

Chapter 05

JSP 내장 객체

〈form〉 태그의 구조

<form> 태그의 구조

■ form 태그의 구조와 속성

<form **name**=이름 **method**=전달방식 **action**=데이터전달문서 **enctype**=인코딩방식 >

속성 값	설명
name	form 태그의 이름을 의미 문서 내에 존재하는 form 태그를 식별하는 식별자 역할을 수행
method	form 태그 내의 구성 요소들에 입력된 값을 웹 서버로 전송하는 방식을 나타냄 POST와 GET이라는 값을 가질 수 있음
action	Form의 구성 요소들에 입력된 값들이 서버 내의 어느 문서로 전달되어야 하는지를 지정
enctype	POST 방식으로 전달되는 인코딩 방식을 지정 - application/x-www-form-urlencoded 방식 (기본값) - multipart/form-data 방식

<form> 태그의 구조

■ POST 방식

- 주로 <FORM>태그의 구성요소에 입력된 값을 전달할 때 사용
 - ▶ 입력된 데이터는 BODY를 통해 전달되므로 전달하는 데이터의 크기에 제한이 없음
 - ▶ FILE 입력 양식을 통해 파일을 업로드하는 경우 반드시 POST 방식을 사용
- 전달되는 값들은 브라우저의 주소 표시줄에 나타나지 않음
 - ▶ 최소한의 보안성을 유지할 수 있음
 - ▶ 같은 양의 데이터를 전달한다고 가정할 때 GET 방식에 비해 처리 속도가 다소 늦음
- [예]

```
<form name="myform" method="post" action="LoginProc.jsp">  
    <input type="text" name="my_id" size=10>  
    <input type="submit" value="send">  
</form>
```

- ▶ send 버튼을 클릭하면 입력된 값은 my_id라는 파라미터에 실려 서버에 존재하는 LoginProc.jsp 문서로 전달

<form> 태그의 구조

■ post 방식의 인코딩 방식

- 스트림으로 전달된다 하더라도 내부적으로는 인코딩되어 전달됨
 - 수신측에서는 자동으로 디코딩됨
 - application/x-www-form-urlencoded 방식과 multipart/form-data 방식이 있음
- application/x-www-form-urlencoded 방식
 - 텍스트 데이터를 전송하는데 사용되는 방식 (기본값)
- multipart/form-data 방식
 - 텍스트 데이터는 물론 파일을 전송하는데 사용되는 방식
 - 파일의 업로드 작성 시 반드시 사용해야 함

<form> 태그의 구조

■ GET 방식

- form 태그가 존재하지 않는 문서에서 다른 문서로 데이터를 전달하는데 사용

```
http://서버주소?파라미터1=값1&파라미터2=값2& .....
```

- URL의 뒷부분에 쿼리 스트링(Query String)으로 전달
 - ▶ 쿼리 스트링은 [이름=값] 과 같이 파라미터의 이름과 값의 쌍으로 구성
 - ▶ 데이터가 전달되는 서버의 URL과 쿼리 스트링은 '?' 문자를 사용
 - ▶ '이름=값'의 쌍은 '&' 문자로 구분
- GET 방식으로 전달할 수 있는 데이터의 크기가 제한되어 있음
 - ▶ 제한된 크기 이내라면 전달할 수 있는 파라미터의 수에는 제한이 없음
- 데이터를 전달할 경우 전달되는 파라미터가 브라우저의 주소표시줄에 공개됨

<form> 태그의 구조

- <form> 태그의 action 속성에 쿼리스트링을 지정해 전송하기도 함

- ▶ POST와 GET 방식으로 동시에 전달 가능

```
<form name="myform" method="post" action="LoginProc.jsp?name=kim&age=30&address=seoul">  
  <input type="text" name="my_id" size=10>  
  <input type="submit" value="send">  
</form>
```

- ▶ Submit이 발생되면 <input> 입력상자에 입력된 값을 POST 방식으로, 쿼리스트링의 값은 GET 방식으로 전달됨

- 실제 개발에서는 POST 방식보다 GET 방식 처리가 많음

JSP 내장 객체

내장 객체의 개요

■ JSP 내장 객체

- JSP 내에서 선언하지 않고 사용할 수 있는 객체

내장 객체	자바 클래스	설명
request	javax.servlet.http.HttpServletRequest	사용자가 요청한 정보를 추출
response	javax.servlet.http.HttpServletResponse	사용자에 대한 응답을 처리
pageContext	javax.servlet.jsp.PageContext	현재 JSP 실행에 대한 context 참조
session	javax.servlet.http.HttpSession	세션 정보의 처리
application	javax.servlet.ServletContext	어플리케이션 관련 정보 처리
out	javax.servlet.jsp.JspWriter	사용자에게 전달하기 위한 output 스트림 처리
config	javax.servlet.ServletConfig	JSP의 초기화 환경 설정
page	java.lang.Object	현재 JSP의 정보를 추출
exception	java.lang.Throwable	예외 사항 처리

request 내장 객체

request 내장 객체

■ request 내장 객체

- 사용자 요청과 관련된 기능을 제공하는 내장 객체
 - request 내장 객체는 JSP 문서에서 가장 많이 사용되는 객체 중 하나
- 주로 클라이언트에서 서버로 전달되는 정보를 처리하기 위해 사용
 - 대표적으로 HTML 폼을 통해 입력된 값을 JSP에서 가져올 때 사용함
 - 사용 예
 - 사용자가 브라우저의 주소표시줄에 URL을 입력해 요청하면 request 내장 객체를 통해 웹 서버로 전달
 - <FORM> 태그의 구성 요소에 입력된 값은 request 내장 객체를 통해 웹 서버로 전달
- request 내장 객체가 제공하는 메서드의 종류
 - form 태그 구성 요소의 파라미터 처리 관련 메서드
 - 시스템의 정보 추출 관련 메서드
 - 헤더 정보 추출에 관련된 메서드

request 내장 객체의 메서드

■ 파라미터 처리 관련 메서드

- form 태그의 구성 요소로부터 전송된 파라미터를 처리하기 위해 사용

메서드	변환 타입	설명
setCharacterEncoding(charset)	String	전달되는 내용의 문자 집합을 지정한 charset으로 인코딩
getParameter(String name)	String	name이라는 이름을 가진 파라미터의 값을 반환 값이 존재하지 않을 경우 null을 반환
getParameterValues(String name)	String[]	name이라는 이름을 가진 파라미터의 값을 배열 형태로 반환 값이 존재하지 않을 경우 null을 반환
getParameterNames()	Enumeration	전송된 모든 파라미터의 이름을 Enumeration 형태로 반환

request 내장 객체의 메서드

■ getParameter() 메서드

```
Datatype 변수 = request.getParameter("파라미터 이름")
```

- 클라이언트로부터 전달된 파라미터의 값을 추출하는 메소드
 - 인자로 가지는 파라미터의 이름인 name에 해당하는 값을 반환
 - 값이 존재하지 않으면 null을 반환
- POST 방식과 GET 방식으로 전달된 파라미터 값들을 모두 추출
- 파라미터 값의 데이터 타입이 기본적으로 String으로 인식
 - String 타입이 아닌 경우, 래퍼 클래스(wrapper class)와 데이터 타입 변환 메서드를 사용해 데이터 타입으로 변환 필요

request 내장 객체의 메서드

- `getParameter(String name)` 메소드의 사용 예

```
<% String my_id = request.getParameter("id"); %>
```

- ▶ id라는 이름(name=id)을 가진 text 입력 양식에 입력된 값을 추출하여 my_id라는 String 변수에 저장

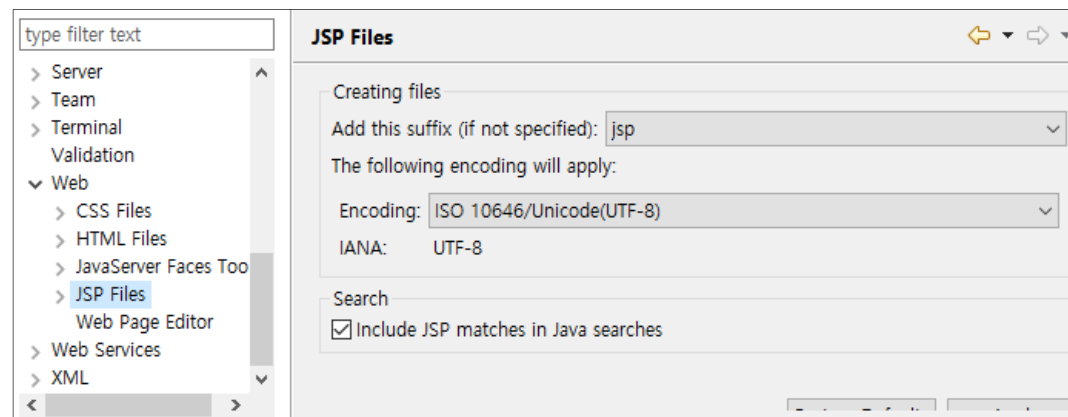
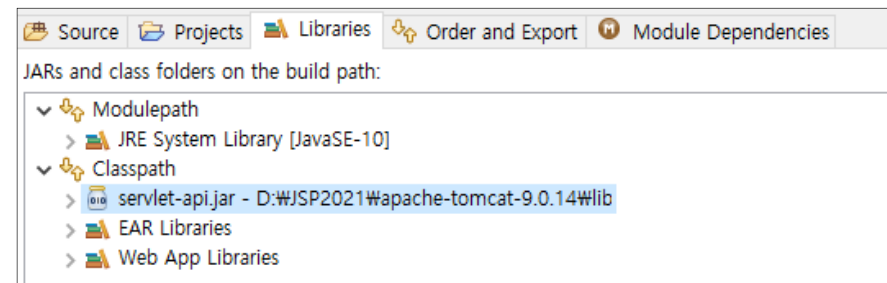
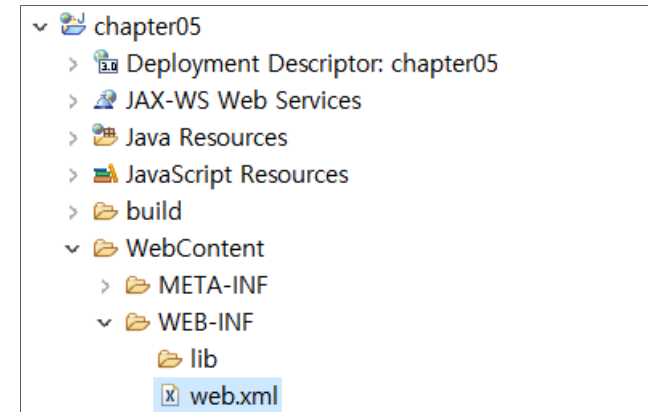
```
<% int my_age = Integer.parseInt( request.getParameter("age") ); %>
```

- ▶ String 타입이 아닌 다른 데이터 타입이라면 wrapper class와 데이터 타입 변환 메서드를 사용
- ▶ form 태그에서 age라는 이름(name=age)을 가진 text 입력 양식의 값을 추출하여 my_age라는 정수형 변수에 저장

request 내장 객체의 메서드

■ 프로젝트 생성

- 프로젝트 이름 : chapter05
 - ▶ web.xml 파일은 반드시 생성해야 함
- servlet-api 라이브러리 추가
- JSP 문서 문자 집합 지정
 - ▶ UTF-8로 지정



```
getParameter1.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="UTF-8">
7 </head>
8 <body>
9
10   <% String depart="software"; %>
11
12   <form name="myform" method=post action="getParameterProc1.jsp?depart=<%=depart%>">
13
14     번호(정수) : <input type=text name="num" size=10><br>
15     이름(문자열) : <input type=text name="name" size=20><br>
16     계정(문자열) : <input type=text name="uid" size=20><br>
17     평점(실수) : <input type=text name="score" size=5><br>
18     <input type=submit value="저장">
19
20   </form>
21
22 </body>
23 </html>
```


request 내장 객체의 메서드

getParameterProc1.jsp

```
getParameterProc1.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%
4
5   String depart = request.getParameter("depart");
6   int num = Integer.parseInt(request.getParameter("num"));
7   String name = request.getParameter("name");
8   String uid = request.getParameter("uid");
9   float score = Float.parseFloat(request.getParameter("score"));
10
11   out.println("학과는 " + depart + "입니다.<BR>");
12   out.println("번호는 " + num + "입니다.<BR>");
13   out.println("이름은 " + name + "입니다.<BR>");
14   out.println("계정은 " + uid + "입니다.<BR>");
15   out.println("평점은 " + score + "입니다.<BR>");
16
17 %>
```

getParameterP... getParameter1... http://localh...
p://localhost:8080/chapter05/getParameter1.jsp

번호(정수):

이름(문자열):

계정(문자열):

평점(실수):

getParameterP... getParameter1... http://localh...
http://localhost:8080/chapter05/getParameterPr

학과는 software입니다.
번호는 30입니다.
이름은 iê,,ë입니다.
계정은 hong입니다.
평점은 4.3입니다.

request 내장 객체의 메서드

■ setCharacterEncoding() 메서드

```
request.setCharacterEncoding("문자 집합");
```

- 클라이언트로부터 POST 방식으로 전달된 데이터를 인자로 지정한 방식으로 인코딩
- form 요소의 입력 양식 중에 한글이 포함되어 있을 경우
 - 한글을 인코딩할 수 있는 문자 집합을 정의해 주어야 한글이 정상적으로 출력됨
- 클라이언트로부터 전달된 파라미터의 값을 추출하기 전에 수행되어야 함
 - 일반적으로 JSP 문서의 상단에 작성

```
getParameterProc1.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3
4 <% request.setCharacterEncoding("utf-8"); %>
5 <%
6
7   String depart = request.getParameter("depart");
8   int num = Integer.parseInt(request.getParameter("num"));
9   String name = request.getParameter("name");
10  String uid = request.getParameter("uid");
11  float score = Float.parseFloat(request.getParameter("score"));
12
13  out.println("학과는 " + depart + "입니다.<BR>");
14  out.println("번호는 " + num + "입니다.<BR>");
15  out.println("이름은 " + name + "입니다.<BR>");
16  out.println("계정은 " + uid + "입니다.<BR>");
17  out.println("평점은 " + score + "입니다.<BR>");
18
19 %>
20
```

http://localhost:8080/chapter05/getParameterProc1.jsp?depart=so...
http://localhost:8080/chapter05/getParameterProc1.j

학과는 software입니다.
번호는 30입니다.
이름은 홍길 동입니다.
계정은 hong입니다.
평점은 4.3입니다.

request 내장 객체의 메서드

- action 속성에 쿼리 스트링 이용한 GET 방식의 한글 데이터의 전달
 - ▶ `setCharacterEncoding` 메서드는 GET 방식으로 전달된 데이터까지는 인코딩하지 못함
- `URLEncoder`의 `encode()` 메서드를 사용해 인코딩해 전달해야 함

wrong !

```
<% String depart="소프트웨어"; %>  
<form name="myform" method=post action="getParameter-proc-1.jsp?depart=<%=depart%>">
```

correct !

```
<%@ page import="java.net.URLEncoder" %>  
<% String depart = URLEncoder.encode("소프트웨어", "UTF-8");%>  
  
<form name="myform" method=post action="getParameter-proc-1.jsp?depart=<%=depart%>">
```

```
getParameter2.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.net.URLEncoder" %>
4
5 <!DOCTYPE html>
6 <html>
7 <head>
8   <meta charset="UTF-8">
9 </head>
10 <body>
11
12   <% String depart = URLEncoder.encode("소프트웨어", "UTF-8");%>
13
14   <form name="myform" method=post action="getParameterProc2.jsp?depart=<%=depart%>">
15
16     번호(정수) : <input type=text name="num" size=10><br>
17     이름(문자열) : <input type=text name="name" size=20><br>
18     계정(문자열) : <input type=text name="uid" size=20><br>
19     평점(실수) : <input type=text name="score" size=5><br>
20     <input type=submit value="저장">
21
22   </form>
23
24 </body>
25 </html>
```

```
getParameter2.jsp  getParameterProc2.jsp ✕
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3
4 <% request.setCharacterEncoding("utf-8"); %>
5 <%
6
7     String depart = request.getParameter("depart");
8     int num = Integer.parseInt(request.getParameter("num"));
9     String name = request.getParameter("name");
10    String uid = request.getParameter("uid");
11    float score = Float.parseFloat(request.getParameter("score"));
12
13    out.println("학과는 " + depart + "입니다.<BR>");
14    out.println("번호는 " + num + "입니다.<BR>");
15    out.println("이름은 " + name + "입니다.<BR>");
16    out.println("계정은 " + uid + "입니다.<BR>");
17    out.println("평점은 " + score + "입니다.<BR>");
18
```

getParameter2.jsp http://localhost:8080/chapter05/ge
http://localhost:8080/chapter05/getParamete
학과는 소프트웨어입니다.
번호는 20입니다.
이름은 홍길동입니다.
계정은 hong입니다.
평점은 4.3입니다.

request 내장 객체의 메서드

■ getParameterValues() 메서드

```
Datatype 변수 = request.getParameterValues("파라미터 이름")
```

- 하나의 파라미터가 가지는 여러 개의 값을 가지는 경우, 각각의 값을 추출해 배열로 반환
 - ▶ 값이 존재하지 않을 경우 null을 반환
- form 태그의 구성요소 중 checkbox는 하나 이상의 값을 가질 수 있음
 - ▶ getParameter()로는 처리 불가능
 - ▶ checkbox 파라미터가 가지는 값들을 배열에 저장
 - ▶ [예]

```
String[] test = request.getParameterValues("lang");
```

- lang이라는 이름의 파라미터가 가지는 값들을 전달받아 test라는 String 배열을 생성

```
getParameterValues.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="UTF-8">
7 </head>
8 <body>
9
10   가능한 외국어를 모두 선택하십시오. <BR>
11
12   <form name="test" method=POST action="getParameterValuesProc.jsp">
13     <input type="checkbox" name="lang" value="영어">영어 <br>
14     <input type="checkbox" name="lang" value="일본어">일본어 <br>
15     <input type="checkbox" name="lang" value="중국어">중국어 <br>
16     <input type="checkbox" name="lang" value="스페인어">스페인어 <br>
17     <input type="SUBMIT" value="확인">
18
19   </form>
20
21 </body>
22 </html>
23
```



```
getParameterValuesProc.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <% request.setCharacterEncoding("utf-8");%>
4 <%
5
6   String[] languages = request.getParameterValues("lang");
7   String str="";
8
9   for( String data : languages) {
10      str += data;
11      str += " ";
12   }
13
14   out.println("선택한 외국어는 " + str + "입니다.");
15
```

```
for (int i=0; i<languages.length; i++) {
    str += languages[i];
}
```

/localhost:8080/chapter05/getParameterValues.jsp

가능한 외국어를 모두 선택하시오.

☐ 영어

☒ 일본어

☐ 중국어

☒ 스페인어

확인

http://localhost:8080/chapter05/getParameterValu

선택한 외국어는 일본어 스페인어 입니다.

request 내장 객체의 메서드

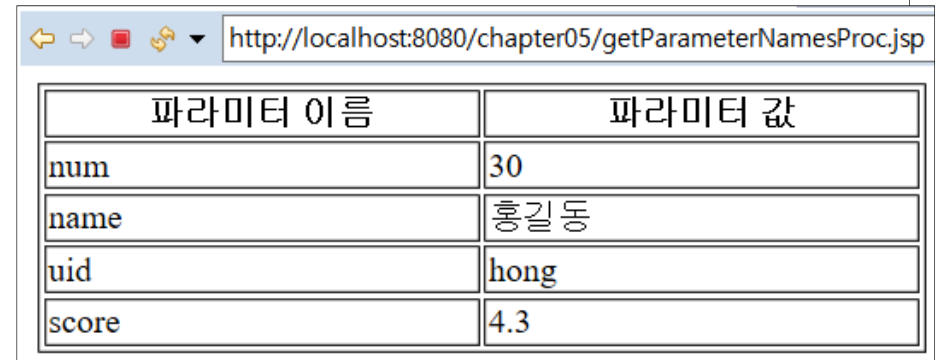
■ getParameterNames() 메서드

```
Enumeration 변수 = request.getParameterNames("추출할 파라미터 이름")
```

- 요청에 존재하는 파라미터의 값을 추출하지 않고 파라미터의 이름들을 추출
 - ▶ 추출된 파라미터의 이름은 Enumeration 타입으로 반환됨
- 파라미터의 값까지 함께 추출하기 위해서는 getParameter() 메서드를 함께 사용해야 함

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3
4 <!DOCTYPE html>
5 <html>
6 <head>
7   <meta charset="UTF-8">
8 </head>
9 <body>
10
11   <form name="myform" method=post action="getParameterNamesProc.jsp">
12
13     번호(정수) : <input type=text name="num" size=10><br>
14     이름(문자열) : <input type=text name="name" size=20><br>
15     계정(문자열) : <input type=text name="uid" size=20><br>
16     평점(실수) : <input type=text name="score" size=5><br>
17     <input type=submit value="저장">
18
19   </form>
20
21 </body>
22 </html>
```

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.io.*, java.util.*"%>
4 <% request.setCharacterEncoding("UTF-8"); %>
5
6 <!DOCTYPE html>
7 <html>
8 <head>
9   <meta charset="UTF-8">
10 </head>
11 <body>
12   <table border="1">
13     <tr>
14       <th width=200>파라미터 이름</th>
15       <th width=200>파라미터 값</th>
16     </tr>
17     <%
18       Enumeration paramNames = request.getParameterNames();
19       while (paramNames.hasMoreElements()) {
20         String name = (String) paramNames.nextElement();
21         out.print("<tr><td>" + name + " </td>\n");
22         String paramValue = request.getParameter(name);
23         out.println("<td> " + paramValue + "</td></tr>\n");
24       }
25     %>
26   </table>
27 </body>
28 </html>
```



파라미터 이름	파라미터 값
num	30
name	홍길동
uid	hong
score	4.3

request 내장 객체의 메서드

■ 프로토콜 및 URL 정보를 추출하는 메서드

메서드	반환 타입	설명
getRemoteAddr()	String	서비스를 요청한 클라이언트의 IP 주소를 반환.
getMethod()	String	현재 클라이언트로부터의 요청이 POST인지 GET 인지를 반환
getProtocol()	String	현재의 프로토콜 및 버전을 반환
getServerName()	String	클라이언트로부터 요청을 받은 서버의 이름을 반환
getServerPort()	int	클라이언트로부터 요청을 받은 서버의 포트 번호를 반환
getRequestURI()	String	요청된 클라이언트의 URL 중 호스트 이름 이후의 문자열을 반환
getRequestURL()	String	요청된 클라이언트의 URL의 모든 정보 반환
getQueryString()	String	요청된 클라이언트의 URL 중 쿼리 스트링만 반환
getContextPath()	String	웹 어플리케이션의 컨텍스트(웹 어플리케이션 이름)를 반환

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <meta charset="UTF-8">
7 </head>
8 <body>
9
10     <form name="myForm" method=post action="requestInfoProc.jsp?str=korea">
11         <input type=submit value="저장">
12     </form>
13
14 </body>
15 </html>
```

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3
4 <% out.println("<h2> 프로토콜과 URL 관련 정보</h2><hr>"); %>
5
6 1. 클라이언트 IP 주소: <%= request.getRemoteAddr() %><br>
7 2. 요청 메서드: <%= request.getMethod() %> <br>
8 3. 프로토콜: <%= request.getProtocol() %> <br>
9 4. 서버 호스트 이름: <%= request.getServerName() %> <br>
10 5. 서버 포트: <%= request.getServerPort() %> <br>
11 6. 요청 URI: <%= request.getRequestURI() %> <br>
12 7. 요청 URL: <%= request.getRequestURL() %> <br>
13 8. 요청 URL중 쿼리 스트링: <%= request.getQueryString() %> <br>
14 9. 컨텍스트 패스 정보: <%= request.getContextPath() %>
```

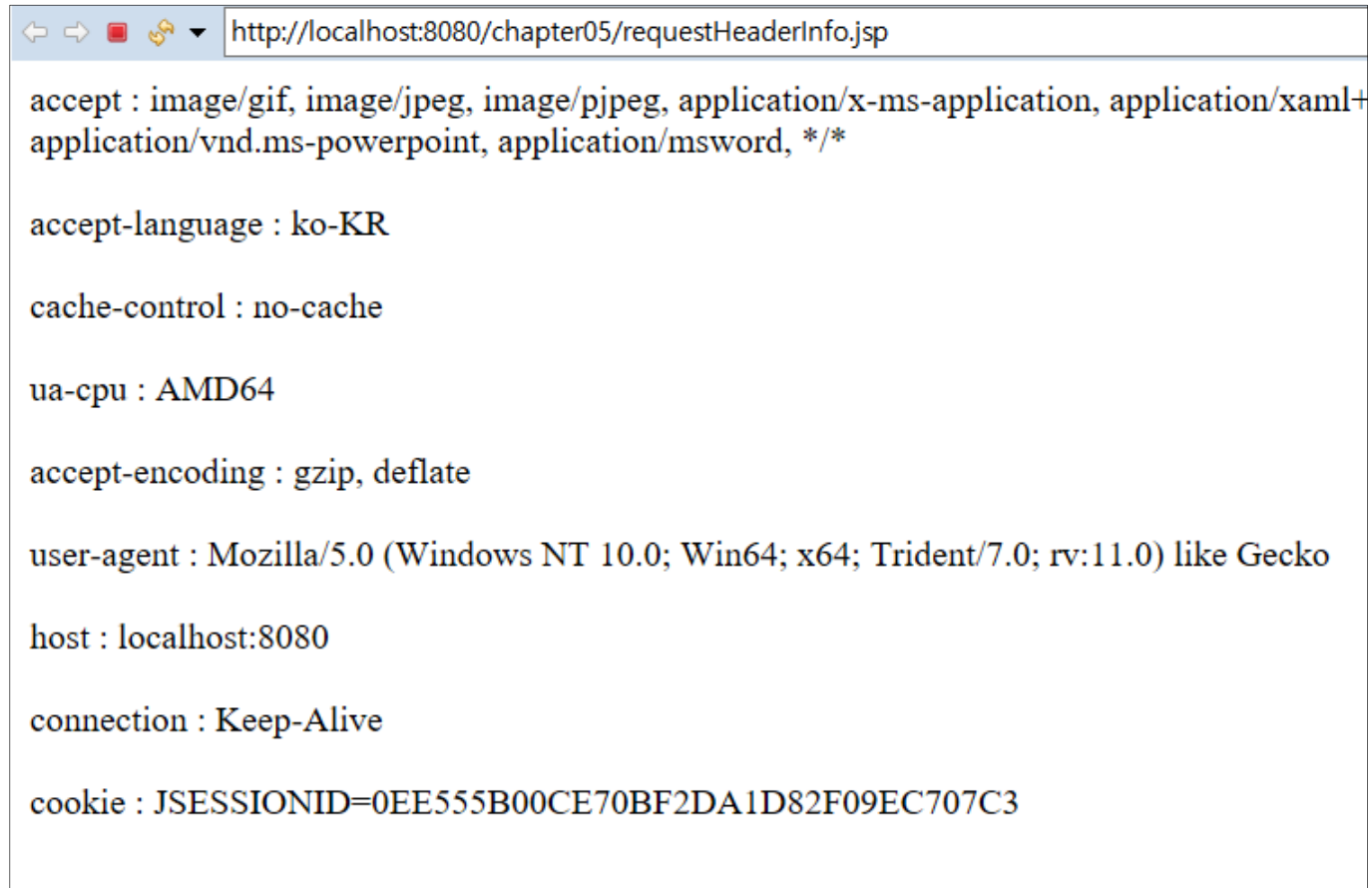


request 내장 객체의 메서드

■ 헤더 정보를 추출하는 메서드

메서드	반환 타입	설명
getHeader(String name)	String	인자로 가지는 이름의 헤더 정보를 문자열로 반환
getHeaderNames()	String	모든 헤더의 이름을 Enumeration 타입으로 반환


```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@page import="java.util.Enumeration"%>
4
5 <!DOCTYPE html>
6 <html>
7 <head>
8   <meta charset="UTF-8">
9 </head>
10 <body>
11   <%
12     Enumeration<String> en = request.getHeaderNames();
13
14     while (en.hasMoreElements()) {
15       String headerName = (String) en.nextElement();
16       String headerValue = request.getHeader(headerName);
17
18       out.println(headerName + " : " + headerValue);
19       out.println("<br><br>");
20
21     }
22   %>
23 </body>
24 </html>
```



The screenshot shows a web browser window with the address bar displaying `http://localhost:8080/chapter05/requestHeaderInfo.jsp`. The main content area lists the following request headers:

```
accept : image/gif, image/jpeg, image/pjpeg, application/x-ms-application, application/xhtml+xml, application/vnd.ms-powerpoint, application/msword, */*  
  
accept-language : ko-KR  
  
cache-control : no-cache  
  
ua-cpu : AMD64  
  
accept-encoding : gzip, deflate  
  
user-agent : Mozilla/5.0 (Windows NT 10.0; Win64; x64; Trident/7.0; rv:11.0) like Gecko  
  
host : localhost:8080  
  
connection : Keep-Alive  
  
cookie : JSESSIONID=0EE555B00CE70BF2DA1D82F09EC707C3
```

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="UTF-8">
7 </head>
8 <body>
9   <h2>학생정보 입력</h2>
10  <hr width=400 align=left>
11  <form method="get" action="requestStudentResult.jsp">
12    <table width=400 border=1 cellspacing="1" cellpadding="5">
13
14      <tr>
15        <td width=100>이름</td>
16        <td width=300><input type="text" size="10" name="username"></td>
17      </tr>
18
19      <tr>
20        <td>학년</td>
21        <td>
22          <input type="radio" name="grade" value="1학년">1학년</input>
23          <input type="radio" name="grade" value="2학년">2학년</input>
24          <input type="radio" name="grade" value="3학년">3학년</input>
25          <input type="radio" name="grade" value="4학년">4학년</input>
26        </td>
27      </tr>
```

```
28
29<tr>
30    <td>관심분야</td>
31<td>
32    <input type="checkbox" name="favorite" value="JSP">JSP</input>
33    <input type="checkbox" name="favorite" value="JAVA">JAVA</input>
34    <input type="checkbox" name="favorite" value="C++">C++</input>
35</td>
36</tr>
37
38<tr>
39<td colspan="2" align="center">
40    <input type="submit" value="확인">
41    <input type="reset" value="취소">
42</td>
43</tr>
44
45</table>
46</form>
47
48</body>
49</html>
```

[실습과 과제]

■ 과제

- 다음과 같이 입력된 값을 출력하는 문서를 완성하여 제출하시오.
 - ▶ 입력 문서는 이전의 예제 (requestStudentInfo.jsp)를 사용
 - ▶ 출력 문서는 작성해 제출 (파일명 : requestStudentResult.jsp)
 - 단, 입력 문서의 '이름'과 '학년' 입력 양식에는 본인의 이름과 학년을 입력해 출력해야 함
 - ▶ 제출 방법 일시 : e-calpus / 이번주 일요일까지
 - ▶ 제출 내용
 - JSP 문서 (requestStudentInfo.jsp, requestStudentResult.jsp)와 실행 화면(이미지나 워드 문서)를 하나의 파일로 압축
 - 압축파일의 이름은 '본인이름.zip'

http://localhost:8080/chapter05/requestStudentForm.jsp

학생정보 입력

이름	<input type="text" value="이광"/>
학년	<input type="radio"/> 1학년 <input checked="" type="radio"/> 2학년 <input type="radio"/> 3학년 <input type="radio"/> 4학년
관심분야	<input checked="" type="checkbox"/> JSP <input type="checkbox"/> JAVA <input checked="" type="checkbox"/> C++
<input type="button" value="확인"/> <input type="button" value="취소"/>	

http://localhost:8080/chapter05/requestStudentResult.jsp

학생정보 출력

이름	이광
학년	2학년
관심분야	JSP C++

1. 클라이언트 IP 주소: 0:0:0:0:0:0:1
2. 프로토콜: HTTP/1.1
3. 서버 호스트 이름: localhost
4. 서버 포트: 8080

response 내장 객체

■ response 내장 객체

- 클라이언트로의 응답 정보를 저장하는 기본 객체
- response 객체의 주요 역할
 - ▶ 헤더 정보를 설정
 - ▶ request를 처리하고 응답을 다른 페이지로 전달하는 리디렉션(redirection) 기능을 수행

메서드	반환 타입	설명
setContentType(String type)	viod	클라이언트로 전달할 데이터의 MIME 타입을 지정
setHeader(String name, value)	viod	name에 해당하는 헤더의 값을 value로 설정
addHeader(String name, value)	viod	name에 해당하는 헤더의 값에 value를 추가
sendRedirect(url)	viod	브라우저에 url에 지정된 JSP문서나 사이트로 리디렉트를 수행

response 내장 객체의 메서드

■ setContentType() 메서드

- 클라이언트로 전송되는 정보의 MIME 타입을 지정

- ▶ page 지시문에서 contentType 속성을 사용해 MIME 타입을 지정하는 것과 같은 효과를 가짐

- [예]

- ▶ 전달할 데이터가 JPG 이미지일 경우

```
response.setContentType("image/jpeg");
```

- ▶ 전송하고자 하는 정보가 일반 파일일 경우

```
response.setContentType("application/octet-stream");
```

response 내장 객체의 메서드

■ setHeader()와 addHeader() 메서드

- 웹 서버는 클라이언트로 데이터를 전송하기 전 JSP 문서를 처리하기 위한 정보를 포함한 헤더 정보를 전송
 - setHeader()와 addHeader() 메서드를 사용하여 기존 헤더를 변경하거나 새로운 헤더를 추가 가능
- setHeader() 메소드
 - 기존 헤더가 가지는 값을 다른 값으로 변경하는 기능을 수행
- addHeader() 메소드
 - 새로운 헤더를 추가하는 기능을 수행
 - 같은 이름을 가진 헤더가 이미 존재한다면 지정한 값이 하나 더 추가됨

response 내장 객체의 메서드

- **setHeader() 메소드 사용의 예**

- ▶ setHeader() 메소드를 사용해 캐시(cache)를 사용하지 않도록 지정하는 예

```
<%  
    if(request.getProtocol().equals("HTTP/1.1")) {  
        response.setHeader("Cache_Control","no-cache");  
    } else {  
        response.setHeader("Pragma","no-cache");  
    }  
%>
```

- ▶ Cache_Control 헤더의 값을 'no-cache'로 지정하거나 Pragma 헤더의 값을 'no-cache'로 지정해 캐시를 사용하지 않음
- ▶ HTTP/1.0 프로토콜에서는 Pragma 헤더를 사용해 캐시의 사용 여부를 지정
- ▶ HTTP/1.1 프로토콜에서는 Cache_Control 헤더를 사용
- ▶ 주로 파일 다운로드 구 현시 사용 (차후 학습)

response 내장 객체의 메서드

■ sendRedirect() 메서드

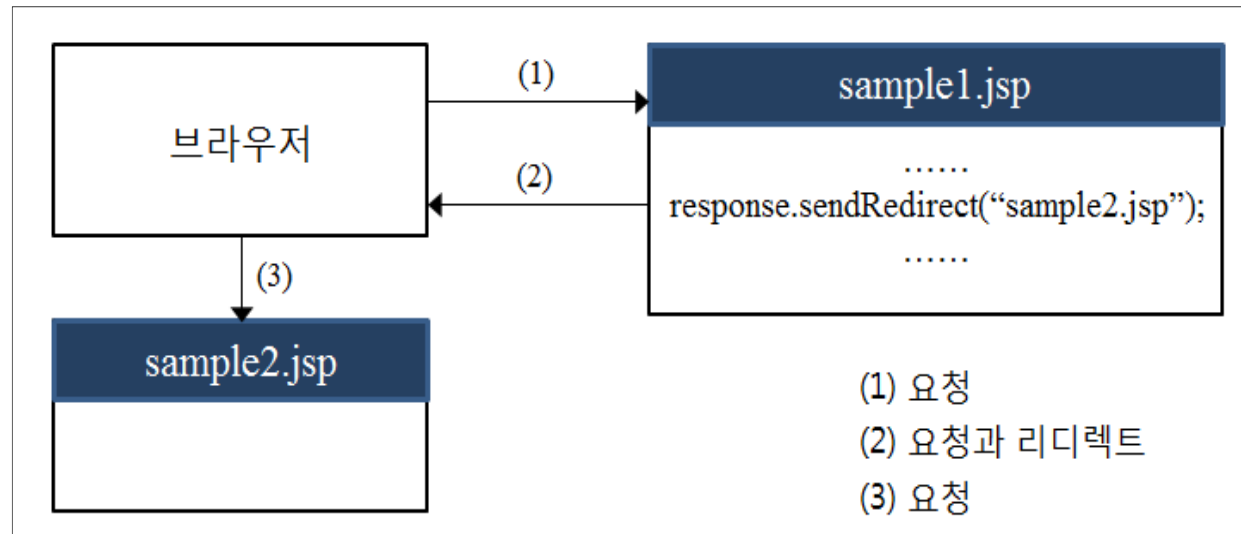
`response.sendRedirect(이동할문서나 사이트)`

- 특정 문서를 출력하고 있는 브라우저에게 다른 문서를 출력하라고 지시하는 리디렉트 기능 수행
 - ▶ 제어의 흐름 전환(제어의 이동)을 의미
 - ▶ Response 기본 객체의 메서드 중 비교적 자주 사용되는 메서드
 - ▶ 리디렉트의 예
 - 로그인 문서에서 정상적인 로그인이 이루어지면 메인 페이지로 이동
 - 게시판에 글을 입력한 다음 [저장] 버튼을 클릭하면 게시 글의 목록을 출력하는 페이지로 이동
- 제어의 이동을 수행하는 또다른 방법
 - ▶ <forward> 액션 태그
 - ▶ 자바스크립트의 location객체의 replace() 메서드나 href 속성
 - ▶ 차후 학습

response 내장 객체의 메서드

■ sendRedirect() 메서드의 동작 방식

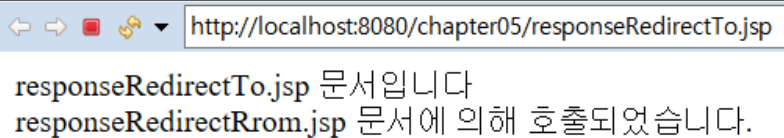
- sample1.jsp 문서 내에 sample2.jsp 문서로 리디렉트를 지시하는 경우의 예



- ▶ (1) 브라우저는 sample1.jsp 문서를 웹 서버에 요청
- ▶ (2) 웹 서버는 요청된 sample1.jsp 문서를 서비스
 - sample1.jsp는 sendRedirect() 메서드를 사용해 sample2.jsp를 요청할 것을 브라우저에 지시
- ▶ (3) 브라우저는 다시 sample2.jsp 문서를 웹 서버에 요청
 - 브라우저는 실제로 웹 서버에 두 번의 요청을 수행

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <meta charset="UTF-8">
7 </head>
8 <body>
9
10 <%
11
12     out.println("responseRedirectFrom.jsp의 내용입니다.");
13     response.sendRedirect("responseRedirectTo.jsp");
14
15 %>
16 </body>
17 </html>
```

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <meta charset="UTF-8">
7 </head>
8 <body>
9
10 <%
11
12     out.println("responseRedirectTo.jsp 문서입니다<BR>");
13     out.println("responseRedirectRrom.jsp 문서에 의해 호출되었습니다.");
14
15 %>
16
17 </body>
18 </html>
```



http://localhost:8080/chapter05/responseRedirectTo.jsp

responseRedirectTo.jsp 문서입니다
responseRedirectRrom.jsp 문서에 의해 호출되었습니다.

response-sendRedirect1.jsp의 내용이 출력되지 않았음

- sendRedirect() 메서드가 리디렉트를 수행하기 전 버퍼를 비우기 때문
- 자세한 내용은 out 내장 객체에서 학습함

out 내장 객체

out 내장 객체

■ out 내장 객체

- JSP 문서가 생성하는 출력 내용들을 브라우저에 전송하는 내장 객체
 - out을 이용해서 출력한 내용은 서버의 콘솔이 아닌 사용자의 브라우저에 출력됨
 - 데이터 출력을 위한 메서드와 버퍼 관련 메서드를 제공

메서드	반환 타입	설명
print(content)	void	인자로 가지는 content를 출력
println(content)	void	인자로 가지는 content 마지막에 <code>\r\n</code> 를 함께 출력
getBufferSize()	int	현재 버퍼의 전체 크기를 바이트 단위로 반환.
getRemaining()	int	현재 버퍼의 남아있는 크기를 바이트 단위로 반환한다.
flush()	void	현재 버퍼에 있는 내용을 브라우저로 전송하고 버퍼를 비움
isAutoFlush()	boolean	버퍼가 가득 찼을 때 자동으로 flush되면 true를 반환
clear()	void	현재의 버퍼를 비움 - flush() 가 수행된 후 호출되면 버퍼를 비운 후 IOException이 발생
clearBuffer()	void	현재의 버퍼를 비움 - flush() 가 수행된 후 호출되면 버퍼를 비운 후에도 IOException이 발생되지 않음
close()	void	현재 버퍼의 모든 데이터를 브라우저로 전송하고 출력 스트림을 닫음

out 내장 객체의 메서드

■ getBufferSize()메서드

```
out.getBufferSize()
```

- 현재 JSP 문서에 설정된 버퍼의 크기를 바이트 단위로 반환
 - 기본 값은 8K임

■ getRemaining() 메서드

```
out.getRemaining()
```

- 현재의 버퍼 크기 중 남아있는 공간의 크기를 바이트 단위로 반환

out 내장 객체의 메서드

■ flush() 메소드

```
out.flush()
```

- 현재 버퍼에 있는 모든 내용을 브라우저로 출력하기 위해 전송하는 기능을 수행

■ isAutoFlush() 메소드

```
out.isautoFlush()
```

- 버퍼가 가득 찼을 경우 자동으로 flush되도록 설정되어 있다면 true를 반환
 - 주로 제어문이나 순환문에서 조건으로 사용
- page 지시문에서 autoFlush 속성이 true로 지정되어 있을 경우 true를 반환
 - false로 지정되어 있을 경우 false를 반환

out 내장 객체의 메서드

■ clear() 메서드

```
out.clear()
```

- 현재 버퍼의 내용을 브라우저에 출력하지 않고 버퍼를 비움
 - flush()가 수행된 후 clear()가 수행될 경우 현재 버퍼를 비운 후 IOException이 발생

■ clearBuffer() 메소드

```
out.clearBuffer()
```

- 현재 버퍼의 내용을 브라우저에 출력하지 않고 버퍼를 비움
 - flush()가 수행된 후 clearBuffer()가 수행될 경우 버퍼를 비운 후에도 IOException이 발생되지 않음

out 내장 객체의 메서드

■ close() 메서드

```
out.close()
```

- 버퍼에 있는 모든 내용을 브라우저에 출력하고 출력 스트림을 닫는 기능을 수행

out 내장 객체의 메서드

out1.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6   <meta charset="UTF-8">
7 </head>
8 <body>
9
10 <h2> out 내장객체의 사용 </h2>
11 <hr>
12 1. 설정된 버퍼크기 : <%= out.getBufferSize() %> <br>
13 2. 남아있는 버퍼크기 : <%= out.getRemaining() %> <br>
14 <% out.flush(); %>
15
16 3. flush 후 남아있는 버퍼크기 : <%= out.getRemaining() %> <br>
17 <% out.clearBuffer(); %>
18
19 4. clear 후 남아있는 버퍼크기 : <%= out.getRemaining() %> <br>
20 <% out.close(); %>
21
22 5. close 후 남아있는 버퍼크기 : <%= out.getRemaining() %> <br>
23
24 </body>
25 </html>
```

http://localhost:8080/chapter05/out1.jsp

out 내장객체의 사용

1. 설정된 버퍼크기 : 8192
2. 남아있는 버퍼크기 : 8038
4. clear 후 남아있는 버퍼크기 : 8163

session 내장 객체

session 내장 객체

■ HTTP 프로토콜의 특성

- HTTP 프로토콜은 무상태(stateless) 프로토콜임
 - ▶ HTTP 프로토콜은 클라이언트의 정보를 저장하지 않음
 - ▶ 클라이언트의 상태를 저장하지 않는 것은 웹 어플리케이션 개발에 많은 문제를 발생시킴
 - 예를 들어, 웹 메일 시스템에서 여러 메일을 참조하고자 할 경우
 - 예를 들어, 쇼핑몰 사이트에서 로그인 상태나 장바구니를 구현하는 경우
- 상태를 저장하지 않는 것을 해결할 수 있는 방법이 필요
 - ▶ 쿠키(cookies)나 세션(session)을 사용해 해결
- 세션 객체는 세션을 다루는 객체임
 - ▶ session 은 접속하는 사용자 별로 따로 생성되며 일정시간 유지되고 소멸
 - ▶ 차후 쿠키와 세션을 다룰 때 자세히 학습할 예정

config 내장객체

config 내장 객체

■ config 내장 객체

- 서블릿이 최초로 메모리에 적재될 때 컨테이너가 서블릿 초기화와 관련된 정보를 읽어 저장하는 객체
- 어플리케이션을 구성하는 각종 서블릿, JSP, 빈즈 사이의 데이터를 공유하기 위해 사용 가능
 - ▶ 순수하게 JSP 만을 사용하거나 데이터를 공유하는 다른 방법을 사용한다면 config 객체를 사용하는 일은 거의 없음
- config는 거의 사용되지 않음
 - ▶ config 내장 객체를 통해 생성되는 application 내장 객체를 주로 사용
 - ▶ 어플리케이션에서 사용되는 초기와 파라미터 지정도 web.xml 이 아닌 application 객체의 메서드로 이루어짐

application 내장 객체

application 내장 객체

■ application 내장 객체

- 어플리케이션이 실행되는 환경 정보들을 담고 있으며, 웹 어플리케이션 전체에서 사용되는 객체
- 웹 어플리케이션에 포함되어 있는 모든 문서들은 application 내장 객체를 공유
 - JSP가 제공하는 내장 객체 중 역할이나 기능으로 봤을 때 가장 큰 객체
 - 하나의 웹 어플리케이션 당 하나의 application 내장 객체가 생성
- 유형별로 많은 메서드를 제공하므로 주로 관리 기능의 웹 애플리케이션 개발에 유용
 - 어플리케이션 전체가 공유할 수 있는 초기 파라미터 정보
 - 컨테이너와 관련된 정보, 로그 정보 등

application 내장 객체의 메서드

■ application 객체의 메서드

• 웹 컨테이너의 정보를 추출하기 위한 메서드

메서드	반환 타입	설명
getServletInfo()	String	웹 컨테이너의 이름과 버전을 반환
getMajorVersion()	String	서블릿 API의 버전 중 정수 부분(메이저 버전)을 반환
getMinorVersion()	String	서블릿 API의 버전 중 소수 부분(마이너 버전)을 반환

• 웹 어플리케이션 속성 관련 메서드

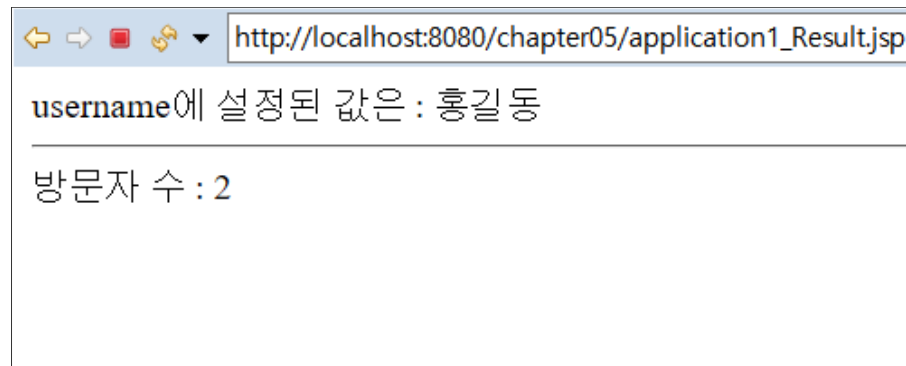
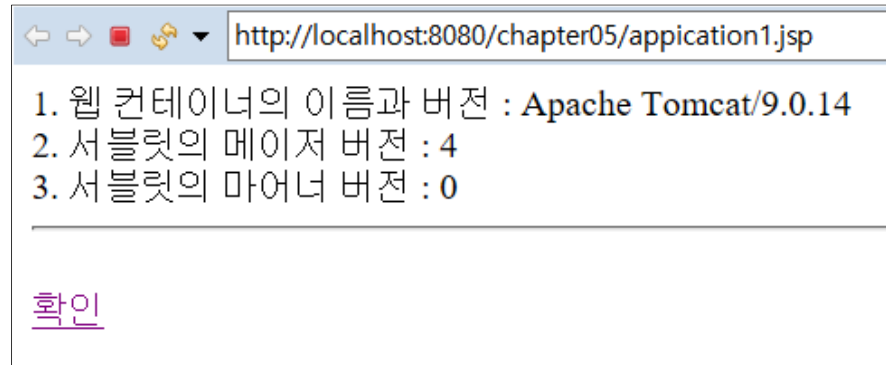
메서드	반환 타입	설명
getAttribute(name)	object	이름이 name인 속성의 값을 반환(name이 존재하지 않으면 null을 반환)
getAttributeNames()	Enumeration	속성 이름들을 Enumeration 타입으로 반환
setAttribute(name, value)	void	이름이 name인 속성 값을 value로 지정
removeAttribute(name)	String	이름이 name인 속성을 삭제

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <meta charset="UTF-8">
7 </head>
8 <body>
9
10     1. 웹 컨테이너의 이름과 버전 : <%=application.getServerInfo()%> <br>
11     2. 서블릿의 메이저 버전 : <%=application.getMajorVersion()%> <br>
12     3. 서블릿의 마이너 버전 : <%=application.getMinorVersion()%> <br>
13     <hr>
14 <%
15     application.setAttribute("username", "홍길동");
16     application.setAttribute("count", 1);
17 %>
18     <br>
19     <a href="application1_Result.jsp">확인</a>
20
21 </body>
22 </html>
```

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2     pageEncoding="UTF-8"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6     <meta charset="UTF-8">
7 </head>
8 <body>
9
10     username에 설정된 값은 : <%=application.getAttribute("username") %>
11     <hr>
12 <%
13     Integer count = (Integer)application.getAttribute("count");
14     int cnt = count.intValue()+1;
15
16     application.setAttribute("count", cnt);
17 %>
18
19 방문자 수 : <%=cnt%>
20
21 </body>
22 </html>
```

수정 가능

```
int cnt = (int)application.getAttribute("count") + 1;
```



application 내장 객체의 메서드

- 어플리케이션의 정보를 추출하기 위한 메서드

메서드	반환 타입	설명
getMimeType(filename)	String	인자로 지정된 파일의 MIME 타입을 반환
getResource(path)	URL	인자로 지정된 경로의 자원에 접근할 수 있는 URL 객체를 반환
getRealPath(path)	String	인자로 지정된 경로의 자원에 대한 절대 경로를 반환
getResourceAsStream(path)	InputStream	인자로 지정된 경로의 자원에 대한 InputStream을 반환
getContextPath()	String	어플리케이션의 이름을 반환

application 내장 객체의 메서드

■ 실습

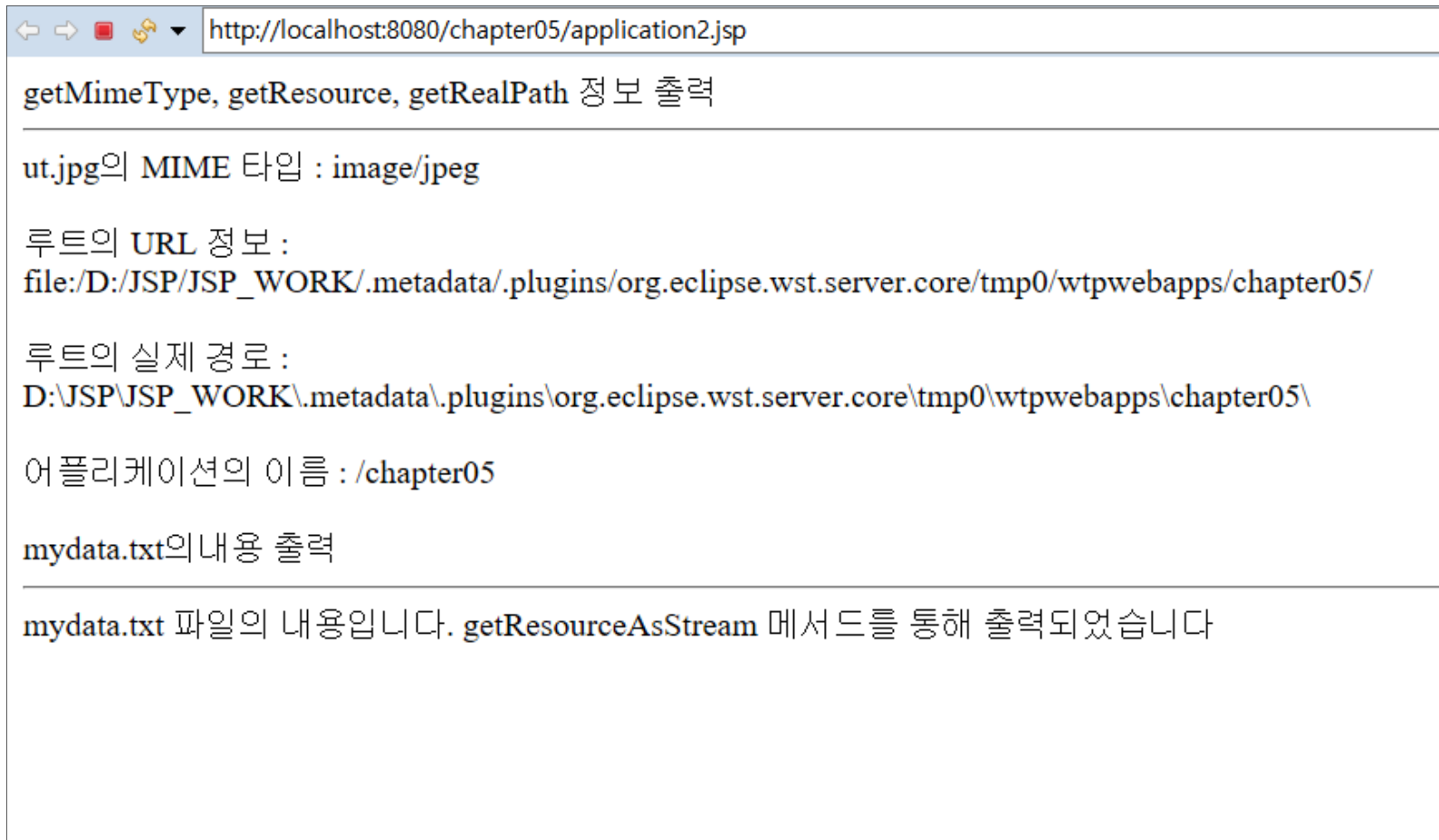
- WebContent 디렉터리에 파일을 생성
 - ▶ 이미지 파일을 생성
 - coffee.jpg 이미지를 저장(다른 이미지도 무관)
 - ▶ 텍스트 파일을 생성
 - mydata.txt 파일을 생성

```
mydata.txt ✕  
1 mydata.txt 파일의 내용입니다.  
2  
3 getResourceAsStream 메서드를 통해 출력되었습니다
```

```
chapter05  
  > Deployment Descriptor: chapter05  
  > JAX-WS Web Services  
  > Java Resources  
  > JavaScript Resources  
  > Referenced Libraries  
  > build  
  > WebContent  
    > META-INF  
    > WEB-INF  
      application1.jsp  
      application1_Result.jsp  
      coffee.jpg  
      getParameter1.jsp
```



```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import = "java.io.*" %>
4
5   getMimeType, getResource, getRealPath 정보 출력
6   <HR>
7   ut.jpg의 MIME 타입 : <%=application.getMimeType("coffee.jpg")%> <br><br>
8   루트의 URL 정보 : <br><%=application.getResource("/")%> <br><br>
9   루트의 실제 경로 : <br> <%=application.getRealPath("/")%> <br><br>
10  어플리케이션의 이름 : <%=application.getContextPath()%> <br><br>
11 <%
12   String PathAndName = "./mydata.txt";
13
14   char[] my_buff = new char[512];
15   int length = -1;
16
17   InputStreamReader isr = new InputStreamReader(application.getResourceAsStream(PathAndName));
18   BufferedReader br = new BufferedReader(isr);
19
20   out.println("mydata.txt의내용 출력<HR>");
21   while( (length = br.read(my_buff)) != -1) {
22       out.println(new String(my_buff, 0, length));
23   }
24
25   br.close();
26   isr.close();
27 %>
```



```
← → ⌂ ↻ http://localhost:8080/chapter05/application2.jsp

getMimeType, getResource, getRealPath 정보 출력

ut.jpg의 MIME 타입 : image/jpeg

루트의 URL 정보 :
file:/D:/JSP/JSP_WORK/.metadata/.plugins/org.eclipse.wst.server.core/tmp0/wtpwebapps/chapter05/

루트의 실제 경로 :
D:\JSP\JSP_WORK\metadata\plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps\chapter05\

어플리케이션의 이름 : /chapter05

mydata.txt의내용 출력

mydata.txt 파일의 내용입니다. getResourceAsStream 메서드를 통해 출력되었습니다
```

[참고] web.xml 파일의 활용

■ web.xml 파일을 이용한 초기 파라미터 지정

- 초기 파라미터 값 지정
 - ▶ [동적 웹 프로젝트]WEB-INF\web.xml 문서에 지정
 - ▶ 초기 파라미터는 웹 어플리케이션이 시작될 때 활성화
- application 기본 객체를 사용해 초기 파라미터를 추출해 사용 가능
- 다른 프로젝트에서 실습
 - ▶ 새로운 프로젝트를 생성하여 실습 수행
 - ▶ test 프로젝트를 생성
 - web.xml 파일을 반드시 생성해야 함

[참고] web.xml 파일의 활용

■ 초기 파라미터 생성

- 초기 파라미터는 web.xml 문서에 <context-param> 태그를 사용해 생성

```
<context-param>  
    <description> 초기 파라미터 설명 문자열 </description>  
    <param-name> 초기 파라미터 이름 </param-name>  
    <param-value> 초기 파라미터 값 </param-value>  
</context-param>
```

태그	설명
<context-param>	초기 파라미터를 지정하기 위한 정보를 포함하는 태그
<description>	초기 파라미터의 설명을 나타냄 (생략 가능)
<param-name>	초기 파라미터의 이름을 나타냄
<param-value>	<param-name> 태그로 지정한 파라미터의 값을 나타냄

[참고] web.xml 파일의 활용

- test 프로젝트의 web.xml 파일

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="
3   <display-name>test</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.htm</welcome-file>
7     <welcome-file>index.jsp</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.htm</welcome-file>
10    <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12 </web-app>
```

[참고] web.xml 파일의 활용

- test 프로젝트의 web.xml 파일 수정

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.
3   <display-name>test</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.htm</welcome-file>
7     <welcome-file>index.jsp</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.htm</welcome-file>
10    <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12
13    <context-param>
14      <description>소프트웨어 버전</description>
15      <param-name>Version</param-name>
16      <param-value>version2.0</param-value>
17    </context-param>
18
19    <context-param>
20      <description>개발자</description>
21      <param-name>name</param-name>
22      <param-value>김철수</param-value>
23    </context-param>
24
25 </web-app>
```

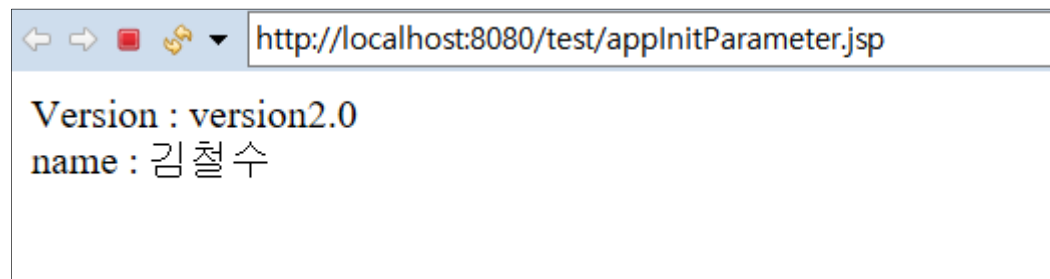
[참고] web.xml 파일의 활용

■ 초기 파라미터의 추출

- web.xml 문서에 설정된 초기 파라미터를 추출하는 메서드

메서드	반환 타입	설명
getInitParameterNames()	Enumeration	초기 파라미터의 이름을 Enumeration 타입으로 반환
getInitParameter(String name)	String	이름이 name인 초기 파라미터의 값을 반환 (존재하지 않으면 null을 반환)

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.util.Enumeration" %>
4
5 <%
6
7   Enumeration<String> InitParm = application.getInitParameterNames();
8
9   while(InitParm.hasMoreElements()) {
10     String ParamName = (String)InitParm.nextElement();
11     out.println(ParamName + " : " + application.getInitParameter(ParamName) + "<BR>");
12   }
```



page 내장 객체

page 내장 객체

■ page 내장 객체

- JSP에서 자기 자신을 참조할 때 사용
 - ▶ JSP는 page 참조 변수를 참조하지 않고 생성된 서블릿 클래스의 멤버 변수나 메서드에 직접 접근할 수 있음
- page 참조 변수는 거의 사용하지 않음

pageContext 내장 객체

pageContext 내장 객체

■ pageContext 내장 객체

- pageContext 내장 객체의 주요 용도
 - 다른 모든 내장 객체에 대한 프로그램적인 접근 방법을 제공
 - 제어를 다른 페이지로 넘기기 위해 사용 (제어의 이동)
 - 특정 문서의 실행 결과를 포함시킬 때 사용

pageContext 내장 객체의 메서드

■ 다른 내장 객체에 대한 프로그램적인 접근

pageContext.기본객체참조메서드.기본객체메서드

- pageContext 내장 객체를 사용해 다른 내장 객체의 메서드를 사용

▶ [예] pageContext.getOut().println() = out.println()

메서드	반환 타입	설명
getRequest()	ServletRequest	request 내장 객체를 반환
getResponse()	ServletResponse	response 내장 객체를 반환
getOut()	JspWriter	out 내장 객체를 반환
getSession()	HttpSession	session 내장 객체를 반환
getPage()	Object	page 내장 객체를 반환
getServletConfig()	ServletConfig	config 내장 객체를 반환
getServletContext()	ServletContext	application 내장 객체를 반환
getException()	Exception	exception 객체를 반환

pageContext 내장 객체의 메서드

■ 제어의 전달과 실행 결과의 포함

• 제어의 전달

- ▶ 어떤 JSP 문서가 다른 JSP 문서를 브라우저에 출력하도록 하는 것
- ▶ response 내장 객체의 sendRedirect() 메서드와 유사한 기능 수행

• 실행 결과의 포함

- ▶ 다른 JSP 문서의 수행 결과를 현재의 문서에 포함시키는 것
- ▶ include 지시문과 동일한 기능을 수행

메서드	반환 타입	설명
forward(String path)	void	현재 문서에서 path에 지정된 문서로 제어를 이동
include(String path)	void	path에 해당하는 문서의 수행 결과를 현재 문서에 포함

pageContext 내장 객체의 메서드

■ forward() 메서드

```
pageContext.forward("이동할 페이지나 URL");
```

- 인자로 가지는 JSP 문서나 URL로 제어를 이동시키는 기능을 수행
 - 제어가 이동된 후 브라우저의 주소표시줄에 제어를 이동시킨 JSP 문서가 출력됨
 - 제어를 이동한 문서가 출력되지 않음
- 제어가 이동되면서 새로운 request 객체가 생성되지 않고 기존 request 객체가 전달됨
 - sendRedirect() 메서드의 경우 기존 request 객체는 소멸되고 새로운 request 객체가 생성
 - response 내장 객체의 sendRedirect() 메서드와 유사한 기능 수행
- forward() 메서드는 forward 액션과 동일한 기능을 수행
 - 액션 태그에서 자세히 설명할 것임

pageContext 내장 객체의 메서드

pageContextForwardFrom.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%
4   pageContext.forward("pageContextForwardTo.jsp");
5 %>
```

pageContextForwardTo.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%
4   out.println("pageContextForwardTo.jsp 문서의 내용입니다");
5 %>
```

← → 🛑 🔄 ▼ http://localhost:8080/chapter05/pageContextForwardFrom.jsp

pageContextForwardTo.jsp 문서의 내용입니다

pageContext 내장 객체의 메서드

■ include() 메소드

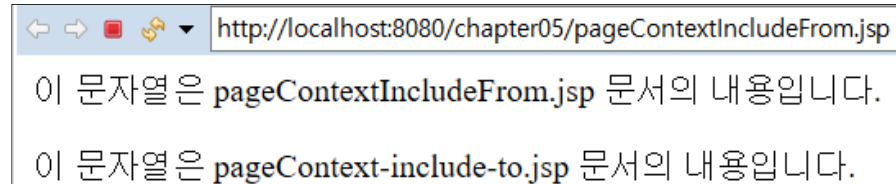
```
pageContext.include("실행결과를 포함시킬 문서이름");
```

- 인자로 가지는 JSP 문서의 실행 결과를 현재의 문서에 포함시키는 기능을 수행
- include() 메서드와 include 디렉티브와의 동일한 결과를 출력하지만 차이점이 있음
 - ▶ include 디렉티브의 경우
 - 인자로 가지는 문서를 현재 문서에 포함시킨 후 서블릿으로 변환시키고 컴파일을 수행
 - ▶ include() 메서드의 경우
 - 인자로 가지는 문서를 별도로 컴파일해 실행한 다음 그 결과만을 현재 문서에 포함시킴
 -
- include 액션 태그와는 동일한 기능을 수행
 - ▶ 액션 태그에서 자세히 설명

pageContext 내장 객체의 메서드

pageContextIncludeFrom.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%
4   out.println("이 문자열은 pageContextIncludeFrom.jsp 문서의 내용입니다.<BR><BR>");
5   pageContext.include("pageContextIncludeTo.jsp");
6 %>
```



http://localhost:8080/chapter05/pageContextIncludeFrom.jsp

이 문자열은 pageContextIncludeFrom.jsp 문서의 내용입니다.

이 문자열은 pageContext-include-to.jsp 문서의 내용입니다.

pageContextIncludeTo.jsp

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%
4   out.println("이 문자열은 pageContext-include-to.jsp 문서의 내용입니다.<BR><BR>");
5 %>
```

Exception 객체

exception 내장 객체

■ exception 내장 객체

- JSP 문서를 실행하는 도중 예외가 발생할 경우 예외를 처리할 문서에 전달되는 객체
 - 하나의 어플리케이션 마다 하나의 exception 객체가 생성됨
 - 주로 try~catch 구문에서 예외 처리를 위해 사용

• 주요 메서드

메서드	반환 타입	설명
getMessage()	String	오류 메시지를 문자열로 반환
printStackTrace()	void	발생된 예외를 추적하기 위해 표준 스트림으로 스택 정보를 출력
toString()	String	예외를 발생시킨 클래스의 이름과 함께 오류 메시지를 문자열로 반환

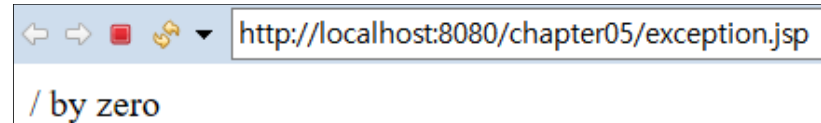
```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%
4   try{
5
6       int a = 20, b = 0;
7       int c = 0;
8
9       //인위적으로 오류 발생
10      c = a/b;
11
12   } catch(Exception e) {
13
14       out.println(e.toString());
15
16   }
17 %>
```

← → ⏏ 🔄 ▼ http://localhost:8080/chapter05/exception.jsp

java.lang.ArithmeticException: / by zero

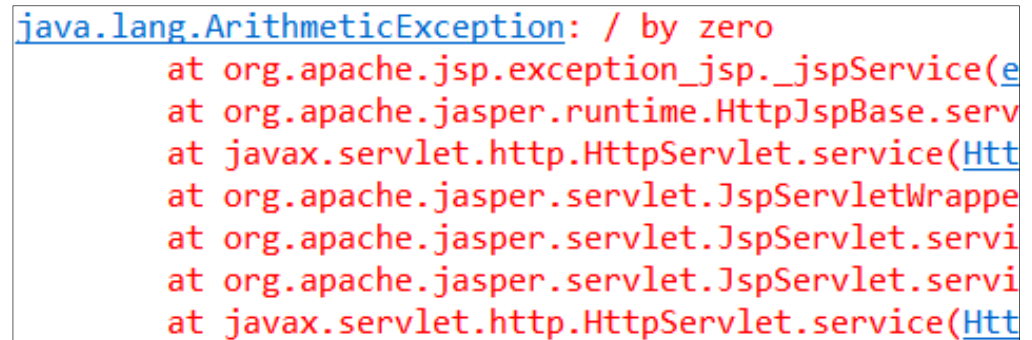
exception 내장 객체

```
12 } catch(Exception e) {  
13  
14     out.println(e.getMessage());  
15  
16 }
```



A screenshot of a web browser window. The address bar shows the URL `http://localhost:8080/chapter05/exception.jsp`. The main content area displays the text `/ by zero`, which is the message returned by `e.getMessage()` for a `java.lang.ArithmeticException`.

```
12 } catch(Exception e) {  
13  
14     e.printStackTrace();  
15  
16 }
```



A screenshot of a web browser window showing the output of the second code snippet. The output is a full Java stack trace for a `java.lang.ArithmeticException: / by zero`. The stack trace includes the following frames (from top to bottom):
`java.lang.ArithmeticException: / by zero`
`at org.apache.jsp.exception_jsp._jspService(e`
`at org.apache.jasper.runtime.HttpJspBase.serv`
`at javax.servlet.http.HttpServlet.service(Http`
`at org.apache.jasper.servlet.JspServletWrappe`
`at org.apache.jasper.servlet.JspServlet.servi`
`at org.apache.jasper.servlet.JspServlet.servi`
`at javax.servlet.http.HttpServlet.service(Http`

[참고] JSP에서 오류 처리

■ JSP에서 오류 처리

- JSP의 실행은 오류를 발생할 수 있으므로 오류를 대처하는 방법이 필요함
 - ▶ 작은 오류로 인해 발생하는 오류 메시지도 사용자 입장에서는 큰 오류로 인식될 수 있음

HTTP Status 404 – Not Found

Type Status Report

Message /pro12/sum.jsp

Description The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

Apache Tomcat/9.0.6

HTTP Status 500 – Internal Server Error

Type Exception Report

Message An exception occurred processing [/test02/add.jsp] at line [5]

Description The server encountered an unexpected condition that prevented it from fulfilling the request.

Exception

org.apache.jasper.JasperException: An exception occurred processing [/test02/add.jsp] at line [5]

2: pageEncoding="UTF-8" %>

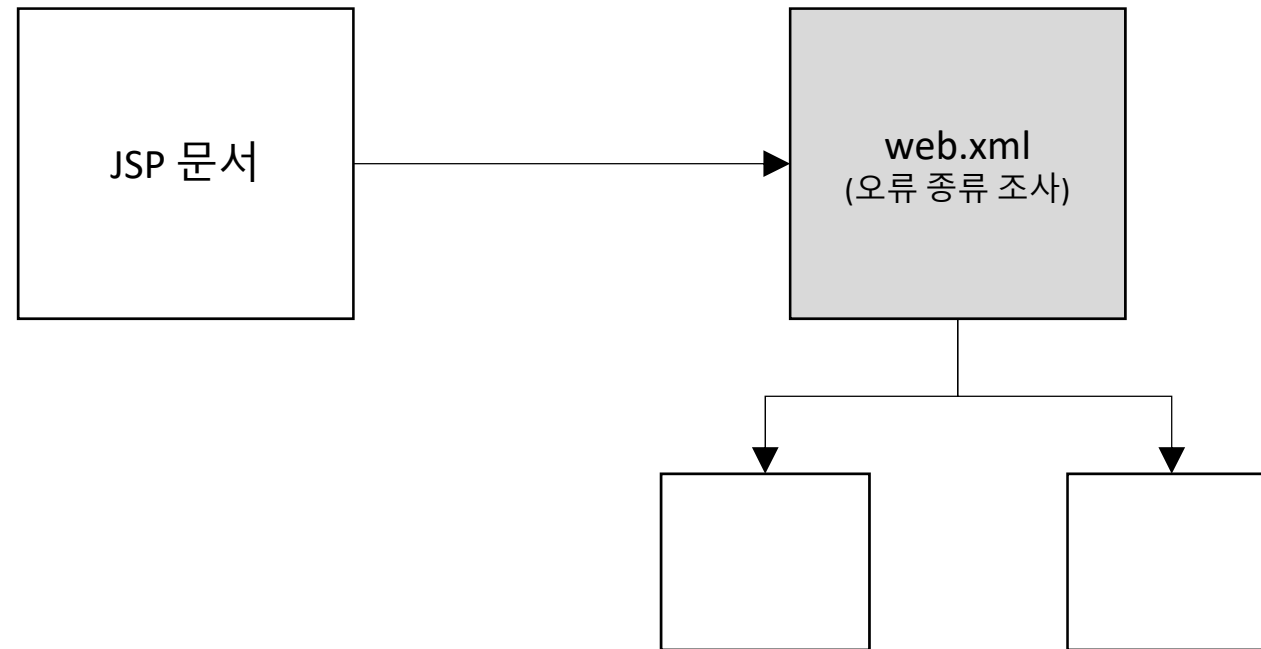
[참고] JSP에서 오류 처리

■ 오류 발생을 위한 예외 처리의 방법

- page 디렉티브의 isErrorPage와 errorPage 속성을 사용하는 방법
 - ▶ 오류 처리를 담당하는 문서를 작성
 - page 지시어의 isErrorPage 속성에 true를 지정
 - ▶ 오류 발생 시 오류 처리를 담당하는 JSP 파일을 지정
 - page 지시어의 errorPage 속성에 오류 처리 페이지의 이름을 지정
- web.xml 문서를 사용하는 방법
 - ▶ web.xml 문서에 오류의 종류에 따라 수행할 예외처리 문서를 지정하는 방법
- JSP 2.0이상에서는 errorPage와 isErrorPage 속성을 사용해 예외를 처리하지 않음
 - ▶ web.xml 파일의 <error-page> 태그를 사용해 예외를 처리하도록 권고하고 있음

[참고] JSP에서 오류 처리

- web.xml 문서를 사용하는 방법



오류 종류에 따른 처리 문서 실행

[참고] JSP에서 오류 처리

■ web.xml 문서를 이용한 오류 처리 실습

- 다른 프로젝트에서 실행
 - ▶ 이전에 web.xml을 이용한 초기화 파라미터를 실습한 test 프로젝트를 사용
- 오류 처리를 위한 문서 생성
 - ▶ 404번 오류 : WebContent/err_404.jsp
 - ▶ 500번 오류 : WebContent/err_500.jsp
- 인위적인 오류 발생
 - ▶ WebContent/error.jsp

[참고] JSP에서 오류 처리

err_404.jsp

```
9 <body>
10 <div align=center>
11     [404 오류]요청하신 페이지는 찾을 수 없습니다.
12 </div>
13 </body>
--
```

err_500.jsp

```
9 <body>
10 <div align=center>
11     [500 오류]요청한 문서를 처리할 수 없습니다.
12 </div>
13 </body>
--
```

[참고] JSP에서 오류 처리

- test 프로젝트의 web.xml 문서 수정

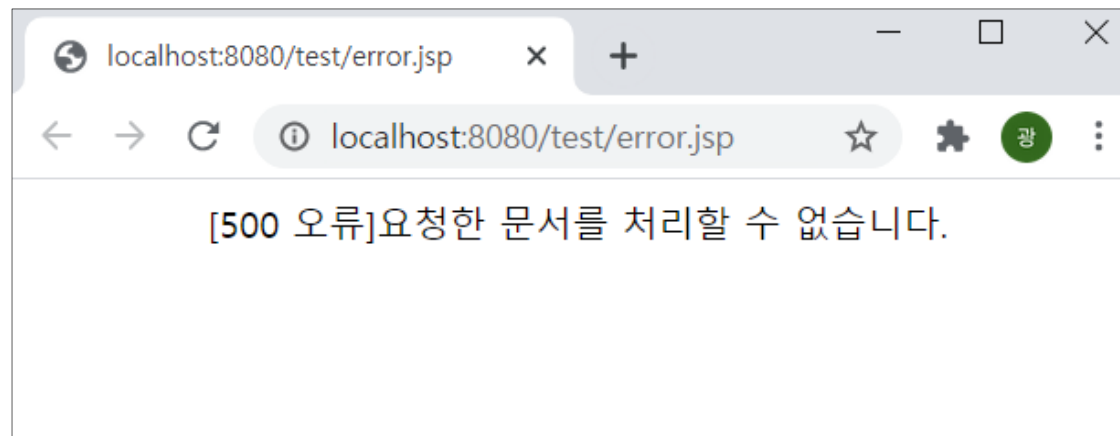
```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xmlns-com
3   <display-name>test</display-name>
4   <welcome-file-list>
5     <welcome-file>index.html</welcome-file>
6     <welcome-file>index.htm</welcome-file>
7     <welcome-file>index.jsp</welcome-file>
8     <welcome-file>default.html</welcome-file>
9     <welcome-file>default.htm</welcome-file>
10    <welcome-file>default.jsp</welcome-file>
11  </welcome-file-list>
12
13  <error-page>
14    <error-code>404</error-code>
15    <location>/err_404.jsp</location>
16  </error-page>
17
18  <error-page>
19    <error-code>500</error-code>
20    <location>/err_500.jsp</location>
21  </error-page>
22
23 </web-app>
```

[참고] JSP에서 오류 처리

error.jsp

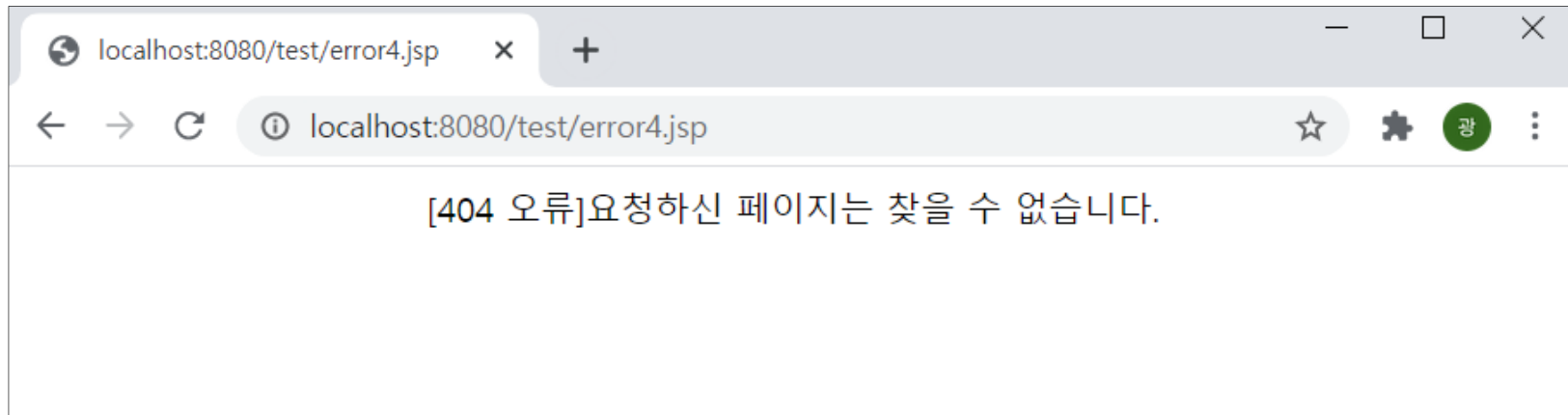
```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%@ page import="java.util.Date" %>
4
5 <%
6
7   Date my_date = new Date();
8   String today = my_data.toString();           // 인위적으로 오류 발생
9   out.println(today);
10
11 %>
```

브라우저를 크롬으로 변경하여 실행



[참고] JSP에서 오류 처리

- 존재하지 않는 문서를 요청하는 경우



내장 객체와 속성

JSP 내장 객체와 영역

■ HTTP 프로토콜 특징과 내장객체 속성 관리

- JSP는 HTTP 프로토콜의 사용하는 웹 환경에서 구동되는 프로그램임
 - ▶ HTTP는 비연결형으로 사용자가 서버에 특정 페이지를 요청에 대한 응답이 수행되면 서버와의 연결이 끊기는 형태임
- 비연결형 문제점의 예
 - ▶ 게시판에 글을 작성하는 페이지에서 작성한 내용은 다른 jsp에서 처리되므로 서버는 글을 작성한 사람을 모름
 - ▶ 쇼핑몰에서 여러 상품 페이지를 이동하면서 장바구니에 물건을 담아 두고 한꺼번에 구매하기 어려움
- JSP에서는 내장 객체를 통해 서로 다른 페이지에서 처리된 값을 저장하고 공유하기 위한 방법을 제공함
 - ▶ page, request, session, application 내장 객체에 대한 영역을 사용
 - ▶ 이는 컨테이너 기반 프로그램의 특징 중 하나로 실제 프로그램 구현 시 매우 중요한 기법임

JSP 내장 객체와 영역

■ 내장 객체의 영역

- JSP에는 네 개의 내장 객체와 연관성이 있는 네 개의 영역(scope)가 존재

영역	설명
page 영역	하나의 JSP 문서를 처리할 때 사용되는 영역
request 영역	하나의 HTTP 요청을 처리할 때 사용되는 영역
session 영역	하나의 브라우저와 관련해 사용되는 영역
application 영역	하나의 웹 어플리케이션과 관련해 사용되는 영역

JSP 내장 객체와 영역

- application

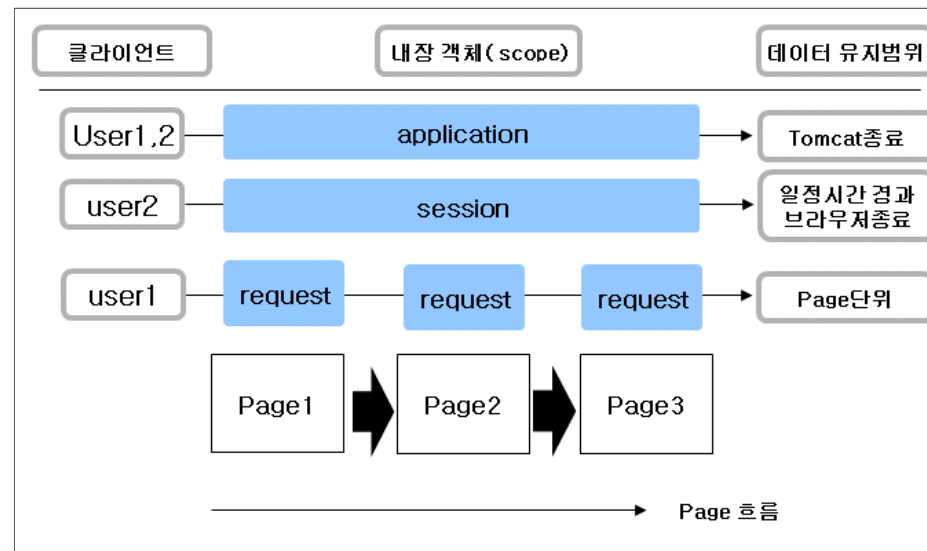
- ▶ 모든 사용자가 공유하는 데이터를 저장할 수 있음
 - 톰캣이 종료될 때 까지 데이터를 유지할 수 있음

- session

- ▶ 사용자마다 분리된 저장 영역이 있음
 - Page1, Page2, Page3 모두에서 공유되는 정보를 관리할 수 있음 (이 데이터는 사용자 간에는 공유되지 않음)

- request

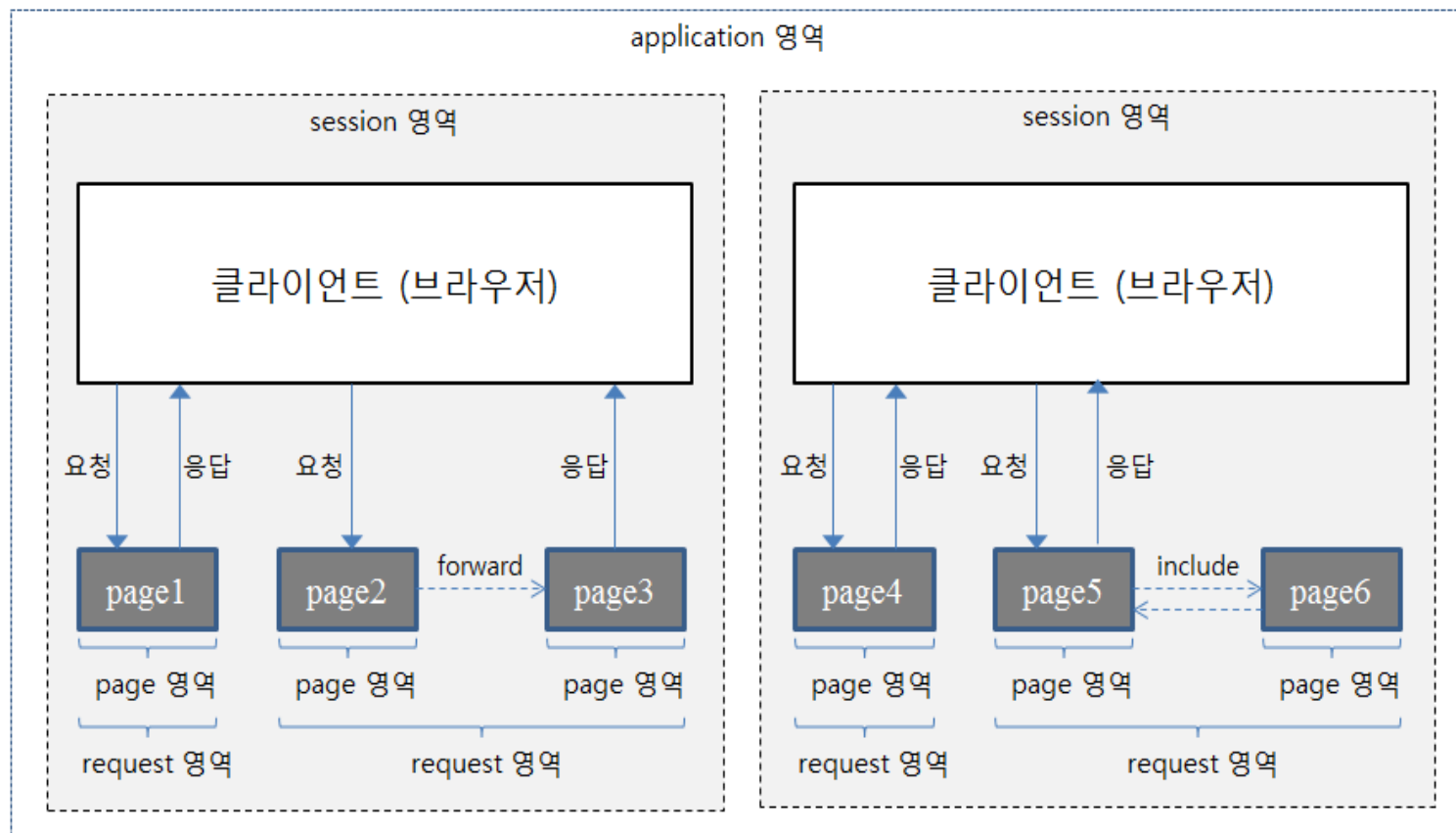
- ▶ 한 페이지에서 다른 페이지로 데이터를 전달할 경우 request 내장 객체가 생성



JSP 내장 객체와 영역

■ 내장 객체 영역 간의 관계

- page1부터 page6이 모두 동일한 어플리케이션 내에 존재한다고 가정할 경우



JSP 내장 객체와 영역

■ page 영역

• page 영역의 범위

- ▶ 한 번의 HTTP 요청에 대해 하나의 JSP 문서가 처리되는 동안은 영역의 범위로 가짐
- ▶ page1부터 page6까지의 모든 JSP 문서가 요청에 의해 처리될 경우
 - 각각 자신만의 page 영역을 가지게 됨
- ▶ 만약, page1에 A라는 속성이 존재할 경우
 - A라는 속성은 page1에서만 사용가능하며 다른 모든 문서에서는 참조할 수 없음

• 페이지 영역의 생성과 소멸

- ▶ JSP 문서의 처리를 시작할 때 생성
- ▶ JSP 문서의 처리가 완료될 때 소멸

JSP 내장 객체와 영역

■ request 영역

- request 영역의 범위

- ▶ 하나의 HTTP 요청이 처리되는 동안은 영역의 범위로 가짐
- ▶ URL을 입력하거나 한 JSP 문서에서 다른 JSP 문서를 요청할 때 request 영역이 생성
 - request 내장 객체가 생성될 때 마다 하나의 request 영역이 생성
- ▶ page1과 page4는 요청에 의해 request 내장 객체가 생성되므로 page 영역과 함께 request 영역을 가지게 됨
- ▶ page2에서 forward라는 액션 태그가 실행될 경우 page2에서 page3으로 제어가 이동
 - request 내장 객체가 전달되므로 page2와 page3은 같은 request 영역을 가짐
- ▶ page5에서 include라는 액션 태그가 실행될 경우 page5와 page6 사이에 제어가 이동
 - request 내장 객체가 전달되므로 page5와 page6은 같은 request 영역을 가지게 됨

- request 영역의 생성과 소멸

- ▶ request 영역은 브라우저로 HTTP 요청이 시작될 때 생성
- ▶ 브라우저로 응답이 완료될 때 소멸

JSP 내장 객체와 영역

■ session 영역

- 하나의 브라우저에서 수행되는 동일한 어플리케이션 내의 모든 JSP 문서와 관련
 - ▶ 기본적으로 하나의 문서를 브라우저를 통해 열면 하나의 세션이 시작
 - JSP 문서는 session 속성의 기본 값이 true로 지정되어 있음
- 같은 어플리케이션에 존재하는 문서들은 동일한 브라우저로 출력되는 동안 같은 session 영역을 가지게 됨
 - ▶ 하나의 브라우저에서 출력되는 page1, page2, page3는 하나의 session 영역에 포함
 - ▶ 다른 브라우저에서 출력되는 page4, page5, page6은 또 다른 session 영역에 포함
 - ▶ session 영역은 브라우저가 닫힐 때까지 유지
 - 새로운 브라우저를 열어 이전과 동일한 JSP 문서를 실행할 경우, 아닌 새로운 session 영역이 생성됨
- session 영역의 생성과 소멸
 - ▶ 브라우저로 첫 번째 JSP 문서의 요청이 시작될 때 생성
 - ▶ 브라우저가 종료되거나, 세션 타이머가 종료되거나, 인위적으로 세션을 종료시킬 때 소멸

JSP 내장 객체와 영역

■ application 영역

- 하나의 웹 어플리케이션 전체와 관련

- ▶ 현재 개발하고 있는 어플리케이션의 모든 문서들이 하나의 application 영역에 포함
- ▶ page1부터 page6은 동일한 어플리케이션 내에 존재하므로 이들은 모두 하나의 application 영역에 포함

- application 영역의 생성과 소멸

- ▶ 서버가 시작될 때 모든 어플리케이션이 활성화되며, 각각의 어플리케이션 마다 하나의 application 영역이 생성
- ▶ tomcat 서버가 종료될 때 소멸됨

JSP 내장 객체와 영역

■ 내장 객체 영역의 용도

영역	용도
page	하나의 JSP 문서 내에서 공유되는 값을 지정하는데 사용 (별도의 속성을 만들어 사용하는 경우는 거의 없음)
request	하나의 HTTP 요청에 의해 처리될 모든 JSP 문서에서 공유되는 값을 지정하기 위해 사용 주로 request 영역 내의 문서에 데이터를 전달할 목적으로 사용
session	하나의 브라우저에 실행되는 모든 JSP 문서에 공유되는 값을 지정하기 위해 사용 주로 로그인 처리에 사용
application	어플리케이션 전반에 걸쳐 공유되는 값을 지정하기 위해 사용 주로 어플리케이션의 설정 정보를 저장

JSP 내장 객체의 속성

■ 내장 객체에 대한 속성을 관리하기 위한 메서드

메서드	반환 타입	설명
setAttribute(String name, Object value)	void	이름이 name인 속성 값을 value로 지정
getAttribute(String name)	Object	이름이 name인 속성의 값을 반환 (name이 존재하지 않으면 null을 반환)
removeAttribute(String name)	void	이름이 name인 속성을 삭제
getAttributeNames()	Enumeration	속성 이름들을 Enumeration 타입으로 반환

```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%
4
5   pageContext.setAttribute("season1", "봄");
6   String season1 = (String)pageContext.getAttribute("season1");
7
8   request.setAttribute("season2", "여름");
9   String season2 = (String)request.getAttribute("season2");
10
11   session.setAttribute("season3", "가을");
12   String season3 = (String)session.getAttribute("season3");
13
14   application.setAttribute("season4", "여름");
15   String season4 = (String)application.getAttribute("season4");
16
17 %>
18   page 영역 : <%=season1 %> <br>
19   request 영역 : <%=season2 %> <br>
20   session 영역 : <%=season3 %> <br>
21   application 영역 : <%=season4 %> <br>
```

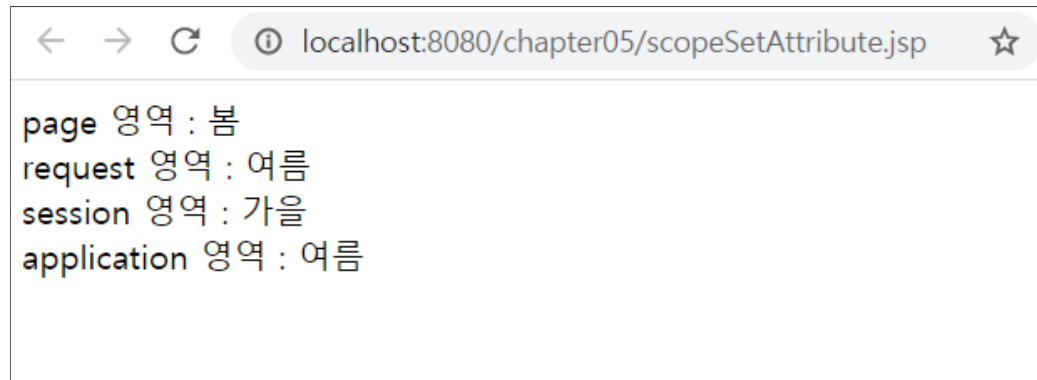
```
1 <%@ page language="java" contentType="text/html; charset=UTF-8"
2   pageEncoding="UTF-8"%>
3 <%
4   String season1 = (String)pageContext.getAttribute("season1");
5   String season2 = (String)request.getAttribute("season2");
6   String season3 = (String)session.getAttribute("season3");
7   String season4 = (String)application.getAttribute("season4");
8 %>
9   page 영역 : <%=season1 %> <br>
10  request 영역 : <%=season2 %> <br>
11  session 영역 : <%=season3 %> <br>
12  application 영역 : <%=season4 %> <br>
13
```

JSP 내장 객체의 속성

■ 실습 1

- scopeSetAttribute.jsp 문서 실행

- ▶ 브라우저를 크롬으로 지정하고 실행

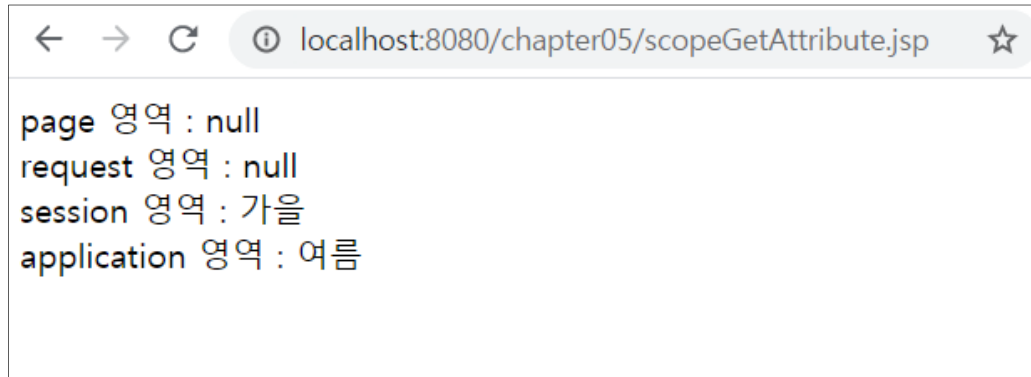


- ▶ 현재 4개의 영역에 각각 4개의 속성이 정의되어 있으므로 모두 출력됨

JSP 내장 객체의 속성

■ 실습 2

- object-attribute-1.jsp가 실행된 브라우저에서 object-attribute-2.jsp를 실행
 - ▶ object-attribute-1.jsp가 실행된 브라우저를 닫지 말고 주소표시줄에 object-attribute-2.jsp를 입력해 실행



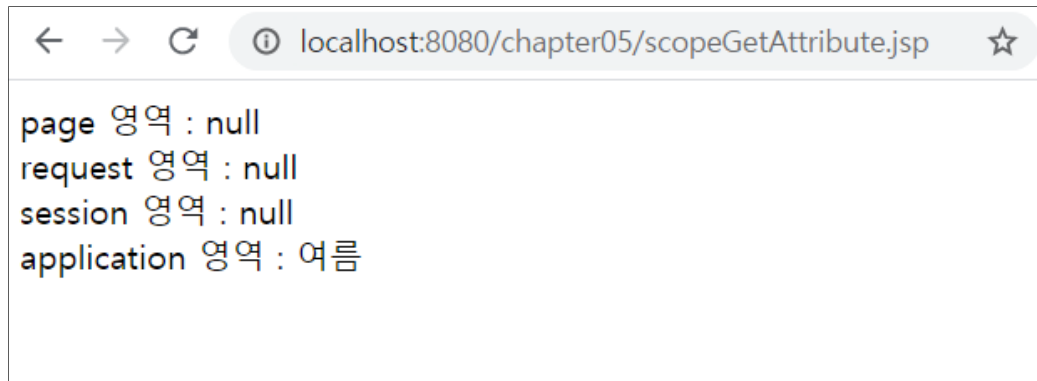
```
← → ↻ ⓘ localhost:8080/chapter05/scopeGetAttribute.jsp ☆  
page 영역 : null  
request 영역 : null  
session 영역 : 가을  
application 영역 : 여름
```

- ▶ 브라우저에 수행되는 페이지가 변경되었으므로 새로운 page 객체가 생성됨
 - 기존 page 객체의 season1 속성은 제거됨
- ▶ object-attribute-1.jsp에서 object-attribute-2.jsp로 제어가 이동된 경우가 아니므로 새로운 request 객체가 생성됨
 - 기존 request 객체의 season2 속성은 제거됨
- ▶ 이전 브라우저가 종료되지 않았으므로 session 객체는 유지되고 있음
- ▶ 톰캣이 종료되지 않았으므로(어플리케이션이 종료되지 않았으므로) application 객체는 유지됨

JSP 내장 객체의 속성

■ 실습 3

- 현재의 브라우저를 닫고 새로운 브라우저를 열어 object-attribute-2.jsp를 실행

A screenshot of a web browser window. The address bar shows 'localhost:8080/chapter05/scopeGetAttribute.jsp'. The main content area displays the following text:

```
page 영역 : null
request 영역 : null
session 영역 : null
application 영역 : 여름
```

- ▶ 기존 브라우저가 종료되고 새로운 브라우저에서 생성되므로 새로운 session 객체가 생성됨
 - 기존 session 객체의 season3 속성은 제거됨
- ▶ 톰캣이 종료되지 않았으므로(어플리케이션이 종료되지 않았으므로) appliaocation 객체는 유지됨

수고하셨습니다