



## 3장 JSP로 시작하는 웹 프로그래밍

### 3.1 JSP에서 HTML 문서를 생성하는 기본 코드 구조

- JSP 코드를 작성하는 주된 목적은 웹 브라우저에 보여 줄 HTML 문서를 생성하는 것이다.

HTML 문서를 생성하는 전형적인 JSP 코드

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


### 3.2 JSP 페이지의 구성 요소

#### 3.2.1 디렉티브

- JSP 페이지에 대한 설정 정보를 지정할때 사용한다.
- 구문: <%@ 디렉티브이름 속성1="값1" 속성2="값2" ... %>
- 제공 디렉티브
  - page : JSP 페이지에 대한 정보(문서의 타입, 출력 버퍼의 크기, 에러 페이지 등)를 지정
  - taglib : 사용할 태그 라이브러리를 지정
  - include : 다른 문서를 포함

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

#### 3.2.2 스크립트 요소

- JSP에서 문서의 내용을 동적으로 생성하기 위해 사용한다.
- 스크립트 요소 
  - 표현식(Expression) : 값을 출력
  - 스크립트릿(Scriptlet) : 자바 코드를 실행
  - 선언부(Declaration) : 자바 메서드(함수)를 정의

### 3.2.3 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

### 3.2.4 표현 언어(Expression language)

- 자바 문법을 그래도 사용할 수 있기 때문에 자바 언어의 특징을 그대로 사용할 수 있다는 장점이 있다. 하지만 스크립트 요소를 사용하면 JSP 코드가 다소 복잡해진다.

```
<%  
    int a = Integer.parseInt(request.getParameter("a"));  
    int b = Integer.parseInt(request.getParameter("b"));  
%>  
a * b = <%= a*b %>
```

- 위 코드를 표현 언어(Expression Language)를 사용하면 다음과 같이 간결하게 작성할 수 있다.

```
a * b = ${param.a * param.b}
```

### 3.2.5 표준 액션 태그와 태그 라이브러리

- 액션 태그는 특정한 페이지의 실행 결과를 현재 위치에 포함시킬 때 사용한다.

```
<jsp:include page="header.jsp" flush="true" />
```

## 3.3 page 디렉티브

- JSP 페이지에 대한 정보를 입력하기 위해서 사용한다.
  - 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" %>
- 주요 속성
  - contentType : JSP가 생성할 문서의 타입을 지정
  - import : JSP 페이지에서 사용할 자바 클래스를 지정
  - session : JSP 페이지가 세션을 사용할 지의 여부를 지정

- info : JSP 페이지에 대한 설명을 입력한다.
- errorPage : 에러가 발생할 때 보여 줄 페이지를 지정
- isErrorPage : 에러 페이지인지의 여부를 지정

### 3.3.1 contentType 속성과 캐릭터 셋

- JSP 페이지가 생성할 문서의 타입을 지정한다.
- contentType: 생성할 문서의 MIME 타입
  - text/html, text/xml, text/plain 등
- charset: 응답 문서의 문자 인코딩 지정
  - EUC-KR, UTF-8, ISO-8859-1 등

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

[chap03\invalidCharset.jsp]

```
01 <%@ page contentType="text/html; charset=iso-8859-1"%>
02 <%@ page import="java.util.Date"%>
03 <%
04     Date now = new Date();
05 %>
06 <html>
07 <head>
08 <title>현재 시간</title>
09 </head>
10 <body>
11     현재 시각:
12     <%= now %>
13 </body>
14 </html>
```

### 3.3.2 import 속성

- JSP 페이지에서 사용할 자바 클래스(인터페이스) 지정한다.
- import 한 클래스는 단순 클래스 이름으로 사용 가능

```
<%@ page contentType = "text/html; charset=euc-kr" %>
<%@ 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();
%>
```

### 3.3.3 trimDirectiveWhitespaces 속성을 이용한 공백 처리

- 불필요하게 생성되는 줄바꿈 공백 문자를 제거한다.

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

### 3.3.4 JSP 페이지의 인코딩과 pageEncoding 속성

- 웹 컨테이너가 JSP 페이지를 읽어올 때 사용할 캐릭터 셋을 결정하는 기본 과정은 다음과 같다.
  - 기본 인코딩을 이용해서 파일을 처음부터 읽고, 파일을 읽어올 때 속성값을 캐리터 셋으로 사용한다.
  - pageEncoding 속이 값을 갖고 있다면, 파일을 읽어올 때 속성값을 캐릭터 셋으로 사용한다.
  - pageEncoding 속성이 없다면, contentType 속성을 검색한다.
  - 모두 해당되지 않을 경우, ISO-8859-1을 캐릭터 셋으로 사용한다.

## 3.4 스크립트 요소

### 3.4.1 스크립트릿(Scriptlet)

- JSP 페이지에서 자바 코드를 실행할 때 사용하는 코드 블록이다.

```
<%
자바코드1;
자바코드2;
...
%>
```

[chap03/oneToTen.jsp]

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

```
</body>
</html>
```

- 아래와 같이 두 개의 스크립트릿을 사용해서 구현할 수 있다.

[chap03/sum.jsp]

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

    <br>

    <%
        int sum2 = 0;
        for (int i = 11; i <= 20; i++) {
            sum2 = sum2 + i;
        }
    %>
    11 부터 20까지의 합은
    <%=sum2%>
    입니다.
</body>
</html>
```

### 3.4.2 표현식(Expression)

- 표현식은 어떤 값을 출력 결과에 포함시키고자 할 때 사용된다.

<%= 값 %>

[chap03/oneToTen2.jsp]

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

### 3.4.3 선언부(Declaration)

- JSP 페이지의 스크립트릿이나 표현식에서 사용할 수 있는 메서드를 작성할 때에는 선언부를 사용한다.

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

[chap03/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>
```




## 3.5 request 기본 객체

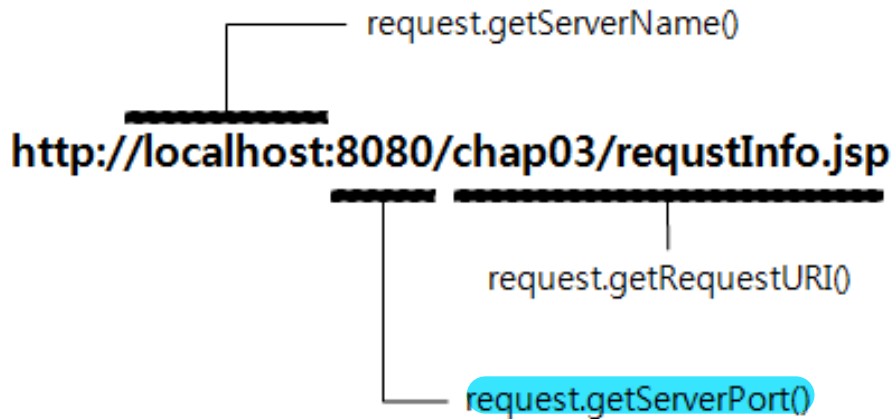
- 웹 브라우저가 웹 서버에 전송한 요청 관련 정보 제공
- 주요 기능
  - 클라이언트(웹 브라우저)와 관련된 정보 읽기 기능
  - 서버와 관련된 정보 읽기 기능
  - 클라이언트가 전송한 요청 파라미터 읽기 기능
  - 클라이언트가 전송한 요청 헤더 읽기 기능
  - 클라이언트가 전송한 쿠키 읽기 기능
  - 속성 처리 기능

### 3.5.1 클라이언트 정보 및 서버 정보 읽기

- request 기본 객체는 웹 브라우저, 즉 클라이언트가 전송한 정보와 서버 정보를 구할 수 있는 메서드를 제공하고 있다.

메서드	리턴 타입	설명
-----	-------	----

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.
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 실행중인 포트 번호를 구한다.



[chap03/requestInfo.jsp]

```
<%@ page contentType = "text/html; charset=utf-8" %>
<html>
<head><title>클라이언트 및 서버 정보</title></head>
<body>

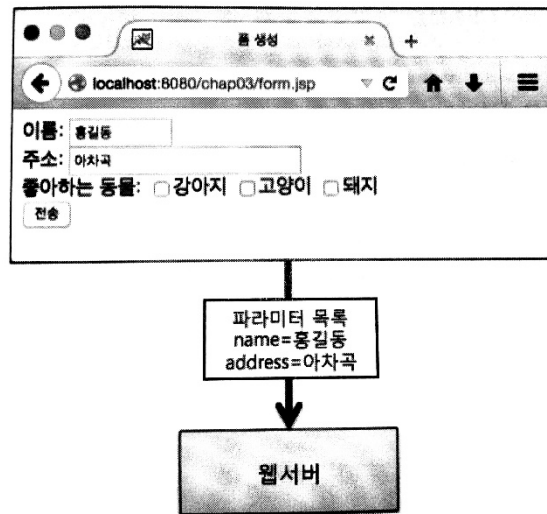
클라이언트 IP = <%= request.getRemoteAddr() %> <br>
요청정보길이 = <%= request.getContentLength() %> <br>
요청정보 인코딩 = <%= request.getCharacterEncoding() %> <br>
요청정보 컨텐츠타입 = <%= request.getContentType() %> <br>
요청정보 프로토콜 = <%= request.getProtocol() %> <br>
요청정보 전송방식 = <%= request.getMethod() %> <br>
요청 URI = <%= request.getRequestURI() %> <br>
컨텍스트 경로 = <%= request.getContextPath() %> <br>
서버이름 = <%= request.getServerName() %> <br>
서버포트 = <%= request.getServerPort() %> <br>

</body>
</html>
```

### 3.5.2 요청 파라미터 처리

#### (1) HTML 폼과 요청 파라미터

- 폼에 입력한 정보를 파라미터로 전송



## (2) request 기본 객체의 요청 파라미터 관련 메서드

메서드	리턴타입	설명
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.

[chap03\form.jsp]

```
<%@ page contentType = "text/html; charset=utf-8"%>
<html>
<head><title>폼 생성</title></head>
<body>

<form action="/chap03/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>
```

[chap03/viewParameter.jsp]

```
<%@ page contentType="text/html; charset=utf-8" %>
<%@ page import="java.util.Enumeration" %>
<%@ page import="java.util.Map" %>
<%
    request.setCharacterEncoding("utf-8");
%>
<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>
<b>request.getParameterNames() 메서드 사용</b><br>
<%
    Enumeration paramEnum = request.getParameterNames();
    while(paramEnum.hasMoreElements()) {
        String name = (String)paramEnum.nextElement();
        <%= name %>
    }
%>
<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>


```

### (3) GET 방식 전송과 POST 방식 전송

#### ■ 파라미터를 전송하는 방식

- GET : 쿼리문자열로 전송
- POST : 요청 몸체 데이터로 전송

#### ■ GET 방식

- 요청 URL에 파라미터를 붙여서 전송한다.
- 폼을 사용하지 않아도 파라미터를 전송할 수 있다. 

#### ■ POST 방식

- 데이터 영역을 이용해서 파라미터를 전송한다.
- 웹 브라우저나 웹 서버 등에 상관없이 전송할 수 있는 파라미터의 길이에 제한이 없다.

```

// GET 방식 전송 예
GET /chap03/viewParameter.jsp?name=cbk&address=seoul HTTP/1.1
Host: localhost:8080

```

```

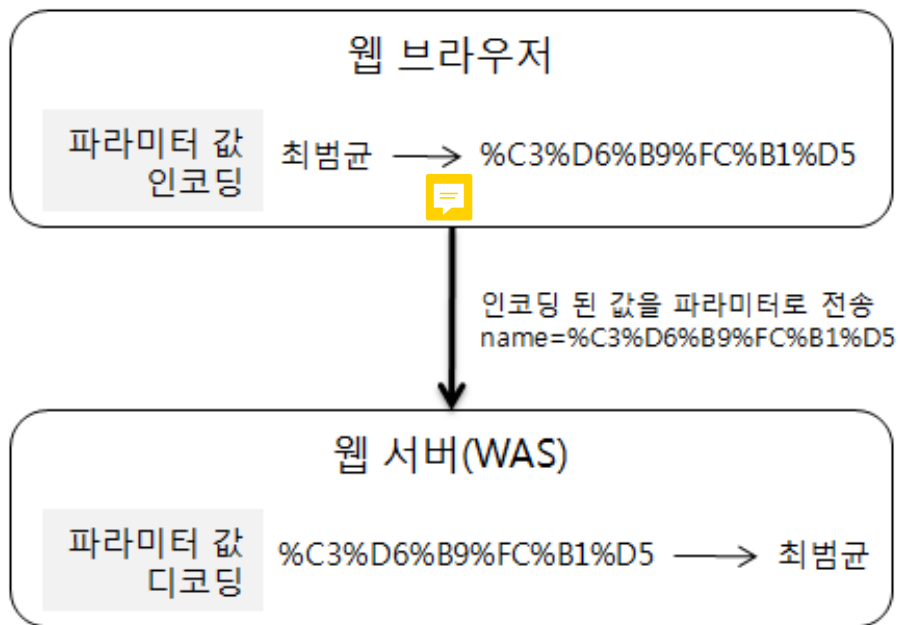
User-Agent: Mozilla/5.0 (Windows; U; Windows NT 6.0; ...
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: ko-kr,ko;q=0.8,en-us;q=0.5,en;q=0.3

// 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

```

#### (4) 요청 파라미터 인코딩

- 웹 브라우저는 웹 서버에 파라미터를 전송할 때 알맞은 캐리터 셋을 이용해서 파라미터 값을 인코딩한다.
- 반대로 웹 서버는 알맞은 캐리터 셋을 이용해서 웹 브라우저가 전송한 파라미터 데이터를 디코딩한다.



```

<!-- name을 utf-8 캐리터 셋을 이용해서 인코딩 한 값을 서버에 전송 -->
<%@ page contentType="text/html; charset=utf-8" %>
...
이름: <input type="text" name="name" size="10"><br>

<!--name 파라미터 값을 utf-8로 디코딩해서 가져옴-->
<%
    request.setCharacterEncoding("utf-8");
    String name = request.getParameter("name");
%>
...
<%= request.getParameter("name") %>

```

### (5) 톰캣에서 GET 방식 파라미터를 위한 인코딩 처리하기

- 톰캣 8 버전에서 GET 방식으로 전달된 파라미터 값을 읽어올 때 사용하는 캐릭터 셋의 기본 값은 UTF-8이다. 따라서 GET 방식으로 전송한 파라미터값을 UTF-8로만 인코딩한다면, 톰캣 8에 별도 설정을 하지 않아도 된다.

### 3.5.3 요청 헤더 정보의 처리

- HTTP 프로토콜은 헤더 정보에 부가적인 정보를 담도록 하고 있다. 예를 들어, 웹 브라우저는 웹 브라우저의 종류, 선호하는 언어에 대한 정보를 헤더에 담아서 전송한다.
- request 기본 객체가 제공하는 헤더 관련 메서드

메서드	리턴 타입	설명
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.

[chap03\viewHeaderList.jsp]

```
<%@ page contentType = "text/html; charset=utf-8" %>
<%@ page import = "java.util.Enumeration" %>
<html>
<head><title>헤더 목록 출력</title></head>
<body>
<%
    Enumeration headerEnum = request.getHeaderNames();
    while(headerEnum.hasMoreElements()) {
        String headerName = (String)headerEnum.nextElement();
        String headerValue = request.getHeader(headerName);
    }
<%>
<%= headerName %> = <%= headerValue %> <br>
<%
</body>
</html>
```

## 3.6 response 기본 객체

- 웹 브라우저에 보내는 응답 정보를 담는다.
- 주요 기능
  - 헤더 정보 입력
  - 리다이렉트 처리

### 3.6.1 웹 브라우저에 헤더 정보 전송하기

- 응답 정보에 헤더를 추가하는 기능을 제공한다.

- response 기본 객체가 제공하는 헤더 추가 메서드

메서드	리턴 타입	설명
addDateHeader(String name, long date)	void	name 헤더에 date를 추가한다. date는 1970년 1월 1일 이후 흘러간 시간을 1/1000초 단위로 나타낸다.
addHeader(String name, String value)	void	name 헤더에 value를 값으로 추가한다.
addIntHeader(String name, int value)	void	name 헤더에 정수 값 value를 추가한다.
setDateHeader(String name, long date)	void	name 헤더의 값을 date로 지정한다. date는 1970년 1월 1일 이후 흘러간 시간을 1/1000초 단위로 나타낸다.
setHeader(String name, String value)	void	name 헤더의 값을 value로 지정한다.
setIntHeader(String name, int value)	void	name 헤더의 값을 정수 값 value로 지정한다.

### 3.6.2 웹 브라우저 캐시 제어를 위한 응답 헤더 입력

- HTTP는 특수한 응답 헤더를 통해서 웹 브라우저가 응답 결과를 캐시 할 것인지에 대한 여부를 설정할 수 있다.
- response 기본 객체가 제공하는 헤더 추가 메서드

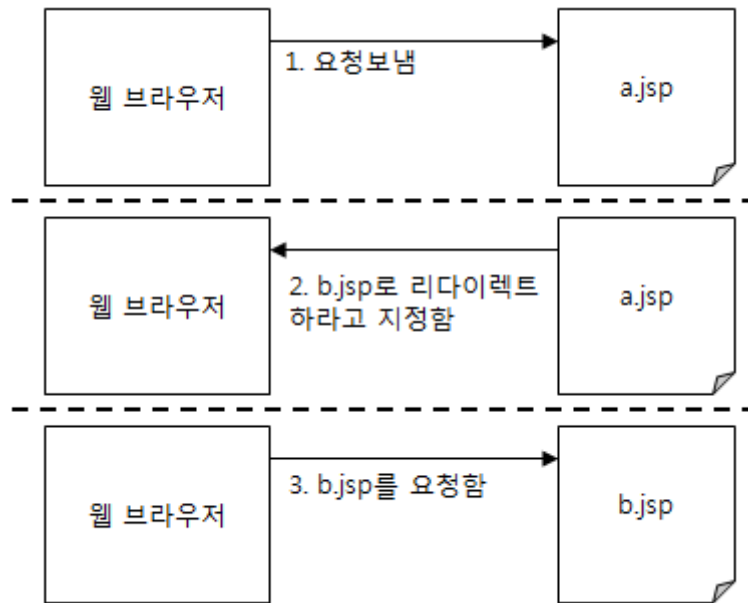
응답 헤더	설명
Cache-Control	이 헤더의 값을 'no-cache'로 지정하면 웹 브라우저는 응답 결과를 캐시하지 않는다.
Pragma	이 헤더의 값을 'no-cache'로 지정하면 웹 브라우저는 응답 결과를 캐시에 저장하지 않는다.
Expires	응답 결과의 만료일을 지정한다.

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



### 3.6.3 리다이렉트를 이용해서 페이지 이동하기

- 웹 서버가 웹 브라우저에게 다른 페이지로 이동하라고 응답하는 기능이다.



```

<%
    response.sendRedirect(String location);
%>

```

[chap03/login.jsp]

```

<%@ page contentType="text/html; charset=utf-8"%>
<%
    String id = request.getParameter("memberId");
    if (id != null && id.equals("madvirus")) {
        response.sendRedirect("/chap03/index.jsp");
    } else {
%>
<html>
<head>
<title>로그인에 실패</title>
</head>
<body>잘못된 아이디입니다.
</body>
</html>
<%
    }
%>

```

- `URLEncoder.encode()` 메서드를 사용하면 파라미터 값으로 사용될 문자열을 지정한 캐리터 셋으로 인코딩할 수 있다.

[chap03/redirectEncodingTest.jsp]

```

<%@ page pageEncoding="utf-8" %>
<%@ page import = "java.net.URLEncoder" %>
<%
    String value = "자바";
    String encodedValue = URLEncoder.encode(value, "utf-8");
    response.sendRedirect("/chap03/index.jsp?name=" + encodedValue);
%>

```

## 3.7 JSP 주석

[chap03/jspComment.jsp]

```
<%@ page contentType="text/html; charset=utf-8"%>
<html>
<head>
<title>JSP 주석</title>
</head>
<body>
    <!-- JSP 주석입니다. -->
    주석은 출력 결과에 포함되지 않습니다.
</body>
</html>
```