百見不如一打

Servlet&JSP

Ecipse
&
Oracle

- 예제 소스 및 퀴즈 해답 등의 자료 다운로드
  http://www.roadbook.co.kr/126

- 질의 응답 사이트
  http://roadbook.zerois.net

- 특별부록(온라인 쇼핑몰) 다운로드
  http://www.roadbook.co.kr/126

백견불여일타 JSP&Servlet : Ecipse&Oracle

지은이 성윤정
1판 1쇄 발행일 2014년 7월 28일

펴낸이 임성춘
펴낸곳 로드북
편집 장미경
디자인 이호용(표지), 박진희(본문)

주소 서울시 관악구 신림로29길 8 101-901호
출판 등록 제 2011-21호(2011년 3월 22일)
전화 02)874-7883
팩스 02)6280-6901
정가 27,000원
ISBN 978-89-97924-10-3 93000

이메일 chief@roadbook.co.kr
블로그 www.roadbook.co.kr

지은이의 글

"백견불여일타!"

처음 프로그래밍을 시작할 때 필자가 늘 학생들에게 강조하는 "주저함 없이 도전하라"라는 필자의 일관된 철학이 들어가 있는 것 같아 참 마음에 드는 말입니다. 누구나 고수가 되는 길의 첫 발걸음의 설렘과 도전은 깊은 '사유'가 아니라 무심코 따라한 간단한 코드에서 그리고 컴파일되어 나오는 시시한(?) 결과화면에서 시작됩니다.

이 책은 웹 서비스 개발자가 되고자 하는 입문자를 대상으로 하고 있습니다. 필자가 '실습'에 포커스를 두고 이 책을 설계한 이유입니다. 입문자는 손이 닳도록 코딩을 반복해봐야 합니다. 과정에 익숙해지면 그 원리가 보이고 원리가 보이면 전체적인 그림이 보이기 때문입니다.

이 책은 "원리와 개념"을 말로써 알려주지는 않습니다. 차례만 보아도 실습하고 결과를 바로 확인할 수 있는 [직접해봅시다] 코너만 200개가 넘고 각 장 말미에 나오는 [도전해보세요] 과제만 보아도 이 책이 얼마나 "실습"을 강조하고 있는지 금방 알아챌 수 있을 것입니다.

필자는 사실 1980년대 학번으로 IT 기술의 초창기에 입문해서 지금까지 현업과 강의 현장을 누벼왔습니다. 대학 시절에ㅇ는 파스칼, 포트란, 코볼을 배웠고 회사에서 이들을 활용한 개발을 했습니다. 시간이 지나면서 웹 서비스가 등장하자 콘솔 화면에서의 개발 영역은 좁아지고 이클립스와 같은 복잡해보이지만 상당히 편한 개발 툴이 등장하게 되었습니다. 더불어 학습 방법도 예전과는 많이 달라야 한다는 생각을 하게 되었습니다. 복잡한 웹 서비스 개발을 배워야 하는데, 콘솔 화면에서 문자열 찍고 있는 것을 알려주거나 개발 방식은 빠르게 변해가는데, 과거의 패턴을 그대로 학생들에게 가르치는 것은 개발 현장에 나갈 그들에게 이중고를 주는 것과 다름없다고 생각했습니다. 그래서 이 책을 집필할 때 크게 두 가지에 주안점을 두었습니다. 현업에서 활용하는 개발 환경과 개발 방식입니다.

현업에서 활용하는 개발 방식은 이클립스와 오라클 등이며 개발 방식은 모델과 뷰를 분리해서 개발하는 방식입니다. 처음부터 끝까지 책을 따라 학습을 마치면 이러한 개발 환경에 상당히 익숙해질 수 있을 것입니다.

이 책은 여러분들이 스스로 확장해가면서 일부로 에러도 내보고 또 덧붙여보면서 실습을 해보면 좋겠습니다. 눈으로 보아서는 절대 아무것도 얻을 수 없는 책입니다. 그리고 단순히 따라만 해서도 좋은 성과를 얻을 수 없습니다. 고쳐보고 확장해보고 다른 기능을 붙여보면서 응용력을 키워보시기 바랍니다. 그리고 자바 개발자라고 해서 HTML이나 CSS, 자바스크립트 등을 대강만 알면 안 됩니다. 이 책에서는 거의 모든 예제에 클라이언트에 사용되는 대표적인 기술들이 들어가 있습니다. 자세한 설명은 없지만 역시 레이아웃이나 자바스크립트 기능도 변경해보면서 클라이언트 기술에도 익숙해지는 계기가 되었으면 좋겠습니다.

그리고 [도전해보세요] 코너는 정말 여러분들이 정답을 보지 않고 몇일이 걸리더라도 꼭 직접 풀어보라고 권하고 싶습니다. 정답을 찾아는 과정에서 여러분이 배울 수 있는 게 너무 많기 때문입니다.

자바로 밥 먹고 살려면 이 책만으로는 절대 부족합니다. 이 책은 정말 JSP&서블릿을 배우는 데 진입 장벽을 낮춰주는 책이지 이 책을 학습하고 바로 현업 개발자로 나설 수 있는 책이 절대 아닙니다. 이 책으로 드넓은 자바 웹 프로그래밍의 세계에서 진정한 고수가 되는 날을 만날 수 있기를 충심으로 기원합니다.

마지막으로 집필을 마치고 마지막 정리하는 과정에서 갑자기 아프게 된 바람에 고생을 많이 하게 만든 임성춘 편집장에게 진심으로 감사를 드립니다.

2014년 7월

성윤정

편집자이자 베타테스터의 글

1990년대에 IT 편집자로 입문하고 제일 처음 만들었던 책이 Servlet 책이었습니다. 번역서였고 국내에서 첫 번째 책이었던 것으로 기억합니다. 바로 1년 정도 뒤에 JSP 책을 만들고 EJB 책을 만들었습니다. 국내서였죠. 그로부터 십수 년이 흘러 다시 한번 JSP&서블릿 책을 기획해서 출간을 앞두고 있습니다. 어쩌다가 10년도 훌쩍 넘은 후에 다시 같은 주제의 책을 만들고 있는지 죽을 때까지 IT 편집자로 살아야 할 운명 같습니다.

처음 서블릿 책을 접했을 때는 아키텍처 자체마저 생소했고 영어로 된 용어를 어떻게 번역해야 할지 정말 어려웠습니다. 그리고 국내에서도 이제 막 도입했던 시기였기 때문에 게시판 예제 하나 없었고 단순히 Servlet 클래스들을 분석하고 어떤 역할을 하는지에 관한 책이었습니다. 물론 MVC니 하는 패턴 이야기는 전혀 없었죠. 하지만, 지금은 이러한 패턴을 기본으로 지키고 있고 당연시 하고 있습니다. 시간이 지날수록 서블릿은 JSP를 낳고 또 서블릿과 JSP는 스트럿츠니 스프링이니 하는 수많은 프레임워크를 낳게 되는데, 이걸 보면 스스로 진화하는 기술을 보고 있는 듯한 이상한 느낌마저 듭니다.

이 책을 집필하신 성윤정 강사님은 오래 전부터 IT 책을 다수 집필하셨습니다. 처음 로드북에 이 책을 제안하였을 때는 '개념과 원리'를 중시하는 로드북의 출간 철학에는 맞지 않아 많은 고민을 하였습니다. 하지만, 곰곰이 생각해보면 너무나 큰 장점이 있었습니다. 초보자들이 가장 쉽게 접근할 수 있도록 직접 해보면서 배울 수 있다는 점이었습니다. 그래서 "백견불여일타"라는 제목을 붙이게 되었습니다.

그리고 프로그래머가 아닌 편집자가 일일이 테스트를 해보았습니다. 처음엔 한글 워드에 있는 소스를 복사해서 붙여다가 테스트를 했습니다(시간상 일일이 타이핑은 못했습니다). 그리고 책이 디자인된 뒤에도 마지막 교정시에 PDF 상태에서도 역시 소스를 복사해서 일일이 테스트를 하였습니다. 완전 소스에는 반영이 되어 있어도 책에는 반영이 안 되는 경우도 있고 편집 과정에서 실수가 있을 수 있어 독자를 괴롭힐 수가 있기 때문입니다.

처음엔 이클립스도 엉뚱한 버전을 설치해서 당황한 적도 있었습니다. 서블릿 컨테이너에 대한 개념이 잡히질 않아 수많은 문서를 뒤적이며 겨우 이해를 할 수 있었는데, 이 부분은 저자와 상의하여 초고에서 약간의 수정을 함으로써 깔끔하게 해결되었습니다. 다른 JSP 페이지로 포워딩을 해야 하는데, JSP 파일 이름을 잘못 입력해서 계속해서 에러가 나 황당한 적도 있었고 임포트가 안 되었거나 오라클에서 커밋을 해주지 않아 한창을 헤맸던 적도 있었습니다. 대부분 아주 사소한 오타 때문에 에러가 많이 난다는 사실에 놀라지 않을 수 없었습니다. 점차 에러에 익숙해지기 시작했고 근원지를 찾아 고쳐내고자 하는 도전 의식까지 생기더군요. 이런 게 프로그래밍의 재미가 아닌가 싶었습니다.

마지막에 모델2로 해보는 게시판 예제까지 테스트를 하고 나서 이 글을 쓰고 있습니다. 물론 지금도 어떤 기능상의 버그가 있어 저자에게 수정 의뢰를 해놓았는데, 깔끔하게 처리하고 책을 내놓도록 기다리고 있습니다.

집필 도중에 건강이 많이 안좋았지만 최선을 다해 편집자의 엉뚱한 요청에도 잘 대해주신 성윤정 저자에게 진심으로 감사를 드립니다.

2014년 7월

편집자 & 베타테스터 임성춘

이 책을 제대로 보는 방법

1. 이론적인 내용은 최대한 앞부분에 핵심만 간추려 설명

JSP&서블릿의 역사나 단순한 API 등의 나열은 과감히 생략하였습니다. 이론적인 내용을 최대한 간결하게 정리하여 실습을 바로 해볼 수 있게 하였습니다.

2. 예제소스를 보는 방법

이클립스에서 자동으로 생성되는 소스 외에 별도로 입력해야 할 소스는 별색으로 처리하였습니다. 일부 자동으로 입력된 소스가 변경된 부분이 있는 부분이 있지만, 실행상에는 문제가 없습니다.

3. [직접해보세요] 코너는 반드시 직접 해보아야

이 코너에서는 실습뿐만 아니라 개발 환경 설정, DB 설정 등 여러 환경 설정까지 함께 들어가 있기 때문에 처음부터 끝까지 직접 해보아야 합니다.

[직접해보세요] get과 post 전송 방식의 폼과 서블릿 테스트

1. 웹 프로젝트(web-study-02)에서 마우스 오른쪽 버튼을 클릭하여 나타난 바로가기 메뉴에서 [New → JSP File]을 선택합니다. [New JSP File] 창이 나타나면 파일 이름을 입력합니다. 이 책에서는 파일 이름을 '04_method'로 합니다. 파일 이름만 입력하면 확장자는 자동으로 .jsp로 붙습니다. get 방식 전송 버튼과 post 방식 전송 버튼을 갖는 입력 폼을 작성합니다.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>서블릿</title>
</head>
<body>
<form method="get" action="MethodServlet">
  <input type="submit" value="get 방식으로 호출하기" >
</form>
<br><br>
<form method="post" action="MethodServlet">
  <input type="submit" value="post 방식으로 호출하기" >
</form>
</body>
</html>
```

4. [퀴즈로 정리합시다]로 이론적인 내용을 정리

각 장에서 배운 내용을 객관식과 서술 형식을 섞어 테스트하는 과정입니다. 무슨 지식을 습득하든지 연습문제를 풀어봐야 그 지식에 더 익숙해질 수 있습니다.

퀴즈로 정리합시다

문제의 답은 로드북 홈페이지(http://roadbook.co.kr/126)에서 확인할 수 있습니다.

1. JSP가 무엇의 약어인지 표시하고 정의하시오.

2. 사용자가 JSP를 요청하면 이를 서블릿 컨테이너에서 어떻게 처리하는지 순서대로 각 단계를 상세히 기술하시오.

5. [도전해보세요] 코너는 직접 풀어봐야

이 코너는 반드시 직접 풀어보기를 바랍니다. 답안 소스를 제공하고 있지만, 여러분이 같은 실행결과가 나올 때까지 끝까지 도전하여 결과를 내고 답안과 비교를 하면서 어떤 부분이 다른지를 꼭 파악해보시기 바랍니다.

도전해보세요

문제의 답은 로드북 홈페이지(http://roadbook.co.kr/126)에서 확인할 수 있습니다.

"회원 가입 페이지 작성하기"

목표 <form> 태그를 사용하여 입력 폼의 대표할 만한 사용 예인 회원 가입 페이지를 작성하고 여기서 입력 받은 값을 서블릿에 받아 처리합니다.

난이도 상

조건 표시된 내용은 회원 가입을 위해 반드시 입력해야 하는 항목입니다. 자바스크립트로 유효성을 체크하세요.

6. 궁금하면 여기 물어보세요

로드북 전용 Q&A 게시판입니다. 질문 앞머리에 [JSP]만 붙여주세요. 저자분께 바로 질문 의뢰가 가고 빠른 답변을 받아볼 수 있습니다.

로드북 ReadBook

질문과 답변 | FAQ | 자료실

LOG IN

로그인 유지

회원가입 ID/PW 찾기

질문과 답변

글 수 219

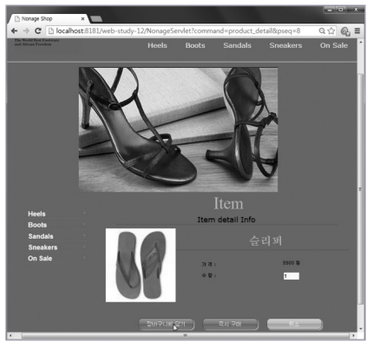
번호 제목

공지 (예제소스 업로드) 처음부터 다시 배우는 HTML5&CSS3 트 재작까지

공지 (예제소스 업로드) 처음 시작하는 CSS&워드프레스

7. 특별부록 제공

이 책은 "온라인 쇼핑몰 개발"을 특별부록으로 예제소스와 함께 해설을 PDF 형태로 제공합니다. 우선은 한번 따라해보면서 감을 익혀보고 자신만의 쇼핑몰을 만들어보시기 바랍니다.



Nonage Shop

localhost:8181/web-study-12/NonageServlet?command=product_detail&pseq=8

Heels Boots Sandals Sneakers On Sale

Item

Item detail Info

슬리퍼

Heels
Boots
Sandals
Sneakers
On Sale

목차

4장.내장 객체와 액션 태그

5장.쿠키와 세션

6장.자바 빈과 액션 태그

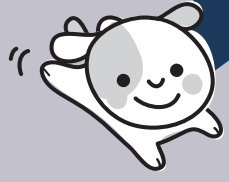
7장.표현 언어와 JSTL

10장.파일 업로드

11장.MVC 패턴(모델2)을 사용한 게시판

1장

서블릿과 JSP 개요



이 장을 시작하기 전에

만약 여러분이

- 자바 웹 애플리케이션 개발 환경(톰캣+JDK+이클립스)을 구축할 수 있거나,
- JSP와 서블릿으로 HelloWorld 수준의 웹 애플리케이션을 만들어 실행할 수 있다면,

다음 장으로 넘어가도 좋습니다.

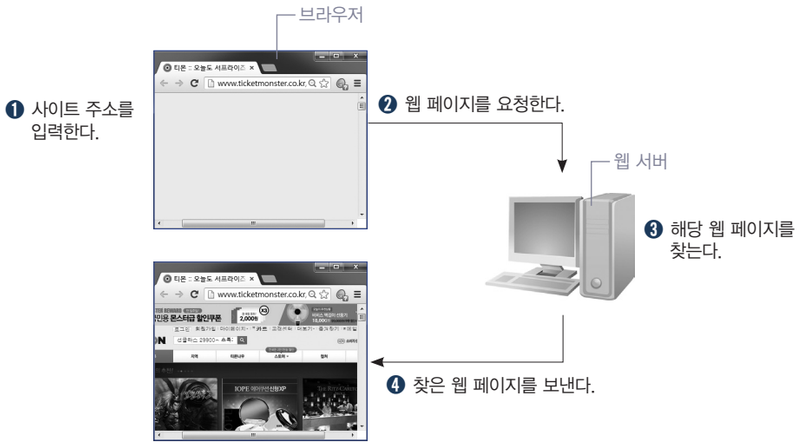
- 자바 문법에 대해 아직 아리송하고 잘 모르고 있다면,

우선은 자바 기본서를 한번 정도는 정독하며 학습한 후에 배우기를 권장합니다. 이 책은 가능하면 자바의 기본 문법을 알고 있다는 전제 하에 설명을 하고 있기 때문입니다. 물론 좀 어려운 기초 문법은 간단히 설명하고 있으니 겁먹지 마세요.

자, 이제 시작입니다. 코드를 직접 입력하고 실행해보고 변경해보면서 그 의미를 파악하며 이 책의 끝까지 완주할 수 있기를 바랍니다. 화이팅!

웹 프로그래밍이란?

인터넷은 컴퓨터들이 연결된 거대한 네트워크 구조입니다. 예를 들어 인터넷 쇼핑을 하기 위해 브라우저에 사이트 주소를 입력하면 이는 해당 사이트에 있는 웹 페이지를 내가 사용하는 브라우저에 보내달라고 요청하는 것을 의미합니다. 이렇게 요청을 받아 웹 페이지를 찾아서 보내주는 일을 하는 컴퓨터나 프로그램을 웹 서버라고 하고 요청된 페이지를 받아보는 브라우저나 컴퓨터를 클라이언트라고 합니다. 아래 그림은 인터넷 쇼핑 사이트의 동작 원리를 나타내고 있습니다.



브라우저를 통해서 각종 정보를 제공해주는 웹 페이지는 HTML을 이용하여 웹 프로그래밍을 한 것입니다. 하지만 HTML만으로는 시시각각 변경되는 새로운 정보를 제공해주지 못합니다. 왜냐하면 HTML은 같은 내용만 표시해주는 정적인 페이지이기 때문입니다.

우리가 사용하는 인터넷은 바로바로 새로운 내용을 제공해주어야 하기 때문에 HTML만 가지고 웹 프로그래밍을 하는 데 문제가 있습니다. 그래서 등장하게 된 것이 동적인 페이지입니다.

동적인 페이지에서 새로운 정보를 제공해주기 위해서는 방대한 정보를 관리할 데이터베이스가 필요합니다. 예를 들어 게시판에 게재되는 글은 데이터베이스에 저장되었다가 보여주는 것입니다. 이렇듯 다양한 정보를 데이터베이스에서 얻거나 저장하기 위해서 등장한 언어가 PHP, ASP, 서블릿/JSP입니다.

인터넷을 통해 카멜레온처럼 변화무쌍한 정보를 얻거나 쇼핑을 할 수 있는 이유는 웹 애플리케이션 언어(PHP, ASP, 서블릿/JSP)로 개발한 웹 애플리케이션, 예를 들면 네이버Naver나 다음Daum과 같은 포털 사이트, GMarket이나 티켓 몬스터와 같은 온라인 쇼핑몰이 웹 서버에 구축되어 있기 때문입니다.

참고

웹 애플리케이션이란?

웹 애플리케이션은 웹(인터넷)을 기반으로 실행되는 애플리케이션(프로그램)을 말합니다. 즉, 웹 브라우저로 접근하여 사용되는 애플리케이션을 말합니다. 이 책에서 학습하게 되는 웹 프로그래밍(Web Programming)이 바로 웹 애플리케이션을 제작하는 과정을 뜻합니다. 기억하세요. 이 책에서 의미하는 웹 프로그래밍이란 '웹 애플리케이션을 제작'하는 과정입니다.

이 책을 읽는 독자의 학습목표는 서블릿/JSP를 사용하여 웹 애플리케이션을 개발하는 것입니다. 여러분이 쇼핑몰을 웹 애플리케이션으로 구축하는 것을 목표로 한다면 사용자가 원하는 상품을 검색한 후 구입을 하는 과정을 모두 서블릿/JSP를 사용하여 제작해야 합니다.

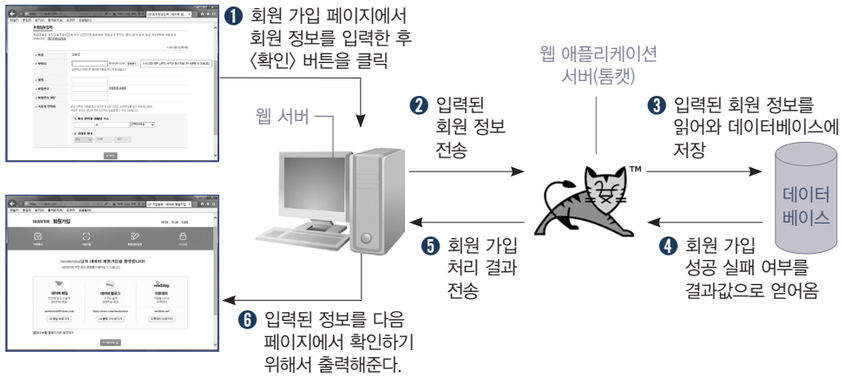
서버는 일반적으로 사용자(클라이언트)의 요청이 들어오면 이에 대한 처리를 한 결과 페이지를 전송하는 웹 서버Web Server와 실질적으로 요청한 페이지의 로직이나 데이터베이스와의 연동을 처리할 수 있는 비즈니스 로직이 구현되어야 하는 웹 애플리케이션 서버(Web Application Server : WAS)로 이루어져 있습니다.

대표적인 WAS로는 BEA사의 웹로직WebLogic, IBM의 웹스피어WebSphere, SUN사의 iPlanet, Oracle 9iAS, 티맥스의 제우스Jeus, 우리 책에서 사용하는 톰캣Tomcat 등이 있습니다.

여기에서 톰캣은 웹 서버 기능이 내장되어 있어 별도로 웹 서버를 설치하지 않고 WAS 역할까지 합니다.

WAS라는 서버 프로그램이 웹 애플리케이션을 어떻게 동작시키는지 그 원리를 이해하기 위해 '회원 가입을 위한 프로그램(애플리케이션)'이 어떤 절차를 거쳐서 실행되는지를 살펴보겠습니다. 회원 가입을 하기 위해서는 회원 가입 페이지에서 이름,

아이디, 별명, 비밀번호 등을 입력한 후 〈확인〉 버튼을 클릭할 것입니다. 그러면 브라우저는 입력된 회원 정보를 서버로 전송하고 서버에서는 이 정보를 WAS를 통해 데이터베이스에 저장합니다.



웹 서버는 요청이 있을 경우 이를 받아들여 요청한 페이지를 응답할 책임을 지고 있습니다. 웹 서버가 요청을 받아 응답할 때까지의 웹 애플리케이션의 동작 순서는 다음과 같습니다.

❶ 사용자가 브라우저 주소 입력란에 특정 사이트의 주소를 입력하게 되면 브라우저가 해당 웹 서버에 웹 페이지를 요청하는 것이 됩니다. 단 요청한 페이지가 단순한 정적인 페이지라면 웹 서버에서 바로 클라이언트에게 해당 페이지를 전송합니다.

❷ 해당 웹 서버는 입력된 회원 정보를 웹 애플리케이션 서버(WAS : Web Application Server)로 전송합니다.

❸ 요청한 페이지는 웹 서버에서 바로 제공되지 못하고 웹 애플리케이션 서버에서 다양한 로직이나 데이터베이스와의 연동을 통해서 완성되어야만 제공됩니다. 웹 애플리케이션 서버는 웹 서버가 클라이언트에게 제공할 페이지를 완성하기 위해 이에 필요한 로직이나 데이터베이스와의 연동과 데이터 처리를 담당합니다.

❹ 로직이나 데이터베이스 작업 처리 결과를 웹 서버에게 보냅니다.

❺ 웹 서버는 이 결과를 다시 클라이언트 측 브라우저에 응답하게 됩니다.

❻ 회원 가입이 성공적으로 이루어졌다면 가입 당시에 입력된 정보를 확인하기 위해서 출력해줍니다.

앞에서도 언급하였지만 톰캣은 웹 서버를 내장하고 있다고 했습니다. 위의 그림에서는 웹 서버와 톰캣이 분리되어 있지만 사실 두 기능을 함께 수행하고 있는 것입니다.

지금까지 웹 프로그래밍의 전체적인 그림을 살펴보았다면, 다음 절에서는 구체적으로 서블릿과 JSP가 무엇인지 아주 기초적인 내용을 살펴볼 것입니다.

웹 애플리케이션 개발 환경 구축하기 – 프로그램 설치

웹 애플리케이션 개발 환경 구축을 위한 설치 프로그램은 JDKJava Development Kit, 톰캣Apache Tomcat, 이클립스Eclipse입니다.

이번 절에서 서블릿/JSP 웹 프로그래밍을 학습하기 위해서 웹 애플리케이션 개발 환경을 구축하는 방법을 학습해보도록 하겠습니다. 다음은 이들 설치 프로그램들이 각각 어떤 역할을 하는지 정리한 표입니다.

설치 프로그램	설명
JDK (Java Development Kit)	제일 먼저 무료로 제공해주는 자바 개발 도구인 JDK(Java Development Kit)를 다운받아 설치해야 합니다. 자바는 플랫폼에 독립적이므로 어떠한 플랫폼에서도 설치할 수 있습니다. 우리는 JDK를 설치하기 위한 개발 플랫폼으로 Windows를 선택했습니다.
톰캣 (Apache Tomcat)	톰캣(Tomcat)은 아파치와 썬 마이크로시스템즈에서 공동 프로젝트로 만든 웹 애플리케이션 서버입니다. 웹 애플리케이션 서버가 무엇인지 모르면 앞부분을 다시 한번 살펴보세요.
이클립스(Eclipse)	이클립스는 애플리케이션 개발을 위한 코딩과 컴파일을 함께 할 수 있는 종합 개발 툴(IDE)입니다. 최근 개발자들이 가장 많이 사용하는 툴입니다.

설치 프로그램들의 버전은 자주 최신 버전으로 변경됩니다. 설치 방법은 [직접해보세요] 코너에서 자세히 알려드리겠습니다. 각각의 버전은 설치 시점에 따라 약간 다를 수 있으며 해당 사이트에 가면 이 책에 언급한 버전과 다른 버전이 제공될 수 있습니다. 버전에 따라 사용 방법이 크게 다르지 않기에 자유롭게 버전을 선택해도 상관없습니다. 다음은 이 책의 집필 시점의 버전입니다.

- JDK(Java Development Kit)
  http://www.oracle.com/technetwork/java/javase/downloads/index.html
  'Windows x86(jdk-7u17-windows-i586.exe)'라는 윈도우 32비트용 버전을 다운로드하여 설치합니다. 윈도우가 64비트라면 64비트용을 다운로드해도 상관 없습니다.
- 톰캣(Apache Tomcat)
  http://tomcat.apache.org
  톰캣(Tomcat) 7.0을 다운로드하여 압축을 풀면 됩니다.
- 이클립스 개발 도구
  http://www.eclipse.org/downloads
  이클립스를 다운로드하여 압축을 풀면 됩니다. 현재 웹 애플리케이션 개발은 대부분 이클립스를 사용합니다. 회사에서 업무를 수행하기 위해서 워드프로세스를 공부하듯이 웹 애플리케이션 개발을 위해서는 이클립스를 반드시 익혀야 합니다. 학습할 때는 메모장과 같은 윈도우즈의 기본 에디터를 이용해서 프로그램을 작성한 후 콘솔 창에서 컴파일하고 브라우저에 직접 입력하여 요청하여 실행할 수도 있습니다. 하지만, 실제 실무에서 이와 같은 방식으로는 웹 애플리케이션 개발이 거의 불가능합니다. 이클립스를 사용하면 소스 작성은 물론 컴파일, 실행을 원스톱으로 한꺼번에 실행할 수 있어 빠른 개발이 가능합니다. 자주 사용하면서 익혀두기 바랍니다.

JDK 설치하기

이클립스가 설치되면서 자바의 개발 환경을 자동으로 설정하기 때문에 JDK를 먼저 설치한 후 이클립스를 설치해야 합니다. 또한 JDK는 서블릿과 JSP를 포함한 자바로 작성한 프로그램을 컴파일하기 위한 툴로서 이를 WAS(WAS란 앞에서 이미 언급한 바 있는 Web Application Server의 약어로서 톰캣과 같은 웹 애플리케이션 서버를 말합니다)를 설치하기 전에 JDK를 먼저 설치해야 합니다.

참고

서블릿과 JSP 학습을 위해 JDK를 설치하는 이유

서블릿과 JSP를 구동시키는 것은 WAS입니다. 하지만, WAS에서 서블릿과 JSP를 구동하기 위해서는 자바 컴파일러가 필요합니다. 왜냐하면 이미 설명한 대로 서블릿과 JSP는 자바를 기반으로 한 기술이기 때문입니다.

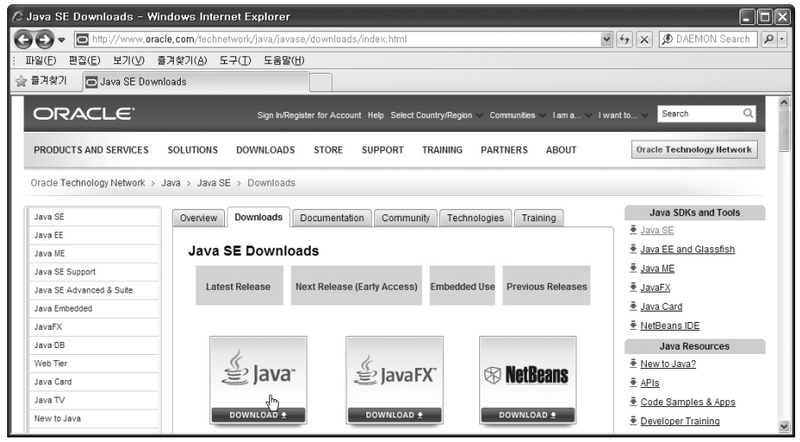
우선 JDK를 다운로드하여 설치하도록 합시다. JDK는 오라클사의 홈페이지에서 무료로 다운로드한 후 설치할 수 있습니다. Java SE의 최신 버전(집필 시점에서는 Java SE 7u17)을 다운로드하기 바랍니다.

JDK는 자주 버전이 갱신되므로 설치 시점에 따라 버전이 약간 다를 수 있다는 점에 주의하세요. 버전이 갱신되어 찾기가 어려우면 Java SE(Standard Edition)를 찾으면 됩니다. 다운로드하는 시점에 따라 최신 버전이 다를 수 있으므로 '7u17 버전'이 아니더라도 가장 최신의 버전을 다운로드하면 됩니다.

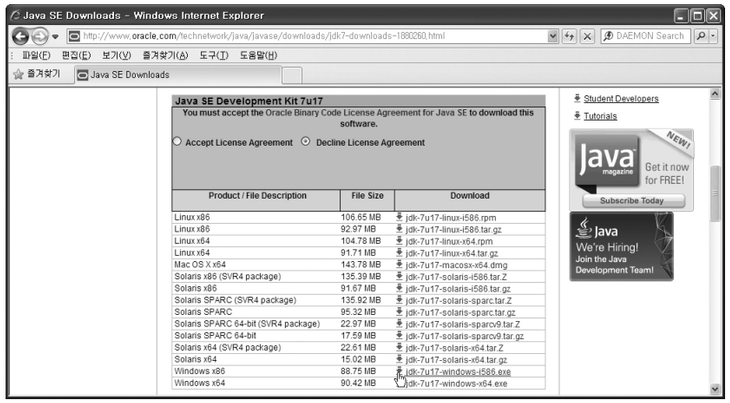
[직접해보세요] JDK 다운로드하여 설치하기

1. 브라우저를 열어 주소란에 다음 URL을 입력하여 오라클 사이트의 JDK 다운로드 페이지에 접속합니다.

```
http://www.oracle.com/technetwork/java/javase/downloads/index.html
```

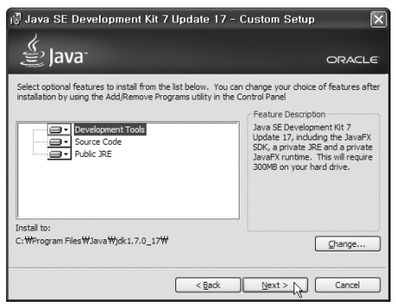


2. 다운로드에 앞서 'Accept License Agreement'에 체크하여 동의를 거칩니다. 플랫폼(운영체제)별로 JVM 설계가 다르므로 JDK는 다음과 같이 여러 종류로 나뉘어 있습니다. 사용자의 PC에 설치된 플랫폼에 맞는 JDK를 선택합니다. 윈도우즈 32비트인 경우 'Windows x86(jdk-7u17-windows-i586.exe)'을 다운로드 합니다.



3. 다운로드 페이지를 통해 받은 파일을 더블클릭하면 설치가 시작됩니다. [Next] 버튼을 클릭하여 설치를 진행합니다.

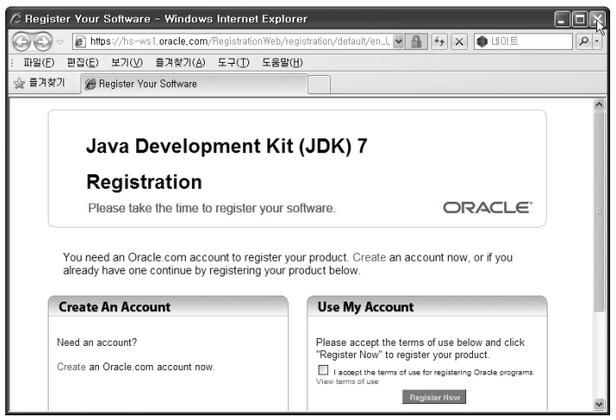
4. JDK의 설치 위치를 변경하겠느냐는 화면이 나타납니다. 본서에서는 JDK의 설치 위치를 변경하지 않습니다. [Next] 버튼을 클릭한 후 계속 설치합니다.



5. 이번에는 JRE의 설치 위치를 변경하겠느냐는 화면이 나타납니다. JRE의 설치 위치 역시 변경하지 않기로 합니다. [Next] 버튼을 클릭하여 설치를 진행합니다.

6. 설치 진행 화면이 나타나게 된 후에 JDK가 설치가 완료되면 다음 그림과 같이 설치 종료 화면이 나타나게 됩니다. [Close] 버튼을 눌러 설치를 종료합니다.

7. JDK 설치가 완료되면 등록 화면이 뜹니다. 오라클 계정 없이 설치하고 나면 바로 사용할 수 있으므로 익스플로러 창을 닫습니다.

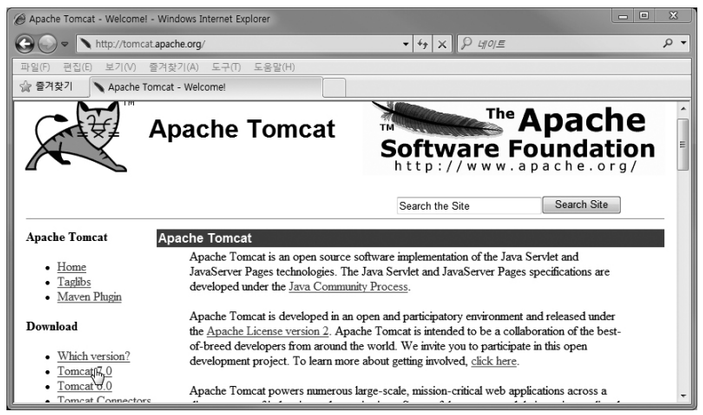


톰캣 설치하기

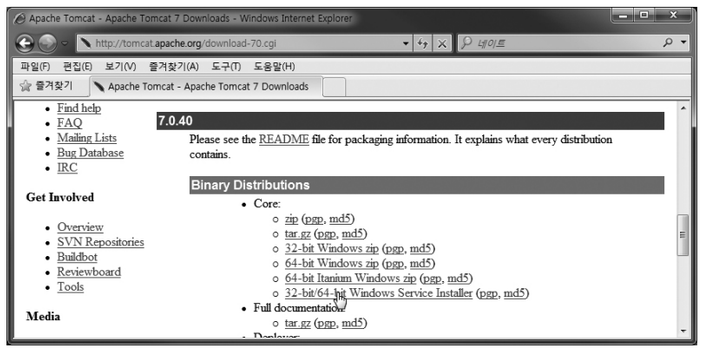
JDK 설치가 완료되었으면 이제 실질적으로 서블릿과 JSP를 구동시키는 WAS를 설치해보도록 하겠습니다. WAS로는 BEA사의 웹로직, IBM의 웹스피어 SUN사의 iPlanet, Oracle 9iAS, 티맥스의 제우스 등이 있다고 이미 언급했습니다. 이 책에서는 이렇게 다양한 WAS 중에서 오픈소스 프로젝트로 개발되어 무료로 제공되는 톰캣을 사용하겠습니다.

[직접해보세요] 톰캣 다운로드하여 설치하기

1. 브라우저를 열어 아파치 톰캣 사이트(http://tomcat.apache.org)에 접속한 후 화면 왼쪽에 있는 메뉴에서 Download → Tomcat 7.0을 선택하여 톰캣 다운로드 페이지로 이동합니다.

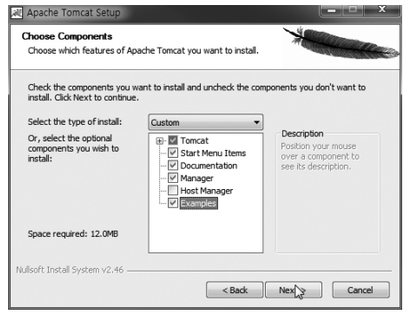


2. 화면을 스크롤하여 중앙에 있는 7.0.40 밑에 Binary Distributions / Core 영역의 32-bit/64-bit Windows Service Installer (pgp, md5)를 선택해 다운로드 합니다.

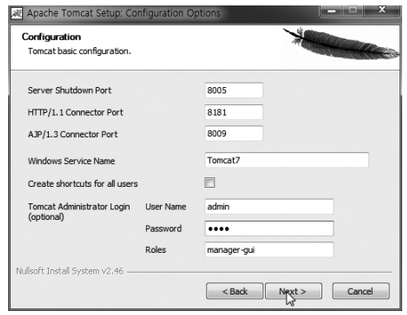


3. 받은 파일을 더블클릭하여 설치를 시작합니다. 설치 화면이 나타나면 [Next] 버튼을 클릭한 후에 나타난 라이센스 동의화면에서 라이선스에 동의하겠다는 의미로 〈I Agree〉 버튼을 클릭합니다.

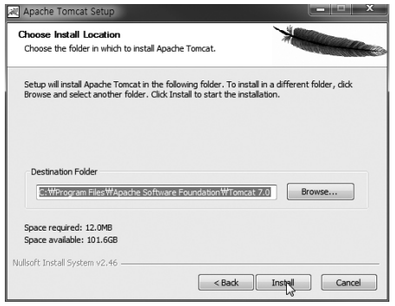
4. 환경 설정 부분에서 Examples에 추가로 체크한 후 [Next] 버튼을 클릭합니다.



5. Tomcat 내부 환경설정을 변경해야 합니다. **기존 오라클에서 사용하는 포트와 충돌이 발생할 우려가 있기 때문에 port를 8181로 수정합니다.** 오라클 사용자 이름인 User Name과 오라클에 접속하기 위해서 필요한 비밀 번호인 Password를 각각 **admin, 1234**로 지정한 후 [Next] 버튼을 클릭합니다.

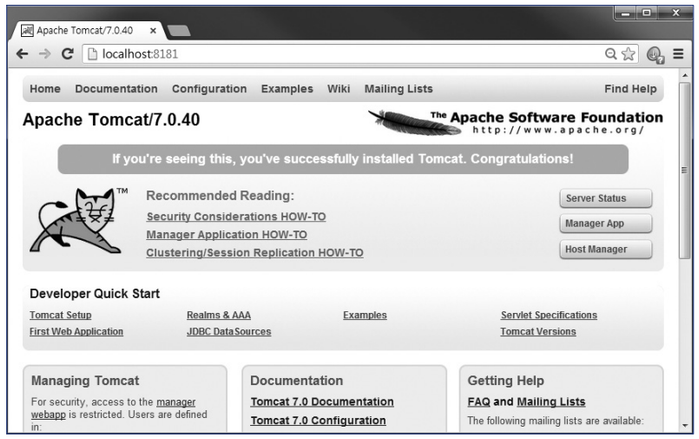


6. 톰캣 설치 경로는 기본 설정값을 사용할 것이므로 [Next] 버튼을 클릭하여 설치를 진행합니다. 설치가 완료되면 오른쪽과 같은 화면이 나타납니다. [Finish] 버튼을 클릭하여 설치를 마무리합니다.



7. 설치하자마자 톰캣이 구동됩니다. 확인은 화면 아래의 트레이 아이콘으로 확인할 수 있습니다.

8. 브라우저를 실행시켜 "http://localhost:8181"을 입력하여 다음과 같이 톰캣의 시작 페이지가 나타나면 톰캣이 제대로 설치된 것입니다.

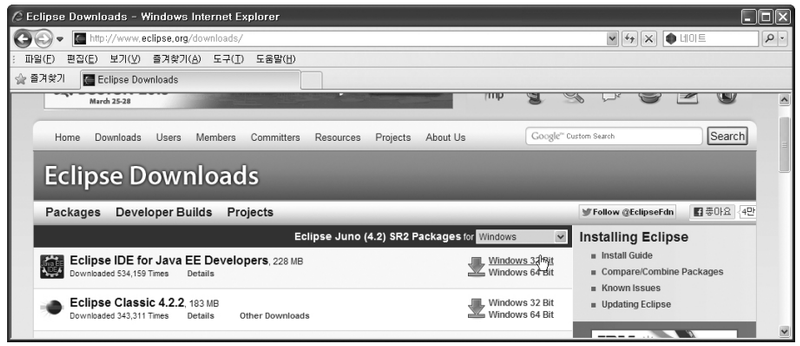


이클립스 설치하기

JDK와 톰캣이 설치되었으면 웹 프로그램을 작성하기 위한 이클립스를 설치합니다. 이클립스까지 설치가 끝나면 서블릿과 JSP를 학습하기 위한 예제를 작성하고 실행해 보도록 하겠습니다.

[직접해보세요] 이클립스 다운로드하여 설치하기

1. 웹 브라우저를 열고, http://www.eclipse.org 사이트로 이동합니다. 위쪽의 메뉴에서 [downloads] 메뉴를 클릭한 후 이클립스(Eclipse IDE for Java EE Developers)를 클릭합니다. 참고로 이클립스는 Standard가 아니라 EE Developers를 위한 IDE로 다운을 받아야 합니다.



2. 이클립스(Eclipse IDE for Java EE Developers)를 클릭하면 미러 사이트로 이동합니다. 미러 사이트란 인기있는 웹사이트에서 통신량이 폭주하는 장거리 또는 국제 회선을 경유하지 않고도 파일을 전송받을 수 있도록 2개 이상의 파일 서버를 두는 것을 말합니다. 인터넷상에는 유명한 사이트의 경우 전 세계에 몇군데의 미러 사이트가 있으므로 사용자들은 가까운 곳 또는 국내에 있는 미러 사이트를 이용하는 것이 바람직합니다.

3. 미러 사이트에서 이클립스를 다운로드 합니다.

4. 받은 압축 파일을 푸는 것으로 이클립스 설치가 끝납니다.

지금까지 JDK, 톰캣, 이클립스를 설치하였습니다. 이제부터는 본격적으로 서블릿과 JSP로 능숙하게 웹 애플리케이션을 개발할 수 있도록 서블릿과 JSP 문법을 학습할 것입니다.

이클립스로 첫 웹 애플리케이션 작성하기

이클립스를 사용하기에 앞서 먼저 워크스페이스와 프로젝트 개념을 알아봅시다.

우리는 일반적으로 파일을 관리하기 위해 폴더를 만듭니다. 예를 들어 문서 파일을 관리하기 위해서 문서라는 폴더를 만들어 놓지요. 그런데 문서에도 종류가 많기 때문에 이를 구분하기 위해 문서 폴더 내부에 워드 문서 관리 폴더를 만들고 거기에 워드 파일들을, 엑셀 문서 관리 폴더를 만들어 놓고 거기에는 엑셀 파일을 나누어 관리합니다.

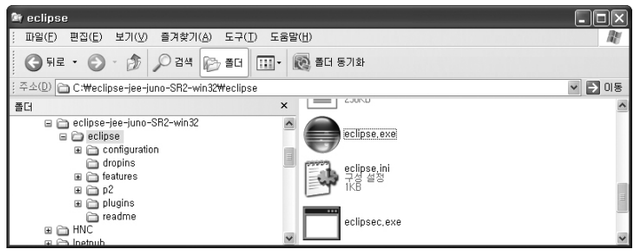
이와 마찬가지로 병원관리 프로젝트를 위한 파일을 [병원 관리] 폴더에 저장하고 학사 관리 프로젝트는 [학사 관리] 폴더에 저장합니다. 또한 [병원 관리] 폴더와 [학사 관리] 폴더와 같이 개발을 위한 프로젝트 폴더들을 워크스페이스란 작업을 위한 공간(폴더)에 저장해 둡니다. 만일 프로젝트를 할 때마다 여기저기에 저장해 두면 찾기 힘들까봐 프로젝트를 한꺼번에 작업을 위한 공간인 워크스페이스(작업 공간) 내에 모아서 관리하는 것입니다.

대규모 프로젝트나 소규모 프로젝트는 모두 프로젝트이고 프로젝트 폴더들을 관리의 편의를 위해서 워크스페이스라는 작업 공간에 저장해 두는 것입니다.

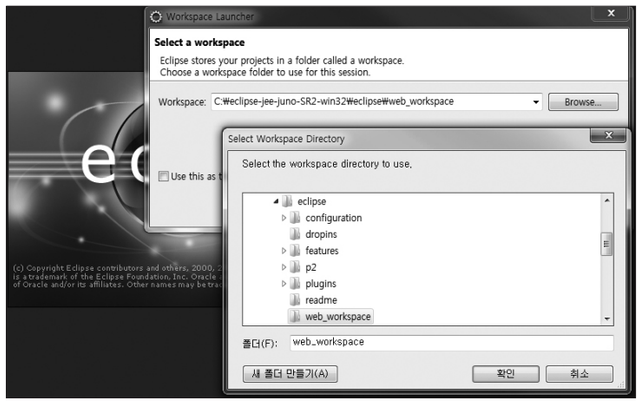
우리가 지금 이클립스로 간단한 예제를 작성하기 위해서 우선적으로 해야 할 일은 워크스페이스와 프로젝트와 JSP 파일을 만드는 일입니다. 생성된 JSP 파일에 실행 결과로 얻고 싶은 내용을 코딩한 후 저장하고, 이를 실행하여 결과를 확인합니다. 이들을 만드는 순서는 다음과 같이 워크스페이스를 생성한 후 생성된 워크스페이스를 작업공간으로 하여 프로젝트를 생성합니다. 생성된 프로젝트 내에 서블릿이나 JSP 등 필요한 파일을 생성한 후 원하는 결과가 나타날 수 있도록 코드를 입력한 후 저장하여 이를 실행합니다. 원하는 결과가 나오지 않으면 원하는 결과가 나올 때까지 코드를 수정한 후 다시 저장하여 실행하는 작업을 반복합니다. 이러한 작업은 이 책의 전반에 걸쳐 반복되니 지금은 이런 절차를 거치는구나 정도만 알면 됩니다.

[직접해보세요] 이클립스 실행하기

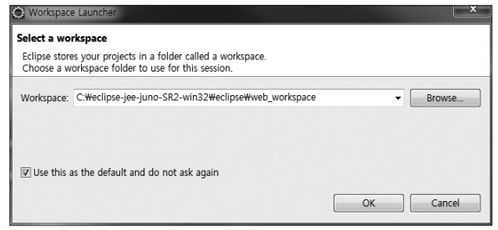
1. 제대로 설치가 되었는지 확인하기 위해서 탐색기에서 C:₩eclipse-jee-juno-SR2-win32₩eclipse 폴더 안에 있는 eclipse.exe 파일을 더블클릭하여 이클립스를 실행합니다. 바탕화면에 바로가기를 해놓으면 편리합니다.



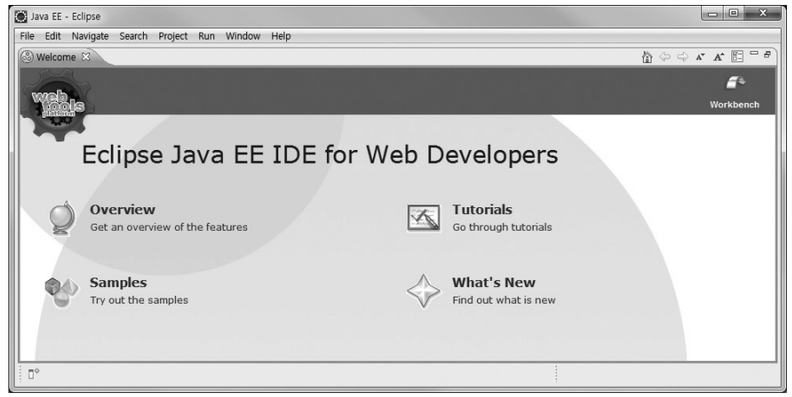
2. 이클립스가 실행되면 작업 공간(workspace)을 선택하는 창이 나타납니다. 기본으로 설정된 작업공간이 아닌 다른 작업 공간을 만들고자 할 경우에는 [Browse...] 버튼을 클릭합니다. [Select a workspace] 창이 나타나면 eclipse 폴더 하위 폴더에 워크스페이스를 생성하기 위해서 eclipse 폴더를 선택한 후 여기에서 [새 폴더 만들기] 버튼을 클릭합니다. [새 폴더] 항목의 이름을 원하는 폴더 명으로 변경합니다. 필자의 경우에는 폴더 이름을 'web_workspace'로 변경하고 변경된 폴더를 선택한 후 [확인] 버튼을 누릅니다.



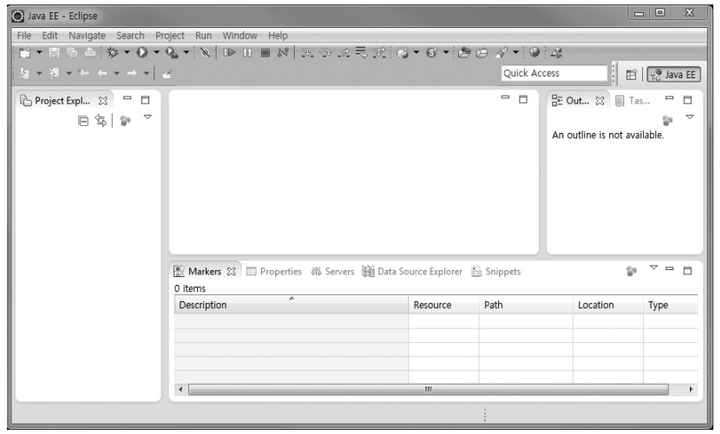
3. 그러면 [Workspace] 항목의 값이 web_workspace로 변경되어 적용되는 것을 알 수 있습니다. 화면 아래에 [Use this as file default and do not ask again] 항목을 체크하지 않으면 이클립스를 실행할 때마다 워크스페이스를 지정하라고 [Select a workspace] 창이 매번 나타나므로 귀찮을 경우 이 항목에 체크하여 이클립스 실행할 때마다 해당 창이 나타나지 않도록 합니다.



4. 이클립스가 실행되면 [Welcome] 창이 나타납니다. [Welcome] 창에 표시된 아이콘들은 자바를 학습하기 위한 튜토리얼이나 예제들과 연결하는 기능을 합니다. 화면 상단 왼쪽의 닫기 버튼을 눌러 [Welcome] 창을 닫습니다.



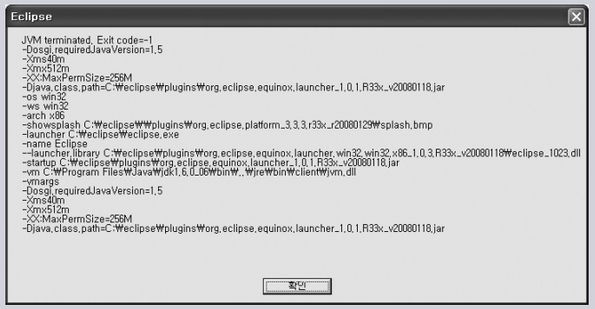
5. [Welcome] 창을 닫으면 다음과 같은 화면이 제공됩니다.



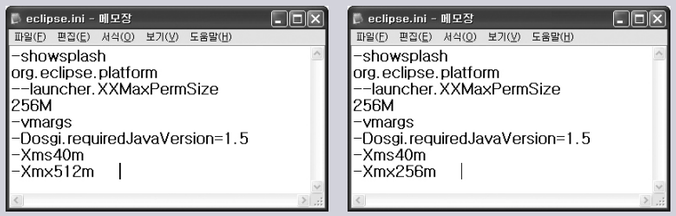
참고

이클립스가 실행이 안 된다면 이렇게 해보세요.

이러한 문제는 메모리 사이즈 때문인 경우가 대부분이므로 eclipse.ini 파일을 열어서 아래 xmx 부분이 메모리 사이즈를 나타내는데, 이 부분만 잘 조정하면 됩니다.

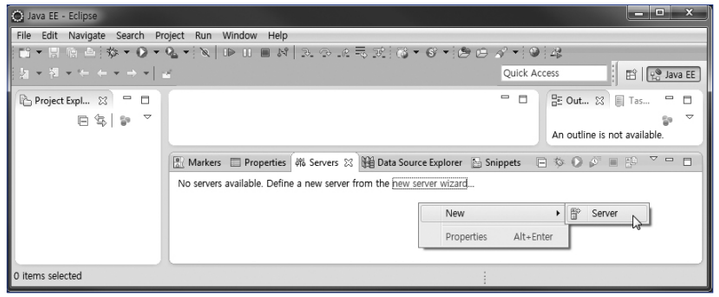


eclipse.ini 파일의 맨 아래 줄 Xmx512m을 Xmx256m으로 변경합니다.

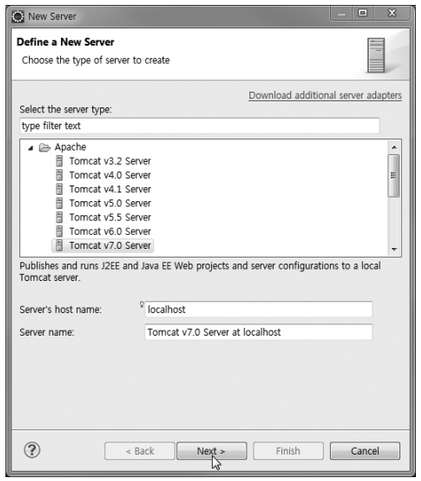


[직접해보세요] 이클립스에서 톰캣 연동하기

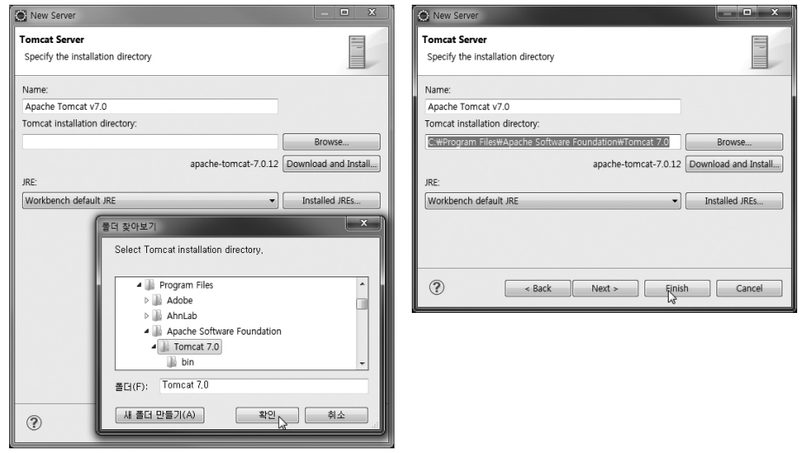
1. 이클립스의 화면 아래의 [Server] 창에서 마우스 오른쪽 버튼을 클릭하여 나타난 바로가기 메뉴에서 [New → Server]를 선택합니다.



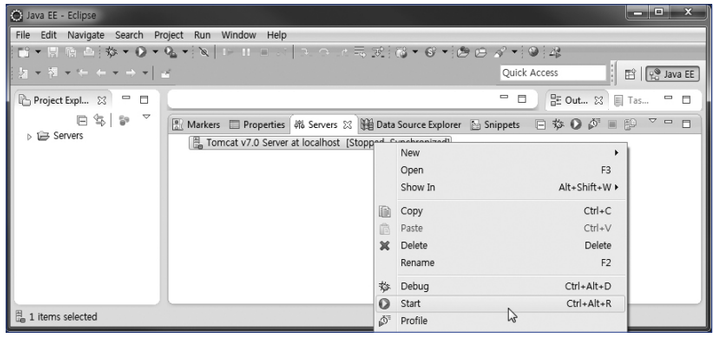
2. [New Server] 창이 뜨면, [Define a New Server]에서 [Apache → Tomcat v7. 0 Server]를 선택한 후 [Next] 버튼을 클릭합니다.



3. 그 다음 [Tomcat Server] 창에서, [Browser] 버튼을 클릭하여 Tomcat 경로(C:₩Program Files₩Apache Software Foundation₩Tomcat 7.0)를 찾아 지정합니다. [확인] 버튼을 클릭한 후 Tomcat 경로가 제대로 지정되었는지 확인한 후에 [Finish] 버튼을 클릭합니다.

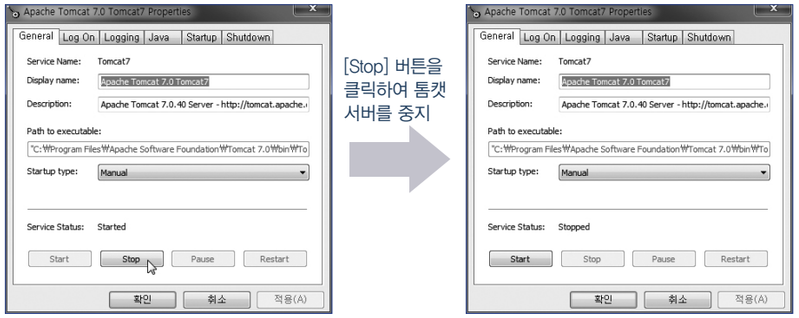


4. [Server] 창에 만들어진 Tomcat 7.0 Server에서 마우스 오른쪽 버튼을 클릭하여 나타난 바로가기 메뉴에서 [Start]를 선택하여 서버를 구동시킵니다.

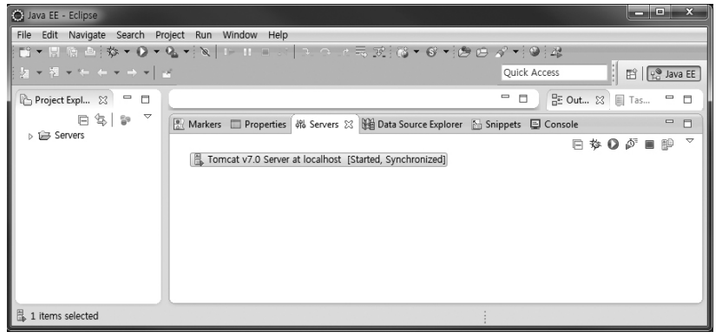


5. 서버가 구동되지 않고 다음과 같은 에러 창이 뜨면 이미 톰캣 서버가 실행중이기 때문입니다. 이클립스에서 서버를 구동하기 위해서는 이미 실행중인 톰캣 서버를 중지시켜야 합니다. 그러기 위해서 화면 하단의 작업 표시 줄의 톰캣 트레이 아이콘을 클릭하거나 시작 메뉴에서 [Monitor Tomcat]를 선택하여 [Tomcat Monitor] 창을 띄웁니다.

6. [Tomcat Monitor] 창이 나타나면 [Stop] 버튼을 클릭하여 톰캣 서버를 중지시킵니다. [Stop] 버튼이 비활성화되면 톰캣 서버가 중지된 것입니다. [확인] 버튼을 클릭합니다.



7. 톰캣 서버를 중지했으므로 다시 이클립스에서 톰캣 서버를 시작시켜 봅시다. [Server] 창에 만들어진 [Tomcat 7.0 Server]에서 마우스 오른쪽 버튼을 클릭하여 나타난 바로가기 메뉴에서 [Start]를 선택합니다. 다음과 같이 나타나면 톰캣 서버가 성공적으로 시작된 것입니다.

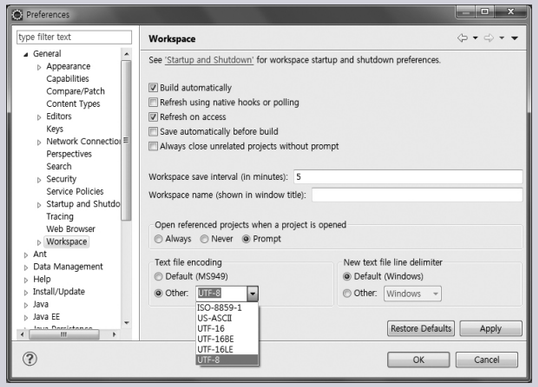


참고

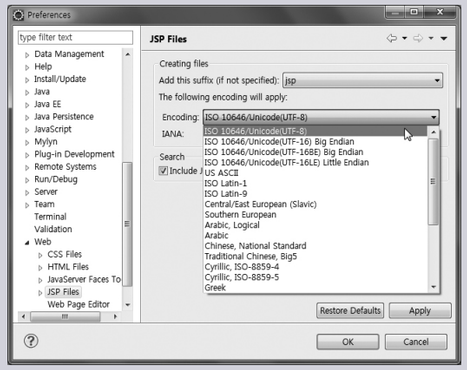
이클립스에서 인코딩 방식을 UTF-8로 변경하기

웹 프로그래밍에서 한글을 위한 작업을 위해서는 인코딩을 지정해야 합니다. 이클립스의 기본 인코딩은 시스템 인코딩에 따릅니다. 한글 윈도우에 설치된 이클립스의 경우 기본 인코딩이 MS949입니다. 이것을 다국어 지원을 위해서 일반적으로 보편화된 UTF-8로 설정해 줍시다.

[Window → Preferences] 메뉴를 선택하여 [Preferences] 창이 나타나면 화면 왼쪽에서는 [General] 하위 항목으로 [Workspace]를 선택하고 화면 오른쪽 상세 내용 중 맨 하단에서는 [Text file encoding] 항목의 선택박스에서 UTF-8로 선택하고 [Apply] 버튼을 클릭합니다.



또한 [Preferences] 창의 왼쪽에서 [Web] 하위 항목으로 [JSP Files]를 선택하고 화면 오른쪽 상세 내용 중 [Encoding] 항목의 선택박스에서 ISO 10646/Unicode(UTF-8)를 선택합니다. 역시 [Apply]를 클릭한 후 [OK] 버튼을 클릭하여 설정을 마무리합니다.



문자 인코딩이란?

인코딩은 문자셋을 컴퓨터가 이해할 수 있는 바이트와 매핑한 규칙을 말합니다. 예를 들면 ASCII Code에서 ABC 등은 문자셋이고 A는 코드 65, B는 코드 66 등 바이트 순서와 매핑한 것이 인코딩입니다. 따라서 문자셋을 어떻게 매핑하느냐에 따라 하나의 문자셋이 다양한 인코딩을 가질 수 있습니다.

컴퓨터는 영미권에서 만들어졌기 때문에 영어를 표현하는 경우는 아무런 문제가 없지만 다른 문자를 사용하는 나라에서 자국의 언어로 표현하려면 문제가 되는 경우가 있습니다. 그렇기 때문에 한글을 표현하는 인코딩을 적용하여 문서를 만들어야 합니다. 한글 인코딩은 MS949(윈도 OS에서 기본으로 사용), UTF-8 등이 있습니다.

브라우저는 내부적으로 모두 유니코드로 처리합니다. 그렇기 때문에 HTML5에서는 문자 인코딩에 UTF-8을 권장하고 있습니다. UTF-8은 전 세계적으로 모두 통용될 수 있는 표준화된 텍스트 데이터를 표현하기 위해서 만들어진 인코딩 방식입니다.

UTF-8 방식을 이용하면 다음과 같이 HTML에서 제공되는 〈meta〉 태그에 charset 속성을 추가한 후 인코딩 방식을 지정해야 합니다.

```
<meta charset="UTF-8">
```

또한 MS949 방식을 이용하면 다음과 같이 표기합니다.

```
<meta charset="EUC-KR">
```

위 방식은 HTML5에서 지정하는 인코딩 방식이고 HTML4의 인코딩 방식을 지정하는 〈meta〉 태그는 다음과 같습니다.

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

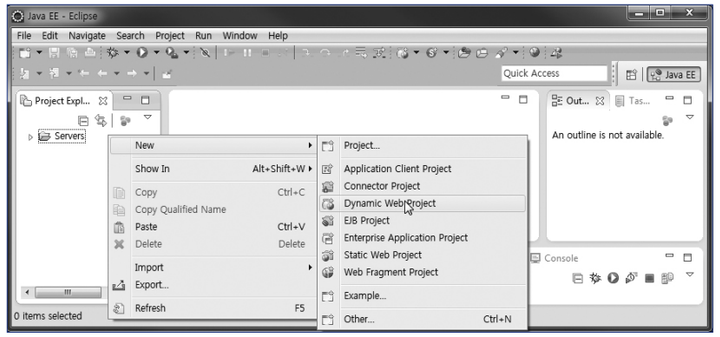
charset(Character Set, 문자셋)은 하나의 언어권에서 사용하는 언어를 표현하기 위한 모든 문자(활자)의 모임을 말하며 HTML 문서뿐만 아니라 다른 애플리케이션에도 언어권에 맞는 문자들의 집합을 언급해야 할 때 charset이란 키워드를 사용합니다. 아직은 문자셋에 익숙하지 않겠지만 앞으로 예제를 만들어가면서 확실히 이해를 할 수 있을겁니다.

톰캣 서버가 이클립스와 연동되었으므로 이제 웹 프로젝트를 생성하고 JSP로 간단한 웹 애플리케이션을 생성하여 실행해 보도록 합시다.

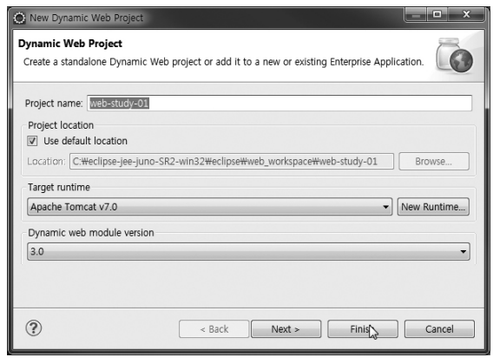
서블릿과 JSP로 게시판이나 회원 관리 애플리케이션 더 나아가서 쇼핑몰까지 만들 수 있습니다. 이런 다양한 웹 애플리케이션은 프로젝트 단위로 작성합니다. 프로젝트는 개발에 필요한 파일을 관리할 뿐 아니라 각종 라이브러리나 디버깅 정보 등을 관리합니다. 자, 웹 프로그래밍을 학습하기 위한 첫 번째 단계로 프로젝트를 생성합시다.

[직접해보세요] Dynamic Web Project 만들고 jsp 파일 만들기

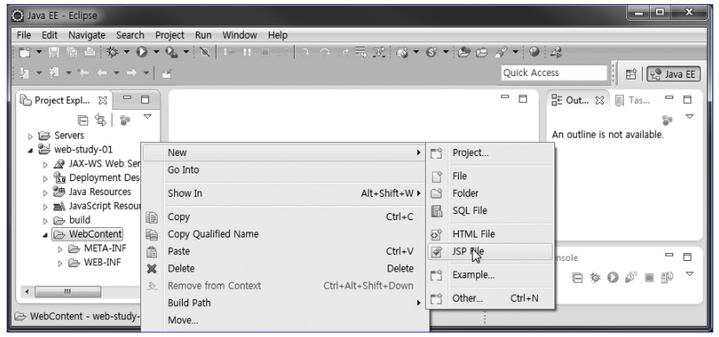
1. 웹 프로젝트를 만들기 위해서는 화면 왼쪽에 [Project Explorer]에서 마우스 오른쪽 버튼을 클릭하여 나타난 바로가기 메뉴에서 [New → Dynamic Web Project]를 선택합니다.



2. 프로젝트 이름을 입력합니다. 이 책에서는 프로젝트 이름을 "web-study-01"로 하고 [Finish] 버튼을 클릭합니다.

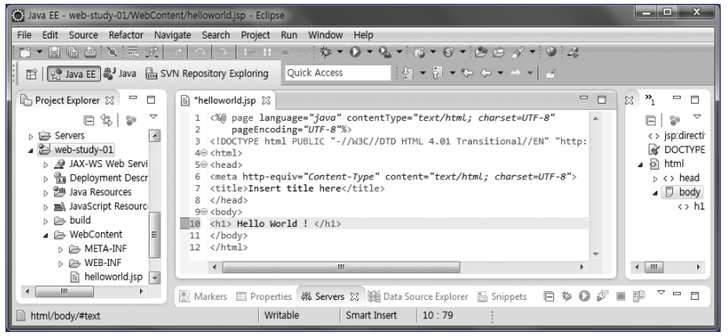


3. 이클립스의 화면 왼쪽에 [Project Explorer]에 웹 프로젝트가 추가되어 나타납니다. 새롭게 나타난 웹 프로젝트를 선택한 후 마우스 오른쪽 버튼을 클릭하여 나타난 바로가기 메뉴에서 [New → JSP File]을 선택합니다.

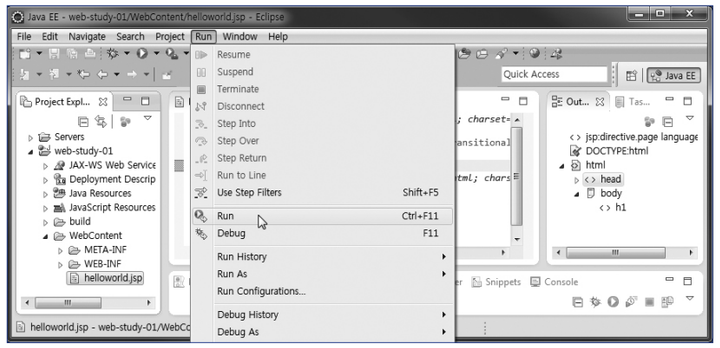


4. [New JSP File] 창이 나타나면 파일 이름을 입력합니다. 이 책에서는 파일 이름을 "helloworld"로 합니다. 파일 이름만 입력하면 확장자는 자동으로 .jsp가 됩니다. 역시 [Finish] 버튼을 클릭합니다.

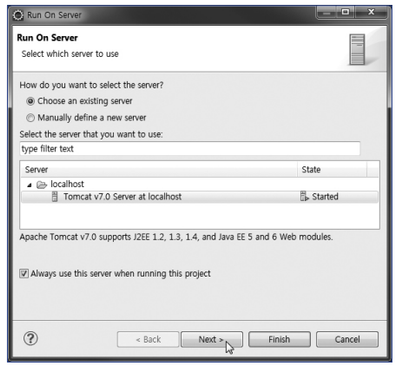
5. 웹 애플리케이션이 제대로 실행되는지 확인을 위해 생성된 JSP 파일의 〈body〉 태그 안에 `<h1> Hello World ! </h1>`을 기술합니다.

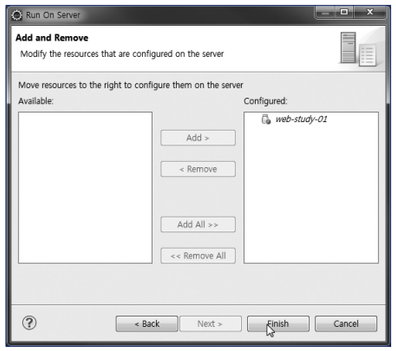


6. 실행을 하기 위해서는 JSP 파일을 선택한 후 [Run → Run]을 선택합니다. 단축키를 사용하고자 할 경우에는 [Ctrl+F11]을 누릅니다.

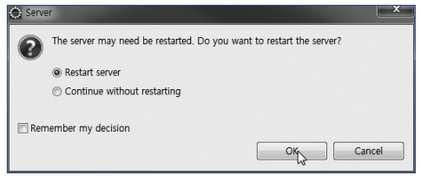


7. JSP 파일을 실행시킬 서버를 선택하라는 창이 나타납니다. 화면 아래 [Always use this server when running this project] 체크 박스를 클릭하면 실행시킬 서버를 선택하라는 창이 더 이상 나타나지 않습니다. [Next] 버튼을 누르면 웹 서버에 웹 프로젝트를 추가하는 화면이 제공됩니다. [Finish] 버튼을 클릭합니다.

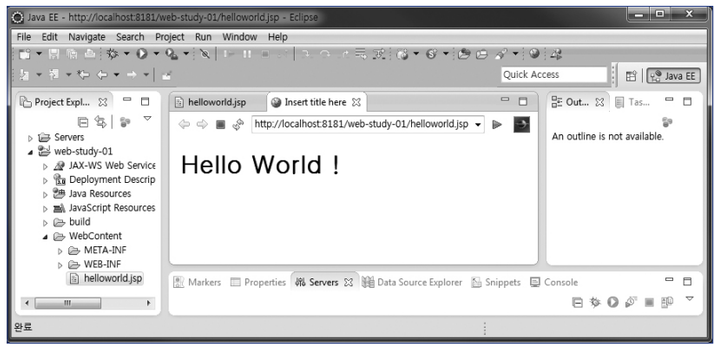




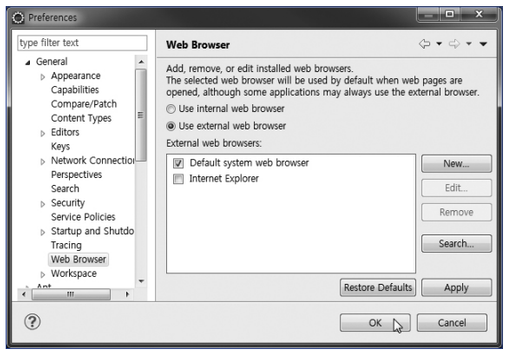
8. 서버에 새로운 웹 프로젝트를 추가하려면 서버가 새로 구동되어야 합니다. [OK] 버튼을 클릭하면 서버에 웹 프로젝트가 추가된 후 서버가 새로 구동되어 JSP 파일을 실행합니다.



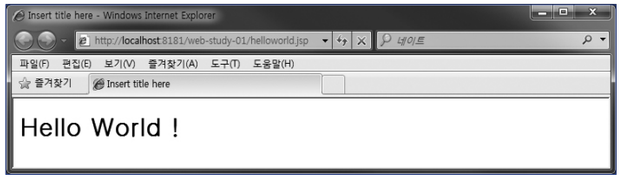
9. JSP 실행 결과는 이클립스 내부에 있는 브라우저에 표시됩니다.



10. 결과 화면을 이클립스 내부가 아닌 팝업 창 형태로 외부에서 확인하고자 한다면 [Window→ Preferences]를 선택한 후에 [Preferences] 창이 나타나면, 화면 왼쪽에서는 [General] 하위 항목으로 [Web Browser]를 선택하고 [Use external web browser] 라디오 버튼을 선택합니다.

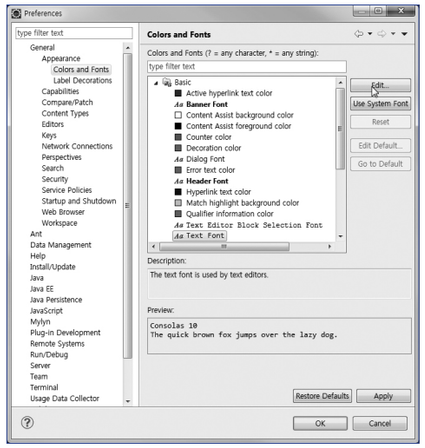


11. 다음은 이클립스 외부의 브라우저에서 JSP 파일을 실행한 결과를 보여주는 화면입니다.



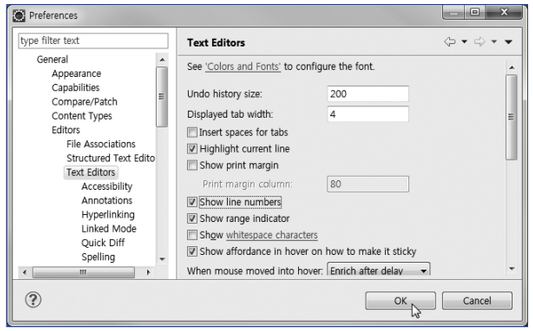
[직접해보세요] 글꼴 변경과 줄 번호 출력하기

1. 글꼴을 변경하기 위해서는 [Window → Preferences] 메뉴를 선택합니다. 이곳에서 다양한 환경설정을 할 수 있습니다. [Preferences] 창이 나타나면 화면 왼쪽에서는 [General] 하위 항목으로 [Appearance → Colors and Fonts]를 선택하고 화면 오른쪽에서는 [Basic → Text Font] 항목을 선택한 후 [Edit...] 버튼을 클릭합니다.

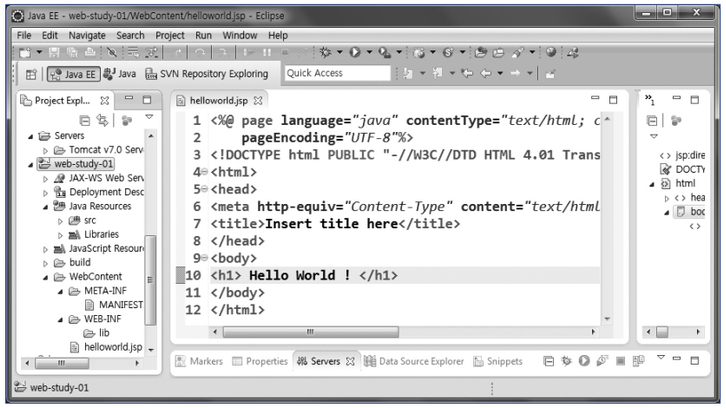


2. [글꼴] 창이 나타나면 원하는 글꼴을 선택합니다. 보통은 Consolas 서체를 사용합니다.

3. [에디터] 뷰에 라인번호가 표시되지 않아 불편하므로, 라인번호가 표시되도록 이번에는 [Preferences] 창 왼쪽에 General의 [Editors] 하위 항목으로 [Text Editors]를 선택하고 화면 오른쪽에서는 [Show line numbers] 항목을 체크한 후에 [Ok] 버튼을 클릭합니다.



4. 글꼴이 변경되고 줄 번호가 생성된 것을 확인할 수 있습니다.



지금까지 이클립스를 설치하고 간단하게 JSP 파일을 생성하여 실행까지 해보았습니다. 성공했다는 기쁨도 있지만, 이클립스에 의해 생성된 JSP가 어떻게 톰캣에 의해 구동되는지 궁금한 게 한두 가지가 아닐 겁니다. 아직은 서블릿에 대해 자세한 학습을 하지 않은 상태이기 때문에 이러한 궁금증을 풀기란 시기상조이고 서블릿과 JSP의 개념을 우선 파악하고 톰캣이 JSP를 어떻게 동작시키지 알아보도록 하겠습니다. 이제 본격적으로 서블릿으로 웹 프로그래밍을 하는 방법을 익히도록 합시다.

서블릿과 JSP의 기초 개념

이제야 본론으로 들어왔습니다. 앞서 웹 프로그래밍이 필요한 이유에 대해서 학습했고 개발 환경까지 세팅하였고 간단하게 JSP 페이지로 만들어 실행해 보았습니다. 이제는 웹 프로그래밍의 핵심 기술인 서블릿과 JSP의 기초 개념을 알아보고 어떻게 개발하는지 알아볼 차례입니다.

서블릿

서블릿Servlet은 Server + Applet의 합성어로 서버에서 실행되는 Applet이란 의미로 자바를 이용하여 웹에서 실행되는 프로그램을 작성하는 기술을 말합니다.

웹 애플리케이션을 제작하기 위해 제공되는 언어는 이미 언급한 PHP, ASP와 같이 많지만 요즘에 많은 기업에서는 JSP&서블릿을 사용하고 있습니다. 이는 다른 웹 기술에 비해 빠른 응답을 해 줄 수 있다는 장점이 있기 때문입니다. PHP처럼 JSP라 하지 않고 JSP&서블릿이라고 한 이유는 독특한 탄생비화 때문입니다. 이 부분은 뒤이어 설명합니다.

하지만 앞으로 서블릿을 공부하면서 상속이나 출력 스트림, 예외 처리와 같은 자바 기술에서 사용되는 용어들이 아주 자연스럽게 나오는데, 이들 용어에 대해서 낯선 분들은 자바 기술에 대한 별도의 학습이 필요합니다.

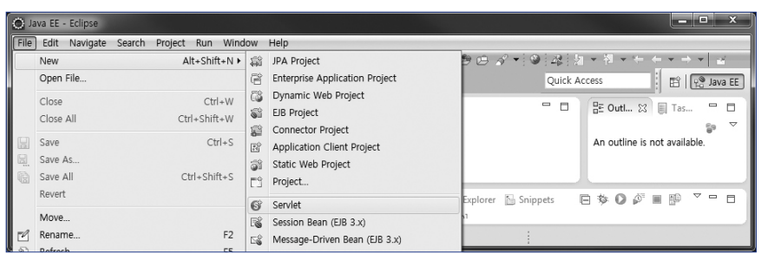
서블릿은 자바 클래스 형태의 웹 애플리케이션을 말하는데, 브라우저를 통해 자바 클래스가 실행되도록 하기 위해서는 `javax.servlet.http` 패키지에서 제공하는 `HttpServlet` 클래스를 상속받아 구현해야 합니다. `HttpServlet` 클래스를 상속 받아 만든 서브 클래스를 서블릿 클래스라고 합니다.

서블릿 또한 자바 프로그램의 다른 클래스들처럼 자바 가상머신인 JVM에서 동작해야 하므로 클래스 파일이 생성되어야 합니다. 그래서 클래스의 형태로 작성합니다. JDK에는 웹 애플리케이션을 제작할 수 있는 클래스가 제공되지 않고 톰캣을 설치하고 나면 웹 애플리케이션을 제작할 수 있는 클래스가 제공되는데, 그 클래스가 바로 `HttpServlet`입니다. `HttpServlet`은 웹 서비스가 가능한 웹 애플리케이션을 제작할 수 있도록 자바를 확장해 놓은 클래스로 톰캣을 설치하면 제공됩니다. `HttpServlet`을 상속받은 클래스를 서블릿이라고 합니다. 이미 여러 기능들이 미리 만들어져 있기 때문에 개발자는 편리하게 `HttpServlet`을 활용하여 새로운 기능의 웹 프로그램을 만들 수 있는 것입니다.

이제 서블릿 클래스가 어떤 구조로 되어 있는지 살펴보기 위해 지금부터 덧셈을 해주는 서블릿을 이클립스를 이용하여 단계적으로 만들어 보겠습니다.

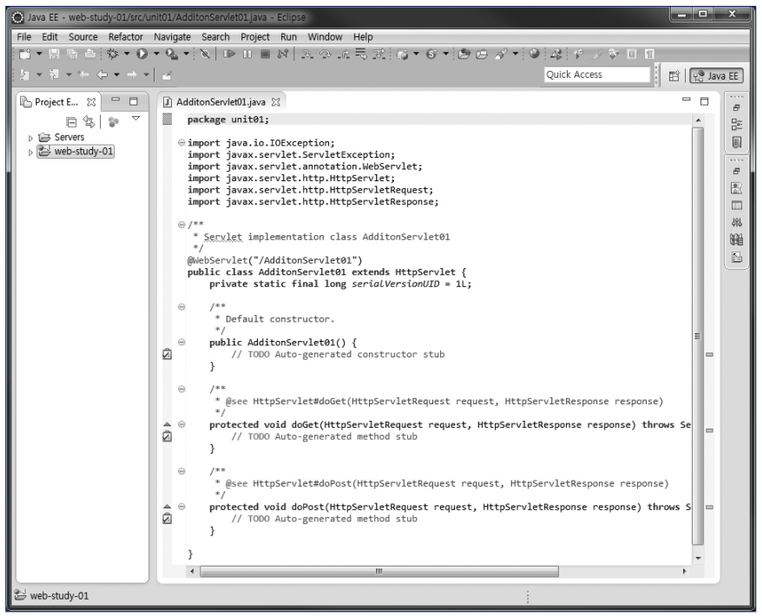
[직접해보세요] 두 수에 대한 합을 구하여 결과를 출력하는 서블릿 클래스

1. 이클립스의 [Project Explorer]에서 웹 프로젝트(web-study-01)를 클릭하여 선택한 후 [File → New → Servlet]을 선택합니다.

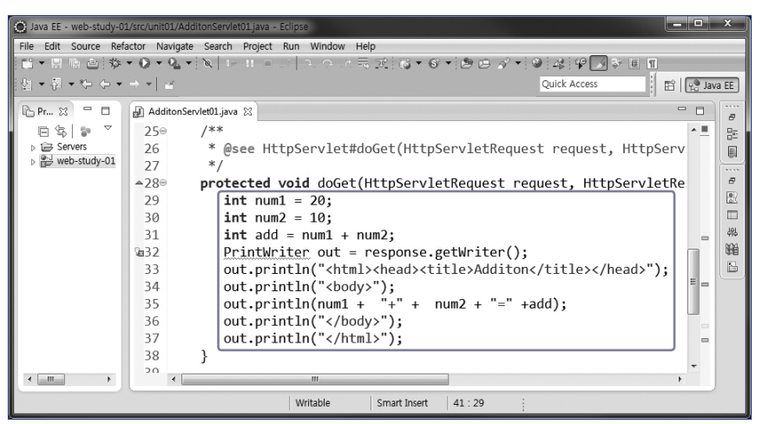


2. [Create Servlet] 창이 나타나면 패키지(unit01)와 서블릿 클래스 이름(AdditonServlet01)을 입력한 후 [Finish] 버튼을 클릭합니다.

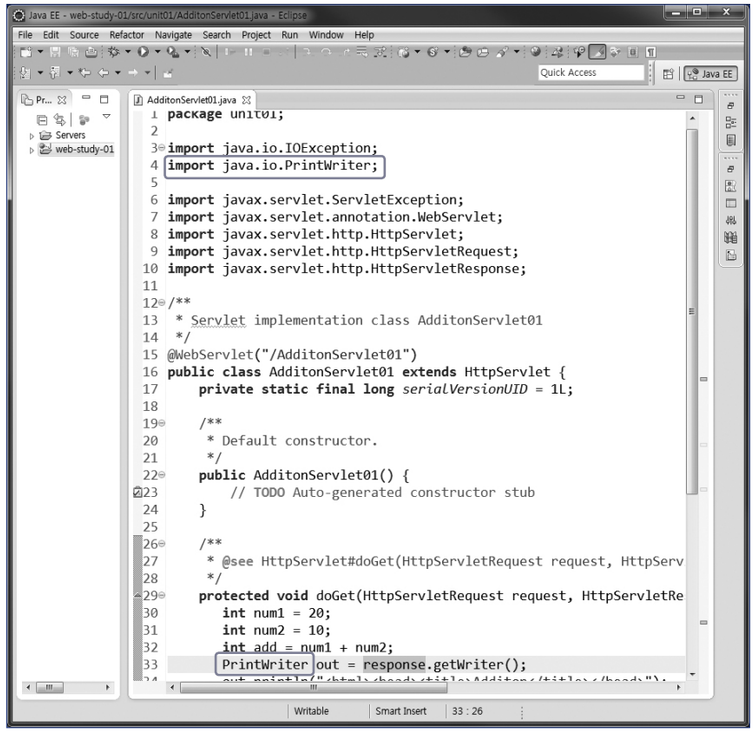
3. 서블릿 클래스가 생성됩니다.



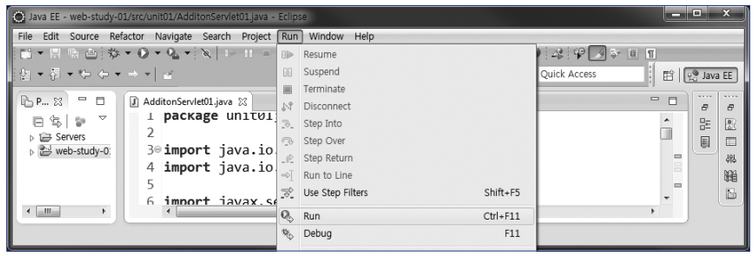
4. 29번 라인부터 다음과 같이 입력합니다.



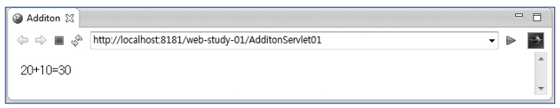
5. 32줄에서 에러가 발생하는데 이는 PrintWriter 클래스 사용을 위한 import문이 없기 때문입니다. Ctrl+Shift+O를 동시에 누르면 `import java.io.PrintWriter;`와 같은 임포트 구문이 자동 추가되면서 발생했던 에러가 해결됩니다.



6. 작성한 서블릿을 실행하여 결과를 얻으려면 [Run → Run]을 선택합니다.



7. 웹 브라우저 주소입력란에 http://localhost:8181/web-study-01/AdditonServlet01가 자동 입력되어 다음과 같은 결과가 출력됩니다.



다음은 위에서 실습한 AdditonServlet01.java 파일의 전체 소스 내용입니다. 2장에서 서블릿에 대해서 자세하게 설명할 것이기에 소스코드 다음의 설명을 보고 전체적인 내용만 이해하도록 합시다.

```
1  package unit01;
2
3  import java.io.IOException;
4  import java.io.PrintWriter;
5
6  import javax.servlet.ServletException;
7  import javax.servlet.annotation.WebServlet;
8  import javax.servlet.http.HttpServlet;
9  import javax.servlet.http.HttpServletRequest;
10 import javax.servlet.http.HttpServletResponse;
11
12 /**
13  * Servlet implementation class AdditonServlet01
14  */
15 @WebServlet("/AdditonServlet01")
16 public class AdditonServlet01 extends HttpServlet {
17   private static final long serialVersionUID = 1L;
18
19     /**
20      * Default constructor.
21      */
22     public AdditonServlet01() {
23         // TODO Auto-generated constructor stub
24     }
25
26   /**
27    * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
28    */
29   protected void doGet(HttpServletRequest request, HttpServletResponse response)
30 throws ServletException, IOException {
31      int num1 = 20;
32      int num2 = 10;
33      int add = num1 + num2;
34      PrintWriter out = response.getWriter();
35      out.println("<html><head><title>Additon</title></head>");
36      out.println("<body>");
37      out.println(num1 +  "+" +  num2 + "=" +add);
38      out.println("</body>");
```

```
39        out.println("</html>");
40     }
41     /**
42      * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
43      */
44     protected void doPost(HttpServletRequest request, HttpServletResponse response)
45  throws ServletException, IOException {
46        // TODO Auto-generated method stub
47     }
48  }
```

1 : 패키지를 만듭니다. 패키지는 비슷한 프로그램들을 묶을 때 유용합니다. 패키지의 유용성에 대해서는 자바 기본서를 참고하세요.

3 : 입출력 시 예외처리를 위한 클래스로 doGet 메소드의 thorws 절에서 IOException을 사용했기에 임포트합니다.

4 : 클라이언트에 결과를 출력하기 위한 out 객체를 PrintWriter 클래스로 선언하였기 때문에 PrintWriter를 임포트합니다(위에서 자동으로 임포트하는 방법도 배웠죠?)

6 : 서블릿에서 발생하는 예외 처리를 위한 클래스로 doGet 메소드의 thorws 절에서 ServletException을 사용했기에 임포트합니다.

7 : 15줄의 @WebServlet을 사용하기 위해 임포트합니다.

8 : HttpServlet 클래스를 16줄에서 사용하기 때문에 임포트해야 합니다.

9~10 : 30줄과 46줄의 doGet, doPost 메소드의 매개 변수에서 사용한 모든 HttpServletRequest, HttpServletResponse 클래스를 사용하기 위한 임포트 구문입니다.

15 : 서블릿을 요청할 때 직접 클래스를 요청하는 것이 아니고 @WebServlet( ) 안에 기술된 URL로 요청을 하는 것이기에 요청 URL을 정하는 것입니다.

16 : HttpServlet에는 웹 애플리케이션으로 동작하도록 하는 기본 동작 즉 요청에 대한 응답이 가능하도록 하는 내용이 기술되어 있기 때문에 상속받아야 합니다.

29 : 서블릿이 요청을 받으면 이벤트 처리 방식으로 자동으로 호출되는 메소드입니다. HttpServlet에 정의된 메소드인데, 이를 오버라이딩해서 개발자가 요청이 있을 경우 어떤 처리를 해야 하는지 명시해 주어야 합니다. 그래서 오버라이딩해 놓은 것입니다. 오버라이드 등의 필요성에 대해서는 자바 기본서를 참고하세요.

웹 프로그래밍에서 가장 중요한 것은 클라이언트가 어떻게 서버에 요청하는지를 알아야 합니다. 이것을 이해해야 서블릿의 동작 방식을 이해할 수 있기 때문입니다. 서블릿 클래스에 대한 문법적인 내용은 2장에서 자세히 살펴보고 이번 예제에서는 서블릿 동작 방식을 이해하기 위해 클라이언트가 어떻게 서버에 요청하는지부터 살펴봅시다.

클라이언트는 서버에 get과 post 두 가지 방식 중 하나로 요청을 합니다. 두 전송 방식의 차이점은 다음과 같습니다.

전송 방식	설명
get 방식	주소 창을 타고 넘어가기 때문에 서버로 보내는 데이터를 사용자가 그대로 볼 수 있습니다. 그래서 보안에 취약합니다. 255자 이하의 적은 양의 데이터를 전송합니다.
post 방식	html header를 타고 넘어가기 때문에 보안에 강합니다. 255자 이상의 대용량의 데이터를 전송합니다.

1바이트가 8비트인 것처럼, 연필 한 자루가 12개인 것처럼, 주소 입력란에 데이터를 전송하는 데는 제한이 있습니다. 컴퓨터는 2진수 체계이므로 2^8에 저장할 수 있는 최댓값이 255이다보니 255자까지 저장 가능하도록 한 것이다.

서블릿 클래스에는 `doGet()` 혹은 `doPost()`가 있는데, 요청 방식에 따라 호출되는 메소드가 달라집니다. get 방식으로 요청하면 `doGet()`이 호출되고 post 방식으로 요청하면 `doPost()`가 호출됩니다. 그렇기 때문에 요청 방식에 따라 `doGet()` 혹은 `doPost()` 메소드 내부에 호출되었을 때 해야 할 일을 써 넣어야 합니다.

`doGet()` 혹은 `doPost()` 메소드 중 어떤 메소드가 호출되는지 실질적인 HTML 코드로 설명하겠습니다. 보통 〈form〉 태그를 통해 서버에 무엇인가를 전달하거나 요청을 하게 됩니다. 다음과 같이 말입니다.

```
                    요청할 서블릿
<form action="CallServlet">
   <input type="submit" value="전송">
</form>
                    클릭하면 서블릿이 요청된다.
```

위 그림은 〈form〉 태그로 서버 측에 존재하는 많은 서블릿 중 하나를 정해서 요청하고 있습니다. 이를 위해 action이라는 속성값에 요청할 서블릿 이름을 기술해야 합니다. 전송 버튼이 눌리는 순간에 action 속성값에 지정된 서블릿이 요청됩니다. 전송 버튼은 일반 버튼이 아닌 〈input〉 태그의 type 속성값을 "submit"으로 지정하여 만들어야 합니다.

〈form〉 태그가 서블릿을 요청할 때는 get과 post 두 가지 전송 방식 중 한 가지로 전송됩니다. 개발자가 원하는 전송 방식을 결정해 줄 수 있는데, 그러기 위해서는

method 속성값을 〈form〉 태그에 추가하면 됩니다. method 속성값으로 get을 기술하면 doGet() 메소드가, post를 기술하면 doPost() 메소드가 호출됩니다.

그런데 폼 태그에 서블릿 이름을 넣어놓으면 서버에서는 어떻게 해당 서블릿을 찾을 수 있을까요? 그것은 WAS(톰캣 엔진)가 운영체제와 같은 시스템 프로그램이므로 확장자가 .class인 파일을 톰캣이 가지고 있다가 요청이 들어오면 해당 서블릿 클래스들을 찾아서 실행하는 원리입니다. 뒷부분으로 갈수록 이런 동작원리는 확실하게 이해할 수 있을 겁니다.

■ 〈form〉 태그를 이용한 get 방식의 요청의 예

```
<form method="get" action="CallServlet">
  <input type="submit" value="전송">
</form>
```

■ 〈form〉 태그를 이용한 post 방식의 요청의 예

```
<form method="post" action="CallServlet">
  <input type="submit" value="전송">
</form>
```

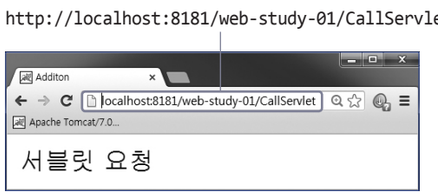
method 속성을 생략한 채 전송 방식을 결정하지 않으면 기본값인 get 방식으로 요청을 하게 됩니다.

〈form〉 태그 외에도 HTML의 〈a〉 태그를 사용하여 링크를 걸어 주면 서블릿은 get 방식으로 요청한 것으로 인식합니다.

■ 〈a〉 태그를 이용한 get 방식 요청의 예

```
<a href="CallServlet"> get 방식의 요청 </a>
```

주소 입력란에서 직접 서블릿 요청을 위한 URL을 입력해도 get 방식으로 요청한 것으로 인식합니다.

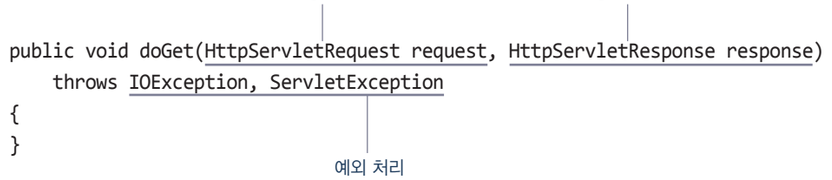


요청에 대한 처리를 위한 doGet()과 doPost() 메소드는 어떤 요청이 왔느냐에 따라 둘 중 하나가 호출된다는 것 빼고는 메소드의 형태는 완전히 동일합니다. 다음은 doGet(), doPost()의 형태입니다.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
{
}

public void doPost(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
{
}
```

doGet(), doPost()에서 동일한 방식으로 처리되기 때문에 doGet() 메소드만 대표로 살펴보도록 합시다.



그렇다면 어떤 경우에 post 방식을 쓰고 어떤 경우에 get 방식을 쓸까요?

앞에서도 잠깐 언급했지만 get 방식은 서버로 데이터가 전송될 때 주소 창을 타고 넘어가기 때문에 보안에 취약합니다. 그렇기 때문에 로그인 폼을 만들면서 get 방식을 사용한다면 회원이 입력한 암호가 그대로 노출됩니다. 그래서 get 혹은 post를 선택할 수 있는 일반적인 폼에서는 post 방식을 사용하는 것이 일반적입니다.

반면 폼이 아닌 〈a〉 태그를 통해서도 페이지를 이동할 수 있는데, 이렇게 하이퍼링크를 통해서 서버가 요청되는 경우에는 무조건 get 방식으로 요청됩니다.

doGet() 메소드는 throws 절로 메소드에서 발생하는 IOException, Servlet Exception 예외를 외부에서 처리하도록 정의되어 있고 두 개의 매개 변수를 갖습니다. HttpServletRequest 형으로 선언된 첫 번째 매개 변수는 클라이언트의 요청(request)을 처리하고, HttpServletResponse 형으로 선언된 두 번째 매개 변수는 요청 처리 결과를 클라이언트에게 되돌리기(응답하기, response) 위해 사용됩니다.

서버가 요청에 대한 처리를 마치고 클라언트에게 결과를 되돌려주기 위해서는 아래와 같이 doGet() 혹은 doPost()의 두 번째 매개 변수인 HttpServletRequest로부터 PrintWriter 형의 출력 스트림 객체를 얻어 와야 합니다.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
{
        PrintWriter out = response.getWriter();
}
```

또한 아래와 같이 PrintWriter 출력 스트림 객체의 println()을 호출하면 브라우저에 HTML 코드를 보내주어 결과를 얻어 볼 수 있게 됩니다.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
    throws IOException, ServletException
{
    PrintWriter out = response.getWriter();
    out.println("<html><head><title>Additon</title></head>");
}
```

자바의 문법적인 형태를 그대로 닮은 서블릿에 대해서 살펴보았으니 이제 같은 목적을 위해 나왔지만 자바의 성격은 거의 느끼지 못하는 HTML과 유사한 JSP의 전반적인 개요에 대해서 살펴보도록 하겠습니다.

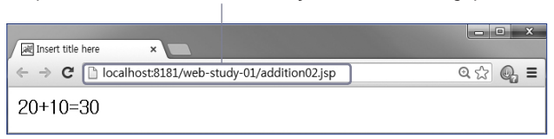
JSP

JSP는 Java Server Page의 줄임말로 자바로 서버 페이지를 작성하기 위한 언어입니다. HTML과 JSP 태그(스크립트릿)로 구성되어 화면을 작성하는 데 유리한 웹 프로그래밍 기술입니다. 서버 페이지는 웹 서버에서 실행되는 페이지를 말하며 요청에 필요한 페이지를 위한 로직이나 데이터베이스와의 연동을 위해 필요한 것들을 포함합니다.

이러한 서버 페이지에서 실행되는 로직을 구현하기 위해서는 프로그래밍 언어가 필요한데, JSP에서는 자바를 사용하고 있습니다. 여기서 말하는 자바는 우리가 알고 있는 썬마이크로시스템즈(현 오라클로 변경)의 '자바'를 말합니다. 즉 JSP에서는 자바 언어로 로직(프로그램)을 구현합니다.

다음은 서블릿 클래스를 학습하면서 살펴본 두 수에 대한 합을 구하여 결과를 출력하는 예제를 JSP로 변환한 예입니다. 이 예제를 실행하기 위해서는 주소 입력란에서 직접 JSP 페이지를 기술하여 실행합니다.

http://localhost:8181/web-study-01/addition02.jsp



■ 두 수에 대한 합을 구하여 결과를 출력하는 JSP [파일 이름 : addition02.jsp]

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<html>
<head>
<title>Addition</title>
</head>
<body>
   <%
      int num1 = 20;
      int num2 = 10;
      int add = num1 + num2;
   %>
   <%=num1%>+<%=num2%>=<%=add%>
</body>
</html>
```

앞에서 이클립스로 JSP 페이지를 만들고 실행하는 방법을 알려주었는데, 그 방법대로 여러분이 직접 JSP 페이지를 만들고 실행해 보기 바랍니다. 위 JSP 예제를 살펴보면 서블릿과는 사뭇 다르다는 것을 느낄 것입니다. 서블릿은 자바 코드 내부에

HTML 코드가 들어가는 구조이지만, JSP는 이와 반대로 HTML 문서 내부에 자바 코드가 들어가는 구조입니다.

HTML 문서 일부분에서 자바를 사용할 수 있도록 하기 위해서 JSP는 다양한 태그를 제공합니다. 위 예(addition02.jsp)에서는 <%@ page %> 태그가 사용되었는데 이 태그는 해당 페이지 내에 사용되는 전반적인 환경을 결정해주는 태그입니다. 이 페이지에서 language="java"는 사용하는 언어가 자바이며 contentType="text/html"; 은 이 페이지가 html 문서이며 charset=UTF-8 pageEncoding="UTF-8"은 한글 인코딩을 UTF-8로 처리하겠다는 의미입니다.

<% %> 태그를 스크립트릿scriptlet이라고 하고 <%= %> 태그는 표현식(expression)이라고 합니다. JSP 페이지에 기술한 내용은 HTML로 간주되기 때문에 자바 코드를 기술하기 위해서는 <% %> 태그 내부에 기술해야 하며 변수에 저장된 값이나 함수의 결과값을 출력하기 위해서는 <%= %> 태그를 사용합니다. 자세한 내용은 뒤에서 살펴볼 예정이니 이런 게 있구나 정도만 기억하고 넘어가시기 바랍니다.

서블릿과 JSP를 비교해 보면 JSP로 개발하는 편이 훨씬 쉽고 간단하다는 것을 알 수 있을 것입니다. 그렇다면 서블릿은 필요 없을까요? 지금까지 살펴본 예제에서는 로직이 복잡하지 않기 때문에 HTML 코드 중심의 구조인 JSP로 페이지를 개발하는 것이 효율적으로 보이지만 쇼핑몰과 같은 웹 애플리케이션을 개발하다 보면 복잡한 자바 코드가 기술되어야 하는데, 이를 JSP 페이지에 기술해 두면 디자이너가 실수로 코드를 건드려 문제가 발생하게 됩니다.

그렇기 때문에 실무에서는 아래에서 보여주는 예(AditonServlet03.java, addition03.jsp)와 같이 복잡한 로직은 서블릿에 기술해 놓고 결과만을 JSP 페이지를 통해 클라이언트에 제공하는 형태로 개발합니다.

■ 두 수에 대한 합을 구하는 서블릿 클래스 [파일 이름 : unit03₩AditonServlet03.java]

```
package unit02;

import java.io.IOException;

import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet("/AditonServlet03")
public class AditonServlet03 extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
                        throws ServletException, IOException {
    int num1 = 20;
    int num2 = 10;
    int add = num1 + num2;

    request.setAttribute("num1", num1);
    request.setAttribute("num2", num2);
    request.setAttribute("add", add);

    RequestDispatcher dispatcher =
      request.getRequestDispatcher("02_addition.jsp");
    dispatcher.forward(request, response);
  }
}
```

AditonServlet03 클래스의 내용은 AditonServlet01.java 내용보다 간단합니다. 이클립스에서 자동으로 생성해주는 클래스에는 불필요한 내용이 많기 때문에 코드 길이가 길어져서 이해하기 힘들어질 수 있기 때문에 필요한 내용만 남겨 두고 정리해 둔 것입니다. 이후에 나오는 서블릿 클래스도 이와 마찬가지로 이클립스에서 생성해준 코드보다는 훨씬 간단한 형태일 것입니다. 직접 만든 서블릿 클래스와 내용이 다르다고 혼동하지 말기 바랍니다.

■ 두 수의 합을 출력하는 JSP [파일 이름 : addition03.jsp]

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
   pageEncoding="UTF-8"%>
<html>
<head>
<title>Addition</title>
</head>
<body>
   ${num1}+${num2}=${add}
</body>
</html>
```

AdditonServlet03.java에서는 두 수에 대한 합을 구하는 자바 코드를 기술하고 출력할 데이터를 request 객체의 속성값으로 저장한 후 서버 상에서 페이지가 이동되는 포워드 방식으로 addition03.jsp 페이지로 이동을 하면 addition03.jsp 페이지에서는 request 객체에 저장된 속성값을 얻어와 출력해줍니다. 포워드 방식은 4장에서 자세히 학습합니다. 이와 같이 로직과 표현을 분리하여 프로그래밍을 하는 것을 MVC 패턴이라고 합니다. 서블릿이 비즈니스 로직을 구현하는 Model 역할을 하고 JSP가 결과를 출력하는 View 역할을 하고 있습니다. MVC 패턴으로 웹 프로그래밍을 하는 것은 복잡하고 까다로운 것이므로 서블릿과 JSP 문법을 다 습득한 후에 다루도록 하겠습니다.

퀴즈로 정리합시다

문제의 답은 로드북 홈페이지(http://roadbook.co.kr/126)에서 확인할 수 있습니다.

1. 다음 Java 기술 중 웹 애플리케이션을 지원하며 HTML과 JSP 태그(스크립트릿)로 구성되어 화면을 작성하는 데 유리한 것은?
   ① JSP ② Servlet ③ Java Bean ④ JDBC

2. 웹 서버와 웹 애플리케션 서버가 무엇인지 개념을 설명하시오.

3. 알고 있는 WAS를 모두 기술하시오.

4. 어떤 식으로 요청을 하면 doGet 메소드가 호출되는지 사례를 들어 설명하시오.

5. 서블릿과 JSP의 차이점을 기술하시오.

6. 한글 인코딩은 무엇이며 왜 해야 하는지 설명하시오.

7. localhost는 자신의 컴퓨터를 지칭하는 도메인 네임이다. 이 도메인 네임에 대응되는 IP 주소는 무엇인가?

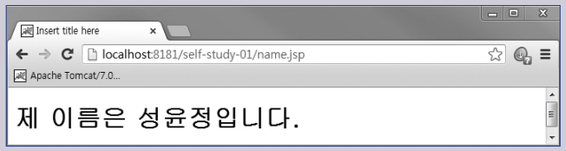
8. 웹 프로그래밍을 하기 위해 WAS 서버로 톰캣(tomcat)을 사용했을 때에, 사용할 웹 포트 번호를 변경하기 위해 수정해야 할 파일명은 ( )이다.

9. Tomcat에 대한 설명으로 바른 것은?
   ① Apache Software Foundation에서 개발한 서블릿/JSP 컨테이너
   ② C 언어로 구현된 HTTP 웹 서버
   ③ 오픈 소스의 통합 개발 환경
   ④ Java 플랫폼

도전해보세요

문제의 답은 로드북 홈페이지(http://roadbook.co.kr/126)에서 확인할 수 있습니다.

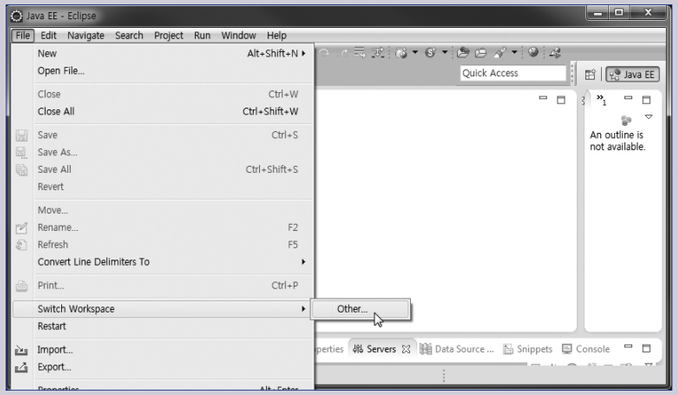
"자신의 이름을 출력하는 JSP 작성하기"



목표 이클립스에서 워크스페이스(self_study)를 생성하고 서버를 설정하고 웹 프로젝트(self-study-01)를 새롭게 만들고 환경설정을 하며 JSP 파일(name.jsp)을 만들어 실행하는 과정을 익힙니다.

난이도 중

힌트 새롭게 워크 스페이스를 만들 경우 [File → Switch Workspace → Other]를 선택하여야 합니다. 또한 톰캣 서버도 새롭게 등록해야 합니다.



2장 서블릿의 기초

이 장을 시작하기 전에

이제 서블릿을 본격적으로 학습할 시간입니다.

- 서블릿의 라이프사이클을 잘 알고 있다면
- 서블릿에서 한글 처리와 데이터 통신을 잘 할 수 있다면
- get 방식과 post 방식을 자세하게 알고 싶다면

이 장을 꼭 보셔야 합니다.

- 폼을 만들어 데이터를 가져와서 브라우저에 표현하는 서블릿을 능숙하게 만들 수 있다면,

다음 장으로 건너뛰어도 좋습니다.

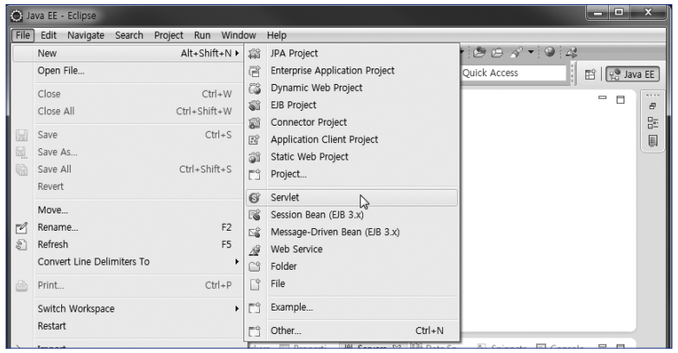
서블릿 프로그램을 만들어 보자

복잡한 개념을 먼저 설명하기 전에 우선 서블릿이라는 프로그램을 한번 만들어봅시다. 보통 처음 프로그래밍 언어를 배울 때처럼 여기에서도 역시 "Hello Servlet"을 출력하기 위한 서블릿 클래스를 만들어 보겠습니다.

또한 서블릿을 요청하기 위한 URL Mapping(바로 뒤에 설명)을 실제 자바 클래스 명과는 다른 이름으로 지정하는 방법도 알아보겠습니다.

[직접해보세요] Dynamic Web Project 만들고 서블릿 만들기

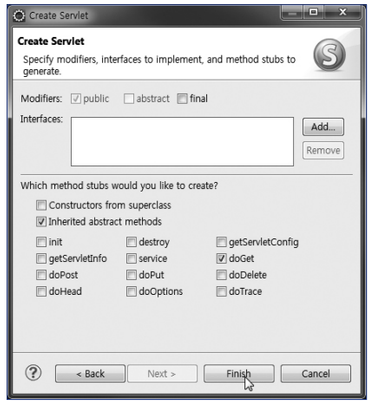
1. [File → New → Dynamic Web Project]를 선택하여 프로젝트 이름(web-study-02)을 입력합니다. [Project Explorer]에서 새로 생성된 웹 프로젝트(web-study-02)를 클릭하여 선택한 후 [New → Servlet]을 선택합니다.



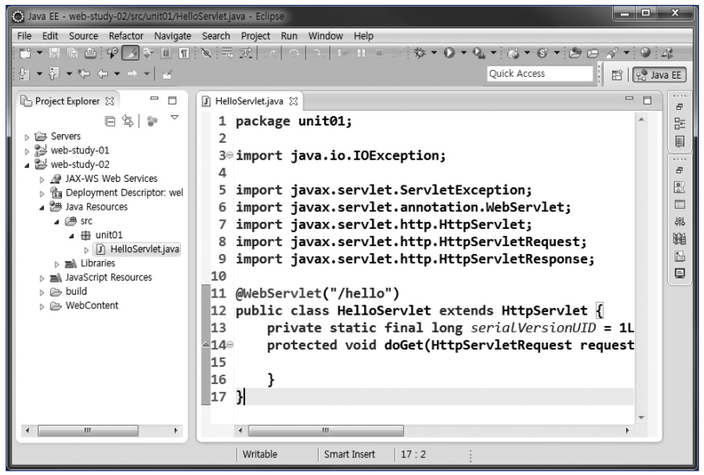
2. [Create Servlet] 창이 나타나면 패키지(unit01)와 서블릿 클래스 이름(HelloServlet)을 입력한 후 [Next] 버튼을 클릭합니다.

3. URL Mapping이란 서블릿을 동작시키기 위해서 실제 자바 클래스 이름 대신에 사용하는 문자열을 말합니다. 즉 그 문자열을 부르면 맵핑된 해당 서블릿이 호출되는 것이죠. [URL mappings:] 목록에서 항목을 선택한 후 [Edit] 버튼을 클릭합니다. [URL Mappings] 창이 나타나면 [Pattern:] 입력란에 패턴명(/hello)을 입력한 후에 [OK] 버튼을 클릭합니다.

4. 이번에 작성하는 서블릿 클래스는 브라우저의 주소란에 서블릿 이름을 직접 입력해서 실행시키는 get 방식으로 요청할 것이므로 doGet만 체크한 후 [Finish] 버튼을 클릭합니다.



5. 다음과 같은 서블릿 클래스가 생성됩니다.



6. 생성된 서블릿 클래스의 doGet( ) 내부에 클라이언트에게 응답해 줄 메시지("Hello Servlet")를 HTML 코드로 작성합니다.

```
1   package unit01;
2
3   import java.io.IOException;
4   import java.io.PrintWriter; // PrintWriter 클래스 사용을 위한 import문
5
6   import javax.servlet.ServletException;
7   import javax.servlet.annotation.WebServlet;
8   import javax.servlet.http.HttpServlet;
9   import javax.servlet.http.HttpServletRequest;
10  import javax.servlet.http.HttpServletResponse;
11
12  @WebServlet("/hello")
13  public class HelloServlet extends HttpServlet {
14    private static final long serialVersionUID = 1L;
15    protected void doGet(HttpServletRequest request, HttpServletResponse response)
16                            throws ServletException, IOException {
17      // 클라이언트에게 응답할 페이지 정보를 셋팅한다.
18      response.setContentType("text/html");
19      // [Ctrl+Shift+오우(알파벳)] : 자동 import
20      PrintWriter out=response.getWriter();
21      out.print("<html><body><h1>");
22      out.print("Hello Servlet");
23      out.print("</h1></body></html>");
24      out.close();
25    }
26  }
```

4 : 출력 스트림인 PrintWriter를 사용하기 위해서는 import를 해야 하는데, 이클립스에서는 [Ctrl+Shift+오우(알파벳)]을 누르면 자동으로 import됩니다. 혹시 PrintWriter를 입력하는 동안에 import 구문이 자동으로 추가될 수도 있으므로 컴파일 에러 메시지가 나타날 경우에만 [Ctrl+Shift+오우(알파벳)]을 눌러 자동 import하면 됩니다.

20~24 : 서블릿은 실행 결과를 클라이언트에게 HTML 문서로 응답해 주기 때문에 response로부터 얻어온 출력 스트림인 out 객체의 출력 메소드인 print에 일일이 하드코딩한 HTML 태그를 기술해 주어야 합니다.

24 : 출력 스트림과 같은 자원들은 사용이 끝나면 이를 안전하게 닫아 주어야 합니다.

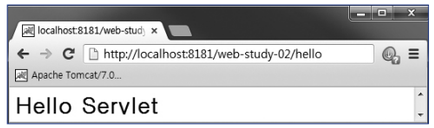
참고

private static final long serialVersionUID = 1L;는 무엇일까요?

자동으로 추가된 14라인은 클래스를 구분하기 위한 값으로 사람에게 주민등록번호와 같은 역할을 합니다.

객체의 직렬화와 같이 객체에 저장된 데이터를 일렬로 입출력할 때 JVM은 같은 클래스 이름과 버전 ID를 가진 객체를 출력합니다. JVM은 버전 ID가 다른 객체의 직렬화된 형태와 연결하는 것을 거부합니다. 클래스는 명시적으로 serialVersionUID 필드를 정의해 클래스 버전에 따른 고유 번호를 포함하기 때문입니다.

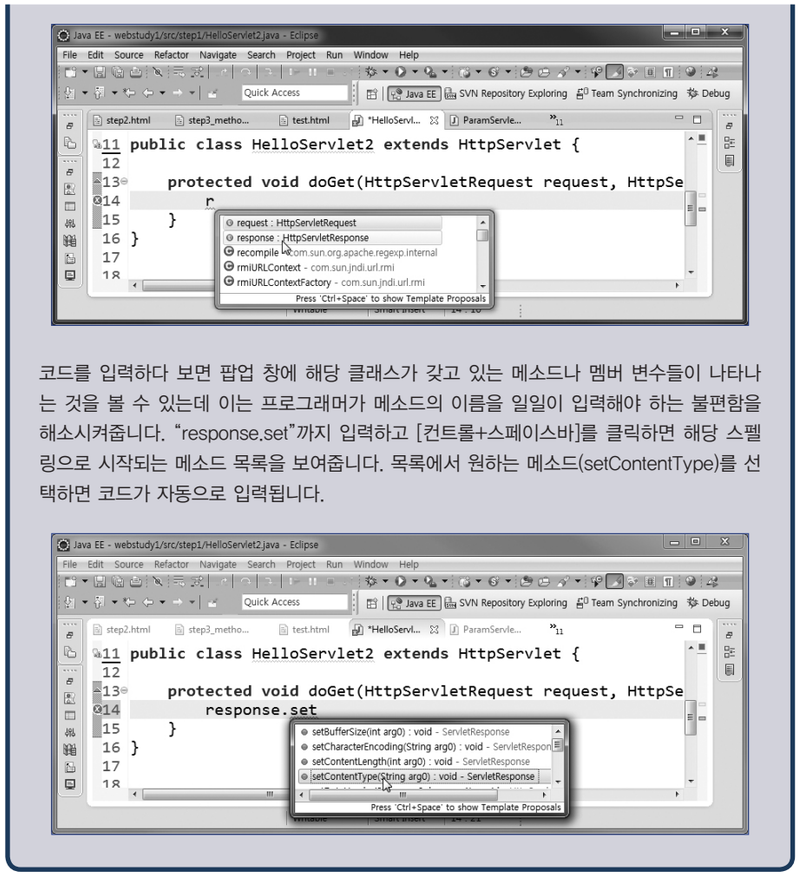
7. 실행을 하기 위해서는 서블릿 클래스를 선택한 후에 [Run → Run]을 선택합니다. 단축키를 사용하고자 할 경우에는 [Ctrl + F11]을 누릅니다.



참고

코드 인사이트로 코딩을 손쉽게 합시다.

코드 편집 과정에서 클래스의 앞부분("r")만 입력한 후에 [컨트롤+스페이스바]를 누르면 해당 스펠링으로 시작되는 클래스들을 목록으로 보여줍니다. 목록에서 원하는 클래스 이름("response")을 클릭하면 코드가 자동 입력됩니다. 이를 코드 인사이트라고 합니다.



코드를 입력하다 보면 팝업 창에 해당 클래스가 갖고 있는 메소드나 멤버 변수들이 나타나는 것을 볼 수 있는데 이는 프로그래머가 메소드의 이름을 일일이 입력해야 하는 불편함을 해소시켜줍니다. "response.set"까지 입력하고 [컨트롤+스페이스바]를 클릭하면 해당 스펠링으로 시작되는 메소드 목록을 보여줍니다. 목록에서 원하는 메소드(setContentType)를 선택하면 코드가 자동으로 입력됩니다.

서블릿을 요청하기 위한 URL은 다음과 같습니다.

http://localhost:8181/ web-study-02 / hello

켄텍스트 패스 서블릿 요청 URL 패턴

서블릿을 요청하기 위한 URL에서 http://localhost는 웹 서버에 접속하기 위한 IP 주소이고 8181은 톰캣을 설치하면서 지정한 포트 번호입니다. 그렇기 때문에 http://localhost:8181은 톰캣 서버에 접속하겠다는 의미입니다.

컴퓨터가 웹 서버로 동작하도록 하기 위해서 1장에서 WAS의 한 종류인 톰캣 서버를 설치했습니다. 우리가 작성하는 웹 애플리케이션은 톰캣 서버에 의해서 클라이언트에 서비스가 되는 것입니다.

하나의 웹 서버는 병원 관리나 학원 관리, 영화 예매 관리, 온라인 쇼핑몰 등 다양한 서비스를 제공할 수 있습니다. 이러한 각각의 서비스는 개별적인 웹 애플리케이션으로 작성해야 하며 웹 애플리케이션 하나당 하나의 프로젝트를 생성합니다. 병원 관리를 위한 웹 애플리케이션은 병원 관리 프로젝트로 학원 관리 웹 애플리케이션은 학원 관리 프로젝트를 개별적으로 생성합니다. 이클립스에서 생성하는 하나의 프로젝트는 하나의 웹 애플리케이션이 됩니다. http://localhost:8181까지 입력하여 웹 서버까지 접근했다면 어떤 서비스를 받을지에 따라 그 이후에 기술되는 내용이 달라지는데, 이후에 기술하는 문자열을 컨텍스트 패스라고 하고 이에 의해서 요청되는 웹 애플리케이션이 달라집니다.

컨텍스트 패스Context Path란 개념을 다시 정리해서 말하자면 웹 서버에서 제공하는 다양한 웹 애플리케이션을 구분하기 위해서 사용하는 것입니다. 병원 관리를 위한 웹 애플리케이션을 위한 병원 관리 프로젝트를 hospital이란 이름으로 이클립스에서 생성하면 hospital이란 컨텍스트 패스가 추가되고 외부에서 이 애플리케이션에 접근할 때에는 다음과 같은 URL을 입력합니다.

http://localhost:8181/ hospital

병원관리 애플리케이션에 접근하기 위한 컨텍스트 패스

영화 예매 웹 애플리케이션을 movie란 이름으로 프로젝트를 생성하면 movie란 컨텍스트 패스가 추가되고 외부에서 이 애플리케이션에 접근할 때에는 다음과 같은 URL을 입력합니다.

http://localhost:8181/ movie

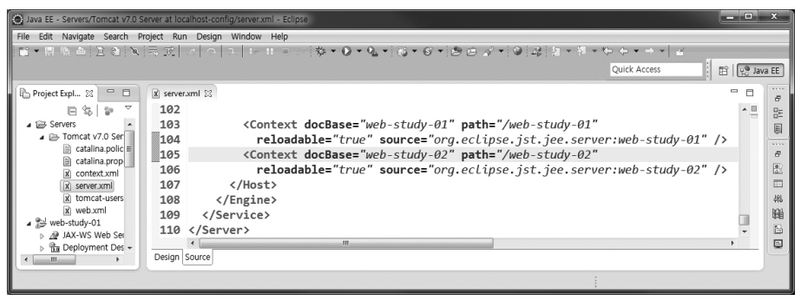
영화예매 애플리케이션에 접근하기 위한 컨텍스트 패스

톰캣 서버에서 클라이언트에게 웹 애플리케이션을 서비스해 주기 위해서는 톰캣 서버에 웹 애플리케이션을 등록해야 합니다. 등록 방법은 톰캣 서버의 server.xml 파일의 〈Context〉 태그를 사용하여 컨텍스트 패스를 추가합니다.

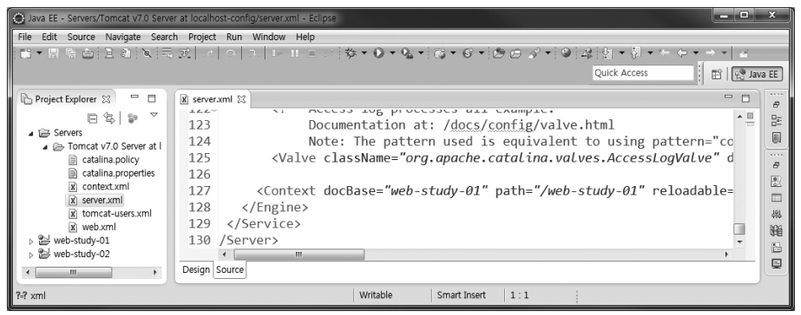
```
<Context docBase="web-study-02" path="/web-study-02"
    reloadable="true" source="org.eclipse.jst.jee.server:web-study-02" />
```

이클립스를 사용하지 않고 웹 애플리케이션을 개발할 때에는 일일이 〈Context〉 태그를 개발자가 기술해야 했지만 이클립스는 컨텍스트 패스를 프로젝트 단위로 자동 생성해 줍니다.

톰캣 서버의 환경 설정을 위한 server.xml 파일을 열어보면 이클립스에서 자동으로 추가해 주는 컨텍스트 패스를 확인할 수 있습니다.

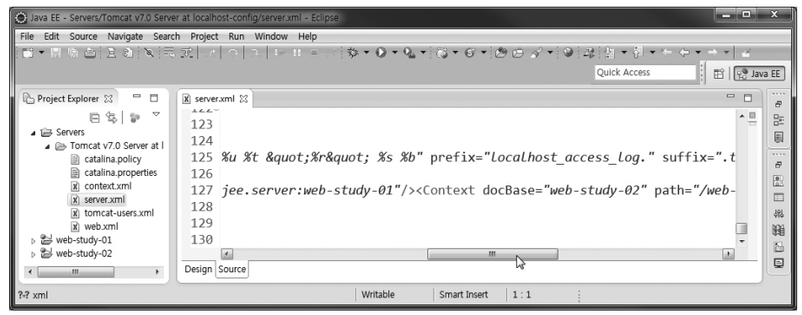


server.xml을 이클립스에서 열어보면 위와 같이 나오지 않고 1장에서 만든 〈Context docBase="web-study-01".../〉만 보이고 〈Context docBase="web-study-02".../〉는 보이지 않습니다.



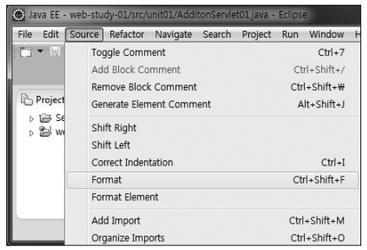
이러한 현상이 나타나는 이유는 〈Context〉 태그가 웹 애플리케이션을 최초로 실행시키면서 추가되는데, 새로운 〈Context〉 태그가 이미 존재하는 〈Context〉 태그

뒤에 추가되어 한 줄에 〈Context〉 태그가 여러 번 기술되기 때문에 스크롤바를 움직여 오른쪽 끝으로 가야 보입니다.



나중에 추가된 웹 프로젝트에 대한 〈Context〉 태그를 찾으려면 스크롤바를 움직여서 오른쪽 끝으로 가서 확인하는 번거로운 작업을 반복해야 하기 때문에 들여쓰기를 하여 〈Context〉 태그가 서로 다른 라인에 출력되도록 하면 됩니다.

하지만 들여쓰기를 개발자가 직접 하는 것이 번거롭다면 [Source → Format] 메뉴를 선택하거나 단축키인 [Ctrl+Shift+F]를 사용하면 코드가 자동으로 들여쓰기가 됩니다.



〈Context〉 태그에 path 속성이 바로 서블릿을 요청할 때 지정할 URL에 기술할 가상 패스입니다. 앞의 그림에서 보면 path 속성값이 프로젝트 이름인 "/web-study-02"로 지정되어 있음을 확인할 수 있습니다. 이클립스에 의해 serever.xml 파일에 자동 추가된 〈Context〉 태그 덕분에 톰캣이 컨텍스트 패스 "/web-study-02"를 인식할 수 있게 됩니다.

이제 위에서 실습한 서블릿 클래스의 구조에 대해서 살펴보도록 합시다.

```
@WebServlet("/hello")                                          ❶
public class HelloServlet extends HttpServlet {                ❷

  protected void doGet(HttpServletRequest request,
                       HttpServletResponse response)
                throws ServletException, IOException {         ❸
    response.setContentType("text/html");                      ❹
    PrintWriter out=response.getWriter();                      ❺
    out.print("<html><body><h1>");                             ❻
    out.print("Hello Servlet");
    out.print("</h1></body></html>");
    out.close();                                               ❼
  }
}
```

@WebServlet 어노테이션으로 URL 매핑

@WebServlet(❶)은 서블릿 3.0에서부터 제공되었으며 서블릿 클래스의 요청을 위한 URL 매핑을 보다 쉽게 자바 클래스에서 설정할 수 있도록 제공되는 어노테이션입니다. 서블릿 3.0 이전에는 web.xml에서 매핑을 했기 때문에 다소 불편함이 있었습니다.

참고

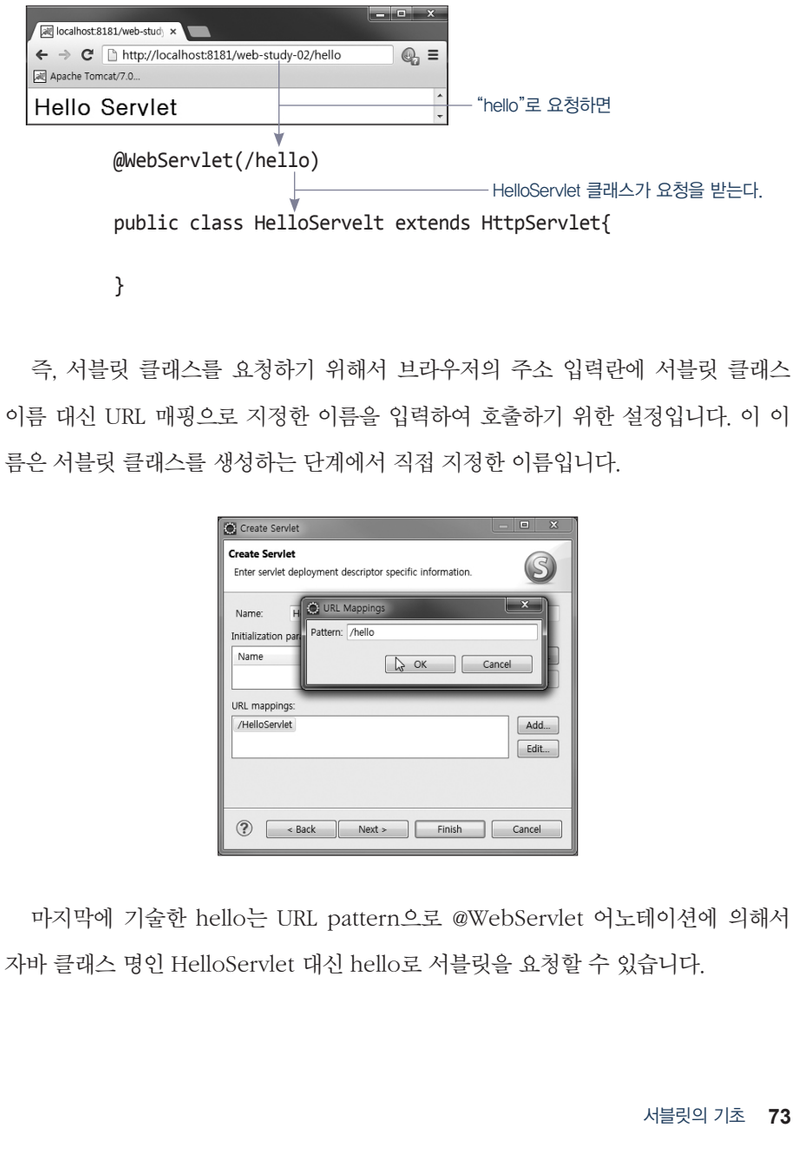
어노테이션(Annotation)

Java 5.0부터 AT 사인(@)으로 시작하는 어노테이션이 지원되었습니다. 어노테이션은 문장이나 문서에 추가적인 정보를 기입하는 것을 말합니다. 자바 프로그램에 영향을 주는 것이 아니라 컴파일할 때 환경 설정을 변경해 줄 것을 알려주는 주석 형태를 말합니다. 이전에는 환경설정을 XML 파일에서 직접 해왔습니다. 하지만 XML 파일을 열어서 일일이 환경 설정하는 일이 번거롭기도 하고 XML 문법을 시간 내어 학습해야만 하기 때문에 개발자가 직접 XML 파일에서 작업하지 않고 자바 코드에서 어노테이션을 사용하는 방식으로 쉽게 환경을 설정하기 위해 자바 5.0에서부터 등장하게 된 것입니다. 어노테이션 등장 덕분에 개발 시간이 단축되었습니다.

URL Mapping이란 서블릿을 동작시키기 위해서 실제 자바 클래스 명(HelloServlet)을 사용하는 대신 서블릿을 요청하기 위한 문자열(hello)을 서블릿 클래스와 매핑시키는 것을 말합니다.

그렇다면 URL 매핑을 하는 이유는 뭘까요?

실제 서블릿 클래스를 공개하지 않기 위해서입니다. 실제 호출되는 서블릿 클래스는 HelloServlet이지만 외부에서 이 서블릿을 요청할 때에는 서블릿 클래스 이름이 아닌 서블릿 클래스와 매핑된 URL인 hello로 접근합니다.



즉, 서블릿 클래스를 요청하기 위해서 브라우저의 주소 입력란에 서블릿 클래스 이름 대신 URL 매핑으로 지정한 이름을 입력하여 호출하기 위한 설정입니다. 이 이름은 서블릿 클래스를 생성하는 단계에서 직접 지정한 이름입니다.

마지막에 기술한 hello는 URL pattern으로 @WebServlet 어노테이션에 의해서 자바 클래스 명인 HelloServlet 대신 hello로 서블릿을 요청할 수 있습니다.

참고

개발자가 어노테이션으로 패턴을 지정하는 것과 이클립스에서 직접 패턴을 지정하는 것의 차이가 있나요?

개발자가 URL pattern을 이클립스 메뉴에서 변경하든 하지 않았든, 사용자가 서블릿을 요청하기 위해서는 브라우저 주소 입력란에 URL pattern을 기술해야만 서블릿이 요청됩니다. URL Mapping 없이는 서블릿을 호출하지 못합니다.

개발자는 자신이 만든 서블릿 클래스가 어느 경로에 무슨 이름으로 만들어졌는지 알아야 하지만 클라이언트는 어느 디렉토리에 어느 파일명으로 존재하는지는 관심 없고 URL 이름을 입력하여 원하는 서비스만 받을 수 있으면 됩니다.

만일 사용자가 서블릿의 실제 경로와 파일 이름을 직접 입력해야 한다면 디렉토리 구조가 바뀌었을 때 사용자에게 일일이 변경된 위치를 통보해 주어야 합니다. 개발자가 아닌 사용자가 실제 개발 구조를 다 알아야만 사용할 수 있다면 수정된 내용을 다 알고 있어야 합니다. 하지만 매핑을 통해 사용자가 접근하는 URL 이름은 실제 물리적인 위치 정보가 아니기에 이런 문제에 유연하게 대처할 수 있게 됩니다. 또한 사용자에게 디렉토리 구조와 파일명을 모두 공개한다는 것은 보안 측면에서도 심각한 문제가 발생할 수 있습니다.

요청은 URL pattern으로 하고

```
http://localhost:8181/web-study-02/hello
                                   ↑
                              서블릿 요청 URL 패턴
```

이 패턴을 @WebServlet 어노테이션 코드의 URL Mapping에서 찾아서 일치하면 이 URL Mapping 바로 아래 선언된 서블릿 클래스가 요청됩니다.

```
@WebServlet(/hello)
                    ↓ HelloServlet 클래스가 요청을 받는다.
public class HelloServelt extends HttpServlet{

}
```

다시 한 번 언급하지만 URL Mapping은 URL pattern과 서블릿 클래스 이름을 매핑해 놓는 것을 의미합니다.

개발자가 URL pattern을 변경하지 않은 경우	개발자가 URL pattern을 변경한 경우

@WebServlet("/HelloServlet ")
public class HelloServlet extends HttpServlet {
 	
}
	@WebServlet("/hello ")
	public class HelloServlet extends HttpServlet {
	
	}

개발자가 직접 패턴을 지정하지 않아도 서블릿 클래스를 생성하면서 자동으로 서블릿 클래스 이름을 URL pattern으로 하는 @WebServlet 어노테이션 코드가 추가되어 있습니다. 왜냐하면 서블릿 요청은 URL pattern으로 해야 하고 이렇게 패턴으로 요청하면 패턴과 매핑해 놓은 서블릿이 호출되기 때문입니다. 우리가 이클립스 메뉴를 열어서 패턴을 바꾼 것은 서블릿 클래스 이름과 동일한 패턴을 사용하지 않고 전혀 다른 이름으로 서블릿을 호출하기 위한 작업입니다. 메뉴에서 패턴을 지정한 작업에 의해서 @WebServlet 어노테이션 코드 내의 패턴 이름만 바뀐 것입니다. 여전히 @WebServlet 어노테이션 코드는 존재하고 이 @WebServlet 어노테이션 코드를 통해서만 URL pattern과 클래스 이름이 매핑됩니다.

서블릿 클래스 정의하기

1장에서 이미 언급한 대로 서블릿 클래스를 정의하는 것은 정형화되어 있습니다. 새롭게 서블릿 클래스를 정의하기 위해서는 javax.servlet.http 패키지에서 제공하는 HttpServlet(❷) 클래스를 상속받아 구현해야 하고 브라우저를 통해 외부에서 실행되기 때문에 접근 제한자는 반드시 public(❷)이어야 합니다.

```
접근 제한자는 반드시
public이어야 함
   ↓
public class HelloServelt extends HttpServlet {
                   ↑                    ↑
            서블릿 클래스 이름      HttpServlet의 상속을 받아야 함
}
```

우스운 질문일 수 있겠으나, 왜 서블릿은 위와 같이 정형화한 형태를 띄게 된 걸까요? 그냥 일반 자바 클래스처럼 만들면 안 될까요?

서블릿이 정형화 되어 있다는 말은 자바 클래스에서 사용하던 상속이란 개념을 그대로 사용하는데 자바 클래스에서는 어떤 클래스의 상속을 받아도 되지만 서블릿 클래스는 HttpServlet 클래스의 상속만을 받아야 하는 것으로 정해디져 있다는 것을 말합니다.

요청 메소드

1장에서 설명한 것처럼 요청 방식에 따라 `doGet()` 혹은 `doPost()`가 호출되기 때문에 요청 방식에 맞추어서 HttpServlet 클래스의 `doGet()` 혹은 `doPost()`를 오버라이딩해야 합니다. 이 메소드는 요청이 되면 호출되기 때문에 요청 메소드라고 불리기도 합니다.

참고

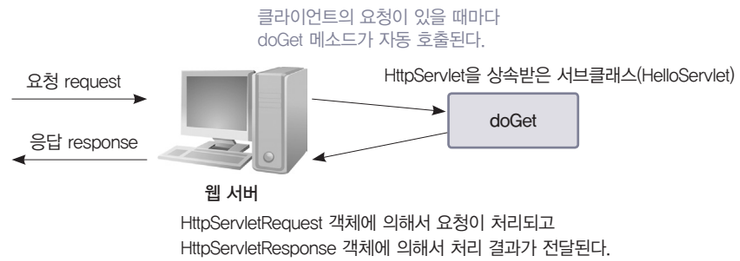
오버라이딩을 기억하시나요?

자바의 클래스들은 부모(수퍼) 클래스에 모든 필드나 메소드를 상속받아 사용합니다. 상속받아 사용하던 메소드의 기능을 더 이상 사용하지 않고 자식(서브) 클래스에서 새롭게 메소드를 정의해서 사용하는 것을 오버라이딩이라고 합니다.

우리가 처음으로 작성할 서블릿은 브라우저의 주소란에 직접 서블릿 이름을 입력하여 수행시킬 것이기 때문에 HttpServlet의 `doGet()`을 오버라이딩(❸)하여 처리를 위한 코드를 입력하겠습니다.

```
                                    요청 처리                          응답 처리
                                        |                                  |
public void doGet(HttpServletRequest request, HttpServletResponse response)
                throws IOException, ServletException,
{                                                   |
                                                    예외 처리
}
```

`doGet()`은 두 개의 매개 변수(HttpServletRequest, HttpServletResponse)를 갖습니다. HttpServletRequest는 클라이언트의 요청(request)을 처리하고, HttpServletResponse는 요청 처리 결과를 클라이언트에게 되돌리기(응답하기, response) 위해 사용됩니다. `doGet()`은 반드시 예외 처리(IOException, ServletException)를 해주어야 하는데, 일반적으로 throws 절을 이용해서 doGet 메소드를 호출한 웹 서버에게 예외처리를 넘깁니다.



참고로 위의 서블릿은 get 방식으로만 요청을 처리하는데, 하나의 서블릿이라도 get과 post 방식에 따라 서로 다른 기능을 제공해야 하는 경우도 있기 때문에 그럴 경우에는 `doGet()`과 `doPost()`를 모두 오버라이딩해야 합니다.

응답 객체에 콘텐트 타입 지정하기

HttpServletResponse 객체인 response로 `setContentType()` 메소드(❹)를 호출하여 클라이언트에게 응답할 페이지에 대한 환경 설정을 결정해 주어야 합니다. 응답 방식이 "text/html"로 지정되어 있으므로 text나 html로 보여주겠다는 의미입니다.

참고

서블릿의 실행 방식

일반적인 자바 클래스를 실행하기 위해서는 main 메소드가 있는 클래스에서 객체를 생성하여 실행해야 합니다. 하지만 서블릿은 이런 방식으로 실행되지 않고 웹 서버가 실행을 해주는 독특한 방식을 갖습니다.

서블릿은 Event-Driven Programming으로 사용자의 요청이 들어오면 동작(실행)을 시작합니다. 요청이 들어오면 톰캣 서버가 서블릿 객체를 생성한 후에 init( ) 메소드를 호출한 후 요청 방식에 따라 doGet( ) 혹은 doPost( ) 메소드가 호출됩니다.

뒤에 자세하게 배우게 되니 지금은 이 정도만 알고 넘어갑시다.

결과로 출력할 내용이 한글일 경우 인코딩 방식을 지정하지 않으면 한글이 깨지는 현상이 나타납니다.