Appendix

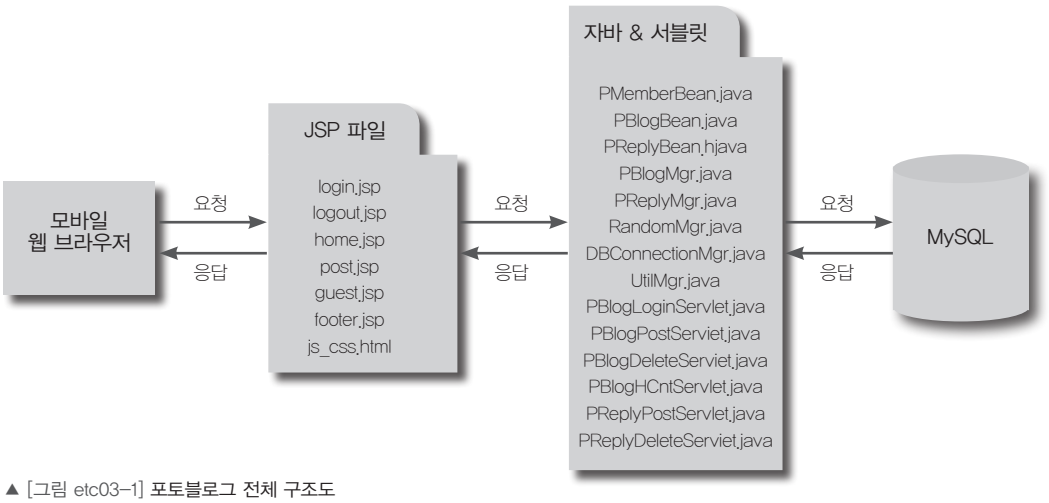
# 03

# 모바일웹 PhotoBlog

본 교재의 대부분은 예제는 PC 버전 화면 중심의 예제입니다. 이번 예제는 처음으로 모바일웹 중심의 예제로 만들어 보았습니다.

이번 장에서는 모바일웹 버전으로 PhotoBlog 구현을 하도록 하겠습니다. 포토블로그는 회원 및 포토포스트 sample data 입력, 회원 로그인, 포토포스트 올리기, '좋아요', 포토포스트 리스트, 포토포스트 삭제, 댓글 달기, 댓글 삭제, 댓글 리스트 기능을 구현하도록 하겠습니다.

## 포토블로그 전체적인 구조




▲ [그림 etc03-1] 포토블로그 전체 구조도

**소스목록**

```
JSP&HTML  - login.jsp (로그인 페이지)
          - logout.jsp (로그아웃 페이지)
          - home.jsp (포토블로그 메인 페이지)
          - post.jsp (포토 및 메시지 입력 페이지)
          - guest.jsp (포토블로그 게스트 리스트 페이지)
          - footer.jsp (모든 JSP 화면 하단 페이지)
          - js_css.html (자바스크립트 및 CSS 페이지)
-------------------------------------------------------------------
서블릿    - PBlogLoginServlet.java (로그인 서블릿)
          - PBlogPostServlet.java (포토포스트 저장 서블릿)
          - PBlogDeleteServlet.java (포토포스트 삭제 서블릿)
          - PBlogHCntServlet.java (포토포스트 '좋아요' 서블릿)
          - PReplyPostServlet.java (댓글 저장 서블릿)
          - PReplyDeleteServlet.java (댓글 삭제 서블릿)
-------------------------------------------------------------------
자바&빈즈 - PMemberBean.java (회원 테이블 빈즈)
          - PBlogBean.java (포토블로그 테이블 빈즈)
          - PReplyBean.java (댓글 테이블 빈즈)
          - PBlogMgr.java (포토블로그 SQL 실행 자바)
          - PReplyMgr.java (댓글 SQL 실행 자바)
          - RandomMgr.java (회원 및 포토포스트 sample data 입력 자바)
          - DBConnectionMgr.java (데이터베이스 Connectionpool 자바)
          - UtilMgr.java (유틸 기능 자바)
```

**포토블로그의 기능**

- 회원 로그인 기능이 있습니다.
- 회원 로그아웃 기능이 있습니다.
- 포토포스트와 메시지 저장 기능이 있습니다.
- 포토포스트 리스트를 보여줍니다.
- 포토포스트 '좋아요' 기능이 있습니다.
- 포토포스트 삭제 기능이 있습니다.
- 포토포스트에 댓글 저장 기능이 있습니다.
- 포토포스트에 댓글 리스트를 보여줍니다.
- 포토포스트에 댓글 삭제 기능이 있습니다.

# 01 _ 데이터베이스 설계

먼저 프로그래밍에 앞서 필요한 SQL 테이블을 만들어 보겠습니다. 포토블로그에 필요한 테이블은 3개입니다. 첫 번째는 회원테이블 두 번째는 포토블로그 테이블 세 번째는 댓글 테이블입니다.

| 칼럼명 | 데이터타입 | 설명 |
|---|---|---|
| id | char(20) – primary key | 회원의 id를 저장하는 칼럼입니다. id는 유일한 값이므로 주키(primary key)로 지정하였습니다. |
| pwd | char(20) | 회원의 비밀번호를 저장하는 칼럼입니다. |
| name | char(20) | 회원의 이름을 저장하는 칼럼입니다. |
| profile | char(20) | 회원의 프로필 포토를 저장하는 칼럼입니다. |

▲ [표 etc03-1] 회원 테이블 (tblPMember)

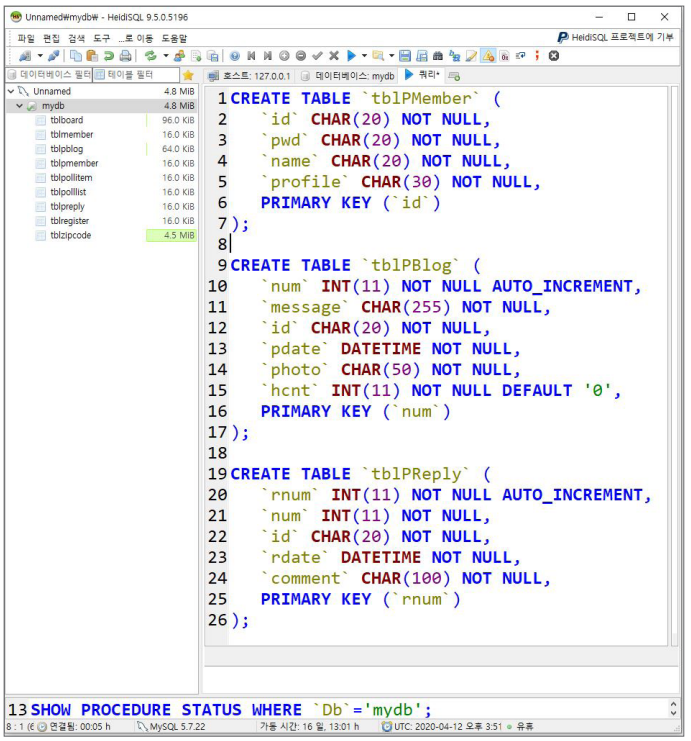
| 칼럼명 | 데이터타입 | 설명 |
|---|---|---|
| num | int(11) – primary key auto_increment | 포토포스트의 유일성을 보장하는 번호를 저장하는 칼럼입니다. num은 유일한 값이므로 주키(primary key)와 자동적으로 증가하는 auto_increment 속성으로 지정하였습니다. |
| message | char(255) | 포토포스트의 메세지를 저장하는 칼럼입니다. |
| id | char(20) | 포토포스트를 저장한 id를 저장하는 칼럼입니다. |
| pdate | datetime | 포토포스트를 저장한 날짜와 시간을 저장하는 칼럼입니다. |
| photo | char(20) | 포토포스트의 photo 명을 저장하는 칼럼입니다. |
| hcnt | int(11) | 포토포스트의 '좋아요' count를 저장하는 칼럼입니다. |

▲ [표 etc03-2] 포토블로그 테이블 (tblPBlog)

| 칼럼명 | 데이터타입 | 설명 |
|---|---|---|
| num | int(11) – primary key auto_increment | 댓글의 유일성을 보장하는 번호를 저장하는 칼럼입니다. rnum은 유일한 값이므로 주키(primary key)와 자동적으로 증가하는 auto_increment 속성으로 지정하였습니다. |
| num | int(11) | 포토포스트의 num를 저장하는 칼럼입니다. 어떤 포토포스트의 댓글인지 구분되는 값입니다. |
| id | char(20) | 댓글을 저장한 id를 저장하는 칼럼입니다. |
| rdate | datetime | 댓글을 저장한 날짜와 시간을 저장하는 칼럼입니다. |
| comment | char(100) | 댓글의 내용을 저장하는 칼럼입니다. |

▲ [표 etc03-3] 포토블로그 댓글 테이블 (tblPReply)

위의 명세표대로 다음과 같이 SQL문를 작성하고 F9 키를 누르면 10장에서 만들어 놓은 mydb 데이터베이스에 3개의 테이블이 생성됩니다.




▲ [그림 etc03-2] HeidiSQL 테이블 만들기

## 02 _ 설계 및 구현

포토블로그는 회원 로그인, 포토포스트 올리기, 포토포스트 리스트, 포토포스트 삭제, '좋아요', 댓글 달기, 댓글 삭제, 댓글 리스트 등의 기능들이 있습니다. 한 번에 모든 기능을 구현하는 것보다는 단계적으로 흐름에 맞게 구현하는 방법이 처음 JSP 프로그래밍을 하는 입장에서는 더 좋은 방법이라고 생각합니다. 지금부터 2단계로 나누어서 첫 번째 단계는 회원 로그인, 포토포스트 올리기, 포토포스트 리스트, 포토포스트 삭제를 구현하고 두 번째 단계는 '좋아요', 댓글 달기, 댓글 삭제, 댓글 리스트를 단계적으로 구현하겠습니다. 그리고 회원가입 기능은 따로 구현하지 않았습니다. 만약 회원가입 기능이 필요하다면 14장을 참고하면 될 것 같습니다. 먼저 포토블로그를 구현하기 전에 10명의 회원과 100개의 포토포스트를 사진과 함께 sample data 작업을 먼저 하도록 하겠습니다. 그리고 100개의 사진은 소스와 함께 제공 됩니다.

**TIP**

이번 장은 필요한 내용 위주의 파일에 대한 설명을 하고 앞 장에서 설명한 중복적인 내용은 생략하도록 하겠습니다. 만약 생각나지 않거나 어렵다고 생각을 하면 앞 장의 내용을 참고 하시길 바랍니다.

설명이 생략되는 파일 리스트

- footer.jsp
- js_css.html
- DBConnectionMgr.java
- PMemberBean.java
- PBlogBean.java
- PReplyBean.java
- UtilMgr.java

## 02-1 회원 및 포토포스트 sample data 입력

포토블로그에 사용되는 10명의 회원과 회원마다 각각 10개의 포토포스트를 입력

**01** tblPMember와 tblPBlog 저장하기 위해 다음과 같이 작성하고 저장합니다.

source/etc03/RandomMgr.java

```
01 : package etc03;
02 :
03 : import java.sql.Connection;
04 : import java.sql.PreparedStatement;
05 : import java.sql.ResultSet;
06 :
07 : public class RandomMgr {
08 :
09 :    private DBConnectionMgr pool;
10 :    String id[] = {"a100","a101","a102","a103","a104","a105","a106","a107","a108","a109"};
11 :        String pwd = "1234";
12 :        String name[] = {"유재석","이광수","하동훈","김종국","전소민","지석진","강개리","양세
찬","송지효","양세형"};
13 :
14 :        public RandomMgr() {
15 :          try {
16 :            pool = DBConnectionMgr.getInstance();
17 :          } catch (Exception e) {
18 :             e.printStackTrace();
19 :          }
20 :        }
21 :
22 :        //회원 10명 입력하는 메소드입니다.
23 :        public void postPMember() {
24 :           Connection con = null;
25 :           PreparedStatement pstmt = null;
```

샘플로 회원 id 10개를 배열로 선언합니다.

샘플로 저장되는 모든 회원의 비밀번호로 선언합니다.

샘플로 입력되는 회원 이름 10개를 배열로 선언합니다.

```
26 :            String sql = null;
27 :            try {
28 :              con = pool.getConnection();
29 :              sql = "insert tblPMember(id,pwd,name,profile)values(?,?,?,?)";
30 :              pstmt = con.prepareStatement(sql);
31 :              for (int i = 0; i < id.length; i++) {
32 :              pstmt = con.prepareStatement(sql);
33 :              pstmt.setString(1, id[i]);
34 :              pstmt.setString(2, pwd);
35 :              pstmt.setString(3, name[i]);
36 :              pstmt.setString(4, "profile"+(i+1)+".jpg");
37 :              pstmt.executeUpdate();
38 :          }
39 :          System.out.println("회원 10개 입력 완료");
40 :          } catch (Exception e) {
41 :              e.printStackTrace();
42 :          } finally {
43 :            pool.freeConnection(con, pstmt);
44 :            }
45 :         }
46 :
47 :         //회원 1명당 10개의 포토포스트 입력 메소드입니다.
48 :         public void postPBlog() {
49 :            Connection con = null;
50 :            PreparedStatement pstmt = null;
51 :            ResultSet rs = null;
52 :            String sql = null;
53 :            try {
54 :            con = pool.getConnection();
55 :            sql = "select id from tblPMember limit 0, 10";
56 :            pstmt = con.prepareStatement(sql);
57 :            rs = pstmt.executeQuery();
58 :            sql = "insert tblPBlog(id,message,pdate,photo)values(?,?,now(),?)";
59 :            int cnt = 1;
60 :            while(rs.next()) {
61 :                 String id = rs.getString(1);
62 :              for (int i = 0; i < 10; i++) {
63 :                 pstmt = con.prepareStatement(sql);
64 :                 pstmt.setString(1, id);
65 :                 pstmt.setString(2, id + " " + (i+1)+" 메세지입니다.");
66 :                 pstmt.setString(3, "photo"+cnt+".jpg");
67 :                 pstmt.executeUpdate();
68 :                 cnt++;
69 :                  }
70 :            }
71 :             System.out.println("회원 1명당 10개 포토포스트 입력 완료");
```

배열로 준비된 id 값 10개가 테이블에 저장되기 위해 for문을 선언합니다.

저장되는 회원 프로필 포토명이 profile1.jpg 이런 형식으로 저장됩니다. 회원 프로필 포토도 sample data로 제공이 됩니다.

10개 입력이 완료되면 Console에 출력되는 성공 메시지입니다.

저장된 회원 id 10개를 가져오는 SQL문 입니다.

포토포스트를 저장하기 위한 SQL문 입니다.

id당 10개의 포토포스트를 입력하기 위해 for문을 선언합니다.

message는 간단하게 'a1011 메시지입니다.' 이런 형식으로 저장됩니다.

저장되는 포토명은 photo1.jpg에서 photo100.jpg로 저장이 됩니다. 100개의 포토는 sample data로 제공이 됩니다.

100개 포토포스트 입력이 완료되면 Console에 출력되는 성공 메시지입니다.

```
72 :         } catch (Exception e) {
73 :            e.printStackTrace();
74 :         } finally {
75 :            pool.freeConnection(con, pstmt, rs);
76 :         }
77 :       }
78 :
       자바 Application 실행을 위해 main 메소드를 선언합니다.
79 :    public static void main(String[] args) {
80 :         RandomMgr mgr = new RandomMgr();
81 :         mgr.postPMember();   <- 10명의 회원 정보가 저장되는 메소드를 호출합니다.
82 :         mgr.postPBlog();
83 :    }
84 : }        100개의 포토포스트가 저장되는 메소드를 호출합니다.
```

**여기서 잠깐!**

RandomMgr.java는 main 메소드가 있는 자바 파일이기 때문에 실행을 하기 위해서는 이클립스에서 Run As –〉 Java Application으로 실행을 해야 합니다.

결과

```
회원 10개 입력 완료
회원 1명당 10개 포토포스트 입력 완료
```

**23 ~ 45** : postPMember() 메소드는 매개변수가 없고 void 타입이므로 반환 값은 없습니다. 11~13라인에 선언한 id와 pwd 그리고 name 값을 10번의 insert문 실행을 통해 sample data를 입력하는 메소드입니다. 프로필 포토 명은 profile1.jpg부터 profile10.jpg으로 입력되고 profile 포토는 소스랑 같이 같이 제공됩니다.

**48 ~ 77** : postPBlog() 메소드는 매개변수가 없고 void 타입이므로 반환 값은 없습니다. postPMember() 메소드 통해서 입력한 10개의 id를 가져와서 id 당 10개의 포토포스트를 입력하는 메소드입니다. 포토명은 photo1.jpg부터 photo100.jpg 입력되고 sample photo 포토는 소스랑 같이 같이 제공됩니다.

**79 ~ 83** : 위에 선언한 메소드 두 개를 실행하기 위해 RandomMgr 객체를 생성하고 postPMember()와 postPBlog() 메소드는 순서대로 실행해야 합니다. sample data는 한 번만 실행해야 합니다. 만약 두 번 실행 한다면 tblPMember과 tblPBlog table의 PK(주키) 속성 위배로 인해 에러가 발생 됩니다.

## 02-2 회원 로그인, 포토포스트 올리기, 리스트, 삭제 만들기

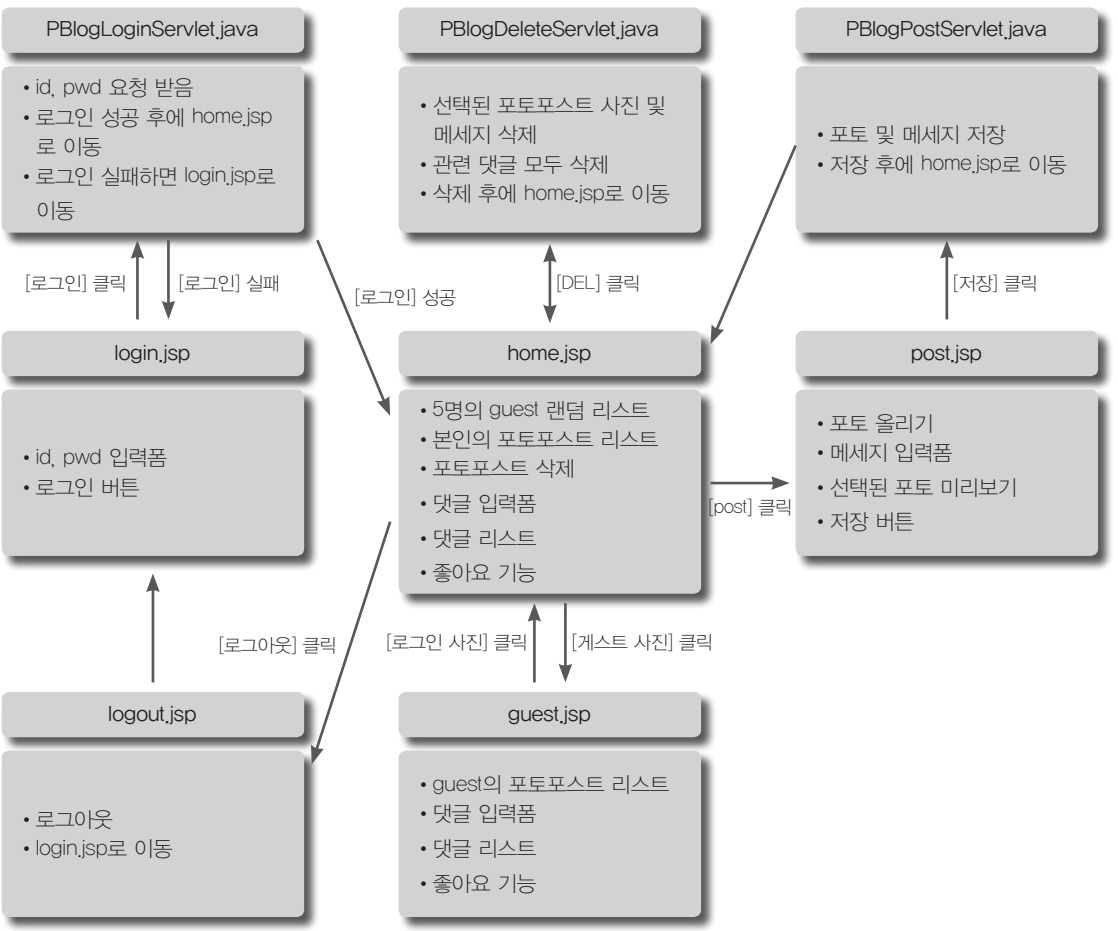
포토블로그의 가장 기본적인 기능은 회원 로그인, 포토포스트의 올리기, 읽기(리스트), 삭제입니다. 첫 번째 단계에서 이러한 기본적인 기능들을 잘 구현한다면 두 번째 단계의 기능인 댓글 달기, 댓글 읽기, 댓글 삭제, '좋아요' 기능은 그렇게 어렵지는 않을 겁니다.

먼저 로그인 기능입니다. login.jsp 페이지에서 id와 pwd를 입력하고 PBlogLoginServlet 서블릿에서 데이터베이스와 연동하여 로그인을 처리하는 부분입니다. PBlogLoginServlet 서블릿에서 로그

인을 성공하면 자동적으로 포토블로그 페이지 home.jsp 페이지로 이동합니다. home.jsp 페이지는 포토포스트 리스트 및 포토포스트 삭제 그리고 로그인 사용자를 제외하고 랜덤한 다섯 명의 포토블로그(guest.jsp)에 접속을 할 수 있는 기능을 제공하는 페이지입니다. home.jsp에서 상단 오른쪽에 있는 대문자[P]를 클릭하면 post.jsp로 이동합니다.

post.jsp는 스마트폰에 저장되는 있는 이미지 및 촬영을 통한 포토를 메시지와 함께 전송하는 기능의 페이지입니다. 전송된 포토와 메시지는 PBlogPostServlet 서블릿에서 저장 기능을 처리하고 자동으로 home.jsp로 이동합니다.

home.jsp에서는 포토의 하단에 [DEL]을 클릭하면 포토포스트와 관련된 댓글이 함께 삭제가 됩니다. 삭제 기능은 PBlogDeleteServlet 서블릿에서 처리합니다. 여기까지가 첫 번째 단계에서 구현할 흐름입니다.
지금부터 흐름을 잘 생각하면서 본격적으로 포토블로그를 구현하겠습니다.

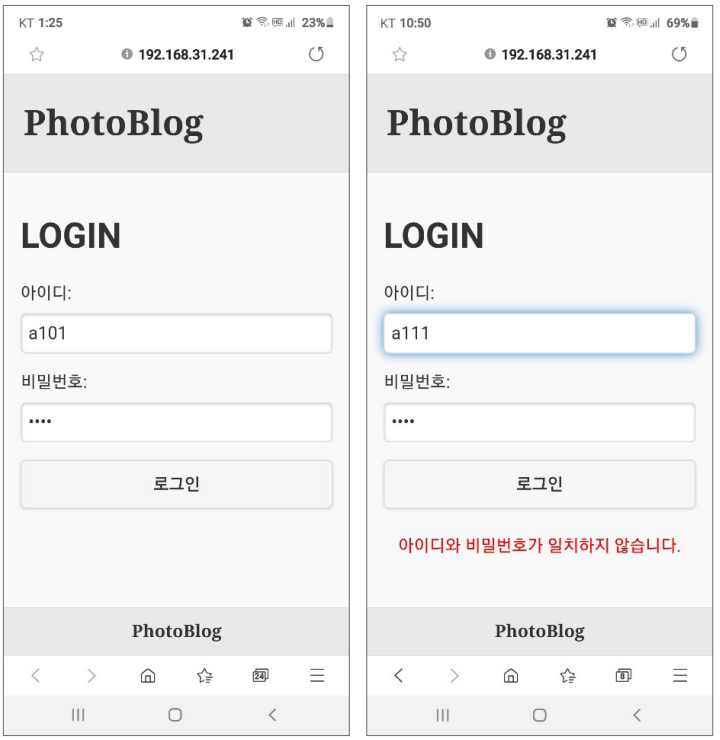



▲ [그림 etc03-3] 회원 로그인, 포토포스트 올리기, 리스트, 삭제 흐름도

## 회원 로그인 입력폼

**01** 회원 로그인 입력폼을 다음과 같이 작성하고 저장합니다.

- Preview – login.jsp 페이지

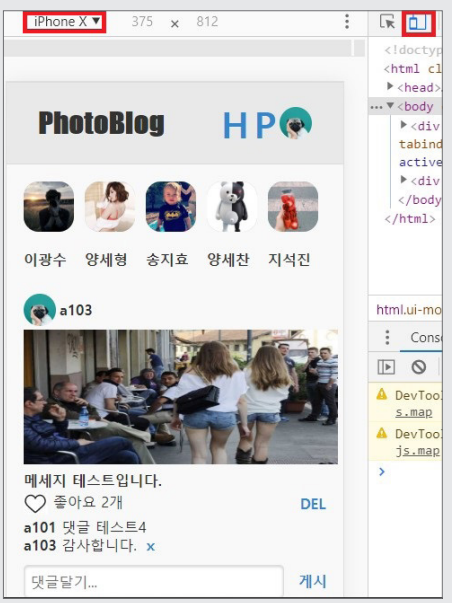



▲ [그림 etc03-4] login.jsp 로그인 입력폼 및 로그인 실패 실행화면

**여기서 잠깐!**

교재에서 보여주는 화면은 실제 스마트폰에서 실행한 화면입니다. 만약 스마트폰에서 실행하지 않고 PC 브라우저에서 실행한다면 다음과 같이 실행을 하시길 바랍니다.

브라우저에서 F12을 클릭하고 오른쪽 위에 있는 핸드폰 모양의 아이콘(Toggle device toolbar)을 선택하고 난 뒤에 보고 싶은 화면의 스마트폰 종류를 선택하면 됩니다.




▶ [그림 etc03-5] PC 브라우저에서 home.jsp 실행화면

source/etc03/login.jsp

```
01 : <%@page contentType="text/html; charset=EUC-KR"%>
02 : <%
03 :    String msg = (String)session.getAttribute("msg");
04 : %>
05 : <!DOCTYPE html>
06 : <html>
07 : <head>
08 :   <meta charset="EUC-KR"/>
09 :   <meta name="viewport" content="width=device-width, initial-scale=1"/>
10 :   <title>PhotoBlog</title>
11 :   <%@include file="js_css.html" %>
       화면 구성에 필요한 jQuery와 CSS로 구현된 파일을 include 지시자를 사용하여 가져옵니다.
12 :   <script type="text/javascript">
13 :     function login() {
14 :       document.frm.submit();
15 :     }
16 :   </script>
17 : </head>
18 : <body>
19 : <div data-role="page">
20 :   <div data-role="header">
21 :     <table>
22 :       <tr>
23 :         <td align="left" width="200">
24 :        <h1 style="font-family: fantasy;" align="left">  PhotoBlog</h1>
25 :         </td>
26 :       </tr>
27 :     </table>
28 :   </div>
29 :   <div data-role="content">
30 :   <h1>LOGIN</h1>
31 :    <form method="post" name="frm" action="pBlogLogin">
32 :      <p>
33 :       <label for="user_id">아이디:<input name="id"></label>
34 :      </p>
35 :      <p>
36 :        <label for="user_password">
37 :        비밀번호:<input type="password" name="pwd">
38 :        </label>
39 :      </p>
40 :        <input type="button" value="로그인" onclick="login()">
41 :    </form>
42 :   </div>
43 :   <%if(msg!=null){%>
```

PBlogLogServlet.java 서블릿에서 로그인 실패 시 세션에 저장한 메시지를 가져옵니다.

로그인에 필요한 id와 pwd를 입력받기 위한 form을 선언합니다.

id 값을 입력받기 위한 input 태그를 선언합니다. input 태그 type의 기본값은 text이므로 생략을 하였습니다.

pwd 값을 입력받기 위한 input 태그를 선언합니다.

로그인 실패 시 보이는 메시지입니다.

```
44 :     <div align="center"><font color="red"><%=msg%></font></div>
45 :   <%}%>
46 :   <%@include file="footer.jsp" %>
47 : </div>
48 : </body>
49 : </html>
```

포토블로그의 모든 화면의 하단을 담당하는 footer.jsp를 include 지시자로 가져옵니다.

**여기서 잠깐!**

JSP 및 서블릿을 설명하는 교재이므로 화면을 구성하는 jQuery와 CSS에 대한 설명은 생략하겠습니다. 만약 jQuery와 CSS 학습이 필요하신 분들은 다른 교재나 인터넷의 동영상 강좌를 참고하시길 바랍니다.

20 ~ 28 : 모든 화면의 header를 담당하는 div 영역의 코드입니다.

31 ~41 : 로그인을 위한 id와 pwd을 위한 form입니다. form의 메소드가 post이기 때문에 PBlogLoginServlet.java 서블릿에서 처리되는 메소드는 doPost 입니다. action값은 PBlogLoginServlet.java 서블릿의 URL 맵핑 값으로 선언된 값이므로 결과적으로 로그인 처리는 PBlogLoginServlet.java 서블릿에서 담당합니다.

43 ~ 45 : 로그인 실패 시 보이는 메시지 영역입니다. 처음 로그인 실행을 시도한다면 보이지 않는 메시지입니다. PBlogLoginServlet.java 서블릿에서 로그인 실패를 하면 세션에 msg를 저장하기 때문에 로그인 실패를 했을 경우 에만 보이는 메시지입니다.

**02** 로그아웃 페이지를 다음과 같이 작성하고 저장합니다.

source/etc03/logout.jsp

```
01 : <%@page contentType="text/html; charset=EUC-KR"%>
02 : <%
03 :         session.invalidate();
04 :         response.sendRedirect("login.jsp");
05 : %>
```

현재의 세션 무효화(제거)합니다.

현재의 세션이 서버에서 제거가 되고 다시 login.jsp로 이동합니다.

**03** 로그인 처리 서블릿을 다음과 같이 작성하고 저장합니다.

source/etc03/PBlogLoginServlet.java

```
01 : package etc03;
02 :
03 : import java.io.IOException;
04 : import javax.servlet.*;...
09 :
10 : @WebServlet("/etc03/pBlogLogin")
11 : public class PBlogLoginServlet extends HttpServlet 서블릿 상속 클래스 {
12 :
13 :    protected void doPost(HttpServletRequest request, HttpServletResponse response)
14 :             throws ServletException, IOException {
15 :        request.setCharacterEncoding("EUC-KR");
```

지면 관계상 다른 클래스의 import는 생략합니다.

서블릿 맵핑 이름을 선언합니다.

post.jsp에서 form의 method가 post이기 때문에 서블릿에서 doPost 메소드를 선언합니다.

login.jsp에서 요청 받은 문자열의 인코딩을 한글로 선언합니다.

```
16 :        PBlogMgr pMgr = new PBlogMgr();
17 :       String id = request.getParameter("id");
18 :       String pwd = request.getParameter("pwd");
19 :       String url = "login.jsp";
20 :       if(pMgr.loginPMember(id, pwd)) {
21 :         request.getSession().setAttribute("idKey", id);
22 :         url = "home.jsp";
23 :       }else {
24 :         request.getSession().setAttribute("msg", "아이디와 비밀번호가 일치하지 않습니다.");
25         }
26 :       response.sendRedirect("url");
27 :    }
28 : }
```

login.jsp에서 요청한 id와 pwd 값을 받습니다.

loginPMember 메소드는 일치하는 id와 pwd이면 true이고 일치하지 않으면 false로 리턴하는 메소드입니다.

로그인 실패 시 이동하게 되는 페이지입니다.

request 객체에서 session 객체를 리턴받고 session에 idKey 라는 name 값으로 로그인에서 입력한 id 값을 저장합니다.

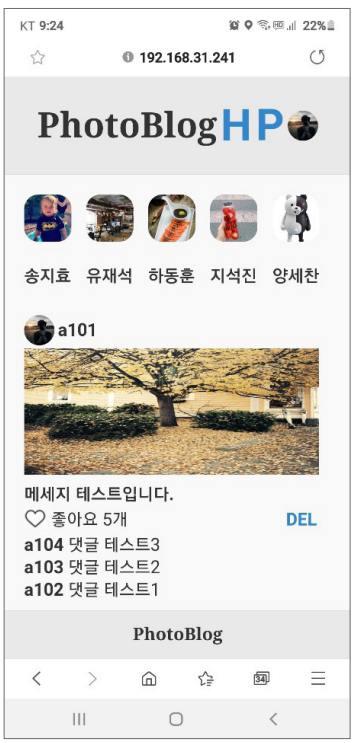
로그인 성공 시 이동하게 되는 페이지입니다.

request 객체에서 session 객체를 리턴 받고 session에 msg 라는 name 값으로 로그인 실패 메세지를 저장합니다.

로그인 결과에 따라서 url 변수에 적용된 페이지로 이동합니다.

**04** 포토블로그 리스트 페이지를 다음과 같이 작성하고 저장합니다.

- Preview – home.jsp 페이지




▲ [그림 etc03-6] home.jsp 실행화면

source/etc03/home.jsp

```
01 : <%@page contentType="text/html; charset=EUC-KR"%>
02 : <%@page import="java.util.Vector, etc03.*"%>
03 : <jsp:useBean id="pmgr" class="etc03.PBlogMgr"/>
04 : <jsp:useBean id="rmgr" class="etc03.PReplyMgr"/>
05 : <%
06 :   String id = (String)session.getAttribute("idKey");
07 :   if(id==null)
08 :     response.sendRedirect("login.jsp");
09 :   PMemberBean mbean = pmgr.getPMember(id);
10 :   Vector<PMemberBean> mvlist = pmgr.listPMember(id);
11 :   Vector<PBlogBean> pvlist = pmgr.listPBlog(id);
12 : %>
13 : <!DOCTYPE html>
14 : <html>
15 : <head>
16 :   <meta charset="EUC-KR"/>
17 :   <meta name="viewport" content="width=device-width, initial-scale=1"/>
18 :   <title>PhotoBlog</title>
19 :   <%@include file="js_css.html" %>
20 :   <script type="text/javascript">
21 :     function del(num) {
22 :       document.frm.action = "pBlogDelete";
23 :       document.frm.num.value=num;
24 :       document.frm.submit();
25 :     }
26 :     function rDel(rnum) {
27 :       document.frm.action = "pReplyDelete";
28 :       document.frm.rnum.value=rnum;
29 :       document.frm.submit();
30 :     }
31 :     function heart(num) {
32 :       document.frm.action = "pBlogUpHCnt";
33 :       document.frm.num.value=num;
34 :       document.frm.submit();
35 :     }
36 :     function cmtPost(num) {
37 :       document.frm.action = "pReplyPost";
38 :       cmt = document.getElementById("comment"+num);
39 :       document.frm.comment.value=cmt.value;
40 :       document.frm.num.value=num;
41 :       document.frm.id.value="<%=id%>";
42 :       document.frm.submit();
43 :     }
44 :     function goURL(url, gid) {
45 :       document.frm1.action=url;
```

포토포스트 및 댓글에 필요한 PBlogMgr과 PReplyMgr 객체를 생성합니다.

세션에서 id 값을 가져옵니다.

세션에서 가져온 id 값이 null이면 login.jsp로 이동합니다.

id를 매개변수로 getPMember 메소드를 호출하여 로그인 사용자의 정보를 가져옵니다.

로그인 사용자를 제외한 5명의 랜덤한 회원 리스트를 Vector 타입으로 가져옵니다.

로그인 사용자의 포토포스트 리스트를 Vector 타입으로 가져옵니다.

```
46 :       document.frm1.gid.value=gid;
47 :       document.frm1.submit();
48 :     }
49 :   </script>
50 : </head>
51 : <body>
52 : <div data-role="page" align="center">
53 :   <div data-role="header">
54 :     <table>
55 :       <tr>
56 :         <td align="left" width="200">
57 :        <h1 style="font-family: fantasy;" align="left">PhotoBlog</h1></td>
58 :         <td>
59 :           <a style="font-size:40px;" href="javascript:goURL('home.jsp','')">H</a>
60 :           <a style="font-size:40px;" href="javascript:goURL('post.jsp','')">P</a>
61 :         </td>
62 :         <td>
63 :        <div class="box" style="background: #BDBDBD;">
64 :        <img align="bottom" class="profile" src="photo/<%=mbean.getProfile()%>"
65 :        width="30" height="30" onclick="location='logout.jsp'">
66 :        </div>
67 :         </td>
68 :       </tr>
69 :     </table>
70 :   </div>
71 :   <div data-role="content">
72 :     <table>
73 :       <tr>
74 :       <%
75 :        for(int i=0;i<mvlist.size();i++){
76 :          PMemberBean mvbean = mvlist.get(i);
77 :        %>
78 :         <td width="80">
79 :        <div class="box1" style="background: #BDBDBD;">
80 :          <a href="javascript:goURL('guest.jsp','<%=mvbean.getId()%>')">
81 :          <img class="profile1" src="photo/<%=mvbean.getProfile()%>"
82 :           width="30" height="30">
83 :          </a>
84 :        </div>
85 :        <div>
86 :             <h4><%=mvbean.getName()%></h4>
87 :        </div>
88 :         </td>
89 :       <%}%>
90 :       </tr>
91 :     </table>
```

로그인 id의 프로필 포토가 보이고 포토를 클릭하면 로그아웃 페이지로 이동합니다.

프로필 포토를 클릭하면 클릭한 사용자의 포토블로그로 이동합니다.

```
92 :     <table>
93 :     <%
94 :        for(int i=0;i<pvlist.size();i++){
95 :        PBlogBean pbean = pvlist.get(i);
96 :        PMemberBean tmbean = pmgr.getPMember(pbean.getId());
97 :     %>
98 :       <tr>
99 :         <td width="30">
100 :       <div class="box" style="background: #BDBDBD;">
101 :         <img class="profile" src="photo/<%=tmbean.getProfile()%>"
102 :          width="30" height="30">
103 :       </div>
104 :        </td>
105 :       <td width="250"><b><%=tmbean.getId()%></b></td>
106 :       <td> </td>
107 :     </tr>
108 :     <tr>
109 :       <td colspan="3">
110 :      <img src="photo/<%=pbean.getPhoto()%>" width="350" height="150">
111 :       </td>
112 :     </tr>
113 :     <tr>
114 :      <td colspan="3"><b><%=pbean.getMessage()%></b></td>
115 :     </tr>
116 :     <tr>
117 :       <td colspan="2">
118 :      <a href="javascript:heart('<%=pbean.getNum()%>')">
119 :      <img src="img/heart.jpg" align="top"></a> 좋아요 <%=pbean.getHcnt() %>개</td>
120 :       <td align="center">
121 :      <a href="javascript:del('<%=pbean.getNum()%>')">DEL</a>
122 :       </td>
123 :     </tr>
124 :     <tr>
125 :       <td colspan="3" width="200">
126 :       <%
127 :      Vector<PReplyBean> rvlist = rmgr.listPReply(pbean.getNum());
128 :      for(int j=0;j<rvlist.size();j++){
129 :        PReplyBean rbean = rvlist.get(j);
130 :      %>
131 :        <b><%=rbean.getId()%></b> <%=rbean.getComment()%> 
132 :        <%if(id.equals(rbean.getId())){%>
133 :       <a href="javascript:rDel('<%=rbean.getRnum()%>')">x</a><%}%><br>
134 :        <%}%>
135 :       </td>
136 :     </tr>
137 :     <tr>
```

[DEL]을 클릭하면 포토포스트가 삭제됩니다.

매개변수는 포토포스트의 num 값으로 댓글 리스트를 Vector로 리턴 받습니다.

댓글을 입력자가 로그인한 사용자와 같은 id 라면 댓글을 삭제할 수 있는 [x]모양이 (활성화)보여집니다.

```
138 :         <td colspan="2">
139 :           <input id="comment<%=pbean.getNum()%>" placeholder="댓글달기..." size="50">
140 :         </td>
141 :         <td align="center">
142 :        <a href="javascript:cmtPost('<%=pbean.getNum()%>')">게시</a>
143 :         </td>
144 :       </tr>
145 :       <tr>
146 :         <td colspan="3"><br></td>
147 :       </tr>
148 :     <%}%>
149 :   </table>
150 :   </div>
151 :   <form method="post" name="frm">
152 :     <input type="hidden" name="num">
153 :     <input type="hidden" name="comment">
154 :     <input type="hidden" name="rnum">
155 :     <input type="hidden" name="id">
156 :   </form>
157 :   <form method="post" name="frm1" action="home.jsp">
158 :     <input type="hidden" name="gid">
159 :   </form>
160 :   <%@include file="footer.jsp" %>
161 : </div>
162 : </body>
163 : </html>
```

댓글을 입력하는 input 태그입니다.

하단 부분의 페이지 footer.jsp를 페이지 지시자로 가져옵니다.

**TIP**

meta viewport 태그는 애플이 아이폰, 아이패드 등 자사의 모바일 브라우저의 뷰포트(viewport) 크기 조절을 위해 만들었습니다. (뷰포트에 대한 설명은 본문 참조) meta viewport 태그는 W3C 명세에는 없기 때문에 표준은 아닙니다. 그러나 iOS 장치(아이폰 운영체제 브라우저 safari)가 널리 사용됨에 따라 사실상 표준처럼 사용되고 있고, 다른 브라우저들(Opera, Android, Mobile firefox(Fennec) 등)도 이 태그를 채택하게 됩니다.

- width=device-width : 웹 페이지의 크기가 모니터의 실제 크기를 따라가도록 만든 설정으로 모니터, 스마트폰 등의 화면에 맞는 비율로 화면이 뜨도록 돕습니다.
- initial-scale = 1 : 화면의 줌 정도를 1배율로 한다는 뜻으로 이 값을 크게 키우면 화면이 줌이 되어 크게 보입니다. 스마트 폰에서만 효과가 있습니다.

**21 ~ 25** : 151라인에 선언한 frm폼에 삭제하고 싶은 포토포스트의 num을 지정하고 pBlogDelete로 URL 맷핑 값으로 지정한 PBlogDeleteServlet.java 서블릿을 호출하는 자바스크립트 함수입니다. 이 함수는 121라인에서 호출합니다.

**26 ~ 30** : 151라인에 선언한 frm폼에 삭제하고 싶은 포토포스트 댓글을 rnum을 지정하고 pReplyDelete로 URL 맷핑 값으로 지정한 PReplyDeleteServlet.java 서블릿을 호출하는 자바스크립트 함수입니다. 이 함수는 133라인에서 호출합니다.

31 ~ 35 : 151라인에 선언한 frm폼에 '좋아요' 카운트를 증가하고 싶은 포토포스트 num을 지정하고 pBlogUpHCnt로 URL 맵핑 값으로 지정한 PBlogHCntServlet.java 서블릿을 호출하는 자바스크립트 함수입니다. 이 함수는 118라인에서 호출합니다.

36 ~ 43 : 151라인에 선언한 frm폼에 댓글 입력하고 싶은 포토포스트 num을 지정하고 pReplyPost로 URL 맵핑 값으로 지정한 PReplyPostServlet.java 서블릿을 호출하는 자바스크립트 함수입니다. 이 함수는 142라인에서 호출합니다. 139라인에 선언된 id값은 comment+num 값으로 동적으로 id값이 만들어집니다. home.jsp 페이지에 다수개의 포토포스트가 존재하기 때문에 id값을 동적으로 만들고 38라인에 댓글로 입력한 comment 값을 가져올수가 있습니다. 만약 입력폼이 전체 페이지에 정적으로 존재한다면 이런 기능이 필요가 없지만 동적으로 입력폼이 존재를 하면 이런 기능으로 구현을 해야 합니다.

44 ~ 48 : 157라인에 선언한 frm1폼에 url과 gid값을 지정하고 지정된 url값으로 gid 값과 같이 요청하여 이동합니다. gid 값은 상황에 따라 있을 수도 있고 없을 수도 있습니다. 이 함수는 59, 60, 80라인에서 호출합니다.

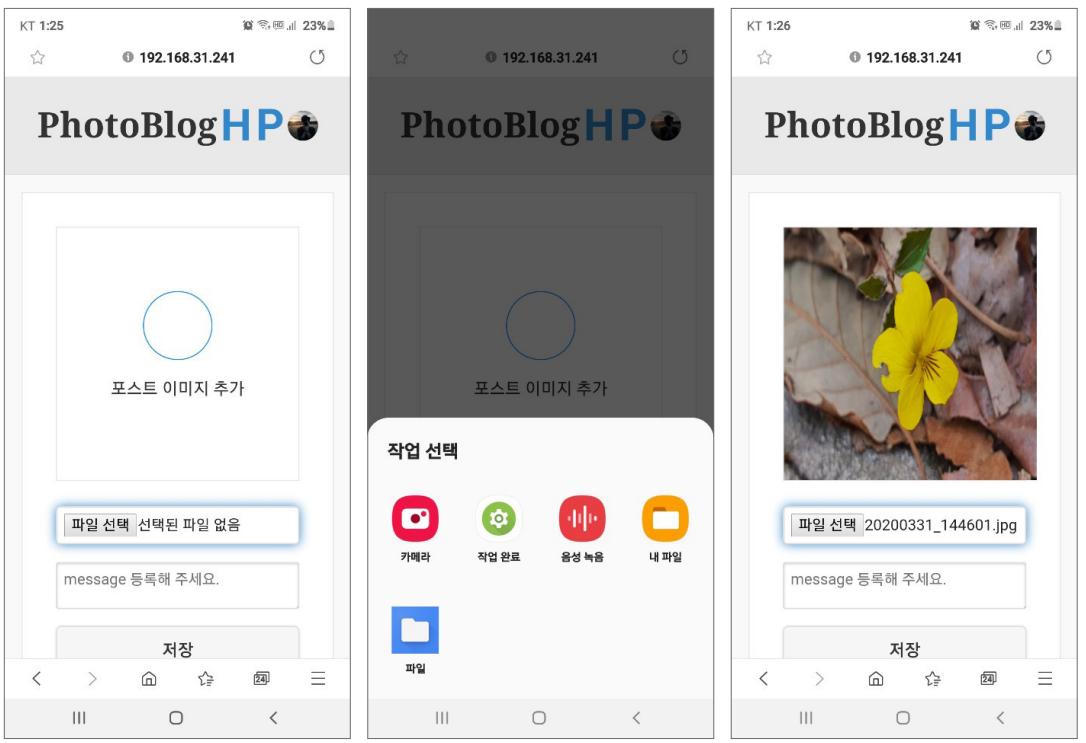
72 ~ 91 : 로그인 사용자를 제외한 전체 회원 중에 5명을 랜덤하게 선택하여 프로필 포토와 이름을 출력합니다. 프로필 포토를 클릭하면 클릭한 사람의 포토블로그 페이지(guest.jsp)로 이동합니다.

92 ~ 149 : 로그인 사용자의 포토블로그를 보여주는 영역입니다. 프로필 포토 및 id 그리고 포토포스트 및 메시지를 보여주고 만약 댓글이 있으면 댓글 리스트를 보여줍니다. (118~119) 하트 모양의 아이콘을 클릭하면 포토포스트의 '좋아요' 카운트가 증가합니다.

151 ~ 159 : frm과 frm1의 폼은 자바스크립의 함수에서 사용을 하기 위해 선언된 폼이고 화면에는 보이지 않는 hidden 타입으로 전부 선언을 하였습니다.

**05** 포토포스트 저장을 위해 입력폼 페이지를 다음과 같이 작성하고 저장합니다.

- Preview – post.jsp 페이지




▲ [그림 etc03-7] post.jsp 포토 입력폼 및 선택된 사진 미리보기 실행화면

source/etc03/post.jsp

```
01 : <%@page contentType="text/html; charset=EUC-KR"%>
02 : <%@page import="etc03.*"%>
03 : <jsp:useBean id="pmgr" class="etc03.PBlogMgr" />
04 : <%
05 :   String id = (String) session.getAttribute("idKey");
06 :   if (id == null)
07 :     response.sendRedirect("login.jsp");
08 :   PMemberBean mbean = pmgr.getPMember(id);
09 : %>
10 : <!DOCTYPE html>
11 : <html>
12 : <head>
13 : <meta charset="EUC-KR" />
14 : <meta name="viewport" content="width=device-width, initial-scale=1" />
15 : <title>PhotoBlog</title>
16 : <%@include file="js_css.html"%>
17 : <script type="text/javascript">
18 :   function send() {
19 :     document.frm.submit();
20 :   }
21 :   function goURL(url) {
22 :     document.frm1.action = url;
23 :     document.frm1.submit();
24 :   }
25 : </script>
26 : </head>
27 : <body>
28 : <div data-role="page" align="center">
29 :   <div data-role="header">
30 :     <table>
31 :       <tr>
32 :         <td align="left" width="200">
33 :        <h1 style="font-family: fantasy;" align="left">PhotoBlog</h1>
34 :         </td>
35 :         <td>
36 :        <a style="font-size:40px;" href="javascript:goURL('home.jsp','')">H</a>
37 :        <a style="font-size:40px;" href="javascript:goURL('post.jsp','')">P</a>
38 :         </td>
39 :         <td>
40 :        <div class="box" style="background: #BDBDBD;">
41 :            <img class="profile" src="photo/<%=mbean.getProfile()%>" width="30"
42 :           height="30" onclick="location='logout.jsp'">
43 :        </div>
44 :         </td>
```

[H]를 클릭하면 home.jsp로 이동합니다.

프로필 포토를 클릭하면 로그아웃 페이지로 이동합니다.

```
45 :       </tr>
46 :     </table>
47 :   </div>
48 :   <div data-role="content">
49 :     <form method="post" name="frm" action="pBlogPost"
50 :       enctype="multipart/form-data" class="post_form">
51 :     <div class="preview">
52 :       <div class="upload">
53 :         <div class="post_btn">
54 :        <div class="plus_icon"></div>
55 :          <p>포스트 이미지 추가</p>
56 :          <canvas id="imageCanvas"></canvas>
57 :       </div>
58 :         </div>
59 :       </div>
60 :       <p>
61 :         <input type="file" name="photo" id="id_photo" required="required">
62 :       </p>
63 :       <p>
64 :        <textarea name="message" cols="50" rows="5"
65 :         placeholder="message 등록해 주세요."></textarea>
66 :       </p>
67 :         <input type="hidden" value="<%=id%>" name="id">
68 :         <input type="button" value="저장" onclick="send()">
69 :       </form>
70 :     </div>
71 :   <form method="post" name="frm1"></form>
72 :   <%@include file="footer.jsp"%>
73 :   </div>
74 :   <script>
75 :   var fileInput = document.querySelector("#id_photo"), button = document
76 :     .querySelector(".input-file-trigger"), the_return = document
77 :     .querySelector(".file-return");
78 :   fileInput.addEventListener('change', handleImage, false);
79 :   var canvas = document.getElementById('imageCanvas');
80 :   var ctx = canvas.getContext('2d');
81 :
82 :   function handleImage(e) {
83 :     var reader = new FileReader();
84 :     reader.onload = function(event) {
85 :       var img = new Image();
86 :       img.onload = function() {
87 :         canvas.width = 300;
88 :         canvas.height = 300;
89 :         ctx.drawImage(img, 0, 0, 300, 300);
```

파일 업로드 기능을 위해서는 반드시 method는 post 방식이고 enctype을 multipart/form-data로 선언해야 합니다.

선택한 포토가 웹서버(Tomcat)로 업로드 되기 전에 미리보기 기능을 위해 선언한 canvas 태그입니다.

포토가 저장하기 위해 타입은 file로 선언하고 반드시 요청되기 위한 옵션으로 required로 선언합니다.

textarea 태그는 포토포스트에 포토와 같이 간단한 메세지가 저장이 됩니다.

로그인 id가 포토포스트에 저장하기 위해 hidden으로 선언을 합니다.

```
90 :      };
91 :       img.src = event.target.result;
92 :     };
93 :      reader.readAsDataURL(e.target.files[0]);
94 :    }
95 :    </script>
96 : </body>
97 : </html>
```

05 ~ 07 : 세션에서 가져온 id가 null이면 로그인 페이지로 이동을 합니다.

29 ~ 47 : post.jsp의 header 부분을 표시하는 코드 영역입니다.

71 21라인에 선언한 자바스크립트 함수에서 호출되는 폼입니다. method가 post를 사용하기 위해 form을 선언하였습니다.

74 ~ 95 : 포토포스트에 저장될 포토를 저장하기 미리보기 기능을 jQuery를 이용한 함수를 선언한 코드 영역입니다. jQuery의 설명은 JSP 및 서블릿 영역이 아니므로 구체적인 설명은 생략하도록 하겠습니다.

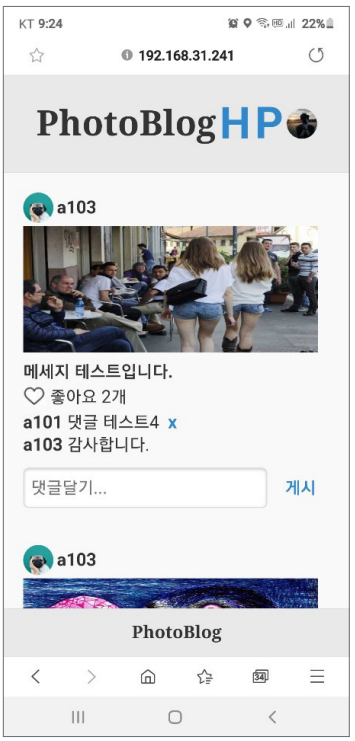
**06** 포토포스트 입력 처리 서블릿을 다음과 같이 작성하고 저장합니다.

source/etc03/PBlogPostServlet.java

```
01 : package etc03;
02 :
03 : import java.io.IOException;
04 : import javax.servlet.*;...   지면 관계상 다른 클래스의 import는 생략합니다.
09 :
10 : @WebServlet("/etc03/pBlogPost")   서블릿 맵핑 이름을 선언합니다.
11 : public class PBlogPostServlet extends HttpServlet {   서블릿 상속 클래스
12 :                    post.jsp에서 form의 method가 post이기 때문에 서블릿의 doPost를 호출합니다.
13 :    protected void doPost(HttpServletRequest request, HttpServletResponse response)
14 :             throws ServletException, IOException {
15 :        request.setCharacterEncoding("EUC-KR");   post.jsp에서 입력받은 값에 대한 인코딩을 EUC-KR로 선언합니다.
16 :        PBlogMgr pMgr = new PBlogMgr();
17 :        pMgr.insertPBlog(request);   웹서버(Tomcat)에 포토 저장도 하고 테이블에 저장도 하는 메소드입니다.
18 :        response.sendRedirect("home.jsp");
19 :    }
20 : }                    포토포스트 저장 후에 home.jsp로 페이지로 이동합니다.
```

**07** 다른 회원 포토블로그 방문 기능의 페이지를 다음과 같이 작성하고 저장합니다.

- Preview – guest.jsp 페이지




▲ [그림 etc03-8] guest.jsp 실행화면

source/etc03/guest.jsp

```
01 : <%@page contentType="text/html; charset=EUC-KR"%>
02 : <%@page import="java.util.Vector, etc03.*"%>
03 : <jsp:useBean id="pmgr" class="etc03.PBlogMgr" />
04 : <jsp:useBean id="rmgr" class="etc03.PReplyMgr" />
05 : <%
06 :   String id = (String) session.getAttribute("idKey");
07 :   String gid = request.getParameter("gid");
08 :   if (id == null){
09 :     response.sendRedirect("login.jsp");
10 :     return;
11 :   }
12 :   if (gid == null){
13 :     response.sendRedirect("home.jsp");
14 :     return;
15 :   }
16 :   PMemberBean mbean = pmgr.getPMember(id);
17 :   Vector<PBlogBean> pvlist = pmgr.listPBlog(gid);
18 : %>
19 : <!DOCTYPE html>
```

home.jsp에서 선택한 게스트(guest) id를 gid 변수로 받습니다.

로그인 사용자의 정보를 세션에 가져온 id의 매개변수로 리턴 받습니다.

게스트의 포토포스트 리스트를 Vector로 리턴 받습니다.

```
<html>
<head>
<meta charset="EUC-KR" />
<meta name="viewport" content="width=device-width, initial-scale=1" />
<title>PhotoBlog</title>
<%@include file="js_css.html"%>
<script type="text/javascript">
   function del(num) {
     document.frm.action = "pBlogDelete";
     document.frm.num.value=num;
     document.frm.submit();
   }
   function rDel(rnum) {
     document.frm.action = "pReplyDelete";
     document.frm.rnum.value=rnum;
     document.frm.gid.value="<%=gid%>";
     document.frm.submit();
   }
   function heart(num) {
     document.frm.action = "pBlogUpHCnt";
     document.frm.num.value=num;
     document.frm.gid.value="<%=gid%>";
     document.frm.submit();
   }
   function cmtPost(num) {
     document.frm.action = "pReplyPost";
     cmt = document.getElementById("comment"+num);
     document.frm.comment.value=cmt.value;
     document.frm.num.value=num;
     document.frm.id.value="<%=id%>";
     document.frm.gid.value="<%=gid%>";
     document.frm.submit();
   }
   function goURL(url) {
     document.frm1.action = url;
     document.frm1.submit();
   }
</script>
</head>
<body>
   <div data-role="page" align="center">
     <div data-role="header">
       <table>
         <tr>
        <td align="left" width="200">
          <h1 style="font-family: fantasy;" align="left">PhotoBlog</h1>
```

```
66 :         </td>
67 :         <td>
68 :           <a style="font-size:40px;" href="javascript:goURL('home.jsp')">H</a>
69 :           <a style="font-size:40px;" href="javascript:goURL('post.jsp')">P</a>
70 :         </td>
71 :         <td>
72 :           <div class="box" style="background: #BDBDBD;">
73 :          <img class="profile" src="photo/<%=mbean.getProfile()%>" width="30"
74 :           height="30" onclick="location='logout.jsp'">
75 :           </div>
76 :            </td>
77 :         </tr>
78 :       </table>
79 :     </div>
80 :     <div data-role="content">
81 :       <table>
82 :       <%
83 :        for (int i = 0; i < pvlist.size(); i++) {
84 :           PBlogBean pbean = pvlist.get(i);
85 :           PMemberBean tmbean = pmgr.getPMember(pbean.getId());
86 :       %>
87 :         <tr>
88 :        <td width="30">
89 :           <div class="box" style="background: #BDBDBD;">
90 :             <img class="profile" src="photo/<%=tmbean.getProfile()%>" width="30" height="30">
91 :           </div>
92 :        </td>
93 :        <td  width="250"><b><%=tmbean.getId()%></b></td>
94 :        <td> </td>
95 :         </tr>
96 :         <tr>
97 :        <td colspan="3">
98 :        <img src="photo/<%=pbean.getPhoto()%>" width="350" height="150">
99 :        </td>
100 :        </tr>
101 :        <tr>
102 :       <td colspan="3"><b><%=pbean.getMessage()%></b></td>
103 :        </tr>
104 :        <tr>
105 :       <td colspan="2" width="250">
106 :         <a href="javascript:heart('<%=pbean.getNum()%>')">
107 :         <img src="img/heart.jpg" align="top"></a> 좋아요 <%=pbean.getHcnt()%>개
108 :       </td>
109 :       <td> </td>
110 :        </tr>
111 :        <tr>
```

```
112 :       <td colspan="3" width="200">
113 :       <%
114 :          Vector<PReplyBean> rvlist = rmgr.listPReply(pbean.getNum());
115 :          for (int j = 0; j < rvlist.size(); j++) {
116 :            PReplyBean rbean = rvlist.get(j);
117 :       %>
118 :           <b><%=rbean.getId()%></b> <%=rbean.getComment()%> 
119 :           <%if (id.equals(rbean.getId())) {%>
120 :             <a href="javascript:rDel('<%=rbean.getRnum()%>')">x</a>
121 :           <%}%><br>
122 :       <%}%>
123 :       </td>
124 :        </tr>
125 :        <tr>
126 :       <td colspan="2">
127 :         <input id="comment<%=pbean.getNum()%>"
128 :          placeholder="댓글달기..." size="50">
129 :       </td>
130 :          <td align="center">
131 :         <a href="javascript:cmtPost('<%=pbean.getNum()%>')">게시</a>
132 :       </td>
133 :        </tr>
134 :        <tr>
135 :          <td colspan="3"><br></td>
136 :        </tr>
137 :        <%}%>
138 :      </table>
139 :    </div>
140 :    <form method="post" name="frm">
141 :      <input type="hidden" name="num">
142 :      <input type="hidden" name="comment">
143 :      <input type="hidden" name="rnum">
144 :      <input type="hidden" name="id">
145 :      <input type="hidden" name="gid">
146 :    </form>
147 :    <form method="post" name="frm1"></form>
148 :      <%@include file="footer.jsp"%>
149 :  </div>
150 : </body>
151 : </html>
```

06 ~ 15 : 세션에서 가져온 id 값이 null이면 login.jsp로 이동을 하고 request에서 가져온 gid 값이 null이면 home.jsp로 이동을 합니다. 기본적으로 web은 url이 오픈이 되어 있는 프로그램이기 때문에 정상적인 방법으로 링크를 타고 들어오면 이런 일이 없지만 정상적인 링크가 아닌 직접적인 url로 입력을 방식을 사용 할 경우에는 id와 gid가 null 값으로 리턴 받을 수 있는 상황을 고려한 코드입니다.

19 ~ 151 : home.jsp에서 랜덤 사용자 5명의 리스트 영역을 제외하고는 거의 똑같은 코드이므로 이후 코드 설명은 생략하고 home.jsp를 참고하시길 바랍니다.

**08** 포토포스트 삭제 서블릿을 다음과 같이 작성하고 저장합니다.

source/etc03/PBlogDeleteServlet.java

```
01 : package etc03;
02 :
03 : import java.io.IOException;
04 : import javax.servlet.*;...
09 :
10 : @WebServlet("/etc03/pBlogDelete")
11 : public class PBlogDeleteServlet extends HttpServlet {
12 :
13 :    protected void doPost(HttpServletRequest request, HttpServletResponse response)
14 :             throws ServletException, IOException {
15 :       request.setCharacterEncoding("EUC-KR");
16 :       PBlogMgr pMgr = new PBlogMgr();
17 :       int num = Integer.parseInt(request.getParameter("num"));
18 :       pMgr.deletePBlog(num);
19 :       response.sendRedirect("home.jsp");
20     }
21: }
```

지면 관계상 다른 클래스의 import는 생략합니다.

서블릿 맷핑 이름을 선언합니다.

home.jsp에서 form의 method가 post이기 때문에 서블릿에서 doPost 메소드를 선언합니다.

home.jsp에서 삭제할 포토포스트의 num 값을 받습니다.

요청받은 num 값으로 포토포스트를 삭제합니다.

토포스트 삭제 후에 home.jsp로 페이지로 이동합니다.

**09** JSP 페이지 및 서블릿에서 사용된 포토블로그와 관련된 SQL 기능을 다음과 같이 작성하고 저장합니다.

source/etc03/PBlogMgr.java

```
01 : package etc03;
02 :
03 : import java.sql.*;...
13 :
14 : public class PBlogMgr {
15 :
16 :    private DBConnectionMgr pool;
17 :    private static final String  SAVEFOLDER = "C:/Jsp/myapp/WebContent/etc03/photo/";
18 :    private static final String ENCTYPE = "EUC-KR";
19 :    private static int MAXSIZE = 5*1024*1024;
20 :
21 :    public PBlogMgr() {
22 :      try {
23 :        pool = DBConnectionMgr.getInstance();
24 :      } catch (Exception e) {
25 :         e.printStackTrace();
26 :      }
27 :    }
28 :
29 :    //PMember Login
30 :    public boolean loginPMember(String id, String pwd) {
```

지면 관계상 다른 클래스의 import는 생략합니다.

파일 업로드 기능에 파일(포토)이 저장되는 폴더 위치입니다.

요청되는 파일이 한글 인식에 필요한 인코딩으로 세팅입니다.

업로드 파일의 크기를 5MB로 제한합니다.

회원 로그인 메소드입니다.

```
31 :     Connection con = null;
32 :     PreparedStatement pstmt = null;
33 :     ResultSet rs = null;
34 :     String sql = null;
35 :     boolean flag = false;
36 :     try {
37 :       con = pool.getConnection();
38 :       sql = "select id from tblPMember where id=? and pwd=?";
39 :       pstmt = con.prepareStatement(sql);
40 :       pstmt.setString(1, id);
41 :       pstmt.setString(2, pwd);
42 :       rs = pstmt.executeQuery();
43 :       flag = rs.next();
44 :     } catch (Exception e) {
45 :        e.printStackTrace();
46 :     } finally {
47 :       pool.freeConnection(con, pstmt, rs);
48 :     }
49 :    return flag;
50 :   }
51 :
52 :   //PMember Get
53 :   public PMemberBean getPMember(String id) {
54 :     Connection con = null;
55 :     PreparedStatement pstmt = null;
56 :     ResultSet rs = null;
57 :     String sql = null;
58 :     PMemberBean bean = new PMemberBean();
59 :     try {
60 :       con = pool.getConnection();
61 :       sql = "select name, profile from tblPMember where id=?";
62 :       pstmt = con.prepareStatement(sql);
63 :       pstmt.setString(1, id);
64 :       rs = pstmt.executeQuery();
65 :       if (rs.next()) {
66 :         bean.setId(id);
67 :         bean.setName(rs.getString(1));
68 :         bean.setProfile(rs.getString(2));
69 :       }
70 :     } catch (Exception e) {
71 :        e.printStackTrace();
72 :      } finally {
73 :       pool.freeConnection(con, pstmt, rs);
74 :     }
75 :     return bean;
```

id와 pwd의 조건에 맞는 id를 검색하는 SQL문 입니다.

조건에 맞는 id와 pwd이면 true 값이고 일치하지 않으면 false 값을 리턴 받습니다.

회원 정보를 가져오는 메소드입니다.

매개변수로 받은 id의 조건에 맞는 이름과 프로필 포토명을 가져오는 SQL문 입니다.

```
76 :   }
77 :
78 :   //Random PMember List
79 :   public Vector<PMemberBean> listPMember(String id) {
80 :     Connection con = null;
81 :     PreparedStatement pstmt = null;
82 :     ResultSet rs = null;
83 :     String sql = null;
84 :     Vector<PMemberBean> vlist = new Vector<PMemberBean>();
85 :     try {
86 :       con = pool.getConnection();
87 :       sql = "select id, name, profile from tblPMember where id !=? order by rand() limit 5";
88 :       pstmt = con.prepareStatement(sql);
89 :       pstmt.setString(1, id);
90 :       rs = pstmt.executeQuery();
91 :       while (rs.next()) {
92 :         PMemberBean bean = new PMemberBean();
93 :         bean.setId(rs.getString(1));
94 :         bean.setName(rs.getString(2));
95 :         bean.setProfile(rs.getString(3));
96 :         vlist.addElement(bean);
97 :       }
98 :     } catch (Exception e) {
99 :        e.printStackTrace();
100 :    } finally {
101 :       pool.freeConnection(con, pstmt, rs);
102 :    }
103 :    return vlist;
104 :  }
105 :
106 :  //PBlog Insert
107 :  public void insertPBlog(HttpServletRequest req) {
108 :    Connection con = null;
109 :    PreparedStatement pstmt = null;
110 :    MultipartRequest multi = null;
111 :    String sql = null;
112 :    try {
113 :      con = pool.getConnection();
114 :      multi = new MultipartRequest(req, SAVEFOLDER,MAXSIZE, ENCTYPE,
115 :            new DefaultFileRenamePolicy());
116 :      String photo = null;
117 :      if (multi.getFilesystemName("photo") != null) {
118 :        photo = multi.getFilesystemName("photo");
119 :      }
120 :      sql = "insert tblPBlog(message,id,pdate,photo)values(?,?,now(),?)";
121 :      pstmt = con.prepareStatement(sql);
```

본인을 제외한 5명 회원 정보 리스트를 가져오는 메소드입니다.

tblPMember에서 매개변수로 받은 id를 제외하고 5개의 랜덤한 회원 정보를 검색하는 SQL문 입니다.

포토포스트를 저장하는 메소드입니다.

파일 업로드 기능을 위해 MultipartRequest 객체를 생성합니다. 5번째 매개변수는 중복된 파일명이 업로드 될 경우에 중복을 피하는 기능을 위한 객체입니다.

post.jsp에서 포토 업로드 위해 file type의 input 태그의 name 값이 photo입니다.

post.jsp에서 입력받은 값들을 저장하는 SQL문 입니다.

```
122 :       pstmt.setString(1, multi.getParameter("message"));
123 :       pstmt.setString(2, multi.getParameter("id"));
124 :       pstmt.setString(3, photo);
125 :       pstmt.executeUpdate();
126 :     } catch (Exception e) {
127 :       e.printStackTrace();
128 :      } finally {
129 :       pool.freeConnection(con, pstmt);
130 :     }
131 :   }
132 :
133 :   //PBlog Delete
134 :   public void deletePBlog(int num) {
135 :     Connection con = null;
136 :     PreparedStatement pstmt = null;
137 :     String sql = null;
138 :     try {
139 :       String photo = getPhoto(num);
140 :       if (photo!=null) {
141 :         File file = new File(SAVEFOLDER + "/" + photo);
142 :        if (file.exists())
143 :          UtilMgr.delete(SAVEFOLDER + "/" + photo);
144 :        }
145 :       con =pool.getConnection();
146 :       sql = "delete from tblPBlog where num=?";
147 :       pstmt = con.prepareStatement(sql);
148 :       pstmt.setInt(1, num);
149 :       if(pstmt.executeUpdate()>0) {
150 :        PReplyMgr pMgr = new PReplyMgr();
151 :        pMgr.deleteAllPReply(num);
152 :        }
153 :      } catch (Exception e) {
154 :        e.printStackTrace();
155 :      } finally {
156 :        pool.freeConnection(con, pstmt);
157 :     }
158 :   }
159 :
160 :   //PBlog Get Photo
161 :   public String getPhoto(int num) {
162 :     Connection con = null;
163 :     PreparedStatement pstmt = null;
164 :     ResultSet rs = null;
165 :     String sql = null;
166 :     String photo = null;
```

포토포스트를 삭제하는 메소드입니다.

포토포스트를 삭제할 때 업로드된 파일(포토)을 삭제 합니다.

조건에 맞는 포토포스트를 삭제하는 SQL문 입니다.

삭제한 포토포스트와 관련된 댓글도 모두 삭제하는 메소드의 호출입니다.

포토포스트의 포토 파일명을 가져오는 메소드입니다.

```
167 :    try {
168 :      con = pool.getConnection();
169 :      sql = "select photo from tblPBlog where num=?";
170 :      pstmt = con.prepareStatement(sql);
171 :      pstmt.setInt(1, num);
172 :      rs = pstmt.executeQuery();
173 :      if (rs.next()) {
174 :       photo = rs.getString(1);
175 :       }
176 :    } catch (Exception e) {
177 :      e.printStackTrace();
178 :    } finally {
179 :      pool.freeConnection(con, pstmt, rs);
180 :    }
181 :    return photo;
182 :  }
183 :
184 :  //PBlog List
185 :  public Vector<PBlogBean> listPBlog(String id) {
186 :    Connection con = null;
187 :    PreparedStatement pstmt = null;
188 :    ResultSet rs = null;
189 :    String sql = null;
190 :    Vector<PBlogBean> vlist = new Vector<PBlogBean>();
191 :    try {
192 :      con = pool.getConnection();
193 :      sql = "select * from tblPBlog where id=? order by num desc";
194 :      pstmt = con.prepareStatement(sql);
195 :      pstmt.setString(1, id);
196 :      rs = pstmt.executeQuery();
197 :      while (rs.next()) {
198 :        PBlogBean bean = new PBlogBean();
199 :        bean.setNum(rs.getInt(1));
200 :        bean.setMessage(rs.getString(2));
201 :        bean.setId(rs.getString(3));
202 :        bean.setPdate(rs.getString(4));
203 :        bean.setPhoto(rs.getString(5));
204 :        bean.setHcnt(rs.getInt(6));
205 :        vlist.addElement(bean);
206 :      }
207 :    } catch (Exception e) {
208 :       e.printStackTrace();
209 :    } finally {
210 :        pool.freeConnection(con, pstmt, rs);
211 :     }
212 :    return vlist;
```

조건에 맞는 포토 파일명을 검색하는 SQL문 입니다.

포토포스트 리스트를 가져오는 메소드입니다.

조건에 맞는 포토포스트 리스트를 검색하는 SQL문 입니다.

```
213 :   }
214 :
215 :  //HCnt Up
216 :  public void upHCnt(int num) {
217 :    Connection con = null;
218 :    PreparedStatement pstmt = null;
219 :    String sql = null;
220 :    try {
221 :      con = pool.getConnection();
222 :      sql = "update tblPBlog set hCnt=hCnt+1 where num=?";
223 :      pstmt = con.prepareStatement(sql);
224 :      pstmt.setInt(1, num);
225 :      pstmt.executeUpdate();
226 :    } catch (Exception e) {
227 :      e.printStackTrace();
228 :    } finally {
229 :      pool.freeConnection(con, pstmt);
230 :    }
231 :   }
232 : }
```

'좋아요'의 카운트가 증가하는 메소드입니다.

조건에 맞는 포토포스트 '좋아요'(hCnt)의 카운트가 1씩 증가하는 SQL문 입니다.

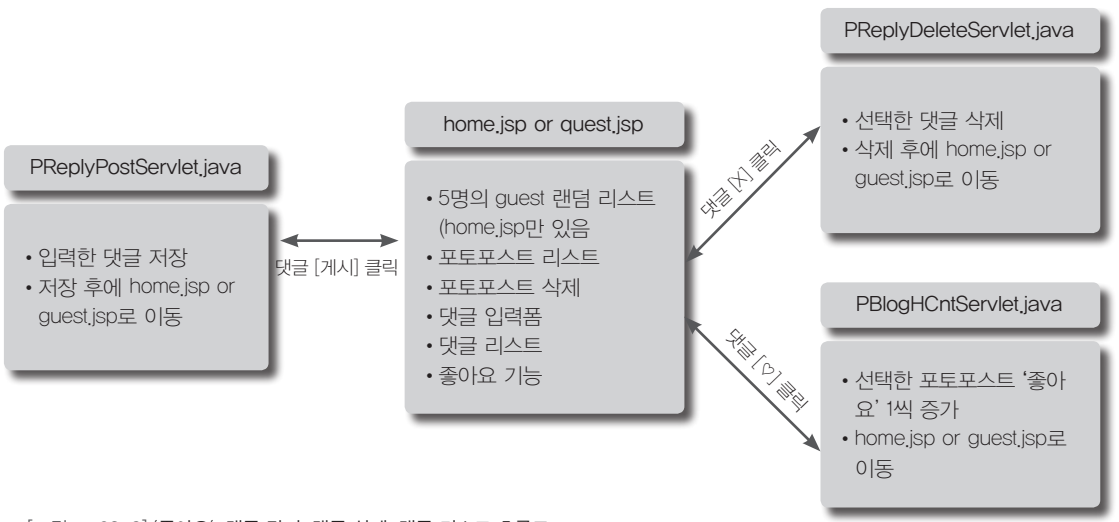
**30 ~ 50** : loginPMember 메소드 (PBlogLoginServlet.java의 20라인에서 사용)
PMember Login : 회원 로그인 메소드입니다.
**53 ~ 76** : getPMember 메소드 (home.jsp의 9, 96라인 및 guest.jsp의 16,85라인에서 사용)
PMember Get : 회원 정보를 가져오는 메소드입니다.
**79 ~ 104** : listPBlog 메소드(PBlogLoginServlet.java의 home.jsp의 11라인에서 사용)
Random PMember List : 본인을 제외한 5명 회원 정보 리스트를 가져오는 메소드입니다.
**107 ~ 131** : insertPBlog 메소드 (PBlogPostServlet의 17라인에서 사용)
PBlog Insert : 포토포스트를 저장하는 메소드입니다.
**133 ~ 158** : deletePBlog 메소드 (PBlogDeleteServlet의 18라인에서 사용)
PBlog Delete : 포토포스트를 삭제하는 메소드입니다.
**160 ~ 182** : getPhoto 메소드 (PBlogMgr.java의 139라인에서 사용)
PBlog Get Photo : 포토포스트의 포토 파일명을 가져오는 메소드입니다.
**185 ~ 213** : listPBlog 메소드 (home.jsp의 11라인 및 guest.jsp의 17라인에서 사용)
PBlog List : 포토포스트의 리스트를 가져오는 메소드입니다.
**216 ~ 232** : upHCnt 메소드 (PBlogHCntServlet.java의 18라인에서 사용)
HCnt Up : '좋아요'의 카운트가 증가하는 메소드입니다.

## 02-3 포토포스트 좋아요, 댓글 달기, 댓글 삭제, 댓글 리스트

두 번째 단계는 포토포스트의 '좋아요', 댓글 달기, 댓글 리스트, 댓글 삭제 기능입니다.

포토포스트의 '좋아요'는 home.jsp or guest.jsp 페이지에서 각각의 포토포스트에 있는 [♡]를 클릭하면 PBlogHCntServlet 서블릿에서 데이터베이스와 연동하여 카운트가 올라갑니다. 포토블로그에는 댓글을 달 수 있는 기능이 있습니다. 댓글을 달고자 하는 포토포스트에 댓글을 입력하고 [게시]을 클릭하면 PReplyPostServlet 서블릿에서 데이터베이스에 저장 처리합니다. 입력한 댓글은 home.jsp or guest.jsp에서 댓글 리스트를 보여주고 마지막으로 본인이 입력한 댓글은 댓글 끝에 [X]를 클릭하면 PReplyDeleteServlet 서블릿에서 데이터베이스와 연동하여 삭제 처리합니다.

지금까지 두 번째 단계에 대한 흐름의 설명입니다. 첫 번째 단계를 잘 이해하셨다면 두 번째 단계도 쉽게 구현할 수 있을 것입니다.




▲ [그림 etc03-9] '좋아요', 댓글 달기, 댓글 삭제, 댓글 리스트 흐름도

### 포토포스트에 사용되는 '좋아요', 댓글 달기, 댓글 삭제, 댓글 리스트 부분

**01** 포토포스트에서 '좋아요' 기능의 서블릿을 다음과 같이 작성하고 저장합니다.

source/etc03/PBlogHCntServlet.java

```
01 : package etc03;
02 :
03 : import java.io.IOException;
04 : import javax.servlet.*;...  지면 관계상 다른 클래스의 import는 생략합니다.
09 :
10 : @WebServlet("/etc03/pBlogUpHCnt")  웹브라우저에서 사용되는 url값으로 맵핑 합니다.
11 : public class PBlogHCntServlet extends HttpServlet {
```

```
12 :
13 :   protected void doPost(HttpServletRequest request, HttpServletResponse response)
14 :             throws ServletException, IOException {
15 :       request.setCharacterEncoding("EUC-KR");
16 :       PBlogMgr pMgr = new PBlogMgr();
17 :       int num = Integer.parseInt(request.getParameter("num"));
18 :       pMgr.upHCnt(num);
19 :       String gid = request.getParameter("gid");
20 :       if(gid==null)
21 :         response.sendRedirect("home.jsp");
22 :       else
23 :         response.sendRedirect("guest.jsp?gid="+gid);
24 :   }
25: }
```

home.jsp or guest.jsp에서 좋아요를 하고 싶은 포토포스트의 num 값을 받습니다.

매개변수 num으로 ' 좋아요 ' 카운트를 증가하는 메소드를 호출합니다,

gid의 요청의 유무로 home.jsp or guest.jsp로 이동할지 판단합니다.

gid 요청이 없다면 home.jsp로 페이지로 이동합니다.

gid 요청이 있다면 guest.jsp로 페이지로 이동합니다.

**02** 포토포스트에서 '댓글 달기' 기능의 서블릿을 다음과 같이 작성하고 저장합니다.

source/etc03/PBlogHCntServlet.java

```
01 : package etc03;
02 :
03 : import java.io.IOException;
04 : import javax.servlet.*;...
09 :
10 : @WebServlet("/etc03/pReplyPost")
11 : public class PReplyPostServlet extends HttpServlet {
12 :
13 :   protected void doPost(HttpServletRequest request, HttpServletResponse response)
14 :             throws ServletException, IOException {
15 :       request.setCharacterEncoding("EUC-KR");
16 :       PReplyMgr rMgr = new PReplyMgr();
17 :       PReplyBean rbean = new PReplyBean();
18 :       rbean.setNum(Integer.parseInt(request.getParameter("num")));
19 :       rbean.setId(request.getParameter("id"));
20 :       rbean.setComment(request.getParameter("comment"));
21 :       rMgr.insertPReply(rbean);
22 :       String gid = request.getParameter("gid");
23 :       if(gid==null)
24 :         response.sendRedirect("home.jsp");
25 :       else
26 :         response.sendRedirect("guest.jsp?gid="+gid);
27 :   }
28: }
```

지면 관계상 다른 클래스의 import는 생략합니다.

웹브라우저에서 사용되는 url값으로 맵핑합니다.

댓글을 저장하는 메소드를 호출합니다.

gid의 요청의 유무로 home.jsp or guest.jsp로 이동할지 판단합니다. gid값이 null이면 home.jsp에서 호출을 하였고 gid값이 null이 아니면 guest.jsp에서 호출을 한 것입니다.

gid 요청이 없다면 home.jsp로 페이지로 이동합니다.

gid 요청이 있다면 guest.jsp로 페이지로 이동합니다. guest.jsp로 이동할 때 조건값으로 다시 gid 값을 넘깁니다.

**17 ~ 20** : home.jsp or guest.jsp에서 댓글 달기를 위해 입력받은 값들을 PReplyBean 객체를 생성하고 setter 메소드를 사용하여 빈즈 객체에 담습니다.

**03** 포토포스트에서 '댓글 삭제' 기능의 서블릿을 다음과 같이 작성하고 저장합니다.

source/etc03/PReplyDeleteServlet.java

```
01 : package etc03;
02 :
03 : import java.io.IOException;
04 : import javax.servlet.*;...
09 :
10 : @WebServlet("/etc03/pReplyDelete")
11 : public class PReplyDeleteServlet extends HttpServlet {
12 :
13 :   protected void doPost(HttpServletRequest request, HttpServletResponse response)
14 :             throws ServletException, IOException {
15 :       request.setCharacterEncoding("EUC-KR");
16 :       PReplyMgr rMgr = new PReplyMgr();
17 :       int rnum = Integer.parseInt(request.getParameter("rnum"));
18 :       rMgr.deletePReply(rnum);
19 :       String gid = request.getParameter("gid");
20 :       if(gid==null)
21 :         response.sendRedirect("home.jsp");
22 :       else
23 :         response.sendRedirect("guest.jsp?gid="+gid);
24 :   }
25: }
```

지면 관계상 다른 클래스의 import는 생략합니다.

웹브라우저에서 사용되는 url값으로 맵핑 합니다.

home.jsp or guest.jsp에서 삭제할 댓글의 rnum 값을 받습니다.

rnum 값을 매개변수로 댓글 삭제 메소드를 호출합니다.

gid의 요청의 유무로 home.jsp or guest.jsp로 이동할지 판단합니다. gid값이 null이면 home.jsp에서 호출을 하였고 gid 값이 null이 아니면 guest.jsp에서 호출을 한 것입니다.

home.jsp로 페이지로 이동합니다.

gid 요청이 있다면 guest.jsp로 페이지로 이동합니다. guest.jsp로 이동할 때 조건 값으로 다시 gid 값을 넘깁니다.

**04** JSP 페이지 및 서블릿에서 사용된 댓글과 관련된 SQL 기능을 다음과 같이 작성하고 저장합니다.

source/etc03/PReplyMgr.java

```
01 : package etc03;
02 :
03 : import java.sql.Connection;
04 : import java.sql.PreparedStatement;
05 : import java.sql.ResultSet;
06 : import java.util.Vector;
07 :
08 : public class PReplyMgr {
09 :
10 :   private DBConnectionMgr pool;
11 :
12 :   public PReplyMgr() {
13 :     try {
14 :       pool = DBConnectionMgr.getInstance();
15 :     } catch (Exception e) {
16 :       e.printStackTrace();
17 :     }
```

```
18 :   }
19 :
20 :   //PReply Insert
21 :   public void insertPReply(PReplyBean bean) {
22 :     Connection con = null;
23 :     PreparedStatement pstmt = null;
24 :     String sql = null;
25 :     try {
26 :       con = pool.getConnection();
27 :       sql = "insert tblPReply(num,id,rdate,comment)values(?,?,now(),?)";
28 :       pstmt = con.prepareStatement(sql);
29 :       pstmt.setInt(1, bean.getNum());
30 :       pstmt.setString(2, bean.getId());
31 :       pstmt.setString(3, bean.getComment());
32 :       pstmt.executeUpdate();
33 :     } catch (Exception e) {
34 :       e.printStackTrace();
35 :     } finally {
36 :       pool.freeConnection(con, pstmt);
37 :     }
38 :   }
39 :
40 :   //PReply Delete
41 :   public void deletePReply(int rnum) {
42 :     Connection con = null;
43 :     PreparedStatement pstmt = null;
44 :     String sql = null;
45 :     try {
46 :       con = pool.getConnection();
47 :       sql = "delete from tblPReply where rnum=?";
48 :       pstmt = con.prepareStatement(sql);
49 :       pstmt.setInt(1, rnum);
50 :       pstmt.executeUpdate();
51 :     } catch (Exception e) {
52 :       e.printStackTrace();
53 :     } finally {
54 :       pool.freeConnection(con, pstmt);
55 :     }
56 :   }
57 :
       //PReply All Delete
58 :   public void deleteAllPReply(int num) {
59 :     Connection con = null;
60 :     PreparedStatement pstmt = null;
61 :     String sql = null;
62 :     try {
```

댓글을 입력하는 메소드입니다.

home.jsp 및 guest.jsp에서 요청한 댓글을 저장하는 SQL문 입니다.

댓글을 삭제하는 메소드입니다.

home.jsp 및 guest.jsp에서 요청한 댓글을 삭제하는 SQL문 입니다.

포토포스트의 관련된 댓글을 모두 삭제하는 메소드

```
63 :       con = pool.getConnection();
64 :       sql = "delete from tblPReply where num=?";
65 :       pstmt = con.prepareStatement(sql);
66 :       pstmt.setInt(1, num);
67 :       pstmt.executeUpdate();
68 :     } catch (Exception e) {
69 :       e.printStackTrace();
70 :     } finally {
71 :       pool.freeConnection(con, pstmt);
72 :     }
73 :   }
74 :
75 :   //PReply List
76 :   public Vector<PReplyBean> listPReply(int num){
77 :     Connection con = null;
78 :     PreparedStatement pstmt = null;
79 :     ResultSet rs = null;
80 :     String sql = null;
81 :     Vector<PReplyBean> vlist = new Vector<PReplyBean>();
82 :     try {
83 :       con = pool.getConnection();
84 :       sql = "select * from tblPReply where num=? order by rnum desc";
85 :       pstmt = con.prepareStatement(sql);
86 :       pstmt.setInt(1, num);
87 :       rs = pstmt.executeQuery();
88 :       while (rs.next()) {
89 :         PReplyBean bean = new PReplyBean();
90 :         bean.setRnum(rs.getInt(1));
91 :         bean.setNum(rs.getInt(2));
92 :         bean.setId(rs.getString(3));
93 :         bean.setRdate(rs.getString(4));
94 :         bean.setComment(rs.getString(5));
95 :         vlist.addElement(bean);
96 :       }
97 :     } catch (Exception e) {
98 :        e.printStackTrace();
99 :     } finally {
100 :      pool.freeConnection(con, pstmt, rs);
101 :    }
102 :    return vlist;
103 :  }
104 : }
```

BlogMgr.java에서 포토포스트 삭제 시 관련된 모든 댓글 삭제하는 SQL문 입니다.

댓글 메소드입니다.

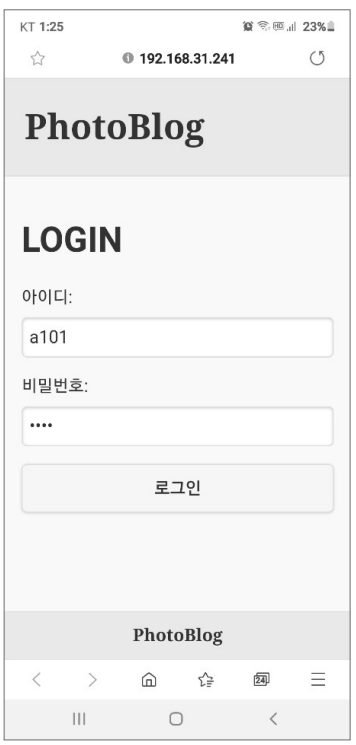
조건에 맞는 댓글 리스트를 가져오는 SQL문 입니다.

21 ~ 38 : insertPReply 메소드 (PReplyPostServlet.java의 21라인에서 사용)
PReply Insert : 댓글을 입력하는 메소드입니다.
41 ~ 56 : deletePReply메소드 (PReplyDeleteServlet.java의 18라인에서 사용)
PReply Delete : 댓글을 삭제하는 메소드입니다.
58 ~ 73 : deleteAllPReply 메소드 (PBlogMgr.java의 151라인에서 사용)
PReply All Delete : 포토포스트의 관련된 댓글을 모두 삭제하는 메소드입니다.
75 ~ 103 : listPReply 메소드 (home.jsp의 122라인 및 guest.jsp의 114라인에서 사용)
PReply List : 댓글 리스트입니다.

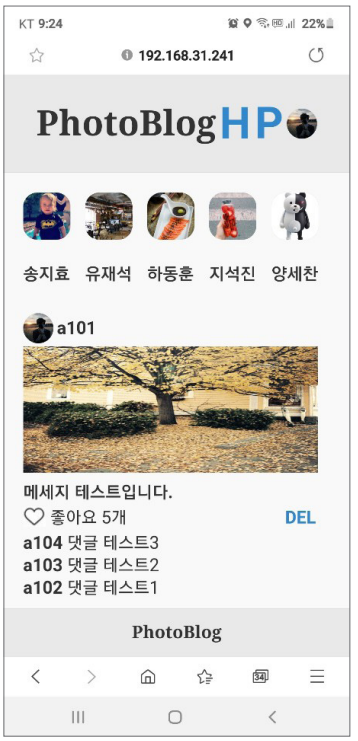
## 02-4 포토블로그 페이지 실행

앞에서 만든 포토블로그을 실행시켜 모든 기능을 화면으로 살펴보도록 하겠습니다.
회원 로그인, 포토포스트 올리기, 포토포스트 리스트, 포토포스트 삭제, '좋아요', 댓글 달기, 댓글 삭제, 댓글 리스트

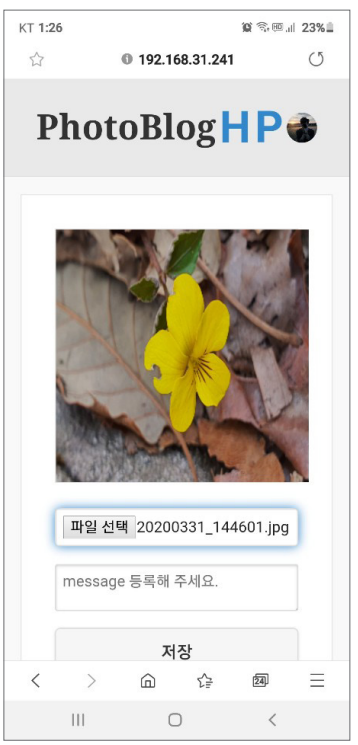
**01** 이클립스에서 login.jsp를 실행시켜 id와 pwd를 입력하고 [로그인] 버튼을 클릭합니다.



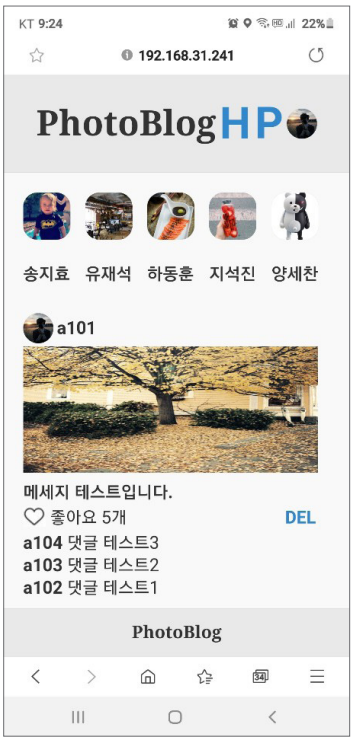
**02** 입력한 id의 포토포스트 리스트를 home.jsp에서 확인을 합니다. 그리고 사진 밑에 있는 [DEL]를 클릭하면 포토포스트가 삭제되고 [♡] 클릭하면 '좋아요'의 카운트가 올라갑니다.



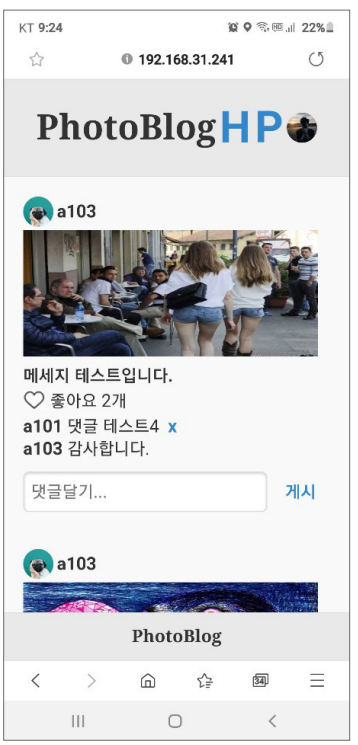

**03** home.jsp에서 상단 오른쪽 [P]를 클릭하면 post.jsp로 이동합니다. post.jsp에서 포토와 메시지를 선택하고 입력하여 [저장] 버튼을 클릭합니다.



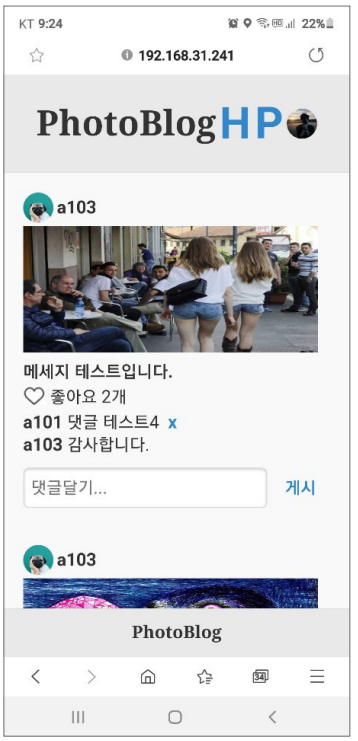
**04** post.jsp에서 포토와 메시지를 저장하고 다시 home.jsp에서 랜덤으로 가져온 다섯 명의 guest 포토블로그에 방문하기 위해 한 개를 선택하여 [프로필사진] 클릭합니다.




**05** guest.jsp는 선택한 게스트 포토블로그이고 여기에서 포토포스트에 댓글을 달고 [게시]를 클릭합니다. 자신이 입력한 댓글은 댓글 끝에 [X]를 클릭하면 댓글이 삭제됩니다.



**06** guest.jsp에서 상단 오른쪽 [H]를 클릭하면 자신의 포토블로그 화면으로 이동합니다.




**07** home.jsp 또는 guest.jsp에서 상단 오른쪽 [프로필사진]을 클릭하면 로그아웃을 하고 login.jsp 화면으로 이동합니다.


이상으로 포토블로그의 설명을 모두 마치도록 하겠습니다.