

# JSP로 시작하는 웹 프로그래밍

# JSP에서 HTML 문서를 생성하는 기본 코드 구조

```
<%@ page contentType = "text/html; charset=utf-8" %>
<html>
<head>
    <title>HTML 문서의 제목</title>
</head>
<body>
<%
    String title = "JSP 프로그래밍";
    String author = "홍길동";
%>
    <b><%= title %></b>(<%= author %>)입니다.
</body>
</html>
```

설정 부분:  
JSP 페이지에 대한  
설정 정보

생성 부분:  
HTML 코드 및  
JSP 스크립트

# JSP 페이지의 구성 요소

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

# 디렉티브(Directive)

- JSP 페이지에 대한 설정 정보를 지정
- 디렉티브 구문
  - `<%@ 디렉티브이름 속성1="값1" 속성2="값2" ... %>`
  - 예) `<%@ page contentType = "text/html; charset=utf-8" %>`
- 제공 디렉티브
  - page : JSP 페이지에 대한 정보를 지정
    - 문서의 타입, 출력 버퍼의 크기, 에러 페이지 등 정보 지정
  - taglib : JSP 페이지에서 사용할 태그 라이브러리를 지정
  - include : JSP 페이지의 특정 영역에 다른 문서를 포함

# page 디렉티브

- JSP 페이지에 대한 정보를 입력
  - JSP가 생성할 문서의 타입, 사용할 클래스, 버퍼 여부, 세션 여부
- JSP 디렉티브의 작성 예
  - `<%@ page contentType="text/html; charset=utf-8" %>`
  - `<%@ page import="java.util.Date" %>`

# page 디렉티브의 주요 속성

속성	설명	기본값
contentType	JSP가 생성할 문서의 타입을 지정한다.	Text/html
import	JSP 페이지에서 사용할 자바 클래스를 지정한다.	
session	JSP 페이지가 세션을 사용할 지의 여부를 지정한다. "true"일 경우 세션을 사용하고 "false"일 경우 세션을 사용하지 않는다.	true
buffer	JSP 페이지의 출력 버퍼 크기를 지정한다. "none"일 경우 출력 버퍼를 사용하지 않으며, "8kb"라고 입력한 경우 8킬로바이트 크기의 출력 버퍼를 사용한다.	
autoFlush	출력 버퍼가 다 찼을 경우 자동으로 버퍼에 있는 데이터를 출력 스트림에 보내고 비울 지의 여부를 나타낸다. "true"일 경우 버퍼의 내용을 웹 브라우저에 보낸 후 버퍼를 비우며, "false"일 경우 에러를 발생시킨다.	true
info	JSP 페이지에 대한 설명을 입력한다.	
errorPage	JSP 페이지를 실행하는 도중에 에러가 발생할 때 보여줄 페이지를 지정한다.	
isErrorPage	현재 페이지가 에러가 발생할 때 보여줄 페이지인지의 여부를 지정한다. "true"일 경우 에러 페이지이며, "false"일 경우 에러 페이지가 아니다.	false
pageEncoding	JSP 페이지 자체의 캐릭터 인코딩을 지정한다.	
isELIgnored	"false"일 경우 표현 언어를 지원하며, "true"일 경우 표현 언어를 지원하지 않는다.	false
deferredSyntaxAllowedAsLiteral	#{ 문자가 문자열 값으로 사용되는 것을 허용할 지의 여부를 지정한다.	false
trimDirectiveWhitespaces	출력 결과에서 템플릿 텍스트의 공백 문자를 제거할 지의 여부를 지정한다.	false

# page 디렉티브: contentType 속성과 캐릭터 셋

- JSP 페이지가 생성할 문서의 타입을 지정
- contentType 속성 형식

```
TYPE  
or  
TYPE; charset=캐릭터 셋
```

- TYPE: 생성할 문서의 MIME 타입
  - text/html, text/xml, text/plain 등
- 캐릭터 셋 - 응답 문서의 문자 인코딩 지정
  - EUC-KR, UTF-8, ISO-8859-1 등
  - 대소문자 구별하지 않음.
- 설정 예

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

# page 디렉티브: import 속성

- JSP 페이지에서 사용할 클래스(인터페이스) 지정
- import 속성의 사용 예

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

- import 한 클래스는 단순 클래스 이름으로 사용 가능

```
<%@ page contentType = "text/html; charset=utf-8" %>  
<%@ page import = "java.util.Date" %>  
<html>  
<head> <title>Calendar 클래스 사용</title> </head>  
<body>  
<%  
    Date date = new Date();  
    java.util.Calendar cal = java.util.Calendar.getInstance();  
%>
```



# page 디렉티브: trimDirectiveWhitespaces 속성

```
<%@ page contentType="text/html; charset=utf-8" %>
<%@ page import="java.util.Date" %>
<%@ page trimDirectiveWhitespaces="true" %>
<%
    Date now = new Date();
%>
<html>
<head> <title>현재 시간</title> </head>
<body>
    현재 시각:
    <%= now %>
</body>
</html>
```

# page 디렉티브:

## JSP 페이지의 인코딩과 pageEncoding 속성

- charset=utf-8 : 브라우저에게 알려줄 캐릭터 셋 설정
- pageEncoding="utf-8" : 웹컨테이너가 현재 파일을 읽어올 때 캐릭터 셋 설정(설정값 없으면 charset 값으로 셋팅)

```
<%@ page contentType="text/html; charset=utf-8" %>
<%@ page pageEncoding="utf-8" %>
<%@ page import="java.util.Date" %>
<%
    Date now = new Date();
%>
<html>
<head> <title>현재 시간</title> </head>
<body>
    현재 시각:
    <%= now %>
</body>
</html>
```

# 스크립트 요소

- 요청을 처리하는 데 필요한 코드를 실행
  - 실시간으로 문서의 내용을 생성하기 위해 사용되는 것
- 동적으로 응답 결과를 생성하기 위해 사용
  - 사용자가 폼에 입력한 정보를 데이터베이스에 저장할 수 있음
  - 데이터베이스로부터 게시글 목록을 읽어와 출력할 수도 있음
  - 자바가 제공하는 다양한 기능들도 사용할 수 있음
- 스크립트 요소 세 가지
  - 스크립트릿(Scriptlet) : 자바 코드 실행
  - 표현식(Expression) : 값을 출력
  - 선언부(Declaration) : 자바 메서드(함수)를 정의

# 스크립트릿(Scriptlet)

- 자바 코드를 실행할 때 사용되는 코드의 블록
- 스크립트릿의 구조

```
<%  
    자바코드1;  
    자바코드2;  
    ....  
%>
```

- 예제 코드

```
<%@ page contentType = "text/html; charset=utf-8" %>  
<%  
    int sum = 0;  
    for (int i = 1 ; i <= 10 ; i++) {  
        sum = sum + i;  
    }  
%>  
1 부터 10까지의 합은 <%= sum %> 입니다.
```

# 표현식(Expression)

- 값을 출력 결과에 포함시키고자 할 때 사용
- 표현식 구문
  - `<%= 값 %>`
- 표현식 예

```
<%= 1 + 2 + 3 + 4 + 5 + 6 + 7 + 8 + 9 + 10 %>
```

```
<%
```

```
    int sum = 0;
```

```
    for (int i = 1 ; i <= 10 ; i++) {
```

```
        sum = sum + i;
```

```
    }
```

```
%>
```

1 부터 10까지의 합은 `<%= sum %>` 입니다.

# 선언부(Declaration)

- 스크립트릿이나 표현식에서 사용할 수 있는 함수를 작성할 때 사용
- 선언부 형식

```
<%!  
    public 리턴타입 메서드이름(파라미터목록) {  
        자바코드1;  
        자바코드2;  
  
        ...  
        자바코드n;  
        return 값;  
    }  
%>
```

# 선언부와 파라미터 값 전달

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

```
<%!
```

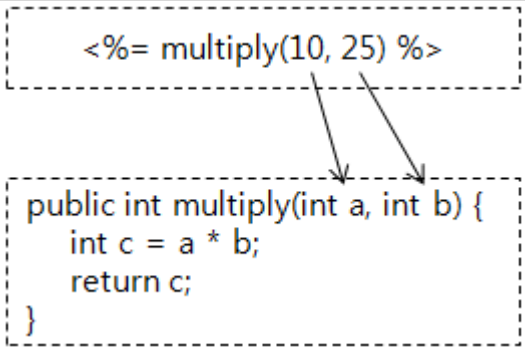
```
    public int multiply(int a , int b) {  
        int c = a * b;  
        return c;  
    }
```

```
%>
```

```
<html>
```

```
<head><title>선언부를 사용한 두 정수값의 곱</title></head>
```

```
<body>
```



```
<%= multiply(10, 25) %>
```

```
public int multiply(int a, int b) {  
    int c = a * b;  
    return c;  
}
```

```
10 * 25 = <%= multiply(10, 25) %>
```

# 기본 객체(implicit object)

- 웹 프로그래밍에 필요한 기능을 제공
- JSP에서 별도 선언 없이 사용 가능
- 주요 기본 객체
  - request : 요청 정보를 구할 때 사용
  - response : 응답과 관련된 설정(헤더, 쿠키 등)시 사용
  - out : 직접 응답을 출력할 때 사용
  - session : 세션 관리에 사용

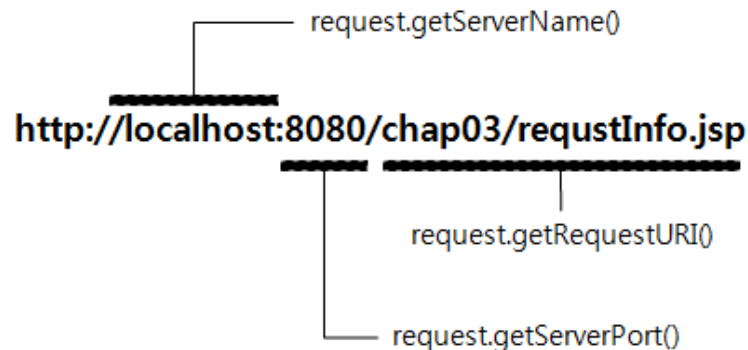


# request 기본 객체

- 웹 브라우저가 웹 서버에 전송한 요청 관련 정보 제공
- 주요 기능
  - 클라이언트(웹 브라우저)와 관련된 정보 읽기 기능
  - 서버와 관련된 정보 읽기 기능
  - 클라이언트가 전송한 요청 파라미터 읽기 기능
  - 클라이언트가 전송한 요청 헤더 읽기 기능
  - 클라이언트가 전송한 쿠키 읽기 기능
  - 속성 처리 기능

# request 기본 객체 - 주요 정보 제공 메서드

메서드	리턴 타입	설 명
getRemoteAddr()	String	웹 서버에 연결한 클라이언트의 IP 주소를 구한다. 게시판이나 방명록 등에서 글 작성자의 IP 주소가 자동으로 입력되기도 하는데, 이때 입력되는 IP 주소가 바로 이 메서드를 사용하여 구한 것이다.
getContentLength()	long	클라이언트가 전송한 요청 정보의 길이를 구한다. 전송된 데이터의 길이를 알 수 없는 경우 -1을 리턴한다.
getCharacterEncoding()	String	클라이언트가 요청 정보를 전송할 때 사용한 캐릭터의 인코딩을 구한다.
getContentType()	String	클라이언트가 요청 정보를 전송할 때 사용한 콘텐츠의 타입을 구한다.
getProtocol()	String	클라이언트가 요청한 프로토콜을 구한다.
getMethod()	String	웹 브라우저가 정보를 전송할 때 사용한 방식을 구한다.
getRequestURI()	String	웹 브라우저가 요청한 URL에서 경로를 구한다.
getContextPath()	String	JSP 페이지가 속한 웹 어플리케이션의 컨텍스트 경로를 구한다.
getServerName()	String	연결할 때 사용한 서버 이름을 구한다.
getServerPort()	int	서버가 실행 중인 포트 번호를 구한다.



# 요청 파라미터

The screenshot shows the 'make' page of a Daum Cafe. The browser address bar displays 'http://cafe.daum.net/make/cafe\_make.html'. The page contains several input fields and options for creating a new cafe:

- 이름 (Name):** A text input field with a placeholder '0/20bytes'.
- 주소 (Address):** A text input field with a placeholder 'http://cafe.daum.net/' and a '0/200bytes' limit.
- 공개여부 (Privacy):** Radio buttons for '공개' (Public) and '비공개' (Private).
- 카테고리 (Category):** A dropdown menu with '전체' (All) selected, and a sub-menu showing '대중/취미' (General/Hobby).
- 카카오 검색어 (Kakao Searcher):** A text input field with a placeholder '0/200bytes'.
- 소개글 (Introduction):** A large text area with a placeholder '0/1000bytes'.

파라미터 목록  
(폼 데이터)

웹서버

# request 기본 객체 - 파라미터 읽기 메서드

메서드	리턴 타입	설 명
getParameter(String name)	String	이름이 name인 파라미터의 값을 구한다. 존재하지 않을 경우 null을 리턴한다.
getParameterValues(String name)	String[]	이름이 name인 모든 파라미터의 값을 배열로 구한다. 존재하지 않을 경우 null을 리턴한다.
getParameterNames()	java.util.Enumeration	웹 브라우저가 전송한 파라미터의 이름을 구한다.
getParameterMap()	java.util.Map	웹 브라우저가 전송한 파라미터의 맵을 구한다. 맵은 <파라미터 이름, 값> 쌍으로 구성된다.

# GET 방식(METHOD)/POST 방식(METHOD)

- 파라미터를 전송하는 방식
  - GET : 쿼리문자열로 전송(길이 제한)
  - POST : 요청 몸체 데이터로 전송
- GET 방식 전송 예

```
GET /chap03/viewParameter.jsp?name=cbk&address=seoul HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; ...
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ko-kr,ko;q=0.8,en-us;q=0.5,en;q=0.3
...
```

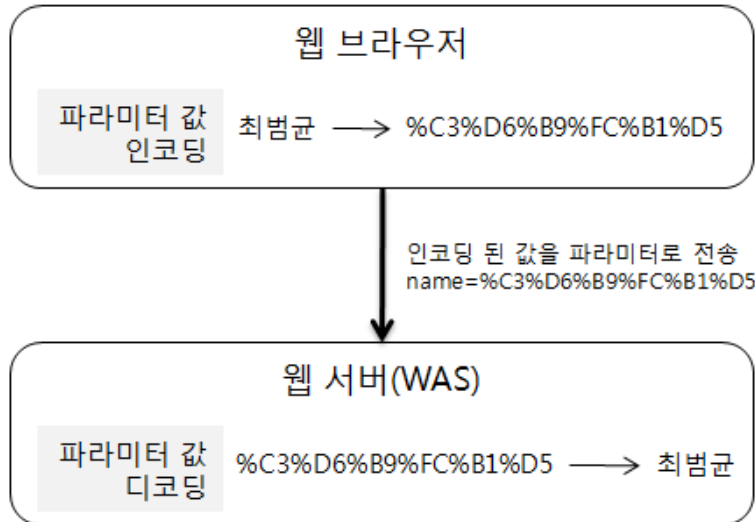
- POST 방식 전송 예

```
POST /chap03/viewParameter.jsp HTTP/1.1
Host: localhost:8080
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; ko; rv:1.9.0.3) ...
...
Content-Type: application/x-www-form-urlencoded
Content-Length: 22

name=cbk&address=seoul
```

# 파라미터 값의 인코딩/디코딩

- 파라미터 값의 인코딩/디코딩



- JSP에서 파라미터 로딩 시 인코딩 지정 필요(POST 방식)

```
<%  
    request.setCharacterEncoding("utf-8");  
    String name = request.getParameter("name");  
%>
```

# GET 방식을 이용해서 파라미터 전송 방법

- <a> 태그의 링크에 쿼리 문자열 추가

- <a href="viewParameter.jsp?name=홍길동&address=역삼동">링크</a>

- HTML 폼(form)의 method 속성 값을 "get"으로 지정해서 전송

```
<%@ page contentType = "text/html; charset=utf-8" %>
...
<form action="viewParameter.jsp" method="get">
<input type="text" name="name" size="10"> <br>
<input type="text" name="address" size="30"> <br>
<input type="submit" value="전송">
</form>
```

- 웹브라우저 주소에 직접 쿼리 문자열 포함한 URL 입력
  - HTTP, URL, URI 등의 표준에는 get 방식으로 전달되는 파라미터 값을 인코딩 할 때 어떤 캐릭터 셋을 사용해야 하는지에 대한 규칙이 정해져 있지 않다.
  - 톰캣은 기본적으로 ISO-8859-1 캐릭터 셋으로 파라미터 값을 읽어온다.

# 톰캣에서 GET 방식 파라미터를 위한 인코딩 처리

- 방법1 : Server.xml 파일에서 <Connector>의 useBodyEncodingForURI 속성의 값을 true로 지정
  - [톰캣설치디렉토리]/conf/server.xml

```
<Connector port="8090" protocol="HTTP/1.1"
            connectionTimeout="20000"
            redirectPort="8443"
            useBodyEncodingForURI="true" />
```

- request.setCharacterEncoding("utf-8"); 이 적용됨

- 방법2 : Server.xml 파일에서 <Connector>의 URLEncoder 속성의 값으로 원하는 캐릭터 셋을 지정

```
<Connector port="8090" protocol="HTTP/1.1"
            connectionTimeout="20000"
            redirectPort="8443"
            URLEncoder="utf-8" />
```

- request.setCharacterEncoding("utf-8"); 은 적용되지 않음



# request 기본 객체 - 요청 헤더 정보 읽기

메서드	리턴 타입	설 명
getHeader(String name)	String	지정한 이름의 헤더 값을 구한다.
getHeaders(String name)	java.util.Enumeration	지정한 이름의 헤더 목록을 구한다.
getHeaderNames()	java.util.Enumeration	모든 헤더의 이름을 구한다.
getIntHeader(String name)	Int	지정한 헤더의 값을 정수 값으로 읽어 온다.
getDateHeader(String name)	long	지정한 헤더의 값을 시간 값으로 읽어 온다.

# response 기본 객체

- 웹 브라우저에 전송하는 응답 정보 설정
- 주요 기능
  - 헤더 정보 입력
  - 리다이렉트 처리

# response 기본 객체 - 헤더 설정 메서드

메서드	리턴 타입	설 명
<code>addDateHeader(String name, long date)</code>	void	name 헤더에 date를 추가한다. date는 1970년 1월 1일 이후 흘러간 시간을 1/1000초 단위로 나타낸다.
<code>addHeader(String name, String value)</code>	void	name 헤더에 value를 값으로 추가한다.
<code>addIntHeader(String name, int value)</code>	void	name 헤더에 정수 값 value를 추가한다.
<code>setDateHeader(String name, long date)</code>	void	name 헤더의 값을 date로 지정한다. date는 1970년 1월 1일 이후 흘러간 시간을 1/1000초 단위로 나타낸다.
<code>setHeader(String name, String value)</code>	void	name 헤더의 값을 value로 지정한다.
<code>setIntHeader(String name, int value)</code>	void	name 헤더의 값을 정수 값 value로 지정한다.
<code>containsHeader(String name)</code>	boolean	이름이 name인 헤더를 포함하고 있을 경우 true를 그렇지 않을 경우 false를 리턴한다.

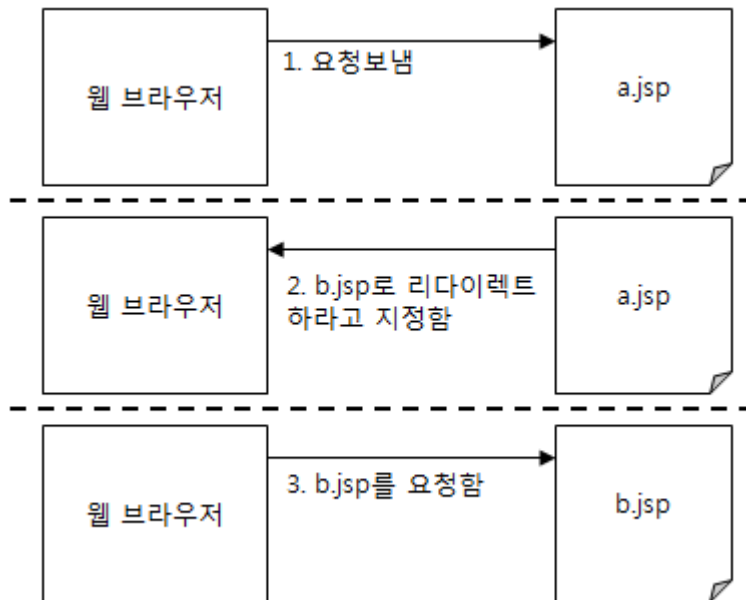
# 웹 브라우저 캐시 제어

- 응답 헤더 입력

```
<%  
    response.setHeader("Pragma", "No-cache");  
    response.setHeader("Cache-Control", "no-cache");  
    response.addHeader("Cache-Control", "no-store");  
  
    response.setDateHeader("Expires", 1L);  
%>
```

# 리다이렉트(Redirect)

- 특정 페이지로 이동하라고 웹 브라우저에 응답
- 웹 브라우저는 실질적으로 요청을 두 번하게 됨



- `response.sendRedirect(String location)`로 구현