

HTTP

3. HTTP, Servlet, JSP, JSTL Review

◆ HyperText Transfer Protocol (HTTP)

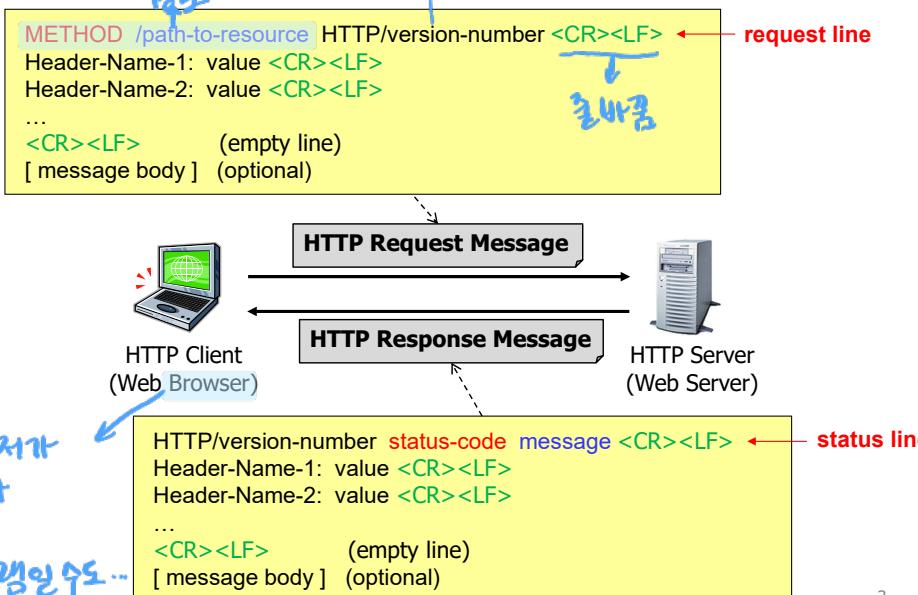
- 웹 메시지 전송을 위한 응용 계층(application-layer) 프로토콜
- Request-response protocol in the client-server computing model
- Stateless protocol: client와 server 사이에 상태 데이터를 유지 및 공유하지 않음
- [HTTP - Hypertext Transfer Protocol Overview \(w3.org\)](#)



HTTP Messages

client가 사용하는 ver

1.3



3

HTTP Messages

◆ 예: <http://cs.dongduk.ac.kr> 요청

Request message

Chrome 개발자 도구

이 뒤에 오는 부분 (근데 없어서 /로 넣겠)

요청
/index.html

Name	Value
Host	cs.dongduk.ac.kr
Connection	keep-alive
Cache-Control	max-age=0
sec-ch-ua	"Chromium";v="92", "Not A;Brand";v="99", "Google Chrome";v="92"
sec-ch-ua-mobile	?0
Upgrade-Insecure-Requests	1
User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/92.0.4515.159 Safari/537.36
Accept	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site	same-origin
Sec-Fetch-Mode	navigate
Sec-Fetch-User	?1
Sec-Fetch-Dest	document
Referer	https://cs.dongduk.ac.kr/
Accept-Encoding	gzip, deflate, br
Accept-Language	ko-KR,ko;q=0.9,en-US;q=0.8,en;q=0.7

request line
GET /index.html

request headers

2

4

HTTP Messages

Response message

status line

response headers

message body (HTML문서)

5

GET vs. POST

◆ GET

- 가장 일반적인 request method
- Message body가 필요 없음
- Parameters(data) passing
 - URL 내에 미리 포함되거나, HTML form을 통해 입력된 값
 - URL 내에 query string(name-value 쌍)으로 포함되어 전달됨
 - ASCII 문자열만 가능하고 길이의 제한이 있으며 URL 상에 노출됨

◆ POST

- 증액 문자는 ASCII 문자로 변환*
- Message body를 가짐
 - Parameters(data) passing
 - HTML form을 통해 입력된 값
 - Message body 내에 byte stream으로 포함되어 전달됨
 - Binary data도 가능하고 길이의 제한이 없으며 URL에 노출되지 않음
 - 대량의 중요 데이터 전송이나 file upload 등에 적합

7

Request Methods

◆ Request Methods

Method	Description
GET	Requests data from a specified resource
POST	Submits data to be processed to a specified resource
HEAD	Same as GET but returns only HTTP headers and no document body
PUT	Uploads a representation of the specified URI
DELETE	Deletes the specified resource
OPTIONS	Returns the HTTP methods that the server supports
TRACE	Echoes back the received request
CONNECT	Converts the request connection to a transparent TCP/IP tunnel

6

GET vs. POST

◆ GET method 사용 예

- 커리스토리*
- Web browser의 주소창에 URL 입력
 - `http://www.example.org/app/multiply.jsp?num1=3&num2=7`
 - Web page 내에 정의된 hyper-link click
 - `Multiply!`
 - HTML Form에서 입력

```
<!- /app/multiplyForm.html -->
<form method="GET" action="multiply.jsp">
  <p>Please specify the multiplicands:
  <input type="text" name="num1" size="5"> ← 텍스트 입력창
  <input type="text" name="num2" size="5"><BR> ← 텍스트 입력창
  <input type="submit" value="Multiply!"> ← submit 버튼
</form>
```

3, 7 입력 후 submit button click

`GET /app/multiply.jsp?num1=3&num2=7 HTTP/1.1`
Host: `www.example.org`
...(other headers)...

8

GET vs. POST

◆ POST method 사용 예

```
<!-- /app/registerForm.html -->
<form method="POST" action="registerServlet">
  Name : <input type="text" name="name"/><br/>
  SSN : <input type="text" name="birth"/> - <input type="text" name="ssn"/><br/>
  Id : <input type="text" name="id"/><br/>           ← 텍스트 입력창
  Pass : <input type="password" name="pw"/><br/>    ← password 입력창
  <input type="hidden" name="secret" value="secret_val"/> ← hidden parameter
  <input type="submit" value="전송"/>
</form>
```



```
POST /app/registerServlet HTTP/1.1
Host: www.example.org
...(other headers)...
name=kim&birth=240101&ssn=3456789&id=scott&pw=TIGER&secret=secret_val
```

Status Codes

◆ 예

Status Code & Message	Meaning
200 OK	Request가 성공적으로 처리되고 요청된 resource가 client에게 송신되었음
201 Created	Request가 성공적으로 처리되고 새로운 resource가 서버에 생성되었음
301 Moved Permanently, 302 Found (or Moved Temporarily)	요청된 자원이 새로운 장소로 이동되었음. client는 Location 헤더의 값으로 지정된 URL로 새로운 요청을 보내야 함 (redirection)
400 Bad request	Request의 구문이 잘못됨
403 Forbidden	Request는 올바르나 server가 처리를 거부함
404 Not found	요청된 자원을 찾을 수 없음
500 Internal Server Error	Server에 오류가 발생하여 request를 처리하지 못함

Status Codes

◆ Status Codes & Messages

- Response message의 status line에 포함됨
 - Web server가 request를 수신 및 처리한 결과의 종류를 나타냄
 - Web client가 해야 할 action을 지시
-
- ◆ 종류
 - 1xx (Informational): Request received, server is continuing the process
 - 2xx (Successful): The request was successfully received, understood, accepted and serviced
 - 3xx (Redirection): Further action must be taken in order to complete the request
 - 4xx (Client Request Error): The request contains bad syntax or cannot be understood
 - 5xx (Server Error): The server failed to fulfill an apparently valid request

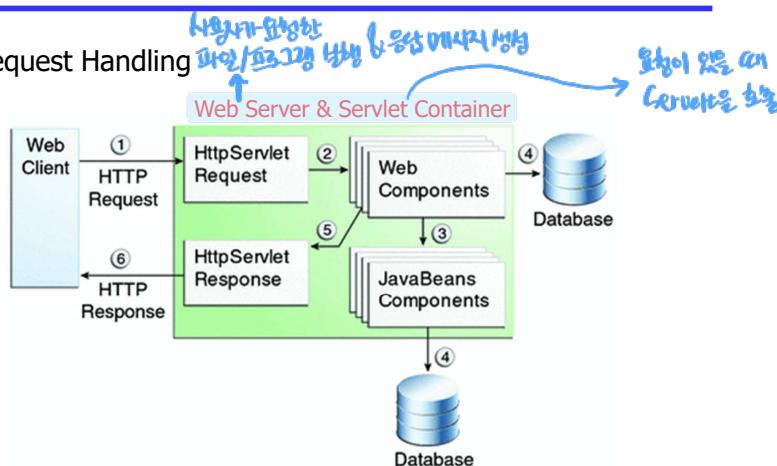
HTTP Headers

◆ HTTP Headers

종류	설명	예
General Headers	요청/응답 메시지에 대한 일반적인 정보	<ul style="list-style-type: none">• Date: Sat, 26 Oct 2023 22:28:31 GMT• Connection: keep-alive• Warning: 199 Miscellaneous warning
Request Headers	클라이언트 및 요청과 관련된 정보	<ul style="list-style-type: none">• User-Agent: Mozilla/4.75 [en] (WinNT; U)• Host: www.example.org• Accept: image/gif, image/x-bitmap• Referer: http://www.cs.rutgers.edu/~shklar/index.html• Authorization: Basic [encoded-credentials]• Cookie: userId=JohnDoe; login=true;
Response Headers	서버 및 응답과 관련된 정보	<ul style="list-style-type: none">• Location: http://www.server.com/relocatedPage.html• WWW-Authenticate: Basic• Server: Apache/2.4.1 (Unix)• Set-Cookie: userId=JohnDoe; max-age=3600;
Entity Headers	message body 또는 target resource에 관한 정보	<ul style="list-style-type: none">• Content-Type: application/x-www-form-urlencoded• Content-Length: 348• Last-Modified: Sat, 26 Oct 2023 22:28:31 GMT

Web Programming by Servlet & JSP

◆ HTTP Request Handling



- **Web Components:** Servlets 또는 JSP pages
- **javax.servlet.http.HttpServletRequest, HttpServletResponse:**
HTTP 요청 및 응답 메시지에 대한 정보(데이터)를 갖고 관련된 메소드들을 제공

13

Web Programming by Servlet & JSP

◆ Servlet

- Java 언어 기반의 웹 프로그래밍 기술(API)

- 모든 Java API 사용 가능
- 호환성 및 확장가능성 높음

- Multi-thread로 실행되어 빠른 처리 속도를 가짐

- 기본 구현 클래스인 **HTTPServlet**을 상속해서 **Servlet** 클래스 구현
 - 기 지정된 URL에 대한 요청 발생 시 **Servlet Container**에 의해 호출됨
 - 요청 및 응답 정보를 포함하는 **HttpServletRequest** 및 **HttpServletResponse** 객체를 전달받음

- 웹 애플리케이션을 **Servlet**으로만 구현 시 동적인 결과 화면 출력을 위한 Java code를 작성해야 함

- **MVC design pattern**에서 **Controller** 구현을 위해 사용됨

14

Web Programming by Servlet & JSP

◆ Servlet 구현 예

```
@WebServlet(urlPatterns="/helloWorld")
public class HelloWorldServlet extends HttpServlet{
    private String time;
    private SimpleDateFormat format;
    public void init() {
        dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    }
    public void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException { // service() 대신 doGet()이나 doPost() 메소드 정의 가능
        time = dateFormat.format(new Date());
        String message = "Hello " + request.getParameter("name"); // request parameter 값 추출
        response.setContentType("text/html;charset=utf-8"); // Content-Type header 생성
        PrintWriter out = response.getWriter();
        out.print("<HTML><HEAD><TITLE>"); out.print(message);
        out.println("</TITLE></HEAD></HTML>");
        out.println("<BODY>");
        for (int i = 1; i <= 5; i++) {
            out.print(i); out.print(" : "); out.print(time); out.println("<BR/>");
        }
        out.println("</BODY>"); out.println("</HTML>"); } }
```

결과 화면(html) 생성 코드

Web Programming by Servlet & JSP

◆ JSP(Java Server Pages)

- Servlet 기반의 server-side script 언어

- Java web application에서 presentation layer의 구현 기술로 사용됨

- JSP page는 최초 요청 처리 시 servlet으로 변환되고 메모리에 load됨 → 미리 컴파일 X

- 이후의 요청은 변환 과정 없이 servlet 객체에서 바로 처리됨

- 정적인 HTML page 내에 Java 언어를 사용하여 동적인 프로그램 작성 가능

- Expression Language(EL), JSP Standard Tag Library(JSTL) 등 다양한 확장 기술 이용 가능

- UI, Business logic 및 data 처리 구현을 동시에 할 수 있으나 프로그램의 복잡도 및 유지보수의 어려움 증가

- **MVC design pattern**에서 **View** 구현을 위해 사용됨

별도의 클래스로 따로 정의 후
MVC 중 Model이 담당

16

JSP 개요

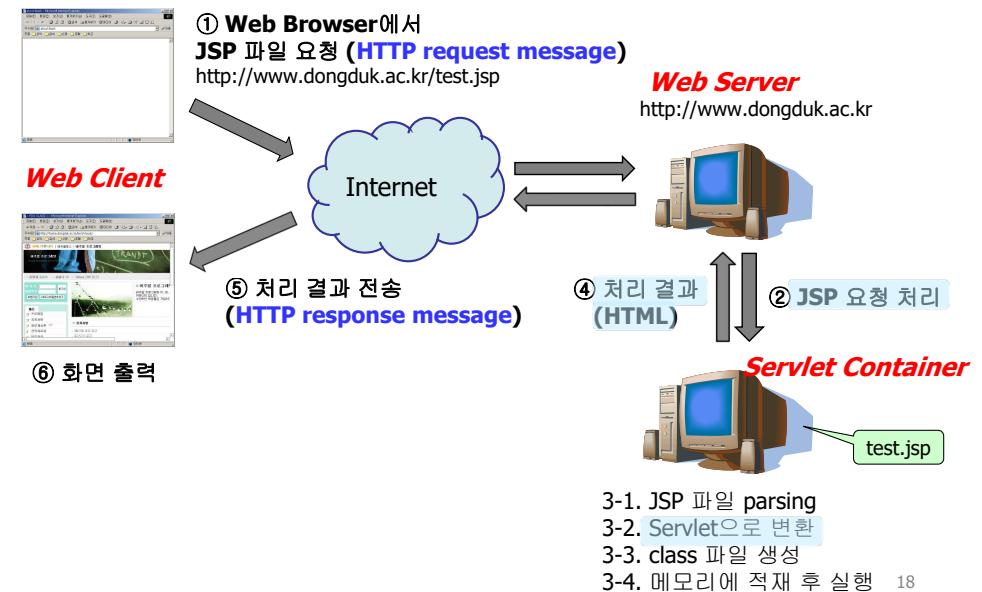
◆ JSP 구현 예

```
<%@ page contentType="text/html;charset=utf-8" pageEncoding="utf-8" %>
<%@ page import="java.util.* , java.text.*" %>      <!-- 지시자(directive) -->
<%! //declaration
private String time;
private SimpleDateFormat dateFormat;

public void init() {                                // life-cycle init method
    dateFormat = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
}
%>
<% // scriptlet
time = dateFormat.format(new Date());
String message = "Hello " + request.getParameter("name"); // request 객체 이용
%>
<HTML>
<HEAD><TITLE><%= message %></TITLE></HEAD>  <!-- 표현식(expression) -->
<BODY>
<% // scriptlet
for (int i = 1; i <= 5; i++) {
%
    <%= i %> : <%= time %><BR/>           <!-- 표현식 -->
%
}<% } %>
</BODY>
</HTML>
```

17

JSP Request 처리 과정



Servlet & JSP 개발 환경

◆ Eclipse IDE for Enterprise Java and Web Developers

- Enterprise application 개발을 지원하는 통합개발도구(IDE)
- 웹 애플리케이션 개발에 필요한 WTP(Web Tools Project) 모듈 포함
- Installation
 - <http://www.eclipse.org/downloads/>에서 installer를 다운로드 후 실행
 - 설치 목록에서 Eclipse IDE for Enterprise Java and Web Developers 선택

◆ Apache Tomcat

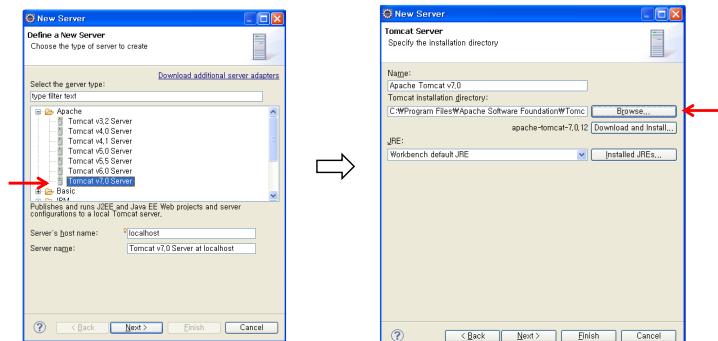
- Web Server & Servlet Container
 - Servlet과 JSP 실행을 위한 Enterprise Java 기능 제공
- Installation
 - <https://tomcat.apache.org/download-90.cgi>에서 Core > 32-bit/64-bit Windows Service Installer를 다운로드 후 실행

19

참고: Eclipse 설정

◆ Tomcat server 등록

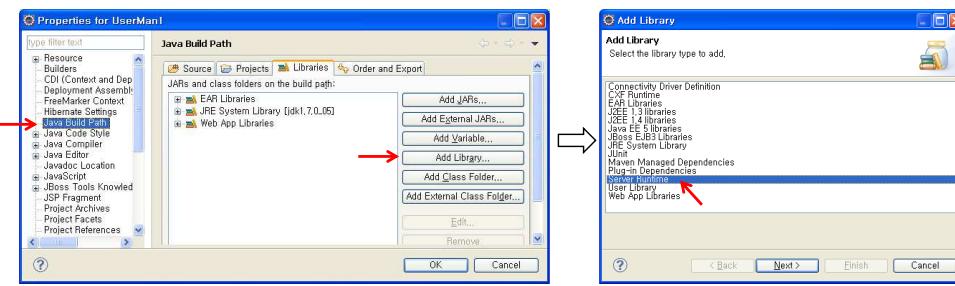
- Servers 창에서 right-click > context menu에서 new > server 실행
 - Apache > Tomcat v9.0 Server 선택 > Browse 버튼 클릭 > Tomcat 설치 폴더 지정
- Tomcat Server Start
 - 포트 관련 오류 발생 시 Server를 double-click해서 설정 메뉴를 연 후 Tomcat admin port 번호 입력, HTTP/1.1 포트 번호를 수정 및 저장한 후 Server를 restart함



20

참고: Eclipse 설정

- ◆ Web project의 build path 설정 (**→ Maven project 이용 시 불필요**)
 - Dynamic Web Project 생성 또는 WAR file import 시 Tomcat library 추가
 - Project 이름 선택 후 right-click, context menu에서 "Properties" 선택
 - 또는 "Build Path" > "Configure Build Path..." 선택
 - Java Build Path 확인
 - Libraries 탭에서 "Apache Tomcat v9.0" 포함 여부 확인
 - 없을 경우 Add Library > Server Runtime > Apache Tomcat v9.0 선택 및 추가



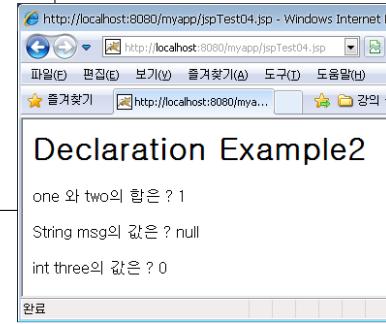
21

Script : 선언부(Declaration)

- ◆ 예

```
<h1>Declaration Example2</h1> <!-- jspTest04.jsp -->
<%!
    int one;
    int two = 1;
    public int plusMethod(){
        return one + two;
    }
    String msg;      // null 문자열로 초기화
    int three;       // 0으로 초기화
%>

one과 two의 합은 ? <%=plusMethod()%><p>
String msg의 값은 ? <%=msg%><p>
int three의 값은 ? <%=three%>
```



23

Script : 선언부(Declaration)

- ◆ JSP page 안에서 사용할 변수나 메소드를 선언하는 문장

- 형식

<%!

변수 선언 또는 메소드 선언

%>

- 선언문으로 선언한 변수는 JSP 파일이 parsing되어 만들어지는 클래스의 member 변수 및 메소드로 바뀜
- 해당 JSP page 어디에서나 참조 및 호출 가능
- 변수는 자동으로 초기화됨
- JSP 내의 메소드 선언은 가급적 피하고 별도의 Java class나 JavaBeans를 사용하는 것이 바람직함
- 선언부 내에서는 JSP 내장 객체(default object) 참조 불가
 - 예: request, response, session, out 등

22

Script : Scriptlet

- ◆ JSP page 내에 Java code를 기술하는 부분

- 형식

<%

Java codes ...

%>

- JSP page 작성 시 가장 많이 쓰임
- JSP page가 Servlet으로 변환될 때 _jspService() 메소드 안에 포함됨
 - 변수 선언 시 local 변수가 되므로 자동으로 초기화되지 않음
- 복잡한 로직을 포함하는 scriptlet은 개발 및 유지보수의 어려움이 있으므로, 표현 언어(EL), JSTL, Custom Tag, JavaBeans의 사용을 권장함

24

Script : 표현식 (Expression)

- JSP code에서 생성한 값을 출력

- 형식

- <%= 변수명, 산술식, 또는 메소드 호출 %>

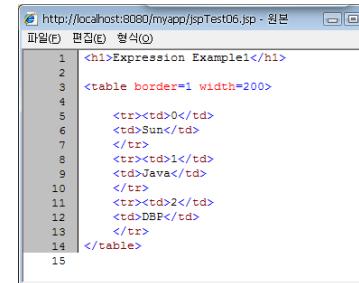
- JSP code 내의 변수 값 또는 메소드 호출 결과 값을 출력할 때 사용
- 세미콜론(;)은 생략
- out.print() 메소드 호출과 동일한 효과

25

Script : 표현식 (Expression)

- 예

```
<h1>Expression Example1</h1>      <!-- jspTest06.jsp -->
<%! String name[] = {"Sun", "Java", "DBP"}; %>
<table border=1 width=200>
<% for (int i=0; i<name.length; i++) {%
    <tr><td> <%=i%> </td>
    <td><%=name[i]%></td></tr>
<% } %>
</table>
```

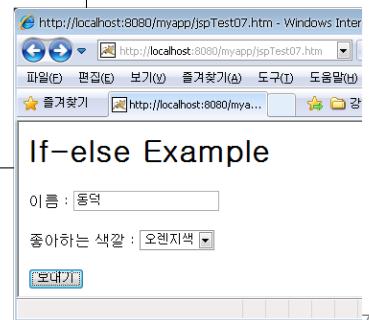


26

제어문 : if 문

- if - else 문

```
<!-- jspTest07.htm -->
<h1>If-else Example</h1>
<form method="POST" action="jspTest07.jsp">
이름 : <input type="text" name="name"><p>
좋아하는 색깔 :
<select name="color">
    <option value="blue" selected>파란색</option>
    <option value="red">붉은색</option>
    <option value="orange">오렌지색</option>
    <option value="etc">기타</option>
</select><p>
<input type="submit" value="보내기">
</form>
```

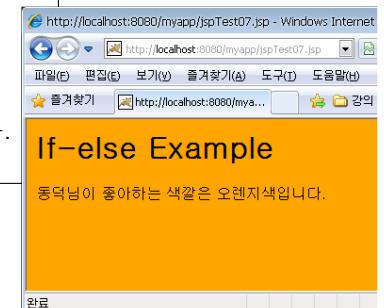


제어문 : if 문

- if - else 문

```
<h1>If-else Example</h1>      <!-- jspTest07.jsp -->
<%! String msg; %>
<%
String name = request.getParameter("name");
String color = request.getParameter("color");

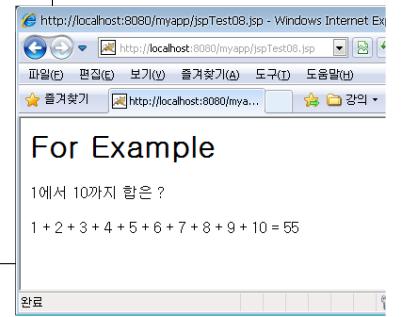
if (color.equals("blue")) { msg = "파란색"; }
else if (color.equals("red")) { msg = "붉은색"; }
else if (color.equals("orange")){ msg = "오렌지색"; }
else { color = "white"; msg = "기타색"; }
%>
<body bgcolor=<%=color%>>
<%=name%>님이 좋아하는 색깔은 <%=msg%>입니다.
</body>
```



제어문 : for 문

◆ for 문

```
<h1>For Example</h1> <!-- jspTest08.jsp -->
1에서 10까지 합은 ?<p>
<%
    int i,sum = 0;
    for (i=1; i<=10; i++) {
        if (i<10){
%>
        <%=(i + " ")%>
<% }
        else {
            out.println(i + " =");
        }
        sum += i;
    }
%>
<%=sum%>
```



29

제어문 : while 문

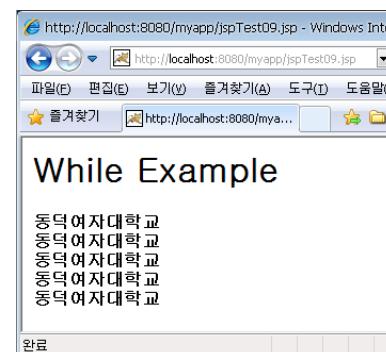
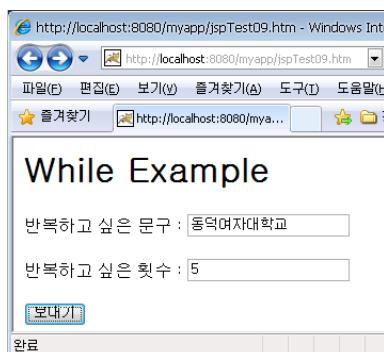
◆ while 문

```
<h1>While Example</h1> <!-- jspTest09.htm -->
<form method="POST" action="jspTest09.jsp">
    반복하고 싶은 문구 : <input type="text" name="msg" size="20"><p>
    반복하고 싶은 횟수 : <input type="text" name="number"><p>
    <input type="submit" value="보내기">
</form>
```

```
<h1>While Example</h1> <!-- jspTest09.jsp -->
<%
    String msg = request.getParameter("msg");
    int number = Integer.parseInt(request.getParameter("number"));
    int count = 0;
    while(count < number) {
%>
        <b><%=msg%></b><br>
<% count++;
    }
%>
```

30

제어문 : while 문



31

JSP 내장 객체

- ◆ JSP page를 작성할 때 필요한 여러 기능을 제공하기 위해 Servlet Container가 기본적으로 생성 및 제공하는 객체(default object)
 - Servlet으로 변환될 때 _jspService() 메소드 내에 선언되고 참조됨

객체명	타입	설명
request	javax.servlet.http.HttpServletRequest	클라이언트가 전송한 요청 정보를 저장
response	javax.servlet.http.HttpServletResponse	클라이언트에게 전송할 응답 정보를 저장
out	javax.servlet.jsp.JspWriter	JSP page가 생성하는 결과를 출력하는 스트림
session	javax.servlet.http.HttpSession	세션 정보를 저장
application	javax.servlet.ServletContext	웹 어플리케이션에 대한 정보를 저장
pageContext	javax.servlet.jsp.PageContext	JSP page에 대한 정보 및 관련 객체들을 저장
page	javax.servlet.jsp.HttpJspPage	JSP page가 변환된 servlet 클래스 객체
config	javax.servlet.ServletConfig	JSP page에 대한 설정 정보를 저장
exception	java.lang.Throwable	예외 처리 페이지에 전달된 예외 객체

32

request 객체

- Client가 Server에 전달한 요청 정보를 저장하는 JSP의 기본 객체



- Servlet 클래스의 service(), doGet(), doPost()의 첫 번째 parameter로 전달되는 HttpServletRequest 객체와 동일
- 주요 기능
 - Client가 전송한 요청에 대한 세부 정보 참조
 - ex. getMethod(), getRequestURI(), getServletPath(), getContentType(), ...
 - 요청 매개변수(request parameter) 참조
 - ex. getParameter(), getParameters(), getParameterMap(), ...
 - 요청 헤더(request header) 참조
 - ex. getHeader(), getHeaders(), ...
 - cookie 및 session 객체 참조(생성)
 - ex. getCookies(), getSession()
 - 속성(attribute) 처리
 - ex. getAttribute(), setAttribute(), ...

33

request 객체

- 요청에 대한 세부 정보 참조

메소드 명	설명
String getRemoteAddr()	웹 서버에 접속한 클라이언트의 IP 주소를 반환함
long getContentLength()	클라이언트가 전송한 요청 정보(body)의 길이를 반환함
String getCharacterEncoding()	요청 정보(body)의 인코딩 방식을 반환함 (ex. utf-8)
String getContentType()	요청 정보(body)의 MIME type을 반환함 (ex. text/html)
String getMethod()	요청에 사용된 HTTP method를 반환함 (ex. GET, POST)
String getScheme()	요청의 전송 scheme를 반환함 (ex. http, https, ftp)
String getServerName()	서버의 호스트 이름을 반환함 (ex. www.example.org)
int getServerPort()	서버가 실행중인 포트 번호를 반환함 (ex. 8080)
String getRequestURI()	요청 URL의 일부를 반환함 (ex. /userManager/user/login)
String getContextPath()	요청 URL에 포함된 문맥 경로를 반환함 (ex. /userManager)
String getServletPath()	요청 URL에 포함된 servlet 경로를 반환함 (ex. /user/login)

34

request 객체

- Request parameter 참조
 - Client가 HTML Form의 입력 요소를 사용하여 전송한 정보를 다음 메소드를 이용해서 조회 가능

메소드 명	설명
String getParameter(String name)	name이라는 이름의 요청 파라미터의 값을 문자열로 반환함 (존재하지 않을 경우 null 반환)
String[] getParameterValues(String name)	name이라는 이름의 요청 파라미터들의 값을 문자열 배열로 반환함 (존재하지 않을 경우 null 반환)
java.util.Enumeration getParameterNames()	전송된 모든 요청 파라미터의 이름들을 열거형으로 반환함
java.util.Map getParameterMap()	전송된 모든 요청 파라미터들을 Map으로 반환함

35

response 객체

- 요청에 대한 처리 결과로 Client에게 보낼 응답 정보를 저장하는 객체
- Servlet의 service(), doGet(), doPost() 등의 두 번째 parameter로 전달되는 HttpServletRequest 객체와 동일
- response 객체의 기능
 - Response message의 header 정보 설정
 - 문서 encoding 형식, 암호화 형식, 전송 주소, cookie 등에 대한 정보
 - Error 응답 처리, Redirection 수행 등

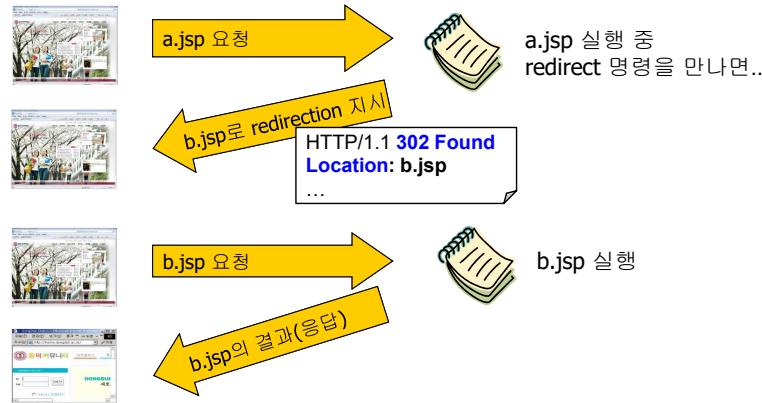
메소드 명	설명
setContent-Type(String type)	문자열 형태의 MIME Type으로 ContentType 을 설정
setHeader(String name, String value)	문자열 name 이름과 문자열 value 값을 헤더로 설정
setDateHeader(String name, long date)	문자열 name 이름으로 date에 설정된 milisecond 시간 값을 헤더로 설정
sendError(int status, String msg)	상태 코드와 메시지를 설정하여 에러 응답 전송
sendRedirect(String url)	지정한 URL로 redirection하도록 지시

36

Redirection

◆ response 객체를 이용한 redirection 실행

- 웹 서버가 클라이언트에게 다른 URL로 이동하라고 지시
- 클라이언트로부터 새로운 요청이 생성 및 전송됨
 - 기존 request 객체는 삭제되어 재사용 불가!



37

Redirection

◆ `HttpServletResponse#sendRedirect(String location)` 메소드

- 주어진 location(URL)로 redirection을 지시하는 응답 생성
 - `response.sendRedirect("http://www.dongduk.ac.kr");`
 - ✓ 외부의 URL로 요청
 - `response.sendRedirect("test.jsp");`
 - ✓ 같은 서버 프로그램 내의 jsp 파일을 요청

- 예

```
<html>                                         <!-- jspTest10.htm -->
<head> <title> Redirect Test </title></head>
<body>
    jspTest10.jsp 로 이동 후 아래 내용 중 하나로 Redirection<br>
    <form name="redirectForm" action="jspTest10.jsp">
        <input type="radio" name="direction" value="home" checked="true">
        학교 홈페이지<br>
        <input type="radio" name="direction" value="jsp"> jspTest11.jsp 파일<p>
        <input type="submit" value="이동">
    </form>
</body></html>
```

38

Redirection

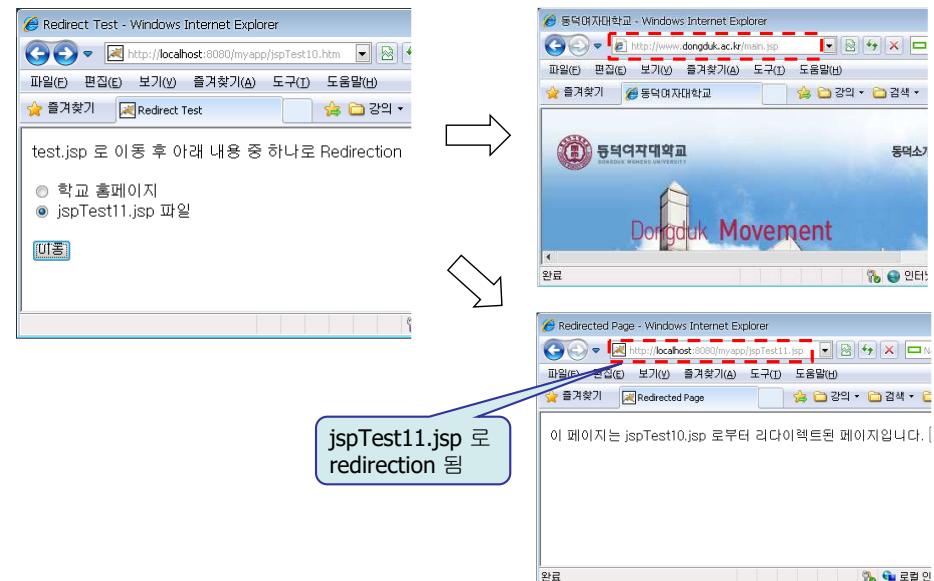
```
<%@ page contentType="text/html;charset=utf-8" %>      <!-- jspTest10.jsp -->
<%
    String info = request.getParameter("direction");
    if (info.equals("home"))
        response.sendRedirect("http://www.dongduk.ac.kr");
    else if (info.equals("jsp"))
        response.sendRedirect("jspTest11.jsp");
%>
```



```
<%@ page contentType="text/html;charset=utf-8" %>      <!-- jspTest11.jsp -->
<html>
<head><title> Redirected Page </title>
<script language="javascript">
    function back() { history.go(-1); }
</script></head>
<body>
    <% String info = request.getParameter("direction"); %>      <!-- null -->
    이 페이지는 jspTest10.jsp 로부터 리다이렉트된 페이지입니다.
    <input type="button" value="돌아가기" onClick="javascript:back()">
</body>
</html>
```

39

Redirection



데이터 저장 영역

◆ pageContext, request, session, application 객체

개방자

- 사용자가 속성(attribute)을 통해 값이나 객체를 저장 및 참조할 수 있음
- 서로 다른 저장 영역과 scope(생성 및 소멸 시점)를 가짐
- **pageContext (javax.servlet.jsp.PageContext 객체)**
 - 각 JSP page에 관련된 정보 저장
- **request (javax.servlet.http.HttpServletRequest 객체)**
 - 각 요청에 관련된 정보 저장
 - 다른 page를 include하거나 다른 page로 forward할 때 전달됨
- **session (javax.servlet.http.HttpSession 객체)**
 - 각 client로부터 전송된 여러 요청들에 대해 유지 및 공유되는 영역
 - Servlet에서는 `request.getSession()`으로 생성 또는 획득
- **application (javax.servlet.ServletContext 객체)**
 - Web container에서 관리하며 모든 client에 대해 공유 가능
 - Servlet에서는 `this.getServletContext()`로 획득

41

데이터 저장 영역

◆ pageContext, request, session, application 객체

- 속성 저장 및 참조를 위한 메소드

메소드명	설명
<code>void setAttribute(String name, Object value)</code>	이름이 name인 속성에 value 값을 저장함
<code>Object getAttribute(String name)</code>	이름이 name인 속성의 값을 읽어옴 (속성이 없는 경우 null을 반환)
<code>void removeAttribute(String name)</code>	이름이 name인 속성을 삭제함
<code>java.util.Enumeration getAttributeNames()</code>	속성 이름 목록을 반환함 (pageContext에서는 제공되지 않음)

42

JSP 지시자(Directive)

◆ JSP page가 요청되어 실행될 때 해당 페이지를 어떻게 처리할 것인지 를 container에게 알리는 역할 수행

◆ 형식

```
<%@ 지시자이름 속성1="값1" 속성2="값2"... %>
```

◆ 종류

- **page**
- **include**
- **taglib**

43

page 지시자

◆ JSP page의 여러 가지 속성을 정의

속성	설명	기본 값
<code>contentType</code>	Content의 MIME type과 character set 설정	text/html; charset=ISO-8859-1
<code>pageEncoding</code>	JSP page 내의 character set 지정	ISO-8859-1
<code>import</code>	JSP 파일 내에서 사용할 외부 자바 패키지나 클래스 지정	
<code>language</code>	script 언어를 지정	java
<code>session</code>	세션 생성 여부 지정	true
<code>buffer</code>	버퍼 크기 지정	8kb
<code>autoFlush</code>	버퍼 내용 자동 비움 지정	true
<code>isThreadSafe</code>	단일 쓰레드 모델을 사용하여 동시 성 제어 여부 지정	true
<code>info</code>	JSP page 설명	
<code>errorPage</code>	에러 발생 시 호출 페이지 지정	
<code>isErrorPage</code>	에러만 처리하는 페이지 지정	false

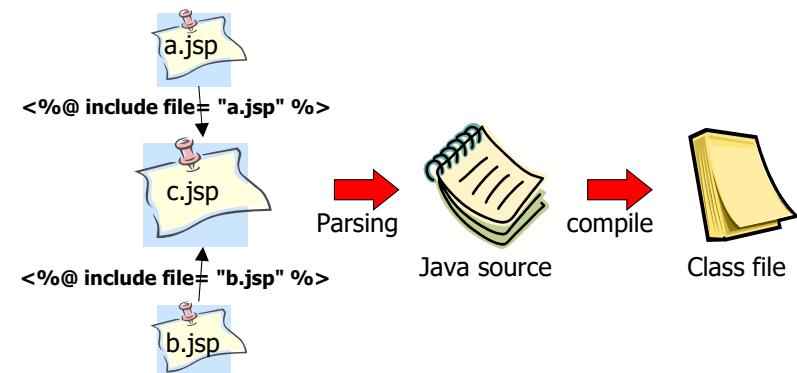
page 지시자 속성

- ◆ contentType
 - JSP page가 생성하는 문서의 종류 및 문자 집합을 지정하는 속성
 - 예: 유니코드 문자 집합을 사용하는 HTML 문서를 생성할 경우
 - <%@ page contentType="text/html; charset=utf-8" %>
- ◆ pageEncoding
 - JSP page 자체의 인코딩 방식을 지정하는 속성
 - 예 : <%@ page pageEncoding="utf-8" %>
- ◆ import
 - JSP page에서 사용할 자바 클래스를 import할 때 지정하는 속성
 - 예: java.util.* 과 java.io.* 를 포함하는 경우
 - <%@ page import="java.util.*, java.io.*" %>

45

include 지시자

- ◆ 여러 JSP page에서 공통으로 사용되는 부분을 별도의 JSP page로 만들어 놓고 이를 특정 JSP page에 포함시킬 때 사용
- ◆ 형식: <%@ include file="포함할 파일명" %>
- ◆ include한 JSP page를 포함하여 하나의 servlet을 생성
 - 변수 공유가 가능하나, 변수의 이름을 중복 사용할 수 없음



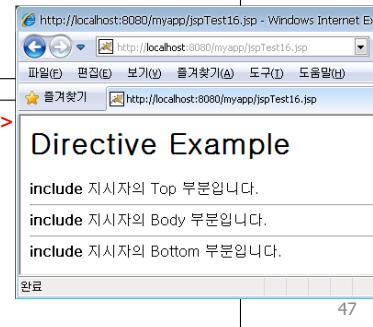
46

include 지시자 사용 예

```
<%@ page contentType="text/html;charset=utf-8" %>    <!-- jspTest16.jsp -->
<%! String name = "include"; %>
<%@ include file="jspTest17.jsp"%>
<b>include</b> 지시자의 Body 부분입니다.
<%@ include file="jspTest18.jsp"%>
```

```
<%@ page pageEncoding="utf-8" %>                <!-- jspTest17.jsp -->
<html>
<body>
<h1>Directive Example</h1>
<b><%=name%></b> 지시자의 Top 부분입니다.
<hr>
```

```
<%@ page import="java.util.*"      <!-- jspTest18.jsp -->
pageEncoding="utf-8" %>
<% Date date = new Date(); %>
<hr>
<b>include</b> 지시자의 Bottom 부분입니다.<p>
<%=date.toLocaleString()%>
</body>
</html>
```



47

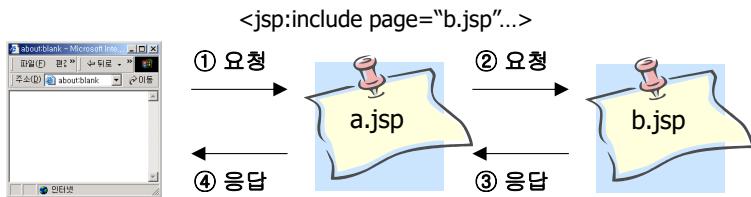
Action Tags

- ◆ 특정 동작이 일어날 때 JSP page를 제어하기 위해 사용
 - JSP page들 사이의 흐름 제어
 - Java Applet 지원
 - JavaBeans와의 상호작용 지원
- ◆ Action Tag의 종류
 - include
 - forward
 - plug-in
 - useBean
 - setProperty
 - getProperty

48

include Action

- ◆ JSP page에 다른 resource(JSP page, servlet, HTML file)를 포함시킴
 - include 지시자와 유사하나 포함되는 page를 먼저 실행 후 그 실행 결과를 현재 page에 포함시키는 점이 다름
 - 각 페이지는 독립적으로 컴파일되어 실행됨
 - 변수 공유 불가
 - **request 및 response 객체가 전달됨**
 - jsp:param 태그를 이용해서 추가적인 파라미터 전달 가능
- ◆ 형식: <jsp:include page="local URL" />



49

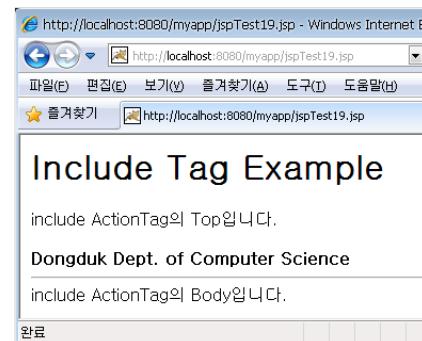
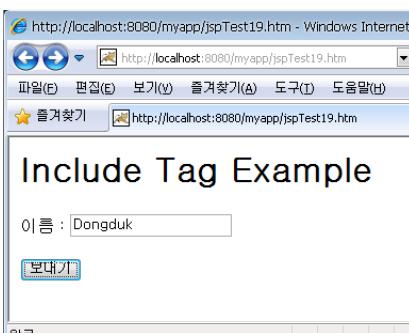
include Action 사용 예

```
<h1>Include Tag Example</h1>           <!-- jspTest19.htm -->
<form method="POST" action="jspTest19.jsp">
    이름 : <input type="text" name="name"><p>
    <input type="submit" value="보내기">
</form>
```

```
<%@ page contentType="text/html;charset=utf-8"%>      <!-- jspTest19.jsp -->
<%
    String name = "Dong-duk";
%>
<html> <body>
    <h1>Include Tag Example</h1>
    <jsp:include page="jspTest20.jsp"/>
    include ActionTag의 Body입니다.
</body></html>
```

```
<%@ page contentType="text/html;charset=utf-8"%>      <!-- jspTest20.jsp -->
<% String name = request.getParameter("name"); %>
include ActionTag의 Top입니다.<p>
<b><%=name%> Dept. of Computer Science</b><hr>
```

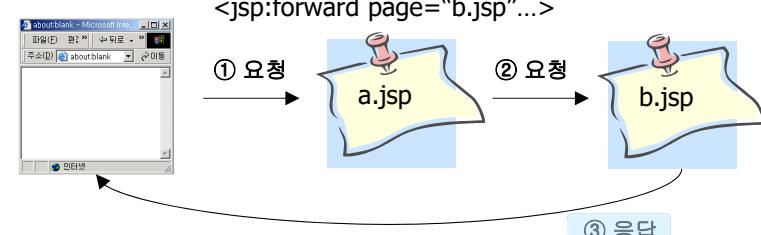
include Action 사용 예



51

forward Action

- ◆ JSP page에서 다른 resource로 제어를 넘김
 - 사용자의 선택에 따라 다른 JSP page로 이동하고자 할 때 사용
 - **request 및 response 객체가 전달됨 (Redirection과 차이점)**
 - jsp:param 태그를 이용해서 추가적인 파라미터 전달 가능
 - 기존 page로 되돌아올 수 없음
- ◆ 형식: <jsp:forward page="local URL" />



52

forward Action 사용 예

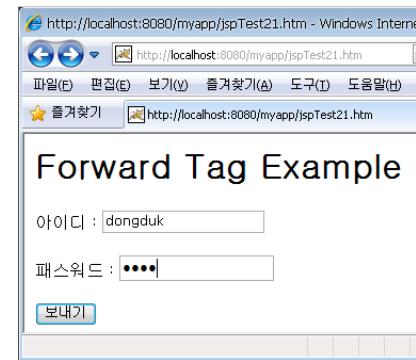
```
<h1>Forward Tag Example</h1>          <!-- jspTest21.htm -->
<form method="POST" action="jspTest21.jsp">
    아이디 : <input type="text" name="id"><p>
    패스워드 : <input type="password" name="password"><p>
    <input type="submit" value="보내기">
</form>

<%@ page contentType="text/html;charset=utf-8"%>      <!-- jspTest21.jsp -->
<% request.setCharacterEncoding("utf-8"); %>
<html><body>
    Forward Tag의 포워딩 되기 전의 페이지입니다.
    <jsp:forward page="jspTest22.jsp"/>
</body></html>

<%@ page contentType="text/html;charset=utf-8"%>      <!-- jspTest22.jsp -->
<%
    String id = request.getParameter("id");
    String password = request.getParameter("password");
%>
    당신의 아이디는 <%=id%>이고 <p>패스워드는 <%=password%>입니다.
```

53

forward Action 사용 예



Forward Tag Example

아이디 :

패스워드 :



Forward Tag Example

당신의 아이디는 dongduk이고
패스워드는 univ입니다.

54

pageContext 내장 객체 활용

- ◆ [javax.servlet.jsp.PageContext](#) pageContext
 - Servlet에서 자동으로 생성된 내장 객체 관리
 - forward 및 include 기능 제공
 - abstract void [forward\(String relativeUrlPath\)](#)
 - abstract void [include\(String relativeUrlPath\)](#)

```
<%@ page contentType="text/html;charset=utf-8"%>
<html><body>
    <jsp:forward page="jspTest22.jsp"/>
    <jsp:include page="jspTest20.jsp"/>
</body></html>
```



```
<%@ page contentType="text/html;charset=utf-8"%>
<html><body>
<%
    pageContext.forward("jspTest22.jsp");
    pageContext.include("jspTest20.jsp");
%
</body></html>
```

55

Servlet에서 JSP로의 forward/include

- ◆ [javax.servlet.RequestDispatcher](#) interface의 메소드 이용
 - void [forward\(ServletRequest request, ServletResponse response\)](#)
 - void [include\(ServletRequest request, ServletResponse response\)](#)
- ◆ 예

```
public class jspTest21 extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");
        PrintWriter out = response.getWriter();
        out.println("<html><body>"); → JSP가 처리해 줍니다
        RequestDispatcher dispatcher = request.getRequestDispatcher("jspTest22.jsp");
        dispatcher.forward(request, response); // 또는 dispatcher.include(...);
        out.println("</body></html>");
    }
}
```

// 주의: 위 행은 include 후에는 실행되나 forward 후에는 실행 안 됨

Servlet에서 JSP로의 forward/include

◆ Login 구현 예

- login/loginForm.jsp

```
<%@ page contentType="text/html;charset=utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>...</head>
<body>
    <form method="POST" action="
```

57

Servlet에서 JSP로의 forward/include

- LoginServlet.jsp

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    protected void service(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
        String id = request.getParameter("id");
        String pw = request.getParameter("password");

        LoginService loginSvc = new LoginService();
        User userInfo = loginSvc.login(id, pw);           // invoke the login service
        if (userInfo != null) {                           // login succeeds
            HttpSession session = request.getSession(); // create a new session object
            session.setAttribute("userInfo", userInfo);   // save to the session
            RequestDispatcher rd = request.getRequestDispatcher("login/loginResult.jsp");
            rd.forward(request, response);               // forwarding to the result view
        } else {                                         // login fails
            response.sendRedirect("login/loginForm.jsp"); // redirection to loginForm
        }
    }
}
```

Servlet에서 JSP로의 forward/include

- LoginService.jsp

```
public class LoginService {          // service class

    public User login(String id, String pw) {
        // process login ...
        // ex:
        // User user = userDao.findUserById(id);
        // if (user != null && user.getPassword().equals(pw)) {
        //     return user;
        // }
        // return null;

        if (id.equals(pw))           // assume that login succeeds
            return new User(id, pw, "Jain", 22, "010-3333-4444");
        return null;
    }
}
```

59

기타 Action

◆ plug-in

- 자바 애플릿을 JSP page에서 포함시켜 실행하기 위해 사용
- 형식: <jsp:plugin type="bean|applet" code="objectCode" codebase="objectCodeBase">

◆ useBean

- JavaBeans 객체를 통해 요청 파라미터 값 전달 등을 위해 사용
- 형식: <jsp:usebean id="..." class="..." scope="..." /> (bean 객체 생성)
<jsp:setProperty name="..." property="..." value="..." /> (값 저장)
<jsp:getProperty name="..." property="..." /> (값 이용)

- id/name: bean 객체의 식별자
- class: JavaBeans 클래스
- scope: 객체의 유효 범위
- property: JavaBeans 객체의 멤버 변수, value: 멤버 변수에 저장할 값

60

JavaBeans 클래스 정의

◆ JavaBeans 클래스 작성 규칙

- JavaBeans 클래스가 갖는 멤버 변수(property)는 private으로 선언
 - 예: `private String message;`
- private으로 선언한 변수의 값을 설정하기 위한 public 메소드를 선언
 - 메소드 이름은 반드시 "set변수이름" 형태로 명명
 - ✓ 변수이름의 첫 글자는 반드시 대문자
 - 예: `public void setMessage(String message) { this.message = message; }`
- private으로 선언한 변수의 값을 읽기 위한 public 메소드를 선언
 - 메소드 이름은 반드시 "get변수이름" 형태로 명명
 - ✓ 변수이름의 첫 글자는 반드시 대문자
 - 예: `public String getMessage() { return message; }`

61

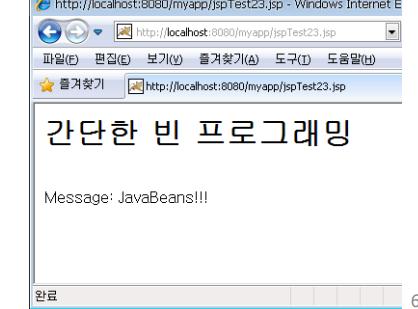
JSP에서 JavaBeans 이용

```
<%@ page contentType="text/html; charset=utf-8" %>      <!-- jspTest23.jsp -->
<jsp:useBean id="test" class="dbp.beans.SimpleBean" scope="page" />
<jsp:setProperty name="test" property="message" value="JavaBeans!!!" />
<html>
<body>
    <h1>간단한 빈 프로그래밍</h1><br>
    Message: <jsp:getProperty name="test" property="message" />
</body></html>
```

```
package dbp.beans;

public class SimpleBean {
    private String message="";

    public String getMessage() {
        return message;
    }
    public void setMessage(String m) {
        this.message = m;
    }
}
```



62

JSP에서 JavaBeans 이용

```
<%@page contentType="text/html; charset=utf-8" %>
<%@page import="user.*" %>
<jsp:useBean id="user" class="user.User"/>    <!-- User 객체를 JavaBean 객체로 생성 -->
<jsp:setProperty name="user" property="*"/>    <!-- parameter를 추출하여 user 객체에 저장 -->
<%
    UserManager.getInstance().manager.create(user);    // user 객체를 create() 호출에 전달
    response.sendRedirect("user_list.jsp");
%>
```

User 클래스가 userId, password, name, email 멤버변수(property)를
가진다고 가정하면, 아래 코드와 동일한 기능을 실행함

```
String userId = request.getParameter("userId");
String password = request.getParameter("password");
String name = request.getParameter("name");
String email = request.getParameter("email");

user.setId(userId);
user.setPassword(password);
user.setName(name);
user.setEmail(email);
```

63

JSP Standard Tag Library(JSTL)

◆ Custom Tag

- JSP에서 반복적으로 사용되는 프로그램 로직(logic)을 캡슐화(encapsulation)하여 태그 형태로 정의한 것
- JSP 페이지에서 scriptlet(즉, Java code)을 대체하기 위해 사용

◆ JSP Standard Tag Library(JSTL)

- JSP와 함께 Java EE Platform에 포함됨
- JSP 개발에서 공통적으로 사용되는 유용한 custom tag들을 정의
- MVC 구조에서 collection과 같이 여러 원소들을 포함하는 데이터를 처리할 때 특히 유용
- Expression Language(EL)와 함께 사용

64

JSP Standard Tag Library

◆ 설치

- Maven repository에서 JSTL API 및 구현체(class) library를 직접 download
 - <https://mvnrepository.com/artifact/javax.servlet/jstl/1.2>
 - jstl-1.2.jar 파일을 Web application 내의 WEB-INF/lib 폴더에 복사
- Maven dependency 설정을 통해 자동 download (권장)

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
</dependency>
```

65

JSP Standard Tag Library

◆ JSTL 구성

Library	URI	prefix
Core 기능: 변수 설정 및 제거, 흐름제어, URL 사용 등	http://java.sun.com/jsp/jstl/core	c
XML Processing 기능: 변수설정/제거, 흐름 제어, XML 변환	http://java.sun.com/jsp/jstl/xml	x
I18N capable formatting 기능: 국가에 따른 메시지 처리, 숫자/날짜 형식	http://java.sun.com/jsp/jstl/fmt	fmt
Database access(SQL) 기능: SQL query 처리	http://java.sun.com/jsp/jstl/sql	sql
Function 기능: 다양한 기능의 함수 제공	http://java.sun.com/jsp/jstl/functions	fn
Tag Library Validator 기능: translation-time validation of the XML view of a JSP page	http://java.sun.com/jstl/permittedTaglibs	

66

Expression Language(EL)

◆ Expression Language

- JSP에서 JavaBeans나 Map과 같은 Java component를 Java code(scriptlet)를 사용하지 않고 쉽게 접근하게 해 주는 언어
- standard part of JSP 2.0+
- JSP 또는 JSTL과 함께 사용됨
- \${expression} 형식을 사용하여 다음과 같은 것들을 접근 가능
 - JavaBean properties
 - Map, List, 또는 배열(array)의 원소
 - Servlet 내장 객체의 속성

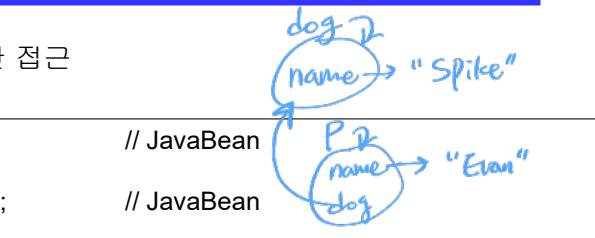
67

Expression Language

◆ 예 1: JavaBean에 대한 접근

- Servlet

```
Dog dog = new Dog();           // JavaBean
dog.setName("spike");
Person p = new Person();        // JavaBean
p.setName("Evan");
p.setDog(dog);                 // Person은 Dog 타입의 property를 포함
request.setAttribute("person", p); // request 객체에 저장 후 forwarding
request.getRequestDispatcher("result.jsp").forward(request, response);
```



- result.jsp

```
#{person.name} 또는 #{person["name"]}
→ <%= ((Person)request.getAttribute("person")).getName()%>과 동일

#{person.dog.name}
→ <%= ((Person)request.getAttribute("person")).getDog().getName()%>
```

68

Expression Language

◆ 예 2: Map에 대한 접근

- Servlet

```
mMap  
java.util.Map mMap = new java.util.HashMap();  
musicMap.put("Ambient", "Zero 7");  
musicMap.put("Surf", "Tahiti 80");  
musicMap.put("DJ", "BT");  
musicMap.put("Indie", "Frou Frou");  
  
request.setAttribute("musicMap", mMap);  
request.setAttribute("Genre", "Ambient"); → "Genre" 2개의 이름으로 "Ambient" 차지  
request.getRequestDispatcher("result.jsp").forward(request, response);
```

- result.jsp



```
 ${musicMap["Ambient"]} 또는 ${musicMap[Genre]} // = Zero 7
```

69

Expression Language

◆ EL 내장 객체

- pageScope, requestScope, sessionScope, applicationScope
- param, paramValues
- header, headerValues, pageContext
- cookie
- initParam

◆ pageScope, requestScope, sessionScope, applicationScope

- \${name}: page, request, session, application 순으로 검색해서 찾음
- \${requestScope.name}: request에서만 검색
 - <%=request.getAttribute("name") %> 와 동일
- \${sessionScope.name }: session에서만 검색
 - <%=session.getAttribute("name") %> 와 동일

Expression Language

◆ 예 3: 배열 또는 List에 대한 접근

- Servlet

```
String[] favoriteMusic = {"Zero 7", "Tahiti 80", "BT", "Frou Frou"};  
// 또는  
// List favoriteMusic = new ArrayList();  
// favoriteMusic.add("Zero 7");  
// favoriteMusic.add("Tahiti 80");  
// ...  
  
request.setAttribute("musicList", favoriteMusic);  
request.getRequestDispatcher("result.jsp").forward(request, response);
```

- result.jsp



```
 ${musicList} // 모든 원소들을 출력  
 ${musicList[0]} // = Zero 7  
 ${musicList["1"]} // = Tahiti 80
```

70

Expression Language

◆ param, paramValues: request parameter를 접근

- 예

```
<form action="TestBean.jsp">  
  <input type="text" name="name">  
  <input type="text" name="empID">  
  <input type="checkbox" name="food">  
  <input type="checkbox" name="food">  
</form>
```



```
 ${param.name} // name 파라미터의 값  
 ${param.empID}  
 ${param.food}  
 ${paramValues.food[0]}  
 ${paramValues.food[1]}  
 ${paramValues.name[0]} // ${param.name}과 동일
```

72

71

Expression Language

- ◆ header: request header 정보 접근
 - 예: \${header.host} 또는 \${header["host"]} // host header
- ◆ pageContext: pageContext 내장 객체 접근
 - 예: \${pageContext.request.method} // http request method
\${pageContext.request.contextPath} // servlet context path
- ◆ initParam: 애플리케이션의 컨텍스트 초기화 파라미터 접근
 - 예:

```
<!-- web.xml -->
<context-param>
    <param-name>breed</param-name>
    <param-value>Great Dane</param-value>
</context-param>
```

 - \${initParam.breed} or <%=application.getInitParameter("breed") %>
→ bread 파라미터의 값 참조

73

JSTL Core Library

◆ Tag Summary

기능	태그	설명
Variable Support Actions	set	<ul style="list-style-type: none">· JSP의 setAttribute()와 같은 역할· (page request session application) 범위의 변수(속성)를 설정
	remove	<ul style="list-style-type: none">· JSP의 removeAttribute()와 같은 역할· (page request session application) 범위의 변수(속성)를 제거
Iterator Actions	forEach	<ul style="list-style-type: none">· 객체 전체에 걸쳐 반복 실행시 사용
	forTokens	<ul style="list-style-type: none">· 자바의 StringTokenizer 클래스와 동일
Conditional Actions	choose	<ul style="list-style-type: none">· 자바의 switch문과 같지만, 조건에 문자열 비교도 가능· 한 개 이상의 <when>과 한 개의 <otherwise> 서브 태그를 가짐
	when	<ul style="list-style-type: none">· <choose>의 서브 태그로, 특정 조건을 만족하는 경우 사용
	otherwise	<ul style="list-style-type: none">· <choose>의 서브 태그로, 조건을 만족하지 못하는 경우 사용
	if	<ul style="list-style-type: none">· 하나의 조건문 표현

74

JSTL Core Library

◆ Tag Summary (계속)

기능	태그	설명
URL Related Actions	import	<ul style="list-style-type: none">· 웹 애플리케이션 내부의 자원과 http, ftp와 같은 외부에 있는 자원을 가져옴
	redirect	<ul style="list-style-type: none">· HttpServletRequest#sendRedirect()를 대체하는 태그로, 지정된 다른 페이지로 redirection을 지시
	url	<ul style="list-style-type: none">· query parameter로부터 URL을 생성함.
	param	<ul style="list-style-type: none">· <import> 태그 사용 시 parameter 전달을 위해 사용
Miscellaneous Actions	out	<ul style="list-style-type: none">· JSP 표현식을 대체하기 위해 사용됨
	catch	<ul style="list-style-type: none">· body 위치에서 실행되는 코드의 예외를 잡아냄

75

Variable Support Tags

◆ <c:set> Tag

- Sets the value of a scoped variable or a property of a target object
 - 변수: var와 value 속성 이용
 - JavaBean이나 Map: target과 property 속성 이용
- 변수의 scope: page, request, session, application 순으로 탐색

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:set var="userLevel" scope="session" value="Cowboy"></c:set>
userLevel : ${userLevel}<br>                                // Cowboy

<c:set var="Fido" value="${person.dog}"></c:set>      // ${person.dog}의 값
<c:set var="userLevel" scope="session">
    Sheriff, Bartender, Cowgirl
</c:set>
userLevel : ${userLevel}<br>                                // Sheriff, Bartender, Cowgirl

<c:set target="${person}" property="name">$<foo.name></c:set>
<c:set target="${PetMap}" property="dogName" value="Clover"></c:set>
```

76

Variable Support Tags

- ◆ <c:remove> Tag
 - Removes a scoped variable

```
<c:set var="userStatus" scope="request" value="Brilliant" />
userStatus : ${userStatus} <br> // Brilliant

<c:remove var="userStatus" scope="request" />
userStatus : ${userStatus} <br> // 값 없음
```

77

Iteration Tags

2. Looping over data structures

- Array

```
<c:forEach var="name" items="${arrayOrCollection}">
    ${name} // 원소를 하나씩 출력
</c:forEach>
```

- Collection

```
<c:forEach var="bean" items="${collectionOfBeans}">
    ${bean.property} // 원소인 bean의 property 값 출력
</c:forEach>
```

bean의 각각의 주소가 출력된다.

79

Iteration Tags

1. Looping with explicit numeric values

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<c:forEach var="val" begin="x" end="y" step="z"> // x, y, z는 숫자
    ${val}
</c:forEach> // 숫자값들을 차례로 이용
```

```
<UL>
    <c:forEach var="i" begin="1" end="10">
        <LI>${i}</LI>
    </c:forEach>
</UL>
```

```
<UL>
    <c:forEach var="seconds" begin="0"
        end="${pageContext.session.maxInactiveInterval}" step="100">
        <LI>${seconds} seconds. </LI>
    </c:forEach>
</UL>
```

78

Iteration Tags

- JSP without JSTL

```
<UL>
<% String[] messages = (String[])request.getAttribute("messages");
    for (int i=0; i < messages.length; i++) {
        String message = messages[i];
    }
    <LI><%=message%></LI>
<% } %>
</UL>
```



더 간결

- JSP with JSTL

```
<UL>
    <c:forEach var="message" items="${messages}">
        <LI>${message}</LI>
    </c:forEach>
</UL>
```

80

Iteration Tags

- Example: Looping Down Array or List

- Servlet

```
public class ArrayServlet extends HttpServlet {  
    public void public void doGet(HttpServletRequest request, ...) throws ServletException, IOException {  
  
        String[] words = {"foo", "bar", "baz";  
        // 또는 List<String> words = Arrays.asList("foo", "bar", "baz");  
        request.setAttribute("words", words);  
        String address = "/WEB-INF/result/array-loop.jsp";  
        RequestDispatcher dispatcher =  
            request.getRequestDispatcher(address);  
        dispatcher.forward(request, response);  
    }  
}
```

81

Iteration Tags

- Example: Looping Down Array or List

- JSP (/WEB-INF/result/array-loop.jsp)

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<H2>Key Words:</H2>  
<UL>  
    <c:forEach var="word" items="${words}">  
        <LI>${word} </LI>  
    </c:forEach>  
</UL>  
  
<H2>Values of the test Parameter:</H2>  
<UL>  
    <c:forEach var="val" items="${paramValues.test}">  
        <LI>${val} </LI>  
    </c:forEach>  
</UL>
```

82

Iteration Tags

- Example: Accessing Sub-elements
- JavaBean

```
public class Name {  
    private String firstName;  
    private String lastName;  
  
    public Name(String firstName, String lastName) {  
        this.firstName = firstName; this.lastName = lastName;  
    }  
    public String getFirstName() { return(firstName); }  
    public void setFirstName(String firstName) {  
        this.firstName = firstName;  
    }  
    public String getLastname() { return(lastName); }  
    public void setLastName(String lastName) {  
        this.lastName = lastName;  
    }  
}
```

83

Iteration Tags

- Example: Accessing Sub-elements
- Servlet

```
public class ArrayServlet2 extends HttpServlet {  
    public void public void doGet(HttpServletRequest request, ...) throws ServletException, IOException {  
        Name[] names = { new Name("Bill", "Gates"),  
                        new Name("Larry" "Ellison"),  
                        new Name("Sam", "Palmisano"),  
        request.setAttribute("names", names);  
        String[][] sales = { {"2005", "12,459", "15,622"},  
                            {"2006", "18,123", "17,789"},  
                            {"2007", "21,444", "23,555"} };  
        request.setAttribute("sales", sales);  
        String address = "/WEB-INF/result/array-loop2.jsp";  
        request.getRequestDispatcher(address).forward(request, response);  
    }  
}
```

84

Iteration Tags

- Example: Accessing Sub-elements

- JSP

```
<!-- /WEB-INF/result/array-loop2.jsp -->
<UL>
  <c:forEach var="name" items="${names}" >
    <LI>${name.firstName} ${name.lastName} </LI>
  </c:forEach>
</UL>
<H2>Comparing Apples and Oranges</H2>
<TABLE BORDER="1">
  <TR><TH>Year</TH> <TH>Apples Sold</TH> <TH>Oranges Sold</TH>
  </TR>
  <c:forEach var="row" items="${sales}">
    <TR>
      <c:forEach var="col" items="${row}">
        <TD>${col}</TD>
      </c:forEach>
    </TR>
  </c:forEach>
</TABLE>
```

85

→ 2년간 매출이라서

Iteration Tags

- ◆ Loop Status

- varStatus 속성 이용

```
<c:forEach var="name" items="${names}" varStatus="status" >
```

- Status sub-properties

- index (int: the current index)
 - first (boolean: is this the first entry?)
 - last (boolean: is this the last entry?)
 - begin (Integer: value of 'begin' attribute)
 - end (Integer: value of 'end' attribute)
 - step (Integer: value of 'step' attribute)

Iteration Tags

- Example

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%
  String[] names = {"Joe", "Jane", "Juan", "Juana"};
  request.setAttribute("names", names);
%>

Names:
<c:forEach var="name" items="${names}" varStatus="status">
  ${name}<c:if test="${!status.last}">, </c:if>
</c:forEach>
```

마지막 처리가 아니면 ,

출력 결과 - Names: Joe, Jane, Juan, Juana

87

Iteration Tags

3. Looping down delimited strings

- Looping Down Comma-Delimited Strings

```
<UL>
  <c:forEach var="country"
    items="Australia,Canada,Japan,Philippines,Mexico,USA" >
    <LI>${country} </LI>
  </c:forEach>
</UL>
```

- Looping Down Arbitrarily-Delimited Strings (<c:forTokens>)

```
<UL>
  <c:forTokens var="color"
    items="(red (orange) yellow)(green)((blue) violet)" delims="()" >
    <LI>${color} </LI>
  </c:forTokens>
</UL>
```

86

88

Conditional Evaluation Tags

- ◆ One choice: <c:if> Tag

```
<c:if test="${someTest}">  
    Content  
</c:if>
```

- example

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>  
<UL>  
    <c:forEach var="i" begin="1" end="10">  
        <LI>${i}  
        <c:if test="${i > 7}">  
            (greater than 7)  
        </c:if>  
        </LI>  
    </c:forEach>  
</UL>
```

89

Conditional Evaluation Tags

- ◆ Lots of choices: <c:choose> Tag

```
<c:choose>  
    <c:when test="test1">Content1</c:when>  
    <c:when test="test2">Content2</c:when>  
    ...  
    <c:when test="testN">ContentN</c:when>  
    <c:otherwise>Default Content</c:otherwise>  
</c:choose>
```

- Caution: resist use of business logic!

Conditional Evaluation Tags

- 비교 연산자
 - eq: equal (==)
 - ne: not equal (!=)
 - empty: list, set, map 등 collection 객체가 원소를 갖고 있는지 여부 검사
- 사용 예

```
<c:if test="${ name eq null }">          // null 인지 검사  
<c:if test="${ name eq 'scott' }">        // 문자열 'scott' 인지 검사  
<c:if test="${ number eq 10 } " >          // 숫자10 인지 검사
```

```
<c:if test="${ name ne null }">          // null 이 아닌지 검사  
<c:if test="${ name ne 'scott' }">        // 문자열 'scott' 이 아닌지 검사  
<c:if test="${ number ne 10 }">
```

```
<c:if test="${ empty userList }">          // list가 비어있는지 검사  
<c:if test="${ !empty userMap }">           // map이 원소가 있는지 검사
```

91

<c:out> Tag

- ◆ 식 계산 및 결과 화면 출력

```
<UL>  
    <c:forEach var="word" items="${words}">  
        <LI><c:out value="${word}" /></LI>  // == ${word}  
    </c:forEach>  
</UL>
```

- ◆ c:out escapes HTML (XML) characters

- <c:out value="x > 10 and y < 20"/> → x > 10 and y < 20 출력
- Disable with <c:out value="

" escapeXml="false"/> → <h1> 출력
- ◆ c:out lets you supply a default value *(태그를 그대로 출력하고 넣으면)*
 - <c:out value="\${foo}" default="explicit value"/> or
<c:out value="\${foo}" default="\${calculatedValue}"/>
 - The default is output when \${foo} evaluates to null

92

JSTL Functions

◆ Tag Summary

기능	태그	기능
Collection length	<code>length</code>	Returns the length of the collection (when applied to a collection supported by the <code>c:forEach</code>) or the number of characters in the string (when applied to a String)
String manipulation	<code>toUpperCase</code> , <code>toLowerCase</code>	Changes the capitalization of a string
	<code>substring</code> , <code>substringBefore</code> , <code>substringAfter</code>	Gets a subset of a string
	<code>trim</code>	Trims white space from a string
	<code>replace</code>	Replaces characters in a string
	<code>indexOf</code> , <code>startsWith</code> , <code>endsWith</code> , <code>contains</code> , <code>containsIgnoreCase</code>	Checks whether a string contains another string
	<code>split</code>	Splits a string into an array
	<code>join</code>	Joins a collection into a string
	<code>escapeXml</code>	Escapes XML characters in a string

JSTL Functions

◆ Examples

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn"%>
<%@ page import="java.util.ArrayList"%>
<%
    ArrayList<String> numList = new ArrayList<String>();
    numList.add("one"); numList.add("two"); numList.add("three");
    request.setAttribute("numList", numList);
    request.setAttribute("str", "This is a test string");
%>
<html>
<body>
    The length of the numList: ${fn:length(numList)}<br/> -- 3
    The length of the test String: ${fn:length(str)}<br/> -- 21
    ${fn:contains(str, "test")}<br/> -- true
    ${fn:indexOf(str, "test")}<br/> -- 10
    ${fn:substring(str, 8, 20)}<br/> -- a test string
    <c:set var="array" value="${fn:split(str, ' ')}" />
    ${fn:join(array,'|')}<br/> -- This|is|a|test|string
</body>
</html>
```

94

References

- Introduction to HTTP Basics
 - https://www3.ntu.edu.sg/home/ehchua/programming/webprogramming/HTTP_Basics.html
- Official Tutorial: The Java EE 5 Tutorial, Chapter 7, JavaServer Pages Standard Tag Library
 - <http://docs.oracle.com/javaee/5/tutorial/doc/bnkc.html>
- Servlet API
 - <https://docs.oracle.com/javaee/6/api/javax/servlet/http/package-summary.html> (HttpServletRequest, HttpServletResponse, HttpSession, ...)
- JSTL in Oracle Technology Network
 - <https://www.oracle.com/java/technologies/Java-server-tag-library.html>
- JSTL References (JavaDoc)
 - <http://download.oracle.com/javaee/5/jstl/1.1/docs/tlddocs/>
- JSTL Functions
 - <http://www.javatips.net/blog/2011/10/jstl-functions>

