**[ 5 ] JSP 웹프로그래밍**

목표 : 내부객체, JSP 태그, 동작원리,

1. JSP 내부 객체
   * 개발자가 객체를 생성하지 않고 바로 사용할 수 있는 객체가 내부객체.
   * JSP에서 제공되는 내부객체는 JSP컨테이너에 의해 Servlet으로 변화될 때 자동으로 객체가 생성
   * 대표적인 내부 객체
     1. request
     2. response,
     3. out
2. JSP 태그의 개념 이해
   * Servlet은 JAVA언어를 이용하여 문서를 작성하고, 출력객체(PrintWriter객체)를 이용하여 HTML코드를 삽입한다
   * JSP는 Servlet과 반대로 HTML코드에 JAVA언어를 삽입하여 동적 문서를 만들 수 있다..
   * HTML코드안에 JAVA코드를 삽입하기 위해서는 다음과 같은 태그를 이용한다
     1. 지시자 : <%@ %> : 페이지 속성 ex.import, page
     2. 주석 : <%-- --%> ; 응답 페이지에서는 이 주석은 보이지 않는다. WAS에서 이 주석은 제거하고 응답페이지를 만든다. cf. HTML 주석과 JSP 주석과 비교해 보자
     3. 선언 : <%! %> : 전역변수 및 메소드 선언
     4. 표현식 : <%= %> : 결과값 출력
     5. 스크립트릿 : <% %> : JAVA 코드
     6. 액션태그 : <jsp:action></jsp:action> : 표준액션태그 ex.파일연결 및 자바빈 연결

<c:set></c:set> : 커스텀액션태그

1. JSP 내부 객체
   1. request 객체 ; 웹브라우저를 통해 서버에 어떤 정보를 요청하는 것을 request라 하며, 이러한 요청 정보는 request 내부객체가 관리한다. 다음은 request 객체 메소드.
      * + 객체 타입 : javax.servlet.http.HttpServletRequest
2. getRemoteAddr(); 웹서버에 연결한 클라이언트의 IP주소
   * + IP 체크 할일이 생겨서 검색을 했더니 <%=request.getRemoteAddr()%> 결과가 0:0:0:0:0:0:0:1 만 나온다.
     + WIN7 이상은 기본이 IPV6
     + 그래서 톰캣 실행시 JVM 환경변수를 추가 해야 한다고 한다.
     + 0:0:0:0:0:0:0:1 이거랑 127.0.0.1 이랑 같으나 IPv4를 원한다면 아래의 과정을 거친다.
     + eclipse > Run > Run Configuration > Apache Tomcat > Tomcat v10.1 Server >
     + Arguiments >
     + Program arguments에 “-Djava.net.preferIPv4Stack=true”입력
     + VM arguments:에 "-Djava.net.preferIPv4Stack=true" 입력
     + 톰캣 재 실행후 재 확인 하면 제대로 나온다.

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

<style>

**td** {text-align: *center*;}

</style>

</head>

<body>

<form action=*"memo.jsp"* method=*"post"*>

<table>

<tr><td>작성자</td><td><input type=*"text"* name=*"name"* required=*"required"*></td></tr>

<tr><td>한줄평</td><td><input type=*"text"* name=*"memo"* required=*"required"*></td></tr>

<tr><td colspan=*"2"*><input type=*"submit"* value=*"메모작성"*></td></tr>

</table>

</form>

</body>

</html>

<%@page import=*"java.util.Date"*%>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<% request.setCharacterEncoding("utf-8"); %>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<%! String name, memo, ip, date; %>

<%

name = request.getParameter("name");

memo = request.getParameter("memo");

ip = request.getRemoteAddr();

date = **new** Date().toLocaleString();

%>

<h3>이름 <%=name %></h3>

<h3>한줄평 <%=memo %></h3>

<h3>ip <%=ip %></h3>

<h3>쓴시간 <%=date %></h3>

</body>

</html>

1. request.getContextPath() : contextPath를 리턴
2. getParameter(String name) : name에 해당하는 파라미터 값을 구함.
3. getParameterValues(String name) : name에 해당하는 파라미터값들을 구함.
4. getContextPath() : 웹어플리케이션의 컨텍스트 패스(http://localhost:8181**/prjName**/folderName/ex.jsp중 **/prjName**)를 얻습니다.
5. getMethod() : get방식과 post방식을 구분할 수 있습니다.
6. getRequestURL() : 요청 URL를 얻습니다.
7. getRequestURI() : 요청 URI를 얻습니다.
8. 위의 메소는 서블릿에서도 사용 가능

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response) **throws** ServletException, IOException {

String conPath = request.getContextPath();

String uri = request.getRequestURI();

String command = uri.substring(conPath.length());

response.getWriter().append("<h2>conPath :"+conPath+"</h2>");

response.getWriter().append("<h2>uri :"+uri+"</h2>");

response.getWriter().append("<h2>command :"+command+"</h2>");

}

1. Parameter 메소드 : getParameter(String name); name에 해당하는 파라미터 값 구함

getParameterValues(String name);파라미터 값들 구함

* 1. response 객체 ; 웹브라우저의 요청에 응답하는 것을 response한다고 하며, 이러한 응답(response)의 정보는 response 내부객체가 갖고 있다. doGet, doPost 메서드의 두번째 파라미터와 동일
     + - 객체타입 : javax.servlet.http.HttpServletResponse

sendRedirect(URL) : 지정한 URL로 이동

ageInput.jsp → agePro.jsp → pass.jsp

ng.jsp

성인인증 로직

age.html

<body>

<form action=*"age.jsp"*>

나이 <input type=*"number"* name=*"age"*>

<input type=*"submit"*>

</form>

</body>

age.jsp

<body>

<%

String str = request.getParameter("age");

**int** age = Integer.parseInt(str);

**if**(age>=20)

response.sendRedirect("pass.jsp?age="+age);

**else**

response.sendRedirect("ng.jsp?age="+age);

%>

</body>

pass.jsp

<body>

<%

String str = request.getParameter("age");

**int** age = Integer.parseInt(str);

%>

<h2><%=age %>살 성인이니 주류구매 가능합니다</h2>

</body>

ng.jsp

<body>

<%

String str = request.getParameter("age");

**int** age = Integer.parseInt(str);

%>

<h2><%=age %>살 성인이 아니니, 주류판매가 불가합니다</h2>

<button onclick="history.go(-1)">처음으로</button>

</body>

* 1. out : 웹 브라우저로 HTML 코드를 출력하는 기능. PrintWriter처럼 스트림 형태로 데이터를 출력하는 클래스는 송신측과 수신측 사이에 가상의 통로를 만든다. out 내장 객체는 이 통로를 관리하는 객체 변수.
     + - * 객체 타입 : javax.servlet.jsp.JspWriter

1. JSP 태그들
   1. 스크립트릿, 선언, 표현식 ; JSP문서안에 JAVA언어를 넣기 위한 방식들 -> 이 세개를 묶어 scriptlet이라 한다
      1. 스크립트릿(scriptlet) : <% java 코드 %>

**예제> 원하는 배경색을 select 태그를 이용하여 입력받아 배경색을 바꾸도록 jsp페이지를 구현.**



<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

<%

String color = request.getParameter("color");

**if**(color==**null**){

color="white";

}

%>

<style>

**body** {background-color: <%=color%>;}

</style>

</head>

<body>

<h1>배경색 바꾸기</h1>

<form action=*"#"* method=*"get"*>

<select name=*"color"*>

<option value=*"red"*>빨강</option>

<option value=*"orange"*>주황</option>

<option value=*"yellow"*>노랑</option>

<option value=*"green"*>초록</option>

<option value=*"blue"*>파랑</option>

<option value=*"navy"*>남색</option>

<option value=*"violet"*>보라</option>

</select>

<input type=*"submit"* value=*"해당 배경색 창으로"*>

</form>

</body>

</html>

* + 1. 선언(declaration) : <%! java 코드 기술 %>

변수 또는 메소드 선언할 때 사용. 여기서 선언된 변수 및 메소드는 페이지 어느곳에서나 사용가능한 전역의 의미로 사용된다.

* + 1. 표현식(expression) : <%= java 코드 기술 %>

JSP 페이지 내에서 사용되는 변수의 값 또는 메소드 호출 결과값을 출력하기 위해 사용.

;를 사용하지 않는다.

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

<style>

</**head**>

<**body**>

<!-- **HTML** **주석** -->

<%-- JSP 주석 --%>

<% // 스크립트릿 안에 자바 소스

**int** j=0; // 초기화하지 않으면 쓰레기 데이터

String localStr = "몰라요";

i++; j++;

globalStr += "<span> ♥ </span>";

localStr += "<span> @ </span>";

out.println("전역변수 i : " + i + "<br>");

out.println("전역변수 globalStr = " + globalStr + "<br>");

out.println("지역변수 j : " + j + "<br>");

out.println("지역변수 localStr : " + localStr + "<br>");

out.println("sum(3,4) = " + sum(3,4));

%>

<%!

**int** i;

String globalStr = "Good";

**private** **int** sum(**int** a, **int** b){

**return** a+b;

}

%>

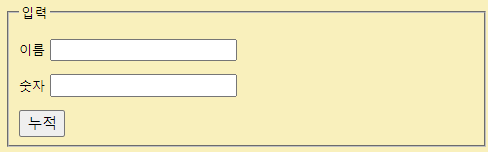
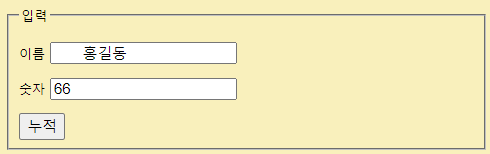
</bod**y>**

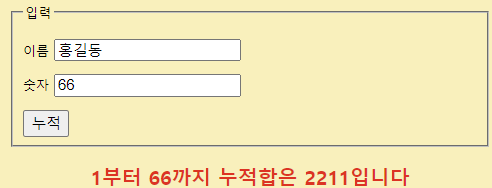
</ht**ml>**

**실습)숫자를 입력받아 아래에 누적합을 출력하는 jsp컴포넌트를 구현하시오. jsp 첫실행화면은 바로 아래 첫번째 캡처화면과 같고,**

**두번째 화면과 같이 입력할 경우,**

**“누적”버튼을 클릭하면 세번째 화면과 같이 실행되도록 구현하시오. 이름에 “ 홍길동 ”를 입력시 “홍길동”으로 처리**



* + 1. 지시자

; JSP페이지의 전체적인 속성을 지정할 때 사용. page, include, taglib 가 있으며, <%@ 속성 %>형태

* + - * page : 해당 페이지의 전체적인 속성 지정. 주로 사용되는 언어 지정 및 import문을 많이 사용, 문서 타입(html, excel..)
      * include : 별도의 페이지를 현재 페이지에 사용 (footer 등 모든 페이지에서 반복되는 작업들에 이용)
      * taglib : 태그라이브러리(사용자가 만든 tag들)의 태그 사용. 태그라이브러리를 사용하기 위해 taglib지시자 사용. taglib 지시자에 대한 학습은 추후에 살펴볼 JSTL학습 때 사용될 예정.

<%@page import=*"java.sql.Date"*%>

<%@page import=*"java.util.Arrays"*%>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<%

**int**[] iArr = {10,20,30};

out.println(Arrays.toString(iArr));

System.out.println(Arrays.toString(iArr));

%>

<hr>

<%@include file=*"../member/footer.jsp"* %>

<hr>

<% Date now = **new** Date(System.currentTimeMillis()); %>

현재는 <%=now %>입니다

</body>

</html>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"UTF-8"*>

<title>Insert title here</title>

<link href=*"../css/footer.css"* rel=*"stylesheet"*>

</head>

<body>

<footer>

<div class=*"office\_logo"*>

<img src=*"https://www.google.com/images/branding/googlelogo/1x/googlelogo\_color\_272x92dp.png"*>

</div>

<div class=*"office\_address"*>

<ul>

<li>서울시 XX구 XX로 000 XXX빌딩 8F | TEL : 02-999-9999 | FAX : 02-999-9998</li>

<li>사업자등록번호 : 000-12-00000 | 통신판매업신로 : 종로 제0000호 | 대표이사 : 홍길동</li>

<li>좋은시스템 문의 메일 : webmaster@joeun.com</li>

</ul>

</div>

<div class=*"copyright"*>

ⓒ2022 JOEUNSYSTEM.Co.,Ltd. All Rights Reserved.

</div>

</footer>

</body>

</html>

@charset *"UTF-8"*;

**footer** {

overflow:*hidden*;

width: *1000px*;

margin: *0* *auto*;

}

**footer** **div** {

float:*left*;

height: *90px*;

}

**footer** **div***.office\_logo* {

width: *300px*;

text-align: *center*;

}

**footer** **div***.office\_logo* **img** {

margin-left:*100px*;

width:*150px*;

height: *60px*;

padding:*20px*;

}

**footer** **div***.office\_address*{

width:*700px*;

}

**footer** **div***.office\_address* **ul**{

margin: *15px* *20px* *20px* *20px*;

}

**footer** **div***.office\_address* **ul** **li**{

padding:*2px*;

color:*gray*;

font-size: *0.8em*;

list-style: *none*;

}

**footer** **div***.copyright*{

clear: *both*;

border: *0*;

width: *1000px*;

height: *25px*;

margin: *0* *auto*;

background-color: *#D4A190*;

color: *white*;

text-align: *center*;

line-height: *25px*;

font-size: *0.9em*;

}

* + 1. 주석 : 실제 프로그램에는 영향이 없고, 프로그램 설명들의 목적으로 사용되는 태그입니다

<%-- comments -->, //, /\* \*/

<body>

<h3>주석 4가지 다른 거 한번 볼까</h3>

<!-- 여기는 HTML 주석을 쓴 곳 -->

<%-- 여기는 JSP 주석을 쓴 곳 --%>

<% // 여기는 스크립트릿 안 주석 쓴 곳 %>

<% /\* 여기는 스크립트릿

\* 안 주석 쓴 곳

\*/

%>

</body>

1. JSP 동작 원리
   * 클라이언트가 웹브라우저로 a.jsp를 요청하게 되면 **JSP컨테이너가 JSP파일을 Servlet파일**(a\_jsp.java)로 변환된다. 그리고 Servlet파일(.java)은 컴파일 된 후 클래스 파일(.class)로 변환되고, 요청한 클라이언트한테 html파일 형태로 응답한다.

**= [웹 브라우저 요청] → [a.jsp 를 a\_ jsp.java로 변환] → [a\_ jsp.java를 a\_ jsp.class로 변환] → [웹브라우저에 HTML 형태로 응답]**

jsp요청이 들어오면 Servlet이 있는지 없는지 보고 없으면 만들고 .class를 만들고 메모리에 올리고 실행되고 응답해준다. 그 후 또 요청이 들어오면 이미 자바와 클래스 객체가 있으므로 그 객체 재사용

|  |
| --- |
| 폴더들  소스 : d:/ai/source/06\_jsp/ch05\_jsp/src/main/webapp/ex/a.jsp  (내가 이클립스로 작업하면서 저장)  http://localhost:8181/ch05\_jsp/ex/a.jsp 로 요청  1. d:/ai/IDE/톰캣폴더/wtpwebapps/ch05\_jsp/웹소스  (\*.html, \*.jsp, \*.js, \*.css, \*.xml)  (실행하는 소스 : 톰캣이 웹소스들을 옮겨 저장)  2. d:/ai/IDE/톰캣폴더/work/~/a\_jsp.java 전환  3. d:/ai/IDE/톰캣폴더/work/~/a\_jsp.class컴파일  4. 메모리에 객체 생성 |

|  |
| --- |
| <내가 웹서버에 배포 : 나의 톰캣의 webapps 폴더에 x라는 이름의 context root로 배포>  1. 톰캣의 webapps 폴더 밑에 x폴더 생성  2. 소스폴더의 webContent폴더(D:\ai\source\06\_jsp\ch05\_jsp\webapp)에 있는 모든 내용을 1에서 생성한 x폴더 밑에 복사  3. 소스폴더의 build(D:\ai\source\06\_jsp\ch05\_jsp\build)밑에 있는 classes폴더를 2에서 생성된 x/WEB-INF밑에 복사  4. 톰캣 startup (bin/startup) (反)shutdown  5. 브라우저의 url에 다음을 입력하여 실행 : http://localhost:8090/x  <war파일을 이용해서 웹서버에 배포 : 나의 톰캣의 webapps 폴더에 y라는 이름의 context root로 배포>  1. 프로젝트 오른쪽마우스 클릭 -> Export - WAR file - Destination 선택하여 war 생성(webapps폴더로)  2. 톰캣 startup  3. 브라어저의 url에 다음을 입력하여 실행 : http://localhost:8090/war파일명 |