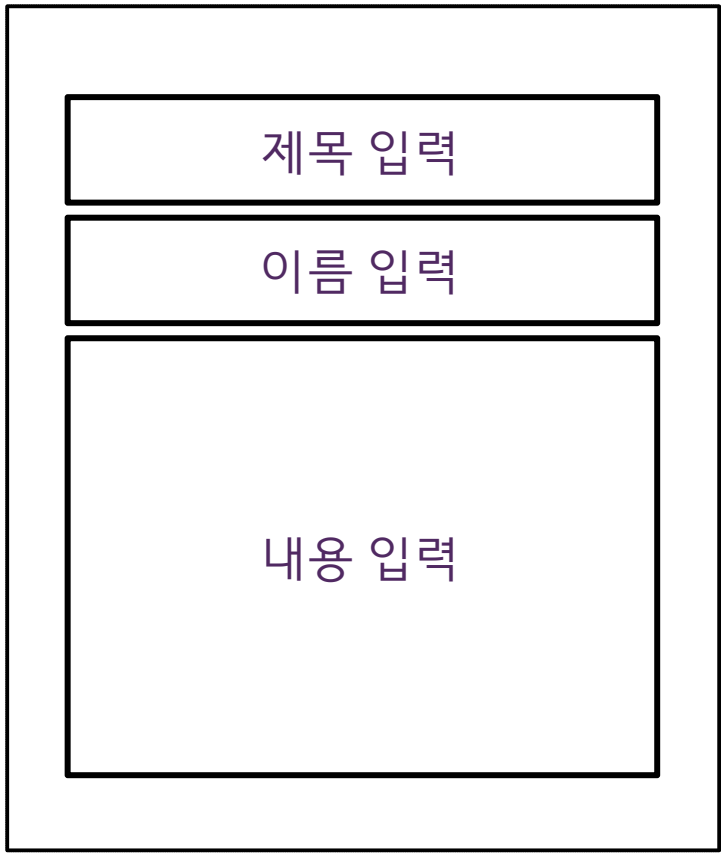
수정할 글 이름 출력
(지우고입력)

수정할 글 내용 출력
(지우고입력)



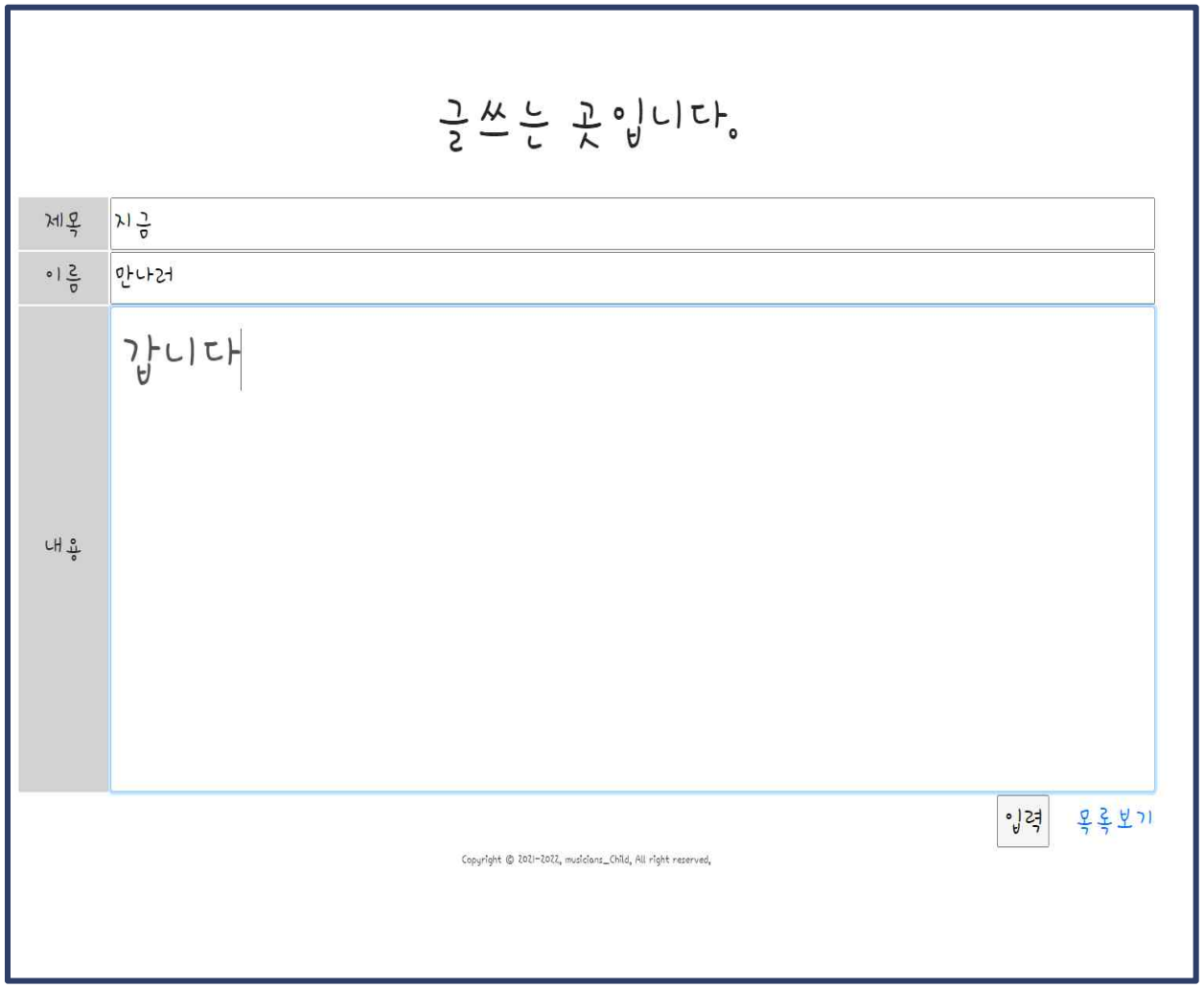
게시글 수정 페이지

출력 화면에는 기존의 글이 함께 뜨도록 하여, 보고서 수정이 가능하도록 구현한다.



게시글 작성 페이지

제목, 이름, 내용을 입력하는 페이지



3. 게시판 설계

a. 데이터베이스 설계

b. MVC Model2 설계

c. 게시판 설계 로직

MVC_BOARD 테이블 생성 sql문

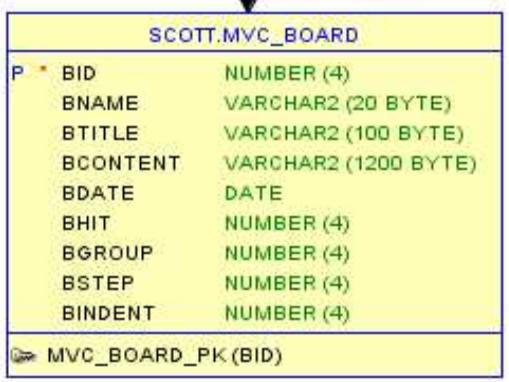
```
create table mvc_board(
    bid NUMBER(4) PRIMARY KEY,
    bname VARCHAR2(20),
    btitle VARCHAR2(100),
    bcontent VARCHAR2(1200),
    bdate DATE DEFAULT SYSDATE,
    bhit NUMBER(4) DEFAULT 0,
    bgroup NUMBER(4),
    bstep NUMBER(4),
    bindent NUMBER(4)
);

create SEQUENCE mvc_board_seq;
```

MVC_BOARD의 데이터 구성요소 설명

bid	게시글 번호
bname	이름(작성자)
btitle	글 제목
bcontent	내용
bdate	글 작성 날짜
bhit	히트(조회수)
bgroup	원본글과 댓글의 그룹 번호
bstep	리스트에서 원본글 대비 댓글의 상, 하 위치
bindent	리스트에서 댓글의 들여쓰기 위치

MVC_BOARD Data dictionary



글 정렬 기준을 BID로 하기 위해
BID(게시글 번호)에 Primary key를 설정하여
중복되지 않고, null값이 입력되지 않도록 설정.

REPLY_BOARD 테이블 생성 sql문

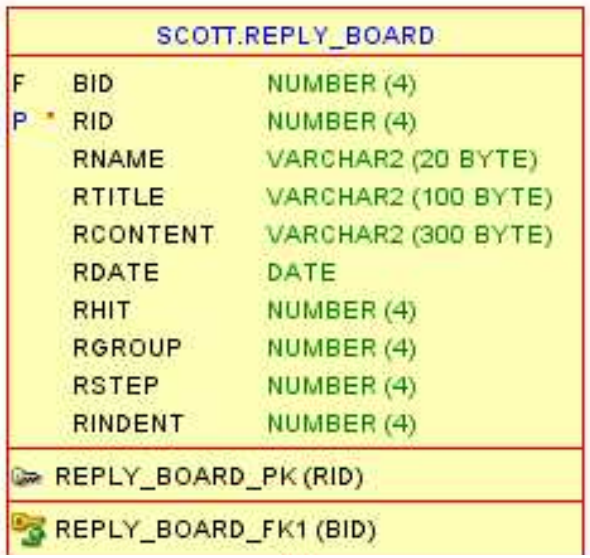
```
create table reply_board(
    bid NUMBER(4) references bid,
    rid NUMBER(4),
    rname VARCHAR2(20),
    rtitle VARCHAR2(100),
    rcontent VARCHAR2(300),
    rdate DATE DEFAULT SYSDATE,
    rhit NUMBER(4) DEFAULT 0,
    rgroup NUMBER(4),
    rstep NUMBER(4),
    rindent NUMBER(4)
);

create SEQUENCE reply_board_seq;
```

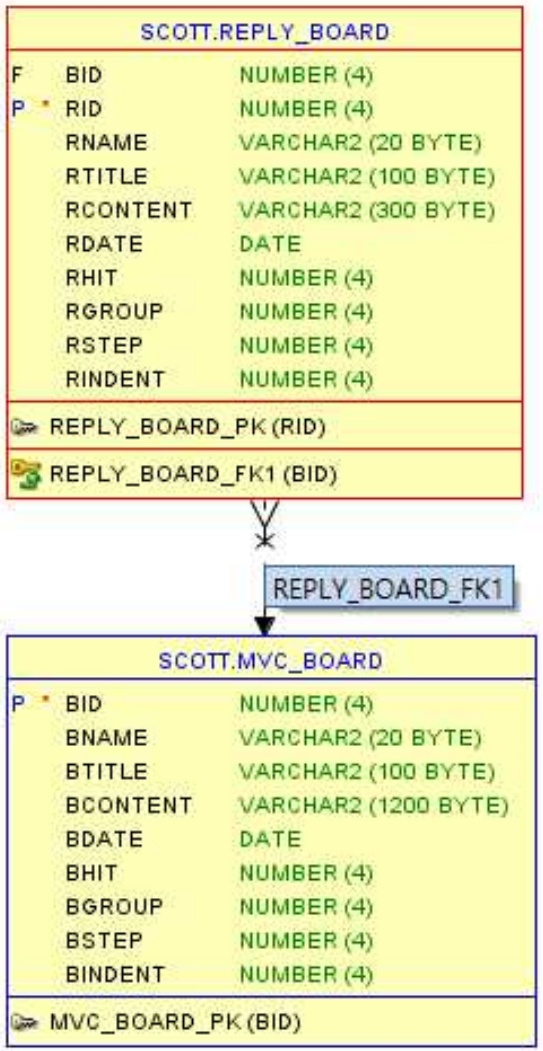
REPLY_BOARD의 데이터 구성요소 설명

bid	게시글 번호
rid	댓글 번호(primary key: 전체 중복 안 되며, 값이 꼭 들어가게 함)
bname	이름(작성자)
btitle	글 제목
bcontent	내용
bdate	글 작성 날짜
bhit	히트(조회수)
bgroup	원본글과 댓글의 그룹 번호
bstep	리스트에서 원본글 대비 댓글의 상, 하 위치
bindent	리스트에서 댓글의 들여쓰기 위치

REPLY_BOARD Data dictionary



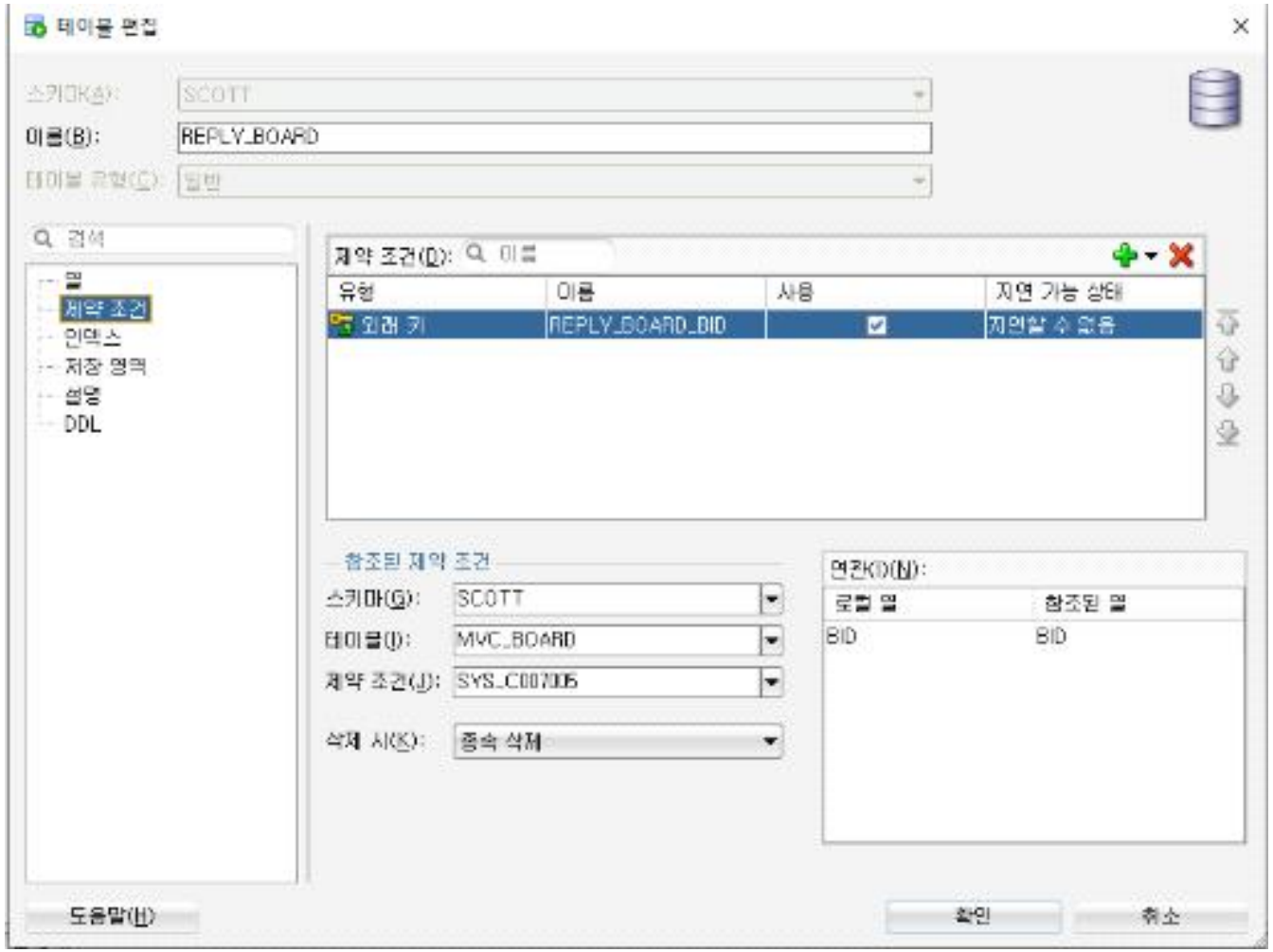
글 정렬 기준을 RID로 하기 위해
RID(게시글 번호)에 Primary key를 설정하여
중복되지 않고, null값이 입력되지 않도록 설정.



MVC_BOARD의 게시글을 불러올 때 댓글을

REPLY_BOARD에서 불러올 수 있도록,

reply_board_F1 이라는 이름으로 **forigner key**를 부여하였다.



MVC_BOARD의 글을 삭제할 경우
댓글을 삭제할 수 있도록
MVC_BOARD와 REPLY_BOARD의 BID
에 부여된 FORIGNER KEY의 조건에
삭제 시 종속삭제를 설정.

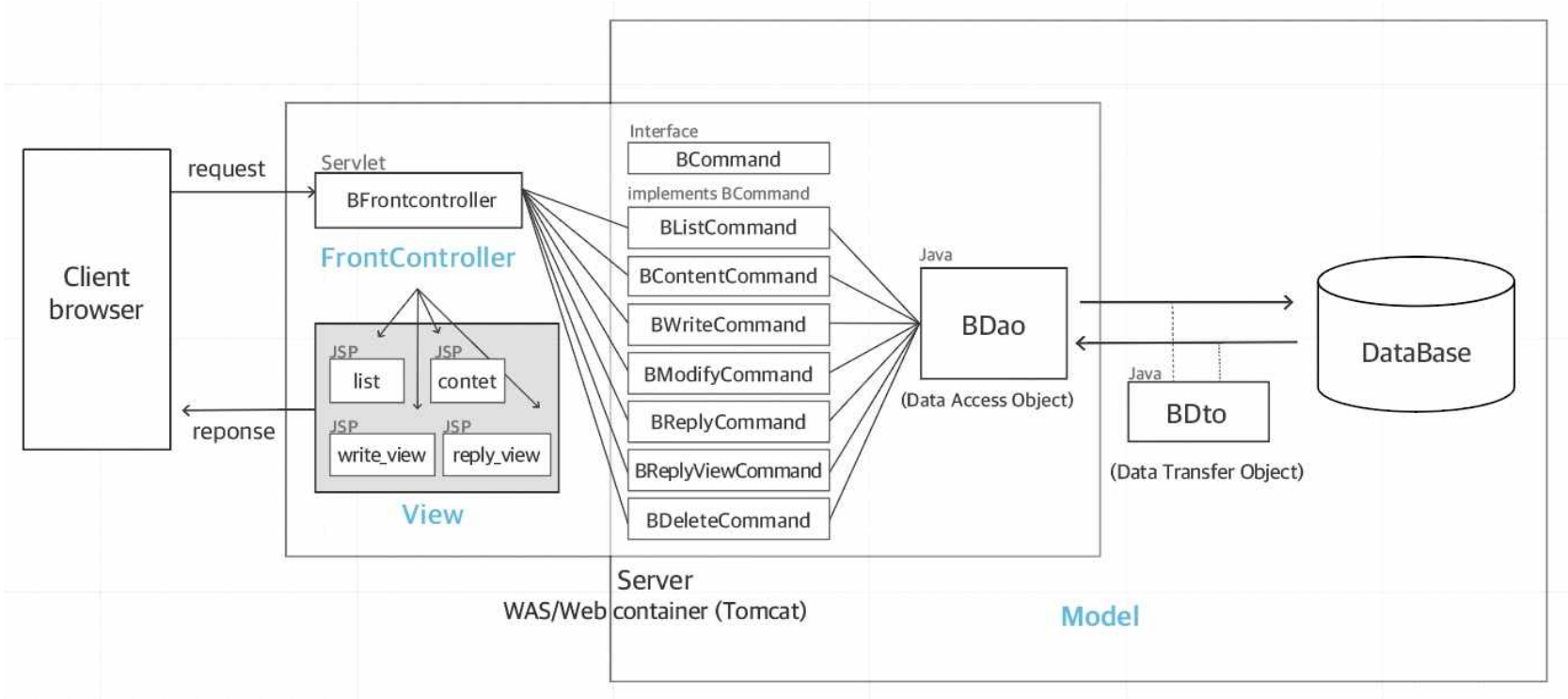
3. 게시판 설계

a. 데이터베이스 설계

b. MVC Model2 설계

c. 게시판 설계 로직

설계 모델 - MVC Model2



Model, View, Controller로 모듈화를 시켜서 역할을 구분해서 설계함

- 역할이 나누어져 있으므로 개발 과정에서 유지 보수 가 용이함
- 확장 가능성을 염두에 둠

Model

BCommand.java

BContentCommand.java
BContentNextCommand.java
BContentPreviousCommand.java

BDeleteCommand.java

BListCommand.java
BListCommand2.java

BModifyCommnad.java
BModifyViewCommand.java

BReplyCommand.java
BReplyDeleteCommand.java
BReplyViewCommand.java

BSearchCommand.java
BSearchCommand2.java

BWriteCommand.java

BDao.java

BDto.java
BtitleDto.java
RDto.java

Criteria.java
PageVO.java

View

content_view.jsp
list.jsp
modify_view.jsp
search_view.jsp
write_view.jsp

Controller

BController.java

MVC 구성

MVC Model2 게시판 설계

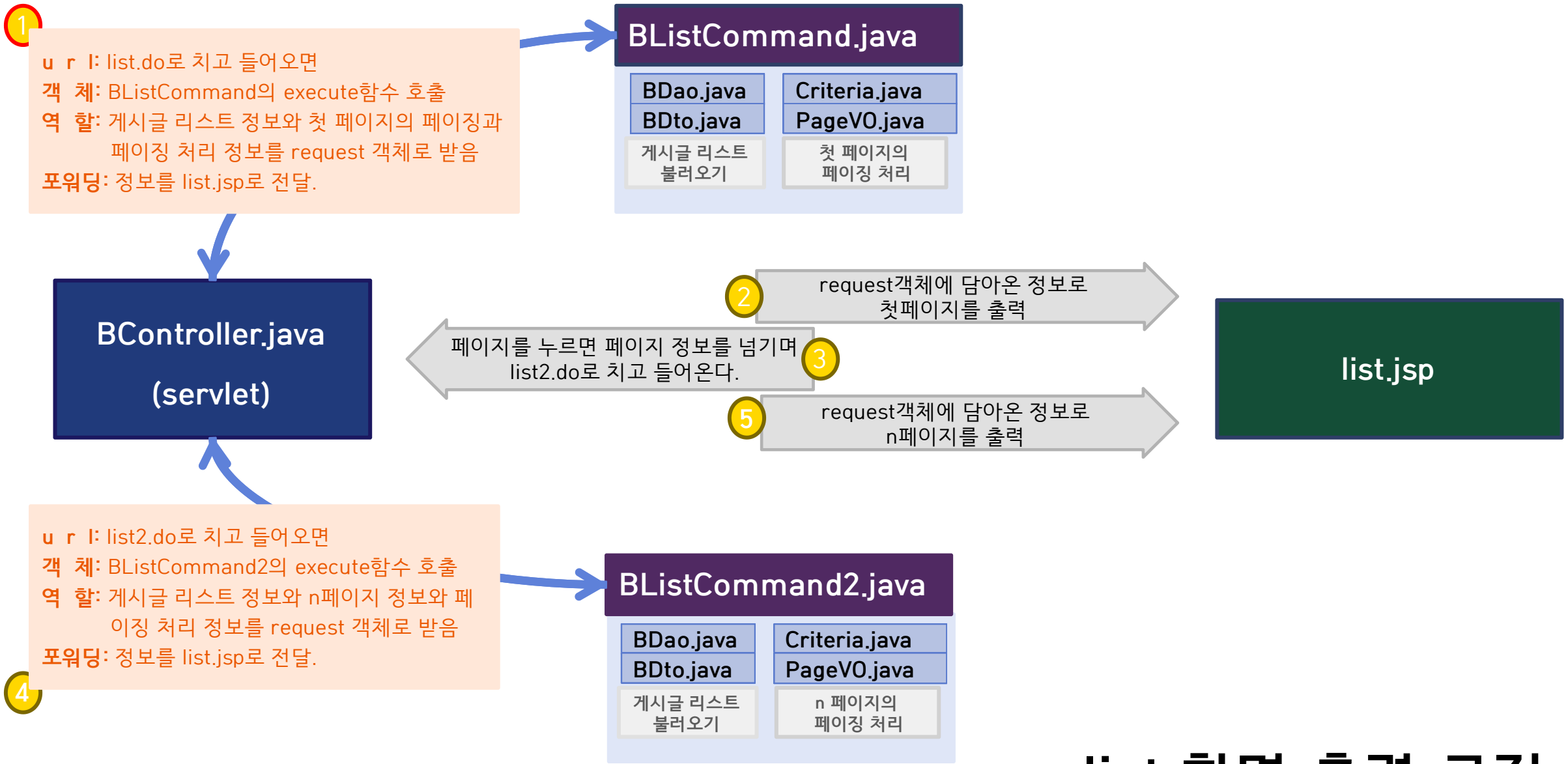
	Controller	Model	View
게시글 리스트	BController.java (servlet)	BListCommand.java BListCommand2.java	list.jsp
게시글 보기		BContentCommand.java BContentNextCommand.java BContentPreviousCommand.java	content_view.jsp
게시글 쓰기		BWriteCommand.java	write_view.jsp
게시글 수정		BModifyCommnad.java BModifyViewCommand.java	modify_view.jsp
게시글 삭제		BDeleteCommand.java	content_view.jsp
댓글		BReplyCommand.java BReplyDeleteCommand.java BReplyViewCommand.java	content_view.jsp
게시글 검색		BSearchCommand.java BSearchCommand2.java	search_view.jsp
		BDao.java Criteria.java PageVO.java BDto.java BtitleDto.java RDto.java	

3. 게시판 설계

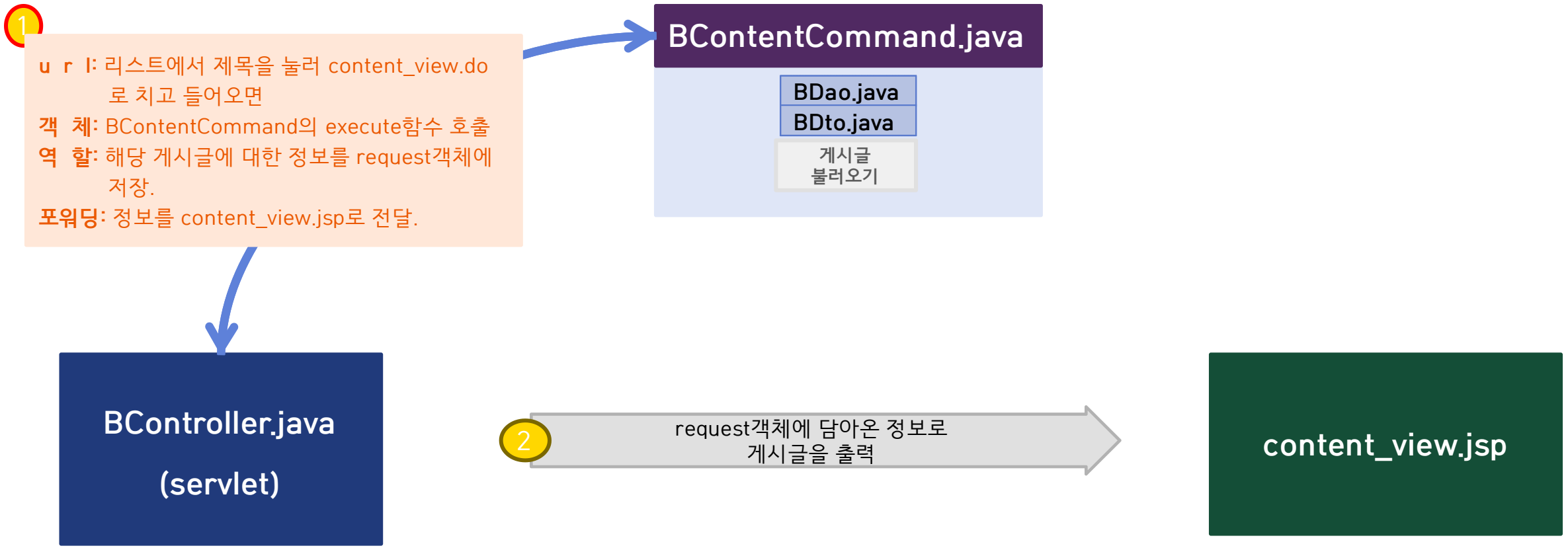
a. 데이터베이스 설계

b. MVC Model2 설계

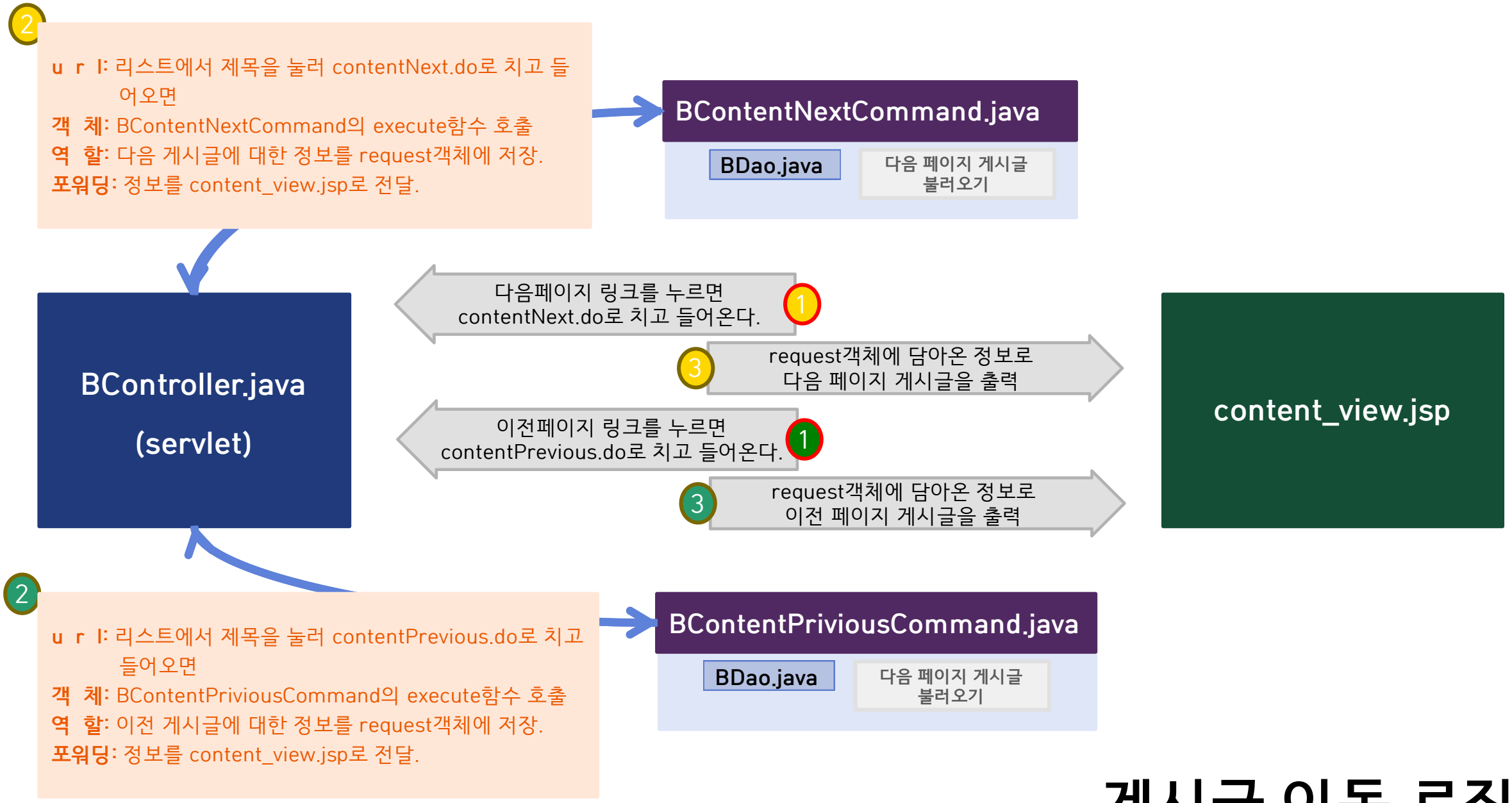
c. 게시판 설계 로직


1
url: list.do로 치고 들어오면
객 체: BListCommand의 execute함수 호출
역 할: 게시글 리스트 정보와 첫 페이지의 페이징과 페이징 처리 정보를 request 객체로 받음
포워딩: 정보를 list.jsp로 전달.
BListCommand.java
BDao.java
BDto.java
게시글 리스트 불러오기
Criteria.java
PageVO.java
첫 페이지의 페이징 처리
BController.java (servlet)
2
request객체에 담아온 정보로 첫페이지를 출력
페이지를 누르면 페이지 정보를 넘기며 list2.do로 치고 들어온다.
3
5
request객체에 담아온 정보로 n페이지를 출력
list.jsp
url: list2.do로 치고 들어오면
객 체: BListCommand2의 execute함수 호출
역 할: 게시글 리스트 정보와 n페이지 정보와 페이징 처리 정보를 request 객체로 받음
포워딩: 정보를 list.jsp로 전달.
4
BListCommand2.java
BDao.java
BDto.java
게시글 리스트 불러오기
Criteria.java
PageVO.java
n 페이지의 페이징 처리

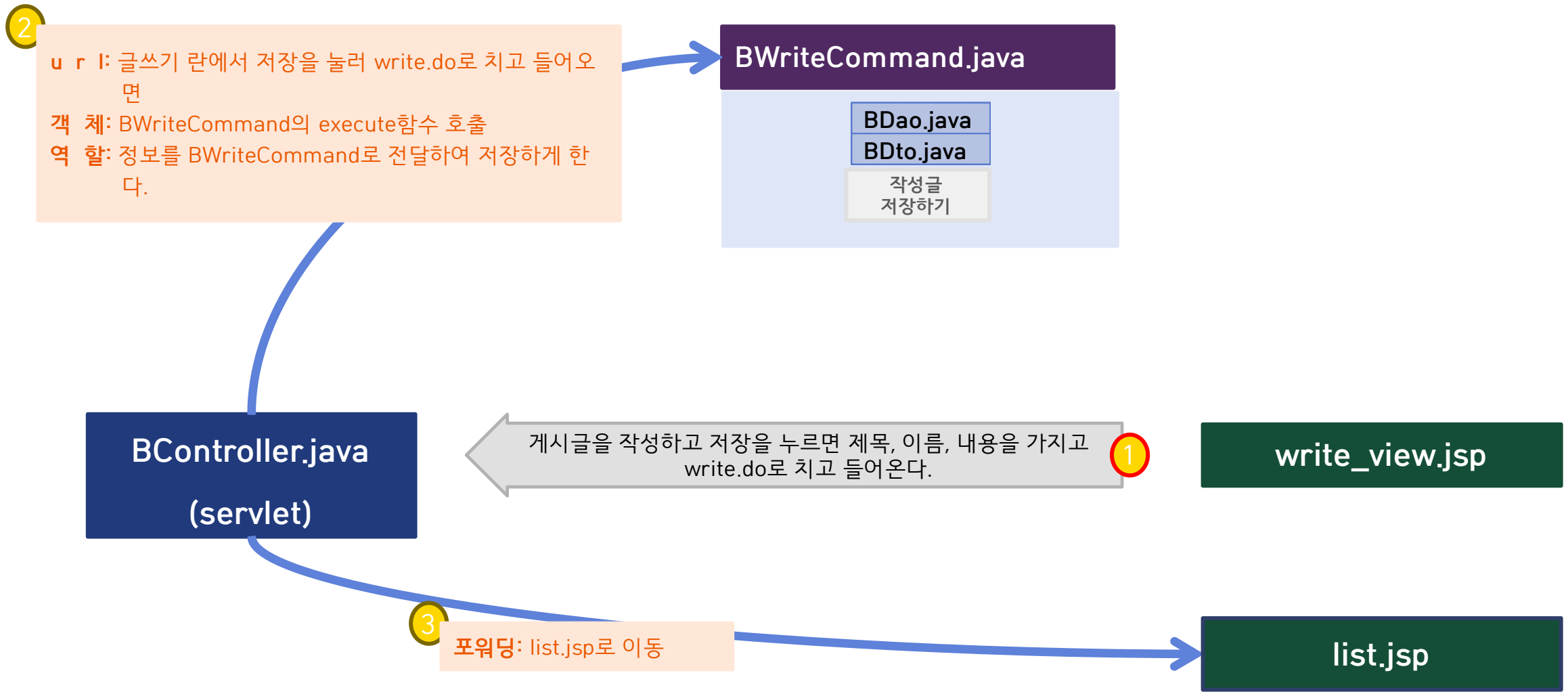
list 화면 출력 로직


1
url: 리스트에서 제목을 눌러 content_view.do 로 치고 들어오면
객 체: BContentCommand의 execute함수 호출
역 할: 해당 게시글에 대한 정보를 request객체에 저장.
포워딩: 정보를 content_view.jsp로 전달.
BContentCommand.java
BDao.java
BDto.java
게시글 불러오기
BController.java (servlet)
2
request객체에 담아온 정보로 게시글을 출력
content_view.jsp

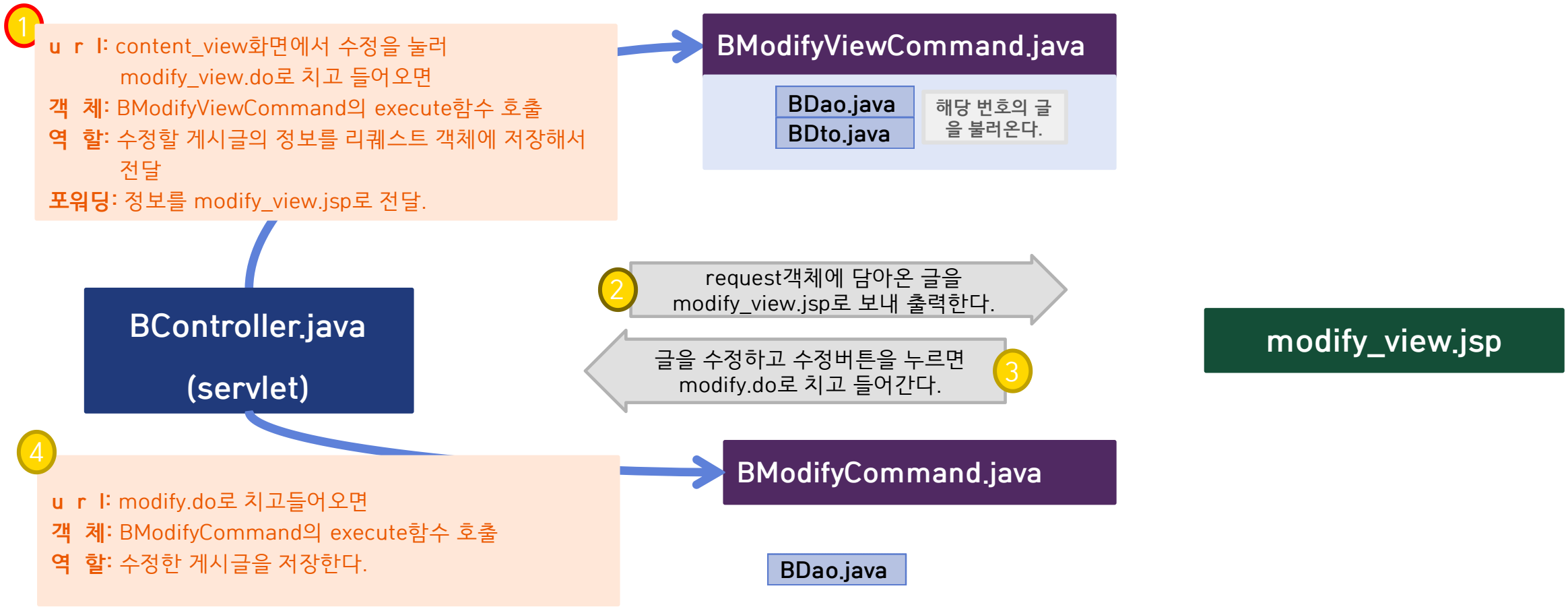
게시글 출력 로직


2
url: 리스트에서 제목을 눌러 contentNext.do로 치고 들어오면
객 체: BContentNextCommand의 execute함수 호출
역 할: 다음 게시글에 대한 정보를 request객체에 저장.
포워딩: 정보를 content_view.jsp로 전달.
BContentNextCommand.java
BDao.java
다음 페이지 게시글 불러오기
BController.java
(servlet)
다음페이지 링크를 누르면 contentNext.do로 치고 들어온다.
1
3
request객체에 담아온 정보로 다음 페이지 게시글을 출력
content_view.jsp
이전페이지 링크를 누르면 contentPrevious.do로 치고 들어온다.
1
3
request객체에 담아온 정보로 이전 페이지 게시글을 출력
2
url: 리스트에서 제목을 눌러 contentPrevious.do로 치고 들어오면
객 체: BContentPriviousCommand의 execute함수 호출
역 할: 이전 게시글에 대한 정보를 request객체에 저장.
포워딩: 정보를 content_view.jsp로 전달.
BContentPriviousCommand.java
BDao.java
다음 페이지 게시글 불러오기

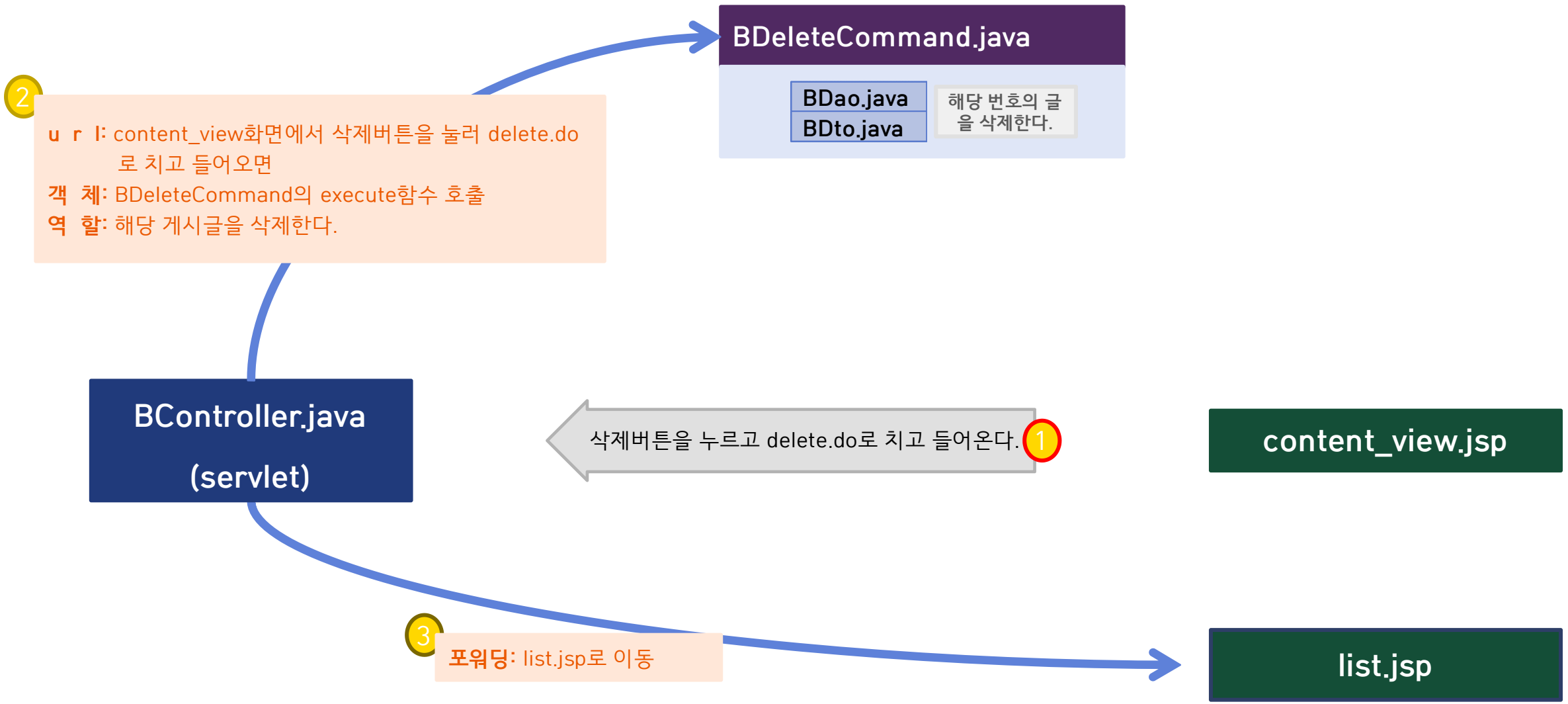
게시글 이동 로직


2
url: 글쓰기 란에서 저장을 눌러 write.do로 치고 들어오면
객 체: BWriteCommand의 execute함수 호출
역 할: 정보를 BWriteCommand로 전달하여 저장하게 한다.
BWriteCommand.java
BDao.java
BDto.java
작성글 저장하기
BController.java (servlet)
게시글을 작성하고 저장을 누르면 제목, 이름, 내용을 가지고 write.do로 치고 들어온다.
1
write_view.jsp
3
포워딩: list.jsp로 이동
list.jsp

게시글 작성 로직


1
u r l: content_view화면에서 수정을 눌러
modify_view.do로 치고 들어오면
객 체: BModifyViewCommand의 execute함수 호출
역 할: 수정할 게시글의 정보를 리퀘스트 객체에 저장해서
전달
포워딩: 정보를 modify_view.jsp로 전달.
BModifyViewCommand.java
BDao.java
BDto.java
해당 번호의 글
을 불러온다.
BController.java
(servlet)
2
request객체에 담아온 글을
modify_view.jsp로 보내 출력한다.
글을 수정하고 수정버튼을 누르면
modify.do로 치고 들어간다.
3
modify_view.jsp
4
u r l: modify.do로 치고들어오면
객 체: BModifyCommand의 execute함수 호출
역 할: 수정한 게시글을 저장한다.
BModifyCommand.java
BDao.java

게시글 수정 로직


BDeleteCommand.java
BDao.java
BDto.java
해당 번호의 글을 삭제한다.
2
url: content_view화면에서 삭제버튼을 눌러 delete.do로 치고 들어오면
객 체: BDeleteCommand의 execute함수 호출
역 할: 해당 게시글을 삭제한다.
BController.java
(servlet)
삭제버튼을 누르고 delete.do로 치고 들어온다.
1
content_view.jsp
3
포워딩: list.jsp로 이동
list.jsp

게시글 삭제 로직

2

u r l: reply.do로 치고 들어오고
객 체: BReplyCommand의 execute함수도 호출되어
역 할: 해당 게시글과에 입력된 댓글이 저장된다.

BReplyCommand.java

BDao.java RDto.java	해당 번호의 댓글을 단다.

BController.java

(servlet)

list.jsp에서 특정 글을 눌러 content_view.jsp로 치고 들어간다. 1

content_view.jsp에서 이름과 내용을 입력하여 댓글을 쓰고 입력버튼을 누르면 1

3 content_view.jsp를 포워딩 한다.

content_view.jsp

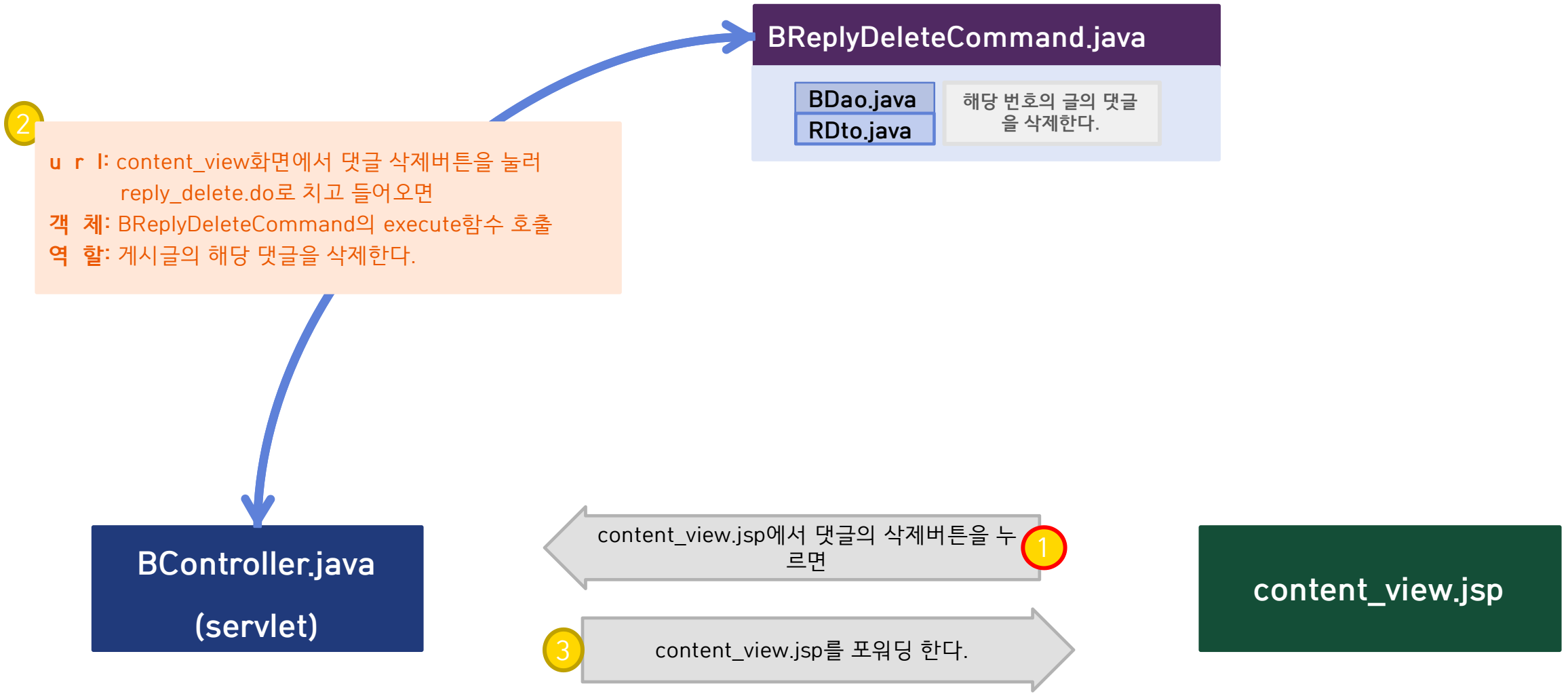
2

u r l: content_view.do로 치고 들어오면,
객 체: BContnentCommand의 execute함수 호출과 함께
BReplyViewCommand의 execute함수도 호출되어
역 할: 해당 게시글과 해당 게시글의 댓글도 출력한다.

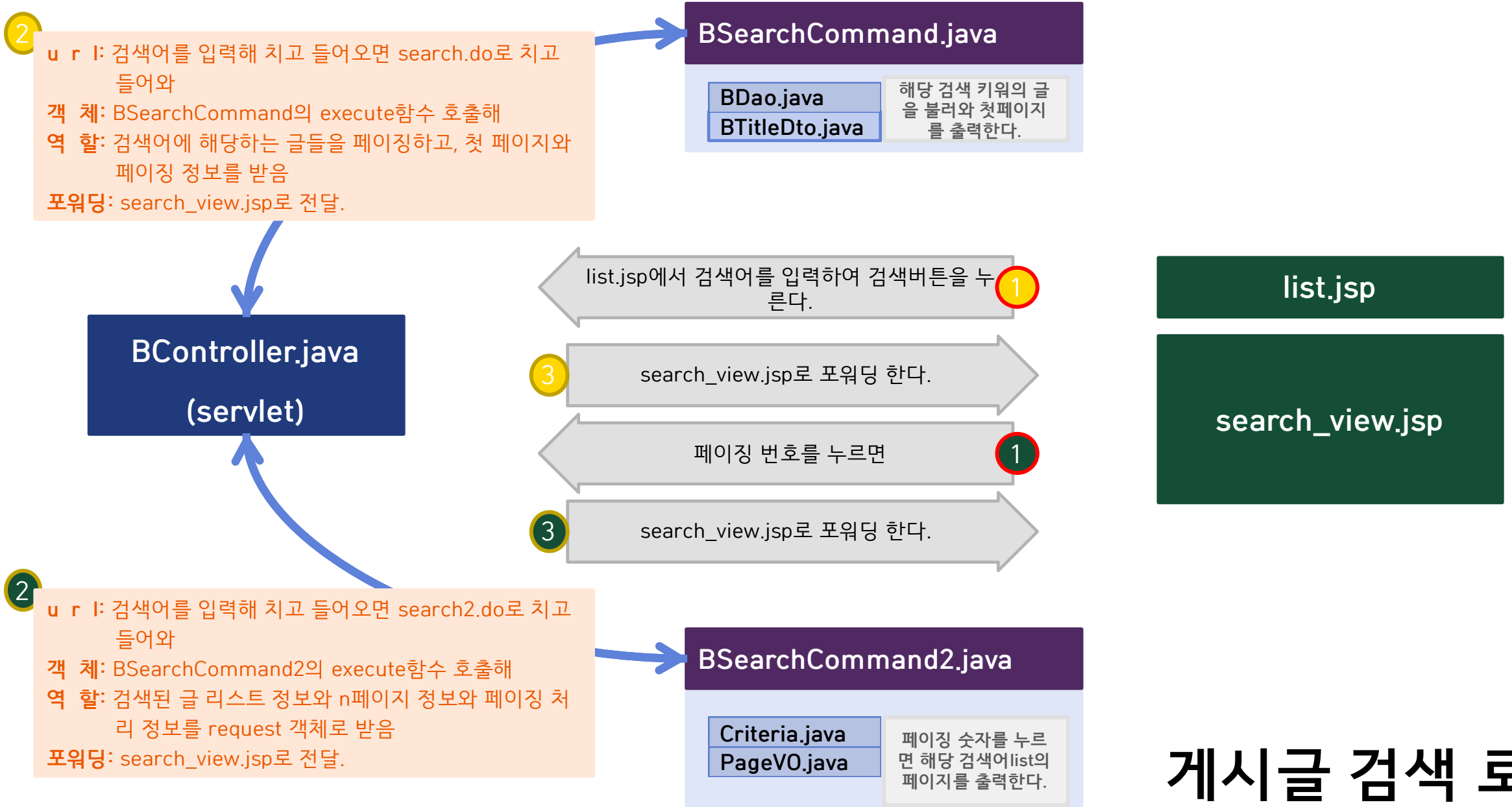
BReplyViewCommand.java

BDao.java RDto.java	해당 번호의 댓글을 출력한다.

댓글 입*출력 로직


BReplyDeleteCommand.java
BDao.java
RDto.java
해당 번호의 글의 댓글을 삭제한다.
2
u r l: content_view화면에서 댓글 삭제버튼을 눌러 reply_delete.do로 치고 들어오면
객 체: BReplyDeleteCommand의 execute함수 호출
역 할: 게시글의 해당 댓글을 삭제한다.
BController.java
(servlet)
content_view.jsp에서 댓글의 삭제버튼을 누르면
1
3
content_view.jsp를 포워딩 한다.
content_view.jsp

댓글 삭제 로직


2
url: 검색어를 입력해 치고 들어오면 search.do로 치고 들어와
객 체: BSearchCommand의 execute함수 호출해
역 할: 검색어에 해당하는 글들을 페이징하고, 첫 페이지와 페이징 정보를 받음
포워딩: search_view.jsp로 전달.
BSearchCommand.java
BDao.java
BTitleDto.java
해당 검색 키워의 글을 불러와 첫페이지를 출력한다.
BController.java
(servlet)
list.jsp에서 검색어를 입력하여 검색버튼을 누른다.
1
3
search_view.jsp로 포워딩 한다.
페이징 번호를 누르면
1
3
search_view.jsp로 포워딩 한다.
list.jsp
search_view.jsp
2
url: 검색어를 입력해 치고 들어오면 search2.do로 치고 들어와
객 체: BSearchCommand2의 execute함수 호출해
역 할: 검색된 글 리스트 정보와 n페이지 정보와 페이징 처리 정보를 request 객체로 받음
포워딩: search_view.jsp로 전달.
BSearchCommand2.java
Criteria.java
PageVO.java
페이징 숫자를 누르면 해당 검색어list의 페이지를 출력한다.

게시글 검색 로직

4. 상세 구현 내용

a. controller

b. command

c. dao

d. dto

e. view

```java
package edu.kosmo.ex.controller;

import java.io.IOException;

/**
 * Servlet implementation class BController
 */
@WebServlet("*.do")
public class BController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public BController() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("doGet");
        actionDo(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
     *      response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("doPost");
        actionDo(request, response);
```

*.do를 치고 들어오는 모든 경로는
controller가 받고,
형식이 doget이든, dopost이든
모두 actoinDo를 타도록 구현.

```java
    protected void actionDo(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
        System.out.println("actionDo");
        request.setCharacterEncoding("UTF-8");

        String viewPage = null;
        BCommand command = null;
        BCommand command1 = null;

        String uri = request.getRequestURI();
        String conPath = request.getContextPath();
        String com = uri.substring(conPath.length());
        System.out.println(uri);
        System.out.println(conPath);
        System.out.println(com);

        if (com.equals("/list.do")) {
            command = new BListCommand();
            command.execute(request, response);
            viewPage = "list.jsp";
        }else if (com.equals("/list2.do")) {
            command = new BListCommand2();
            command.execute(request, response);
            viewPage = "list.jsp";
        }else if (com.equals("/search.do")) {
            command = new BSearchCommand();
            command.execute(request, response);
            viewPage = "search_view.jsp";
        }else if (com.equals("/search2.do")) {
            command = new BSearchCommand2();
            command.execute(request, response);
            viewPage = "search_view.jsp";
        }else if (com.equals("/content_next.do")) {
            String bgroup = request.getParameter("bgroup");
            System.out.println("이동 받았나?" + bgroup);
            command = new BContentNextCommand();
            command.execute(request, response);
            viewPage = "content_view.jsp";
        } else if (com.equals("/content_previous.do")) {
            String bgroup = request.getParameter("bgroup");
            System.out.println("이동 받았나?" + bgroup);
            command = new BContentPreviousCommand();
            command.execute(request, response);
            viewPage = "content_view.jsp";
        } else if (com.equals("/write_view.do")) {
            viewPage = "write_view.jsp";
        } else if (com.equals("/write.do")) {
            command = new BWriteCommand();
            command.execute(request, response);
            viewPage = "list.do";
        }else if (com.equals("/content_view.do")) {
            command = new BContentCommand();
            command.execute(request, response);
            command2 = new BReplyViewCommand();
            command2.execute(request, response);
            viewPage = "content_view.jsp";
        }else if (com.equals("/reply.do")) {
            command = new BReplyCommand();
            command.execute(request, response);
            viewPage = "content_view.do";
        }else if (com.equals("/delete.do")) {
            command = new BDeleteCommand();
            command.execute(request, response);
            viewPage = "list.do";
        } else if (com.equals("/modify_view.do")) {
            command = new BModifyViewCommand();
            command.execute(request, response);
            viewPage = "modify_view.jsp";
        }else if(com.equals("/modify.do")) {
            command = new BModifyCommad();
            command.execute(request, response);
            viewPage = "content_view.do";
        }else if (com.equals("/reply_delete.do")) {
            command = new BReplyDeleteCommand();
            command.execute(request, response);
            viewPage = "content_view.do";
        }

        RequestDispatcher dispatcher = request.getRequestDispatcher(viewPage);
        dispatcher.forward(request, response);
    }

}
```

```java
package edu.kosmo.ex.controller;

import java.io.IOException;

/**
 * Servlet implementation class BController
 */
@WebServlet("*.do")
public class BController extends HttpServlet {
    private static final long serialVersionUID = 1L;

    /**
     * @see HttpServlet#HttpServlet()
     */
    public BController() {
        super();
        // TODO Auto-generated constructor stub
    }

    /**
     * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
     */
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("doGet");
        actionDo(request, response);
    }

    /**
     * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse
     *      response)
     */
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        System.out.println("doPost");
        actionDo(request, response);
```

```java
protected void actionDo(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
    System.out.println("actionDo");
    request.setCharacterEncoding("UTF-8");

    String viewPage = null;
    BCommand command = null;
    BCommand command1 = null;

    String uri = request.getRequestURI();
    String conPath = request.getContextPath();
    String com = uri.substring(conPath.length());
    System.out.println(uri);
    System.out.println(conPath);

    if (com.equals("/list.do")) {
        command = new BListCommand();
        command.execute(request, response);
        viewPage = "list.jsp";
    }else if (com.equals("/search.do")) {
        command = new BSearchCommand();
        command.execute(request, response);
        viewPage = "search_view.jsp";
    }else if (com.equals("/search2.do")) {
        command = new BSearchCommand2();
        command.execute(request, response);
        viewPage = "search_view.jsp";
    }else if (com.equals("/content_next.do")) {
        String bgroup = request.getParameter("bgroup");
        System.out.println("이동 합니다" + bgroup);
        command = new BContentNextCommand();
        command.execute(request, response);
        viewPage = "content_view.jsp";
    } else if (com.equals("/content_previous.do")) {
        String bgroup = request.getParameter("bgroup");
        System.out.println("이동 합니다" + bgroup);
        command = new BContentPreviousCommand();
        command.execute(request, response);
        viewPage = "content_view.jsp";
    } else if (com.equals("/write_view.do")) {
        viewPage = "write_view.jsp";
    } else if (com.equals("/write.do")) {
        command = new BWriteCommand();
        command.execute(request, response);
        viewPage = "list.do";
    }else if (com.equals("/content_view.do")) {
        command = new BContentCommand();
        command.execute(request, response);
        command2 = new BReplyViewCommand();
        command2.execute(request, response);
        viewPage = "content_view.jsp";
    }else if (com.equals("/reply.do")) {
        command = new BReplyCommand();
        command.execute(request, response);
        viewPage = "content_view.do";
    }else if (com.equals("/delete.do")) {
        command = new BDeleteCommand();
        command.execute(request, response);
        viewPage = "list.do";
    } else if (com.equals("/modify_view.do")) {
        command = new BModifyViewCommand();
        command.execute(request, response);
        viewPage = "modify_view.jsp";
    }else if(com.equals("/modify.do")) {
        command = new BModifyCommad();
        command.execute(request, response);
        viewPage = "content_view.do";
    }else if (com.equals("/reply_delete.do")) {
        command = new BReplyDeleteCommand();
        command.execute(request, response);
        viewPage = "content_view.do";
    }

    RequestDispatcher dispatcher = request.getRequestDispatcher(viewPage);
    dispatcher.forward(request, response);
}
```

.do로 타고 들어오면
커맨드 객체를 생성 후
커맨드 객체의 execute 함수를 실행시켜 실행 결과를
viewPage로 던진다.

입력된 viewPage로
forwarding 시킨다.

4. 상세 구현 내용

a. controller

b. command

c. dao

d. dto

e. view

BCommand.java

```
package edu.kosmo.ex.command;

public interface BCommand {
        void execute(HttpServletRequest request, HttpServletResponse response);

}
```

void execute( ) 함수를자손이 구현

BListCommand.java

```
package edu.kosmo.ex.command;

public class BListCommand implements BCommand{
        @Override
        public void execute(HttpServletRequest request, HttpServletResponse response) {
                Criteria cri = new Criteria();
                BDao dao = new BDao();

                dao.getTotalCount();
                int total = dao.getTotalCount();
                ArrayList<BDto> dtos = dao.list(cri);

                request.setAttribute("list", dtos);
                request.setAttribute("pageMaker", new PageVO(cri, total));

        }
}
```

처음 접속 시
게시글 리스트를 보여주는 커맨드

BListCommand2.java

```
package edu.kosmo.ex.command;

public class BListCommand2 implements BCommand{
        @Override
        public void execute(HttpServletRequest request, HttpServletResponse response) {
                int pageNum =Integer.valueOf(request.getParameter("pageNum"));
                int amount = Integer.valueOf(request.getParameter("amount"));

                Criteria cri = new Criteria(pageNum, amount);
                BDao dao = new BDao();
                ArrayList<BDto> dtos = dao.list(cri);

                dao.getTotalCount();
                int total = dao.getTotalCount();

                request.setAttribute("list", dtos);
                request.setAttribute("pageMaker", new PageVO(cri, total));
        }
}
```

페이징된 숫자를 클릭 시
게시글 리스트를 보여주는 커맨드

BCommand.java

```
package edu.kosmo.ex.command;

public interface BCommand {
	void execute(HttpServletRequest request, HttpServletResponse response);

}
```

void execute( ) 함수를자손이 구현

BContentPreviousCommand.java

```
package edu.kosmo.ex.command;

public class BContentPreviousCommand implements BCommand{
	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		String bid = request.getParameter("bid");

		BDao dao = new BDao();
		BDto dto = dao.contentPrevious(bid);

		request.setAttribute("content_view", dto);
	}
}
```

게시글을 이전 게시글로 넘겨주는 커맨드

BContentCommand.java

```
package edu.kosmo.ex.command;

public class BContentCommand implements BCommand{
	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		String bid = request.getParameter("bid");

		BDao dao = new BDao();
		BDto dto = dao.contentView(bid);

		request.setAttribute("content_view", dto);

	}
}
```

게시글을 보여주는 커맨드

BContentNextCommand.java

```
package edu.kosmo.ex.command;


public class BContentNextCommand implements BCommand{
	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		String bid = request.getParameter("bid");

		BDao dao = new BDao();
		BDto dto = dao.contentNext(bid);

		request.setAttribute("content_view", dto);
	}
}
```

게시글을 다음 게시글로 넘겨주는 커맨드

BCommand.java

```java
package edu.kosmo.ex.command;

public interface BCommand {
	void execute(HttpServletRequest request, HttpServletResponse response);

}
```

void execute( ) 함수를자손이 구현

BWriteCommand.java

```java
package edu.kosmo.ex.command;

public class BWriteCommand implements BCommand{
	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		String bid = request.getParameter("bid");
		String bname = request.getParameter("bname");
		String btitle = request.getParameter("btitle");
		String bcontent = request.getParameter("bcontent");
		System.out.println(bid);
		BDao dao = new BDao();
		dao.write(bname, btitle, bcontent);
	}
}
```

작성된 게시글을 저장하는 커맨드

BCommand.java

```java
package edu.kosmo.ex.command;

public interface BCommand {
	void execute(HttpServletRequest request, HttpServletResponse response);

}
```

void execute( ) 함수를자손이 구현

BModifyCommand.java

```java
package edu.kosmo.ex.command;

public class BModifyViewCommand implements BCommand{

	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		String bid = request.getParameter("bid");

		BDao dao = new BDao();
		BDto dto = dao.modifyView(bid);

		request.setAttribute("content_view", dto);
	}

}
```

수정된 게시글을 저장하는 커맨드

BModifyViewCommand.java

```java
package edu.kosmo.ex.command;

public class BModifyViewCommand implements BCommand{

	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		String bid = request.getParameter("bid");

		BDao dao = new BDao();
		BDto dto = dao.modifyView(bid);

		request.setAttribute("content_view", dto);
	}

}
```

게시글 수정화면을 보여주는 커맨드

BCommand.java

```java
package edu.kosmo.ex.command;

public interface BCommand {
	void execute(HttpServletRequest request, HttpServletResponse response);

}
```

void execute( ) 함수를자손이 구현

BDeleteCommad.jav
a

```java
package edu.kosmo.ex.command;

import edu.kosmo.ex.dao.BDao;

public class BDeleteCommand implements BCommand{
	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		String bid = request.getParameter("bid");

		BDao dao = new BDao();
		dao.delete(bid);

	}
}
```

게시글을 삭제하는 커맨드

BCommand.java

```
package edu.kosmo.ex.command;

public interface BCommand {
	void execute(HttpServletRequest request, HttpServletResponse response);

}
```

void execute( ) 함수를 자손의 구현

BReplyViewCommand.java

```
package edu.kosmo.ex.command;

public class BReplyViewCommand implements BCommand{
	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		String bid = request.getParameter("bid");

		BDao dao = new BDao();
		ArrayList<RDto> rtos = dao.reply_view(bid);

		request.setAttribute("reply_view", rtos);

	}
}
```

게시글에 달린 댓글을 보여주는 커맨드

BReplyCommand.java

```
package edu.kosmo.ex.command;

public class BReplyCommand implements BCommand {

	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		String bid = request.getParameter("bid");
		String rname = request.getParameter("rname");
		String rcontent = request.getParameter("rcontent");

		BDao dao = new BDao();
		dao.reply(bid, rname, rcontent);
	}

}
```

게시글에 댓글을 다는 커맨드

BReplyDeleteCommand.java

```
package edu.kosmo.ex.command;

public class BReplyDeleteCommand implements BCommand{
	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		String rid = request.getParameter("rid");

		BDao dao = new BDao();
		dao.reply_delete(rid);
	}
}
```

게시글에 달린 댓글을 삭제하는 커맨드

BCommand.java

```
package edu.kosmo.ex.command;

public interface BCommand {
	void execute(HttpServletRequest request, HttpServletResponse response);

}
```

void execute( ) 함수를자손이 구현

BSearchCommand.java

```
package edu.kosmo.ex.command;

public class BSearchCommand implements BCommand{
	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {

		Criteria cri = new Criteria();
		String btitle = request.getParameter("btitle");

		BDao dao = new BDao();
		ArrayList<BDto> dtos = dao.search(btitle, cri);
		request.setAttribute("search", dtos);

		int total = dao.getTotalCount(btitle);
		request.setAttribute("pageMaker", new PageVO(cri, total));

		BtitleDto title = dao.title(btitle);
		request.setAttribute("btitle", title);
	}
}
```

처음 게시글 검색 시
페이징이 된 화면을 출력하는 커맨

BSearchCommand2.java

```
package edu.kosmo.ex.command;

public class BSearchCommand2 implements BCommand{
	@Override
	public void execute(HttpServletRequest request, HttpServletResponse response) {
		String btitle = request.getParameter("title");
		int pageNum =Integer.valueOf(request.getParameter("pageNum"));
		int amount = Integer.valueOf(request.getParameter("amount"));

		Criteria cri = new Criteria(pageNum, amount);
		BDao dao = new BDao();
		ArrayList<BDto> dtos = dao.search(btitle, cri);
		request.setAttribute("search", dtos);

		int total = dao.getTotalCount(btitle);
		request.setAttribute("pageMaker", new PageVO(cri, total));

		BtitleDto title = dao.title(btitle);
		request.setAttribute("btitle", title);

	}
}
```

게시글 검색 결과 화면에서 페이징 된 숫자를 누르면
해당 화면으로 넘어가게 해 주는 커맨드

4. 상세 구현 내용

a. controller

b. command

c. dao

d. dto

e. view

클래스 변수, 생성자 함수

```
private DataSource dataSource;

public BDao() {
        try {

                Context context = new InitialContext();
                dataSource = (DataSource) context.lookup("java:comp/env/jdbc/oracle");
        } catch (Exception e) {
                e.printStackTrace();
        }

}
```

Context객체와 DataSource객체를 이용해 커넥션 풀을 불러온다.

게시글 리스트 관련 함수

ArrayList<BDto> list()

```
public ArrayList<BDto> list(Criteria cri){
        System.out.println("list(Criteria cri)");
        ArrayList<BDto> dtos = new ArrayList<BDto>();
        Connection connection =null;
        PreparedStatement preparedStatement=null;
        ResultSet rs = null;
        try {

                String query = "select * from("
                                + "select rownum as rnum, a.* from("
                                + "select * from smw_board order by bgroup desc, bstep asc"
                                + ") a where rownum <= ?*?"
                                + ") where rnum >?-1)*?";
                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);
                preparedStatement.setInt(1, cri.getPageNum());
                preparedStatement.setInt(2, cri.getAmount());
                preparedStatement.setInt(3, cri.getPageNum());
                preparedStatement.setInt(4, cri.getAmount());

                rs = preparedStatement.executeQuery();
                while(rs.next()) {
                        int bid = rs.getInt("bid");
                        String bname = rs.getString("bname");
                        String btitle = rs.getString("btitle");
                        String bcontent = rs.getString("bcontent");
                        Timestamp bdate = rs.getTimestamp("bdate");
                        int bhit = rs.getInt("bhit");
                        int bgroup = rs.getInt("bgroup");
                        int bstep = rs.getInt("bstep");
                        int bindent = rs.getInt("bindent");

                        BDto dto = new BDto(bid, bname, btitle, bcontent, bdate, bhit, bgroup, bstep, bindent);
                        dtos.add(dto);
                }

        } catch (Exception e) {
                e.printStackTrace();
        }finally {
                try {
                        if(rs != null) rs.close();
                        if(preparedStatement != null) preparedStatement.close();
                        if(connection != null) connection.close();
                } catch (Exception e2) {
                        e2.printStackTrace();
                }

        }
        return dtos;
}

public BtitleDto title(String btitle) {
        String tit = btitle;
        BtitleDto title = new BtitleDto(tit);
        System.out.println("제목 : " + title);
        System.out.println(title);
        return title;
}
```

페이징 된 게시글 리스트를
출력하는 함수

BDto contentView()

```java
public BDto contentView(String bid){//글 하나만 가져오면 되기 때문에 ArrayList는 필요 없음
		//bhit 업데이트
		System.out.println("contentView()....");
		upHit(bid);
		BDto dto = null;
		Connection connection =null;
		PreparedStatement preparedStatement=null;
		ResultSet rs = null;
		try {
			String query = "select * from mvc_board where bid=?";
			connection = dataSource.getConnection();
			preparedStatement = connection.prepareStatement(query);

			preparedStatement.setInt(1, Integer.valueOf(bid));

			rs = preparedStatement.executeQuery();
			while(rs.next()) {
				int id = rs.getInt("bid");
				String bname = rs.getString("bname");
				String btitle = rs.getString("btitle");
				String bcontent = rs.getString("bcontent");
				bcontent = bcontent.replaceAll([illegible]); // 치환된 작은따옴표 원래 작은따옴표로 원상처리
				bcontent = bcontent.replaceAll("\r\n", "<br>"); // 줄바꿈처리
				bcontent = bcontent.replaceAll("\u0020", " "); // 스페이스바로 띄운 공
				Timestamp bdate = rs.getTimestamp("bdate");
				int bhit = rs.getInt("bhit");
				int bgroup = rs.getInt("bgroup");
				int bstep = rs.getInt("bstep");
				int bindent = rs.getInt("bindent");

				dto = new BDto(id, bname, btitle, bcontent, bdate, bhit, bgroup, bstep, bindent);
			}

		} catch (Exception e) {
			e.printStackTrace();
		}finally {
			try {
				if(rs != null) {rs.close();}
				if(preparedStatement != null) {preparedStatement.close();}
				if(connection != null) {connection.close();}
			} catch (Exception e2) {
				e2.printStackTrace();
			}

		}
		return dto;
	}
```

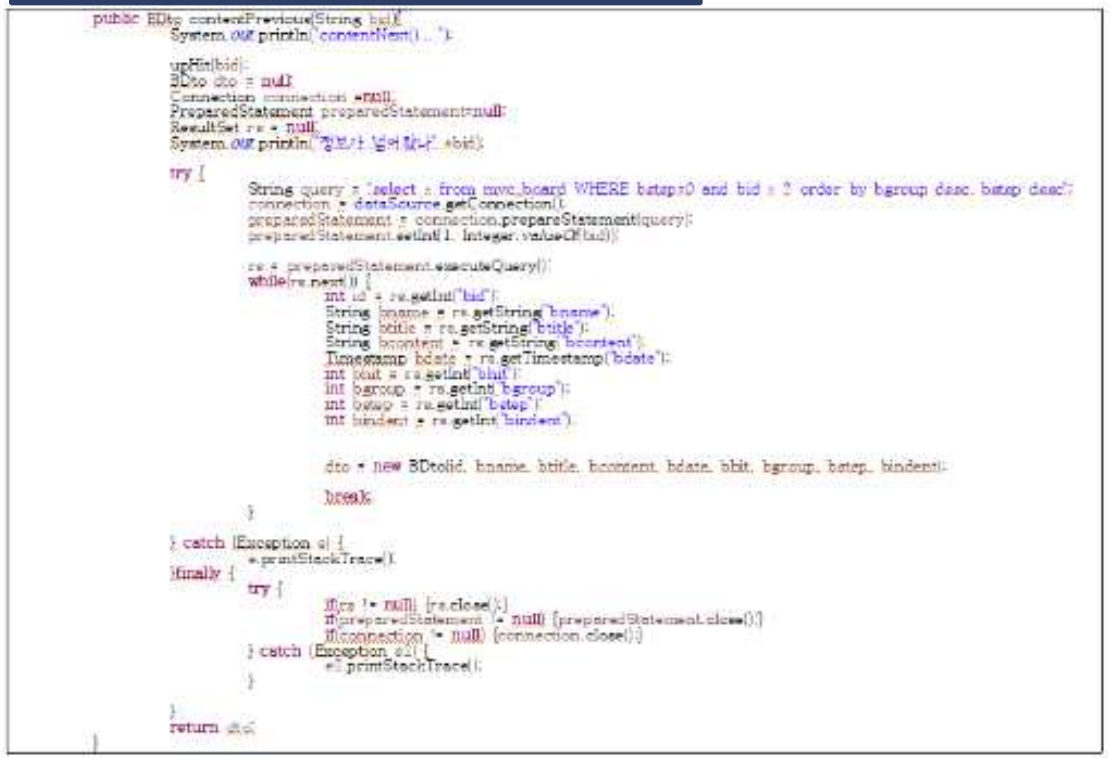
게시글을 보여주는 함수

게시글 출력 관련 함수

BDto contentNext()

다음 게시글로 넘겨주는 함수

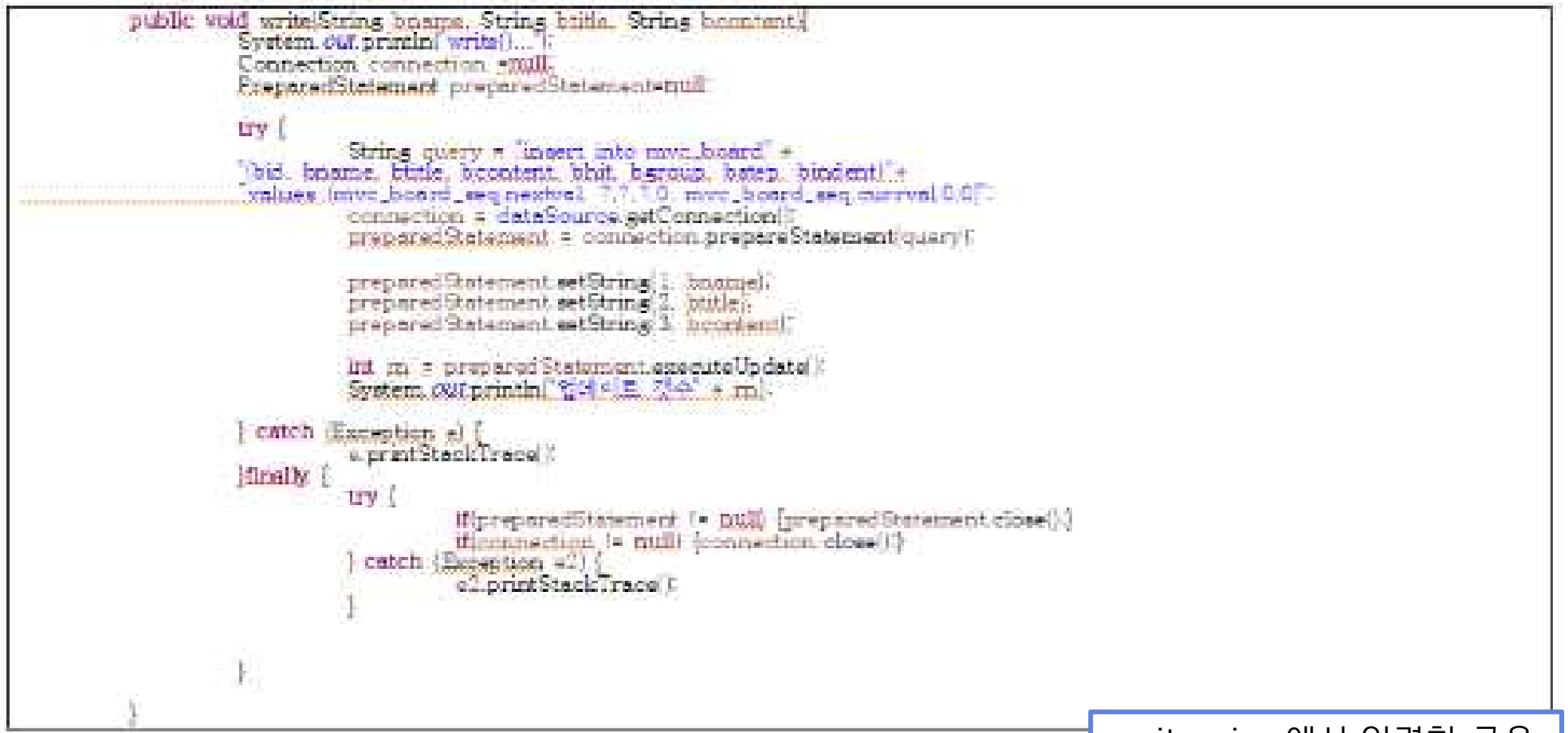
BDto contentPrevious()



이전 게시글로 넘겨주는 함수

게시글 출력 관련 함수

void write()



```
public void write(String bname, String btitle, String bcontent){
        System.out.println("write()....");
        Connection connection = null;
        PreparedStatement preparedStatement=null;

        try {
                String query = "insert into mvc_board" +
        "(bid, bname, btitle, bcontent, bhit, bgroup, bstep, bindent)"+
        "values (mvc_board_seq.nextval, ?,?,?,0, mvc_board_seq.currval,0,0)";
                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);

                preparedStatement.setString(1, bname);
                preparedStatement.setString(2, btitle);
                preparedStatement.setString(3, bcontent);

                int rn = preparedStatement.executeUpdate();
                System.out.println("입력 결과 " + rn);

        } catch (Exception e) {
                e.printStackTrace();
        }finally {
                try {
                        if(preparedStatement != null) preparedStatement.close();
                        if(connection != null) connection.close();
                } catch (Exception e2) {
                        e2.printStackTrace();
                }

        }
}
```

write_view에서 입력한 글을 저장하는 함수

게시글 작성 관련 함수

void delete()

```
public void delete(String bid) {

        System.out.println("delete() ...");
        Connection connection = null;
        PreparedStatement preparedStatement = null;

        try {
                String query = "delete from mvc_board where bid = ?";

                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);
                preparedStatement.setInt(1, Integer.valueOf(bid));

                int rn = preparedStatement.executeUpdate();
                System.out.println("업데이트 갯수 " + rn);

        } catch (Exception e) {
                e.printStackTrace();
        } finally {
                try {
                        if(preparedStatement != null) preparedStatement.close();
                        if(connection != null) connection.close();
                } catch (Exception e2) {
                        e2.printStackTrace();
                }
        }
}
```

게시글된 글을 삭제하는 함수

게시글 삭제 관련 함수

void upHit()

```
public void upHit(String bid) {
        System.out.println("upHit() ...");

        Connection connection = null;
        PreparedStatement preparedStatement = null;

        try {
                String query = "update mvc_board set bhit = bhit+1 where bid=?";

                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);

                preparedStatement.setInt(1, Integer.valueOf(bid));

                int rn = preparedStatement.executeUpdate();
                System.out.println("업데이트 갯수: " + rn);

        } catch (Exception e) {
                e.printStackTrace();
        } finally {
                try {
                        if (preparedStatement != null)
                                preparedStatement.close();
                        if (connection != null)
                                connection.close();
                } catch (Exception e2) {
                        e2.printStackTrace();
                }
        }
}
```

컨텐츠를 눌렀을 때
조회수를 올려주는 함수

조회수 관련 함수

BDto modifyView()

```
public BDto modifyView(String bid){
        System.out.println("modifyView() ...");
        BDto dto = null;
        Connection connection =null;
        PreparedStatement preparedStatement=null;
        ResultSet rs = null;
        try {
                String query = "select * from mvc_board where bid=?";
                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);

                preparedStatement.setInt(1, Integer.valueOf(bid));

                rs = preparedStatement.executeQuery();
                while(rs.next()) {
                        int id = rs.getInt("bid");
                        String bname = rs.getString("bname");
                        String btitle = rs.getString("btitle");
                        String bcontent = rs.getString("bcontent");
                        Timestamp bdate = rs.getTimestamp("bdate");

                        int bhit = rs.getInt("bhit");
                        int bgroup = rs.getInt("bgroup");
                        int bstep = rs.getInt("bstep");
                        int bindent = rs.getInt("bindent");
                        dto = new BDto(id, bname, btitle, bcontent, bdate, bhit, bgroup, bstep, bindent);
                }

        } catch (Exception e) {
                e.printStackTrace();
        }finally {
                try {
                        if(rs != null) rs.close();
                        if(preparedStatement != null) preparedStatement.close();
                        if(connection != null) connection.close();
                } catch (Exception e2) {
                        e2.printStackTrace();
                }

        }
        return dto;
}
```

수정할 게시글을 보여주는 함수

void modify()

```
public void modify(String bid, String bname, String btitle, String bcontent) {

        System.out.println("modify() ...");

        Connection connection = null;
        PreparedStatement preparedStatement = null;

        try {
                String query = "update mvc_board set bname=?, btitle=?, bcontent=? where bid=?";

                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);

                preparedStatement.setString(1, bname);
                preparedStatement.setString(2, btitle);
                preparedStatement.setString(3, bcontent);
                preparedStatement.setInt(4, Integer.valueOf(bid));

                int rn = preparedStatement.executeUpdate();
                System.out.println("수정된 개수: " + rn);

        } catch (Exception e) {
                e.printStackTrace();
        } finally {
                try {
                        if (preparedStatement != null)
                                preparedStatement.close();
                        if (connection != null)
                                connection.close();
                } catch (Exception e2) {
                        e2.printStackTrace();
                }
        }
}
```

게시글을 수정하는 함수

게시글 수정 관련 함수

댓글 관련 함수

ArrayList<RDto> reply_view()

```
public ArrayList<RDto> reply_view(String bid) {
        System.out.println("reply_view() ....");
        ArrayList<RDto> rdto = new ArrayList<RDto>();
        Connection connection = null;
        PreparedStatement preparedStatement = null;
        ResultSet rs = null;
        try {
                String query = "select * from reply_board where bid=? order by rgroup desc, rstep asc";
                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);

                preparedStatement.setInt(1, Integer.valueOf(bid));
                rs = preparedStatement.executeQuery();
                while(rs.next()) {
                        int bid = rs.getInt("bid");
                        int rid = rs.getInt("rid");
                        String rname = rs.getString("rname");
                        String rtitle = rs.getString("rtitle");
                        String rcontent = rs.getString("rcontent");
                        Timestamp rdate = rs.getTimestamp("rdate");
                        int rhit = rs.getInt("rhit");
                        int rgroup = rs.getInt("rgroup");
                        int rstep = rs.getInt("rstep");
                        int rindent = rs.getInt("rindent");

                        RDto rto = new RDto(bid, rid, rname, rtitle, rcontent, rdate, rhit, rgroup, rstep, rindent);
                        rdto.add(rto);
                }
        } catch (Exception e) {
                e.printStackTrace();
        }finally {
                try {
                        if(rs != null) {rs.close();}
                        if(preparedStatement != null) {preparedStatement.close();}
                        if(connection != null) {connection.close();}
                } catch (Exception e2) {
                        e2.printStackTrace();
                }
        }

        return rdto;
}
```

입력된 댓글을 보여주는 함수

void reply()

```
public void reply(String bid, String rname, String rcontent) {

        System.out.println("reply() ...");
        Connection connection = null;
        PreparedStatement preparedStatement=null;

        try {
                String query = "insert into reply_board (bid, rid, rname, rcontent, rgroup, rstep, rindent)"+
                                "values (?, reply_board_seq.nextval, ?,?,reply_board_seq.currval,0,0)";
                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);

                preparedStatement.setInt(1, Integer.parseInt(bid));
                preparedStatement.setString(2, rname);
                preparedStatement.setString(3, rcontent);

                int rn = preparedStatement.executeUpdate();
                System.out.println("입력이된 갯수" + rn);

        } catch (Exception e) {
                e.printStackTrace();
        }finally {
                try {
                        if(preparedStatement != null) {preparedStatement.close();}
                        if(connection != null) {connection.close();}
                } catch (Exception e2) {
                        e2.printStackTrace();
                }
        }
}
```

작성한 댓글을 저장하는 함수

void reply_delete()

```
public void reply_delete(String rid) {

        System.out.println("reply_delete() ...");
        Connection connection = null;
        PreparedStatement preparedStatement = null;

        try {
                String query = "delete from reply_board where rid = ?";

                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);
                preparedStatement.setInt(1, Integer.valueOf(rid));

                int rn = preparedStatement.executeUpdate();
                System.out.println("입력이된 갯수" + rn);

        } catch (Exception e) {
                e.printStackTrace();
        } finally {
                try {
                        if(preparedStatement != null) preparedStatement.close();
                        if(connection != null) connection.close();
                } catch (Exception e2) {
                        e2.printStackTrace();
                }
        }
}
```

댓글을 삭제하는 함수

ArrayList<BDto> search()

```
public ArrayList<BDto> search(String btitle, Criteria cri){
        System.out.println("Search()");
        ArrayList<BDto> dtos = new ArrayList<BDto>();
        Connection connection = null;
        PreparedStatement preparedStatement=null;
        ResultSet rs = null;
        try {
                System.out.println(btitle);
                String keyword = "%" + btitle + "%";
                String query = "select *from("
                                + "select rownum rn rnum, a.*from("
                                + "select *from mvc_board WHERE btitle like ? order by bgroup desc, bstep asc"
                                + ")a where rownum<=? *?"
                                + ")where rnum>(?-1)*?";
                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);

                preparedStatement.setString(1, keyword);
                preparedStatement.setInt(2, cri.getPageNum());
                preparedStatement.setInt(3, cri.getAmount());
                preparedStatement.setInt(4, cri.getPageNum());
                preparedStatement.setInt(5, cri.getAmount());
                rs = preparedStatement.executeQuery();
                while(rs.next()) {
                        int id = rs.getInt("bid");
                        String bname = rs.getString("bname");
                        String title = rs.getString("btitle");
                        String bcontent = rs.getString("bcontent");
                        Timestamp bdate = rs.getTimestamp("bdate");
                        int bhit = rs.getInt("bhit");
                        int bgroup = rs.getInt("bgroup");
                        int bstep = rs.getInt("bstep");
                        int bindent = rs.getInt("bindent");

                        BDto dto = new BDto(id, bname, title, bcontent, bdate, bhit, bgroup, bstep, bindent);
                        dtos.add(dto);
                }
        } catch (Exception e) {
                e.printStackTrace();
        }finally {
                try {
                        if(rs != null) {rs.close();}
                        if(preparedStatement != null) {preparedStatement.close();}
                        if(connection != null) {connection.close();}
                } catch (Exception e2) {
                        e2.printStackTrace();
                }
        }
        return dtos;
}
```

검색한 키워드가 포함된
제목의 게시글을 찾아주는 함수

BtitleDto title()

```
public BtitleDto title(String btitle) {
        String tit = btitle;
        BtitleDto title = new BtitleDto(tit);
        System.out.println("여기서 title()... 이다.");
        System.out.println(title);
        return title;
}
```

게시글 검색 후 페이징 숫자를 눌렀을 때
BSearchCommand에서
Get방식으로 받은 검색 키워드를
BSearchCommand2로 넘겨주는 함수

검색 관련 함수

int getTotalCount()

```
public int getTotalCount() {
        System.out.println("getTotalCount()");
        Connection connection = null;
        PreparedStatement preparedStatement=null;
        ResultSet rs = null;
        int total = 0;
        try {
                String query = "select count(*) from mvc_board";
                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);
                rs = preparedStatement.executeQuery();
                while(rs.next()) {
                        total = rs.getInt(1);
                }
        } catch (Exception e) {
                e.printStackTrace();
        }finally {
                try {
                        if(rs != null) {rs.close();}
                        if(preparedStatement != null) {preparedStatement.close();}
                        if(connection != null) {connection.close();}
                } catch (Exception e2) {
                        e2.printStackTrace();
                }
        }
        return total;
}
```

게시글 리스트 출력 시 페이징을 위해 전체 게시글의 갯수를 출력하는 함수

int getTotalCount()

```
public int getTotalCount(String btitle) {
        System.out.println("getTotalSearchCount()");
        Connection connection = null;
        PreparedStatement preparedStatement=null;
        ResultSet rs = null;
        String keyword = "%" + btitle + "%";
        int total = 0;
        try {
                String query = "select count(*) from mvc_board WHERE btitle like ? order by bgroup desc, bstep asc";
                connection = dataSource.getConnection();
                preparedStatement = connection.prepareStatement(query);
                preparedStatement.setString(1, keyword);
                rs = preparedStatement.executeQuery();
                while(rs.next()) {
                        total = rs.getInt(1);
                }
        } catch (Exception e) {
                e.printStackTrace();
        }finally {
                try {
                        if(rs != null) {rs.close();}
                        if(preparedStatement != null) {preparedStatement.close();}
                        if(connection != null) {connection.close();}
                } catch (Exception e2) {
                        e2.printStackTrace();
                }
        }
        return total;
}
```

특정 키워드로 검색된 게시글 리스트 출력 시 페이징을 위해 검색된 전체 게시글의 갯수를 출력하는 함수

페이징처리 관련 함수

4. 상세 구현 내용

a. controller

b. command

c. dao

d. dto

e. view

public class Criteria

```
package edu.kosmo.ex.page;

public class Criteria {

	private int pageNum;
	private int amount;

	public Criteria(){
		this(1,10);
	}
	public Criteria(int pageNum, int amount) {
		this.pageNum = pageNum;
		this.amount = amount;
	}
	public int getPageNum() {
		return pageNum;
	}
	public void setPageNum(int pageNum) {
		this.pageNum = pageNum;
	}
	public int getAmount() {
		return amount;
	}
	public void setAmount(int amount) {
		this.amount = amount ;
	}
	@Override
	public String toString() {
		return "Criteria [pageNum=" + pageNum + ", amount=" + amount + "]";
	}

}
```

게시글 리스트 출력 화면에서 페이지를 넘어갈 때 '페이지'와 '한 페이지에 들어갈 게시글의 총수'를 받아 PageVO에 넘길 수 있도록 정보를 조합하는 클래스

public class PageVO

```
package edu.kosmo.ex.page;

public class PageVO {
	private int startPage;
	private int endPage;
	private boolean pre, next;
	private int total;
	private Criteria cri;

	public PageVO(Criteria cri, int total) {
		this.cri = cri;
		this.total = total;
		this.endPage = (int) (Math.ceil(cri.getPageNum() / 10.0)) * 10;
		this.startPage = this.endPage - 9;
		int realEnd = (int) (Math.ceil((total * 1.0) / cri.getAmount()));
		if (realEnd <= this.endPage) {
			this.endPage = realEnd;
		}
		this.pre = this.startPage > 1;
		this.next = this.endPage < realEnd;

	}
	public String makeQuery(int page) {
		System.out.println("makeQuery()");
		return "?pageNum=" + page + "&amount=" + cri.getAmount();
	}

	public PageVO() { }

	public int getStartPage() { return startPage; }

	public void setStartPage(int startPage) { this.startPage = startPage; }

	public int getEndPage() { return endPage; }

	public void setEndPage(int endPage) { this.endPage = endPage; }

	public boolean isPre() { return pre; }

	public void setPre(boolean pre) { this.pre = pre; }

	public boolean isNext() { return next; }

	public void setNext(boolean next) { this.next = next; }

	public int getTotal() { return total; }

	public void setTotal(int total) { this.total = total; }

	public Criteria getCri() { return cri; }

	public void setCri(Criteria cri) { this.cri = cri; }
}
```

페이징에 필요한 변수들을 설정하는 클래스

public class BDto

```
package edu.koemo.mvc.dto;

public class BDto {
	private int rownum;
	private int bid;
	private String bname;
	private String btitle;
	private String bcontent;
	private Timestamp bdate;
	private int bhit;
	private int bgroup;
	private int bstep;
	private int bindent;
	public BDto() {

	}
	public BDto(int bid, String bname, String btitle, String bcontent, Timestamp bdate, int bhit, int bgroup, int bstep,
			int bindent) {
		this.bid = bid;
		this.bname = bname;
		this.btitle = btitle;
		this.bcontent = bcontent;
		this.bdate = bdate;
		this.bhit = bhit;
		this.bgroup = bgroup;
		this.bstep = bstep;
		this.bindent = bindent;
	}

	public int getBid() { return bid; }
	public void setBid(int bid) { this.bid = bid; }
	public String getBname() { return bname; }
	public void setBname(String bname) { this.bname = bname; }
	public String getBtitle() { return btitle; }
	public void setBtitle(String btitle) { this.btitle = btitle; }
	public String getBcontent() { return bcontent; }
	public void setBcontent(String bcontent) { this.bcontent = bcontent; }
	public Timestamp getBdate() { return bdate; }
	public void setBdate(Timestamp bdate) { this.bdate = bdate; }
	public int getBhit() { return bhit; }
	public void setBhit(int bhit) { this.bhit = bhit; }
	public int getBgroup() { return bgroup; }
	public void setBgroup(int bgroup) { this.bgroup = bgroup; }
	public int getBstep() { return bstep; }
	public void setBstep(int bstep) { this.bstep = bstep; }
	public int getBindent() { return bindent; }
	public void setBindent(int bindent) { this.bindent = bindent; }
}
```

DataBase에서
MVC_BOARD의 프로세스 간에 데이터를
전달하는 클래스

public class RDto

```
package edu.kosmo.ex.dto;

public class RDto {
	private int bid;
	private int rid;
	private String rname;
	private String rtitle;
	private String rcontent;
	private Timestamp rdate;
	private int rhit;
	private int rgroup;
	private int rstep;
	private int rindent;

	public RDto() { }

	public RDto(int bid, int rid, String rname, String rtitle, String rcontent, Timestamp rdate, int rhit, int rgroup,
			int rstep, int rindent) {
		this.bid = bid;
		this.rid = rid;
		this.rname = rname;
		this.rtitle = rtitle;
		this.rcontent = rcontent;
		this.rdate = rdate;
		this.rhit = rhit;
		this.rgroup = rgroup;
		this.rstep = rstep;
		this.rindent = rindent;
	}

	public int getBid() { return bid; }
	public void setBid(int bid) { this.bid = bid; }
	public int getRid() { return rid; }
	public void setRid(int rid) { this.rid = rid; }
	public String getRname() { return rname; }
	public void setRname(String rname) { this.rname = rname; }
	public String getRtitle() { return rtitle; }
	public void setRtitle(String rtitle) { this.rtitle = rtitle; }
	public String getRcontent() { return rcontent; }
	public void setRcontent(String rcontent) { this.rcontent = rcontent; }
	public Timestamp getRdate() { return rdate; }
	public void setRdate(Timestamp rdate) { this.rdate = rdate; }
	public int getRhit() { return rhit; }
	public void setRhit(int rhit) { this.rhit = rhit; }
	public int getRgroup() { return rgroup; }
	public void setRgroup(int rgroup) { this.rgroup = rgroup; }
	public int getRstep() { return rstep; }
	public void setRstep(int rstep) { this.rstep = rstep; }
	public int getRindent() { return rindent; }
	public void setRindent(int rindent) { this.rindent = rindent; }
}
```

DataBase에서
REPLY_BOARD의 변수를 불러와서 사용할 때
잠시 저장해두는 클래스

public class BtitleDto

```
package edu.kosmo.ex.dto;

public class BtitleDto {
	String title;

	public BtitleDto() {

	}

	public BtitleDto(String title) {
		this.title = title;
	}

	public String getTitle() {
		return title;
	}

	public void setTitle(String title) {
		this.title = title;
	}
}
```

게시글 검색 시
페이징 숫자를 눌렀을 때
Get방식으로 검색 키워드를 넘기기 위해
키워드를 저장하는 클래스

4. 상세 구현 내용

a. controller

b. command

c. dao

d. dto

e. view

```jsp
<%@page import="java.io.PrintWriter"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!doctype html>
<html lang="en">
<head>
<!-- Required meta tags -->
<meta charset="utf-8">
<meta name="viewport"
    content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->
<link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
    integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
    crossorigin="anonymous">

<title>list</title>
<style>
@import url('https://fonts.googleapis.com/css2?family=Hi+Melody&display=swap');

    body {
        width: 80%;
        margin: 0 auto;
        font-family: 'Hi Melody', cursive;

    }

    div {
        width: 100%;
        margin: 0 0 0 0;
        text-align: right;
        margin-bottom: 10px;
    }

    table {
        width: 100%;
        text-align: center;
        font-family: 'Hi Melody', cursive;
    }
</style>
</head>
<body>
    <div style="height: 150px; font-size: 3em; line-height: 150px; text-align: center;">
        게시판입니다.
    </div>
    <table style="margin: 0 0 5px 0">
        <tr>
            <td style="text-align: left;">
                <form action="search.do">
                    제목 <input type="text" name="btitle" size="50"></input>
                        <input type="submit" value="검색" />
                </form>
            </td>
            <td style="text-align: right;">
                <a href="write_view.do" style="text-align: right;">글작성</a>
            </td>
        </tr>
    </table>
    <table class="table table-hover">
        <thead>
            <tr>
                <td style="width: 10%; background-color: aliceblue;">번호</td>
                <td style="width: 45%; background-color: aliceblue;">제목</td>
                <td style="width: 15%; background-color: aliceblue;">글쓴이</td>
                <td style="width: 20%; background-color: aliceblue;">날짜</td>
                <td style="width: 10%; background-color: aliceblue;">조회수</td>
            </tr>
        </thead>
        <c:forEach var="dto" items="${list}">
            <tr>
                    <!-- 출처: https://baessi.tistory.com/144 -->
                    <th scope="row">${dto.bid}</th>
                    <td>
                        <a href="content_view.do?bid=${dto.bid}">
                            ${dto.btitle}
                            <c:if test="${dto.bhit > 10}">
                                <span class="badge badge-pill badge-danger">Hot</span>
                            </c:if>
                        </a>
                    </td>
                    <td>${dto.bname}</td>
                    <td>${dto.bdate}</td>
                    <td>${dto.bhit}</td>
            </tr>
        </c:forEach>
    </table>
```

게시글 리스트를 출력하는 부분

list.jsp

```
91  <!-- 링크를 걸어주다 1-10페이지까지 페이지를 만들어준느낌 -->
92      <div class="btn-group me-2" role="group" aria-label="First group" style="height: 30px">
93          <c:if test="${pageMaker.pre}">
94              <a href="list2.do${pageMaker.makeQuery(pageMaker.startPage - 1) }">
95              <button type="button" class="btn btn-secondary">«</button>
96              </a>
97          </c:if>
98
99          <c:forEach var="idx" begin="${pageMaker.startPage }" end="${pageMaker.endPage }">
100             <a href="list2.do${pageMaker.makeQuery(idx)}">
101             <button type="button" class="btn btn-outline-secondary">${idx}</button></a>
102         </c:forEach>
103
104         <c:if test="${pageMaker.next && pageMaker.endPage > 0}">
105             <a href="list2.do${pageMaker.makeQuery(pageMaker.endPage +1)}">
106             <button type="button" class="btn btn-secondary"> » </button></a>
107         </c:if>
108     </div>
109 <br>
110 <hr>
111 <div style="height: 100px; text-align: center;">
112     Copyright © 2021-2022. musicians_Child. All right reserved.
113 </div>  <!-- Optional JavaScript; choose one of the two! -->
```

게시글 리스트를 페이징 처리하는 부분

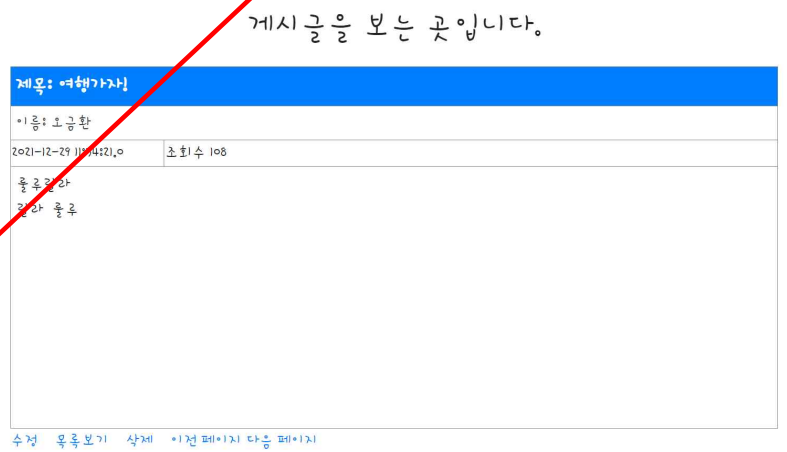
1028	사랑합니다.	사랑합니다.	2021-12-28 01:32:05.0	4
1018	테스트 Hot	test	2021-12-27 11:17:17.0	20
1027	테스트	test	2021-12-27 18:11:37.0	1
1016	테스트	test	2021-12-27 11:17:17.0	0
1015	테스트	test	2021-12-27 11:17:17.0	0

1 2 3 4 5 6 7 8 9 10 »

list.jsp

게시글을 보여주는 부분

수정, 목록보기, 삭제, 이전페이지, 다음페이지



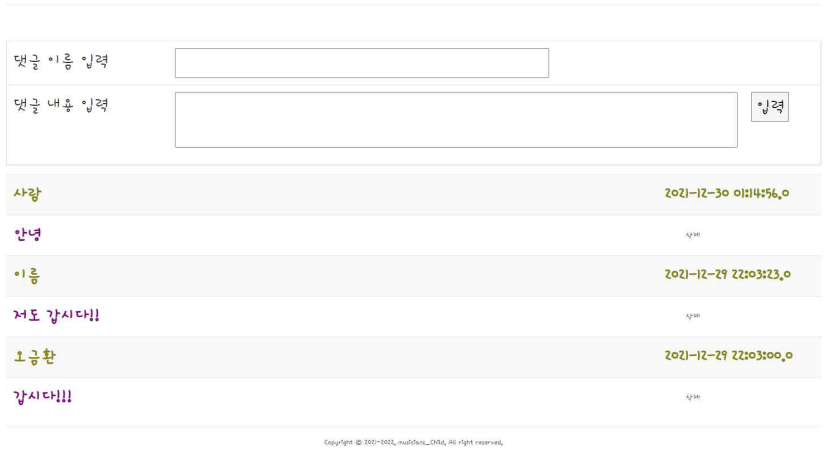
content_view.jsp

```
<table class="table" id="table2" width="80%">
    <form action="reply.do?bid=${content_view.bid}" method="post">

        <tr style="border: 1px solid #cccccc;">
            <td>댓글 이름 입력</td>
            <td><input type="text" name="rname" size="50"></td>
            <td></td>
        </tr>
        <tr style="border: 1px solid #cccccc;">
            <td>댓글 내용 입력</td>
            <td><textarea style="resize: none; width: 100%; height: 100px; name="rcontent"></textarea></td>
            <td><input type="submit" value="입력"></td>
        </tr>
        <br><br>
    </form>
</table>
<table class="table table-striped">
    <c:forEach var="rdto" items="${reply_view}">
        <tr style="height: 10px; font-weight: 900; color: olive;">
            <td style="text-align: left;" colspan="2">${rdto.rname}</td>
        </tr>
        <tr style="padding-top: 10px; padding-bottom: 10px; height: 50px; color:purple; ; font-weight: 600; ;">
            <td style="text-align: left; background-color: white; width: 80%">${rdto.rcontent}</td>
            <td>
                <button type="button" class="btn btn-warn">
                    <a class="dropdown-item" href="reply_delete.do?rid=${rdto.rid}&bid=${rdto.bid}">삭제</a>
                </button>
            </td>
        </tr>
    </c:forEach>
</table>
```

게시글에 대해 댓글을 입력하는 부분

댓글을 출력하는 부분



content_view.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<!doctype html>
<html lang="en">
    <head>
        <!-- Required meta tags -->
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

        <!-- Bootstrap CSS -->
        <link
            rel="stylesheet"
            href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
            integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
            crossorigin="anonymous">

        <title>write_view</title>
        <style>
        @import url('https://fonts.googleapis.com/css2?family=Hi+Melody&display=swap');

            body {
                margin: 0 auto;
                width: 90%;
                font-family: 'Hi Melody', cursive;

            }
            table {
                width: 100%;
                margin: 0 0 0 0;
            }
            .input-group {

                height: 500px;
            }
            textarea {
                height: 90%;
                resize: none;
            }
        </style>
    </head>
```

```jsp
<body>
    <div style="height: 100px; font-size: 3em; line-height: 150px; text-align: center;">
        글쓰는 곳입니다.
    </div>
<table>
    <form action="write.do?" method="post">
        <tr>
            <td style="border: 1px solid #cccccc; width: 8%; background-color: #cccccc;border: 2px solid white;text-align: center;">
                제목
            </td>
            <td >
                <input type="text" name="btitle" size="200%">
            </td>

        </tr>
        <tr>
            <td style="border: 2px solid #cccccc; width: 8%; background-color: #cccccc; border: 2px solid white; text-align: center;"
                이름
            </td>
            <td >
                <input type="text" name="bname" size="200%">
            </td>                </tr>
        <tr>
            <td style="border: 2px solid #cccccc; width: 8%; background-color: #cccccc;border: 2px solid white;text-align: center;">
                내용
            </td>
            <td>
                <div class="input-group">
                    <textarea class="form-control" aria-label="With textarea" name="bcontent"></textarea>
                </div>
            </td>
        </tr>
        <tr style="text-align: right;">
            <td colspan="2"> <input type="submit" value="입력">    <a href="list.do">목록보기</a></td>
        </tr>
    </form>
</table>
<div style="height: 100px; text-align: center;">
Copyright © 2021-2022. musicians_child. all right reserved.
</div>
```

게시글을 입력하는 부분

글쓰는 곳입니다.

제목	
이름	
내용	

입력 목록보기

Copyright © 2021-2022. musicians_child. All right reserved.

write_view.jsp

```jsp
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!doctype html>
<html lang="en">
  <head>
    <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <!-- Bootstrap CSS -->
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
    integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2" crossorigin="anonymous">

    <title>modify_view</title>
    <style>
            @import url('https://fonts.googleapis.com/css2?family=Hi+Melody&display=swap');

            body {
                margin: 0 auto;
                width: 90%;
                font-family: 'Hi Melody', cursive;

            }
            table {
                width: 100%;
                margin: 0 0 0 0;
            }
            .input-group {

                height: 500px;
            }
            .form-control{
                height: 500px;
                resize: none;
            }
    </style>
  </head>
```

게시글을 수정하는 부분

```jsp
<body>
  <div style="height: 150px; font-size: 3em; line-height: 150px; text-align: center;">
      수정합니다.
  </div>
  <table>
      <form action="modify.do" method="post">
          <input type="hidden" name="bid" value="${content_view.bid}">

          <tr>
              <td style="border: 1px solid #cccccc; width: 8%; background-color: #cccccc;border: 2px solid white;text-align: center;">
                  제목
              </td>
              <td> <input type="text" name="btitle" value="${content_view.btitle}" size="120%"></td>
          </tr>
          <tr>
              <td style="border: 1px solid #cccccc; width: 8%; background-color: #cccccc; border: 2px solid white;text-align: center;">
                  이름
              </td>
              <td>
                  <input type="text" name="bname" value="${content_view.bname}" size="120%">
              </td>
          </tr>
          <tr>
              <td style="border: 1px solid #cccccc; width: 8%; background-color: #cccccc;border: 2px solid white;text-align: center;">
                  내용
              </td>
              <td>
                  <div class="input-group">
                      <textarea class="form-control" aria-label="with textarea" name="bcontent" >${content_view.bcontent}</textarea>
                  </div>
              </td>
          </tr>
          <tr style="text-align: right;">
              <td colspan="2"> <input type="submit" value="수정">    <a href="list.do">목록보기</a>   </td>
          </tr>
      </form>
  </table>
  <div style="height: 100px; text-align: center;">
      copyright © 2021-2022. musicians_child. All right reserved.
  </div>
```

수정합니다.

제목	여행가자!
이름	고경환
내용	좋은장소 장소 좋죠!

수정 목록보기

Copyright © 2021-2022. musicians_child. All right reserved.

modify_view.jsp

```jsp
<%@page import="java.io.Writer"%>
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!doctype html>
<html lang="en">
<head>
<!-- Required meta tags -->
<meta charset="utf-8">
<meta name="viewport"
    content="width=device-width, initial-scale=1, shrink-to-fit=no">

<!-- Bootstrap CSS -->
<link rel="stylesheet"
    href="https://cdn.jsdelivr.net/npm/bootstrap@4.5.3/dist/css/bootstrap.min.css"
    integrity="sha384-TX8t27EcRE3e/ihU7zmQxVncDAy5uIKz4rEkgIXeMed4M0jlfIDPvg6uqKI2xXr2"
    crossorigin="anonymous">

<title>search_view</title>
<style>
@import url('https://fonts.googleapis.com/css2?family=Hi+Melody&display=swap');

    body {
        width: 80%;
        margin: 0 auto;
        font-family: 'Hi Melody', cursive;

    }

    div {
        width: 100%;
        margin: 0 0 0 0;
        text-align: right;
        margin-bottom: 10px;
    }

    table {
        width: 100%;
        text-align:
        font-family:

    }
</style>
</head>
```

```jsp
<body>

    <div style="height: 150px; font-size: 3em; line-height: 150px; text-align: center;">
        제목 검색 결과입니다.
    </div>
    <div>
        <a href="write_view.do" style="text-align: right;">글작성</a>
    </div>
    <table border="1" class="table table-hover">
        <thead>
            <tr>
                <td style="width: 10%; background-color: aliceblue;">번호</td>
                <td style="width: 45%; background-color: aliceblue;">제목</td>
                <td style="width: 15%; background-color: aliceblue;">글쓴이</td>
                <td style="width: 20%; background-color: aliceblue;">날짜</td>
                <td style="width: 10%; background-color: aliceblue;">조회수</td>
            </tr>
        </thead>
        <c:forEach var="dto" items="${search}">
            <tr style="border: 0.5px solid #cccccc">
                <!-- 출처: https://baessi.tistory.com/144 -->
                <td>${dto.bid}</td>
                <td>
                    <a href="content_view.do?bid=${dto.bid}">
                        ${dto.btitle}
                        <c:if test="${dto.bhit > 10}">
                            <span class="badge badge-pill badge-danger">Hot</span>
                        </c:if>
                    </a>
                </td>
                <td>${dto.bname}</td>
                <td>${dto.bdate}</td>
                <td>${dto.bhit}</td>
            </tr>
        </c:forEach>
</table>
```

검색한 게시글 리스트를 출력하는 부분

localhost:8282/jsp_board/search.do?btitle=여행

제목 검색 결과입니다.

번호	제목	글쓴이	날짜	조회수
1032	여행가자!! Hot	오금환	2021-12-29 11:54:21.0	107
1031	여행가자!!	오금환	2021-12-29 11:53:42.0	2
1030	여행가자!!	오금환	2021-12-29 11:53:13.0	2
1029	여행가자!!	오금환	2021-12-29 11:50:38.0	1

search_view.jsp

```
<div class="btn-group me-2" role="group" aria-label="First group" style="height: 20px">
        <c:if test="${pageMaker.pre}">
            <a href="search2.do${pageMaker.makeQuery(pageMaker.startPage - 1) }&title=${btitle.title}">
            <button type="button" class="btn btn-secondary">«</button>
            </a>
        </c:if>
        <c:forEach var="idx" begin="${pageMaker.startPage }" end="${pageMaker.endPage }">
                <a href="search2.do${pageMaker.makeQuery(idx)}&title=${btitle.title}">
                <button type="button" class="btn btn-outline-secondary">${idx}</button></a>
        </c:forEach>
        <c:if test="${pageMaker.next && pageMaker.endPage > 0}">
            <a href="search2.do${pageMaker.makeQuery(pageMaker.endPage +1)}&title=${btitle.title}">
            <button type="button" class="btn btn-secondary"> » </button></a>
        </c:if>
    </div>
    <br>
 <div>   <a href="list.do">목록보기</a>
 </div>
 <hr>
 <div style="height: 100px; text-align: center;">
    Copyright © 2021-2022. musicians_Child. All right reserved.
</div>
```

검색한 게시글 리스트의 페이징 처리

search_view.jsp

5. 후기 및 보완점

후기

- 게시판이 어떤 경로로 정보를 옮기고 저장하지를 이해할 수 있었다.

- 데이터 베이스의 활용의 중요성을 인지할 수 있었다.

보완점

- ctrl C, ctrl V 를 활용을 높여, 코드 작성의 정확성을 높여야 겠다.
- 데이터 베이스를 익숙하게 다루기 위한 연습이 필요하다.
- 게시판 구현을 2시간 안에 끝낼 수 있도록 연습이 필요하다.