[[1]]

CNT는 BoardVO에서 관리하지만 테이블 생성시에는 없는 변수이다.
왜냐하면 단순 JAVA에서 로직으로만 활용할 기능이기 때문에 모든 게시글(Board 객체)들이 가질 필요가 없기 때문

[MainAction]

```
boolean flag = true;
int cnt;
String paramCnt=request.getParameter("cnt");
if(paramCnt==null || paramCnt.equals("")){
// index에서 처음 실행 시 or 파라미터에 cnt 값이 없을 시 초기 값을 2로 설정한다.
            cnt=2;
            bvo.setCnt(cnt);
        }
        else { // 받아온 값이 있을 경우에는 받아온 값을 cnt에 세팅한다.
            cnt=Integer.parseInt(paramCnt);
            bvo.setCnt(Integer.parseInt(paramCnt));
            //null이 아닌 값이면 해당 값을 set한다.
        }
        //System.out.println("if문 전: "+paramMid);
        if(paramMid == null) {
            bvo.setMid("");
            // mid에 해당하는 데이터가 없는 경우 = 검색에 mid가 없는 경우 ==> 전체보기
        }else {
            bvo.setMid(paramMid);
            // mid에 해당하는 데이터가 있는 경우 = 검색에 mid로 검색 ==> 해당 mid로 검색하기
        }
        //System.out.println("if문 후"+bvo);

        ArrayList<BoardSet> datas=bdao.selectAll(bvo); // cnt 만큼 게시글 조회
        bvo.setCnt(cnt+1);
        ArrayList<BoardSet> datasNext=bdao.selectAll(bvo); // cnt + 1 만큼 게시글 조회

        if(datas.size() == datasNext.size()) { // cnt만큼 조회한 게시글 사이즈 == cnt +1 만큼 조회한 게시글 사이즈
                        flag = false;
        }
        request.setAttribute("more", flag);

        ArrayList<MemberVO> member = mdao.selectAll(mvo); // 최근 가입한 3명 출력용

        request.setAttribute("boardMidCheck", bvo.getMid());
        request.setAttribute("cnt", bvo.getCnt());
```

[main]

```
            <c:if test="${boardMidCheck.equals('')}"> // 전체보기일때 더보기 기능
                <c:if test="${more==true}"> // 더 보기할 데이터가 없을때 더보기 기능 비활성화
                    <a href="main.do?cnt=${cnt+2}">더보기&gt;&gt;</a>
                </c:if>
            </c:if>

            <c:if test="${!boardMidCheck.equals('')}"> // 특정회원이 작성한 글 검색시 더보기 기능
```

```
            <c:if test="${more==true}}"> // 더 보기할 데이터가 없을때 더보기 기능 비활성화
                <a href="main.do?mid=${b.mid}&cnt=${cnt+2}">더보기&gt;&gt;</a>
                                        // 더보기를 눌렀을 때에도 mid(검색한 특정 회원)가 계속 유지되도
록(세션유지)
            </c:if>
        </c:if>
```

[[2]]
[web.xml]

```
    <context-param>
        <param-name>encoding</param-name>
        <param-value>UTF-8</param-value>
    </context-param>
```

[EncFilter]

```
@WebFilter({"*.do" , "*.jsp"}) // .do로 실행하는 모든것들에 EncFilter.java의 내용 적용
public class EncFilter extends HttpFilter implements Filter {

    private String encoding; // 변수 설정

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

    /
     * @see Filter#destroy()
     */
    public void destroy() {
        // TODO Auto-generated method stub
    }

    /
     * @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)
     */
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws
IOException, ServletException {
            // Parameter로 ServletRequest, ServletResponse를 가지며, FilterChain에 대한 참조도 포함한다.
        request.setCharacterEncoding(this.encoding);
        response.setCharacterEncoding(this.encoding);

        // pass the request along the filter chain
        chain.doFilter(request, response);
    }
            // doFilter()는 EncFilter가 적용되는 모든 상황 = *.do를 실행했을 때 적용되는 메서드

    /
     * @see Filter#init(FilterConfig)
     */
```

```
    public void init(FilterConfig fConfig) throws ServletException {
        this.encoding=fConfig.getServletContext().getInitParameter("encoding");
            //==web.xml에서 encoding의 value
    }

            // 초기화 파라미터 에서 인코딩 정보 가져와서 설정.
            // Filter 초기화로 한 번만 호출한다. 여기서는 web.xml의 초기화 매개변수에
            // 'encoding'으로 설정된 값을 문자열 변수 encoding에 저장한다.
            // 이후 실제 필터 적용은 doFilter()에 의해 이루어진다.

}
```

서블릿에서 PrintWriter와 같은 클래스을 통해 한글을 입력하여 jsp 파일로 내보내는 경우,
그리고 폼 태그에서 입력한 값이 서블릿으로 넘어올 경우에 인코딩 설정이 되어 있지 않으면 한글이 정상적으로 출력되지 않습니다.

아래와 같이 서블릿의 메소드 시작 부분에 인코딩 설정을 해주면 정상적으로 출력됩니다.
response.setContentType("text/html; charset=UTF-8");

[[3]]

[FrontController]

```
if(forward==null) { // forward == null 이면 에러발생
            //System.out.println("로그f2");
            forward=new ActionForward();
            forward.setPath("error/error.jsp");
            forward.setRedirect(false);
            RequestDispatcher dispatcher=request.getRequestDispatcher(forward.getPath());
            dispatcher.forward(request, response);
}else{ // 에러 발생이 아닐 시
            //System.out.println("로그f3");
            if(forward.isRedirect()) { // 새로운 요청을 한다.
                        response.sendRedirect(forward.getPath());
            }
            else { // 기존 요청 정보를 유지
            RequestDispatcher dispatcher=request.getRequestDispatcher(forward.getPath());
            dispatcher.forward(request, response);
            }
}
// : 타겟페이지(인자)로 request,response 객체를 전달하는 메서드
// : 제어권을 넘겨줌 -> 클라이언트가 응답을 확인할 수 있음
```

ex)
[InsertMemberAction]

```
            if(dao.insert(vo)) {
            forward=new ActionForward();
            forward.setPath("/close.jsp");
            forward.setRedirect(true);
        }else { // 입력이 제대로 되지 않은 경우 = 에러가 발생하는 경우이기 때문에
            request.setAttribute("errormsg", "회원가입실패"); // "회원가입실패" 정보를 담아 넘겨준다.
        }
```

[error.jsp]
```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8" isErrorPage="true" %> // 필수
<!DOCTYPE html>
<html>
<body>

<h1>요청하신 서비스를 처리하던중에 문제가 발생했습니다...</h1>
<a href="main.do">메인으로 돌아가기</a>
<hr>
발생한 예외: ${errormsg} // 셋팅되어 넘어온 errormsg 출력

</body>
</html>
```

[[4]]
[main]
```
<%@ taglib prefix="kim" tagdir="/WEB-INF/tags%>
```
// 사용하려는 태그를 WEB-INF > tags 폴더로 관리하였다.
board.tag => 게시글 삭제, 좋아요 기능 구현
login.tag => 로그인한 회원의 정보표시, 로그아웃, 내글보기, 탈퇴하기, 회원가입 기능 구현
reply.tag => 댓글 삭제 기능 구현
serach.tag => 최근에 가입한 회원 3명 표시, 검색 기능 구현
write.tag => (로그인 상태를 구분하여 로그인이 되어 있을 때에만) 게시글 작성, 댓글 작성 기능

[[5]]
```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
```
jsp에서 자주 사용하는 기능을 미리 구현해 놓은 커스텀 태그 라이브러리 모음이다.
반복문 제어 foreach / 복합조건문제어 (if~else, switch) choose when otherwise / 조건문제어 if / 저장영역에 객체를저장 set

EL식 = ${표현식}
view에서 주로 사용 => ${변수명} or ${객체명.변수명} or ${컬렉션객체[인덱스]}로 사용
단, 사용하기 위해서는 scope(request, session, application ...)에 담겨있는 setAttribute를 해주어야한다.
=> 즉, Controller가 Model과 DB에서 전달받아 넘겨주는 데이터를 View가 표현하는 방식이다.

[[6]]
MVC2 방식을 설명하라는 것 같음

[FrontController]

공통 처리가 어려운 것이 문제

FrontController(이후 FC)를 통해서 관리하였다.
이는 기존에 Controller가 jsp였던것에 비해 서블릿 파일이다.
@WebServlet("*.do")를 통해 do 요청을 수행하면 FC로 오게 된다.

```
    public FrontController() {
// FrontController fc = new FrontController(); xxx
// 객체화를 하지 않았는데, 메서드를 사용할 수 있었다.
// 서블릿 컨테이너(== 객체를 관리하는 것) == 웹 서버 == 톰캣이 서블릿을 객체화해주고있음!!
```

```
        super();
        // TODO Auto-generated constructor stub
    }



            protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        actionDO(request, response);
    }

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException {
        actionDO(request, response);
    }

            // Get방식과 Post방식 모두 actionDO를 수행하도록 설정


            private void actionDO(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException{
        String uri = request.getRequestURI(); // 전체URI
        String cp = request.getContextPath(); // 요청경로
        String command = uri.substring(cp.length()); // 전체경로에서 요청경로만 보기 위해 설정
        System.out.println(command); // console에 출력
        response.setContentType("text/html; charset=UTF-8"); // 필터에서 걸러진 페이지에 인코딩 설정

        ActionForward forward =null; // 초기값 설정


            .
            .
            .           // forward에는 수행하려는 기능(###action)에 필요한 경로와
                                                                    == New ###Action
                 request, response가 담겨야 한다.
            .
            .

            이후는 기존 Controller 사용방식과 동일하다.
```

[###Action]

필요한 기능 수행하는 자바코드를 입력 = 그전까지 사용해오던 일반적인 Controller 역할의 클래스파일
단, Action을 implements 해야함
-> request와 response가 필요하기 때문

[Action]

서블릿의 request와 response를 포함하는 (추상 메서드를 가지는 why> 메서드 강제가 필요하기 때문)인터
페이스이다.
-> request에서 데이터 추출, vo에 set, dao에 vo전달, response를 통해 데이터 v에게 전달

```
public interface Action {
    public ActionForward execute(HttpServletRequest request, HttpServletResponse response) throws
```

Exception;
}

[ActionForward]
```
public class ActionForward {
    private String path; // 어디로갈지
    private boolean redirect; // 어떻게 갈지

    public ActionForward() {

    }

    public String getPath() {
        return path;
    }
    public void setPath(String path) {
        this.path = path;
    }
    public boolean isRedirect() {
        return redirect;
    }
    public void setRedirect(boolean redirect) {
        this.redirect = redirect;
    }

}
```

[[우리기능]]
1. 최근 가입한 회원 3명의 이름을 main에 가로로 출력
2. 특정 회원(최근 가입한 회원)이 작성한 글 모아보기

[MemberDAO]
```
final String sql_selectAll="SELECT * FROM (SELECT A.,ROWNUM AS RNUM FROM (SELECT FROM
MEMBER ORDER BY MPK DESC) A WHERE ROWNUM<=3) WHERE RNUM>=1";
// ROWNUM을 통해 테이블 수정 없이 최근에 가입한 순서대로 내림차순으로 정렬하여 3명만 선정

    public ArrayList<MemberVO> selectAll(MemberVO mvo){
        ArrayList<MemberVO> datas=new ArrayList<MemberVO>();
        conn=JDBCUtil.connect();
        try {
            pstmt=conn.prepareStatement(sql_selectAll);
            ResultSet rs=pstmt.executeQuery();
            while(rs.next()) {
                MemberVO memberVO=new MemberVO();
                memberVO.setMid(rs.getString("MID"));
                memberVO.setMname(rs.getString("MNAME"));
                datas.add(memberVO);
            }
        }catch (SQLException e) {
            e.printStackTrace();
        } finally {
            JDBCUtil.disconnect(pstmt, conn);
        }
```

```
        return datas;
    }

[search.tag]
<c:if test="${member.size() == 0}">
    최근에 가입한 회원이 없습니다. // 최근에 가입한 회원이 없을 때 출력
</c:if>
<c:forEach var="member" items="${member}">
    <tr>
        <th><a href="main.do?mid=${member.mid}">[${member.mname}] </a></th>
            // 최근에 가입한 회원의 mname출력 + a태그를 통해 해당 mname 클릭 시 mid를 main.do에게
전달
    </tr>
</c:forEach>

[mainAction]
            if(paramMid == null) {
            bvo.setMid("");
            // mid에 해당하는 데이터가 없는 경우 = 검색에 mid가 없는 경우 ==> 전체보기
        }else {
            bvo.setMid(paramMid);
            // mid에 해당하는 데이터가 있는 경우 = 검색에 mid로 검색 ==> 해당 mid로 검색하기
        }
        System.out.println("if문 후"+bvo);
        ArrayList<BoardSet> datas=bdao.selectAll(bvo); // 게시글 모두 출력용
        ArrayList<MemberVO> member = mdao.selectAll(mvo); // 최근 가입한 3명 출력용

        request.setAttribute("boardMidCheck", bvo.getMid());
```