# JSP 기초

## 기본 코드 구조

### JSP 페이지에 대한 정보

JSP 페이지가 생성하는 문서의 타입(종류)

JSP 페이지에서 사용할 커스텀 태그

JSP 페이지에서 사용할 자바 클래스 지정

### JSP 페이지 구성

일반적인 HTML 코드 + 자바 코드

|  |
| --- |
| HTML 문서를 생성하는 전형적인 JSP 코드 |
| <!-- JSP 페이지에 대한 설정 정보 -->  <!-- 문서 타입의 종류, 커스텀 태그, 자바 클래스 지정등의 설정이 들어갈 수 있다. -->  <!-- 현재는 문서의 타입 지정에 대한 설정만 있다. -->  <%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"* pageEncoding=*"UTF-8"*%>  <!-- JSP 페이지가 생성할 문서가 HTML이며 문서의 인코딩(캐릭터셋)이 UTF-8이다. -->  <%-- <%@ page ... %>를 "page 디렉티브"라고 한다. --%>  <!-- "page 디렉티브"는 JSP 페이지에 대한 정보를 설정할 때 사용한다. -->  <!DOCTYPE html>  <html><!-- HTML 코드 -->  <head><!-- 헤더 부분 -->  <title>HTML 문서의 제목</title>  </head> <!-- 헤더 부분 끝 -->  <body><!-- 바디 부분 시작 -->  <!-- JSP 스크립트 (스크립트릿) -->  <%  String bookTitle = "JSP 프로그래밍";  String author = "아무개";  %>  <b>  <!-- JSP 스크립트 (표현식) -->  <%= bookTitle %></b><%= author %>입니다.  </body> <!-- 바디 부분 끝 -->  </html><!-- HTML 코드 끝 --> |

## JSP 주석

스크립트릿과 선언부의 블록은 자바 코드이므로 자바의 주석을 사용할 수 있다.

JSP 코드 자체를 주석 처리할 때 : <%-- JSP 코드 주석 --%>

## JSP 페이지의 구성 요소

* 디렉티브(Directive)
* 스크립트: 스크립트릿(Scriptlet), 표현식(Expression), 선언부(Declaration)
* 표현 언어(Expression Language)
* 기본 객체(Implicit Object)
* 표준 액션 태그(Action Tag)
* 커스텀 태그(Custom Tag)와 표준 태그 라이브러리(JSTL)

## 디렉티브(Directive)

JSP 페이지에 대한 설정 정보를 지정할 때 사용

|  |
| --- |
| <%@ 디렉티브이름 속성1=”값1” 속성2=”값2” … %> |

|  |  |
| --- | --- |
| **디렉티브** | **설명** |
| page | **문서 타입**, **출력 버퍼 크기**, **에러 페이지** 등 JSP 페이지에서 필요로 하는 정보를 설정 |
| taglib | JSP 페이지에서 **사용할 태그 라이브러리를 지정** |
| Include | JSP 페이지의 **특정 영역에 다른 문서를 포함** |

### Page 디렉티브의 주요 속성

|  |  |  |
| --- | --- | --- |
| **속성** | **설명** | **기본값** |
| **contentType** | JSP가 생성할 문서의 MIME 타입과 캐릭터 인코딩을 지정 | text/html |
| **import** | JSP 페이지에서 사용할 자바 클래스를 지정 |  |
| **Buffer** | JSP 페이지의 출력 버퍼 크기를 지정  “none” – 사용안함, “8kb” 입력 시 8kb 크기의 출력 버퍼 사용 | 최소 8kb |
| **autoFlush** | 버퍼가 다 찼을 경우 자동으로 데이터를 출력 스트림에 보내고 비울지 여부  “true” - 버퍼의 내용을 브라우저에 보낸 후 버퍼를 비우며, “false” - 에러를 발생 | true |
| **info** | JSP 페이지에 대한 설명을 입력 |  |
| **errorPage** | JSP 페이지를 실행하는 도중에 에러가 발생할 때 보여줄 페이지를 지정 |  |
| **isErrorPage** | 현재 페이지가 에러가 발생될 때 보여주는 페이지인지의 여부를 지정  “true” - 에러 페이지 | False |
| **pageEncoding** | JSP 페이지 소스 코드의 캐릭터 인코딩을 지정 |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| MIME(Multipurpose Internet Mail Extensions)-클라이언트에게 전송된 문서의 다양성을 알려주기 위한 메커니즘.  브라우저들이 리소스를 내려 받았을 때 해야 할 기본 동작이 무엇인지를 결정하기 위해 사용.   |  |  | | --- | --- | | 타입 | 설명 | | text | 텍스트를 포함하는 모든 문서를 나타내며 이론상으로는 인간이 읽을 수 있어야 합니다 | | image | 모든 종류의 이미지를 나타냅니다. (animated gif 포함) | | audio | 모든 종류의 오디오 파일들을 나타냅니다. | | video | 모든 종류의 비디오 파일들을 나타냅니다. | | application | 모든 종류의 이진 데이터를 나타냅니다. | |

### 한글 설정 관련(문자셋 처리 부분이 여러가지인 이유)

Page 디렉티브의 charset은 서버내부에서 처리되는 문자의 인코딩을, Meta 태그의 charset은 HTML 코드 부분의 문자 인코딩을 지정. 그 밖에 request나 response에 포함된 문자의 처리도 개별적으로 수행해야 함. pageEncoding은 JSP 부분의 자바 코드의 문자셋을 지정.

## 스크립트 요소

JSP에서 문서의 내용을 동적으로 생성하기 위해 사용되는 것이 스크립트 요소이다.

스크립트릿(Scriptlet): 자바 코드를 실행한다.

표현식(Expression): 값을 출력한다.

선언부(Declaration): 자바 메서드(함수)를 만든다.

### 스크립트릿과 표현식

|  |
| --- |
| <html>  <head><title>1-10까지의 합</title></head>  <body>  **<%**  **int sum = 0;**  **for (int i = 1 ; i <= 10 ; i++) {**  **sum = sum + i;**  **}**  **%>**  1 부터 10까지의 합은 **<%= sum %>** 입니다.<!-- 출력은 표현식 사용 -->  </body>  </html> |

표현식 내에서 연산 가능

|  |
| --- |
| <html>  <head><title>1-10까지의 합:표현식만 사용</title></head>  <body>  1 부터 10까지의 합은  <%= 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 %>  입니다.  </body>  </html> |

### 선언부

스크립트릿이나 표현식에서 사용할 수 있는 메서드를 작성

|  |
| --- |
| <!-- 예제 1 (useDecl.jsp) -->  <%@ page contentType = "text/html; charset=utf-8" %>  <%!  public int multiply(int a , int b) {  int c = a \* b;  return c;  }  %>  <html>  <head><title>선언부를 사용한 두 정수값의 곱</title></head>  <body>  10 \* 25 = <%= multiply(10, 25) %>  </body>  </html> |
| <!-- 예제 2 -->  <%@ page contentType = "text/html; charset=utf-8" %>  <%!  public int add(int a, int b) {  int c = a + b;  return c;  }  public int subtract(int a, int b) {  int c = a - b;  return c;  }  %>  <html>  <head><title>스크립트릿에서 선언부 사용하기</title></head>  <body>  <%  int value1 = 3;  int value2 = 9;    int addResult = add(value1, value2);  int subtractResult = subtract(value1, value2);  %>  <%= value1 %> + <%= value2 %> = <%= addResult %>  <br>  <%= value1 %> - <%= value2 %> = <%= subtractResult %>  </body>  </html> |

## 기본 객체(Implicit Object)

JSP는 웹 애플리케이션 프로그래밍을 하는 데 필요한 기능을 제공해주는 ‘기본 객체(implicit object)’를 제공

request, response, session, application, page 등등.. (out, exception, …)

## request 객체

웹 브라우저의 요청과 관련 있는 기본 객체

request 기본 객체가 제공하는 기능

* 클라이언트(웹 브라우저)와 서버 관련 정보 처리
* 클라이언트가 전송한 요청 파라미터 처리
* 클라이언트가 전송한 요청 헤더 처리
* 클라이언트가 전송한 쿠키 처리
* 속성 처리

### request 기본 객체의 클라이언트 및 서버 정보 관련 메서드

|  |
| --- |
| <head><title>클라이언트 및 서버 정보</title></head>  <body>  클라이언트IP = <%= request.getRemoteAddr() %> <br>  요청정보길이 = <%= request.getContentLength() %> <br>  요청정보 인코딩 = <%= request.getCharacterEncoding() %> <br>  요청정보 컨텐츠타입 = <%= request.getContentType() %> <br>  요청정보 프로토콜 = <%= request.getProtocol() %> <br>  요청정보 전송방식 = <%= request.getMethod() %> <br>  요청 URI = <%= request.getRequestURI() %> <br>  컨텍스트 경로 = <%= request.getContextPath() %> <br>  서버이름 = <%= request.getServerName() %> <br>  서버포트 = <%= request.getServerPort() %> <br>  </body> |

|  |
| --- |
| **URL과 URI의 차이**   * URL(Uniform Resource Locator)   + **http://localhost:8080/chapter03/requestInfo.jsp** * URI(Uniform Resource Identifier)   + http://localhost:8080/**chapter03/requestInfo.jsp** |

### 요청 파라미터 처리

요청 파라미터 : HTML의 FORM 태그를 통해 전송하는 값. Input 태그의 name 속성과 value 속성을 사용

관련 메서드

|  |  |  |
| --- | --- | --- |
| **메서드** | **리턴 타입** | **설명** |
| getParameter(String name) | String | 이름이 name인 파라미터의 값을 구한다.(없으면 null) |
| **getParameterValues**(String name) | **String[]** | 이름이 name인 모든 파라미터의 값을 배열로 구한다.  존재하지 않을 경우 null을 리턴한다. |
| **getParameterNames()** | **Enumeration** | 웹 브라우저가 전송한 파라미터의 이름 목록을 구한다. |
| getParameterMap() | Map | 웹 브라우저가 전송한 파라미터의 맵을 구한다.  맵은 <파라미터 이름, 값> 쌍으로 구성된다. |

form.jsp

|  |
| --- |
| <%@ page contentType = "text/html; charset=utf-8"%>  <html>  <head><title>폼 생성</title></head>  <body>  <form action="./viewParameter.jsp" method="post">  이름: <input type="text" name="name" size="10"> <br>  주소: <input type="text" name="address" size="30"> <br>  좋아하는 동물:  <input type="checkbox" name="pet" value="dog">강아지  <input type="checkbox" name="pet" value="cat">고양이  <input type="checkbox" name="pet" value="pig">돼지  <br>  <input type="submit" value="전송">  </form>  </body>  </html> |

viewParameter.jsp

|  |
| --- |
| <%@ page import="java.util.Enumeration, java.util.Map" %>  <% request.setCharacterEncoding("utf-8"); %><!-- 요청값에 대한 문자인코딩 -->  <html>  <head><title>요청 파라미터 출력</title></head>  <body>  <b>request.getParameter() 메서드 사용</b><br>  name 파라미터 = <%= request.getParameter("name") %> <br>  address 파라미터 = <%= request.getParameter("address") %>  <p>  <b>request.getParameterValues() 메서드 사용</b><br>  <%  String[] values = request.getParameterValues("pet");  if (values != null) {  for (int i = 0 ; i < values.length ; i++) {  %>  <%= values[i] %>  <%  }  }  %>  </p>  <p>  <b>request.getParameterNames() 메서드 사용</b><br>  <%  Enumeration paramEnum = request.getParameterNames();  while(paramEnum.hasMoreElements()) {  String name = (String)paramEnum.nextElement();  %>  <%= name %>  <%  }  %>  </p>  <p>  <b>request.getParameterMap() 메서드 사용</b><br>  <%  Map parameterMap = request.getParameterMap();  String[] nameParam = (String[])parameterMap.get("name");  if (nameParam != null) {  %>  name = <%= nameParam[0] %>  <%  }  %>  </body>  </html> |

**체크박스(<input type=”checkbox” …>)와 라디오 버튼(<input type=”radio” ..>)**은 선택하지 않으면 **파라미터 자체가 전송되지 않는다.**

**텍스트(<input type=”text” …>)**를 위한 입력 요소들은 값을 입력하지 않더라도 **빈 문자열(“”)이 전달된다.**

### 요청 전송 방식

**GET 방식 (노출 되도 상관없는 데이터 전달)**

- URL의 경로 뒤에 물음표(‘?’)와 함께 파라미터를 붙여 전송 이를 쿼리 문자열(query string)이라고 한다.

- 각각의 파라미터는 앰퍼샌드(&) 기호로 구분

- 파라미터의 이름과 값은 등호기호(=)로 구분

**POST 방식 (노출되면 안되는, 또는 양이 많은 데이터 전달)**

- 데이터 영역을 이용해서 데이터를 전송하기 때문에 파라미터의 길이에 제한이 없다.

## Response 객체

웹 브라우저에 보내는 응답 정보. 주로 리다이렉트(redirect) 기능 활용에 이용(페이지 이동)

### 리다이렉트 기능의 흐름



### 관련 함수

response.sendRedirect(String location)

활용 : 로그인 후, 첫 페이지로 이동하길 원할 경우

login.jsp

|  |
| --- |
| <%@ page contentType="text/html; charset=utf-8" %>  <%  String id = request.getParameter("memberId");  if (id != null && id.equals("madvirus")) {  response.sendRedirect("./index.jsp");  } else {  %>  <html>  <head><title>로그인에 실패</title></head>  <body>  잘못된 아이디입니다.  </body>  </html>  <%  }  %> |

index.jsp

|  |
| --- |
| <%@ page contentType="text/html; charset=utf-8" %>  <html>  <head><title>로그인 예제</title></head>  <body>  여기는 첫 화면  </body>  </html> |

URLEncoder.encode() 메서드를 사용하여 파라미터 값으로 사용될 문자열을 지정된 캐릭터 셋으로 설정

|  |
| --- |
| <%@ page pageEncoding=*"utf-8"* %>  <%@ page import = *"java.net.URLEncoder"* %>  <%  String value = "자바";  String encodedValue = URLEncoder.encode(value, "utf-8");  response.sendRedirect("./index.jsp?name=" + encodedValue);  %> |

## 표현 언어(EL, Expression Language)

JSP 스크립트 코드보다 간결하고 이해하기 좋은 언어

|  |
| --- |
| <!-- 스크립트릿 -->  <%  **int** a = Integer.parseInt(request.getParameter("a"));  **int** b = Integer.parseInt(request.getParameter("b"));  %>  <!-- 표현식 -->  a \* b = <%= a \* b %>  <br>  <!-- EL -->  a \* b = ${param.a \* param.b} |

## 표준 액션 태그(Action Tag)와 태그 라이브러리(JSTL)

### 표준 액션 태그(Action Tag)

<jsp:액션태그이름>의 형태로 액션 태그 종류에 따라서 다양한 특별한 기능을 제공한다.

### 커스텀 태그(Custom Tag)

JSP를 확장시켜주는 기능

액션 태그와 마찬가지로 태그 형태로 기능을 제공

개발자가 직접 개발한 기능을 제공

JSTL(JavaServer Pages Standard Tag Library)가 대표적인 표준화한 태그 라이브러리