

제7장 JSP 내장 객체

- 내장 객체

내장 객체 개요

2

□ 내장 객체(Implicit Object)

▣ JSP 페이지에서 선언 없이 이용할 수 있는 객체 변수

■ 객체 변수 **out**은 **JSP** 서블릿에서 자동으로 선언

- JSP 페이지에서는 선언 없이 `out.println()`을 사용 가능
- 스크립트릿과 선언에서 사용

□ 종류

▣ 객체 변수로는 **out**을 비롯하여

▣ **request**와 **response** 등 9개

부류	java.lang	javax.servlet	javax.servlet.http	javax.servlet.jsp
JSP 페이지에 관련된 객체	page	config		
페이지 입출력에 관련된 객체			request, response	out
컨텍스트에 관련된 객체		application	session	pageContext
에러에 관련된 객체	exception			

내장 객체 종류

3

내장 객체	소속 패키지	클래스 이름	사용 용도
request	javax.servlet.http	<<interface>> HttpServletRequest	클라이언트의 요청에 의한 폼 양식 정보 처리
response	javax.servlet.http	<<interface>> HttpServletResponse	클라이언트의 요청에 대한 응답
session	javax.servlet.http	<<interface>> HttpSession	클라이언트에 대한 세션 정보 처리
application	javax.servlet	<<interface>> ServletContext	웹 애플리케이션 정보 처리
config	javax.servlet	<<interface>> ServletConfig	현재 JSP 페이지에 대한 환경 처리
exception	java.lang	<<interface>> Throwable	예외처리를 위한 객체
page	java.lang	<class> Object	현재 JSP 페이지에 대한 클래스 정보
pageContext	javax.servlet.jsp	<class> PageContext	현재 JSP 페이지에 대한 페이지 컨텍스트
out	javax.servlet.jsp	<class> JspWriter	출력 스트림

내장 객체의 선언

4

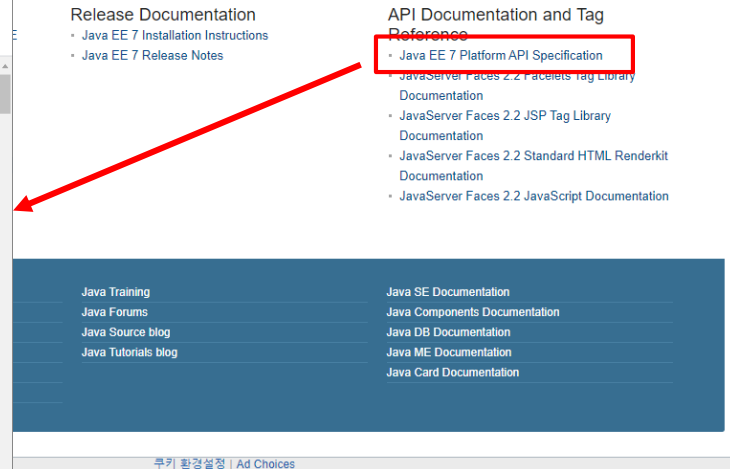
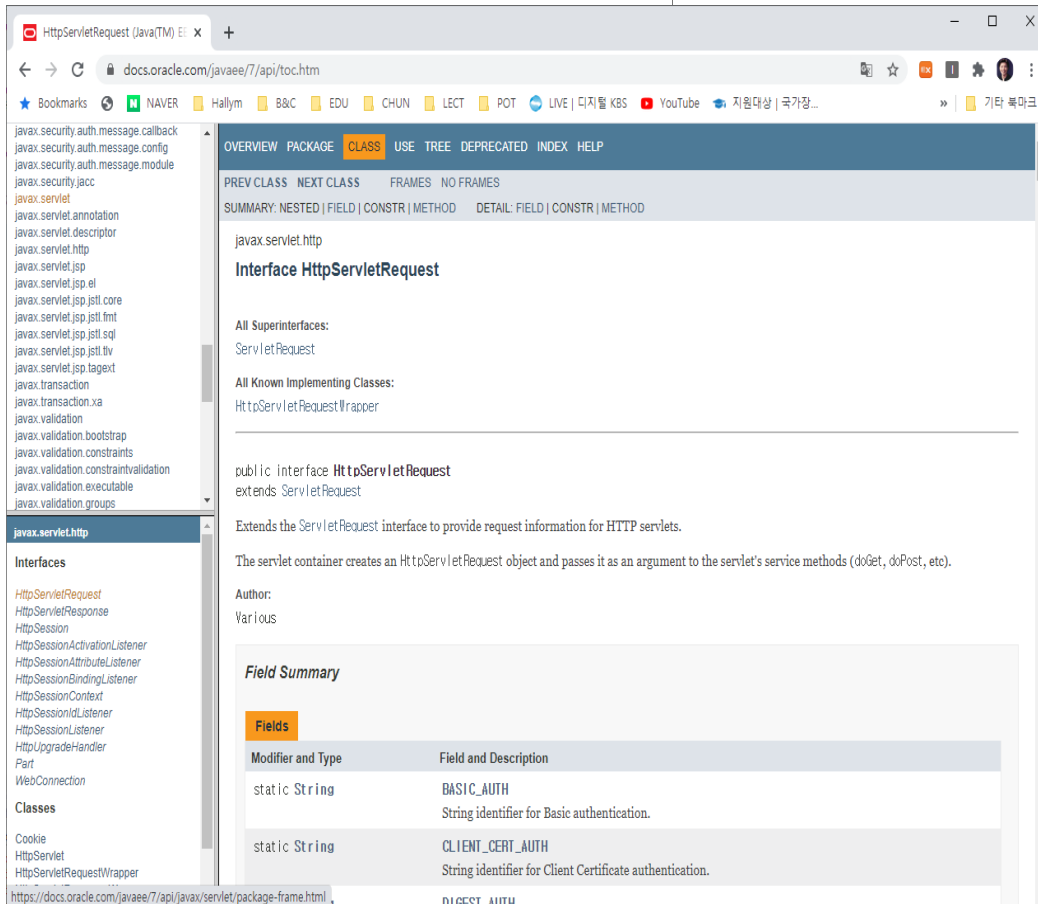
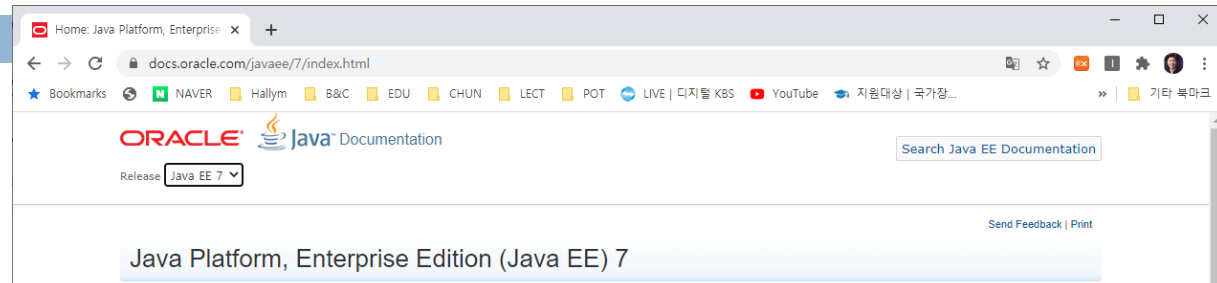
- JSP 서블릿의 메소드 `_jspService()`
 - ▣ 메소드의 첫 부분에 선언되거나 메소드의 매개변수 목록의 변수
 - ▣ exception
 - 페이지 지시자의 속성 `isErrorPage="true"`인 경우에 선언되는 변수
- 내부 객체는 지역 변수 또는 매개 변수
 - ▣ JSP의 선언에서는 이용 불가능
 - ▣ 내부 객체와 같은 이름으로 JSP의 선언에 선언하더라도
 - 지역 변수인 내부 객체와 이름이 충돌하므로 소속 변수로 이용 불가능
 - `<%! int application = 0; %>`
 - `<%= application /* 정수 0이 아니라 내부 객체 application임 */ %>`

```
1 package org.apache.jsp;
2
3 import javax.servlet.*;
4 import javax.servlet.http.*;
5 import javax.servlet.jsp.*;
6
7 public final class error_jsp extends
8     implements org.apache.jasper.runt
9
28
29 public void _jspService(HttpServletRequest request, HttpServletResponse response)
30     throws java.io.IOException, ServletException {
31
32     PageContext pageContext = null;
33     HttpSession session = null;
34     Throwable exception = org.apache.jasper.runtime.JspRuntimeLibrary.getThrowable(request);
35     if (exception != null) {
36         response.setStatus(HttpServletResponse.SC_INTERNAL_SERVER_ERROR);
37     }
38     ServletContext application = null;
39     ServletConfig config = null;
40     JspWriter out = null;
41     Object page = this;
42     JspWriter _jspx_out = null;
43     PageContext _jspx_page_context = null;
44
```

J2EE API 문서

<https://docs.oracle.com/javaee/7/index.html>

5

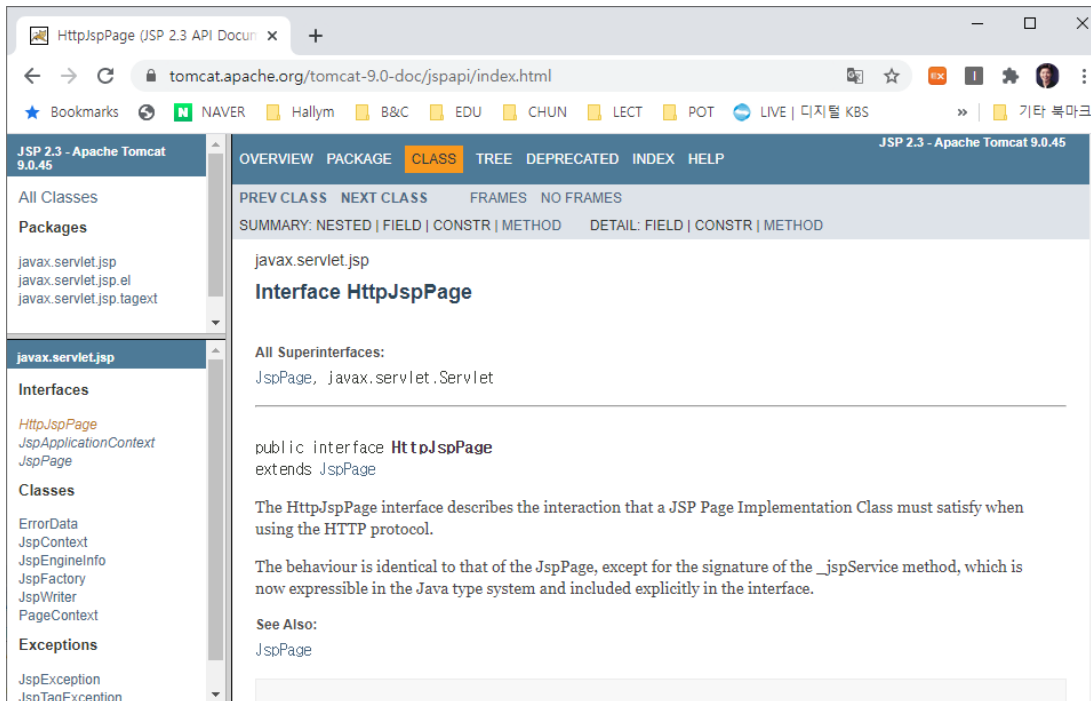


톰캣 엔진의 JSP API 문서

6

□ 웹 사이트

- <http://tomcat.apache.org> 에 접속
- 왼쪽 메뉴 [Documentation]에서 원하는 버전 선택
- 다시 왼쪽 메뉴 [Reference] 에서 [JSP 2.3 Javadocs] 선택

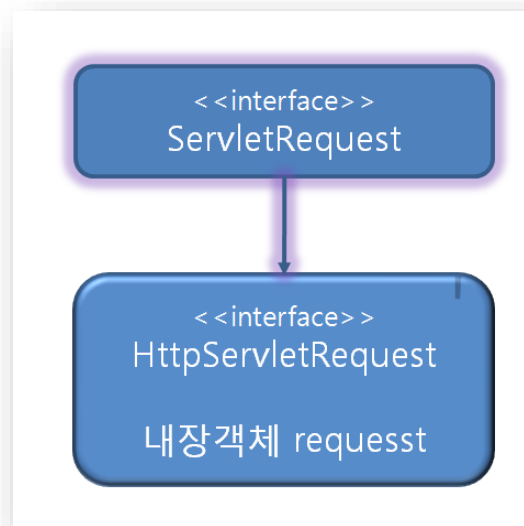


내장 객체 request

7

□ 인터페이스 HttpServletRequest

- 내장 객체 **request**는 클라이언트가 서버에게 전송하는 관련 정보를 처리하는 객체
- 즉 **HTML** 폼에 입력하여 값을 전달하는 경우
- 인터페이스 `javax.servlet.ServletRequest`가 상위 인터페이스



내장 객체 request의 주요 메소드(1)

8

□ 인터페이스 javax.servlet.ServletRequest의 메소드

반환값	메소드	사용 용도
void	setCharacterEncoding(String env)	요청 페이지에 env의 인코딩 방법을 적용
String	getParameter(String name)	name의 요청 인자 값을 반환, 없으면 null을 반환, 만일 값이 여러 개이면 첫 번째 값만 반환
String[]	getParameterValues(String name)	지정한 name의 요청 인자 값을 문자열 배열로 반환, 없으면 null을 반환
Enumeration	getParameterNames()	모든 인자의 이름을 Enumeration으로 반환
String	getProtocol()	사용중인 프로토콜을 반환
String	getRemoteAddr()	클라이언트의 IP 주소를 반환
String	getRemoteHost()	클라이언트의 호스트 이름을 반환
String	getServerName()	요청된 서버의 호스트 이름을 반환
int	getServerPort()	요청된 서버의 포트 번호를 반환

HTML 폼 정보의 전달

9

□ request.html

```
1<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>예제 request</title>
6</head>
7<body>
8
9<h2> 학생 정보 입력</h2>
10
11<form method="post" action="request.jsp">
12
13    성명 : <input type="text" name="name"><p>
14    학번 : <input type="text" name="studentNum"><p>
15    성별 : 남자 <input type="radio" name="sex" value="man" checked>
16           여자 <input type="radio" name="sex" value="woman"><p>
17    국적 : <select name="country">
18           <option SELECTED value="대한민국">대한민국</option>
19           <option value="일본">일본</option>
20           <option value="중국">중국</option>
21           <option value="터키">터키</option>
22           <option value="태국">태국</option>
23       </select><p>
24
25    <input type="submit" value="보내기">
26</form>
27
28</body>
29</html>
```

[실습1]

http://localhost:8080/eun01/implicit_object/request.html

학생 정보 입력

성명 :

학번 :

성별 : 남자 ☒ 여자 ☐

국적 :

HTML 폼 정보의 전달 처리

10

□ request.jsp

```
request.html request.jsp
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 request.jsp</title>
6</head>
7<body>
8
9<%
10    request.setCharacterEncoding("euc-kr");
11%>
12
13<%
14    String name = request.getParameter("name");
15    String studentNum = request.getParameter("studentNum");
16    String sex = request.getParameter("sex");
17    String country = request.getParameter("country");
18
19    if (sex.equalsIgnoreCase("man")) {
20        sex = "남자";
21    } else {
22        sex = "여자";
23    }
24%>
25
26<h2> 학생 정보 입력 결과</h2>
27
28성명 : <%= name%><p>
29학번 : <%= studentNum%><p>
30성별 : <%= sex%><p>
31국적 : <%= country%><p>
32
33</body>
34</html>
```

예제 request

http://localhost:8080/ch05/request.html

학생 정보 입력

성명 :

학번 :

성별 : 남자 ☐ 여자 ☒

국적 :

JSP 예제 request.jsp

http://localhost:8080/ch05/request.jsp

학생 정보 입력 결과

성명 : 홍길동

학번 : 2010-1289

성별 : 여자

국적 : 대한민국

내장 객체 request의 주요 메소드(2)

11

반환값	메소드	사용 용도
Cookie[]	getCookies()	클라이언트에 보내진 쿠키 배열을 반환
String	getQueryString()	URL에 추가된 Query 문자열을 반환
String	getRequestURI()	클라이언트가 요청한 URI 반환, URI는 프로토콜, 서버이름, 포트번호를 제외한 서버의 컨텍스트와 파일의 문자열
String	getRequestURL()	클라이언트가 요청한 URL 반환, URL은 프로토콜과 함께 주소 부분에 기술된 모든 문자열
HttpSession	getSession()	현재의 세션을 반환, 세션이 없으면 새로 만들어 반환
String	getMethod()	요청 방식인 get, post 중의 하나를 반환

태그 select 처리(1)

12

□ request2.html

The image shows a web browser window with two tabs: 'request2.html' and '예제 request'. The 'request2.html' tab is active, displaying the source code of the HTML file. The code is as follows:

```
1<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>예제 request2</title>
6</head>
7<body>
8
9<h2> 학생 정보 입력</h2>
10
11<form method="post" action="request2.jsp">
12
13     학번 : <input type="text" name="studentNum"><p>
14     전공 : <select multiple name="major">
15         <option SELECTED value="전산과">전산과 </option>
16         <option value="국문과">국문과 </option>
17         <option value="기계과">기계 </option>
18         <option value="자유전공과">자유전공과 </option>
19         <option value="경영학과">경영학과 </option>
20     </select><p>
21
22     <input type="submit" value="보내기">
23 </form>
24
25</body>
26</html>
```

On the right side of the browser window, the rendered HTML is displayed. It shows the title '[실습2]' and the heading '학생 정보 입력'. Below the heading, there is a form with the following elements:

- A text input field labeled '학번 : '.
- A multiple-select dropdown menu labeled '전공 : ' with the following options: '전산과' (selected), '국문과', '기계', '자유전공과', and '경영학과'.
- A submit button labeled '보내기'.

The browser's address bar shows the URL 'http://localhost:8080/eun01/implicit_c'.

태그 select 처리(2)

13

□ request2.jsp

← → ↻ 📄 http://localhost:8080/

학생 정보 입력

학번 :

전공 :

전산과
국문과
기계
자유전공과

← → ↻ 📄 http://localhost:8080/eun01/implicit_object/request2.jsp

학생 정보 입력 결과

학번 : 2020-1234

전공 : 전산과 자유전공과

요청 정보

요청 방식 : POST

요청 URL : http://localhost:8080/eun01/implicit_object/request2.jsp

요청 URI : /eun01/implicit_object/request2.jsp

클라이언트 주소 : 0:0:0:0:0:0:1

클라이언트 호스트 : 0:0:0:0:0:0:1

프로토콜 방식 : HTTP/1.1

서버 이름 : localhost

서버 포트 번호 : 8080

```
request2.jsp
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 request2.jsp</title>
6</head>
7<body>
8
9<%
10    request.setCharacterEncoding("euc-kr");
11%>
12
13<%
14    String studentNum = request.getParameter("studentNum");
15    String[] majors = request.getParameterValues("major");
16%>
17
18<h2> 학생 정보 입력 결과</h2>
19
20학번 : <%= studentNum%><p>
21전공 : <%
22    if (majors == null) {
23        out.println("전공 없음.");
24    } else {
25        for (int i=0; i < majors.length; i++)
26            out.println(majors[i] + " ");
27
28        //JDK 1.5 이후부터 다음 코딩 가능
29        //for ( String eachmajor : majors )
30            //out.println(eachmajor + " ");
31    }
32    %>
33
34<h2> 요청 정보</h2>
35요청 방식 : <%= request.getMethod() %><p>
36요청 URL : <%= request.getRequestURL() %><p>
37요청 URI : <%= request.getRequestURI() %><p>
38클라이언트 주소 : <%= request.getRemoteAddr() %><p>
39클라이언트 호스트 : <%= request.getRemoteHost() %><p>
40프로토콜 방식 : <%= request.getProtocol() %><p>
41서버 이름 : <%= request.getServerName() %><p>
42서버 포트 번호 : <%= request.getServerPort() %><p>
43
44</body>
45</html>
```

for each 문장

14

- 메소드 `request.getParameterValues("major")`
 - ▣ 반환 값이 문자열 배열
 - ▣ 선택된 전공이 없다면
 - 메소드 `request.getParameterValues("major")`는 `null` 값을 반환
 - 변수 `majors`에는 `null` 값이 저장
 - ▣ `<% String[] majors = request.getParameterValues("major"); %>`
- 문자열 배열 변수 `majors`를 브라우저에 출력
 - `for (int i=0; i < majors.length; i++)`
 - `out.println(majors[i] + " ");`
- JDK 1.5(5.0) 이후, `for each` 문장으로도 가능
 - `for (String eachmajor : majors)`
 - `out.println(eachmajor + " ");`

메소드 getParameterNames()

15

□ 반환 값이 Enumeration 유형

▣ 요청 페이지의 모든 인자 이름 목록을 반환

JSP 예제 request3.jsp request3.html

```
1<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>예제 request3</title>
6</head>
7<body>
8
9<h2> 취미와 가보고 싶은 국가</h2>
10
11<form method="post" action="request3.jsp">
12
13    1. 좋아하는 취미를 선택하십시오. <p>
14    영화 <input type="checkbox" name="hobby" value="영화"><br>
15    독서 <input type="checkbox" name="hobby" value="독서"><br>
16    스키 <input type="checkbox" name="hobby" value="스키"><br>
17    자전거 <input type="checkbox" name="hobby" value="자전거"> <p> <hr>
18
19    2. 여행하고 싶은 국가를 하나 선택하십시오. <p>
20    영국 <input type="radio" name="country" value="영국" checked><br>
21    미국 <input type="radio" name="country" value="미국"><br>
22    브라질 <input type="radio" name="country" value="브라질"><br>
23    터키 <input type="radio" name="country" value="터키"> <p>
24
25    <input type="submit" value="보내기">
26</form>
27
28</body>
29</html>
```

[실습3]

http://localhost:8080/eun01/implicit

취미와 가보고 싶은 국가

1. 좋아하는 취미를 선택하십시오.

영화 ☐
독서 ☐
스키 ☐
자전거 ☐

2. 여행하고 싶은 국가를 하나 선택하십시오.

영국 ☒
미국 ☐
브라질 ☐
터키 ☐

반환 유형 Enumeration 처리 방법

16

- 메소드 `getParameterNames()`를 이용
 - 반환 유형 `java.util.Enumeration`

```
JSP 예제 request3.jsp request3.html request3.jsp x
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="utf-8"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 request3.jsp</title>
6</head>
7<body>
8
9<%@ page import="java.util.Enumeration" %>
10<% request.setCharacterEncoding("euc-kr"); %>
11
12<h2> 취미와 가보고 싶은 국가 결과</h2>
13
14<%
15 //Enumeration e = request.getParameterNames();
16 Enumeration<String> e = request.getParameterNames();
17
18 while ( e.hasMoreElements() ) {
19     //String name = (String) e.nextElement();
20     String name = e.nextElement();
21     String [] data = request.getParameterValues(name);
22     if (data != null) {
23         for ( String eachdata : data )
24             out.println(eachdata + " ");
25     }
26     out.println("<p>");
27 }
28%>
29
30</body>
31</html>
```

http://localhost:8080/eun01/implicit_

취미와 가보고 싶은 국가

1. 좋아하는 취미를 선택하시오.

영화 ☒
독서 ☐
스키 ☒
자전거 ☒

2. 여행하고 싶은 국가를 하나 선택하시오.

영국 ☐
미국 ☐
브라질 ☐
터키 ☒

보내기

http://localhost:8080/eun01/implicit_object/req

취미와 가보고 싶은 국가 결과

영화 스키 자전거
터키

전송방식 post의 한글 처리(1)

17

□ post 방식

- 전송 자료 크기의 제한 없이 사용자가 입력한 내용을 공개하지 않고 전송하는 방식
- JSP 파일에서 내장객체 request를 사용하기 이전에
 - 메소드 `request.setCharacterEncoding("euc-kr")`을 호출
- 예제 `postrequest.html`

```
1<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>예제 postrequest</title>
6</head>
7<body>
8
9<h2> 메소드 post 방식에서 한글 처리</h2>
10
11<form method="post" action="postrequest.jsp">
12    한글 성명 : <input type="text" name="korname"><p>
13    영문 성명 : <input type="text" name="engname"><p>
14    <input type="submit" value="보내기"/>
15</form>
16
17</body>
18</html>
```

[실습4]

메소드 post 방식에서 한글 처리

한글 성명 :

영문 성명 :

http://localhost:8080/eun01/implicit_object/post

전송방식 post의 한글 처리(2)

18

□ postrequest.jsp

```
postrequest.html  postrequest.jsp x
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 postrequest.jsp</title>
6</head>
7<body>
8
9<% request.setCharacterEncoding("euc-kr"); %>
10
11<h2> 메소드 post 방식에서 한글 처리</h2>
12<hr>
13한글 성명 : <%= request.getParameter("korname") %><p>
14영문 성명 : <%= request.getParameter("engname") %><p>
15
16</body>
17</html>
```

http://localhost:8080/eun01/implicit_object/c

메소드 post 방식에서 한글 처리

한글 성명 : 홍길동

영문 성명 : Hong Gil Dong x

보내기



http://localhost:8080/eun01/implicit_object/po

메소드 post 방식에서 한글 처리

한글 성명 : 홍길동

영문 성명 : Hong Gil Dong

전송방식 get의 한글 처리

19

□ 폼 양식 get 전송 방식

- ▣ post와는 달리 전송 자료 크기의 제한이 있으며
- ▣ 사용자가 입력한 내용을 공개하여 전송하는 방식

□ 한글 처리

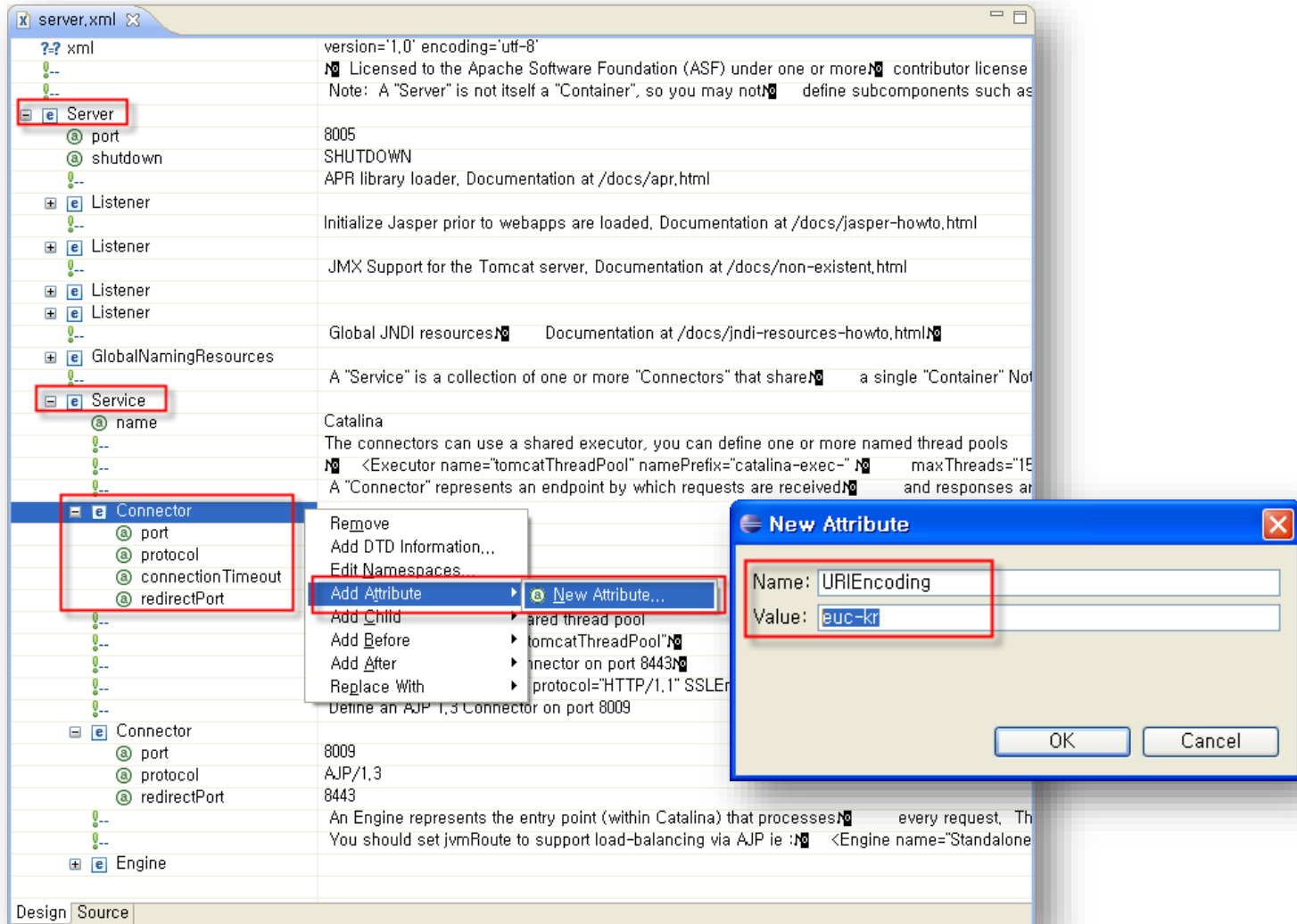
- ▣ 수정 파일
 - [톰캣 설치 폴더]/[conf]/server.xml 파일
- ▣ <connector port="8080" ... />에서
 - 속성 [URIEncoding="euc-kr"]을 추가

```
<Connector port="8080" protocol="HTTP/1.1"  
    connectionTimeout="20000"  
    redirectPort="8443" URIEncoding="euc-kr"> </Connector>
```

이클립스에서 server.xml 편집(1)

20

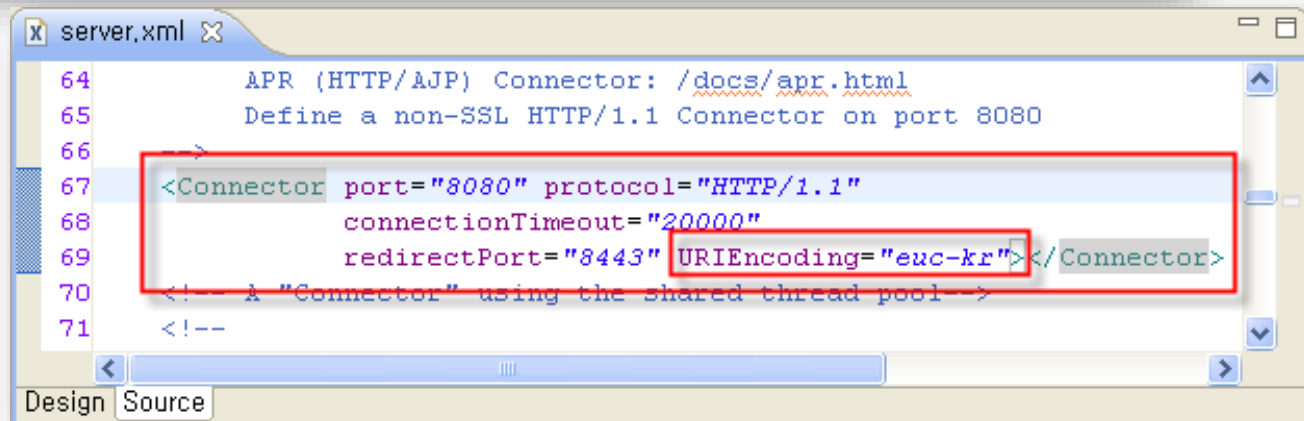
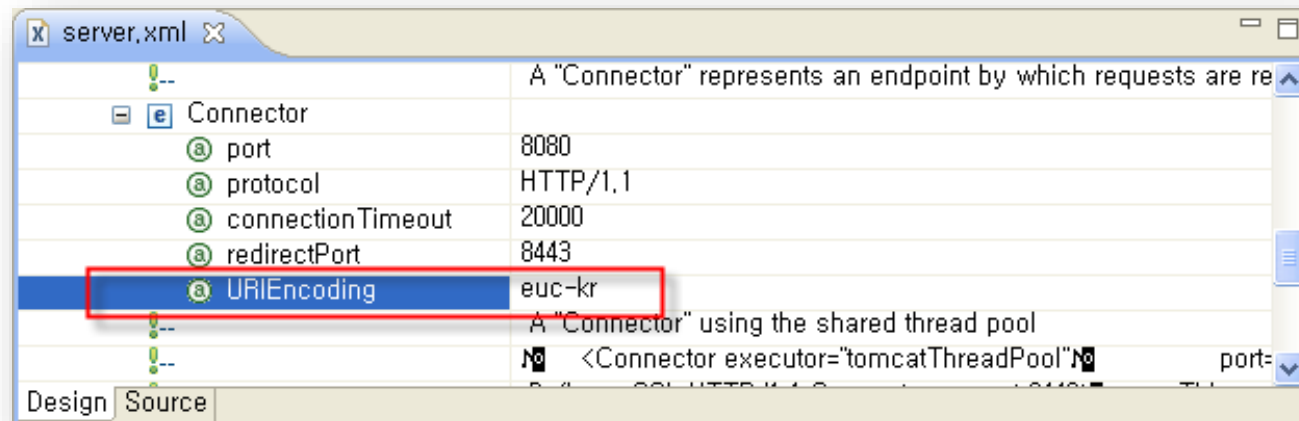
- 메뉴 [File -> open file... -> server.xml파일 선택]



이클립스에서 server.xml 편집(2)

21

- URIEncoding="euc-kr" 혹은 "utf-8" 추가



전송방식 get의 한글 처리 예제 (1)

22

□ getrequest.html

```
1<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>예제 getrequest</title>
6</head>
7<body>
8
9<h2> 메소드 get 방식에서 한글 처리</h2>
10
11<form method="get" action="getrequest.jsp">
12    한글 성명 : <input type="text" name="korname"><p>
13    영문 성명 : <input type="text" name="engname"><p>
14    <input type="submit" value="보내기">
15</form>
16
17</body>
18</html>
```

[실습5]

메소드 get 방식에서 한글 처리

한글 성명 :

영문 성명 :

전송방식 get의 한글 처리 예제 (2)

23

□ getrequest.jsp

The image displays the source code of a JSP file named `getrequest.jsp` and its rendered output in a web browser.

Source Code (getrequest.jsp):

```
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 getrequest.jsp</title>
6</head>
7<body>
8
9<h2> 메소드 get 방식에서 한글 처리</h2>
10<hr>
11한글 성명 : <%= request.getParameter("korname") %><p>
12영문 성명 : <%= request.getParameter("engname") %><p>
13
14</body>
15</html>
```

Browser Output (JSP 예제 getrequest.jsp):

The browser displays the rendered HTML output. The title is "JSP 예제 getrequest.jsp". The main heading is "메소드 get 방식에서 한글 처리". Below the heading, there are two input fields for "한글 성명" (Korean Name) and "영문 성명" (English Name). The "한글 성명" field contains the text "홍길동" (Hong Gil Dong). The "영문 성명" field contains the text "Hong Gil Dong". A "보내기" (Send) button is located below the input fields.

The browser's address bar shows the URL: `http://localhost:8080/ch05/getrequest.jsp?korname=%C8%AB%B1%E6%B5%BF&engname=Hong+Gil+Dong`. The rendered output shows the Korean name as "한글 성명 : 홍길동" and the English name as "영문 성명 : Hong Gil Dong".

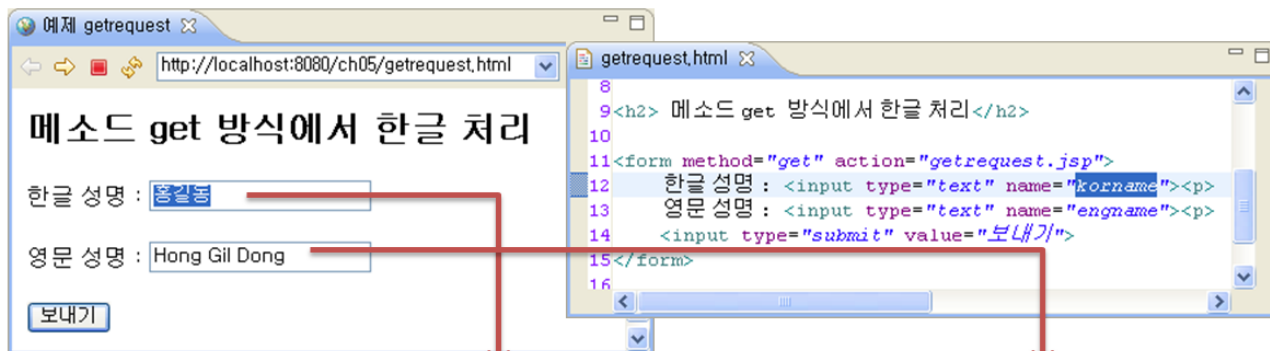
질의문자열(query string)

24

- 메소드 get 방식에서
 - ▣ URL 부분에 전송 자료
 - [name1=값1&name2=값2] 형식으로 추가

전체 URL

`http://localhost:8080/ch05/getrequest.jsp?korname=%C8%AB%B1%E6%B5%BF&engname=Hong+Gil+Dong`



질의문자열

`korname=%C8%AB%B1%E6%B5%BF&engname=Hong+Gil+Dong`

질의문자열 형식

`name1=value1&name2=value2`

내장 객체 response

25

□ 인터페이스 HttpServletResponse

- ▣ 서버가 클라이언트에게 요청에 대한 응답을 보내기 위한 객체
- ▣ 인터페이스 HttpServletResponse
 - 상위 인터페이스가 **ServletResponse**
- ▣ 메소드 **sendRedirect()**
 - 원하는 페이지로 이동

<%

String URL = "http://www.naver.com ";

response.sendRedirect(URL);

%>

반환값	메소드	사용 용도
void	addCookie(Cookie cookie)	쿠키 데이터 기록
void	addHeader(String name, String value)	response 헤더 내용 기록
void	sendRedirect(String location)	지정된 location 페이지로 이동
void	setBufferSize(int size)	버퍼 크기 지정
void	setContentType(String type)	Content Type 지정
int	getBufferSize(int size)	버퍼 크기 반환

메소드 sendRedirect()

26

- response의 sendRedirect()의 메소드를 이용한 검색 기능

[실습6]

```
1<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/strict.dtd">
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>예제 sendredirect</title>
6</head>
7<body>
8
9<h2> 검색할 단어를 입력하세요.</h2>
10
11<form method="get" action="sendredirect.jsp">
12  검색 키워드 : <input type="text" name="word"><p>
13  <input type="submit" value="보내기">
14  <input type="reset" value="취소">
15</form>
16
17</body>
18</html>
```

```
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 sendredirect.jsp</title>
6</head>
7<body>
8
9<%
10  String URL = "http://search.naver.com/search.naver?where=nexearch";
11  String keyword = request.getParameter("word");
12  URL += "&" + "query=" + keyword;
13  response.sendRedirect (URL);
14%>
15
16</body>
17</html>
```

예제 sendredirect

http://localhost:8080/ch05/sendredirect.html

검색할 단어를 입력하세요.

검색 키워드 : JSP

보내기 취소

NAVER JSP 검색

통합검색 사이트 일문서 지식IN 블로그 카페 이미지 동영상 사진 뉴스 더보기

스폰서링크

- JSP취업 종로아이티뱅크 - JSP, JAVA, JSP, EJB, 자바취업 100% 지원과정 JSP, 자바실무자. <http://www.it-sesang.com>
- 더존그룹웨어 ASP서비스 - 더존그룹웨어 ASP서비스, 사내메신저, 기업메일 호스팅, 전자결재, 웹하드. <http://www.duzongroupware.com>
- 나누미넷 JSP호스팅 전문 - JSP호스팅 7년차 업체, 톨넷, 레진, MYSQL, MSSQL2005 지원. <http://www.nanuminet.co.kr>
- 아이티원 JSP - JSP, 재직자30만원환급, 학생20%할인, 무료재수강, JSP취업, 주말반. <http://www.itclick.or.kr>
- 아이티뱅크자바교육지정홈페이지 - jsp학원중 가장저렴하고 결과가 좋은곳, 대학교수 추천 학원 100%취업보장. <http://www.itzang.co.kr>

파워링크

1. 강남아이티정보처리학원 강건 - JSP전문교육, SCWCD취득, EJB, Ajax, CBD, MVC, 취업과정. <http://www.celli4web.com>
2. 자바 한솔멀티캠퍼스 - 자바전문교육기관, DB, JAVA, JSP, 취업추천, 독석역 5번출구. <http://www.hsmc.net>
3. 중앙정보처리학원 신철우 - 자바전문학원, 자바6단계코스진행, 과목별실무교육, 체계적인교육, 자바개발자양성교육 <http://www.lesson-web.com>
4. ITBANK멀티캠퍼스 이세미 - JAVA, JSP, EJB, 실무 단계학습, SUN 자격증, 고용보험 70%환급.

내장 객체 out

27

□ 클래스 JspWriter

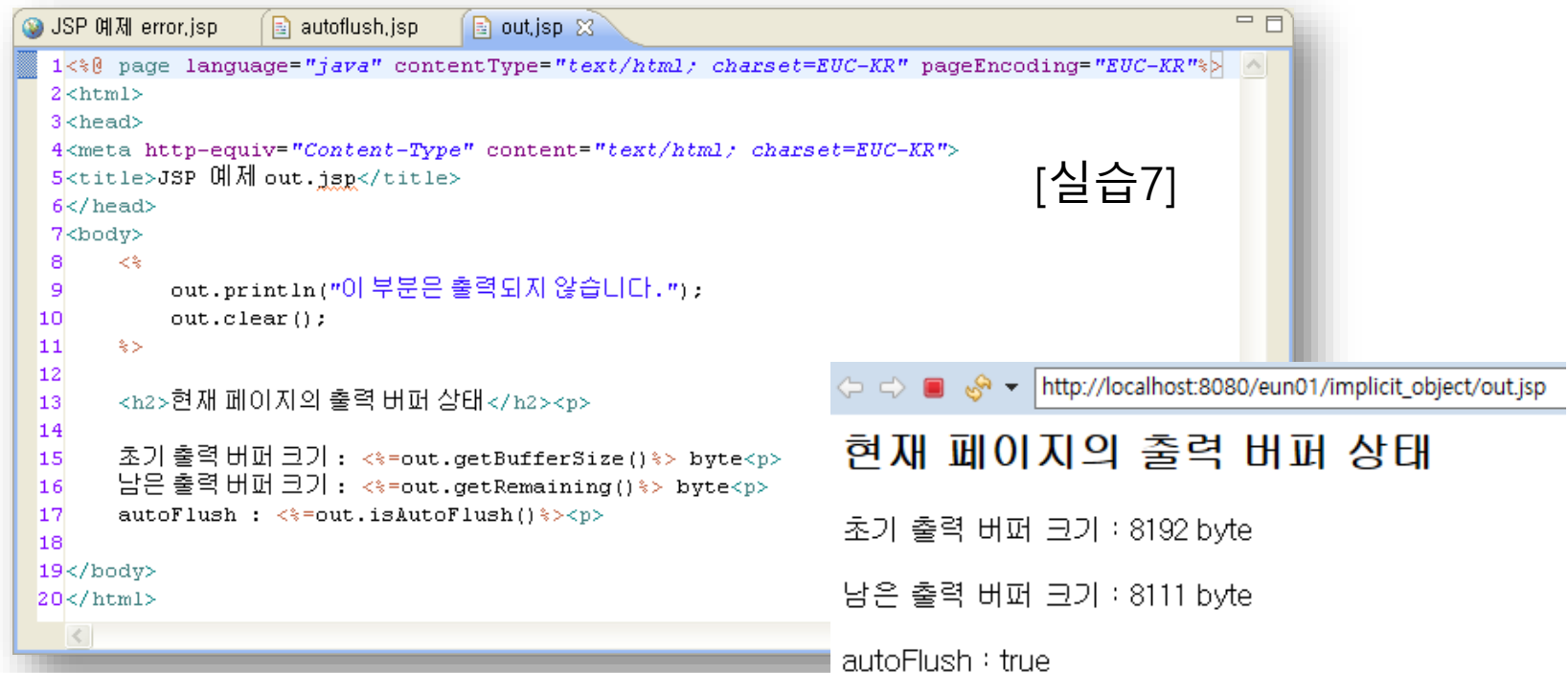
- 클래스 `javax.servlet.jsp.JspWriter` 자료유형
- JSP 페이지의 출력을 위한 객체
- 출력과 버퍼링에 관련된 주요 메소드를 제공

반환값	메소드	사용 용도
void	<code>print(여러 자료 값)</code>	여러 자료유형을 출력
void	<code>println(여러 자료 값)</code>	여러 자료유형을 출력하고 현재 줄을 종료
void	<code>clearBuffer()</code>	버퍼의 현재 내용물을 제거
void	<code>flush()</code>	버퍼의 내용물 비우기
void	<code>clear()</code>	버퍼의 내용물을 제거
void	<code>close()</code>	스트림을 닫음
int	<code>getBufferSize()</code>	버퍼의 전체 크기를 반환
int	<code>getRemaining()</code>	버퍼의 남아 있는 크기를 반환
boolean	<code>isAutoFlush()</code>	현재 <code>autoFlush</code> 상태를 반환

내부 객체 out의 메소드 clear()

28

- 버퍼의 상태 점검
 - 메소드 getBufferSize()
 - getRemaining()
 - isAutoFlush()를 이용



The image shows a web browser window displaying the output of a JSP page. The browser's address bar shows the URL `http://localhost:8080/eun01/implicit_object/out.jsp`. The page content displays the current state of the output buffer. To the left, a code editor shows the JSP source code for `out.jsp`.

[실습7]

현재 페이지의 출력 버퍼 상태

초기 출력 버퍼 크기 : 8192 byte
남은 출력 버퍼 크기 : 8111 byte
autoFlush : true

```
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 out.jsp</title>
6</head>
7<body>
8    <%
9        out.println("이 부분은 출력되지 않습니다.");
10       out.clear();
11    %>
12
13    <h2>현재 페이지의 출력 버퍼 상태</h2><p>
14
15    초기 출력 버퍼 크기 : <%=out.getBufferSize()%> byte<p>
16    남은 출력 버퍼 크기 : <%=out.getRemaining()%> byte<p>
17    autoFlush : <%=out.isAutoFlush()%><p>
18
19</body>
20</html>
```

버퍼링

29

- 페이지 지시자에서 속성 `autoFlush`가 `false`
 - ▣ 버퍼가 가득 차기 전에 `flush()`를 호출하여 수동 출력
 - `flush`하기 전에 버퍼가 가득 차면 버퍼 오버플로(buffer overflow) 오류가 발생

```
JSP 예제 autoflush.jsp autoflush.jsp
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 autoflush.jsp</title>
6</head>
7<body>
8  <%@ page autoFlush="false" buffer="1kb" %>
9  <h2>현재 autoFlush = <%=out.isAutoFlush() %></h2><p>
11
12  <%
13    for (int i = 1; i < 25; i++) {
14      out.println("남은 출력 버퍼 크기 (out.getRemaining()) : " + out.getRemaining() + "<br>");
15      //autoFlush가 false이면 알아서 버퍼를 출력해야 한다.
16      if (out.getRemaining() < 50) {
17        out.println("<br>");
18        out.flush();
19      }
20    }
21  %>
22
23  <hr>
24  초기 출력 버퍼 크기 : <%=out.getBufferSize() %> byte<br>
25  남은 출력 버퍼 크기 : <%=out.getRemaining() %> byte
26</body>
27</html>
```

[실습8]

```
JSP 예제 autoflush.jsp
http://localhost:8080/ch05/autoflush.jsp
현재 autoFlush = false

남은 출력 버퍼 크기(out.getRemaining()) : 841
남은 출력 버퍼 크기(out.getRemaining()) : 798
남은 출력 버퍼 크기(out.getRemaining()) : 755
남은 출력 버퍼 크기(out.getRemaining()) : 712
남은 출력 버퍼 크기(out.getRemaining()) : 669
남은 출력 버퍼 크기(out.getRemaining()) : 626
남은 출력 버퍼 크기(out.getRemaining()) : 583
남은 출력 버퍼 크기(out.getRemaining()) : 540
남은 출력 버퍼 크기(out.getRemaining()) : 497
남은 출력 버퍼 크기(out.getRemaining()) : 454
남은 출력 버퍼 크기(out.getRemaining()) : 411
남은 출력 버퍼 크기(out.getRemaining()) : 368
남은 출력 버퍼 크기(out.getRemaining()) : 325
남은 출력 버퍼 크기(out.getRemaining()) : 282
남은 출력 버퍼 크기(out.getRemaining()) : 239
남은 출력 버퍼 크기(out.getRemaining()) : 196
남은 출력 버퍼 크기(out.getRemaining()) : 153
남은 출력 버퍼 크기(out.getRemaining()) : 110
남은 출력 버퍼 크기(out.getRemaining()) : 67

if (out.getRemaining() < 50) {
    out.println("<br>");
    out.flush();
}

위 조건문이 만족하여 실행된 부분으로
<br>이 출력되어 한 줄을 띄고 출력된
다.

남은 출력 버퍼 크기(out.getRemaining()) : 1024
남은 출력 버퍼 크기(out.getRemaining()) : 980
남은 출력 버퍼 크기(out.getRemaining()) : 937
남은 출력 버퍼 크기(out.getRemaining()) : 894
남은 출력 버퍼 크기(out.getRemaining()) : 851

초기 출력 버퍼 크기 : 1024 byte
남은 출력 버퍼 크기 : 752 byte
```

내장 객체 application

30

- 인터페이스 ServletContext
 - ▣ javax.servlet.ServletContext 인터페이스
 - ▣ 웹 애플리케이션에서 유지 관리되는 여러 환경 정보를 관리
- 웹 애플리케이션
 - ▣ 여러 개의 서블릿과 **JSP**로 구성되는 웹 서비스 응용 프로그램 단위
 - ▣ 내장 객체 **application**은 서블릿과 서버 간의 자료를 교환하는 여러 메소드를 제공

반환값	메소드	사용 용도
String	getServerInfo()	JSP 컨테이너의 이름과 버전 반환
Object	getAttribute(String name)	웹 응용에서 지정된 이름의 속성을 반환
void	log(String msg)	지정된 msg의 로그를 저장
void	setAttribute(String name, Object object)	웹 응용에서 지정된 이름으로 object를 저장
void	removeAttribute(String name)	웹 응용에서 지정된 이름의 속성을 삭제

웹 응용 프로그램에서 조회 수 관리

31

□ 메소드 setAttribute(), getAttribute()

[실습9]

```
JSP 예제 application.jsp application.jsp
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 application.jsp</title>
6</head>
7<body>
8
9<%= int count = 0; %>
10
11<%
12    String scount = (String) application.getAttribute("count");
13
14    if (scount != null) {
15        count = Integer.parseInt(scount);
16    } else {
17        count = 0;
18    }
19
20    application.setAttribute("count", Integer.toString(++count));
21    application.log("현재까지 조회 수 : " + count);
22%>
23    서버 컨테이너 정보 : <%=application.getServerInfo() %> <p>
24    현재까지 조회 수 : <%=count %>
25
26</body>
27</html>
```

http://localhost:8080/eun01/implicit_object/

서버 컨테이너 정보 : Apache Tomcat/9.0.44

현재까지 조회 수 : 1

http://localhost:8080/eun01/implicit_object/

서버 컨테이너 정보 : Apache Tomcat/9.0.44

현재까지 조회 수 : 6

Markers Properties Servers Data Source Explorer Snippets Console

tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jre1.8.0_201\bin\j...
4월 08, 2021 8:22:13 오후 org.apache.coyote.AbstractProtocol start
정보: 프록트콜 핸들러 ["http-nio-8080"]을(를) 시작합니다.
4월 08, 2021 8:22:13 오후 org.apache.catalina.startup.Catalina start
정보: 서버가 [251] 밀리초 내에 시작되었습니다.
4월 08, 2021 8:32:31 오후 org.apache.catalina.core.ApplicationContext log
정보: 현재까지 조회 수 : 1
4월 08, 2021 8:32:52 오후 org.apache.catalina.core.ApplicationContext log
정보: 현재까지 조회 수 : 2
4월 08, 2021 8:32:54 오후 org.apache.catalina.core.ApplicationContext log
정보: 현재까지 조회 수 : 3
4월 08, 2021 8:32:56 오후 org.apache.catalina.core.ApplicationContext log
정보: 현재까지 조회 수 : 4
4월 08, 2021 8:32:57 오후 org.apache.catalina.core.ApplicationContext log
정보: 현재까지 조회 수 : 5
4월 08, 2021 8:33:06 오후 org.apache.catalina.core.ApplicationContext log
정보: 현재까지 조회 수 : 6

내장 객체 exception

32

□ 페이지 지시자에서

- ▣ isErrorPage="true"로 지정한 경우, 이용할 수 있는 내부 객체
- ▣ 지정한 예외 처리 페이지에서 적절한 예외 처리를 구현

반환값	메소드	사용 용도
String	getMessage()	예외를 표시하는 문자열을 반환
String	toString()	예외 자체를 문자열을 반환
void	printStackTrace()	표준 출력으로 스택 추적 정보 출력

버퍼 오버플로 오류 처리

33

□ isErrorPage="true"

```
error.jsp
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 bufferoverflow.jsp</title>
6</head>
7<body>
8    <%@ page autoFlush="false" buffer="1kb" errorPage="error.jsp" %>
9
10    <%
11        for (int i = 1; i < 25; i++) {
12            out.println("남은 출력 버퍼 크기 (out.getRemaining()) : " + out.getRemaining() + "<br>");
13        }
14    %>
15</body>
16</html>
```

[실습10]

```
error.jsp
1<%@ page language="java" contentType="text/html; charset=EUC-KR" pageEncoding="EUC-KR"%>
2<html>
3<head>
4<meta http-equiv="Content-Type" content="text/html; charset=EUC-KR">
5<title>JSP 예제 error.jsp</title>
6</head>
7<body>
8
9    <%@ page isErrorPage="true" %>
10    <h1>예외처리 페이지</h1>
11
12    오류 문자열[exception.toString()] : <%=exception.toString() %><br>
13    오류 메시지[exception.getMessage()] : <%=exception.getMessage() %><br>
14
15</body>
16</html>
```



내장 객체 pageContext

34

□ 클래스 pageContext

- ▣ 자료유형 클래스 javax.servlet.jsp.PageContext
- ▣ JSP 페이지에 관한 정보와 다른 페이지로 제어권을 넘겨줄 때 이
용되는 메소드를 제공

반환값	메소드	사용 용도
void	forward(String)	다른 서블릿 혹은 JSP로 요청을 이동
void	include(String)	지정된 페이지를 현재의 위치에 삽입
Exception	getException()	Exception 객체를 반환
Object	getPage()	page 객체를 반환
JspWriter	getOut()	JspWriter 객체를 반환
ServletRequest	getRequest()	ServletRequest 객체를 반환
ServletResponse	getResponse()	ServletResponse 객체를 반환
ServletConfig	getServletConfig()	ServletConfig 객체를 반환
ServletContext	getServletContext()	ServletContext 객체를 반환
HttpSession	getSession()	HttpSession 객체를 반환
Object	findAttribute(String)	page, request, session, application 범위 내에서 사용 가능한 속성의 값을 반환
void	removeAttribute(String)	지정한 이름의 속성 객체를 제거
Object	getAttribute(String)	page 범위 내에서 특정한 이름에 해당하는 속성 객체를 반환
void	setAttribute(String, Object)	pageContext 객체 안에 지정한 이름과 연관된 속성 객체를 저장

다른 내부 객체를 참조 : 메소드 제공

35

□ 내장 객체 pageContext

- ▣ 8개의 다른 내부 객체를 얻을 수 있는 메소드를 제공

```
28
29 public void _jspService(HttpServletRequest request, HttpServletResponse response)
30     throws java.io.IOException, ServletException {
31
32     PageContext pageContext = null;
33     HttpSession session = null;
34     ServletContext application = null;
35     ServletConfig config = null;
36     JspWriter out = null;
37     Object page = this;
38     JspWriter _jspx_out = null;
39     PageContext _jspx_page_context = null;
40
41
42     try {
43         response.setContentType("text/html; charset=EUC-KR");
44         pageContext = _jspxFactory.getPageContext(this, request, response,
45             null, true, 1024, false);
46         _jspx_page_context = pageContext;
47         application = pageContext.getServletContext();
48         config = pageContext.getServletConfig();
49         session = pageContext.getSession();
50         out = pageContext.getOut();
51         _jspx_out = out;
52
53         out.write("\r\n");
54         out.write("<html>\r\n");
```

메소드 include()

36

□ 외부 파일 삽입

The image displays three overlapping screenshots from an IDE and a web browser, illustrating the use of the `include()` method in JSP.

Top Screenshot (JSP Editor): Shows the code for `pagecontext.jsp`. The code includes a page directive, HTML boilerplate, and a body section. The body contains an `<h2>` tag, a JSP expression `<%= pageContext.getOut().println("include.html을 추가"); %>`, an `<hr>` tag, and an `<%= pageContext.include("include.html"); %>` statement. Line numbers 1 through 15 are visible on the left.

Middle Screenshot (JSP Editor): Shows the code for `include.html`. It contains a blue font tag ``, followed by the text "다른 파일을 삽입하는 include(), 제어권을 넘기는 forward() 메소드 제공", and a closing font tag ``. Line numbers 1 through 3 are visible on the left.

Bottom Screenshot (Web Browser): Shows the rendered output of the JSP page at `http://localhost:8080/ch05/pagecontext.jsp`. The page title is "pageContext 예제". The output displays the text "include.html을 추가" followed by a horizontal line and the text "다른 파일을 삽입하는 include(), 제어권을 넘기는 forward() 메소드 제공". The text "include.html을 추가" is rendered in blue, matching the font color in the `include.html` file.

[실습11]

내장 객체 page

37

- JSP 페이지 자체를 표현
 - ▣ 내장 객체 **page**는 JSP 페이지 자체를 나타내는 객체
 - Object **page = this;**
 - ▣ 자바에서 자기 자신을 나타내는 키워드 **this**로 사용
 - ▣ **this**는 자료유형 **org.apache.jasper.runtime.HttpJspBase**의 객체
 - ▣ 메소드 **getServletInfo()**를 제공
 - JSP 페이지 지시자의 속성 **info**에 지정한 값을 반환



내장 객체 session

38

□ 세션 관리를 위한 내부 객체

- ▣ 인터넷 쇼핑몰에서 상품을 구매하는 경우
 - 장바구니를 생각
 - 장바구니 페이지는 다른 페이지를 이동하더라도 현재 선택된 상품 목록과 관련 정보를 지속적으로 유지 관리
 - 이렇게 클라이언트 사용자의 지속성 서비스를 하기 위해 **session** 내장 객체를 이용

□ 내장 객체 session

- ▣ 클라이언트마다 세션 정보를 저장 및 유지 관리하기 위한 객체
- ▣ 자료유형이 인터페이스 `javax.servlet.http.HttpSession`
- ▣ 세션관리를 위한 다양한 메소드를 제공

내장 객체 config

39

□ 자료유형

- ▣ 인터페이스 `javax.servlet.ServletConfig`
- ▣ 서블릿이 초기화되는 동안, **JSP** 컨테이너가 환경 정보를 서블릿으로 전달할 때 사용하는 객체

스몰과제

40

- 강의 동영상 자료에 있는 실습 문제(총 12개)를 모두 프로그램 한 후 소스와 출력 결과를 편집하여 제출하세요.(번호가 잘못지정되어 있습니다. 실습 문제는 총 12개 입니다.)