
<복습>

< 1개의 **JSP page** 로 작성시**>**

webapp/dbConnection/jspDBConnection.jsp - jsp 페이지에서 DB의 게시글 리스트 읽어오기

- 자바프로그램과 데이터베이스를 연동할때 JSP 페이지에서 요청을 받아, 직접 DB에 접근, 동일한 결과화면도 하나의 JSP 페이지에서 직접 처리

- 변수저장(오라클 사용자 정보, 드라이버, url) : String user = “hr”;
- DBMS 드라이버 로딩 : Class.forName(driver);
- DB 커넥션 생성(DB 연결을 위한) : Connection conn = DriverManager.getConnection(url, user, pass);
- DB에 쿼리를 발행를 하는 활성 커넥션 PreparedStatement 객체 얻음 : PreparedStatement pstmt = conn.prepareStatement(“SELECT*FROM jspbbs);
- 받은 쿼리를 실행할 ResultSet 객체 : ResultSet rs = pstmt.executeQuery();
- 게시글을 위한 객체 생성 : ArrayList boardList = new ArrayList();
- 데이터 출력 : while(rs.next()) { Board b = new Board(); boardList.add(b); }
- 활성 커넥션 PreparedStatement 종료, 생성한 객체를 역순으로 닫음
- HTML - 원하는 데이터 출력

< 1개의 **Java**와 **1**개의 **JSP page** 로 작성시**>**

요청 처리와 화면 처리 모듈 분리 : 요청처리(controller) 와 화면처리(Presentation Layer) 의 구분

DB - Controller(Java) - Presentation Layer(JSP)

1) 게시 글 리스트 요청을 처리하는 컨트롤러 클래스(controller) 77p

- **com.jspstudy.ch06.controller > BoardListController01.java**
- 클라이언트 요청을 받아서 DB에서 게시 글 리스트를 읽어오는 코드를 JSP 페이지에서 복사

- 어노테이션 : @WebServlet("/boardList01")
- httpservlet, doget, req, res
- 변수저장(오라클정보)
- DB 커넥션 : Connection conn = null;
- DB 쿼리 발행 객체 : PreparedStatement pstmt = null;
- DB 쿼리 결과 저장 객체 : ResultSet rs = null;
- 게시글을 위한 객체 : ArrayList boardList = null;
- try { DBMS 드라이버 로딩 및 각 벨류입력(DB 커넥션, DB 쿼리 발행 객체, DB 쿼리 결과 저장 객체, 게시글을 위한 객체(boardList = new ArrayList();), while(rs.next()) {Board b = new Board(); b.setNo(rs.getInt(1)); boardList.add(b); } }
- catch (예외발생시) { 처리 }
- finally (작업완료시) { 객체역순종료 }
- 요청처리 결과 저장(HttpRequest의 속성에 저장) : request.setAttribute("bList", boardList);
- 포워딩(RequestDispatcher) : RequestDispatcher rd = request.getRequestDispatcher("/WEB-INF/board/boardList01.jsp"); rd.forward(request, response);

2) 게시 글 리스트 뷰 페이지 (Presentation Layer) 79p

- **webapp/WEB-INF/board/boardList01.jsp**
- DynamicWebProject에서 WEB-INF 폴더는 외부에서 직접 접근할 수 없지만 서버 내부에서 포워드 기법을 활용하면 이 폴더에 JSP 페이지 저장하고 뷰 페이지로 활용할 수 있다.
- HTML : taglib, foreach

<2개의 **Java** 와 **1**개의 **JSP page** 로 작성시**>**
DAO(Data Access Object) 사용 : 컨트롤러의 DB 중복 작업 제거를 위해 별도의 클래스로 구분
DB - DAO, BoardListController - Presentation Layer
과정마다 try, catch 를 통해 브레이크포인트를 찾는다.

1) 데이터베이스 작업을 전담하는 BoardDao 객체(DAO) 81p

- **com.jspstudy.ch06.dao > BoardDao01.java**
- 앞선 컨트롤러 페이지에서 DB 내용만을 전담
- public BoardDao01() 와 public ArrayList boardList() 가 실행되며, 트라이캐치로 확인

- DAO 클래스 선언 : public class BoardDao01 {
- 변수저장(오라클정보) : private static[1] final String DRIVER, URL, USER, USER
- DB 커넥션 : Connection conn = null;
- DB 쿼리 발행 객체 : PreparedStatement pstmt = null;
- DB 쿼리 결과 저장 객체 : ResultSet rs = null;
- DBMS 드라이버 로딩, 트라이캣치 : public BoardDao01() { try { Class.forName(DRIVER); } catch (없으면){에러} }
- 게시글을 위한 객체, 트라이캣치 : public ArrayList boardList() {
    - 변수에 쿼리문 저장 : String sqlBoardList = "SELECT * FROM jspbbs ORDER BY no DESC";
    - 게시 글 리스트를 위한 ArrayList 객체 선언 : ArrayList boardList = null;

  try { DB 커넥션, DB 쿼리 발행 객체, DB 쿼리 결과 저장, ArrayList 객체 초기화(boardList = new ArrayList();) , while문 데이터 출력}
  catch (예외발생시) { 처리 }
  finally (작업완료시) { 객체역순종료 }
- DB 게시글 리스트 반환 : return boardList; }
- }

2) 게시 글 리스트 컨트롤러 클래스 수정 84p

- **com.jspstudy.ch06.controller > BoardListController.java**
- 앞선 컨트롤러 페이지에서 DB 내용을 제외

- 어노테이션 : @WebServlet("/boardList01")
- httpservlet, doget, req, res
- BoardDao01 객체를 생성하고 데이터베이스에서 게시 글 리스트를 읽어온다. :
    - BoardDao 01 dao = new BoardDao01();
    - ArrayList bList = dao.boardList();
- 결과 데이터를 HttpRequest의 속성에 저장 : request.setAttribute("bList", bList);
- 포워딩 : RequestDispatcher rd = request.getRequestDispatcher("/WEB-INF/board/boardList01.jsp"); rd.forward(request, response);

3) 게시 글 리스트 뷰 페이지 (Presentation Layer) 은 중복이라 생략

- **webapp/WEB-INF/board/boardList01.jsp**

---

[1] static : 이곳에서의 의미는 속성(멤버 변수)중에서 공통속성에 static을 붙이는 것이다. 그외에도 메서드에 static 키워드가 붙으면 클래스 메서드가 되어 객체를 만들지 않아도 '클래스명.메소드명' 형태로 호출할 수 있다.

데이터베이스 커넥션 풀(Database Connection Pool) : DB 효율화, JNDI 방식의 DBCP사용

- JNDI를 이용한 커넥션 풀의 설정은 톰캣의 context.xml에 설정하는 방법과 애플리케이션에서 별도의 context.xml를 작성해 설정하는 방법이 있다.

사전 작업 : 라이브러리 참조, context.xml 설정(DBCP)

DBCP를 활용한 게시판 구현하기

1) DAO(Data Access Object) 클래스 : DBCP를 이용해 DB와 연동하는 DAO 클래스 작성 88p

- **com.jspstudy.ch06.dao > BoardDao.java** //BoardDao01.java에서 DBCP 방식만 추가
- public BoardDao01() 와 public ArrayList boardList() 가 실행되며, 트라이캐치로 확인

- JNDI 방식의 DBCP를 활용한 DAO 클래스 선언 : public class BoardDao01 {
- DB 커넥션 : private Connection conn;
- DB 쿼리 발행 객체 : private PreparedStatement pstmt;
- DB 쿼리 결과 저장 객체 : private ResultSet rs;
- DBCP의 커넥션 객체 대여 : private static DataSource ds;
- JNDI 객체 생성, 트라이캣치 public BoardDao() { try {
    - InitialContext 객체 생성 : Context initContext = new InitialContext();
    - 디렉토리 url : Context envContext = (Context) initContext.lookup("java:/comp/env");
    - DBCP의 커넥션 객체 대여 :ds = (DataSource) envContext.lookup("jdbc/bbsDBPool"); } catch (없으면){에러} }
- 게시글을 위한 객체, 트라이캣치 : public ArrayList boardList() {
    - 변수에 쿼리문 저장 : String sqlBoardList = "SELECT * FROM jspbbs ORDER BY no DESC";
    - 게시 글 리스트를 위한 ArrayList 객체 선언 : ArrayList boardList = null;

  try { DB 커넥션, DB 쿼리 발행 객체, DB 쿼리 결과 저장, ArrayList 객체 초기화(boardList = new ArrayList();) , while문 데이터 출력}
  catch (예외발생시) { 처리 }
  finally (작업완료시) { 객체역순종료 }
- DB 게시글 리스트 반환 : return boardList; }
- }

2) 게시 글 리스트 컨트롤러 클래스 수정 92p

- **com.jspstudy.ch06.controller > BoardListController.java** //기존과 거의 동일함
- 기존 파일에서 DBCP 방식만 추가, 사실상 boardList01 -> boardList 만 바꿔진 것

- 어노테이션 : @WebServlet("/boardList")
- httpservlet, doget, req, res
- BoardDao 객체를 생성하고 데이터베이스에서 게시 글 리스트를 읽어온다. :
    - BoardDao dao = new BoardDao();
    - ArrayList bList = dao.boardList();
- 결과 데이터를 HttpRequest의 속성에 저장 : request.setAttribute("bList", bList);
- 포워딩 : RequestDispatcher rd = request.getRequestDispatcher("/WEB-INF/board/boardList01.jsp"); rd.forward(request, response);

3) 부트스트랩을 활용해 게시 글 리스트 화면 구현하기 93p

- **webapp/WEB-INF/board/boardList.jsp**
- 79p 의 boardList01 에서 CSS만 부트스트랩 적용
- HTML : taglib, foreach

4) DAO(Data Access Object) 클래스에 메서드 추가 95p

- **com.jspstudy.ch06.dao > BoardDao.java**
- 88p DAO에 아래의 내용을 추가하여, 게시판글과 게시판글 상세보기로 나뉜다.
- 앞서 선언한 변수들의 초기화도 진행

- 게시 글 상세보기 요청 시 호출되는 메서드 : public Board getBoard(int no) {
  - 변수에 쿼리문 저장 : String sqlBoard = "SELECT * FROM jspbbs WHERE no=?";
  - Board 객체 선언 : Board board = null;
  - try{
    DataSource 객체를 이용해 커넥션을 대여 : conn = ds.getConnection();
    DB 쿼리 발행 객체(pstmt) 에 쿼리문전달 pstmt = conn.prepareStatement(sqlBoard);
    SELECT 쿼리의 Placeholder(?)와 데이터를 맵핑 : pstmt.setInt(1, no);
    DB 쿼리 결과 저장 객체 받음 : rs = pstmt.executeQuery();
    Board 객체의 각 프로퍼티에 값을 설정 : if(rs.next()) { board = new Board();
    board.setNo(rs.getInt("no")); }
    catch (예외발생시) { 처리 }
    finally (작업완료시) { 객체역순종료 }
  - DB 게시글 리스트 반환 : return boardList; }

5) 게시 글 상세보기 요청을 처리하는 컨트롤러 클래스 99p

- **com.jspstudy.ch06.controller > BoardDetailController.java**

- 어노테이션 : @WebServlet("/boardDetail")
- httpservlet, doget, req, res
- 게시글 no 파라미터 : String no = request.getParameter("no");
- BoardDao 객체 구하고 게시 글 번호(no)에 해당하는 게시 글을 읽어온다. :
  - BoardDao dao = new BoardDao();
  - Board board = dao.getBoard(Integer.valueOf(no));
- 결과 데이터를 HttpRequest의 속성에 저장 : request.setAttribute("board", board);
- 포워딩 : RequestDispatcher rd = request.getRequestDispatcher("/WEB-INF/board/boardDetail.jsp"); rd.forward(request, response);

6) 게시 글 상세보기 뷰 페이지 100p

- webapp/WEB-INF/board/boardDetail.jsp
- HTML : taglib, EL

7) 여러 페이지에서 중복되는 header와 footer 페이지 모듈화 102p

- webapp/WEB-INF/pages/header.jsp
- webapp/WEB-INF/pages/footer.jsp
- 앞에서 작성한 boardList.jsp와 boardDetail.jsp 페이지에서 header 부분과 footer 부분 정리

8) 게시 글쓰기 폼 요청을 처리하는 컨트롤러 클래스

- com.jspstudy.ch06.controller > BoardWriteFormController
- 어노테이션 : @WebServlet("/writeForm"))
- httpservlet, doget, req, res
- 포워딩 : RequestDispatcher rd = request.getRequestDispatcher("/WEB-INF/board/writeForm.jsp"); rd.forward(request, response);

9) 게시 글쓰기 폼 뷰 페이지

- webapp/WEB-INF/board/writeForm.jsp
- HTML