<서블릿 기본 게시판 구현하기>

1. DB TABLE 생성 및 설정

1) 게시판 테이블 생성 //what does mean jspbbs?

- /WEB-INF/sql/JSPStudyBBS.sql 파일을 참고해 데이터를 테이블에 추가
- 게시글 번호, 제목, 내용, 글쓴이, 날짜, 조회수, 비밀번호, 파일정보
- no, title, writer, content, reg_date, read_count, pass, file1

```
CREATE TABLE jspbbs(
 no NUMBER PRIMARY KEY,
 title VARCHAR2(50 CHAR) NOT NULL,
 writer VARCHAR2(20 CHAR) NOT NULL,
 content VARCHAR2(1000 CHAR) NOT NULL,
 reg_date TIMESTAMP NOT NULL,
 read_count NUMBER(5) NOT NULL,
 pass VARCHAR2(20 CHAR) NOT NULL,
 file1 VARCHAR2(100 CHAR)
);
CREATE SEQUENCE jspbbs_seq;
```

2) DBCP(Database Connection Pool) 설정
- webapp/META-INF/context.xml

```
<?xml version="1.0" encoding="UTF-8"?>[1]
<Context>
  <Resource
    name="jdbc/bbsDBPool" //JDBC 이름, 변경가능
    auth="Container" //컨테이너를 자원 관리자로 기술
    type="javax.sql.DataSource" //웹에서 이 리소스를 사용할 때 DataSource로 리턴됨
    driverClassName="oracle.jdbc.driver.OracleDriver" //JDBC 드라이버
    url="jdbc:oracle:thin:@localhost:1521:xe" //@ip주소:포트번호:전역DB이름
    username="hr" //DB 접속 계정
    password="hr" //DB 접속 계정의 비밀번호
    factory="org.apache.commons.dbcp2.BasicDataSourceFactory"
    maxActive="10" //최대 연결가능한 커넥션수(기본 20개)
    maxIdle="5"  /> //Connection pool 유지를 위해 최대 대기 connection 숫자
</Context>
```

3) 백엔드 라이브러리 추가

- Oracle 접속 드라이버 : ojdbc6.jar
- commons dbcp2 라이브러리 : commons-dbcp2-2.11.0.jar, commons-pool2-2.12.0.jar, commons-logging-1.3.jar
- 표준 태그(Java Standard Tag Library – JSTL) 라이브러리 : taglibs-standard-impl-1.2.5.jar, taglibs-standard-spec-1.2.5.jar

4) 프런트엔드 라이브러리 추가

- 부트스트랩 : webapp/bootstrap : bootstrap.bundle.min.js, bootstrap.min.css
- jQuery : webapp/js : jquery-3.3.1.min.js

---

[1] JSP에서 DB연동 하기 - JNDI, DBCP(커넥션풀) 이용

2. Value Object 클래스 (VO≒DTO≒JavaBeans) [2]

1) 하나의 게시 글 정보를 저장하는 Board 클래스를 작성 2p
- com.jspstudy.bbs.vo > board.java

```
public class Board {
  private int no; // 게시판에 필요한 변수 선언
  private String title;
  private String writer;
  private String content;
  private Timestamp regDate;
  private int readCount;
  private String pass;
  private String file1;

  public Board() { } //BoardDao.java 에서 JNDI 객체 생성에 호출됨
  public Board(int no, String title, String writer, String content,
               Timestamp regDate, int readCount, String pass, String file1) {
    this.no = no; // 참조변수 this로 (동일명) 초기화
    this.title = title;
    this.writer = writer;
    this.content = content;
    this.regDate = regDate;
    this.readCount = readCount;
    this.pass = pass;
    this.file1 = file1;
    }
                                                    ▸source >  Generate getter & setter 입력시 자동 생성
    public int getNo() {return no;} // 호출될 함수 get,set 선언
    public void setNo(int no) {this.no = no;}
    public String getTitle() {return title;}
    public void setTitle(String title) {this.title = title;}
    public String getWriter() {return writer;}
    public void setWriter(String writer) {this.writer = writer;}
    public String getContent() {return content;}
    public void setContent(String content) {this.content = content;}
    public Timestamp getRegDate() {return regDate;}
    public void setRegDate(Timestamp regDate) {this.regDate = regDate;}
    public int getReadCount() {return readCount;}
    public void setReadCount(int readCount) {this.readCount = readCount;}
    public String getPass() {return pass;}
    public void setPass(String pass) {this.pass = pass;}
    public String getFile1() {return file1;}
    public void setFile1(String file1) {this.file1 = file1;}
}
```

2

Model
- 애플리케이션의 정보, 데이터를 나타낸다. 데이터베이스(처음의 정의하는 상수, 초기화값, 변수 등)
  데이터, 정보들의 가공을 책임지는 컴포넌트를 말한다.

View
- input 텍스트, 체크박스 항목 등과 같은 사용자 인터페이스 요소를 나타낸다. 다시 말해 데이터 및 객체의 입력, 그리고 보여주는 출력을 담당한다. 데이터를 기반으로 사용자들이 볼 수 있는 화면이다.

Controller
- 데이터와 사용자인터페이스 요소들을 잇는 다리 역할을 한다.
  즉, 사용자가 데이터를 클릭하고, 수정하는 것에 대한 "이벤트"들을 처리하는 부분을 뜻한다.

DAO(Data Access Object)
- DB의 data에 접근하기 위한 객체이다. DB에 접근하기 위한 로직을 분리하기 위해 사용한다.
- 직접 DB에 접근하여 data를 삽입, 삭제, 조회 등 조작할 수 있는 기능을 수행한다.
- MVC 패턴의 Model에서 이와 같은 일을 수행한다.

DTO(Data Transfer Object)
- DTO는 계층 간(Controller, View, Business Layer) 데이터 교환을 위한 자바 빈즈(Java Beans)를 의미한다.
- DTO는 로직을 가지지 않는 데이터 객체이고 getter/setter 메소드만 가진 클래스를 의미한다.

자바빈(JavaBean)이란?
- JSP에서 객체를 가져오기 위한 기법으로 데이터 전달 오브젝트 파일(DTO).자바로 작성된 컴포넌트들(클래스)
- 사용 목적 : JSP 페이지의 로직 부분을 분리해서 코드를 재사용함으로써 프로그램의 효율을 높이기 위해서 사용한다.

자바빈 설계 규약
- 멤버변수마다 별도의 getter, setter 메서드가 존재해야 한다.
- get 메소드는 매개변수가 존재하지 않아야한다.
- set 메소드는 반드시 하나 이상의 매개변수가 존재해야 한다.
- 생성자는 매개변수가 존재하지 않아야 한다.
- 멤버변수의 접근제어자는 private이고 각 getter, setter 메서드의 접근제어자는 public , 클래스의 접근제어자는 public 으로 정의한다.

VO(Value Object)
- VO는 값 오브젝트로써 값을 위해 쓰인다.Read-Only 특징(사용하는 도중에 변경 불가능하며 오직 읽기만 가능)을 가진다. DTO와 유사하지만 VO는 getter 기능만 존재한다.

DTO와 VO 차이점
- DTO는 인스턴스 개념이고 VO는 리터럴 값 개념이다.
- VO는 값들에 대해 Read-Only를 보장해줘야 존재의 신뢰성이 확보되지만,
  DTO의 경우는 단지 데이터를 담는 그릇의 역할일 뿐 값은 그저 전달되어야 할 대상일 뿐이다.
- 값 자체에 의미가 있는 VO와 전달될 데이터를 보존해야하는 DTO의 특성상 개념이 다르다.

[3]

3. 게시 글 리스트 **: Read**

[3] https://velog.io/@mingkiii/JSPMVC%EC%99%80-DAO-DTO-VO%EB%9E%80

boardList.jsp의 title click -> boardDetail page 와 no req -> BoardDetailController 에서 getBoard() req -> BoardDao getBoard() -> board -> view rep

1) DAO 클래스에 게시 글 리스트를 읽어오는 메서드 추가 4p
- com.jspstudy.bbs.dao > BoardDao.java

```
public class BoardDao {  // DBCP를 활용할 DAO 클래스 선언
  // 가장 초반부에는 선언만 해주고, 이후 각 메서드마다 초기화로 사용한다.
  private Connection conn; // DB 커넥션 선언
  private PreparedStatement pstmt; // DB 쿼리 발행 객체 선언
  private ResultSet rs; // DB 쿼리 결과 저장 객체 선언
  private static DataSource ds; // DBCP 커넥션 객체 선언
  public BoardDao() {   // JNDI 설정을 위한 객체 생성
    try {
      Context initContext = new InitialContext(); // InitialContext 객체 생성
      Context envContext = (Context) initContext.lookup("java:/comp/env"); //디렉토리 url
      ds = (DataSource) envContext.lookup("jdbc/bbsDBPool");   // DBCP의 커넥션 객체 초기화
    } catch(NamingException e) { e.printStackTrace(); }
  }
  public ArrayList<Board> boardList() {  // 게시글 목록 객체
    String sqlBoardList = "SELECT * FROM jspbbs ORDER BY no DESC"; // 쿼리 변수
    ArrayList<Board> boardList = null;  // 게시글 객체 ArrayList 선언
    try{
      conn = ds.getConnection(); // DB 커넥션 초기화
      pstmt = conn.prepareStatement(sqlBoardList); // DB 쿼리 발행 객체 초기화
      rs = pstmt.executeQuery(); // DB 쿼리 결과 저장 객체 초기화
      boardList = new ArrayList<Board>(); // ArrayList 객체 초기화
      while(rs.next()) {    //DB 쿼리 결과가 있으면,
        Board board = new Board(); // Board 객체를 생성해
        board.setNo(rs.getInt("no")); // Board 객체에 no ~ file을 담음
        board.setTitle(rs.getString("title"));
        board.setWriter(rs.getString("writer"));
        board.setContent(rs.getString("content"));
        board.setRegDate(rs.getTimestamp("reg_date"));
        board.setReadCount(rs.getInt("read_count"));
        board.setPass(rs.getString("pass"));
        board.setFile1(rs.getString("file1"));
        boardList.add(board);     //게시글 목록 객체에 DB 내용을 추가
      }
    } catch(Exception e) { e.printStackTrace();
    } finally {
      try {
        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
      } catch(SQLException se) {}
    }
    return boardList;
    } //end boardList()
}
```

2) 게시 글 리스트 요청을 처리하는 컨트롤러 클래스 8p
- com.jspstudy.bbs.controller > BoardListController.java

```java
@WebServlet("/boardList")
public class BoardListController extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      BoardDao dao = new BoardDao(); // BoardDao 객체 생성
      ArrayList<Board> bList = dao.boardList(); // 게시글 목록 객체 저장
      request.setAttribute("bList", bList); // HttpRequest 속성에 저장
      RequestDispatcher rd = request.getRequestDispatcher("/WEB-INF/board/boardList.jsp");
      rd.forward(request, response);
    }
}
```

3) 여러 페이지에서 중복되는 header와 footer 페이지 모듈화 (생략)
- webapp/WEB-INF/pages/header.jsp

4) 게시 글 리스트 뷰 페이지
- webapp/WEB-INF/board/boardList.jsp

```jsp
<!DOCTYPE html>
<html lang="ko">
 <head>
  <meta charset="utf-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>게시 글 리스트</title>
  <link href="bootstrap/bootstrap.min.css" rel="stylesheet" >
 </head>
 <body>
  <div class="container">
    <%@ include file="../pages/header.jsp" %>
  <!-- content -->
  <div class="row my-5 text-center">
    <div class="col">
      <h2 class="fs-3 fw-bold">게시 글 리스트</h2>
    </div>
  </div>
  <form name="searchForm" id="searchForm" action="#" class="row justify-content-center my-3">
  <div class="col-auto">
    <select name="type" class="form-select">
      <option value="title">제목</option>
      <option value="writer">작성자</option>
      <option value="content">내용</option>
    </select>
  </div>
  <div class="col-4">
    <input type="text" name="keyword" class="form-control" id="keyword"/>
  </div>
  <div class="col-auto">
    <input type="submit" value="검 색" class="btn btn-primary"/>
  </div>
  </form>
  <div class="row">
```

```
      <div class="col text-end">
        <a href="writeForm" class="btn btn-outline-success">글쓰기</a>
      </div>
    </div>
    <div class="row my-3">
      <div class="col">
        <table class="table table-hover">
          <thead>
            <tr class="table-dark">
              <th>NO</th><th>제목</th><th>작성자</th><th>작성일</th><th>조회수</th>
            </tr>
          </thead>
          <tbody class="text-secondary">
// 게시 글이 있는 경우 게시 글 수만큼 반복하면서 게시 글을 출력
          <c:if test="${ not empty bList }">
          <c:forEach var="b" items="${bList}">
          <tr>
// 유저의 Action 으로 boardDetail 페이지와 no 에 대한 request가 요청되는 것
// 반복문 안에서 한 행의 게시 글을 출력하면서 게시 글 상세보기로 넘어갈 수 있도록 링크를 설정
            <td>${ b.no }</td>
            <td><a href="boardDetail?no=${b.no}" class="text-decoration-none link-secondary">
              ${ b.title }</a></td>
            <td>${ b.writer }</td>
            <td><fmt:formatDate value="${ b.regDate }" pattern="yyyy-MM-dd HH:mm:ss"/></td>
            <td>${ b.readCount }</td>
          </tr>
          </c:forEach>
          </c:if>
// 게시글이 없는 경우
          <c:if test="${ empty bList }">
          <tr>
            <td colspan="5" class="text-center">게시 글이 존재하지 않습니다.</td>
          </tr>
          </c:if>
          </tbody>
          </table>
        </div>
        </div>
<!-- footer  -->
        <%@ include file="../pages/footer.jsp" %>
        </div>
    <script src="bootstrap/bootstrap.bundle.min.js"></script>
  </body>
</html>
```

3-1. 게시글 상세보기 : Read(Detail)

1) 게시 글 상세 보기 : DAO 클래스에 no에 해당하는 게시 글 정보를 읽어오는 메서드 추가 13p
- com.jspstudy.bbs.dao > BoardDao.java

```
public Board getBoard(int no) {
  String sqlBoard = "SELECT * FROM jspbbs WHERE no=?";  // 쿼리 변수
  Board board = null; //쿼리 결과를 담을 객체 선언 및 초기화
  try {
    conn = ds.getConnection(); // DB 커넥션 초기화
    pstmt = conn.prepareStatement(sqlBoardList); // DB 쿼리 발행 객체 초기화
    pstmt.setInt(1,  no); // placeholder 지정
    rs = pstmt.executeQuery();  // DB 쿼리 결과 저장 객체 초기화
    if(rs.next()) { // 쿼리 결과에 담을 데이터들
      board = new Board();
      board.setNo(rs.getInt("no"));
      board.setTitle(rs.getString("title"));
      board.setWriter(rs.getString("writer"));
      board.setContent(rs.getString("content"));
      board.setRegDate(rs.getTimestamp("reg_date"));
      board.setReadCount(rs.getInt("read_count"));
      board.setPass(rs.getString("pass"));
      board.setFile1(rs.getString("file1"));
    }
  } catch(SQLException e) {e.printStackTrace();
  } finally {
    try {
      if(rs != null) rs.close();
      if(pstmt != null) pstmt.close();
      if(conn != null) conn.close();
    } catch(SQLException e) {}
  }
  return board;
} // end getBoard(int no);
```

2) 게시 글 상세보기 요청을 처리하는 컨트롤러 클래스 15p
- com.jspstudy.bbs.controller.BoardDetailController

```
@WebServlet("/boardDetail") //boardDetail로 들어오는 모든 요청 처리
public class BoardDetailController extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      String no = request.getParameter("no");     // 게시글번호 선언 및 초기화
      BoardDao dao = new BoardDao();   // BoardDao 객체 선언 , 컨트롤러는 dao 로 호출한다.
      Board board = dao.getBoard(Integer.valueOf(no)); // BoardDao.dao.no 를 board 로 초기화
      request.setAttribute("board", board); // board = no 를 set
      RequestDispatcher rd = request.getRequestDispatcher("/WEB-INF/board/boardDetail.jsp");
      rd.forward(request, response);
    }}
```

3) 게시 글 상세보기 뷰 페이지 16p (단순 HTML 이므로 생략)
- webapp/WEB-INF/board/boardDetail.jsp

4. 게시글 쓰기 **: Create(insert)**

글쓰기 폼(writeForm) 의 등록하기(submit) -> formcheck.js 의 유효성 검사 및 action="writeProcess" -> BoardWriteController 에서 writeForm의 내용을 전달 -> BoardDao의 insertBoard(board) 메서드가 DB에 저장 -> boardList 로 이동후 종료

1) 게시 글쓰기 폼 요청을 처리하는 컨트롤러 클래스 18p
- com.jspstudy.bbs.controller > BoardWriteFormController.java

```
@WebServlet("/writeForm")
public class BoardWriteFormController extends HttpServlet {
  protected void doGet(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
// 게시 글 쓰기 폼 요청은 게시 글쓰기 폼만 화면에 보여주면 되므로 필요한 모델 데이터가 없기 때문에 아무런 처리 없이 뷰 페이지로 포워딩 하면 된다.
      RequestDispatcher rd = request.getRequestDispatcher("/WEB-INF/board/writeForm.jsp");
      rd.forward(request, response);
    }
}
```

2) 게시 글쓰기 폼 뷰 페이지 (간략) 19p
- 유저가 submit 버튼을 클릭시 action="writeProcess" 에 의해 BoardWriteController.java로 연결된다.
- webapp/WEB-INF/board/writeForm.jsp

```
<form name="writeForm" action="writeProcess" id="writeForm"
                                        class="row g-3 border-primary" method="post">
  <input type="submit" value="등록하기" class="btn btn-primary"/>
```

3) 게시 글쓰기 폼 유효성 검사 자바스크립트 추가 20p
- webapp/js/formcheck.js

```
$(function() {
  $("#writeForm").on("submit", function() {
    if($("#writer").val().length <= 0) {
      alert("작성자가 입력되지 않았습니다.\n작성자를 입력해주세요");
      $("#writer").focus();
      return false;
    }
    if($("#title").val().length <= 0) {
      alert("제목이 입력되지 않았습니다.\n제목을 입력해주세요");
      $("#title").focus();
      return false;
    }
    if($("#pass").val().length <= 0) {
      alert("비밀번호가 입력되지 않았습니다.\n비밀번호를 입력해주세요");
      $("#pass").focus();
      return false;
    }
    if($("#content").val().length <= 0) {
      alert("내용이 입력되지 않았습니다.\n내용을 입력해주세요");
      $("#content").focus();
      return false;
}});
```

4) DAO 클래스에 게시 글을 등록하는 메서드 추가 21p
- com.jspstudy.bbs.dao.BoardDao

```
public void insertBoard(Board board) {
  String sqlInsert = "INSERT INTO jspbbs(no, title, writer, content," // 쿼리 변수
                                      + " reg_date, read_count, pass, file1) "
                                      + " VALUES(jspbbs_seq.NEXTVAL, ?, ?, ?, SYSDATE, 0, ?, ?)";
// 타 함수들은 Board board = null; 객체에 담아 View 로 전달하였으나, insert의 경우 DB에 바로 저장할 것이기에 객체를 생성하지도 전달하지도 않음
  try {
    conn = ds.getConnection();  // DB 커넥션 초기화
    pstmt = conn.prepareStatement(sqlInsert); // DB 쿼리 발행 객체 초기화
    pstmt.setString(1, board.getTitle());  // placeholder 지정
    pstmt.setString(2, board.getWriter());
    pstmt.setString(3, board.getContent());
    pstmt.setString(4, board.getPass());
    pstmt.setString(5, board.getFile1());
    pstmt.executeUpdate();  // DB 쿼리 결과 저장 객체 초기화
  } catch(Exception e) { e.printStackTrace();
  } finally {
    try {
      if(pstmt != null) pstmt.close();
      if(conn != null) conn.close();
    } catch(SQLException se) {}
  }
} // end indertBoard(Board board);
```

5) 게시 글 등록하기 요청을 처리하는 컨트롤러 클래스
- com.jspstudy.bbs.controller > BoardWriteController.java

```
@WebServlet("/writeProcess")
public class BoardWriteController extends HttpServlet {
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      request.setCharacterEncoding("utf-8");
// 사용자가 입력한 데이터를 HttpServletRequest 객체를 통해 파라미터를 읽어 변수에 저장한다.
      String title = request.getParameter("title");
      String writer = request.getParameter("writer");
      String pass = request.getParameter("pass");
      String content = request.getParameter("content");
// 하나의 게시글 정보를 저장하는 자바빈 객체 생성, 요청 파라미터로 받은 데이터를 Board 객체 저장
      Board board = new Board();
      board.setTitle(title);
      board.setWriter(writer);
      board.setPass(pass);
      board.setContent(content);
      BoardDao dao = new BoardDao(); // BoardDao 객체 생성하고 게시 글을 DB에 추가한다.
      dao.insertBoard(board);
// 게시 글쓰기와 같이 DB 입력 작업이 연동되는 경우 사용자의 새로 고침(F5) 동작에 의해 동일한 요청이 다시 발생하여 DB에 입력되는 데이터의 중복이 발생하거나 SQLException을 발생 시킬 수 있어 Redirect 기법을 사용한다. 이외에 다른 사이트로 이동시킬 때 Redirect 기법을 사용 한다.
      response.sendRedirect("boardList");
    }
}
```

5. 게시글 수정하기
- 테이블에서 게시 글을 유일하게 구분할 수 있는 데이터(no)를 요청 파라미터로 보내야 한다

- 게시글번호(no)와 연관된 데이터를 호출, 수정할 내용을 update한다.

1) 게시글 수정 폼 요청에 사용할 숨김 form 추가 24p
- webapp/WEB-INF/board/boardDetail.jsp

```
<form name="checkForm" id="checkForm">
  <input type="hidden" name="no" id="no" value="${ board.no }"/>
  <input type="hidden" name="pass" id="rPass" />
</form>
<input class="btn btn-warning" type="button" id="detailUpdate" value="수정하기"/>
```

2) 게시글 상세 페이지에서 게시글 수정 폼 요청 자바스크립트 추가 25p
- boardDetail.jsp에서 수정하기 버튼이 클릭되면 비밀번호 입력란에 입력된 데이터를 읽어와 checkForm의 hidden 컨트롤인 비밀번호 입력란에 추가하고 서버로 submit 하는 자바스크립트 코드
- webapp/js/formcheck.js

```
$("#detailUpdate").on("click", function() { //업데이트 버튼 클릭시
  var pass = $("#pass").val();
  if(pass.length <= 0) { // pass에 값이 없으면, 비밀번호 입력 요청
    alert("게시 글을 수정하려면 비밀번호를 입력해주세요");
    return false; }
  $("#rPass").val(pass); //#rPass에 변수 pass값을 입력
// #checkForm의 action 속성을 updateForm로 변경, 컨트롤러에서 어노테이션으로 인지, 처리된다.
  $("#checkForm").attr("action", "updateForm");
  $("#checkForm").attr("method", "post"); //#checkForm의 action 속성을 updateForm로, 없으면 생성
  $("#checkForm").submit();
});
```

3) DAO 클래스에 게시글 비밀번호를 체크하는 메서드 추가 26p
- com.jspstudy.bbs.dao.BoardDao

```
public boolean isPassCheck(int no, String pass) {
  String sqlPass = "SELECT pass FROM jspbbs WHERE no=?";
  boolean isPass = false;
    try {
    conn = ds.getConnection(); // DB 커넥션 초기화
    pstmt = conn.prepareStatement(sqlBoardList); // DB 쿼리 발행 객체 초기화
    pstmt.setInt(1,  no); // placeholder 지정
    rs = pstmt.executeQuery();  // DB 쿼리 결과 저장 객체 초기화
    if(rs.next()) { // 쿼리 결과에 담을 데이터들, 쿼리결과로 no에 해당하는 pass를 가진 데이터
      isPass = rs.getString(1).equals(pass);} // rs(DB)의 pass 와 #pass가 동일한지
    } catch(SQLException e) {e.printStackTrace();
    } finally {
      try {
        if(rs != null) rs.close();
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
      } catch(SQLException e) {}
    }
    return isPass;
} // end isPassCheck();
```

4) 게시글 수정 폼 요청을 처리하는 컨트롤러 클래스 28p

- boardDetail.jsp에서 수정하기 버튼이 클릭되면 비밀번호 입력란에 입력된 데이터를 읽어와 hidden속성인 checkForm의 pass가 저장되고, action도 updateForm이 되어 컨트롤러가 인지된다.
- 컨트롤러 클래스는 저장된 게시글번호(no)와 비밀번호(pass)를 받게 되는데, BoardDAO의 **boolean** isPassCheck 메서드를 통해 T/F 여부를 확인, T일경우 게시글번호(no)를 board 인스턴스에 담아서 updateForm.jsp 로 전송한다.
- com.jspstudy.bbs.controller.BoardUpdateFormController

```
@WebServlet("/updateForm")
public class BoardUpdateFormController extends HttpServlet {
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      request.setCharacterEncoding("utf-8");
      String sNo = request.getParameter("no");
      String pass = request.getParameter("pass");
      BoardDao dao = new BoardDao();
      int no = Integer.parseInt(sNo);  // String no 를 Int no 로 저장
      boolean isPassCheck = dao.isPassCheck(no, pass);  //BoardDao의 isPassCheck에 T/F 받음
      if(!isPassCheck) { //False 면 얼러트 , history.back(), return
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println("<script>");
        out.println("alert('비밀번호가 다릅니다.');");
        out.println("history.back();");
        out.println("</script>");
        out.close();
        return;
      }
      Board board = dao.getBoard(no);   // 비번이 True면 no번의 게시글을 board에 담아 전송준비
      request.setAttribute("board", board);  // 요청결과 데이터를 HttpServletRequest의 속성에 저장
      RequestDispatcher rd = request.getRequestDispatcher("/WEB-INF/board/updateForm.jsp");
      rd.forward(request, response);
    }}
```

5) 게시 글 수정 폼 뷰 페이지 29p
- 비밀번호가 일치될 경우 게시글번호(no)를 통해 no와 관련된 데이터가 표현된다.
- 게시 글을 수정하기 위해서는 테이블에서 게시 글을 유일하게 구분할 수 있는 데이터가 필요하다. 아래에서 no는 테이블에서 하나의 게시 글을 유일하게 구분할 수 있는 데이터로 아래 폼이 서버로 전송될 때 이 no도 같이 서버로 전송되어야 게시 글 정보를 제대로 수정할 수 있기 때문에 화면에는 보이지 않고 폼이 전송될 때 요청 파라미터에 추가될 수 있도록 hidden 폼 컨트롤로 폼에 추가하였다.
- 아래폼은 submit 할 경우 action="*updateProcess*" 에 의해 BoardUpdateController.java로 전송된다.
- webapp/WEB-INF/board/updateForm.jsp

```
<form name="updateForm" action="updateProcess" id="updateForm"
  class="row g-3 border-primary" method="post">
  <input type="hidden" name="no" value="${board.no}">
  <div class="col-4 offset-md-2">
    <label for="writer" class="form-label">글쓴이</label>
    <input type="text" class="form-control" name="writer"  id="writer"
     placeholder="작성자를 입력해 주세요" value="${board.writer}">
  </div>
  <div class="col-4 ">
    <label for="pass" class="form-label">비밀번호</label>
    <input type="password" class="form-control" name="pass" id="pass">
  </div>
```

```
  <div class="col-8 offset-md-2">
    <label for="title" class="form-label">제 목</label>
    <input type="text" class="form-control" name="title" id="title" value="${board.title}">
  </div>
  <div class="col-8 offset-md-2">
    <label for="content" class="form-label">내 용</label>
    <textarea class="form-control" name="content" id="content" rows="10">${board.content}
    </textarea>
  </div>
  <div class="col-8 offset-md-2 text-center mt-5">
    <input type="submit" value="수정하기" class="btn btn-primary"/>  
    <input type="button" value="목록보기" onclick="location.href='boardList'" class="btn
      btn-primary"/>
  </div>
</form>
```

6) 게시 글 수정 폼 유효성 검사 자바스크립트 추가 31p

- webapp/js/formcheck.js

```
$("#updateForm").on("submit", function() {
  if($("#writer").val().length <= 0) {
    alert("작성자가 입력되지 않았습니다.\n작성자를 입력해주세요");
    $("#writer").focus();
    return false;
  }
  if($("#title").val().length <= 0) {
    alert("제목이 입력되지 않았습니다.\n제목을 입력해주세요");
    $("#title").focus();
    return false;
  }
  if($("#pass").val().length <= 0) {
    alert("비밀번호가 입력되지 않았습니다.\n비밀번호를 입력해주세요");
    $("#pass").focus();
    return false;
  }
  if($("#content").val().length <= 0) {
    alert("내용이 입력되지 않았습니다.\n내용을 입력해주세요");
    $("#content").focus();
    return false;
  }
});
```

7) DAO 클래스에 게시 글을 수정하는 메서드 추가 32p

- updateForm.jsp 의 수정하기 버튼 클릭시 게시글을 DB에 수정하는 메서드
- com.jspstudy.bbs.dao.BoardDao

```
public void updateBoard(Board board) {
  String sqlUpdate = "UPDATE jspbbs set title=?, writer=?, content=?,"
                                    + " reg_date=SYSDATE, file1=? WHERE no=?";
  try {
    conn = ds.getConnection();
    pstmt = conn.prepareStatement(sqlUpdate);
    pstmt.setString(1, board.getTitle());
    pstmt.setString(2, board.getWriter());
```

```
      pstmt.setString(3, board.getContent());
      pstmt.setString(4, board.getFile1());
      pstmt.setInt(5, board.getNo());
      pstmt.executeUpdate();
    } catch(Exception e) { e.printStackTrace();
    } finally {
      try {
        if(pstmt != null) pstmt.close();
        if(conn != null) conn.close();
      } catch(SQLException se) {}
    }} // end updateBoard(Board board);
```

8) 게시 글 수정하기 요청을 처리하는 컨트롤러 클래스 33p

- com.jspstudy.bbs.controller.BoardUpdateController

```
@WebServlet("/updateProcess")
public class BoardUpdateController extends HttpServlet {
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      request.setCharacterEncoding("utf-8");
      String pass= null, title = null, writer = null, content = null;
      int no = 0;
      no = Integer.parseInt(request.getParameter("no"));
      pass = request.getParameter("pass");
      BoardDao dao = new BoardDao();
      boolean isPassCheck = dao.isPassCheck(no, pass);
      if(! isPassCheck) {
        StringBuilder sb = new StringBuilder(); //문자열을 효율적처리를 위해 StringBuilder 객체 이용
        sb.append("<script>");
        sb.append("alert('비밀번호가 맞지 않습니다.');");
        sb.append("history.back();");
        sb.append("</script>");
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println(sb.toString());
        System.out.println("비밀번호 맞지 않음");
        return;
      }
// 비밀번호가 맞으면 사용자가 게시글 수정 폼에 입력한 데이터를 읽어온다.
      title = request.getParameter("title");
      writer = request.getParameter("writer");
      content = request.getParameter("content");
// 하나의 게시 글 정보를 저장하는 자바빈 객체를 생성하고 파라미터로 넘겨받은 요청 데이터를 Board
객체에 저장한다.
      Board board = new Board();
      board.setNo(no);
      board.setTitle(title);
      board.setWriter(writer);
      board.setPass(pass);
      board.setContent(content);
      dao.updateBoard(board);   //BoardDao의 updateBoard() 에서 pstmt.executeUpdate(); 실행된다.
      response.sendRedirect("boardList");}}
```

6. 게시글 삭제하기

1) DAO 클래스에 게시 글을 삭제하는 메서드 추가

```
public void deleteBoard(int no) {
  String sqlDelete = "DELETE FROM jspbbs WHERE no=?";
  try {
    conn = ds.getConnection();
    pstmt = conn.prepareStatement(sqlDelete);
    pstmt.setInt(1, no);
    pstmt.executeUpdate();
  } catch(Exception e) {e.printStackTrace();
  } finally {
    try {
      if(pstmt != null) pstmt.close();
      if(conn != null) conn.close();
    } catch(SQLException se) {}
  }} // end deleteBoard(int no); }
```

2) 게시 글 상세 페이지에서 게시 글 삭제 요청 자바스크립트 추가

```
$("#detailDelete").on("click", function() {
  var pass = $("#pass").val();
  if(pass.length <= 0) {
    alert("게시 글을 삭제하려면 비밀번호를 입력해주세요");
    return false;
  }
    $("#rPass").val(pass);
    $("#checkForm").attr("action", "deleteProcess");
    $("#checkForm").attr("method", "post");
    $("#checkForm").submit();
});
```

3) 게시 글 삭제하기 요청을 처리하는 컨트롤러 클래스

- com.jspstudy.bbs.controller.BoardDeleteController

```
@WebServlet("/deleteProcess")
public class BoardDeleteController extends HttpServlet {
  protected void doPost(HttpServletRequest request, HttpServletResponse response)
    throws ServletException, IOException {
      request.setCharacterEncoding("utf-8");
      String sNo = request.getParameter("no");
      String pass = request.getParameter("pass");
      int no = Integer.parseInt(sNo);
      BoardDao dao = new BoardDao();
      boolean isPassCheck = dao.isPassCheck(no, pass);
      if(! isPassCheck) {
        StringBuilder sb = new StringBuilder();
        sb.append("<script>");
        sb.append("alert('비밀번호가 맞지 않습니다.');");
        sb.append("history.back();");
        sb.append("</script>");
        response.setContentType("text/html; charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println(sb.toString());
        System.out.println("비밀번호 맞지 않음");
```

```
            return;
        }
        dao.deleteBoard(no); //boardDao 에 의해 삭제 실행
        response.sendRedirect("boardList");
}}
```