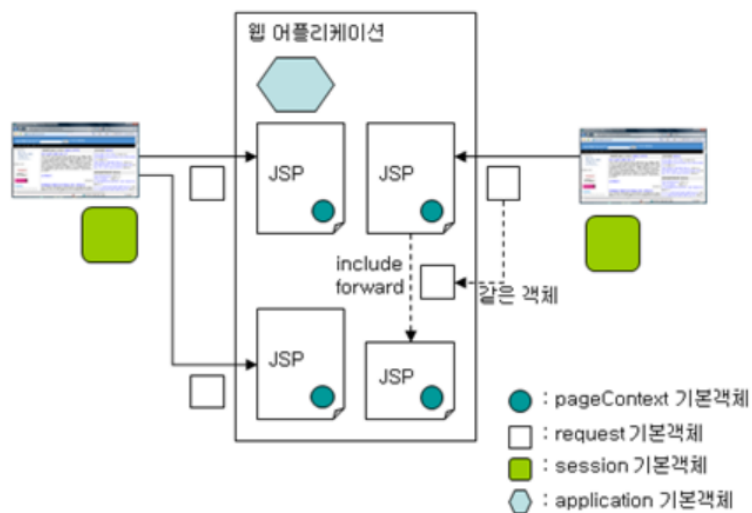


221024 수업 -1


🕒 작성일시	@2022년 10월 24일 오전 9:21
≡ 키워드	
📄 유형	
📎 자료	
☑ 복습	<input type="checkbox"/>

네 영역

- PAGE 영역 - 하나의 JSP 페이지를 처리할 때 사용되는 영역
- REQUEST 영역 - 하나의 HTTP 요청을 처리할 때 사용되는 영역
- SESSION 영역 - 하나의 웹 브라우저와 관련된 영역
- APPLICATION 영역 - 하나의 웹 어플리케이션과 관련된 영역



KOSA_IT

 https://cafe.naver.com/kosait?iframe_url_utf8=%2FArticleRead.nhn%253Fclubid%3D30787723%2526articleid%3D729%2526referrerAllArticles%3Dtrue

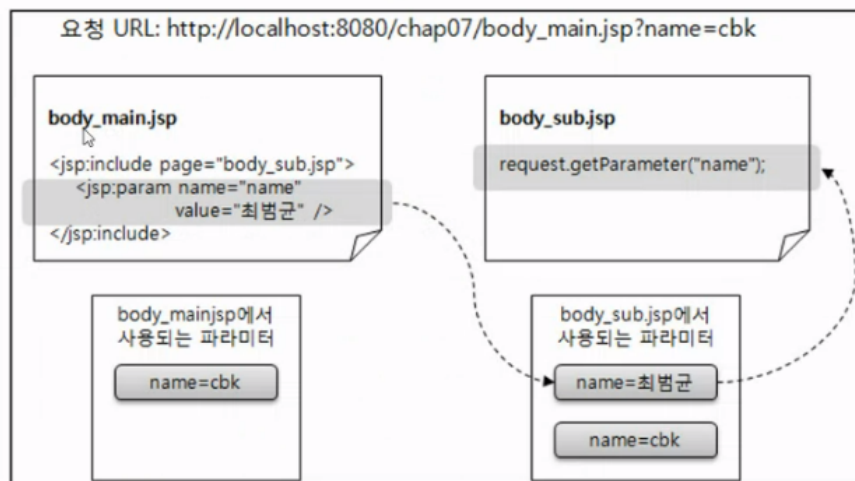


궁금한게 있을 땐
카페 스마트봇

<jsp:param> 액션 태그

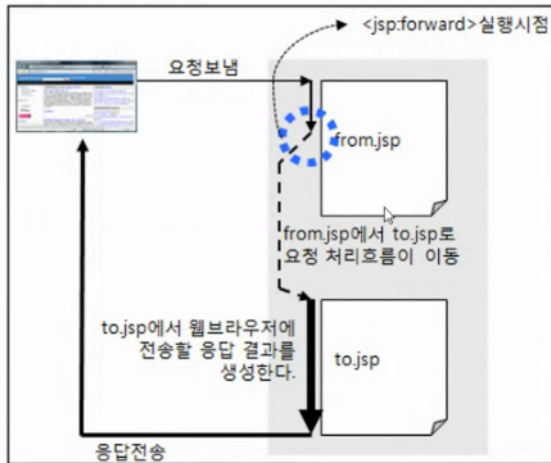
<jsp:param> 액션 태그의 동작 방식

- 기존 파라미터는 유지하고 파라미터를 새로 추가
 - <jsp:include>로 포함되는 페이지에서만 유효



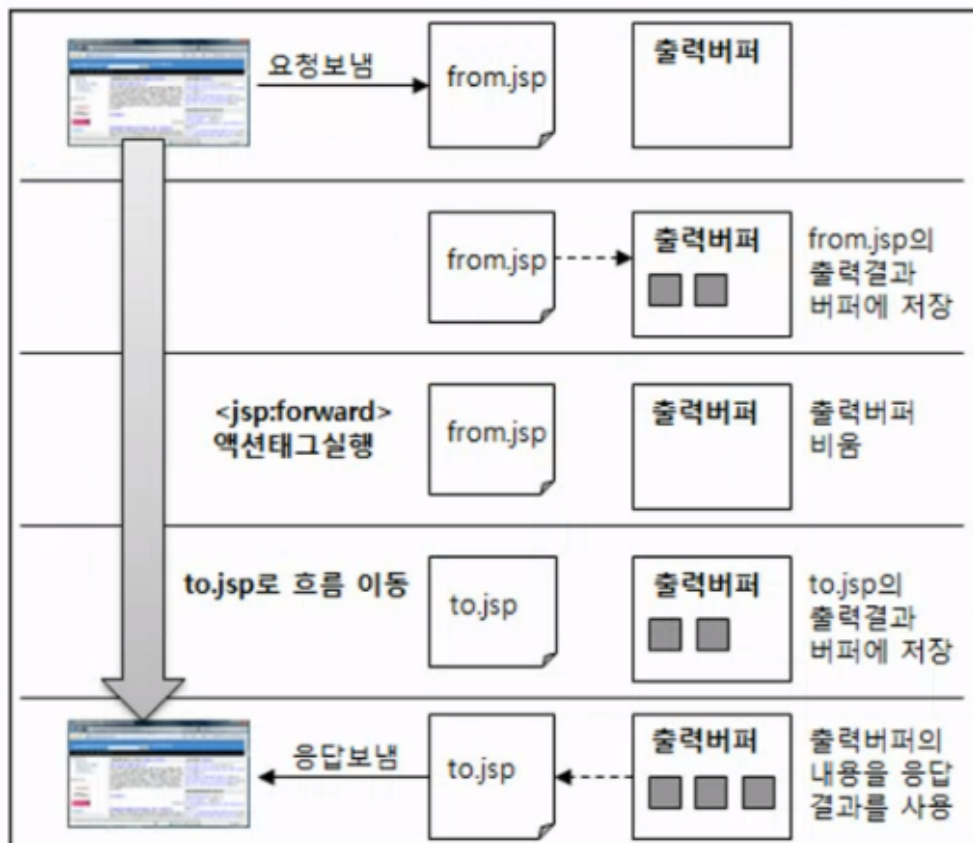
<jsp:forward> 액션 태그

- 하나의 JSP 페이지에서 다른 JSP 페이지로 요청 처리를 전달할 때 사용
- 동작 방식



<jsp:forward> 액션 태그

- 구문
 - `<jsp:forward page="이동할 페이지" />`
- 출력 버퍼와의 관계



- `include`는 디자인 시 공통 페이지를 가져올 때 사용
- `forward`는 특정 조건에 따라 다른 페이지 서비스 가능
 - 페이지 이동 시 출력 버퍼의 내용을 싹 비움

`<jsp:include ... >`

- `include` 페이지는 request 공유 가능

- 변수를 만들어서 include 페이지 전달 가능
-

[요청에 대한 흐름 제어]

1. include - 공통 디자인
2. forward - 통제 권한

공통점 : request 객체를 공유

차이점 : 제어권

> include (제어권을 다시 가지고 온다)

> forward (제어권을 넘겨준다)

POINT

- 요청 주소는 동일한데 다른 page의 내용을 서비스할 수 있다
 - forward 된 페이지는 자신의 데이터가 의미 없음
 - Buffer를 비우고 다른 페이지 내용을 담기 때문
-

에러 처리

- <%@ page errorPage="" %>
- 서버에서 에러 발생시 대신 출력 해줌
- 최종 배포 전 만들어 처리
- 처리 방법
 - 페이지 마다 설정
 - 전역 설정
 - web.xml에 작성

→ 수정시 서버 재 시작

Cookie

- 메모리 쿠키
- 파일 쿠키
 - 소멸시기를 결정하면 파일쿠키로 생성
- 단점 : 크기 제한(4096byte)
 - 보안(사용자 삭제 ... 암호화 해독 해커가 ...)

쿠키 객체 생성

```
Cookie mycookie = new Cookie("cname", "1004");
```

내 웹 서버에 접속한 브라우저(client)에게 cookie 전달 (write) >> response

```
response.addCookie(mycookie);
```

웹 브라우저가 가지는 모든 쿠키

```
<%  
  
//웹 브라우저가 가지는 모든 쿠키  
if(cs != null || cs.length > 0){  
    for(Cookie c : cs){  
        out.print("c.getName" + c.getName());  
        out.print("c.getValue" + c.getValue());  
        out.print("c.getMaxAge" + c.getMaxAge()); // -1 소멸 시간이 없다 : 메모리 쿠키  
        out.print("c.getDomain" + c.getDomain());  
    }  
}  
%>
```

메모리 쿠키

- client가 강제로 지우지 않는 한 유지됨
- 브라우저가 닫기 전까지 유효
- 소멸 타입 : `getMaxAge() >> -1`

파일 쿠키

- 소멸 시간을 가지고 있음
- client 강제로 지우지 않는 한 정해진 시간까지 유효
- `setMaxAge(60) → 60초`
 - 30일 : $(30 \times 24 \times 60 \times 60)$ 일 * 시간 * 분 * 초

쿠키 사용 예제

- 팝업창 : 오늘 하루 보지 않기
-

Local storage(key, value) 형태

- 장점) 크기, value(객체 사용 가능 : JSON)

Server (웹 서버 (메모리) + DB 서버)

1. server memory : session 객체
 - 사이트 접속하는 사용자마다 고유하게 부여하는 메모리 (객체)
2. server memory : application 객체
 - 사이트 접속하는 모든 사용자에게 부여하는 메모리(객체)
 - server 영속적 : 파일.txt (관리 힘들)

- DB server : 보안, 영속적 데이터 관리 >> 비용 추가됨 >> cloudDB 사용

session 객체 : 서버(Web)에 접속한 사용자 마다 고유하게 부여되는 객체

- 고유성을 보장하기 위해 key 값을 생성해 session 객체마다 key값을 부여해 Client에게도 전달함
- 클라이언트마다 고유하게 부여 (session 객체)
- 활용 : 클라이언트마다 고유하게 부여 (session 객체)
→ 로그인 ID, 장바구니(사용자마다 다른), 사용자 정보 (각각 다른 정보)

1. 서버에 요청 보내기
2. session ID (key 값) 생성됨 -> session 객체가 키를 가짐
→ sessionID response 함(cookie)
a. JSESSIONID : 5D52BFC6ADD599DE5E42AA37BE490A74
3. 동기화

ex) 로그 아웃 할 경우

1. Client 로그아웃 버튼을 클릭
2. 로그아웃 판단
3. key 값 가지고 와서
4. key (이름표)
5. 객체를 찾아서
6. abandon (소멸)
7. invalidate

```
<%
    Date time = new Date();
    SimpleDateFormat formatter = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    %>

    <h3>session 객체 정보</h3>
    session 객체의 식별값 : <%= session.getId() %>
```



```

<hr>
<%
    time.setTime(session.getCreationTime());
%>
[session이 생성된 시간] : <%= formatter.format(time) %>
<hr>
<%
    time.setTime(session.getLastAccessedTime());
%>
[session 마지막 접속 시간(client)] : <%=formatter.format(time) %>

```

session 객체의 유효 Scope

- webApp 전체
- 어떤 페이지든 저장한 session 객체 사용 가능

session 객체 활용

- 접속 브라우저마다 sessionID 값을 가지고 객체 관리
- `session.setAttribute("id", request.getParameter("userid"))`
: 접속 사용자마다 고유한 변수 생성
- 세션에 값 쓰기

```

<%
    String userid = request.getParameter("userid");
    //POINT
    session.setAttribute("id", userid);
%>

```

- 세션에서 값 가져오기

```

<%
    String id = (String)session.getAttribute("id");
%>

```