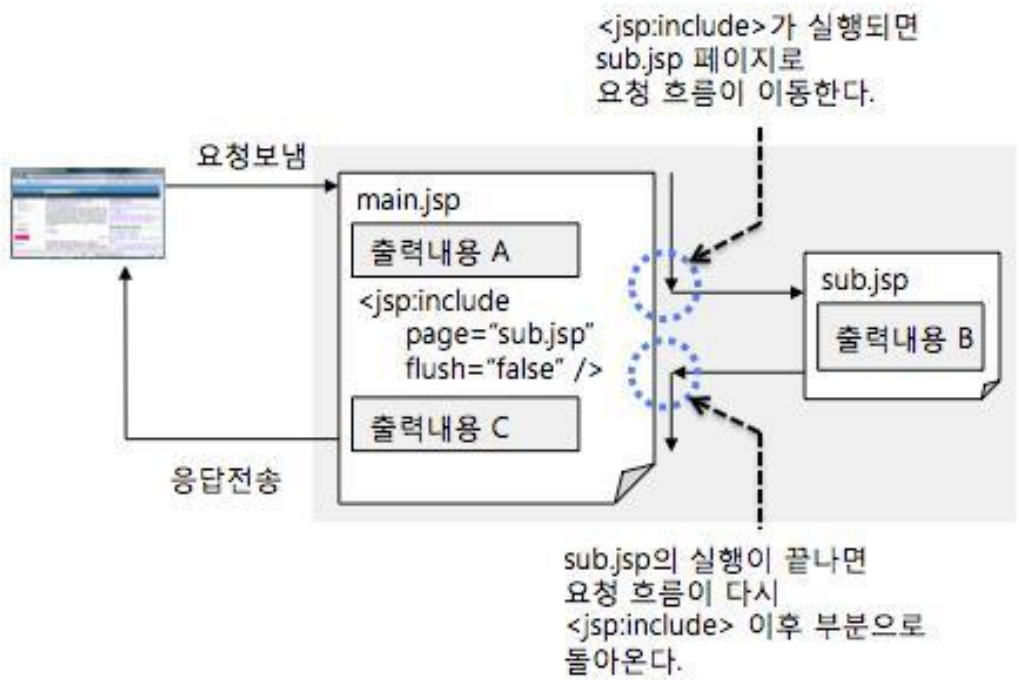
4강 자료

페이지 모듈화 <jsp:include> 액션태그 사용

- <jsp:include> 동작 방식



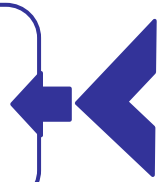
페이지 모듈화 <jsp:include> 액션태그 사용

- <jsp:include> 액션 태그 형식

```
<jsp:include  page="포함할 페이지 이름"  flush="false" />
```

  - page 포함할 jsp 페이지의 경로 지정
  - flush 포함하기 전에 출력 버퍼를 flush할 것인지를 지정, 기본값은 false
    include 이전의 내용을 flush하면, response.setHeader()를 사용하여
    페이지 헤더 변경 불가능.
  - [리스트 7.1] [리스트 7.2] 참고

```
Main 페이지의 내용
<jsp:include page="sub.jsp" flush="false" />
Main 페이지에서 include 이후의 내용
```



sub.jsp 페이지의 내용

페이지 모듈화 <jsp:include> 액션태그 사용

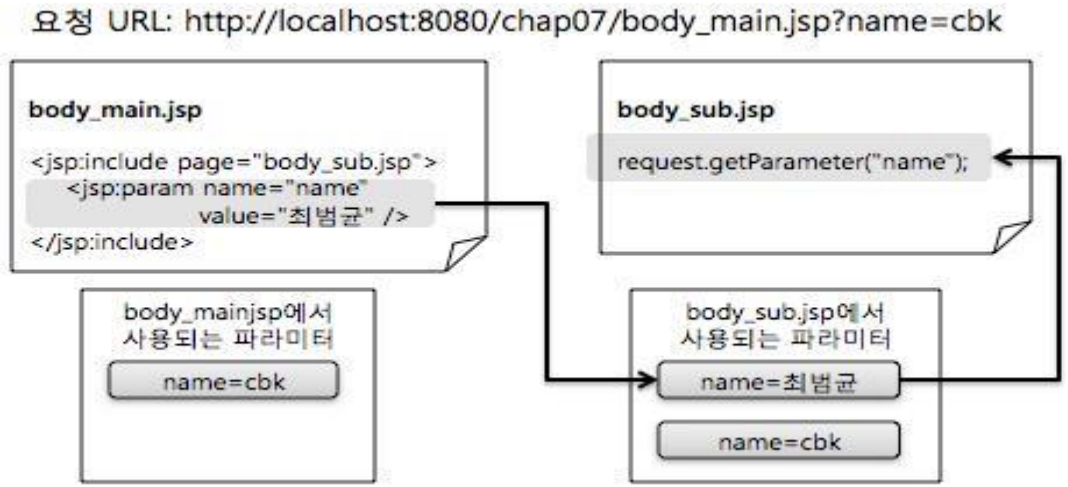
- 페이지 레이아웃 만들기 추가 예제 확인
  - [chap06-include] 프로젝트 확인

<jsp:param> 액션 태그

- <jsp:include>를 사용하여 포함하는 페이지에 파라미터 전달
- 사용 예

```
<jsp:include page="top.jsp" flush="false">
    <jsp:param name="param1" value="value1" />
    <jsp:param name="param2" value="value2" />
</jsp:include>
```

- <jsp:param>으로 전달하는 파라미터의 이름이 중복되는 경우



<jsp:param> 액션 태그

- [리스트 7.8] [리스트 7.9] 참고

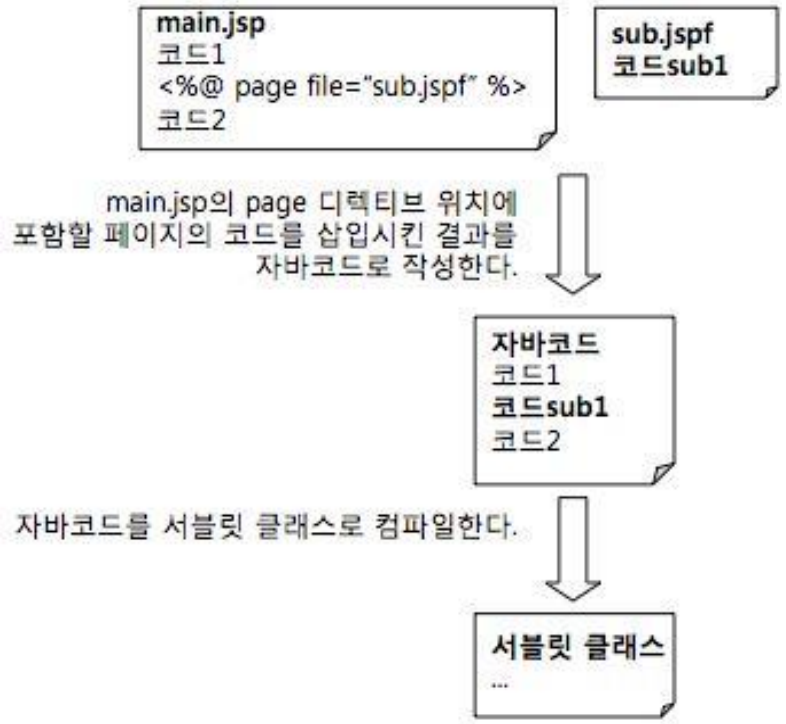
Body_main.jsp를 실행할 때 URL에는
localhost:8080/..../body_main.jsp?name="hgd" 형식으로 요청

```
<%-- body_main.jsp" --%>
include 전 name 파라미터 값: <%= request.getParameter("name") %>
<jsp:include page="body_sub.jsp" flush="false">
   <jsp:param name="name" value="홍길동" />
</jsp:include>
include 후 name 파라미터 값: <%= request.getParameter("name") %>
```

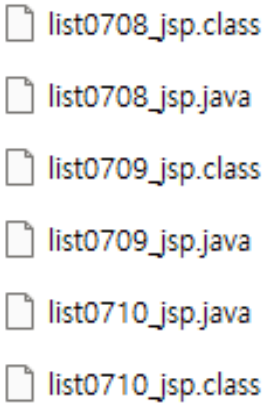
```
<%-- body_sub.jsp --%>
Name 파라미터 값 목록 확인
<ul>
<%
   String[] names = request.getParameterValues("name");
   for(String name: names) {
%>
      <li><%= name %></li>
<%
   }
%>
```

<% include %> 디렉티브를 이용한 중복 코드 삽입

- <% include %> 디렉티브의 처리 방식



jsp 파일이 JAVA 파일로 변환된 결과 위치
?\workspace\.metadata\.plugins\org.eclipse.wst.server.core\tmp1\work\Catalina\localhost\프로젝트 이름\org\apache\jsp에서
<jsp:include> 액션 태그를 사용할 때의 변환 결과와
<% include %> 디렉티브를 사용할 때의 결과 비교



<% include %> 디렉티브를 이용한 중복 코드 삽입

- [리스트 7.10] [리스트 7.11] 참고

```
<%-- list0710.jsp -->
<%
   int num1 = 10;
%>
<%@ include page="list0711.jspf" %>
공통변수 확인 <%= num2 %>
```

```
<%-- list0711.jspf --%>
가져온 num1은 <%= num1 %>
<%
   int num2 = 10;
%>
```

<% include %>없이 코드 자동 포함

- 사용 예

```
<%@ include file="aa.jspf" %>
<html>
…
<%@ include file="footer.jspf" %>
```

```
<%-- web.xml -->
<jsp-config>
   <jsp-property-group>
      <url-pattern>/view/*</url-pattern>
      <include-prelude>aa.jspf</include-prelude>
      <include-coda>footer.jspf</include-coda>
   </jsp-property-group>
</jsp-config>
```

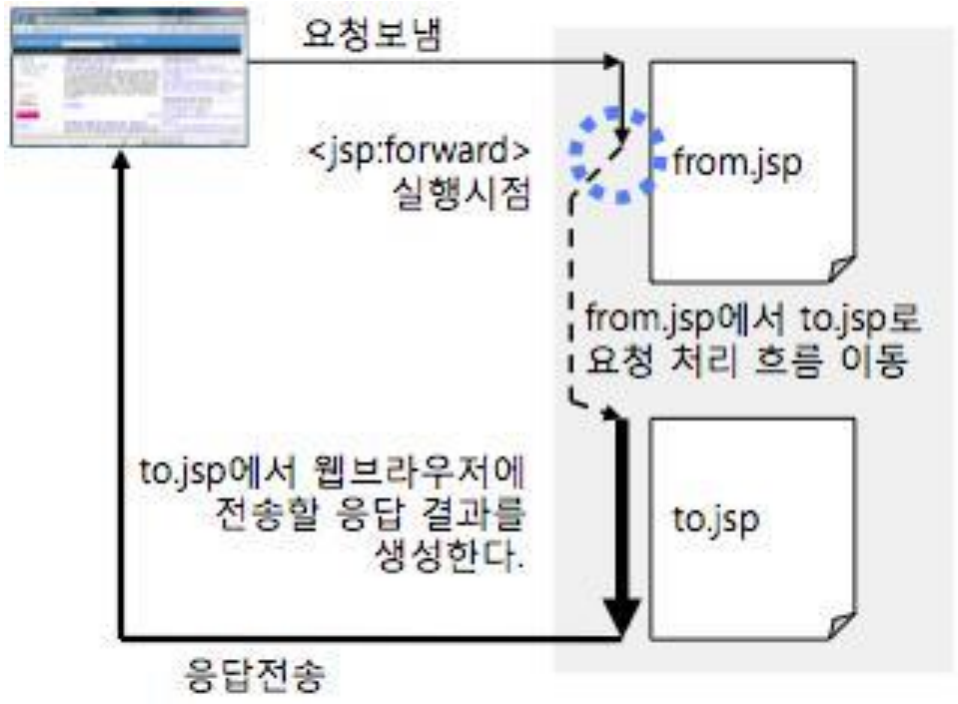
- ✓ <jsp-property-group>
  JSP의 프로퍼티를 포함
- ✓ <url-pattern>
  지정된 URL로 요청되는 경우
- ✓ <include-prelude>
  지정된 jsp 코드 조각을 요청 페이지의 앞에 자동 포함
- ✓ <include-coda>
  지정된 jsp 코드 조각을 요청 페이지의 뒤에 자동 포함

<%@ include %>와 <jsp:include> 비교

비교 항목	<jsp:include> 액션태그	<%@ include %> 디렉티브
처리 시간	요청 시간에 처리	JSP 파일을 JAVA로 변환할 때 처리
기능	별도의 파일로 요청 처리 흐름 이동	현재 파일에 삽입
데이터 전달 방법	request 기본 객체나 <jsp:param>을 이용한 파라미터 전달	페이지 내의 변수를 선언한 후, 변수에 값 저장
용도	화면 레이아웃의 일부분을 모듈화할 때 주로 사용	다수의 JSP 페이지에서 공통으로 사용하는 변수를 지정하는 코드

<jsp:forward>를 이용한 페이지 이동

- 다른 페이지로 요청 처리를 전달할 때 사용



1. from.jsp로 요청
2. from.jsp는 <jsp:forward> 액션 태그 실행
3. 요청 흐름이 to.jsp로 이동(이때 from.jsp에서 생성한 내용은 버퍼에서 삭제됨)하면서, from.jsp의 request와 response 기본 객체를 to.jsp로 전달
4. to.jsp에서 결과 생성
5. to.jsp가 생성한 결과를 사용자에 전달

❖ 주의:
from.jsp 페이지에서는
<%@ page buffer="none" %>으로 설정
from.jsp에서 flush가 이루어진 후에는
<jsp:forward>가 실행되지 않기 때문

<jsp:forward> 페이지 이동 사용 예

- [리스트 7.21] 참고

```
<%
    String code = request.getParameter(“code”);
    String viewPageURI = null;
    if(code.equals(“A”) {
        viewPageURI = “a.jsp”;
    } else if(code.equals(“B”) {
        viewPageURI = “b.jsp”;
    } else if(code.equals(“C”) {
        viewPageURI = “c.jsp”;
    }
%>
<jsp:forward page=“<%= viewPageURI %>” />
```

<jsp:forward> 파라미터 전달

- <jsp:param> 액션 태그를 사용하여 파라미터 전달 가능
  - <jsp:include> 액션 태그에서 <jsp:param> 액션 태그를 사용하는 방법과 같음

Page 속성 값의 파일 경로

- 절대 경로
  - context root부터 "/to/to.jsp"
  - 절대 경로를 사용하는 것이 관련 파일을 찾기에 편리
- 상대 경로
  - 현재 위치부터 "./../to/to.jsp"
  - 이동할 페이지가 현재 폴더 위치와 같거나, 현재 폴더의 하위 폴더에 있을 때

<jsp:param>의 문제점 해결

- <jsp:param> 액션 태그를 사용해서는
  - String 타입의 데이터만 전달 가능
- request 기본 객체 사용
  - request.setAttribute("이름", 값)

```
<%
    Calendar cal = Calendar.getInstance();
    request.setAttribute("calendar", cal);
%>
```

  - request.getAttribute("이름")

```
<%
    Calendar cal = (Calendar)request.getAttribute("calendar");
%>
현재 시간은 <%= cal.get(Calendar.HOUR) %> 시 <%=
cal.get(Calendar.MINUTE) %>
분 <%= cal.get(Calendar.SECOND) %> 초 이다.
```

JavaBean 사용

- JavaBean = Java 객체
- JavaBean 예: [리스트 8.1] 참고(jsp가 아닌 java 파일)

```
Import java.util.Date;

public clas MemberInfo {
    private String id;
    private String password;
    private String name;
    private Date registerDate;
    private String email;

    // getter와 setter는 eclipse에서 자동 생성
    // 편집기에서 마우스 오른쪽 버튼, 단축메뉴에서
    // Source -> Generate Getters/Setters 메뉴 선택
}
```

JavaBean 사용

- JavaBean = Java 객체
- JavaBean 예: [리스트 8.1] 참고(jsp가 아닌 java 파일)

```
import java.util.Date;

public clas MemberInfo {
    private String id;
    private String password;
    private String name;
    private Date registerDate;
    private String email;

    // getter와 setter는 eclipse에서 자동 생성
    // 편집기에서 마우스 오른쪽 버튼, 단축메뉴에서
    // Source -> Generate Getters/Setters 메뉴 선택
}
```

JavaBean 사용

- Java Bean을 사용하는 jsp
- [리스트 8.2] [리스트 8.3] 참고

```
<%-- create Bean Object: list0802.jsp --%>
<jsp:userBean id="member" scope="request" class="MemberInfo" />
<%
   member.setId("shinhan");
   member.setName("신한대");
%>
<jsp:forward page="list0803.jsp" />
```

```
<%-- use Bean Object: list0803.jsp --%>
<jsp:userBean id="member" scope="request" class="MemberInfo" />
   회원 ID는 <%= member.getId();<br />
   회원 이름은 <%= member.getName();<br />
```

<jsp:setProperty>와 <jsp:getProperty>

- <form>에서 전달 받은 데이터를 Bean에 저장하기
- [리스트 8.4] [리스트 8.5]참고

```
<form action="/ch08/list0805.jsp" method="post">
   아이디 <input type="text" name="id" size="30" /><br />
   이름 <input type="text" name="name" size="50" /><br />
   <br />
   <input type="submit" value="회원가입" />
</form>
```

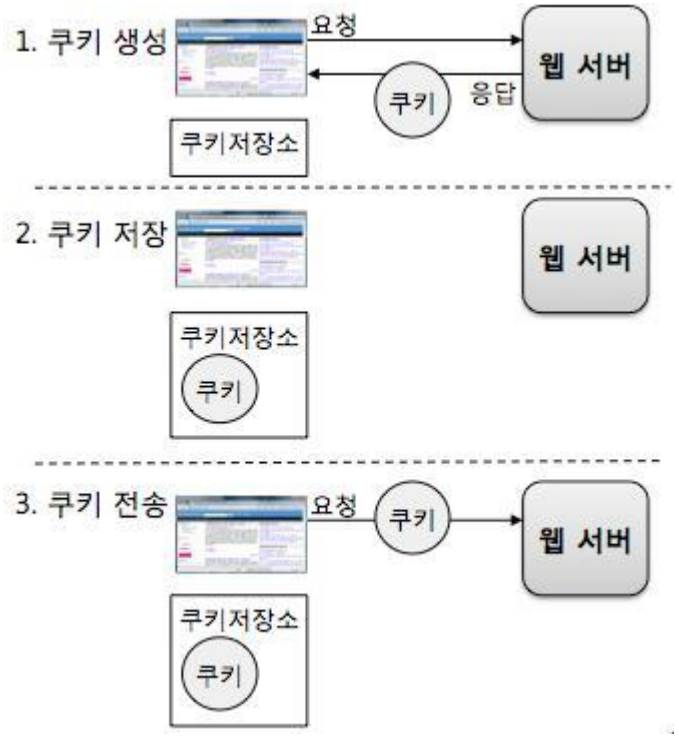
```
<jsp:useBean id="memberInfo" class="ch08.MemberInfo" />
<jsp:setProperty name="memberInfo" property="*" /> <%-- 같은 이름에 대응 --%>
<jsp:setProperty name="memberInfo" property="password"
                                   value="<%= memberInfo.getId() %>" />
…
입력된 이름은 <jsp:setProperty name="memberInfo" property="name" />
…
```

<jsp:setProperty> 값 매핑

프로퍼티 타입	변환 방법	기본값
boolean 또는 Boolean	Boolean.valueOf(String)을 값으로 갖는다.	false
byte 또는 Byte	Byte.valueOf(String)을 값으로 갖는다.	(byte) 0
Short 또는 Short	Short.valueOf(String)을 값으로 갖는다.	(short) 0
Char 또는 Char	입력한 값의 첫 번째 바이트를 값으로 갖는다.	(char) 0
Int 또는 Integer	Integer.valueOf(String)을 값으로 갖는다.	0
Long 또는 Long	Long.valueOf(String)을 값으로 갖는다.	0L
Double 또는 Double	Double.valueOf(String)을 값으로 갖는다.	0.0
float 또는 Float	Float.valueOf(String)을 값으로 갖는다.	0.0f

Cookie 사용하기

- 쿠키(Cookie)는 서버에서 웹 브라우저에 전송한 데이터
- 이후 웹 브라우저는 서버에 요청할 때 쿠키 값을 함께 전달
- JSP에서 사용하는 쿠키는 웹 서버에서 생성



웹 서버에서 쿠키를 생성하여 응답 데이터의 헤더에 저장해서 웹브라우저에 전달

웹 브라우저는 응답 데이터에 포함된 쿠키를 쿠키 저장소에 보관(쿠키의 종류에 따라 메모리 또는 파일 형태로 저장)

웹 브라우저는 요청이 있을 때마다 쿠키를 함께 서버로 전송

Cookie 구성요소

- 이름 각 쿠키를 구분하기 위해 사용하는 쿠키의 이름
  영문자와 숫자만 사용
- 값 각 쿠키가 갖는 값
  콤마, 세미콜론, 공백 문자, '(', ')' 를 제외한 출력 가능한 문자 사용
- 유효시간 쿠키 유지 시간
- 도메인 쿠키를 전송할 도메인
- 경로 쿠키를 전송할 요청 경로

Cookie 생성하기

- 쿠키 생성 기본 형식: [리스트 9.1] 참고

```
<%
    Cookie cookie = new Cookie(“name”, “홍길동”);
    response.addCookie(cookie);
%>
…
쿠키 이름은 <%= cookie.getName() %><br />
쿠키 값은 <%= cookie.getValue() %>
```

- 웹 브라우저의 개발자 도구에서 쿠키 확인
  - F12 -> 네트워크 -> 새로고침(F5) -> 파일 이름 클릭

Cookie 클래스의 메서드

메서드	반환 타입	설명
getName()	String	쿠키 이름을 반환
getValue()	String	쿠키 값을 반환
setValue(String value)	void	쿠키 값을 설정
setDomain(String pattern)	void	쿠키가 전송될 서버의 도메인 지정
getDomain()	String	쿠키의 도메인 반환
setPath(String uri)	void	쿠키를 전송할 경로 설정
getPath()	String	쿠키의 전송 경로 반환
setMaxAge(int expiry)	void	쿠키의 유효 시간을 초 단위로 설정 음수를 설정하면 웹 브라우저를 종료할 때 쿠키를 제거
getMaxAge()	int	큐키의 유효시간 반환

요청과 함께 전송된 Cookie 확인하기

- 쿠키 생성 기본 형식: [리스트 9.2] 참고

```
<%
   Cookie cookies[] = request.getCookies();
   if(cookies != null && cookies.length > 0) {
      for(int i = 0; i < cookies.length; ++i) {
%>
         <%= cookies[i].getName() %> :
         <%= cookies[i].getValue() %> <%= “<br />” %>
<%
      }
   } else {
%>
      쿠키가 존재하지 않습니다.<br />
<%
   }
%>
```

Cookie 값 변경

- 쿠키 값 변경: [리스트 9.3] 참고

```
<%
    Cookie cookies[] = request.getCookies();
    if(cookies != null && cookies.length > 0) {
        for(int i = 0; i < cookies.length; ++i) {
            if(cookies[i].getName().equals("name")) {
                Cookie cookie = new Cookie("name", "새로운값")
                response.addCoolie(cookie);
            }
        }
    }
%>
쿠키 값을 변경했습니다.
```

Cookie 삭제

- 쿠키 삭제: [리스트 9.4] 참고

```
<%
    Cookie cookies[] = request.getCookies();
    if(cookies != null && cookies.length > 0) {
        for(int i = 0; i < cookies.length; ++i) {
            if(cookies[i].getName().equals("name")) {
                Cookie cookie = new Cookie("name", "")
                cookie.setMaxAge(0);
                response.addCoolie(cookie);
            }
        }
    }
%>
쿠키를 삭제했습니다.
```

Cookie 도메인

- 쿠키 도메인: [리스트 9.5] 참고

```
<%
    Cookie cookie1 = new Cookie(“id”, “hgd”);
    cookie1.setDomain(“somehost.com”); <%-- .으로 시작 하면 안됨 --%>
    response.addCookie(cookie1);
    Cookie cookie2 = new Cookie(“id”, “jks”);
    response.addCookie(cookie2);
%>
다음의 쿠키를 생성했습니다.<br />
<%= cookie1.getName() %>:<%= cookie1.getValue() %><br />
<%= cookie2.getName() %>:<%= cookie2.getValue() %>
```

Cookie 경로

- 쿠키 경로: [리스트 9.6] 참고

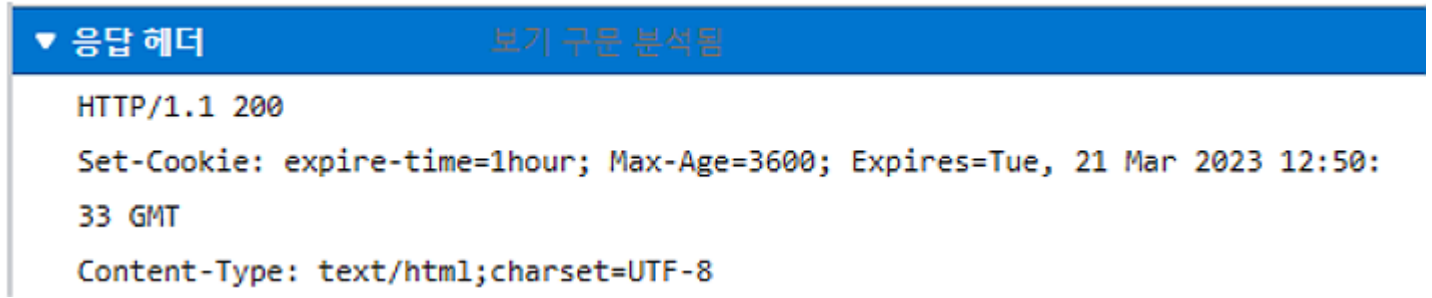
```
<%
    Cookie cookie1 = new Cookie(“path-id”, “hgd”);
    cookie1.setPath (“/ch09/sub”);
    response.addCookie(cookie1);
%>
다음의 쿠키를 생성했습니다.<br />
<%= cookie1.getName() %>:<%= cookie1.getValue() %><br />
```

Cookie 유효시간

- 쿠키 유효시간: [리스트 9.7] 참고

```
<%
    Cookie cookie1 = new Cookie(“expire-time”, “1hour”);
    cookie1.setMaxAge (60*60);   <%-- 1시간 --%>
    response.addCookie(cookie1);
%>
다음의 쿠키를 1시간 수명으로 생성했습니다.<br />
<%= cookie1.getName() %>:<%= cookie1.getValue() %><br />
```

Cooki와 헤더



Cookie를 이용한 로그인 상태 유지

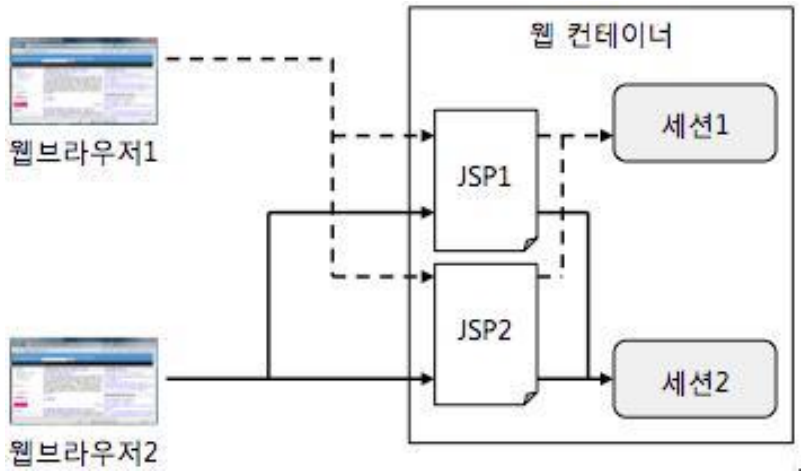
- 로그인 정보를 쿠키로 저장
- 웹 브라우저를 통해 다시 연결할 때 로그인 상태 확인
- 로그인 상태가 아니면, 로그인 페이지로 이동
- 로그인 상태이면, 로그인 후의 페이지로 이동
- 로그아웃할 때 쿠키 삭제
- 수정된 ch09.zip에서
  - list-login.jsp
    - 쿠키에 로그인과 관련된 아이디가 저장되어 있는지 확인
    - 모든 쿠키를 읽어 로그인과 관련된 특정 쿠키 이름이 있는지 확인하여
    - 이름이 있는 경우 값을 읽음
    - 로그인을 위한 form 작성
  - list-loginProcess.jsp
  - list-logout.jsp

Cookie를 이용한 로그인 상태 유지

- 수정된 ch09.zip에서 (계속)
  - list-loginProcess.jsp
    - list-login.jsp에서 전달된 로그인 아이디와 패스워드 확인
    - 예로, 아이디와 패스워드가 같으면 정상 로그인 처리
      - 쿠키 경로 저장
      - 쿠키 수명 지정
    - 아이디와 패스워드가 다르면 오류 메시지 출력
    - <button onclick="location='페이지 이름'">으로 버튼 만들어 페이지 이동
  - List-logout.jsp
    - 로그아웃 처리
    - response.sendRedirect("페이지이름"); 으로 로그인 페이지로 이동

Session 사용: session 기본 객체

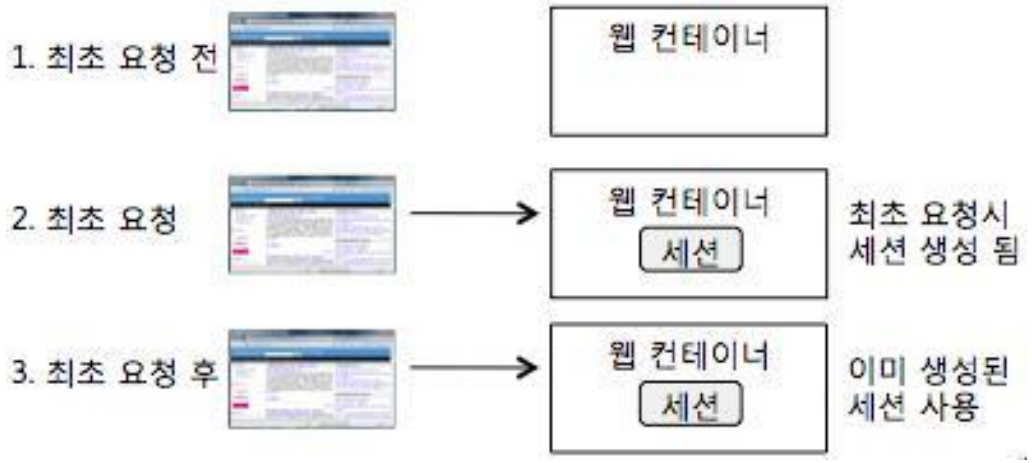
- 쿠키는 웹 브라우저에 정보 저장
- 세션은 웹 컨테이너(톰캣)에서 생성하고 정보 저장
  - 웹 브라우저마다 한 개의 세션 생성
  - 웹 브라우저 관련 정보 저장 가능



Session 사용: session 생성

- 세션 생성

```
<%@ page session = “true” %> <%-- 기본값 --%>
…
<%
   session.setAttribute(“이름”, 값);
   …
   session.getAttribute(“이름”);
   …
%>
```



같은 브라우저이면, 몇 개의 브라우저가 실행되어도
같은 세션이 된다.
다른 브라우저이면, 다른 세션이 된다.

Session 기본 객체

- session 기본 객체가 제공하는 메서드

메서드	반환 타입	설명
getId()	String	세션의 고유 ID를 반환(SessionId)
getCreationTime ()	long	세션이 생성된 시간을 반환 1970년 1월 1일 이후의 시간으로, 1/1000초 단위
getLastAccessTime()	long	마지막 세션 접근 시간을 반환

- 웹 브라우저는 세션 ID를 공유
  - 쿠키 이름: SESSIONID

Session 기본 객체

- [리스트 10.1] 참고

```
세션 ID는 <%= session.getId() %>
세션 생성 시간은 <%= session.getCreationTime() %>
마지막 세션 접근 시간은 <%= session.getLastAccessedTime() %>
```

- [리스트 10.2] 참고

```
<%-- 세션에 데이터 쓰기 --%>
session.setAttribute("이름", 값");

<%-- 세션의 데이터 읽기
session.getAttribute("이름" );
%>
```

Session 유효 시간과 세션 종료

- Session 유효시간: 웹 컨테이너의 메모리 부족 현상 발생 가능
  - web.xml 파일에서 설정(분 단위)

```
<session-config>
    <session-timeout>60</session-timeout>
</session-config>
```

  - session 객체를 이용한 설정(초 단위)

```
<%
    session.setMaxInactiveInterval(60*60);
%>
```

- Session 종료

```
<%
    session.invalidate();
%>
```

Session을 사용한 로그인 상태 유지

- 수정된 ch10 프로젝트 확인
  - session/login.jsp
  - session/process.jsp
  - session/logout.jsp

Session을 사용한 로그인 상태 유지

- 수정된 ch10 프로젝트 확인
  - session/login.jsp
  - session/process.jsp
  - session/logout.jsp

쿠키와 세션의 차이

구분	쿠키 Cookie	세션 Session
공통점	웹 통신간 유지하려는 정보(로그인 정보 등)을 저장하기 위해	
저장위치	클라이언트(=접속자 PC)	서버(세션 키값만 클라이언트)
라이프사이클(만료시점)	쿠키 저장 시 설정 (브라우저가 종료되어도 만료시점이 지나지 않으면 자동삭제되지 않음)	브라우저 종료 시 삭제(기간 지정 가능)
보안	비교적 취약	안전
속도	빠름	비교적 느림
저장 형식	text	Object
사용하는 자원(리소스)	클라이언트 리소스	웹 서버 리소스
용량 제한	총 300개 하나의 도메인당 20개 하나의 쿠키 당 4KB	서버가 허용하는 한 용량 제한 없음

End of Document

Thank you