

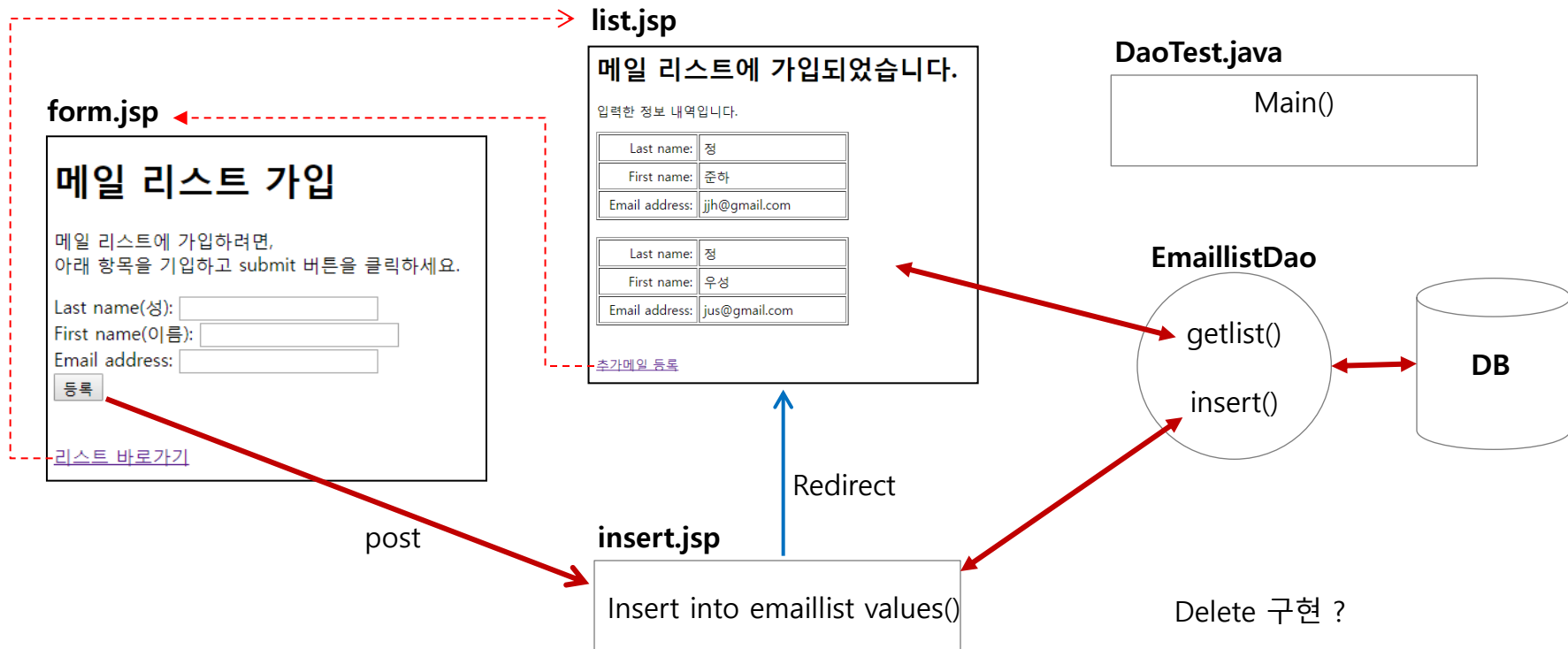
chapter04

emalilist 만들기 (Model 1)

1. emalilist 개요 및 분석
2. JSP 태그
3. request 객체 메소드
4. Get방식, Post방식
5. 웹페이지 강제 이동 처리, 경로 지정 방법
6. guestbook 만들기 : Model 1

01 maillist 개요 및 분석

■ 분석



■ 분석

form.jsp

메일 리스트 가입

메일 리스트에 가입하려면,
아래 항목을 기입하고 submit 버튼을 클릭하세요.

✓ Last name(성):

✓ First name(이름):

✓ Email address:

[리스트 바로가기](#)

list.jsp

메일 리스트에 가입되었습니다.

입력한 정보 내역입니다.

| | |
|----------------|---------------|
| Last name: | 정 |
| First name: | 준하 |
| Email address: | jjh@gmail.com |

| | | |
|----------------|---------------|---|
| Last name: | 정 | ✓ |
| First name: | 우성 | ✓ |
| Email address: | jus@gmail.com | ✓ |

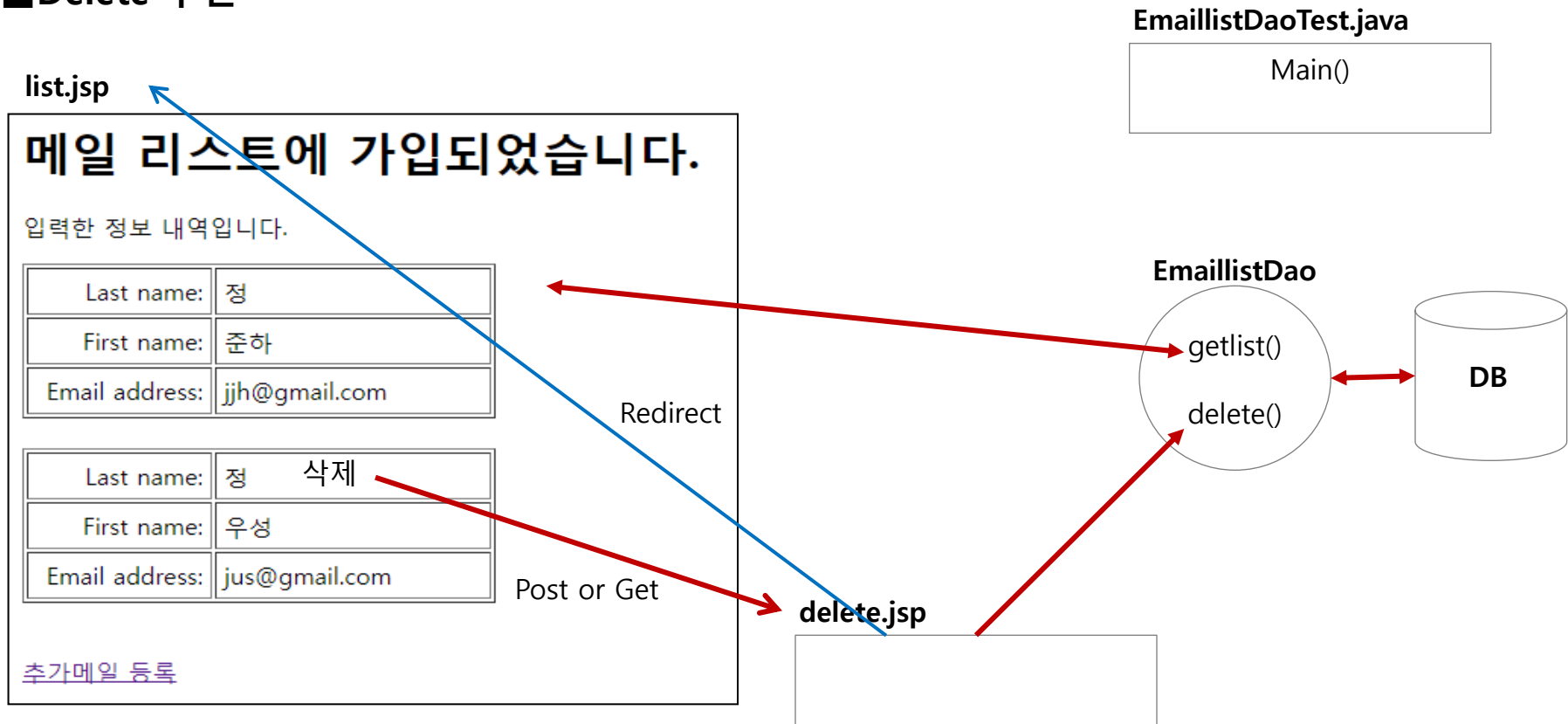
[추가메일 등록](#)

maillist

| | | | |
|------------|---------------|----------|-------|
| no | number | PK | 식별번호 |
| last_name | varchar2(10) | | 성 |
| first_name | varchar2(10) | | 이름 |
| email | varchar2(100) | NOT NULL | 이메일주소 |

```
create table maillist(
    no number(10),
    last_name varchar2(10),
    first_name varchar2(10),
    email varchar2(100) not null enable,
    primary key(no)
);
```

■Delete 구현



■ 기본태그

| Tag | Name | Purpose |
|-----------|-----------|-------------------------|
| <% %> | JSP 스크립틀릿 | Java 구문을 JSP 페이지에 삽입 |
| <%= %> | JSP 표현식 | Java 표현식을 문자열로 출력 |
| <%@ %> | JSP 지시자 | JSP 페이지 전체에 적용되는 조건을 설정 |
| <%-- --%> | JSP 주석 | JSP 페이지에 주석 구문을 삽입 |
| <%! %> | JSP 선언문 | 인스턴스 변수 및 메소드를 선언 |

■ Java 클래스를 import하는 JSP 코드

```
<%@ page import="java.util.List" %>
```

```
<%@ page import="com.javaex.emaillist.dao.EmaillistDao" %>
```

```
<%@ page import="com.javaex.emaillist.vo.EmaillistVo" %>
```

■ request 객체 메소드

| 메소드 | 설명 |
|---|---|
| getParameter (String param) | 주어진 이름의 파라미터가 갖는 값을 리턴한다. 지정된 파라미터가 없는 경우에는 null을 리턴한다. |
| getParameterValues (String param) | 주어진 이름의 파라미터가 갖는 모든 값을 String 타입의 배열로 리턴한다. 파라미터가 다중 선택이 가능한 리스트 또는 체크 박스 값이라면, 여러 개의 값이 하나의 이름으로 전달 될 수 있다. |
| getParameterNames() | request 객체에 포함되어 있는 모든 파라미터의 이름을 Enumeration(열거형) 객체로 리턴한다. request에 파라미터가 하나도 없는 경우에는 비어있는 Enumeration 객체를 리턴한다. |

03 request 객체 메소드

http://localhost:8088/helloJSP/requestTest.jsp?num=4321

jsp & Servlet

■ request.getParameter() 메소드 - test1.html -> test1.jsp

```
<form action="test1.jsp" method="post">
  <input type="text" name="num"/>
  <input type="submit"/>
</form>
```

```
<!-- num값으로 123을 넘겨받았을때 -->
<%=request.getParameter("num") %><br/> <!-- 123 -->
<%=request.getAttribute("num") %> <!-- null -->
<br/>
<% request.setAttribute("num", "4321"); %>
<%=request.getParameter("num") %><br/> <!-- 123 -->
<%=request.getAttribute("num") %> <!-- 4321 -->
```

03 request 객체 메소드

■ request.getParameterValues() 메소드 – test2.html -> test2.jsp

```
<form action="test2.jsp" method="post">
  <p>
    <select name="color">
      <option>red</option>
      <option>green</option>
      <option>blue</option>
      <option>white</option>
      <option>black</option>
    </select>
  </p>

  <p>
    <input type="checkbox" name="choice" value="A"> A 타입 <br/>
    <input type="checkbox" name="choice" value="B"> B 타입 <br/>
    <input type="checkbox" name="choice" value="C"> C 타입 <br/>
  </p>
  <input type="submit"/>
</form>
```


■ request.getParameterValues() 메소드 – test2.html -> test2.jsp

```
String colorParam = request.getParameter("color");
out.println("<br>colorParam: " + colorParam);

String[] colorParamVal = request.getParameterValues("color");
for(String c : colorParamVal){
    out.println("<br>colorParamVal: " + c);
}

String one = request.getParameterValues("choice")[0];
out.println("<br>choice[0]: " + one);

String[] choice = request.getParameterValues("choice");

for(String c : choice){
    out.println("<br>choice: " + c);
}
```

03 request 객체 메소드

■ request.getParameterNames() 메소드 – test3.html -> test3.jsp

```
<form action="test3.jsp" method="post">
    <input type="text" name="var1"/>
    <input type="text" name="var2"/>
    <input type="text" name="myVar"/>
    <input type="submit"/>
</form>
```

```
<%
Enumeration params = request.getParameterNames();
while(params.hasMoreElements()) {
    String name = (String) params.nextElement();
    out.print(name + " : " + request.getParameter(name) + "<br>");
}
%>
```

■Get 방식

- 가능한 한 빠르게 데이터를 전송하기 원하는 경우
- 4KB 보다 적은 데이터를 보내는 경우
- 파라미터가 URL에 표시되어도 상관없는 경우
- 사용자가 웹 페이지를 브라우저에 등록(bookmark)할 때 파라미터를 포함하는 경우

■Post 방식

- 4KB가 넘는 데이터를 전송하는 경우
- URL 뒤에 파라미터가 덧붙여지면 안 되는 경우

04 HTTP Get방식의 데이터 전송

■ 데이터를 URL에 포함하여 링크를 통해 값을 전달 하는 방식

- URL의 총 길이가 1024 byte로 제한 되기 때문에 전송할 수 있는 데이터에 한계가 있다
- 모든 데이터가 URL에 그대로 노출되기 때문에 보안이 필요한 경우 적합하지 않다
- 한글이나 공백은 URLEncoder 처리 해야만 전송할 수 있다
- 브라우저별로 길이 제한이 다를 수 있다 (<http://brainage.egloos.com/4323903>)

■ 데이터 전송 방법

- URL 뒤에 ? 를 명시하고 이름=값&이름=값 형태로 전송한다

```
1 <%
2     String name= "개발자";
3     int w = 72;
4     int h = 175;
5 %>
6
7 <a href="hello.jsp?name<%=URLEncoder.encode(name)%>&width=<%=w%>&height=<%=h%>">click</a>
```

■ 데이터 수신 방법

- 데이터가 전송된 페이지에서는 request 내장객체에 이전 페이지로부터 전송된 변수가 저장된다.
- URL에 명시한 변수이름을 `getParameter()`메서드에 전달하여 값을 리턴받는다.

```
1  <%  
2      // 모든 페이지에서 파라미터 수신 전 1회 수행해야 함.  
3      request.setCharacterEncoding("UTF-8");  
4  
5      // 전달한 값의 형식에 상관 없이 모두 String으로 리턴된다.  
6      String name = request.getParameter("name");  
7      String w = request.getParameter("width");  
8      String h = request.getParameter("height");  
9  %>
```

■ HTML의 <form>태그 내에 <input> 태그에 입력한 내용을 전송 하는 방식

- 전송할 수 있는 데이터에 한계가 없다
- 보통 20MB로 설정되어 있기때문에 GET방식에 비해서 상대적으로 한계가 없다
- 전송하는 내용이 URL에 노출되지 않기 때문에 GET방식에 비해서 상대적으로 보안에 유리하다. (절대적으로 안전하지는 않다!)
- 한글이나 공백에 대한 URLEncoding 처리가 필요 없다.

■ 데이터 전송 방법

- <form> 태그의 method 속성에 post라고 명시
- action속성에 데이터를 수신할 jsp페이지의 URL(경로)를 지정
- 이 <form> 내의 submit버튼이 눌러졌을때 모든 input 요소에 대한 입력값(value속성 값)이 전송된다

```
1 <form method="post" action="post.jsp">
2     <input type="text" name="hello" />
3     <input type="text" name="world" />
4     <button type="submit">click</button>
5 </form>
```

■ 데이터 수신 방법

- 데이터가 전송된 페이지에서는 get 방식과 동일하게 처리한다
- <input>태그에 명시한 name속성의 값을 request.getParameter()에 파라미터로 전달하여 입력값을 리턴받는다

```
1  <%  
2      // 모든 페이지에서 파라미터 수신 전 1회 수행해야 함.  
3      request.setCharacterEncoding("UTF-8");  
4  
5      // 전달한 값의 형식에 상관 없이 모두 String으로 리턴된다.  
6      String hello = request.getParameter("hello");  
7      String world = request.getParameter("world");  
8  %>
```

■ 예외 처리 필요

- 전송된 값이 null일 수 있다.
- null이 아니더라도 빈 문자열("")일 수 있다

```
1  <%  
2      String hello = request.getParameter("hello");  
3  
4      if(hello == null || hello.equals("")){  
5          hello = "안녕하세요";  
6      }  
7  %>
```


■ GET방식의 경우

- 이전 페이지에서 전달된 파라미터를 URL에 누적시켜 전달

```
1  <%  
2      String w = request.getParameter("w");  
3      String h = request.getParameter("h");  
4  %>  
5  <a href="hello.jsp?w=<%=w%>&h=<%=h%>&name=helloworld">click</a>
```

■ POST방식의 경우

- 이전 페이지에서 전달된 파라미터를 hidden field에 포함시켜 submit 처리

```
1  <%  
2      String w = request.getParameter("w");  
3      String h = request.getParameter("h");  
4  %>  
5  <form method="post" action="foo.jsp">  
6      <input type="hidden" name="w" value="<%=w%>" />  
7      <input type="hidden" name="h" value="<%=h%>" />  
8  </form>
```

■ Cookie

- 변수값을 사용자의 PC에 텍스트 형태로 저장
- 초단위의 유효시간과 유효 도메인을 설정해야함
- 서로 다른 도메인간에는 공유할 수 없지만 서브도메인간에는 공유 가능함
- 보안에 취약 하고 데이터 저장시 URLEncoder 처리, 데이터 읽어올 때 URLDecode 처리 필요
- 지정된 시간동안은 브라우저를 닫았다가 다시 열어도 삭제되지 않으며, 사이트 내의 모든 페이지에서 읽을 수 있는 전역 변수의 역할을 한다.

■ Cookie 저장하기

```
1 String input = URLEncoder.encode("저장할 값");
2 //쿠키 생성
3 Cookie info = new Cookie("mycookie", input);
4 // 쿠키의 유효시간 (초)
5 info.setMaxAge(60);
6 // 쿠키가 유효한 경로 설정
7 info.setPath("/");
8 // 쿠키가 유효한 도메인 설정
9 // --> 상용화시에는 사이트에 맞게 수정해야 함
10 info.setDomain("localhost");
11 // 쿠키저장하기
12 response.addCookie(info);
```

■ Cookie 읽어오기

```
1 // 저장된 쿠키 목록을 가져온다.
2 Cookie[] cookies = request.getCookies();
3 // 쿠키값을 저장할 문자열
4 String mycookie = null;
5
6 // 쿠키목록이 있다면
7 if (cookies != null) {
8     for (int i=0; i<cookies.length; i++) {
9         // 쿠키의 이름을 취득한다.
10        String cookieName = cookies[i].getName();
11        // 이름이 내가 원하는 값일 경우 값을 복사한다.
12        if (cookieName.equals("mycookie")) {
13            mycookie = cookies[i].getValue();
14            // 원하는 값을 찾으면 break;
15            break;
16        }
17    }
18 }
```

■ Session

- 각 클라이언트의 데이터를 서버의 메모리에 직접 저장 하고 사용하는 방식
- 데이터가 외부로 노출되지 않기 때문에 상대적으로 보안에 유리 (하지만 절대적이지 않음!)

■ Session 사용시간 설정(초단위)

```
1 session.setMaxInactiveInterval(30);
```

■ Session 데이터 사용

```
1 // 세션 저장하기 -> 모든 형태의 객체 저장 가능
2 String input = "Hello world";
3 session.setAttribute("mysession", input);
4
5 // 세션 읽기 -> 리턴값을 원래의 형태로 변환 필요함
6 String input = (String) session.getAttribute("mysession");
```

■ Session 삭제(개별, 전체)

```
1 session.removeAttribute("mysession");
```

```
1 session.invalidate();
```

■ JS

- 브라우저의 히스토리에 남지 않도록 하기 위해서는 js방식을 사용하면 안된다

`window.location = "이동할 페이지 URL";`

■ JSP

- servlet 컨트롤러에 의해 즉시 이동

`<% response.sendRedirect("이동할 페이지 URL"); %>`

■ HTML

- <form>요소의 submit처리에 의해 호출된 action 페이지는 요청에 따른 프로그램 로직을 수행한 후 결과 표시를 위한 페이지로 이동해야 한다.
이때, action페이지가 브라우저의 히스토리에 남지 않도록 하기 위해 <meta>태그를 이용한 방식을 사용해야 한다.

`<meta http-equiv="refresh" content="지연시간(초); url=이동할 페이지 주소">`

■ 상대경로, 절대경로

- / ←루트디렉토리
- ./ 또는 _ ←현재위치

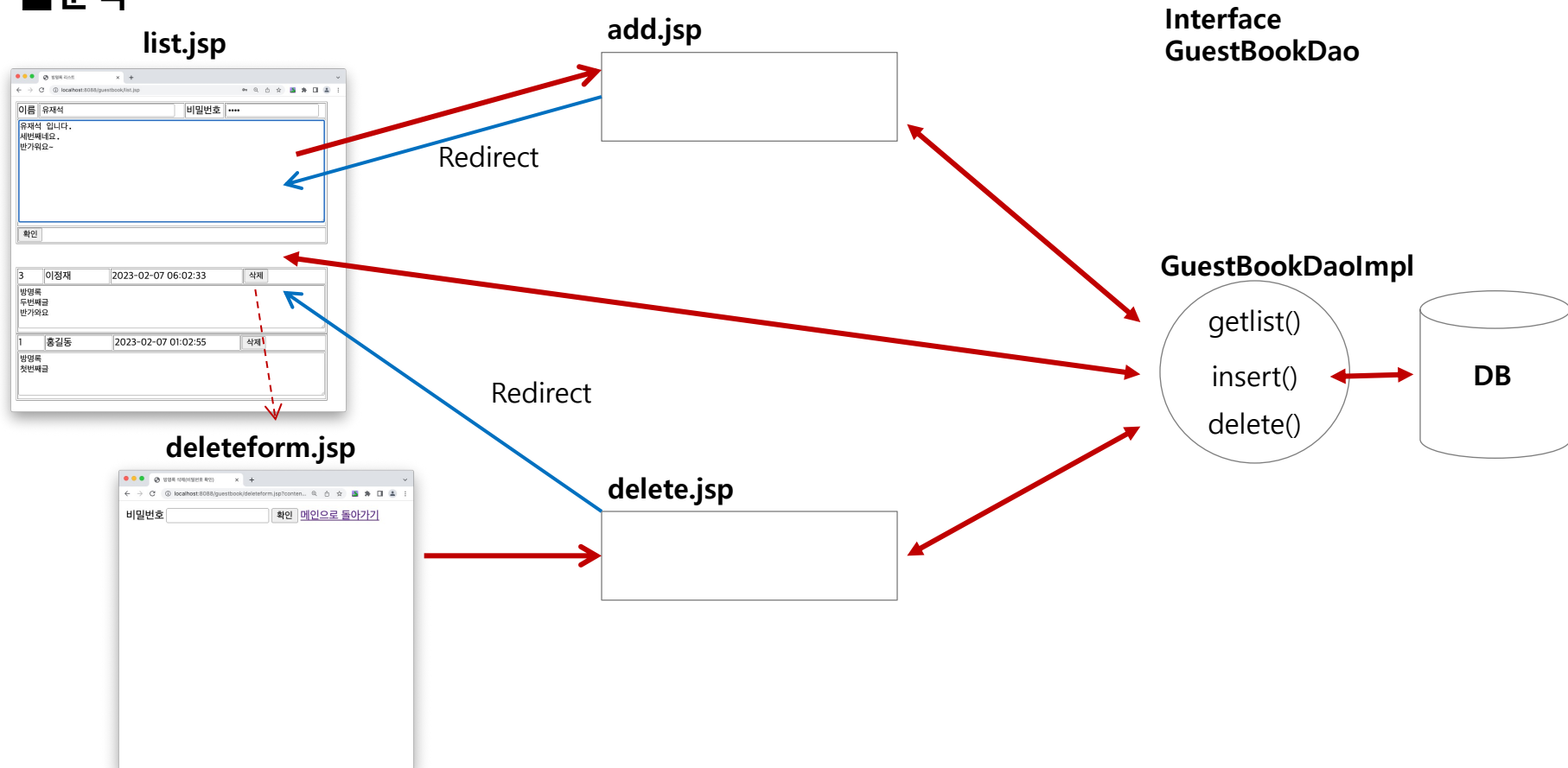
■ 상대경로, 절대경로

/ ← 루트디렉토리

./ 또는 _ ← 현재위치

06 guestbook 만들기

■ 분석



06 guestbook 만들기

■ 분석

방명록 리스트

이름 유재석 비밀번호

유재석 입니다.
세번째네요.
반가워요~

확인

| | | | |
|---------------------|-----|---------------------|----|
| 3 | 이정재 | 2023-02-07 06:02:33 | 삭제 |
| 방명록 두번째글 반가워요 | | | |
| 1 | 홍길동 | 2023-02-07 01:02:55 | 삭제 |
| 방명록 첫번째글 | | | |

방명록 삭제 (비밀번호 확인)

비밀번호 확인 [메인으로 돌아가기](#)

■ 분석

guestbook

| | | | |
|----------|----------------|----------|------|
| no | number | PK | 식별번호 |
| name | varchar2(80) | | 이름 |
| password | varchar2(20) | Not Null | 비밀번호 |
| content | varchar2(2000) | | 본문 |
| reg_date | date | | 등록일 |

```
create table guestbook(  
    no          number(10),  
    name        varchar2(80),  
    password    varchar2(20) not null,  
    content     varchar2(2000),  
    reg_date    date,  
    primary key(no)  
);
```

```
CREATE SEQUENCE seq_guestbook_no  
INCREMENT BY 1 START WITH 1 ;
```