게시판 만들기

먼저 게시글 쓰기가 있어야 한다.
게시판 리스트 페이지가 필요하므로 board/boardList.jsp 파일을 생성시킨다.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
  <body>
   게시글 목록<br />
  </body>
</html>
```

리스트 페이지가 열리도록하기 위해 FrontController를 만들어주어야 한다.

java resource에서 BoardFrontController클래스 파일을 만들어준다.

패키지는 controller.board로 하도록 하겠다.

```
package controller.board;
public class BoardFrontController  {


}
```

extends HttpServlet **implements** javax.servlet.Servlet를 클래스에 상속을 시켜야한다.

```
public class BoardFrontController extends HttpServlet
        implements javax.servlet.Servlet{

}
```

상속을 시킨 후 ctrl + shift + o를 눌러서 import를 시킨다.

Ctrl + space키를 눌러

```
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

}
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

}
```

를 추가한다. 그리고 ctrl + shift + o를 눌러서 import를 시킨다.

먼저 boardList.jsp 파일이 웹브라우저에서 열리도록 컨트롤러에 코드를 작성한다. 처음 페이지는 get방식이므로 doGet메서드에 작성을 한다.
먼저 주소를 찾기 위해 URI와 contextPath를 불러온 후 뒤 주소만 가져오기 위해 substring을 사용해서 가지고 온다.

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws ServletException,
IOException {

    String requestURI = request.getRequestURI();
    String contextPath = request.getContextPath();
    String command = requestURI.substring(contextPath.length());

}
```

주소를 가지고 온후 주소가 /boardList.board인지 확인 할수 있게 조건문을 사용한다.

```
protected void doGet(HttpServletRequest request,
                     HttpServletResponse response) throws
ServletException, IOException {
 ...
 if(command.equals("/boardList.board")) {

        RequestDispatcher dispatcher =
            request.getRequestDispatcher("/board/boardList.jsp");
        dispatcher.forward(request, response);
    }
}
```

web-inf.xml에 servlet을 추가한다.

```
 <servlet>
  <servlet-name>board</servlet-name>
  <servlet-class>controller.board.BoardFrontController</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>board</servlet-name>
  <url-pattern>*.board</url-pattern>
  </servlet-mapping>
```

boardList.jsp에 글쓰기 링크가 있어한다.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
  <body>
   게시글 목록<br />
    <a href="boardWrite.board">글쓰기</a>
  </body>
</html>
```

먼저 "boardWrite.board"주소에서 열릴 boardForm.jsp파일을 만들어야 할 것이다.
Webapps 밑에boardFrom.jsp를 만든다.

```
<form action="boardRegist.board" method="get">
        <table>
                <caption>게시글 쓰기</caption>
                <tr><td>글쓴이</td>
                        <td><input type="text" name="boardWriter"></td>
                </tr>
                <tr><td>제목</td>
                        <td><input type="text" name="boardSubject"></td>
                </tr>
                <tr><td>내용</td>
                        <td><textarea rows="6" cols="40" name="boardContent"></textarea></td>
                </tr>
                <tr><th colspan=2>
                        <input type="submit" value="게시글 등록">
                </th></tr>
        </table>
</form>
```

FrontController에서 boardForm.jsp파일을 웹브라우저에 전송하는 코드를 작성해준다.

```
public class BoardFrontController extends HttpServlet
        implements javax.servlet.Servlet{
@Override
    protected void doGet(HttpServletRequest request,
        HttpServletResponse response) throws.
            ServletException, IOException {
    …
    if(){
    …
    }else if(command.equals("/boardWrite.board"){
        RequestDispatcher dispatcher =
                request.getRequestDispatcher("/board/boardForm.jsp");
        dispatcher.forward(request, response);
    }
}
```

FrontController에 의해 boardForm.jsp가 웹브라우자에 열리면 자료항목에 자료를 작성을 하고 submit을 하게 된다.

Submit이 된 데이터는 서버단에서 저장이 된 후 목록페이지로 넘어가게 된다.

일단 boardForm.jsp에서 submit을 하면 *boardRegist.board로 전송이 되고 다시 목록 페이지가 열려야 하므로 boardRegist.board* 에서 목록 페이지로 가게 FrontController에 작성을 한다,

```
else if(command.equals("/boardRegist.board"){
        response.sendRedirect("boardList.board");
}
```

이때는 페이지 이동이므로 response.sendRedirect()를 이용하여 목록 페이지로 이동이 되게 만든다.

목록 페이지로 이동이 되는 것을 확인되었다면 전송된 데이터를 저장하기 위한 page-controller를 만들어준다.

Page-controller의 이름은 BoardWriteContoller라고 만들고 패키지는 controller.board로 하자.

```
package controller.board;
public class BoardWriteContoller {
}
```

Front-controller로 부터 데이터를 전달 받기 위한 메서드를 작성한다,

```
public class BoardWriteContoller {
    public void execute(HttpServletRequest request) {

    }
}
```

이제 Front-controller에서 데이터를 전달할 수 있게 BoardWriteContoller객체를 생성하고 메서드를 실행시켜준다.

```
else if(command.equals("/boardRegist.board")) {
        BoardWriteContoller action = new BoardWriteContoller();
        action.execute(request);

        response.sendRedirect("boardList.board");
}
```

BoardWriteContoller에 전달된 값을 받을 수 있게 변수를 정의하고 각 변수에는 boardForm.jsp의 input으로 부터 전송된 값을 request.getParameter() 메서드를 이용하여 받는다.

```
public void execute(HttpServletRequest request) {
	try {
		request.setCharacterEncoding("utf-8");
	} catch (UnsupportedEncodingException e) {}

///		request.getParameter("input의 name을 적어준다.")
	String boardWriter = request.getParameter("boardWriter");
	String boardSubject = request.getParameter("boardSubject");
	String boardContent = request.getParameter("boardContent");
		// 웹브라우저를 사용하는 사용자의 ip주
	String writerIP = request.getRemoteAddr();
}
```

각 변수에 저장된 값은 디비에 전송하기 위해 먼저 DTO에 저장을 해야하므로 DTO를 만든다.

```
package model;

public class BoardDTO {
	Integer boardNum;
	String boardWriter;
	String boardSubject;
	String boardContent;
	String writerIP;
	Integer visitCount;
// getter/setter 만든다.
}
```

DTO를 만들었다면 BoardWriteContoller에서 DTO를 객체 생성후 변수에 저장되어 있는 값을 DTO에 저장한다.

```
public class BoardWriteContoller {
        public void execute(HttpServletRequest request) {

                ...
                BoardDTO dto = new BoardDTO();
               dto.setBoardContent(boardContent);
               dto.setBoardSubject(boardSubject);
               dto.setBoardWriter(boardWriter);
               dto.setWriterIp(writerIP);
        }
}
```

DTO에 있는 값을 데이터베이스에 저장하기 위해 테이블을 만들어야 한다.

```
Create table board(
        board_num number,
        board_Writer varchar2(50),
        board_Subject varchar2(100),
        board_Content varchar2(2000),
        writer_IP varchar2(20),
        visit_Count number,
        board_date date default sysdate
) ;
```

DTO에 있는 값을 DAO로 전달해야 한다,
전달 받을 DAO를 만든다. 그리고 데이터베이스 정보를 적어준다.

```
package model.DAO;
public class BoardDAO{
	String jdbcURL;
	String jdbcDriver;
	Connection con;
	PreparedStatement pstmt;
	ResultSet rs;
	public BoardDAO() {
	jdbcDriver = "oracle.jdbc.driver.OracleDriver";
	jdbcURL = "jdbc:oracle:thin:@192.168.0.69:1521:xe";
	}
	public Connection getConnection() {
		Connection conn = null;
		try {
		Class.forName(jdbcDriver);
		conn = DriverManager.getConnection(jdbcURL,"kosa123","oracle");
		}catch(Exception e) {e.printStackTrace();}
		return conn;
	}
}
```

그리고 데이터를 BoardWriteContoller에서 DTO로 전달 받은 값을 저장하기 위한 메서드를 추가한다.

```
public void boardInsert(BoardDTO dto) {
		con = getConnection();
		String sql = " insert into board(BOARD_NUM,BOARD_WRITER,BOARD_SUBJECT,"
				+ " BOARD_CONTENT, WRITER_IP, visit_Count )"
				+ " values((select nvl(max(board_num),0)+1 from board)"
				+ " ,?,?,?,?,0)";
		try {
				pstmt= con.prepareStatement(sql);
				pstmt.setString(1, dto.getBoardWriter());
				pstmt.setString(2, dto.getBoardSubject());
				pstmt.setString(3, dto.getBoardContent());
				pstmt.setString(4, dto.getWriterIp());
				int i = pstmt.executeUpdate();
				System.out.println(i + " 개 행이(가) 삽입되었습니다.");
		} catch (SQLException e) {e.printStackTrace();
		} finally {
				if(pstmt != null) try{pstmt.close();}catch(Exception e) {}
				if(con != null) try{con.close();}catch(Exception e) {}
		}
}
```

DAO객체를 생성한 후 boardInsert를 통해 **DTO를 전달해준다.**

```
public class BoardWriteContoller {
        public void execute(HttpServletRequest request) {

                ...
                 BoardDTO dto = new BoardDTO();
                dto.setBoardContent(boardContent);
                dto.setBoardSubject(boardSubject);
                dto.setBoardWriter(boardWriter);
                dto.setWriterIp(writerIP);

                BoardDAO dao = new BoardDAO();
                dao.boardInsert(dto);
        }
}
```

```
public class BoradListController {
	public void execute(HttpServletRequest request) {
		BoardDAO dao = new BoardDAO();
	}
}
```

```
public List<BoardDTO> selectAll() {
List<BoardDTO> list = new ArrayList<BoardDTO>();
    con = getConnection();
    sql="select board_num,board_writer,board_subject,"
    + "board_content,writer_ip,visit_count,board_date"
    + "from board"
    + "order by board_num desc";


return list;
}
```

```
try {
		pstmt = con.prepareStatement(sql);
		rs = pstmt.executeQuery();// 출력될 모든 레코드를갖디고 옮
		while(rs.next()) {
			BoardDTO dto = new BoardDTO();
			dto.setBoardContent(rs.getString("board_content"));
			dto.setBoardNum(rs.getInt("BOARD_NUM"));
			dto.setBoardSubject(rs.getString("BOARD_SUBJECT"));
			dto.setBoardWriter(rs.getString("BOARD_WRITER"));
			dto.setVisitCount(rs.getInt("VISIT_COUNT"));
			dto.setWriterIp(rs.getString("WRITER_IP"));
			dto.setBoardDate(rs.getDate("board_date"));
			list.add(dto);
		}
} catch (Exception e) {
		e.printStackTrace();
}
```

```
public class BoradListController {
    public void execute(HttpServletRequest request) {
        BoardDAO dao = new BoardDAO();
        ///// 추가
        List<BoardDTO> list = dao.selectAll();
        request.setAttribute("lists", list);
    }
}
```

BoardFrontController에 boardList.board에 아래 내용을 추가한다.

```
if(command.equals("/boardList.board")) {
    //// 추가
    BoradListController action = new BoradListController();
    action.execute(request);
   ///
    RequestDispatcher dispatcher =
                    request.getRequestDispatcher("/board/boardList.jsp");
    dispatcher.forward(request, response);
}
```

boardList.jsp에서 BoradListController에서 전달된 List에 있는 값을 출력하자.

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html>

...
<body>
게시글 목록<br />
<table border= 1 width="600px">
<thead>
<tr><th>글번호</th><th>글쓴이</th><th>제목</th><th>조회수</th></tr>
</thead>
<tbody>
<c:forEach items="${lists }" var="dto">
<tr><td><a href="boardDetail.board?num=${dto.boardNum }">${dto.boardNum }</a></td><td>${dto.boardWriter }</td>
    <td>${dto.boardSubject }</td><td>${dto.visitCount }</td></tr>
</c:forEach>
</tbody>
</table><br />
<a href="boardWrite.board">글쓰기</a>
</body>
```

BoardFrontController에서 boardDetail.board 주소에서 보여줄 /board/boardInfo.jsp 파일을 전송할 수 있게 한다.

```
else if(command.equals("/boardDetail.board")) {
    RequestDispatcher dispatcher =
    request.getRequestDispatcher("/board/boardInfo.jsp");
    dispatcher.forward(request, response);
}
```

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
</head>
<body>
상세페이지 | 조회수 : 0 | 0.0.0.0<br />
글번호 : 0<br />
글쓴이 : 0<br />
제목 : 0<br />
내용 : 0<br />
수정 | 삭제 | 게시글 리스트
</body>
</html>
```

BoardFrontController의 boardDetail.board에 데이터를 가지고 올 수 있게 page-controller가 있어야 한다.

BoardDetailController를 만들어 상세정보를 가지고 오도록한다.

쿼리스트링으로 받아온 num값을 이용해서 상세정보를 가지고 오자.

```
public class BoardDetailController {
	public void execute(HttpServletRequest request) {
		String num = request.getParameter("num");
		BoardDAO dao = new BoardDAO();
	}
}
```

```
public BoardDTO selectOne(String num) {
        BoardDTO dto = new BoardDTO();
        con = getConnection();
        String sql = "select BOARD_NUM, BOARD_WRITER,
                                BOARD_SUBJECT, "
                    + " BOARD_CONTENT, WRITER_IP,
                        VISIT_COUNT"
                    + " from board "
                    + " where BOARD_NUM = ?";



        return dto;
}
```

```
try {
		pstmt = con.prepareStatement(sql);
		pstmt.setString(1, num);
		rs = pstmt.executeQuery();
		if(rs.next()) {
			dto.setBoardContent(rs.getString("BOARD_CONTENT"));
			dto.setBoardNum(rs.getInt("BOARD_NUM"));
			dto.setBoardSubject(rs.getString("BOARD_SUBJECT"));
			dto.setBoardWriter(rs.getString("BOARD_WRITER"));
			dto.setVisitCount(rs.getInt("VISIT_COUNT"));
			dto.setWriterIp(rs.getString("WRITER_IP"));
		}

}catch(Exception e) {
		e.printStackTrace();
}finally {
		if(rs != null) try{rs.close();}catch(Exception e) {}
		if(pstmt != null) try{pstmt.close();}catch(Exception e) {}
		if(con != null) try{con.close();}catch(Exception e) {}
}
```

```
public void visitCount(String num) {
        con = getConnection();
        sql = " update board "
                + "     set visit_count = visit_count + 1 "
                + "     where board_num = ?";
        try {
                pstmt = con.prepareStatement(sql);
                pstmt.setString(1, num);
                int i = pstmt.executeUpdate();
                System.out.println(i + "개 행이(가) 수정되었습니다.");
        }catch(Exception e) {e.printStackTrace();
        }finally {
                if(pstmt != null) try{pstmt.close();}catch(Exception e) {}
                if(con != null) try{con.close();}catch(Exception e) {}
        }
}
```

```
public class BoardDetailController {
	public void execute(HttpServletRequest request) {
		String num = request.getParameter("num");
		BoardDAO dao = new BoardDAO();

		//// 추가
		dao.visitCount(num);
		BoardDTO dto = dao.selectOne(num);
		request.setAttribute("dto", dto);
	}
}
```

```
else if(command.equals("/boardDetail.board")) {
        //// 추가
        BoardDetailController action = new
                                BoardDetailController();
        action.execute(request);


        RequestDispatcher dispatcher =
        request.getRequestDispatcher("/board/boardInfo.jsp");
        dispatcher.forward(request, response);
}
```

/board/boardInfo.jsp에 수정페이지로 가도록 링크를 만들어준다.

```
<body>
상세페이지  |  조회수 : ${dto.visitCount } | ${dto.writerIp }<br />
글번호 : ${dto.boardNum}<br />
글쓴이 : ${dto.boardWriter }<br />
제목 : ${dto.boardSubject }<br />
내용 : ${dto.boardContent }<br />
<a href="boardUpdate.board?num=${dto.boardNum}">수정</a> | 삭제 | 게시글 리스트
</body>
```

수정하기 위한 주소 *boardUpdate.board에서 boardModifyForm.jsp 파일이 열리도록 Front-Controller에 작성하자.*

수정 후 디테일 페이지로 이동하도록 작성을 한다.
이렇게 하면 페이지가 이동하지 않은 것 처럼 보인다.

```
else if(command.equals("/boardUpdate.board")) {
        BoardDetailController action = new BoardDetailController();
        action.execute(request);
        RequestDispatcher dispatcher =
        Request.getRequestDispatcher("/board/boardModifyForm.jsp");
        dispatcher.forward(request, response);
}
```

boardModifyForm.jsp페이지를 만든다.

```
<form action="boardModify.board" method="get">
		<input type="hidden" name="boardNum" value="${dto.boardNum }"/>
		<table>
		<caption>게시글 수정</caption>
			<tr><td>글쓴이</td>
			<td><input type="text" name="boardWriter" value="${dto.boardWriter }"></td>
			</tr>
			<tr><td>제목</td>
			<td><input type="text" name="boardSubject" value="${dto.boardSubject }"></td>
			</tr>
			<tr><td>내용</td>
			<td><textarea rows="6" cols="40" name="boardContent">${dto.boardContent }</textarea></td>
			</tr>
			<tr><th colspan=2>
			<input type="submit" value="게시글 수정 완료" />
			<input type="button" value="뒤로가기"  onclick="javascript:history.back()" />
			</th></tr>
		</table>
</form>
```

Front-Controller에서 수정완료를 위한 주소를 작성한다,
수정이 완료가 되면 다시 상세페잊로 이동할 수 있게 만든다.

```
else if(command.equals("/boardModify.board")) {
        response.sendRedirect("boardDetail.board?num="+
        request.getParameter("boardNum"));
}
```

데이터베이스에서 수정할 수 있게 page-controller를 만들어 준다.
BoardModifyController를 만든다.

```
public class BoardModifyController {
	public void execute(HttpServletRequest request) {
		try {
			request.setCharacterEncoding("utf-8");
		}catch(Exception e) {}
		BoardDTO  dto = new BoardDTO();
		dto.setBoardContent(request.getParameter("boardContent"));
		dto.setBoardNum( Integer.parseInt(request.getParameter("boardNum")));
		dto.setBoardSubject(request.getParameter("boardSubject"));
		dto.setBoardWriter(request.getParameter("boardWriter"));
		BoardDAO dao = new BoardDAO();
	}
}
```

```
public void boardUpdate(BoardDTO dto) {
		con = getConnection();
		String sql = "update board"
				+ " set BOARD_WRITER = ? , BOARD_SUBJECT = ?, "
				+ "	BOARD_CONTENT = ? "
				+ " where board_num = ?";
		try {
				pstmt = con.prepareStatement(sql);
				pstmt.setString(1, dto.getBoardWriter());
				pstmt.setString(2, dto.getBoardSubject());
				pstmt.setString(3, dto.getBoardContent());
				pstmt.setInt(4, dto.getBoardNum());
				int i = pstmt.executeUpdate();
				System.out.println(i + "개 행이(가) 수정되었습니다.");
		}catch(Exception e) {e.printStackTrace();
		}finally {
				if(pstmt != null) try{pstmt.close();}catch(Exception e) {}
				if(con != null) try{con.close();}catch(Exception e) {}
		}
}
```

```
public class BoardModifyController {
	public void execute(HttpServletRequest request) {
		try {
			request.setCharacterEncoding("utf-8");
		}catch(Exception e) {}
		BoardDTO  dto = new BoardDTO();
		…
		BoardDAO dao = new BoardDAO();
		/// 추가
		dao.boardUpdate(dto);
	}
}
```

Front-Comntrolller 에 추가한다.

```
else if(command.equals("/boardModify.kosa")) {
        //// 추가
        BoardModifyController action = new
                                BoardModifyController();
        action.execute(request);
        /////

        response.sendRedirect("boardDetail.kosa?num="+
                request.getParameter("boardNum"));
}
```

/board/boardInfo.jsp에 삭제를 할 수 있게 링크를 만들어준다.

```
<body>
상세페이지  |  조회수 : ${dto.visitCount } | ${dto.writerIp }<br />
글번호 : ${dto.boardNum}<br />
글쓴이 : ${dto.boardWriter }<br />
제목 : ${dto.boardSubject }<br />
내용 : ${dto.boardContent }<br />
<a href="boardUpdate.board?num=${dto.boardNum}">수정</a> | <a href="boardDel.board?num=${dto.boardNum}">삭제</a> | 게시글 리스트
</body>
```

Front-Controller에서 삭제가 되면 목록 페이지로 이동하도록한다.

```
else if(command.equals("/boardDel.kosa")) {
        response.sendRedirect("boardList.kosa");
}
```

데이터베이스에서 삭제가 되도록 page-controller를 만들어준다.

```
public class BoardDelController {
        public void execute(HttpServletRequest request) {
                String num = request.getParameter("num");
                BoardDAO dao = new BoardDAO();
        }
}
```

DAO에 메서드를 추가해준다.

```
public void boardDel(String num) {
        con = getConnection();
        String sql = "delete from board where board_num = ?";
        try {
                pstmt = con.prepareStatement(sql);
                pstmt.setString(1, num);
                int i = pstmt.executeUpdate();
                System.out.println(i +" 개 행이(가) 삭제되었습니다.");
        }catch(Exception e) {e.printStackTrace();
        } finally {
                if(pstmt != null) try{pstmt.close();}catch(Exception e) {}
                if(con != null) try{con.close();}catch(Exception e) {}
        }
}
```

page-controller메서드를 호출할 수 있게 추가해준다.

```
public class BoardDelController {
        public void execute(HttpServletRequest request) {
                String num = request.getParameter("num");
                BoardDAO dao = new BoardDAO();
                /// 추가
                 dao.boardDel(num);
        }
}
```

Front-Controller에서 BoardDelController의 메서드를 추가한다.

```
else if(command.equals("/boardDel.kosa")) {
      //// 추가
        BoardDelController action = new BoardDelController();
        action.execute(request);
       ////

        response.sendRedirect("boardList.kosa");
}
```