JSP 기초 #1

2018-03-05

이승진

**학습 목표**

- 한글이 깨지지 않도록 eclipse 인코딩을 UTF-8로 설정한다

- 강의 예제 소스 코드를 작성할 프로젝트를 이클립스에서 생성한다.

- JSP 작동 원리를 이해한다.

- JSP scriptlet tag를 이해한다.

- JSP expression tag를 이해한다.

- JSP에서 날짜를 출력하는 방법을 배운다.

**목차**

[1. 배경지식 2](#_Toc508029775)

[1) JSP 페이지를 실행할 때 한글이 깨진다면? 2](#_Toc508029776)

[2. 강의 예제 프로젝트 생성 3](#_Toc508029777)

[1) 폴더명 조심 3](#_Toc508029778)

[2) 프로젝트 생성 3](#_Toc508029779)

[3) hello.html 파일 생성 5](#_Toc508029780)

[4) hello.html 실행 과정 설명 6](#_Toc508029781)

[3. JSP 스크립릿 태그 (scriptlet tag) 8](#_Toc508029782)

[1) hello.jsp 파일 생성 8](#_Toc508029783)

[2) JSP 태그의 종류 10](#_Toc508029784)

[3) hello.jsp 실행 과정 설명 10](#_Toc508029785)

[4) hello.jsp 실행 결과 출력 12](#_Toc508029786)

[5) JSP 스크립릿 태그 (JSP scriptlet tag) 12](#_Toc508029787)

[6) hello2.html 13](#_Toc508029788)

[7) hello2.jsp 14](#_Toc508029789)

[4. JSP 표현식 태그 16](#_Toc508029790)

[1) JSP 표현식 태그 (JSP expression tag) 16](#_Toc508029791)

[2) hello3.jsp 18](#_Toc508029792)

[5. 날짜 출력 20](#_Toc508029793)

[1) Date1.java 20](#_Toc508029794)

[2) SimpleDateFormat 클래스 23](#_Toc508029795)

[3) date1.jsp 24](#_Toc508029796)

[6. 연습문제 26](#_Toc508029797)

[1) JSP 코드 분석 26](#_Toc508029798)

[2) 구현 실습 27](#_Toc508029799)

# 배경지식

## JSP 페이지를 실행할 때 한글이 깨진다면?

eclipse에서 소스 코드를 작성할 때 그 소스 코드 파일의 인코딩과

그 소스 코드 파일을 톰캣 웹서버가 읽어서 실행할 때의 인코딩과

실행 결과 데이터가 웹브라우저로 전달되어서 웹브라우저 화면에 페이지가 보일 때의 인코딩이

모두 같아야 한다. 그렇지 않으면 한글 문자가 처음 입력했던 그 문자가 아니라 이상한 문자로 출력되게 된다.

웹에서 주로 사용하는 문자 인코딩 두개는 다음과 같다.

(1) UTF-8 인코딩

(2) ANSI 인코딩 (MS949, EUC-KR 인코딩이라고도 부른다)

Java 개발 도구들의 인코딩을 UTF-8로 모두 통일하는 것이 안전하다.

- eclipse 개발도구의 인코딩도 UTF-8

- 톰캣 서버의 인코딩도 UTF-8

- 웹브라우저의 인코딩도 UTF-8

### eclipse 개발도구의 인코딩도 UTF-8

eclipse 에서 소스코드 파일을 저장할 때, UTF-8 인코딩으로 저장해야 한다.

### 톰캣 웹서버의 인코딩도 UTF-8

톰캣 서버에서 소스코드 파일을 읽어서 실행할 때, UTF-8 인코딩을 사용해야 한다.

### 웹브라우저의 인코딩도 UTF-8

서버로부터 전달된 문자열을 웹브라우저 창에 표시할 때, UTF-8 인코딩을 사용해야 한다.

Windows 앱들의 기본 인코딩은 ANSI (MS949) 이다.

Windows 메모장에서 ANSI 인코딩으로 저장한 소스 파일을, eclipse에서 UTF-8로 인코딩으로 해석해서 열면

알파벳 문자는 괜찮은데 한글만 다 이상한 문자로 출력될 것이다.

# 강의 예제 프로젝트 생성

강의에서 사용할 예제를 작성하기 위한 Dynamic Web Project를 생성하자.

## 폴더명 조심

Java 도구들 중에서 한글 파일명이나 한글 폴더명을 제대로 인식 못하는 것들이 있다.

한글 뿐만 아니라 공백 문자가 경로명(path)에 포함된 것도 인식 못하는 것들이 있다.

프로젝트를 생성할 때, 프로젝트 폴더의 경로명(path)에 한글이나 공백 문자가 없어야 한다.

바탕화면에 프로젝트를 생성하지 말자.

옛날 버전의 Windows에서는 바탕화면 폴더의 이름이 "바탕화면"이다.

(최근 버전의 Windows에서 바탕화면 폴더의 이름은 "Desktop" 이다.)

Windows 사용자 계정(로그인 아이디)에 한글이나 공백이 있어도 안된다.

계정 이름으로 사용자의 홈(home) 폴더가 만들어지는데,

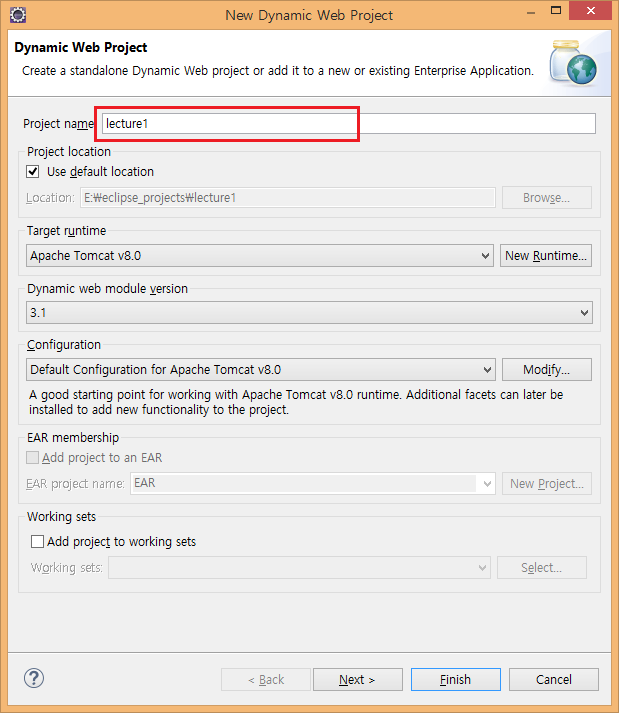
여기에 설정 파일들이나 데이터 파일들이 저장되기 때문이다.

## 프로젝트 생성

이클립스 메뉴:

File - New - **Dynamic Web Project**

File – New – Other - Web - **Dynamic Web Project**

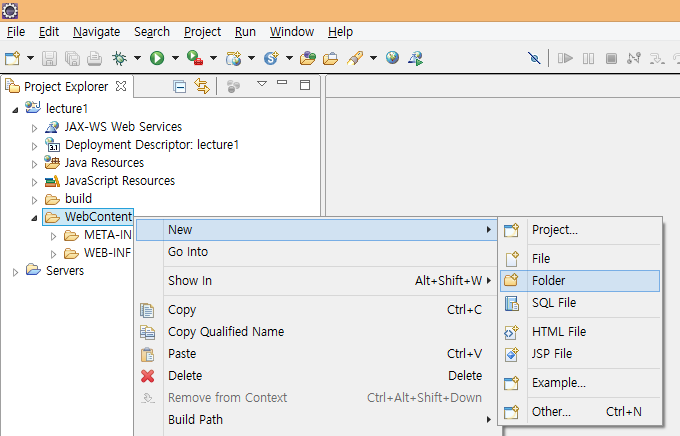


Project name: lecture1

Finish 버튼 클릭

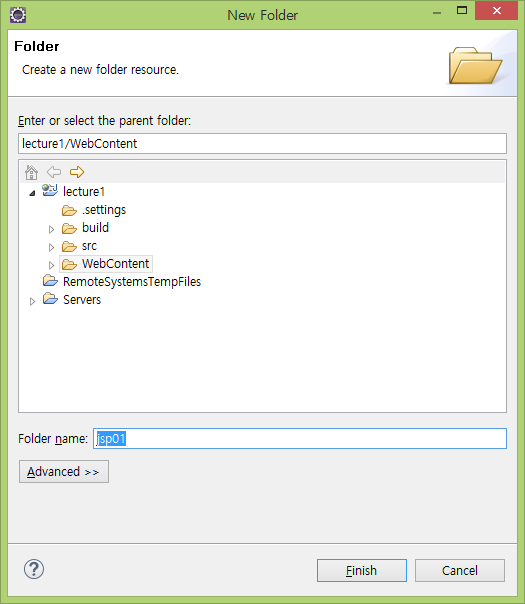
### 폴더 생성

강의 예제 파일들을 작성할 jsp01 폴더를 생성하자.



Project Explorer창에서 lecture1 프로젝트의 아래 WebContent 폴더를

마우스 오른쪽 버튼 클릭 - New - Folder 클릭하면 아래의 대화상자가 나타난다.



폴더명으로 jsp01을 입려하고 Finish 클릭.

## hello.html 파일 생성

WebContent 아래에 방금 생성한 jsp01 폴더를 마우스 오른쪽 버튼 클릭 - New - HTML File 클릭하면

[New HTML File] 대화상자가 나타난다.

이 대화상자에서

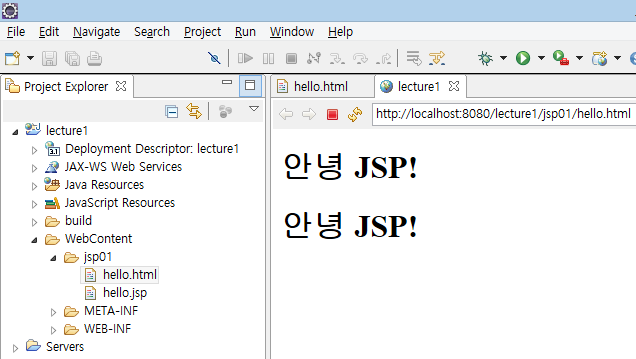
File name: hello.html 입력하고, Finish 버튼 클릭.

hello.html 파일을 다음과 같이 수정한다.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  </head>  <body>  <h1>안녕 JSP!</h1>  <H1>안녕 JSP!</H1>  </body>  </html> |

### hello.html 실행

hello.html 편집창에서 Ctrl+F11 단축키를 누르면 hello.html가 실행된다.



## hello.html 실행 과정 설명

hello.html 파일은 정적인 문서(static document)이다.

이클립스의 편집창에서 Ctrl+F11 단축키를 누르면, 이클립스 내부에서 다음 절차가 진행된다.

1) 이클립스에서 작성중이던 현재 프로젝트가 컴파일되어

톰캣 서버에서 실행될 수 있도록 배치(publish) 된다.

2) 이클립스의 내부 창에서 웹브라우저가 실행된다.

3) 이 웹브라우저는 이클립스 편집창에서 편집 중이던 파일에 대한

서버 URL을 인터넷을 통해 톰캣 서버에 요청한다. (http request)

URL [http://localhost:8080/lecture1/jsp01/hello.html]

톰캣 서버가 웹브라우저와 같은 컴퓨터에 있어도, 이 둘은 인터넷을 통해서 통신한다.

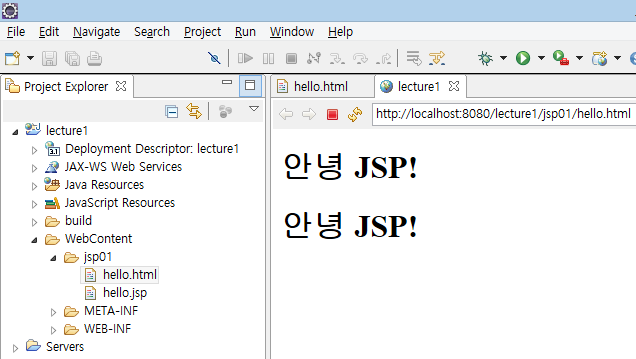
4) 톰캣 서버는 요청된 URL의 파일을 찾는다.

5) 요청된 파일(hello.html)이 정적인 문서이므로

톰캣 서버는 그 파일의 내용을 그대로 웹브라우저에게 인터넷을 통해 전달한다. (http response)

6) hello.html 파일의 내용을 전달받은 웹브라우저는 그 내용을 웹브라우저 창에 그린다.

그 결과는 아래 화면이다.



위 그림에서 오른쪽 웹브라우저 창에 hello.html 파일의 내용이 표시되었다.

그런데 웹브라우저는 웹서버의 디스크에 저장된 hello.html 파일의 내용을 알지 못한다.

웹브라우저는 그저 웹서버에 URL http://localhost:8080/lecture1/jsp01/hello.html을 요청하였고,

그 요청에 대한 응답으로 웹서버가 인터넷을 통해 보내온 내용을 받아서

웹브라우저 창에 그렸을 뿐이다.

웹서버가 파일의 내용을 그대로 보내주었는지 아니면 내용중에서 무엇을 수정하고 보냈는지

웹브라우저는 알지 못한다. 그저 받은 내용 그대로 창에 그릴뿐이다.

웹브라우저와 웹서버가 같은 컴퓨터에서 실행되지 않아도 상관없다.

이 둘은 인터넷으로 통신하기 때문이다.

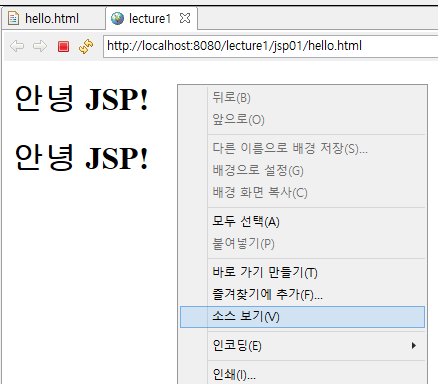
만약 웹서버가 다른 컴퓨터에서 실행되고 있다면,

웹브라우저가 웹서버에 요청하는 URL에 서버의 IP 주소가 들어있어야 한다.

http://**서버IP주소**:8080/lecture1/jsp01/hello.html

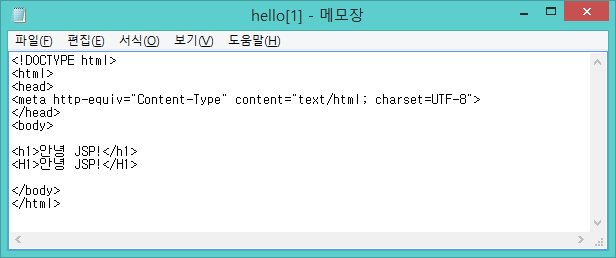
웹브라우저가 웹서버로부터 받은 내용이 무엇인지 보는 방법은 다음과 같다.

웹브라우저 창에서 마우스 오른쪽 버튼을 클릭하면 다음과 같은 메뉴가 보인다.



이 메뉴에서 [소스보기]를 클릭한다.

### hello.html 실행결과 소스보기



위 내용이 바로 URL http://localhost:8080/lecture1/jsp01/hello.html 요청에 대해서

웹서버가 응답으로 보냈고, 그래서 웹브라우저가 받은 내용이다.

hello.html 파일의 내용과 일치함을 확인할 수 있다.

### 웹서버와 인터넷 통신

개발할 때는 보통 한 컴퓨터에 이클립스와 톰캣 웹서버를 같이 실행한다.

이클립스에서 웹페이지를 실행하면 이클립스 내부 창에서 웹브라우저가 실행되고 이 웹브라우저가 톰캣 웹서버와 통신한다.

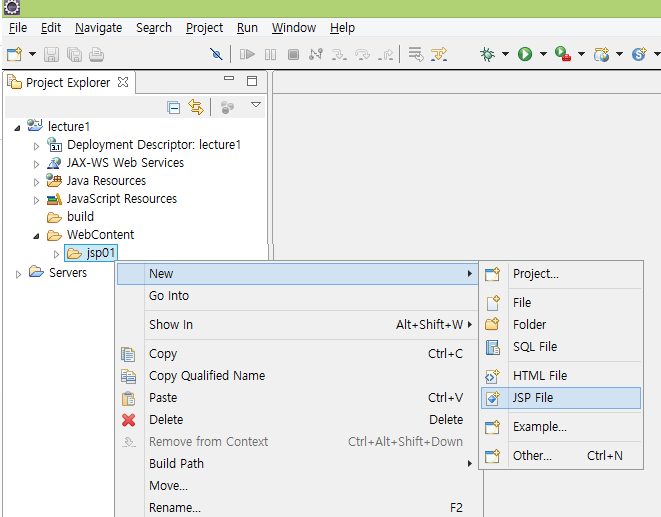
비록 웹브라우저 창과 톰캣 웹서버가 한 컴퓨터에서 실행되지만, 이들의 요청과 응답은 HTTP 프로토콜 규격대로 인터넷 통신(TCP/IP)으로 진행된다.

그래서 Windows 방화벽이 톰캣 웹서버의 인터넷 열결을 허용하도록 설정하지 않으면, 같은 컴퓨터에 있는 톰캣 웹서버인데도 연결되지 않는다.

개발이 끝난 웹프로그램은 24시간 365일 실행되는 서버 컴퓨터의 웹서버에 설치되어 실행된다.

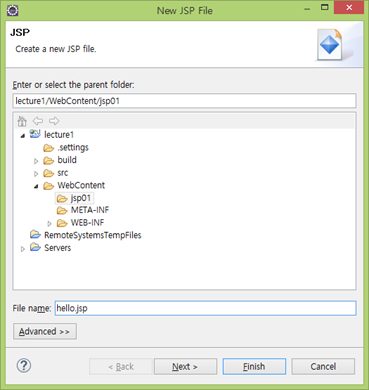
# JSP 스크립릿 태그 (scriptlet tag)

## hello.jsp 파일 생성



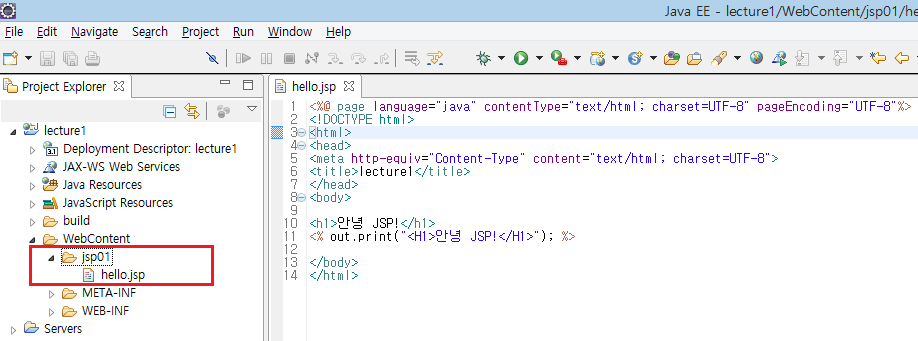
WebContent 아래에 방금 생성한 jsp01 폴더를 마우스 오른쪽 버튼 클릭 - New - JSP File 클릭하면

아래의 대화상자가 나타난다.



File name: hello.jsp 입력하고, Finish 버튼 클릭.

WebContent/jsp01 폴더 아래에 hello.jsp가 생성된다.

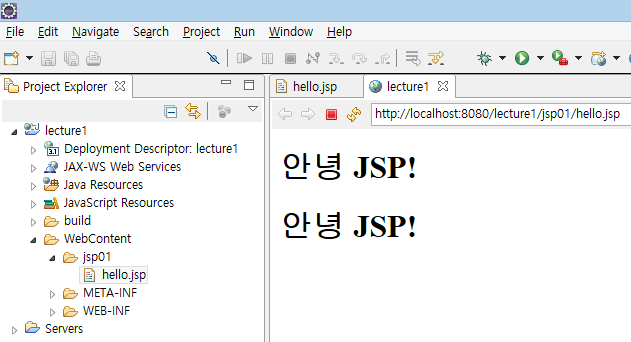


hello.jsp 파일을 다음과 같이 수정하자.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  </head>  <body>  <h1>안녕 JSP!</h1>  <% out.print("<H1>안녕 JSP!</H1>"); %>  </body>  </html> |

### hello.jsp 실행

hello.jsp 편집창에서 Ctrl+F11 단축키를 누르면 hello.jsp가 실행된다.



## JSP 태그의 종류

|  |  |
| --- | --- |
| 형태 | 이름 |
| **<%@** 선언 **%>** | JSP 선언 태그 |
| **<%**  Java Statement **%>** | JSP Scriptlet 태그 |
| **<%=** Java Expression **%>** | JSP Expression 태그 |

웹브라우저에서 JSP 파일의 URL을 서버에 요청하면

서버에서 JSP 파일이 실행되고

그 실행결과 출력이 웹브라우저로 전달된다.

JSP 선언 태그 부분은 출력되지 않는다.

JSP Scriptlet 태그 부분은, out.print(...) 메소드로 출력한 내용만 출력되어 웹브라우저로 전달된다.

JSP Expression 태그 부분은, 그 표현식(java expression)의 값이 출력되어 웹브라우저로 전달된다.

JSP 태그가 아닌 부분은 있는 그대로 출력되어 웹브라우저로 전달된다.

### Java 문장 (Java Statement)

변수 선언문, 대입문, for 문, while 문, if 문, switch 문 등

### Java 표현식 (Java Expression)

상수, 변수, 계산식 등

## hello.jsp 실행 과정 설명

확장자가 JSP인 hello.jsp 파일은 동적인 문서(dynamic document)이다.

이클립스의 hello.jsp 편집창에서 Ctrl+F11 단축키를 누르면, 이클립스 내부에서 다음 절차가 진행된다.

1) 이클립스에서 작성중이던 현재 프로젝트가 컴파일되어

톰캣 서버에서 실행될 수 있도록 배치(publish) 된다.

2) 이클립스의 내부 창에서 웹브라우저가 실행된다.

3) 이 웹브라우저는 이클립스 편집창에서 편집 중이던 파일을 실행하기 위해

그 파일에 대한 URL을 인터넷을 통해 톰캣 웹서버에 요청한다.

URL [http://localhost:8080/lecture1/jsp01/hello.jsp]

4) 톰캣 웹서버는 요청된 URL의 파일을 찾는다.

5) 요청된 파일(hello.jsp)이 동적인 문서이므로

톰캣 웹서버는 그 파일의 내용중에서 Java 코드에 해당하는 부분을 실행한다.

6) 웹서버는 확장자가 jsp인 파일의 내용을 다음과 같이 구분하여 웹브라우저에게 전달한다.

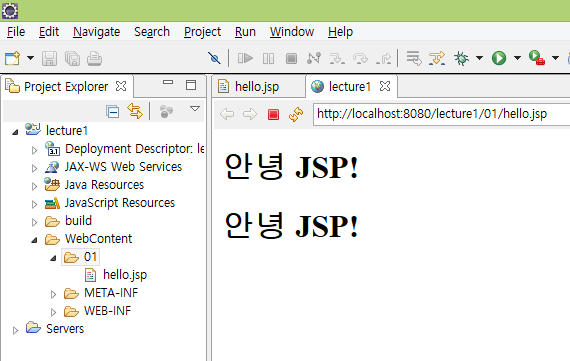
- jsp 파일의 내용 중에서 java 코드에 해당하지 않은 부분은 읽은 내용 그대로 전달한다.

- java 코드에 해당하는 부분은, 그 java 코드가 out 객체의 print 메소드를 호출하여 출력한 내용만

웹브라우저에게 전달한다. java 코드는 웹브라우저에게 전달하지 않는다.

6) hello.jsp 파일의 실행 결과 전달된 내용을 받은 웹브라우저는 그 내용을 웹브라우저 창에 그린다.

그 결과는 다음 화면이다.

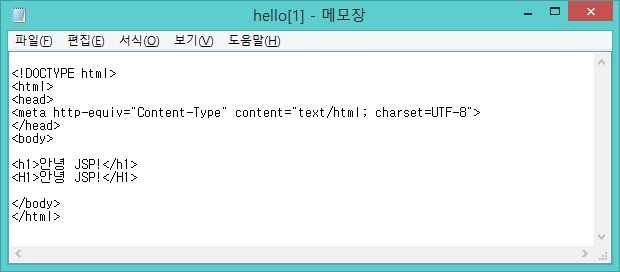


위 그림에서 오른쪽 웹브라우저 창에 hello.jsp 파일의 실행 결과로 출력된 내용이 표시되었다.

웹브라우저 창에서 마우스 오른쪽 버튼을 클릭하고 [소스보기]를 클릭하여

웹서버로부터 웹브라우저에게 전달된 hello.jsp 파일의 실행 결과를 확인하자.

### hello.jsp 실행결과 소스보기



위 내용이 바로 URL [http://localhost:8080/lecture1/jsp01/hello.jsp] 요청에 대해서

웹서버가 hello.jsp를 실행하고,

그 출력을 웹서버가 웹브라우저에게 보내고

웹브라우저가 받은 창에 그린 내용이다.

hello.jsp 파일의 내용이 수정되어 전달되었음을 확인할 수 있다.

## hello.jsp 실행 결과 출력

서버에서 hello.jsp를 실행하고, 실행 결과를 웹브라우저에 출력한다.

|  |  |
| --- | --- |
| 1 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%> |

이 부분은 출력되지 않는다.

JSP 파일의 선두에 언제나 위 한 줄이 포함되어 있어야 한다. 그래야 한글이 깨지지 않는다.

|  |  |
| --- | --- |
| 2  3  4  5  6  7  8  9 | <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  </head>  <body>  <h1>안녕 JSP!</h1> |

이 부분은 그대로 웹브라우저에 출력된다.

JSP 태그가 아닌 부분은 그대로 출력되어 웹브라우저에 전달된다.

hello.jsp 줄11은 java 코드가 실행되어 out.print(...) 메소드로 출력된 내용만 웹브라우저에 전달된다

|  |  |
| --- | --- |
| 10 | <% out.print("<H1>안녕 JSP!</H1>"); %> |

출력된 내용

|  |  |
| --- | --- |
|  | <H1>안녕 JSP!</H1> |

<% ... %> → 이것은 JSP 스크립릿 태그라고 부른다. (JSP scriptlet tag)

톰캣 웹서버는 이 태그 내부의 Java 소스코드를 실행하고,

out.print(...) 메소드로 출력한 내용만 웹브라우저에 전달한다.

|  |  |
| --- | --- |
| 11  12  13 | </body>  </html> |

이 부분은 그대로 웹브라우저에 출력된다.

JSP 태그가 아닌 부분은 그대로 출력되어 웹브라우저에 전달된다.

## JSP 스크립릿 태그 (JSP scriptlet tag)

JSP 파일에서 <% ... %> 부분을 JSP scriptlet tag라고 부른다.

이 태그 사이에는 자바 문장(Java Statement)만 올 수 있다.

웹 서버에서 JSP 파일이 실행될 때 스크립릿 태그 내부의 자바 코드가 실행된다.

실행결과 out.print(..) 나 out.println(..) 메소드로 출력한 내용만 웹브라우저에 전달될 뿐이고

스크립릿 태그 내부의 Java 코드는 웹브라우저로 전달되지 않는다.

이때 주의할 부분은, out.print(..) 메소드로 출력해야 한다는 점이다.

System.out.print(..) 메소드로 출력하면 안된다.

System.out.print(..) 메소드로 출력한 것은 웹브라우저로 전달되지 않고,

이클립스 console창에 표시된다.

이클립스 메뉴에서 Window - Show View - Console 메뉴를 클릭하면 console 창이 보인다.

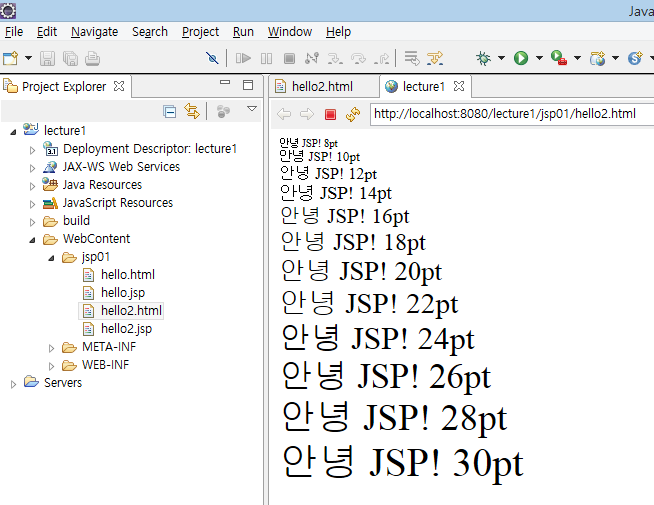
이클립스에서 개발 도중 디버깅을 위해 변수값을 출력해보고 싶다면,

System.out.print(..) 메소드로 console 창에 출력하는 것도 하나의 방법이다.

## hello2.html

### WebContent/jsp01/hello2.html

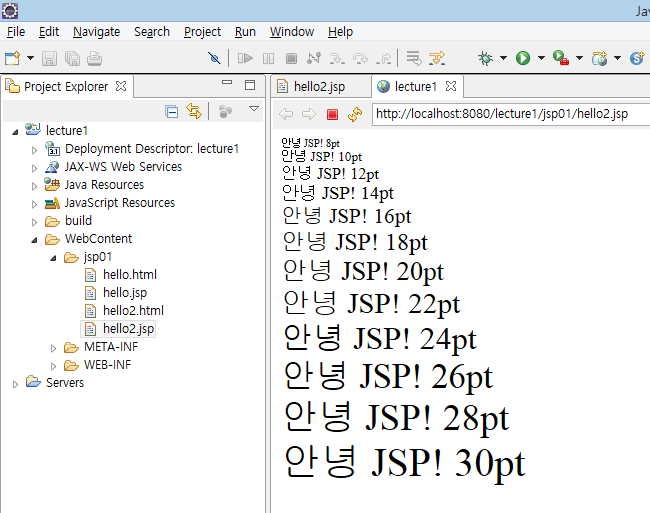
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46 | <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  </head>  <body>  <div style="font-size: 8pt;">  안녕 JSP! 8pt  </div>  <div style="font-size: 10pt;">  안녕 JSP! 10pt  </div>  <div style="font-size: 12pt;">  안녕 JSP! 12pt  </div>  <div style="font-size: 14pt;">  안녕 JSP! 14pt  </div>  <div style="font-size: 16pt;">  안녕 JSP! 16pt  </div>  <div style="font-size: 18pt;">  안녕 JSP! 18pt  </div>  <div style="font-size: 20pt;">  안녕 JSP! 20pt  </div>  <div style="font-size: 22pt;">  안녕 JSP! 22pt  </div>  <div style="font-size: 24pt;">  안녕 JSP! 24pt  </div>  <div style="font-size: 26pt;">  안녕 JSP! 26pt  </div>  <div style="font-size: 28pt;">  안녕 JSP! 28pt  </div>  <div style="font-size: 30pt;">  안녕 JSP! 30pt  </div>  </body>  </html> |



## hello2.jsp

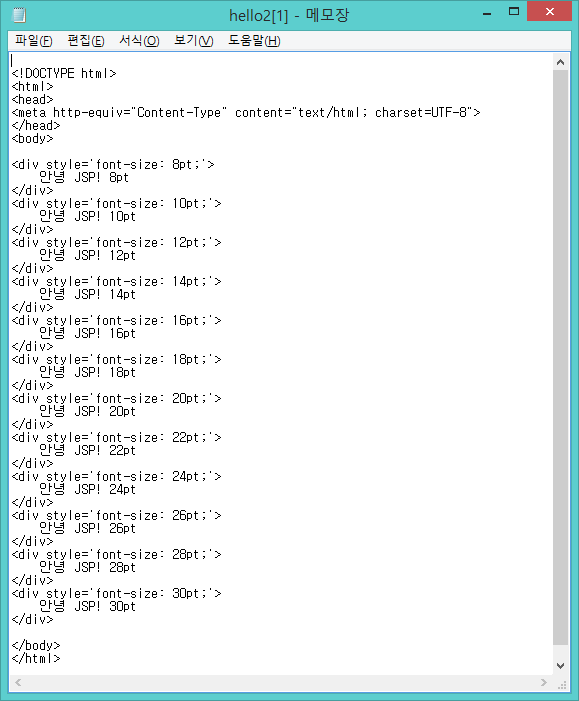
### WebContent/jsp01/hello2.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  </head>  <body>  <%  for (int i=8; i <= 30; i+=2) {  out.println("<div style='font-size: " + i + "pt;'>");  out.println(" 안녕 JSP! " + i + "pt");  out.println("</div>");  }  %>  </body>  </html> |



hello2.jsp의 실행결과 출력되는 내용을 마우스 오른쪽 버튼을 클릭하고 [소스보기] 메뉴를 클릭하여 확인해보자.

### hello2.jsp 실행결과 소스보기



hello2.jsp의 자바 소스코드의 실행 결과로 위와 같은 내용이 출력되어 웹브라우저에 전달되었다.

위 출력과 hello2.jsp의 Java 코드를 비교해 보자.

JSP 파일의 Java 코드만 보고, 출력될 내용이 무엇인지 정확하게 판단할 수 있어야 한다.

# JSP 표현식 태그

## JSP 표현식 태그 (JSP expression tag)

JSP 파일에서 <%= ... %> 부분을 JSP 표현식 태그 (JSP expression tag)라고 부른다.

이 태그 사이에는 자바 표현식(Java Expression)만 올 수 있다.

표현식이 아닌 자바 문장(Java Statement)은 올 수 없다.

표현식(expression)이란, 값이 있는 자바 구문을 말한다.

변수에 대입하는 형태로 구현할 수 있는 자바 구문은, 값이 있는 것이다.

변수값으로 변수에 대입할 수 있는 자바 구문은, 값이 있는 것이므로 표현식(expression)에 해당한다.

### 대입 문장

|  |
| --- |
| i = 3;  i = a;  i = 3 + a;  s = "hello" + b;  i = Math.max(a, b); |

위 Java 코드는 변수에 어떤 값을 대입하는 문장들이다.

대입 문장에서 = 연산자 오른쪽에 있는 부분이, 변수에 대입할 값이다.

이렇게 변수에 대입할 수 있는 Java 코드 부분은 Java 표현식이다.

### Java 표현식

|  |
| --- |
| 3  a  3 + a  "hello" + b  Math.max(a, b) |

표현식은 세미콜론(;) 문자를 포함하지 않는다.

if 문장이나 for 문장을 변수에 대입하는 것은 불가능하다.

따라서 if 문이나 for 문은 표현식(expression)이 아니고 문장(statement)이다.

int 변수에 대입할 수 있는 정수 상수는 표현식이다.

int 변수에 대입할 수 있는 계산식도 표현식이다.

String 변수에 대입할 수 있는 문자열 상수도 표현식이다.

리턴값이 있는 메소드 호출도 그 리턴값을 변수에 대입할 수 있으므로 표현식이다.

즉 변수에 대입 가능한 자바 구문은 모두 표현식이다.

### 표현식의 예:

3.14159 → double 상수 값

13 - 4 → 계산식의 결과 값

Math.abs(-3) → 메소드 호출의 리턴값

Math.PI → public static final 멤버 변수의 값

"hello" → String 상수 값

"hello " + "world" → 계산식의 결과 값

"hello".toUpperCase() → 메소드 호출의 리턴값

Math 클래스의 abs 메소드는 절대값을 리턴한다.

Math 클래스의 public static final 멤버 변수 PI는 수학의 파이값 상수이다.

String 클래스의 toUpperCase 메소드는 this 문자열을 대문자로 변환한 새 문자열을 리턴한다

### JSP 표현식 태그 출력의 예:

웹 서버에서 JSP 파일이 실행될 때 JSP 표현식 태그 내부의 표현식의 값이 웹브라우저에 출력된다.

JSP 스크립릿 태그에서 출력할 때 out.print(..) 메소드 호출이 필요하지만,

JSP 표현식 태그는 그 식의 값이 그냥 출력되므로 out.print(..) 메소드 호출을 할 필요 없다.

out.print(...) 메소드는 리턴값이 없으므로 표현식이 아니다. 따라서 JSP 표현식 태그에 올 수 없다.

|  |  |
| --- | --- |
| JSP 표현식 태그 | 실행 결과 출력 |
| <%= 3.14159 %> | 3.14159 |
| <%= 13 - 4 %> | 9 |
| <%= Math.abs(-3) %> | 3 |
| <%= Math.PI %> | 3.14159265358979 |
| <%= "hello" %> | hello |
| <%= "hello " + "world" %> | hello world |
| <%= "hello".toUpperCase() %> | HELLO |

### 표현식의 값 출력 방법

(1) JSP 스크립릿 태그를 이용한 표현식의 값 출력

<% out.print( 표현식 ); %>

(2) JSP 표현식 태그를 이용한 표현식의 값 출력

<%= 표현식 %>

위 두 JSP 태그의 출력은 모두 표현식의 값이다. 출력 결과가 동일하다.

표현식의 값을 출력할 때는 JSP 표현식 태그가 간결해서 좋다.

## hello3.jsp

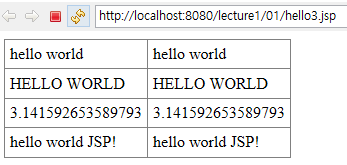
### WebContent/jsp01/hello3.jsp

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  <style>  table { border-collapse: collapse; }  td { padding: 5px; border: solid 1px gray; }  </style>  </head>  <body>  <%  String s1 = "hello world";  String s2 = " JSP!";  %>  <table>  <tr>  <td><% out.print(s1); %></td>  <td><%= s1 %></td>  </tr>  <tr>  <td><% out.print(s1.toUpperCase()); %></td>  <td><%= s1.toUpperCase() %></td>  </tr>  <tr>  <td><% out.print(Math.PI); %></td>  <td><%= Math.PI %></td>  </tr>  <tr>  <td><% out.print(s1 + s2); %></td>  <td><%= s1 + s2 %></td>  </tr>  </table>  </body>  </html> |

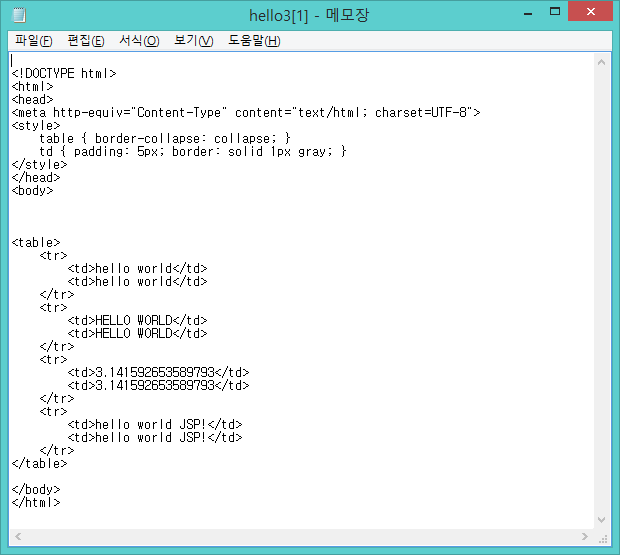
동일한 표현식을 JSP 스크립릿 태그에서 out.print(..) 메소드로 출력하고, JSP 표현식 태그에서 출력하여,

두 출력의 결과가 동일한지 비교해 보자.

### hello3.jsp 실행결과



### hello3.jsp 실행결과 소스보기



JSP 스크립릿 태그로 출력한 내용과, JSP 표현식 태그로 출력한 내용이 동일함을 확인할 수 있다.

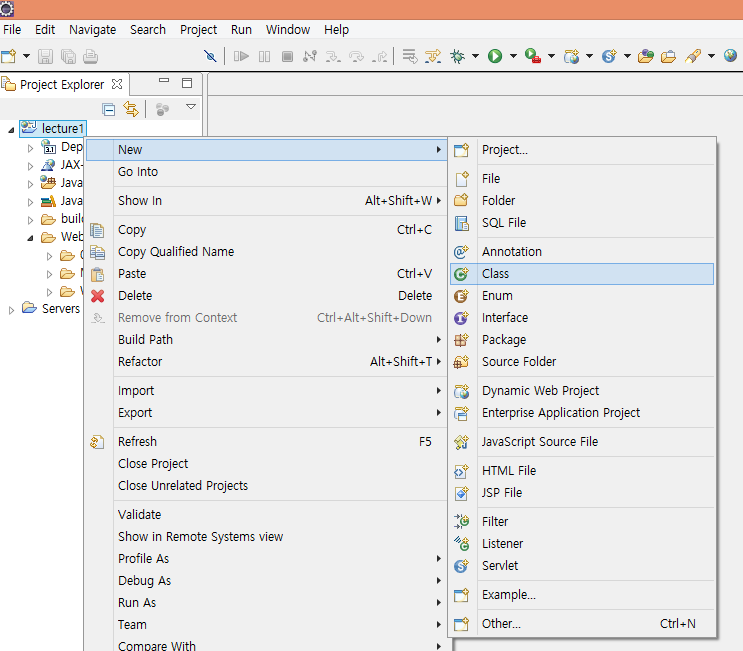
# 날짜 출력

## Date1.java

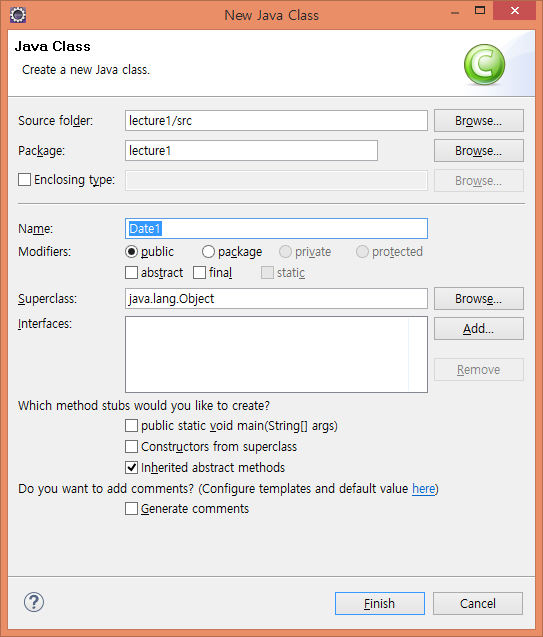
lecture1 프로젝트에 Date1 자바 클래스를 생성하자.

이클립스 화면 왼쪽의 Project Explorer창에서 lecture1 프로젝트를 마우스 오른쪽 버튼으로 클릭하고

메뉴에서 New - Class를 클릭한다.



New Java Class 대화상자에서 Name: 칸에 Date1 을 입력하고 Finish 버튼을 클릭한다.



생성된 Data1.java 파일에 다음과 같이 입력하자

### src/lecture1/Date1.java

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | package lecture1;  import java.util.Date;  import java.text.SimpleDateFormat;  public class Date1 {  public static void main(String[] args) {  Date today = new Date();    SimpleDateFormat f1 = new SimpleDateFormat("yyyy-MM-dd a hh:mm:ss");  String s1 = f1.format(today);  System.out.println(s1);    SimpleDateFormat f2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  String s2 = f2.format(today);  System.out.println(s2);  SimpleDateFormat f3 = new SimpleDateFormat("yy-M-d H:m:s");  String s3 = f3.format(today);  System.out.println(s3);  }  } |

|  |
| --- |
| import java.util.Date;  import java.text.SimpleDateFormat; |

Date 클래스, SimpleDateFormat 클래스를 사용하려면 이 import 문이 필요하다.

아래 그림과 같이 import 문장을 주석으로 막아보자.



import 선언이 없으면 위와 같이 SimpleDatFormat 클래스명에 컴파일 에러를 뜻하는 빨간색 밑줄이 표시된다.

그 빨간색 밑줄에 마우스 커스를 놓고 조금 기다리면

에러메시지에 대한 설명과 해결책을 제시하는 팝업 상자가 위와 같이 나타난다.

import 선언이 없으면 발생하는 에러 메시지는 "클래스명 cannot be resolved to a type" 이다.

자주 발생하는 에러이므로, 이 에러 메시지를 기억해 두자.

import 선언을 안했거나, 아니면 클래스명에 오타가 있을 때 이 컴파일 에러가 발생한다.

오타가 아니라면 import 선언을 추가하면 되는데,

위 팝업 상자에서 "import 'SimpleDateFormat' (java.text)"를 클릭하면 자동 추가된다.

|  |
| --- |
| Date today = new Date(); |

Date 클래스 객체를 생성한다. 생성된 Date 객체에는 생성된 날짜와 시각이 들어있다.

|  |
| --- |
| SimpleDateFormat f1 = new SimpleDateFormat("yyyy-MM-dd a hh:mm:ss"); |

Date 객체에 들어있는 날짜와 시각을 원하는 형태로 출력하려면,

SimpleDateFormat 클래스 객체가 필요하다.

SimpleDateFormat 클래스의 생성자 파라미터로 날짜와 시각의 형태를 지정한다.

|  |
| --- |
| String s1 = f1.format(today); |

SimpleDateFormat 클래스의 format 메소드를 호출한다.

이 메소드의 파라미터는 날짜와 시각을 가지고 있는 Date 객체이다.

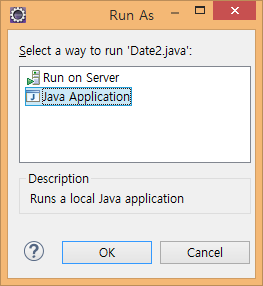
이 메소드는 Date 객체에 들어있는 날짜와 시각을 정해진 형태의 문자열로 만들어 리턴한다.

|  |
| --- |
| System.out.println(s1); |

문자열 s1의 값을 화면(console)에 출력한다.

### Date1.java 실행

Date1.java의 편집창에서 Ctrl+F11 단축키를 눌러서 실행한다.



Run As 대화상자에서 "Java Application"을 선택한다.

"Run on Server"는 톰캣 웹서버에서 JSP 파일을 실행할 때 선택한다.

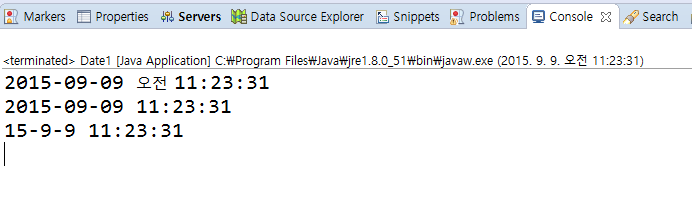
Data1.java의 public static main(String[] args) 메소드를 실행하려면, "Java Application"을 선택해야 한다.

이 경우에는 톰캣 웹서버를 거치지 않고, 바로 화면(console)에서 실행된다.

Date1.java의 실행결과 System.out.print(..) 메소드로 출력한 내용은

이클립스의 console창에 출력된다.

메뉴: Windows - Show View - Console



## SimpleDateFormat 클래스

Date 객체에 들어있는 날짜와 시각을 원하는 형태의 문자열로 만들기 위해 사용하는 클래스이다.

생성자 파라미터로 날짜와 시각의 변환 형태를 지정한다.

형태 지정 방법

|  |  |
| --- | --- |
| yyyy | 연도를 4자리 문자열로 (예: 2014) |
| yy | 연도를 2자리 문자열로 (예: 14) |
| MM | 월을 2자리 문자열로 (예: 08, 09, 10, 11, 12) |
| M | 월을 1~2자리 문자열로 (예: 8, 9, 10, 11, 12) |
| dd | 일을 2자리 문자열로 (예: 08, 09, 10, 11, 12) |
| d | 일을 1~2자리 문자열로 (예: 8, 9, 10, 11, 12) |
| E | 요일 (예: 월, 화, 수, 목, 금, 토, 일) |
| a | (예: 오전, 오후) |
| h | 1부터 12까지 시각을 1~2자리 문자열로 (예: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12) |
| hh | 1부터 12까지 시각을 2자리 문자열로 (예: 01, 02, 03, 04, 05, 06, 07, 08, 09, 10, 11, 12) |
| H | 0부터 23까지 시각을 1~2자리 문자열로 (예: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10 ... 20, 21, 22, 23) |
| HH | 0부터 23까지 시각을 2자리 문자열로 (예: 00, 01, 02, 03, 04, 05, 06, 07, ... 20, 21, 22, 23) |

|  |  |
| --- | --- |
| **SimpleDateFormat 클래스 생성자 파라미터의 예** | **format 메소드 리턴 값의 예** |
| yyyy-MM-dd a hh:mm:ss | 2014-07-10 오후 01:40:23 |
| yy-MM-dd E HH:mm:ss | 14-07-10 목 13:40:23 |
| yyyy년 M월 d일 E요일 H시 m분 s초 | 2014년 7월 10일 목요일 13시 40분 23초 |
| yyyy-MM-dd | 2014-07-10 |
| HH:mm:ss | 13:40:23 |

## date1.jsp

날짜 출력을 JSP에서도 구현해보자.

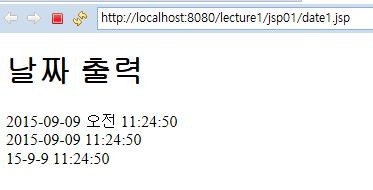
### WebContent/jsp01/date1.jsp 생성

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>  <%@ page import="java.util.Date, java.text.SimpleDateFormat" %>  <!DOCTYPE html>  <html>  <head>  <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">  </head>  <body>  <%  Date today = new Date();    SimpleDateFormat f1 = new SimpleDateFormat("yyyy-MM-dd a hh:mm:ss");  String s1 = f1.format(today);    SimpleDateFormat f2 = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");  String s2 = f2.format(today);  SimpleDateFormat f3 = new SimpleDateFormat("yy-M-d H:m:s");  String s3 = f3.format(today);  %>  <h1>날짜 출력</h1>  <div>  <%= s1 %>  </div>  <div>  <%= s2 %>  </div>  <div>  <%= s3 %>  </div>  </body>  </html> |

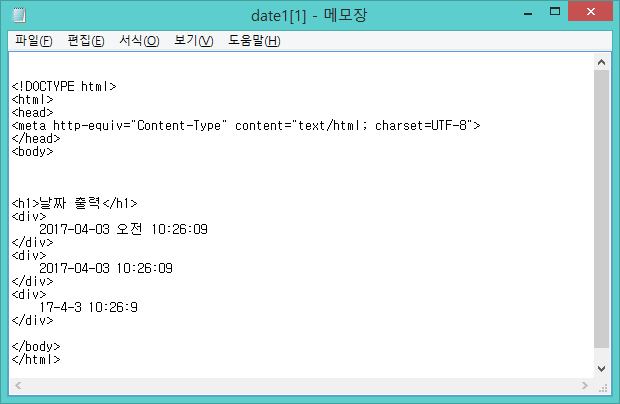
|  |
| --- |
| <%@ page import="java.util.Date, java.text.SimpleDateFormat" %> |

JSP 파일에서는 import를 위와 같이 구현해야 한다.

### 실행결과



### date1.jsp 실행결과 소스보기



톰캣 웹서버에서 date1.jsp의 실행결과로 출력되어 웹브라우저에 전달된 내용은 위와 같다.

# 연습문제

## JSP 코드 분석

아래 코드의 실행 결과 출력되는 내용을 적으시오.

<%

String s1 = "hello";

String s2 = "world";

out.println(s1);

out.println(s2);

out.println(s1 + s2);

out.println("s1");

out.println("s2");

out.println("s1" + "s2");

out.println("s1 + s2");

%>

<%= s1 %>

<%= s2 %>

<%= s1 + s2 %>

<%= "s1" %>

<%= "s2" %>

<%= "s1" + "s2" %>

<%= "s1 + s2" %>

아래 코드의 실행 결과 출력되는 내용을 적으시오.

4 x 7 = <%= 4 \* 7 %>

4 x 8 = <%= 4 \* 8 %>

4 x 9 = <%= 4 \* 9 %>

아래 코드의 실행 결과 출력되는 내용을 적으시오.

<% int i = 4; %>

4 x 7 = <%= i \* 7 %>

4 x 8 = <%= i \* 8 %>

4 x 9 = <%= i \* 9 %>

아래 코드의 실행 결과 출력되는 내용을 적으시오.

<%

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

out.print("4 x " + i);

out.print(" = ");

out.println(4 \* i);

}

%>

아래 코드의 실행 결과 출력되는 내용을 적으시오.

<table border=1>

<tr>

<%

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

out.print(" ");

out.print("<td>");

out.print(i);

out.println("</td>");

}

%>

</tr>

</table>

아래 코드의 실행 결과 출력되는 내용을 적으시오.

<table border=1>

<tr>

<%

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

out.println(" <td>" + i + "</td>");

}

%>

</tr>

</table>

아래 코드의 실행 결과 출력되는 내용을 적으시오.

<table border=1>

<% for (int i=0; i <= 3; ++i) {

out.println(" <tr>");

for (int j=1; j <= 4; ++j) {

out.println(" <td>" + (i \* 4 + j) + "</td>");

}

out.println(" </tr>");

}

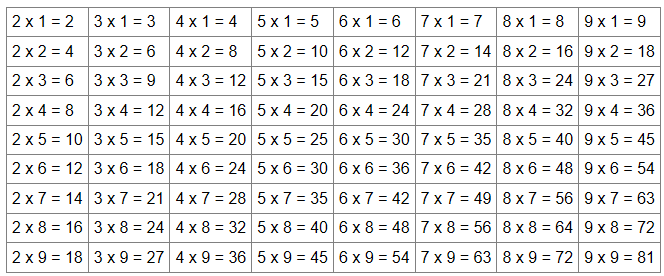
%>

</table>

위 코드들의 실행 결과 출력을 계산해서 적어보라.

그리고 나서, 위 코드들을 실행해 보고 실행 결과 출력과 자신이 계사한 것을 비교해 보자.

## 구현 실습



위와 같이 구구단을 출력하는 gugu1.jsp를 구현해 보자.