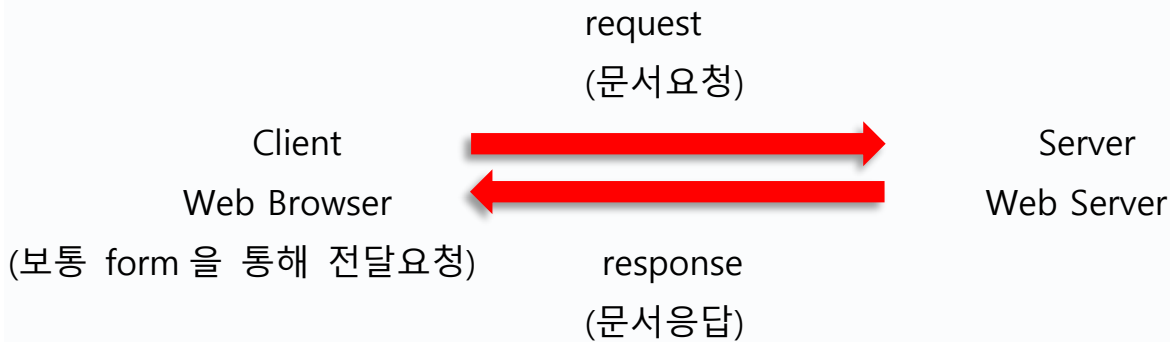


00. Web 의 작동 구조



∴ 클라이언트(유저)가 링크를 **클릭**하면 서버로 **요청**이 전달되고 서버는 그 요청에 따른 **응답**을 해준다.

01. JSP 사용법 및 배포방법

1) JSP (Java Server Page)

: 서블릿 기반 위에 보다 편리하게 웹 프로그래밍을 할 수 있도록 만든 **동적 웹 페이지 작성 언어(Dynamic Web)** → 웹 + 자바 + 소스코드

→ HTML5 문서 + 선언문이 추가된 문서형태 / 확장자는 ~.jsp

※ cf) 선언문

```
<%@ page language="java" contentType="text/html; charset=utf-8" pageEncoding = "utf-8" %>
```

위 예시와 같이 <% %>로 java 의 코드가 묶여있는 구문

2) 주소체계

<http://localhost:8080/JspStudy/hello.jsp>

프로토콜 : localhost(혹은 도메인):접속할 포트명/프로젝트명/요청받아 보여줄 문서명

3) JSP 4 대 구성요소

(1) **스크립트릿(ScriptLet) : <% java code %>**

① 웹상에서 자바코드를 사용할 수 있도록 해주는 영역

→ 간단한 변수(=지역변수), 간단한 제어문

② html 태그를 사용할 수 없다(자바스크립트 구문도 마찬가지로)

③ 스크립트릿 내부에서는 표현식(<%= %>) 사용이 안된다

④ JSP 내부에 여러군데 선언해서 사용이 가능하다.

(다만, 순차적으로 선언되어야한다. 뒤에 선언된 변수를 앞에서 사용할 수 없다)

(2) 표현식(Expression) : <%= %>

① 간단하게 출력문 대용으로 사용 ➔ 자주 사용하거나 반복되는 내용

② 사용목적 : 변수값 출력, method 호출

③ 형식)

<%=변수명 %>

<%=객체명.일반 method 명 %>

<%=정적 method 명 %>

※ 표현식 안쪽의 method 명 뒤에 ; (semi-colon)을 쓰면 안된다.

```
/abc/imsi.jsp
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
</head>
<% //int count = 3; // ScriptLet이 어디에 있건, 나뉜 Scriptlet 들은 1개의 문서로 인식된다....?%>
<body>
스크립트위에 출력하는 count: <%=count%> // 표현식
<%
    //int count =3;
    for(int i=0; i<count;i++){
        out.println("<h1>JSP 테스트"+i+"!</br>");
        //document.write("<h1>JSP 테스트"+i+"!</br>");
    }
    out.print("count: "+count);
    //!=count ScriptLet 내부에는 Expression을 쓸 수 없다.
%>
스크립트 아래에 출력하는 count: <%=count%>
<%
    //int count = 3; // 하지만 '순차적'으로 실행되기 때문에 어떠한 변수를 넣으려면 앞에 선언을
    해주어야한다.
%>
<%!
//선언된 위치에 상관없이 페이지세 선언만 해놓으면 언제든지 그 페이지에서 불러다 사용이 가능한
//변수 또는 [메소드]를 선언한 영역
//static 느낌
int count =3;
// Declaration(선언문) 형식으로, 선언 위치에 상관없이 어디서든 불러서 사용할 수 있게 해주는 형식%>
</body>
</html>
```

(3) 선언문(Declaration) : <%! %>

- ① 자바코드 사용이 가능한 영역. → 마치 정적멤버변수를 선언하는 느낌
- ② *** 선언된 위치에 상관없이 이 영역에 선언된 변수를 불러다 사용이 가능하다 → 스크립트릿과 선언문의 차이점
- ③ ** 내부에 method 작성이 가능 → 현실적으로 잘 사용하지 않음

```
<!DOCTYPE html>
<%! //Declaration
    //변수와 메소드를 이 jsp 문서내의 어디서든 불러서 사용하고 싶다
    String name = "홍길동"; //static String name 같은 느낌
    // 메소드는 따로 뽑아내서 클래스를 작성 -> 메소드를 불러와서 작업 -> 빈즈 클래스
    public String getName(){ // 내부 public String getName(){return name}
        return name;
    }
%>
<html>
```

(4) 주석(Comment) 또는 지시어 : JSP 주석 (보이지 않는 주석 <%-- --%>

4-1) 주석

① 주석의 종류

눈에 보이는 주석 : <!-- -->

눈에 보이지 않는 주석 : <%-- --%>

자바 주석 : // , /* */

② 주석 사용시 주의할 점

주석 내부에 '표현식 <%= %>' 사용이 가능

```
<html>
<!-- JSP 주석 연습 페이지 -->
<%-- 눈에 안보이는 주석입니다. (외부에서 접속하는 유저(클라이언트)에게는 보이지 않는다.) --%>
<head>
<meta charset="UTF-8">
<title>JSP 4번째 예제(주석,comment)</title>
</head>
<!-- 5+3 = <%=5+3%> 주석안에 "표현식"을 쓸 수 있다. -->
<!-- 9+3 = <%=9+3 /*표현식 내부에 자바주석 사용이 가능하다*/ %> -->
<!-- 10*30 = <%=10*30%> -->
<body bgcolor="yellow">
    <h1>JSP 주석을 확인하는 예제</h1>

</body>
</html>
```

4-2) 지시어 **<%@ %>**

: **jsp 파일의 선언문** → **톰캣서버에 요청, 선언문 속성으로 요청**

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

→ **language = "java" : jsp 가 자바로 작성됩니다.**

→ **contentType = "text/html; charset="utf-8"**

: 텍스트형태의 html 문서로 만들어서 전송하는데, 한글이 있으면
한글이 깨지지 않도록 해서 보내주세요

→ **pageEncoding="UTF-8" : 한글로 Encoding 해주세요**

→ **<%@ %> 내부에 import : 외부에서 package 를 불러올 때 사용**

지시어의 활용

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"
import="java.util.Date" %>
```

<%@ page import="java.io.*,java.util.*" %> =>속성값을 여러개 나열->,,,

<%@ page import="java.sql.*" %> =>세로로 여러개의 import속성을 쓸 수있다.

->buffer,autoFlush=>입출력과 관련->액션태그와 연관

errorPage ,isErrorPage->에러페이지를 작성할때 사용

->현재 8.x에서는 사용X ---->Tomcat 4.x까지 사용했던 속성

isThreadSafe=false=>동시 접속X=>관리자만 접속해서 홈페이지 수정할때
외부에서 접속불가

isThreadSafe=true(default)->홈페이지가 동시 접속이 가능하게 할때 사용하는 방법

<%@page info="현재 페이지 연습중입니다">=>주석대용

형식)

<%@ 지시어종류 -> 3가지 요청할 문장(톰캣서버한테)

<%@ page ->page지시어

<%@ include ->include지시어> =>모델1(웹사이트 기본)

개인블로그,중소규모의 사이트

<%@ taglib ->taglib지시어 =>모델2->대규모 사이트->스프링

※ ScriptLet 의 활용 (꼭 이해하고 활용해야함)

```
<html>
<head>
<meta charset="UTF-8">
<title>중간점검(배열의 값을 출력)</title>
</head>
```

```
<%!
```

```
String[] keyword = {"Scriptlet","Expression","Declaration","Comment"};
```

```
%>
```

```
<body>
```

```
<table style="border:1px solid #333; width:200px; height:15px;">
```

```
<%
```

```
for(int i=0; i<keyword.length;i++){ %>
```

```
<tr>
```

```
<td style="border:1px solid #333; margin:5px;"><%=i%></td>
```

```
<td style="border:1px solid #333;"><%=keyword[i]%></td>
```

```
</tr>
```

```
<%
```

```
%>
```

```
</table>
```

```
</body>
```

```
</html>
```

Declaration

ScriptLet

Expression

자바구문
HTML구문

∴ <% %> 의 영역을 이용해 자바구문의 영역과 HTML 구문의 영역을 넘나들며 코딩할 수 있다.

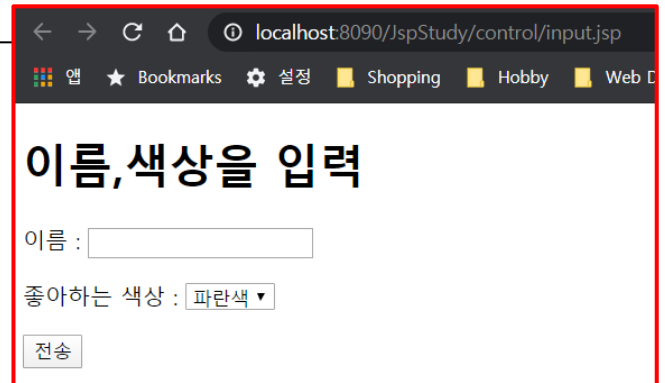
ScriptLet 영역은 영역이 끊겨 있다해도 서로 이어져있어서 변수들을 계속해서 사용할 수 있다.

5) 웹프로그래밍의 특성 : Request ↔ Response

[요청하는 페이지 + 서버로부터 응답받은 페이지] 로 구성된다.

요청하는 페이지 : input.jsp (form)

```
<html>
<head>
<meta charset="UTF-8">
<title>사용자로부터 값을 입력(전송폼)</title>
</head>
<body>
  <h1>이름,색상을 입력</h1>
  <form method="post" action="iftest.jsp">
    <p>이름 : <input type="text" name="name" id="name"> </p>
    <p>좋아하는 색상 :
      <select name="color" id="color" required>
        <option value="blue">파란색</option>
        <option value="red">빨간색</option>
        <option value="orange">주황색</option>
        <option value="black">검정색</option>
        <option value="white">하얀색</option>
        <option value="etc">기타</option>
      </select>
    <p>
      <input type="submit" value="전송" name="btn" id="btn">
    </p>
  </form>
</body>
</html>
```



form method = post // 정보가 가려져서 전송된다.

form action = "iftest.jsp" // 이 form 의 정보는 iftest.jsp 페이지로 전송된다.

- > html 태그 속성 중 name 속성의 값들이 일종의 flag 역할을 하면서 Parameter 를 넘겨준다.
- > form 내의 name 값을 iftest.jsp 에서도 response 내장객체를 활용하여 받는다.

※ 문자열 1 개를 전달할 때에는

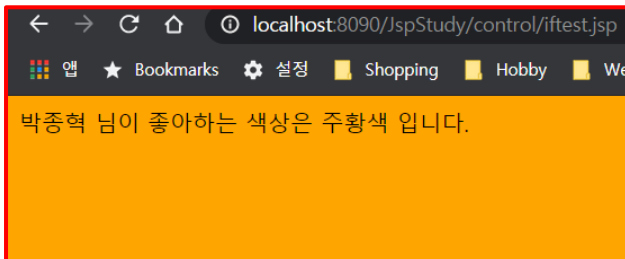

형식) `String name = request.getParameter("name");`

※ checkbox 로 여러값을 전달할 때에는 "배열"의 형태로 전달하고 출력한다.

형식) `String 변수[] = request.getParameterValues("매개변수배열");`

응답받은 페이지 : iftest.jsp

```
<head>
<meta charset="UTF-8">
<title>요청을 받아서 처리해주는 페이지</title>
<%!
    String msg; // 전달받은 색상 -> 한글로 바꿔 출력
%>
<% request.setCharacterEncoding("utf-8"); %> <!-- 따로 넣어줘도 됨 -->
<%
    // 외부에서 전달해주는 변수를 받아서 처리해주는 내장 객체 > request
    // 요구사항을 분석하고 처리결과를 html과 결합 하여 재전송해주는 객체 > response
    // 형식) String 반환값 = request.getParameter("전달받은 매개변수명");
    // 반환값과 매개변수명의 이름을 가능한한 같게
    //request.setCharacterEncoding("utf-8"); // 한글이 깨지지 않도록 encoding
    String name = request.getParameter("name"); // "name"은 input.jsp로부터 전달받은 개체
    String color = request.getParameter("color");
    System.out.println("name:"+name+",color:"+color); // color는 영어로 전달됨. 이 특성으로->배경색을 지정
    //문자열 비교 -> equals() or contentEquals()를 사용
    if(color.equals("blue")){
        msg = "파란색";
    }else if(color.equals("red")){
        msg="빨간색";
    }else if(color.equals("orange")){
        msg="주황색";
    }else if(color.equals("black")){
        msg="검정색";
    }else if(color.equals("white")){
        msg="하얀색";
    }else{
        msg="기타";
    }
%>
</head>
<body bgcolor="<%=color%>">
<%= name %> 님이 좋아하는 색상은 <%= msg %> 입니다.
</body>
```



- ①클라이언트로부터 넘겨받은 **Parameter**를 **변수에 저장** 시키고
- ②그 변수를 이용해 **조건문,반복문 등의 제어문**을 통해 요청을 처리하고 결과값을 도출한다.
- ③최종적으로 나온 결과값을 **스크립트릿** 혹은 **표현식**을 통해 작성하여 클라이언트에게 보여주도록 코딩한다.

6) 작성한 페이지의 배포 및 불러오기

(1) 배포하기

- ① 배포할 프로젝트명에서 우클릭
- ② export → WAR 선택
- ③ 경로를 지정한 후

Export source file 체크 : 프로젝트의 자바파일을 포함 시켜서 배포

Overwirte existing file 체크 : 기존에 파일이 있다면, 덮어쓰겠다.

- ④ Finish

(2) 불러오기

- ① 탐색기 창에서 우클릭
- ② import → WAR 선택
- ③ 경로를 지정한 후

Export source file 체크 : 프로젝트의 자바파일을 포함 시켜서 배포

Overwirte existing file 체크 : 기존에 파일이 있다면, 덮어쓰겠다.

- ④ Finish