

[dev-rary(데브러리)] JSP 빠른 스캔(원리X)

2. session과 response

前) 광고데이터 분석, SEO 최적화, SEM 세팅 및 관리

前) 기초프로그래밍, 웹개발, 앱개발 강의

前) 머신러닝을 활용한 데이터 분석

前) 기아자동차 광주공장 외주

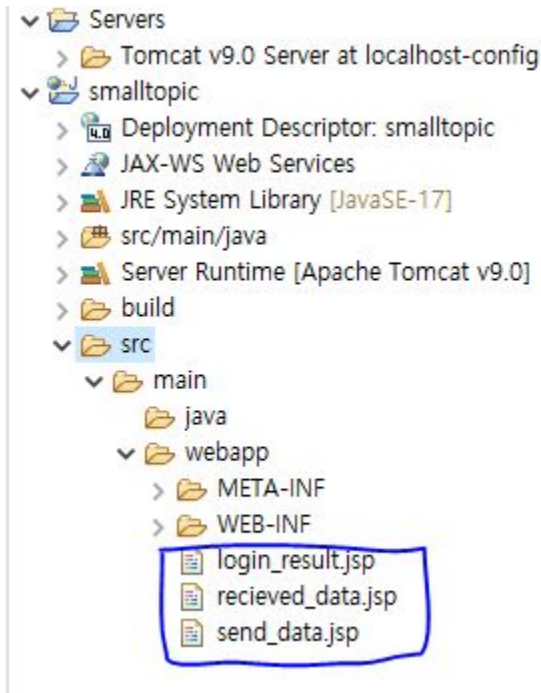
前) 리그오브 레전드 데이터 분석 등...

現) 국비반 강의 진행중

現) 유튜브 데브러리의 개발도서관 운영중

<https://www.youtube.com/channel/UCJJsnLh62qx00sN3yzRcW3Q>

기초 자바에서는 눈에 보이는 코드만 신경써도 되었는데
점점 여러 클래스파일을 넘나들며 호출 순서를 고려해야 하고
웹개발에서는 보이지 않는 데이터의 전송이 많아 어려우셨을 겁니다.
본 토막강의 영상에서는 정말 간단하게 메모리상의 변화만 다룹니다.
원리를 다루지 않기때문에 이 동영상만으로는 절대 100% 현업에서 요구하는
지식을 충족시킬수 없으니 심화된 내용을 공부하기 위한 토대로만 삼아주세요



지난 강의에 이어서 로그인 성공시 넘어갈 창인 **login_result.jsp**가 새로 생겼습니다.

이외의 프로젝트 구성은 지난 강의와 동일합니다.

recieved_data.jsp공간(페이지 이동시 삭제)

```
String formId = request.getParameter("id");  
String formPw = request.getParameter("pw");  
System.out.println("폼에서 받아온 id : " + formId);  
System.out.println("폼에서 받아온 pw : " + formPw);  
out.write("폼에서 받아온 id : " + formId + "</br>");  
out.print("폼에서 받아온 pw : " + formPw + "<br/>");
```

http 데이터(폼에서 전달받음)

id : abc1234
pw : 3434

자바 데이터

formId : abc1234
formPw : 3434

이제 로그인 처리 및 세션 생성부터 해보겠습니다.

먼저 지난 강의에 의해 send_data.jsp에서 폼에 입력한 데이터를

recieved_data.jsp로 전송해 자바 데이터로 저장하는것까지 성공했습니다.

```
send_data.jsp  recieved_data.jsp  login_result.jsp
1  <%@ page language="java" contentType="text/html; charset=EUC-KR"
2  pageEncoding="EUC-KR"%>
3  <%
4  String formId = request.getParameter("id");
5  String formPw = request.getParameter("pw");
6  System.out.println("폼에서 받은 id : " + formId);
7  System.out.println("폼에서 받은 pw : " + formPw);
8  out.write("폼에서 받은 id : " + formId + "<br>");
9  out.print("폼에서 받은 pw : " + formPw + "<br/>");
10
11  session.setAttribute("session_id", formId);
12 %>
13
14 <!DOCTYPE html>
15 <html>
16 <head>
17 <meta charset="EUC-KR">
```

세션 생성

비휘발성 메모리 공간(페이지 이동에도 데이터 유지)

세션 스토리지

session_id : abc1234

쿠키 스토리지

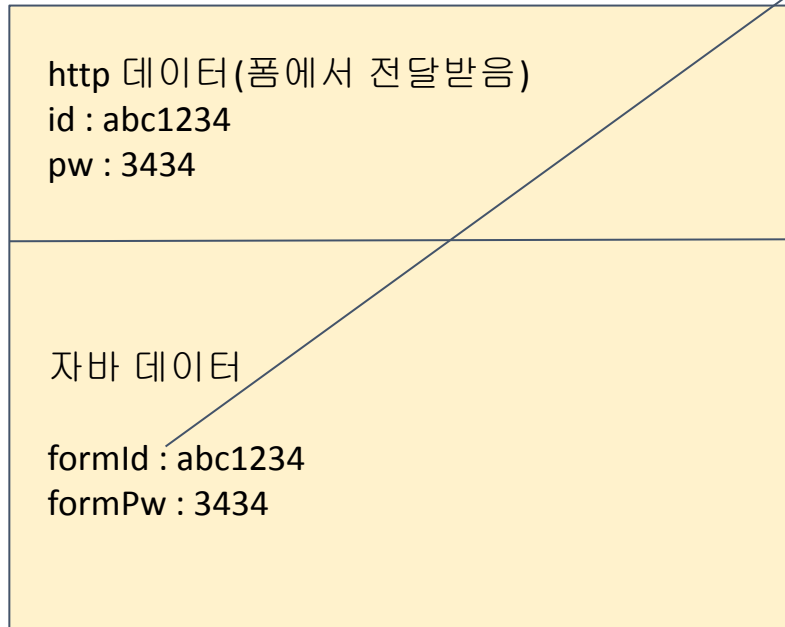
이제 휘발성 데이터 뿐만이 아닌 비휘발성 데이터까지 함께 다뤄야 합니다.

먼저 세션을 생성하겠습니다. 생성된 세션은 세션스토리지에 저장됩니다.

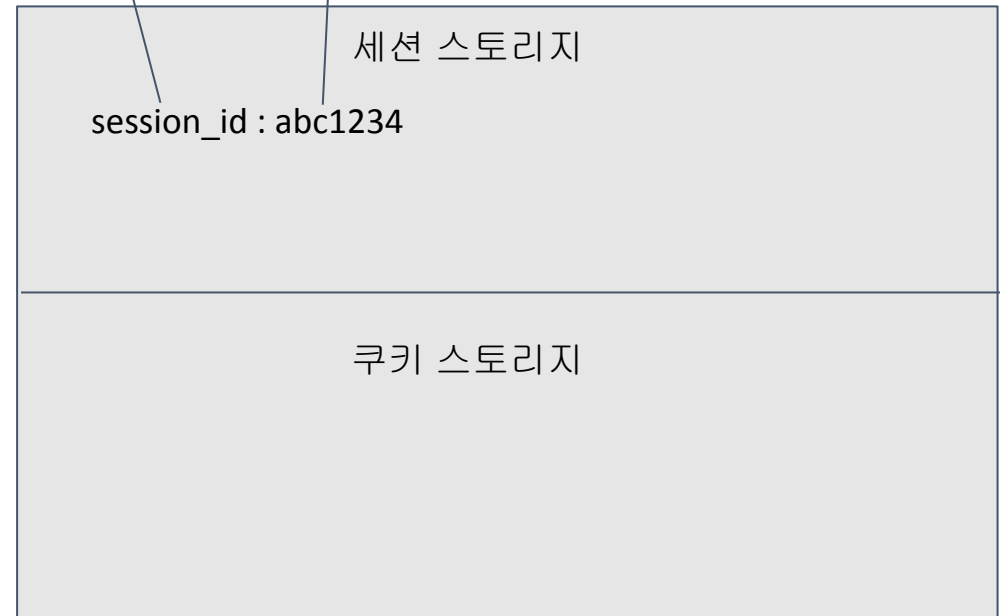
자바 데이터 formId를 저장값으로 처리

```
session.setAttribute("session_id", formId);
```

recieved_data.jsp공간(페이지 이동시 삭제)



비휘발성 메모리 공간(페이지 이동에도 데이터 유지)



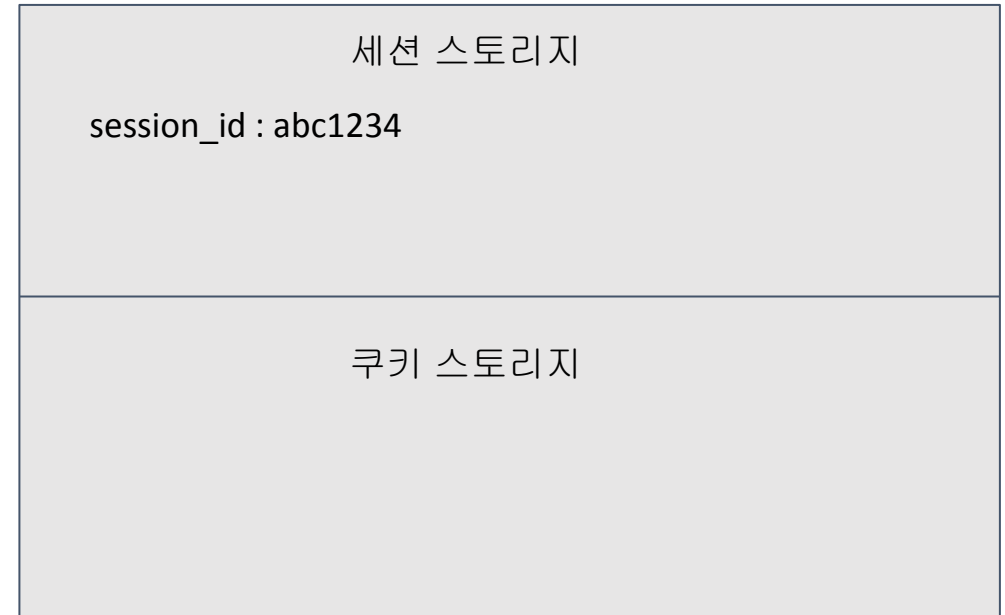
session.setAttribute(“세션명”, “세션값”); 형식으로 저장하는 구문입니다.

세션스토리지에 “세션명”을 변수명으로, “세션값”을 저장값으로 저장합니다.

비휘발성 메모리 공간(페이지 이동에도 데이터 유지)

```
out.write("폼에서 받은 id : " + formId + "<br>")
out.print("폼에서 받은 pw : " + formPw + "<br/>")

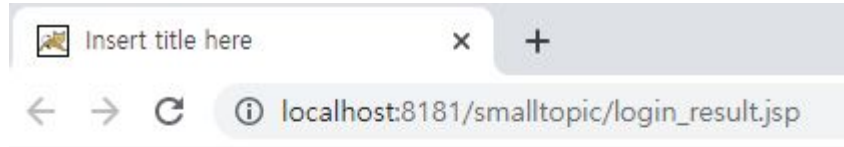
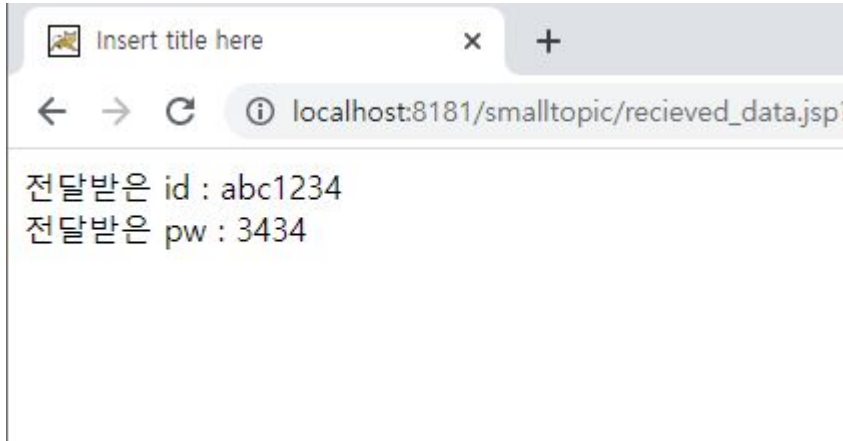
session.setAttribute("session_id", formId);
response.sendRedirect("login_result.jsp");
</>
```



이제 아래 `response.sendRedirect("목적지");` 구문을 이용해

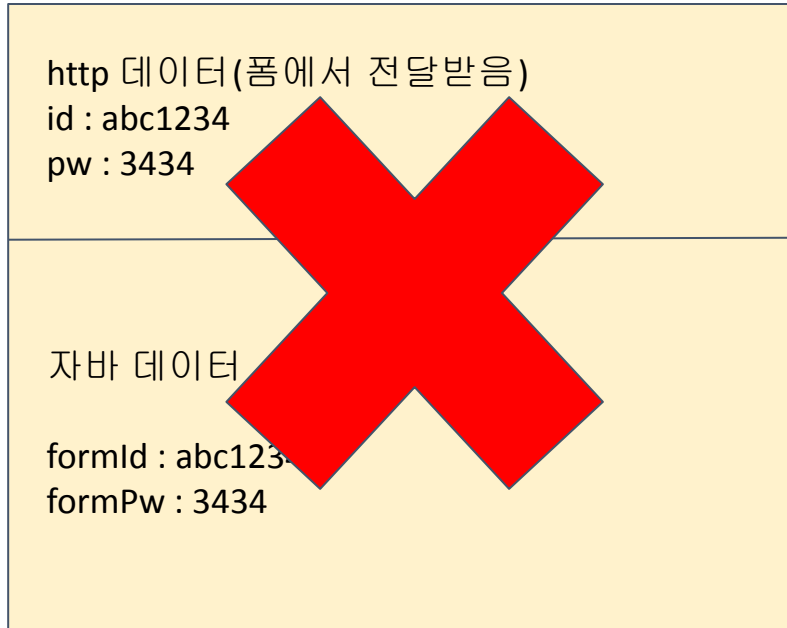
강제로 현재 보고 있는 페이지를 `recieved_data.jsp`에서 `login_result.jsp`로 옮겨줍니다.


```
response.sendRedirect("login_result.jsp");
```



위와 같이 `recieved_data.jsp`로 진입하면 그 즉시 바로 `login_result.jsp`로 보내주면서, 페이지가 변경되는데 이 때 휘발성 데이터 영역에도 큰 변화가 생깁니다.

recieved_data.jsp공간(페이지 이동시 삭제)

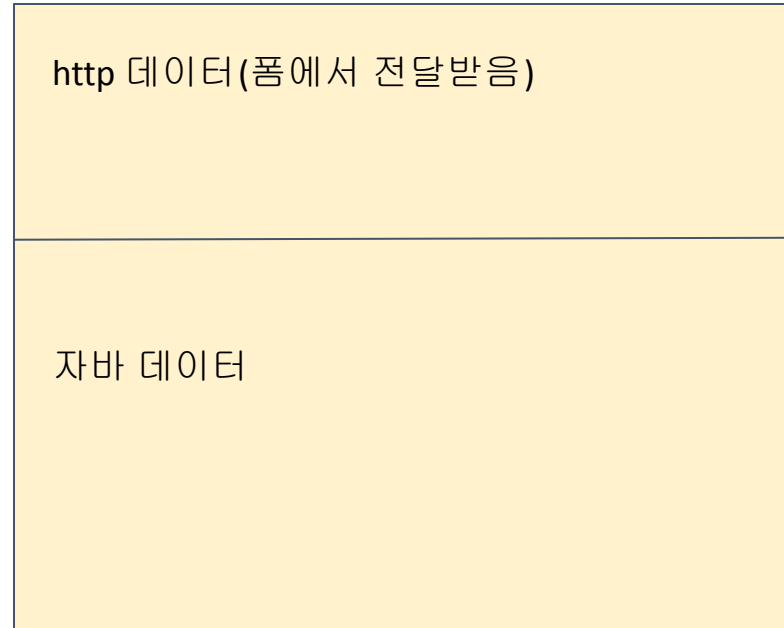


페이지 이동



내부 데이터
전부 삭제됨

login_result.jsp공간(페이지 이동시 삭제)

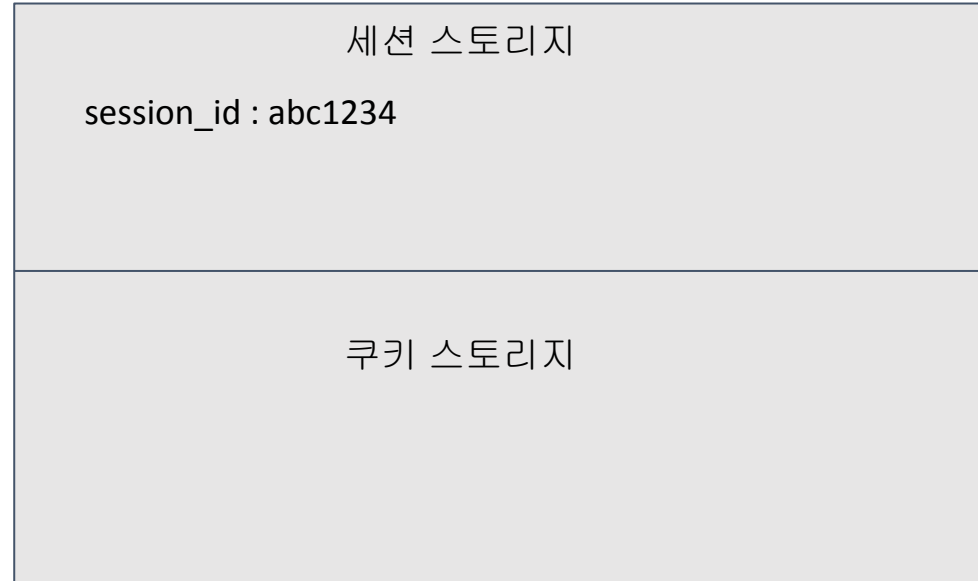


recieved_data.jsp -> login_result.jsp로 이동하면서

recieved_data.jsp공간이 삭제되고, 새로운 공간인 login_result.jsp

공간이 생기게 됩니다.

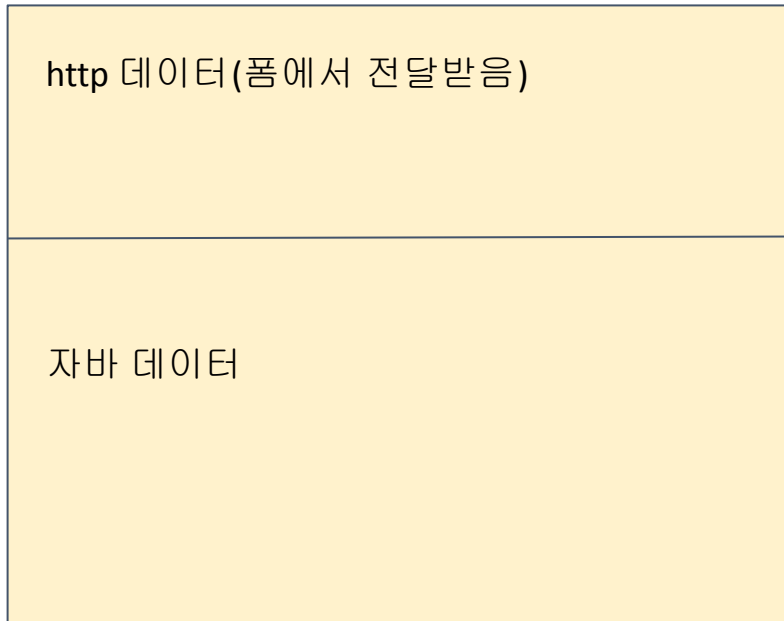
비휘발성 메모리 공간(페이지 이동에도 데이터 유지)



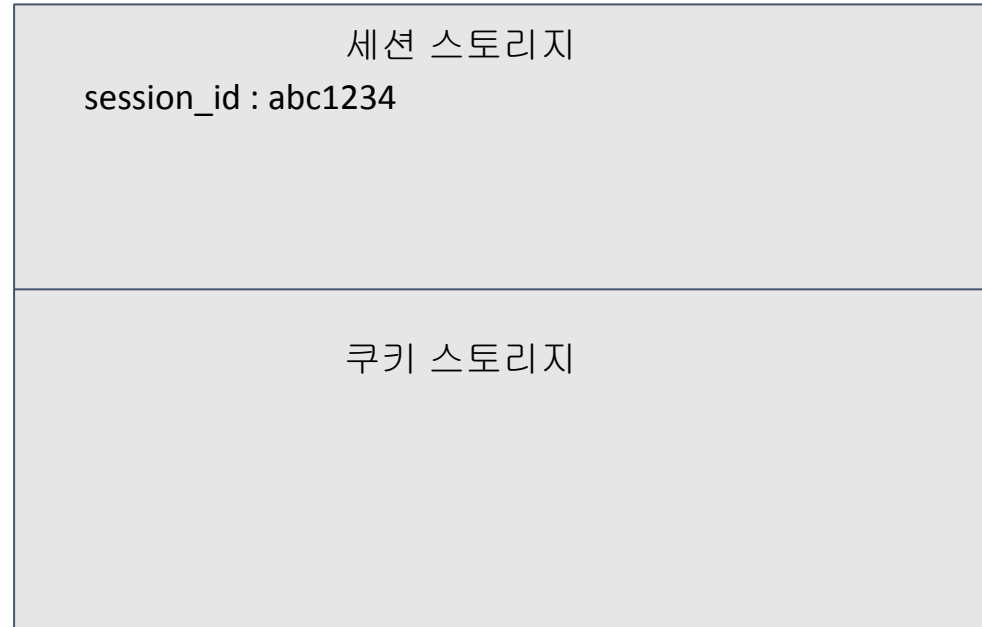
그러나, 페이지 이동과는 상관없이 세션스토리지는
저장 데이터를 유지합니다.

페이지 이동에 따라 `recieved_data.jsp`영역의 자료는 모두 사라지고
텅텅 빈 `login_result.jsp`와 자료가 유지된 세션스토리지만 남은것입니다.

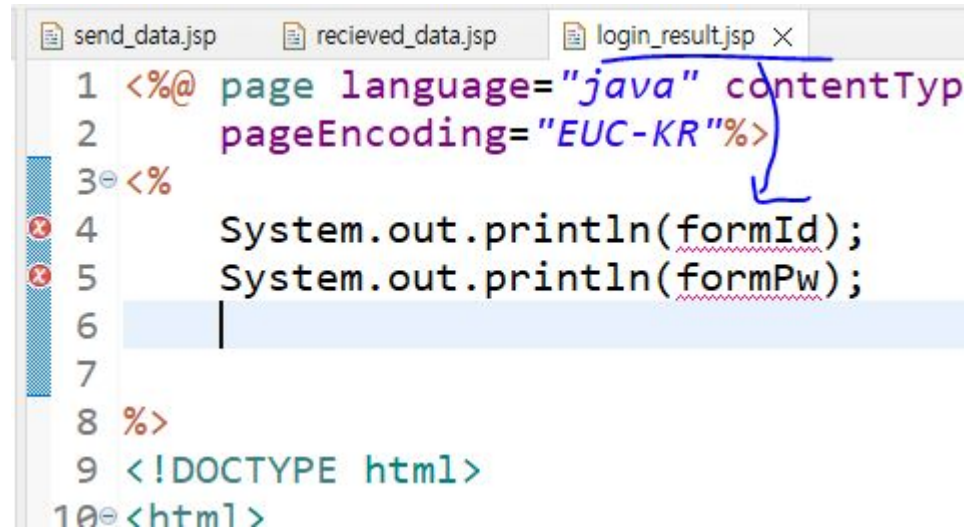
login_result.jsp 공간(페이지 이동시 삭제)



비휘발성 메모리 공간(페이지 이동에도 데이터 유지)



이제 옮겨온 login_result.jsp 페이지는 휘발성 메모리 공간에
모든 데이터가 삭제되었지만 세션스토리지에는 이전 페이지인
recevied_data.jsp에서 생성한 세션이 그대로 유지되고 있습니다.



```
1 <%@ page language="java" contentType
2     pageEncoding="EUC-KR"%>
3 <%
4     System.out.println(formId);
5     System.out.println(formPw);
6
7
8 %>
9 <!DOCTYPE html>
10 <html>
```

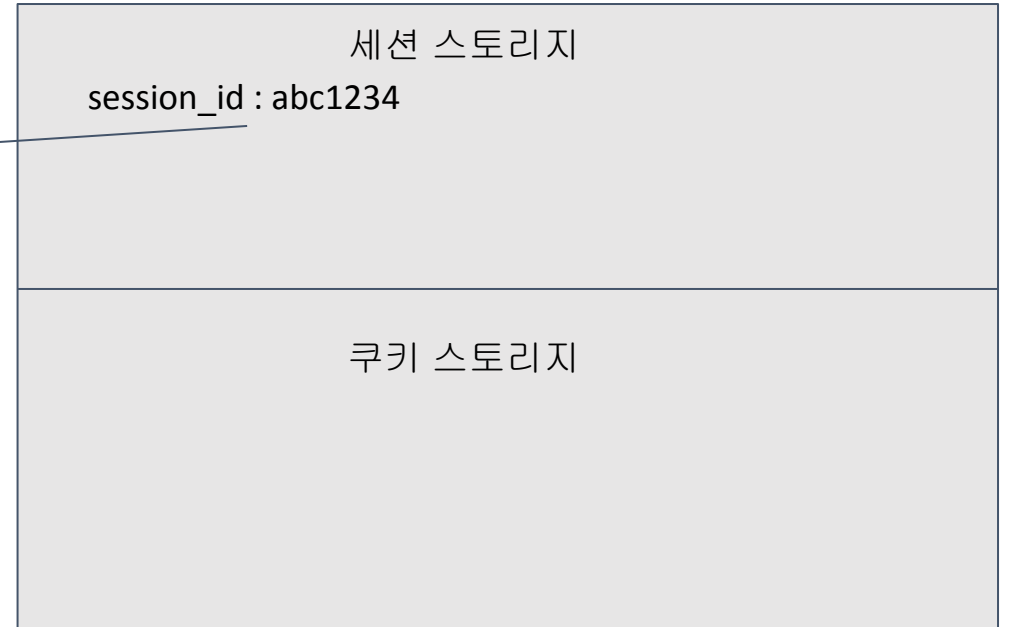
보시다시피 login_result.jsp에서는 이전 페이지에 있던 formId, formPw를 조회해보려 해도 에러가 납니다. 이동으로 삭제된 것을 볼 수 있습니다.

```
1 <%@ page language="java" contentType="text/html" %>
2 <%@ pageEncoding="EUC-KR"%>
3 <%
4     session.setAttribute("session_id", "abc1234");
5 %>
6
7 <%
8 <!DOCTYPE html>
```

- 아래의 폼 데이터 얻어오기와 비슷함!

```
<%
String formId = request.getParameter("id");
String formPw = request.getParameter("pw");
// ...
```

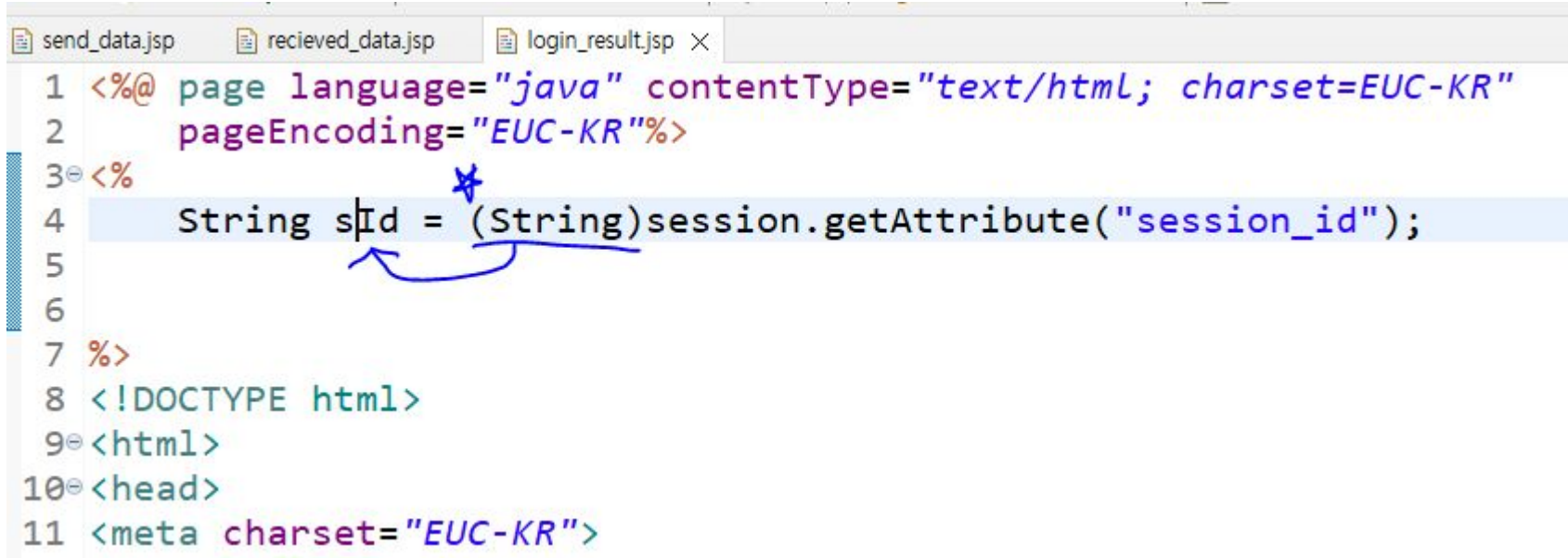
비휘발성 메모리 공간(페이지 이동에도 데이터 유지)



세션을 얻어오는 방법도 폼에서 얻어오던 문법처럼 세션 내 변수명을 적으면

매칭된 값을 가져오는 방식입니다.

위의 코드는 session_id에 매칭된 abc1234를 자바 자료로 가져오는 구문입니다.



```
1 <%@ page language="java" contentType="text/html; charset=EUC-KR"
2     pageEncoding="EUC-KR"%>
3 <%
4     String sId = (String)session.getAttribute("session_id");
5
6
7 %>
8 <!DOCTYPE html>
9 <html>
10 <head>
11 <meta charset="EUC-KR">
```

단, String 변수에 저장하기 위해서는 (String)을 붙여서 자료형변환을 해야 하는데
이유는 session.getAttribute(); 로 얻어오는 자료는 Object 형으로 리턴됩니다.
애매하다면 그냥 session.getAttribute() 왼쪽에 (String)을 붙인다고 암기하세요.

```
String sId = (String)session.getAttribute("session_id");
```

login_result.jsp 공간(페이지 이동시 삭제)

http 데이터(폼에서 전달받음)

자바 데이터

sId : abc1234

비휘발성 메모리 공간(페이지 이동에도 데이터 유지)

세션 스토리지

session_id : abc1234

쿠키 스토리지

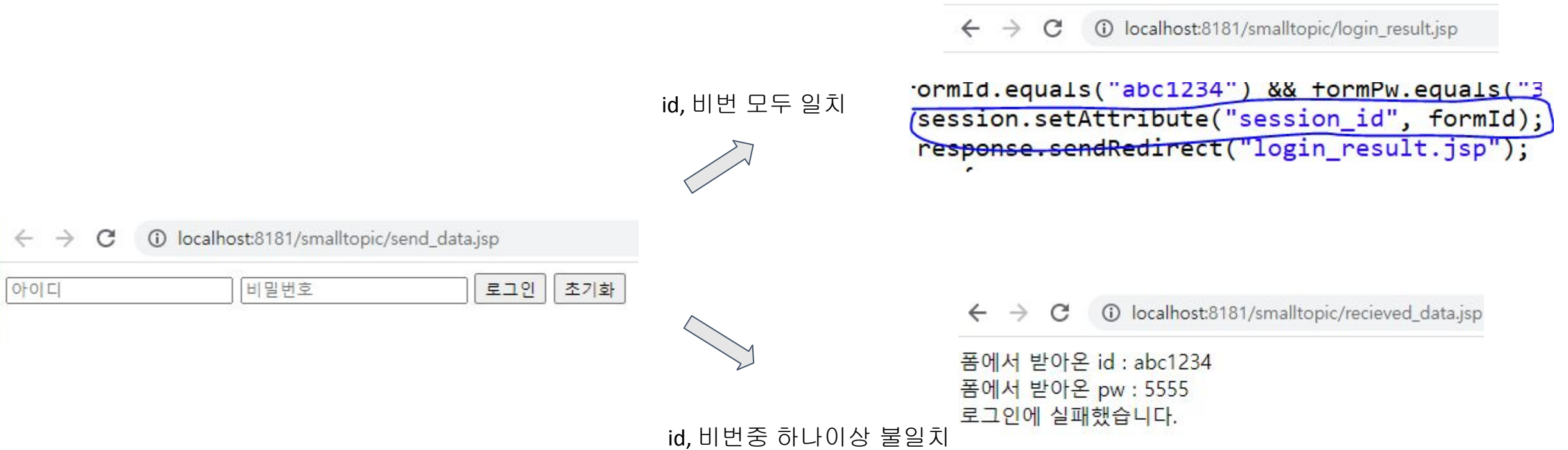
이제 세션에 있던 abc1234라는 문자열을 session_id를 조회해서 얻은 뒤
sId라는 변수명으로 휘발성 메모리 공간인 login_result.jsp에 저장했습니다.


```
send_data.jsp x recieved_data.jsp x login_result.jsp
1 <%@ page language="java" contentType="text/html; charset=EU
2   pageEncoding="EUC-KR"%>
3 <%
4   String formId = request.getParameter("id");
5   String formPw = request.getParameter("pw");
6   System.out.println("폼에서 받은 id : " + formId);
7   System.out.println("폼에서 받은 pw : " + formPw);
8   out.write("폼에서 받은 id : " + formId + "<br>");
9   out.print("폼에서 받은 pw : " + formPw + "<br/>");
10
11   if(formId.equals("abc1234") && formPw.equals("3434")){
12       ↳ session.setAttribute("session_id", formId);
13       ↳ response.sendRedirect("login_result.jsp");
14   } else {
15       ↳ out.write("로그인에 실패했습니다.");|
16   }
17
```

문자열 비교는 .equals()로 함에 주의!

그럼 이제 코드를 조금 수정해서, 로그인 로직을 구현해보겠습니다.

폼에서 보낸 아이디와 비밀번호가 각각 abc1234, 3434인지 비교하는 조건문을
추가합니다. 추후에 DB에서 가져온 데이터로 비교하도록 수정할 예정입니다.



임의로 정한 아이디, 비번과 일치하면 `login_result.jsp`로 넘어가고

아니면 로그인에 실패했습니다 라는 창이 뜹니다.

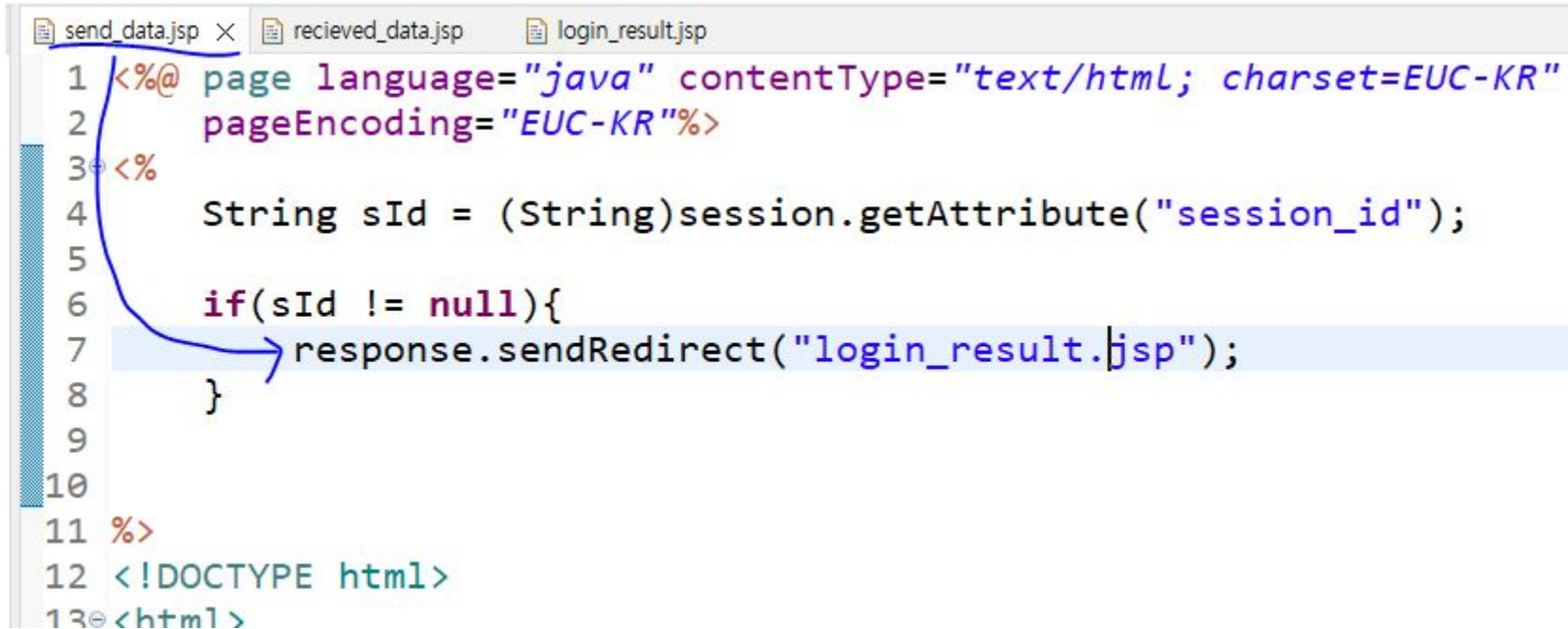
성공시 `login_result.jsp`로 넘어가기 전에 세션생성이 되는 코드가 있습니다.

```
send_data.jsp  recieved_data.jsp  *login_result.jsp X
1  <%@ page language="java" contentType="text/html; charset=EUC-KR"
2  pageEncoding="EUC-KR"%>
3  <%
4      String sId = (String)session.getAttribute("session_id");
5
6      if(sId == null){
7          response.sendRedirect("send_data.jsp");
8      }
9
10
11 %>
12 <!DOCTYPE html>
13 <html>
14 <head>
15 <meta charset="EUC-KR">
16 <title>Insert title here</title>
17 </head>
18 <body>
19 <h1><%= sId %>님 로그인을 환영합니다!</h1>
20 </body>
21 </html>
```

로그인 성공후 세션에 **session_id** 라는 명칭의 세션이 있는지 조회해 저장합니다.

session.getAttribute()는 “없는” 세션 명칭을 조회시 **null**을 저장합니다.

null 여부를 검사해 **null**이면 로그인창으로, 아니면 환영문구를 보여줍니다.



```
1 <%@ page language="java" contentType="text/html; charset=EUC-KR"
2     pageEncoding="EUC-KR"%>
3 <%
4     String sId = (String)session.getAttribute("session_id");
5
6     if(sId != null){
7         response.sendRedirect("login_result.jsp");
8     }
9
10
11 %>
12 <!DOCTYPE html>
13 <html>
```

The screenshot shows a code editor with three tabs: `send_data.jsp`, `recieved_data.jsp`, and `login_result.jsp`. The `send_data.jsp` tab is active, displaying the code above. A blue arrow points from the `if(sId != null){` block to the `login_result.jsp` tab, indicating a redirect action.

마찬가지로 로그인한 사람이 `send_data.jsp` 로 접근하면

로그인창을 보여줄 필요가 없으니 바로 `response.sendRedirect()`를 이용해

`login_result.jsp`로 보내줍니다.

해당 로직은 세션은 사용자가 생성하는게 아니고

오직 **JSP** 코드를 통해서만 생성되니

로그인에 성공한 경우만 세션이 저장된다는 점을 이용한 것입니다.

아직은 고정 문자열을 활용해 로그인 로직을 처리하지만

추후 **JDBC**를 공부한 뒤에는 **DB연동**을 통해서 회원인지 아닌지 여부를 조사합니다.